UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Conteo de multitudes a través de redes neuronales y aprendizaje profundo

Informe de Proyecto de Grado presentado por

Renzo Gambone

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Mathías Etcheverry ..... Universidad de la República
Pablo Musé ..... Universidad de la República

Tribunal

Diego Garat ..... Universidad de la República
Martín Llofriu ..... Universidad de la República
Federico Lecumberry ..... Universidad de la República

Montevideo, 4 de agosto de 2023

Agradecimientos

Me gustaría agradecer a mis amigos quienes me escucharon y me prestaron su hombro cuando estuve al borde de abandonar este proyecto, en particular a mi pareja Cecilia, quién siempre estuvo ahí para darme apoyo cuando lo necesitaba, que durante meses me relevó en varias tareas del hogar para que yo pueda enfocarme en la tesis, y me ayudó como revisora de este informe, encontrando y enumerando un sinfín de errores gramaticales.

A mi lugar de trabajo que permitió desarrollar mis conocimientos en el área de aprendizaje automático, y a mis compañeros de trabajo, que siempre me animaron y apoyaron a seguir adelante.

A mis tutores, Mathias que por más que estuvimos meses sin avance, en todo momento y a toda hora siempre fue ágil y dispuesto a dar una mano con lo que necesite, y también me recordó que es importante descansar y cuidar la salud mental; y a Pablo quien pese a todas sus responsabilidades siempre respondía emails con consultas, no dudó en ofrecer su ayuda cuando le comunicamos que el trabajo iba a ser retomado y sus conocimientos en el área de visión artificial fueron de gran ayuda para este trabajo.

Y por último, pero para nada menor, a mi madre Nibia y difunto padre Silvio, quienes lo dejaron todo trabajando con el fin de regalarme un privilegio que ellos no pudieron tener, el de poder estudiar. Siempre me apoyaron a perseguir algo que me gustara.

Resumen

Este trabajo es un acercamiento al área del conteo de multitudes, es decir, dada una imagen predecir correctamente cuántas personas hay en la misma, y opcionalmente brindar una noción de su posición en la imagen (en forma de un mapa de calor por ejemplo). Para resolver dicho problema se emplean técnicas de aprendizaje automático, más concretamente aprendizaje profundo orientado a la visión artificial.

En este trabajo se estudió y plasmó en este informe cómo estas técnicas aplicadas al conteo de multitudes evolucionaron en la última década, posteriormente enfocándose en detallar las direcciones, técnicas y dificultades presentes en el estado del arte contemporáneo, así como también enumerando posibles direcciones en las que el mismo podría dirigirse en el futuro cercano (basadas en trabajos y arquitecturas emergentes y de gran promesa).

Como parte de la experimentación se etiquetó un conjunto de datos a través de un módulo de taller, se evaluó la similitud entre las anotaciones realizadas, discutiendo sobre la dificultad de la tarea y el posible ruido en anotaciones de conjuntos ya existentes.

También se evaluó el desempeño de varios modelos pre-entrenados en múltiples conjuntos de evaluación, discutiendo sobre particularidades, relaciones y anomalías encontradas en los resultados de sus métricas generales y algunos resultados destacados.

Palabras clave: Visión artificial, Conteo de multitudes, Aprendizaje profundo.

Tabla de contenidos

Capítulo 1

Introducción

En este capitulo se presenta el objetivo del proyecto, su alcance y como este fue alterado a lo largo del mismo. Por último se detalla la organización y estructura de este informe.

1.1. Objetivo del proyecto

En la actualidad, el área computacional denominada **visión artificial** (*computer vision*) se encuentra en un auge histórico, desde un punto de vista tanto académico como productivo. Debido a su utilidad para una gran diversidad de problemas, continuamente se están haciendo avances en el área, donde cada avance contribuye a la vorágine de información actual; de ella nacen soluciones cada vez más precisas frente al diverso rango de problemáticas atacadas y emergentes dentro de este campo de estudio.

Buscando tanto el aprendizaje propio como la contribución académica, nace la necesidad de estudiar una tarea de la visión artificial tan específico como desafiante: el **conteo de multitudes** (*crowd counting).* El desafío del problema radica tanto en su complejidad intrínseca como en la dificultad de generar soluciones realmente efectivas que lo resuelvan en distintos escenarios heterogéneos. Teniendo en cuenta estas características, resulta de interés enfocar un proyecto de grado a un estudio del problema, encarando lo mismo desde múltiples enfoques y abordando sus diferentes aristas.

Cabe destacar que al comienzo del proyecto, se enumeraron distintas posibles contribuciones al problema. Dentro de dichas opciones se manejó la posibilidad de generar un *conjunto de datos*† de calidad para el *entrenamiento*† de modelos, diseñar una arquitectura innovadora en comparación a las ya existentes, o reajustar un modelo implementando técnicas actuales y apuntando a mejorar su resultados.

A medida que se avanzó en la etapa de investigación se fueron identificando dificultades para realizar algunas de las lineas de trabajo consideradas de manera efectiva.

Hoy en día existen muchas preguntas sin responder en lo que refiere al conteo de multitudes utilizando visión artificial; gran parte de los avances en el campo se realizan buscando mejorar *métricas de evaluación*† pero por dificultades del campo, o de la metodología de investigación, las optimización de métricas no necesariamente apunta a la resolución de todas estas preguntas.

Es por esto que se descartó la idea de invertir energía generando nuevos avances en ese frente (como un modelo que intente sobrepasar el estado del arte) y se decidió centrar la totalidad de la energía en realizar un estudio tan profundo como sea posible en el alcance del proyecto, que permita delimitar las fronteras de este problema tan desafiante y servir de punto de partida para todo aquel que busque introducirse o ampliar su conocimiento en el conteo de multitudes con *redes neuronales*†.

Se destaca también como no son pocas las contribuciones existentes en el área y desde el comienzo de este proyecto hasta su culminación, se publicaron múltiples avances que en su mayoría se intentaron tomar en cuenta a medida que iban siendo publicados, lo que incidió en otra etapa de investigación con el fin de complementar aún más el contexto actual del conteo de multitudes. Un ejemplo de esto es la discusión de capacidad de generalización de modelos en la sección 4.11.1.

1.2. Alcance del proyecto

La tarea de conteo de multitudes es relativamente nueva y al inicio de este trabajo no había sido estudiada por los estudiantes, ni por los tutores, contando sí con conocimientos previos en el área más general de visión artificial.

El planteamiento inicial del proyecto consistía en:

- Desarrollar los conocimientos de visión computacional los estudiantes.
- Estudiar y analizar distintos enfoques existentes en la literatura para abordar el problema de conteo de multitudes.
- Compilar la evolución del conteo de multitudes a lo largo de los años.
- Analizar el estado del arte actual, determinando cuáles son sus fortalezas, carencias y direcciones en las que podría llegar a avanzar.
- En base a los resultados obtenidos en el estudio realizado, terminar de desarrollar el alcance de este trabajo de una forma que agregue valor al mismo.

Trece meses tras haber iniciado el proyecto, se había realizado una extensa investigación y discusión sobre los antecedentes del problema, pero debido a diversos problemas, el esfuerzo fue disminuyendo, llegando a pararse por completo durante los últimos cuatro meses. Tras plantear la reactivación del mismo, tres de los cuatro estudiantes ya no tenían la motivación o dedicación para continuar con el proyecto y optaron por abandonar el mismo.

Con la ayuda de los tutores revisamos el alcance del proyecto y enviamos carta a la comisión de proyecto de grado comunicando la situación e intención de continuar con el proyecto de grado.

La separación del grupo impactó significativamente en el alcance del proyecto, limitando la cantidad de trabajo a paralelizar de 4 a 1 sólo estudiante.

Se descartaron por completo elementos tentativos y/o complementarios del alcance como:

- Proponer y entrenar un modelo.
- Re-entrenar modelos con nuevos *conjuntos de datos*†.
- Contribuir a proyectos de código abierto, mejorando el proyecto base de *entrenamiento*† *NWPU-Crowd Sample Code* (*Wang et al.*, 2020c), o agregando más modelos y capacidad de correr en GPU a la biblioteca *LWCC* (*Teršek and Kljun*, 2021).
- Profundizar en total completitud en todas las aristas en las que se ha direccionado el campo del conteo de multitudes en estos últimos años.
- Experimentar la capacidad de generalización de modelos a otros dominios de conteo, como puede ser en la industria de agricultura y ganadería, a través de aplicar *aprendizaje por transferencia*†.

De igual forma, a modo de entregar un trabajo completo, se agregaron los siguientes elementos al alcance del proyecto:

- Identificar áreas poco exploradas en la evolución del conteo, que puedan estar sesgando la dirección del estado del arte sin ser consideradas por los investigadores. Por ejemplo la calidad de las anotaciones en los *conjuntos de datos*† existentes.
- Participar en la organización y ejecución de la anotación de un *conjunto de datos*† de pequeño porte y realizar un análisis estadístico sobre la calidad de anotaciones adquiridas.
- Estudiar el grado de *adaptación al dominio*† y la capacidad de generalización de algunos modelos que forman el estado del arte.
- Evaluar mediante la experimentación la aplicabilidad de modelos de conteo de multitudes en distintos dominios.

1.3. Organización del documento

Este documento se organiza de la siguiente manera:

- La sección 2 define el conteo de multitudes, enumerando también las dificultades intrínsecas a la tarea.
- La sección 3 compila los recursos de apoyo a la investigación del área.
- La sección 4 compila todos los conceptos aprendidos y necesarios para comprender este trabajo, así como a la tarea del conteo de multitudes. La misma se divide en:
  - Las secciones 4.1, 4.2, 4.3 y 4.5, que profundizan en el conteo de multitudes a lo largo de su historia y definen conceptos generales del campo, como son las *métricas de evaluación*†.
  - La sección 4.4, que presenta un abordaje de los conceptos relevantes al *aprendizaje automático*† y *aprendizaje profundo (deep learning)*† necesarios para comprender el resto del trabajo.
  - Las secciones 4.6 y 4.7, que evalúan las dos grandes arquitecturas contemporáneas del campo, de múlti columna y una sola columna.
  - Las secciones 4.8 y 4.9, dedican el resto del estudio a la arquitectura de una sola columna (por su relevancia en estos últimos años). Estas secciones profundizan en problemas, técnicas y módulos que trabajos anteriores han investigado y propuesto para mejorar el desempeño de los modelos.
  - La sección 4.10, que plasma el estado del arte actual, mientras que la sección 4.11 complementa la investigación con áreas de reciente interés experimental que parecen prometer avanzar el campo del conteo de multitudes.
- La sección 5 encapsula una de las experimentaciones de este trabajo, que consta de la participación en generación y anotación, así como en el análisis de anotaciones del *conjunto de datos*† *MT*.
- La sección 6 encapsula los recursos y resultados de evaluar el desempeño de modelos pre-entrenados en varios *conjuntos de evaluación*†.
- La sección 7 da un cierre al trabajo, sintetizando el mismo y listando sus conclusiones generales y enumerando posibles trabajos a futuro.
- El apéndice A amplia y complementa los resultados presentados en la sección 6.2.4.
- El apéndice B es un complemento al marco teórico y registra arquitecturas, técnicas y heurísticas que fueron evaluadas y/o estudiadas en la investigación, pero cuya profundización o inclusión quedaron por fuera del alcance de este estudio.
- El glosario C documenta varios términos usados a lo largo de este informe. Referenciando en la misma a las secciones en donde los mismos fueron definidos, o en el caso de no haber sido un término central al trabajo su definición, generado por el tutor de este trabajo y anotado por alumnos en un módulo de taller.
- Por último se encuentra la bibliografía del trabajo, que contiene todos los trabajos citados por el mismo.

El glosario C fue creado con el fin de permitir referenciar a definiciones de conceptos previamente descritos, o agregar definiciones a conceptos no tan centrales al estudio sin tener que definirlos en el marco teórico, como son los conceptos más propios al *aprendizaje automático*† y no centrales al conteo de multitudes.

Se destaca como las referencias al glosario están en itálica y poseen el símbolo † como por ejemplo el término *aprendizaje automático*† definido en el párrafo anterior. Al leer el informe en formato de PDF, realizar click en los términos conduce a la entrada respectiva en el glosario, la cual lista todos sus términos en orden alfabético, y mantiene registro de en qué páginas cada término ha sido referenciado.

Capítulo 2

Definición del problema

Este capítulo introduce y define la tarea del conteo de multitudes, así como cuáles son las mayores dificultades y problemas intrínsecas a la misma.

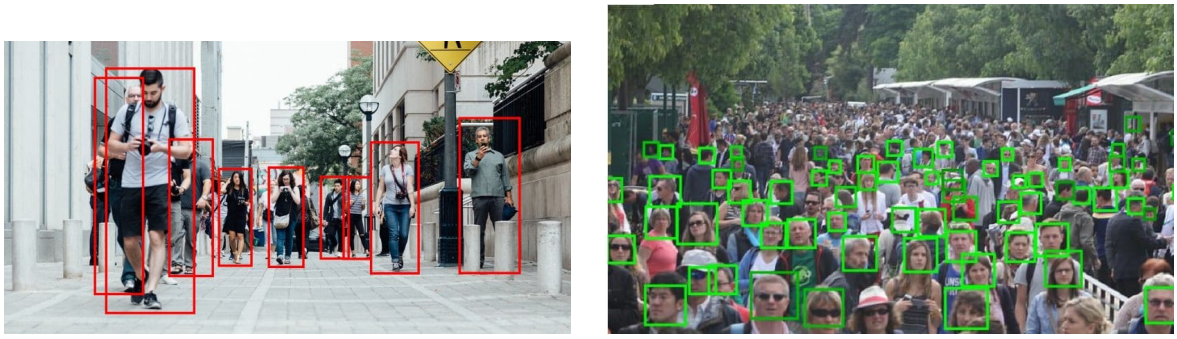
2.1. Conteo de multitudes

La tarea del conteo tiene como objetivo de, dada una imagen, poder indicar el número de elementos a contar presentes en la misma (y opcionalmente alguna noción de su posición en el espacio). En particular, el conteo de multitudes (o *Crowd Counting* en inglés) refiere a poder indicar la cantidad de personas dentro de una multitud.

Este campo es interesante debido a su gran potencial de aplicación, como por ejemplo:

- Monitoreo y estimación de la cantidad personas en espacios abiertos como ciudades, plazas y eventos (vigilancia y control de tránsito).
- Análisis histórico de densidades de personas en espacios públicos a lo largo del tiempo, que es de gran utilidad para el campo del urbanismo, la seguridad pública y los planes de evacuación de emergencia o terremoto. (*Bhangale et al.*, 2020)
- En situaciones con alta densidad de individuos en un espacio, un sistema con este mecanismo puede emitir alertas ante multitudes cuya población supere el rango normal, con lo cual se podría haber anticipado un riesgo antes del incidente de la *estampida de Año Nuevo en Shanghai*, ((*Corporation*, 2015)) donde fallecieron 36 personas en una avalancha durante las celebraciones de año nuevo.

Un pensamiento intuitivo al imaginarse este problema podría ser compararlo con el campo de la detección de objetos (como es la detección de personas), que es un campo de gran auge hoy en día.



(a) Detección de personas (*Visailabs*, 2021) (b) Detección de cabezas (*Sam et al.*, 2019)

Figura 2.1: Aplicación de algoritmos de detección de objetos

Sin embargo, una gran dificultad del conteo de multitudes suele recaer en la *variación de escala*†. Es aquí donde los modelos de detección sufren gravemente con los problemas de *variación de escala*† y *oclusión*†, como se puede apreciar en la Figura 2.1

Figura 2.2: Ejemplos de multitudes (*Zhang et al.*, 2016b)

Se puede ver en la figura 2.2 que cuanto mayor densidad de personas en la multitud, más difícil será el problema de contar correctamente la cantidad de individuos, aunque también más tolerante al error podría llegar a ser un resultado (dado que no es lo mismo contar de menos 10 personas en una imagen que tenga 100 personas que en una con 1000). En la siguiente sección se comentan los principales los factores que dificultan la resolución del problema.

2.2. Dificultades del conteo de multitudes

Un modelo robusto de conteo de multitudes apunta a predecir correctamente la cantidad de individuos en una multitud, así como estimar la densidad en dicha multitud, ofreciendo buenos resultados en diversas condiciones del dominio. (*Khan et al.*, 2022)

2.2.1. Adaptación al dominio

Dependiendo del *conjunto de datos*†, puede que el mismo conste de ejemplos de una cámara fija (figura 2.3), o de distintas vistas o escenas (figura 2.4). Esto es un problema general en el *aprendizaje automático*† y debido a la naturaleza del *entrenamiento*† de modelos con *redes neuronales*†, esta característica es un arma de doble filo.

Figura 2.3: Escena única (*Chen et al.*, 2012)

Por un lado, la dificultad del problema se reduce si todos los ejemplos son de la misma escena. Pero por otro lado, el modelo resultante estará ajustado a dicho dominio, desarrollando sesgos y dependencias a la escena, y muy seguramente teniendo un rendimiento peor si se quiere aplicar en otros dominios disimilares a la escena con la que fue entrenado el modelo (*Khan et al.*, 2022).

Figura 2.4: Escenas múltiples (*Zhang et al.*, 2016b)

2.2.2. Densidad de la multitud

La cantidad de personas varía entre imágenes, como se puede apreciar en la figura 2.5. Por lo general las imágenes con poca densidad de personas son más fáciles de resolver que las de mayor densidad. (*Khan et al.*, 2022)

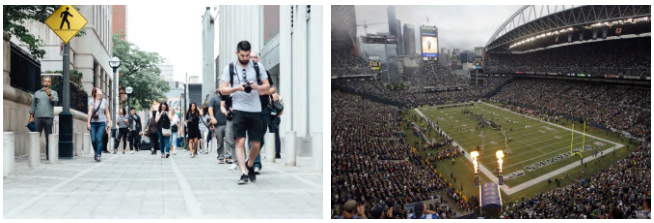


Figura 2.5: Densidad de la multitud (*Sindagi et al.*, 2020)

2.2.3. Variación de escala

La variación de la escala refiere a la situación en la que objetos relacionados (como el tamaño de las personas y sus cabezas) aparecen en distintos tamaños en una única imagen (o a lo largo de múltiples imágenes del *conjunto de datos*†).

La *variación de escala*† dependerá de la distancia y ángulo entre la cámara y los objetos, pero en particular se puede conectar con la *distorsión de perspectiva*†, que consiste en que para la perspectiva de un observador, el tamaño de los objetos disminuyen cuanto más cerca están a un punto de fuga.

Como se aprecia en la figura 2.6, este es uno de los problemas más comunes y que más impacta el desempeño de los modelos de conteo, por lo que es un área de interés y frecuente investigación para los trabajos en el campo (como veremos más adelante). (*Khan et al.*, 2022)

Figura 2.6: *Variación de escala*† (*Idrees et al.*, 2018b)

2.2.4. Distribución no uniforme

Dentro de una imagen, es importante tener en cuenta como se distribuye la *densidad de la multitud*†. Se puede apreciar una imagen con densidad uniforme y otra con densidad no uniforme en la figura 2.7.

En algunos casos la misma suele estar distribuida de manera uniforme, como es en casos de manifestaciones o individuos en asientos (a distancias constantes).

Sin embargo, dependiendo del dominio, es común que pese a la existencia de sub-grupos suficientemente uniformes entre sí, existan individuos aislados o espacios vacíos (o en contraparte, aglomeraciones). Estos casos son aún más presentes en imágenes de vías públicas.

En general, cuanto más uniforme sea la distribución de *densidad de la multitud*† más fácil es resolver el problema de conteo sobre la misma. (*Khan et al.*, 2022)

Figura 2.7: Uniformidad en distribuciones (*Zhang et al.*, 2016b)

2.2.5. Fondo complejo

Las regiones de una imagen que no contienen objetos de interés suelen variar entre y dentro de las imágenes. En particular cuando el color de los píxeles en el fondo se parece mucho al de los objetos a detectar, puede que un modelo genere falsos positivos al procesar esa región. (*Khan et al.*, 2022)

También puede que haya colores o patrones en un fondo que un modelo nunca haya visto como parte de su *entrenamiento*†, y esto pueda terminar generando resultados inesperados.

Otro problema puede darse cuando en una imagen hay objetos similares a una persona, como estatuas o carteles con fotografías de personas.

Ejemplos de estos fenómenos se pueden apreciar en la figura 2.8.

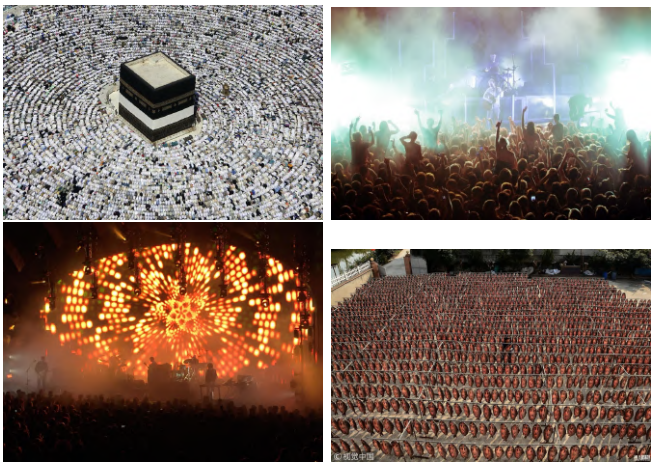


Figura 2.8: *Fondo complejo*† (*Idrees et al.*, 2018b; *Sindagi et al.*, 2020; *Wang et al.*, 2020c)

2.2.6. Oclusión

La *oclusión*† refiere a objetos superponiéndose entre sí. Cuando objetos similares se superponen se le llama *oclusión intraclase*, mientras que cuando objetos distintos se superponen (personas con paredes, autos o árboles) se le llama *oclusión interclase*. Ejemplos se pueden apreciar en la figura 2.9

Adicionalmente, la *oclusión*† puede ser parcial (alguna parte del objeto es visible), o total (un objeto obstruye totalmente al otro). Idealmente si la distribución de la *densidad de la multitud*† es muy cercana a uniforme, en los puntos que rodean a la *oclusión*† (como un árbol tapando una vista aérea de la calle), es posible inferir que atrás del árbol la distribución se mantiene. Sin embargo, esto no deja de ser una suposición que no necesariamente refleja la realidad.

Que un modelo sea robusto *oclusiones*† parciales suele ser un desafío. No sólo es difícil para quienes anotan las imágenes anotar objetos en la presencia de la *oclusión*†, pero también lo es para el aprendizaje en base a ejemplos a la hora del *entrenamiento*†, dado que limita la información visible de los objetos y dificulta poder discriminar entre los bordes de los mismos (*Khan et al.*, 2022).

Figura 2.9: *Oclusión*† (*Sindagi et al.*, 2020)

2.2.7. Variación de iluminación

La iluminación de una imagen puede variar dependiendo de múltiples factores.

A nivel general variará con distintos fenómenos del día, como por ejemplo el paso del tiempo. Cuanto más oscurezca en el día, más difícil será diferenciar entre los colores, hasta el punto que una persona puede llegar a ser completamente indistinguible del resto del fondo. Esto se puede apreciar en la *Figura 2.10*, que proviene de un *conjunto de datos*† con cámaras térmicas.

Figura 2.10: Dificultades extremas de iluminación (*Liu et al.*, 2021a)

Además, la iluminación puede variar a nivel local, como por ejemplo según como estén dispuestos los focos de luz en el espacio. Como se aprecia en las imágenes en la *Figura 2.11*, estos pueden variar su color, lo que puede cambiar el color de objetos en el escenario (que reflejen dicha luz). Adicionalmente, si objetos obstruyen la fuente de luz, esto genera áreas de sombra, que también pueden agregar dificultad a la tarea del conteo.

Figura 2.11: *Variación de iluminación*† (*Sindagi et al.*, 2020)

Cualquiera de estos factores alterará los valores de píxeles que componen a una entidad, lo cual suele hacer el problema de reconocimiento y conteo más difícil.

2.2.8. Fenómenos climáticos

Las imágenes en el exterior variarán según el tiempo atmosférico en el que fueron capturadas. Normalmente estos factores causan *variación de iluminación*† u *oclusión*†.

Figura 2.12: Variación climática (*Sindagi et al.*, 2020)

En general no son problema si el cielo se encuentra despejado o nublado, pero otros tiempos atmosféricos (como pueden ser la lluvia y tormenta) volverán más compleja la imagen, como se puede apreciar en las imágenes de la figura 2.12.

Dentro de las condiciones más extremas se encuentran las tormentas eléctricas, la nieve y la presencia de niebla.

2.2.9. Otras dificultades

La lista anterior no es exhaustiva, y otros múltiples factores pueden afectar el desempeño de un modelo de conteo de multitudes.

Algunos se puden apreciar en las imágenes de la figura 2.2.9 y son por ejemplo:

- Imágenes en **escala de grises**, que disminuyen la cantidad de información de entrada (un solo canal en lugar de los tres canales RGB).
- Imágenes **distorsionadas**, debido a una foto fuera de foco, en movimiento o de mala calidad.
- Imágenes con **ruido**.
- El hecho de que las personas pueden tener apariencias distintas, sea por motivos biológicos o por los tipos y colores de prendas que usen.

Figura 2.13: Otras dificultades. Primeras dos imágenes de (*Idrees et al.*, 2013), la tercera del conjunto *MT* (sección 5.1) y la cuarta extraída de Google imágenes.

Capítulo 3

Recursos disponibles

Dentro del campo del conteo de multitudes existe una gran cantidad de recursos útiles en su desarrollo. Esta sección apunta a enumerar los mismos, focalizándose principalmente en los *conjuntos de datos*† 3.1

Se destaca que un recurso central al estudio del conteo de multitudes es el repositorio: *Awesome Crowd Counting* en *GitHub* (*Gao and Zeng*, 2022). El mismo se encarga de documentar y compilar muchos de los recursos disponibles al conteo de multitudes que se documentan en esta sección.

Dentro de los artículos disponibles, se destacan tres estudios que fueron centrales a este trabajo:

- *CNN-based Density Estimation and Crowd Counting: A Survey* (*Gao et al.*, 2020). La continuación de una secuencia de resúmenes del estado del arte del conteo de multitudes. El mismo se centra más en la parte histórica y en arquitecturas a alto nivel.
- *Real-time implementation of counting people in a crowd on the embedded reconfigurable architecture on the unmanned aerial vehicle* (*Gong*, 2020). Tesis que hace un análisis más técnico del campo y sus modelos, centrándolo a la implementación en vehículos aéreos.
- *Revisiting Crowd Counting: State-of-the-art, Trends, and Future Perspectives* (*Khan et al.*, 2022). Trabajo muy reciente, que a diferencia de los anteriores busca ser un resumen más sencillo y enfocado a los conceptos prácticos y al estado actual.

Estos trabajos tuvieron un rol fundamental en este estudio, tanto sirviendo como guías y punteros a qué artículos estudiar en mayor profundidad, así como brindando definiciones acerca de técnicas y métodos empleados (como *funciones de pérdida*† 4.4.2 y *métricas de evaluación*† 4.3).

El conocimiento extraído de los mismos ha sido en gran medida expandido, complementando su contenido con información de otros resúmenes o artículos estudiados, así como yendo a la fuente principal para buscar más detalles cuando era necesario.

3.1. Conjuntos de datos

3.1.1. Formato de anotación

Las instancias de un *conjunto de datos*† para el conteo de multitudes constan de imágenes, y para poder entrenar o evaluar modelos con estas, es deseable que para cada imagen hayan valores anotados acerca de la misma.

Si bien el objetivo elemental del problema de conteo es determinar la cantidad total de personas (y por ende este podría ser el único dato anotado), es deseable contar con más información de ser posible.

Los *conjuntos de datos*† están anotados de forma tal que, dada una imagen, la anotación es un conjunto de puntos en un espacio bidimencional restringido al tamaño de la imagen (ancho $\times$ altura en píxeles), siendo un punto cada entidad etiquetada (personas en el caso del conteo de multitudes).

Dependiendo del *conjunto de datos*†, estas anotaciones pueden estar en formatos distintos. En general, siempre contienen una lista de coordenadas (x, y), que marca la ubicación de cada individuo en la imagen.

- En la mayoría de los casos, los datos etiquetados se encuentran en formato `.mat`, y contienen una matriz $N \times 2$ con las coordenadas para cada uno de los N individuos anotados (*Zhang et al.*, 2016b; *Idrees et al.*, 2013, 2018b; *Chan et al.*, 2008; *Chen et al.*, 2012; *Ding et al.*, 2018).
- En otros casos el formato es `.mat`, pero los datos son una matriz binaria con las mismas dimensiones que la imagen y marcando un 1 en cada punto de anotación (*Hu et al.*, 2020a).
- Otro formato común es una una estructura `.json` marcando una lista de coordenadas (x, y) (*Wang et al.*, 2020c; *Liu et al.*, 2021a; *Yan et al.*, 2019b). Este formato se puede apreciar en la figura 3.2
- Anotaciones que cuenten con aún más datos (como por ejemplo las coordenadas del rectángulo que encierre cada cabeza, o clasificaciones relevantes sobre cada individuo) pueden incluirse dentro de la estructura del formato `.json`, o también en un formato tabular como por ejemplo `.csv`, dedicando cada columna a un atributo (*Sindagi et al.*, 2020; *Lim et al.*, 2014; *Li et al.*, 2022a). Este formato se puede apreciar en la figura 3.1.
- Por último, los conjuntos que tengan otro tipo de información de la escena, como puede ser un sensor de temperatura, o un sensor de profundidad para medir la distancia al foco de la cámara, suelen proveer otra imagen o matriz conteniendo esta información para cada punto en la imagen (*Liu et al.*, 2021a; *Peng et al.*, 2020b; *Lian et al.*, 2019). Este formato se puede apreciar en la figura 3.3

Una particularidad de los *conjuntos de datos*† asociados a este problema, es que suelen estar ya previamente divididos en cuanto a sus *conjuntos de entrenamiento*† y *conjuntos de evaluación*†.

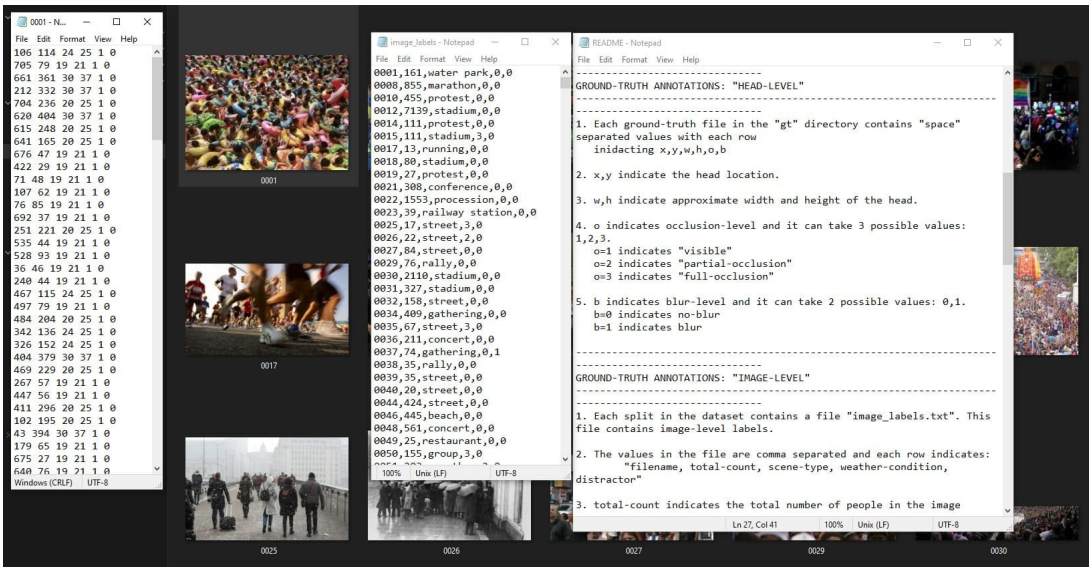


Figura 3.1: Anotación tabular de imagen y del conjunto (*Sindagi et al.*, 2020)

```
{
    "img_id": "aj04.jpg",
    "human_num": 79,
    "boxes": [],
    "points": [
        {
            "x": 35.04000000000001,
            "y": 896.4
        },
        {
            "x": 422.76,
            "y": 840.24
        },
        {
            "x": 645.24,
            "y": 778.68
```


Figura 3.2: Anotación .json (*Wang et al.*, 2020c)

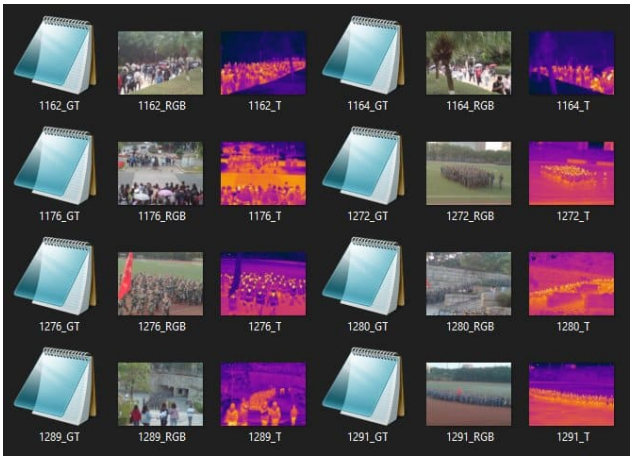


Figura 3.3: Anotaciones térmica (*Liu et al.*, 2021a)

3.1.2. Clasificación de conjuntos de datos

A continuación se definen los términos y atributos de un *conjunto de datos*† en el área de conteo de multitudes.
Los conjuntos de datos se pueden clasificar según cómo fueron obtenidos de la siguiente manera:

- **Cámara Fija**: También referenciada como CCTV (acrónimo de *Closed-Circuit Television*) y sinónimo de *Video Surveillance*, estas imágenes fueron capturadas por cámaras de seguridad situadas en edificios o la vía pública.
- **Cámara Libre**: Estas imágenes fueron capturadas por una persona con una cámara desde varios lugares distintos; pueden tener fotos de la misma localización, pero no suelen ser desde un punto y ángulo anclados.
- **Internet**: Estas imágenes fueron recolectadas de búsquedas por Internet, suelen cubrir un gran conjunto de localizaciones y ser más diversas en cuanto a su dominio.
- **Drones**: Estas imágenes fueron capturadas por drones, lo cual no implica necesariamente que hayan sido capturadas a una gran altitud, pero sí propone un sesgo al dominio de dicho conjunto en cuanto a la distancia y al ángulo entre la cámara y el plano.
- **Sintético**: Estas imágenes fueron generadas artificialmente. En este momento existe un solo conjunto de datos sintético llamado **GCC**, acrónimo de *GTAV Crowd Counting* (*Wang et al.*, 2019). Acorde a su nombre, estas imágenes fueron capturadas usando como motor de simulación el videojuego *Grand Theft Auto V*.

Adicionalmente, respecto a las relaciones entre instancias de un *conjunto de datos*†, otra posible particularidad del conjunto es si el mismo tiene la noción de escenas o no. Se considera que dos instancias pertenecen a la misma escena si tienen el mismo fondo y perspectiva, y por ende fueron capturadas por cámaras en el mismo ángulo y posición.

Esto permite clasificar los conjuntos de datos en los siguientes tipos:

- **1 escena**: En estos conjuntos todas las imágenes fueron capturadas con la misma cámara, en el mismo ángulo y localización. Son los conjuntos más susceptibles a la *adaptación al dominio*†. Los conjuntos de **Cámara fija** suelen ser de este tipo.
- **K escenas**: En estos conjuntos las cámaras se fijaron en distintas localizaciones, manteniendo su ángulo fijo y capturando imágenes en distintas escenas. Los conjuntos del tipo **Cámara fija** suelen ser de este tipo (cuando cuentan con múltiples cámaras).

- **Sin escenas**: En estos conjuntos existen imágenes que no comparten escenas con otras imágenes. Los conjuntos del tipo **Cámara Libre**, **Internet** y **Drones** suelen ser de este tipo.

Por último, los conjuntos de datos pueden tener **otros atributos** además de la localización de los individuos, o la lista de coordenadas que encuadran las cabezas de los mismos. Por ejemplo:

- **Térmico**: Estas imágenes fueron capturadas con una cámara térmica, y para cada punto adjuntan la temperatura capturada en el mismo (*Peng et al.*, 2020b; *Liu et al.*, 2021a).
- **Profundidad**: Estas imágenes fueron capturadas con una cámara con sensor de distancia, y para cada punto se adjunta su profundidad (*Lian et al.*, 2019).
- **Aéreo**: Estas imágenes fueron capturadas a una altura particularmente elevada (*Bahmanyar et al.*, 2019).
- **Sonido**: Estas imágenes fueron capturadas de videos que también capturaban sonido, y para cada fotograma se adjuntan unos segundos de sonido que ocurrieron al mismo tiempo (*Hu et al.*, 2020a).
- **Escala de grises**: Estas imágenes contienen un solo canal que representa el nivel de gris (*Idrees et al.*, 2013).
- **Estimación de perspectiva**: Las instancias del conjunto **WorldExpo'10** (*Zhang et al.*, 2015) contienen una matriz que estima la perspectiva en la imagen, en base a un modelo de regresión entrenado con anotaciones de pies a cabeza para cada persona. Estas anotaciones asumen que todas las personas miden 1.75m.
- **Muestras distractoras**: Presentes por ejemplo en el conjunto JHU-Crowd (*Sindagi et al.*, 2020) y llamadas negativas en el conjunto NWPU-Crowd (*Wang et al.*, 2020c). Estas imágenes no contienen personas, y pueden además tener *fondos complejos*[†] y una multitud de objetos/criaturas que puedan llegar a engañar a un modelo de conteo de multitudes (contándolas erróneamente como personas). Ejemplos de las mismas pueden apreciarse en la figura 3.4.

(a) Distractors in JHU-CROWD

(b) Negative Samples in NWPU-Crowd

Figura 3.4: Ejemplos de muestras distractoras en *JHU-Crowd* (*Sindagi et al.*, 2020) y *NWPU-Crowd* (*Wang et al.*, 2020c)

3.1.3. Conjuntos de datos

A continuación se presentan los *conjuntos de datos*† disponibles para la tarea del conteo de multitudes, así como su clasificación, observaciones, la resolución de sus imágenes, y el promedio de personas capturadas en cada instancia del mismo.

Se destaca además que los conjuntos de datos más relevantes al campo de estudio, y que son usados en *tablas de resultados* para comparar el modelo de desempeño entre modelos, tendrán su nombre en **negrita**.

Conjunto	Año	Obtención	Imágenes	Resolución	Promedio	Observaciones
UCSD	2008	Cámara fija	2000	238 × 158	25	1 escena
Mall	2012	Cámara fija	2000	640 × 480	31	1 escena
UCF_CC_50	2013	Internet	50	Varía	1280	Escala de grises
AHU-Crowd	2014	Cámara fija	107	576 × 576	421	20 videos
WorldExpo'10	2015	Cámara fija	3920	576 × 720	50	108 escenas, Est. Persp.
ShanghaiTech-A	2016b	Internet	482	Varía	501	-
ShanghaiTech-B	2016b	Cámara libre	716	768 × 1024	123	-
CityUHK-X	2017	Cámara fija	3191	Varía	33	55 escenas
UCF-QNRF	2018b	Internet	1535	800 × 600	815	-
Beijing-BRT	2018	Cámara fija	16795	360 × 640	13	4 escenas
SmartCity	2018	Cámara fija	50	1920 × 1080	7	10 escenas
Drone-Crowd	2019	Drones	33600	1920 × 1080	145	-
DLR-ACD	2019	Drones	33	Varía	6857	Aéreo
ShanghaiTechRGBD	2019	Cámara libre	2193	1920 × 1080	66	Profundidad
Fudan-ShanghaiTech	2019	Cámara fija	15000	1920 × 1080	27	13 escenas
Crowd Surveillance	2019b	Internet	13945	1342 × 840	28	-
Venice	2019c	Video libre	167	1280 × 720	?	1 escena
CityStreet	2019	Cámara fija	500	2704 × 1520	?	5 escenas
GCC	2019	Sintético	15211	1920 × 1080	501	GTAV. 400 escenas
JHU-CROWD++	2020	Internet	4372	1450 × 900	346	100 muestras negativas
NWPU-Crowd	2020c	Internet	5109	Varía	418	351 muestras negativas
DISCO	2020a	Cámara fija	1935	1920 × 1080	88	Sonido, múltiples escenas
DroneRGBT	2020b	Drones	3600	640 × 512	49	Térmico
VisDrone-People	2021a	Drones	3347	Varía	32	-
RGBT-CC	2021a	Cámara libre	2030	640 × 480	68	Térmico, día-noche
VSCrowd	2022a	Cámara fija	62938	1920 × 1080	37	634 Videos
Fudan-UCC	2022a	Internet	4000	Varía	-	No anotado

Tabla 3.1: Conjuntos de datos para el conteo de multitudes. En **negrita** se destacan los empleados en *tablas de resultados*

En general, las distribuciones de densidad entre estos conjuntos son muy distintas. *ShanghaiTech A* (*Zhang et al.*, 2016b) tiende a mostrar multitudes congestionadas, mientras que *ShanghaiTech B* (*Zhang et al.*, 2016b) contiene imágenes de relativamente baja densidad, estando todas en la vía pública (calles). *UCF-QNRF* (*Idrees et al.*, 2018b) tiende a contener multitudes aún más densas que *ShanghaiTech A* y contiene mayor diversidad en los fondos de las imágenes, y las de *NWPU-Crowd* (*Wang et al.*, 2020c) presentan una enorme diversidad en cuanto a escala, densidades y fondos. Todas estas diferencias sesgan a los modelos entrenados con uno solo de estos conjuntos a funcionar mejor para instancias que presenten similitudes (*Chen et al.*, 2021).

3.1.4. Técnicas de aumentado de datos

El aumentado de datos, o *data augmentation* en inglés, refiere al método usado para incrementar la cantidad de datos, introduciendo a su vez mayor diversidad en los mismos, a partir de generar más instancias de un *conjunto de datos*† en base a las pertenecientes al mismo. El objetivo principal de la misma es mejorar el desempeño de un modelo entrenado, pero también se busca fabricar datos ya previamente anotados que vuelvan al modelo menos dependiente del dominio, reduciendo el problema del *sobreajuste (overfitting)*†.

A las instancias del *conjunto de datos*† se le aplican diversos tipos de técnicas como por ejemplo: invertir imágenes horizontalmente, recortarlas, cambiar su resolución, rotarlas, agregarles ruido aleatorio, cambiar su brillo o contraste, entre otras.

Dependiendo del problema a abordar es importante elegir correctamente un subconjunto de dichas técnicas. En el contexto del conteo de multitudes, las transformaciones más aplicadas comúnmente son:

- **Recortes aleatorios en la imagen** (*Zhang et al.*, 2015; *Boominathan et al.*, 2016; *Zhang et al.*, 2016a; *Li et al.*, 2018; *Liu et al.*, 2018a; *Thanasutives et al.*, 2021; *Gao et al.*, 2019b), los cuales ayudan a mejorar el *entrenamiento*† con respecto a la *variación de escala*†.
- **Inversión horizontal** (*Li et al.*, 2018; *Liu et al.*, 2018a; *Thanasutives et al.*, 2021; *Gao et al.*, 2019b), la cual ayuda con el problema de la *distorsión de perspectiva*† (*Khan et al.*, 2022).

Otras transformaciones usadas, aunque menos frecuentemente, han sido:

- Introducción de **ruido aleatorio** (*Liu et al.*, 2018a)
- **Inversión vertical** (*Liu et al.*, 2018a)
- **Rotación** de imágenes (*Peng et al.*, 2020b).

3.1.5. Sesgos y particularidades observadas con los conjuntos de datos

En el estudio de este campo, se destacan ciertas observaciones sobre el manejo y la anotación de los *conjuntos de datos*†:

- Como se puede observar en la sección 3.1.1, no hay un formato estándar para guardar y registrar las anotaciones.
- No se pudo encontrar un estudio estadístico sobre la calidad de anotaciones en los *conjuntos de datos*†.
- No parece haber una guía de anotación que establezca la metodología de etiquetado para los datos (aparte de etiquetar cabezas). Esto deja muchas preguntas y ambigüedades en el campo, tales como son las anotaciones de personas cuya cabeza está ocluida por algo, o fuera de la imagen.

Este último punto es particularmente interesante a la hora de pensar en el problema de estimar la cantidad de personas en una imagen. Uno podría verse tentado a frente a la oclusión dar una anotación conservadora, puesto que por particularidades del *aprendizaje automático*† será probablemente más fácil entrenar un modelo con dichos datos. Sin embargo, para una imagen con una alta uniformidad en la densidad y distribución de su multitud (como por ejemplo la foto de una manifestación), es irreal pensar que no hay personas detrás de carteles, o abajo de las ramas de árboles que obstruyan en la foto.

Por otro lado, es sumamente difícil que un anotador llegue a una anotación correcta de la cantidad de personas. Dado que las *oclusiones*† totales eliminan completamente la información de qué hay detrás, como se puede apreciar en la figura 3.5.

Figura 3.5: Casos de *oclusión*† total (*Wang et al.*, 2020c)

También se destaca que existen imágenes en las que ya sea por su calidad de imagen, o por la *variación de escala*†, llega un punto al ampliar la imagen en el que no es posible diferenciar claramente individuos en el horizonte. Como se puede apreciar en la figura 3.6.

Figura 3.6: Casos de personas en el horizonte (*Wang et al.*, 2020c)

3.1.6. Conjuntos de datos de conteo alternativo

Por último, si bien la gran mayoría de recursos y avance del campo del conteo están enfocados al conteo de multitudes, gracias a los avances en la investigación de este campo, y a la naturaleza de que los datos etiquetados están en un formato asociado al problema general del conteo (no necesariamente de multitudes), se han realizado *conjuntos de datos*† y estudios en distintos dominios. A continuación se destacan algunos de estos conjuntos de datos alternativos:

Conjunto	**Año**	**Obtención**	**Imágenes**	**Resolución**	**Promedio**	**Observaciones**
TRANCOS (Vehículos)	2015	Cámara fija	1244	640 × 480	37.6	-
Penguins (Pingüinos)	2016	Cámara fija	80095	Varía	7	40 escenas
CARPK (Vehículos)	2017	Drones	1448	1280 × 720	62	212 escenas, clima
WebCamT (Vehículos)	2017	Cámara fija	60000	352 × 240	?	-
MTC (Borlas de maíz)	2017	Cámara fija	361	Varía	?	16 escenas
DCC (Células)	2018	Cámara fija	177	Varía	34	-
Wheat (Espigas de trigo)	2018	Cámara fija	20	Varía	1005	-
VisDrone-Vehicles (Vehículos)	2022	Drones	5303	991 × 1511	37.5	-

Tabla 3.2: Conjuntos de datos para el conteo de dominios alternativos

Estos conjuntos pueden ser de interés tanto para entrenar un modelo en base a los mismos, como para aplicar un re-*entrenamiento*† sobre un modelo de conteo, re-ajustando sus pesos con estos ejemplos y beneficiándose del *aprendizaje por transferencia*† para adaptarlos a otro dominio (*Gao et al.*, 2020).

3.2. Herramientas de etiquetado

Se destaca la herramienta de código abierto ***CCLabeler*** (*Wang et al.*, 2020c) que ofrece una interfaz web en la que anotar imágenes para generar nuevos conjuntos de datos, y guarda los mismos en una estructura *.json* registrando los objetos etiquetados en una lista de coordenadas (x, y) referentes a las dimensiones de la imagen.

Cabe mencionar que la misma fue desarrollada como parte de la creación del *conjunto de datos*† **NWPU-Crowd** (*Wang et al.*, 2020c).

Es esta misma herramienta la que fue empleada en la anotación de imágenes realizada en la sección 5.1 de este trabajo. En la figura 3.7 se puede apreciar la interfaz de la herramienta.

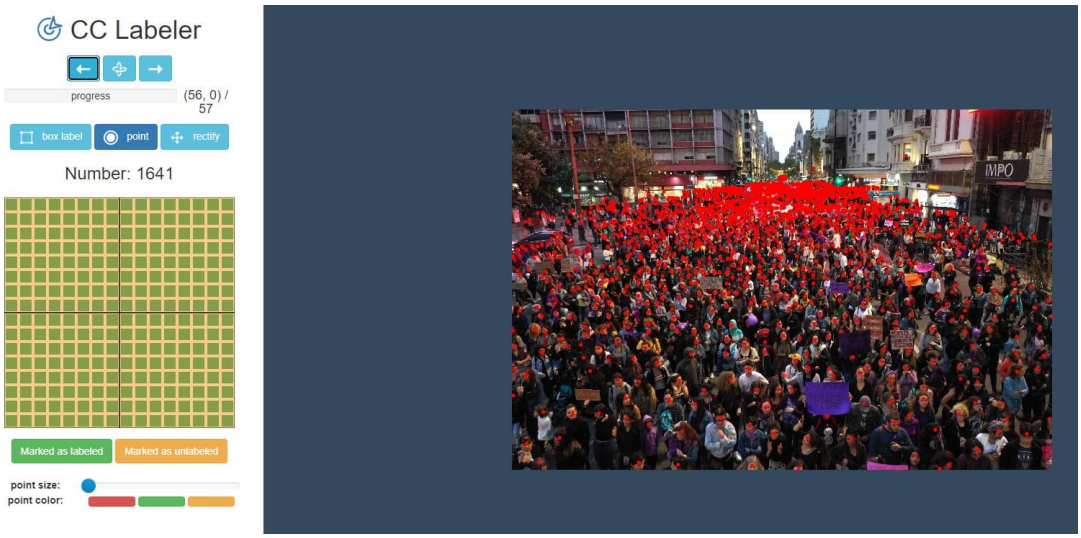


Figura 3.7: **CCLabeler**: Herramienta de anotación

3.3. Herramientas de entrenamiento y evaluación

Si bien el *entrenamiento*† es algo que quedó por fuera del alcance de este estudio, se destaca la herramienta *NWPU-Crowd Sample Code* (*Wang et al.*, 2020c). La misma se presenta como un proyecto base en el cual es posible crear, entrenar y validar modelos reutilizando funcionalidades en común.

Esta herramienta fue basada en un proyecto anterior, el C^3 *Framework* (*Gao et al.*, 2019a), que dejó de ser mantenido por la comunidad.

Si bien escapa del alcance de este estudio, se destacan estas herramientas como punteros para futuros trabajos.

3.3.1. LWCC

En cuanto a evaluar o realizar *inferencias*† en un proyecto de software, se destaca también la biblioteca de Python ***LWCC*** (*Teršek and Kljun*, 2021), que si bien no permite entrenar modelos, sí permite evaluar y obtener resultados de varios modelos contemporáneos del conteo de multitudes.

Esta biblioteca es empleada para la *evaluación*† de modelos pre-entrenados realizada en la sección 6.

Capítulo 4

Marco teórico

En la siguiente sección se presenta una breve compilación histórica de cómo ha evolucionado el campo del conteo de multitudes a lo largo de los años. La evolución del mismo ha sido impulsada por tres áreas:

1. Los avances en estudios dentro del mismo campo.
2. Los avances en estudios dentro del área general del *aprendizaje automático*†.
3. La disponibilidad creciente de recursos, tanto en forma de *conjuntos de datos*† como en poder computacional.

Además de recorrer el avance histórico del estado del arte, esta sección se centrará en detallar en profundidad los elementos que componen las soluciones contemporáneas al problema, vinculando sus técnicas y componentes con las aristas y desafíos del problema que buscan atacar.

Adicionalmente, se destaca que a nivel elemental, conceptual y general, esta sección apunta a ser lo más exhaustiva posible, basando su contenido en múltiples referencias de compilación como (*Gong*, 2020; *Gao et al.*, 2020; *Khan et al.*, 2022), profundizando en el contenido de las mismas e indagando en sus fuentes primarias.

Sin embargo, por la enorme cantidad de material disponible, el contenido cubierto en esta sección no es exhaustivo en su totalidad, ya que múltiples conceptos generales o de paradigma son priorizados frente a otros (como es el caso de los modelos basados en *aprendizaje supervisado*† frente al *aprendizaje débilmente supervisado*†, o modelos especializados en *conjuntos de datos*† más generales frente a los basados en drones).

Yendo a los antecedentes estudiados en particular, se mencionan y referencian una gran cantidad de modelos para dar una perspectiva general y cubrimiento del campo, pero sólo se profundiza los que fueron considerados de mayor relevancia.

4.1. Estrategias pioneras de resolución

Inicialmente los modelos seguían dos estrategias para la resolución del problema de conteo de multitudes: los métodos basados en **detección** y los métodos basados en **regresión**.

4.1.1. Métodos basados en detección

Con el avance del *aprendizaje automático*† resultó intuitivo intentar resolver este problema con técnicas fruto de la investigación de otros campos, como la **detección de personas** (*Leibe et al.*, 2005; *Tuzel et al.*, 2008; *Lin and Davis*, 2010), reconociendo en una imagen los recuadros que encuadran a cada persona de pies a cabeza, para luego procesar los resultados contando las entidades detectadas.

Debido al problema de la *oclusión*† en los conjuntos de datos que constan de **multitudes**, este enfoque plantea dos problemas. El primero es un problema de datos: en una gran multitud con *oclusión*† resulta difícil para los anotadores encuadrar el espacio que ocupa una persona, debido a que rara vez es posible ver debajo de la cabeza de las personas en el fondo de la multitud; por ende, es difícil etiquetar un *conjunto de datos*† con ejemplos de multitudes. El segundo es que incluso contando con dichos datos, es difícil direccionar el *entrenamiento*† de estos modelos a funcionar en contextos de *oclusión*† y alta *densidad de la multitud*†

Generalizando esta idea, otros modelos optan por usar conjuntos que encuadren partes del cuerpo, como las cabezas de los individuos, y entrenar un modelo que reconozca cabezas (*Lin et al.*, 2001; *Viola and Jones*, 2004; *Wu and Nevatia*, 2007; *Li et al.*, 2008; *Felzenszwalb et al.*, 2010; *Topkaya et al.*, 2014). Esto mejora el desempeño de los modelos, pero como se puede apreciar en la figura 2.1 siguen sufriendo en menor medida del problema de la *oclusión*†, y tampoco obtienen buen desempeño al problema de la *variación de escala*† (*Gong*, 2020).

La mayoría de estos modelos son previos al auge de las *redes neuronales*†. Debido a esto, se basan en métodos de *aprendizaje automático*† tradicionales o heurísticas, como son la regresión lineal, la regresión de cresta, procesos gaussianos, SVMs y los bosques aleatorios. Luego del auge de las *redes neuronales*†, varios modelos pioneros adoptaron las mismas (*Khan et al.*, 2022).

4.1.2. Métodos basados en regresión

Debido a las limitaciones de los *métodos basados en detección de objetos*†, se adoptaron estos métodos, que dada una imagen o un parche de la misma, intentan aprender una función para inferir el número total de personas presentes en esta a través de regresión lineal o de regresión con mezcla gaussiana (*Paragios and Ramesh*, 2001; *Tian et al.*, 2010).

En lugar de aprender a detectar partes del cuerpo, estos métodos aprenden a detectar patrones más globales, como la textura (*Haralick et al.*, 1973; *Chen et al.*, 2012), gradientes, bordes, entre otras características (*Davies et al.*, 1995; *Dalal and Triggs*, 2005; *Chan and Vasconcelos*, 2009).

Los *métodos basados en regresión*† funcionan mejor contra el problema de la *oclusión*†, baja resolución, y los *fondos complejos*†. Pero desafortunadamente no mostraron buenos resultados con imágenes que poseen una alta *densidad de la multitud*†, y terminan ignorando gran parte de la información espacial en la escena (*Gao et al.*, 2020).

4.2. Métodos basados en estimación de densidad

Los métodos anteriores fueron incursiones en el tema, pero tenían diversas carencias. Además de no tener buen desempeño en las *métricas de evaluación*†, los *métodos basados en regresión*† no retornaban ningún tipo de información espacial en la imagen; esta información existe en los datos anotados (conjuntos de puntos el espacio dimensional de la imagen), por lo que sería de interés que los modelos retornen una matriz similar que permita identificar y localizar áreas con mayor *densidad de la multitud*†.

El formato de anotación presentado en la sección 3.1.1 resulta intuitivo y coherente para la resolución de este problema, y ciertamente es sencillo y eficaz a la hora de anotar y almacenar anotaciones. Sin embargo, es extremadamente desafiante entrenar *redes neuronales*† y obtener buen desempeño si la *valor anotado*† que el modelo aprende a replicar es una matriz binaria dispersa (*Idrees et al.*, 2018a).

En 2010 (*Lempitsky and Zisserman*, 2010) propone el primer modelo que adopta un *método basado en estimación de densidad*†. Para las mismas es necesario convertir las matrices binarias anotadas en el *conjunto de datos*† a **matrices de densidad**, una matriz de igual relación de aspecto a la imagen original, pero compuesta por elementos en punto flotante, distribuidos de forma similar a la *densidad de la multitud*†, y cuya suma de elementos (integral) es igual a la cantidad de individuos anotados en la imagen. El resultado de esta transformación se puede apreciar en la figura 4.1.

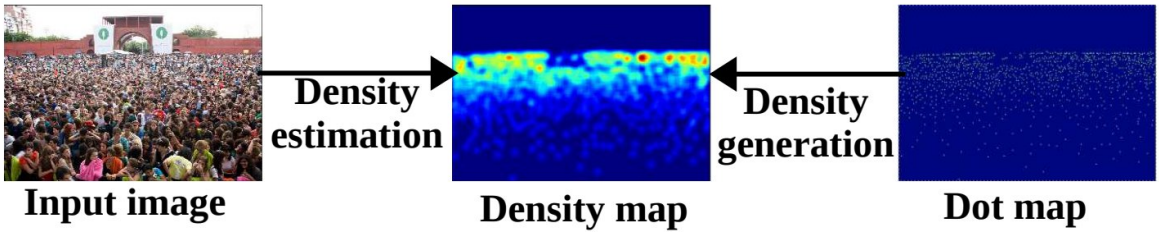


Figura 4.1: *Mapa de densidad*† (*Wan and Chan*, 2019b)

El pre-procesamiento necesario para en base a las anotaciones del *conjunto de datos*† generar para cada instancia una *mapa de densidad*† será detallado en la sección siguiente 4.2.1.

El modelo propuesto por (*Lempitsky and Zisserman*, 2010) se basa en aprender un mapeo lineal entre características locales a la imagen, y *mapas de densidad*† previamente etiquetados. Luego esta idea se generalizó a aprender un mapeo no lineal a través de bosques aleatorios (*Pham et al.*, 2015).

Estos métodos consideraban la información espacial, pero se veían limitados debido a que sus atributos tenían que ser diseñados manualmente (por ejemplo segmentar el fondo y frente de la imagen, bordes de las multitudes, etc), a causa de la naturaleza de los métodos de *aprendizaje automático*† tradicionales (*Gao et al.*, 2020).

Es aquí en donde entran los modelos basados en *redes neuronales*†, particularmente los que hacen uso de las *redes neuronales convolucionales (CNN)*†. Dichos modelos, debido a la naturaleza de su estrategia de *entrenamiento*†, pueden aprender a inferir funciones de gran complejidad a partir de los datos crudos (por ejemplo mapas de perspectiva o segmentación del frente de la imagen (*Zhang et al.*, 2015)); se profundizará sobre esto en la sección 4.4.

Se destaca que los modelos pioneros basados en *redes neuronales*† obtuvieron una mejora sustancial en comparación a los métodos anteriores, gracias a emplear *CNNs*†, (*Wang et al.*, 2015; *Fu et al.*, 2015; *Zhang et al.*, 2015; *Walach and Wolf*, 2016).

Como será presentado a lo largo del marco teórico, con el paso de los años las arquitecturas y técnicas empleadas en modelos con *redes neuronales*† *CNNs*† fueron iterando e innovando con el fin de mejorar el desempeño de los modelos.

A su vez, como vimos en la sección 3.1, con el paso del tiempo se han generado *conjuntos de datos*†, y se han mostrado más ejemplos de las dificultades que tiene el problema del conteo de multitudes en la sección 2.2. Para superar estos desafíos y obtener una precisión alta sobre estos nuevos conjuntos, arquitecturas complejas de *CNN*†, métodos de aprendizaje contemporáneos, y criterios de *evaluación*†

más sofisticados se han desarrollado en los últimos años (*Khan et al.*, 2022). Se explorarán más en detalle estas particularidades en las siguientes secciones.

4.2.1. Preprocesamiento de los datos

Como se mencionó en la sección anterior, es necesario procesar los datos anotados con el fin de generar *mapas de densidad*†, para emplearlas a lo largo del *entrenamiento*† y la *evaluación*† del modelo. Para esto se suele aplicar una sumatoria de kernels gaussianos posicionados en cada cabeza.

Para cada píxel $X_i = (x_i, y_i)$ que contenga una anotación, es posible representar una función *delta de Dirac* $\bar{\delta}(X - X_i) = \delta(x - x_i)\delta(y - y_i)$, que vale cero en todos punto excepto en X_i en donde vale infinito y su integral vale 1. El *mapa de densidad*† es generado aplicando *convoluciones*† con esta misma función junto a un *filtro gaussiano*† G_σ. (*Khan et al.*, 2022)

$$Y(X) = \sum_{i=1}^{N} \bar{\delta}(X - X_i) * G_\sigma(X), \tag{4.1}$$

en donde N es el número total de puntos anotados en la imagen (la cantidad de personas) y σ es un *hiperparámetro*† que indica la desviación estándar de una gaussíana isotrópica. Se aprecia que como la integral de la *delta de Dirac* es 1, la integral de $Y(X)$ es igual a la cantidad de personas en la imagen.

Visualmente esta transformación genera un difuminado de cada anotación por cabeza, variando su escala según el valor de σ. Se puede apreciar la generación del mismo en la figura 4.2 y cómo varía según σ en la figura 4.3.

