UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

Proyecto de Grado
Licenciatura en Computación

Aprendizaje Profundo para el procesamiento de Imágenes - Optimización del conjunto de datos de entrenamiento

---

Juan Ignacio Cabrera García
Noviembre 2021

Tutores:
Mercedes Marzoa Tanco
Mario González Olmedo

Resumen

Los proyectos de aprendizaje profundo se enfrentan a múltiples desafíos, entre ellos el gran coste de generar conjuntos de datos etiquetados a gran escala, su depuración y el impacto en el resultado obtenido. La elección de qué datos utilizar, cuáles etiquetar y cómo afectan a estos modelos tienen un alto impacto en la viabilidad económica y computacional de los proyectos. Por lo tanto, es crucial elegir qué herramientas y técnicas utilizar durante el proceso de creación de estos conjuntos de datos en varios dominios, entre ellos la visión artificial.

En este proyecto se propone analizar el estado del arte de las herramientas, técnicas y modelos que permiten optimizar el conjunto de imágenes sobre el cual se entrena un modelo de aprendizaje profundo para la visión artificial. Además, se diseña una solución para la selección de imágenes relevantes a etiquetar (a partir de un banco de imágenes sin etiquetas) para entrenar los modelos y se implementa una prueba de concepto de esta solución sobre un caso de uso real que supera significativamente el etiquetado aleatorio.

Índice

Capítulo 1

Introducción

1.1. Presentación del problema y contexto

Cuando nos plantean problemas relacionados con la inteligencia artificial (IA) generalmente pensamos en construir algoritmos y modelos para resolverlos, tomando un conjunto de datos de referencia. Invertimos horas de trabajo en implementar, testear y corregir esos modelos, sin tener en cuenta que varias veces, el problema de obtener resultados no satisfactorios está relacionado con los datos con los que estamos trabajando. Andrew Ng [1] hace hincapié en la importancia de los datos en la construcción de algoritmos de IA, afirmando que estos sistemas no son solo código como el software tradicional, sino que son código más datos.

Si bien hay varios conjuntos de datos muy buenos, y han sido punto de partida de muchos trabajos de investigación del aprendizaje automático, no todos se adaptan a las necesidades de cada problema. Además, pueden causar inconvenientes con algunas anotaciones sesgadas [11] como ocurrió con los sistemas de reconocimiento de imágenes de Google [2] que confundía algunas personas con gorilas, o con los sistemas de LinkedIn [3] que recomendaba empleos mejor remunerados a personas de sexo masculino.

Los algoritmos de IA aprenden una realidad representada por los datos que fueron utilizados para sus entrenamientos y toman decisiones a partir de ello, por lo tanto si los datos no son de buena calidad o no representan todo el dominio en el que se quiere trabajar, el sistema no aprenderá de forma correcta y fallará. Por lo tanto, es de vital importancia tener un buen conjunto de datos para lograr el éxito de nuestra solución.

Una de las tareas esenciales y que requiere gran parte del tiempo en proyectos relacionados con IA es la clasificación y etiquetado de los datos. El etiquetado manual, si bien es el más preciso, ya que por el momento ningún algoritmo se puede comparar con el ojo humano, es el más costoso en términos de tiempo y esfuerzo humano, dos recursos generalmente valiosos y acotados [11].

Basado en el libro **Human-in-the-Loop, Machine Learning** [4] (en el que se explican varias técnicas para seleccionar datos de forma eficiente para el etiquetado manual) en este proyecto implemento una solución alternativa donde el etiquetado de los datos es una tarea que se realiza en cooperación entre el sistema y el humano, lo que en inglés se conoce como Human-in-the-Loop, buscando la intervención humana donde el sistema tiene más incertidumbre al momento de clasificar. Si bien en este proyecto trabajo sobre la clasificación de imágenes para tareas de visión artificial, los algoritmos implementados son fácilmente adaptables a otro tipo de datos, como puede ser texto para tareas de procesamiento de lenguaje natural (PLN).

1.2. Estructura del documento

Este documento está compuesto por seis capítulos organizados de la siguiente manera.

En el Capítulo 2 se presenta el estado del arte, donde se introducen una serie de trabajos relacionados con el tema y se da una descripción de la solución propuesta en cada uno de ellos.

En el Capítulo 3 se presenta el marco teórico, donde se describen conceptos y técnicas utilizadas para el desarrollo del proyecto.

En el Capítulo 4 se presenta la solución propuesta, donde se describe el trabajo realizado, cómo éste se vincula con los conceptos detallados en el marco teórico y las tecnologías y herramientas utilizadas.

En el Capítulo 5 se presentan los experimentos realizados, los cuales se dividen en dos partes: la primera es para visualizar en forma gráfica el funcionamiento de los algoritmos propuestos y la segunda es para comparar el rendimiento de la solución propuesta en comparación con un algoritmo base.

En el Capítulo 6 se presenta la conclusión, describiendo lo abarcado por este proyecto con respecto al objetivo inicial del mismo y los trabajos a realizar a futuro como forma de continuar con la investigación y el desarrollo de este proyecto.

Capítulo 2

Estado del arte

2.1. Introducción

En este capítulo se presentan una serie de artículos de relevancia relacionados con la temática tratada en este proyecto. En las primeras tres secciones se presentan artículos que están fuertemente relacionados con la solución propuesta en este proyecto, por lo que profundizaré en los detalles de la tarea realizada dando una descripción de la misma, pruebas realizadas y resultados obtenidos.
En las últimas dos secciones, se presentan artículos que si bien están enmarcados en la tarea de la construcción de un conjunto de datos, la tarea realizada es diferente, por lo tanto solo doy una breve descripción de cada uno, sin entrar en los detalles.

2.2. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop

Trabajo presentado por Yu et al. en el año 2015 [5] donde presentan un framework que utiliza aprendizaje profundo con humanos en el ciclo de aprendizaje para etiquetar un conjunto de imágenes a gran escala. Crean un sistema de propagación de etiquetado que amplifica automáticamente el esfuerzo humano creando un conjunto de imágenes clasificadas en 10 escenas y 20 categorías de objetos.

El trabajo se divide en dos grandes tareas:

- Recopilación de imágenes
- Procesamiento

2.2.1. Recopilación de imágenes

Para recopilar las imágenes generan consultas para las categorías de escenas y objetos combinando el nombre o sinónimo de la escena u objeto con un adjetivo adecuado. En el caso de las escenas los adjetivos provienen de una lista de los más populares del inglés. Estas consultas les permite aumentar el límite habitual en los resultados de búsqueda recopilando un conjunto grande y diverso de URLs de imágenes candidatas mediante la búsqueda de palabras clave en el motor de búsqueda de imágenes de Google.
Luego se eliminan las URL duplicadas, se descargan las imágenes y por último, se eliminan las de dimensión menor a 256.

2.2.2. Procesamiento

El proceso consta de un ciclo de cuatro pasos principales:

- Pedir a personas que etiqueten un pequeño subconjunto de imágenes elegidas al azar.

- Entrenar un clasificador en ese subconjunto.
- Ejecutar el clasificador sobre el conjunto de imágenes sin etiquetar para que prediga etiquetas .
- Seleccionar las imágenes clasificadas como dudosas por el clasificador y agregarlas al conjunto de imágenes no etiquetadas.

En el último paso de la iteración, se calculan dos umbrales para la discriminacion. Las imágenes que puntúan por encima o por debajo de estos umbrales se etiquetan como positivas o negativas respectivamente. Todas las imágenes que puntúan entre estos dos umbrales se consideran imágenes dudosas y se envían a la siguiente iteración.

El objetivo de este proceso es obtener etiquetas correctas para todos los ejemplos de un conjunto específico de imágenes, en lugar de entrenar un clasificador que generalize a nuevos ejemplos. La principal implicancia de esta diferencia es que es aceptable que el sistema sobreajuste un modelo para adaptarse localmente a su conjunto objetivo, sin considerar la posibilidad de generalización fuera del conjunto.
El método iterativo aprende una cascada de clasificadores, cada uno de los cuales divide un conjunto de imágenes en subconjuntos positivos, negativos y sin etiquetar. Los ejemplos positivos se agregan al conjunto de datos; los ejemplos negativos se descartan; y el subconjunto sin etiquetar pasa a la siguiente iteración para su posterior procesamiento. La cascada termina cuando el subconjunto sin etiquetar es lo suficientemente pequeño como para que todas las imágenes se puedan etiquetar manualmente.

Para el etiquetado manual utilizaron Amazon Mechanical Turk (AMT) [29] logrando obtener etiquetas rápidamente y a un costo mínimo, proporcionándoles una interfaz adecuada con imágenes de ejemplos, definiciones de cada etiqueta, etc. Para asegurar la calidad de las etiquetas adoptaron el enfoque del etiquetado redundante, manteniendo solamente etiquetas doblemente confirmadas.

Realizaron pruebas para medir la precisión del etiquetado comparando 2000 imágenes de 11 categorías con un etiquetado manual hecho por expertos, los experimentos indican que la precisión ronda el 90%.

Verificaron cuánto amplifica el esfuerzo de etiquetado humano, es decir, la relación entre el número de imágenes etiquetadas por el pipeline implementado frente al número de imágenes etiquetadas por personas. El resultado indica que el pipeline propaga las etiquetas hechas por el humano a otras imágenes con similares características en un promedio de 40 veces.

Probaron si el nuevo conjunto de datos puede ayudar a los modelos actuales a mejorar el rendimiento de clasificación. Para eso hicieron dos tipos de pruebas:

- Usaron el modelo AlexNet [14] entrenado en PLACES [28], tomaron 10 categorcoías de LSUN y compararon el modelo, para eso hicieron por separado dos ajustes finos (fine-tuning en inglés) del mismo modelo. Por un lado ajustaron solo con los datos de PLACES. Por otro lado ajustaron con los datos de PLACES y LSUN. La evaluación la hicieron con datos de PLACES, compararon el porcentaje de error y vieron que la prueba es desfavorable para LSUN concluyendo que es debido al sesgo del conjunto de datos.
- Compararon los modelos AlexNet y VGG [27] entrenando de dos formas. Primero entrenaron ambos modelos con LSUN e hicieron un ajuste fino (fine-tuning) de cada modelo con imágenes de PASCAL VOC 2012 [30]. Luego repitieron el experimento entrenado cada modelo con LSUN, pero inicializados con ImageNet [12]. Concluyeron que el modelo entrenado con LSUN de cero funciona significativamente mejor.

Para medir la representatividad de las imágenes de objetos, entrenaron AlexNet con las imágenes de objetos de LSUN por un lado y con ImageNet por el otro usando las mismas categorías de objetos de LSUN. Luego compararon el filtro de la primera capa y observaron que el patrón de filtro aprendido con los datos de LSUN es más limpio (con menos ruido) que los filtros aprendidos con los datos de ImageNet.

2.3. Construction of Diverse Image Datasets from Web Collections with Limited Labeling

En el año 2016, Mithun et al. [6] presentaron este trabajo que recopila imágenes en forma incremental teniendo en cuenta la representatividad y la diversidad.
El proceso se realiza en tres grandes pasos:

- Recopilación de imágenes y extracción de características
- Selección diversa de conjuntos representativos
- Aprendizaje activo para el etiquetado de imágenes

2.3.1. Recopilación de imágenes y extracción de características

Recopilan imágenes de diferentes fuentes web, utilizando un esquema de expansión de consultas en Google Search y ConceptNet [31], seleccionando solo sinónimos y frases derivadas como consultas ampliadas.
Una vez descargadas y filtradas las imágenes de baja calidad, por ejemplo, desenfocadas o borrosas, demasiado blancas o negras, vacías o demasiado pequeñas, se procesan con una red neuronal convolucional (CNN) a la que se quita la última capa de la red para la extracción de características [32].

2.3.2. Selección diversa de conjuntos representativos

Para resolver el problema de diversidad y representatividad proponen seleccionar un subconjunto X del lote de imágenes actual que sean representativas del mismo y diferentes del conjunto de imágenes actual D. La idea es establecer lo que se denomina un modelo parsimonioso (en inglés, sparse model), realizando una regularización l^1 en las filas de la

matriz de dispersión [33]. Introduciendo el regularizador de dispersión de filas, el problema lo plantean como:

$$\min \ ||X - XZ||_F^2$$
$$\text{s.t. } ||Z||_{2,1} \leq \tau, \ ||D^T XZ||_F^2 \leq \kappa$$

Donde:

- Z : matriz de dispersión.
- X : matriz de características de todas las imágenes del lote actual.
- D : matriz de características del conjunto de datos actual.
- τ y k : parámetros de compensación.
- $||A||_F$: norma de Frobenius de A.
- $||Z||_{2,1} \simeq \sum_{i=1}^{N} \left||z^i\right||_2$

Lo que pretenden con la ecuación anterior es encontrar un subconjunto de imágenes pequeño con el cual se pueda reconstruir de la mejor manera todas las imágenes del conjunto X mediante combinaciones lineales.

La primera restricción induce parsimonia a nivel de fila. La matriz de dispersión Z contiene pocas filas distintas de cero que permite seleccionar el conjunto representativo. La segunda restricción selecciona imágenes que estén menos correlacionadas con las imágenes del conjunto actual (diversidad).

La optimización de la ecuación intenta obtener un subconjunto pequeño de imágenes de X no redundantes con las imágenes previamente seleccionadas. Para ello utilizan multiplicadores de Lagrange, por lo que el problema se escribe como:

$$\min \frac{1}{2}\|X - XZ\|_F^2 + \lambda\|Z\|_{2,1} + \frac{\alpha}{2}||D^T XZ||_F^2$$

Obteniendo así, el conjunto de imágenes representativo diverso al que le llaman Y.

2.3.3. Aprendizaje activo para el etiquetado de imágenes

El siguiente objetivo es estimar la similitud de cada imagen en Y con las imágenes del diccionario D. En función de la puntuación de similitud, el módulo de aprendizaje activo determina las imágenes que se etiquetan.

El objetivo que plantean aquí es encontrar la probabilidad de que una imagen pertenezca a una clase en particular contando solo con algunos ejemplos de la misma clase.

Para cada muestra $y_i \in Y$, se calcula su representación parsimoniosa $c_i \in C$ con base en el conjunto D.

Una muestra $y_i \in Y$ es relevante dependiendo de qué tan bien se asocian las entradas distintas de cero en la estimación $c_i \in C$ con las columnas de D.

El problema lo plantean como:

$$\min \ ||Y - DC||_F^2 \ \text{ s.t.} ||c_i||_1 \leq s$$

Donde:

- C : matriz de dispersión
- Y : matriz de características de las imágenes representativas.
- D : matriz de características del conjunto de datos actual.
- s : parámetro de compensación.
- $||A||_F$: norma de Frobenius de A.

Una vez más, para la optimización de la ecuación utilizan los multiplicadores de Lagrange.
Calculada la matriz C, la puntuación de similitud para cada imagen en el conjunto representativo la calculan de la siguiente manera:

$$\zeta_i = 1 - \frac{\|y_i - Dc_i\|_2}{\|y_i\|_2}$$

Luego hacen la siguiente discriminación:

- Si la puntuación de similitud es mayor que un umbral δ la imagen se etiqueta con la etiqueta de la consulta.
- Las instancias que tienen una puntuación de similitud menor que un umbral γ se eliminan.
- Entre las muestras restantes, se eligen las instancias con la puntuación de similitud más baja primero para el etiquetado humano, ya que estos ejemplos tienen más posibilidades de aumentar la diversidad en el conjunto de datos.

Para testear el framework propuesto crearon un conjunto de datos llamado DivNet.
La proporción de etiquetado humano utilizado en comparación con el número total de imágenes en el conjunto de datos, es del 11,7%.
Para medir la precisión seleccionaron un conjunto de imágenes al azar e inspeccionaron en forma manual el etiquetado, concluyeron que la precisión promedio es del 97,2%, que es ligeramente inferior al 99,7%, reportado por ImageNet [12].

Evaluaron la capacidad de generalización de DivNet comparándolo con VOC2012 [31] e ImageNet [12], dos conjuntos de datos de imágenes etiquetados en forma manual. Para ello realizaron un cruzamiento de entrenamiento y prueba, entrenando en un conjunto de datos y evaluando en otro. Concluyeron que el entrenamiento con DivNet muestra la mejor generalización entre los conjuntos de datos, ya que la caída promedio en el rendimiento del conjunto de datos cruzados es mínima.

2.4. ActiveCrowds: A Human-in-the-Loop Machine Learning Framework

Toumanidis et al. presentaron en el año 2021 [7] un framework de clasificación de un conjunto de datos con la estrategia “Human in the Loop”, el cual se puede aplicar en diferentes dominios de aprendizaje automático como son visión artificial o procesamiento de texto y audio.
El proceso consta de los siguientes pasos:

1. Inicialización del modelo utilizando un modelo existente previamente entrenado (Transfer Learning).
2. Entrenar y validar el modelo, calculando el accuracy en un conjunto de datos de prueba.
3. Obtener nuevas muestras utilizando una de las estrategias de muestreo disponibles.
4. Solicitar el etiquetado a los humanos.
5. Agregar las muestras etiquetadas al conjunto de entrenamiento y volver al paso 2.

El framework consta de tres grandes módulos:

1. Módulo principal, es el encargado del entrenamiento, validación y prueba de los modelos disponibles y del muestreo de nuevos datos para el etiquetado humano.
2. Servicio web, actúa como interfaz con el módulo principal.
3. Aplicación móvil, utilizada por los usuarios para completar las tareas de etiquetado.

Para el etiquetado manual usa la técnica llamada crowdsourcing [35], la cual es una técnica común tanto para la adquisición como para el etiquetado de datos, en la que se solicita a un grupo grande de personas, no necesariamente expertos en el dominio, que proporcionen o etiqueten las muestras existentes.

Como forma de testeo, se utilizó el framework para clasificar imágenes de derrumbe de edificios (clasificación binaria). Para ello contaron con un conjunto de 5270 imágenes, 1850 de ellas corresponden a edificios colapsados y 3420 a no colapsados.
Como conjunto de entrenamiento inicial tomaron 30 imágenes las cuales fueron clasificadas en forma manual para después comenzar el ciclo de aprendizaje activo.

Las estrategias de muestreo utilizadas en el ciclo de aprendizaje activo fueron las siguientes:

- Muestreo de margen de confianza mínimo.
- Muestreo de entropía.
- Muestreo aleatorio.

Con las tres estrategias de muestreo lograron un accuracy de 93.5% con un conjunto de datos de entrenamiento final de menos de 420 imágenes.
Además compararon sus resultados con los resultados reportados en el trabajo de Yeum et al. [36] y pudieron comprobar que lo superaron en casi un 1% de accuracy, 92.47% contra 91.5% reportado en [36], con un conjunto de 150 imágenes etiquetadas.

Por último probaron el modelo con 500 muestras del conjunto de prueba y obtuvieron un resultado de 315 predicciones verdaderas negativas y 154 verdaderas positivas.

2.5. Automatic Image Dataset Construction from Click-through Logs Using Deep Neural Network

Este trabajo fue presentado por Bai et al. en el año 2015 [8] donde proponen un método basado en el aprendizaje profundo para construir un conjunto de datos de imágenes a gran escala de forma automática.
La idea básica es automatizar los siguientes pasos:

1. Formación de consultas.
2. Eliminación de imágenes ruidosas.

Para lograr esto definen dos métricas de similitud.

1. Similitud palabra-palabra lo cual es usado para expandir cada categoría a un conjunto de palabras similares.
2. Similitud imagen-palabra para eliminar imágenes que no son relevantes para un conjunto de palabras similares.

