Mecanismos de Atención e Interpretabilidad en el Aprendizaje Automático para la Detección de Enfermedades Oculares a través del Uso de Tomografías

David Camilo Alba Acosta *
Universidad Santo Tomás
davidalba@usantotomas.edu.co

Robert Gil Romero †
Universidad Santo Tomás
gilromero@usantotomas.edu.co

2 de noviembre de 2020

Resumen

El Deep learning o "aprendizaje profundo" vino para quedarse por mucho tiempo; es así que, en los últimos años producto del desarrollo de la inteligencia artificial, su actuar ha venido aplicándose en varias áreas del conocimiento principalmente en las ciencias de la salud. Ante este fenómeno señalado, la medicina tradicionalmente ha venido desarrollando los diagnósticos de enfermedades como de traumatologías, incorporando el uso de rayos x, resonancias magnéticas y tomografías entre otras. En ese sentido, la ciencia de la medicina ha venido implementando sistemas autónomos, los cuales, han venido a complementar la forma de analizar la información médica.

La ciencia médica hoy en día viene desarrollando en sus diferentes áreas del conocimiento la aplicación de la inteligencia artificial mediante el uso del deep learning, como es el caso de la oftalmología y de la optometría. De conformidad con lo explicitado, el trabajo propuesto tiene como objetivo principal: Encontrar la mejor arquitectura CNN "Convolutional neural network" que se adapte a un conjunto de datos que contiene tomografías oculares en la región de la retina, y que sea capaz de clasificar estos cuatro tipos de diagnóstico ocular: Normal, Drusen, CNV (Neurovascularización coroidea) y DME(Edema Macular Diabético).

Palabras Claves— Deep Learning, Neural networks, CNN, Módulos de atención, enfermedades oculares.

1. Introducción

En el ejercicio de la medicina los profesionales de la salud requieren tomar decisiones oportunas y rápidas para la detección de cualquier tipo de enfermedad, que muchas veces, son tomadas de acuerdo con el conocimiento adquirido, o desarrollado a través del tiempo por parte del profesional. En ese contexto, el profesional de la salud o el médico cuando se enfrenta a situaciones de alta complejidad para la detección de algún tipo de enfermedad acude al consenso de sus pares profesionales para poder efectuar un diagnóstico acertivo que pueda dar respuesta a un tipo de enfermedad. Conforme a lo anterior ha surgido dentro del diagnóstico en la medicina contemporánea los diagnósticos de la salud mediante la incorporación de nuevos desarrollos vía inteligencia artificial, que es uno de los frutos de la cuarta revolución industrial. La incorporación del ejercicio de la inteligencia artificial en el área de la salud ha sido de gran beneficio y utilidad en el desarrollo de la medicina, ya que ha incorporado un mejor análisis del ejercicio profesional mediante la identificación de procesos, rutinas, e incluso competencias en el diagnóstico y tratamiento de enfermedades(Jaiswal and Jaiswal, 2019).

Las ventajas en el desarrollo de la inteligencia artificial son muchas entre las que se destacan, por ejemplo, la prescripción de medicamentos, obtención de diagnósticos ágiles y oportunos en pruebas de laboratorio, que han permitido la reducción de costos clínicos. Sin embargo, a pesar de que estos procesos mejoran el análisis de predicción de enfermedades, aún continua siendo un reto para la ciencia el disminuir los índices de morbilidad y mortalidad asociadas a enfermedades de alto costo, como lo son el cáncer en sus diferentes modalidades (óseo, hepático, cerebral de mama, etc) y los pacientes renales crónicos, los cuales en los últimos años ha venido creciendo en forma exponencial, y generan grandes déficit en el sistema de salud(Jaiswal and Jaiswal, 2019).

De otro lado, a pesar de los grandes avances que se han dado en los últimos años, todavía existe un punto de quiebre entre la lectura médica hecha por un profesional de la salud en yuxtaposición con el análisis a través de los sistemas de inteligencia artificial; no obstante, a pesar que aún existe resistencia por algunos profesionales de la salud a este tipo de desarrollo, lo que es cierto, es que cada día se viene incorporando los sistemas de inteligencia artificial como herramientas fundamentales para identificar y predecir enfermedades como también traumatologías(Jaiswal and Jaiswal, 2019).

En ese orden de ideas, el avance tecnológico ha venido contribuyendo de forma significativa a las ciencias de la salud, mejorando los procesos de lectura médica en el diagnóstico e identificación de enfermedades. Historicamente las ciencias de la salud venían utilizando la lectura rayos x, resonancias magnéticas y tomografías computari-

*Estudiante de Estadística
†Tutor de tesis, docente de la Facultad de Estadística

zadas como instrumento de diagnóstico en enfermedades y traumatologías; sin embargo, con la aparición de los sistemas de inteligencia artificial vía Deep Learning "aprendizaje profundo", se ha podido encontrar que mediante este mecanismo, los diagnósticos han mejorado sustancialmente, contribuyendo en la mejor toma de decisiones por parte del profesional de la salud. El aprendizaje profundo en su desarrollo o ejecución requiere como insumos: grandes volúmenes de imágenes que corresponden a fotos de rayos x, resonancias magnéticas y tomografías computarizadas. La incorporación de imágenes en el aprendizaje profundo se realiza mediante el tratamiento de los pixeles que arrojan las fotos, las cuales, son modeladas a través de las capas ocultas de una red neuronal(Razzak et al., 2018).

Cabe anotar que, con la intensidad de las imágenes es decir, el cambio en el flujo de luz, se convierte,en el aprendizaje profundo, en datos de entrada que son procesados a través de capas de neuronas ocultas. Estas capas identifican patrones como pueden ser : bordes, líneas, bucles(Razzak et al., 2018).

En la actualidad el deep learning(aprendizaje profundo) ha hecho que el tema del diagnóstico de las enfermedades sea más eficaz, empleando un análisis estructural de los patrones que constituyen el área de interés de la imagen, para la correcta clasificación de la enfermedad. Dentro del deep learning se encuentra la arquitectura Convolutional Neural Network(CNN), la cual ha adquirido últimamente protagonismo, en cuanto al excelente desempeño para detectar patrones complejos en donde se haya la característica de interés, en imágenes de dos dimensiones. La arquitectura CNN implementa varios filtros a la imagen, con el fin de extraer patrones relevantes a través de la reducción de dimensionalidad(Alom et al., 2019).

En el caso de las tomografías oculares existen estudios como Kermany et al. (2018), quienes encontaron evidencia, que el uso del transfer Learning, es una metodología adecuada para la clasificación de enfermedades oculares; aunque esta técnica, es bastante éficaz en el aumento de la capacidad predictiva, todavía se encuentran nuevas arquitecturas que han revolucionado la clasificación de imágenes con la implementación de módulos de atención(Sarkar, 2018).

De acuerdo con lo descrito anteriormente, en el proceso investigativo, se realizó una adaptación de diferentes arquitecturas(CNN) a los datos de entrenamiento, con el propósito de clasificar de forma conjunta los tipos de clases expuestos en la base de datos ZhangLabData (2018). Asimismo, se planteó en este ejercicio, el uso de metodologías como los módulos de atención para el aumento de la capacidad predictiva del modelo.

Por otro lado, en el desarrollo investigativo surgió la importancia de dar interpretabilidad a los resultados arrojados por el modelo; para tal efecto, se logró con el algoritmo LIME, analizar de manera local las regiones de la imagen que proporcionaron más información de la clase de diagnóstico ocular que se deseo predecir; Por otra parte, se realizó un análisis comparativo entre las arquitecturas(CNN) y otras metodologías estadísticas (Support Vector Machines).

A continuación se hace un análisis conceptual de los principales tópicos que se utilizaron en la investigación.

2. Análisis Conceptual

2.1. Machine Learning

El Machine Learning(Aprendizaje automático), nació fruto de la investigación en la construcción de modelos de neuronas artificiales basado en la adaptación de la neurociencia a los sistemas computacionales. Las neuronas artificiales, simulan el funcionamiento del cerebro, en donde existen en las redes nodos capaces de recibir información, desde fuentes de tipo externo, para el tratamiento y transformación de ésta en una salida, que denota la respuesta del flujo de la información (Goodfellow et al., 2016a).

Los neuronas o percetrons, consisten en capas de nodos conectados de manera conjunta, que tienen como objetivo: transmitir el flujo de información desde las entradas hasta las salidas, formando lo que se denomina: Neural Network(NN). Estos nodos, se activan mediante pesos o funciones de transferencia, que permiten el procesamiento de la información. Dichos pesos son ajustados mediante el proceso de entrenamiento de las Neural Networks(NN); de tal forma, que se minímice la diferencia entre el resultado derivado del proceso, y el resultado real(Ravì et al., 2016).