Existen múltiples formas de generar distintos *mapas de densidad*†; una de las más básicas y empleadas en trabajos tempranos es mantener σ con un valor constante (como por ejemplo $\sigma = 15$ (*Lempitsky and Zisserman*, 2010; *Zhang et al.*, 2015)). Esta estrategia funciona bien para caracterizar la densidad de objetos circulares del mismo tamaño (como células y bacterias), pero no funciona bien para objetos dispuestos en una escala variable (*Gao et al.*, 2020).

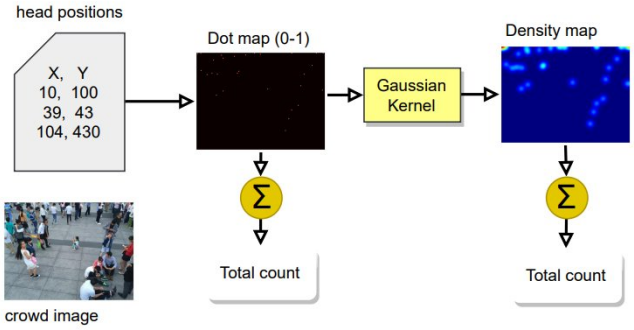


Figura 4.2: Generación de un *mapa de densidad*† (*Khan et al.*, 2022)

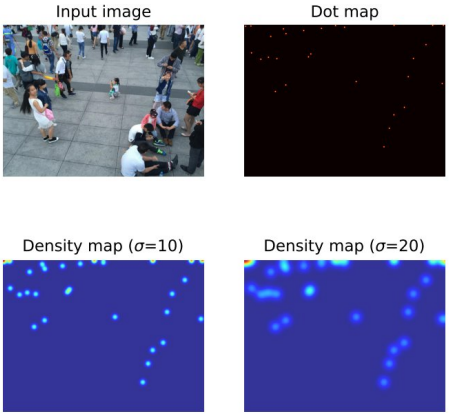


Figura 4.3: Variación según σ (*Liu et al.*, 2021a)

De mayor popularidad hoy en día es el modelado de núcleos adaptativos, en donde para cada punto anotado, se determina un valor de σ que depende de la distancia promedio (en píxeles) entre las k anotaciones más cercanas en la imagen. Con un sigma calculado se aplica el *filtro gaussiano*† en una matriz de ceros con 1 en el valor del punto, y los resultados se suman en una matriz final que se convertirá en el *mapa de densidad*†.

Esta estrategia visualmente genera un filtro más difuso en áreas de baja densidad, y menos en las de alta densidad. El valor de k es otro *hiperparámetro*† que suele variar en los trabajos: se han visto ejemplos de $k = 4$ (*Li et al.*, 2018), $k = 10$ (*Cao et al.*, 2018), $k = 20$ (*Zeng et al.*, 2017). Estos algoritmos son más complejos computacionalmente, y demoran más en ejecutarse, pero al emplearlos durante el *entrenamiento*† suelen ofrecer una mejora en el desempeño de los modelos.

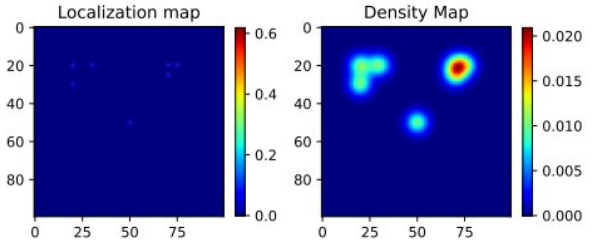


Figura 4.4: Densidad con σ variable según distancia (*Khan et al.*, 2022)

Dependiendo de la *densidad de la multitud*† en la región, varía el valor del *mapa de densidad*†. A modo de ejemplo, en la figura 4.4 hay un grupo de tres personas a cierta distancia, otro de tres personas muy solapadas y una persona sola abajo. Cuanto mayor es la *densidad de la multitud*† mayor es el valor en el *mapa de densidad*†.

Se destaca como si bien esta estrategia modela la densidad de la imagen, las distancias en base a píxeles no son una medida traducible a las distancias tridimensionales en donde la imagen fue capturada. Dos objetos muy próximos en la imagen pueden estar lejos físicamente, pero por el ángulo de la cámara sus píxeles se encuentran próximos. Esta estrategia tampoco necesariamente plantea una noción de escala, dado que un área de alta densidad no necesariamente significa que los objetos están a una mayor distancia del lente (aunque esto se encuentra correlacionado en la mayoría de los casos).

Debido a que no existe un estándar en la generación de *mapas de densidad*†, y cada enfoque tiene su propio conjunto de *hiperparámetros*†, el método de generación del *mapa de densidad*† en sí mismo es un factor con el que se puede experimentar en la creación de un modelo de conteo de multitudes, buscando generar un *mapa de densidad*† que optimice el *entrenamiento*† del modelo y logre brindar estimaciones más certeras de la cantidad de personas en una multitud.
Varios trabajos se han enfocado en alterar la generación de *mapas de densidad*† para que reflejen más correctamente los contextos locales del mundo real en regiones de cada imagen, como *Gao et al.* (2020) enumera las siguientes:

- El artículo propuesto por *Lian et al.* (2019) explota las anotaciones de profundidad para modelar el espacio tridimensional y usar sus distancias en la aplicación de K vecinos más cercanos (*Lian et al.*, 2019).
- El artículo propuesto por *Oghaz et al.* (2019) aplica un algoritmo de fuerza bruta y segmentación para a partir de las imágenes bidimensionales encontrar los K vecinos más cercanos a cada anotación (*Oghaz et al.*, 2019).
- El artículo propuesto por *Xu et al.* (2022) observa que para regiones de extremadamente alta *densidad de la multitud*†, los *mapa de densidad*† resultantes del algoritmo presentado pueden terminar causando que regiones baja densidad tengan un valor muy cercano a cero. El trabajo propone un auto escalado del mapa de densidad, para aliviar cambios bruscos entre regiones del mismo.
- El artículo propuesto por *Ma et al.* (2019) emplea métodos bayesianos de máxima verosimilitud para generar los *mapas de densidad*†.
- El artículo propuesto por *Wan and Chan* (2019a) propone refinar los *mapas de densidad*† a través de un modelo de generación secuencial.
- El artículo *Distribution Matching for Crowd Counting* propuesto por *Wang et al.* (2020a) emplea la matriz binaria de anotaciones directamente en su *función de pérdida*†.

4.3. Métricas para evaluar el rendimiento de un modelo

Existen múltiples *métricas de evaluación*† que permiten evaluar el rendimiento de un modelo de conteo de multitudes. Estas se pueden clasificar en tres categorías:

- **Métricas de evaluación globales (4.3.1)**, las cuales trabajan sobre la cantidad total de personas C_i en una imagen i, siendo C_i^{pred} la cantidad total de personas detectadas por el modelo para la imagen i y C_i^{gt} la cantidad de personas anotadas para la imagen i.
- **Métricas de evaluación por parches (4.3.2)**, las cuales segmentan los *mapas de densidad*† (D_i^{pred} y D_i^{gt}) en parches y para cada parche calculan la *métrica de evaluación*†.
- **Métricas de evaluación a nivel de píxel (4.3.3)**, las cuales trabajan comparando cada punto (j, k) del los *mapas de densidad*† comparando $D_{i,j,k}^{pred}$, con su correspondiente $D_{i,j,k}^{gt}$.

Una particularidad de la aplicación de estas métricas en la metodología de evaluación del área del conteo de multitudes, es que la aplicación de las mismas no pasa por un proceso de normalización.

A modo de ejemplo, definiendo el error de una imagen como la diferencia en la cantidad de personas predicha y la cantidad anotada $E(i) = |C_i^{pred} - C_i^{gt}|$:

- Si $C_a^{gt} = 10$ y $C_a^{pred} = 20$, entonces $E(a) = 10$.
- Mientras que si $C_b^{gt} = 4000$ y $C_b^{pred} = 4040$, entonces $E(b) = 40$.

El error $E(b)$ es cuantitativamente cuatro veces más mayor. Sin embargo, para la imagen a se predijo incorrectamente el doble de personas anotadas 200 % del valor real, mientras que para la imagen b se predijo un 101 % del valor real.

El no normalizar o regularizar las *métrica de evaluación*† acorde a la cantidad de individuos de su *conjunto de evaluación*† dificulta la ya delicada tarea de comparación entre evaluaciones en distintos *conjuntos de evaluación*†. Resulta útil que las tablas de resultados adjunten datos como la media y desviación estándar de cada conjunto.

4.3.1. Métricas de evaluación globales

Estas *métricas de evaluación*† tienen la ventaja de ser las más sencillas de implementar, sin embargo, su desventaja es que trabajando sólo con la cantidad total de personas C_i no consideran para su evaluación que la localización de personas en el *mapa de densidad*† sea correcta.

Una predicción que acierte a la cantidad total $C_i^{pred} = C_i^{gt}$ tendrá excelente desempeño, pese a que su *mapa de densidad*† D_i^{pred} refleje la realidad, o que por ejemplo agrupe a todas las personas en un área vacía.

4.3.1.1. Error Absoluto Medio (MAE)

El *Error Absoluto Medio (MAE)*†, es el promedio de diferencias absolutas entre C_i^{pred}, la cantidad total de personas detectadas y C_i^{gt}, la cantidad total de personas anotadas.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |C_i^{pred} - C_i^{gt}|, \tag{4.2}$$

en donde N es la cantidad de imágenes en el *conjunto de evaluación*†, y para cada instancia i se cuenta con un dato anotado C^{gt} y otro inferido por el modelo C^{pred}. Una *MAE*† de 0 indica que no hay error alguno en el *conjunto de evaluación*† (es decir que para toda instancia el modelo acertó correctamente en su predicción)

MAE† es **una de las *métricas de evaluación*† más usadas** en el conteo de multitudes. Una particularidad de la misma es que no es muy sensible a valores atípicos (*outliers*), por lo que un error significativo en una imagen (sea a causa de mal desempeño, o por errores en las anotaciones) acarreará menos impacto que con otras métricas (*Gao et al.*, 2020).

4.3.1.2. Error Cuadrático Medio (MSE) y Raíz del Error Cuadrático Medio (RMSE)

El *Error Cuadrático Medio (MSE)*† es el promedio de la suma de las diferencias al cuadrado entre el valor etiquetado y el valor inferido. Mientras que el *Raíz del Error Cuadrático Medio (RMSE)*† es la raíz cuadrada del *MSE*†.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(C_i^{pred} - C_i^{gt})^2; \tag{4.3}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(C_i^{pred} - C_i^{gt})^2}, \tag{4.4}$$

en donde N es la cantidad de imágenes en el *conjunto de evaluación*†. A diferencia de *MAE*†, esta *métrica de evaluación*† es más sensible a aumentar cuando pocas imágenes dentro del *conjunto de evaluación*† tienen un error muy alto (sea por mal desempeño, o por errores en las anotaciones) *Gao et al.* (2020).

Se destaca cómo los artículos y las *comparaciones de evaluación en el estado del arte* comúnmente referencian al *MSE*† para comparar modelos. Sin embargo, en realidad emplean la raíz cuadrada de la misma (*RMSE*†).

4.3.2. Métricas de evaluación por parches

Buscando una *métrica de evaluación*† que además de evaluar la cantidad total de personas anotadas, también evalué algún grado del error de localización en los *mapas de densidad*†. Varios trabajos proponen *métricas de evaluación*† basadas en parches, en donde dividen la imagen en parches mutuamente exclusivos, computan una métrica para cada par de parches y combinan (ej. sumando) las *métricas de evaluación*† de los parches que componen cada imagen.

Los artículos que proponen estas *métricas de evaluación*† discuten que son más adecuadas que las métricas globales para evaluar problemas de conteo basados en *métodos basados en estimación de densidad*†.

Pese a esto, en la práctica, debido a que la gran mayoría de trabajos emergentes sólo evalúan con *MAE*† y *RMSE*†, estas *métricas de evaluación*† son rara vez empleadas (exceptuando los artículos en donde fueron propuestas).

4.3.2.1. Error Absoluto Medio por Grillas (GAME)

Propuesta por (*Guerrero-Gómez-Olmedo et al.*, 2015), en un trabajo de **conteo de vehículos**. La *métrica de evaluación*† *Error Absoluto Medio por Grillas (GAME)*† propone partir la imagen i en 4^L parches no superpuestos, computar la *MAE*† en separado para cada parche $C_{(i,l)}$ y sumar la *MAE*† de los parches que componen la imagen:

$$GAME = \frac{1}{N}\sum_{i=1}^{N}(\sum_{l=1}^{4^L}|C_{(i,l)}^{pred} - C_{(i,l)}^{gt}|), \tag{4.5}$$

en donde N es la cantidad de imágenes en el *conjunto de evaluación*†. De esta forma *GAME*† evalúa la eficacia de un modelo de forma más robusta, considerando errores de localización en el *mapa de densidad*† (*Khan et al.*, 2022).

Nótese en la figura 4.5 que cuanto mayor el valor de L más restrictiva será la *métrica de evaluación*†, mientras que si $L = 0$, *GAME*† tendrá el mismo valor que *MAE*†.

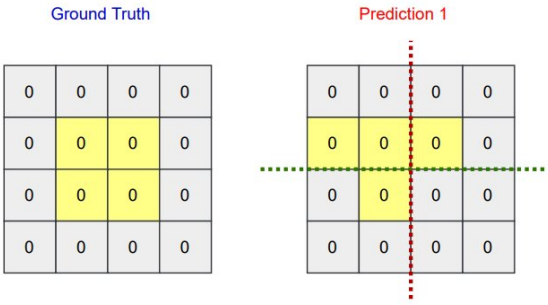


Figura 4.5: Ilustración de robustez de *GAME*† (con $L = 1$), para la predicción la *MAE*† es 0 (ignorando el error de localización), sin embargo la *GAME*† retorna 1. (*Khan et al.*, 2022)

4.3.2.2. Error Absoluto Medio por Parches (PMAE)

Buscando cubrir la misma necesidad que *GAME*†, en el trabajo que propone el modelo **PaDNet** (*Tian et al.*, 2019) se propone la *métrica de evaluación*† *Error Absoluto Medio por Parches (PMAE)*†. En la misma cada imagen se parte en m parches, y se calcula la *métrica de evaluación*† individualmente para cada uno de estos parches (como si fueran más imágenes en el *conjunto de evaluación*†):

$$PMAE = \frac{1}{m \times N} \sum_{i=1}^{m \times N} |C_{I_i}^{pred} - C_{I_i}^{gt}|, \tag{4.6}$$

en donde N es la cantidad de imágenes en el *conjunto de evaluación*† y m es el número de parches no superpuestos en cada imagen. Cuando $m = 1$ *PMAE*† es igual a *MAE*†, sin embargo difiere de *GAME*† debido a que aplica el promedio para el resultado de cada parche (en lugar de promediar la suma de *MAE*† de los parches para cada imagen).

4.3.2.3. Error Cuadrático Promedio por Parches (PMSE y RPMSE)

Análogo a *RMSE*†, se define *Raiz del Error Cuadrático Medio por Parches (RPMSE)*† como el promedio de las distancias cuadráticas para cada parche m de las imágenes N (*Tian et al.*, 2019):

$$PMSE = \frac{1}{m \times N} \sum_{i=1}^{m \times N} (C_{I_i}^{pred} - C_{I_i}^{gt})^2); \tag{4.7}$$

$$PRMSE = \sqrt{\frac{1}{m \times N} \sum_{i=1}^{m \times N} (C_{I_i}^{pred} - C_{I_i}^{gt})^2}, \tag{4.8}$$

en donde N es la cantidad de imágenes en el *conjunto de evaluación*† y m es el número de parches no superpuestos en cada imagen. Cuando $m = 1$ *RPMSE*† es igual a *RMSE*†.

4.3.3. Métricas de evaluación a nivel de píxel

Las *métricas de evaluación*† presentadas a continuación trabajan a nivel de píxel, por lo que son de mayor sensibilidad a la distribución de valores en el *mapa de densidad*†.

Cabe destacar que para su correcta aplicación, dado un *mapa de densidad*† estimado por un modelo, estas *métricas de evaluación*† tienen que ser calculadas comparando contra un *mapa de densidad*† generado con el mismo algoritmo de preprocesamiento (ver sección 4.2.1) empleado para generar el *valor anotado*† en el entrenamiento del modelo.

4.3.3.1. Error Absoluto Medio a nivel de Píxel (MPAE)

El *Error Absoluto Medio a nivel de Píxel (MPAE)*† a diferencia de las anteriores, trabaja comparando la estimación del *mapa de densidad*† píxel a píxel, acumulando el error de cada uno de los píxeles respecto al *mapa de densidad*† que se desea aprender:

$$MPAE = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W} ||D_{i,j,k}^{gt} - D_{i,j,k}^{pred}||_1 \times 1_{\{D_{i,j,k}\in R_i\}}, \tag{4.9}$$

en donde N es la cantidad de imágenes en el *conjunto de evaluación*†, $D_{i,j,k}^{gt}$ es el valor etiquetado en el *mapa de densidad*† para la i-ésima imagen en el píxel (j,k) de el *valor anotado*†, mientras que $D_{i,j,k}^{pred}$ es su análogo pero para el valor inferido por el modelo a evaluar.

El trabajo que propone esta *métrica de evaluación*† (*Liu et al.*, 2019a) le integra el concepto de *Región de Interés (RoI)*†, en donde es posible para una imagen i designar una región en donde se desean ignorar los valores etiquetados e inferidos para la imagen. De esta manera, se define un mapa binario para la *RoI*† R_i, en donde 1 indica un píxel a ser considerado y 0 a ser descartado. En general, para la mayoría de *conjuntos de datos*†, la *RoI*† es la imagen en su totalidad.

Ignorando el concepto de *RoI*†, esta métrica es análoga a la *función de pérdida* (ℓ_1)†, o MAE a nivel de píxel para las matrices de los *mapas de densidad*†.

4.3.3.2. Peak SNR (PSNR)

Peak SNR (PSNR)† mide el error entre los píxeles correspondientes entre el *mapa de densidad*† inferido y el verdadero. Sea $pixMSE_i$ el MSE a nivel de píxel para los *mapas de densidad*† D_i^{gt} y D_i^{pred}, el $PSNR$† se calcula como:

$$PSNR = \frac{1}{N}\sum_{i=1}^{N} 10 \cdot log_{10}(\frac{255^2}{pixMSE_i}), \tag{4.10}$$

en donde N es la cantidad de imágenes en el *conjunto de evaluación*†, el valor del $PSNR$† es un número real; un $PSNR$† alto significa un error bajo (las imágenes similares píxel a píxel), mientras que un valor bajo significa una diferencia mayor entre las imágenes (*Khan et al.*, 2022).

Esta *métrica de evaluación*† ha tenido mayor uso que las anteriores, aunque muy poco en comparación con MAE† y $RMSE$† Por ejemplo en los trabajos *Li et al.* (2018) y *Jiang et al.* (2019).

4.3.3.3. Índice de Medida de Similitud Estructural (SSIM)

Al medir sólo diferencias de valores píxel a píxel, el *PSNR*† no necesariamente refleja la percepción humana, dos *mapas de densidad*† completamente idénticos salvo por un pequeño sector en donde haya mucha variación (un gran cumulo de gente), no necesariamente tendrá un *PSNR*† muy bajo, dado que en general las matrices son casi idénticas.

El trabajo *Image Quality Assessment: From Error Visibility to Structural Similarity* (*Wang et al.*, 2004) define la métrica *Índice de Medida de Similitud Estructural (SSIM)*†, como una *métrica de evaluación*† perceptual para la comparación de imágenes.

La misma mide la similitud entre imágenes, teniendo en cuenta aspectos como el **brillo**, el **contraste** y la **estructura**. Su valor es un número real que oscila entre el rango de [0-1]; cuanto más bajo, más distintas serán las imágenes, y se calcula como:

$$SSIM(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma}, \tag{4.11}$$

en donde α, β y γ son hiperparámetros para cuanto pesa cada subcomponente de la métrica, y los mismos se basan en la diferencia de luminosidad ($l(x,y)$ 4.12), contraste ($c(x,y)$ 4.13) y estructura ($s(x,y)$ 4.14)

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}, \tag{4.12}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}, \tag{4.13}$$

$$s(x,y) = \frac{\sigma_{xy} + \frac{c_2}{2}}{\sigma_x\sigma_y + \frac{c_2}{2}}, \tag{4.14}$$

en donde:

- x, y son dos imágenes de tamaño $N \times N$.
- μ_x y μ_y es el valor promedio de los píxeles de x e y.
- σ_x^2 y σ_y^2 es la varianza de x e y.
- σ_{xy} es la covarianza de x e y.
- c_1 y c_2 son dos constantes hiperparamétricas que estabilizan la operación cuando el denominador es pequeño.

A la hora de aplicar *SSIM*† a un *conjunto de evaluación*† se realiza el promedio del *SSIM*† para cada instancia de evaluación del conjunto.

4.4. Aprendizaje profundo con redes convolucionales

Los *métodos basados en estimación de densidad*† y el uso de las *redes neuronales*† plantean un conjunto de desafíos interesantes a la hora de entrenar y ejecutar el modelo con el fin de generar *mapas de densidad*†. A continuación se destacan conceptos de las varias aristas que componen este tipo de metodología y sus modelos.

4.4.1. Conceptos destacados de las redes neuronales

Las *redes neuronales*†, también llamadas *redes neuronales artificiales*, tienen sus origenes en un modelo matemático algorítmico propuesto en el año 1943 por (*McCulloch and Pitts*, 1943) que apela a emular las características de las redes neuronales biológicas (para el entendimiento en dicha época).

El método extiende las ideas de los modelos de regresión lineal, agregando elementos no lineales. El modelo plantea como unidad mínima la **neurona**, o perceptrón (*Rosenblatt*, 1958), que recibe una cantidad arbitraria de entradas $(x_1, x_2, x_3, ...)$, a las cuales se agrega una entrada extra de **sesgo** b (bias), se les aplica una transformación lineal en base a sus **pesos** o coeficientes w, y luego al resultado se le aplica una *función de activación*† h, con lo que genera una salida $h_{w,b}(x)$ (*Gong*, 2020).

En una *red neuronal*† las neuronas se organizan en capas, comunicándose desde la entrada hasta llegar a la salida de la red.

Existen diversas arquitecturas de *redes neuronales*†, en este estudio se profundizará en algunas de estas, en particular entre aquellas diseñadas y optimizadas para el procesamiento de imágenes, o la visión artificial.

Existe una propiedad importante a destacar de las *redes neuronales*†, y es que con el suficiente tamaño y profundidad, estos modelos son capaces de aproximar cualquier función (*Cybenko*, 1989; *Hornik*, 1991). Por último, el trabajo *Montufar et al.* (2014) muestra como las *redes neuronales*† tienen mejor capacidad de expresión escalando en su profundidad (mayor número de capas) que en la cantidad de neuronas que contiene cada capa.

4.4.1.1. Perceptrón Multi-Capa (MLP)

La arquitectura más elemental de una *red neuronal*† es el *Perceptrón Multi Capa (MLP)*†. El mismo consta de una **capa de entrada** (que recibe la entrada del modelo), la cual es propagada por un conjunto de *capas totalmente conectadas*†, en las que para cada neurona existe una conexión unidireccional entre la salida de la capa subyacente y la entrada de la capa que le sigue, agregándole a cada neurona un sesgo con valor de 1 que permite mayor holgura a la hora de ajustar los pesos de la red durante el *entrenamiento*†. El resultado de este grupo de capas (denominado capa oculta) es finalmente propagado hasta la **capa de salida**, que retorna la salida del modelo. La figura 4.6 diagrama el *MLP*†.

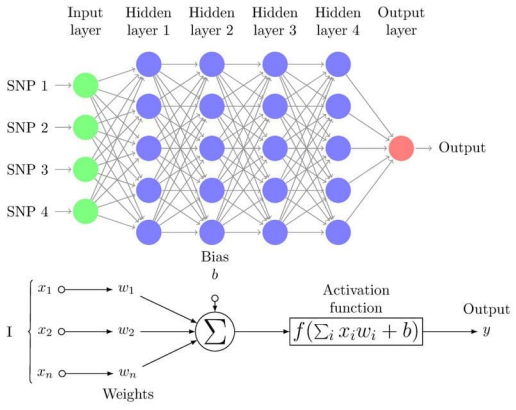


Figura 4.6: Perceptrón multi capa (*Pérez and Zingaretti*, 2019)

4.4.1.2. Inferencia

En las redes neuronales de tipo *feed-forward*, el proceso *inferencia*† (o pasaje hacia adelante) consta de propagar una entrada a lo largo de todas las capas que conforman la *red neuronal*†.

Esto consiste en, para cada neurona de cada capa, multiplicar la entrada de las mismas por los coeficientes o pesos almacenados en la neurona y evaluar la *función de activación*†, a modo de obtener la salida de la capa que será enviada a la siguiente. Debido a que en general, las multiplicaciones dentro de una capa son independientes al resto de neuronas, es posible definir la operación como una multiplicación tensorial (seguida de la aplicación vectorial de la *función de activación*†).

4.4.1.3. Función de activación

Las neuronas de una red aplican la transformación lineal entre su entrada y los pesos de la misma, y al resultado le aplican una *función de activación*†, lo que agranda el espacio de soluciones a funciones no lineales (*LeCun et al.*, 2012).

La *función de activación*† no lineal más básica y usada en los primeros estudios es la **función umbral**, que dado un umbral z fijo, y un valor x a evaluar, retorna 1 si $x \geq z$ y 0 en el caso contrario, lo cual convierte la salida de las neuronas en un resultado binario. El problema de esta función es que posee gradiente igual a 0 para $x \neq z$, y no es diferenciable en $x = z$ (el umbral); esto causa que no sea optimizable con métodos de *descenso por gradiente*† en el algoritmo de *propagación hacia atrás*† que es de gran importancia como se describe más adelante en la sección 4.4.1.5.

Una *función de activación*† con mayor éxito es la **función Sigmoide** o *Logística* (*LeCun et al.*, 2012), la cual es monótona creciente (por lo que tiene un gradiente positivo), y presenta sus asíntotas en ambos extremos de un valor finito. Dos funciones con estas propiedades son la **función logística estándar** y la **tangente hiperbólica**:

$$logistica(x) = \frac{1}{1 + e^{-x}}; \tag{4.15}$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{4.16}$$

Se destaca que al ser asintóticas, sus salidas siempre se encuentran dentro de un rango acotado, lo que normaliza su salida y puede facilitar el *entrenamiento*†, evitando tanto el crecimiento de sus pesos hasta el infinito, como el fenómeno de explosión del gradiente.

Sin embargo, estas funciones también pueden llegar causar el *problema del desvanecimiento del gradiente*†, que ocurre debido a que los gradientes de las sigmoides tienden a 0 en sus extremos, por lo que la capacidad de mejora durante el *entrenamiento*† decrece si las neuronas devuelven valores cercanos a los extremos (lo que se podría combatir con una asignación inteligente de pesos iniciales) (*Balduvino and Lorenzo*, 2019). Las funciones sigmoides además son costosas de calcular.

Para aliviar los problemas anteriores, pero mantener gran parte del poder de expresividad que las funciones sigmoides aportan, se emplean funciones **rectificadoras**, las cuales son casi lineales y rápidas de computar.

La función **Unidad Lineal Rectificada (ReLU)** (*Nair and Hinton*, 2010) es una de las más eficientes *funciones de activación*† empleadas hoy en día. Dado un valor de umbral z (usualmente $z = 0$):

$$ReLU(x) = \begin{cases} 0, & \text{si } x \leq z \\ x, & \text{en otro caso.} \end{cases} \tag{4.17}$$

4.4.1.4. Aprendizaje supervisado

La tarea de determinar los pesos para que una *red neuronal*† aproxime correctamente la función deseada es extremadamente compleja de modelar con exactitud. Para ello se recurre al *aprendizaje automático*†; en particular, para la mayoría de modelos se utiliza el *aprendizaje supervisado*†, es decir, entrenar un

modelo en base a un *conjunto de datos*† que para cada instancia del conjunto, cuente con anotaciones etiquetadas (las posición de cada persona en la imagen, para este caso).

Si bien existen otros tipos de estrategia de aprendizaje, como el *aprendizaje no supervisado*†, o el *aprendizaje débilmente supervisado*†, y han sido explorados en el campo del conteo de multitudes, los mismos no entran en el foco de este estudio.

4.4.1.5. Propagación hacia atrás

Las *redes neuronales*† fueron propuestas hace varias décadas, y conceptualmente eran muy poderosas, pero varios factores limitaron su aplicabilidad. Uno de estos era contar con un algoritmo de *entrenamiento*† efectivo y eficiente.

El algoritmo de *propagación hacia atrás*† se basó en el método de diferenciación automática (*Linnainmaa*, 1976), que fue refinado mediante estudios para su aplicación en funciones no lineales (*Werbos*, 1974, 1982), y por último llegó a ser conocido como el método de *propagación hacia atrás*† (*backpropagation* en inglés).

Este algoritmo es vital para el *entrenamiento*† de *redes neuronales*† profundas (con varias capas ocultas), y consiste en propagar el gradiente de la *función de pérdida*† hacia atrás, actualizando los pesos de cada neurona en dirección opuesta al gradiente calculado, minimizando localmente el valor de la *función de pérdida*†.

Esto durante la etapa de *entrenamiento*† permite corregir el valor de los pesos (los cuales se inicializan con distintos valores según la estrategia de inicialización aplicada, como por ejemplo valores aleatorios), en una dirección pequeña hasta que converjan a valores en los que se minimice el error obtenido (medido acorde a la *función de pérdida*† aplicada) para las instancias del *conjunto de entrenamiento*†.

Para lograr esto, sin embargo, es necesario conocer el gradiente de la *función de activación*† de cada neurona en cada capa, los cuales son calculados iterativamente desde la capa de salida hasta la capa de entrada. Es por esto que antes de ajustar los pesos es necesario realizar una *inferencia*† con una instancia, para luego evaluar el desempeño de los pesos actuales, es decir un *forward pass* de la entrada, y ajustar las capas de la red hacia atrás en base a los mismos (*Balduvino and Lorenzo*, 2019).

La propagación del gradiente funciona gracias a la regla de la cadena, y este proceso de cálculo y ajustes de gradientes a lo largo de todas las capas de la *red neuronal*† es lo que compone el *entrenamiento*† de las mismas.

Nótese que para su aplicación en la práctica, los gradientes son calculados empleando métodos numéricos de de *descenso por gradiente*† (como $lbfgs$, sgd, o $adam$), y el método empleado afectará los resultados finales del modelo (debido a estar directamente relacionado a los ajustes de pesos y su convergencia), dado que usualmente las funciones de costo son altamente no convexas.

Por último, las *redes neuronales*† tuvieron otro problema en su época, y es que no se contaba con el poder computacional necesario para entrenarlas y llegar a una convergencia del valor de cada peso en la red. En las últimas décadas hubo un resurgimiento lento y consistente, que explotó en esta última década gracias al incremento de poder de computo y el surgimiento de grandes *conjuntos de datos*†.

4.4.2. Funciones de pérdida

Una *función de pérdida*† es una función empleada para cuantificar la diferencia entre el *valor anotado*† y el valor predicho por el modelo. En el contexto del conteo de multitudes con *mapas de densidad*†, estas diferencias se miden en base a las matrices de densidad generadas en base a las anotaciones de los datos y las generadas por ejecutar el modelo.

El objetivo de la misma es poder cuantificar la distancia entre la predicción y el valor deseado, con el fin de poder ajustar y direccionar los pesos del modelo a lo largo del *entrenamiento*†.

A continuación se presentan las *funciones de pérdida*† más relevantes que han sido empleadas en el campo del conteo de multitudes.

Se destaca que algunas *funciones de pérdida*† más particulares se definen en la sección 4.9, mientras que otras compiladas por *Gao et al.* (2020) no se consideraron de suficiente relevancia para ser incluidas esta sección, y se definen en el apéndice B.1.

4.4.2.1. Función de pérdida euclidiana (ℓ_2)

La *función de pérdida euclidiana* $(\ell_2)^\dagger$ es la *función de pérdida*† más comúnmente usada en la investigación del conteo de multitudes y en los problemas de regresión en general. Durante este trabajo, en caso de no aclararse explícitamente, los modelos presentados emplean esta *función de pérdida*†:

$$\ell_D(\Theta) = \frac{1}{N}\sum_{i=1}^{N} ||F(X_i;\Theta) - D_i||_2^2, \tag{4.18}$$

en donde N es la cantidad total de imágenes, X_i es una imagen, Θ representa los parámetros de la *red neuronal*† (como sus pesos), N es el número total de muestras, $F(X_i;\Theta)$ es el mapa que el modelo predijo, y D_i es el *mapa de densidad*† generado en base a los datos anotados.

Si bien la *función de pérdida euclidiana* $(\ell_2)^\dagger$ es la más usada, posee varias desventajas, tales como su sensibilidad a valores atípicos o a mapas difuminados. Tampoco prioriza la coherencia local en el *mapa de densidad*†, así como no refleja la correlación espacial en el mismo (*Gao et al.*, 2020); en particular, trabaja asumiendo que los valores en un píxel son estadísticamente independientes al resto de sus píxeles vecinos (*Khan et al.*, 2022).

Aunque no es común, otro enfoque para la pérdida euclidiana consiste en basarlo en la cantidad total de elementos detectados en la instancia (personas), sumando el valor del *mapa de densidad*† predicho y generado en base a las anotaciones para obtener la cantidad total de personas C_i y en base a ello calcular la *función de pérdida*†:

$$\ell_C(\Theta) = \frac{1}{N}\sum_{i=1}^{N} (\hat{C}_i - C_i)^2. \tag{4.19}$$

4.4.2.2. Función de pérdida (ℓ_1)

La *función de pérdida* $(\ell_1)^\dagger$ es calculada como la distancia de Manhattan (norma l_1) entre los píxeles del *mapa de densidad*† predicho y su valor esperado generado en base a los datos anotados:

$$\ell_D(\Theta) = \frac{1}{N}\sum_{i=1}^{N} ||F(X_i;\Theta) - D_i||_1. \tag{4.20}$$

De forma similar a la *función de pérdida euclidiana* $(\ell_2)^\dagger$, también se podría computar usando C_i, la cantidad total de individuos contados.

$$\ell_C(\Theta) = \frac{1}{N}\sum_{i=1}^{N} |\hat{C}_i - C_i|. \tag{4.21}$$

4.4.2.3. Función de pérdida compuesta

La *función de pérdida de composición*†, o *función de perdida de combinación* es la idea de combinar múltiples *funciones de pérdida*† para mejorar el aprendizaje durante el *entrenamiento*†. Por ejemplo en el trabajo CrowdCNN (*Zhang et al.*, 2015) y en (*Idrees et al.*, 2018b), ambos proponen combinar la *función de pérdida euclidiana* $(\ell_2)^\dagger$ en base a la matriz de densidad y en base a la cantidad total de individuos en la imagen:

$$\ell(\Theta) = \ell_D(\Theta) + \lambda\ell_C(\Theta). \tag{4.22}$$

Cabe destacar que esta idea es aplicable para cualquier grupo de *funciones de pérdida*†. En particular se destaca su utilidad en el contexto de los métodos de *aprendizaje multitarea*†, en los que durante el *entrenamiento*† es deseable ajustar el desempeño de varias tareas para generar el resultado final.

4.4.2.4. Función de pérdida atenta a la región piramidal (PRA)

Iterando sobre la *función de pérdida AP*† (definida en el apéndice B.1.2), que apunta a reducir los errores de estimación en las regiones más densas de la imagen, el trabajo **SASNet** (*Song et al.*, 2021) introduce la esta *función de pérdida*† como una extensión natural de la *función de pérdida AP*†.

La intuición detrás de la *función de pérdida atenta a la región piramidal (PRA Loss)*† es que en cada imagen de una multitud, algunas regiones (y por ende algunos píxeles) son más difíciles de aprender para el modelo. Estas regiones y píxeles son típicamente sobre-estimados, lo que contribuye a más errores en el conteo. Se puede lograr solventar el *entrenamiento*† para que aprenda a resolver dichas regiones a través de incrementar el peso que tienen las mismas en la *función de pérdida*†.

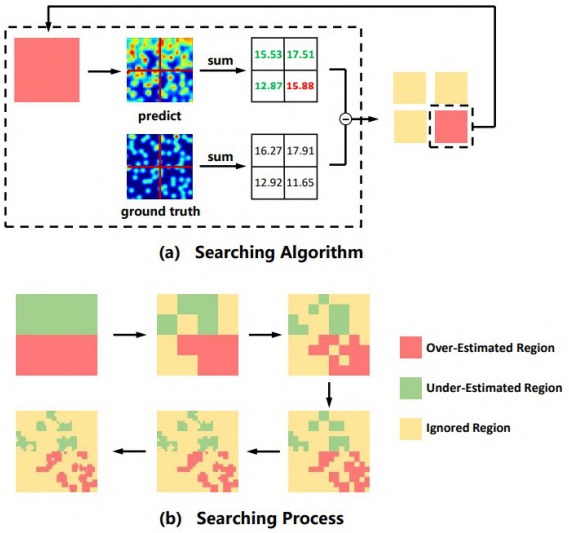


Figura 4.7: Búsqueda recursiva para la función de perdida PRA (*Song et al.*, 2021)

La *función de pérdida PRA*† divide la imagen en cuatro parches, y el parche con mayor error de conteo (generalmente sobre-estimado) es seleccionado. Este proceso se repite recursivamente hasta que los píxeles con el error más alto sean determinados y se puede apreciar en la figura 4.7. La *función de pérdida*† se calcula como:

$$\ell_{PRA} = ||D_{est}^{p\in G} - D_{GT}^{p\in G}||_2^2 + \gamma||D_{est}^{p\in H} - D_{GT}^{p\in G}||_2^2, \tag{4.23}$$

en donde p es un píxel en el *mapa de densidad*† D^{est}, G representa todos los conjuntos de píxeles en D^{est}, H es el conjunto de todos los píxeles difíciles en D^{est} y por último γ es un *hiperparámetro*† que determina el peso que aumentará el valor de la *función de pérdida*† para este conjunto de píxeles más difíciles.

4.4.3. Conceptos destacados de las redes neuronales convolucionales

En un modelo neuronal que emplee por ejemplo *capas totalmente conectadas*†, sería necesario conectar cada píxel de la imagen a cada una de las neuronas de la capa de entrada, por lo que cuanto mayor sea el tamaño de la imagen, más pesos será necesario almacenar en cada neurona y más compleja será la suma ponderada para el proceso de *inferencia*†.

Una particularidad de las imágenes y el espacio dimensional definido por las mismas es que su información y nuestra percepción sobre las mismas va desde elementos locales (píxeles cercanos entre sí) hacia elementos globales (la imagen en su totalidad). Esto es coherente con la relación espacial de la imagen, dado que en general un píxel suele estar más relacionado o próximo a sus píxeles cercanos que al resto (*Gong*, 2020).

Las *CNNs*† (*Fukushima and Miyake*, 1982; *LeCun et al.*, 2010) proponen que en espacios dimensionales como el anteriormente descripto, no es necesario que cada neurona perciba la imagen en su totalidad, sino que para cada píxel puede percibir sólo los píxeles adyacentes, y a lo largo de la red el proceso de *inferencia*† puede ir combinando información local de distintos píxeles para obtener información global de la imagen.

Esta idea también tiene una inspiración en la estructura del sistema visual biológico, en donde las neuronas en la corteza visual reciben información local (a base de estímulos) y el resto de su *red neuronal*† en el cerebro combina esa información para darnos la información global de lo que estamos viendo (*Hubel and Wiesel*, 1998).

4.4.3.1. Convolución

La *convolución*† es la operación principal que define a las *capas convolucionales*†, y por ende a las *CNNs*†. Consiste de una operación matemática que provee una forma de multiplicar en conjunto dos arreglos de números no necesariamente del mismo tamaño, pero sí de igual dimensión vectorial (por ejemplo entre vectores, matrices o tensores)

Sea $F : \mathbb{Z}^2 \to \mathbb{R}$ una función discreta. Sea $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$ y $k : \Omega_r \to \mathbb{R}$ un filtro discreto de soporte $(2r+1) \times (2r+1)$. Se define el operador de *convolución*† discreto $*$ como: $(F * k)(p) = \sum_{s+t=p} F(s)k(t)$.

Al aplicarlas a matrices (como en el caso del conteo de multitudes), se emplean *convoluciones*† tridimensionales (ancho, largo y canal de color), convolucionando a lo largo de la matriz por cada filtro, o núcleo, cuyo **tamaño** varía según la arquitectura de la red (entre por ejemplo 3×3, 5×5, 7×7 y 9×9). Cuanto más pequeño el filtro se enfocará en detalles más granulares, mientras que un filtro grande considerará detalles de menor localidad (*Sassisegarane*, 2021).

Acorde a los coeficientes almacenados en el filtro, es que cada neurona puede alcanzar un valor alto y activarse al detectar cierto patrón en la distribución de su entrada. En una *capa convolucional*† habrán múltiples filtros con distintos coeficientes, cada uno produciendo una salida a un canal distinto. Se observa que las neuronas aplican los mismos filtros a lo largo de la imagen, siendo los pesos de un filtro compartidos, a esto se le llama *weight sharing*.

Además del **tamaño de filtro**, también son necesarias dentro de una *capa convolucional*† otras características, como por ejemplo: (*Dertat*, 2017)

- El **paso** (*stride*): La distancia entre píxeles de la imagen para los que se aplica el núcleo. Un paso de 1 ejecutará la *convolución*† para cada píxel, uno de 2 dejará un píxel entre medio, etc.
- El **margen** (*padding*): Determina cuánto del borde de la imagen queda fuera de la operación. Con un espaciado de 0, una *convolución*† de 3×3 en una esquina asume que los píxeles afuera de la imagen valen 0, mientras que un espaciado del tamaño del filtro asegura que el filtro no considere píxeles fuera de la imagen. El efecto que tiene el margen puede apreciarse en las figuras 4.8 y 4.9.
- La **dilatación**: Los filtros además pueden estar dilatados, en cuyo caso se les llama *convolución dilatada*†. Esto propone que un filtro 3×3 por ejemplo, capture 9 puntos no necesariamente en un cuadrado de 3×3, sino en algún patrón que considere píxeles más lejanos.

Con esta información se puede apreciar que la aplicación de un filtro de *convolución*† no necesariamente preserva las dimensiones de la entrada (a menos que su **paso** y **espaciado** sean de 1 y 0 respectivamente).

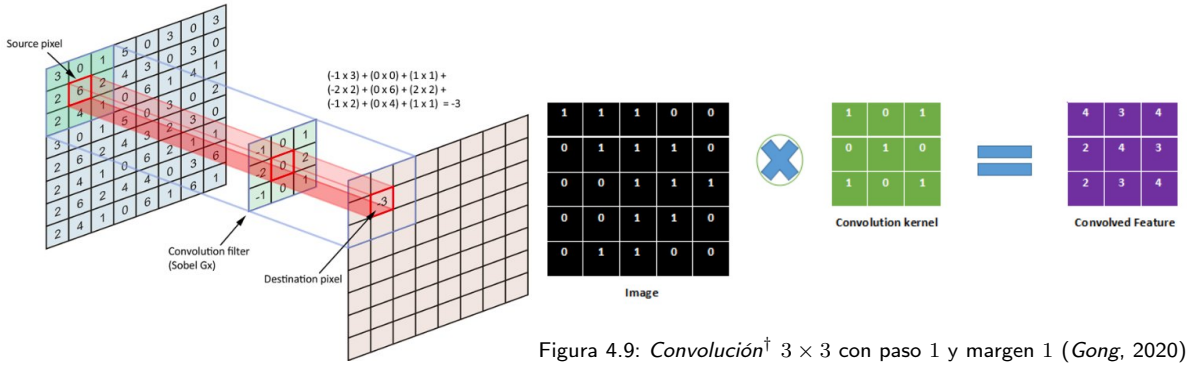


Figura 4.9: *Convolución*† 3×3 con paso 1 y margen 1 (*Gong*, 2020)

Figura 4.8: *Convolución*† 3×3 con paso 1 y margen 0 (*Dertat*, 2017)

Es importante destacar que, para cada atributo generado por la *capa convolucional*†, al valor resultante de la *convolución*† es necesario aplicarle una *función de activación*† (como ReLU), para que la operación ofrezca no linealidad (*Dertat*, 2017).

Cabe destacar cómo, si bien la operación de *convolución*† aplicada es bidimensional, los filtros convolucionales aplicados son tridimensionales, dado que la entrada contiene 3 canales de color y luego a lo largo de la red la cantidad de filtros convolucionales en una *capa convolucional*† genera N canales de salida.
A los efectos de notación se omite la dimensión de canal del filtro (cubriendo siempre todos los canales), por ende cuando se hable de un filtro 3×3, en realidad es un filtro $3 \times 3 \times N$ en donde N es la cantidad de canales de salida de la capa anterior. Esto se puede apreciar en la figura 4.10.

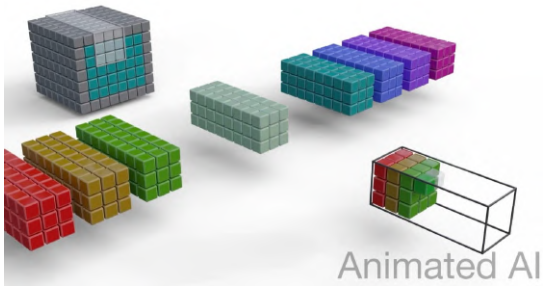


Figura 4.10: *Convolución*† 3×3 2d multicanal (*AnimatedAI*, 2022)

Otro beneficio de las *capas convolucionales*† es que requieren de una cantidad menor de pesos a entrenar. Dado que cada uno de los filtros es usado como ventana deslizante a lo largo de las dimensiones de la imagen, pueden funcionar N filtros sin importar el tamaño de la imagen de entrada, lo cual facilita el *entrenamiento*† de las *CNNs*†. Se destaca cómo cada filtro generará un canal de salida en la *capa convolucional*† (*Balduvino and Lorenzo*, 2019; *Gong*, 2020). La figura 4.11 ilustra una *capa convolucional*†, mientras que la figura 4.12 muestra la combinación de las mismas para formar una *CNN*†.

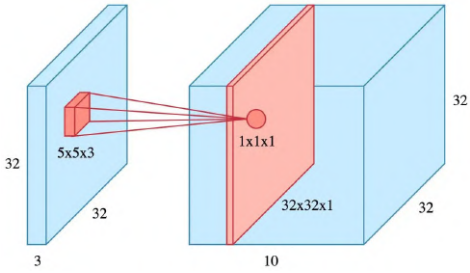


Figura 4.11: *Capa convolucional*† 5×5 con paso 1, margen 0 y 10 canales (*Dertat*, 2017)

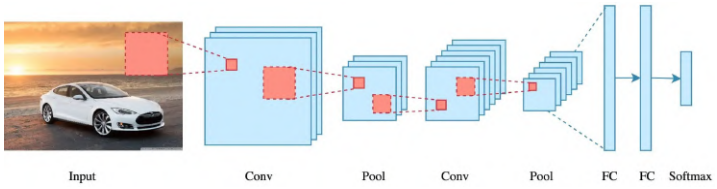


Figura 4.12: Combinación de *capas convolucionales*† y *capas totalmente conectadas*† (*Dertat*, 2017)

4.4.3.2. Convolución 1×1

Las *convoluciones* 1×1† tienen un rol muy interesante en el mundo de la visión artificial, y son universalmente usadas en los modelos contemporáneos de conteo de multitudes.
Una *convolución* 1×1† de paso 1 y margen 0, parece no ser útil a primera vista a los efectos del objetivo detrás de una *convolución*†; sin embargo, al ser sus verdaderas dimensiones $1 \times 1 \times D$, esta *convolución*† permite condensar la información de cada coordenada bidimensional a lo largo de sus D canales. Esto se ilustra en la figura 4.13.
En el contexto del conteo de multitudes, son empleadas como capa final en la *red neuronal*†, por lo que para generar un *mapa de densidad*† se mantiene la resolución de largo y ancho de la penúltima capa (resolución del *mapa de densidad*† generado), y se condensan todos los canales a uno solo.

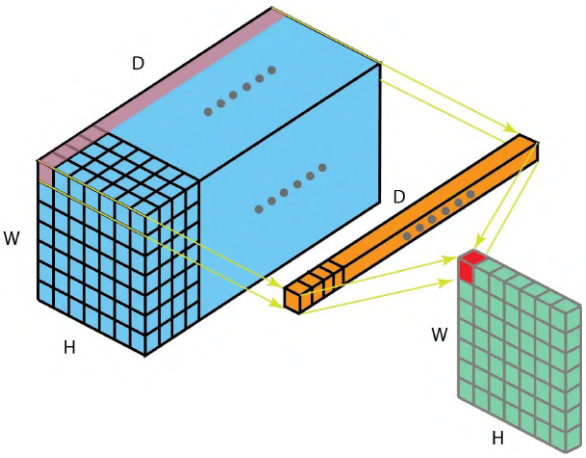


Figura 4.13: *Convolución*† 1×1 (*Bai*, 2019)

4.4.3.3. Pooling

El *pooling*† es aplicado para disminuir la dimensionalidad de un tensor, lo cual permite reducir el número de parámetros, lo que acorta el tiempo de *entrenamiento*† y combate el *sobreajuste*†. Las *capas de pooling*† achican cada canal de los mapas de atributos independientemente, reduciendo su altura y ancho, pero manteniendo su profundidad; esto se puede apreciar en la figura 4.15.

Similar a las *convoluciones*†, el *pooling*† también se aplica a lo largo de una ventana por la imagen, y por ende también tiene definido un **paso** (*stride*) y un **margen** (*padding*) (*Dertat*, 2017).

El tipo más común de *pooling*† es *max pooling* (ejemplo en figura 4.14), en el que dados los valores de la ventana de *pooling*†, el valor de salida es el valor de mayor magnitud. Pero también existe el *average pooling*, que promedia los atributos para determinar el valor final.

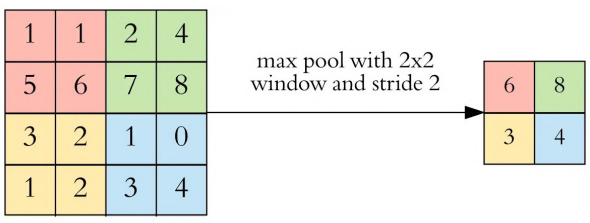


Figura 4.14: Max Pooling 2×2 con paso de 2 (*Dertat*, 2017)

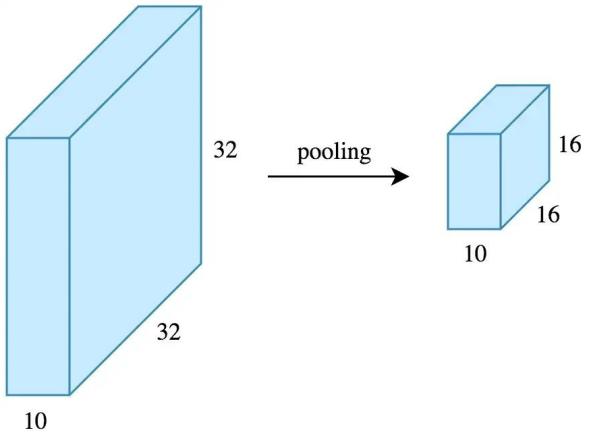


Figura 4.15: Efecto de max pooling 2×2 con paso de 2 (*Dertat*, 2017)

En general para las *CNN*† se emplean *poolings*† con tamaño de ventana 2×2, paso de 2 y margen de 0, lo que reduce las dimensiones del mapa de atributos a la mitad, y por ende reduce la cantidad de atributos en $\frac{1}{4}$.

Las *CNN*† son usadas para identificar patrones bidimensionales con la invarianza del desplazamiento y escalado. Las *capas de pooling*† tienen un rol fundamental en esto, dado que concentran la información en un espacio de menor dimensionalidad pero que representa la información de forma conceptualmente jerárquica, las primeras capas son muy locales y codifican cosas como bordes en la imagen, mientras que al ser refinadas a lo largo de la red las matrices pasan a contener información más semántica (*Gong*, 2020).

4.4.3.4. Redes Convolucionales

En general las *CNN*† incluyen dos grupos de capas. Las del primer grupo se agrupan al principio en la **capa de extracción de atributos**, que tiene el rol de refinar la información recibida por la entrada a atributos de mayor valor semántico para la *red neuronal*†. El segundo grupo es el de capas de **mapeo de atributos**, que dados atributos refinados, infieren la salida de la red (esta última puede ser compuesta por *capas totalmente conectadas*† por ejemplo). Un ejemplo de red *CNN*† es la arquitectura de la VGG-16 (*Simonyan and Zisserman*, 2014) que se puede apreciar en la figura 4.16.

Las *CNNs*† son extremadamente potentes a la hora de procesar imágenes. Las *capas convolucionales*† clásicas aceptan entradas de tamaño arbitrario, son mucho más eficientes al procesar entradas de gran tamaño en comparación a las *capas totalmente conectadas*†, priorizan refinar en detalle las relaciones locales entre píxeles (lo que permite reconocer patrones), y la cantidad de pesos que debe aprender es relativamente baja, dado que para *capas convolucionales*† tradicionales, no dependen del tamaño de entrada de la red (*Balduvino and Lorenzo*, 2019).

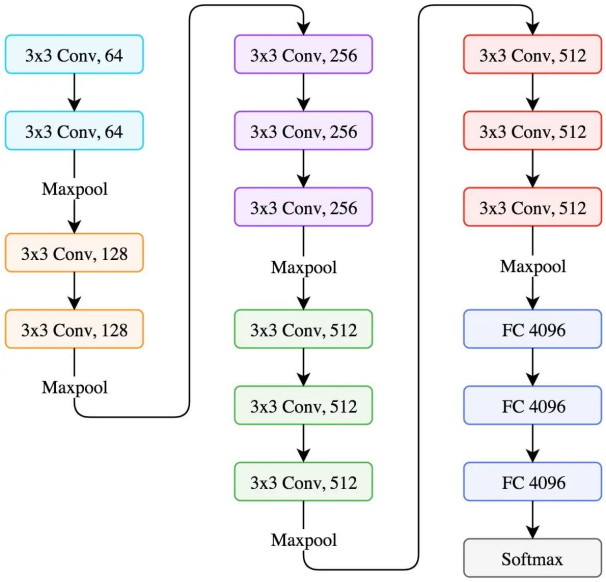


Figura 4.16: Arquitectura de la VGG-16 (*Simonyan and Zisserman*, 2014) imagen obtenida de (*Dertat*, 2017)

4.4.4. Conceptos destacados del aprendizaje profundo

El *aprendizaje profundo*† fue propuesto por la **Deep Belief Network** (*Hinton*, 2009), y ha hecho gran progreso en el campo del *aprendizaje automático*†; en particular en las áreas que trabajan con datos de una alta dimensionalidad, como es el caso de la visión artificial cuyos datos de entrada suelen ser imágenes o videos (*Gong*, 2020).

La técnica de incrementar el número de capas ocultas fue demostrado en la práctica y en la teoría por *Montufar et al.* (2014), que evalúa redes con la misma cantidad de parámetros (sea con capas ocultas más grandes, o mayor cantidad de capas ocultas) y muestra como a mayor cantidad de capas ocultas la red adquiere mayor poder de expresividad.

4.4.4.1. Aprendizaje por transferencia

En el contexto del *aprendizaje automático*†, el *aprendizaje por transferencia*† consiste en integrar un modelo previamente entrenado a la arquitectura de otro modelo (que busca resolver una tarea relacionada).

Esta técnica puede verse de forma más explícita integrando modelos ya entrenados como módulos en un algoritmo, con el fin de afectar la manipulación y transformación de datos.

Por otro lado, en el contexto de las *redes neuronales*† se puede ver de una forma más implícita, agregando capas pre-enetrenadas en una tarea (en general convolucionales) al principio de la arquitectura de la *red neuronal*†; en general a la red reutilizada se le denomina *backbone*†.

En general, para la visión artificial, un *backbone*† es conformado por las primeras *capas convolucionales*† de un modelo de visión artificial (como puede ser uno de clasificación de imágenes), que ha sido entrenado con un gran *conjunto de datos*†. Si el *entrenamiento*† de una *CNN*† funciona como se espera, las **capas de extracción de atributos** (o *Feature Map Extractor (FME)*) aprenden patrones y forman atributos semánticos, mientras que las **capas de mapeo de atributos** aprenden a inferir la solución al problema a partir de dichos atributos.

Por ende, contar con un *backbone*† entrenado con un gran *conjunto de datos*† (generalmente **ImageNet** (*Deng et al.*, 2009)) permite aprovechar la capacidad de extraer patrones en las primeras capas de la red (y opcionalmente alterando levemente sus pesos durante el *entrenamiento*†), lo cual refina los atributos generados durante el *entrenamiento*† y acelera el entrenamiento del modelo.

En el campo del conteo de multitudes se han empleado los siguientes *backbones*†:

4.4.4.2. VGG-16

La VGG-16 o ConvNet (*Simonyan and Zisserman*, 2014) es una *CNN*† entrenada para la detección y clasificación de imágenes. Entrenada y evaluada con ImageNet (*Deng et al.*, 2009), es capaz de clasificar imágenes de 1000 categorías distintas con 92,7 % de precisión. Si bien existen modelos de mejor desempeño en estas tareas (como Inception (*Szegedy et al.*, 2015) y ResNet (*He et al.*, 2016)), también son modelos de mayor complejidad, mientras que la VGG-16 tiene una arquitectura bastante sencilla, lo que la ha posicionado como el *backbone*† de mejor desempeño en el conteo de multitudes. Se destaca su arquitectura en la figura 4.16

La red utiliza *convoluciones*† de filtro 3×3, con margen y paso de 1, y va aumentando la cantidad de canales (y por ende filtros) al incrementar su profundidad.

En el contexto del conteo de multitudes, en general se emplean las primeras 13 capas de la misma (incluyendo las capas de max *pooling*†). La idea es que dichas capas permiten detectar detalles entre bordes y patrones (esenciales para diferenciar regiones congestionadas), pero puede que no tengan suficiente información para distinguir regiones de cabezas entre las hojas de un árbol, edificios o fondos desordenados (*Zhu et al.*, 2019).

Este es uno de los *backbones*† más empleados, y es usado por ejemplo en los siguientes modelos:

CrowdNet (*Boominathan et al.*, 2016), **Switching-CNN** (*Babu Sam et al.*, 2017),
CSRNet (*Li et al.*, 2018), **GSP** (*Aich and Stavness*, 2018), **CANNet** (*Liu et al.*, 2019b)
PACNN (*Shi et al.*, 2019), **C-CNN** (*Shi et al.*, 2020), **SFANet** (*Zhu et al.*, 2019),
MBTTBF-SCFB (*Sindagi and Patel*, 2019), **LibraNet** (*Liu et al.*, 2020), **ADSCNet** (*Bai et al.*, 2020)
SASNet (*Song et al.*, 2021), **M-SFANet** (*Thanasutives et al.*, 2021),
M-SegNet (*Thanasutives et al.*, 2021), **FusionCount** (*Ma et al.*, 2022), **TAFNet** (*Tang et al.*, 2022).

Otros modelos emplean la **VGG-19**, es decir la VGG entrenada con 3 *capas convolucionales*† adicionales, algunos de estos son: **BayL** (*Ma et al.*, 2019), **MAN** (*Lin et al.*, 2022).

4.4.4.3. CSRNet

CSRNet (*Li et al.*, 2018) es un modelo de conteo de multitudes presentado en 2018, y el cual propuso la arquitectura de una sola columna detallada más adelante en la sección 4.7 (que a su vez incluye una sección sobre este modelo 4.7.1).

Si bien algunos trabajos han empleado la **CSRNet** como *backbone*†, la **CSRNet** a su vez emplea la **VGG-16** (*Simonyan and Zisserman*, 2014) de *backbone*†. Por lo tanto, los modelos que integran CSRNet como *backbone*† también están basándose en la **VGG-16**.

Cuando se emplea la **CSRNet** como *backbone*†, se emplean todas sus capas salvo la última *convolución* 1×1†

Estos son algunos de los modelos que emplean la **CSRNet** como *backbone*†: **PGCNet** (*Yan et al.*, 2019a), **DeepCount** (*Chen et al.*, 2019), **MATT** (*Lei et al.*, 2021), **DMPNet** (*Li et al.*, 2022b).

Algunos de estos modelos se centran en agregar una técnica modular para mejorar el problema de conteo, y por ende son aplicables a cualquier modelo, pero emplean **CSRNet** por su simpleza y efectividad como *backbone*†.

En general siguen surgiendo modelos que no usan la **CSRNet**, usando solo la **VGG-16** como *backbone*†. Esto no significa que su arquitectura no presente *convoluciones dilatadas*† (empleadas por **CSRNet** y presentadas en más detalle en la sección 4.7.1), como es el caso con **ADSCNet** (*Wu et al.*, 2019) y **LA-Batch** (*Zhou et al.*, 2021).

Otros modelos contemporáneos que no usan la **CSRNet** emplean *conexiones residuales*† con las primeras capas de la red (**MBTTBF-SCFB** (*Sindagi and Patel*, 2019), **SFANet** (*Zhu et al.*, 2019) y **SASNet** (*Song et al.*, 2021)), propiedad que **CSRNet** carece, el beneficio de estas conexiones se presenta en la sección 4.8.2.

4.4.4.4. Inception

La red y arquitectura Inception (*Szegedy et al.*, 2015, 2016), ha sido de bastante importancia en el mundo de la visión artificial, y plantea en lugar de sólo aumentar la profundidad de la *CNN*†, aumentar la complejidad de la misma, agregando por ejemplo canales con distinto tamaño en los filtros convolucionales. Este módulo se puede apreciar en la figura 4.17, y la arquitectura completa de la red en la figura 4.18.