Las palabras son representadas mediante una técnica similar a la técnica de incrustación de palabras muy común en tareas de PLN, pero en lugar de aprender de un corpus de texto, se aprende directamente usando registros de clics junto con la representación de imágenes.
Las imágenes son representadas utilizando una red con cuatro capas convolucionales y una capa completamente conectada.
La asociación entre la consulta y la imagen se aprende en una red neuronal profunda mediante la supervisión de registros de clics. En los registros de clics, se establecen un gran número de asociaciones entre consultas e imágenes a través de interacciones masivas de usuarios con el motor de búsqueda de imágenes. Todas las consultas en las que se hace clic en una imagen se combinan en un documento para formar la representación basada en palabras de la imagen, pudiendo establecer la asociación o similitud entre la imagen y la palabra.

La construcción del conjunto de datos consta de dos grandes pasos:

2.5.1. Formación de consultas:

Para la formación de consultas se define un umbral de similitud ε, aquellas palabras que cuya similitud con la clase sea mayor a ε se consideran similares.
Para una clase c, el conjunto de palabras similares S_c se define como:

$$\mathcal{S}_c = \{w_i | \text{sim}(c, w_i) > \xi_w, w_i \in \mathcal{V}\}$$

donde $sim(c, w_i)$es la similitud de coseno.

La recopilación de imágenes se realiza mediante dos pasos:

- Para una categoría *c* se expande a un conjunto de palabras similares S_c.
- Con el conjunto de palabras similares S_c se recopilan imágenes candidatas I_c para la categoría *c* agregando imágenes cuyas consultas asociadas a estas, tienen palabras que pertenecen al conjunto S_c.

2.5.2. Eliminación de imágenes ruidosas

Para la eliminación de imágenes ruidosas se define un umbral ε. Aquellas imágenes cuya similitud con la clase sea menor a ε se descartan.
Por lo que, dado el conjunto de imágenes candidatas I_c y el conjunto de palabras similares S_c para una clase *c* , el conjunto de imágenes D_c se define como:

$$\mathcal{D}_c = \{I | \text{sim}(I, \mathcal{S}_c) > \xi_i, I \in \mathcal{I}_c\}$$

donde la similitud de una imagen y el conjunto de palabras similares se mide como:

$$\text{sim}(I, \mathcal{S}_c) = \max_{w \in \mathcal{S}_c} \text{sim}(w, c) \cdot \text{sim}(w, I).$$

Este criterio selecciona imágenes que son similares a alguna palabra en S_c y la palabra es similar a la categoría *c*.

2.6. CurriculumNet: Weakly Supervised Learning from Large-Scale Web Images

Este trabajo, presentado por Guo et al. en el año 2018 [9] consiste en investigar la mejora de la capacidad del modelo de las redes neuronales estándar mediante la introducción de una nueva estrategia de entrenamiento.

La estructura de CurriculumNet se compone de dos grandes módulos:

1. Curriculum Design.
2. Curriculum Learning.

2.6.1. Curriculum Design

El objetivo es dividir todo el conjunto de entrenamiento en varios subconjuntos aplicando un algoritmo de agrupación en clusters basado en densidad, que se clasifican desde un subconjunto fácil que tiene imágenes limpias con etiquetas más confiables, hasta un subconjunto más complejo que contiene etiquetas ruidosas masivas.

Para ello se entrena un modelo inicial con todo el conjunto de entrenamiento usando una arquitectura Inception-v2 [37], se extraen las características de las imágenes y se calcula un valor de similitud utilizando la distancia euclidiana entre cada imagen. Con esa similitud se puede calcular la densidad local para cada imagen (esto es, cuántas imágenes están “cerca” de ella).

Luego se calcula un centro de grupo, que es la imagen que tiene mayor densidad, o sea que tiene la mayor cantidad de imágenes cerca en el mapa de características.

Las imágenes con similar apariencia se proyectan muy cerca unas de otras (alta densidad), por el contrario las imágenes distintas se proyectan más alejadas (densidad baja).

Utilizando el algoritmo k-means se generan tres grupos para cada categoría midiendo la densidad de la distribución dentro de cada grupo:

- Conjunto limpio: alta densidad, todas las imágenes tienen una gran similitud lo que significa que todas las etiquetas dentro de este grupo son correctas.
- Conjunto ruidoso: densidad media, puede incluir imágenes irrelevantes con etiquetas incorrectas.
- Conjunto muy ruidoso: densidad alta, similar al anterior pero con imágenes más dispersas en el mapa de características.

El Curriculum Learning final implementa el entrenamiento secuencialmente en los subconjuntos limpios, ruidosos y muy ruidosos.

2.6.2 Curriculum Learning

El entrenamiento se realiza con un modelo convolucional Inception-v2 que se entrena a través de tres etapas mezclando continuamente subconjuntos de entrenamiento desde un subconjunto limpio hasta uno muy ruidoso.

- En primer lugar se entrena un modelo utilizando solo los datos limpios, donde las imágenes dentro de cada categoría tienen una apariencia visual cercana. Esto permite que el modelo aprenda información visual básica pero clara de cada categoría, distinguiéndose características fundamentales para el siguiente proceso.
- En segundo lugar, cuando el modelo entrenado en la primera etapa converge, continuamos el proceso de aprendizaje agregando los datos de ruido, donde las imágenes tienen una diversidad visual más alta, lo que permite que el modelo aprenda características distintas y discriminatorias de muestras más difíciles.
- En tercer lugar, el modelo se entrena aún más agregando los datos de alto ruido que contienen una gran cantidad de imágenes visuales irrelevantes con etiquetas incorrectas.

  Se observa que los datos ruidosos agregados mejoran la capacidad de generalización del modelo y permite que el modelo evite el ajuste excesivo sobre los datos limpios, proporcionando una forma de regularización.

Capítulo 3

Marco teórico

Existen diferentes tipos de sistemas de aprendizaje automático [10], dentro de los cuales se encuentra el aprendizaje supervisado. En el aprendizaje supervisado, los datos de entrenamiento que alimentan al algoritmo incluyen las soluciones deseadas (etiquetas). El etiquetado corresponde a la tarea de asociar la solución deseada a cada dato de entrenamiento, donde el tipo del valor de la etiqueta depende de la tarea que queremos realizar. Por ejemplo, para clasificar si una imagen corresponde a un hombre o una mujer, o si un email es un spam o no, alcanzará con etiquetas binarias, mientras que si la tarea, por ejemplo, es predecir el valor de una casa dado un conjunto de características como la ubicación, cantidad de habitaciones, etc. se utilizará un valor numérico (este tipo de tarea se llama regresión).
Construir un conjunto de datos con etiquetado enteramente manual es una tarea que insume muchos recursos, pero contar con la participación humana en el etiquetado para la construcción de un conjunto de datos nos da cierta confianza en la calidad de los datos. Podemos seleccionar un conjunto de datos al azar y presentarlos para el etiquetado manual para así obtener un porcentaje de datos que sabemos (o esperamos que así sea) que están bien etiquetados. El problema es que si tenemos un conjunto de datos para etiquetar donde la mayoría, o una porción importante de ellos, tienen características similares, existe una alta probabilidad de seleccionar datos que no aportan nueva información a nuestro modelo. De esta forma estamos desperdiciando un recurso muy importante que es el etiquetado humano en el ciclo de aprendizaje.
Para comenzar, se parte de un modelo pre-entrenado en un problema de clasificación similar, que se va ajustando poco a poco para clasificar nuestros datos a medida que recibimos nuevas etiquetas.
El punto clave es seleccionar los datos de forma inteligente de manera de poder ayudar al modelo, lo cual no deja de ser un algoritmo de búsqueda que tiene que tener en cuenta dos factores:

- Explotación: guía la búsqueda teniendo en cuenta las mejores soluciones encontradas hasta el momento.
- Exploración: guía la búsqueda hacia regiones sin explorar y evita la convergencia prematura.

En este capítulo presentaré dos temas que tratan de resolver este problema:

- Muestreo por incertidumbre (Explotación)
- Muestreo por diversidad (Exploración)

3.1. Muestreo por incertidumbre

Como su nombre lo indica, el muestro por incertidumbre es una estrategia que consiste en seleccionar un conjunto de elementos sin etiquetar tomados de una región del espacio de características donde el modelo no está seguro de cómo clasificarlos. Como muestra la Figura 3.1, son los elementos que están más cerca de la frontera de decisión.

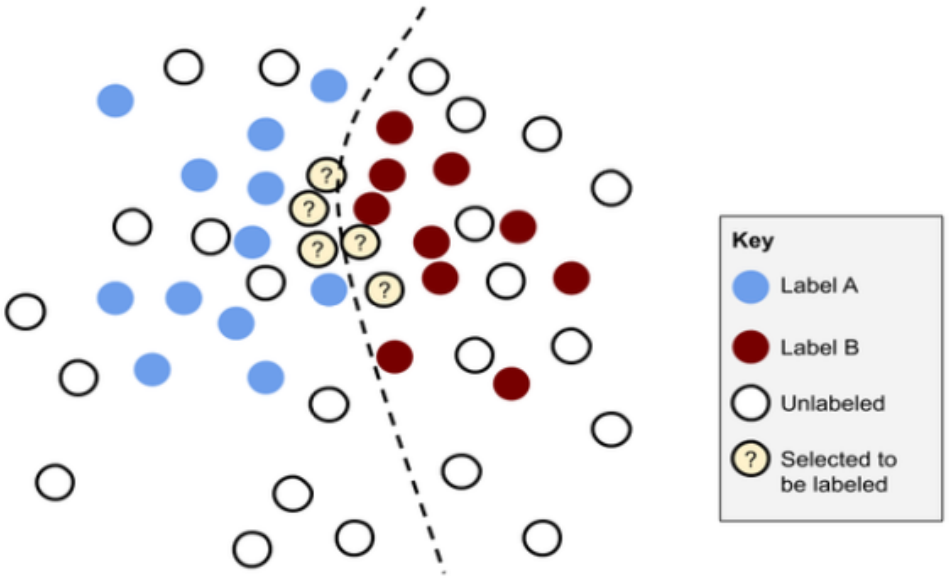


Figura 3.1.(Imagen tomada de [4])
Muestreo por incertidumbre: Los elementos seleccionados para etiquetar son los que están más cerca del límite de decisión (menor confianza).

El correcto etiquetado de estos datos ayudará al modelo a predecir de forma acertada la etiqueta de los datos con similares características en la próxima iteración.

3.1.2. Algoritmos para el muestreo por incertidumbre

Estos algoritmos buscan encontrar elementos que se encuentren cerca del límite de decisión, como muestra la Figura 3.1. Son algoritmos simples y constan de cuatro pasos [4]:

- Aplicar el algoritmo de muestreo por incertidumbre a un conjunto grande de datos no etiquetados, generando una única puntuación de incertidumbre por elemento.
- Clasificar las predicciones según la puntuación de incertidumbre.
- Seleccionar los N elementos más inciertos para la revisión por humanos.
- Obtener etiquetas generadas por humanos para los N elementos principales, reentrenar/ajustar el modelo con esos elementos y volver al primer paso.

Existen varias estrategias para realizar el muestro por incertidumbre, dentro de las cuales se encuentran:

- Muestreo de mínima confianza

La confianza se mide como el valor de probabilidad de la etiqueta y * predicha con mayor seguridad para cada elemento x.
Se escribe como:

$$P_{\theta}(y^* | x) \quad (3.1)$$

Los elementos muestreados son los que tienen el menor valor de probabilidad.

- Muestreo de margen de confianza

Consta de tomar la diferencia entre las probabilidades de las dos etiquetas y^*_1 y y^*_2 predichas con mayor seguridad para cada elemento x.
La diferencia de la predicción de ambas etiquetas se escribe como:

$$P_{\theta}(y^*_1|x) - P_{\theta}(y^*_2|x) \quad (3.3)$$

Los elementos muestreados son aquellos cuya diferencia entre las probabilidades es cercana a cero, o sea que tienen valores de probabilidad similares.

- Muestreo de porcentaje de confianza

Consta de tomar la relación entre las probabilidades de las dos etiquetas y^*_1 y y^*_2 predichas con mayor seguridad para cada elemento x. Esto se escribe como:

$$P_{\theta}(y^*_1|x) / P_{\theta}(y^*_2|x) \quad (3.5)$$

Los elementos muestreados son aquellos cuya relación entre las probabilidades es cercana a uno, o sea que tienen valores de probabilidad similares.

- Muestreo basado en entropía

Está relacionado a la diferencia entre todas las predicciones, es decir, cuánto difiere cada confianza de las demás.
La entropía aplicada a una distribución de probabilidad consta de multiplicar cada probabilidad por su logaritmo y tomar la suma negativa de ellos. Esto se escribe como:

$$-\sum_{y} P_{\theta}(y|x) \, log_2 P_{\theta}(y|x) \quad (3.6)$$

Los elementos muestreados son los que tienen entropía más alta, esto sucede cuando las probabilidades de cada etiqueta son similares.

3.2. Muestreo por diversidad

Muchas veces el sesgo en los datos está dado por la sobrerrepresentación de los mismos. Si imaginamos que se están clasificando imágenes de aves y existen muchas imágenes de una misma especie, y pocas de otra porque las fotografías fueron tomadas en una región donde estas últimas no son muy frecuentes, el modelo aprenderá mucho sobre las aves que están más representadas y poco sobre las otras.

El muestreo por diversidad trata el problema de datos que el algoritmo no conoce, datos escasos o que ocurren con menor frecuencia pero que en muchos problemas son de vital importancia para equilibrar los datos donde el algoritmo se entrena y adquiere conocimiento.

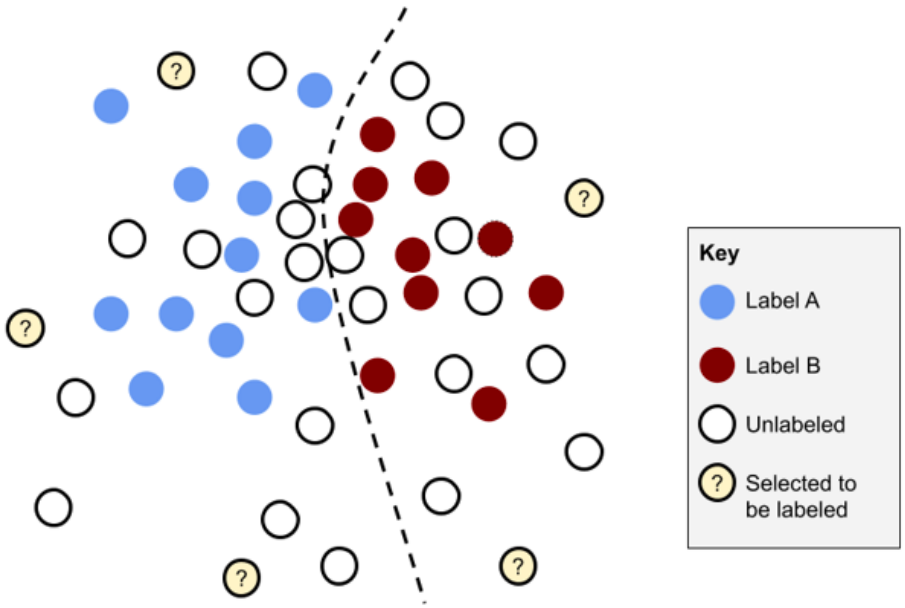


Figura 3.2. (Imagen tomada de [4])
Muestreo por diversidad: Muestreo diverso de datos sin etiquetar.

En la Figura 3.2. se puede apreciar una selección diversa de datos sin etiquetar (marcados con el símbolo de pregunta), lo cual expande el espacio de búsqueda. Esto hace que el límite de decisión se mueva hacia las nuevas características tratando de cubrir todo el dominio del problema.

El muestreo por diversidad puede verse de cuatro formas:

- Muestreo de valores atípicos basado en modelos

Consta en determinar qué elementos son desconocidos para el modelo. Representan los datos que el modelo no ha visto hasta el momento.

- Muestreo basado en clusters

Consta en realizar una selección diversa de los elementos sin etiquetar, de forma de no perder elementos que pueden resultar atípicos pero que tienen características significativas para el dominio del problema.

- Muestreo representativo

Consta en realizar una selección de elementos sin etiquetar que se parezcan más al dominio de destino (región del espacio de características donde queremos ajustar nuestros modelos), en comparación con los datos de entrenamiento. Esta estrategia es útil cuando el conjunto de entrenamiento y los datos que queremos etiquetar pertenecen a distintas distribuciones. Por ejemplo, si las personas usan principalmente su asistente de voz para solicitar canciones, el muestreo representativo se centraría en ejemplos de solicitudes de canciones sobre otros tipos de comandos.

- Muestreo para el mundo real

Consta en realizar una selección de elementos que cubran todo el dominio del problema de forma de reducir el sesgo. Volviendo al ejemplo de las aves, el muestreo para el mundo real debe garantizar la selección de imágenes de todas las especies existentes.

3.3. Combinando estrategias (Incertidumbre + Diversidad)

Todo algoritmo de búsqueda y optimización debe manejar el balance entre explotación (Incertidumbre) y exploración (Diversidad)
En este capítulo presentaré algunas posibles combinaciones que resuelven este problema. Se debe tener en cuenta que estas no son las únicas combinaciones posibles, pero todas tienen en común que primero se aplica un método y sobre el resultado del mismo se aplica el otro método.

- Muestreo de incertidumbre + Muestreo basado en clusters

El primer paso es hacer un muestreo de incertidumbre sobre los datos sin etiquetar. El resultado se agrupa en clusters y se muestrean algunos elementos de cada grupo de forma de lograr diversidad en los datos seleccionados, siendo que los elementos que están cerca unos de otros tienen características similares, no aportando nueva información al modelo.

Generalmente los elementos seleccionados en cada cluster son: centroides, outliers (son los elementos más alejados del centroide) y algunos elementos random. La cantidad de elementos outliers y random depende de la necesidad de cada problema.

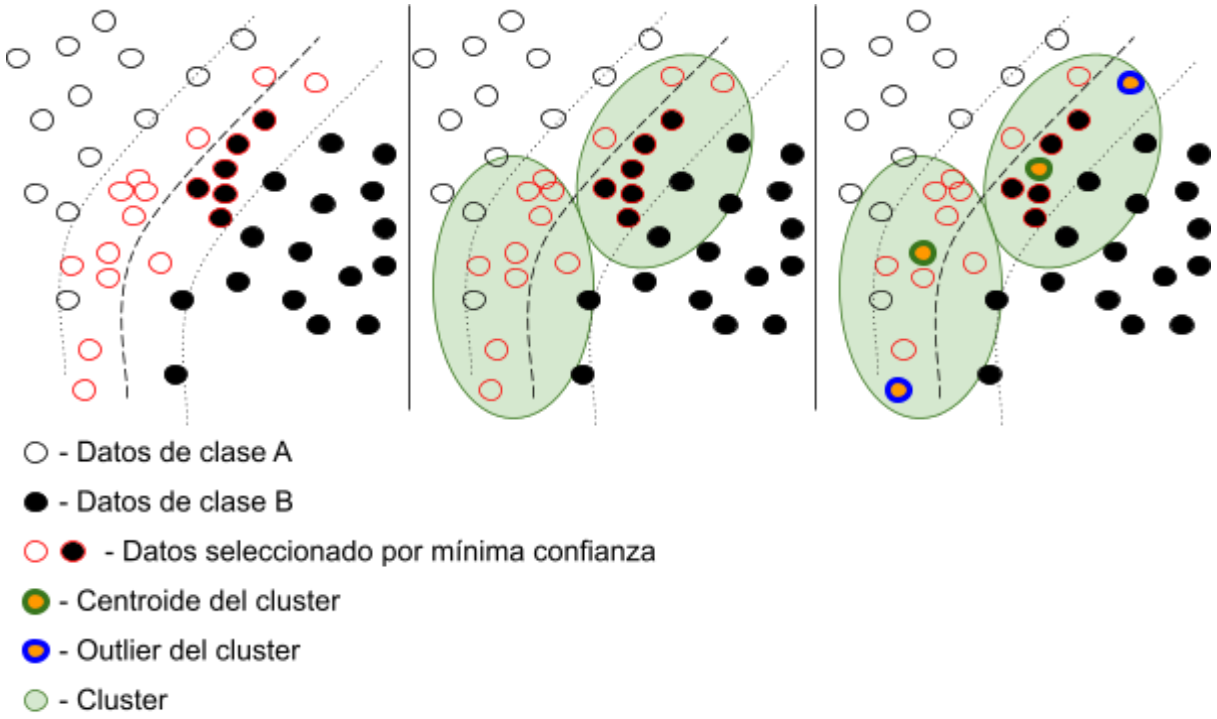


Figura 3.3: Muestreo por incertidumbre + clusters
La imagen de la izquierda muestra los elementos seleccionados por muestreo de menor confianza, la imagen del centro muestra la agrupación en clusters y la imagen de la derecha muestra los centroides y los outliers de cada cluster.