2.2. Tipos de Aprendizaje

Existen 4 tipos de aprendizaje que utiliza el machine Learning en la etapa de entrenamiento para lograr una buena capacidad predictiva, los cuales son(Alom et al., 2019):

- Aprendizaje supervisado: Técnica que necesita que los datos tengan una especie de etiqueta, con el propósito que la máquina adquiera conocimiento del tipo de objeto que va a clasificar.
- Aprendizaje semi-supervisado: Técnica en la que se necesita, que parte de los datos estén etiquetados para su correcta clasificación.
- Aprendizaje no supervisado: Técnica en la cual no se necesita, que los datos estén etiquetados, ya que, la máquina tiene la experticia necesaria en la clasificación de objetos.
- Aprendizaje reforzado o Reinforcement learning: Técnica en la que la máquina aprende por medio de un ambiente con el fin de adquirir, a través de la interacción, el conocimiento necesario para clasificar la información.

2.3. Deep learning

Con base a las Neural Networks(NN), la adición de más capas de nodos a la red es lo que constituye las Deep Neural Networks(DNN). Estas redes poseen la habilidad de procesar datos de carácter multidimensional, identificando las características comunes para su correcta clasificación(Alom et al., 2019).

El multilayer perceptron, son modelos por excelencia de las Deep Neural Networks(DNN), los cuales, aproximan una función de salida de acuerdo con los valores de entrada junto con los parámetros óptimos. La información fluye a lo largo de la red, realizando cálculos intermedios para estimar la respuesta en la salida de la red. Los procedimientos de entrenamientos de éstas redes especifican, cuál debe ser resultado que debe arrojar

la última capa de la red, el cual, debe ser cercano al valor real(Goodfellow et al., 2016a).

El algoritmo del proceso de transferencia, de la información se da de la siguiente forma:

- X = Entrada de la información.
- $f(x)$ = Procesamiento.
- $f(x)^*$ = Estimación, salida de la información.
- y = Valor objetivo.

Comparación entre f(x)* = y

$$X \rightarrow f(x) \rightarrow f(x)^* \approx y \tag{1}$$

Los valores de entrada de la red estan dados de la siguiente forma:

- $x(input)$ = Valor de la intensidad del pixel de la imágen.
- $w(peso)$ = Valor de la importancia asociada a la intensidad del pixel.
- $b(bias)$ = Número de penitencia que activa y desactiva neuronas.

El procesamiento de la información esta dado por la siguiente operación:

$$f_{(w,x,b)} = wx + b \tag{2}$$

Con ayuda de una función de activación , el valor arrojado en el procesamiento de la información de los datos de entrada es transformado a una estimación del valor objetivo.

$F(x)$ = Función de activación.

$$F(f_{(w,x,b)}) = f(x)^*$$

$$f(x)^* \approx y \tag{3}$$

2.3.1. Estructura

La capa de neuronas que se encuentra en el extremo izquierdo de la red se le denomina la capa de entrada de los datos. En contraposición, la capa en el extremo derecho de la red neuronal, se le atribuye, el nombre de capa de salida de datos. La capa intermedia, que se encuentra en medio de la capa de entrada, y la capa de salida recibe el nombre de capa oculta(Nielsen, 2015).

El flujo de información dentro de las capas de la red neuronal va en un sentido izquierda a derecha, en donde hay una entrada, un proceso y una salida de la información. El número de neuronas en la capa de entrada de los datos esta determinado por, la dimensión que tiene la imagen en término de sus pixeles; de igual manera, el número de neuronas en la capa de salida de los datos, está determinado por el número de clases que se desean predecir(Nielsen, 2015).

Cabe resaltar, que a pesar, de que no existe una regla específica en cuanto a cuál debe ser: el número de neuronas por capa oculta de la red neuronal, existen diferentes tipos de diseños desarrollados por investigadores, que permiten obtener un buen desempeño en el entrenamiento; por ende, se obtiene una buena clasificación de la información. Por otro lado, la salida de información de una neurona se vuelve la entrada de información para otra, lo que significa, que hay un proceso de transmisión de la información en el sentido hacia adelante(Nielsen, 2015).

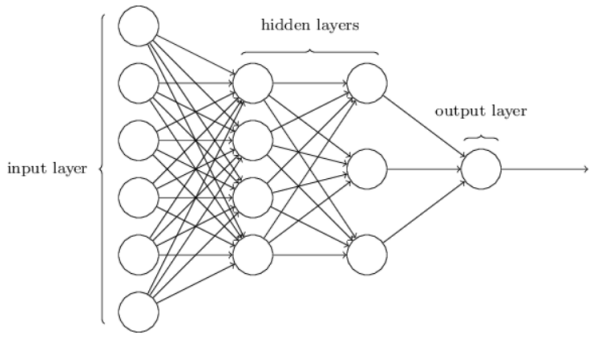


Figura 1: Deep neuronal network, fuente (Nielsen, 2015)

2.3.2. Neuronal Networks

- **Perceptron**

  El concepto de perceptron dentro de una Neural Network(NN), parte de la relación que este posee con la célula del sistema nervioso central, denominada neurona. Las neuronas son unidades celulares que tienen la función de recibir, almacenar y transmitir información a través de pulsos nerviosos dentro del cerebro. Cada neurona tiene la capacidad de activarse o inhibirse en el proceso de la sinapsis, en donde las neuronas, se conectan entre sí, con el fin de procesar la información por medio del impulso nervioso. Así como la neurona tiene la capacidad de procesar información, el perceptron a través de una función de activación, emula la función que hace una neurona dentro del cerebro humano(Ravì et al., 2016).

  De otro lado, el perceptron o nodo realiza operaciones de multiplicación y suma con los pesos y biases para la toma de decisiones por medio de una salida . La salida se hace de acuerdo con una función de activación que transforma los resultados cálculados en outputs binarios(Nielsen, 2015).

  El resultado de la operación del perceptron esta dada por la siguiente función de activación:

  - w = Peso.
  - b = Bias.
  - x = Intensidad del píxel de la imagen o valor de entrada.

$$Salida(w,x,b) = \begin{cases} 0 \quad si \quad wx + b \leq 0 \\ 1 \quad si \quad wx + b > 0 \end{cases} \tag{4}$$

De acuerdo con la fórmula anterior se puede decir:

- Valores positivos de bias harán que la salida de la información sea igual a 1.
- Valores negativos de bias harán que la salida de la información sea igual a 0.

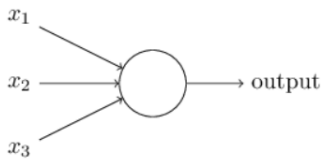


Figura 2: Perceptron (Nielsen, 2015)

Funciones de Activación

Sigmoide: De acuerdo con la salida del perceptron, la salida sigmoide propone realizar un pequeño cambio en el valor de la salida. Este pequeño cambio en la salida, hace que exista de la misma forma, pequeños cambios en los pesos y los biases de la red; por lo tanto, se plantea una función de activación que generé una salida de valores en el intervalo 0 y 1, dejando atrás la salida binaria propuesta por el perceptron(Nielsen, 2015).

La función de activación está dada de la siguiente forma:

$$z = wx + b$$

$$Salida(w, x, b) = \sigma_{(z)}$$

$$\sigma_{(z)} = \frac{1}{1 + e^{-z}} \tag{5}$$

$$0 \leq Salida(w, x, b) \leq 1$$

De acuerdo con, la fórmula anterior se pueden hacer los siguientes planteamientos:

- Si z toma valores grandes positivos, la salida de la función va tender a ser igual a 1.

$$\lim_{z \to \infty +} \sigma(x) = 1 \tag{6}$$

- Si z toma valores grandes negativos, la salida de la función va tender a ser igual a 0.

$$\lim_{z \to \infty -} \sigma(x) = 0 \tag{7}$$

De acuerdo con lo anterior, se puede evidenciar la aproximación que tiene la función sigmoide a la función de salida del perceptron.

ReLU: La función de activación ReLU realiza un proceso de transformación a las entradas de los nodos de la red neuronal de la siguiente manera (Goodfellow et al., 2016b):

$$z = wx + b$$

$$ReLU(z) = max(0, z) \begin{cases} 0 & si \quad z < 0 \\ z & si \quad z \geq 0 \end{cases} \tag{8}$$

De acuerdo con la fórmula anterior, se puede afirmar:

- Todo valor de z negativo va a ser igual a "0"
- Todo valor de z positivo mayor o igual a "0" va a tomar el valor de z.

Softmax: Es una función de activación que estima la distribución de probabilidad de las distintas clases que se desean predecir(Denker and LeCun, 1991). La función cumple con los dos principales axiomas de probabilidad, los cuales son:

- La probabilidad debe ser positiva en un rango de 0 a 1.
- La suma de las probabilidades de las clases debe ser igual a 1.

Las principales ventajas que se tiene al aplicar esta función de activación son las siguientes:

- Es fácil tratar con números que no tienen un comportamiento estandarizado, simétrico con respecto al valor 0.
- Las ganancias calculadas debido a la aplicación de las exponenciales dependen de la categoría a clasificar.
- Las ganancias pueden ser perfectamente calculadas de acuerdo con los valores arrojados por la red neuronal.

La función softmax esta dada por la siguiente fórmula:

$$Softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{9}$$

Se puede observar la relación que existe entre una de las categorías con referencia a las demás de manera implícita.