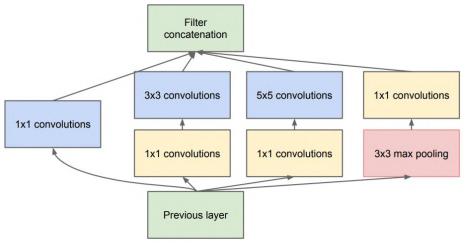


Figura 4.17: Modulo convolucional de filtros de Inception-V1 (*Szegedy et al.*, 2015)

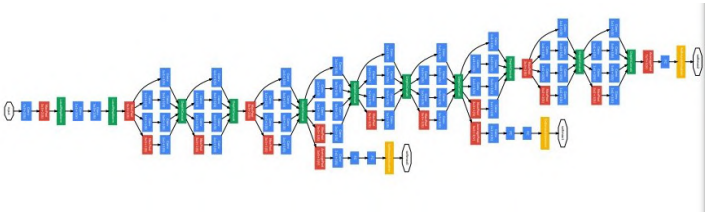


Figura 4.18: Arquitectura de Inception-V1 (*Szegedy et al.*, 2015)

Ha habido varias versiones de esta arquitectura a lo largo de los años. Un trabajo reciente en particular (**SGANet** (*Wang and Breckon*, 2022)) empleó como *backbone*† Inception-V3, no llegando sin embargo a superar el estado del arte.

MSCNN (*Zeng et al.*, 2017) y **SANet** (*Cao et al.*, 2018) no emplean ningún *backbone*† en particular, pero inspiraron sus módulos convolucionales en la arquitectura Inception (*Szegedy et al.*, 2015) (*Gong*, 2020).

4.4.4.5. ResNet

ResNet, o Residual Network (*He et al.*, 2016), es un tipo de *CNN*† que plantea incluir *conexiones residuales*† a lo largo de las *capas convolucionales*†. Una *conexión residual*†, o *skip connection*, consta de alimentar la entrada de una red con la salida de capas no adyacentes a la misma. La arquitectura de esta red comparada con la VGG-19 se puede apreciar en la figura 4.19.

Esta red es de mayor profundidad que la **VGG-16** (*Simonyan and Zisserman*, 2014), lo que aumenta su complejidad y tiempo de *inferencia*† y *entrenamiento*†, pero a su vez las *conexiones residuales*† le permiten aliviar el *problema del desvanecimiento del gradiente*†, aprendiendo con mayor facilidad funciones de mayor complejidad.

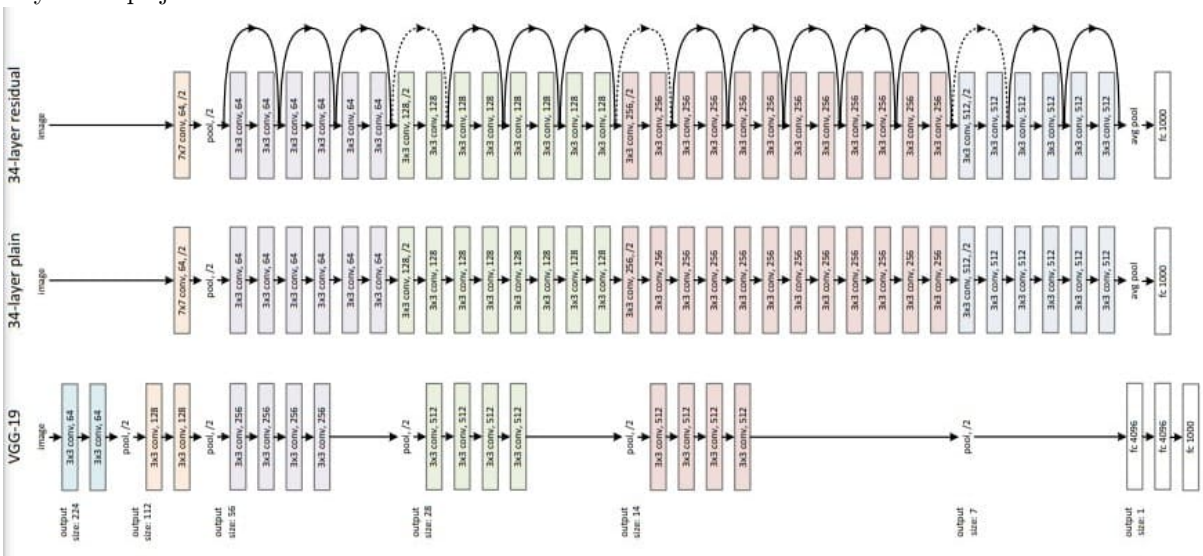


Figura 4.19: Arquitectura de ResNet-34 (*He et al.*, 2016)

ResNet-50 (es decir, una variante de ResNet con 50 capas de profundidad) ha sido solo empleada como *backbone*† en los modelos:

- **MMCNN** (*Peng et al.*, 2020a) y **MFCC** (*Gu and Lian*, 2022), modelos enfocados al conteo de multitudes con imágenes de drones, en particular empleando el *conjunto de datos*† **DroneRGBT** (*Peng et al.*, 2020b), que tiene anotaciones térmicas.
- **DUBNet** (*Oh et al.*, 2019), que emplea una similar a **CSRNet** (*Li et al.*, 2018), sólo que reemplazando su *backbone*† por **ResNet-50**.

4.4.5. Mecanismos de atención

Inspirado en el éxito del articulo *Attention is all you need* (*Vaswani et al.*, 2017), varios trabajos incorporan esta técnica a la hora de refinar o combinar la información de distintas salidas.

Se entrena con una *función de pérdida*†, usualmente compuesta, que además de ajustar los pesos para mejorar la resolución del problema, también intenta aprender a inferir qué atributos a lo largo de la red son más importantes para la tarea a resolver (y de esa forma dar menos valor al resto).

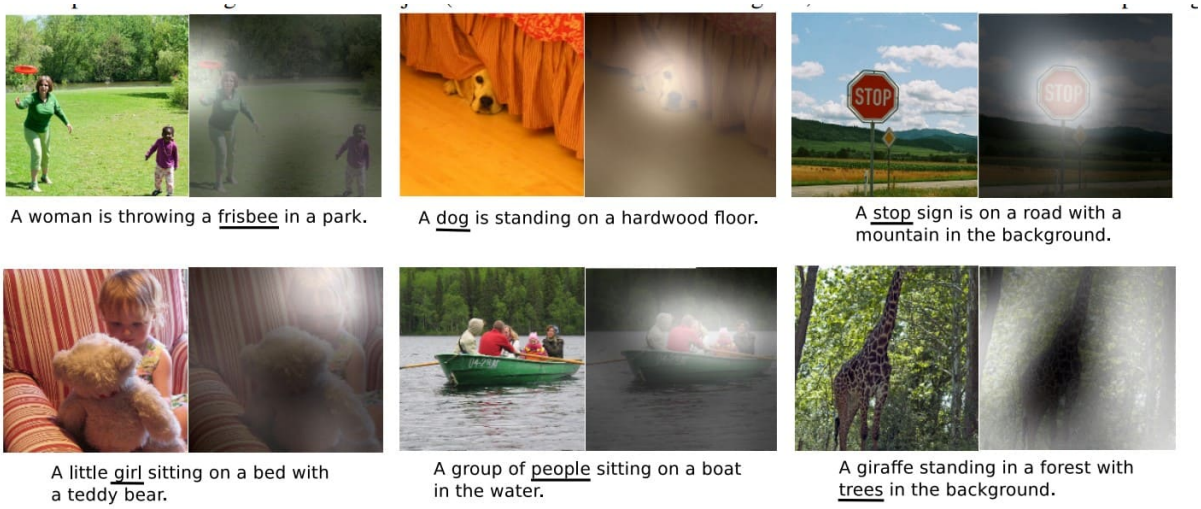


Figura 4.20: Ejemplo de un mecanismo de *atención*† entrenado a variar según palabras (*Xu et al.*, 2015)

En el contexto del *aprendizaje automático*† aplicado al conteo de multitudes, un mecanismo de *atención*† implica que el modelo genere como subproducto un **mapa de atención**. Siendo similar al *mapa de densidad*† generado, sólo que orientado a marcar con valores altos áreas de la entrada de mayor interés para el modelo (como puede ser segmentar una multitud del fondo, o dar más peso al fondo por la densidad de la misma). Las imágenes en la figura 4.20 ilustran los mapas de *atención*†.

Esta información es combinada con el *mapa de densidad*† generado por el modelo, usualmente empleando el operador de Hadamard (multiplicación elemento-elemento) entre ambos mapas. Se puede ver un ejemplo de arquitectura con *atención*† en la figura 4.35.

4.5. Trabajos pioneros con redes convolucionales

Los primeros trabajos que empleaban *CNNs*† en la tarea del conteo de multitudes, utilizaba una arquitectura compuesta por *capas convolucionales*†, *capas de pooling*†, y *capas totalmente conectadas*† (*Fu et al.*, 2015; *Wang et al.*, 2015; *Zhang et al.*, 2015; *Walach and Wolf*, 2016). Se destaca como si bien estos trabajos emplean *CNNs*†, carecen de un *backbone*† (lo cual les diferencia del resto de trabajos posteriores.

Otra particularidad de estos modelos, era que su tipo de entrada eran parches con dimensión fija recortados de las imágenes del *conjunto de datos*†. Para el *entrenamiento*† se fabricaban instancias en base a recortes aleatorios en las imágenes, mientras que para la *evaluación*† se dividía una imagen en parches no superpuestos hasta cubrir la misma, y los *mapas de densidad*† resultantes para cada parche se combinan para obtener el resultado final (*Gao et al.*, 2020).

En general, el desempeño de dichos modelos fue una mejora para la época, pero aún así tenía sus problemas. Gran parte de esto se debe a las dificultades que estos tenían a la hora de comprender y manejar el espacio y la perspectiva capturada en la imagen a partir de sus píxeles. Los parches que componen una imagen pueden ser muy heterogéneos, algunos no conteniendo personas, otros con muy poca densidad (en general al frente), mientras que otros tienen mucha *densidad de la multitud*† (en general al fondo de la imagen).

La *distribucion no uniforme*†, así como la *variación de escala*† entre parches, constituyeron un desafío importante para estos métodos (*Gao et al.*, 2020).

4.6. Arquitecturas Multicolumna

Con el fin de mejorar el desempeño de los modelos a la hora de procesar imágenes con gran *variación de escala*† y un dominio en el que las multitudes ocurren siguiendo una *distribucion no uniforme*†, *Zhang et al.* (2016a) proponen la **arquitectura multicolumna**.

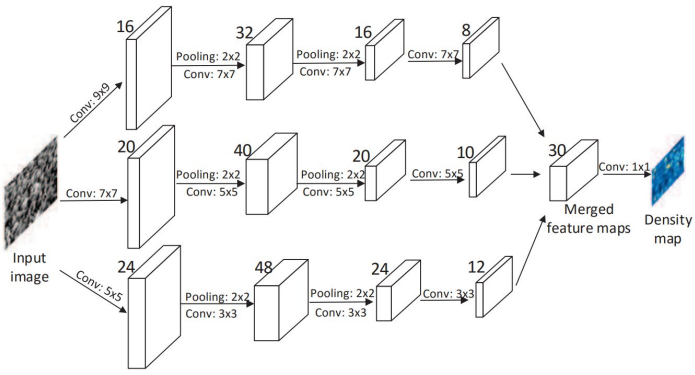


Figura 4.21: Arquitectura de la red multicolumna pionera MCNN (*Zhang et al.*, 2016a)

Como se puede apreciar en la figura 4.21, la misma consiste en multiplicar la red en ramas independientes (columnas) y propagar la entrada a lo largo de las mismas, cada rama contiene distintos campos receptivos, y en base al entrenamiento la red aprende a especializar cada rama a un tipo distinto de parche (de acuerdo a atributos como su escala y perspectiva) y a cómo combinar dicha información (*Khan et al.*, 2022).

4.6.1. MCNN

La arquitectura de este modelo se ilustra en la figura 4.21, y consta en enviar la entrada de la *red neuronal*† a tres ramas *CNN*†, de iguales características, salvo por la cantidad de filtros y el tamaño de núcleo en sus *convoluciones*† (5×5, 7×7 y 9×9) respectivamente.

El trasfondo de este diseño es que un problema sufrido por los modelos pioneros con *CNNs*† era determinar un tamaño de núcleo. Dado que las *convoluciones*† con un núcleo más grande funcionan mejor para parches de baja densidad (como al frente de la imagen), mientras que núcleos más pequeños funcionan mejor para parches más densos (como al horizonte de una multitud), esto permite que cada columna sea adaptativa a la escala del parche (*Zhang et al.*, 2016a).

Este, además, fue el primer trabajo de conteo en incluir una *convolución* 1×1† en la salida de la red (en contraposición a *capas totalmente conectadas*†). Esta *convolución*† efectivamente suma el valor en cada canal del *mapa de densidad*†, reduciendo su dimensionalidad a dimensiones con la misma relación de aspecto que la imagen de entrada. Esto es lo que permite que la entrada de la red sean imágenes o recortes de cualquier dimensión sin que ocurra una distorsión de la misma (*Gong*, 2020).

Otra particularidad de esta red es que al emplear dos capas de max pooling, la resolución espacial es reducida a $\frac{1}{4}$ para cada imagen (antes de generar su mapa de densidad), por lo que para los datos también redujeron (*downsample*) la resolución de los mismos a $\frac{1}{4}$ (*Zhang et al.*, 2016a) . Es posible ver la **MCNN** como una agrupación (*ensemble*) de regresores sencillos (*Gao et al.*, 2020).

4.6.2. Hydra-CNN

El modelo **Hydra-CNN** (*Onoro-Rubio and López-Sastre*, 2016) surge rápidamente después de la **MCNN** (*Zhang et al.*, 2016a). Dicho trabajo plantea, para una imagen, hacer recortes centrados en forma de pirámide (de mayor a menor tamaño), ajustar el tamaño de cada recorte a una resolución fija (72×72 píxeles) y distribuir cada uno a lo largo de las columnas de la red.

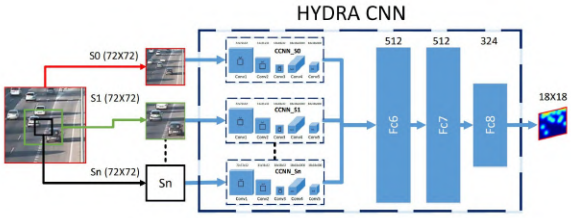


Figura 4.22: Arquitectura de Hydra-CNN (*Onoro-Rubio and López-Sastre*, 2016)

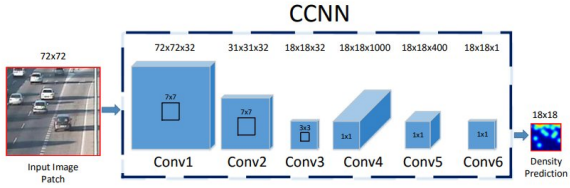


Figura 4.23: Subcomponente CCNN de Hydra-CNN (*Onoro-Rubio and López-Sastre*, 2016)

Cada una de las N columnas (3 por defecto) es encargada de producir un *mapa de densidad*†; debido a las dos *capas de pooling*† empleadas, la resolución del *mapa de densidad*† generado es $\frac{1}{4}$ del parche procesado. Se destaca que durante el *entrenamiento*†, cada columna de la red es entrenada por separado (enviándole y ajustándola solo con los elementos correspondientes a la escala de parche para la mismo).

Por último, la salida de cada una de las columnas es concatenada en un conjunto de *capas totalmente conectadas*† que refinan y determinan el *mapa de densidad*† final (fijando la dimensión de salida).

La arquitectura de la misma se puede apreciar en la figura 4.22, mientras que la arquitectura de cada una de sus columnas CCNN se ilustra en la figura 4.23.

4.6.3. Switching CNN

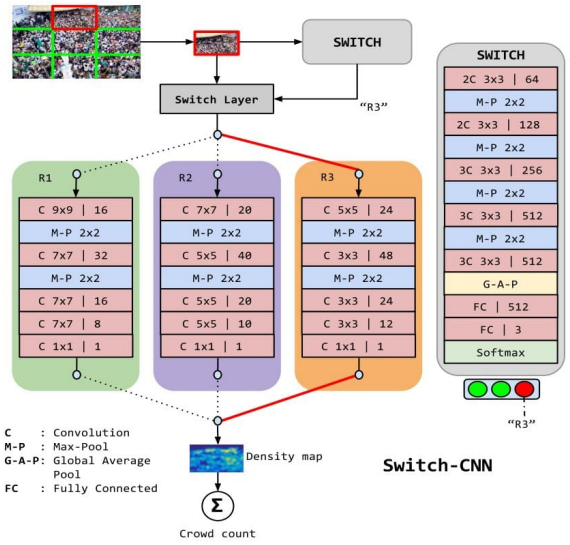


Figura 4.24: Arquitectura de la Switch-CNN (*Babu Sam et al.*, 2017)

Las arquitecturas multicolumna presentan un mayor coste computacional, debido a la cantidad de columnas. Una mejora respecto a **MCNN** (*Zhang et al.*, 2016a), es omitir *inferencias*† de un parche en todas las columnas, salvo la destinada para predecir el mismo acorde a su nivel de densidad (*Khan et al.*, 2022).
El modelo **Switching-CNN** (*Babu Sam et al.*, 2017) propone una arquitectura similar a la de la **MCNN** (*Zhang et al.*, 2016a), pero agrega además una capa de switch; que se encarga de, dado un parche, clasificar *grosso modo* su densidad, con el fin de dirigir la entrada a su columna correspondiente (con la capa de switch). Cada uno de los regresores R_i, así como el switch, pasan por una etapa de pre-entrenamiento antes del *entrenamiento*† del modelo completo (*Babu Sam et al.*, 2017).
La arquitectura de **Switching-CNN** se puede apreciar en la figura 4.24; se destaca como las primeras capas del clasificador para el switch son formadas en su mayoría empleando la **VGG-16** (*Simonyan and Zisserman*, 2014) pre-entrenada como *backbone*†.

4.6.4. Problemas experimentados

El paradigma multicolumna surgió en el año 2016 con la necesidad de mejorar el desempeño del modelo ante el problema de la *variación de escala*†, dado que según la *densidad de la multitud*†, el problema del conteo en base a generación de *mapas de densidad*† para una imagen o parche es drásticamente distinto.

Contar con capas con distintos tamaños de filtro convolucional es una idea que permite dosificar bien la información capturada de los píxeles más próximos acorde a la densidad, pero también tiene sus problemas (*Li et al.*, 2018):

- Estas redes son difíciles de entrenar. Debido a su arquitectura, es complicado evitar que hagan ***sobreajuste***†; más aún para el contexto del conteo de multitudes en esta época, en donde los *conjuntos de datos*† contaban con muy pocas instancias.
- Dependiendo de la estrategia de combinación de información entre columnas, las columnas no diseñadas para la instancia procesada pueden terminar agregando **ruido** al resultado.
- Vinculado a los problemas anteriores, estas redes **demoraban bastante tiempo en entrenarse y converger** a un resultado, y aumentar la profundidad en cada columna tenía un impacto y coste mayor que en una arquitectura de red más sencilla y con menos redundancia.

Como se presentará en la siguiente sección, estos problemas han impulsado a la exploración de otro paradigma que trajo mejores resultados al campo del problema. Pese a esto, se destaca también que, en base a los nuevos paradigmas y a estas lecciones aprendidas, las redes multicolumna han de cierta manera resurgido con arquitecturas híbridas; ninguna de estas es central para este estudio, pero se destacan algunas de ellas en la sección B.2.

4.7. Arquitecturas de una columna (*Single column*)

Posterior a las arquitecturas multicolumna 4.6, *Li et al.* (2018) propone esta arquitectura junto al modelo **CSRNet**, este paradigma plantea una sola columna, de mayor profundidad que los modelos anteriores, contando siempre con un *backbone*† pre-entrenado, que comúnmente es la **VGG-16** (*Simonyan and Zisserman*, 2014). De esta forma se beneficia del *aprendizaje por transferencia*†, lo cual ayuda a la red a extraer y refinar información a partir de la imagen de entrada (tarea previamente sobrellevada con los distintos tamaños de filtros convolucionales en cada columna). Este modelo obtuvo mejores resultados que el modelo multicolumna CrowdNet (*Boominathan et al.*, 2016) que fue el pionero en introducir la **VGG-16** como *backbone*†.

El uso de una única columna no descarta sin embargo el concepto de *aprendizaje multitarea*†, es decir, considerar la arquitectura de la red en módulos que puedan encargarse de tareas de diversa índole, como por ejemplo los mecanismos de *atención*†. Si bien estos módulos pueden ser parcialmente independientes, suelen estar mucho más conectadas a lo largo de la red que los modelos multicolumna.

Para el *aprendizaje multitarea*† es frecuente la aplicación de una *función de pérdida de composición*†, evaluando con cada función las distintas tareas de la red.

Estas redes además tienden a ser más profundas, lo que permitió dejar de trabajar con recortes de la imagen (aunque aún existen modelos que trabajan con parches) y pasan a usar la imagen entera como entrada. Esto, si bien vuelve el problema más difícil, también ofrece el contexto global de la imagen en lugar de recortes aislados.

4.7.1. CSRNet: Convoluciones dilatadas

Como se mencionó anteriormente, **CSRNet** (*Li et al.*, 2018) presenta una estructura más sencilla que los modelos del paradigma multicolumna, siendo conformada por dos componentes.

El primero es su *backbone*†, o *Frontend Map Extractor (FME)*, el cual emplea las primeras 10 capas convolucionales de la **VGG-16** (*Simonyan and Zisserman*, 2014), con un tamaño de filtro de 3 y tres *capas de pooling*† entre las mismas.

Estas capas se cargan inicialmente con pesos pre-entrenados para hacer uso del *aprendizaje por transferencia*†. Las mismas reducen la dimensionalidad de la imagen entrante a $\frac{1}{8}$ de sus dimensiones, refinando el nivel de información a un espacio más semántico en el que debido a las *convoluciones*† cada punto contempla información sobre sus elementos adyacentes.

El segundo componente de la red, siguiendo la arquitectura *codificador-decodificador*†, agrega 6 capas de *convoluciones dilatadas*† y una *convolución* 1×1† al final. Cada capa de *convoluciones dilatadas*† 2-D se define como:

$$y(m,n) = \sum_{i=1}^{M} \sum_{j=1}^{N} x(m + r \times i, n + r \times j) w(i,j), \tag{4.24}$$

en donde $w(i,j)$ es un filtro con longitud y ancho de M y N, y r es el ratio de dilatación. Se observa que con $r = 1$ esta formula se convierte en una *convolución*† normal (*Li et al.*, 2018). Aquí un núcleo pequeño con un filtro $k \times k$ es agrandado a $k + (k-1)(r-1)$. Un ejemplo de su aplicación se puede apreciar en la figura 4.25.

Las capas de *convolución dilatada*† han tenido bastante éxito en tareas de segmentación (*Yu and Koltun*, 2015; *Chen et al.*, 2017a,b), y son una buena alternativa a las *capas de pooling*†, las cuales si bien son útiles para mantener invarianza y controlar el *sobreajuste*†, también reducen drásticamente la resolución espacial, causando la pérdida de información espacial en el mapa de atributos. Una comparación de esto se muestra en la figura 4.26.

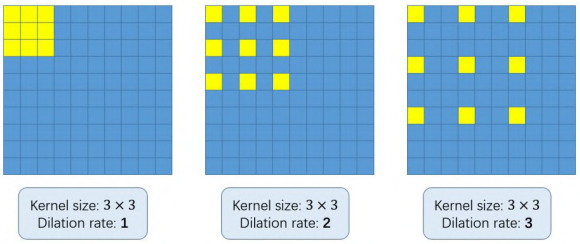


Figura 4.25: *Convolución dilatada*† 3×3 variando su ratio de dilatación r(*Li et al.*, 2018)

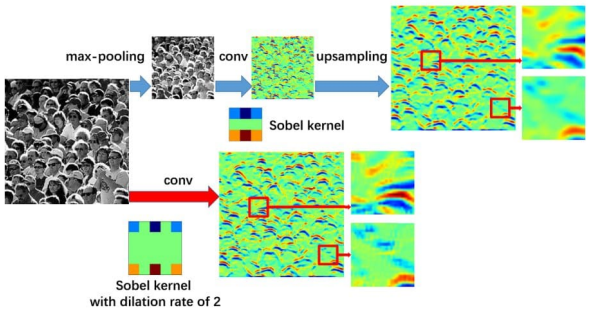


Figura 4.26: Comparación entre *convolución dilatada*† (rojo) y combinar *max pooling*†, *convolución*† y *upsampling* (*Li et al.*, 2018)

Como se puede ver, estas *convoluciones*† expanden las dimensiones del espacio a través de filtros con núcleo disperso, lo cual permite agrandar los campos receptores sin incrementar el número de parámetros o de coste computacional (como podría ser agregando *capas convolucionales*†)

Configurations of CSRNet			
A	B	C	D
input(unfixed-resolution color image)			
front-end (fine-tuned from VGG-16)			
conv3-64-1 conv3-64-1			
max-pooling			
conv3-128-1 conv3-128-1			
max-pooling			
conv3-256-1 conv3-256-1 conv3-256-1			
max-pooling			
conv3-512-1 conv3-512-1 conv3-512-1			
back-end (four different configurations)			
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-256-1	conv3-256-2	conv3-256-4	conv3-256-4
conv3-128-1	conv3-128-2	conv3-128-4	conv3-128-4
conv3-64-1	conv3-64-2	conv3-64-4	conv3-64-4
conv1-1-1			

Figura 4.27: Configuraciones de la arquitectura de CSR-Net (*Li et al.*, 2018).

Distintas configuraciones de la arquitectura de **CSRNet** se muestra en la figura 4.27. En el estudio se experimentaron con varias configuraciones de la arquitectura, siendo **CSRNet-B** la que se desempeñó mejor en la evaluación y por ende la solución propuesta. La notación en las *capas convolucionales*† de *conv3-k-f-r*, en donde k es el tamaño del filtro, f es la cantidad de filtros y r es el ratio de dilatación.
Las capas de max *pooling*† son conducidas por una ventana de 2×2 píxeles con un desplazamiento de 2 (*Li et al.*, 2018).
Debido a la estructura de la red, la salida de los *mapas de densidad*† generados son de $\frac{1}{8}$ de la dimensión de la imagen de entrada.
Este modelo es tan sencillo y eficaz que múltiples trabajos posteriores lo han empleado de *backbone*†, enfocándose en cómo refinar aún más los atributos que esta red *codificador-decodificador*† extrae.

En estos últimos años han ido surgiendo diversos modelos, en su mayoría dentro de este paradigma, y los mismos serán expuestos en las secciones subsiguientes del marco teórico. La estructura del mismo pasará a tener un enfoque orientado a técnicas y módulos propuestos para resolver los problemas a los que se enfrentan los modelos de conteo de multitudes.

4.8. Técnicas para atacar el problema de escala

Gran parte del avance en el campo se ha dedicado a modelar explícita o implícitamente la noción de **escala** en las imágenes. Esto es debido a que un modelo de fusión tradicional que ignore la escala de la imagen tendrá una mayor dificultad en llegar a una buena solución. Si bien lo ideal sería contar con anotaciones de profundidad en las imágenes (como ofrece el *conjunto de datos*† ShanghaiTechRGBD (*Lian et al.*, 2019)), esto no es la norma para el resto.

Para combatir el problema de la escala, los modelos o bien agregan **módulos de escala** a su arquitectura (que a través del *entrenamiento*† refinen esta información de la imagen), o la modelan de forma **implícita** a través de *conexión residual*†, apelando a combinar información de distintas capas.

En los artículos procesados durante la investigación de este trabajo, se observaron dos enfoques para el modelado de escala **explícito**: modelar la escala como un intervalo **discreto** (que modela intervalos de distancia), o modelar la escala como un valor en un intervalo **continuo**.

4.8.1. Modelado de escala explícito

4.8.1.1. Modelado de escala discreta

Modelar la escala en **intervalos discretos** es una tarea más simple de resolver que en un intervalo continuo, y es un área que se ha trabajado con anterioridad en el campo (por ejemplo a la hora de combinar información de las distintas columnas de un modelo multicolumna). Pese a no tener información tan refinada o exacta (que sea capaz de representar la realidad), este tipo de modelado aún así logra cumplir el objetivo de refinar e incorporar información de la escala en el modelo.

Una forma de realizar esta tarea es aproximar la escala en base a las anotaciones (para poder con ella asistir el *entrenamiento*†).

El trabajo **MBTTBF-SCFB** (*Sindagi and Patel*, 2019) propone basarse en los principios de la segmentación superpíxel, empleando el algoritmo superpíxel *SLIC* (*Achanta et al.*, 2010), y combinarlo con la *segmentación watershed* (*Beucher*, 1992), con lo cual se permite estimar la transformación de distancia para la localización de cada cabeza en un modelo de campos aleatorios Markovianos (MRF).

Figura 4.28: Estimación de escala (*Sindagi and Patel*, 2019).

En la figura 4.28 se puede apreciar la comparación entre este tipo de modelado (c), en contraste a:

- Emplear una escala constante (a), como puede ser el tamaño promedio de las cabezas, lo cual falla cuando hay una alta varianza de escalas debido a la *distorsión de perspectiva*†
- Emplear un modelo de tamaño de escala según los vecinos más cercanos (b), que falla cuando la multitud sigue una *distribucion no uniforme*†, en la cual por ejemplo hay zonas vacías de gente en una gran profundidad.

Este algoritmo le asigna a cada punto una escala discreta, y tales datos pueden ser empleados para supervisar el *entrenamiento*† de bloques de extracción de escala dentro de una *CNN*†.

4.8.1.2. Modelado de escala continua

Modelar la escala en un intervalo continuo es una tarea de mayor dificultad que el modelado de escala discreta, por lo cual es más difícil de lograr con exactitud y a su vez mantener baja la complejidad del modelo. Sin embargo, también es un modelado que aporta información mucho más refinada en el problema del conteo de multitudes, y que con un modelo bien entrenado sí podría ser capaz de representar la realidad tridimensional en base a las imágenes (bidimensionales).

El trabajo **PGCNet** (*Yan et al.*, 2019a) propone usar *convoluciones*† guiadas por perspectiva (**PCG**), en las que estos bloques combinan la información de la salida anterior junto con la salida de un autocodificador (*autoencoder*) **PENet** (*Pespective Estimation Network*) previamente entrenado para estimar mapas de perspectiva. Su arquitectura se puede apreciar en la figura 4.29.

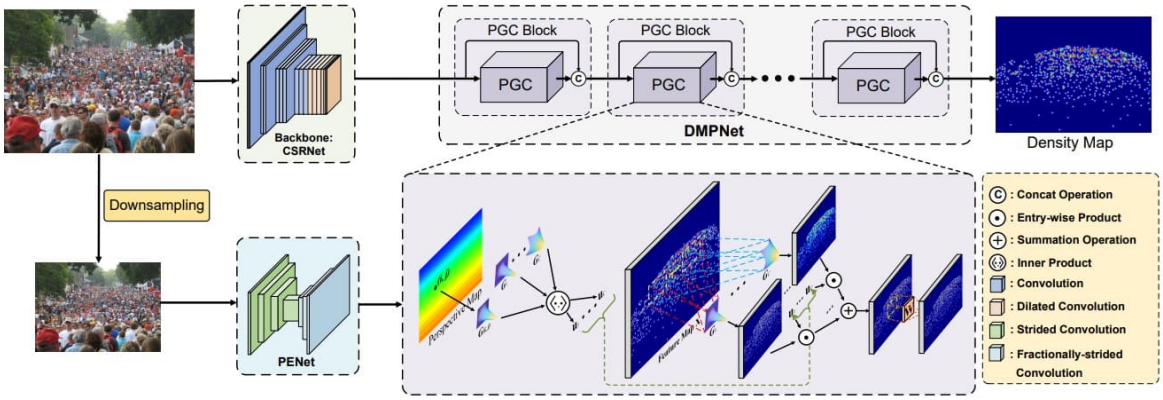


Figura 4.29: PGCNet: Arquitectura (*Yan et al.*, 2019a)

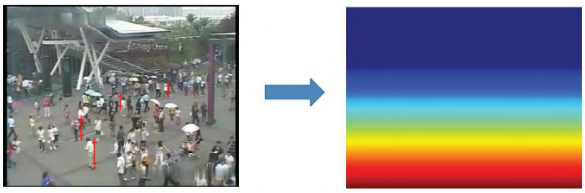


Figura 4.30: Mapa de perspectiva WorldExpo'10 (*Zhang et al.*, 2015).

En el conjunto **WorldExpo'10** (*Zhang et al.*, 2015), se cuentan con anotaciones de pies a cabeza de los individuos, las mismas fueron usadas con un modelo de regresión para generar un mapa de perspectiva para cada imagen del conjunto, el cual refleja la profundidad del píxel. Para generar el mapa de perspectiva, este modelo asume que cada individuo mide 1.75m y en base al tamaño en píxeles de su *bounding box* le asigna a cada píxel una anotación de profundidad. Un ejemplo de mapa de perspectiva se ilustra en la figura 4.30.

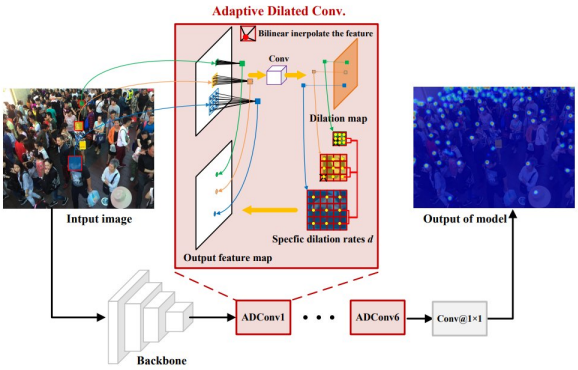


Figura 4.31: Convoluciones dilatadas adaptativas (*Bai et al.*, 2020).

Otro enfoque interesante que se puede ver en la figura 4.31 es planteado por el modelo **ADSCNet** (*Bai et al.*, 2020), que propone el uso de convoluciones dilatadas adaptativas **ADConv**, en donde en base a una estimación supervisada de escala cada **ADConv** varía su ratio de dilatación ampliando de ser necesario el área de la *convolución*†.

4.8.2. Modelado de escala implícito

Esta técnica parece ser bastante poderosa, y consta de enriquecer la información de una capa en base a *conexiones residuales*† con capas anteriores (más superficiales).

El fundamento detrás de esto, es que en una *CNN*† (*Long et al.*, 2014):

- Los mapas de atributos en **capas superficiales** tienden a inferir detalles de más bajo nivel local, y tienen más presente la información espacial de la escena. Sin embargo, estos mapas también contienen ruido (que en un flujo normal se va filtrando a lo largo de la red).
- Los mapas de atributos en **capas profundas** están mucho más refinados y su información es de más alto nivel; en particular, capturan mucha información semántica gracias a su dimensión reducida, y por como funcionan las *convoluciones*† aportan un contexto global y de mayor claridad. Sin embargo, en el camino pierden información sobre la resolución espacial, lo que causa problemas de localización (*Sindagi and Patel*, 2019).

Es por esto que resulta de gran interés considerar en las capas finales ciertos atributos inferidos inicialmente sobre la resolución espacial (en lugar de emplearlos sólo para generar la información semántica).

4.8.2.1. Multi-Level Bottom-Top and Top-Bottom Feature Fusion (MBTTBF-SCFB)

En 2019 el artículo **MBTTBF-SCFB** (*Sindagi and Patel*, 2019) propone un flujo de retroalimentación y extracción de atributos entre las últimas capas de la red (a diferencia de sólo la última).

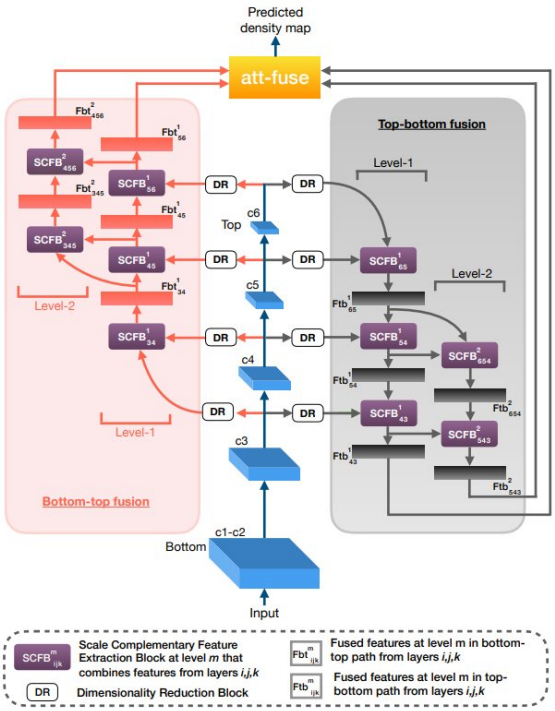


Figura 4.32: MBTTBF-SCFB: Multi-Level Bottom-Top and Top-Bottom Feature Fusion (*Sindagi and Patel*, 2019).

- Como se ve en su arquitectura en la figura 4.32, la salida del *backbone*† (**VGG** (*Simonyan and Zisserman*, 2014)) está conectada a ramas de fusión que tienen *conexión residual*† en ambas direcciones de la red.
- La salida final es determinada por los mapas generados en las cuatro ramas siendo procesadas a través de un mecanismo de *atención*†.

Esta técnica permite:

- Aprender atributos de mayor complejidad, permitiendo que información espacial y semántica sea intercambiada entre capas de manera bidireccional.
- El camino *Bottom-Top* asegura el flujo de información espacial, mientras que el *Top-Bottom* propaga información contextual a las capas inferiores.
- La retroalimentación entre ambos caminos minimiza el ruido propagado a la capa superior (en la dirección *bottom-top*), así como logra que la información del contexto no suprima los detalles en las capas inferiores.

Este trabajo también concluye que si bien es posible combinar mapas de atributos residuales sumando o concatenando, esto limita la complementación de información, por lo que es recomendable que los

bloques que unan la información **SCFB** crucen los atributos de ambas ramas convolucionales (*Sindagi and Patel*, 2019).
Una observación interesante de este trabajo es que se realizó una experimentación de arquitecturas (figura 4.33), probando varias maneras de combinar la información entre capas y obteniendo una mejora al evaluar con los *conjuntos de datos*† *ShanghaiTech-A* (*Zhang et al.*, 2016b) y *UCF-QNRF* (*Idrees et al.*, 2018b) cuanto más comunicadas estaban. La figura 4.34 muestra los resultados obtenidos en la evaluación.

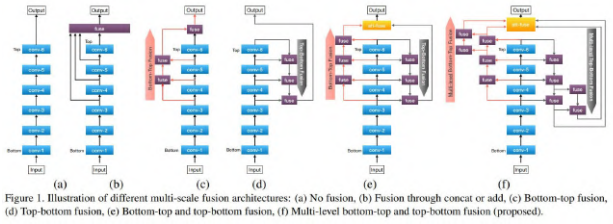


Figure 1. Illustration of different multi-scale fusion architectures: (a) No fusion, (b) Fusion through concat or add, (c) Bottom-top fusion, (d) Top-bottom fusion, (e) Bottom-top and top-bottom fusion, (f) Multi-level bottom-top and top-bottom fusion (proposed).

Figura 4.33: Experimentación de arquitectura MBTTBF-SCFB (*Sindagi and Patel*, 2019)

Dataset	Shanghaitech-A[74]		UCF-QNRF[19]	
Method	MAE	MSE	MAE	MSE
Baseline (Fig. 1a)	78.3	126.6	150.2	220.1
Baseline + fuse-a (Fig. 1b)	73.6	118.4	140.3	210.8
Baseline + fuse-c (Fig. 1b)	73.4	115.6	135.2	200.2
Baseline + BT + fuse-c (Fig. 1c)	68.1	122.2	114.1	185.2
Baseline + TB + fuse-c (Fig. 1d)	70.2	118.5	120.1	188.1
Baseline + BTTB + fuse-c (Fig. 1e)	66.9	112.2	115.4	174.5
Baseline + MBTTB + fuse-c (Fig. 1f)	63.2	108.5	105.5	169.5
Baseline + MBTTB + SCFB-NS (Fig. 2)	62.5	105.1	102.1	168.1
Baseline + MBTTB + SCFB (Fig. 2)	60.2	94.1	97.5	165.2

Figura 4.34: MBTTBF-SCFB: Experimentación (*Sindagi and Patel*, 2019).

4.8.2.2. SFANet

El modelo **SFANet** *(Dual Path Multi-Scale Fusion Networks with Attention for Crowd Counting)* (*Zhu et al.*, 2019), se compone de la **VGG-16** (*Simonyan and Zisserman*, 2014) como *backbone*†, y luego parte la red en dos ramas, alimentando ambas con *conexiones residuales*† desde el *backbone*†.

Las ramas son:

- Una de estimación inicial del ***mapa de densidad***†: Emplea la información refinada y la de capas anteriores para formar una *inferencia*† inicial del *mapa de densidad*†.
- Otra de *inferencia*† de un **mapa de *atención***†: Infiere el nivel de importancia que tiene cada píxel a los efectos de la generación del *mapa de densidad*†. Su valor es análogo a un mapa de probabilidad que indica las regiones donde es más probable que haya individuos.

Por último, realiza una multiplicación elemento-elemento entre el *mapa de densidad*† y el mapa de *atención*†. Esto suprime del *mapa de densidad*† sus valores en las regiones con probabilidad baja o nula de contener individuos, permitiendo que el *entrenamiento*† se enfoque en la tarea de aprendizaje para estas regiones. La arquitectura de este modelo se ilustra en la figura 4.35.

La supervisión de ambas ramas se realiza a través de una *función de pérdida de composición*†, que combina la *función de pérdida euclidiana* (ℓ_2)† (para la generación del *mapa de densidad*†) con una *función de pérdida*† diseñada para corregir los mapas de *atención*†, la cual se basa en la entropía de clase binaria:

$$\ell_{att} = -\frac{1}{N}\sum_{i=1}^{N}(A_i^{GT} log(P_i) + (1 - A_i^{GT}) log(1 - P_i)), \tag{4.25}$$

en donde A_i^{GT} es el *valor anotado*† del mapa de atención, calculada tras aplicar un *filtro gaussiano*† a la matriz de anotaciones, y P_i es la probabilidad de que cada píxel en la región de atención sea activado por la función sigmoide.

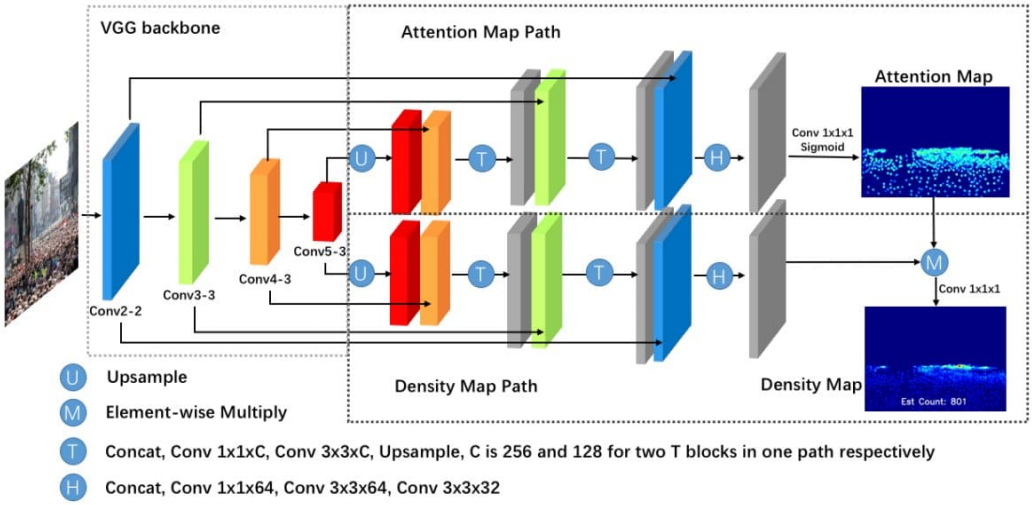


Figura 4.35: SFANet: Arquitectura (*Zhu et al.*, 2019)

4.8.2.3. Scale-Adaptive Selection (SASNet)

En 2021 el modelo **SASNet** (*Song et al.*, 2021) plantea que modelos como **SaCNN** (*Zhang et al.*, 2018) y **W-Net** (*Valloli and Mehta*, 2019) generan una predicción explotando los mapas de atributos de distintas capas, pero lo hacen de forma agnóstica a la escala, lo cual ignora la correspondencia entre la profundidad de los mapas de atributos y las escalas que captan mejor. Una particularidad interesante de este modelo es que trabaja **partiendo la imagen por parches** (como era popular con los modelos multicolumna). Este modelo propone combinar la información de los niveles de atributos a través de selección adaptativa y sin emplear anotaciones o estimación de la escala. Su arquitectura se puede apreciar en la figura 4.36.

Esta selección se logra con un mecanismo de *atención*†, en el cual para cada parche se predice un puntaje de confianza por cada *mapa de densidad*† generado y, tras normalizar los mapas de confianza entre sus píxeles, se realiza un promedio ponderado por cada píxel del *mapa de densidad*† final. Esto ayuda a reducir la inconsistencia entre mapas de atributos discretos y una variación continua de escala. Una visualización del resultado al aplicar esta técnica se aprecia en la figura 4.38 (*Song et al.*, 2021).

Este trabajo emplea una *función de pérdida de composición*†, que combina las siguientes funciones:

$$\mathcal{L} = \mathcal{L}_{den} + \lambda \mathcal{L}_{conf} + \mathcal{L}_{pra}, \tag{4.26}$$

en donde:

- Para supervisar las ramas de densidad, se evalúa la *función de pérdida euclidiana* $(\ell_2)^\dagger$:
  - $\mathcal{L}_{den} = \sum_i^5 ||D_i - D^{gt}||_2^2$
  - En donde D_i es el *mapa de densidad*† generado por la rama P_i
- Para supervisar las ramas de confianza la *función de pérdida*† es:
  - $\mathcal{L}_{ce}(\mathcal{C}_{i,m,n}, \mathcal{C}^{gt}_{i,m,n}) = \mathcal{C}^{gt}_{i,m,n} \cdot log(\mathcal{C}_{i,m,n}) + (1 - \mathcal{C}^{gt}_{i,m,n}) \cdot (1 - log(\mathcal{C}_{i,m,n}))$
  - $\mathcal{L}_{conf} = \frac{1}{\sum_{i=1}^5 |\mathcal{K}_i|} \sum_{i=1}^5 \sum_{(m,n)\in\mathcal{K}_i} \mathcal{L}_{ce}(\mathcal{C}_{i,m,n}, \mathcal{C}^{gt}_{i,m,n})$
  - En donde: $\mathcal{L}_{ce}$ es la *función de pérdida*† de entropía binaria cruzada (BCE), $\mathcal{C}^{gt}_{i,m,n}$ es el *valor anotado*† generado en la Fig 4.37 para el sub-parche (m, n) con el mapa de atributos D_i. $\mathcal{K}_i$ es el conjunto de parches con etiqueta de confianza 0 o 1.

- Este modelo, para mejorar su convergencia en *entrenamiento*†, propuso además la *función de pérdida PRA*†, de ecuación 4.23, que ajusta priorizando corregir las regiones más sobre- o sub- estimadas (de mayor error).

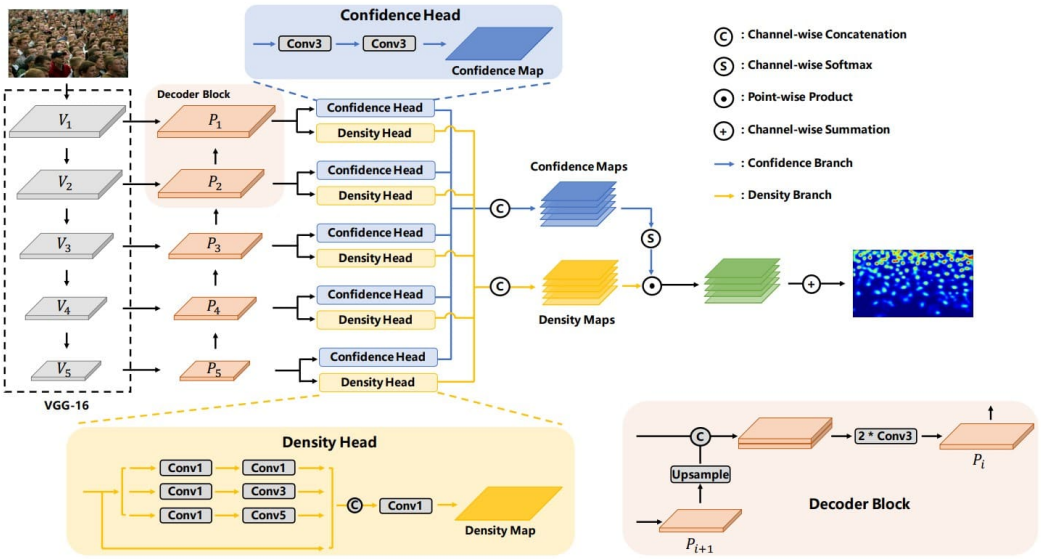


Figura 4.36: SASNet: Arquitectura (*Song et al.*, 2021)

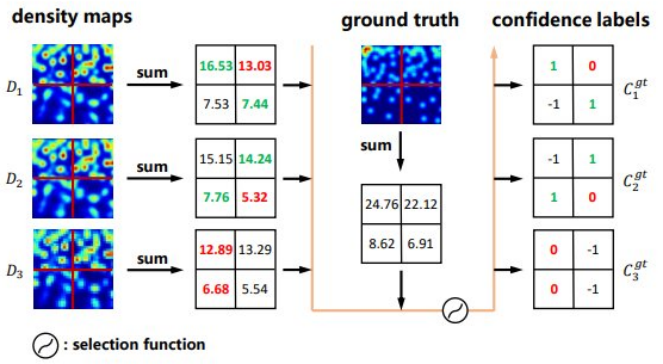


Figura 4.37: SASNet: *Evaluación*† con el *valor anotado*† para la rama de confianza. 1 se asigna al valor más cercano, 0 al más lejano, el resto es ignorado durante el *entrenamiento*†. (*Song et al.*, 2021)

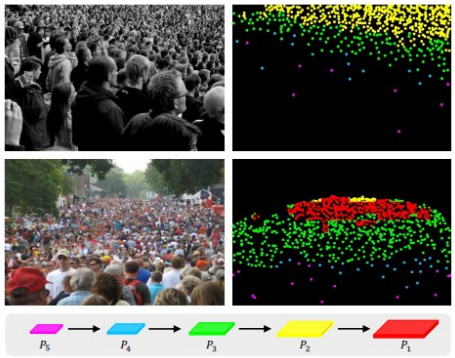


Figura 4.38: SASNet: Selección adaptiva, cada color es el feature level seleccionado por SASNet para el punto (entre los cinco niveles $\{P_1, P_2, P_3, P_4, P_5\}$) (*Song et al.*, 2021).

Se destaca como la figura 4.38 es una visualización, pero en tiempo de *inferencia*† todos los mapas de atributos son considerados por la suma ponderada a través del mecanismo de *atención*†; esto ayuda a mitigar la diferencia del modelado entre que los niveles de mapas de atributos sean discretos, frente a que la escala real es continua.

Se puede apreciar como la información rica en detalles se encuentra en mapas de atributos de alta resolución convolucional, lo que ayuda a predecir cabezas pequeñas, mientras que los mapas de atributos más superficiales de baja resolución tienen rica información contextual que funciona mejor para cabezas grandes. Se destaca como no es trivial esta correspondencia, puesto que la partición de los bordes entre zonas no es homogénea (de la misma forma que la *densidad de la multitud*† suele tener una *distribucion no uniforme*†).

4.9. Trabajos basados en funciones de pérdida probabilísticas

Si bien en la elaboración de este trabajo la investigación no se centró puntualmente en estos modelos, los mismos siguen siendo de interés, habiendo sido empleados para generar resultados en la sección de experimentación 6.

4.9.1. Bayesian Loss

El modelo **BayL**, también conocido como Bayes, BL o Bay, propuesto en *Bayesian Loss for Crowd Count Estimation with Point Supervision* (*Ma et al.*, 2019), es el que introduce la *función de pérdida bayesiana*†.

4.9.1.1. Función de pérdida bayesiana

Esta función busca construir un modelo de *probabilidad de contribución* a partir de las anotaciones de los puntos. En lugar de restringir el valor a cada píxel en el *mapa de densidad*†, la *función de pérdida*† durante el *entrenamiento*† adopta una forma de supervisión basada en el principio de estimación por máxima verosimilitud, que logra una supervisión más confiable sobre la esperanza de conteo de cada punto:

$$\ell_{Bayes} = \sum_{n=1}^{N} \mathcal{F}(1 - \sum_{m=1}^{M} p(y_n|x_m) D^{est}(x_m)), \tag{4.27}$$

en donde $\mathcal{F}(\cdot)$ es una función de distancia (l_1 en el trabajo (*Ma et al.*, 2019)), $p(y_n|x_m)$ es la probabilidad a posteriori, D^{est} es la densidad estimada, mientras que y_n es la densidad objetivo deseada.

4.9.1.2. BayL

El resto de la arquitectura del modelo es bastante sencilla, empleando la VGG-19 (*Simonyan and Zisserman*, 2014) de *backbone*†, seguida de dos *capas convolucionales*† con filtro 3×3 y terminando en una *convolución* 1×1†.

Para entrenar dicho modelo, sin embargo, es necesario a partir de las anotaciones crear un *valor anotado*† que permita aplicar la *función de pérdida bayesiana*†. Para hacerlo, es necesario primero analizar los datos etiquetados, luego en base a cada anotación se deben construir las funciones de verosimilitud para cada punto, y finalmente a cada punto anotado se le aplica un *filtro gaussiano*† con dicha función.

Para mejorar la calidad de este método, el trabajo propone además modelar los **píxeles del fondo**. Siendo dichos puntos lejanos de las anotaciones, en lugar de asignarles una etiqueta y_n se les asigna una etiqueta de fondo y_0. Asumir que para todo n, $p(y_n) = p(y_0)$ permite simplificar la fórmula de Bayes y modelar la probabilidad como:

$$p(y_n|x_m) = \frac{p(x_m|y_n)}{\sum_{n=1}^{N} p(x_m|y_n) + p(x_m|y_0)}. \tag{4.28}$$

Ejemplos del desempeño de este modelo se pueden apreciar en la figura 4.40.

Estos mapas pueden ser visualizados como mapas de entropía, calculando para cada punto x_m el siguiente valor:

$$Ent(x_m) = -\sum_{n=0}^{N} p(y_n|x_m) \cdot ln(p(y_n|x_m)), \tag{4.29}$$

el mismo mide la incertidumbre de de cada píxel a aportar un valor al *mapa de densidad*[†] o a pertenecer a un elemento (como el fondo). Variaciones de los mapas de entropía se aprecian en la figura 4.39

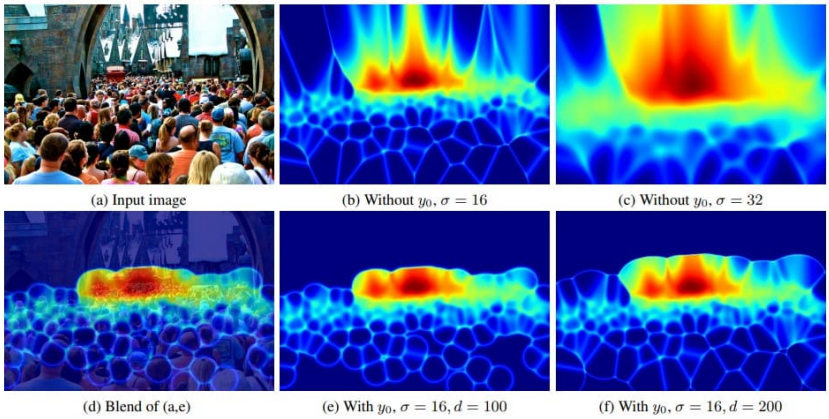


Figura 4.39: Bayesian Loss: Configuraciones de mapas de entropía. σ controla la suavidad de la probabilidad a posteriori. d controla el margen entre frente y fondo. (*Ma et al.*, 2019)

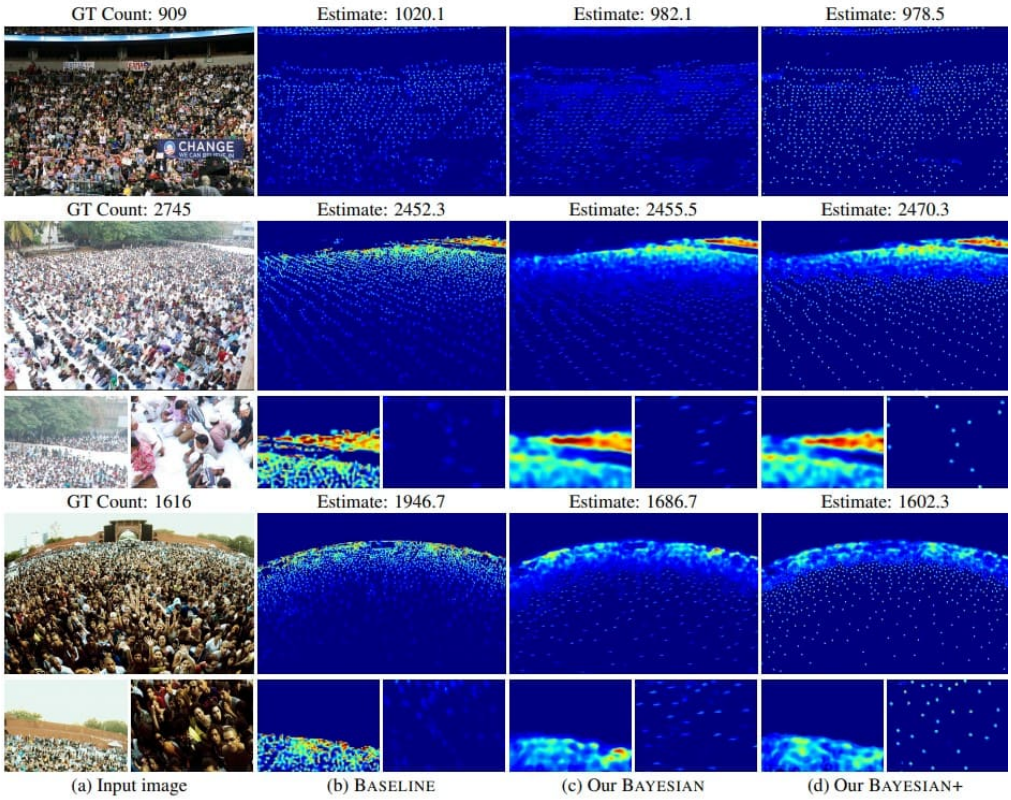


Figura 4.40: Bayesian Loss: Ejemplos de resultados. *Baseline* es la misma arquitectura pero con *función de pérdida euclidiana* $(\ell_2)^{\dagger}$, y *Bayesian+* aplica la mejora de representar los píxeles del fondo en el modelo probabilístico (*Ma et al.*, 2019)

4.9.2. DM-Count

El modelo DM-Count *(Distribution Matching for Crowd Counting)* (*Wang et al.*, 2020a), se inspira en el trabajo presentado en la sección anterior (Bayesian Loss (*Ma et al.*, 2019)). Ataca varias de las debilidades que dicho método presentaba:

- DM-Count no necesita un método para construir un mapa de probabilidades en base a las funciones de verosimilitud en cada punto. Sino que hace uso del mapa binario de anotaciones, más concretamente emplea la cantidad total de personas anotadas $\hat{C}$ y un mapa de anotaciones normalizado en el que cada 1 pasa a ser $\frac{1}{\hat{C}}$).
- Emplear *función de pérdida bayesiana*† podría conducir a un sistema de ecuaciones indeterminado con infinitas soluciones.
- Emplear *función de pérdida bayesiana*† podría además dar un valor de 0 para un *mapa de densidad*† estimado si la imagen de entrada es muy disimilar a las presentes en el *conjunto de entrenamiento*†

Este trabajo propone usar principios de teoría de la información, y de **transporte óptimo** (*Villani*, 2009), que refiere al coste óptimo de transformar una distribución de probabilidad en otra.

Con esto el modelo se puede entrenar para que ajuste los valores del *mapa de densidad*† generado a una distribución que minimice el coste de transporte óptimo respecto al mapa de puntos anotado.

Para esto, el método propone la aplicación de una *función de pérdida de composición*† por la diferencia de personas anotadas en la imagen respecto a las detectadas por el modelo: $\ell_C(z, \hat{z}) = ||z||_1 - ||\hat{z}||_1$, en donde z es el mapa de densidad estimado para una imagen, y $\hat{z}$ es el mapa binario de anotaciones. La norma de los mismos es la cantidad de individuos (estimados y anotados respectivamente) en la imagen.

Y a modo de poder ajustar los mapas de densidad generados a la distribución deseada, se le compone ℓ_{OT} (la *función de pérdida de OT (Transporte Óptimo)*†) y ℓ_{TV} (la *función de pérdida de TV (Variación Total)*†), ambas detalladas en los apéndices B.1.4, B.1.5 respectivamente:

$$\ell(z, \hat{z}) = \ell_C(z, \hat{z}) + \lambda_1 l_{OT}(z, \hat{z}) + \lambda_2 ||z||_1 \ell_{TV}(z, \hat{z}), \tag{4.30}$$

en donde λ_1 y λ_2 son *hiperparámetros*†. Se destaca como ℓ_{TV} es multiplicada por la cantidad total de personas, para convertirla a una escala similar a las anteriores.

El resto de la arquitectura del modelo es idéntica a la empleada en el trabajo **BayL** (*Ma et al.*, 2019), a modo de minimizar las variables en la experimentación y destacar el contraste entre las técnicas. Una visualización de las mismas se aprecia en la figura 4.41.