La Figura 3.3. ilustra este mecanismo, donde el conjunto de datos seleccionados para etiquetar son los que están más cerca de la frontera de decisión pero a su vez son diversos entre sí ya que pertenecen a distintos clusters.

- Muestreo de incertidumbre + Muestreo de valores atípicos basado en modelos

El primer paso es hacer un muestreo de incertidumbre y, sobre ese resultado, buscar los elementos que menos se parecen al conjunto de entrenamiento actual. Básicamente el algoritmo consiste en buscar elementos cerca del límite de decisión, pero que además, sean desconocidos para el modelo en su estado actual. Esta técnica se puede ver en la Figura 3.4.

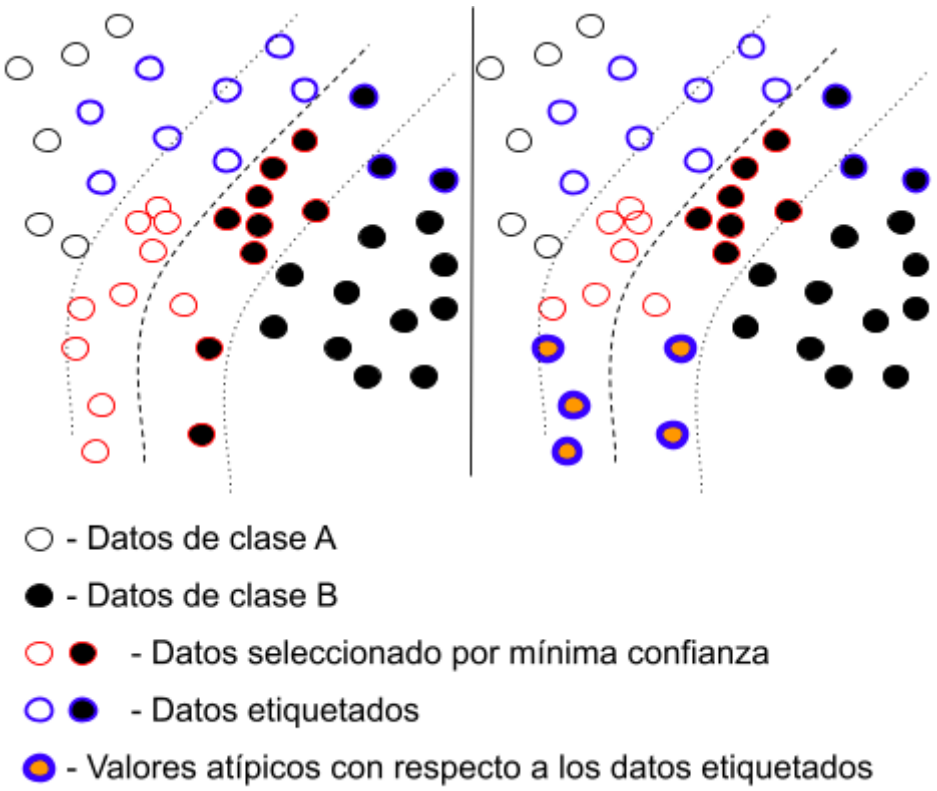


Figura 3.4 : Muestreo de incertidumbre + valores atípicos basado en modelos
La imagen de la izquierda muestra los elementos seleccionados con muestreo de menor confianza. La imagen de la derecha muestra los elementos que están más cerca de la frontera de decisión pero que son distintos del conjunto de entrenamiento actual.

- Muestreo de incertidumbre + Muestreo representativo + Muestreo basado en clusters

El primer paso es hacer un muestreo de incertidumbre de los datos sin etiquetar y los elementos resultantes se agrupan en clusters. De forma independiente se agrupan en clusters los datos del conjunto de entrenamiento. Luego se buscan los elementos de los grupos sin etiquetar que son más representativos en comparación con los grupos del conjunto de entrenamiento. En otras palabras, se muestrean elementos que se alejan del conjunto de entrenamiento actual, pero que son representativos del conjunto de datos sin etiquetar. Esto se puede ver en la Figura 3.5.

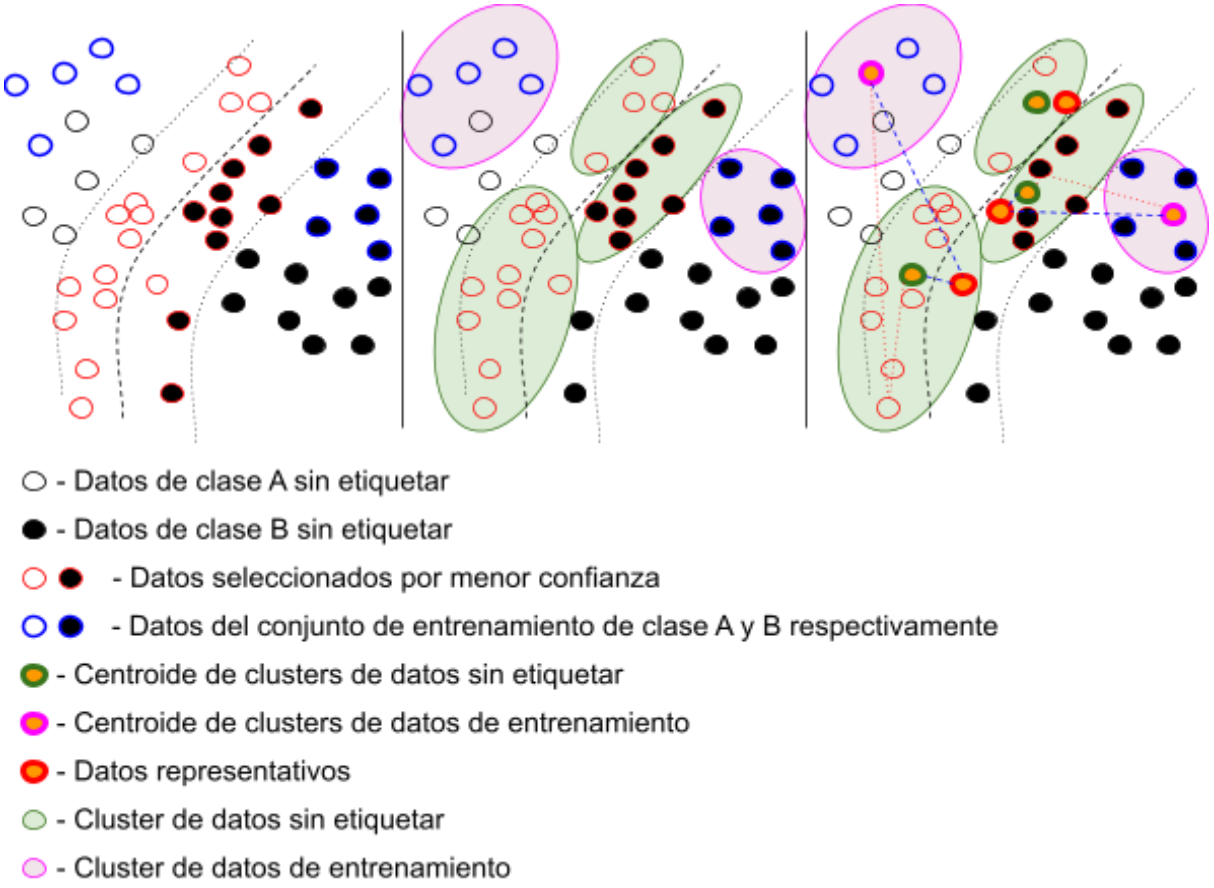


Figura 3.5: Muestreo por incertidumbre + Muestreo representativo + Muestreo basado en cluster
La imagen de la izquierda muestra los elementos seleccionados por muestreo de menor confianza. La imagen del centro muestra la agrupación en clusters de los datos no etiquetados seleccionados por menor confianza y la agrupación de los datos de entrenamiento. La imagen de la derecha muestra los centroides y los datos representativos de cada cluster.

Capítulo 4

Solución propuesta

4.1. Introducción

El objetivo de este proyecto es diseñar un algoritmo que permita generar conjuntos de datos etiquetados para tareas de visión artificial minimizando el costo de producción, donde el costo puede medirse en esfuerzo humano, tiempo, dinero, etc. Para ello se propone un framework al que llamé TrainingSet (TS), que utiliza redes neuronales pero que agrega el beneficio del ojo humano para retroalimentar el modelo y así guiarlo hacia el óptimo de la clasificación con la menor cantidad de datos etiquetados posible. Esta estrategia se denomina "Human in the loop" y el proceso se compone de tres grandes pasos [4] ilustrados en la Figura 4.1:

1. Muestrear los datos para etiquetar.
2. Usar esos datos para entrenar un modelo.
3. Usar ese modelo para muestrear más datos para etiquetar.

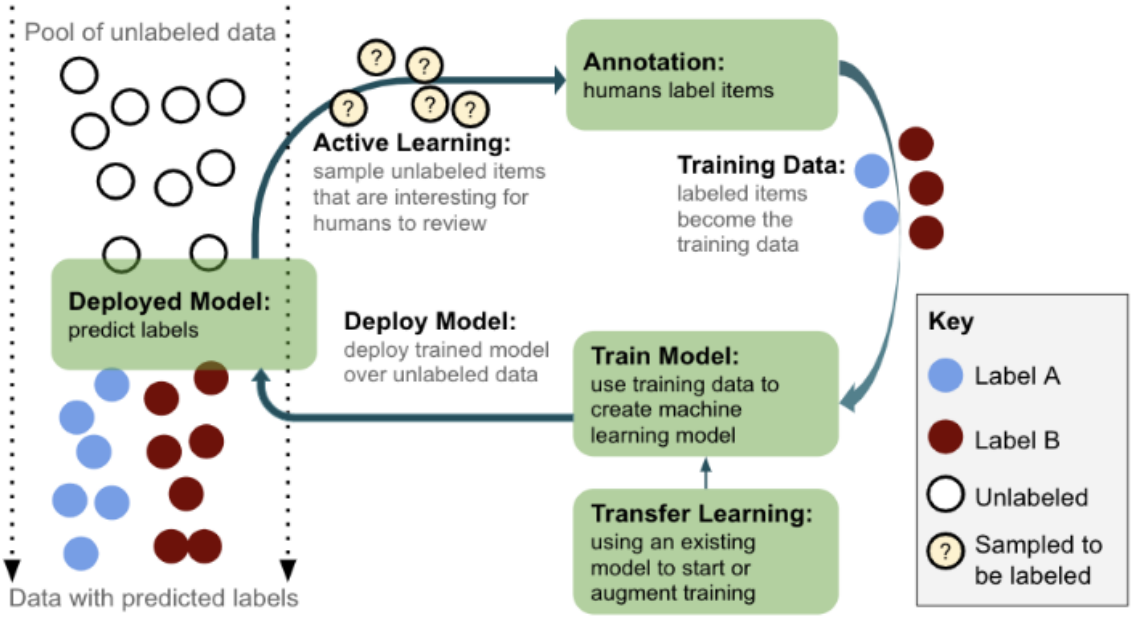


Figura 4.1. (Imagen tomada de [4])
Proceso de etiquetado de datos con humanos en el ciclo.

Optimizar el muestreo de datos es la parte fundamental del proceso, ya que la correcta selección de los mismos para que el humano etiquete hace que el sistema alcance un alto rendimiento en menor tiempo y por lo tanto con menor costo.
En base a esto propongo dos algoritmos para la selección inteligente de las imágenes que se etiquetarán manualmente. Los algoritmos implementan combinaciones de las estrategias

descritas en la sección anterior, específicamente “Muestreo de incertidumbre + muestreo basado en clusters“ y “Muestreo de incertidumbre + muestreo representativo basado en clusters”.

Los datos que estos algoritmos muestrean, una vez etiquetados, pasan a formar parte del conjunto de entrenamiento de un modelo de clasificación. Al entrenar el clasificador con nuevas imágenes, éste adquiere nuevos conocimientos sobre el dominio del problema, que luego son aplicados para la selección de nuevas imágenes para el etiquetado humano, que a su vez son el alimento del modelo de clasificación para su entrenamiento, creando así un sistema “Human in the loop”.

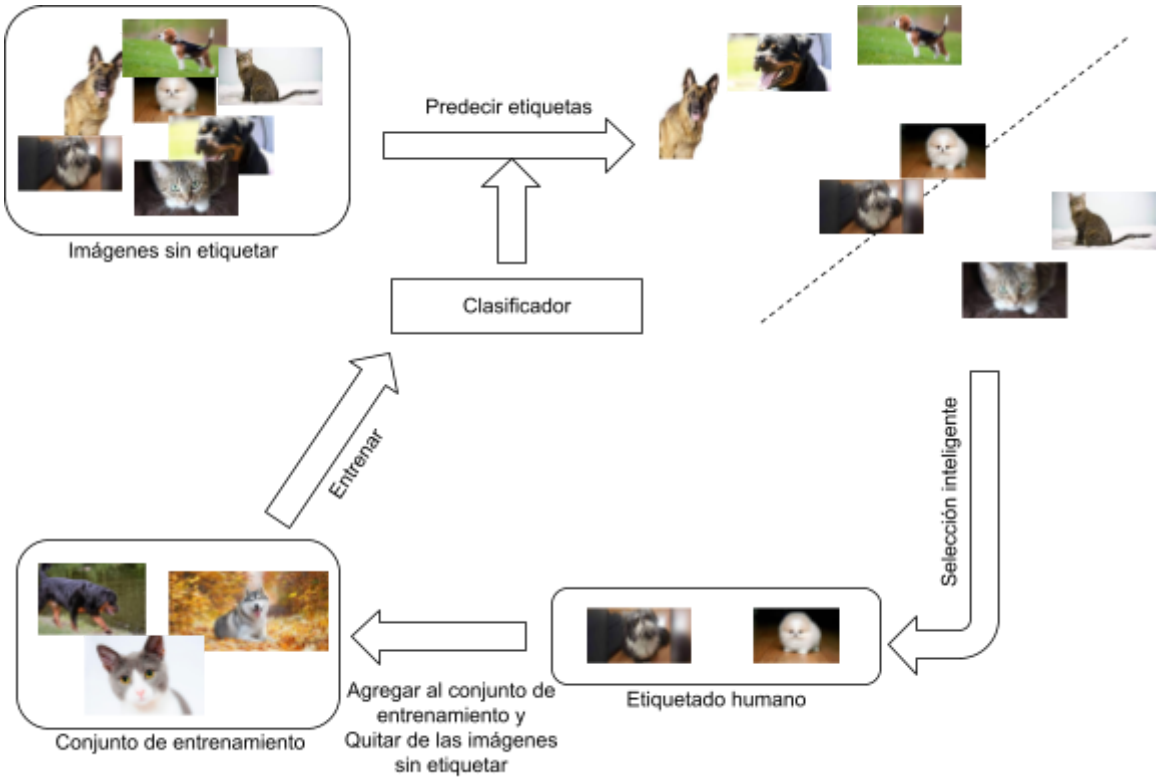


Figura 4.2. Sistema “Human in the loop” con selección inteligente de imágenes para el etiquetado manual. La figura inferior izquierda muestra el conjunto de entrenamiento actual con los que se entrena el clasificador, la figura superior izquierda muestra los datos sin etiquetar que son pasados por el clasificador (previamente entrenado) para predecir etiquetas, la figura superior derecha muestra el porcentaje de confianza en las etiquetas predichas, y la figura inferior derecha muestra la selección inteligente de imágenes para ser etiquetadas por el humano, que se quitan del conjunto de imágenes sin etiquetar y pasan a formar parte del conjunto de entrenamiento actual usados para entrenar el clasificador y posteriormente predecir etiquetas.

Entrenar un modelo desde cero con unas pocas imágenes no da buenos resultados, por lo que es necesario inicializar el modelo de alguna forma que ayude a mejorar la clasificación. Por ese motivo hay un paso extra que se puede ver en la Figura 4.1, llamado Aprendizaje por Transferencia (“Transfer Learning” en la figura). El Transfer Learning o transferencia de aprendizaje [4],[25] es una técnica muy común para inicializar un nuevo modelo, que trata de usar lo aprendido por una red en la resolución de un problema y aplicarlo (transferirlo) en otro.

Una red pre-entrenada como ImageNet [4],[12] ha aprendido texturas y bordes mientras discrimina entre 1000 categorías de imágenes. Este conocimiento adquirido suele ser más general que las 1000 clases discriminadas y se puede usar como punto de partida (inicialización) de un nuevo modelo para clasificar nuevos tipos (clases) de imágenes.

En este proyecto he utilizado la transferencia de aprendizaje obteniendo las características de las imágenes de una red VGG16 pre-entrenada en ImageNet, siendo ésta una de las arquitecturas más utilizadas para este propósito. La arquitectura se puede ver en la Figura 4.3.

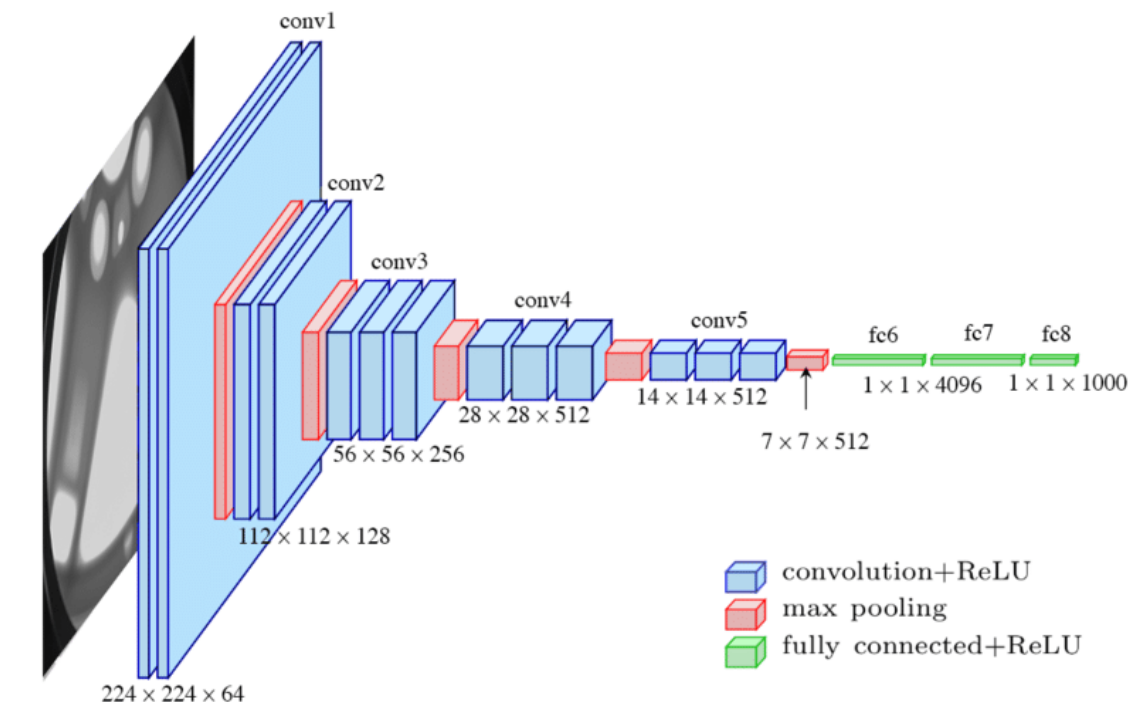


Figura 4.3. Arquitectura de red VGG16 [27]

Las capas convolucionales de esta red se congelan y se extraen las características de la capa posterior (en la Figura 4.3, es la capa señalada de dimensión 7x7x512) para inicializar un clasificador con la arquitectura que se describe en la Figura 4.4 Este clasificador es el responsable de clasificar las nuevas imágenes.