2.3.3. Optimización y aprendizaje

- **Función de Costo**

La función de costo tiene como finalidad encontrar los pesos y biases óptimos que permitan disminuir la distancia entre el valor estimado y real de acuerdo con el procesamiento de los datos de entrenamiento en la red. Este proceso, va acompañado del algoritmo de gradiente descendente(SGD), el cual calcula la caída de la función de costo en dirección al mínimo local(Nielsen, 2015).

Error Cuadrático Medio(MSE): Con base en la función de costo de los modelos parámetricos, se puede construir de la misma forma para los modelos de redes neuronales de la siguiente manera(Nielsen, 2015):

- w = Peso.
- b = Bias.
- n = Número total de datos de entrenamiento.
- a = Vector de salida de los resultados arrojados.
- $Y(x)$ = Vector de valores reales.

$$C(w, b) = \frac{1}{2n} \sum_x \|Y(x) - a\|^2 \tag{10}$$

De acuerdo a la fórmula anterior se puede afirmar:

- Si $C(w, b) \approx 0$, se dice que el valor calculado en la red neuronal es aproximadamente igual al valor objetivo o real. Lo anterior, quiere decir que, el entrenamiento de la red ha sido un éxito; por lo tanto, la red neuronal se dice que está "aprendiendo".

- Si $C(w, b)$ es grande, se dice que el valor calculado está alejado de ser aproximadamente igual al valor objetivo o real. Lo anterior, quiere decir que, posiblemente los valores iniciales dados en la entrada de la red ,no fueron los adecuados; por lo tanto, la red no muestra estar "aprendiendo".

Cross Entropy: Es una medida que determina la diferencia entre el vector de probabilidades, dadas por la función softmax, y el vector de etiquetas binario, en donde 1 indica la existencia de cierta categoría y 0 indica lo contrario(Goodfellow et al., 2016a).

La fórmula de cross entropy está dada de la siguiente manera:

- s_i = Vector de probabilidades dado por la función Softmax.
- l_i = Vector de etiquetas.

$$CE(s, l) = -\sum_i l_i \log(s_i) \tag{11}$$

Se puede decir, que el resultado, que se obtiene, a través de esta función, no es simétrico, debido a la existencia de un logaritmo. También, se puede afirmar que el logaritmo dentro de la función nunca tomará el valor de cero, ya que, la función softmax garantiza que cada clase tenga su propia probabilidad mayor a cero.

- **Tasa de Aprendizaje**

Se puede decir que la tasa de aprendizaje son los saltos que se dan dentro de la grilla de parámetros, con el objetivo de encontrar los parámetros óptimos que mínimicen la función de costo. Esta tasa, hace que el proceso de entrenamiento de la red sea eficaz en la convergencia hacia el mínimo de la función de costo(Alom et al., 2019).

$$\eta = \text{Tasa de Aprendizaje}$$
$$\eta > 0$$

La tasa de aprendizaje debe ser reducida para lograr una buena convergencia; sin embargo, la tasa debe procurar no ser tan pequeña para no caer en la divergencia.

- **Algoritmos de aprendizaje**

Stochastic Gradient Descent(SGD)

Es un algoritmo que tiene como objetivo hacer que la red neuronal aprenda a identificar las representaciones de las imágenes a través de un proceso iterativo en la disminución del valor de la función de costo. Este proceso, utiliza el gradiente con dirección negativa para hallar los parámetros óptimos que mínimicen la distancia entre el resultado estimado y el resultado que se desea obtener(Nielsen, 2015).

El algoritmo plantea las derivadas de la función de costo con respecto a los parámetros(pesos y biases) con el fin de saber cuales cambios deben aplicarse como punta de partida con dirección hacia el mínimo de la función.

El gradiente esta representado por las derivadas parciales de la función de costo y los parámetros de la red neuronal; como se puede denotar en la siguiente expresión:

$$w = \text{Peso}$$
$$b = \text{Bias}$$

$$\nabla C = \left(\frac{\partial C}{\partial w}, \frac{\partial C}{\partial b}\right)^T$$

El proceso iterativo para encontrar los valores óptimos con ayuda del gradiente descendente esta dado por las siguientes exprésiones:

$$w_i = w_i^{(n-1)} - \eta \frac{\partial C}{\partial w}$$

$$b_i = b_i^{(n-1)} - \eta \frac{\partial C}{\partial b}$$

Con base a las anteriores fórmulas se llega a la siguiente generalización:

$$\nu = w, b$$

$$\Delta\nu = (\Delta\nu_1, \ldots, \Delta\nu_n)^T$$

$$\nabla C = \left(\frac{\partial C}{\partial \nu_1}, \ldots, \frac{\partial C}{\partial \nu_n}\right)^T$$

$$\Delta\nu = -\eta \nabla C \tag{12}$$

$$\nu = \nu' - \eta \nabla C \tag{13}$$

Como resultado, se obtiene que, el cambio de los parámetros de la red está en función de la dirección de los ajustes de la función de costo, con saltos que debe hacer el algoritmo para encontrar el mínimo.

SGD con momentum

A la hora de hallar el mínimo óptimo, el stochastic gadrient descent(SGD) puede llegar a presentar problemas de convergencia. De acuerdo con lo anterior, se plantea el momentum(γ) como método de aceleración en la convergencia de SGD, logrando la reducción en la actualización de los parámetros en el proceso de optimización (Ruder, 2016).

El SGD con momentum esta dado por la siguiente expresión:

$$v_t = \gamma v_{t-1} + \eta \nabla C \tag{14}$$
$$\theta = \theta - v_t \tag{15}$$

El hiperparametro γ suele ser un valor cercano a 0.9. Gracias a este hiperparametro, los gradientes calculados apuntan a una misma dirección, dismuyendo la oscilación que debe hacer el algoritmo hasta el punto óptimo.

ADAM

Este método de optimización sigue la metodología del algoritmo de gradiente descendente estocástico(SGD), calculando los primeros y segundos momentos de los gradientes, para la adaptación de las tasas de aprendizaje de los diferentes parámetros de la red neuronal(Kingma and Ba, 2014). Este tipo de optimización posee las siguientes ventajas :

- Requiere poca memoria en cuanto toma los gradientes de primer orden.

- No requiere de un parámetro estacionario objetivo . Esto se puede observar con la tasa de aprendizaje , la cual cambia de acuerdo a la dirección del gradiente.
- El cambio que se da en la escala del gradiente no hace que varíen los demás parámetros.

★ **Algoritmo**

- ◇ Valores de entrada:
  - ◇ α = El tamaño del paso inicial.
  - ◇ $\beta_1, \beta_2 \in [0, 1)$ = Las tasas de decrecimiento exponencial para las estimaciones de momentos de los gradientes.
  - ◇ $f(\theta)$ = la función estocástica objetivo o función de costo con respecto a los parámetros el vector inicial de parámetros θ.
  - ◇ θ_0 = El vector inicial de parámetros.
  - ◇ m_0 = Media móvil exponencial 1º momento con valor de "0".
  - ◇ v_0 = Media móvil exponencial 2º momento con valor de "0".
  - ◇ t = Tiempo iniciado en "0".
  - ◇ ϵ= Nivel de tolerancia.
- ◇ Proceso Iterativo:
  - ◇ $g_t = \nabla_\theta f_t(\theta_{t-1})$, los gradientes en el tiempo.
  - ◇ $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$, actualizar la estimación sesgada el primer momento.
  - ◇ $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$, actualizar la estimación sesgada del segundo momento.
  - ◇ $\hat{m}_t = \frac{m_t}{(1-\beta_1^t)}$, el cómputo de la estimación del primer momento con corrección.
  - ◇ $\hat{v}_t = \frac{v_t}{(1-\beta_2^t)}$, el cómputo de la estimación segundo momento con corrección.
- ◇ Valor de salida:
  - ◇ $\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{(\sqrt{\hat{v}_t}+\epsilon)}$, actualizar parámetros.

De acuerdo con el algoritmo es importante tener en cuenta las siguientes afirmaciones:

- Dado que m y v inicializan sus valores en cero, es necesario corregir el sesgo que se genera con esta opción. Para esto se divide la estimación sesgada sobre 1 menos tasa. Esto permite tener una escala de los pasos hacia el óptimo más controlada, evitando estimar a pasos demasiado grandes.
- El tamaño de paso inicial α establece el límite de la magnitud de los pasos en el espacio de parámetros.
- Cuando el ratio $(\hat{m}_t/\sqrt{\hat{v}_t})$ tiende a ser más pequeño, el tamaño de los pasos tenderá a estar en la misma sintonía, cerca del valor óptimo.

▪ **Técnicas para evitar el sobreajuste**

Regularización L1 Y L2

Las regularizaciones son estrátegias que se usan para evitar el sobreajuste de los datos en la etapa de entrenamiento. Estas constan de coeficientes de penalización en la función de perdida de la red neuronal(Goodfellow et al., 2016a). De acuerdo con lo anterior se proponen dos tipos de regularizaciones inspiradas en los modelos de regresión Ridge y Lasso; las cuales son:

- L1: Basado en el modelo Ridge, se le añade un coeficiente elevado al cuadrado a la función de perdida.
- L2: Basado en el modelo Lasso, se le añade un coeficiente en valor absoluto a la función de pérdida.