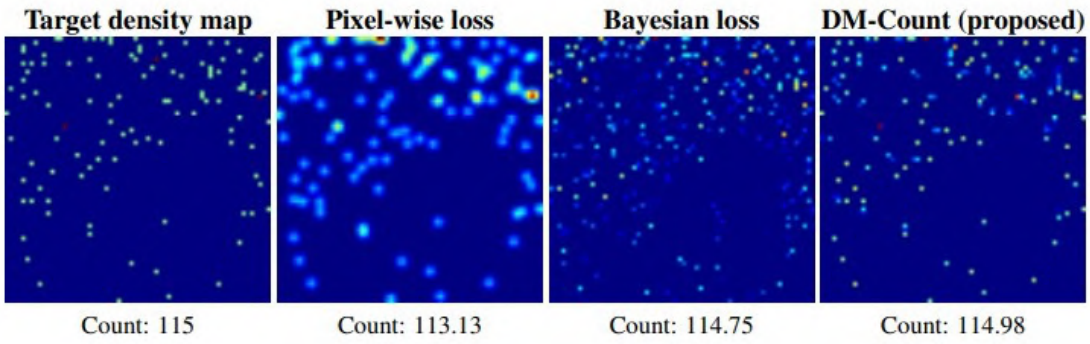


Figura 4.41: DM-Count: Comparación de *mapas de densidad*† generados por distintas *funciones de pérdida*†. El basado en error de píxel (como *función de pérdida euclidiana* (ℓ_2)† es más borroso, el generado por la *función de pérdida bayesiana*† es más nítido pero disimilar al mapa objetivo. (*Wang et al.*, 2020a)

4.10. Contexto actual

Del trabajo *Revisiting Crowd Counting: State-of-the-art, Trends, and Future Perspectives* (*Khan et al.*, 2022) se destacan las figuras 4.42 y 4.43 que compilan el *MAE*† para varios modelos a lo largo de varios *conjuntos de evaluación*†.

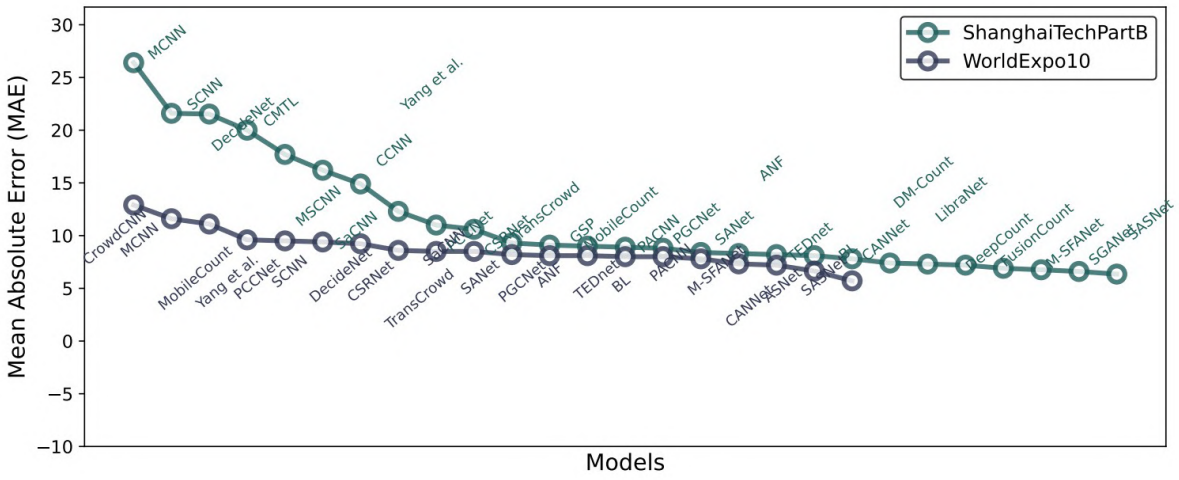


Figura 4.42: (*Khan et al.*, 2022). Tabla de resultados compilada para ShanghaiTech B (*Zhang et al.*, 2016b) y WorldExpo'10 (*Zhang et al.*, 2015).

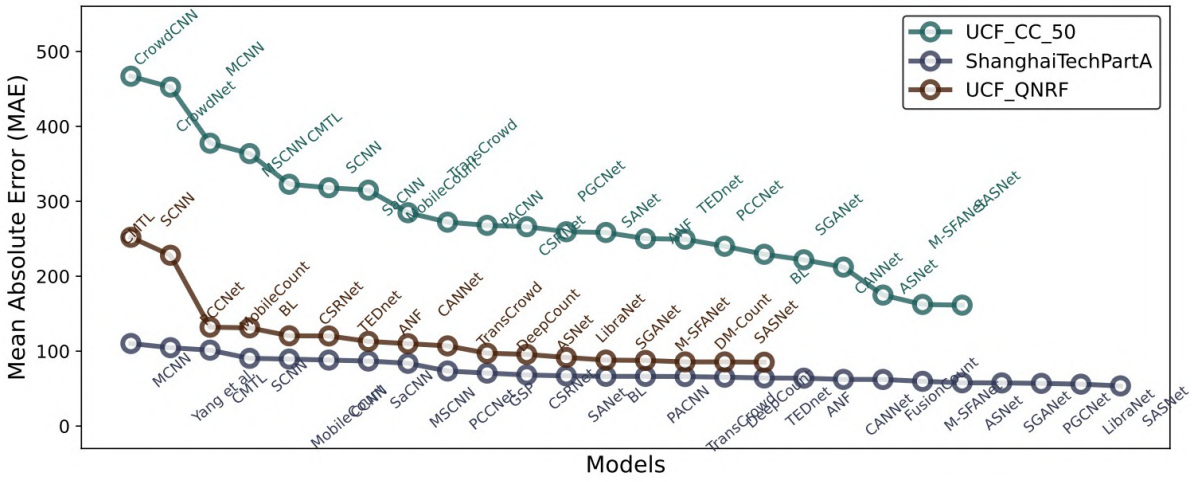


Figura 4.43: (*Khan et al.*, 2022). Tabla de resultados compilada para ShanghaiTech A (*Zhang et al.*, 2016b), UCF-CC50 (*Idrees et al.*, 2013) y UCF-QNRF (*Idrees et al.*, 2018b).

4.10.1. Observaciones sobre el estado del arte

A lo largo de este trabajo, se han observado algunas particularidades que de una forma u otra sesgan la *evaluación*† y por ende el desarrollo del estado del arte para los modelos de conteo de multitudes. Las mismas son las siguientes:

- Si bien se han propuesto muchas *métricas de evaluación*† (sección 4.3), y se ha justificado la motivación de las mismas, la gran mayoría de trabajos calculan solo *MAE*† y *MSE*†. Esto se debe no sólo a la simpleza de dichas *métricas de evaluación*†, sino también a que debido al ser estas las *métricas de evaluación*† históricamente empleadas, son las que permiten comparación de desempeño con trabajos anteriores.

- Durante una gran cantidad de años los *conjuntos de datos*† fueron bastante reducidos en este campo. Por otra parte ahora, con conjuntos como **Crowd Surveillance** (*Yan et al.*, 2019b), **NWPU-Crowd** (*Wang et al.*, 2020c) y **JHU-Crowd++** (*Sindagi et al.*, 2020), el *entrenamiento*† de modelos es capaz de hacer uso de muchos más recursos, lo que podría suponer un cambio en el desempeño de arquitecturas anteriores.

- Debido a una metodología ya establecida, en general no se entrenan modelos cruzando datos de múltiples *conjuntos de datos*†, sino que para cada *conjunto de datos*† se usa como conjunto de test un sub-conjunto del mismo, salvo para conjuntos de *evaluación*† muy pequeños como **UCFCC-50** (*Idrees et al.*, 2013).
  Esto en parte tiene sentido, dado que ofrece una metodología de *evaluación*† agnóstica a los datos con los que se entrene. Por otra parte sesga los modelos a ser entrenados con una menor diversidad de lo que se podría obtener combinando conjuntos de datos, lo cual sesga su desempeño a su aplicabilidad en conjuntos similares; esto podría ser de mayor utilidad para productos finales de ingeniería (*Chen et al.*, 2021; *Yan et al.*, 2021).

- En el momento en que este estudio fue desarrollado no se logró encontrar fuentes que analicen la calidad, precisión y estandarización de anotaciones entre *conjuntos de datos*†. El artículo (*Gao et al.*, 2020) comenta que se han encontrado muestras de *UCFCC-50* (*Idrees et al.*, 2013) y *ShanghaiTech A* (*Zhang et al.*, 2016b) con errores en su anotación, pero no cita fuentes o ejemplos.

4.11. Exploración de estrategias recientes

A continuación se destacan ciertas estrategias contemporáneas que prometen buen desempeño, o gran crecimiento del área de conteo de multitudes.

Por restricciones de alcance, no se realizó un abordaje en profundidad sobre estas técnicas durante la etapa de investigación del proyecto. Aún así, se destacan como punteros de interés a estrategias o técnicas que parecen prometer grandes mejoras en esta área.

4.11.1. Estudios sobre la capacidad de generalización en modelos

La capacidad de generalización de los modelos es de gran interés para la correcta aplicación del conteo de multitudes, y es sólo en estos últimos años que se publicaron estudios relacionados a la misma.

Trata sobre qué tan capaces son los modelos pre-entrenados de garantizar su desempeño en dominios generales y desconocidos, en contraste a su desempeño en un *conjunto de evaluación*† de dominio igual o similar al de *entrenamiento*†. Es decir, estudiar y mitigar el problema de la *adaptación al dominio*†.

Los tres estudios presentados a continuación proponen a grandes rasgos que el modelo debe ser capaz de tener un buen desempeño a lo largo de múltiples dominios, y por ende es de interés que el modelo logre discriminar y clasificar el dominio de una entrada dada. Este es un atributo a tener en cuenta a lo largo de la red, por ejemplo con mecanismos de *atención*† aplicados a nivel de canales.

Una forma intuitiva de pre-etiquetar estos dominios puede ser asumir que cada *conjunto de datos*† representa un dominio, y etiquetar cada imagen de dicho conjunto con un valor específico. Esta suposición, sin embargo, no ofrece buenos resultados, dado que:

- Existen *conjuntos de datos*† cuya densidad entre instancias o escenas varía ampliamente, por lo que sería erróneo agruparlas bajo el mismo dominio. Un conjunto se ve compuesto, por lo tanto por varios sub-conjuntos de datos.
- Existen *conjuntos de datos*† con instancias muy similares entre sí, por lo que sería deseable agrupar sub-conjuntos de imágenes de múltiples conjuntos de datos en un solo dominio.
- El modelado y noción de un dominio es difícil de determinar, pudiendo variar las representaciones según *densidad de la multitud*†, *variación de escala*†, escenas en común, tipos de localización (urbano, interior, parque), o incluso todos los anteriores. Además, pares de dominios podrán estar a mayor o menor distancia de otros según su similitud. Es por esto que una etiqueta discreta no es suficiente para representar esta información en su totalidad.

Para inferir información sobre los dominios, es necesario entonces aplicar técnicas más elaboradas, como por ejemplo técnicas de *clusterización*† sobre particularidades del dominio (como *K-Means*†) para llegar a una clasificación inicial y posibles refinamientos de la misma (presentados a continuación).

Se observa una similitud con el problema que el modelo **Switching-CNN** (*Babu Sam et al.*, 2017), presentado en la sección 4.24, propuso atacar: discriminar la densidad de parches para elegir con qué tamaño de filtros convolucionales es mejor procesarlos. Estos modelos, sin embargo, trabajan con la *inferencia*† y *atención*† en dominios a nivel de imagen, y no a nivel de parche.

4.11.1.1. Aprendizaje multidominio

El aprendizaje multidominio consiste en mejorar el desempeño general de un modelo entrenándolo a base de dominios combinados; por ejemplo, combinando varios *conjuntos de entrenamiento*† durante la etapa de *entrenamiento*†.

El aprendizaje multidominio se diferencia del fine-tuning o re-*entrenamiento*† en que realiza un *entrenamiento*† desde cero, en lugar de aplicar *aprendizaje por transferencia*† y re-entrenar con un nuevo conjunto objetivo cuyo desempeño es deseable mejorar (sacrificando potencialmente precisión en el dominio con el que originalmente se entrenó).

Los artículos **DCANet** (*Yan et al.*, 2021) y **DKPNet** (*Chen et al.*, 2021) proponen mejorar el desempeño de los modelos de conteo empleando múltiples *conjuntos de datos*† en el *entrenamiento*† de los mismos.

Ambos artículos hallaron no trivial combinar los *conjuntos de datos*†, encontrando que por ejemplo al combinar *ShangaiTech A*, *ShanghaiTech B* (*Zhang et al.*, 2016b) y *UCF-QNRF* (*Idrees et al.*, 2018b), se observaba una mejora en el desempeño en los *conjuntos de evaluación*† de *ShanghaiTech A* y *UCF-QNRF*, pero una degradación en el desempeño al evaluar en *ShanghaiTech B* (*Chen et al.*, 2021). Esto se debe a que los modelos de *aprendizaje profundo*† tienden a enfocarse en aprender regularidades estadísticamente superficiales, en lugar de conceptos abstractos más generales (*Chen and Deng*, 2019). Por lo que, como se menciona en la sección 4.11.1, se vieron obligados a implementar un mecanismo de *inferencia*† y *atención*† del dominio al que una imagen pertenece.

DKPNet

El trabajo **DKPNet** (*Domain-specific Knowledge Propagating Network*) (*Chen et al.*, 2021), de *código abierto*, propone la técnica de *Atención Variacional (VA)*, que basándose en los *Variational Auto-Encoders* (*Kingma and Welling*, 2013) logra explícitamente modelar la distribución de *atención*† para diferentes dominios.

La *atención variacional* presume que teniendo distintos conjuntos de datos $\mathcal{X}^*$, cada uno con una distribución claramente definida y diferente al del resto, cada conjunto se anota con una etiqueta l^*.

Tras procesar todas las imágenes, la red se pueden modelar como un tensor tridimensional x que tiene capas para aprender las *atenciones*† y. Luego sigue la idea de *VAE* (*Kingma and Welling*, 2013), introduciendo una variable de espacio latente z para controlar la distribución $p_\theta(y)$ del dominio, y maximizar su verosimilitud $p_\theta(y|x, l)$.

Como ya se discutió anteriormente, un conjunto puede cubrir distintos dominios y efectivamente hay solapamiento entre conjuntos. Debido a ello, el trabajo agrega la *Atención Variacional Intrínseca (InVA)* que se ocupa de modelar la superposición de de dominios y sub-dominios.

Esta técnica entrena **un primer modelo** sólo con *Atención Variacional (VA)*, y en base a resultados obtenidos con este modelo, aplica un modelo de mezcla de gaussianas con C componentes, para agrupar los datos en C conjuntos. Además, para anotar los k posibles sub-dominios contenidos dentro de un conjunto, cada conjunto se modela nuevamente con una mezcla de gaussianas. Lo cual logra dividir el conjunto combinado de *entrenamiento*† en sub-mezclas de gaussianas.

Emplear este tipo de descomposición del espacio latente explícitamente en la red permite guiar al mecanismo de *atención*† para discriminar caminos y receptores en dicha red de acuerdo a la similitud con el dominio en el que se aplique (*Chen et al.*, 2021). La figura 4.45 muestra los resultados que este trabajo obtuvo en su evaluación. Un diagrama de este modelo se puede apreciar en la figura 4.44.

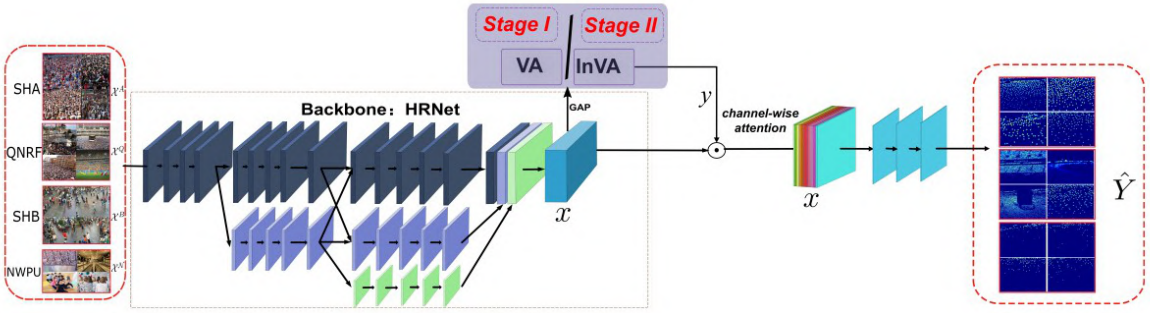


Figura 4.44: DKPNet: Arquitectura (*Chen et al.*, 2021), emplea de *backbone*† el reciente modelo de visión computacional **HRNet-W40** (*Wang et al.*, 2020b)

Methods	SHA[64]		SHB[64]		QNRF[15]		NWPU[54] (V)		NWPU[54] (T)	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Training Dataset: Individual										
CSRNet [21]	68.2	115.0	10.6	16.0	-	-	104.8	433.4	121.3	387.8
CANet [26]	62.3	100.0	7.8	12.2	107.0	183.0	93.5	489.9	106.3	386.5
SFCN [55]	64.8	107.5	7.6	13.0	102.0	171.0	95.4	608.3	105.4	424.1
DSSINet [23]	60.6	96.0	6.9	10.3	99.1	159.2	-	-	-	-
Bayes [32]	62.8	101.8	7.7	12.7	88.7	154.8	93.6	470.3	105.4	454.2
DM-Count [52]	59.7	95.7	7.4	11.8	85.6	148.3	70.5	357.6	88.4	388.6
IT(baseline1)	60.6	99.2	8.8	12.6	97.7	155.7	81.7	516.0	94.0	371.9
Training Dataset: 3-Joint										
JT(baseline2)	60.2	99.6	9.3	13.7	92.8	159.7	-	-	-	-
MB [34]	59.4	101.2	8.3	13.2	91.9	159.6	-	-	-	-
DKPNet (c=3,k=3)	***56.7***	***97.1***	***6.9***	***12.0***	***85.2***	***151.4***	-	-	-	-
Training Dataset: 4-Joint										
JT(baseline2)	59.9	96.7	9.7	15.2	91.1	160.4	73.2	509.5	81.9	351.5
MB [34]	59.2	97.7	8.9	13.4	90.6	157.1	72.7	504.0	80.5	377.8
DKPNet (c=5,k=2)	***55.6***	***91.0***	***6.6***	***10.9***	***81.4***	***147.2***	***61.8***	***438.7***	***74.5***	***327.4***

Figura 4.45: DKPNet: Tabla de resultados (*Chen et al.*, 2021). **DKPNet (c,k)** significa que el modelo usa c dominios en la clusterización, y como máximo k sub-dominios. JT es el resultado de entrenar DKPNet sólo uniendo *conjuntos de entrenamiento*†, (3-Joint usa SHA, SHB, QNRF y 4-Joint agrega NWPU, que está particionado en **V**alidation y **T**est)

DCANet

El trabajo **DCANet** (*Domain guided Channel Attention Network*) (*Yan et al.*, 2021), de *código abierto*, propone un módulo de extracción de atributos CAMD *(channel attention-guided multi-dilation)*, que a través de un mecanismo de *atención*† por canales extrae información del dominio y la refina en una representación vectorial, atributo que se puede emplear en el resto de la red a la hora de construir el *mapa de densidad*†; por ejemplo, para saber que canal de *convoluciones dilatadas*† valorar más.

Este trabajo propone dos técnicas de supervisión para el aprendizaje por dominio, en donde la *atención*† por canales CAMD puede ser explícitamente optimizada para enfatizar canales correspondientes al dominio de la imagen de entrada:

- **DDK** *(dataset-level domain kernel)*, el cual asume que las imágenes de un conjunto reflejan un dominio y que la importancia del canal se puede computar con un modelo pre-entrenado.
- **IDK** *(image-level domain kernel)*, el cual adaptativamente durante el *entrenamiento*† actualiza el dominio inferido y la *atención*† de canal para cada imagen concreta. Esto da un modelado más real del cubrimiento de dominios en un *conjunto de datos*† (y su solapamiento), y además mejora la habilidad del modelo para generalizar en escenas desconocidas.

Dicho trabajo aplica además esta estrategia propuesta al modelo DM-Count (*Wang et al.*, 2020a), modificándolo para incluir el módulo **CADM** y entrenándolo de cero, con la estrategia propuesta de supervisión.

Un diagrama de arquitectura de este modelo se aprecia en la figura 4.46, mientras que los resultados que el estudio obtuvo en su evaluación se muestran en la figura 4.47.

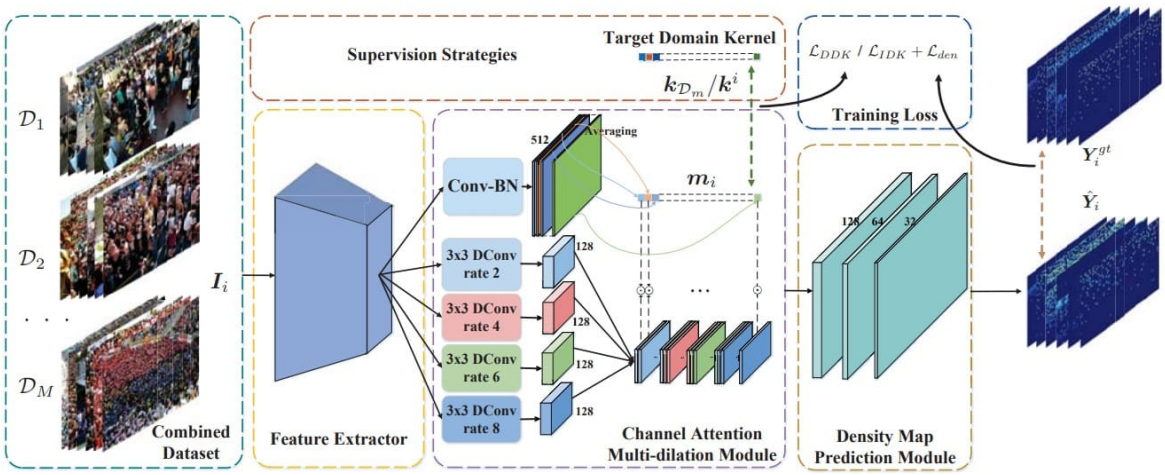


Figura 4.46: DCANet: Arquitectura (*Yan et al.*, 2021)

Method	Observed domains						Unseen domains		
	SHA		SHB		QNRF		UCF_50		WE'10
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Single-domain Method									
CSRNet [35]	68.4	107.2	13.5	18.9	101.1	176.1	343.1	469.8	16.3
CANet [12]	66.4	103.1	12.8	20.6	104.2	170.5	340.9	486.3	15.1
Multi-domain Method									
DGD* [42]	59.2	101.4	8.4	13.5	94.6	167.8	326.3	446.4	14.8
Cross-domain Method									
DSM* [20]	59.8	100.8	8.5	14.3	94.1	166.7	376.3	452.8	19.7
Our Method									
DCANet$_{base}$	62.5	99.3	9.4	14.8	95.9	170.6	330.7	453.5	15.9
DCANet ($\mathcal{L}_D$)	59.0	**99.2**	7.9	12.9	93.8	164.9	345.1	460.5	17.3
DCANet ($\mathcal{L}_I$)	**58.3**	99.3	**7.2**	**11.8**	**88.9**	**160.2**	**309.6**	**431.4**	**12.4**
DM-Count$_{base}$	61.1	101.5	8.4	13.0	84.3	156.3	311.2	422.6	14.7
DM-Count ($\mathcal{L}_D$)	58.2	98.6	7.0	11.1	82.2	147.3	341.5	450.7	17.0
DM-Count ($\mathcal{L}_I$)	**56.8**	**97.4**	**6.1**	**10.3**	**80.4**	**144.1**	**296.6**	**412.5**	**11.5**

Figura 4.47: DCANet: Tabla de resultados con modelos entrenados con el **multidominio** (*Yan et al.*, 2021). $\mathcal{L}_{base}$ es el modelo sin supervisión por dominio. $\mathcal{L}_D$ es re-iterar en el *entrenamiento*† con **DDK** (usando la base), y $\mathcal{L}_I$ re-itera en el *entrenamiento*† con la estrategia **IDK**.

4.11.1.2. Generalización en modelos en contextos desconocidos

El artículo *Domain-general Crowd Counting in Unseen Scenarios* (*Du et al.*, 2022), cuya implementación está disponible en *su repositorio oficial*, se basa en los presentados en la sección anterior (4.11.1.1), pero propone una técnica más robusta para con **un sólo *conjunto de datos*†** mejorar el desempeño de los modelos en **dominios generales (no conocidos)**.

De forma similar a los trabajos anteriores aplica *K-Means*† para tener una división inicial de dominios dentro del *conjunto de datos*†.
Además, el trabajo bifurca la red en dos ramas, que permiten identificar y re-codificar la información **especifica** e **invariante** al dominio de una imagen:

- **DICM** (Domain-Invariant Crowd Memory): En donde un conjunto de vectores se aplica para codificar **todas las imágenes** e inferir un *mapa de densidad*† inicial.
- **DSCM** (Domain-Specific Crowd Memory): En donde varios conjuntos de vectores (uno por sub-dominio) se aplican para **cada imagen de un dominio**, y dicha información complementa la

rama anterior.

Los mismos son respectivamente supervisados por las *funciones de pérdida*† propuestas:

- **Pérdida de reconstrucción de atributos:** supervisa que los atributos re-codificados sean similares a sus valores pre-codificados.
- **Pérdida de atributos ortogonales:** complementa la función anterior, permitiendo que los atributos invariantes al dominio converjan a valores distintos a sus contrapartes específicas al dominio.

La figura 4.48 muestra la arquitectura del modelo. Mientras que la figura 4.49 muestra su evaluación en dominios conocidos (*conjuntos de datos*† entrenados con) y la figura 4.50 muestra su evaluación en dominios desconocidos (*conjuntos de datos*† que no formaron parte del entrenamiento).

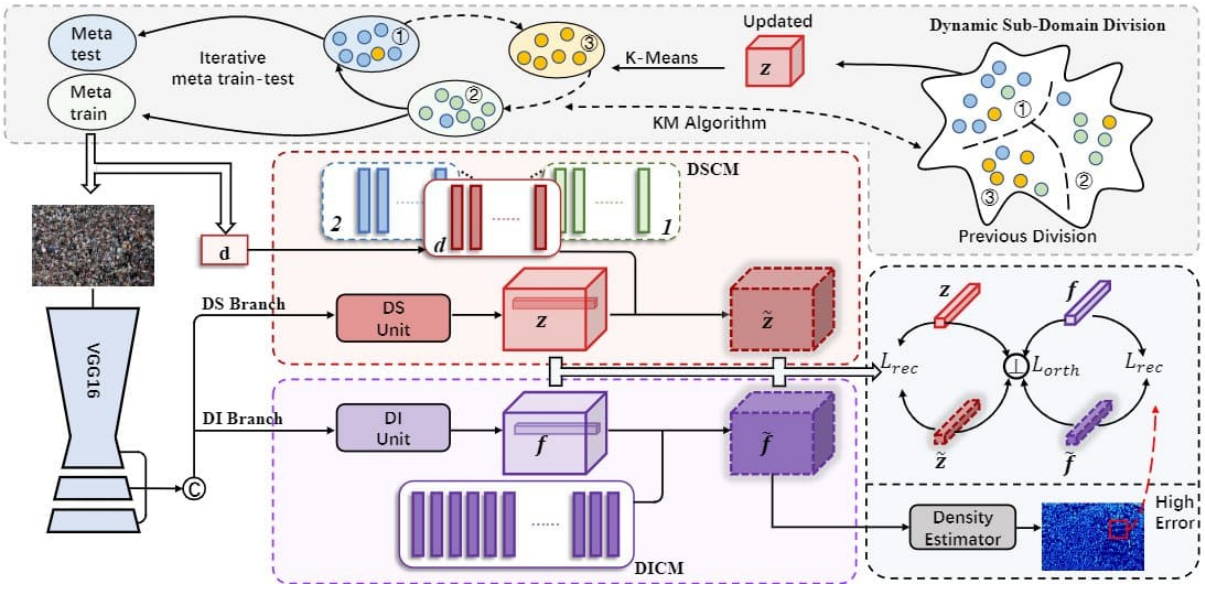


Figura 4.48: Domain-General-CC: Arquitectura (*Du et al.*, 2022)

Dataset	*Source*	SHA				SHB			
	Target	SHB		QNRF		SHA		QNRF	
Method	*Adaptation*	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
MCNN (Zhang et al. 2016)	✗	85.2	142.3	–	–	221.4	357.8	–	–
DSSINet (Liu et al. 2019a)	✗	21.7	37.6	198.7	329.4	148.9	273.9	267.3	477.6
BL (Ma et al. 2019)	✗	15.9	25.8	166.7	287.6	138.1	228.1	226.4	411.0
DMCount (Wang et al. 2020a)	✗	23.1	34.9	134.4	252.1	143.9	239.6	203.0	386.1
D2CNet (Cheng et al. 2021)	✗	21.6	34.6	126.8	245.5	164.5	286.4	267.5	486.0
SASNet (Song et al. 2021b)	✗	21.3	33.2	211.2	418.6	132.4	225.6	273.5	481.3
MAN (Lin et al. 2022)	✗	22.1	32.8	138.8	266.3	133.6	255.6	209.4	378.8
DG-MAN (Mansilla et al. 2021)	✗	17.3	28.7	129.1	238.2	130.7	225.1	182.4	325.8
Ours	✗	12.6	24.6	119.4	216.6	121.8	203.1	179.1	316.2
Cycle GAN (Zhu et al. 2017)	✓	25.4	39.7	257.3	400.6	143.3	204.3	257.3	400.6
SE CycleGAN (Wang et al. 2019)	✓	19.9	28.3	230.4	384.5	123.4	193.4	230.4	384.5
SE+FD (Han et al. 2020)	✓	16.9	24.7	221.2	390.2	129.3	187.6	221.2	390.2
RBT (Liu et al. 2020)	✓	13.4	29.3	175.0	294.8	112.2	218.2	211.3	381.9
C^2MoT (Wu, Wan, and Chan 2021)	✓	12.4	21.1	125.7	218.3	120.7	192.0	198.9	368.0

Figura 4.49: Domain-General-CC: Tabla de resultados, los modelos sin adaptación fueron entrenados sólo con la fuente, mientras que los otros tuvieron un re-*entrenamiento*† para adaptarlos al conjunto destino (*Du et al.*, 2022)

Source	SHA		SHB		SHA+SHB	
Target	NWPU					
Method	MAE	MSE	MAE	MSE	MAE	MSE
DMCount (Wang et al. 2020a)	146.9	**563.8**	191.6	747.4	144.6	592.8
SASNet (Song et al. 2021b)	158.8	588.0	195.7	716.8	155.3	583.6
MAN (Lin et al. 2022)	148.2	586.5	193.6	802.5	147.8	605.3
Ours	**143.1**	567.6	**175.0**	**688.6**	**139.6**	**553.6**

Figura 4.50: Domain-General-CC: Tabla de resultados en dominios desconocidos. (*Du et al.*, 2022)

4.11.2. Transformers

Los *transformers*† (*Vaswani et al.*, 2017), con sus mecanismos de auto-atención, han estado floreciendo e incrementando enormemente el desempeño del estado del arte en tareas de procesamiento del lenguaje natural (NLP). Luego el modelo ViT generaliza esta idea (*Dosovitskiy et al.*, 2020) a imágenes, trabajando con las mismas en parches, lo cual permite la aplicación de los mismos a mejorar el desempeño en áreas de la visión artificial. Este éxito sugiere que los mecanismos de auto-atención y de un sesgo inductivo local son importantes para tareas de visión, y varios trabajos los han aplicado con éxito en tareas de reconocimiento de imágenes (*Touvron et al.*, 2021), detección de objetos (*Carion et al.*, 2020; *Zhu et al.*, 2021b) y segmentación en imágenes (*Ye et al.*, 2019). (*Lin et al.*, 2022; *Khan et al.*, 2022)

El estudio y la aplicabilidad de los *transformers*† al área de visión artificial aún en su etapa preliminar, con nuevas arquitecturas y módulos surgiendo constantemente (*Liu et al.*, 2021b; *Zhang et al.*, 2021b). Un problema en particular es que estos modelos usan un tamaño fijo en su núcleo de *atención*†, lo cual no es ideal para las grandes variaciones de escala (y por ende de densidad de individuos) que pueden tener los parches de una imagen en el conteo de multitudes (*Lin et al.*, 2022).

Trabajos recientes han aplicado el uso de *transformers*† al conteo de multitudes (*Ranjan et al.*, 2019; *Gao et al.*, 2022b; *Tian et al.*, 2021; *Sun et al.*, 2021; *Do*, 2021; *Liang et al.*, 2022b; *Wei et al.*, 2021; *Liang et al.*, 2022a; *Lin et al.*, 2022).

Si bien la limitación de tener que partir la imagen en parches para generar una secuencia (y el cómo lograrlo sin perder la coherencia espacial de la imagen) son problemas de dicha arquitectura que estos trabajos aún están explorando cómo resolver, todos estos modelos han tenido un desempeño destacable, y han llegado a igualar o superar mínimamente al estado del arte evaluado en algunos *conjuntos de datos*†, por lo que son un área promisoria.

4.11.3. Aprendizaje débilmente supervisado

Si bien no cuentan con un desempeño tan bueno como los modelos basados en *aprendizaje supervisado*† que se han presentado en las secciones anteriores, un pequeño número de trabajos (*Lei et al.*, 2021; *Liang et al.*, 2022a; *Wang et al.*, 2019; *Liu et al.*, 2019d) han investigado cómo resolver el problema de conteo contando con anotaciones más sencillas (como el número de individuos en la imagen).

El trabajo **MATT** (*Lei et al.*, 2021), por ejemplo, usa un enfoque de *aprendizaje débilmente supervisado*† con *mapas de densidad*† para una pequeña cantidad de instancias, y luego itera en el *entrenamiento*† del modelo con anotaciones de conteo (supervisión débil).

Los *transformers*† también han sido aplicados en esta área, dado su mecanismo de auto-atención, que permite mayor facilidad para aprender información semántica. Un ejemplo de esto es el modelo TransCrowd (*Liang et al.*, 2022a) basado en ViT (*Dosovitskiy et al.*, 2020) (*Khan et al.*, 2022).

Capítulo 5

Estudio sobre la calidad de anotaciones

Resulta de interés para el problema de conteo de multitudes a través de *aprendizaje automático*†, tener en cuenta la calidad y el posible ruido en los *conjuntos de datos*†. Al momento de escribir este trabajo, y dentro del alcance de investigación del mismo, no se pudo encontrar ningún artículo que realice una *evaluación*† sobre la calidad de anotación en los *conjuntos de datos*† disponibles. El artículo (*Gao et al.*, 2020) comenta que se han encontrado muestras de *UCFCC-50* (*Idrees et al.*, 2013) y *ShanghaiTech A* (*Zhang et al.*, 2016b) con errores en su anotación, pero no cita fuentes ni ejemplos.

Todo *conjuntos de datos*† contendrá cierto grado de ruido en sus anotaciones; el error humano es un factor inevitable en este campo. Sin embargo, tareas como el conteo en una multitud muy densa (y/o con particularidades que afecten la calidad de la imagen) son particularmente difíciles de realizar para los seres humanos.

En este estudio se decidió por explorar el proceso de etiquetado, creando un nuevo *conjunto de datos*† (denominado **MT** de ahora en adelante), y organizando un módulo de taller para que estudiantes de ingeniería en computación contribuyan al etiquetado del mismo, el estudiante que realizó este trabajo de grado formó parte de los anotadores del módulo de taller.

El *conjunto de datos*†, así como el código diseñado para observar y comparar las anotaciones se encuentra accesible en el *repositorio público de este trabajo* (`https://github.com/renzodgc/fing-crowdcounting`).

5.1. Conjunto de datos Módulo de Taller

Las imágenes que componen el dataset **MT** fueron compiladas por uno de los tutores de este trabajo, y provienen de las siguientes fuentes:

- Fotografías capturadas por el escritor y fotógrafo Jorge Fierro (quién consintió a su uso para este trabajo)
- Fotografías descargadas de la página de *fotos de prensa de la página web de la intendencia de Montevideo*.

En total el conjunto se compone de 56 imágenes, promediando acorde a los datos anotados 953,9 personas por imagen; la desviación estándar es de 1848,1, por lo que la densidad varía mucho entre imágenes.

Para focalizar mejor el análisis para los varios dominios del *conjunto de datos*†, se divide el mismo en cuatro subconjuntos:

- **Paraguas:** Conjunto de 6 imágenes formado por fotos de multitudes bajo la lluvia, en donde hay un gran porcentaje de *oclusión*† debido a los paraguas. Promediando entre los anotadores 93,4 personas por instancia con una desviación estándar de 71,2. La figura 5.1 muestra un ejemplo de este subconjunto.
- **Baja densidad:** Conjunto de 13 imágenes formado por fotos de grupos de personas de densidad particularmente baja; en general todas las personas presenten están en una escala muy grande (cercanas a la cámara). Promediando entre los anotadores 38,3 personas por instancia con una desviación estándar de 17,9. La figura 5.2 muestra un ejemplo de este subconjunto.
- **Densidad media:** Conjunto de 26 imágenes formado por fotos de multitudes, de densidad variable. Promediando entre los anotadores 217,4 personas por instancia con una desviación estándar de 229,6. La figura 5.3 muestra un ejemplo de este subconjunto.
- **Aéreos:** Conjunto de 12 imágenes formado por fotos de multitudes, de muy alta densidad, y que debido al ángulo de la cámara incluyen en el punto de fuga un enorme número de personas. Promediando entre los anotadores 3971,7 personas por instancia con una desviación estándar de 2132,1. La figura 5.4 muestra un ejemplo de este subconjunto.

Figura 5.1: **MT aj04**: Paraguas

Figura 5.2: **MT am01**: Densidad baja

Figura 5.3: **MT bm09**: Densidad media

Figura 5.4: **MT cm04**: aéreo

5.1.1. Dificultad de las imágenes

Si bien el conjunto presenta varias imágenes de buena calidad, nitidez y facilidad para anotar (por ejemplo las figuras 5.5, 5.6 y 5.7), también presenta varias otras imágenes de particular dificultad, como por ejemplo las figuras 5.8, 5.9, 5.10 y 5.11:

Figura 5.5: **MT aj14**: Densidad baja

Figura 5.6: **MT bm01**: Densidad media

Figura 5.7: **MT bm02**: Densidad media

Figura 5.8: **MT aj07 dificultad**: Densidad media, desenfocada

Figura 5.9: **MT bj02 dificultad**: Paraguas, *oclusiones*†

Figura 5.10: **MT cm01 dificultad**: Aérea, iluminación

Figura 5.11: **MT cm05 dificultad**: Aérea, perspectiva

5.2. Metodología de anotación

Para anotar el conjunto **MT** se empleó la herramienta ***CCLabeler*** (*Wang et al.*, 2020c), presentada en la sección 3.2. Se contó con un total de 12 anotadores, quienes anotaron independientemente cada imagen del conjunto.

Durante el Módulo de Taller se siguió las siguiente metodología de etiquetado:

- En el caso de una persona con su cabeza visible se etiqueta con un punto su cabeza.
- En el caso de una persona con parte de su cuerpo visible, pero cabeza ocluida, sea por estar fuera de la foto, u ocluida por otro objeto. Se etiqueta el punto de su cuerpo visible más cercano a la cabeza.
- En el caso de personas completamente ocluidas (no visibles en la imagen), que se encuentren rodeadas por una multitud de densidad aparentemente uniforme, se asume que efectivamente lo es y se etiquetan puntos de la misma detrás del obstáculo. Por ejemplo en la multitud de la figura 5.13.
- En el caso de personas completamente ocluidas (no visibles en la imagen), que no se encuentren dentro de una multitud uniforme, pero que sea posible inferir una estimación conservadora de su presencia (por ejemplo un auto en movimiento), se anota un individuo Por ejemplo los paraguas en la figura 5.12.

Figura 5.12: **MT aj01**: *Oclusión*†, se anota una persona por paraguas sin portador visible.

Figura 5.13: **MT bm07**: *Oclusión*† en densidad uniforme, se anotan personas atrás de la estatua (manteniendo la distribución).

5.3. Resultados obtenidos

En el etiquetado del conjunto **MT** se obtuvieron las anotaciones que se presentan a continuación.

5.3.1. MT paraguas

Se aprecian las anotaciones obtenidas en la tabla 5.1 y se muestra su distribución en la figura 5.14.

Imagen													Media (std)
aj01	**27**	35	**28**	**44**	**29**	30	**41**	39	35	38	39	35	35.0 (5.3)
aj02	53	47	**37**	**58**	44	**35**	**62**	47	50	52	**61**	41	48.9 (8.5)
aj03	52	52	**41**	63	47	**45**	64	51	**69**	**81**	60	55	56.7 (10.8)
aj04	76	**60**	**60**	77	63	63	**78**	**56**	**79**	**78**	64	77	69.2 (8.5)
bj01	**150**	**139**	**89**	130	**76**	103	130	121	114	**83**	125	125	115.4 (22.1)
bj02	**284**	**290**	208	273	**180**	210	**293**	**188**	253	**191**	252	203	235.4 (41.2)

Tabla 5.1: Cantidad de anotaciones en **MT paraguas**: En **negrita** valores fuera del rango de desviación estándar

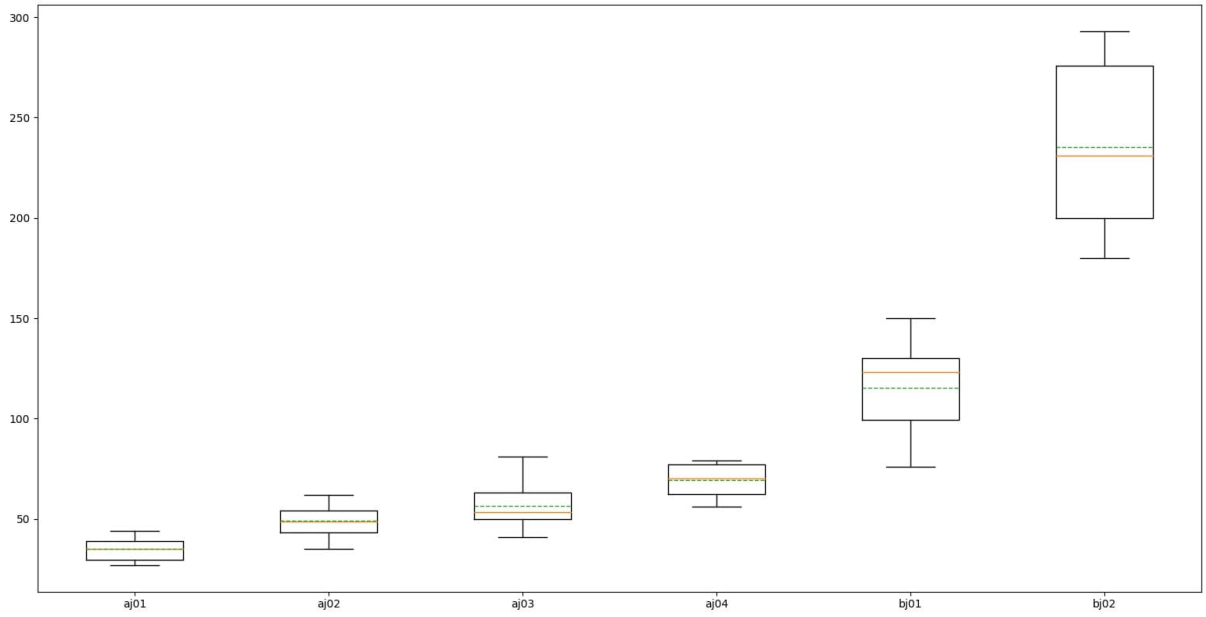


Figura 5.14: Diagrama de caja: **MT paraguas**. La línea punteada es la media, la línea naranja es la mediana, las cajas delimitan el segundo y tercer cuartil, las líneas verticales el primer y cuarto cuartil. Los puntos por fuera son valores atípicos

5.3.2. MT densidad baja

Se aprecian las anotaciones obtenidas en la tabla 5.2 y se muestra su distribución en la figura 5.15.

Imagen													Media (std)
aj06	30	**35**	**26**	33	31	**26**	30	28	30	**23**	**35**	31	29.8 (3.5)
aj08	**40**	**58**	**37**	45	45	47	**57**	**40**	53	43	**59**	48	47.7 (7.2)
aj09	35	44	**31**	43	36	39	44	38	40	37	**51**	40	39.8 (5.0)
aj10	**41**	46	**37**	50	47	47	**52**	47	44	50	**56**	46	46.9 (4.8)
aj11	**61**	**111**	**56**	77	**62**	75	86	68	78	**95**	87	71	77.2 (15.0)
aj12	**20**	30	**20**	**36**	22	30	**33**	26	27	24	27	23	26.5 (4.8)
aj13	31	**35**	**27**	33	30	32	**38**	33	29	30	30	**28**	31.3 (3.0)
aj14	23	26	23	26	24	26	**28**	**21**	26	**21**	23	22	24.1 (2.2)
aj15	53	**61**	51	**63**	49	49	**61**	**46**	51	**43**	55	48	52.5 (6.1)
aj16	40	**53**	41	45	39	39	**51**	**36**	47	**35**	**52**	**36**	42.8 (6.3)
aj17	**19**	17	15	15	**14**	**13**	**18**	15	16	16	17	16	15.9 (1.6)
am01	13	13	12	**11**	13	12	13	12	12	12	13	**10**	12.2 (0.9)
am04	51	52	50	50	**46**	51	52	49	**53**	51	52	51	50.7 (1.7)

Tabla 5.2: Cantidad de anotaciones en **MT densidad baja**: En **negrita** valores fuera del rango de desviación estándar

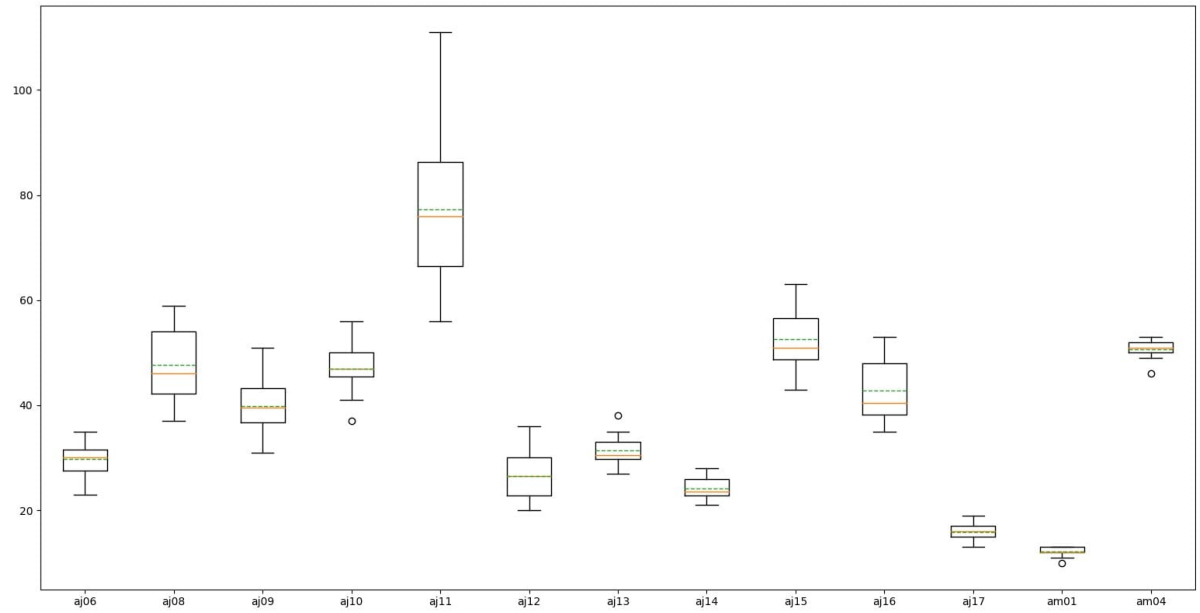


Figura 5.15: Diagrama de caja: **MT densidad baja**. La línea punteada es la media, la línea naranja es la mediana, las cajas delimitan el segundo y tercer cuartil, las líneas verticales el primer y cuarto cuartil. Los puntos por fuera son valores atípicos

5.3.3. MT densidad media

Se aprecian las anotaciones obtenidas en la tabla 5.3 y se muestra su distribución en la figura 5.16.

Imagen													Media (std)
aj05	100	**113**	**88**	97	96	91	97	**89**	101	91	**115**	99	98.1 (8.2)
aj07	77	**111**	72	80	69	73	69	77	81	**101**	**118**	81	84.1 (15.9)
am02	**93**	94	**93**	**95**	94	94	94	94	**93**	94	94	94	93.8 (0.6)
am03	47	54	53	54	**40**	43	52	**37**	48	55	**60**	49	49.3 (6.4)
am05	85	82	81	**87**	79	82	**87**	**74**	**87**	**77**	**88**	84	82.8 (4.2)
am06	76	75	73	75	**68**	72	75	**70**	74	75	76	75	73.7 (2.4)
am07	45	**46**	41	44	**39**	41	**46**	**37**	42	43	**46**	44	42.8 (2.8)
bj03	76	**161**	79	92	**63**	81	**127**	82	97	96	74	76	92.0 (26.0)
bj04	**50**	**53**	44	43	43	48	45	**42**	45	**51**	49	**42**	46.2 (3.6)
bj05	99	**139**	**84**	102	95	109	103	**76**	105	87	111	88	99.8 (15.6)
bj06	54	**71**	54	59	55	59	59	**44**	60	60	**67**	54	58.0 (6.5)
bj07	85	**129**	85	94	**69**	95	97	98	105	**62**	106	79	92.0 (17.1)
bj08	58	**77**	49	**67**	**48**	51	**69**	**48**	61	53	60	52	57.8 (9.0)
bj09	**134**	**137**	**113**	117	130	122	124	**109**	**141**	116	130	**112**	123.8 (10.1)
bj10	86	**93**	77	79	78	77	**88**	**74**	86	77	87	76	81.5 (5.9)
bj11	81	83	69	72	**66**	77	82	**64**	**90**	70	**84**	71	75.8 (7.9)
bj12	216	**255**	227	217	206	206	236	207	222	213	**243**	**200**	220.7 (16.1)
bm01	356	**400**	**307**	339	341	338	353	323	317	**306**	353	351	340.3 (24.8)
bm02	696	788	772	**603**	**654**	726	**859**	661	790	689	**915**	734	740.6 (85.5)
bm04	**285**	262	**228**	248	246	264	259	**305**	273	259	283	262	264.5 (19.4)
bm07	362	**427**	336	**320**	349	331	**403**	342	**404**	363	357	340	361.2 (31.8)
bm08	876	1067	863	**792**	839	**791**	**1292**	962	**1120**	949	906	839	941.3 (144.0)
bm09	348	355	333	**296**	324	**288**	348	330	**374**	**379**	**369**	318	338.5 (27.9)
bm10	379	434	347	369	443	373	**928**	458	**767**	474	417	358	478.9 (172.6)
bm12	203	296	261	260	197	270	**435**	262	**344**	214	210	**164**	259.7 (70.8)
bm13	380	535	460	415	**366**	510	**680**	388	510	372	417	437	455.8 (87.3)

Tabla 5.3: Cantidad de anotaciones en **MT densidad media**: En **negrita** valores fuera del rango de desviación estándar

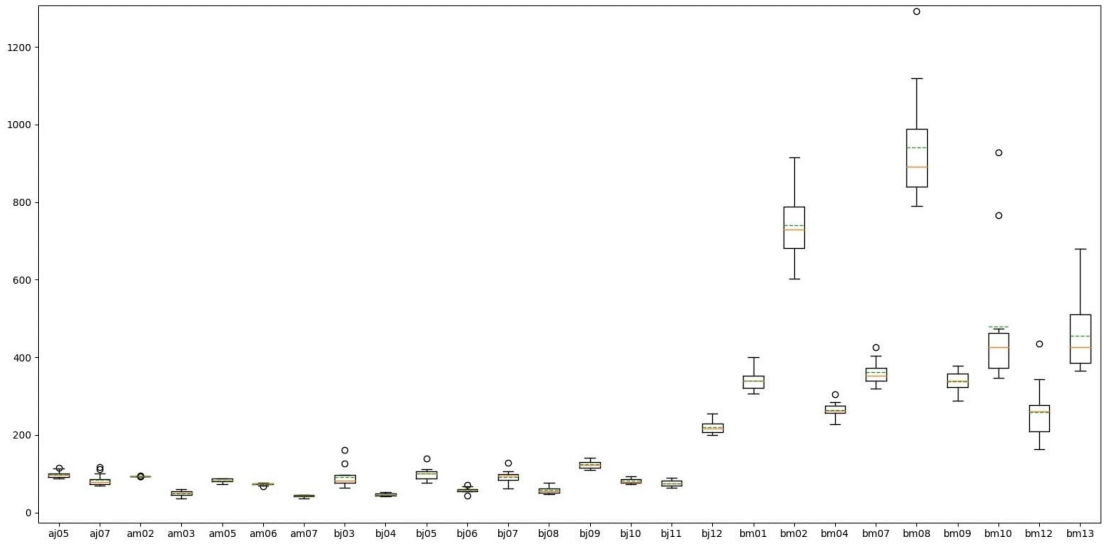


Figura 5.16: Diagrama de caja: **MT densidad media**. La línea punteada es la media, la línea naranja es la mediana, las cajas delimitan el segundo y tercer cuartil, las líneas verticales el primer y cuarto cuartil. Los puntos por fuera son valores atípicos

5.3.4. MT aéreo

Se aprecian las anotaciones obtenidas en la tabla 5.4 y se muestra su distribución en la figura 5.17.

Se destaca que por un error en la selección de imágenes del módulo de taller, la imagen *bm*03 y *cm*03 son la misma imagen. Todos los anotadores las anotaron una segunda vez, excepto por tres anotadores que no anotaron una de estas, para la realización de este estudió se copió su respectiva anotación de la otra imagen.

Si bien su repetición fue un error, es interesante observar como varían significativamente las anotaciones de los 9 anotadores que etiquetaron ambas imágenes.

Imagen													Media (std)
bm03	5778	7967	**4038**	**9592**	6491	6034	**10261**	6318	8763	**2245**	**3344**	8680	6625.9 (2422.8)
bm05	**1385**	**1833**	1467	1409	1441	**1750**	1626	**1358**	1641	**1367**	1638	**1767**	1556.8 (164.5)
bm06	2157	2440	1947	2522	2015	2042	**2844**	**1431**	2680	2011	1834	**3240**	2263.6 (476.6)
bm11	3444	6697	3541	**2470**	3239	3462	**8778**	3863	**7133**	3612	3055	**7721**	4751.2 (2077.3)
cm01	1807	1824	1753	1608	**1393**	1640	**2178**	**2695**	2044	1597	1708	1675	1826.8 (328.0)
cm02	2137	2468	1975	2160	2403	2043	**5194**	2510	**3633**	2197	2763	3381	2738.7 (891.5)
cm03	5778	3833	3984	**2866**	6491	4538	**7096**	3453	**8763**	**3020**	3902	**8060**	5148.7 (1940.0)
cm04	**3201**	4897	4135	3680	**5320**	3704	**5682**	4131	**6156**	3646	3844	4678	4422.8 (882.8)
cm05	5575	7064	4290	**3087**	5456	3805	6790	4682	**8827**	**3454**	3694	**8483**	5433.9 (1877.7)
cm06	**8179**	5260	4092	**3266**	4779	**3300**	**8623**	4474	7028	5473	4230	6025	5394.1 (1694.7)
cm07	3297	3232	**2144**	**2283**	3080	2873	**6645**	3901	4115	3041	2946	**6009**	3630.5 (1324.2)
cm08	2696	2449	3009	**2310**	3471	3161	**6609**	4203	4453	3247	3815	**6991**	3867.8 (1452.6)

Tabla 5.4: Cantidad de anotaciones en **MT aéreo**: En **negrita** valores fuera del rango de desviación estándar

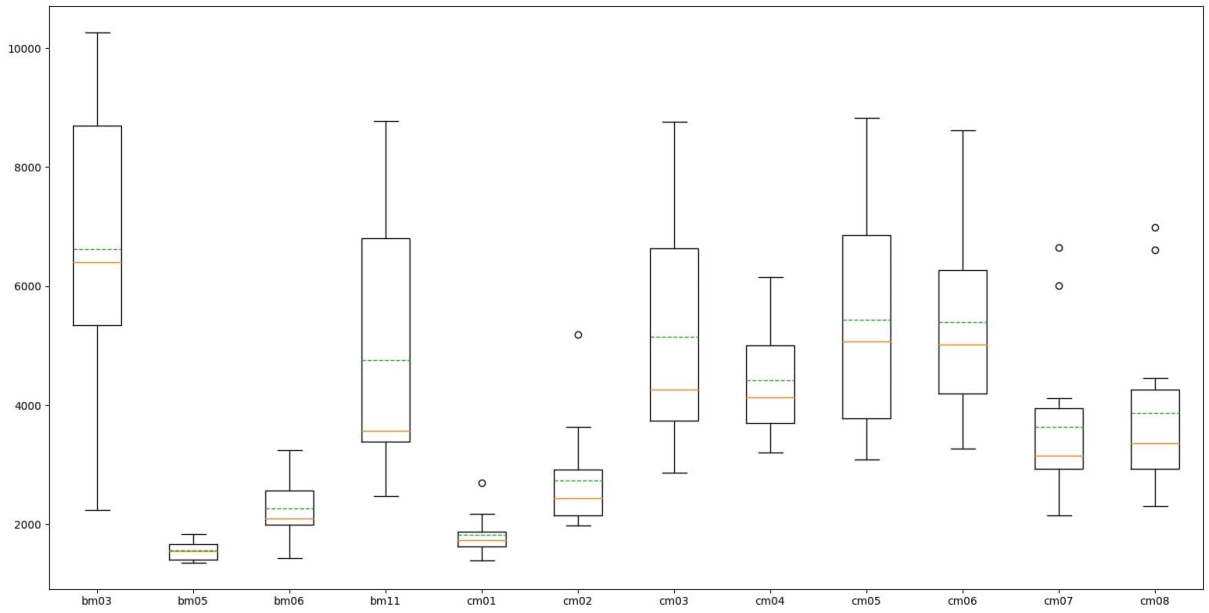


Figura 5.17: Diagrama de caja: **MT aéreo**. La línea punteada es la media, la línea naranja es la mediana, las cajas delimitan el segundo y tercer cuartil, las líneas verticales el primer y cuarto cuartil. Los puntos por fuera son valores atípicos

5.4. Análisis de resultados

En esta sección se analizan los resultados presentados en la sección anterior. En la sección 5.4.1 se visualizan las imágenes de menor y mayor varianza para cada subconjunto. Luego en la sección 5.4.2 se enumeran datos y observaciones en los resultados obtenidos.

5.4.1. Visualización de resultados

A continuación se presentan las imágenes de menor (sección 5.4.1.1) y de mayor (sección 5.4.1.2) varianza entre anotadores.

5.4.1.1. Resultados de baja varianza

Se destacan las siguientes imágenes cuyas anotaciones variaron poco entre anotadores. Se sitúa **para cada conjunto** a la **izquierda** la de menor varianza (figuras 5.18, 5.20, 5.22 y 5.24), y a la **derecha** la segunda imagen de menor varianza (figuras 5.19, 5.21, 5.23 y 5.25).

Figura 5.18: **MT paraguas aj01**: Menor varianza

Figura 5.19: **MT paraguas aj02**: Segunda menor varianza

Figura 5.20: **MT densidad baja am01**: Menor varianza

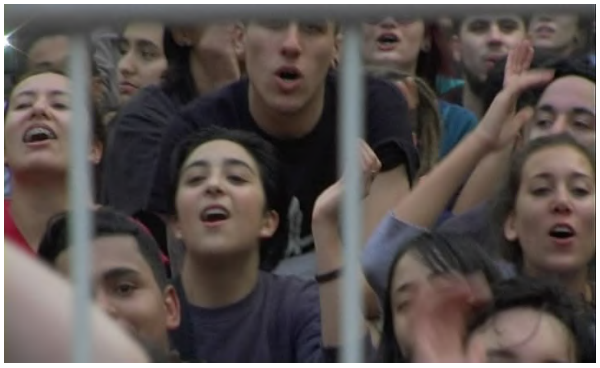


Figura 5.21: **MT densidad baja am04**: Segunda menor varianza

Figura 5.22: **MT densidad media am02**: Menor varianza

Figura 5.23: **MT densidad media bj04**: Segunda menor varianza

Figura 5.24: **MT aéreo bm05**: Menor varianza

Figura 5.25: **MT aéreo cm01**: Segunda menor varianza

5.4.1.2. Resultados de alta varianza

Se destacan las siguientes imágenes cuyas anotaciones variaron sustancialmente entre anotadores. Se sitúa **para cada conjunto** a la **izquierda** la de mayor varianza (figuras 5.26, 5.28, 5.30 y 5.32), y a la **derecha** la segunda imagen de mayor varianza (figuras 5.27, 5.29, 5.31 y 5.33).