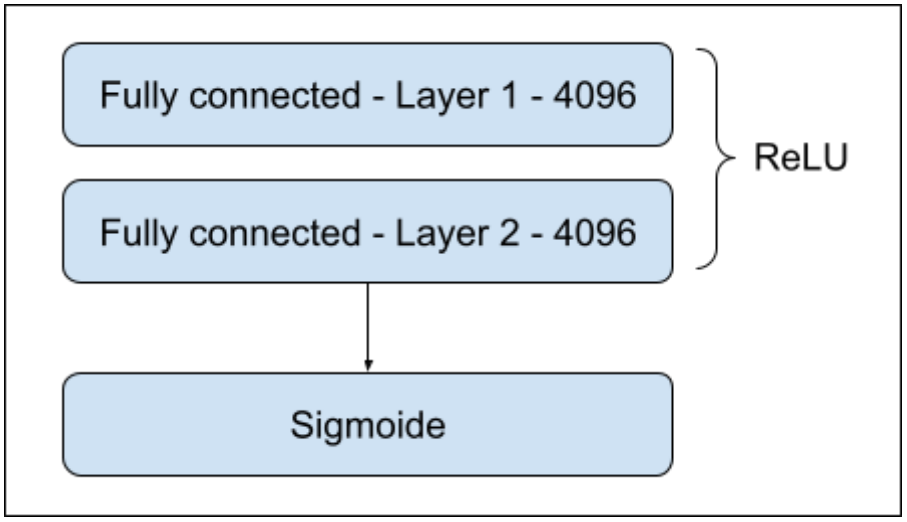


Figura 4.4. Arquitectura del clasificador
Arquitectura del clasificador, consta de dos capas ocultas fully connected, cada una con 4096 neuronas y función de activación ReLU y una capa de salida Sigmoide.

Dado que la clasificación que hago en este proyecto es binaria, la capa de salida del clasificador tiene una función de activación Sigmoide. La función Sigmoide transforma la entrada en valores entre 0 y 1.

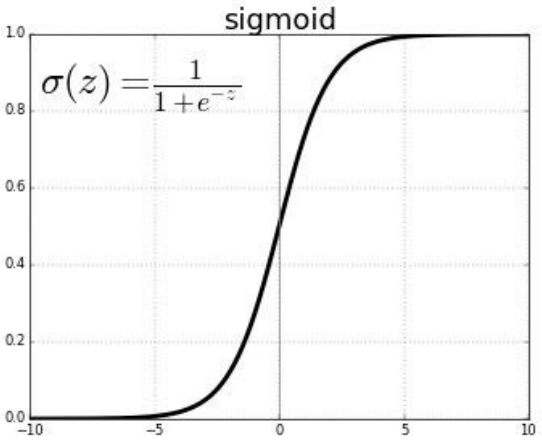


Figura 4.5. Función Sigmoide.
Función Sigmoide, transforma el valor de entrada *z* en un valor entre 0 y 1.

Notar que si bien este proyecto está orientado a resolver un problema de visión artificial, y el clasificador es alimentado con una representación de imágenes dado por el vector de características obtenido de la red VGG16, la independencia del clasificador permite que éste, y por consiguiente los algoritmos propuestos en este proyecto, puedan ser utilizados con otro tipo de representación de datos como puede ser una representación de palabras Word2Vec [26], por lo que, por ejemplo, es fácil la adaptación para resolver un problema de clasificación de textos.

4.2. Algoritmos para la selección de imágenes

Los algoritmos implementados para la selección de imágenes fueron dos:

- Muestreo de incertidumbre + Muestreo basado en clusters
- Muestreo de incertidumbre + Muestreo representativo basado en clusters

En ambos algoritmos se realiza en primer lugar un muestreo por incertidumbre y sobre este subconjunto se aplican dos estrategias distintas, ambas con el objetivo de tener diversidad en el muestreo. De este modo se evitan imágenes con similares características y que aportan poco al aprendizaje del nuevo modelo.

4.2.1 Algoritmo de Muestreo de incertidumbre + Muestreo basado en clusters (Low Confidence + Cluster)

El algoritmo de la Figura 4.6 ilustra el proceso completo de "Human in the loop" con muestreo de incertidumbre combinado con muestreo de clusters.

1. *model = Train(model, current_training_set) //** train model with current training set*
2. *random_samples = GetRandomSamples(mode, unlabeled_data_set) //** a percentage of the samples are random*
3. *low_confidence_samples = GetSamplesLowConfidence(mode, unlabeled_data_set) //** sample unlabeled data where the model has lower confidence*
4. *centroids, outliers, random_per_cluster = GetSamplesClustering(low_confidence_samples) //** groups the data into clusters and samples the centroids, outliers and some random items within each cluster*
5. *samples_to_label = random_samples + low_confidence_clustering_ samples //** prepare all data to label*
6. *samples_labeled = GetSamplesAnnotations(samples_to_label) //** get human tags*
7. *current_training_set += samples_labeled //** add labeled samples to current training and repeat the loop.*
8. *back to 1.*

Figura 4.6. Algoritmo de Muestreo de incertidumbre + Muestreo basado en clusters

Notar que en el primer paso del algoritmo de la Figura 4.6 se entrena el modelo en el conjunto de entrenamiento actual (ver Figura 4.1) lo que implica que debe existir un conjunto no vacío de entrenamiento inicial, el cual, dado la naturaleza de los algoritmos propuestos que utilizan el conjunto de entrenamiento actual para poder predecir el siguiente muestreo, debe ser seleccionado con algún criterio distinto, en este caso la selección se realiza de forma aleatoria para garantizar diversidad en el muestreo inicial.

La selección inteligente de datos para etiquetar se logra con la ejecución secuencial de las funciones:

- *"GetRandomSamples"* retorna un muestreo aleatorio sobre todos los datos sin etiquetar de forma de tener más diversidad (exploración) en los datos de entrenamiento. La cantidad de elementos muestreados en forma aleatoria debe ser un porcentaje menor ($\leq$10%) del total de elementos muestreados para el etiquetado manual de modo que el algoritmo no quede demasiado influenciado por la aleatoriedad.
- *"GetSamplesLowConfidence"* retorna un porcentaje de los datos sin etiquetar donde el modelo tiene más incertidumbre, que son los datos que están más cerca de la

frontera de decisión. Como se puede ver en la Figura 4.4 la clasificación implementada es binaria, lo que implica que los datos con mayor incertidumbre son los que su etiqueta tiene una probabilidad cercana a 0.5.

- *"GetSamplesClustering"* agrupa los datos con mayor incertidumbre en clusters usando el algoritmo K-means [13] con distancia de coseno [4], y de cada cluster se retorna el centroide, el elemento que está más lejos del centroide (outlier) usando la similitud de coseno [4] y una cantidad random de elementos. De esta forma garantizamos la diversidad en los datos muestreados.

Como último paso la función *"GetSampledAnnotation"* interactúa con el humano presentando la selección inteligente de imágenes y obteniendo una clasificación manual. Una vez clasificados estos datos se agrega al conjunto de entrenamiento actual.

Este proceso se ejecuta hasta que se cumpla algún criterio que nos satisfaga. Estos criterios pueden ser:

- La medida seleccionada para medir el rendimiento, por ejemplo la precisión en la clasificación, alcanza un determinado valor.
- La sucesión de ejecuciones hace que la medida seleccionada para medir el rendimiento, por ejemplo la precisión en la clasificación, no supere un valor τ previamente definido.
- Se alcanzó un porcentaje de imágenes clasificadas en forma manual.
- Se alcanzó el tiempo límite para el presupuesto dado para la clasificación manual.

4.2.2. Algoritmo de Muestreo de incertidumbre + Muestreo representativo basado en clusters (Low Confidence + Cluster Representative)

El algoritmo de la Figura 4.7 ilustra el proceso Human in the loop con muestreo por incertidumbre combinado con muestreo representativo basado en clusters.

1. *model = Train(model, current_training_set) //** train model with current training set*
2. *random_samples = GetRandomSamples(mode, unlabeled_data_set) //** a percentage of the samples are random*
3. *low_confidence_samples = GetSamplesLowConfidence(mode, unlabeled_data_set) //** sample unlabeled data where the model has lower confidence*
4. *representative_samples_clustered = GetRepresentativeClusteringSamples(low_confidence_samples, current_data_set) //** samples representative unlabeled with lower confidence data relative to current training set*
5. *samples_to_label = random_samples + representative_samples_clustered //** prepare all data to label*
6. *samples_labeled = GetSamplesAnnotations(samples_to_label) //** get human tags*
7. *current_training_set += samples_labeled //** add labeled samples to current training and repeat the loop.*
8. *back to 1.*

Figura 4.7. Algoritmo de Muestreo de incertidumbre + Muestreo Representativo basado en clusters

La estructura del algoritmo es igual al comentado en la sección 4.2.1 con la diferencia de la selección inteligente de imágenes: sobre el muestreo por incertidumbre ahora se aplica un muestreo representativo basado en clusters.

La función que lo implementa es:

- *"GetRepresentativeClusteredSampled"* agrupa en clusters el conjunto de datos sin etiquetar muestreados con menor confianza y de forma independiente hace los mismo con los datos de entrenamiento actual, en ambos casos usando el algoritmo K-means [13] con distancia de coseno [4]. Luego para cada dato sin etiquetar muestreados con menor confianza se calcula cuál es el mejor cluster para él entre los clusters de datos de entrenamiento y se calcula la similitud sc_i^{se} al centroide de su cluster de datos sin etiquetar y la similitud sc_i^{ent} a su mejor cluster de datos de entrenamiento usando la similitud de coseno [4] y se establece un valor de representatividad como la diferencia de las dos similitudes $r_i = sc_i^{se} - sc_i^{ent}$. Se retornan los datos de mayor valor r_i para cada cluster. En otras palabras, para cada grupo de datos sin etiquetar la función muestrea el elemento que está más cerca del centroide de ese grupo, en relación con los grupos de datos de entrenamiento.

4.3. Implementación

La implementación del framework fue realizada utilizando las siguientes tecnologías:

- El código fue escrito en Python 3 [21].
- Se utilizó la biblioteca para aprendizaje automático TensorFlow v2.6.0 [18] para la implementación y entrenamiento de los modelos, específicamente la biblioteca Keras. [19]
- Se utilizó el algoritmo de clusterización K-Means implementado en la biblioteca scikit-learn [15]. Esta biblioteca se utilizó también para la creación de un conjunto de datos de juguete.
- Jupyter-notebook como entorno de desarrollo [20].
- Google Colab para la ejecución de las pruebas primarias [17].

El código fuente de este proyecto está disponible en: https://gitlab.fing.edu.uy/juan.cabrera/proyecto-de-grado-2021/-/tree/master/codigo

Capítulo 5

Experimentos

En este capítulo se presentan los experimentos realizados. En la primera sección se presenta la experimentación realizada sobre un conjunto de datos de juguete, llevada a cabo para la visualización gráfica y posterior validación de la solución propuesta. En la segunda sección se presentan las bases de datos utilizadas para la experimentación con datos reales. En la tercera sección se presentan las métricas utilizadas para la evaluación, y por último, en la cuarta sección se presentan los experimentos realizados para medir el rendimiento de los algoritmos propuestos.

5.1. Prueba en un conjunto de datos de juguete

El primer experimento tiene por objetivo visualizar el comportamiento de la solución propuesta, presentada en las secciones 4.2.1 y 4.2.2 y probar la teoría presentada en el marco teórico. Visualizar la teoría de forma gráfica ayuda a ganar intuición sobre lo que se está realizando, pero es complejo cuando se trabaja con dimensiones mayores a tres. Por tal motivo, utilicé un conjunto de datos de juguete de la biblioteca scikit-learn para Python [15] con datos en dos dimensiones. El conjunto de datos contiene dos clases de instancias distribuidos en dos semicírculos entrelazados [16], con lo cual el límite de decisión se confunde, siendo apropiado para probar los algoritmos.
Los algoritmos se ejecutaron en forma independiente partiendo de un conjunto de datos con 4000 elementos. La Figura 5.1. muestra la distribución de los mismos en el plano.

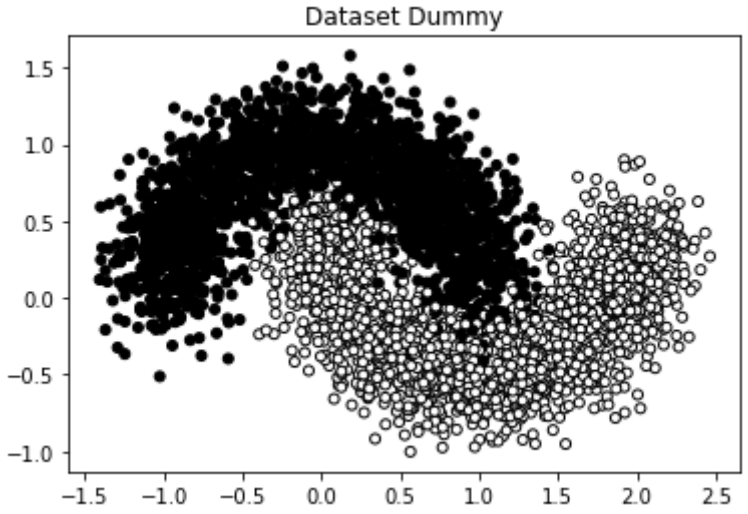


Figura 5.1. Conjunto de datos de juguete
Muestra la distribución de los elementos en un conjunto de datos de juguete con dos clases de elementos en un plano de la siguiente manera: Negro: Datos de clase A, Blanco: Datos de clase B

El conjunto de entrenamiento inicial contiene 20 elementos etiquetados elegidos al azar y el resto son datos sin etiquetar. El motivo de seleccionar 20 elementos de entrenamiento está relacionado con la necesidad del algoritmo Low Confidence + Representative Cluster (que se

verá en la sección 5.1.2) y su correcto funcionamiento, donde se agrupan los datos de entrenamiento en 20 clusters para el muestreo de datos sin etiquetar. Si bien para el algoritmo Low Confidence + Cluster se podría utilizar menos datos de entrenamiento, uso la misma cantidad de datos en ambos algoritmos para para no beneficiar a un algoritmo con respecto al otro.

A continuación se presentan las pruebas realizadas con ambos algoritmos en este conjunto de datos.

5.1.1. Prueba del algoritmo del algoritmo de Muestreo de incertidumbre + Muestreo basado en clusters (Low Confidence + Cluster)

Para la ejecución se estableció que la cantidad de datos seleccionados donde el modelo tiene menor confianza ("$numero\_items\_menor\_confianza$") sea 150, para así lograr una mayor claridad y comprensión de las gráficas.
Dado que el algoritmo muestrea para cada cluster el centroide, un outlier y algunos datos aleatorios de los datos donde el modelo tiene menor confianza, la cantidad de muestras en cada iteración está dada por la ecuación:

$$numero\_items\_muestreados = numero\_items\_random + lc\_numero\_cluster * (2 + numero\_items\_random\_por\_clusters) \quad (5.1)$$

definiéndose para este ejemplo:

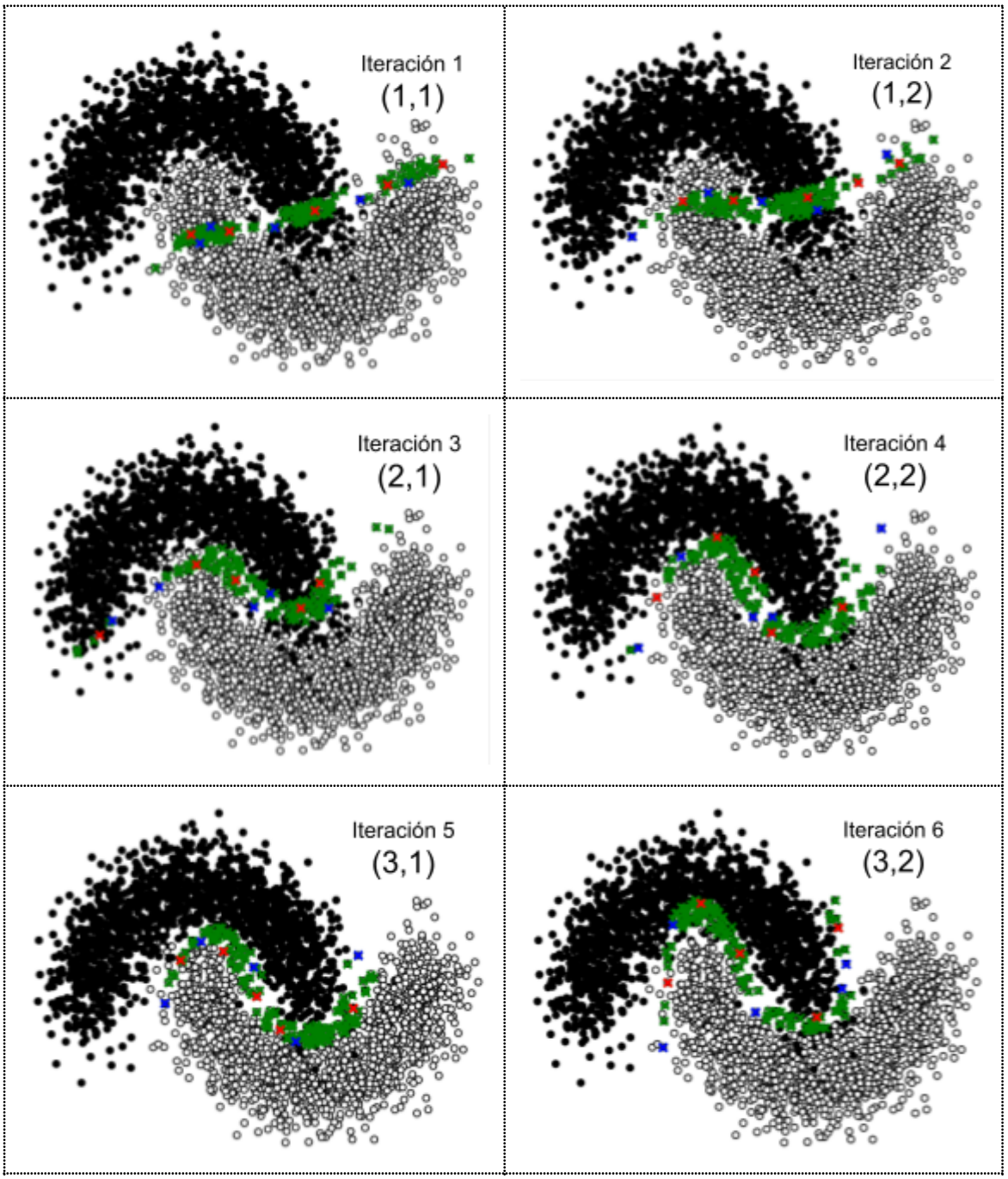
$$numero\_items\_random = 2$$
$$lc\_numero\_cluster = 5$$
$$número\_items\_random\_por\_cluster = 2$$

Estas constantes definen la cantidad de elementos que quiero agregar en cada iteración pudiendo variar según la conveniencia. En este caso se agregan 22 elementos al conjunto de datos de entrenamiento actual en cada iteración. El algoritmo se ejecuta una vez iterando 10 veces, totalizando 220 elementos agregados.