Dropout

El dropout consiste en apagar las activaciones de las neuronas de la red, agregando un valor de cero en estas de forma aleatoria. Esto se hace con el objetivo, de agilizar el cálculo de los valores óptimos que mínimicen la función de pérdida de la red neuronal(Goodfellow et al., 2016a).

Batch Normalization

El problema de tener muchas distribuciones asociadas a los mapas de características, producto de la aplicación de filtros a la imagen, es que a medida que pasan por las capas de la red neuronal, hay una constante transformación en la distribución de los pixeles que estos poseen. Lo anterior, ocasiona que en el algoritmo de gradiente descendente, se tengan tasas de aprendizaje(η) demasiado bajas; por ende, el algoritmo no logra convergencia en la búsqueda de los parámetros óptimos. Esta problemática tiende a agravarse cuando la red neuronal es más profunda, generando una constante adaptación y cambios de los parámetros de la red.
Con base a esta problemática, se propone el método de bach-normalization, el cual, utiliza la normalización de las muestras de los mapas de caracteristicas en el proceso de entrenamiento, para contar con altas tasas de aprendizaje que permitan encontrar los parámetros óptimos que minimicen la función de costo. Este método reduce la dependencia de los gradientes hacia la distribución de los parámetros de la red, asimismo, de los valores iniciales dados al principio del entrenamiento(Ioffe and Szegedy, 2015).
Bach-normalization tiene como objetivo mejorar el tiempo de entrenamiento de la red, reduciendo el cambio interno de la distribución de los píxeles de las imágenes. De acuerdo con lo anterior, es importante asegurar que cada parámetro de la red siempre se produzca la distribución esperada de los datos; es así que, se debe permitir que los parámetros en el gradiente con respecto a la función de pérdida tenga en cuenta la normalización para lograr una misma distribución de los parámetros(Ioffe and Szegedy, 2015).
El proceso de normalización consta de escalar la información(x) de manera independiente, teniendo en cuenta que la varianza de las entradas, de los mapas de características, debe ser igual a “1” y su media respectiva debe ser igual a “0”.

$$\hat{x} \sim N(0,1) \tag{16}$$

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \tag{17}$$

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{18}$$

$$\sigma_B = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{19}$$

- Normalización:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{20}$$

- Transformación lineal:

$$y_i = \gamma \hat{x}_i + \beta \tag{21}$$

$$y_i = BN_{\gamma,\beta}(x_i) \tag{22}$$

Cabe resaltar que una simple normalización de una capa de la red puede cambiar el flujo de información que se transmite; es por ello, que se hace una transformación lineal con los parámetros γ y β para guardar la identidad de las capas y el flujo de información.

2.3.4. Convolutional Neural Network(CNN)

Es un tipo de red neuronal profunda que se especializa en procesar información que tiene una estructura de cuadrícula como lo es una imagen de dos dimensiones. El origen del nombre que recibe la red proviene de la operación matemática "convolución", la cual, se le aplica a la imagen por medio de filtros. Estos filtros extraen características de la imagen, a través del agrupamiento de píxeles; método que reduce la dimensionalidad de la imagen, obteniendo representaciones menos complejas de procesar en las capas ocultas de la red neuronal(Goodfellow et al., 2016a).

- **Operaciones**

  Filtro: Los filtros son matrices con pesos que son multiplicados por la intensidad de los píxeles de una imagen, con el fin de obtener, a través de una operación lineal, los bordes de los objetos a clasificar.

  Padding: El padding o relleno, consiste en agregar ceros alrededor de la imagen para la adecuada aplicación de filtros que puedan identificar los bordes que componen la imagen.

  Stride: El stride o zancada, es el número de desplazamiento de píxeles que se realiza sobre la matriz de entrada de información; por ejemplo, cuando se habla de una zancada de 1 píxel, da entender la afirmación que los filtros se mueven a lo largo de los datos de a 1 píxel por vez.

  Pooling: El pooling o agrupamiento, radica en la aplicación de filtros con el fin de sintetizar la información, teniendo como resultado la reducción de dimensionalidad de la imagen. Existen dos tipos de agrupamiento; uno de ellos corresponde al promedio de los elementos, y el otro reside en encontrar el número más grande o máximo de los elementos por filtro.

- **Arquitecturas**

  La arquitectura de una CNN se compone de dos aspectos importantes; uno de ellos corresponde a la forma en cómo se extraen los patrones de la imagen; y por otro lado, la manera en cómo se clasifica la imagen de acuerdo con los patrones extraídos. De igual forma, la arquitectura de la CNN posee tres tipos de capas, los cuales son: Convolución, agrupamiento y clasificación.

  A medida que la información es procesada en las capas de la red, la dimensión de la imagen se va reduciendo con la aplicación de operadores de agrupamiento y convolución. Por otra parte, en la capa de clasificación se utiliza aproximaciones de distribuciones de probabilidad, con el objetivo de asociar el valor más alto de salida de información de la red con el valor más alto de probabilidad conforme a lo arrojado también en las otras salidas de la capa.

- **DenseNet**

  Es una arquitectura convolucional que tiene como objetivo mejorar al máximo el flujo de información dentro de las capas de la red, proponiendo una forma de conexión directa entre las capas inciales y las capas posteriores. Esta conexión, se basa en que cada capa de la red recibe de entrada los mapas de características de las capas anteriores; lo anterior quiere decir, que hay un proceso de unión entre los mapas de caracteristicas(concatenación), mejorando la representación de las imágenes dentro de la red(Huang et al., 2017).

  La estructura de la DenseNet esta compuesta de sub-bloques, los cuales, en su interior realizan operaciones de convolución y de agrupamiento. Dentro de estos sub-bloques, se aplica el método de batch-normalization, que consta de normalizar los mapas de características; asimismo, el uso de la función de activación ReLU entre las capas (Huang et al., 2017).

  Es importante mencionar, que en el proceso de concatenación en cada sub-bloque, no se realiza reducción de dimensionalidad, ya que, en la misma concatenación, se necesita que los mapas posean una misma dimensión; sin embargo, en la transición entre los bloques, si se puede aplicar la operación de convolución (Huang et al., 2017).

  Una vez procesada la información entre los sub-bloques, se realiza el proceso de compresión para que la información sea reducida hasta llegar a la última capa de la red que corresponde a la clasificación.

- **Modulos de Atención CBAM**

  Son bloques de módulos de atención convolucional que tienen como objetivo refinar los mapas de características, dados en el proceso de convolución. Estos módulos, aumentan la representación de las imágenes dentro de la red, obteniendo mejores resultados en la clasificación(Woo et al., 2018). CBAM posee dos submódulos, los cuales son :

  - Módulo de atención por canal.
  - Módulo de atención por espacio.

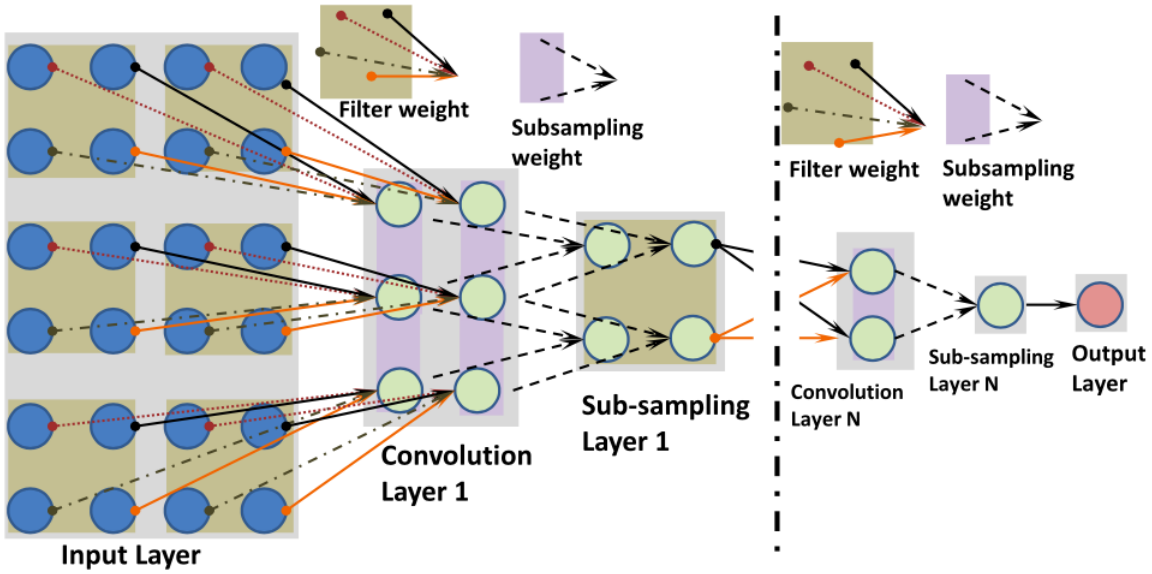


Figura 3: Convolutional neural network, fuente(Ravì et al., 2016).