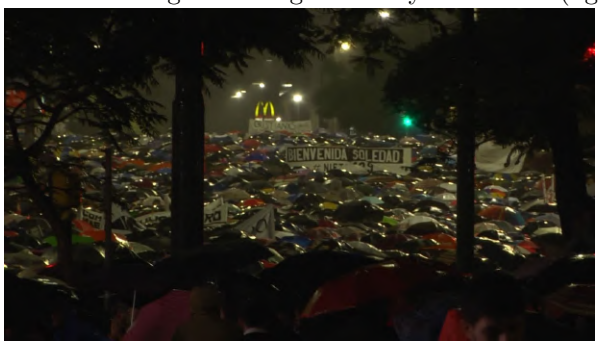


Figura 5.26: **MT paraguas bj02**: Mayor varianza

Figura 5.27: **MT paraguas bj01**: Segunda mayor varianza

Figura 5.28: **MT densidad baja aj11**: Mayor varianza

Figura 5.29: **MT densidad baja aj08**: Segunda mayor varianza

Figura 5.30: **MT densidad media bm08**: Mayor varianza

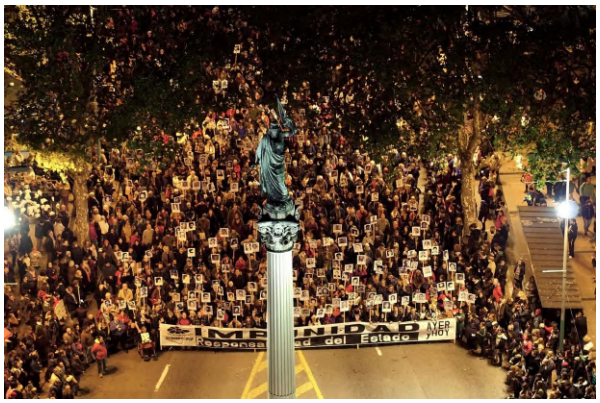


Figura 5.31: **MT densidad media bm13**: Segunda mayor varianza

Figura 5.32: **MT aéreo bm03/cm03**: Mayor varianza

Figura 5.33: **MT aéreo bm11**: Segunda mayor varianza

5.4.2. Observaciones e interpretaciones

Observando estos resultados, se realizan las siguientes observaciones a nivel general:

- En ninguna imagen del *conjunto de datos*† los 12 anotadores coincidieron en la cantidad de personas anotadas.
- Salvo por las tres anotaciones que no anotaron **cm03** y se copió su resultado de **bm03**, toda otra anotación de **bm03** discrepó significativamente de las anotaciones de **cm03**.
- Para cada subconjunto, las anotaciones más seguras (con menor desviación estándar) son de:
  - MT paraguas: **aj01** ($\mu = 35{,}0$, $\sigma = 5{,}3$) con tres anotadores contando 35 personas, y los otros contando cantidades distintas que varían entre 27 y 41.
  - MT densidad baja: **am01** ($\mu = 12{,}2$, $\sigma = 0{,}9$) con cinco anotadores contando 12 personas, cinco contando 13, uno contando 11 y otro contando 10.
  - MT densidad media: **am02** ($\mu = 93{,}8$, $\sigma = 0{,}6$) con ocho anotadores contando 94 personas, dos contando 93 y uno contando 95.
  - MT aéreo: **bm05** ($\mu = 1556{,}8$, $\sigma = 164{,}5$) con la anotación más cercana a la media habiendo contado 69 personas más que la media.
- Para cada subconjunto, las anotaciones menos seguras (con mayor desviación estándar) son de:
  - MT paraguas: **bj02** ($\mu = 235{,}4$, $\sigma = 41{,}2$)
  - MT densidad baja: **aj11** ($\mu = 77{,}2$, $\sigma = 15{,}0$)
  - MT densidad media: **bm10** ($\mu = 478{,}9$, $\sigma = 172{,}6$)
  - MT aéreo: **bm03** ($\mu = 1556{,}8$, $\sigma = 2422{,}8$). Si se combinan las anotaciones con su versión duplicada (**cm03**), estos valores son $\mu = 5887{,}3$ y $\sigma = 2315{,}6$
- La imagen con segunda menor varianza para el conjunto MT aéreo (**cm01**) es una de las que se consideraban de posible dificultad (debido a la iluminación de la imagen).
- Para cada subconjunto del *conjunto de datos*†, el rango de variación porcentual de desviación estándar respecto a la media ($100 * \frac{\sigma}{\mu}$), entre las 12 anotaciones de cada imagen, es de:
  - MT paraguas: 12,3 % (**aj04**) − 19,1 % (**bj01**)
  - MT densidad baja: 3,5 % (**am04**) − 19,4 % (**aj11**)
  - MT densidad media: 0,6 % (**am02**) − 36,0 % (**bm10**)
  - MT aéreo: 10,6 % (**bm05**) − 43,7 % (**bm11**)
- Para cada subconjunto del *conjunto de datos*†, la cantidad de imágenes cuya variación porcentual de desviación estándar respecto a la media ($100 * \frac{\sigma}{\mu}$) es menor al 15 %, 10 % y 5 % es de:
  - MT paraguas (6 imágenes): 2 imágenes con variación ≥ 15 %.
  - MT densidad baja (13 imágenes): 11 imágenes con variación $\geq 15{,}1$ %, 4 imágenes con variación $\geq 10{,}1$ %, 1 imagen con variación ≥ 5 %.
  - MT densidad media (26 imágenes): 17 imágenes con variación ≥ 15 %, 13 imágenes con variación ≥ 10 %, 3 imágenes con variación $\geq 5{,}1$ %.
  - MT aéreo (12 imágenes): 1 imagen con variación ≥ 15 %.

En la figura 5.34 se aprecian las anotaciones del estudiante con mayor diferencia en cantidad de personas al anotar **MT aéreo bm03** y **MT aéreo cm03**. Si bien es claro que ninguna de estas anotaciones es correcta, también es importante visualizar la gran dificultad en etiquetar esta imagen, dado que debido a su gran *variación de escala*† y fondo borroso, es muy difícil anotar a las personas en ella.

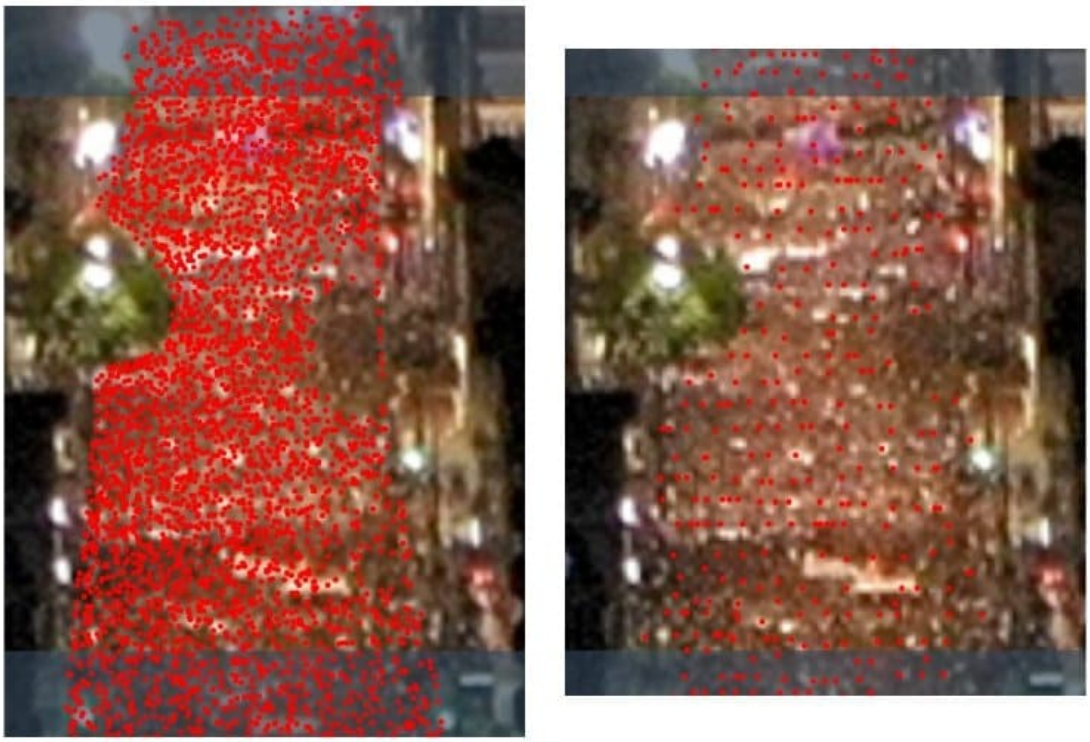


Figura 5.34: Anotaciones un mismo estudiante mayor diferencia en el fondo de la imagen duplicada **MT aéreo bm03** (2866 anotaciones) y **MT aéreo cm03** (9592 anotaciones).

A partir de estos datos y observaciones, se extraen las siguientes conclusiones:

- En general, la tarea de anotación para el problema del conteo de multitudes es de gran dificultad; Incluso en las imágenes de buena calidad, nitidez y claridad en la posición de cada persona en el conjunto **MT** (tales como **am01**, **am04**, **aj17** y **am02**) no se cumplió que las anotaciones coincidan en la cantidad de personas anotadas.
- El conjunto **MT** es particularmente difícil de anotar, conteniendo imágenes en las que, debido a las dificultades de las mismas, la desviación estándar es de una muy gran proporción respecto a la media ($\mu = 4751{,}2, \sigma = 2077{,}3$) siendo **bm11** la imagen de mayor dificultad para anotar.
- Al comparar anotaciones discrepantes de las imágenes duplicadas **bm03** y **cm03**, realza la no trivialidad del dominio. En varias imágenes del conjunto **MT aéreo** la anotación de su fondo termina siendo una tarea de estimación sujeta a la interpretación espacial del anotador.
- En general para el conjunto **MT**, cuanta mayor densidad y oclusión, y menor nitidez en las imágenes, mayor es la proporción de varianza entre anotaciones de la misma.
- Los conjuntos más difíciles de anotar fueron **MT paraguas** y **MT aéreo**, lo cual coincide con la presuposición de dificultad en el conjunto **MT** dado que estas eran las imágenes con mayor *oclusión*† y mayor *variación de escala*†.
- Salvo algunas imágenes particularmente difíciles, los subconjuntos **MT densidad baja** y **MT densidad media** tuvieron anotaciones con una varianza relativamente tolerable respecto a la cantidad de personas anotadas.

5.5. Reflexiones sobre la metodología del estudio

En la selección de imágenes y metodología de etiquetado se tomaron ciertas decisiones que en retrospectiva no fueron las más eficaces para abordar este estudio dentro del alcance del proyecto; en particular, con el fin de poder realizar observaciones que se asemejen a interpretaciones de los otros *conjuntos de datos*† existentes. También hubiera sido deseable analizar anotaciones en conjuntos de datos comúnmente utilizados, pero dicho análisis quedó por fuera del alcance de este proyecto.

Si bien es importante ilustrar y analizar las carencias y problemas que tiene el conjunto generado, también es de importancia destacar que aún así, este recurso aporta valor al trabajo, dado que es posible realizar observaciones específicas a la anotación de datos de este conjunto. Así es como en la sección 6 se presenta el desempeño de modelos pre-entrenados con este nuevo *conjunto de evaluación*†.

5.5.1. Observaciones sobre la metodología de etiquetado adoptada

Como se define en la sección 5.2, este trabajo se adoptó una metodología de etiquetado cuyas pautas buscaban obtener una anotación de personas cercana a la realidad. Anotando no sólo cabezas de personas, sino también anotando personas parcialmente ocluidas (como por un cartel, parcialmente fuera de la imagen), personas totalmente ocluidas (como por su paraguas y una multitud, o estando dentro de un auto en movimiento), y múltiples personas totalmente ocluidas (como un obstáculo tapando parte de una multitud uniforme).

Dicha metodología contrasta con la generalmente usada en el resto de *conjuntos de datos*† en la literatura, que se limita a solamente anotar cabezas visibles en la imagen. Pauta que termina generando una anotación conservadora pero no necesariamente real (anota un mínimo de personas que realmente están en la escena). Sin embargo, esta metodología de anotación también posee el beneficio de ser menos subjetiva al criterio del anotador. Que al seguir un patrón en una multitud ocluida puede también estar anotando personas de más (que tal vez no se encuentran detrás de ese obstáculo).

También estas anotaciones conservadoras en los *conjuntos de entrenamiento*† ocasiona que los modelos de conteo de multitudes sean entrenados para contar cabezas, esto se apreciará claramente en la visualización de estimaciones por modelos en la sección 6.2.4.4.

Se observa que debido a la dificultad intrínseca del conjunto **MT** (principalmente para los subconjuntos de mayor dificultad como **MT Aéreo** y **MT Paraguas**) y el criterio de anotación que fue elegido, la anotación del conjunto no es una tarea trivial, y esto se refleja en las incongruencias, discrepancias y ruido que fueron encontradas observando y comparando las anotaciones de los estudiantes.

Si bien esto dificulta generalizar las conclusiones de este trabajo y presuponer sobre la calidad de anotaciones en otros *conjuntos de datos*†, dado que estos suelen componerse de imágenes más nítidas, de mayor calidad y menor problema de oclusión; además, su política de etiquetado conservadora es más resistente a dar lugar a ruido y subjetividad en las anotaciones.

5.6. Síntesis del estudio

A modo de síntesis y conclusión del estudio de anotaciones se enumera lo siguiente:

- Si bien el conjunto **MT** posee imágenes de un gran orden de dificultad, resultó de interés ver la varianza entre anotaciones para dichas imágenes.
- Para las imágenes de dificultad tolerable y de posible comparación al resto de *conjuntos de datos*† públicos en el conteo de multitudes, resultó de gran interés observar que un total de 12 anotadores no coincidieron en ninguna imagen en cuanto a la cantidad de personas anotadas (inclusive en las de dificultad baja). Esto parecería indicar que los *conjuntos de datos*† en el área del conteo de multitudes tienen una cantidad no menospreciable de ruido en sus anotaciones.
- Sin embargo, como se mencionó en la sección anterior, no es trivial generalizar este estudio a los *conjuntos de datos*† empleados en el área de conteo. Debido a la gran dificultad de las imágenes y a que los anotadores anotaron en un tiempo acotado la totalidad de las 56 imágenes del mismo.
- A su vez, si se desease puntualmente estudiar la calidad de uno de los conjuntos disponibles en la literatura, sería de interés tomar las imágenes ya anotadas de dicho conjunto.
- Por otro lado, si se quisiera incrementar la calidad de la anotación, una posible estrategia consistiría en dar menos imágenes a cada anotador (o más tiempo), descartar las anotaciones atípicas (*outliers*) y compaginar las anotaciones de cada imagen a modo de obtener una anotación final. Estos lineamientos posiblemente otorgarían anotaciones de mayor calidad y menor discrepancia entre los anotadores.

Capítulo 6

Experimentación con modelos pre-entrenados

Como se comentó en la sección 4.11.1.2, es de interés evaluar el desempeño de modelos dentro y fuera del contexto en el que fueron entrenados; es decir, evaluar qué tanto impacta el *conjunto de datos*† en el desempeño del modelo para datos similares y otros distintos.

A modo de cubrir una comparación con resultados diversos y variados, así como de mantener la misma acotada dentro del alcance del trabajo, se propone evaluar modelos pre-entrenados en varios *conjuntos de evaluación*†.

El código que implementa esta experimentación se encuentra en el archivo *CC-Evaluation.ipynb* (disponible en el *repositorio público de este proyecto* (`https://github.com/renzodgc/fing-crowdcounting`)). Se empleó *Google Colab* como equipo en donde correr la misma.

6.1. Modelos seleccionados

Para disminuir el esfuerzo de implementación, se empleó la biblioteca *LWCC* presentada en la sección 3.3.1. Esta misma presenta una interfaz para correr *inferencias*† de los siguientes modelos pre-entrenados:

- **CSRNet** (*Li et al.*, 2018): Presentado en la sección 4.7.1. Cuenta con dos modelos, uno entrenado con ShanghaiTech A y otro con ShanghaiTech B (*Zhang et al.*, 2016b).
- **BayL** (*Ma et al.*, 2019): Presentado en la sección 4.9.1. Cuenta con tres modelos, uno entrenado con ShanghaiTech A, otro con ShanghaiTech B (*Zhang et al.*, 2016b) y otro con UCF-QNRF (*Idrees et al.*, 2018b).
- **DM-Count** (*Wang et al.*, 2020a): Presentado en la sección 4.9.2. Cuenta con tres modelos, uno entrenado con ShanghaiTech A, otro con ShanghaiTech B (*Zhang et al.*, 2016b) y otro con UCF-QNRF (*Idrees et al.*, 2018b).
- **SFANet** (*Zhu et al.*, 2019): Presentado en la sección 4.8.2.2. Cuenta con un modelo, entrenado con ShanghaiTech B (*Zhang et al.*, 2016b).

Además, a modo de expandir la experimentación, se agregó a la misma el modelo **SASNet** (*Song et al.*, 2021), presentado en la sección 4.8.2.3. Basándose en su *implementación oficial*, cuenta con dos modelos, uno entrenado con ShanghaiTech A y otro con ShanghaiTech B (*Zhang et al.*, 2016b).

En total se evaluaron once modelos, con cinco posibles arquitecturas.

6.1.1. Conjuntos de datos seleccionados

Para mantener el alcance de la experimentación diverso pero acotado, se seleccionaron conjuntos de datos comúnmente usados en el conteo de multitudes. Dichos conjuntos, debido a su naturaleza, tienen un número acotado de instancias en su *conjunto de evaluación*†.
Estos conjuntos son:

- **ShanghaiTech A** (*Zhang et al.*, 2016b): Posee 182 instancias de evaluación, promediando $\mu = 433{,}9$ personas por instancia y con una desviación estándar $\sigma = 353{,}7$.
- **ShanghaiTech B** (*Zhang et al.*, 2016b): Posee 316 instancias de evaluación, promediando $\mu = 124{,}1$ personas por instancia y con una desviación estándar $\sigma = 95{,}2$.
- **UCF-CC-50** (*Idrees et al.*, 2013): Posee 50 instancias de evaluación, promediando $\mu = 1279{,}5$ personas por instancia y con una desviación estándar $\sigma = 950{,}8$.
- **UCF-QNRF** (*Idrees et al.*, 2018b): Posee 334 instancias de evaluación, promediando $\mu = 718{,}9$ personas por instancia y con una desviación estándar $\sigma = 746{,}6$.
- **MT**: Con 56 instancias de evaluación, promediando $\mu = 1143{,}1$ personas por instancia y con una desviación estándar $\sigma = 2265{,}1$. **Siendo sus subconjuntos:**
  - **MT paraguas**: Posee 6 instancias de evaluación, promediando $\mu = 100{,}0$ personas por instancia y con una desviación estándar $\sigma = 72{,}7$.
  - **MT densidad baja**: Posee 13 instancias de evaluación, promediando $\mu = 38{,}9$ personas por instancia y con una desviación estándar $\sigma = 17{,}3$.
  - **MT densidad media**: Posee 26 instancias de evaluación, promediando $\mu = 247{,}6$ personas por instancia y con una desviación estándar $\sigma = 270{,}2$.
  - **MT aéreo**: Posee 11 instancias de evaluación (excluyendo la imagen duplicada **bm03**), promediando $\mu = 5133{,}9$ personas por instancia y con una desviación estándar $\sigma = 2467{,}3$.

Debido al momento en el que se ejecutó la experimentación y posteriormente se obtuvieron las anotaciones del **MT**, estos resultados fueron obtenidos empleando como *valor anotado*† el conjunto **MT** anotado por el estudiante que realizó este trabajo (en lugar de por ejemplo combinar la información de las anotaciones de todos los anotadores para cada imagen).

Hubiera sido de interés investigar técnicas para combinar esta información, o incluso probar promediando las matrices. Las mismas dejarían de ser matrices binarias de anotación, y pasarían en teoría a ser algo similar a los *mapas de densidad*† generados por el *filtro gaussiano*†. Sin embargo, por falta de recursos esto quedó fuera del alcance del proyecto.

6.2. Resultados obtenidos

Para cada *evaluación*†, se calculó el *MAE*†, *RMSE*† y *GAME*† (partiendo la imagen en 4 cuadrantes y 16 cuadrantes). No se calculan las métricas *PSNR*† ni *SSIM*† debido a que para aplicar dichas *métricas de evaluación*† con modelos pre-entrenados sería necesario, para cada modelo y *conjuntos de evaluación*†, generar los *mapas de densidad*† con el mismo algoritmo empleados en el *entrenamiento*† del modelo.

Para cada *conjunto de evaluación*† se presentan los resultados obtenidos en:

- Las tablas 6.1, 6.2, 6.3, 6.4 para **ShanghaiTech A** (*Zhang et al.*, 2016b), **ShanghaiTech B** (*Zhang et al.*, 2016b), **UCF-QNRF** (*Idrees et al.*, 2018b) y **UCF-CC-50** (*Idrees et al.*, 2013) respectivamente.
- Las tablas 6.5, 6.6, 6.7, 6.8 para los subconjuntos de **MT**.
- La tabla 6.9 para **MT** en su totalidad.

Cada tabla además marca para cada *métrica de evaluación*† los tres modelos de mejor desempeño, con los colores **azul**, **violeta** y **rojo**, para el mejor, segundo mejor y tercer mejor desempeño.

6.2.1. Resultados obtenidos para conjuntos de datos previamente existentes

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	84.8450	136.8059	92.9638	104.8193
CSRNet-SHB	137.3784	203.7355	149.4631	162.8108
BayL-SHA	**70.6593**	**125.6271**	78.4654	89.5939
BayL-SHB	154.3526	258.9801	158.8308	166.1559
BayL-QNRF	71.0957	139.1065	**77.9976**	**86.5307**
DM-Count-SHA	**69.8156**	**117.4020**	**77.6948**	**87.7487**
DM-Count-SHB	134.7981	228.7521	141.1231	151.2169
DM-Count-QNRF	73.5796	134.3054	80.1491	91.5221
SFANet-SHB	275.0659	433.0353	278.1661	283.3521
SASNet-SHA	**66.8618**	**112.0636**	**74.4048**	**84.5781**
SASNet-SHB	137.7576	224.6520	143.8814	151.6344

Tabla 6.1: Resultados obtenidos para el *conjunto de evaluación*† **ShanghaiTech A** (*Zhang et al.*, 2016b), que promedia $\mu = 433{,}9$ personas con desviación estándar $\sigma = 353{,}7$.

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	17.1379	28.3993	26.4303	32.8175
CSRNet-SHB	18.5044	27.8417	20.7088	25.6379
BayL-SHA	18.6100	28.1117	21.4876	24.9654
BayL-SHB	**9.5518**	**16.3470**	**11.6985**	**15.3639**
BayL-QNRF	19.9739	34.2619	21.4685	24.3849
DM-Count-SHA	25.3636	39.0231	27.3264	30.5361
DM-Count-SHB	**9.8137**	**16.6658**	**12.2151**	**16.4453**
DM-Count-QNRF	16.3029	31.2214	18.3850	22.1044
SFANet-SHB	15.4167	23.0433	21.1743	24.9660
SASNet-SHA	20.7018	32.7109	22.6987	26.0365
SASNet-SHB	**6.7886**	**10.8497**	**9.1115**	**13.0870**

Tabla 6.2: Resultados obtenidos para el *conjunto de evaluación*† **ShanghaiTech B** (*Zhang et al.*, 2016b), que promedia $\mu = 124{,}1$ personas con desviación estándar $\sigma = 95{,}2$.

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	198.9389	382.4800	217.2546	235.0503
CSRNet-SHB	268.0412	458.2792	279.5677	293.5203
BayL-SHA	**187.1092**	**374.7255**	**199.2461**	**213.9938**
BayL-SHB	321.9111	596.4052	329.6540	338.3826
BayL-QNRF	202.7944	400.7214	212.0655	223.8775
DM-Count-SHA	**187.5585**	**364.3632**	**198.2582**	**211.4380**
DM-Count-SHB	266.8697	502.9515	277.5243	290.0523
DM-Count-QNRF	**179.8183**	**347.2329**	**190.9049**	**203.8735**
SFANet-SHB	535.1482	898.0133	538.7654	543.9263
SASNet-SHA	199.5770	401.7283	208.3973	220.7485
SASNet-SHB	286.6907	507.7549	291.8884	298.6263

Tabla 6.3: Resultados obtenidos para el *conjunto de evaluación*† **UCF-QNRF** (*Idrees et al.*, 2018b), que promedia $\mu = 718{,}9$ personas con desviación estándar $\sigma = 746{,}6$.

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	589.7383	927.5038	617.0222	639.1082
CSRNet-SHB	1124.9318	1421.9278	1124.9655	1125.3782
BayL-SHA	**528.8997**	861.5352	**554.6999**	**578.8141**
BayL-SHB	1219.7110	1525.9307	1221.9852	1222.2742
BayL-QNRF	612.9501	918.2360	633.7644	657.8205
DM-Count-SHA	536.3985	**847.4061**	568.6539	601.5511
DM-Count-SHB	1184.5086	1497.1617	1184.5086	1184.6940
DM-Count-QNRF	**468.2744**	**679.6145**	**503.5936**	**540.1029**
SFANet-SHB	1145.0595	1499.0954	1146.3026	1149.8768
SASNet-SHA	**480.5547**	**787.2065**	**511.6908**	**535.1027**
SASNet-SHB	1195.4537	1502.9469	1195.4538	1196.8268

Tabla 6.4: Resultados obtenidos para el *conjunto de datos*† **UCF-CC-50** (*Idrees et al.*, 2013), que promedia $\mu = 1279{,}5$ personas con desviación estándar $\sigma = 950{,}8$.

6.2.2. Resultados obtenidos para conjuntos de datos MT

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	**41.2627**	**74.0834**	**48.2119**	**59.4715**
CSRNet-SHB	82.67295	108.4114	82.7451	83.2961
BayL-SHA	71.6992	98.2291	71.8441	73.5528
BayL-SHB	80.546	111.3801	80.5532	82.3069
BayL-QNRF	77.8774	109.1784	77.8877	78.5353
DM-Count-SHA	**67.3079**	**90.7772**	**67.3602**	**68.3621**
DM-Count-SHB	79.0107	110.0208	79.0373	81.2361
DM-Count-QNRF	76.2074	105.7085	76.2074	76.6062
SFANet-SHB	68.2976	100.3497	70.9559	74.8233
SASNet-SHA	**66.1806**	**85.8648**	**66.2176**	**67.0905**
SASNet-SHB	78.3348	109.1648	78.7003	79.9291

Tabla 6.5: Resultados obtenidos para el subconjunto **MT paraguas**, que promedia $\mu = 100{,}0$ personas con desviación estándar $\sigma = 72{,}7$.

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	12.7512	16.9616	20.4412	28.4359
CSRNet-SHB	11.1505	12.5221	16.9662	19.9244
BayL-SHA	**10.1384**	**11.924**	**12.3152**	**15.1838**
BayL-SHB	13.3462	15.0221	14.9241	16.0506
BayL-QNRF	**8.2913**	**9.4297**	**11.2971**	**13.5835**
DM-Count-SHA	12.9519	14.757	14.0687	15.6431
DM-Count-SHB	13.6476	15.6377	14.9148	16.4813
DM-Count-QNRF	**7.1603**	**7.9256**	**8.4191**	**10.4268**
SFANet-SHB	17.0615	25.2955	28.4125	32.0732
SASNet-SHA	10.7113	12.4571	13.4628	16.1836
SASNet-SHB	14.4341	16.1471	15.28	15.9175

Tabla 6.6: Resultados obtenidos para el subconjunto **MT densidad baja**, que promedia $\mu = 38{,}9$ personas con desviación estándar $\sigma = 17{,}3$.

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	91.3511	167.9605	108.9819	123.5643
CSRNet-SHB	110.3214	189.9662	117.5906	127.3573
BayL-SHA	**65.9103**	**120.0802**	**74.2937**	**87.5476**
BayL-SHB	120.4666	223.2901	123.22	128.9284
BayL-QNRF	**76.7304**	148.6771	**81.3839**	**89.0568**
DM-Count-SHA	76.7693	**132.7744**	86.9773	94.2291
DM-Count-SHB	107.8668	194.2178	112.9208	119.515
DM-Count-QNRF	**60.2941**	**122.2286**	**69.3511**	**77.9778**
SFANet-SHB	148.0384	276.1633	152.9078	160.3787
SASNet-SHA	84.3264	162.543	99.4864	108.4416
SASNet-SHB	128.1936	233.7908	128.5797	133.3845

Tabla 6.7: Resultados obtenidos para el subconjunto **MT densidad media**, que promedia $\mu = 247{,}6$ personas con desviación estándar $\sigma = 270{,}2$.

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	4313.2046	5111.926	4340.4682	4373.8143
CSRNet-SHB	4369.6709	5096.3794	4381.3042	4401.5169
BayL-SHA	**4160.1466**	**4966.2282**	**4179.3975**	**4226.7234**
BayL-SHB	4764.2312	5412.9991	4769.4662	4773.2415
BayL-QNRF	4397.4696	5158.1041	4408.6946	4417.2357
DM-Count-SHA	**4076.0057**	**4907.2557**	**4123.2712**	**4173.5811**
DM-Count-SHB	4494.0983	5215.1816	4505.9972	4510.0484
DM-Count-QNRF	4269.9221	**5089.4628**	4305.3919	**4314.3516**
SFANet-SHB	4971.3344	5571.215	4976.2585	4983.7699
SASNet-SHA	**4277.0921**	5104.9399	**4282.7181**	4324.4161
SASNet-SHB	4636.0247	5317.1038	4642.2164	4646.4352

Tabla 6.8: Resultados obtenidos para el subconjunto **MT aéreo**, que promedia $\mu = 5133{,}9$ personas con desviación estándar $\sigma = 2467{,}3$.

Modelo	MAE	RMSE	GAME4	GAME16
CSRNet-SHA	897.0324	2268.5230	913.0841	929.4525
CSRNet-SHB	920.9485	2262.5576	927.9868	937.2187
BayL-SHA	**857.9364**	**2202.9563**	**866.1373**	**882.3989**
BayL-SHB	1003.5164	2404.1784	1006.1900	1010.0180
BayL-QNRF	909.7641	2288.7248	914.8013	920.6233
DM-Count-SHA	**846.5563**	**2177.0193**	**860.8282**	**874.5521**
DM-Count-SHB	944.5462	2315.5090	949.5132	953.9746
DM-Count-QNRF	**876.7260**	**2257.7554**	**888.1738**	**894.4575**
SFANet-SHB	1056.4672	2476.5233	1062.6151	1068.8300
SASNet-SHA	889.0339	2265.6336	897.8271	910.9049
SASNet-SHB	981.8179	2361.9956	983.4514	986.7919

Tabla 6.9: Resultados obtenidos para el *conjunto de datos*† **MT** (en su totalidad), que promedia $\mu = 1143{,}1$ personas con desviación estándar $\sigma = 2265{,}1$.

6.2.3. Observaciones e interpretaciones de los resultados

Observando estos resultados, se realizan las siguientes observaciones, interpretaciones y conclusiones:

- Para *ShanghaiTech A* (tabla 6.1) y *ShanghaiTech B* (tabla 6.2), los tres modelos con mejor *MAE*† para cada uno están entrenados en dicho conjunto, siendo los de mejor desempeño *SASNet-SHA* y *SASNet-SHB* respectivamente. El que no alcancen mejor desempeño en las otras evaluaciones (como en la evaluación de *UCF-QNRF* (tabla 6.3), donde *SASNet-SHA* tiene peor desempeño que *BayL-SHA* y *DM-Count-SHA*) parece indicar que estos modelos poseen *adaptación al dominio*† en el que fueron entrenados.
  Si bien esto les reduce su capacidad de generalización, también causa que presenten el mejor desempeño (lo que podría ser deseable si se quisieran integrar a soluciones para tales dominios).

- Para *UCF-QNRF* (tabla 6.3), el modelo con mejor *MAE*† es *DM-Count-QNRF*, pero *BayL-QNRF* tiene peor desempeño que *BayL-SHA* y *DM-Count-SHA*.

- En varios casos el desempeño de los modelos entrenados con *UCF-QNRF* es relativamente similar al del mismo modelo entrenado con *ShanghaiTech A*.
  Esto podría ser porque ambos *conjuntos de entrenamiento*† comparten dominios en sus imágenes.

- Salvo en la evaluación en *ShanghaiTech B*, los modelos de mejor desempeño en las evaluaciones siempre fueron entrenados con *ShanghaiTech A* o *UCF-QNRF*, desempeñándose significativamente mejor que los entrenados con *ShanghaiTech B*. Esto indica que *ShanghaiTech B* contiene imágenes de un dominio disimilar al resto, lo cual tiene sentido no sólo porque *ShangaiTech A* y *UCF-QNRF* tienen más individuos en promedio por imagen, sino también porque el conjunto *ShanghaiTech B* está conformado únicamente con fotos tomadas en las calles de Shanghai (mientras que *ShanghaiTech A* y *UCF-QNRF* contienen fotos recopiladas de internet, lo que aumenta su diversidad en dominios).
  Se concluye de esto que los modelos entrenados con el conjunto *ShanghaiTech B* tienen mala capacidad de generalización a dominios desconocidos por más que éstos promedien una cantidad de personas parecida a la que hay en *ShanghaiTech B*, como ocurre en la *evaluación*† de *MT densidad media* (tabla 6.7). Siguen siendo los que mejor se desempeñan en el *conjunto de evaluación*† *ShanghaiTech B*, presumiblemente debido a su *adaptación al dominio*† de dicho conjunto.

- No hay ningún modelo que haya sido el de mejor desempeño en todos los *conjuntos de evaluación*†.
  El de mejor desempeño en la mayor cantidad de conjuntos evaluados es *DM-Count-QNRF* (en 4 de 8 conjuntos).
  Esto indica que los modelos pre-entrenados con los que se evaluó tienen una *adaptación al dominio*† al cual fueron entrenados, presentando un sesgo sobre el tipo de imágenes en las que alcanzan un mejor desempeño (y teniendo peor desempeño en dominios desconocidos).

- En la mayoría de los casos, los modelos de mejor desempeño en la *métrica de evaluación*† *MAE*† tuvieron también el mejor desempeño para las *métricas de evaluación*† *RMSE*†, *GAME*†4, y *GAME*†16.

- Si bien *GAME*† es una *métrica de evaluación*† tal vez más adecuada para la tarea de conteo en base a *mapas de densidad*†, se concluye que *MAE*† también es una buena *métrica de evaluación*†, puesto que sus valores en general no tienen gran diferencia y la mayoría de los casos con mejor *MAE*† también tienen mejor *GAME*†.
  Esto tiene sentido considerando que los modelos deberían de ser robustos al recibir una imagen por parches (tanto por la naturaleza de su *entrenamiento*† como por las *capas convolucionales*†).

- *SFANet-SHA*, *CSRNet-SHA* y *CSRNet-SHB* no estuvieron entre los tres mejores modelos para ningún *conjunto de evaluación*†, exceptuando *CSRNet-SHA* en el conjunto *MT paraguas*.

- Comparando la *MAE*† evaluada en los subconjuntos de *MT* con la evaluada en los otros conjuntos y teniendo en cuenta el promedio de individuos y desviación estándar en cada conjunto, se observa que el desempeño de todos los modelos es significativamente peor para los subconjuntos *MT densidad*

baja y *MT densidad media*, y el desempeño de todos los modelos es particularmente malo para *MT paraguas* y *MT aéreo*.
Esto se debe en parte a que los modelos están adaptados a otros dominios (a diferencia de fotos con paraguas u obtenidas por drones a altas altitudes), pero también se cree que ocurre porque el conjunto *MT* es de mayor dificultad para la tarea del conteo que los otros *conjuntos de evaluación*†.

6.2.4. Visualización de casos destacados

A continuación se incluyen ejemplos de anotaciones a lo largo de los conjuntos y modelos con los que se experimentó. A su vez se comentan para cada imagen observaciones, particularidades o anomalías por las cuales la misma fue seleccionada. Los casos se dividen en las siguientes secciones:

- Estimaciones de calidad aceptable (sección 6.2.4.1)
- Estimaciones de calidad mixta (sección 6.2.4.2)
- Estimaciones de mala calidad (sección 6.2.4.3)
- Efecto de la oclusión en imágenes (sección 6.2.4.4)
- Ejemplos de otras particularidades (sección 6.2.4.5)
- Ejemplos de falsos positivos u artefactos en mapas de densidad generados (sección 6.2.4.6)

6.2.4.1. Ejemplos de estimaciones de calidad aceptable

Se consideran de calidad aceptable a los resultados que varios modelos tuvieron un error menor al del 10 % de la cantidad anotada de personas (figuras 6.1, 6.2, y 6.3). Esta sección es extendida extiende con otros 6 ejemplos en el apéndice A.1.1.

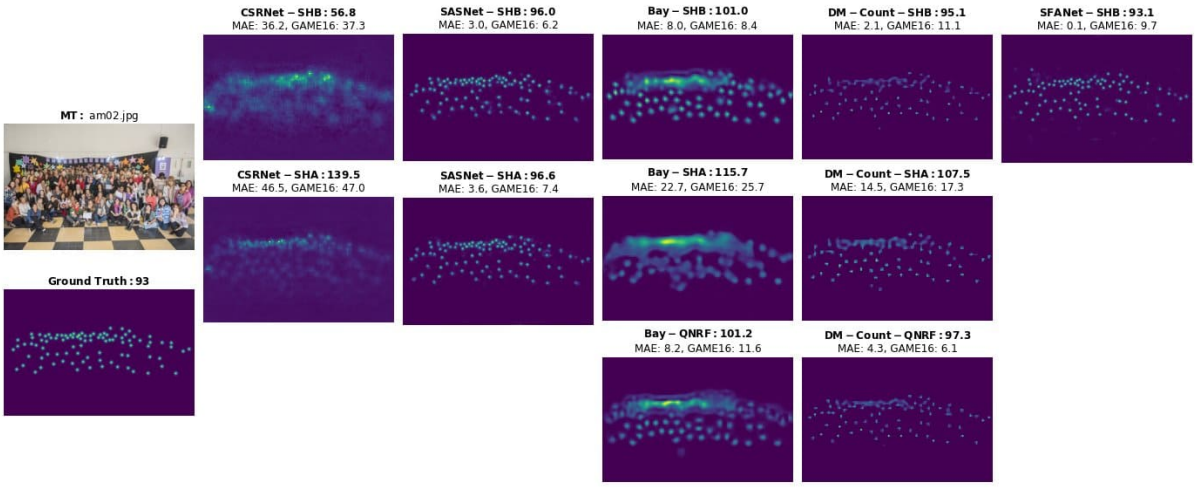


Figura 6.1: MT-media (am02): Imagen de baja dificultad, el modelo *SFANet-SHB* tuvo un desempeño excelente. Mientras que *CSRNet-SHA* y *CSRNet-SHB* obtuvieron muy mal desempeño en relación a los otros modelos, el primero sobreestimando y el segundo infraestimando la cantidad de personas en la imagen.

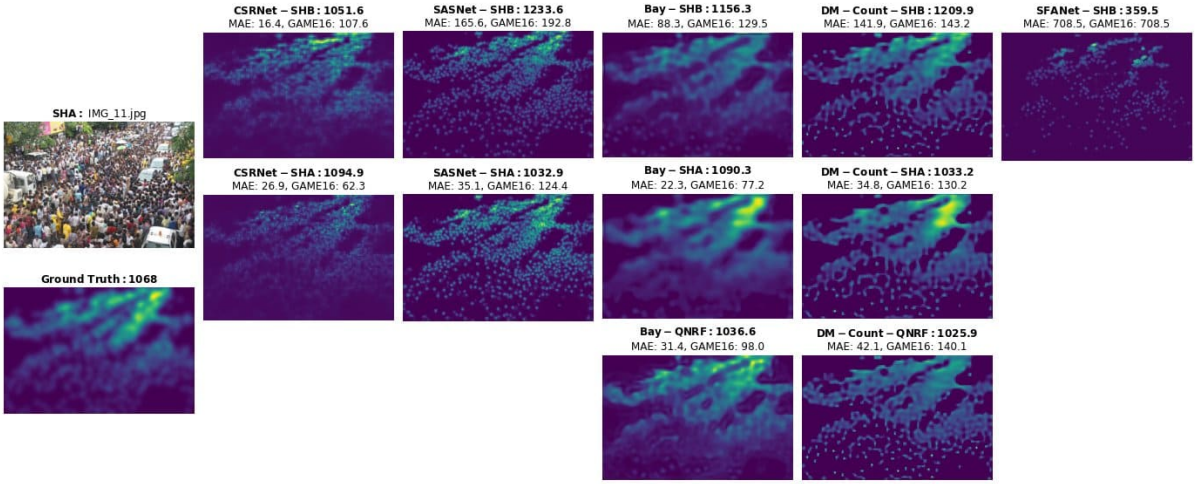


Figura 6.2: SHA (11): Imagen con muy buen desempeño para el tamaño de la multitud, salvo por el modelo *SFANet-SHB*; los *mapas de densidad*† son de muy buena calidad.

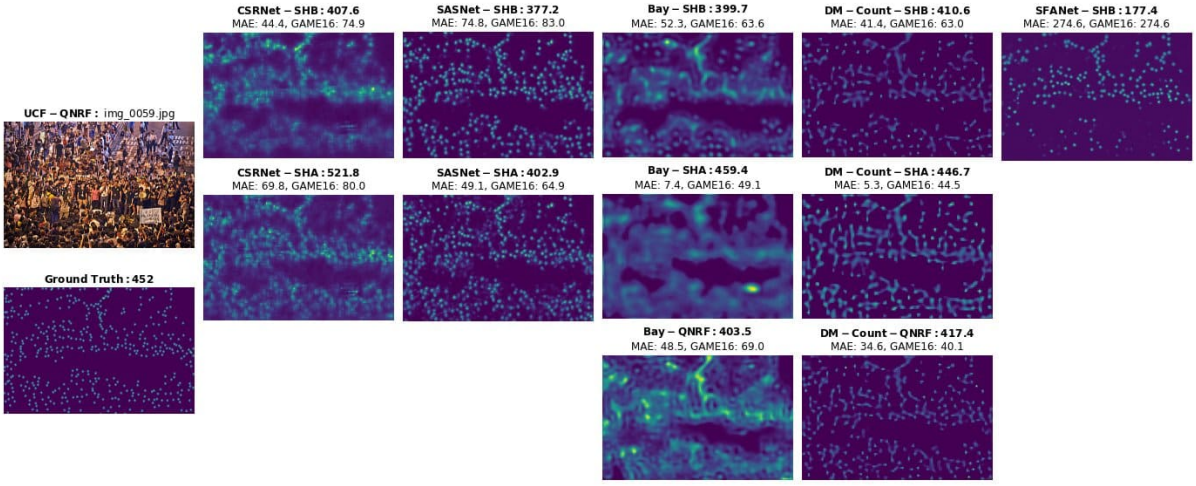


Figura 6.3: UCF-QNRF (59): Imagen de densidad heterogénea, se aprecia cómo cada modelo genera un *mapa de densidad*† con patrones distintos al resto, principalmente *Bay*. También se destaca cómo los mejores resultados son *Bay-SHA* y *DM-Count-SHA* en lugar los que fueron entrenados con *UCF-QNRF* (conjunto del que proviene esta imagen de *evaluación*†). Esto se podría deber a que esta imagen es de un dominio similar a las del *conjunto de entrenamiento*† en *SHA*.

6.2.4.2. Ejemplos de estimaciones de calidad regular

Se consideran de calidad regular a los resultados que varios modelos tuvieron un error entre el 10 % y 25 % de la cantidad anotada de personas (figuras 6.4, 6.5 y 6.6). Esta sección es extendida extiende con otros 5 ejemplos en el apéndice A.1.2.

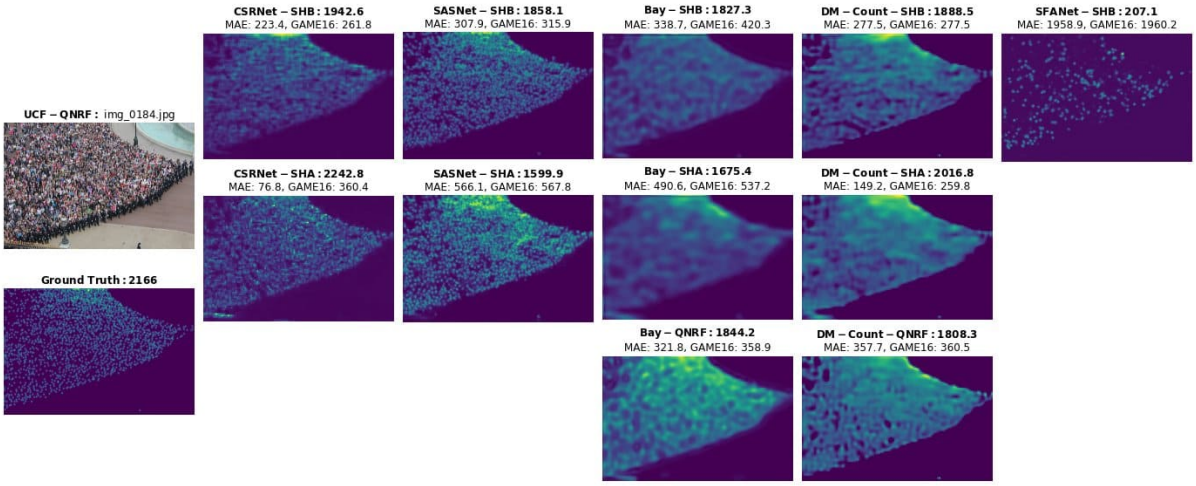


Figura 6.4: UCF-QNRF (184): Desempeño aceptable para el tamaño de la multitud, excepto para *SFANet-SHB*. El mejor resultado se obtuvo con *CSRNet-SHA* en lugar de con los modelos entrenados con *UCF-QNRF*.

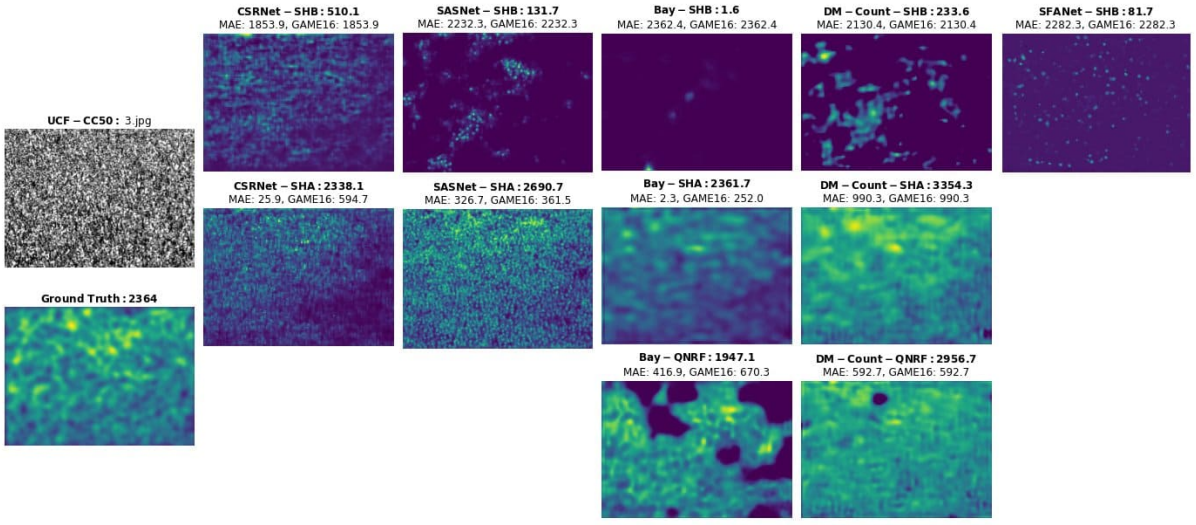


Figura 6.5: UCF-CC50 (3): Imagen de densidad uniforme pero considerable dificultad por estar en escala de grises y que la cámara se encuentra distante a la multitud *Bay-SHA* y *CSRNet-SHA* tienen excelente desempeño aunque su GAME16 es significativamente alta. *Bay-QNRF* y *DM-Count-QNRF* parecen confundir personas en la multitud con el fondo en la imagen, los modelos entrenados con *SHB* tienen muy mal desempeño, esto podría ser porque, a diferencia de *SHA* y *UCF-QNRF*, el *conjunto de entrenamiento*† *SHB* carece de imágenes en escala de grises.

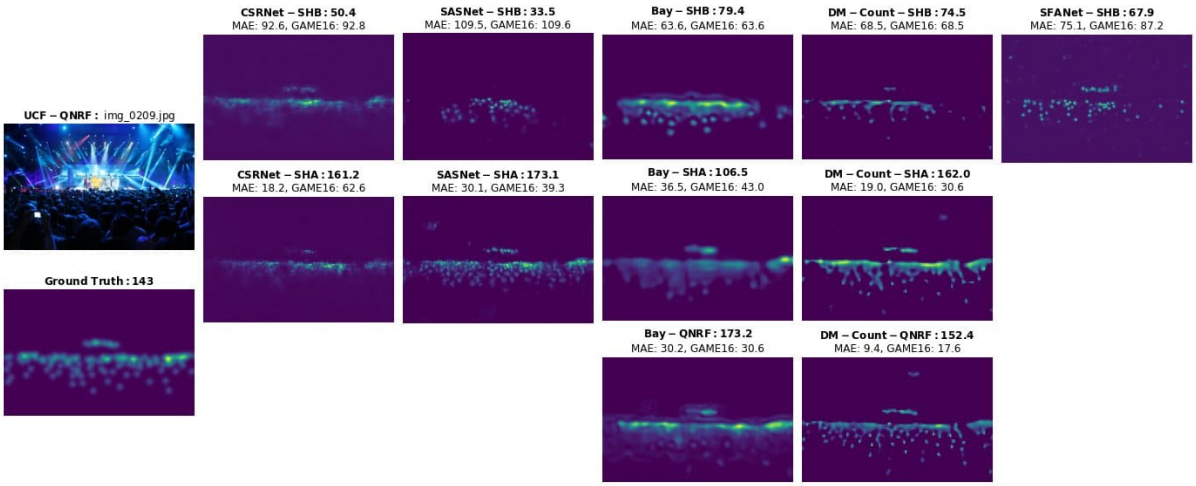


Figura 6.6: UCF-QNRF (209): Imagen de muy mala iluminación, en la que los modelos entrenados con *SHB* infra-estiman la cantidad de personas significativamente. Los otros tienen un desempeño aceptable.

6.2.4.3. Ejemplos de estimaciones de mala calidad

Se consideran de calidad mala a los resultados que varios modelos tuvieron un error mayor al 25 % de la cantidad anotada de personas (figuras 6.7, 6.8 y 6.9). Esta sección es extendida con otros 5 ejemplos en el apéndice A.1.3.

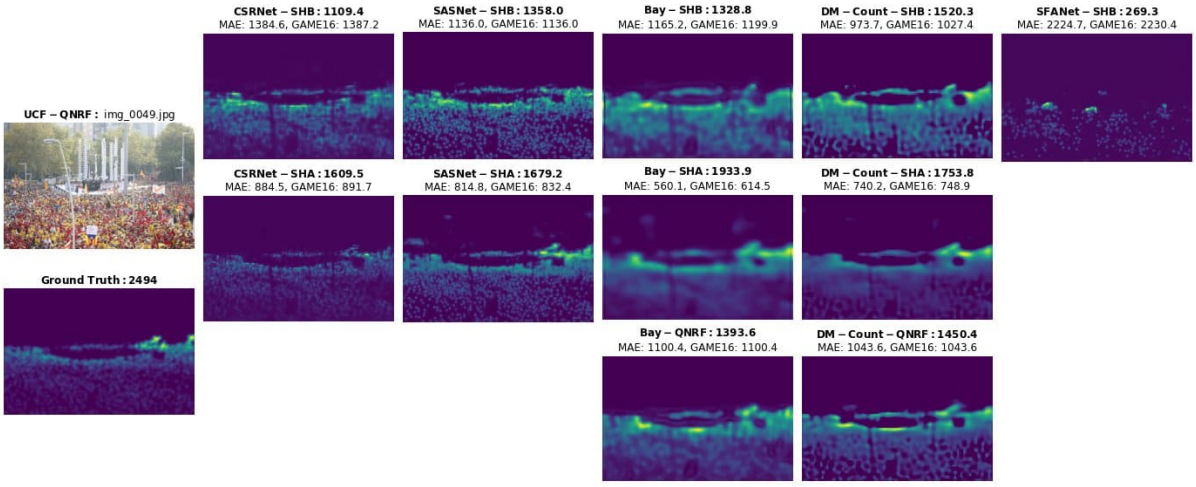


Figura 6.7: UCF-QNRF (49): Imagen infraestimada por todos los modelos.

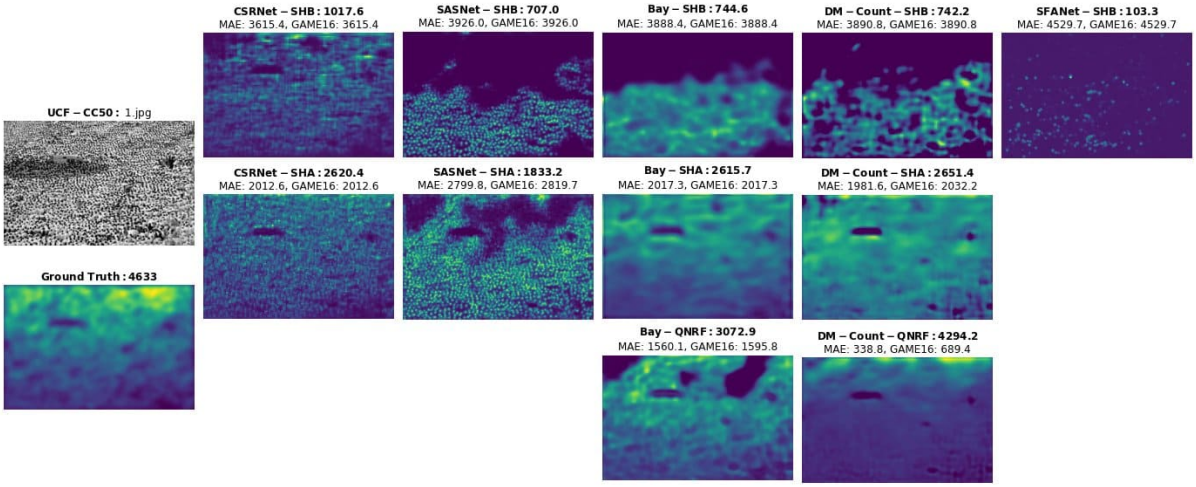


Figura 6.8: UCF-CC50 (1): Imagen en escala de grises de muy alta dificultad, infraestimada por todos los modelos, excepto por *DM-Count-QNRF* (cuyo error es de 339 personas). Los modelos entrenados con *SHB* fueron los de peor desempeño, esto podría ser porque, a diferencia de *SHA* y *UCF-QNRF*, el *conjunto de entrenamiento*† *SHB* carece de imágenes en escala de grises.

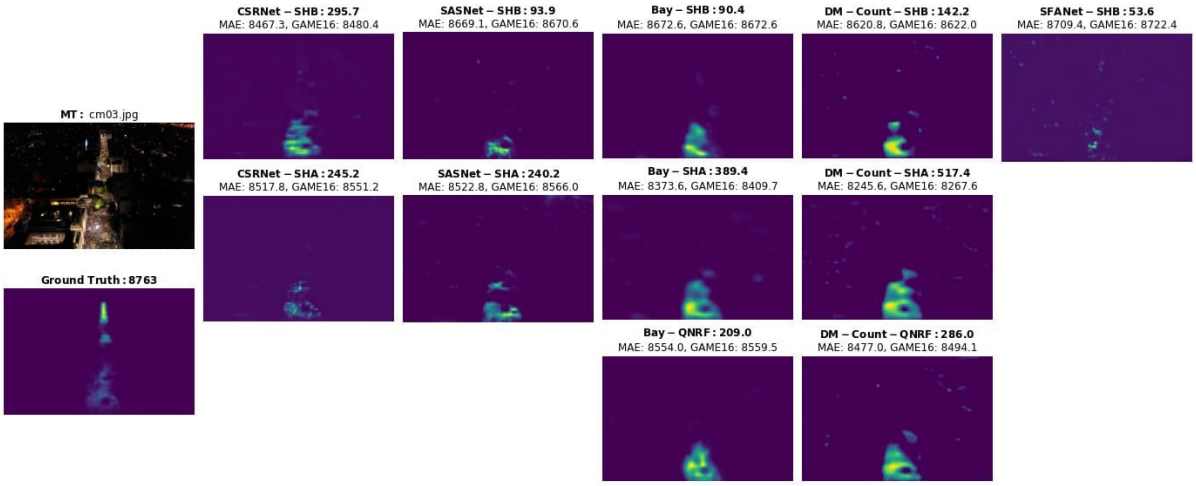


Figura 6.9: MT-aéreo (cm03): Imagen con gran *distorsión de perspectiva*† y tamaño de cabezas pequeño en el frente y diminuto en el horizonte. Todos los modelos tuvieron un muy mal desempeño. Esto es coherente con la alta dificultad de la imagen, siendo una de las más difíciles de anotar en el módulo de taller.

6.2.4.4. Ejemplos del efecto de oclusión en imágenes

En esta sección se destaca como los modelos no realizan estimaciones de personas detrás de *oclusiones*† (figuras 6.10, 6.11, 6.12 y 6.13). Esta sección es extendida con otros 2 ejemplos en el apéndice A.1.4.

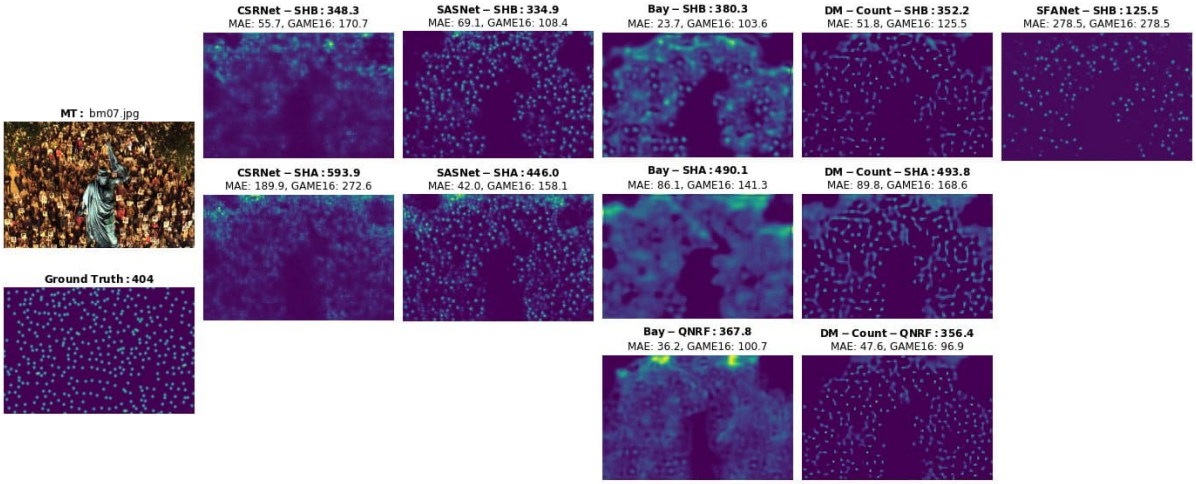


Figura 6.10: MT-media (bm07): Imagen con una densidad uniforme, es razonable presumir la misma se mantiene debajo de la estatua y ramas, pero este no es el caso con las estimaciones de los modelos, se presume que esto ocurre debido a que los *conjuntos de datos*† se limitan a anotar sólo cabezas visibles.

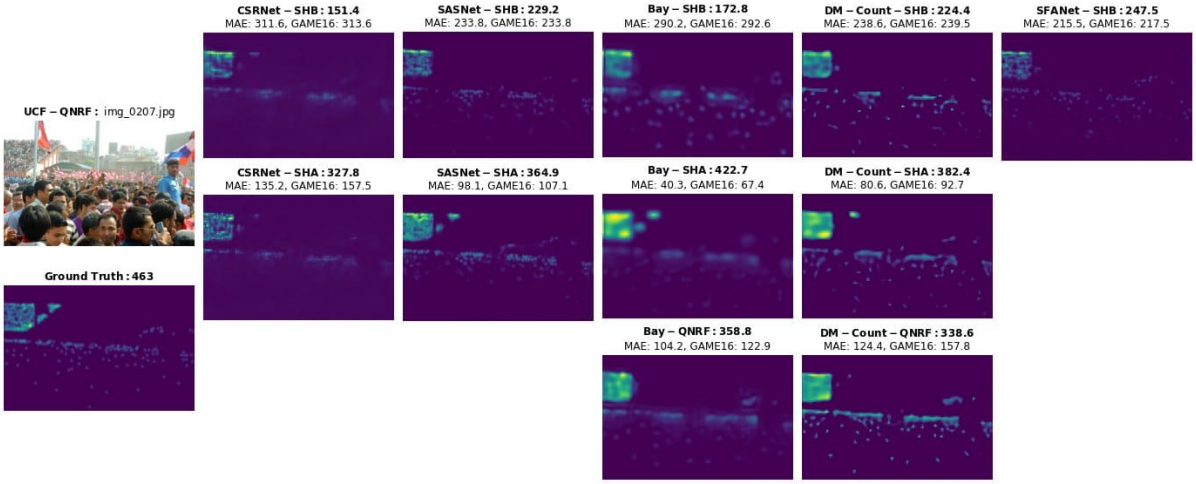


Figura 6.11: UCF-QNRF (207): Imagen en la que la bandera corta parte de las gradas, siendo pocos los modelos (y en general entrenados con *SHA*) que son capaces de detectar y estimar la multitud visible abajo y arriba a la derecha de la bandera.