Para visualizar la evolución del algoritmo, en la Figura 5.2 se presenta un conjunto de 10 imágenes, donde cada una corresponde a un paso de iteración. En cada una de ellas se muestran los datos sin etiquetar, el muestreo en cada paso de iteración y el movimiento de la frontera de decisión.
Cada imagen de la Figura 5.2 muestra los elementos separados en colores de la siguiente manera:

- Negro: Datos de clase A.
- Blanco: Datos de clase B,
- Verde: Datos donde el modelo tiene menor confianza (Muestreo de incertidumbre).
- Rojo: Centroide de cada cluster (Muestreo basado en cluster).
- Azul: Outlier de cada cluster (Muestreo basado en cluster).

Notar que no se muestran los datos seleccionados en forma aleatoria en cada cluster y tampoco los datos seleccionados al azar en todo el conjunto sin etiquetar porque haría la imagen menos legible y no aporta a lo que realmente quiero mostrar que es la selección de los datos en los diferentes grupos.



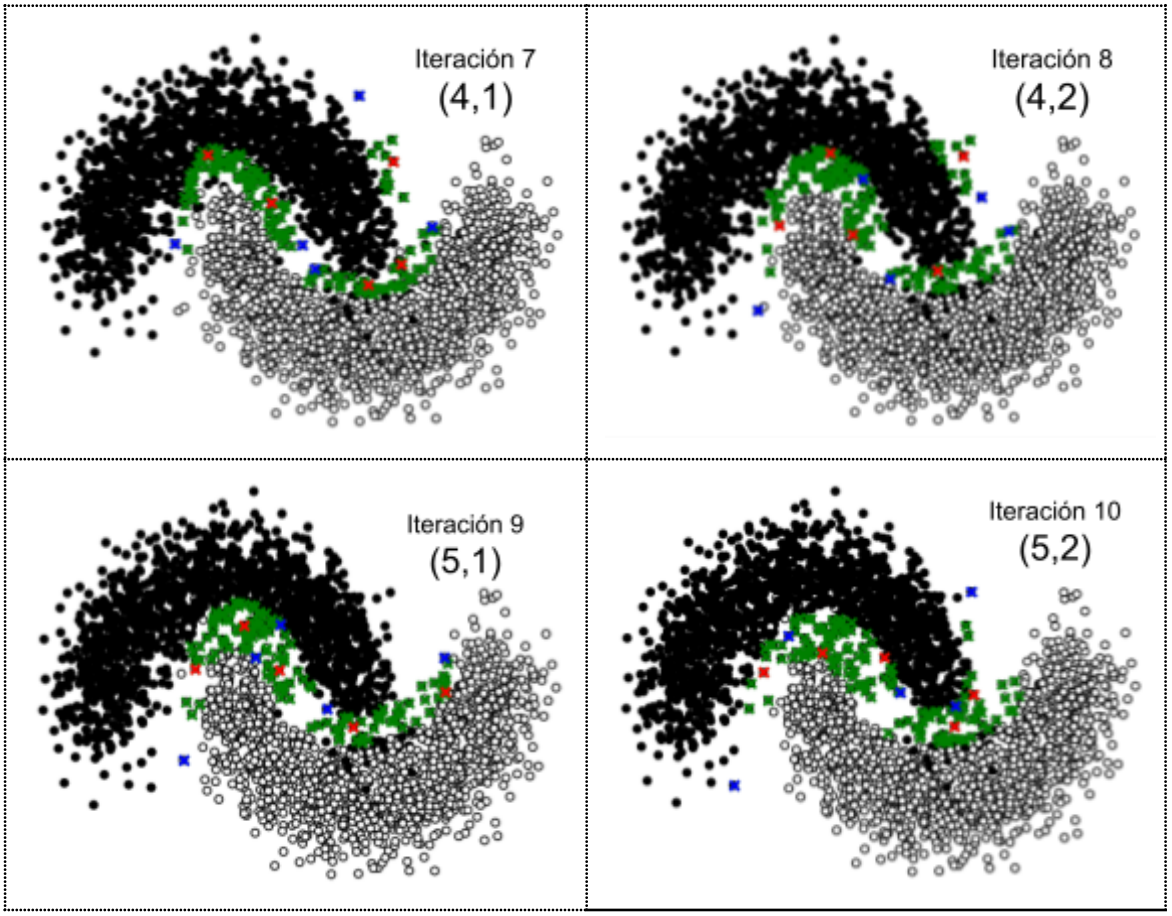


Figura 5.2. Muestra una secuencia de diez ejecuciones del algoritmo de Muestreo por Incertidumbre + Cluster, la imagen de cada celda muestra los datos de la siguiente manera:
Negro: Datos sin etiquetar de Clase A
Blaco: Datos sin etiquetar de Clase B
Verde: Datos sin etiquetar donde el algoritmo tiene mayor incertidumbre (Algoritmo de Muestreo por Incertidumbre)
Rojo: Centroide de cada cluster
Azul: Outlier de cada cluster

Observando la celda (1,1), correspondiente a la primera iteración, se puede apreciar que el modelo no divide correctamente las dos clases de datos, pero si observamos la celda (3,1), correspondiente a la iteración número cinco, el modelo ya tiene correctamente separadas las dos clases (notar que a medida que avanzan las iteraciones se forman huecos sobre la frontera de decisión). Esto indica que a medida que se van seleccionando datos donde el algoritmo tiene mayor incertidumbre, etiquetándolos correctamente y agregándolos al conjunto de entrenamiento, el modelo mejora la clasificación en la siguiente iteración, mostrando que el sistema funciona como se esperaba.

5.1.2. Prueba del algoritmo de Muestreo de incertidumbre + Muestreo representativo basado en clusters (Low Confidence + Representative Cluster)

Del mismo modo que en el algoritmo de la sección anterior, se estableció que la cantidad de datos seleccionados donde el modelo tiene menor confianza (“ $numero\_items\_menor\_confianza$”) sea 150 para la mayor claridad y comprensión de las gráficas.

Dado que este algoritmo separa los datos de menor confianza en clusters y muestrea el elemento más representativo con respecto al conjunto de entrenamiento actual de cada cluster, la cantidad de datos muestreados en cada iteración está dado por el número de clusters, por lo tanto, la cantidad de muestras en cada iteración está dada por la ecuación:

$$numero\_items\_muestreados = numero\_items\_random + rep\_numero\_cluster \quad (5.2)$$

definiéndose para este ejemplo:

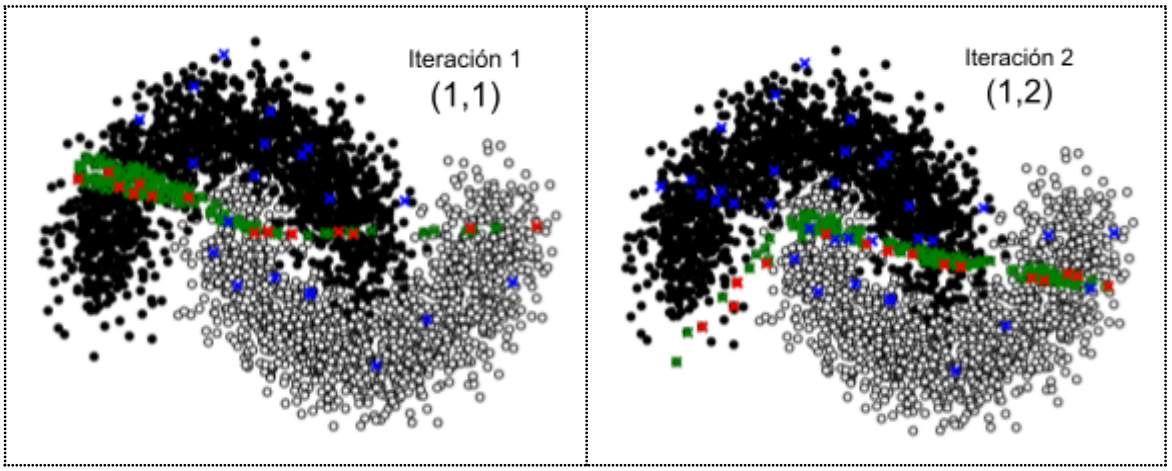
$$numero\_items\_random = 2$$
$$rep\_numero\_cluster = 20$$

Estas constantes definidas indican la cantidad de datos que quiero agregar en cada iteración, de modo que se agregan 22 datos al conjunto de datos de entrenamiento actual en cada iteración. El algoritmo se ejecuta una vez iterando 10 veces, totalizando 220 datos agregados.

La Figura 5.3 se compone de 10 imágenes y cada una representa un paso de iteración. En cada una de ellas se puede ver cómo el algoritmo divide las clases de datos sin etiquetar donde el modelo tiene mayor incertidumbre y los datos que muestrea en cada paso de iteración.
En cada imagen se puede ver los datos separados en colores de la siguiente manera:

- Negro: Datos de clase A.
- Blanco: Datos de clase B.
- Verde: Datos donde el modelo tiene menor confianza (Muestreo de incertidumbre).
- Rojo: Datos más representativos con respecto al conjunto de entrenamiento actual de cada cluster (Muestreo representativo basado en cluster).
- Azul: Datos del conjunto de entrenamiento actual.

Notar que no se muestran los datos seleccionados al azar en todo el conjunto sin etiquetar porque haría la imagen menos legible y no aporta lo que realmente quiero mostrar que es la selección de los datos en los diferentes grupos.



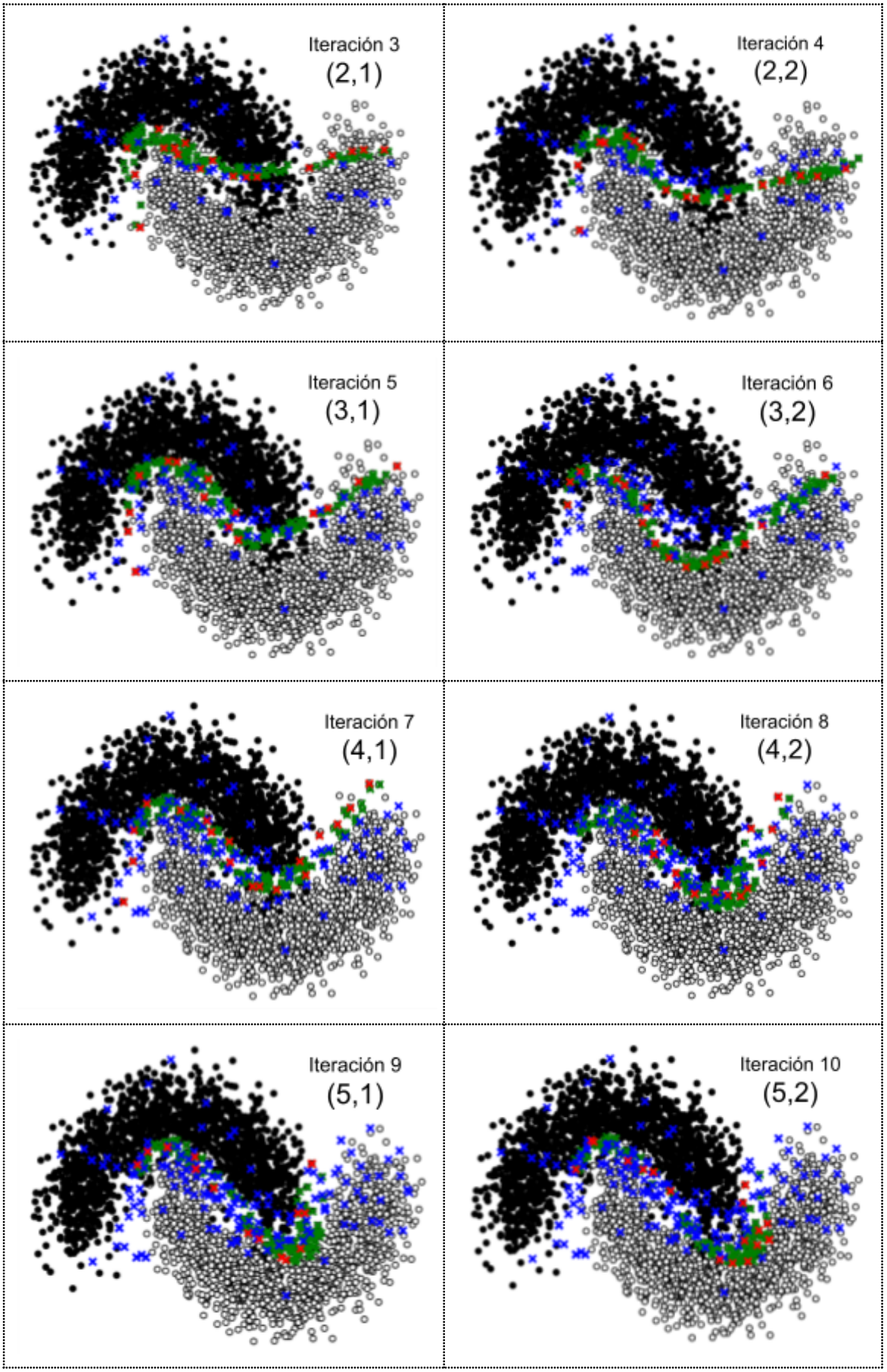


Figura 5.3. Muestra una secuencia de diez iteraciones del algoritmo de Muestreo por Incertidumbre + Muestreo representativo basado en cluster
La imagen de cada celda muestra los datos de la siguiente manera:

Negro: Datos sin etiquetar de Clase A
Blanco: Datos sin etiquetar de Clase B
Verde: Datos sin etiquetar donde el algoritmo tiene mayor incertidumbre (Algoritmo de Muestreo por Incertidumbre)
Rojo: Dato más representativo con respecto al conjunto de entrenamiento actual para cada cluster
Azul: Datos del conjunto de entrenamiento actual

Así como con el algoritmo de la sección 5.1.1, en las primeras iteraciones el modelo no separa correctamente las clases, en este caso no lo hace hasta la sexta iteración (celda 3,2), y del mismo modo, a medida que avanzan las ejecuciones se va poblando el conjunto de entrenamiento (azules) con datos representativos de los grupos de datos sin etiquetar que están más cerca de la frontera de decisión pero que se alejan del conjunto de entrenamiento actual (lo que en la sección 5.1.1 se muestran como huecos). De esta forma también podemos visualizar que el algoritmo funciona como se esperaba.

5.2. Base de datos

A continuación se presentan las dos bases de datos utilizadas para la experimentación, ambas descargadas de Kaggle [22].

Tumores cerebrales

Conjunto de datos de imágenes de resonancias magnéticas de tumores cerebrales [23], el cual consta de 4602 imágenes divididas en 2 categorías, donde 2513 imágenes corresponden a personas con tumores cerebrales y 2087 a personas saludables. La Figura 5.4. muestra un ejemplo de una persona enferma de cáncer (izquierda) y una persona sana (derecha).

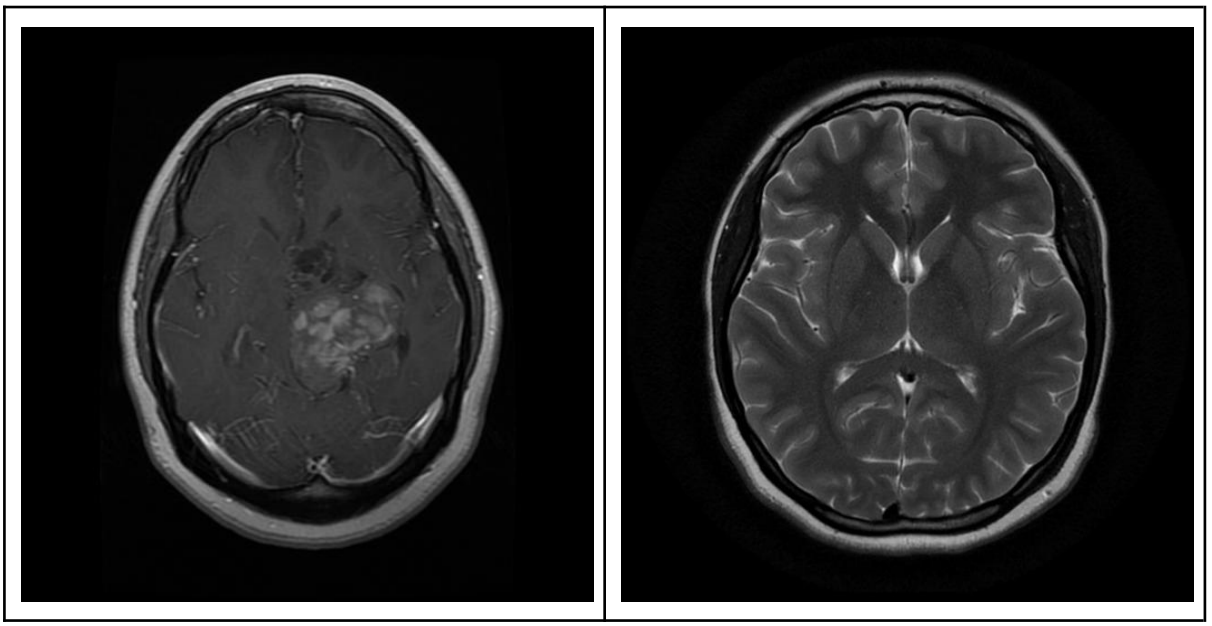


Figura 5.4. (Imágenes tomadas de [23])
Imagen de una resonancia magnética de una persona con tumor cerebral (izquierda)
Imagen de una resonancia magnética cerebral de una persona sana (derecha)

Neumonía

Conjunto de datos de imágenes de rayos X de tórax (neumonía) [24], el cual consta de 5856 imágenes divididas en 2 categorías, donde 1583 imágenes pertenecen a personas saludables y 4273 pertenecen a personas con neumonía. La Figura 5.5. muestra un ejemplo de una persona enferma de neumonía (izquierda) y una persona sana (derecha).