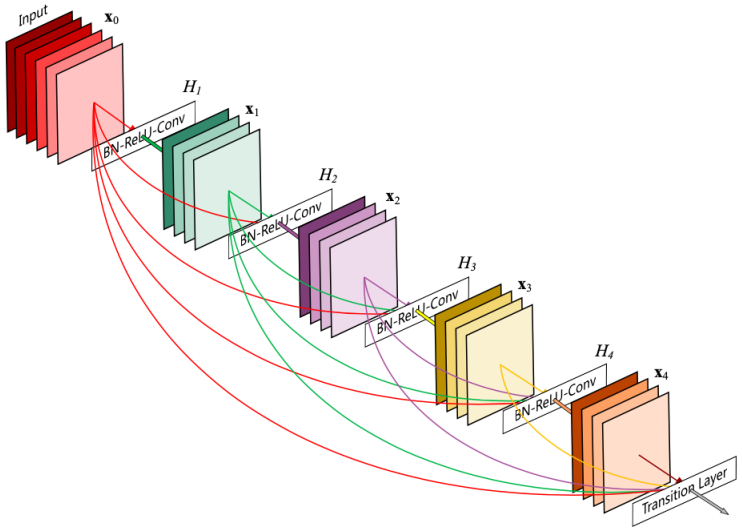


Figura 4: Estructura DesneNet, fuente(Huang et al., 2017).

Módulo de atención por canal: Este módulo explota las diferentes características que poseen los canales de los mapas, en donde se utiliza agrupaciones por el máximo o por el promedio, para tener una mejor representación (Woo et al., 2018).

- $M_c \in \mathbb{R}^{C\times 1\times 1}$= La dimensión del módulo corresponde a un ancho y un largo de 1 píxel, con c-canales.
- F^C_{avg} = Agrupamiento por el promedio por cada canal
- F^C_{max}= Agrupamiento por el máximo por cada canal
- σ = Función de activación sigmoide
- MLP = Multi-layer perceptron con solo una capa oculta.

★ Operación:

$$M_c(F) = \sigma[MlP(AvgPool(F)) + MLP(MaxPool(F))] \tag{23}$$

$$= \sigma[W_1(W_0(F^C_{avg})) + W_1(W_0(F^C_{max}))]$$

$$W_0 \in \mathbb{R}^{\frac{C}{r}\times C}$$

$$W_1 \in \mathbb{R}^{C\times \frac{C}{r}}$$

W_0 y W_1 corresponden a los pesos que componen la MLP. Mientras que r corresponde al ratio de reducción.

Módulo de atención por espacio: Este módulo se enfoca en qué áreas del mapa de características se presenta la mayor información. Se generan dos mapas de dos dimensiones que hacen de filtros por agrupamiento del máximo o del promedio (Woo et al., 2018).

- H = Representa el largo del mapa de características.
- W = Representa el ancho del mapa de características.
- $M_s(F) \in \mathbb{R}^{H\times W}$ = La dimensión del módulo que corresponde a un canal por una dimensión $H \times W$.

★ Operación:

$$M_s(F) = \sigma(f^{n\times n}[AvgPool(F), MaxPool(F)]) \tag{24}$$

$$= \sigma(f^{n\times n}[F_{avg}, F_{max}])$$

$f^{n\times n}$ representa la dimensión del filtro de $n \times n$ dimensiones.

- **Bloque Squeeze and Exitation**

El objetivo de este bloque, es modelar las relaciones que existen entre los canales del mapa de características de la imagen; se puede decir que es un tipo de módulo por canal que analiza y aborda la problemática acerca de la extracción de las características de los canales de una imagen.

Pre-paso: Existe una previa transformación que se le hace a la imagen original, a través de los filtros convolucionales, teniendo como resultado un mapa de características (Hu et al., 2018).

- $F_{tr} : X \to U$, proceso de transformación.
- $X \in \mathbb{R}^{H'\times W'\times C'}$, la imagen original.
- $U \in \mathbb{R}^{H\times W\times C}$, la imagen procesada o mapas de características.

Los mapas de características fueron producto de la operación convolución a la imagen original. Cada canal de U, que es una colección de mapas de características, fue generado de acuerdo a la siguiente expresión:

$$u_c = v_c * X = \sum_{s=1}^{C'} v^s_c * x^s \tag{25}$$

- $U = [u_1, u_2, \ldots, u_c]$: Salidas de la operación.
- $V = [v_1, v_2, \ldots, v_c]$: Filtros con kernel, c-ésimo filtro.
- $X = [x^1, x^2, \ldots, x^{c'}]$: Canales de la imagen.
- v^s_c : Es el parámetro del filtro que se multiplica por el valor del mapa de características del canal(c).

1º Paso: Exprimir(Squeeze)

Una forma de aprovechar la información de todo un canal es reuniendo las características que este tiene a través del uso del promedio. Este método, permite ver de una forma general la información del canal en comparación con los filtros convolucionales, que extraen las características de una forma local (Hu et al., 2018). La operación del promedio se aplica de la siguiente manera:

$$z_c = Fsq(u_c) = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} u_c(i,j) \tag{26}$$

- H = Largo del mapa.
- W = Ancho del mapa.
- $u_c(i,j)$= Entrada por posición de U.
- z_c = Salida de la operación.

2º Paso: Calibración adaptativa(Exitation)

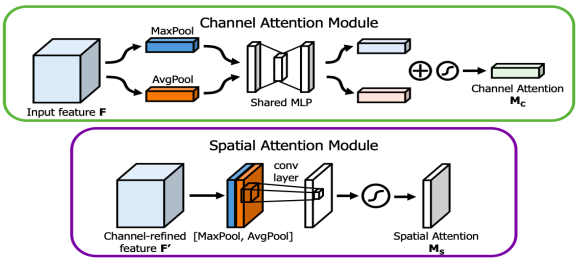
Después de haber aplicado la operación del promedio, se procede a calcular una segunda operación que permita capturar las relaciones entre los canales. Se desea aplicar una función a las salidas de la operación anterior que cumpla con los siguiente criterios (Hu et al., 2018):

- La función debe ser flexible; que sea capaz de aprender de la información a pesar de que está presente un comportamiento no lineal entre los canales.
- La función debe aprender de las relaciones que no son mutuamente excluyentes entre los canales.

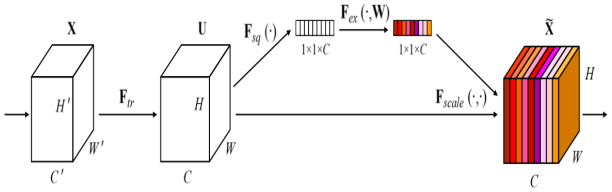
De acuerdo con lo anterior, se plantea una pequeña red neuronal(MLP) con una función de activación sigmoide:

$$S = F_{ex}(Z, W) = \sigma(g(Z, W)) \tag{27}$$

$$= \sigma(W_2\delta(W_1 Z))$$



(a) Módulos CBAM(Woo et al., 2018).



(b) Bloque Squeeze and Exitation(Hu et al., 2018).

Figura 5: Procedimiento que realizan los módulos de atención.

- δ = Función de activación ReLu.
- σ = Función de activación sigmoide.
- MlP = Multi-layer perceptron con una sola capa oculta.
- W_1 y $W2$ = Pesos de la MLP.

Hay una capa de MLP de reducción(r) de dimensionalidad con pesos:

$$W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$$

Luego le sigue una capa de aumento de dimensionalidad con pesos:

$$W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$$

3º Paso: Escalamiento

El resultado obtenido en el proceso de calibración adaptativa, en las salidas de la pequeña MLP, se multiplica con cada entrada de la colección de mapas U. Esto se puede expresar de la siguiente forma Hu et al. (2018):

$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c \tag{28}$$

- s_c = Escalar por canal, resultado de la operación de calibración adaptativa.
- u_c = Mapa por canal $\in \mathbb{R}^{H \times W}$
- $\tilde{x}_c$ = Mapa refinado por canal.

2.4. Interpretabilidad

Entender el resultado arrojado por una red neuronal no es fácil, ya que, dentro de sus capas ocultas se pierde el sentido de la información cuando se aplican varias operaciones matriciales. Al momento de analizar los parámetros de la red, carecen de explicación y entendimiento puesto que solo siguen el proceso de minimizar la función de costo.

Se puede afirmar que las redes neuronales son cajas negras, es por esto que, es necesario añadir en estos modelos la parte de interpretabilidad. La interpretabilidad es la habilidad de explicar en términos claros para la razón humana la esencia que caracteriza un elemento.

De acuerdo con lo anterior, se plantea el modelo de explicaciones locales de interpretación agnóstica (LIME), el cual busca explicar cómo afectan las características de la imagen a la predicción. A través del uso de la representación, LIME señala la región de píxeles(súperpixel) que contribuye a la probabilidad de la clase que se desea explicar (Ribeiro et al., 2016).