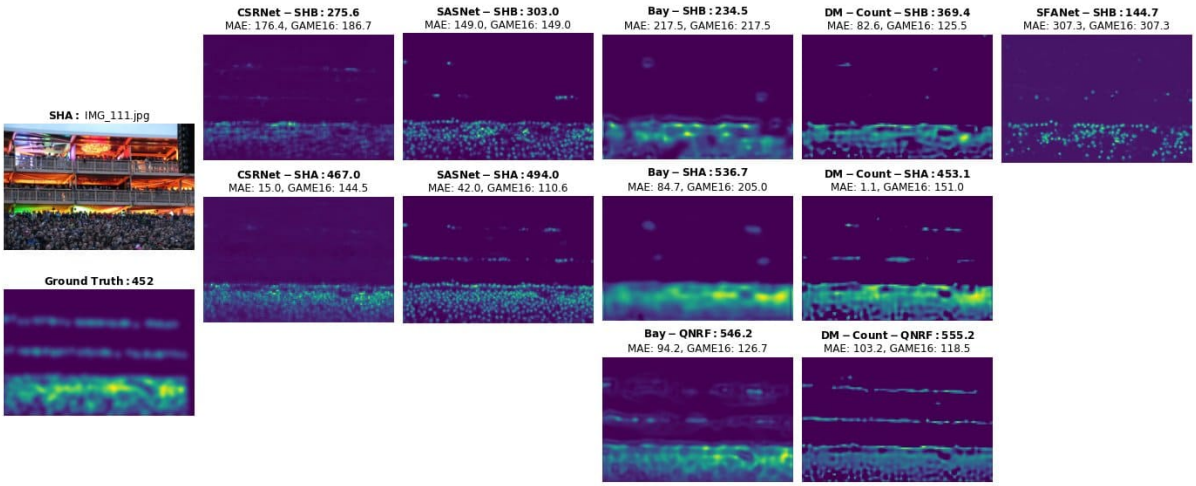


Figura 6.12: SHA (111): Imagen con *fondo complejo*† y *oclusión*†, muchos modelos no detectan personas en el segundo y tercer piso. Los modelos entrenados con *SHA* tienen generalmente un buen desempeño en su *MAE*†, pero su *GAME*†16 es muy alta en comparación, esto indica que su estimación para cada parche no es buena.

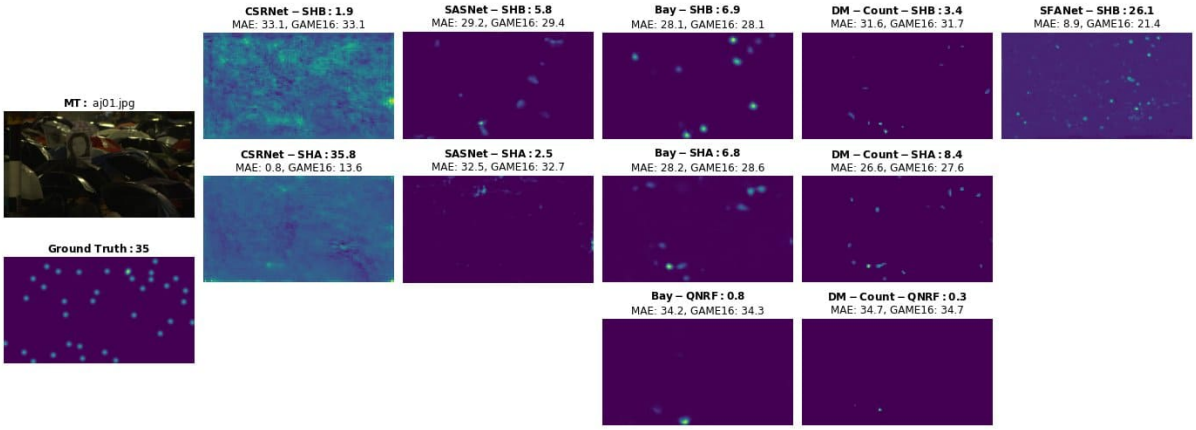


Figura 6.13: MT-paraguas (aj01): Imagen con múltiples *oclusiones*† por paraguas, de muy mal desempeño en los modelos (salvo por *SFANet-SHB* y *CSRNet-SHB*). Se destaca como *CSRNet-SHA* y *CSRNet-SHB* formaron un mapa que no logra localizar a las personas puntualmente en la imagen.

6.2.4.5. Ejemplos de otras particularidades

En esta sección se destacan varias particularidades que se observaron en los resultados obtenidos (figuras 6.14 y 6.15). Esta sección es extendida con otros 4 ejemplos en el apéndice A.1.5.

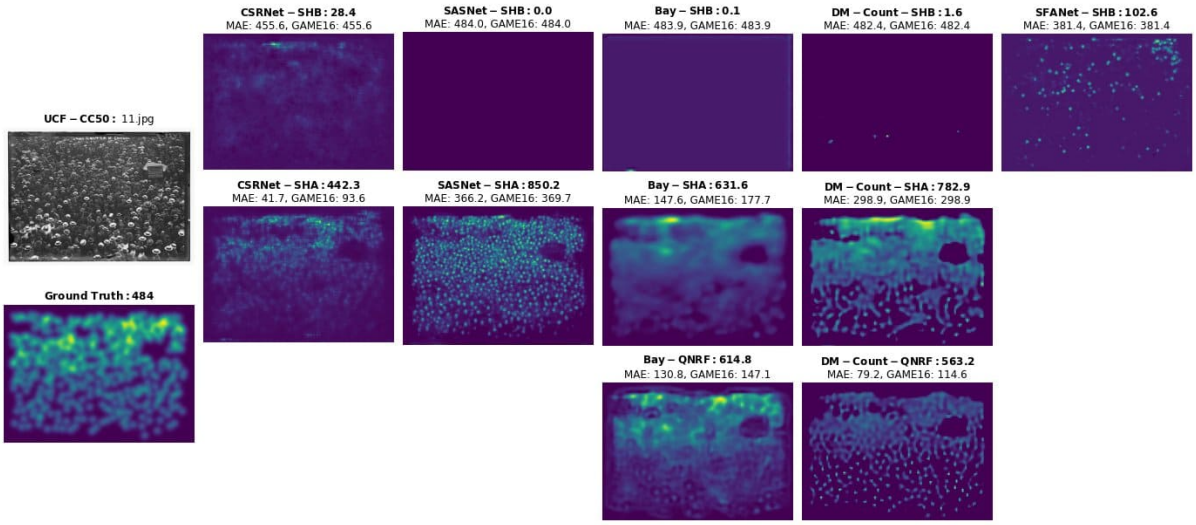


Figura 6.14: UCF-CC50 (11): Imagen en escala de grises y de mala iluminación; es razonablemente bien estimada por *CSRNet-SHA*, muy infraestimada por modelos entrenados con *SHB* (algunos estimando 0 personas), y sobreestimada por el resto de modelos.

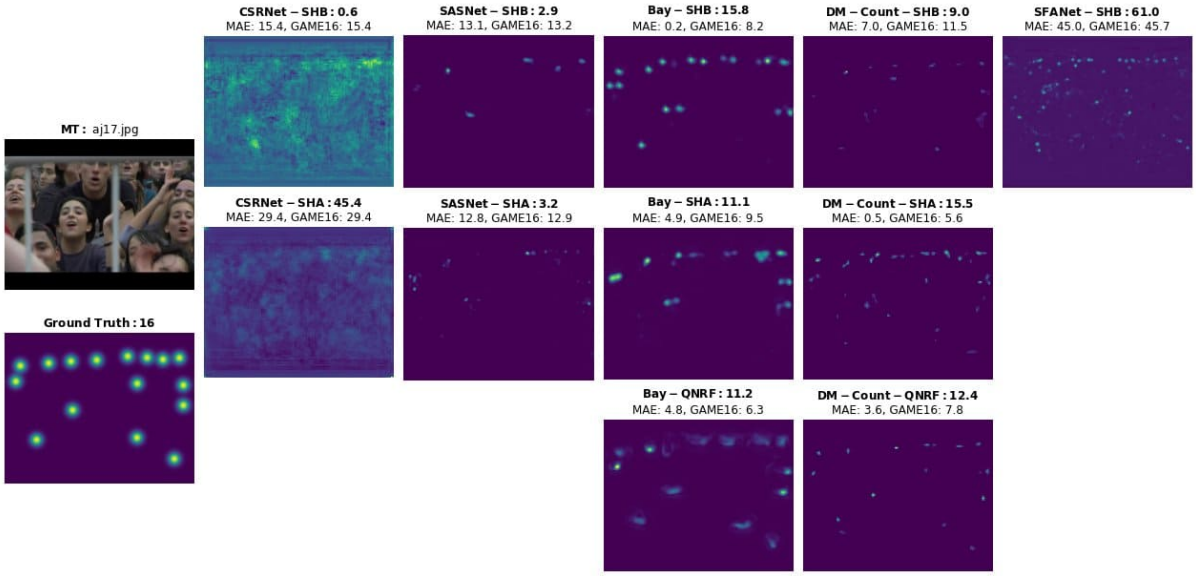


Figura 6.15: MT-baja (aj17): Imagen de muy pocas personas, pese a esto varios modelos fallan estrepitosamente en su estimación (sobreestimando o infraestimando); se destaca como *CSRNet* falla en localizar a las personas, formando una especie de estática distribuida a lo largo del *mapa de densidad*[†].

6.2.4.6. Ejemplos de falsos positivos u artefactos en mapas de densidad generados

En esta sección se destacan casos en los que es claro que un modelo estimó incorrectamente un área en la imagen, introduciendo ruido en la estimación general (figuras 6.16 y 6.17). Esta sección es extendida con otros 4 ejemplos en el apéndice A.1.4.

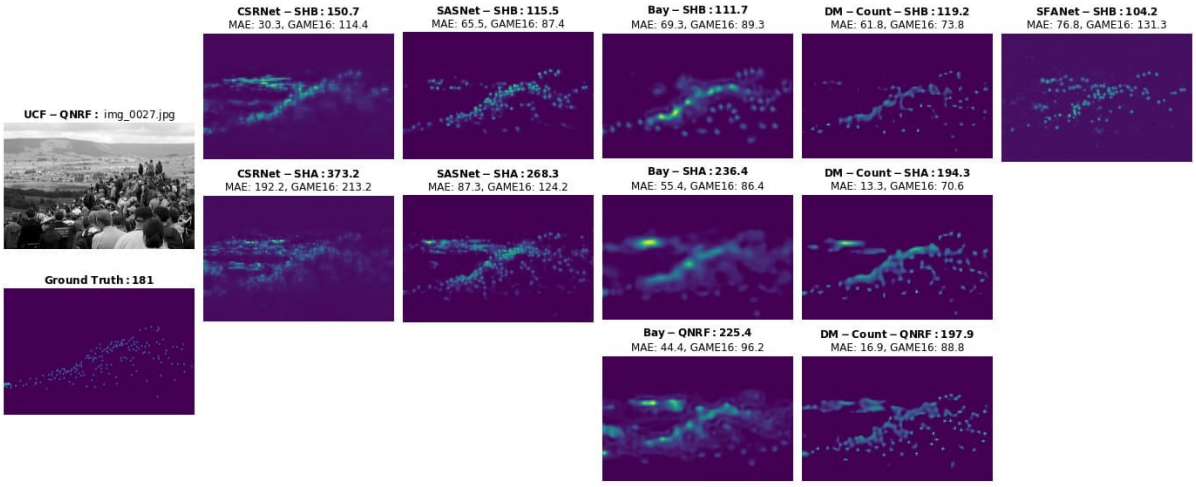


Figura 6.16: UCF-QNRF (27): Se aprecia como varios modelos confunden árboles en el fondo de la imagen con personas.

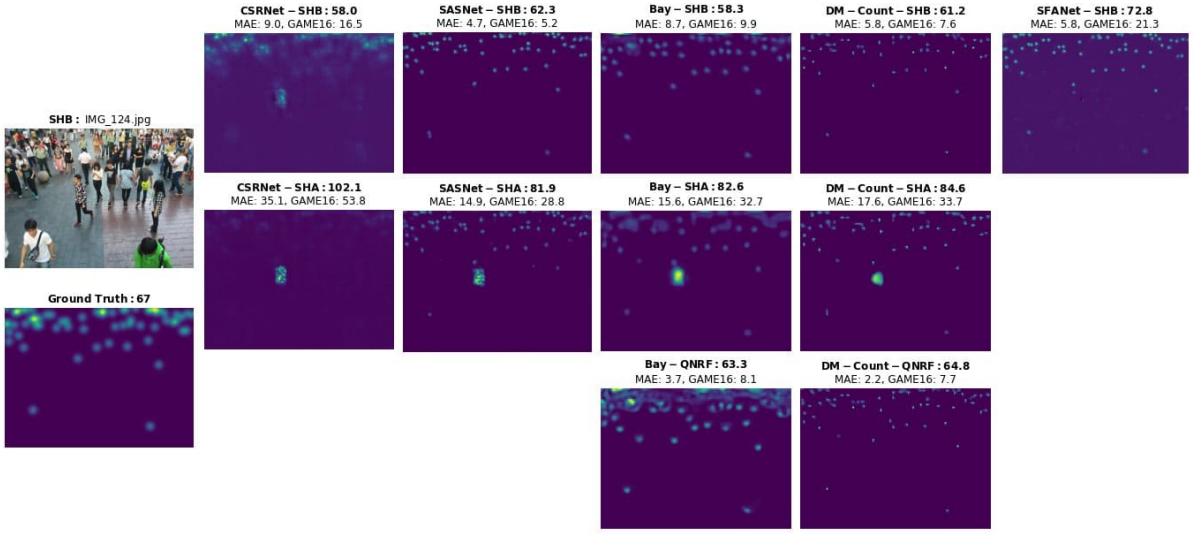


Figura 6.17: SHB (124): Imagen de baja densidad, sobreestimada por todos los modelos entrenados con *SHA*, que anotaron una agrupación de personas sobre un individuo aislado con una camisa de patrón elaborado, se destaca como *CSRNet-SHB* también detecta una agrupación pero su estimación es inferior a la cantidad de personas anotadas en la imagen.

6.2.5. Observaciones e interpretaciones respecto a los casos destacados

Considerando lo previamente observado en la sección 6.2.3 y complementando con un análisis más profundo de la visualización de casos destacados, se enumeran las siguientes observaciones y conclusiones:

- Existe mucha variación en la nitidez y estructura de los *mapas de densidad*† generados entre los modelos.
  En general, los mapas generados por *CSRNet* tienden a ser menos nítidos que los generados por *SASNet* y *DM-Count*, y los mapas generados por *BayL* son estructuralmente distintos al resto, debido al pre-procesamiento de los datos discutido en la sección 4.9.1; pese a esto, no se observa ningún patrón claro en la diferencia de *GAME*†16 respecto a la *MAE*† para los resultados generales de estos modelos (sección 6.2.3).
  Los mapas generados por *CSRNet* a veces no logran localizar a los individuos en la imagen, causando que los valores del mapa se distribuyan como si fueran ruido blanco (se presume esto ocurre mayormente para dominios desconocidos).

- Como se aprecia en las figuras destacadas del *conjunto de evaluación*† *UCF-CC50* (figuras 6.5, 6.8, 6.14, A.14, A.15, A.17 y A.19), los modelos entrenados con el conjunto *ShanghaiTech B* (*Zhang et al.*, 2016b) tienen mal desempeño en imágenes en escala de grises.
  Esto posiblemente se debe a que el *conjunto de entrenamiento*† *ShanghaiTech B* (*Zhang et al.*, 2016b) no presenta ninguna imagen en escala de grises (a diferencia de *ShanghaiTech A* (*Zhang et al.*, 2016b) y *UCF-QNRF* (*Idrees et al.*, 2018b)).
  De ser así, podría ser de valor incluir más escalas de grises en los *conjuntos de entrenamiento*† y *conjuntos de evaluación*† del problema del conteo; sin embargo, también esto se podría argumentar como innecesario, si el caso de uso del modelo aplicado a un problema fuese a ser siempre aplicándolo a imágenes a color.

- Como se aprecia en las figuras 6.3, 6.4, 6.7, 6.11, A.5 a A.8 y A.21, existen casos en los que el mejor desempeño para una imagen se obtiene con un modelo entrenado con un con un *conjunto de entrenamiento*† distinto al que se está evaluando. Esto podría deberse a que estas imágenes podrían ser categorizadas en dominios iguales o similares a los presentes en otros *conjuntos de entrenamiento*†.

- El modelo *SFANet-SHB* tiene un mal desempeño en general (nunca siendo evaluado como uno de los tres mejores modelos acorde a las pruebas realizadas en esta sección): mirando las figuras 6.2 a 6.4, A.5, A.6 y A.8 se puede apreciar como tiene un desempeño significativamente inferior al resto.
  Esto no significa, sin embargo, que su desempeño sea siempre malo. De hecho, en las figuras 6.1, A.1, A.2, A.21 y A.22 se pueden apreciar casos en los cuales el modelo tiene buen desempeño.

- Como se aprecia en las imágenes de la sección 6.2.4.4 y el apéndice A.1.4, los modelos entrenados con *SHA*, *SHB* y *UCF-QNRF* no estiman personas detrás de *oclusiones*†.
  Esto en gran parte se debe a que las anotaciones para sus respectivos *conjuntos de entrenamiento*† se restringen a sólo anotar cabezas visibles.

- En varias ocasiones los modelos entrenados con *SHB* tienden a infraestimar la cantidad de personas en una imagen, en comparación a los resultados de modelos entrenados con *SHA* o *UCF-QNRF*. Ejemplos de esto se pueden apreciar en las figuras 6.3, 6.7, 6.12, A.7, A.12 y A.13; excepciones de esto se pueden apreciar en las figuras A.2, A.3, A.5 y A.6.
  Esto podría deberse a que el *conjunto de datos*† *ShanghaiTech B* tiene un promedios significativamente menor de personas entre sus instancias ($\mu = 123$) en comparación a *ShanghaiTech A* (con $\mu = 501$) y a *UCF-QNRF* (con $\mu = 815$).

- Imágenes que fueron de baja varianza en cuanto a anotaciones en el conjunto *MT* (consideradas fáciles de etiquetar) no necesariamente fueron fáciles de estimar para los modelos pre-entrenados. Ejemplos de esto son las figuras 6.15, A.9 y A.16.

- Imágenes ya previamente identificadas como de gran dificultad para anotar (como las del conjunto *MT paraguas* y *MT aéreo*) fueron efectivamente de gran dificultad para los modelos pre-entrenados, ya sea por las anotaciones detrás de *oclusiones*†, la iluminación, o la *variación de escala*†.
  Estos resultados tienen sentido dado que los *conjuntos de datos*† que se usaron en el *entrenamiento*† de los modelos evaluados no contienen anotaciones detrás de *oclusiones*†, así como tampoco contienen imágenes aéreas (como por ejemplo los *conjuntos de datos*† capturados con drones que fueron mencionados en la sección 3.1).

- Al evaluar el desempeño para casos puntuales (tanto en los casos destacados, como en la preselección de los mismos), los modelos pre-entrenados con las arquitecturas *CSRNet*, *SASNet*, *Bay* y *DM-Count* presentan en general un buen desempeño consistente a la hora de estimar la cantidad de personas en imágenes de multitudes sencillas (sin grandes dificultades, y que no son atípicas). Ejemplos de esto se pueden apreciar en las figuras 5.21, 6.1 y A.2 a A.6.

- Existen situaciones que dificultan particularmente el desempeño de algunos modelos, como el que las imágenes presenten elementos en su *fondo complejo*† (como árboles) o detalles de individuos (como su ropa). Estas llevan a que el modelo detecte una cantidad de personas donde no las hay, como se puede apreciar en la sección 6.2.4.6 y el apéndice A.1.6.
  Si bien estos casos fueron muy pocos en relación al resto de visualizaciones apreciadas, los mismos dejan claro que estos modelos no son libres de falsos positivos o de generar artefactos en base a ciertos valores en las imágenes.
  Se destaca como si bien estos errores estuvieron presentes en modelos entrenados con varios *conjuntos de datos*† (ver figuras 6.16, 6.17 y A.26), ocurrieron con mayor frecuencia para modelos entrenados solamente con el conjunto *SHA*, como se puede apreciar en las figuras A.23 a A.25.

Capítulo 7

Conclusión

En esta sección se enumeran los resultados alcanzados y las dificultades encontradas a lo largo de esta investigación. También se realiza una autocrítica de las fallas cometidas, se enumeran algunas lecciones aprendidas, y las posibles direcciones en las que este estudio podría ser continuado.

7.1. Síntesis del estudio

Al inicio de este trabajo, tanto los estudiantes como los tutores tenían muy poco conocimiento en el área de conteo de multitudes (contando con conocimientos previos en el área más general de la visión artificial).
Este trabajo se planteó con el objetivo de:

- Desarrollar los conocimientos de visión computacional de los estudiantes.
- Estudiar y analizar distintos enfoques existentes en la literatura para abordar el problema de conteo de multitudes.
- Compilar la evolución del conteo de multitudes a lo largo de los años.
- Analizar el estado del arte actual, determinando cuáles son sus fortalezas, carencias y direcciones en las que podría llegar a avanzar.
- Identificar áreas poco exploradas en la evolución del conteo, que puedan estar sesgando la dirección del estado del arte sin ser consideradas por los investigadores. Por ejemplo la calidad de las anotaciones en los *conjuntos de datos*† existentes.
- Participar en la organización y ejecución de la anotación de un *conjunto de datos*† de pequeño porte y realizar un análisis estadístico sobre la calidad de anotaciones adquiridas.
- Estudiar el grado de *adaptación al dominio*† y la capacidad de generalización de algunos modelos que forman el estado del arte.
- Evaluar mediante la experimentación la aplicabilidad de modelos de conteo de multitudes en distintos dominios.

Se considera que estos objetivos fueron alcanzados, destacando particularmente que:

- La sección 4.4 y el glosario C plasman el conocimiento adquirido en el área de visión artificial, complementando los conocimientos previos del alumno.
- La sección 2 define el conteo de multitudes, enumerando también las dificultades intrínsecas al problema, mientras que los recursos de apoyo a la investigación del área se compilan en la sección 3.
- Las secciones 4.1, 4.2, 4.3 y 4.5 se centran en el conteo de multitudes a lo largo de su historia y definen conceptos generales del problema, como son las *métricas de evaluación*†.
- Las secciones 4.6 y 4.7 evalúan las dos grandes arquitecturas contemporáneas, multicolumna y de una sola columna, las cuales han sido focales en la investigación para modelos de conteo, centrando el resto del estudio en la arquitectura de una sola columna, que es la que más relevancia ha tenido en los últimos años.
- Las secciones 4.8 y 4.9 profundizan en problemas, técnicas y módulos que trabajos anteriores han investigado y propuesto para mejorar el desempeño de los modelos. Como son las técnicas para atacar la *variación de escala*† y los trabajos basados en funciones de pérdida probabilísticas.
- La sección 4.10 plasma el estado del arte actual, mientras que la sección 4.11 complementa la investigación, buscando cubrir las áreas de reciente interés experimental.
  Se destacan particularmente los estudios de la capacidad de generalización en modelos (sección 4.11.1), pero también es de interés la adopción de *transformers*† en problemas de conteo (sección 4.11.2), así como la reciente incursión en el *aprendizaje débilmente supervisado*† (sección 4.11.3).
- El apéndice B registra arquitecturas, técnicas y heurísticas que fueron evaluadas y/o estudiadas en la investigación, pero su profundización y/o inclusión quedaron por fuera del alcance de este estudio.
- Se analizaron en profundidad numerosos textos, a saber:
  - 3 resúmenes y compilaciones del conteo de multitudes a lo largo de los años (*Khan et al.*, 2022; *Gao et al.*, 2020; *Gong*, 2020).
  - 4 artículos de modelos que representaron un punto de inflexión en el área del conteo de multitudes (*Zhang et al.*, 2016a; *Babu Sam et al.*, 2017; *Li et al.*, 2018; *Song et al.*, 2021).
  - 12 artículos de modelos contemporáneos, incluyendo cinco que fueron empleados en la experimentación de la sección 6) *Li et al.* (2018); *Zhu et al.* (2019); *Ma et al.* (2019); *Wang et al.* (2020a); *Song et al.* (2021), otros cinco que fueron de gran ayuda para desarrollar un conocimiento más profundo sobre el área (*Liu et al.*, 2018b; *Sindagi and Patel*, 2019; *Yan et al.*, 2019a; *Bai et al.*, 2020), y finalmente dos que proponen la aplicación de heurísticas para intentar mejorar, optimizar o agregar otras funcionalidades a los modelos de conteo (*Oh et al.*, 2019; *Hu et al.*, 2020b).
  - 3 artículos que analizan la capacidad de generación de modelos a través del modelado de dominios y/o el *entrenamiento*† combinando distintos *conjuntos de entrenamiento*† (*Yan et al.*, 2021; *Chen et al.*, 2021; *Du et al.*, 2022).
- Se analizaron superficialmente una gran cantidad de artículos, ya sea buscando profundizar en conceptos citados por artículos estudiados, validando el contexto de los mismos, o complementando la investigación.
- Se generó un *conjunto de datos*† *MT*, publicado en *el repositorio público de este trabajo* (`https://github.com/renzodgc/fing-crowdcounting`). La sección 5 documenta la estructura y anotación del mismo, así como presenta un análisis estadístico de las 12 anotaciones obtenidas para cada imagen del conjunto. Dicho análisis llega a concluir que siempre existe disparidad entre las anotaciones, especialmente cuando estas refieren a imágenes de gran dificultad.

- Se programó y ejecutó un notebook en *Google Colab*, publicado en *el repositorio público de este trabajo* (`https://github.com/renzodgc/fing-crowdcounting`). La sección 6 presenta la experimentación realizada, los resultados obtenidos evaluando 5 arquitecturas (11 modelos pre-entrenados) en 5 *conjuntos de evaluación*†, y las conclusiones que se extrapolaron de dichos resultados.
  El apéndice A incluye más visualizaciones de instancias desatacadas que refuerzan y complementan dicha sección.

Pese a los resultados alcanzados, se destacan las siguientes dificultades y errores que comprometieron la calidad de este trabajo:

- Unos meses después de su arranque, el proyecto comenzó a enlentecer su ritmo, con los estudiantes disminuyendo notoriamente la cantidad de horas dedicadas. Esto fue particularmente perjudicial para la estrategia de investigación *bottom-up* exhaustiva que se había resuelto tomar.
  Al inicio del proyecto el objetivo claro era adentrarse en el problema del conteo de multitudes, y una vez se tuviera un entendimiento terminar de definir el alcance del proyecto, sus objetivos y entregables; trece meses tras su arranque el proyecto tenía gran parte de su investigación realizada y un alcance tentativo definido, pero el informe estaba en etapas muy tempranas, así como el código de la experimentación.

- Tres de los cuatro estudiantes iniciales desistieron del proyecto a los 13 meses de su arranque (octubre de 2022), debido a que no tenían tiempo para o intención de priorizar el mismo.
  Esto tuvo un enorme impacto en el proyecto. Se tuvo que repensar el alcance del mismo, buscando no desechar los recursos generados (investigación e inicio de ejecución del módulo de taller) sin a su vez comprometer la calidad del proyecto.

- La selección de imágenes y política de etiquetado en el módulo de taller impactaron en la calidad del conjunto *MT* y el estudio del mismo.
  Por una parte, si bien se considera de interés al estudio la política de etiquetar detrás de *oclusiones*†, esto agrega mucho ruido a las anotaciones y causa peor desempeño en las *métricas de evaluación*† en modelos pre-entrenados con *conjuntos de entrenamiento*† populares, dado que los mismos no anotan cabezas detrás de *oclusiones*†.
  A su vez, algunas imágenes seleccionadas fueron de una dificultad muy alta, lo cual vuelve muy difícil su anotación (como se puede apreciar en la sección 5.3). Una de las mayores dificultades se vio en las fotos aéreas, que debido a la nitidez de la foto y distancia respecto a las personas en el fondo, cuesta distinguir individuos entre la multitud.
  Estos dos factores dificultan generalizar el análisis del estudio estadístico de anotaciones del conjunto *MT*, y el desempeño evaluado de modelos pre-entrenados en el mismo. Se destaca cómo partir el conjunto en cuatro sub-conjuntos fue de gran ayuda para interpretar los resultados y sacar conclusiones.

- Al ejecutar la experimentación, evaluando con los modelos de la biblioteca *LWCC* y con *SASNet*, se optó por emplear como *valor anotado*† las anotaciones del conjunto *MT* anotadas por el estudiante. Viendo la alta varianza en anotaciones, esta decisión pudo haber sesgado los resultados de las *métricas de evaluación*† en la sección 6.9.
  Un elemento que quedó fuera del alcance del proyecto es la combinación por parches de múltiples anotaciones para una imagen, lo que hubiera permitido usar todos los *valores anotados*† en la *evaluación*†. Este estudio podría ser de interés para trabajos futuros.

7.2. Conclusiones generales

Se enumeran las siguientes conclusiones y lecciones aprendidas a lo largo de este estudio:

- El problema del conteo de multitudes ha tenido un avance consistente a lo largo de la última década. El mismo, sin embargo, está particularmente atado a varios sesgos (la mayoría impuestos con el propósito de reducir el esfuerzo de experimentación y enfocar la misma a tener comparación directa con estudios previos) como por ejemplo:
  - Lo normal es entrenar modelos con un solo *conjunto de entrenamiento*†, para evaluar en el *conjunto de evaluación*† del mismo *conjunto de datos*†. Esto causa que los modelos sean más susceptibles a la *adaptación al dominio*†.
    Recientemente se ha invertido esfuerzo en investigar la capacidad de generalización de los modelos y como el aprendizaje multidominio podría incluso mejorar el desempeño en todos los dominios.
  - La gran mayoría de estudios emplean solo una minoría de los *conjuntos de datos*† publicados, restringiéndose a los comúnmente usados en el *entrenamiento*† y la *evaluación*†. Esto mismo se aplica también a las *métricas de evaluación*† capturadas, siendo generalmente sólo *MAE*† y *RMSE*†.
  - Varios artículos emplean heurísticas o técnicas concisas (a veces de gran complejidad) que en general causan una mejora en cuanto al desempeño del modelo en las *métricas de evaluación*† (*Khan et al.*, 2022). Sin embargo, muy pocas de estas terminan siendo adoptadas por trabajos futuros, en general debido a que su mejora en las *métricas de evaluación*† es baja o también a favor de mantener la simplicidad de los modelos.
- El conteo de multitudes (o su generalización al problema de conteo) está en etapas tempranas para su aplicabilidad en problemas de ingeniería o productos comerciales, por varios motivos:
  - Si bien los modelos en general dan un estimativo aceptable para determinar el **nivel** de aforo en una imagen, son varios los casos en donde esta estimación no es de confianza.
  - Son muy recientes los estudios de la *adaptación al dominio*† de los modelos que dirigen el estado del arte. Si bien al implementar un modelo en una escena fija es deseable que el mismo maximice su desempeño en dicho dominio (estando adaptado a la escena), debido a la complejidad de los dominios y *conjuntos de datos*† es difícil saber cuál será el mejor modelo para una escena, teniendo que probablemente realizar una *evaluación*† con datos capturados en la misma.
    Algunos estudios proponen de hecho reiterar en el entrenamiento de un modelo pre-entrenado, aplicando un *fine-tuning* que mejore su desempeño entrenando con imágenes anotadas de la escena en donde se desee aplicarlo (*Du et al.*, 2022).
  - No se ha destinado suficiente esfuerzo a la *evaluación*† en cuanto a la correctitud de los *conjuntos de datos*† disponibles (ignorando por completo cuanto ruido pueda haber en sus anotaciones) debido a la dificultad de la tarea que es anotar los mismos.
  - No se han realizado trabajos que evalúen el impacto del tiempo de ejecución de los modelos, y que busquen optimizar y lograr disminuir el mismo (a costa de un menor desempeño en su

precisión). Estos trabajos son de gran importancia a la hora de querer integrar un modelo a, por ejemplo, la salida de cámaras de seguridad, en donde podría ser necesario contar con un modelo ligero capaz de correr en tiempo real en hardware de poca potencia (tales como dispositivos edge).

- Las arquitecturas y modelos no están en un punto de gran estabilidad en el campo del conteo de multitudes. Existe hoy en día una cantidad bastante heterogénea de técnicas con las que artículos experimentan para mejorar el estado del arte; sin embargo, estas siguen diversas direcciones, lo cual dificulta solidificar e iterar en patrones de arquitectura (*Khan et al.*, 2022). Ejemplos son:
  - La aplicación de *transformers*† sección 4.11.2, que si bien han revolucionado el área del PLN, todavía no se ha encontrado la forma de aplicarlos al área de conteo, teniendo arquitecturas de gran profundidad y desempeño levemente inferior respecto al estado del arte.
  - La exploración del *aprendizaje no supervisado*† o semi-supervisado (sección 4.11.3), que es de gran interés para el entendimiento de dominios, y podría ampliar enormemente los *conjuntos de datos*† disponibles.
  - La resurgencia de las arquitecturas multicolumna o híbridas (apéndice B.2), que buscan aplicar técnicas contemporáneas a otras estrategias de arquitectura.
  - El aprendizaje multidominio y el foco en evaluar que los modelos entrenados se puedan adaptar a varios dominios generales que desconozcan (sección 4.11.1), que es clave en la aplicación de modelos de conteo en productos finales.
- El problema de conteo, si bien es de nicho, puede ser de gran interés en el futuro por los siguientes motivos:
  - Es de gran utilidad para capturar datos históricos que permitan evaluar y rediseñar ciudades en el área del urbanismo.
  - Puede ser integrado en productos que controlen el aforo de edificios o lugares públicos, como por ejemplo estadios, playas y cruces peatonales.
  - Una enorme parte del avance en el área del conteo de multitudes es agnóstico a las entidades que se estén anotando en los *conjuntos de datos*†, dando lugar a otros nichos que apliquen modelos de conteo como por ejemplo la agricultura, la ganadería, las migraciones de fauna, y el análisis de muestras de laboratorio.

7.3. Trabajo a futuro

Se considera que este proyecto es un muy buen recurso para la inducción de estudiantes a la visión computacional y al problema del conteo de multitudes. Múltiples aristas no pudieron ser cubiertas en este estudio, sea porque requerían de una mayor cantidad de recursos, tiempo o conocimiento, o porque fueron recortadas del alcance debido a la fragmentación del grupo e imposibilidad de disponer de recursos con los que paralelizar tareas.

A continuación se enumeran las posibles direcciones en las que este trabajo podría ser usado como base para desarrollar o complementar para futuros estudios en el área del conteo:

- Profundizar el estudio de la generalización en los modelos, ampliando la experimentación a más *conjuntos de evaluación*†, y ampliando la experimentación a más modelos, en particular modelos que hayan sido entrenados con varios conjuntos de datos (ver sección 4.11.1.1).
- Incursionar en el área del aprendizaje semi-supervisado, con el fin de mejorar el modelado de dominios y sub-dominios en la reciente área de aprendizaje del dominio general o aprendizaje basado en múltiples dominios.
- Complementar la información presentada con un estudio en mayor profundidad de los *transformers*† y su aplicación al área del conteo, con el fin de experimentar y proponer un modelo, entrenarlo y evaluar su desempeño.
- Contribuir a la biblioteca *LWCC*, agregándole más modelos y dando soporte a que corran en GPU.
- Integrar y aplicar un modelo de conteo de multitudes en un producto, el cual podría ser optimizado con técnicas de cuantificación o destilación con el fin de volverlo más ligero y disminuir su tiempo de ejecución en tiempo real o disminuir los recursos que necesite.
- Experimentar con transferir el conocimiento de modelos de conteo de multitudes a otras áreas del conteo (como son la agricultura o la ganadería).
- Anotar otro *conjunto de datos*†, seleccionando imágenes de menor dificultad y empleando una estrategia de anotación que se base en tener más y menor cantidad de imágenes por anotador. Luego de esto, se podría emplear una experimentación similar a la presentada en este trabajo, o expandirla aún más comparando las distribuciones en las matrices de puntos anotados (empleando por ejemplo *distancias entre densidades*, como la *divergencia Kullback-Liebler* (*Hershey and Olsen*, 2007) y la *distancia de Wasserstein* (*Delon and Desolneux*, 2020)), o por ejemplo aplicar algún algoritmo de *clusterización*† como *K-Means*† y comparar la posición de los centroides.
  También se podría anotar un *conjunto de datos*† previamente anotado (como ShanghaiTech (*Zhang et al.*, 2016b)) y comparar las anotaciones de los estudiantes con las anotaciones oficiales.
- Entrenar un modelo propuesto (o arquitecturas del estado del arte) en nuevos *conjuntos de entrenamiento*† empleando técnicas de aprendizaje multidominio.
- Contribuir y mejorar el proyecto abierto *NWPU-Crowd Sample Code* (*Wang et al.*, 2020c), o crear un nuevo recurso para entrenamiento. Sería deseable que el mismo soporte más modelos y más *conjuntos de datos*†, así como que tenga una interfaz más parametrizable para entrenamientos basados en notebooks (como en *Google Colab*) en lugar de depender de archivos de configuración.

Apéndice A

Apéndice: Resultados adicionales de la experimentación

Este apéndice tiene como fin complementar y ampliar los resultados presentados en la sección 6.

A.1. Visualización de casos destacados

A continuación se agrupan y presentan una extensión de la visualización del resultado de modelos en casos destacados presentada en la sección 6.2.4. Al igual que en dicha sección, las imágenes se agrupan en las siguientes secciones:

- Estimaciones de calidad aceptable (sección A.1.1)
- Estimaciones de calidad mixta (sección A.1.2)
- Estimaciones de mala calidad (sección A.1.3)
- Efecto de la oclusión en imágenes (sección A.1.4)
- Ejemplos de otras particularidades (sección A.1.5)
- Ejemplos de falsos positivos u artefactos en mapas de densidad generados (sección A.1.6)

A.1.1. Ejemplos de estimaciones de calidad aceptable

Se consideran de calidad aceptable a los resultados que varios modelos tuvieron un error menor al del 10 % de la cantidad anotada de personas (figuras A.1, A.2, A.3, A.4, A.5 y A.6).

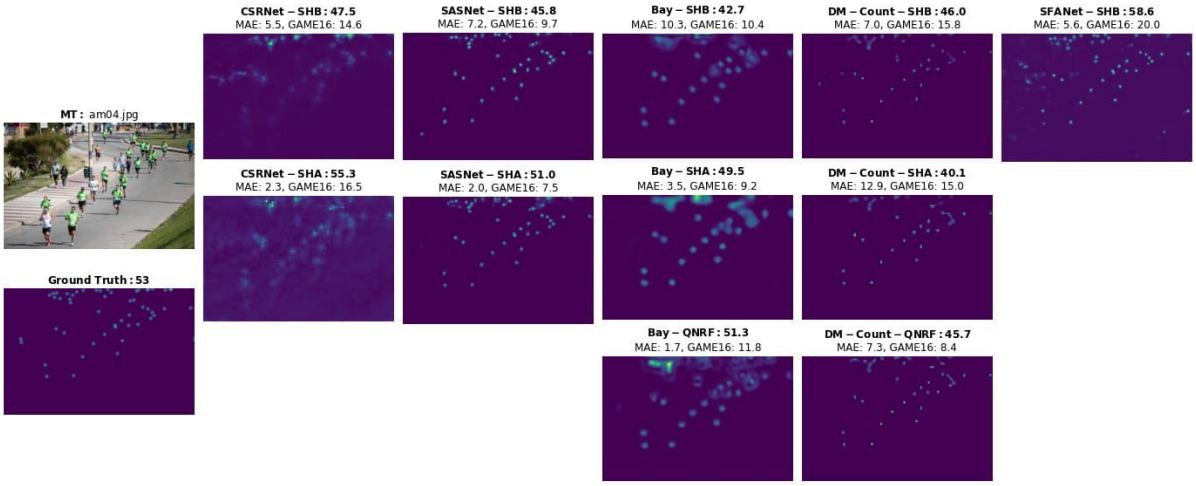


Figura A.1: MT-baja (am04): Imagen de baja dificultad, los *mapas de densidad*†, la diferencia entre la *MAE*† y *GAME*†16 parece indicar que los modelos no localizaron bien a los individuos en la imagen.

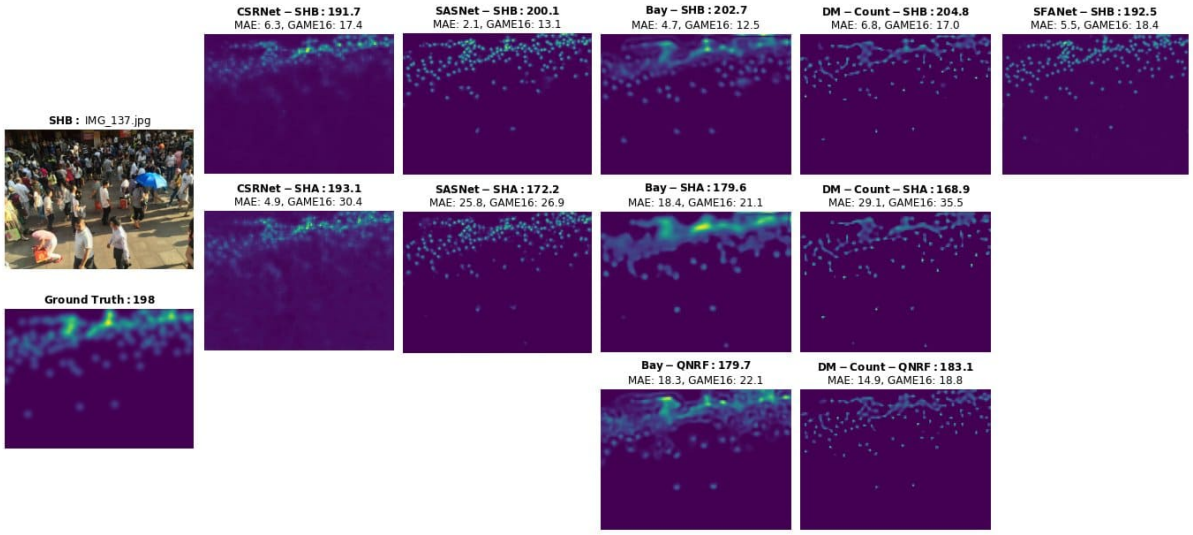


Figura A.2: SHB (137): Imagen de baja dificultad, los modelos entrenados con su *conjunto de entrenamiento*† (SHB) tuvieron mejor desempeño.

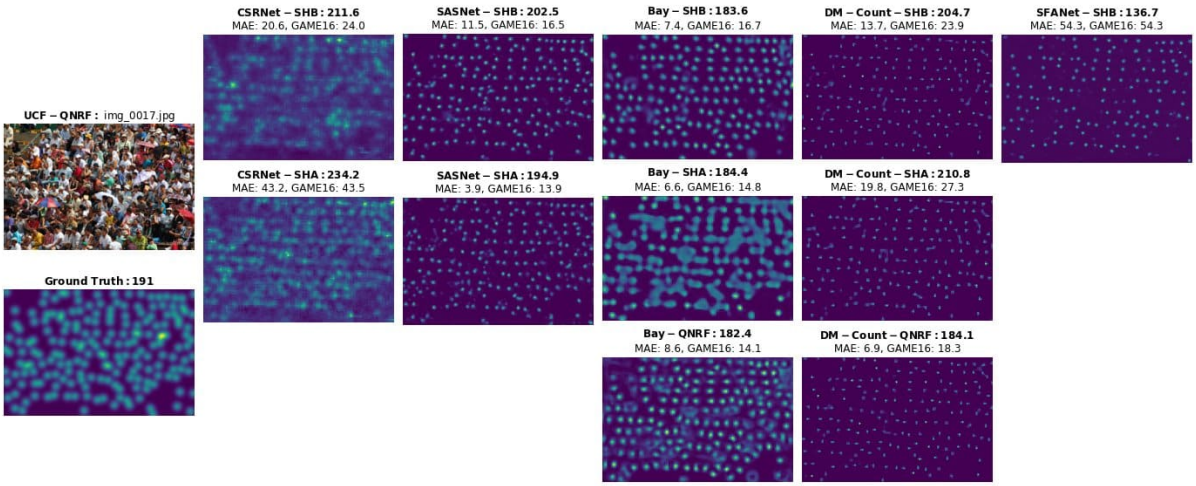


Figura A.3: UCF-QNRF (17): Imagen de densidad homogénea, en general los modelos tienen un buen desempeño, y se aprecian las diferencias en sus *mapas de densidad*[†] .

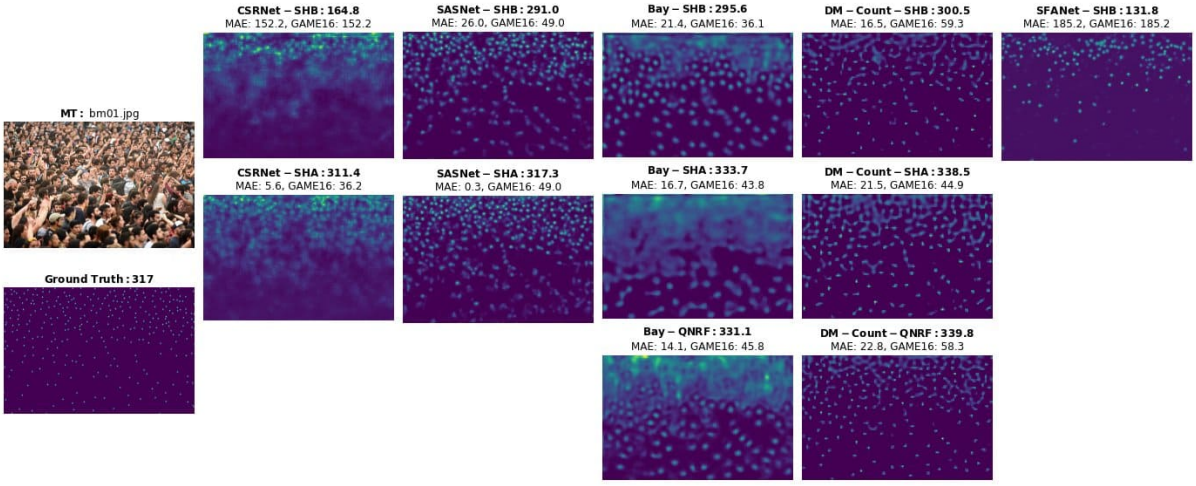


Figura A.4: MT-media (bm01): Esta imagen de baja dificultad de etiquetado dio buenos resultados como se esperaba, salvo para *CSRNet-SHB* y *SFANet-SHB*; se destaca cómo otros modelos entrenados también con *SHB* sí tienen buen desempeño.

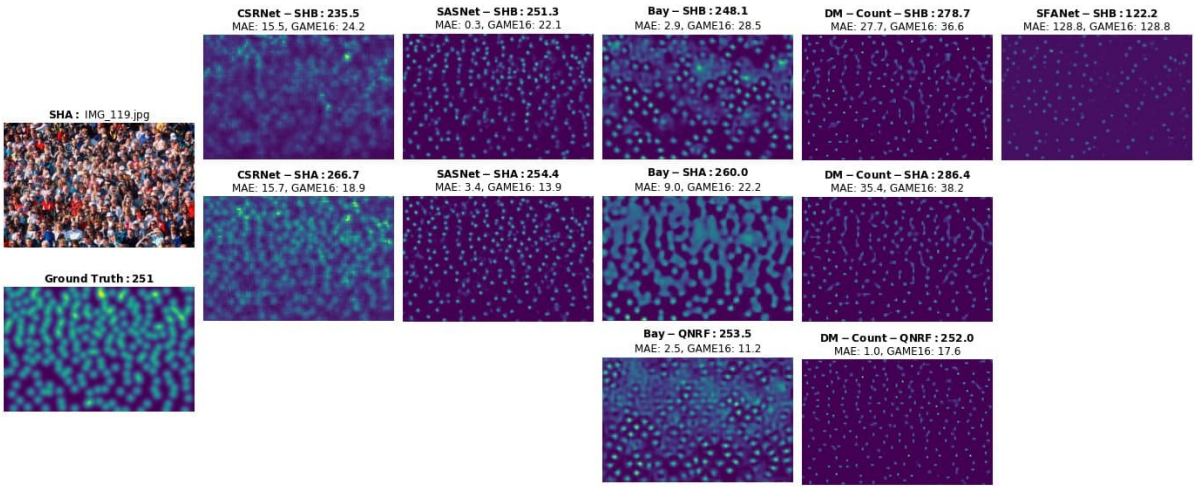


Figura A.5: SHA (119): Imagen de densidad homogénea y baja dificultad; *SASNet-SHB* tiene excelente desempeño.

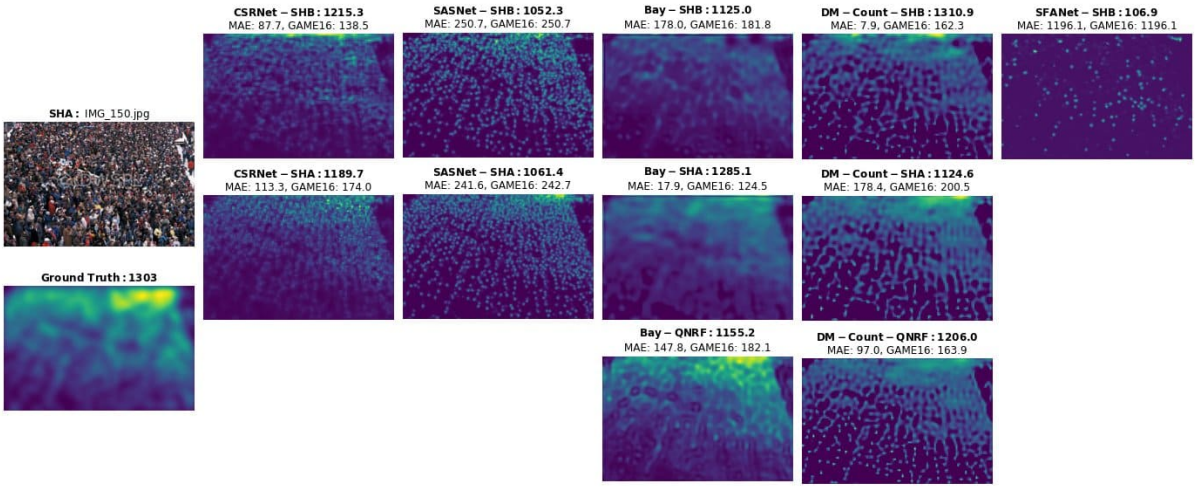


Figura A.6: SHA (150): Imagen con muy buen desempeño el tamaño de la multitud en la misma, excepto por el modelo *SFANet-SHB*.

A.1.2. Ejemplos de estimaciones de calidad regular

Se consideran de calidad regular a los resultados que varios modelos tuvieron un error entre el 10 % y 25 % de la cantidad anotada de personas (figuras A.7, A.8, A.9, A.10 y A.11).

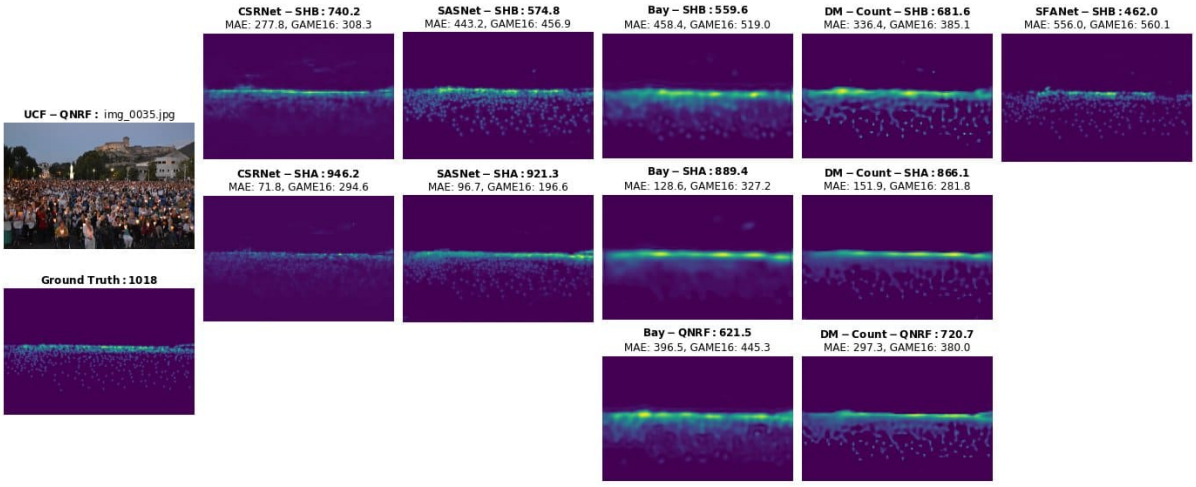


Figura A.7: UCF-QNRF (35): Imagen infraestimada, se destaca cómo siempre los modelos entrenados con *SHA* estimaron más personas, obteniendo mejor desempeño que el resto.

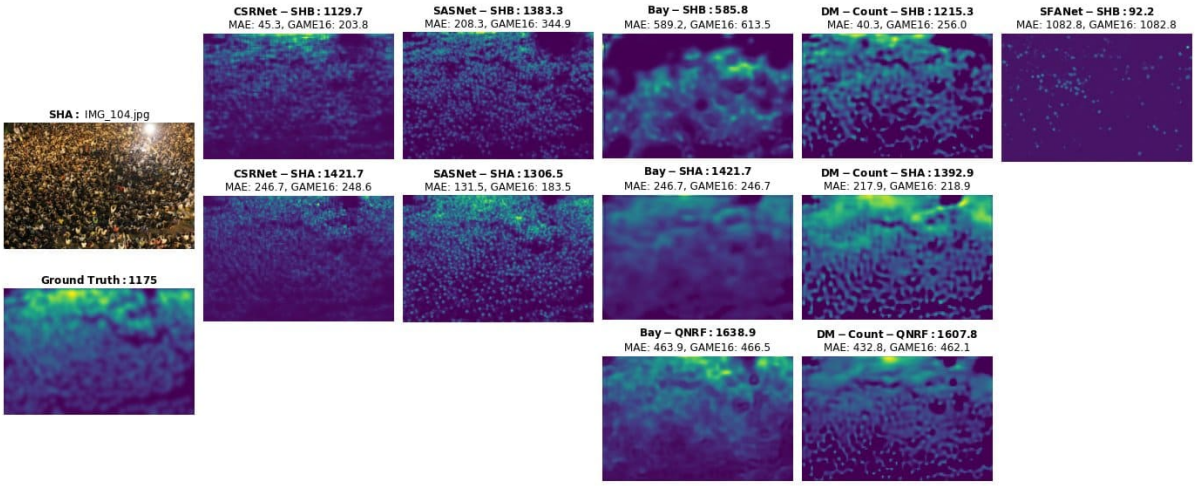


Figura A.8: SHA (104): Resultados en general sobreestimados por un margen aceptable.

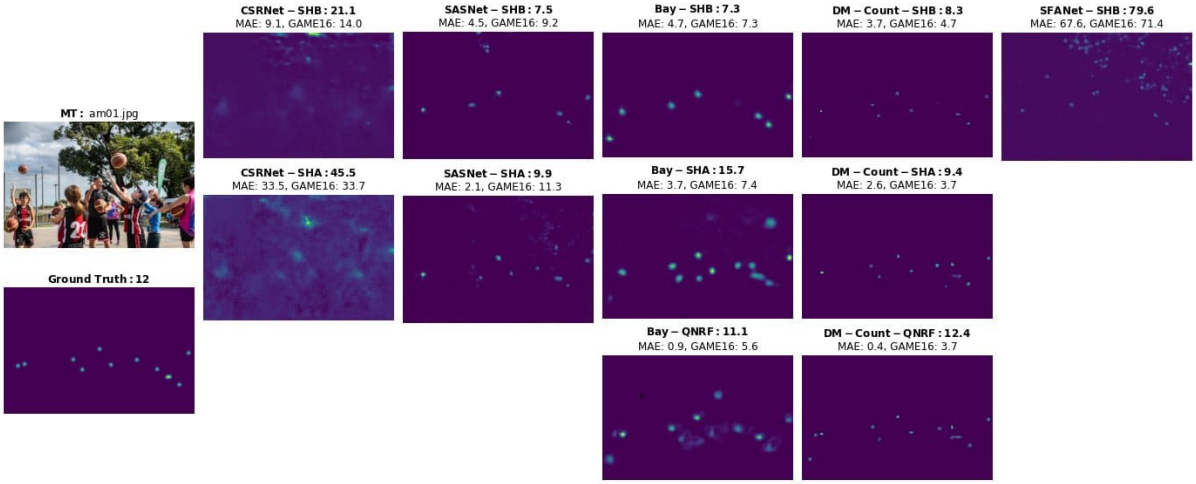


Figura A.9: MT-baja (am01): Esta imagen de baja dificultad tuvo buenos resultados para algunos modelos, pero *CSRNet-SHA*, *CSRNet-SHB* y *SFANet-SHB* sobre-estimaron la poca cantidad de personas en la imagen.

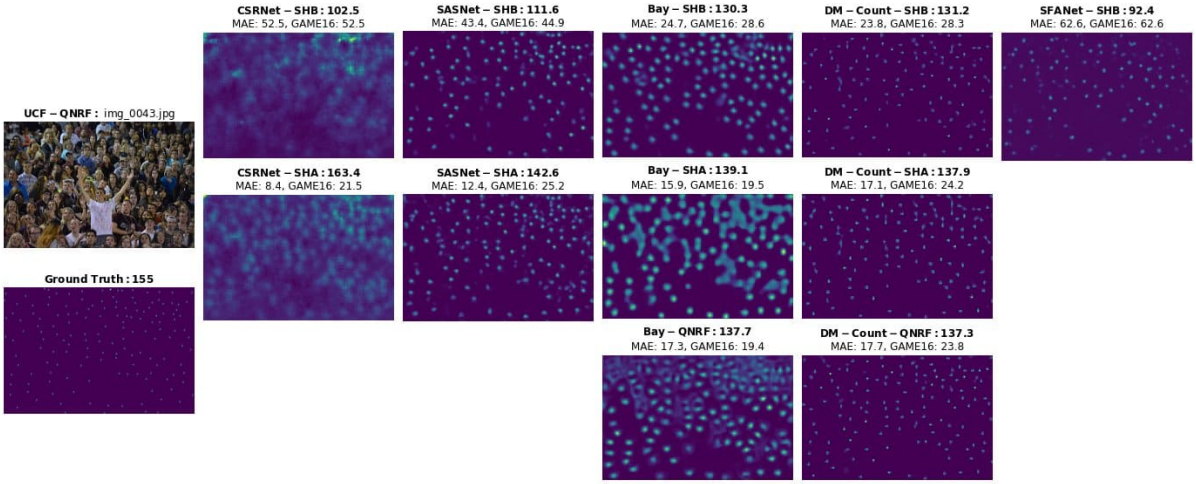


Figura A.10: UCF-QNRF (43): Resultados aceptables, pero debido a la baja dificultad de la imagen, se esperaba un mejor desempeño.

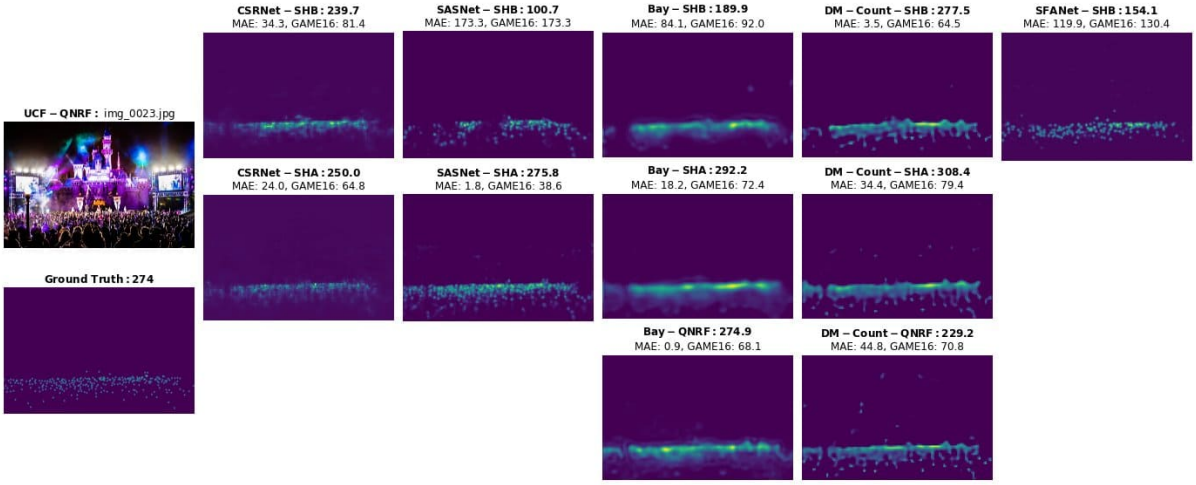


Figura A.11: UCF-QNRF (23): Imagen con variabilidad significativa en su iluminación, pese a esta condición los modelos tienen un desempeño razonablemente bueno, independientemente a su *conjunto de entrenamiento*[†].

A.1.3. Ejemplos de estimaciones de mala calidad

Se consideran de calidad mala a los resultados que varios modelos tuvieron un error mayor al 25 % de la cantidad anotada de personas (figuras A.12, A.13, A.14, A.15 y A.16).