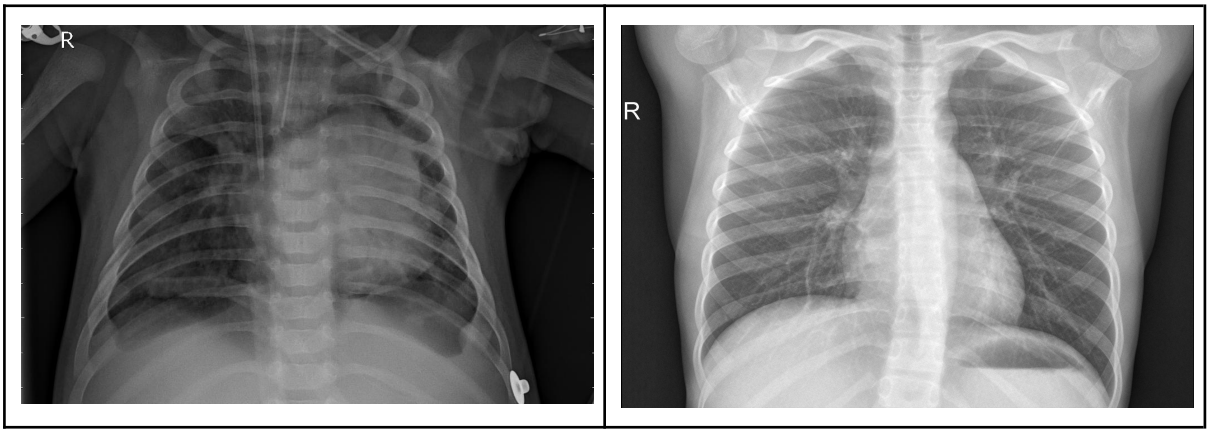


Figura 5.5. (Imágenes tomadas de [24])
Imagen de una radiografía de tórax una persona con neumonía (izquierda)
Imagen de una radiografía de tórax de una persona sana (derecha)

Antes de comenzar con el proceso de active learning es necesario inicializar el sistema, seleccionando algunos datos de entrenamiento y evaluación. Para eso se seleccionan datos al azar para cada conjunto (entrenamiento, evaluación).
Por lo general la cantidad de datos de entrenamiento y evaluación siguen la regla 80-20, esto es 80% de datos de entrenamiento y 20% para evaluación. En este caso, como ambos conjuntos de datos utilizados para los experimentos tienen aproximadamente 5000 imágenes, este valor se fija en 1000 imágenes para el conjunto de evaluación.
El conjunto de datos de entrenamiento se construye en forma incremental utilizando los algoritmos propuestos de forma independiente. Esto implica que se rompe la regla 80-20 y no se toma el 80% de datos para entrenamiento sino que se toma un pequeño subconjunto que se presenta al humano para el etiquetado manual y se usa como conjunto de entrenamiento inicial.
Notar que para estos experimentos estoy utilizando conjuntos de datos etiquetados, por lo que no es necesario realizar el etiquetado manual de estas imágenes sino que se simula dicho proceso consultando las etiquetas provistas por cada base de datos.

5.3. Métricas de rendimiento

Para estudiar los resultados obtenidos es necesario previamente definir las métricas que se van a utilizar. A continuación se definen las mismas.

Accuracy

Es una medida que resulta intuitiva dado que es una relación entre la observación predicha correctamente y el total de observaciones. Funciona bien en conjunto de datos balanceados, pero en caso de conjunto de datos desbalanceados, como lo es el conjunto de datos Neumonía [24], el accuracy puede dar resultados que no reflejan la realidad de la calidad del modelo. Ver Ecuación 5.3.

Medida F

La medida F es el promedio ponderado de precisión (Precision) y recuperación (Recall). Por tanto, esta medida tiene en cuenta tanto los falsos positivos como los falsos negativos. Ver Ecuación 5.4.

Tiempo de muestreo

Tiempo que le toma a un algoritmo muestrear un subconjunto de imágenes.

Error estándar de la media

Desviación estándar de las medias muestrales para todas las posibles muestras escogidas de una población de un determinado tamaño. Ver Ecuación 5.7.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (5.3)$$
$$F\,measure = 2 * (Precision * Recall) / (Precision + Recall) \quad (5.4)$$
$$Precision = TP / (TP + FP) \quad (5.5)$$
$$Recall = TP / (TP + FN) \quad (5.6)$$

donde:

TP= Verdadero positivo
TN= Verdadero negativo
FP= Falso positivo
FN= Falso negativo

$$\sigma_{\bar{x}} = \sigma / \sqrt{n} \quad (5.7)$$

donde:

σ : desviación estándar de la muestra.
n : tamaño de la muestra.

5.4. Prueba de los algoritmos en conjunto de datos reales

El segundo experimento busca probar que utilizando los algoritmos propuestos se necesita etiquetar menos datos para obtener el mismo resultado que seleccionando datos al azar (algoritmo Random).

Para realizar la comparación cada algoritmo se ejecuta 10 veces y en cada ejecución hay 7 iteraciones. Se parte con los mismos datos de entrenamiento y evaluación y se agrega la

misma cantidad de imágenes al conjunto de entrenamiento en cada paso de iteración, entrenando y evaluando el modelo para medir el rendimiento. Cabe destacar que las imágenes son seleccionadas con el algoritmo correspondiente. De este modo, en cada ejecución se crean en forma incremental, tres conjuntos de datos de entrenamiento distintos, uno por cada algoritmo y se evalúa el modelo con cada subconjunto agregado.
Como forma de simplificar la ejecución de los experimentos se sustituye el etiquetado de los datos por parte del humano por una consulta a una base de datos previamente etiquetada, es decir, se simula el ingreso de datos por el humano.

La ejecución se realizó en un servidor con las siguientes características:

Sistema Operativo: Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-81-generic x86_64).
CPU: Intel(R) Core(TM) i7-8700 CPU @ 3.20GH.
GPU: GeForce GTX 1080 Ti.
Memoria RAM: 62 GB.
Capacidad de disco: 216 GB.

5.4.1. Pruebas de ejecución de algoritmos

Para la ejecución se definieron las siguientes constantes:

$$numero\_items\_conjunto\_evaluacion = 1000$$
$$numero\_items\_conjunto\_entrenamiento\_inicial = 150$$
$$numero\_items\_menor\_confianza = 0.1 * cantidad\ datos\ sin\ etiquetar$$
$$numero\_items\_random = 10$$

Constantes necesarias para el algoritmo de Muestreo de incertidumbre + Muestreo basado en clusters (Low Confidence + Cluster)

$$lc\_numero\_cluster = 20$$
$$número\_items\_random\_por\_cluster = 5$$

Constantes para el algoritmo de Muestreo de incertidumbre + Muestreo representativo basado en clusters (Low Confidence + Representative Cluster)

$$rep\_numero\_cluster = 140$$

La cantidad de imágenes muestreadas con menor confianza es variable, siendo el 10% de la cantidad de imágenes sin etiquetar en cada iteración.
El número de imágenes muestreadas para el algoritmo Low Confidence + Cluster está dado por la Ecuación 5.1 la cual indica que se deben agregan 150 imágenes en cada iteración.
El número de imágenes muestreadas para el algoritmo Low Confidence + Representative Cluster está dado por la Ecuación 5.2 la cual indica que se deben agregar 150 imágenes en cada iteración.

Para que los algoritmos puedan ser comparados se debe asegurar que la cantidad de imágenes muestreadas en cada paso de iteración sea la misma.

Dado que ambos algoritmos usan agrupación en clúster y la cantidad de datos en cada uno puede variar, es posible que un cluster no tenga la cantidad suficiente de datos para muestrear en cuyo caso se realizan las siguiente acciones en forma ordenada hasta completar los datos faltantes:

Low Confidence + Cluster:

1. Seleccionar datos al azar dentro de cada cluster que no han sido seleccionados previamente.
2. Seleccionar datos al azar dentro de los datos muestreados con menor confianza que no han sido seleccionados previamente.
3. Seleccionar datos al azar dentro de los datos sin etiquetar que no han sido seleccionados previamente.

Low Confidence + Representative Cluster:

1. Seleccionar para cada cluster el segundo dato más representativo.
2. Seleccionar datos al azar dentro de los datos muestreados con menor confianza que no han sido seleccionados previamente.
3. Seleccionar datos al azar dentro de los datos sin etiquetar que no han sido seleccionados previamente.

Dado que se hacen 7 iteraciones por ejecución, se agregan 1050 imágenes en cada ejecución por cada algoritmo al conjunto de entrenamiento inicial, resultando en un conjunto de entrenamiento de 1200 imágenes.

5.4.1.1. Prueba en el conjunto de datos Tumores Cerebrales

La Figura 5.6 muestra el accuracy promedio con el error estándar de la media de 10 ejecuciones de los algoritmos. El paso 0 es el promedio del accuracy del conjunto de entrenamiento inicial (150 imágenes elegidas al azar). Los pasos siguientes corresponden al accuracy promedio con la selección de 150 imágenes de cada algoritmo.
Si bien el paso 1 muestra una igualdad (contando el margen de error) entre los tres algoritmos, lo cual se explica observando las Figuras 5.2 y 5.3 para los algoritmos Low Confidence + Cluster y Low Confidence + Representative Cluster respectivamente, en esta figura se puede observar que en las primeras iteraciones, donde los algoritmos tienen pocos datos de entrenamiento la frontera de decisión no está cerca de la frontera o límite entre clases. No fue el caso aquí, pero dependiendo del conjunto de entrenamiento inicial (sorteo), en etapas (iteraciones) tempranas puede pasar que el algoritmo solo muestree elementos de una misma clase, por lo qué, el algoritmo Random podría dar mejores resultados en las primeras iteraciones. El incluir un pequeño porcentaje de muestras sin etiquetar elegidas al azar en cada algoritmo disminuye la probabilidad de que esto suceda.

En los pasos sucesivos se puede apreciar cómo se despegan los algoritmos Low Confidence + Cluster y Low Confidence + Representative Cluster con respecto al algoritmo Random logrando una mejora apreciable, aunque no se puede establecer una diferencia entre los dos primeros. Esto se debe a que los conjuntos de datos de entrenamiento y sin etiquetar pertenecen a la misma distribución, por lo tanto el muestreo representativo no encuentra datos que se alejan demasiado del conjunto de entrenamiento actual.
El experimento demuestra que usando los algoritmos propuestos se puede etiquetar todo el conjunto de datos con un accuracy del 97% aproximado, etiquetando en forma manual solo el 33% aproximadamente del total de imágenes, mientras que usando el algoritmo Random se puede etiquetar todo el conjunto de datos pero con un accuracy del 95% aproximado.
Además, se puede observar que agregando 750 imágenes etiquetadas al conjunto de entrenamiento muestreadas con los algoritmos propuestos, se alcanza el mismo accuracy que agregando 1200 imágenes etiquetadas al conjunto de entrenamiento muestreadas con el algoritmo Random.
En el Apéndice A, se pueden ver las Tablas A.1, A.2, A.3, con los resultados del accuracy de cada ejecución/iteración de los algoritmos Low Confidence + Cluster, Low Confidence + Representative Cluster y Random respectivamente.

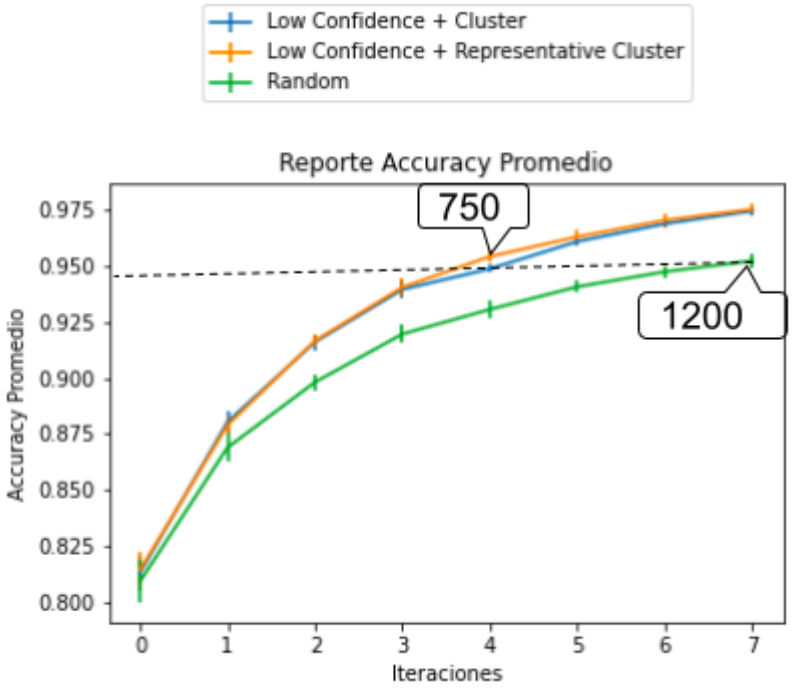


Figura 5.6. Reporte del promedio de accuracy y el error estándar de la media en 10 ejecuciones, cada una con 7 iteraciones, de los algoritmos Low Confidence + Cluster, Low Confidence + Cluster Representative y Random.

La Figura 5.7 muestra la medida F promedio con el error estándar de la media de 10 ejecuciones de los algoritmos. Dado que el conjunto de datos es balanceado, no se observa un cambio notable con respecto a la diferencia entre el accuracy promedio y la diferencia de la medida F promedio de los algoritmos testeados.
En el Apéndice A, se pueden ver las Tablas A.4, A.5, A.6, con los resultados de la medida F de cada ejecución/iteración de los algoritmos Low Confidence + Cluster, Low Confidence + Representative Cluster y Random respectivamente.

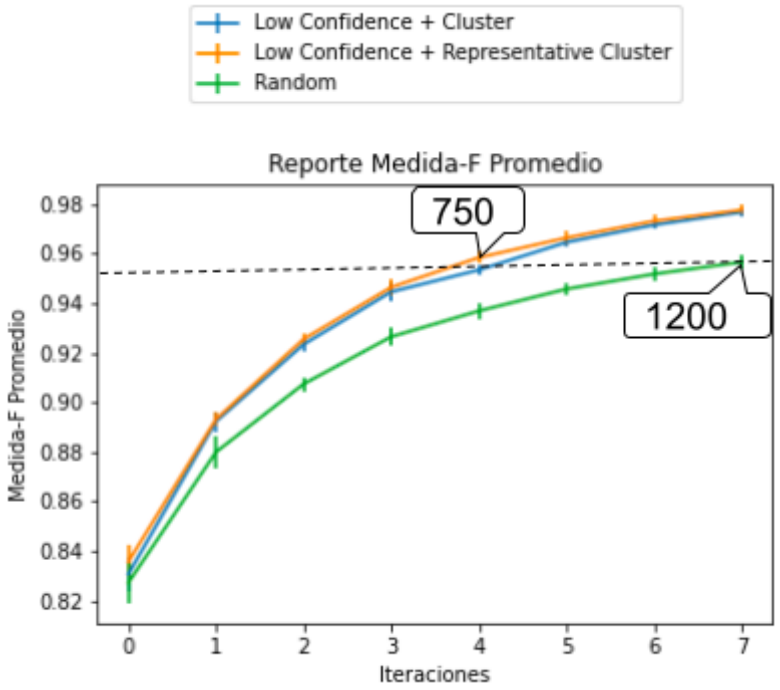


Figura 5.7. Reporte del promedio de la medida F y el error estándar de la media en 10 ejecuciones, cada una con 7 iteraciones, de los algoritmos Low Confidence + Cluster, Low Confidence + Cluster Representative y Random.

La Tabla 5.1. muestra los tiempos de muestreo promedio (en segundos) para cada iteración de cada algoritmo. Claramente se puede apreciar que el algoritmo Random es el que toma menos tiempo lo cual no sorprende porque solo hace una selección al azar de imágenes para muestrear. Por otro lado, cada muestreo del algoritmo Low Confidence + Cluster toma menos de 90 segundos (1.5 minutos) y a medida que se quitan imágenes del conjunto de datos sin etiquetar, el tiempo de muestreo disminuye. Lo contrario ocurre con el tiempo de muestreo del algoritmo Low Confidence + Representative Custer que varía desde los 120 segundos a los 160 segundos aproximadamente (menos de 3 minutos), pero que a medida que se quitan imágenes del conjunto de datos sin etiquetar y se mueven al conjunto de entrenamiento, el tiempo de muestreo crece debido a la agrupación en clúster del conjunto de entrenamiento para el muestreo representativo. De todas formas, en este caso la diferencia de tiempo entre los algoritmos Low Confidence + Cluster y Low Confidence + Representative Custer con respecto al algoritmo Random no es significativa teniendo en cuenta el tiempo que consume el etiquetado humano de 150 imágenes médicas, por lo que estas diferencias en tiempo de muestreo se justifican ampliamente con el ahorro en etiquetado humano resultante de la aplicación de los algoritmos propuestos.

Tiempo de muestreo promedio							
# Iteración	1	2	3	4	5	6	7
Low Confidence + Cluster	84,20	80,21	77,37	74,53	71,93	69,23	66,21
Low Confidence +	122,791	127,99	133,77	140,09	145,96	153,26	162,13

Representative Cluster							
Random	0,36	0,31	0,31	0,31	0,31	0,30	0,30

Tabla 5.1. Tiempo de muestreo promedio (en segundos).
Muestra el tiempo promedio en segundos, que demora cada algoritmo en cada muestreo (iteración).

5.4.1.2. Prueba en un conjunto de datos Neumonía

La Figura 5.8 muestra el accuracy promedio con el error estándar de la media de 10 ejecuciones de los algoritmos. El paso 0 es el promedio del accuracy del conjunto de entrenamiento inicial (150 imágenes elegidas al azar). Los pasos siguientes corresponden al accuracy promedio con la selección de 150 imágenes con cada algoritmo. En el paso 1 y 2 se puede apreciar un gran salto de los algoritmos Low Confidence + Cluster y Los Confidence + Representative Cluster con respecto al algoritmo Random y esta diferencia se mantiene en los siguientes pasos obteniendo una mejora apreciable, aunque, como en el experimento 5.4.1.1 no se puede establecer una diferencia entre los dos primeros por el mismo motivo.
El experimento demuestra que, etiquetando en forma manual aproximadamente el 25% del total de imágenes, se puede etiquetar todo el conjunto de datos con un accuracy del 97% aproximado usando los algoritmos propuestos y con un accuracy del 96% aproximado usando el algoritmo Random.
Además, se puede observar que agregando 450 imágenes etiquetadas al conjunto de entrenamiento muestreadas con los algoritmos propuestos, se alcanza el mismo accuracy que agregando 1200 imágenes etiquetadas al conjunto de entrenamiento muestreadas con el algoritmo Random.
En el Apéndice A, se pueden ver las Tablas A.7, A.8, A.9, con los resultados del accuracy de cada ejecución/iteración de los algoritmos Low Confidence + Cluster, Low Confidence + Representative Cluster y Random respectivamente.

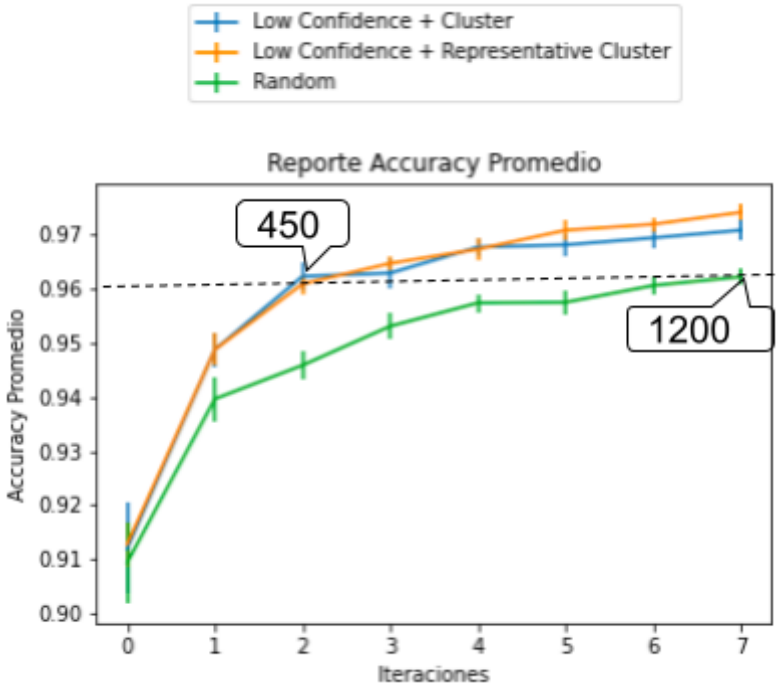


Figura 5.8. Reporte del promedio de accuracy y el error estándar de la media en 10 ejecuciones, cada una con 7 iteraciones, de los algoritmos Low Confidence + Cluster, Low Confidence + Cluster Representative y Random.