LIME hace los siguientes planteamientos:

- g = Modelo de interpretación de la clase a explicar. El dominio de g está entre 0 y 1.
- G = Conjunto de explicaciones tal que $g \in G$
- Z = Perturbación que toma valores (0, 1) en donde:
  - ⊛ "0" marca la ausencia del súperpixel al modelo.
  - ⊛ "1" marca la presencia del súperpixel al modelo.
- $\Omega(g)$= Medida de complejidad o parámetro de penalización que se utiliza para evitar el sobreajuste; se puede decir que, es el número de betas en el modelo g.
- $f(x)$ = La probabilidad de que el súperpixel(x) pertenezca a la clase que se desea explicar.
- $\pi_x(g)$= Es una medida de proximidad entre la imagen perturbada por segmentos Z y la imagen original por segmentos X. Esta medida oscila entre 0 y 1, teniendo como objetivo ser los pesos dentro del modelo de explicación.

$$\pi_x(z) = e^{-[\frac{D(x,z)}{\sigma}]^2} \tag{29}$$

El kernel gaussiano es utilizado para traducir la distancia de coseno a un valor entre 0 y 1, con el fin de obtener los pesos. La distancia de coseno es la distancia lineal que hay entre la imagen perturbada y la imagen original. Esta medida esta denotada por la siguiente expresión (Ribeiro et al., 2016):

$$\begin{aligned} D(x,z) &= 1 - \frac{x \cdot z}{\| x \|_2 \cdot \| z \|_2} \\ &= 1 - \frac{\sum_{i=1}^{n} x_i \times z_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} z_i^2}} \end{aligned} \tag{30}$$

El modelo ponderado en base a la perturbación esta denotado por la siguiente expresión:

$$g(z') = w_g z' \tag{31}$$

- wg = El peso asociado a la perturbación.
- z' = El segmento(súperpixel) asociado con la perturbación.

Cada beta β_g generado por la explicación se puede interpretar como la influencia del segmento(superpixel) de la imagen en la probabilidad de la clase a explicar $g(z')$. Con base a esto se puede afirmar:

- β_g con magnitud positiva(+), aumenta la probabilidad de la clase $g(z')$

Figura 6: Explicación del algoritmo LIME en la segmentación de las regiones que representan la clase a predecir. Teniendo al principio la imagen original y al costado derecho las otras imágenes que representan las siguientes clases: Guitarra electrica, guitarra acústica y labrador. Fuente(Ribeiro et al., 2016).

- β_g con magnitud negativa(-), disminuye la probabilidad de la clase $g(z')$

Para alcanzar una interpretabilidad local confiable, se debe minimizar la siguiente función de costo para el modelo de explicaciones $g(z')$ (Ribeiro et al., 2016):

$$\xi(x) = \underset{g \in G}{arg\ min}\ L(f, g, \pi_x) + \Omega(g) \tag{32}$$

en donde,

$$L(f, g, \pi_x) = \sum_{z, z' \in Z} \pi_x(z)(f(z) - g(z'))^2 \tag{33}$$

★ Pasos:

- Segmentar la imagen a explicar en regiones(súperpixeles).
- Generar valores aleatorios de "0" y "1".
- Multiplicar las regiones con los valores aleatorios generados, con el fin de conocer las regiones que estarán activadas y desactivadas dentro de la imagen.
- Teniendo la imagen perturbada se procede a calcular los pesos.
- Ajustar una regresión lineal ponderada con los pesos , las perturbaciones y las predicciones arrojadas por el modelo, teniendo como entrada las imágenes perturbadas.
- Tomar los valores de beta más altos con magnitud positiva de la regresión.
- Escoger las regiones que tuvieron los valores de beta más altos.

2.5. Metodologías Estadísticas de Ensamble

★ **Support Vector Machine**

Este método consiste en llevar los vectores de los datos, asociados a cada clase, a una función que realiza un proceso de transformación de la información hacia un espacio multidimensional. En ese espacio, se plantea un hiperplano de separación o superficie de decisión, el cual, tiene como objetivo dividir las clases para su correcta clasificación (Cortes and Vapnik, 1995).

Para la construcción, de un hiperplano que sea capaz de generar un límite decisión entras las clases, es necesario contar con los vectores de soporte. Estos vectores de soporte dibujan un margen de decisión con base en las clases, que permite disminuir el error en la clasificación y generalizan sobre el conjunto de datos (Cortes and Vapnik, 1995).

Con el fin de lograr un hiperplano que sea óptimo, es esencial que el margen entre los vectores de soporte debe procurar ser el máximo.

La fórmula del hiperplano óptimo en un espacio multidimensional de parámetros esta dada de la siguiente forma (Cortes and Vapnik, 1995):

$$\mathbf{w}_0 \cdot \mathbf{z} + b_0 = 0 \tag{34}$$

Donde $\mathbf{w}_0$ esta dado de la siguiente forma

$$\mathbf{w}_0 = \sum_{vectores\ de\ soporte} \alpha_i \mathbf{z}_i \tag{35}$$

- $\mathbf{w}_0$= Los pesos, producto de la combinación lineal de los vectores de soporte.
- $\mathbf{b}_0$= Intercepto o umbral.
- $\mathbf{z}$= Vector de entradas.

- **Funciones kernel**

  Estas funciones se utilizan para transformar la información a un plano dimensional donde sea más fácil hallar los vectores de soporte, con la finalidad de construir el hiperplano de separación (Cortes and Vapnik, 1995).

  $$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b \tag{36}$$

  $f(\mathbf{x})$, corresponde al hiperplano de separación, en donde existe una función kernel $\phi(\mathbf{x})$. Esta función se descompone de la siguiente manera:

  $$\phi(\mathbf{x}) = \phi(x)\phi(x_i) \tag{37}$$

  Obteniendo como resultado la expresión de la función kernel

  $$\phi(\mathbf{u}) \cdot \phi(\mathbf{v}) = K(\mathbf{u}, \mathbf{v}) \tag{38}$$

  A continuación, se explicarán algunas de ellas:

  - **Kernel polinómico:**

    $$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d \tag{39}$$

    d, es el grado del polinomio que denota en un espacio de n-dimensiones.

  - **Kernel gausiano:**

    $$K(\mathbf{u}, \mathbf{v}) = exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right) \tag{40}$$

    σ, corresponde a un parámetro de dispersión libre de tomar cualquier valor.

3. Resultados

3.1. Descripción base de datos

Los datos corresponden a un repositorio de imágenes, el cual contiene 4 clases de datos(imágenes). Dentro de estos, se encuentran 3 clases de enfermedades oculares asociadas al área de la retina y una clase que corresponde a la no presencia de ninguna enfermedad en esa zona.

El repositorio de imágenes, teniendo en cuenta las clases, contiene dos grandes grupos de datos, los cuales son: Datos de entrenamiento y datos de evaluación.

A continuación, se explicitará, por medio de una tabla y gráficos de torta, la distribución de los datos por clase y grupo.

Clase	Entrenamiento	Prueba
CNV	37205	250
DME	11348	250
Drusen	8616	250
Normal	51140	250
Total	**108309**	**1000**

Cuadro 1: Distribución de los datos.

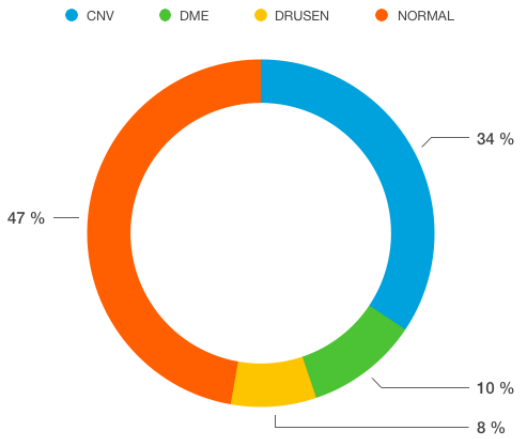


Figura 7: Distribución datos de entrenamiento.

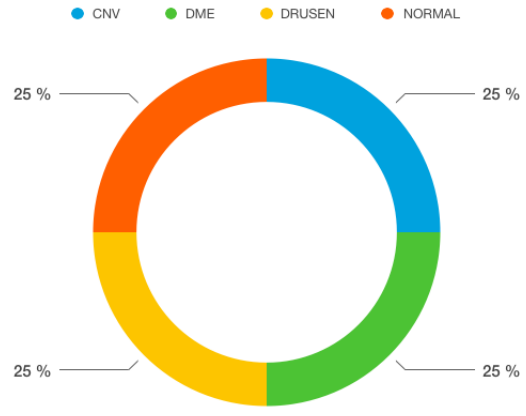


Figura 8: Distribución datos de prueba.