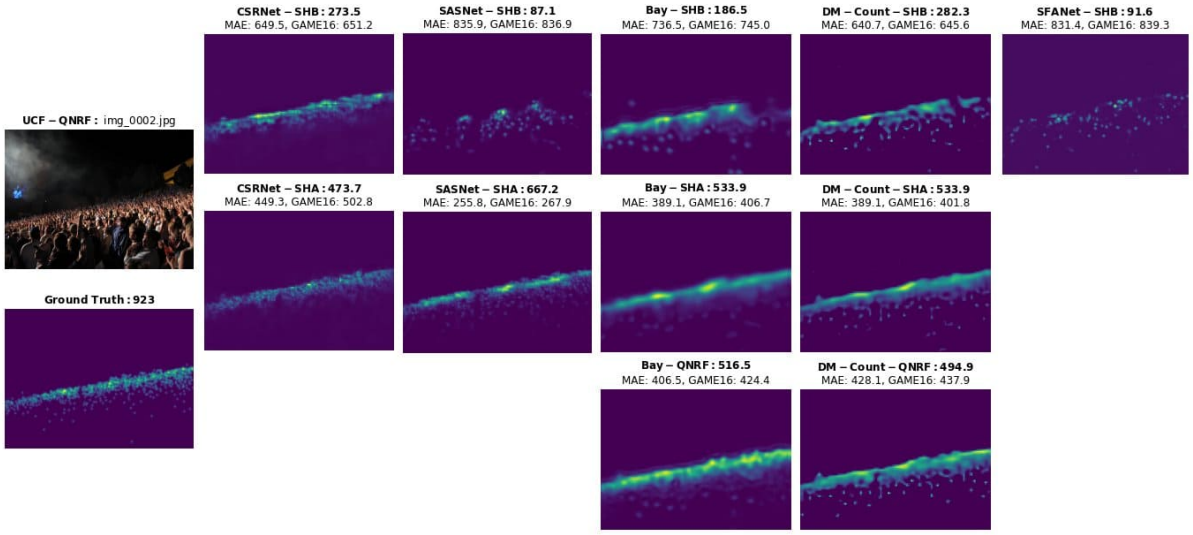


Figura A.12: UCF-QNRF (2): Imagen difícil por su iluminación, es infraestimada por todos los modelos, particularmente los entrenados con *SHB*.

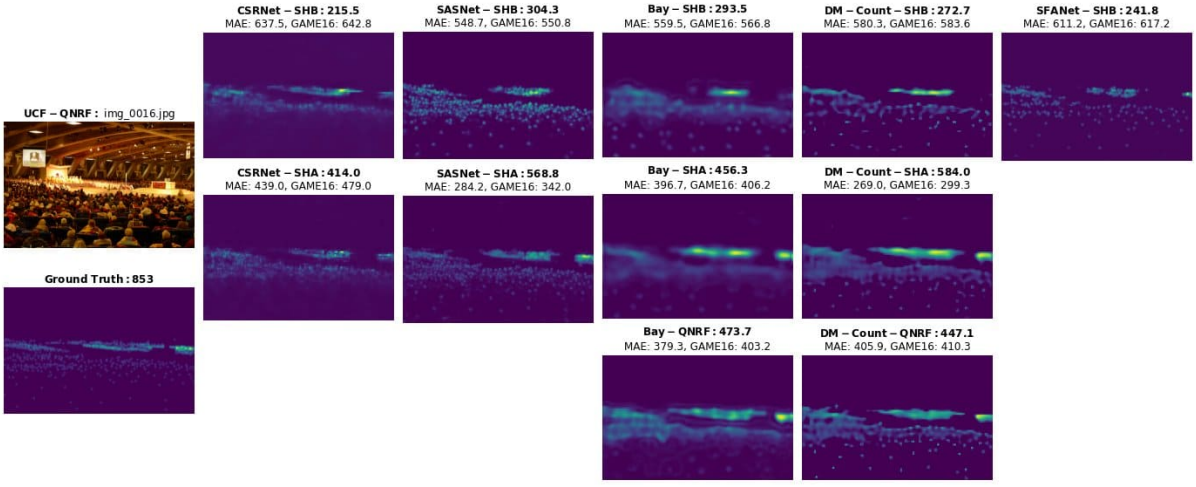


Figura A.13: UCF-QNRF (16): Imagen infraestimada por todos los modelos, particularmente los entrenados con *SHB*.

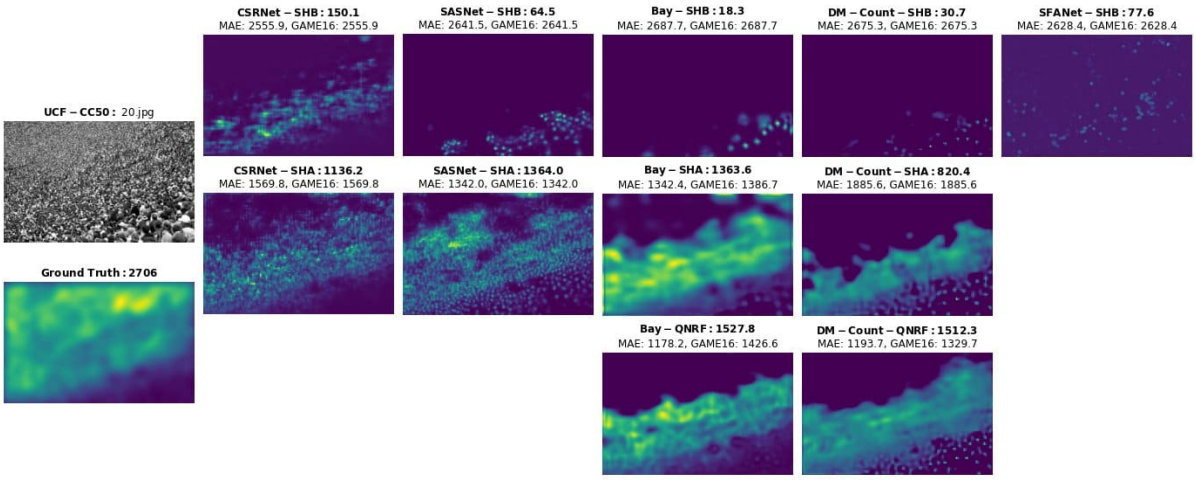


Figura A.14: UCF-CC50 (20): Imagen en escala de grises de muy alta dificultad, infraestimada por todos los modelos, particularmente con los entrenados con *SHB*

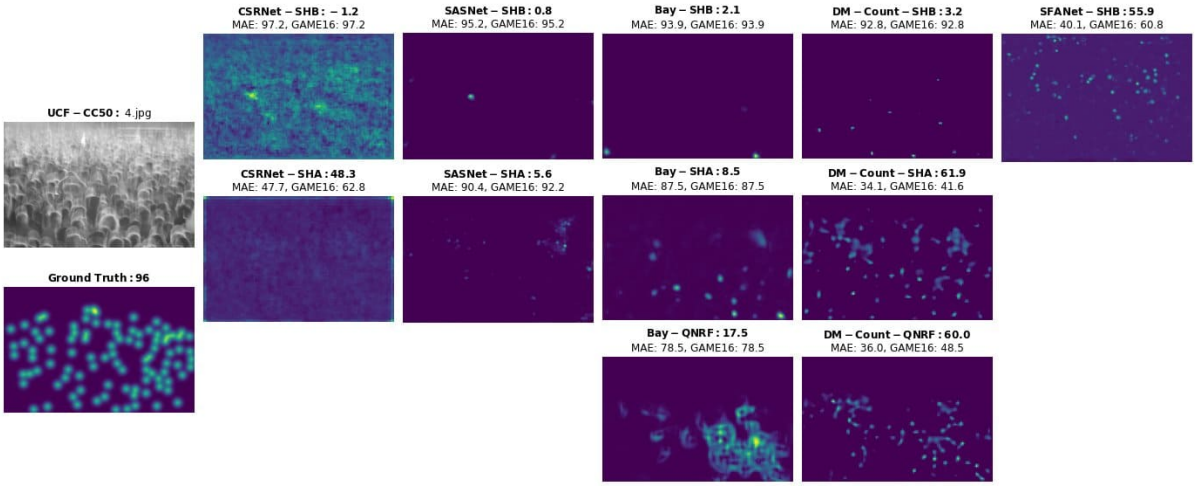


Figura A.15: UCF-CC50 (4): Imagen en escala de grises y borrosa, de baja densidad de personas. Los modelos en general no tuvieron buen desempeño.

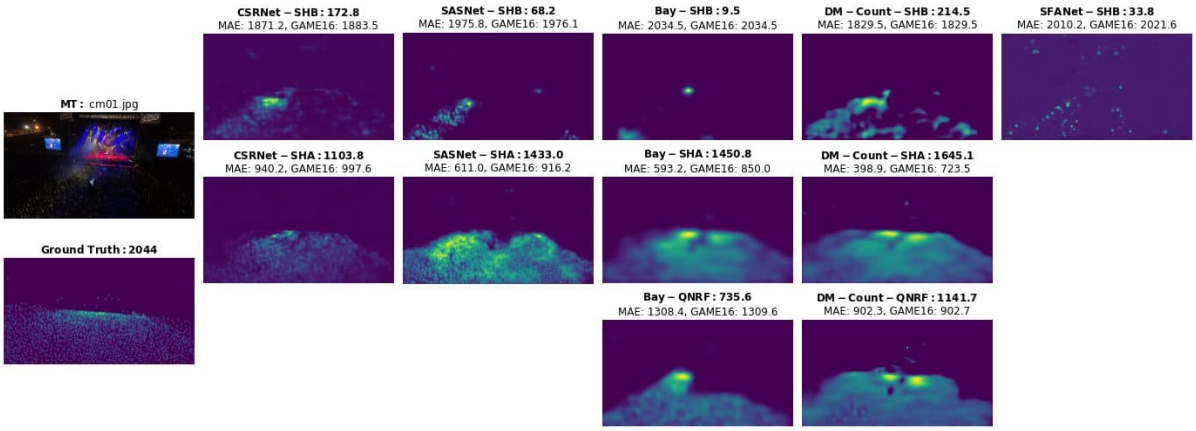


Figura A.16: MT-aéreo (cm01): Imagen que pese a ser la segunda imagen con mejor varianza para el conjunto *MT aéreo* (ver sección 5.4.1.1), fue de gran dificultad para los modelos pre-enetrenados, que infraestimaron la cantidad de personas en la multitud, especialmente los entrenados con *SHB*.

A.1.4. Ejemplos del efecto de oclusión en imágenes

En esta sección se destaca como los modelos *no hacen estimaciones de personas detrás de oclusiones*† (figuras A.17 y A.18).

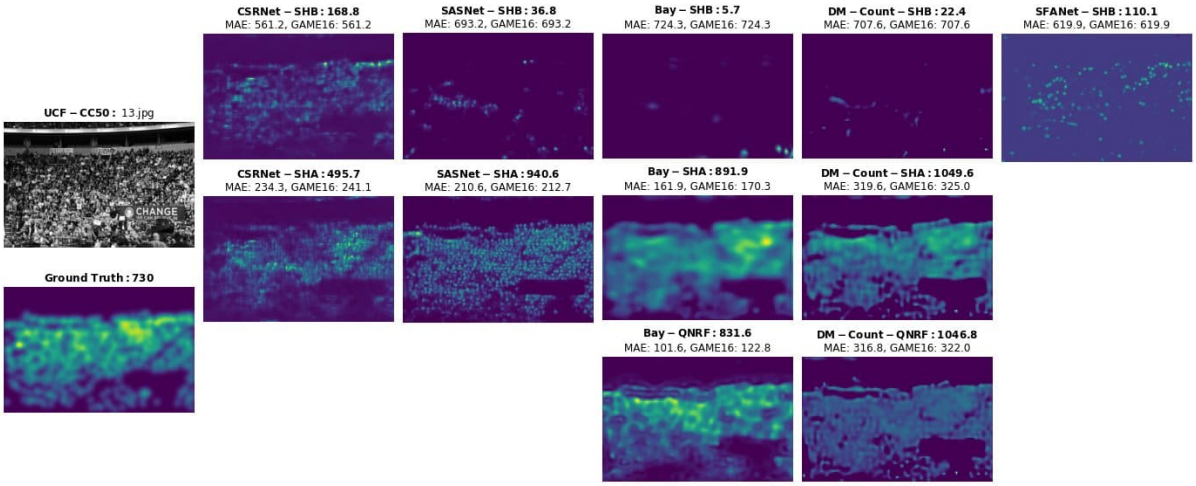


Figura A.17: UCF-CC50 (13): Imagen con una densidad no uniforme, los modelos no estiman la presencia de gente detrás del cartel.

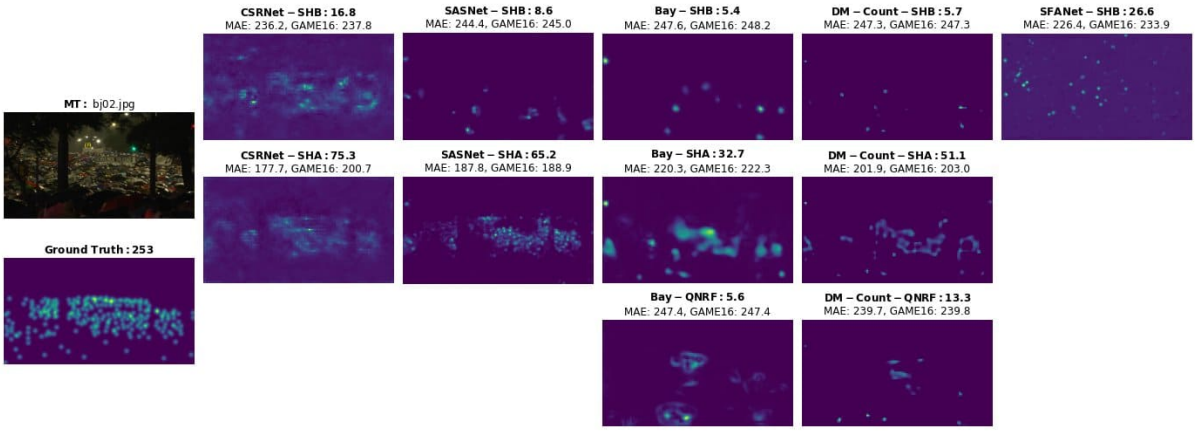


Figura A.18: MT-paraguas (bj02): Imagen con múltiples *oclusiones*† por paraguas, de muy mal desempeño para todos los modelos.

A.1.5. Ejemplos de otras particularidades

En esta sección se destacan varias particularidades que se observaron en los resultados obtenidos (figuras A.19, A.20, A.21 y A.22).

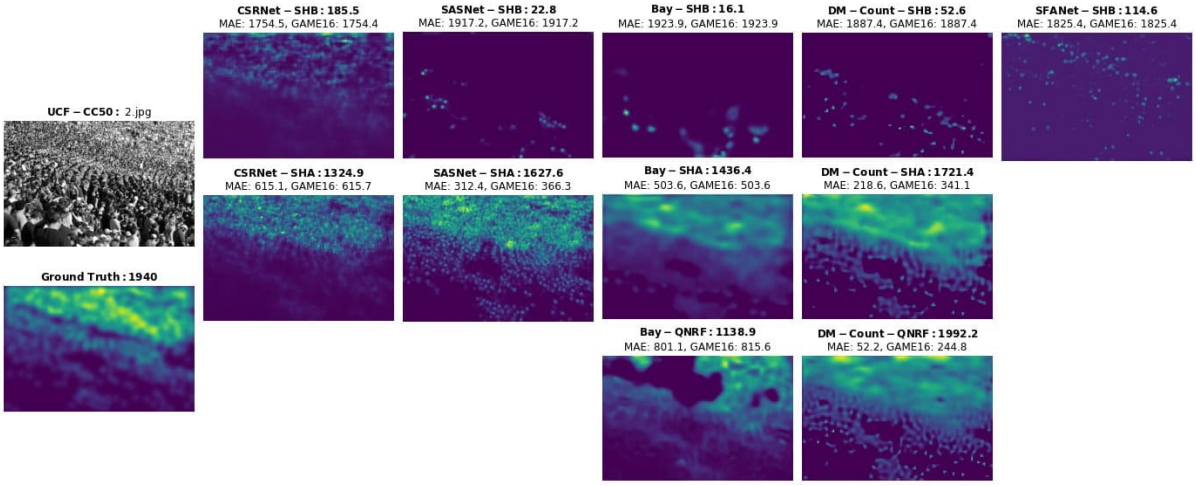


Figura A.19: UCF-CC50 (2): Imagen densa y en escala de grises, se destaca cómo modelos entrenados con *SHB* tienen mal desempeño, así como que *BAY-QNRF* falló en detectar correctamente el fondo.

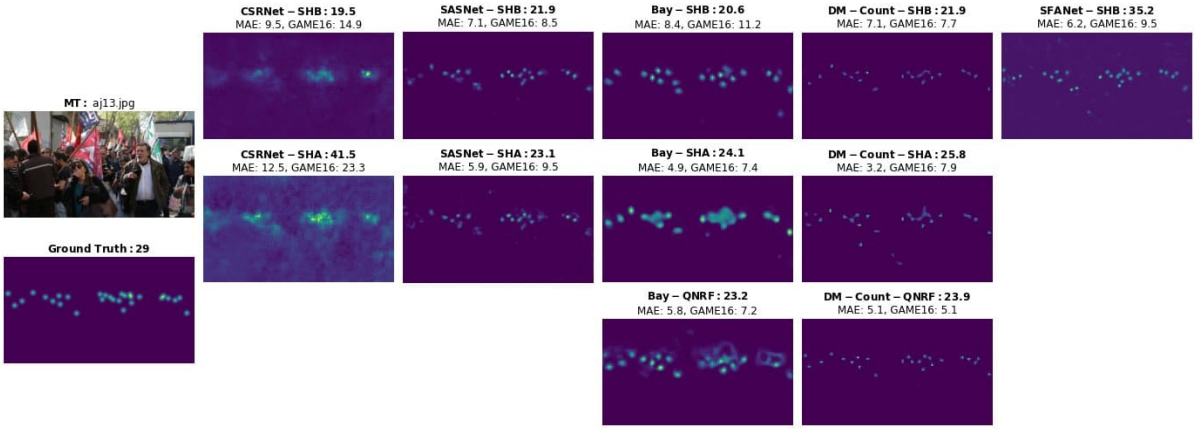


Figura A.20: MT-baja (aj13): Imagen con razonablemente buen desempeño en su estimación, excepto por *CSRNet-SHA* que no logra localizar bien las personas en la imagen y sobre-estima las mismas.

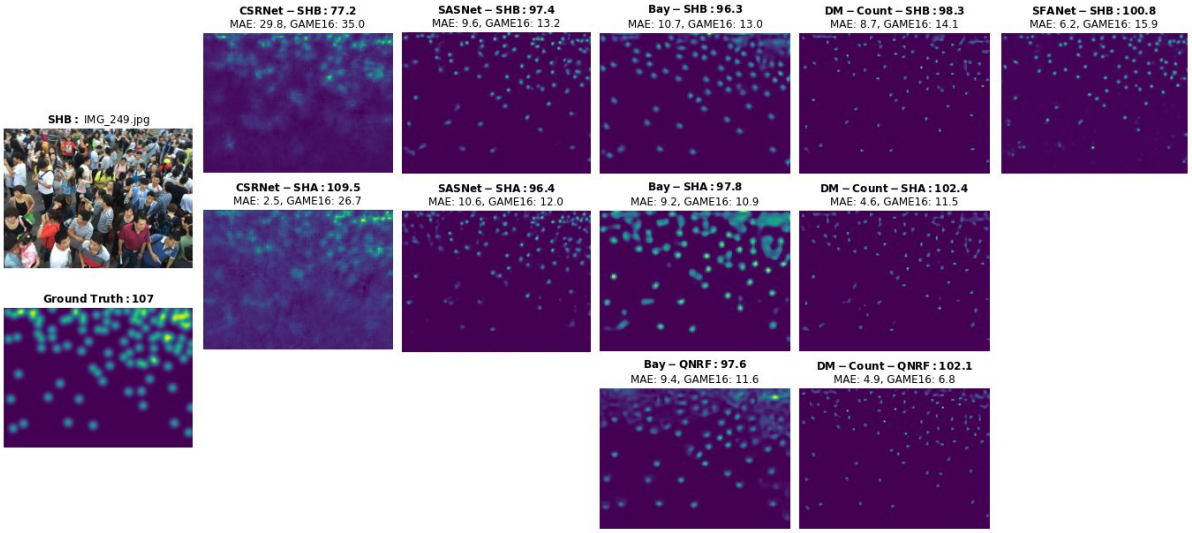


Figura A.21: SHA (249): Imagen del conjunto *SHB*, en la que los mejores tres resultados se obtienen con modelos entrenados con *SHA* o *UCF-QNRF*.

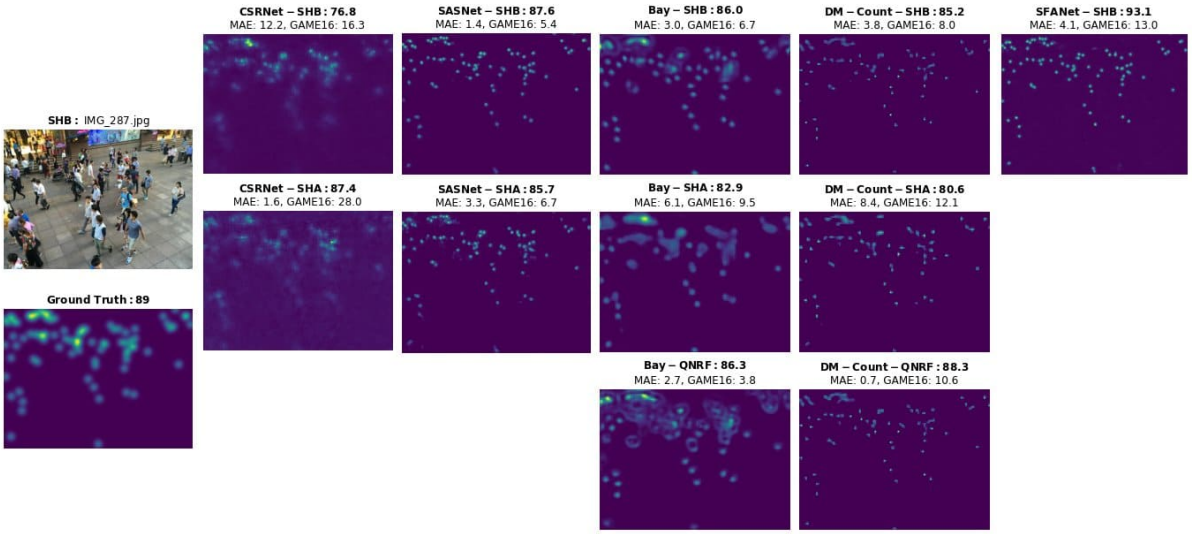


Figura A.22: SHB (287): Imagen cuyas estimaciones son en general de buen desempeño, se destaca la diferencia entre *mapas de densidad*†, particularmente los producidos por el método de la *función de pérdida bayesiana*†.

A.1.6. Ejemplos de falsos positivos u artefactos en mapas de densidad generados

En esta sección se destacan casos en los que es claro que un modelo estimó incorrectamente un área en la imagen, introduciendo ruido en la estimación general (figuras A.23, A.24, A.25 y A.26).

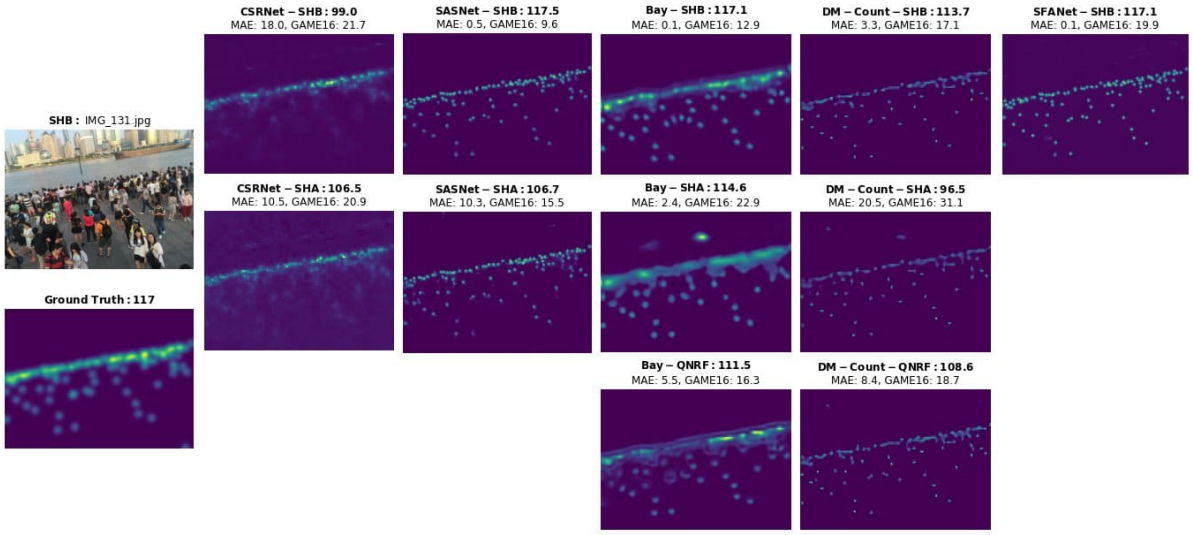


Figura A.23: SHB (131): Se aprecia como algunos modelos confunden el barco y los edificios en el fondo de la imagen con personas.

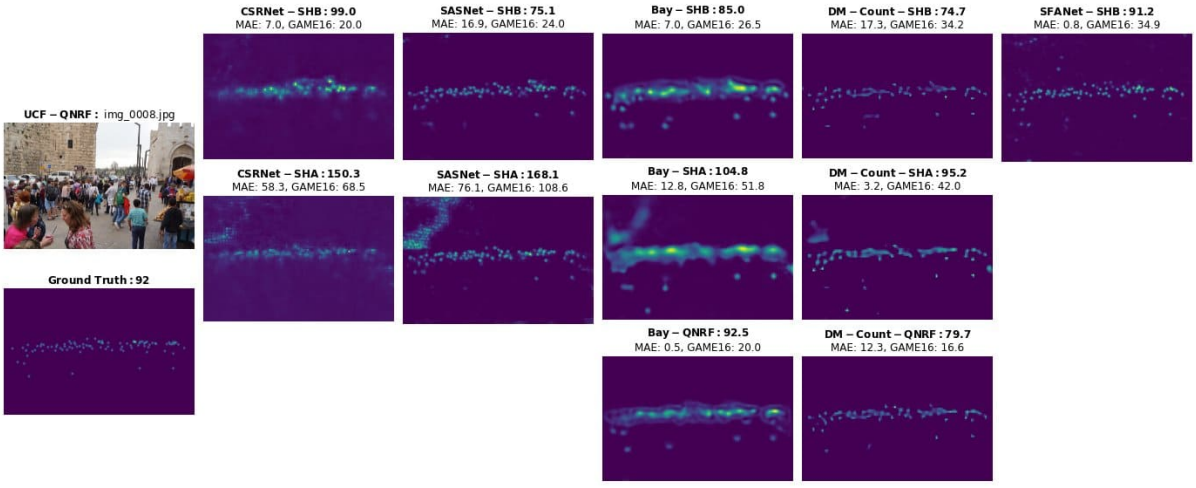


Figura A.24: UCF-QNRF (8): Se aprecia como varios modelos confunden la pared del castillo con personas.

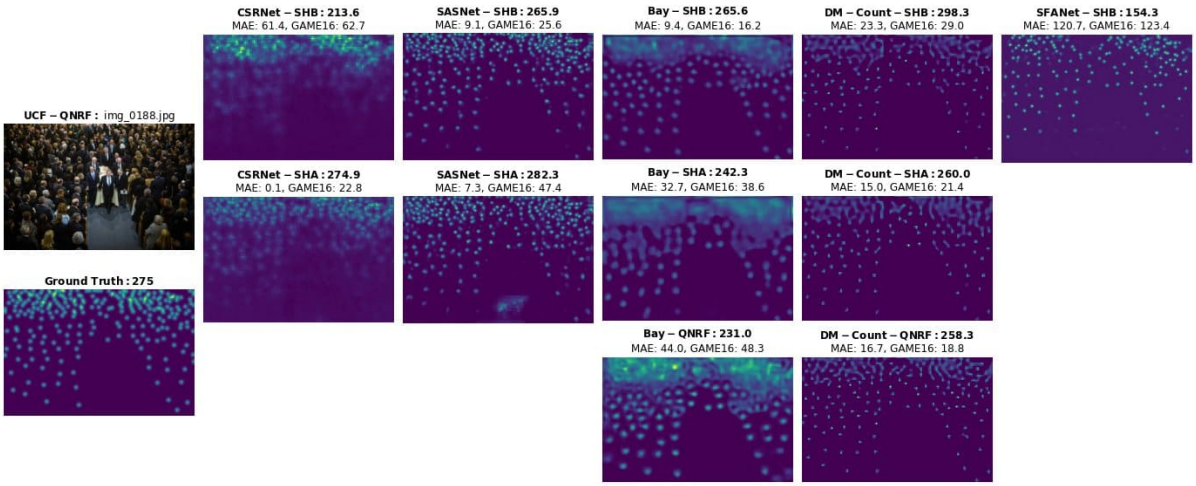


Figura A.25: UCF-QNRF (188): Imagen de baja dificultad y en general buen desempeño entre los modelos, se aprecia como *SASNet-SHA* detecta personas en un espacio vacío al frente de la imagen.

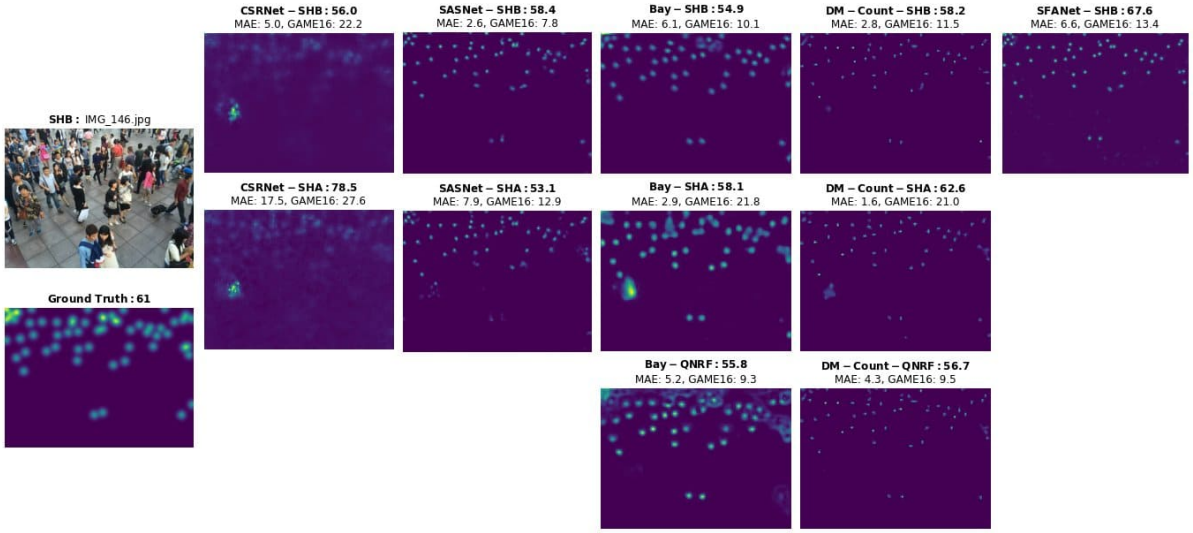


Figura A.26: SHB (146): Imagen de baja densidad, se aprecia como *CSRNet-SHB*, *CSRNet-SHA*, *SASNet-SHA* y *Bay-SHA* estiman una agrupación de personas sobre una persona con una camisa que sigue un patrón elaborado en sus colores.

Apéndice B

Apéndice: Complemento del marco teórico

A continuación se desarrollan conceptos que profundizan en varias aristas generales o anecdóticas del conteo de multitudes a través de *redes neuronales*†, pero que no se consideran centrales al foco de este estudio.

B.1. Funciones de pérdida adicionales

B.1.1. Función de pérdida de correlación espacial (SCL)

La *función de pérdida de correlación espacial (SCL)*† es la correlación cruzada entre el objetivo normalizado, y el *mapa de densidad*† inferido. Se calcula para un *mapa de densidad*† como:

$$\ell_{SC} = 1 - \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(\hat{D}_{ij} \cdot D_{ij})}{\sqrt{\sum_{i=1}^{N}\hat{D}_{ij}^2 \cdot \sum_{i=1}^{N} D_{ij}^2}}, \tag{B.1}$$

en donde D_{ij} y $\hat{D}_{ij}$ son los *mapas de densidad*† objetivos e inferidos (respectivamente), i y j son los índices de fila y columna de cada píxel del *mapas de densidad*†, y N y M representan sus dimensiones.

Esta *función de pérdida*† fue empleada por **TEDNet** (*Jiang et al.*, 2019) en combinación con la *función de pérdida euclidiana* $(\ell_2)^\dagger$, debido a que es facil de computar y menos sensible a cambios lineales en la intensidad del *mapa de densidad*†.

B.1.2. Función de pérdida AP

La *función de pérdida euclidiana* $(\ell_2)^\dagger$ computa la distancia píxel a píxel entre el *mapa de densidad*† inferido y el objetivo. Esto presenta un problema, y es que no considera las variaciones de densidad en las imágenes, y por ende lleva a errores de generalización (*Khan et al.*, 2022).

La *función de pérdida AP*† (de pirámide adaptativa) fue propuesta por *Jiang et al.* (2020) y empleada en el entrenamiento del modelo **ASNet**. Para computarla, el **mapa objetivo** es dividido en cuatro parches no superpuestos. Si la cantidad de personas en cada parche es mayor que un límite T, ese parche es dividido en otros 4 parches adicionales. Este proceso se repite hasta que todos los parches tengan T o menos personas.

Luego de determinar la cantidad y localización de los parches se le aplica la misma división al *mapa de densidad*† inferido por el modelo, y se calcula la pérdida:

$$\ell_{AP} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{J} l_{R_j}^i, \tag{B.2}$$

en donde N es el número de *mapas de densidad*†, J es la cantidad total de parches a evaluar y $l^i_{R_j}$ representa la *función de pérdida* $(\ell_1)^\dagger$ para el parche j de la imagen i.

B.1.3. Función de pérdida de currículum

El *entrenamiento*† de currículum es una estrategia propuesta por *Bengio et al.* (2009), en la cual, en lugar de formar baches aleatorios durante el *entrenamiento*† del modelo, los baches son conformados por un orden incremental de complejidad.

Esta estrategia fue empleada en el conteo de multitudes por el modelo **PSDDN** (*Liu et al.*, 2019e), de forma que a cada imagen se le asignó un índice de dificultad.

Wang and Breckon (2022) emplean *Aprendizaje por currículum*† en el entrenamiento de **SGANet**, pero además proponen la *función de pérdida de currículum*†, que en su diseño se calcula el nivel de dificultad de todos los píxeles en los *mapas de densidad*†. Siguiendo la intuición de que los píxeles con alta densidad son más difíciles de estimar, todos los píxeles con mayor densidad a un umbral son considerados "difíciles".

Para computar el nivel de dificultad y la *función de pérdida*† final, es necesario en base a esto definir una matriz de pesos W con las mismas dimensiones que los *mapas de densidad*†:

$$W(e) = \frac{T(e)}{max\{D_i - T(e), 0\} + T(e)}, \tag{B.3}$$

en donde D_i es el *mapa de densidad*† objetivo, $T(e)$ es un *hiperparámetro*† umbral, el cuál además es dinámico para cada *época*† e y es definido como:

$$T(e) = ke + b, \tag{B.4}$$

y k y b pueden ser determinados basándose en el conocimiento a priori de los valores de los píxeles de los *mapas de densidad*† previamente generados. b es el valor inicial del umbral que limita el intervalo, debería ser equivalente al valor máximo de densidad en una región con un solo individuo; mientras que el valor de k controla la velocidad de incremento de la dificultad, k puede ser derivado a partir de la curva de aprendizaje cuando no se emplea la estrategia del aprendizaje por currículum.

Se aprecia que independientemente del *época*† e, $W(e)$ tendrá un valor de 1 para píxeles más altos que el umbral, y menor a 1 para píxeles más pequeños que el umbral. tendrá un valor más alto para los píxeles cercanos al umbral $T(e)$

Modificando en base a esto la *función de pérdida euclidiana* $(\ell_2)^\dagger$ se obtiene la *función de pérdida*†: (*Wang and Breckon*, 2022)

$$L_{curr} = \frac{1}{2N}\sum_{i=1}^{N} ||W(e) \odot (\hat{M}_i^{den} - M_i^{den}||_F^2), \tag{B.5}$$

en donde $\odot$ es el producto punto (es decir, la suma de la multiplicación elemento a elemento de las matrices).

B.1.4. Función de pérdida del OT (Transporte Óptimo)

La *pérdida de OT*† es introducida por el modelo **DM-Count** (*Wang et al.*, 2020a). Esta se basa en el **transporte óptimo** (*Villani*, 2009) y la formulación *Monge-Kantarovich OT*.

La siguiente sección está fuertemente basada en el trabajo **DM-Count** *(Distribution Matching for Crowd Counting)* (*Wang et al.*, 2020a).

B.1.4.1. Principio del transporte óptimo

El **transporte óptimo** refiere al coste óptimo de transformar una distribución de probabilidad en otra. Se define el Transporte Óptimo de Monge-Kantorovich (OT) como el costo entre $\boldsymbol{\mu}$ y $\boldsymbol{\nu}$ que cumple con:

$$\mathcal{W}(\boldsymbol{\mu}, \boldsymbol{\nu}) = min_{\gamma \in \Gamma} \langle \mathbf{C}, \gamma \rangle, \tag{B.6}$$

en donde:

- $\langle A, B \rangle$ el producto interno entre A y B, que para matrices se define como el producto interno entre los vectores conformados por apilar las columnas de cada matriz en un vector.
- $\mathcal{X} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ e $\mathcal{Y} = \{\mathbf{y}_j | \mathbf{y}_j \in \mathbb{R}^d\}_{j=1}^n$, son dos conjuntos de puntos en un espacio vectorial de dimensión d.
- $\boldsymbol{\mu}$ y $\boldsymbol{\nu}$ son dos medidas de probabilidad definidas sobre $\mathcal{X}$ e $\mathcal{Y}$ respectivamente, con $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbf{R}_+^n$ y que cumplen que $\mathbf{1}_n^T \boldsymbol{\mu} = \mathbf{1}_n^T \boldsymbol{\nu} = 1$, siendo $\mathbf{1}_n$ un vector de n dimensiones en donde todos sus valores son 1.
- $c : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}_+$ es la función de costo de moverse de un punto en $\mathcal{X}$ a un punto en $\mathcal{Y}$, y $\mathbf{C}$ es la matriz de costo $n \times n$ correspondiente para dos conjuntos de puntos $\mathbf{C}_{ij} = c(\mathbf{x}_i, \mathbf{y}_j)$.
- $\Gamma = \{\gamma \in \mathbf{R}_+^{n \times n} : \gamma \mathbf{1} = \boldsymbol{\mu},\ \gamma^T \mathbf{1} = \boldsymbol{\nu}\}$, es el conjunto de todas las posibles formas de transformar la masa de probabilidad de $\mathcal{X}$ a $\mathcal{Y}$.

Conceptualmente se puede visualizar imaginando la distribución $\boldsymbol{\mu}$ como una pila de tierra en $\mathcal{X}$, y $\boldsymbol{\nu}$ como otra pila en $\mathcal{Y}$. Entonces el transporte óptimo es el coste mínimo de reorganizar una distribución para convertirla en la otra (redistribuir los puntos de la misma para que formen la otra distribución con el menor coste posible).

Este coste es una medida útil para cuantificar la disimilitud entre dos distribuciones de probabilidad, así como la distancia entre sus elementos (*Wang et al.*, 2020a).

Dicho coste también se puede computar con la formulación dual, la cual facilita el cálculo del gradiente:

$$\mathcal{W}(\boldsymbol{\mu}, \boldsymbol{\nu}) = max_{\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbf{R}^n} \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle + \langle \boldsymbol{\beta}, \boldsymbol{\nu} \rangle,\ s.t.\ \alpha_i + \beta_j \leq c(x_i, \mathbf{y}_j), \forall i, j. \tag{B.7}$$

B.1.4.2. Función de pérdida del OT (Transporte Óptimo)

En el contexto del conteo de multitudes es posible computar el coste del transporte óptimo para un *mapa de densidad*† estimado $\hat{\mathbf{z}} \in \mathbb{R}_+^n$ (expresado en forma vectorial apilando las columnas de la matriz) y para un mapa de anotaciones binarias $\mathbf{z} \in \mathbb{Z}_2^n$ (también expresado como un vector que apila las columnas de su matriz).

Es posible visualizar a $\mathbf{z}$ y $\hat{\mathbf{z}}$ como funciones de densidad no normalizadas ($||\mathbf{z}||_1$ es igual a la cantidad de personas en el mapa de anotaciones), y tras normalizarlas (dividiéndolas sobre su masa total) las mismas se convierten en funciones de densidad de probabilidad (PDF).

Luego en base a estas distribuciones es posible emplear los principios del *transporte óptimo (OT)*† para computar una *función de pérdida*† que minimice el costo entre las distribuciones. Esta estrategia busca que la convergencia del *entrenamiento*† no sea exactamente en base a los valores que infiere, sino a su espacio de distribución de probabilidad, y se formula como:

$$\ell_{OT}(\mathbf{z}, \hat{\mathbf{z}}) = \mathcal{W}(\frac{\mathbf{z}}{||\mathbf{z}||_1}, \frac{\hat{\mathbf{z}}}{||\hat{\mathbf{z}}||_1}) = \langle \boldsymbol{\alpha}^*, \frac{\mathbf{z}}{||\mathbf{z}||_1} \rangle + \langle \boldsymbol{\beta}^*, \frac{\hat{\mathbf{z}}}{||\hat{\mathbf{z}}||_1} \rangle, \tag{B.8}$$

en donde:

- $\langle a, b \rangle$ el producto interno entre a y b.
- $\boldsymbol{\alpha}^* \in \mathbb{R}^n$ y $\boldsymbol{\beta}^* \in \mathbb{R}^n$ son las soluciones de la ecuación B.7.

- Para calcularlas se emplea la función de costo cuadrática $c(\mathbf{z}(i), \hat{\mathbf{z}}(j)) = ||\mathbf{z}(i) - \hat{\mathbf{z}}(j)||_2^2$, en donde $\mathbf{z}(i)$ y $\hat{\mathbf{z}}(j)$ son coordenadas 2D de i y j respectivamente.
- Para resolver dicha ecuación es posible emplear el algoritmo de Sinkhorn (*Peyré and Cuturi*, 2019)
- Para evitar división por cero se agrega un épsilon al denominador (*Wang et al.*, 2020a).

Como los valores de $\hat{\mathbf{z}}$ son no-negativos, es posible calcular el gradiente de la ecuación B.8 respecto a $\hat{\mathbf{z}}$ como:

$$\frac{\partial \ell_{OT}(\mathbf{z}, \hat{\mathbf{z}})}{\partial \hat{\mathbf{z}}} = \frac{\boldsymbol{\beta}^*}{||\hat{\mathbf{z}}||_1} - \frac{\langle \boldsymbol{\beta}^*, \hat{\mathbf{z}} \rangle}{||\hat{\mathbf{z}}||_1^2}, \tag{B.9}$$

el cual puede ser empleado en el algoritmo de *propagación hacia atrás*† para ajustar los pesos de la *red neuronal*†.

El trabajo demás nota que si bien existen otras formas de medir la disimilitud entre dos funciones de probabilidad de densidad (la *divergencia de Kullback-Leibler*, y la *divergencia de Jensen-Shannon*), las mismas no facilitan gradientes válidos para entrenar una red a no ser que la distribución de la fuente se superponga con la distribución objetiva *Arjovsky et al.* (2017), por lo que no servirían para entrenamientos que tengan de *valor anotado*† mapas binarios de anotaciones.

B.1.5. Función de pérdida de TV (Variación Total)

Esta *función de pérdida*† se propone en conjunto a la *pérdida de OT*†, debido a que para calcularla es necesario aplicar aplicar a cada iteración el *algoritmo de Sinkhorn* (*Peyré and Cuturi*, 2019) para aproximar $\boldsymbol{\alpha}^*$ y $\boldsymbol{\beta}^*$, cuyo orden de ejecución es $O(n^2 log(n)/\epsilon^2)$ (*Dvurechensky et al.*, 2018), siendo ϵ el error tolerado y n el espacio vectorial de la imagen.

Empleando este método numérico en el ajuste del modelo, la distancia respecto al objetivo disminuye sustancialmente al inicio, pero luego tiene una convergencia lenta hasta llegar al *transporte óptimo*†.

En la práctica, con el fin de contar con un tiempo de *entrenamiento*† razonable, se agrega un número máximo de iteraciones, y por ende el algoritmo de Sinkhorn retorna una solución aproximada. Por lo que no basta con usar sólo la *pérdida de OT*†, dado que si bien los *mapas de densidad*† convergerán a un valor parecido, este no será el mismo.

Esta función aproximará bastante bien las áreas densas de la multitud, pero su aproximación puede ser pobre para las áreas de baja densidad. Para atacar este problema de convergencia es que se define la *funcion de pérdida de TV*†, basada en la **variación total**:

$$\ell_{TV}(\mathbf{z}, \hat{\mathbf{z}}) = ||\frac{\mathbf{z}}{||\mathbf{z}||_1} - \frac{\hat{\mathbf{z}}}{||\hat{\mathbf{z}}||_1}||_{TV} = \frac{1}{2}||\frac{\mathbf{z}}{||\mathbf{z}||_1} - \frac{\hat{\mathbf{z}}}{||\hat{\mathbf{z}}||_1}||_1. \tag{B.10}$$

Debido a que optimizar la *pérdida de OT*† con el *algoritmo de Sinkhorn* (*Peyré and Cuturi*, 2019) es un procedimiento de optimización *minimización/maximización de punto en silla de montar*, que es similar a la optimización en el *entrenamiento*† de las *Generative Adversarial Network (GAN)s*† *Goodfellow et al.* (2020), en las que como muestra el trabajo *Pix2Pix GAN* (*Isola et al.*, 2017) es posible incrementar su estabilidad añadiendo una *función de pérdida*† de reconstrucción. Las *funciones de pérdida*† presentadas en este capitulo presentan una similitud, la *funcion de pérdida de TV*† cumple un rol similar a una *función de pérdida*† de reconstrucción e incrementa la estabilidad del proceso de *entrenamiento*† (*Wang et al.*, 2020a).
El gradiente de la ecuación B.10 respecto al *mapa de densidad*† inferido $\hat{\mathbf{z}}$ es:

$$\frac{\partial \ell_{TV}(\mathbf{z}, \hat{\mathbf{z}})}{\partial \hat{\mathbf{z}}} = -\frac{1}{2}\left(\frac{sign(\frac{\mathbf{z}}{||\mathbf{z}||_1} - \frac{\hat{\mathbf{z}}}{||\hat{\mathbf{z}}||_1})}{||\hat{\mathbf{z}}||_1} - \frac{\langle sign(\frac{\mathbf{z}}{||\mathbf{z}||_1} - \frac{\hat{\mathbf{z}}}{||\hat{\mathbf{z}}||_1}), \hat{\mathbf{z}} \rangle}{||\mathbf{z}||_1^2}\right), \tag{B.11}$$

en donde $sign(\cdot)$ es el vector resultante de aplicar la función de *signo* a cada elemento del vector y $\langle a, b \rangle$ el producto interno entre a y b.

B.2. Otros modelos destacados

B.2.1. Multicolumna

A modo de resumen de la evolución en el área se destacan otros trabajos que avanzaron en las arquitecturas multicolumna. En general las mismas han ido diversificando las responsabilidades de cada columna, cambiando de paradigma al *aprendizaje multitarea*†, en el que columnas pueden tener el rol de generar un resultado intermedio, que es utilizado como dato auxiliar en otra capa de la red.

- **CrowdNet:** (*Boominathan et al.*, 2016). Fue el primer modelo que empleó *aprendizaje por transferencia*†, empleando las primeras capas de la *VGG-16* (*Simonyan and Zisserman*, 2014). Una columna consta de la misma, mientras que otra más superficial corre en paralelo; las salidas de ambas columnas son concatenadas antes de pasar por la *convolución* 1×1†.
- **CMTL:** (*Sindagi and Patel*, 2017). Una red de dos columnas (Cascaded Multi-Task Learning). La primer columna predice una estimación *grosso modo* por regresión, mientras que la segunda se ocupa de generar el *mapa de densidad*† (recibiendo la salida de la columna anterior).
- **DecideNet:** (*Liu et al.*, 2018a). Plantea 3 columnas, pero a diferencia de la *MCNN* (*Zhang et al.*, 2016a), cada columna se encarga de una tarea distinta. La primera columna **RegNet**, predice el *mapa de densidad*†. La segunda, **DetNet**, usa la red **Faster R-CNN** (*Girshick*, 2015) y con *bounding boxes* genera un *mapa de densidad*† basado en detección. La última, **QualityNet**, combina los dos *mapas de densidad*† generados por **RegNet** y **DetNet** en el *mapa de densidad*† final.
- **SAAN:** (*Hossain et al.*, 2019) Emplea mecanismos de *atención*† para fusionar los *mapas de densidad*† generados por las distintas columnas de la red.
- **PACNN:** (*Shi et al.*, 2019). Emplea las primeras cuatro capas de la **VGG-16** (*Simonyan and Zisserman*, 2014) como *backbone*†. Las primeras tres columnas generan *mapas de densidad*† con distinto tamaño en filtros, mientras que la cuarta columna genera dos mapas de perspectiva, que son enviados a las capas intermedias de dos de las columnas que generan *mapas de densidad*†.
- **PCCNet:** (*Gao et al.*, 2019b). Emplea tres columnas, de las cuales una clasifica el nivel de densidad (del 1 al 10), otra crea el *mapa de densidad*†, y la última crea un mapa de segmentación de los individuos en la imagen respecto al frente y fondo de la misma.
- **ASNet:** (*Jiang et al.*, 2020). Bifurca la entrada en dos columnas: **DANet**, una red auxiliar que genera mapas de segmentación; y **ASNet**, que luego de pasar por un *backbone*† se bifurca a su vez en dos columnas, una de las cuales se divide internamente en N columnas más y produce N *mapas de densidad*†, mientras que la otra genera atributos auxiliares de escalado para combinar la información de los *mapas de densidad*† generados junto con los mapas de segmentación. El *entrenamiento*† de esta red propuso y emplea la *función de pérdida AP*†.
- **TAFNet:** (*Tang et al.*, 2022). Se diseñó para *conjuntos de datos*† con cámaras térmicas: una columna recibe la imagen, otras la imagen térmica, y una última ambas imágenes.

B.2.2. Enfoque híbrido

Varios trabajos contemporáneos experimentaron combinar el paradigma de única columna con el de múltiples columnas, terminando en un enfoque híbrido. Esto en general consiste en romper la arquitectura multicolumna, separando la estructura de la red en ramales, pero unificando el resultado de los mismos en etapas intermedias.

- **MSCNN:** (*Zeng et al.*, 2017). El primer modelo de este tipo. Contiene tres bloques secuenciales, cada uno teniendo columnas independientes con distintos tamaños en filtros; se inspira en la arquitectura **Inception** (*Szegedy et al.*, 2015)
- **SANet:** (*Cao et al.*, 2018). También se basa en la arquitectura de **Inception** (*Szegedy et al.*, 2015), aplicando un conjunto de bloques multicolumna con distintos campos receptivos para extraer diferentes atributos. Dichos bloques son conectados al final empleando *convoluciones*†.
- **M-SegNet:** (*Thanasutives et al.*, 2021) Se basa en **SFANet** (*Zhu et al.*, 2019), en donde tras pasar por las primeras capas de la **VGG-16** como *backbone*†, la red determina un *mapa de densidad*† y un mapa de *atención*† con el cual ponderar el mapa anteriormente generado (*Zhu et al.*, 2019). Además complementa su inferencia de *mapas de densidad*† con la información de segmentación en la imagen (producida por una rama de segmentación inspirada en **SegNet** (*Badrinarayanan et al.*, 2017)), lo cual define más claro la *RoI*† para la imagen. Este trabajo emplea la *función de pérdida bayesiana*†.
- **M-SFANet:** (*Thanasutives et al.*, 2021) Es una versión más compleja y profunda de **M-SFANet** (propuesta en el mismo trabajo), la cual presenta los mismos componentes pero además extiende el *backbone*† con un bloque **CAN** (*Context-aware module*), inspirado en otro trabajo de conteo de multitudes, y un bloque **ASPP** (*Atrous spatial pyramid pooling*), inspirado en un trabajo sobre segmentación de imágenes (*Chen et al.*, 2017a). La utilidad de estos módulos es extraer información más refinada sobre la escala y fondo de la imagen.
- **SGANet:** (*Wang and Breckon*, 2022) Este trabajo experimenta con cambiar el *backbone*† comúnmente usado (**VGG-16** (*Simonyan and Zisserman*, 2014)) por **Inception-V3** (*Szegedy et al.*, 2016), para luego partirse en dos ramas: una de *atención*†, y otra que genera el *mapa de densidad*†. Este trabajo además propone y emplea la *función de pérdida de currículum*†.

Todos los modelos mencionados anteriormente han presentado resultados interesantes, pero ninguno de ellos estableció puntualmente el estado del arte actual (o convergió hacia este), por lo que este estudio no profundizó en los mismos.

B.3. Técnicas y módulos destacados

B.3.1. Refinamiento iterativo

El modelo **DRSAN** (*Liu et al.*, 2018b), emplea un bloque de extracción de atributos (GFE) para obtener g (la representación vectorial de la imagen) y M_0 una aproximación del *mapa de densidad*† para la imagen, luego M_0 es refinado iterativamente en bloques recurrentes atentos al espacio (RSAR), empleando g como conexión residual y el mapa resultante de la capa anterior. La figura B.1 muestra la arquitectura de este modelo.

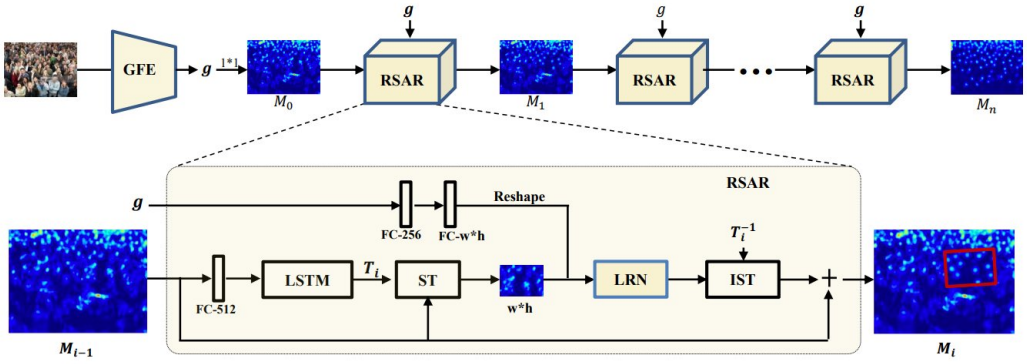


Figura B.1: DRSAN: Deep Recurrent Spatial-Aware Network (*Liu et al.*, 2018b)

Cada módulo RSAR (*Recurrent Spatial-Aware Refinement*) realiza las siguientes operaciones (*Liu et al.*, 2018b):

1. Selecciona iterativamente una región en la imagen, guardan información del *mapa de densidad*† e infiere las transformaciones a aplicar en una *Long Short-Term Memory Network (LSTM)*†.
2. Emplea *transformadores espaciales*† (ST), que aprenden a transformar el espacio del *mapa de densidad*† a un espacio más fácil de trabajar, que no sufra tanto los problemas de rotación y de *variación de escala*†, por la perspectiva y el ángulo de la cámara.
3. El bloque LRN (*Local Refinement Network*) combina esta información con la información global del mapa de atributos (GFE).
4. El bloque IST (*Inverse Spatial Transformer*) aplica la transformación espacial inversa (para volver al espacio original)
5. Agrega la nueva región refinada al *mapa de densidad*† resultado.

Este trabajo sostiene que su arquitectura es aplicable a cualquier modelo de conteo, sustituyendo el módulo GFE por el modelo a refinar (como **CSRNet** (*Li et al.*, 2018) por ejemplo).

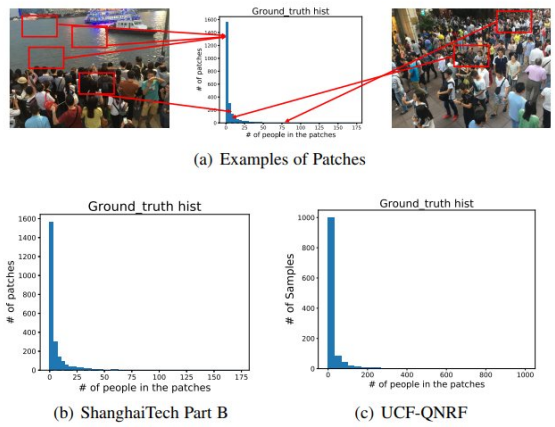
B.4. Heurísticas destacadas

En el área del *aprendizaje profundo*† existen una gran cantidad de técnicas que resultan agnósticas al problema que el modelo resuelve. Estas técnicas, sin atacar desafíos puntuales, permiten dada una solución mejorar la calidad de la misma.

Generalmente estas técnicas son aplicables a cualquier modelo, siempre que se cuente con el poder de cómputo y el tiempo de *entrenamiento*† y ejecución necesario para ejecutar las mismas.

Es por esto que en este trabajo las denominamos *heurísticas*, y a continuación se listan las que se han empleado y discutido en el campo del conteo de multitudes.

B.4.1. Equilibrado de los lotes de entrenamiento



(a) Examples of Patches

(b) ShanghaiTech Part B

(c) UCF-QNRF

Figura B.2: LABatch: Visualización de parches en *conjuntos de datos*† (*Zhou et al.*, 2021)

Los ejemplos de datos para el conteo de multitudes no suelen ser balanceados: por ejemplo, en el *conjunto de datos*† **ShanghaiTech A** (*Zhang et al.*, 2016b), una imagen tiene entre 33 y 3139 personas, promediando 501.
Esto es un problema aún mayor para los métodos basados en parches, que en lugar de trabajar con las imágenes en su totalidad emplean recortes aleatorios de las mismas.
A modo de atacar el problema de las *distribuciones no uniformes*† en una imagen, en 2021 el artículo *LA-Batch (Locality-Aware Batch)* (*Zhou et al.*, 2021) propone una técnica LADP *(Locality-Aware Data Partition)*, que a través de hasheo sensible a la localidad permite agrupar el *conjunto de entrenamiento*†, con el fin de crear lotes más balanceados. La figura B.2 muestra un ejemplo de distribución de lotes de entrenamiento.

La justificación de esta técnica proviene de que si bien el *descenso por gradiente estocástico*† tiene grandes beneficios para problemas de gran escala (debido a que el costo de computación es solamente proporcional al tamaño del lote), también tiene el problema de que frente a un lote desbalanceado el ajuste de los pesos no tiene una visión global del *conjunto de entrenamiento*†; esto agrega un sesgo, dado que los ajustes se realizan siempre desde un subconjunto de datos, en lugar de en base a su totalidad.

Algorithm 1 LSH for Data Partition

INPUT: $\{(\mathbf{x}_i, y_i)|_{i=1}^n\}$ with $\mathbf{x} \in \mathcal{R}^{L\times W\times 3}$, B, G.

1: Group the data into G group with width $\lceil y_{max}/G \rceil$
2: Generate $\mathbf{w}_c \in \mathcal{R}^{LW}$ for each channel, whose element is drawn from Gaussian Distribution.
3: Set hash value $h(i) \leftarrow \frac{1}{3}\sum_{c=1}^{3} \mathbf{w}_c{}^\top vec(\mathbf{x}_i(:,:,c))$
4: We evenly divide the group into a set of B bins according to their hash values.
5: **for** $t = 0, 1, 2, \cdots$, **do**
6: Sample one data point from each bin, compute the corresponding $\ell(\mathbf{x}_i, y_i)$ of the sampled data.
7: Compute $\tilde{\ell} \leftarrow \sum_i^C \ell(\mathbf{x}_i, y_i)$
8: Update gradient and model parameters.
9: **end for**

OUTPUT: Updated network parameters.

Figura B.3: LABatch: Pseudocódigo para la partición de datos para **G** (número de grupos) (*Zhou et al.*, 2021)

La figura B.3 muestra un pseudocódigo para la partición de datos.
Si el lote generado contiene una una gran cantidad de parches muy densos tenderá a estimar de más, mientras que si tiene muchos parches del fondo tenderá a estimar de menos. Debido a esto, en lugar de emplear técnicas como selección aleatoria de lotes, en cada iteración se selecciona un subconjunto representativo de datos.
Contar con lotes balanceados permite una convergencia más estable durante la *propagación hacia atrás*†, dado que no se provocará *sobreajuste*† a casos borde frente a un lote desbalanceado (causando que el modelo final estime de más o de menos).
El mismo artículo a su vez itera sobre esto agregando otro módulo LADA *(Locality-Aware Data Augmentation)*, en donde los parches de las imágenes son adaptativamente mejorados (*augmented*), escalandolos (upsample/downsample) para incluir más o menos individuos en el parche.

B.4.2. Predicción de incertidumbre

DUBNet (*Oh et al.*, 2019) presenta un estudio sobre la incertidumbre de los *mapas de densidad*† generados por los modelos, a modo de tener un índice respecto a la certeza de la estimación en distintas zonas de la imagen.

Para lograr esto construyen un *bootstrap ensemble*† compuesto por *k redes neuronales*†. Sin embargo, debido a que entrenar múltiples *redes neuronales*† es computacionalmente caro (especialmente en el campo del *aprendizaje profundo*†), en este estudio aplican una modificación en donde sólo se entrena **una *red neuronal***† con K cabezas de salida que se ramifican independientemente al resto.

El *entrenamiento*† aplica una idea similar al *dropout*† pero dependiente de los datos, en donde para cada partición j del *conjunto de entrenamiento*†, sólo se evalúan y reajustan los pesos de la cabeza j y la red compartida (congelando e ignorando el resto de las cabezas de salida). La figura B.4 muestra la arquitectura de esta red.