La Figura 5.9 muestra la medida F promedio con el error estándar de la media de 10 ejecuciones de los algoritmos. Dado que este conjunto de datos es desbalanceado es natural que haya una variación en la diferencia de los valores promedio de la medida F y la diferencia del promedio del accuracy. Si bien se observa que los valores promedio de la media F son más bajos que el accuracy promedio, la diferencia entre los algoritmos Low Confidence + Cluster y Low Confidence + Representative Cluster con respecto al algoritmo Random se ha incrementado, lo que representa una mejora en los dos primeros con respecto a este último.

En el Apéndice A, se pueden ver las Tablas A.10, A.11, A.12, con los resultados del accuracy de cada ejecución/iteración de los algoritmos Low Confidence + Cluster, Low Confidence + Representative Cluster y Random respectivamente.

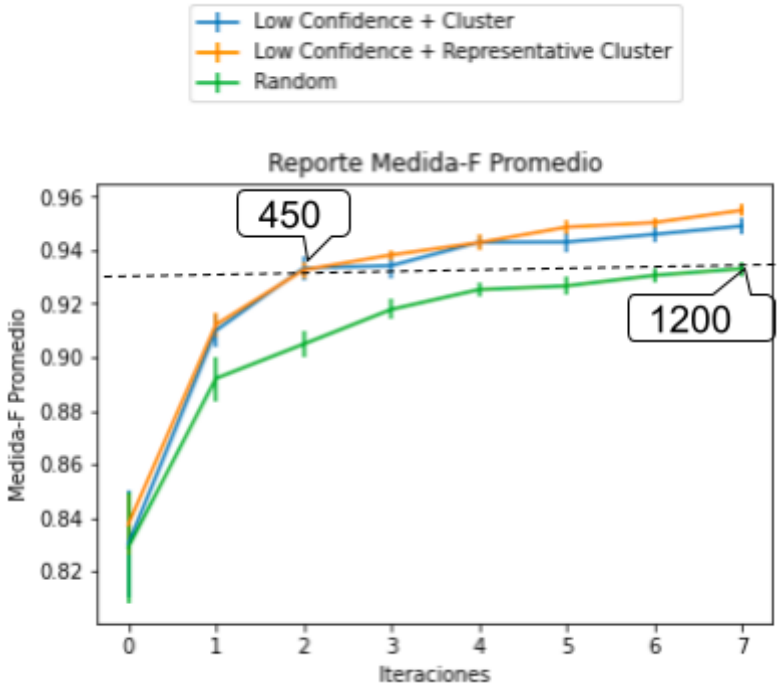


Figura 5.9. Reporte del promedio de la medida F y el error estándar de la media en 10 ejecuciones, cada una con 7 iteraciones, de los algoritmos Low Confidence + Cluster, Low Confidence + Cluster Representative y Random.

La Tabla 5.2 muestra los tiempos de muestreo promedio (en segundos) para cada iteración de cada algoritmo. Como en el caso del experimento en el conjunto de datos de Tumores Cerebrales, el algoritmo Random supera ampliamente al resto de los algoritmos en términos de tiempo de muestreo y la tendencia en los algoritmos Low Confidence + Cluster y Low Confidence + Representative Cluster se mantiene con un leve aumento en los tiempos de muestreo ya que este conjunto de datos tiene unas 1000 imágenes más que el conjunto de datos de Tumores Cerebrales, por lo que los algoritmos tienen mas imagenes para procesar.

Tiempo de muestreo promedio							
# Iteración	1	2	3	4	5	6	7
Low Confidence + Cluster	104,65	100,89	97,71	94,911	91,83	88,91	86,38
Low Confidence + Representative Cluster	154,37	159,85	165,19	172,15	178,51	185,88	193,12
Random	0,45	0,39	0,39	0,39	0,39	0,38	0,39

Tabla 5.2. Tiempo de muestreo promedio (en segundos).
Muestra el tiempo promedio en segundos, que demora cada algoritmo en cada muestreo (iteración).

Capítulo 6

Conclusión y trabajo futuro

6.1. Conclusión

En este proyecto busqué desarrollar un sistema para el etiquetado de un conjunto de datos a gran escala para tareas de visión artificial. El objetivo principal es minimizar el costo de producción, donde el costo puede medirse en esfuerzo humano, tiempo, dinero, etc.

Al enfrentarnos a un problema relacionado con inteligencia artificial, en general pensamos en diseñar algoritmos o modelos para la resolución del mismo con la premisa de que los datos ya están dados, es lo que tenemos para trabajar y debemos mejorar y ajustar el modelo en base a ellos. En los últimos años se ha comenzado a cambiar este paradigma y se propone dar más protagonismo a los datos, mejorando la calidad de los mismos para que no solo el modelo se ajuste a los datos, sino también los datos al modelo. Esto es un enfoque novedoso del cual se está comenzando a hablar, lo que implicó un gran desafío al momento de buscar material relacionado.

En este proyecto implementé un framework, el cual llamé TrainingSet (TS), que implementa la estrategia denominada “Human in the loop”, donde el conjunto de datos se genera de forma incremental seleccionando un pequeño subconjunto de imágenes para ser revisadas y etiquetadas de forma manual. Este subconjunto es agregado al conjunto de datos actual y con él se entrena un clasificador que es utilizado para muestrear un nuevo subgrupo de imágenes para el etiquetado manual.

La parte más importante de este proceso es la selección estratégica de imágenes, detectando dónde el clasificador tiene mayor incertidumbre y presentando esas imágenes para el etiquetado manual. De esta forma es el humano en el ciclo el encargado de resolver la incertidumbre del clasificador y, al agregar estas imágenes etiquetadas al conjunto de entrenamiento en el que se entrena el clasificador, éste adquiere ese conocimiento y puede predecir de manera correcta imágenes con similares características en la siguiente iteración, propagando así esas etiquetas a otras imágenes. Esto implica una reducción de la participación del ojo humano en el proceso (costo).

Para ello implementé dos algoritmos:

- Low Confidence + Cluster: Este algoritmo retorna imágenes diversas donde el clasificador tiene mayor incertidumbre.
- Low Confidence + Representative Cluster: Este algoritmo retorna imágenes diversas y representativas del conjunto de datos sin etiquetar con respecto al conjunto de entrenamiento, donde el clasificador tiene mayor incertidumbre.

Ambos algoritmos fueron testeados en un conjunto de datos de juguete y se verificó el funcionamiento esperado observando en dos dimensiones cómo cada algoritmo selecciona los datos que están cerca de la frontera de decisión.
Posteriormente, ambos algoritmos fueron testeados en dos conjuntos de datos reales y se comparó el accuracy, medida F y tiempo de muestreo con un algoritmo que selecciona imágenes en forma aleatoria. Se observó que ambos algoritmos superan al aleatorio en un 2% aproximado de accuracy y medida F promedio en un conjunto de datos balanceado y en un 1% aproximado del accuracy promedio y 2% aproximado de la medida F promedio en un conjunto de datos desbalanceado. Además, se observó que fue necesario etiquetar de forma manual un 37.5% menos de imágenes para alcanzar el mismo resultado que usando selección aleatoria en el conjunto de datos de tumores cerebrales y un 62.5% menos de imágenes para alcanzar el mismo resultado que usando selección aleatoria en el conjunto de datos de neumonía. Con respecto al tiempo de muestreo el algoritmo Random supera ampliamente a los dos algoritmos propuestos ya que solo muestrea imagenes aleatoriamente, mientras que en los algoritmos Low Confidence + Cluster y Low Confidence + Representative Clusters no se puede establecer una diferencia significativa. De todas formas, las diferencias no son significativas con respecto al tiempo que consume etiquetar manualmente la cantidad de imágenes muestreadas ni con respecto al tiempo que se ahorra en etiquetado humano si se utilizara el algoritmo Random para lograr la misma performance.
En términos generales puedo concluir que conseguí el objetivo de implementar algoritmos que contribuyen a la reducción del costo del esfuerzo humano en la construcción de un conjunto de datos para visión artificial y además en un tiempo de muestreo que no es prohibitivo, 1.5 minutos promedio aproximadamente para cada muestreo usando el algoritmo Low Confidence + Cluster y algo más de 3 minutos promedio usando el algoritmo Low Confidence + Representative Cluster en el conjunto de datos de mayor tamaño (Neumonía).

6.2. Trabajo Futuro

- Selección representativa del conjunto de entrenamiento inicial.

  Los algoritmos propuestos necesitan tener un conjunto de datos de entrenamiento inicial no vacío para comenzar a ser utilizados. En este proyecto la selección de datos inicial fue hecha de forma aleatoria, pero una selección de datos representativos del dominio del problema puede ser un buen comienzo y podría disminuir aún más el costo de generación del conjunto de datos. Se podría utilizar la solución propuesta por Elhamifar, Sapiro y Vidal [33] para este propósito.

- Interfaz de anotación.

Implementar una interfaz de anotación adecuada, con un diseño intuitivo, amigable, con instrucciones claras, texto descriptivo y de ayuda, con opciones para anotar, retroceder, o pasar al siguiente ejemplo, etc. Un diseño incorrecto puede afectar la calidad y la eficiencia del proceso de anotación. Para ello debe cumplir con lo siguiente:

1. Cada componente debe funcionar de la manera que esperamos que funcione, esto quiere decir que si hay un botón con un signo “?”, éste debe mostrarte una ayuda, si hay un botón con un signo “+” éste expanda un contenido oculto, etc.
2. Cada acción en la interfaz debe tener una retroalimentación, es decir que si un usuario hace click en un botón, se debe informar al anotador el resultado de su acción. Ej: botón “Guardar cambios”.
3. Los anotadores deben sentir que la interfaz le permite anotar correctamente y que su trabajo ayuda al proyecto en el que está trabajando.
4. Debe manejar sesiones de usuarios, de forma de poder dejar registro de que anotador etiquetó cada dato. Esto permitirá realizar determinadas acciones, por ejemplo puntuar a un anotador o saber si un dato fue etiquetado por un humano o por el algoritmo.

- Generalizar la solución propuesta para clasificar más categorías de imágenes.

  Actualmente en este proyecto sólo se permite la clasificación binaria de datos, pero sería interesante poder extender la solución a más clases de datos. Estudiar qué impacto tiene esta generalización en el sistema en términos de costos, que modificaciones son necesarias, por ejemplo resolver las siguientes preguntas: ¿qué estrategia usar para medir incertidumbre?, ¿mínima confianza?, ¿porcentaje de confianza?, ¿entropía?, en caso de usar estrategias de clustering, ¿cual es la cantidad mínima de clusters?, etc.

- Otras técnicas para el muestreo.

  Como se mencionó en la introducción de este documento, este proyecto está fuertemente basado en el libro **Human-in-the-Loop, Machine Learning** [4], en el cual se detallan una serie de técnicas para los diferentes tipos de muestreo (incertidumbre, diversidad) y sus posibles combinaciones las que sería interesante explorar. Algunos ejemplos:

  1. Muestreo de valores atípicos (outliers) basado en modelos y muestreo representativo: se enfoca en elementos que actualmente son desconocidos para su modelo, pero que son relativamente comunes (representativos) en el conjunto de datos sin etiquetar.
  2. Agrupación jerárquica de clusters: es una técnica que sirve para muestrear diversidad dentro de un cluster, útil para cuando tenemos clusters con muchos elementos.
  3. Combinación de modelos con estrategias individuales: crear varios modelos y usarlos según la conveniencia, es posible que un modelo bayesiano sea mejor para determinar la incertidumbre, pero un modelo neuronal sea mejor para

el muestreo de valores atípicos basado en modelos.

Referencias

[1]. The Batch - https://read.deeplearning.ai/the-batch/issue-84/ - Consultado el 20/07/2021

[2]. The Verge - https://www.theverge.com/2018/1/12/16882408/google-racist-gorillas-photo-recognition-algorithm-ai - Consultado el 22/07/2021

[3]. The Seattle Times - https://www.seattletimes.com/business/microsoft/how-linkedins-search-engine-may-reflect-a-bias/ - Consultado el 22/07/2021

[4] Monarch, R. M. (2021). *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster.

[5] YU, Fisher, et al. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[6] MITHUN, Niluthpol Chowdhury; PANDA, Rameswar; ROY-CHOWDHURY, Amit K. Generating diverse image datasets with limited labeling. En *Proceedings of the 24th ACM international conference on Multimedia*. 2016. p. 566-570.

[7] TOUMANIDIS, Lazaros, et al. ActiveCrowds: A Human-in-the-Loop Machine Learning Framework. En *Novelties in Intelligent Digital Systems*. IOS Press, 2021. p. 176-184.

[8] BAI, Yalong, et al. Automatic image dataset construction from click-through logs using deep neural network. En *Proceedings of the 23rd ACM international conference on Multimedia*. 2015. p. 441-450

[9] GUO, Sheng, et al. Curriculumnet: Weakly supervised learning from large-scale web images. En *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018. p. 135-150.

[10] Aurélien Géron. (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 2nd Edición*

[11] PAULLADA, Amandalynne, et al. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *arXiv preprint arXiv:2012.05345*, 2020.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image
database. In *CVPR*, pages 248–255. IEEE, 2009.

[13] C. M. Bishop (2006). Pattern Recognition and Machine Learning. Springer.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] scikit-learn - https://scikit-learn.org/stable/ - Consultado el 17/08/2021

[16] Sklearn Datasets

- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html - Consultado el 29/06/2021

[17] Colaboraty - https://colab.research.google.com/notebooks/intro.ipynb - Consultado el 17/08/2021

[18] TensorFlow - https://www.tensorflow.org/ - Consultado el 18/08/2021

[19] TensorFlow Core - https://www.tensorflow.org/guide/keras - Consultado el 18/08/2021
[20] Proyecto Jupyter - https://jupyter.org - Consultado el 16/08/2021
[21] Python - https://www.python.org/downloads/ - Consultado el 16/08/2021
[22] Kaggle - https://www.kaggle.com - Consultado el 15/07/2021
[23] Brian tumor dataset - https://www.kaggle.com/preetviradiya/brian-tumor-dataset - Consultado el 15/07/2021
[24] Chest X-Ray Images (Pneumonia)
- https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia - Consultado el 19/07/2021
[25] https://www.tensorflow.org/tutorials/images/transfer_learning - Consultado el 02/06/2021
[26] MIKOLOV, Tomas, et al. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
[27] SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[28] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition
using places database. In *NIPS*, 2014.
[29] Amazon Mechanical Turk - https://www.mturk.com/ - Consultado el 27/07/2021
[30] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. IJCV, 2010.
[31] R. Speer and C. Havasi. Conceptnet 5: A large semantic network for relational knowledge. In The Peoples Web Meets NLP. 2013.
[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, pages 1097–1105, 2012.
[33] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In CVPR, pages 1600–1607. IEEE, 2012.
[34] Y. Lin, J. Michel, E. Aiden, J. Orwant, W. Brockman, and S. Petrov, “Syntactic annotations for the google books ngram corpus,” ACL 2012 System Demonstrations, 169–174, 2012.
[35] J. Howe, “The rise of crowdsourcing,” Wired magazine, vol. 14, pp. 1–4, June 2006.
[36] C. M. Yeum, S. J. Dyke, and J. Ramirez, “Visual data classification in post-event building reconnaissance,” Engineering Structures, vol. 155, pp. 16–24, 2018.
[37] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167 (2015)

Apéndice A

A.1. Reporte de medidas en conjunto de datos Tumores Cerebrales

A.1.1. Reporte accuracy

Reporte Accuracy Algoritmo Low confidence + Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,7960000038	0,8610000014	0,9120000005	0,9340000153	0,9570000172	0,9620000124	0,9670000076	0,9720000029
2	0,7820000052	0,8740000129	0,898999989	0,925999999	0,94599998	0,9610000253	0,9620000124	0,9750000238
3	0,8090000153	0,8590000272	0,9079999924	0,9279999733	0,9530000091	0,9520000219	0,9639999866	0,9689999819
4	0,8000000119	0,8849999905	0,9250000119	0,9279999733	0,9509999752	0,9589999914	0,9639999866	0,9739999771
5	0,8389999866	0,8849999905	0,90200001	0,9200000167	0,9480000138	0,9629999995	0,9739999771	0,9739999771
6	0,8249999881	0,8799999952	0,9139999747	0,9350000024	0,9440000057	0,9620000124	0,9720000029	0,9800000191
7	0,8230000138	0,8870000243	0,9210000038	0,94599998	0,9599999785	0,9660000205	0,9660000205	0,976000011
8	0,8130000234	0,8690000176	0,9070000052	0,92900002	0,94599998	0,9539999962	0,9700000286	0,976000011
9	0,8309999704	0,8489999771	0,9039999843	0,9190000296	0,9359999895	0,9559999704	0,9700000286	0,9700000286
10	0,8090000153	0,8659999967	0,9160000086	0,9319999814	0,9449999928	0,9589999914	0,9580000043	0,9689999819
Promedio	0,8127000034	0,8715000033	0,910799998	0,9296999991	0,9485999942	0,9594000041	0,9667000055	0,9735000014
Error estándar	0,0054712351	0,0040722645	0,0026449121	0,0024632828	0,0021919048	0,0013678872	0,0015495519	0,0011180364

Tabla A.1. Reporte del accuracy de cada ejecución/iteración del algoritmo Low Confidence + Cluster (Filas 1 a 10).
Promedio de los accuracy para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio del accuracy de todas las ejecuciones en cada iteración (Fila 12).

Reporte Accuracy Algoritmo Low Confidence + Representative Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,824000001	0,8830000162	0,9049999714	0,9250000119	0,9440000057	0,9589999914	0,9649999738	0,9750000238
2	0,8320000172	0,8600000143	0,898999989	0,9219999909	0,9369999766	0,9530000091	0,9649999738	0,9739999771
3	0,7950000167	0,8840000033	0,90200001	0,9279999733	0,9499999881	0,9599999785	0,9610000253	0,9679999948
4	0,8140000105	0,8790000081	0,9169999957	0,9399999976	0,9440000057	0,9599999785	0,9679999948	0,9779999852
5	0,8439999819	0,875	0,9160000086	0,9350000024	0,9480000138	0,9599999785	0,9710000157	0,9660000205
6	0,7620000243	0,8569999933	0,9070000052	0,9210000038	0,9449999928	0,9509999752	0,9649999738	0,97299999
7	0,8000000119	0,8820000291	0,8980000019	0,9150000215	0,9319999814	0,9499999881	0,9570000172	0,9769999981
8	0,7950000167	0,8799999952	0,9079999924	0,9390000105	0,9580000043	0,9700000286	0,97299999	0,976000011
9	0,8159999847	0,8899999857	0,9160000086	0,9449999928	0,9580000043	0,9620000124	0,9750000238	0,9800000191
10	0,8249999881	0,8809999824	0,9229999781	0,9409999847	0,9419999719	0,9530000091	0,9639999866	0,9700000286
Promedio	0,8107000053	0,8771000028	0,9090999961	0,931099999	0,9457999945	0,9577999949	0,9663999975	0,9737000048
Error estándar	0,0074252537	0,0033348325	0,00268514395	0,0032298934	0,0026025655	0,0019310355	0,0017333336	0,0014224382

Tabla A.2. Reporte del accuracy de cada ejecución/iteración del algoritmo Low Confidence + Representative Cluster (Filas 1 a 10).
Promedio de los accuracy para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio del accuracy de todas las ejecuciones en cada iteración (Fila 12).