3.1.1. Clases

A continuación, se describen las clases de las imágenes de la base de datos:

- **Normal:** Imagen que denota la ausencia de alguna enfermedad asociada a la zona de la retina.
- **Drusen:** Imagen que denota la presencia de la enfermedad de drusen en la zona de la retina. Esta enfermedad consiste en la aparición de bultos extracelulares entre el epitelio pigmentado y la membrana de Bruch(Williams et al., 2009).
- **CNV:** Imagen que denota la presencia de la enfermedad neovascularización coroidea en la zona de la retina. Esta enfermedad consiste en la acumulación de líquidos en los vasos sanguíneos de la coroides, ocasionando un proceso de degeneración en la mácula(Yeo et al., 2019).
- **DME:** Imagen que denota la presencia de la enfermedad del edema macular diabético en la zona de la retina. Esta enfermedad consiste en la inflamación de la retina cerca del área de la mácula, por causa de la aparición de líquidos extracelulares, producto de la diabetes(Browning et al., 2018).

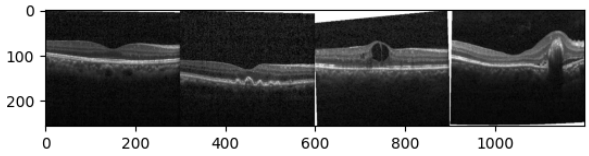


Figura 9: Tomografías de las clases: Normal, Drusen, DME y CNV respectivamente.

3.1.2. Formato de las imágenes

Las imágenes de las 4 clases de la base de datos poseen un formato RGB, que es la combinación de la intensidad de tres colores, los cuales son: Azul, rojo y verde. De acuerdo con esto, cada imagen de 2 dimensiones(H x W) se descompone en tres imágenes, de la misma dimensión, asociadas a cada intensidad de color. Lo anterior, se le denomina los canales que contiene una imagen; por lo tanto, se puede decir que una imagen en formato RGB posee tres canales. Por otro lado, cada canal de la imagen posee su propia distribución de píxeles, los cuales fluctuan entre un rango de valores de 0 a 255; siendo 0 el valor más bajo en intensidad de luz y 255 lo contrario.

3.1.3. Preprocesamiento

Antes de ingresar las imágenes en las entradas de la red neuronal, se procedió a darles un previo tratamiento, con el fin de orientar las imágenes a un formato que sea entendible para la red.

Se realizaron los siguientes procedimientos:

- **Re-escalamiento:** Se ajustaron todas las imágenes de la base de datos a un tamaño fijo de 136x136 píxeles.
- **Estandarización:** A cada canal de la imagen se le aplica la estrategia de estandarización logrando una distribución de los píxeles con media igual a 0 y desviación estándar igual a 1. El proceso consta de restar el valor de cada píxel con el promedio, dividiendo el resultado obtenido sobre la desviación estándar.

3.2. Modelos

3.2.1. Inicialización

- **Entrada de los datos**

  En la entrada de los datos al modelo es importante recalcar que de antemano se hace un muestreo de las imágenes con el fin de introducir al modelo de forma paulatina los datos a través de pequeñas muestras. El tamaño de la muestra esta dado por el

nombre de **bach size**, el cual contiene las imágenes de las clases de la base de datos.

Para los modelos se especificó que el **bach size** fuera de 64 imágenes; 16 imágenes por cada clase; esto quiere decir que existe un balanceo entre las clases y no hay de ninguna forma inequidad entre ellas. Dado que en la base de datos se presenta inequidad entre las clases, el **bach size** siempre cumplirá con las 16 imágenes por cada clase, lo que quiere decir, que las imágenes de ciertas clases se repetirán para cumplir con el tamaño de la muestra. Todo lo anterior se hace de tal modo que, se puedan visualizar la totalidad de los datos de entrenamiento.

La visualización de todos los datos se le denomina **epoch**(época); cuanto más epochs completen los modelos, mejor será el proceso de entrenamiento de la red.

- **Funciones de activación**

  En la última capa de todos los modelos se utilizó, la función de activación **Softmax** para la correcta clasificación de las clases. Esta función transforma los valores dados en la red a probabilidades que son asignadas de acuerdo con las clases.

  En las capas ocultas, se optó por utilizar la función de activación **ReLU**, la cual transforma los valores entrada de cada nodo en valores de salida entre "0" y "z".

- **Hiperparametros**

  - **Tasa de aprendizaje:**
    - Se utilizo una tasa de aprendizaje(η) inicial de 0.02.
    - Se especifico que, por cada 5000 iteraciones, se redujera esta a la mitad, con el fin de no generar saltos tan grandes en el proceso de encontrar los parámetros óptimos.
  - **Epochs:** Se especificó un número de 100 epochs en el proceso de entrenamiento.
  - **Momentum:** Se utilizo un momentum de 0.9.

- **Pesos y biases:**

  - Se generaron los pesos(w) de acuerdo con una distribución normal truncada con los siguientes parámetros:

$$w \sim N(0, \frac{1}{fan-in+fan-out})$$

    "fan-in" corresponde al número de mapas de características generados por la primera capa de la red.

    "fan-out" corresponde al número de mapas de características generados por la última capa de la red.
  - Se especificó que los valores de los biases(b) en la inicialización fueran 0.1.

- **Función de costo**

  Se eligió la función de costo **Cross Entropy** con el objetivo de ajustar los parámetros, en el proceso de entrenamiento, a través de la minimización del error.

- **Algoritmo de aprendizaje**

  Para los modelos se tomó el algoritmo de aprendizaje **SGD con momentum** para el hallazgo de los valores óptimos.

- **Modelos**

  Se realizaron los siguientes modelos con sus correspondientes estructuras:

  - **Red CNN:** Se planteó una convolutional neural network con las siguientes caracteristicas:
    - **Primera capa:** Se aplicaron 8 filtros de una tamaño de 7×7.
    - **Segunda capa:** Se aplicaron 8 filtros de una tamaño de 3×3.
    - **Transición(1):** Se aplicó el agrupamiento por el máximo(max-pooling).
    - **Tercera y cuarta capa:** Se aplicaron 16 filtros de una tamaño de 3×3.
    - **Transición(2):** Se aplicó el agrupamiento por el máximo(max-pooling).
    - **Quinta, sexta y séptima capa:** Se aplicaron 32 filtros de una tamaño de 3×3.
    - **Transición(3):** Se aplicó el agrupamiento por el máximo(max-pooling).
    - **Octava y novena capa:** Se aplicaron 64 filtros de una tamaño de 3×3.
    - **Decima capa:** Se aplicaron 96 filtros de una tamaño de 3×3.
    - **Transición(4):** Se aplicó el agrupamiento por el máximo(max-pooling).
    - **Transición(6):** Se aplicó el agrupamiento por el promedio(average-pooling).
    - **Multilayer Perceptron(MLP):** El resultado obtenido por las capas anteriores es incorporado a una pequeña red(MLP), la cual tiene la siguiente estructura:
      - **Primera capa:** Compuesta de 64 nodos o perceptrons.
      - **Dropout(1):** Aplicación del dropout con una tasa de 0.5
      - **Segunda capa:** Compuesta de 128 nodos o perceptrons.
      - **Dropout(2):** Aplicación del dropout con una tasa de 0.5
      - **Tercera capa:** Corresponde a la clasificación por lo tanto sería de 4 nodos de acuerdo con las 4 clases a clasificar.

    Cabe resaltar que entre las capas se aplico el método de bach-normalization para la conservación de la distribución de los píxeles de las imágenes.
  - **DenseNet con S&E:** Se planteó una densely connected convolutional neural network con el módulo por canal Squeeze and Exitation, de acuerdo con las siguientes caracteristicas:
    - **Bloque(1):** Contiene 4 sub-bloques; cada sub-bloque contiene 2 capas. En cada capa se aplicó 16 filtros.

- **Concatenación(1)**: Se unen los mapas de características, generados por el bloque(1).
- **Bloque(2):** Contiene 6 sub-bloques; cada sub-bloque contiene 2 capas. En cada capa aplicó 16 filtros.
- **Concatenación(2)**: Se unen los mapas de características, generados por el bloque(2).
- **Bloque(3):** Contiene 6 sub-bloques; cada sub-bloque contiene 2 capas. En cada capa aplicó 16 filtros.
- **Concatenación(3)**: Se unen los mapas de características, generados por el bloque(3).
- **Bloque(4):** Contiene 8 sub-bloques; cada sub-bloque contiene 2 capas. En cada capa aplicó 16 filtros.
- **Concatenación(4)**: Se unen los mapas de características, generados por el bloque(4).
- **Bloque(5):** Contiene 8 sub-bloques; cada sub-bloque contiene 2 capas. En cada capa aplicó 16 filtros.
- **Concatenación(5)**: Se unen los mapas de características, generados por el bloque(5).
- **Transición:** Se aplicó el agrupamiento por el promedio(average-pooling).
- **Softmax:** Los valores obtenidos en la operación anterior son introducidos en la función softmax. La salida de esta función, es la probabilidad de la clase con respecto a las demás.

Cabe resaltar que entre las capas se aplicó el método de bach-normalization junto con la plicación de la función reLU.

► **Tiempo de entrenamiento**

El tiempo registrado en el proceso del entrenamiento esta dado por el siguiente cuadro.