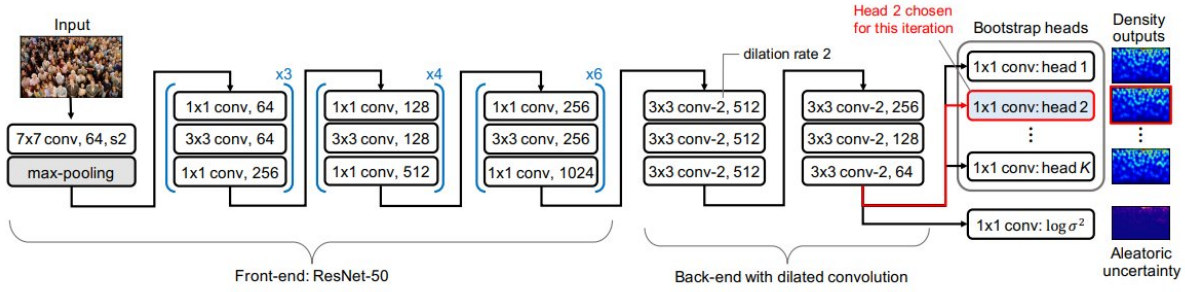


Figura B.4: DUBNet: Arquitectura (*Oh et al.*, 2019)

Nótese que los modelos de predicción de incertidumbre a veces fallan en capturar la distribución real de los datos (*Lakshminarayanan et al.*, 2017), por ejemplo un intervalo de confianza a posteriori de 95 % no necesariamente contiene el verdadero resultado en el 95 % de los casos. En estos casos se considera que el modelo no está calibrado (*Kuleshov et al.*, 2018).

Para combatir esto, este modelo incorpora una técnica propuesta en el artículo *Accurate uncertainties for deep learning using calibrated regression* (*Kuleshov et al.*, 2018), que calibra métodos de regresión (incluyendo *redes neuronales*†). Este proceso, cuyo pseudocódigo es mostrado por la figura B.6, recalibra las predicciones de un clasificador previamente entrenado (post-procesamiento). El estudio que lo propone muestra cómo, aplicado a modelos Bayesianos y con los suficientes datos, este método siempre converge en un modelo calibrado.

Algorithm 1 Decomposed Uncertainty using Bootstrap

Require: Input images $\{x_n\}_{n=1}^N$, GT density $\{y_n\}_{n=1}^N$

Initialize parameters θ
for each epoch **do**
  for all $n = 1$ to N **do**
    Sample a bootstrap head $k \sim \text{Uniform}\{1, ..., K\}$
    Compute predictions $[\hat{y}_n, \hat{s}_n] = f_{\hat{\theta}_k}(x_n)$
    Compute loss:
    $\mathcal{L}(\theta_k) = \frac{1}{D_n} \sum_i \frac{1}{2\exp(\hat{s}_{n,i})} \|y_{n,i} - \hat{y}_{n,i}\|^2 + \frac{1}{2}\hat{s}_{n,i}$
    Update θ_k using gradient $\frac{d\mathcal{L}(\theta_k)}{d\theta_k}$
  end for
end for

Figura B.5: DUBNet: Entrenamiento en batches (*Oh et al.*, 2019)

Algorithm 2 Uncertainty Recalibration

Require: $\{C_n, \bar{C}_n, \bar{\sigma}\}_{n=1}^N$ for validation data

Compute $Z_n = (C_n - \bar{C}_n)/\bar{\sigma}_n$ for all n
Construct a recalibration dataset:
$$\tilde{\mathcal{D}} = \left\{\left(Z_n, \hat{P}(Z_n)\right)\right\}_{n=1,...,N}$$
where $\hat{P}(z) = |\{C_m \mid Z_m \leq z, m = 1, ..., N\}|/N$
Train a isotonic regression model $\mathcal{R}$ on $\tilde{\mathcal{D}}$.

Figura B.6: DUBNet: Recalibracion de la incertidumbre predictiva (*Oh et al.*, 2019)

La técnica consiste en incorporarle a la red **una cabeza extra** $k+1$ que se encarga de **predecir la varianza** σ^2 (ruido de observación) (pseudocódigo en figura B.5). Este parámetro es compartido con el *bootstrap ensemble*†, y se entrena con una *función de pérdida*† que contempla el mínimo cuadrado del residual (*incertidumbre epistémica*†) y un término aleatorio de regularización (*incertidumbre aleatoria*†):

$$L(\Theta) = \frac{1}{D}\sum_i \frac{1}{2\hat{\sigma}_i^2}||\mathbf{y}_i - \hat{y}_i||^2 + \frac{1}{2}log(\hat{\sigma}_i^2). \quad \text{(B.12)}$$

Si durante el *entrenamiento*† el modelo predice un σ^2 muy alto, el término residual no tendrá mucho efecto en los pesos y el segundo término dominará la *función de pérdida*†. El modelo tiene la opción de ignorar los datos ruidosos durante el *entrenamiento*†, pero es penalizado por ello.
Se observa que debido a la estabilidad numérica de predecir σ^2 (que es positiva), se prefiere predecir la varianza logarítmica $s_i = log(\sigma^2)$ (*Kuleshov et al.*, 2018).
A diferencia de una configuración homocedástica, en el contexto del conteo de multitudes no es posible asumir que σ^2, el ruido de observación es constante (varía entre píxeles acorde a la posición de las cámaras, iluminación y distancia).

B.4.3. Búsqueda de arquitectura general (NAS)

El modelo **AMSNet** (*Hu et al.*, 2020b) aplica técnicas de *búsqueda de arquitectura neuronal (NAS)*† con el fin de encontrar la configuración paramétrica de los módulos dentro de la arquitectura que den el mejor resultado.
Debido a que el conteo por densidad es una tarea a nivel de píxel, el mismo requiere de arquitecturas que preserven la información espacial (las matrices de salida deben preservar dimensionalidad con las de entrada), y por ende las transformaciones de escalado (*upsample y downsample*) deben ser coherentes entre sí.

De manera de optimizar el tiempo de búsqueda, el estudio emplea una estrategia de búsqueda *one-shot*, en donde los parámetros de la arquitectura son inferidos al mismo tiempo a través de un optimizador basado en gradiente.

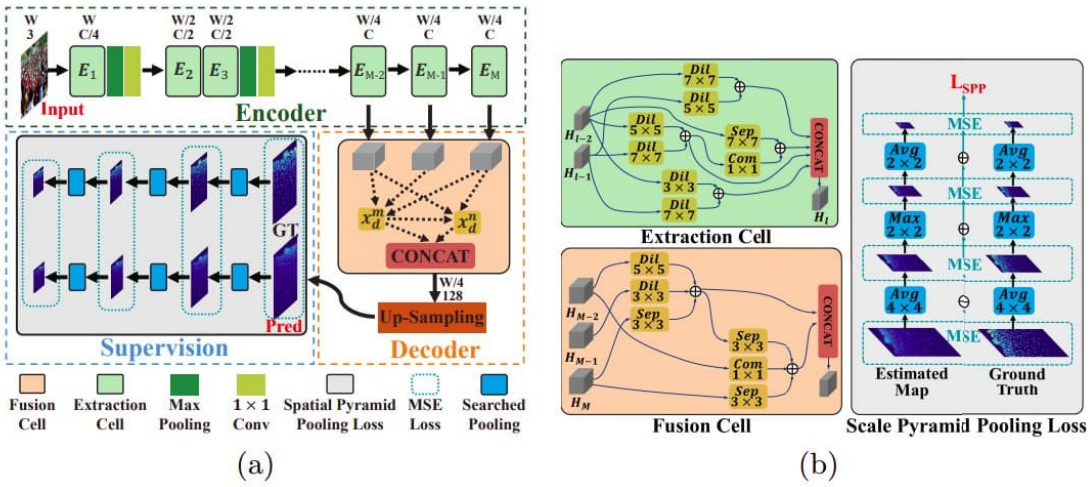


Figura B.7: Automatic Multi-Scale Network: Arquitectura y configuración de celdas de mejor resultado en la búsqueda. (*Hu et al.*, 2020b)

El estudio divide el espacio de búsqueda en dos niveles:

- **Macro:** Si bien está dada la presencia de los módulos *codificador-decodificador*†, se explora cómo estas capas se conectan entre sí, a modo de fusionar características multiescala y formar un *mapa de densidad*† de mayor calidad.
- **Micro:** Celdas multiescala son automáticamente exploradas para extraer y fusionar los atributos multiescala. Nótese como las operaciones entre las *capas de pooling*† no son triviales, dado que están limitadas a preservar la información espacial. Se emplean *convoluciones dilatadas*† para expandir los campos receptivos.

El mismo artículo además observa que la *función de pérdida euclidiana* (ℓ_2)† sólo tiene sentido si se asume un fuerte nivel de aislamiento entre píxeles (suposición que no se cumple entre regiones con *variación de escala*†).

Para remediarlo, propone basarse en el módulo **ASPP** *(Atrous Spatial Pyramid Pooling)* (*Chen et al.*, 2016), cuyo estudio propone, donde capas no paramétricas de *pooling*† están apiladas en una pirámide de dos caminos. Denominan esta supervisión como **SPPLoss** *(Scale Pyramid Pooling Loss)*.

Como muestra la figura B.7 B, luego de inferir el *mapa de densidad*† **E** y dado el *valor anotado*† **G**, el *mapa de densidad*† es refinado progresivamente y entre *capas de pooling*† se computa la *función de pérdida*† (*MSE*† entre píxeles). Esto es el equivalente a computar la diferencia estructural incrementando los campos receptivos a nivel de región, y por ende puede supervisar mejor la estimación a nivel de píxel entre distintas escalas (*Hu et al.*, 2020b).

La arquitectura del modelo **AMSNet** finalmente propuesto por la *NAS*† se puede apreciar en la figura B.7.

Apéndice C

Glosario

Adaptación al dominio

Ver sección 2.2.1. 9, 20, 64, 91, 103, 106

Aprendizaje automático

Sub-área de la inteligencia artificial que permite que un sistema aprenda y mejore de forma autónoma mediante técnicas de *entrenamiento*† en base a *conjuntos de datos*†. Este proceso suele explotar las correlaciones entre los datos para inferir atributos o particularidades. 10, 13, 23, 26–28, 37, 44, 47, 70, 134

Aprendizaje débilmente supervisado

El aprendizaje débilmente supervisado es un híbrido entre el *aprendizaje supervisado*† y el *aprendizaje no supervisado*†. Se puede aprender en base a un conjunto formado por datos anotados y otro subconjunto de datos sin anotación alguna, así como también en base a un conjunto de datos parcialmente anotados (por ejemplo la cantidad de personas en una imagen, sin datos sobre su distribución). 26, 38, 69, 104

Aprendizaje multitarea

Técnica de aprendizaje en donde durante el *entrenamiento*† se emplean distintas *funciones de pérdida*†, técnicas y módulos con el fin de influenciar en el rol que tendrá cada componente de la arquitectura del modelo. En el contexto de la visión artificial se observa una mejora de performance al combinar distintas tareas como clasificación, detección, segmentación, entre otras. Generalmente son diseñadas con múltiples sub redes (*Gao et al.*, 2020). 39, 51, 126

Aprendizaje no supervisado

Este tipo de aprendizaje consiste en entrenar un modelo contando con datos en bruto que no hayan sido procesados por un anotador (como por ejemplo una imagen de una multitud que no cuente con su correspondiente matriz de anotaciones). Estos métodos suelen explotar aún más las similitudes y correlaciones entre los datos, agrupándolos por ejemplo como es el caso con algoritmos de *clusterización*†. 38, 107, 133, 135

Aprendizaje por currículum

Estrategia para el *entrenamiento*† de un modelo en donde se seleccionan ejemplos sencillos al inicio del *entrenamiento*† y gradualmente se incorporan ejemplos de mayor dificultad. Usualmente se define un ranking de ejemplos de *entrenamiento*† que clasifica el nivel de dificultad de una instancia (*Wang and Breckon*, 2022). 123

Aprendizaje por transferencia

Ver sección 4.4.4.1. 9, 24, 44, 51, 64, 126, 134

Aprendizaje profundo

Ver sección 4.4.4. 10, 44, 65, 129, 130, 134

Aprendizaje supervisado

Ver sección 4.4.1.4. 26, 37, 69, 133

Atención

Ver sección 4.4.5. 47, 51, 55–58, 64–66, 69, 126, 127

Backbone

Es común en el *aprendizaje por transferencia*†, emplear como capas iniciales de una *red neuronal*† un subconjunto de capas de un modelo más general previamente entrenado (llamado *backbone*). El razonamiento tras esta técnica se basa en que esas capas aprendieron a detectar patrones o funciones abstractas o estructurales que son de gran uso para refinar los datos de entrada. Durante el *entrenamiento*† los pesos del *backbone* pueden permanecer congelados, o ser ajustados en la etapa de *propagación hacia atrás*† (comúnmente con un distinto ratio de aprendizaje). 44–48, 50–52, 55, 56, 59, 65, 126, 127

Bootstrap Ensemble

Técnica que emplea muestreo con reposición para aproximar distribuciones entre funciones con garantía teórica (en lugar de probabilística), atributo que brinda no sólo inferencia, sino también entendimiento y justificación a la incertidumbre de los valores estimados (*Bickel and Freedman*, 1981). En su forma más común, el método:

- Toma un *conjunto de datos*† D y una función f_θ. Subdivide el *conjunto de entrenamiento*† D en K conjuntos mutuamente exclusivos (tomando muestras uniformes con reemplazamiento).
- Entrena K modelos independientes (como *redes neuronales*†). Cada modelo f_{θ_k} es entrenado con el *conjunto de entrenamiento*† D_k.
- Estos K modelos se pueden usar de distintas formas, tales como consultarlos a todos y someter el resultado a una votación o promedio, o tomar muestras uniformemente aleatorias con un entero de 1 a k.
- Trata a cada modelo como muestra independiente de la distribución de pesos. A diferencia de los métodos Bayesianos, este método es frecuentista, con el uso de la distribución a priori puede aproximar la posteriori.

Nótese cómo el método sólo permite enfrentar el problema de la *incertidumbre epistémica*†, y no la *incertidumbre aleatoria*†. 130, 131

Búsqueda de arquitectura neuronal (NAS)

Subcampo del AutoML que encapsula los procesos que automatizan problemas del *aprendizaje automático*† y del *aprendizaje profundo*†, iniciado en 2016 con modelos ajustados por aprendizaje por refuerzos que trajeron arquitecturas del estado del arte a los campos de reconocimiento de imágenes y modelado de lenguajes (*Zoph and Le*, 2016; *Baker et al.*, 2016).
El método consiste en hacer una exploración paramétrica dentro de un espacio de soluciones (con una complejidad variable de localizada a exhaustiva), y por ende es de alto costo computacional. Cuanto más grande el espacio de búsqueda, más arquitecturas para entrenar, probar y evaluar. Hoy en día esto es mitigado por nuevos modelos de aprendizaje por refuerzos que podan y optimizan la eficiencia de la búsqueda dentro del espacio de soluciones (*Liu et al.*, 2017; *Pham et al.*, 2018).
Pese a esto, el área sufre de múltiples limitaciones. Las arquitecturas son evaluadas con el conjunto de entrenamiento, por lo que es necesario contar con buena calidad de datos. Se destaca como los métodos se pueden dividir en buscar entre arquitecturas de la red (macro) o la arquitectura de una celda (micro). Este último es el más popular, puesto que parte de una arquitectura diseñada a mano y reduce drásticamente el espacio de búsqueda exponencial. 131, 132

Capa convolucional

Capa de una *red neuronal*† que aplica operaciones de *convolución*†. Ver sección 4.4.3.1. 41–46, 48, 52, 59, 91

Capa de pooling

Capa de una *red neuronal*† que aplica operaciones de *pooling*†. Ver sección 4.4.3.3. 43, 48, 49, 51, 132

Capa totalmente conectada (fully connected layer)

Capa de una *red neuronal*† en la cual para cada neurona recibe como entrada la salida de todas las neuronas de la capa anterior, y transmite su salida a todas las neuronas de la capa siguiente. Además, para cada capa se agrega una neurona de *sesgo*, que tiene un valor constante de 1 y permite mayor holgura a la hora de ajustar los pesos durante el *entrenamiento*†. Para un ejemplo ver *MLP*†, detallado en la sección 4.4.1.1. 36, 40, 42, 43, 48, 49

Clusterización

Técnicas de *aprendizaje no supervisado*† que consisten en particionar *conjuntos de datos*† en base a características relacionadas entre los atributos que comparten, donde todo elemento se agrupa en conjuntos discretos basándose en su proximidad. 64, 108, 133, 139

Codificador-Decodificador

Módulo de *red neuronal*† cuyo principal objetivo es servir como modelo de transducción, donde la entrada y la salida son secuencias de no necesariamente la misma dimensión. Se tienen dos principales componentes (*Zhang et al.*, 2021a):

- **Codificador**: este componente tiene como entrada una secuencia de largo posiblemente variable y lo transforma a un estado de dimensiones fijas.
- **Decodificador**: este componente mapea el estado codificado de dimensiones fijas a una nueva secuencia (que también podría ser de largo variable).

. 51, 52, 132

Conexión residual

Una conexión residual, o *skip connection* consiste en alimentar la entrada de una red con la salida de capas no adyacentes a la misma. Ver sección 4.4.4.5. 45, 46, 53, 55, 56

Conjunto de datos

Colección de datos que puede ser procesada y usada por una computadora con fines analíticos y predictivos. 8–10, 13, 14, 16, 18–20, 22–28, 34, 38, 44, 47, 48, 50, 53, 56, 63–71, 81, 83–85, 88, 90, 97, 101–104, 106–108, 126, 129, 133–135, 137, 139–141

Conjunto de entrenamiento

Subconjunto de un *conjunto de datos*†, empleado para el *entrenamiento*† de un modelo. 19, 38, 61, 64, 66, 83, 91, 93, 94, 96, 101, 104–106, 108, 110, 114, 129, 130, 134–138, 140, 141

Conjunto de evaluación

Subconjunto de un *conjunto de datos*†, empleado para la *evaluación*† de un modelo. Debe ser disconjunto del *conjunto de entrenamiento*†. 10, 19, 31–35, 62, 64, 65, 83, 85–88, 91, 92, 101, 105, 106, 108, 140

Convolución

Ver sección 4.4.3.1. 29, 41–43, 45, 48, 49, 51, 52, 54, 55, 127, 135

Convolución 1×1

Ver sección 4.4.3.2. 42, 45, 49, 51, 59, 126

Convolución dilatada

Ver sección 4.7.1. 41, 45, 51, 52, 66, 132

Densidad de la multitud

Ver sección 2.2.2. 14, 15, 27, 28, 30, 48, 50, 58, 64

Descenso por gradiente

El algoritmo de *propagación hacia atrás*†, o *backpropagation* en inglés, es vital para el *entrenamiento*† de *redes neuronales*† profundas (con más de una capa interna), y consiste en propagar el gradiente de la *función de pérdida*† hacia atrás, actualizando los pesos de cada neurona en una dirección que, de acuerdo al gradiente calculado, minimice el valor de la *función de pérdida*† gracias a la regla de la cadena.
Esto durante la etapa de *entrenamiento*† permite corregir el valor de los pesos (los cuales inicializan con valores según la estrategia de inicialización aplicada, como por ejemplo valores aleatorios), en una dirección pequeña hasta que converjan a valores en los que minimicen el error obtenido (medido acorde a *función de pérdida*† aplicada) para las instancias del *conjunto de entrenamiento*†. 37, 38, 136, 140

Descenso por gradiente estocástico

Uno de los algoritmos más populares de *descenso por gradiente*† es el descenso por gradiente estocástico, que consiste en un método numérico aplicado a una función objetivo diferenciable o diferenciable por partes. Dicho método es una aproximación estocástica del algoritmo de *descenso por gradiente*†, debido a que reemplaza el gradiente por un estimado, reduciendo de esta forma el costo computacional del algoritmo. 129

Distorsión de perspectiva

Se da cuando se presenta una deformación o transformación de los objetos y su área circundante difiere significativamente de como se vería el objeto con una distancia focal normal, debido a la escala relativa de las características cercanas y distantes (*Gao et al.*, 2020). 14, 23, 53, 96

Distribucion no uniforme

Ver sección 2.2.4. 48, 53, 58, 129

Dropout

Técnica de regularización empleada para disminuir la ocurrencia del fenómeno de *sobreajuste*†. Consiste en aleatoriamente (o con alguna estrategia) omitir un subconjunto cambiante de pesos de la red (cambiando su valor a 0) a lo largo de las distintas etapas del *entrenamiento*†. Esto permite que la red sea menos dependiente de un pequeño grupo de neuronas, y por ende modele una función más regular a lo largo de sus capas (*Srivastava et al.*, 2014). 130

Entrenamiento

El entrenamiento es la etapa en donde se ajustan los pesos del modelo a una tarea particular. Se entrena en base a un *conjunto de entrenamiento*† y empleando un algoritmo de aprendizaje vinculado según el tipo de modelo a entrenar, siendo entrenado en iteraciones o a lo largo de varias *épocas*†. 8, 9, 13, 15, 23–25, 27–30, 36–40, 42–44, 46, 48–50, 53, 56, 58, 59, 63–69, 86, 91, 102, 104, 106, 123–126, 129–131, 133–136, 140, 141

Error Absoluto Medio (MAE)

Ver sección 4.3.1.1. 31–34, 62, 63, 86, 91, 98, 101, 106, 110

Error Absoluto Medio a nivel de Píxel (MPAE)

Ver sección 4.3.3.1. 34

Error Absoluto Medio por Grillas (GAME)

Ver sección 4.3.2.1. 32, 33, 86, 91, 98, 101, 110

Error Absoluto Medio por Parches (PMAE)

Ver sección 4.3.2.2. 33

Error Cuadrático Medio (MSE)

Ver sección 4.3.1.2. 32, 63, 132

Evaluación

Proceso mediante el cual se cuantifica la calidad de las predicciones de un modelo. Se mide su desempeño empleando un subconjunto del *conjunto de datos*† disconjunto del *conjunto de entrenamiento*†. Se comparan las predicciones del algoritmo con los datos etiquetados (*valor anotado*†) y se emplea una o varias *métricas de evaluación*† para cuantificar el desempeño (*Skyl.ai*, 2019). 25, 28, 29, 48, 58, 63, 70, 86, 91, 93, 106, 135, 138

Filtro en el dominio del espacio

Los filtros en el dominio del espacio bidimensional son transformaciones definidas en base a un punto (x, y) y una matriz M de máscara G compuesta por pesos σ, realiza el producto punto entre la máscara una ventana de la matriz M de igual dimensión que la máscara G, el valor resultado pasa a ser el nuevo valor para (x, y). 137

Filtro Gaussiano

Un filtro gaussiano es un tipo de *filtro en el dominio del espacio*† en donde los coeficientes de la máscara modelan una función gaussiana $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$, teniendo un coeficiente más alto en el centro y disminuyendo el mismo hacia el borde. Para su aplicación al conteo de multitudes ver sección 4.2.1. 29, 56, 59, 86

Fondo complejo

Ver sección 2.2.5. 15, 21, 27, 98, 102

Función de activación

Ver sección 4.4.1.3. 36–38, 41

Función de pérdida

Ver sección 4.4.2. 18, 30, 38–40, 47, 56, 57, 59, 62, 68, 122–125, 131–133, 136, 139, 140

Función de pérdida (ℓ_1)

Ver sección 4.4.2.2. 34, 39, 123

Función de pérdida AP

Ver sección B.1.2. 40, 122, 126

Función de pérdida atenta a la región piramidal (PRA Loss)

Ver sección 4.4.2.4. 40, 58

Función de pérdida bayesiana

Ver sección 4.9.1.1. 59, 61, 62, 119, 127

Función de pérdida de composición

Ver sección 4.4.2.3. 39, 51, 56, 57, 61

Función de pérdida de correlación espacial (SCL)

Ver sección B.1.1. 122

Función de pérdida de currículum

Ver sección B.1.3. 123, 127

Función de pérdida de OT (Transporte Óptimo)

Ver sección B.1.4. 61, 123, 125

Función de pérdida de TV (Variación Total)

Ver sección B.1.5. 61, 125

Función de pérdida euclidiana (ℓ_2)

Ver sección 4.4.2.1. 39, 56, 57, 60, 62, 122, 123, 132

Generative Adversarial Network (GAN)

Red neuronal† entrenada para dado un valor *semilla* (usualmente una realización de ruido), generar datos de un dominio. Consta de dos componentes principales:

- **Generador**: aprende a generar datos plausibles. Su salida se emplea para el entrenamiento del **discriminador**.
- **Discriminador**: Aprende a distinguir los datos falsos del **generador** de los datos reales. El discriminador penaliza al generador por producir resultados no plausibles.

Al principio del proceso el generador produce datos falsos y el discriminador aprende rápidamente a indicar que son falsos. A medida que continua el proceso el generador se acerca a producir datos que puedan engañar al discriminador, mientras el mismo va mejorando su capacidad de distinguir muestras falsas y correctas. Hasta llegar a un equilibrio en que el discriminador se equivoca con una probabilidad de 0.5 (*Documentation*, 2019; *Goodfellow et al.*, 2020). 125

Hiperparámetro

Un hiperparámetro es una variable de configuración externa (ya sea un atributo, función o característica) cuyo valor impacta en el entrenamiento o construcción del modelo. Durante la etapa de *evaluación*† es deseable experimentar con varias combinaciones de los hiperparámetros, seleccionando finalmente la que ofrezca un mejor desempeño observado en las *métricas de evaluación*† medidas. 29, 30, 40, 61, 123

Incertidumbre aleatoria

Incertidumbre que ocurre debido a la genuina estocasticidad de los datos; en esta situación una predicción incierta (de alta entropía) es la mejor posible. Esto ocurre por trabajar con datos con ruido, pues no importa cuántos datos haya visto el modelo, si el *conjunto de entrenamiento*† es incoherente consigo mismo (a causa de ruido en las anotaciones). 131, 134

Incertidumbre epistémica

Incertidumbre asociada a la falta del conocimiento, el entendimiento del problema es parcial. Esto ocurre cuando por falta de datos los modelos no logran ajustar y aprender bien la generalización de la tarea. En teoría al incrementar el volumen y diversidad de datos esta tiende a cero. 131, 134

Inferencia

Ver sección 4.4.1.2. 25, 37, 38, 40, 46, 50, 56, 58, 64, 65, 85, 139

K-Means

Algoritmo iterativo de *clusterización*† en el que dado un grupo deseado de grupos, se itera en el valor de centroides (puntos en el espacio vectorial del *conjunto de datos*†), minimizando en cada iteración la distancia de cada centroide a los puntos más cercanos al mismo. 64, 67, 108

Long short-term memory network (LSTM)

Uno de los tipos de *Red Neuronal Recurrrente (RNN)*† más comúnmente usado, que resuelve varios problemas experimentados por las *RNN*†, entre ellos el *problema del desvanecimiento del gradiente*†. Cada capa recurrente mantiene un valor de *estado*, influenciado por entradas anteriores; dependiendo del tipo de LSTM, el valor guardado en el *estado* puede variar en qué tanto es modificado o incluso ser reiniciado por completo a su valor inicial. 128

Mapa de densidad

Ver sección 4.2.1. 28–32, 34–36, 38–40, 42, 47–50, 52, 56, 57, 59–62, 66, 67, 69, 86, 91, 93, 99, 101, 110, 111, 119, 122–128, 130, 132, 139

Método basado en detección de objetos

Algoritmo que aplica un modelo de detección de objetos y a partir de una entrada retorna una lista de los objetos detectados y su posición en la imagen. Utiliza algoritmos de detección con ventanas deslizantes. (*Sassisegarane*, 2021). **Para su aplicación al conteo de multitudes ver sección 4.1.1**. 27

Método basado en estimación de densidad

Algoritmo que dada una matriz de entrada (como una imagen) retorna una matriz de igual tamaño con valores continuos que representa un *mapa de densidad*†. Estos mapas pueden tener diferentes características entre sí, dado que los métodos para estimarlos varían dependiendo de la selección de la *función de pérdida*† y el tipo de predicción (*Gao et al.*, 2020; *Sassisegarane*, 2021). **Para su aplicación al conteo de multitudes ver sección 4.2**. 28, 32, 36

Método basado en regresión

Algoritmo que aplica un modelo de regresión, el cual a partir de una entrada estima y retorna un número (*Sassisegarane*, 2021). **Para su aplicación al conteo de multitudes ver sección 4.1.2**. 27, 28

Métrica de evaluación

Función empleada para monitorear y medir el desempeño de un modelo. En general refiere a métricas de precisión, en donde se compara a través de alguna función uno o más valores anotados (*valor anotado*†) contra los mismos valores estimados por el modelo. Las métricas también pueden referir a métricas de desempeño, en donde se busca evaluar atributos como por ejemplo el tiempo de ejecución de una *inferencia*†. 8, 10, 18, 28, 31–35, 63, 86, 91, 104–106, 137, 138

Oclusión

Ver sección 2.2.6. 12, 15, 16, 23, 24, 27, 71–73, 82, 97, 98, 101, 102, 105, 117

Peak SNR (PSNR)

Ver sección 4.3.3.2. 34, 35, 86

Perceptrón Multi Capa (MLP)

Ver sección 4.4.1.1. 36, 135

Pooling

Ver sección 4.4.3.3. 43, 45, 52, 132, 135

Problema del desvanecimiento del gradiente

Durante el *entrenamiento*† de una *red neuronal*† empleando aprendizaje basado en un método de *descenso por gradiente*† y *propagación hacia atrás*†, en cada iteración del *entrenamiento*† cada neurona actualiza sus pesos proporcionalmente a las derivadas parciales de la *función de pérdida*† con respecto al peso actual. Este problema ocurre cuando el gradiente se achica hasta tender a cero (*vanishing*), lo que causa que el peso actual de una capa no se siga ajustando y el aprendizaje se detenga, lo que también detiene el ajuste de los pesos de las demás neuronas conectadas a la capa donde se desvaneció el gradiente (*Hochreiter et al.*, 2001). 37, 46, 139

Propagación hacia atrás

Ver sección 4.4.1.5. 37, 38, 125, 130, 134, 136, 140

Raiz del Error Cuadrático Medio por Parches (RPMSE)

Ver sección 4.3.2.3. 33

Raíz del Error Cuadrático Medio (RMSE)

Ver sección 4.3.1.2. 32–34, 86, 91, 106

Red neuronal

Ver sección 4.4.1. 8, 13, 27, 28, 36–40, 42–44, 48, 122, 125, 130, 131, 134–136, 138, 140, 141

Red Neuronal Recurrrente (RNN)

Red neuronal† que utiliza datos secuenciales o series de datos en el tiempo, la cual se distingue por realimentar su entrada con salidas anteriores. Se suele decir que las mismas poseen memoria, dado que para determinar una salida toman en cuenta la salida anterior de la secuencia (*Education*, 2020). 139, 141

Redes Neuronales Convolucionales (CNN)

Ver sección 4.4.3.4. 28, 40–46, 48, 49, 53, 55

Región de Interés (RoI)

Para una instancia (una imagen por ejemplo) se define un subconjunto de la misma, y solo dentro de ese subconjunto se evalúan los resultados de la predicción (*Brinkmann*, 1999). 34, 127

Sobreajuste

El fenómeno de sobreajuste, también conocido como *overfitting* en inglés, ocurre cuando la red tiene demasiada capacidad de expresión para el volumen de datos, lo que conlleva al modelo a aprender a etiquetar el *conjunto de entrenamiento*†, pero no suele obtener buen desempeño con casos no vistos (como los del *conjunto de evaluación*†). Esto ocurre porque el modelo está fallando en aprender la tarea en cuestión, detectando patrones que no resuelven el problema general, sino que sólo funcionan bien para los ejemplos vistos durante el entrenamiento. En las redes neuronales este fenómeno suele ocurrir a partir de cierta *época*† de entrenamiento, y según la heterogeneidad del *conjunto de datos*† puede ser más o menos pronunciado. 23, 43, 50, 51, 130, 136

Transformador espacial

Módulo de visión artificial propuesto en 2015 (*Jaderberg et al.*, 2015) que presenta un mecanismo que incrementa la invariancia espacial de un modelo contra transformaciones como translaciones, escalado, rotación, recortes y deformaciones no rígidas.
Suele ser independiente a la tarea a resolver, y puede ser inyectado en distintas etapas de una *red neuronal*† para intentar inferir, dada una entrada, cuál es la imagen canónica (ajustándola con las transformaciones anteriormente mencionadas).
Es utilizado en tareas de clasificación de imágenes, *Cp-localisation* (localizar en cada imagen dentro

de un conjunto de imágenes con diferentes distancias de una misma clase) y para *spatial attention* (mejorar imágenes de entrada con baja resolución). El módulo tiene 3 componentes principales:

- **Localisation Network**: Tiene de entrada el mapa de atributos, y al pasar por las capas ocultas, da como salida los parámetros de la transformación espacial que debe ser aplicada a la entrada.
- **Grid generator**: Se aplica la transformación encontrada al mapa de atributos de entrada, obteniendo así el sampling grid, el cual es un conjunto de puntos donde el mapa de entrada debe ser muestreado para producir la salida transformada. Se destaca como estas operaciones son determinadas mapeando sobre un subconjunto de la imagen; no la mapean en su totalidad.
- **Sampler**: Al final, se itera sobre las entradas del sampling grid, mapeándolo a la entrada y aplicando interpolación alrededor de los puntos a modo de obtener la salida final.

. 128

Transformer

Los *transformers* son un tipo de arquitectura en las *redes neuronales*†, la misma es capaz de manejar información secuencial, pero a diferencia de las *RNNs*† no se basan en conexiones recurrentes. En cambio se basan en bloques (que combinan capas *feedforward* junto con capas de auto-atención) y reciben una entrada secuencial $(x_1, \ldots, x_n)$ que es mapeada a una salida de igual longitud $(y_1, \ldots, y_n)$. Al procesar cada atributo x_i el modelo tiene acceso a todas las entradas $x_j \mid j \geq i$. Además el cómputo con cada atributo es independiente al resto, lo cual permite fácilmente paralelizar su inferencia. Las capas de auto-atención permiten a la red extraer y usar información de contextos de un largo arbitrario (sin la necesidad de pasar el contexto a través de conexiones recurrentes como en una *RNN*†), en las mismas se computa la relevancia de cada elemento en la secuencia dado su contexto anterior *Jurafsky and Martin* (2023).
Cuando se aplican en tareas de visión, un enfoque puede ser crear la secuencia de entrada partiendo la imagen en parches, y alineando secuencialmente los mismos (*Dosovitskiy et al.*, 2020) . 69, 104, 107, 108

Transporte Óptimo

Ver sección B.1.4.1. 124, 125

Valor anotado (*ground truth*)

Denominado *ground truth* en inglés, es el valor de referencia que es tomado como la verdad absoluta para la propiedad a clasificar o inferir dada una instancia del *conjunto de datos*†. Se adquiere a partir de observaciones y mediciones directas, es decir, evidencia empírica. En el contexto del conteo de multitudes es el conjunto de puntos en una imagen que identifican a cada persona, así como la cantidad de puntos que refleja la cantidad total de personas (*Lemoigne and Caner*, 2008). 28, 34, 38, 56–59, 86, 106, 125, 132, 137, 139

Variación de escala

Ver sección 2.2.3. 12, 14, 23, 24, 27, 48, 50, 64, 82, 102, 104, 128, 132

Variación de iluminación

Ver sección 2.2.7. 16

Época

Etapa del *entrenamiento*† en donde el algoritmo realiza una iteración completa sobre la totalidad del *conjunto de entrenamiento*†. Varios métodos de aprendizaje automático tradicional sólo constan de una época; sin embargo, las *redes neuronales*† suelen necesitar varas épocas entre el *conjunto de entrenamiento*†. 123, 136, 140

Índice de Medida de Similitud Estructural (SSIM)

Ver sección 4.3.3.3. 35, 86

Apéndice D

Referencias

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. *Technical report, EPFL*, 06 2010.

[2] Shubhra Aich and Ian Stavness. Global sum pooling: A generalization trick for object counting with small datasets of large images. *arXiv preprint arXiv:1805.11123*, 2018.

[3] AnimatedAI. Animatedai. `https://animatedai.github.io/`, 2022.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[5] Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. Counting in the wild. In *European conference on computer vision*, pages 483–498. Springer, 2016.

[6] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4031–4039, 2017.

[7] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615.

[8] Reza Bahmanyar, Elenora Vig, and Peter Reinartz. Mrcnet: Crowd counting and density map estimation in aerial and ground imagery. *arXiv preprint arXiv:1909.12743*, 2019.

[9] Kunlun Bai, Feb 2019. URL `https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215`.

[10] Shuai Bai, Zhiqun He, Yu Qiao, Hanzhe Hu, Wei Wu, and Junjie Yan. Adaptive dilated network with self-correction supervision for counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016. URL `http://arxiv.org/abs/1611.02167`.

[12] Gonzalo Balduvino and Matías Lorenzo. Redes neuronales convolucionales aplicadas a demosaicing y denoising. Master's thesis, Universidad de la República (Uruguay). Facultad de Ingeniería, 2019.

[13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[14] Serge Beucher. The watershed transformation applied to image segmentation. *Scanning Microscopy*, 1992(6):28, 1992.

[15] Ujwala Bhangale, Suchitra Patil, Vaibhav Vishwanath, Parth Thakker, Amey Bansode, and Devesh Navandhar. Near real-time crowd counting using deep learning approach. *Procedia Computer Science*, 171:770–779, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.04.084. URL `https://www.sciencedirect.com/science/article/pii/S187705092031053X`. Third International Conference on Computing and Network Communications (CoCoNet'19).

[16] Peter J Bickel and David A Freedman. Some asymptotic theory for the bootstrap. *The annals of statistics*, 9(6):1196–1217, 1981.

[17] Lokesh Boominathan, Srinivas SS Kruthiventi, and R Venkatesh Babu. Crowdnet: A deep convolutional network for dense crowd counting. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 640–644, 2016.

[18] Ron Brinkmann. The art and science of digital compositing. In *The Art and Science of Digital Compositing*, page 184. Morgan Kaufmann, 1999.

[19] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[21] Antoni B Chan and Nuno Vasconcelos. Bayesian poisson regression for crowd counting. In *2009 IEEE 12th international conference on computer vision*, pages 545–551. IEEE, 2009.

[22] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–7. IEEE, 2008.

[23] Binghui Chen and Weihong Deng. Energy confused adversarial metric learning for zero-shot image retrieval and clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8134–8141, 2019.

[24] Binghui Chen, Zhaoyi Yan, Ke Li, Pengyu Li, Biao Wang, Wangmeng Zuo, and Lei Zhang. Variational attention: Propagating domain-specific knowledge for multi-domain learning in crowd counting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16065–16075, 2021.

[25] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang. Feature mining for localised crowd counting. In *Bmvc*, volume 1, page 3, 2012.

[26] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. URL `http://arxiv.org/abs/1606.00915`.

[27] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017a.

[28] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017b.

[29] Zhuojun Chen, Junhao Cheng, Yuchen Yuan, Dongping Liao, Yizhou Li, and Jiancheng Lv. Deep density-aware count regressor. *arXiv preprint arXiv:1908.03314*, 2019.

[30] British Broadcasting Corporation. Shanghai new year crush kills 36, Jan 2015. URL https://www.bbc.com/news/world-asia-china-30646918.

[31] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[32] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[33] Anthony C Davies, Jia Hong Yin, and Sergio A Velastin. Crowd monitoring using image processing. *Electronics & Communication Engineering Journal*, 7(1):37–47, 1995.

[34] Julie Delon and Agnes Desolneux. A wasserstein-type distance in the space of gaussian mixture models. *SIAM Journal on Imaging Sciences*, 13(2):936–970, 2020.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[36] Arden Dertat, Nov 2017. URL https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

[37] Xinghao Ding, Zhirui Lin, Fujin He, Yu Wang, and Yue Huang. A deeply-recursive convolutional network for crowd counting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1942–1946. IEEE, 2018.

[38] Phuc Thinh Do. Attention in crowd counting using the transformer and density map to improve counting result. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 65–70. IEEE, 2021.

[39] Google Developers Documentation. Generative adversarial networks, Oct. 8 2019. URL https://developers.google.com/machine-learning/gan.

[40] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[41] Zhipeng Du, Jiankang Deng, and Miaojing Shi. Domain-general crowd counting in unseen scenarios, 2022. URL https://arxiv.org/abs/2212.02573.

[42] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR, 2018.

[43] IBM Cloud Education. What are recurrent neural networks?, Sep 2020. URL https://www.ibm.com/cloud/learn/recurrent-neural-networks.

[44] Yanyan Fang, Biyun Zhan, Wandi Cai, Shenghua Gao, and Bo Hu. Locality-constrained spatial transformer network for video crowd counting. *arXiv preprint arXiv:1907.07911*, 2019.

[45] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[46] Min Fu, Pei Xu, Xudong Li, Qihe Liu, Mao Ye, and Ce Zhu. Fast crowd density estimation with convolutional neural networks. *Engineering Applications of Artificial Intelligence*, 43:81–88, 2015.

[47] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[48] Guangshuai Gao, Junyu Gao, Qingjie Liu, Qi Wang, and Yunhong Wang. Cnn-based density estimation and crowd counting: A survey. *CoRR*, abs/2003.12783, 2020. URL `https://arxiv.org/abs/2003.12783`.

[49] Jiaqi Gao, Zhizhong Huang, Yiming Lei, James Z. Wang, Fei-Yue Wang, and Junping Zhang. S^2fpr: Crowd counting via self-supervised coarse to fine feature pyramid ranking, 2022a.

[50] Junyu Gao and Xin Zeng. Awesome crowd counting, 2022. URL `https://github.com/gjy3035/Awesome-Crowd-Counting`.

[51] Junyu Gao, Wei Lin, Bin Zhao, Dong Wang, Chenyu Gao, and Jun Wen. C^3 framework: An open-source pytorch code for crowd counting. *arXiv preprint arXiv:1907.02724*, 2019a. URL `https://github.com/gjy3035/C-3-Framework`.

[52] Junyu Gao, Qi Wang, and Xuelong Li. Pcc net: Perspective crowd counting via spatial convolutional network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(10):3486–3498, 2019b.

[53] Junyu Gao, Maoguo Gong, and Xuelong Li. Congested crowd instance localization with dilated convolutional swin transformer. *Neurocomputing*, 513:94–103, 2022b.

[54] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[55] Songchenchen Gong. *Real-time implementation of counting people in a crowd on the embedded reconfigurable architecture on the unmanned aerial vehicle*. Theses, Université Bourgogne Franche-Comté, November 2020. URL `https://tel.archives-ouvertes.fr/tel-03100744`.

[56] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[57] Siqi Gu and Zhichao Lian. A unified multi-task learning framework of real-time drone supervision for crowd counting. *arXiv preprint arXiv:2202.03843*, 2022.

[58] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Onoro-Rubio. Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 423–431. Springer, 2015.

[59] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, 6:610–621, 1973.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. URL `http://arxiv.org/abs/1512.03385`.

[61] John R. Hershey and Peder A. Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–317–IV–320, 2007. doi: 10.1109/ICASSP.2007.366913.

[62] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[63] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[64] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):251–257, 1991.

[65] Mohammad Hossain, Mehrdad Hosseinzadeh, Omit Chanda, and Yang Wang. Crowd counting using scale-aware attention networks. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 1280–1288. IEEE, 2019.

[66] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H Hsu. Drone-based object counting by spatially regularized regional proposal network. In *Proceedings of the IEEE international conference on computer vision*, pages 4145–4153, 2017.

[67] Di Hu, Lichao Mou, Qingzhong Wang, Junyu Gao, Yuansheng Hua, Dejing Dou, and Xiao Xiang Zhu. Ambient sound helps: Audiovisual crowd counting in extreme conditions. *arXiv preprint arXiv:2005.07097*, 2020a.

[68] Yutao Hu, Xiaolong Jiang, Xuhui Liu, Baochang Zhang, Jungong Han, Xianbin Cao, and David S. Doermann. Nas-count: Counting-by-density with neural architecture search. *CoRR*, abs/2003.00217, 2020b. URL `https://arxiv.org/abs/2003.00217`.

[69] David H Hubel and Torsten N Wiesel. Early exploration of the visual cortex. *Neuron*, 20(3): 401–412, 1998.

[70] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554, 2013.

[71] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–546, 2018a.

[72] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–546, 2018b.

[73] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[74] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. URL `http://arxiv.org/abs/1506.02025`.

[75] Xiaoheng Jiang, Li Zhang, Mingliang Xu, Tianzhu Zhang, Pei Lv, Bing Zhou, Xin Yang, and Yanwei Pang. Attention scaling for crowd counting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4706–4715, 2020.

[76] Xiaolong Jiang, Zehao Xiao, Baochang Zhang, Xiantong Zhen, Xianbin Cao, David Doermann, and Ling Shao. Crowd counting and density estimation by trellis encoder-decoder networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6133–6142, 2019.

[77] A Josuttes, A Shubra, I Stavness, C Pozniak, and S Shirtliffe. Utilizing deep learning to predict the number of panicles in wheat (triticum aestivum). In *Soils and Crops Workshop*, 2018.

[78] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Stanford, USA, 3rd edition, 2023. URL `https://web.stanford.edu/~jurafsky/slp3/`.

[79] Di Kang, Debarun Dhar, and Antoni Chan. Incorporating side information by adaptive convolution. *Advances in Neural Information Processing Systems*, 30, 2017.

[80] Muhammad Asif Khan, Hamid Menouar, and Ridha Hamila. Revisiting crowd counting: State-of-the-art, trends, and future perspectives. *arXiv preprint arXiv:2209.07271*, 2022. URL `https://arxiv.org/abs/2209.07271`.

[81] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[82] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pages 2796–2804. PMLR, 2018.

[83] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[84] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[85] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[86] Yinjie Lei, Yan Liu, Pingping Zhang, and Lingqiao Liu. Towards using count-level weak supervision for crowd counting. *Pattern Recognition*, 109:107616, 2021.

[87] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 878–885. IEEE, 2005.

[88] Yves Lemoigne and Alessandra Caner. Molecular imaging: Computer reconstruction and practice. In *Molecular Imaging: Computer Reconstruction and Practice*, pages 269–277. Springer, 2008.

[89] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. *Advances in neural information processing systems*, 23, 2010.

[90] Haopeng Li, Lingbo Liu, Kunlin Yang, Shinan Liu, Junyu Gao, Bin Zhao, Rui Zhang, and Jun Hou. Video crowd localization with multifocus gaussian neighborhood attention and a large-scale benchmark. *IEEE Transactions on Image Processing*, 31:6032–6047, 2022a. doi: 10.1109/TIP.2022.3205210.

[91] Min Li, Zhaoxiang Zhang, Kaiqi Huang, and Tieniu Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *2008 19th international conference on pattern recognition*, pages 1–4. IEEE, 2008.

[92] Pengfei Li, Min Zhang, Jian Wan, and Ming Jiang. Dmpnet: densely connected multi-scale pyramid networks for crowd counting. *PeerJ Computer Science*, 8:e902, 2022b.

[93] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. *CoRR*, abs/1802.10062, 2018. URL `http://arxiv.org/abs/1802.10062`.

[94] Dongze Lian, Jing Li, Jia Zheng, Weixin Luo, and Shenghua Gao. Density map regression guided detection network for rgb-d crowd counting and localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[95] Dingkang Liang, Xiwu Chen, Wei Xu, Yu Zhou, and Xiang Bai. Transcrowd: weakly-supervised crowd counting with transformers. *Science China Information Sciences*, 65(6):1–14, 2022a.

[96] Dingkang Liang, Wei Xu, and Xiang Bai. An end-to-end transformer model for crowd localization. *arXiv preprint arXiv:2202.13065*, 2022b.

[97] Mei Kuan Lim, Ven Jyn Kok, Chen Change Loy, and Chee Seng Chan. Crowd saliency detection via global similarity structure. In *2014 22nd International Conference on Pattern Recognition*, pages 3957–3962. IEEE, 2014.

[98] Hui Lin, Zhiheng Ma, Rongrong Ji, Yaowei Wang, and Xiaopeng Hong. Boosting crowd counting via multifaceted attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19628–19637, 2022.

[99] Sheng-Fuu Lin, Jaw-Yeh Chen, and Hung-Xin Chao. Estimation of number of people in crowded scenes using perspective transformation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 31(6):645–654, 2001.

[100] Zhe Lin and Larry S Davis. Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE transactions on pattern analysis and machine intelligence*, 32(4), 2010.

[101] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160, 1976.

[102] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *CoRR*, abs/1712.00559, 2017. URL `http://arxiv.org/abs/1712.00559`.

[103] Jiang Liu, Chenqiang Gao, Deyu Meng, and Alexander G Hauptmann. Decidenet: Counting varying density crowds through attention guided detection and density estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5197–5206, 2018a.

[104] Liang Liu, Hao Lu, Hongwei Zou, Haipeng Xiong, Zhiguo Cao, and Chunhua Shen. Weighing counts: Sequential crowd counting by reinforcement learning. In *European Conference on Computer Vision*, pages 164–181. Springer, 2020.

[105] Lingbo Liu, Hongjun Wang, Guanbin Li, Wanli Ouyang, and Liang Lin. Crowd counting using deep recurrent spatial-aware network. *CoRR*, abs/1807.00601, 2018b. URL `http://arxiv.org/abs/1807.00601`.

[106] Lingbo Liu, Jiaqi Chen, Hefeng Wu, Guanbin Li, Chenglong Li, and Liang Lin. Cross-modal collaborative representation learning and a large-scale rgbt benchmark for crowd counting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4823–4833, 2021a.

[107] Weizhe Liu, Krzysztof Lis, Mathieu Salzmann, and Pascal Fua. Geometric and physical constraints for drone-based head plane crowd density estimation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 244–249. IEEE, 2019a.

[108] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5099–5108, 2019b.

[109] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5099–5108, 2019c.

[110] Xialei Liu, Joost Van De Weijer, and Andrew D Bagdanov. Exploiting unlabeled data in cnns by self-supervised learning to rank. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1862–1878, 2019d.

[111] Yuting Liu, Miaojing Shi, Qijun Zhao, and Xiaofang Wang. Point in, box out: Beyond counting persons in crowds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6469–6478, 2019e.

[112] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021b.

[113] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL `http://arxiv.org/abs/1411.4038`.

[114] Hao Lu, Zhiguo Cao, Yang Xiao, Bohan Zhuang, and Chunhua Shen. Tasselnet: counting maize tassels in the wild via local counts regression network. *Plant methods*, 13(1):1–17, 2017.

[115] Yiming Ma, Victor Sanchez, and Tanaya Guha. Fusioncount: Efficient crowd counting via multiscale feature fusion. *arXiv preprint arXiv:2202.13660*, 2022.

[116] Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Yihong Gong. Bayesian loss for crowd count estimation with point supervision. *CoRR*, abs/1908.03684, 2019. URL `http://arxiv.org/abs/1908.03684`.

[117] Mark Marsden, Kevin McGuinness, Suzanne Little, Ciara E Keogh, and Noel E O'Connor. People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8070–8079, 2018.

[118] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[119] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

[120] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[121] Mahdi Maktabdar Oghaz, Anish R Khadka, Vasileios Argyriou, and Paolo Remagnino. Content-aware density map for crowd counting and density estimation. *arXiv preprint arXiv:1906.07258*, 2019.

[122] Min-hwan Oh, Peder A. Olsen, and Karthikeyan Natesan Ramamurthy. Crowd counting with decomposed uncertainty. *CoRR*, abs/1903.07427, 2019. URL `http://arxiv.org/abs/1903.07427`.

[123] Daniel Onoro-Rubio and Roberto J López-Sastre. Towards perspective-free object counting with deep learning. In *European conference on computer vision*, pages 615–629. Springer, 2016.

[124] Nikos Paragios and Visvanathan Ramesh. A mrf-based approach for real-time subway monitoring. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[125] Tao Peng, Qing Li, and Pengfei Zhu. Rgb-t crowd counting from drone: A benchmark and mmccn network. In *Proceedings of the Asian Conference on Computer Vision*, 2020a.

[126] Tao Peng, Qing Li, and Pengfei Zhu. Rgb-t crowd counting from drone: A benchmark and mmccn network. In *Proceedings of the Asian Conference on Computer Vision*, 2020b.

[127] Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[128] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *CoRR*, abs/1802.03268, 2018. URL `http://arxiv.org/abs/1802.03268`.

[129] Viet-Quoc Pham, Tatsuo Kozakaya, Osamu Yamaguchi, and Ryuzo Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 3253–3261, 2015.

[130] Enciso Pérez and Laura Zingaretti. A guide for using deep learning for complex trait genomic prediction. *Genes*, 10:553, 07 2019. doi: 10.3390/genes10070553.

[131] Viresh Ranjan, Mubarak Shah, and Minh Hoai Nguyen. Crowd transformer network. *arXiv preprint arXiv:1904.02774*, 2019.

[132] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[133] Deepak Babu Sam, Skand Vishwanath Peri, Mukuntha N. S., Amogh Kamath, and R. Venkatesh Babu. Locate, size and count: Accurately resolving people in dense crowds via detection. *CoRR*, abs/1906.07538, 2019. URL `http://arxiv.org/abs/1906.07538`.

[134] Prithiv Sassisegarane. Dense and sparse crowd counting methods and techniques: A review, Aug 2021. URL `https://medium.com/nanonets/dense-and-sparse-crowd-counting-methods-and-techniques-a-review-bae04fdbf062`.

[135] Miaojing Shi, Zhaohui Yang, Chao Xu, and Qijun Chen. Revisiting perspective information for efficient crowd counting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7279–7288, 2019.

[136] Xiaowen Shi, Xin Li, Caili Wu, Shuchen Kong, Jing Yang, and Liang He. A real-time deep network for crowd counting. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2328–2332. IEEE, 2020.

[137] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[138] Vishwanath Sindagi, Rajeev Yasarla, and Vishal MM Patel. Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[139] Vishwanath A Sindagi and Vishal M Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[140] Vishwanath A. Sindagi and Vishal M. Patel. Multi-level bottom-top and top-bottom feature fusion for crowd counting. *CoRR*, abs/1908.10937, 2019. URL `http://arxiv.org/abs/1908.10937`.

[141] Skyl.ai. Evaluating a machine learning model. Medium, Sep. 10 2019. URL `https://medium.com/@skyl/evaluating-a-machine-learning-model-7cab1f597046`.

[142] Qingyu Song, Changan Wang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Jian Wu, and Jiayi Ma. To choose or to fuse? scale selection for crowd counting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2576–2583, 2021.

[143] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[144] Guolei Sun, Yun Liu, Thomas Probst, Danda Pani Paudel, Nikola Popovic, and Luc Van Gool. Boosting crowd counting with transformers. *arXiv preprint arXiv:2105.10926*, 2021.

[145] Yiming Sun, Bing Cao, Pengfei Zhu, and Qinghua Hu. Drone-based rgb-infrared cross-modality vehicle detection via uncertainty-aware learning. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2022. doi: 10.1109/TCSVT.2022.3168279.

[146] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[147] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[148] Haihan Tang, Yi Wang, and Lap-Pui Chau. Tafnet: A three-stream adaptive fusion network for rgb-t crowd counting. *arXiv preprint arXiv:2202.08517*, 2022.

[149] Matija Teršek and Maša Kljun. Lwcc: A lightweight crowd counting library for python, 2021. URL `https://github.com/tersekmatija/lwcc`.

[150] Pongpisit Thanasutives, Ken-ichi Fukui, Masayuki Numao, and Boonserm Kijsirikul. Encoder-decoder based convolutional neural networks with multi-scale-aware modules for crowd counting. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2382–2389. IEEE, 2021.

[151] Yan Tian, Leonid Sigal, Hernán Badino, Fernando De la Torre, and Yong Liu. Latent gaussian mixture regression for human pose estimation. In *Asian Conference on Computer Vision*, pages 679–690. Springer, 2010.

[152] Ye Tian, Xiangxiang Chu, and Hongpeng Wang. Cctrans: Simplifying and improving crowd counting with transformer. *arXiv preprint arXiv:2109.14483*, 2021.

[153] Yukun Tian, Yiming Lei, Junping Zhang, and James Z Wang. Padnet: Pan-density crowd counting. *IEEE Transactions on Image Processing*, 29:2714–2727, 2019.

[154] Ibrahim Saygin Topkaya, Hakan Erdogan, and Fatih Porikli. Counting people by clustering person detector outputs. In *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 313–318. IEEE, 2014.

[155] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[156] Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1713–1727, 2008.

[157] Varun Kannadi Valloli and Kinal Mehta. W-net: Reinforced u-net for density map estimation. *arXiv preprint arXiv:1903.11249*, 2019.

[158] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[159] Cédric Villani. Optimal transport. old and new, volume 338 of. *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, 2009.

[160] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[161] Visailabs. Technical metrics required to assess of human/people detection solution, Feb 2021. URL `https://visailabs.com/technical-metrics-required-to-asses-of-people-detection-solution/`.

[162] Elad Walach and Lior Wolf. Learning to count with cnn boosting. In *European conference on computer vision*, pages 660–676. Springer, 2016.

[163] Jia Wan and Antoni Chan. Adaptive density map generation for crowd counting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1130–1139, 2019a.

[164] Jia Wan and Antoni B. Chan. Adaptive density map generation for crowd counting. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1130–1139, 2019b.

[165] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai Nguyen. Distribution matching for crowd counting. *Advances in neural information processing systems*, 33:1595–1607, 2020a.

[166] Chuan Wang, Hua Zhang, Liang Yang, Si Liu, and Xiaochun Cao. Deep people counting in extremely dense crowds. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1299–1302, 2015.

[167] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10): 3349–3364, 2020b.

[168] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8198–8207, 2019.

[169] Qi Wang, Junyu Gao, Wei Lin, and Xuelong Li. Nwpu-crowd: A large-scale benchmark for crowd counting and localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020c. doi: 10.1109/TPAMI.2020.3013269. URL `https://arxiv.org/pdf/2001.03360.pdf`.

[170] Qian Wang and Toby P Breckon. Crowd counting via segmentation guided attention networks and curriculum loss. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[171] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

[172] Xing Wei, Yuanrui Kang, Jihao Yang, Yunfeng Qiu, Dahu Shi, Wenming Tan, and Yihong Gong. Scene-adaptive attention network for crowd counting. *arXiv preprint arXiv:2112.15509*, 2021.

[173] Longyin Wen, Dawei Du, Pengfei Zhu, Qinghua Hu, Qilong Wang, Liefeng Bo, and Siwei Lyu. Drone-based joint density map estimation, localization and tracking with space-time multi-scale attention network. *arXiv preprint arXiv:1912.01811*, 2019.

[174] Paul Werbos. Beyond regression:"new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.

[175] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

[176] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[177] Xingjiao Wu, Yingbin Zheng, Hao Ye, Wenxin Hu, Jing Yang, and Liang He. Adaptive scenario discovery for crowd counting. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2382–2386. IEEE, 2019.

[178] Chenfeng Xu, Dingkang Liang, Yongchao Xu, Song Bai, Wei Zhan, Xiang Bai, and Masayoshi Tomizuka. Autoscale: Learning to scale for crowd counting. *International Journal of Computer Vision*, 130(2):405–434, 2022.

[179] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[180] Zhaoyi Yan, Yuchen Yuan, Wangmeng Zuo, Xiao Tan, Yezhen Wang, Shilei Wen, and Errui Ding. Perspective-guided convolution networks for crowd counting. *CoRR*, abs/1909.06966, 2019a. URL `http://arxiv.org/abs/1909.06966`.

[181] Zhaoyi Yan, Yuchen Yuan, Wangmeng Zuo, Xiao Tan, Yezhen Wang, Shilei Wen, and Errui Ding. Perspective-guided convolution networks for crowd counting. In *Proceedings of the IEEE international conference on computer vision*, 2019b.

[182] Zhaoyi Yan, Pengyu Li, Biao Wang, Dongwei Ren, and Wangmeng Zuo. Towards learning multi-domain crowd counting. *IEEE Trans. Circuits Syst. Video Technol*, 2021.

[183] Linwei Ye, Mrigank Rochan, Zhi Liu, and Yang Wang. Cross-modal self-attention network for referring image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10502–10511, 2019.

[184] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[185] Lingke Zeng, Xiangmin Xu, Bolun Cai, Suo Qiu, and Tong Zhang. Multi-scale convolutional neural networks for crowd counting. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 465–469. IEEE, 2017.

[186] Aston Zhang, Zack C. Lipton, Mu Li, and Alex J. Smola. Dive into deep learning, 2021a.

[187] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 833–841, 2015.

[188] Lu Zhang, Miaojing Shi, and Qiaobo Chen. Crowd counting via scale-adaptive convolutional neural network. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1113–1121. IEEE, 2018.

[189] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3008, 2021b.

[190] Qi Zhang and Antoni B Chan. Wide-area crowd counting via ground-plane density maps and multi-view fusion cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 8297–8306, 2019.

[191] Shanghang Zhang, Guanhang Wu, Joao P Costeira, and Jose MF Moura. Understanding traffic density from large-scale web camera data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5898–5907, 2017.

[192] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016a.

[193] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016b.

[194] J. Zhou, L. Zhang, D. Jiawei, X. Peng, Z. Fang, Z. Xiao, and H. Zhu. Locality-aware crowd counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:1–1, feb 2021. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3056518.

[195] Liang Zhu, Zhijian Zhao, Chao Lu, Yining Lin, Yao Peng, and Tangren Yao. Dual path multi-scale fusion networks with attention for crowd counting. *CoRR*, abs/1902.01115, 2019. URL `http://arxiv.org/abs/1902.01115`.

[196] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021a. doi: 10.1109/TPAMI.2021.3119563.

[197] X Zhu, W Su, LW Lu, B Li, XG Wang, and Deformable DETR Dai J F. Deformable transformers for end-to-end object detection. In *Proceedings of the 9th International Conference on Learning Representations. Virtual Event, Austria: OpenReview. net*, 2021b.

[198] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL `http://arxiv.org/abs/1611.01578`.

Esta es la última página.
Compilado el 4 de agosto de 2023.