Reporte Accuracy Algoritmo Random								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,8140000105	0,8740000129	0,8840000033	0,9060000181	0,9179999828	0,9309999943	0,9480000138	0,9470000267
2	0,80400002	0,8519999981	0,8939999938	0,9229999781	0,9399999976	0,9369999766	0,9440000057	0,9549999833
3	0,8159999847	0,8629999757	0,9029999971	0,9139999747	0,9269999862	0,9279999733	0,9419999719	0,9470000267
4	0,7810000181	0,84799999	0,8939999938	0,9129999876	0,9190000296	0,9250000119	0,9430000186	0,94599998
5	0,8320000172	0,8679999709	0,8970000148	0,9160000086	0,9200000167	0,9279999733	0,9369999766	0,9449999928
6	0,773999989	0,8550000191	0,8690000176	0,90200001	0,9139999747	0,9179999828	0,9190000296	0,9279999733
7	0,7990000248	0,8809999824	0,9010000229	0,9169999957	0,9300000072	0,9380000234	0,9480000138	0,9499999881
8	0,7969999909	0,8889999986	0,9160000086	0,9250000119	0,9319999814	0,9440000057	0,9509999752	0,9589999914
9	0,7810000181	0,8640000224	0,8899999857	0,9229999781	0,9219999909	0,9409999847	0,9369999766	0,9530000091
10	0,8270000219	0,8790000081	0,8999999762	0,9169999957	0,9330000281	0,9350000024	0,9390000105	0,94599998
Promedio	0,8025000095	0,8672999978	0,8948000014	0,9155999959	0,9254999995	0,9324999928	0,9407999992	0,9475999951
Error estándar	0,0062950606	0,0042532338	0,0039350273	0,0023295185	0,0025916542	0,0025177166	0,0028511186	0,0026170397

Tabla A.3. Reporte del accuracy de cada ejecución/iteración del algoritmo Random (Filas 1 a 10).
Promedio de los accuracy para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio del accuracy de todas las ejecuciones en cada iteración (Fila 12).

A.1.2. Reporte medida-F

Reporte medida-F Algoritmo Low confidence + Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,8091059923	0,8820872307	0,9229013324	0,9424402118	0,9620088339	0,9665175676	0,9711426497	0,9761253595
2	0,8237006068	0,8855810165	0,9075846672	0,931388855	0,9493657351	0,9628650546	0,9642634392	0,9760066271
3	0,8348118067	0,8714841008	0,9203990698	0,937471509	0,9580112696	0,9566026926	0,9678139687	0,972386539
4	0,8036187887	0,9000108838	0,9315524101	0,9322436452	0,9538098574	0,9624778628	0,9674046636	0,9768586159
5	0,86395818	0,8994249105	0,9137242436	0,9287034869	0,9536220431	0,9670847058	0,9766147733	0,9767489433
6	0,8382132649	0,8890104294	0,919333756	0,9394844174	0,9483798742	0,9656259418	0,9747503996	0,9818166494
7	0,8447879553	0,8907062411	0,9261566401	0,9500934482	0,9625450969	0,9680404663	0,9675565958	0,9771710634
8	0,8102884293	0,8720374107	0,9122487903	0,9310951233	0,9480875134	0,9570096135	0,9710424542	0,9772963524
9	0,8503290415	0,8695735931	0,9087259173	0,923034966	0,9388945699	0,9583231807	0,9716769457	0,971622467
10	0,8397073746	0,8807362318	0,9209678769	0,9356460571	0,9485493898	0,9612406492	0,9608300924	0,9705992937
Promedio	0,831852144	0,8840652049	0,9183594704	0,935160172	0,9523274183	0,9625787735	0,9693095982	0,9756631911
Error estándar	0,00623289457	0,00347849711	0,00243839308	0,00242267566	0,00228851477	0,00134021026	0,00149532435	0,00104590390

Tabla A.4. Reporte de la medida F de cada ejecución/iteración del algoritmo Low Confidence + Cluster (Filas 1 a 10).

Promedio de la medida F para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio de la medida F de todas las ejecuciones en cada iteración (Fila 12).

Reporte medida-F Algoritmo Low Confidence + Representative Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,843097806	0,8944851756	0,9153512716	0,9334648252	0,9507930875	0,9635785222	0,9691666365	0,9784315825
2	0,841389358	0,8770793676	0,9095076323	0,927982688	0,9401485324	0,95416677	0,9663411379	0,9746419787
3	0,8092344999	0,8990756273	0,9110921025	0,9363697767	0,9557543993	0,9641855955	0,9650359154	0,9713460803
4	0,8230400085	0,8902280927	0,9258027077	0,9465885162	0,9502006769	0,9644848108	0,9715348482	0,9802180529
5	0,8679698706	0,8951527476	0,9271422625	0,9436526299	0,9547956586	0,9646853209	0,974131763	0,9692050219
6	0,815905273	0,8794182539	0,9178432226	0,9306725264	0,950267911	0,9566581845	0,9682322741	0,9756817818
7	0,831305027	0,8972943425	0,9082528949	0,9233883023	0,9384915233	0,9548158646	0,9607855678	0,9789409637
8	0,782532692	0,8828607798	0,9138163328	0,9426898956	0,9589602351	0,9720517993	0,9738816023	0,9772911072
9	0,8353402019	0,8987256289	0,9205399752	0,9473365545	0,959871769	0,9630090594	0,9755064249	0,9798947573
10	0,8474811316	0,8920232058	0,9272558093	0,9445292354	0,9464457631	0,95672369	0,9668174982	0,9721251726
Promedio	0,8297295868	0,8906343222	0,9176604211	0,937667495	0,9505729556	0,9614359617	0,9691433668	0,9757776499
Error estándar	0,00746064845	0,00255687515	0,00229690704	0,00268156688	0,00229355714	0,00179156978	0,0014695404	0,00121785784

Tabla A.5. Reporte de la medida F de cada ejecución/iteración del algoritmo Low Confidence + Representative Cluster (Filas 1 a 10).
Promedio de la medida F para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio de la medida F de todas las ejecuciones en cada iteración (Fila 12).

Reporte medida-F Algoritmo Random								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,8347405195	0,8903826475	0,8973048329	0,9169746637	0,9270516634	0,9385288358	0,9536393881	0,9532607794
2	0,8363292813	0,8734502792	0,9035571814	0,9282333255	0,9436688423	0,9405625463	0,9474061131	0,9567320943
3	0,8347759247	0,8752976656	0,914401412	0,9233290553	0,9345693588	0,9352310896	0,9474247694	0,9514745474
4	0,7945235372	0,8671509624	0,902326405	0,923350513	0,9286931753	0,9347991943	0,9499570727	0,9520645142
5	0,8485237956	0,8843982816	0,9099808931	0,9259601831	0,9298470616	0,9360319376	0,9457849264	0,9517143369
6	0,8226754069	0,8773379326	0,8872416615	0,9130220413	0,923306942	0,9270352125	0,9279853106	0,9361154437
7	0,819609046	0,8940017819	0,9086600542	0,9243275523	0,934956193	0,9434807897	0,9518946409	0,9537348747
8	0,7864472866	0,8898268938	0,918083787	0,9269992709	0,933708787	0,9458548427	0,9511864781	0,9600439072
9	0,8209896088	0,8803097606	0,8996882439	0,9282793999	0,9269310236	0,9439032674	0,9408127666	0,9554927945
10	0,8435786963	0,8803024292	0,9026708603	0,9201518893	0,9356662035	0,9373340607	0,9420152903	0,949201107
Promedio	0,8242193103	0,8812458634	0,9043915331	0,9230627894	0,9318399251	0,9382761776	0,9458106756	0,9519834399
Error estándar	0,0063917629	0,0026700453	0,0028117188	0,00158446201	0,0018527084	0,0017484122	0,0023727019	0,00201220769

Tabla A.6. Reporte de la medida F de cada ejecución/iteración del algoritmo Random (Filas 1 a 10).
Promedio de la medida F para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio de la medida F de todas las ejecuciones en cada iteración (Fila 12).

A.2. Reporte de medidas en conjunto de datos Neumonía

A.2.1. Reporte accuracy

Reporte Accuracy Algoritmo Low confidence + Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,9160000086	0,9539999962	0,9649999738	0,9720000029	0,9739999771	0,97299999	0,9769999981	0,9779999852
2	0,9210000038	0,9430000186	0,9599999785	0,9620000124	0,9689999819	0,9639999866	0,9639999866	0,9660000205
3	0,9390000105	0,949000001	0,9570000172	0,9629999995	0,9620000124	0,9689999819	0,9639999866	0,9639999866
4	0,9060000181	0,9449999928	0,9710000157	0,9620000124	0,9670000076	0,9679999948	0,9689999819	0,9700000286
5	0,9350000024	0,9580000043	0,9660000205	0,9670000076	0,9710000157	0,9710000157	0,9689999819	0,9679999948
6	0,8529999852	0,925999999	0,9440000057	0,9440000057	0,9580000043	0,9559999704	0,9620000124	0,9620000124
7	0,9409999847	0,9589999914	0,9750000238	0,976000011	0,976000011	0,9779999852	0,9789999723	0,9779999852
8	0,8909999728	0,949000001	0,9580000043	0,9610000253	0,9649999738	0,9620000124	0,9639999866	0,9670000076
9	0,9029999971	0,9440000057	0,9649999738	0,9580000043	0,9649999738	0,9700000286	0,9700000286	0,976000011
10	0,9150000215	0,9599999785	0,9610000253	0,9629999995	0,9689999819	0,9689999819	0,9750000238	0,9779999852
Promedio	0,9120000005	0,9486999989	0,9622000039	0,9628000081	0,9675999939	0,9679999948	0,9692999959	0,9707000017
Error estándar	0,0083293342	0,0032181761	0,002694851	0,0027030846	0,0017139944	0,0019321847	0,0018976594	0,0019779314

Tabla A.7. Reporte del accuracy de cada ejecución/iteración del algoritmo Low Confidence + Cluster (Filas 1 a 10).

Promedio de los accuracy para todas las ejecuciones en cada iteración (Fila 11).

Error estándar del promedio del accuracy de todas las ejecuciones en cada iteración (Fila 12).

Reporte Accuracy Algoritmo Low Confidence + Representative Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,9060000181	0,9570000172	0,9660000205	0,9679999948	0,9750000238	0,9800000191	0,9800000191	0,9829999804
2	0,9120000005	0,949000001	0,9580000043	0,9620000124	0,9620000124	0,9660000205	0,9679999948	0,9739999771
3	0,9070000052	0,9470000267	0,9639999866	0,9639999866	0,9700000286	0,976000011	0,9720000029	0,976000011
4	0,9070000052	0,9309999943	0,9570000172	0,9639999866	0,9599999785	0,9700000286	0,9750000238	0,9769999981
5	0,9409999847	0,9639999866	0,9720000029	0,9700000286	0,9679999948	0,9710000157	0,9710000157	0,97299999
6	0,9160000086	0,9449999928	0,9559999704	0,9589999914	0,9589999914	0,9620000124	0,9679999948	0,9649999738
7	0,9089999795	0,9499999881	0,9599999785	0,9700000286	0,9769999981	0,97299999	0,97299999	0,976000011
8	0,8899999857	0,9559999704	0,9629999995	0,9660000205	0,9679999948	0,9629999995	0,9660000205	0,9710000157
9	0,9210000038	0,9480000138	0,9499999881	0,9620000124	0,9700000286	0,97299999	0,9720000029	0,9710000157
10	0,9190000296	0,9399999976	0,9620000124	0,9610000253	0,9629999995	0,97299999	0,97299999	0,9739999771
Promedio	0,9128000021	0,9486999989	0,960799998	0,9646000087	0,9672000051	0,9707000077	0,9718000054	0,973999995
Error estándar	0,0041627982	0,0029137596	0,0019194925	0,0012037001	0,0019367813	0,0017891646	0,0012631540	0,0014832398

Tabla A.8. Reporte del accuracy de cada ejecución/iteración del algoritmo Low Confidence + Representative Cluster (Filas 1 a 10).

Promedio de los accuracy para todas las ejecuciones en cada iteración (Fila 11).

Error estándar del promedio del accuracy de todas las ejecuciones en cada iteración (Fila 12).

Reporte Accuracy Algoritmo Random								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,8600000143	0,9390000105	0,9430000186	0,9629999995	0,9610000253	0,9580000043	0,9570000172	0,9610000253
2	0,8980000019	0,9409999847	0,9480000138	0,9509999752	0,9539999962	0,9549999833	0,9629999995	0,9649999738
3	0,9229999781	0,9419999719	0,9480000138	0,9559999704	0,9589999914	0,9629999995	0,9639999866	0,9639999866
4	0,9169999957	0,9470000267	0,9359999895	0,9559999704	0,9530000091	0,9559999704	0,9620000124	0,9599999785
5	0,9330000281	0,949000001	0,9559999704	0,9580000043	0,9589999914	0,9660000205	0,9670000076	0,9649999738
6	0,9169999957	0,9089999795	0,9369999766	0,94599998	0,9520000219	0,9470000267	0,9509999752	0,9499999881
7	0,9399999976	0,9539999962	0,9620000124	0,9639999866	0,9639999866	0,9670000076	0,9620000124	0,9629999995
8	0,9150000215	0,949000001	0,9499999881	0,9520000219	0,9660000205	0,9599999785	0,9639999866	0,9639999866
9	0,9060000181	0,9390000105	0,9380000234	0,9430000186	0,9520000219	0,9509999752	0,9589999914	0,9670000076
10	0,8849999905	0,9269999862	0,9399999976	0,9409999847	0,9530000091	0,9509999752	0,9559999704	0,9620000124
Promedio	0,9094000041	0,9395999968	0,9458000004	0,9529999912	0,9573000073	0,9573999941	0,9604999959	0,9620999932
Error estándar	0,00745534825	0,00413978222	0,00272763633	0,00249888800	0,00164688160	0,00211450185	0,00150000232	0,0014940622

Tabla A.9. Reporte del accuracy de cada ejecución/iteración del algoritmo Random (Filas 1 a 10).
Promedio de los accuracy para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio del accuracy de todas las ejecuciones en cada iteración (Fila 12).

A.2.2. Reporte medida-F

Reporte medida-F Algoritmo Low confidence + Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
	0,8297094107	0,9157284498	0,9341604114	0,949110806	0,952802062	0,9501348734	0,9581950903	0,95959723
	0,8507652283	0,9028050303	0,9259285927	0,932221055	0,9445793033	0,9342278242	0,9364285469	0,9407550097
	0,8912912607	0,914865315	0,9274685979	0,9370519519	0,9357010722	0,9479697943	0,9401197433	0,9400317073
	0,8223651648	0,9016591907	0,9472004771	0,9296405911	0,942007184	0,9442032576	0,9437091947	0,9458168149
	0,8867264986	0,9285152555	0,9408925772	0,9408125877	0,9477497339	0,9494590759	0,9450719953	0,9454110861
	0,6782106757	0,8636105657	0,9033234715	0,9002361298	0,9271893501	0,9169185758	0,9344232678	0,9364404678
	0,8848389387	0,925362289	0,9532881975	0,9557606578	0,9541810751	0,9596992731	0,9584129453	0,958337903
	0,7850853205	0,9131120443	0,927093327	0,9325976372	0,937862277	0,9333087802	0,93753016	0,9433168173
	0,8178226352	0,902539432	0,9403236508	0,9304405451	0,9408545494	0,9476575851	0,948099494	0,9584678411
	0,8541747928	0,9313811064	0,9328810573	0,9343803525	0,9465101361	0,9469943047	0,9572389722	0,9619202614
	0,8300989926	0,9099578679	0,933256036	0,9342252314	0,9429436743	0,9430573344	0,945922941	0,9490095139
	0,0200724608	0,00617440612	0,00437935134	0,0046356471	0,00256834619	0,00378060060	0,00292787837	0,00301391864

Tabla A.10. Reporte de la medida F de cada ejecución/iteración del algoritmo Low Confidence + Cluster (Filas 1 a 10).
Promedio de la medida F para todas las ejecuciones en cada iteración (Fila 11).
Error estándar del promedio de la medida F de todas las ejecuciones en cada iteración (Fila 12).

Reporte medida-F Algoritmo Low Confidence + Representative Cluster								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,809258163	0,9226406217	0,9388241768	0,9406915903	0,9534543753	0,9633652568	0,9627376795	0,9685609937
2	0,848205924	0,9057154655	0,9243162274	0,9324030876	0,9321178198	0,9388658404	0,9441326857	0,9537501335
3	0,8110610843	0,9113644361	0,9410800934	0,9405639768	0,9491244555	0,9577744603	0,9519724846	0,9593812823
4	0,8354995847	0,884293437	0,9234842062	0,9348555803	0,9317327738	0,9465802908	0,9541369677	0,9596303701
5	0,9021632075	0,9363319278	0,9495322108	0,9455887675	0,9416357279	0,9494558573	0,9487922788	0,9529384375
6	0,8527094722	0,910474658	0,9257777929	0,9290508032	0,9291612506	0,9386164546	0,9446817636	0,9411579967
7	0,8016119003	0,9081718326	0,9262361526	0,9446724653	0,9583477974	0,9500548244	0,9496883154	0,9561570883
8	0,7816668749	0,9248136282	0,9369882345	0,9419103861	0,9446055293	0,9345788956	0,9406414032	0,9506198764
9	0,8692388535	0,9156047702	0,9225433469	0,9376444817	0,949857533	0,9534276128	0,952462852	0,9511245489
10	0,8629573584	0,9021964073	0,937279582	0,9348689914	0,9382323027	0,9534529448	0,9539483786	0,9555873871
Promedio	0,8374372423	0,9121607184	0,9326062024	0,938225013	0,9428269565	0,9486172438	0,9503194809	0,9548908114
Error estándar	0,011589545	0,00447565709	0,00294090452	0,00170601930	0,00315019794	0,00288093757	0,00198812143	0,00225516452

Tabla A.11. Reporte de la medida F de cada ejecución/iteración del algoritmo Low Confidence + Representative Cluster (Filas 1 a 10).

Promedio de la medida F para todas las ejecuciones en cada iteración (Fila 11).

Error estándar del promedio de la medida F de todas las ejecuciones en cada iteración (Fila 12).

Reporte medida-F Algoritmo Random								
#Ejecución/#Iteración	0	1	2	3	4	5	6	7
1	0,6695252657	0,8798284531	0,8899424672	0,9295532107	0,9259107709	0,9217388034	0,9197860956	0,9270259738
2	0,8351588249	0,8931736946	0,9059689641	0,9125038385	0,9156314731	0,9179137945	0,9339175224	0,9381723404
3	0,8524569273	0,895236671	0,9131549597	0,9233236313	0,9270541072	0,9366922379	0,9379317164	0,9375478625
4	0,8455885053	0,9073016047	0,891420722	0,9214576483	0,9201947451	0,9233829379	0,932226181	0,9271236658
5	0,878162384	0,9118361473	0,9221725464	0,9274663925	0,927646935	0,9407805204	0,9422742724	0,9354578853
6	0,8540437818	0,8331713676	0,8906975985	0,9066381454	0,9193102717	0,9126351476	0,9170957804	0,9144018292
7	0,8929620981	0,9170514941	0,9293775558	0,9331881404	0,9329311252	0,9407730103	0,9264701009	0,930580318
8	0,8671063185	0,917764008	0,9165479541	0,9236962199	0,9437391162	0,935567975	0,9411169887	0,9405891299
9	0,8265277743	0,8980671167	0,8918613195	0,9015907049	0,9185497165	0,918976903	0,9273158908	0,9437333941
10	0,7665609717	0,8672936559	0,8977176547	0,8990141153	0,9215356708	0,9179161191	0,9274917841	0,9359315038
Promedio	0,8288092852	0,8920724213	0,9048861742	0,9178432047	0,9252503932	0,9266377449	0,9305626333	0,9330563903
Error estándar	0,0207598118	0,00829118832	0,00463222021	0,00382287652	0,0026295878	0,00337012093	0,00268967807	0,00270145439

Tabla A.12. Reporte de la medida F de cada ejecución/iteración del algoritmo Random (Filas 1 a 10).

Promedio de la medida F para todas las ejecuciones en cada iteración (Fila 11).

Error estándar del promedio de la medida F de todas las ejecuciones en cada iteración (Fila 12).