Arquitectura	Tiempo	Iteraciones	# Parámetros
CNN	4h y 38 min	50'000	155'000
DenseNet con S&E	7h y 52 min	50'000	186'000

Cuadro 2: Tiempo, número de iteraciones y parámetros por arquitectura.

Para la lectura y desenvolvimiento de los modelos, se utilizo un ordenador con las siguientes características:

GPU	CPU
AMD ryzen 2600. 4Ghz	NVIDIA RTX 2060 super 16 gb ram

Cuadro 3: GPU y CPU que se utilizaron para correr los modelos.

Curvas de aprendizaje

A continuación, se muestran las curvas de aprendizaje, las cuales señalan el proceso de entrenamiento(azul) y de evaluación(naranja) que realizó cada modelo.

► DenseNet con S&E

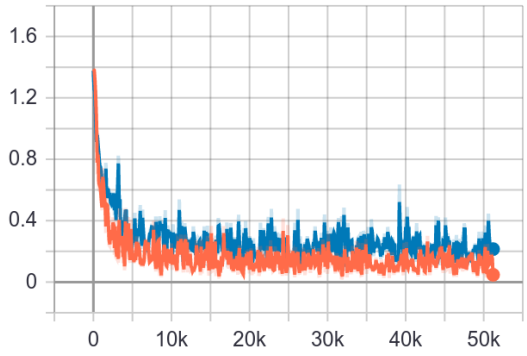


Figura 10: Comportamiento de la pérdida a lo largo de los dos procesos.(Pérdida vs Iteraciones)

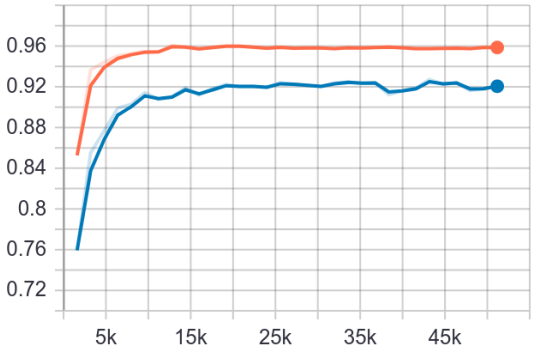


Figura 11: Comportamiento del Accuracy a lo largo de los dos procesos.(Accuracy vs Iteraciones)

► CNN

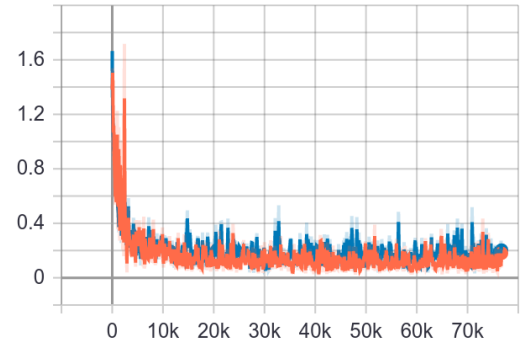


Figura 12: Comportamiento de la pérdida a lo largo de los dos procesos.(Pérdida vs Iteraciones)

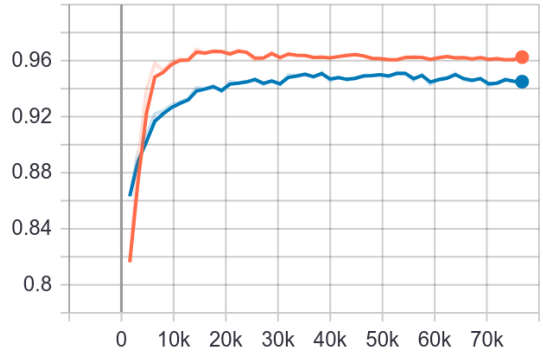


Figura 13: Comportamiento del Accuracy a lo largo de los dos procesos.(Accuracy vs Iteraciones)

De acuerdo con las figuras 10 y 12, se puede decir que tanto la red DenseNet con S&E como la red

CNN, en su proceso de entrenamiento, se logró la convergencia del error; por lo tanto, hubo un éxito en el proceso de aprendizaje, y este se vió reflejado en el proceso de evaluación.

Con respecto a las figuras 11 y 13, se puede decir que tanto la red DenseNet con S&E como la red CNN, en su proceso de entrenamiento, se logró encontrar los parámetros óptimos; por lo tanto, en el proceso de validación se pudo obtener buenas tasas de clasificación.También se puede decir que, no hubo sobreajuste en el proceso de entrenamiento.

3.3. Evaluación

Durante el proceso de entrenamiento, de manera gradual se fue colocando a prueba la eficiencia del modelo mediante la introducción del conjunto de datos de evaluación por medio de pequeñas muestras, las cuales, estaban conformadas por 64 imágenes. En este proceso al cumplir una epoch en el entrenamiento se visualiza todo el conjunto de datos de evaluación, el cual, esta conformado por 1000 imágenes.

La salida del proceso de evaluación esta dada en las matrices de confusión. Estas matrices muestran como se clasificaron las imágenes de acuerdo a los 4 tipos de clase. La lectura de estas matrices se muestra de la siguiente manera:

- Las filas representan las clases reales.
- Las columnas representan las clases predichas.

De acuerdo con lo anterior, se puede afirmar:

- Si la clase predicha es igual a la clase real, se puede decir que el modelo clasificó bien la imagen.
- Si la clase predicha no es igual a la clase real, se puede decir que el modelo no clasificó bien la imagen.

De acuerdo con, las probabilidades arrojadas por la función de activación softmax, se estableció un conjunto de umbrales(0.5, 0.6, 0.7, 0.85, 0.9, 0.95) para realizar las matrices de confusión; de tal forma, que la probabilidad mayor al umbral fue clasificada, como se pueden visualizar a continuación:

- DenseNet con S&E
  - Umbral 0.5

	Normal	CNV	DME	Drusen
Normal	240	0	4	4
CNV	0	247	2	1
DME	0	2	248	0
Drusen	1	25	1	222

  - Umbral 0.6

	Normal	CNV	DME	Drusen
Normal	240	0	4	3
CNV	0	247	2	1
DME	0	1	248	0
Drusen	0	23	1	221

  - Umbral 0.7

	Normal	CNV	DME	Drusen
Normal	236	0	1	2
CNV	0	246	1	0
DME	0	1	246	0
Drusen	0	21	1	214

  - Umbral 0.85

	Normal	CNV	DME	Drusen
Normal	227	0	1	1
CNV	0	239	1	0
DME	0	0	241	0
Drusen	0	16	0	193

  - Umbral 0.90

	Normal	CNV	DME	Drusen
Normal	224	0	0	1
CNV	0	234	1	0
DME	0	0	240	0
Drusen	0	13	0	186

  - Umbral 0.95

	Normal	CNV	DME	Drusen
Normal	202	0	0	1
CNV	0	217	1	0
DME	0	0	232	0
Drusen	0	6	0	155

- CNN
  - Umbral 0.5

	Normal	CNV	DME	Drusen
Normal	245	0	0	5
CNV	0	248	2	0
DME	0	4	246	0
Drusen	1	24	0	223

  - Umbral 0.6

	Normal	CNV	DME	Drusen
Normal	243	0	0	5
CNV	0	245	1	0
DME	0	2	244	0
Drusen	1	22	0	214

  - Umbral 0.7

	Normal	CNV	DME	Drusen
Normal	242	0	0	3
CNV	0	244	1	0
DME	0	2	239	0
Drusen	1	16	0	209

  - Umbral 0.85

	Normal	CNV	DME	Drusen
Normal	235	0	0	3
CNV	0	234	1	0
DME	0	2	234	0
Drusen	0	11	0	189

  - Umbral 0.9

	Normal	CNV	DME	Drusen
Normal	228	0	0	1
CNV	0	230	1	0
DME	0	1	232	0
Drusen	0	7	0	178

  - Umbral 0.95

	Normal	CNV	DME	Drusen
Normal	186	0	0	1
CNV	0	224	1	0
DME	0	1	226	0
Drusen	0	6	0	161

- Comparación de medidas

  La matriz de confusión refleja la conconcordancia entre los datos predichos y los datos reales, que permiten poder inferir, si el modelo posee una buena capacidad predictiva. De acuerdo con lo descrito,

los valores dados en la diagonal de la matriz muestran el número de imágenes que fueron bien clasificadas por el modelo; esto quiere decir que, el resultado predicho por el modelo es igual al resultado real. Por otra parte, los valores que se encuentran por fuera de la diagonal, corresponden al número de imágenes que fueron mal clasificadas por el modelo; esto quiere decir que, el resultado predicho por el modelo no es igual al resultado real.

Verdadero(+)	**Falso(+)**
Falso(-)	**Verdadero(-)**

Cuadro 4: Matriz de confusión

De conformidad con, las tablas de confusión arrojadas anteriormente , se procedio a calcular diferentes tipos de ratio con el objetivo de analizar las clasificaciones dadas por los modelos.

▷ Especificidad(precision):

$$Pr = \frac{Verdadero(+)}{Verdadero(+) + Falso(+)}$$

▷ Recuperación(recall):

$$Rc = \frac{Verdadero(+)}{Verdadero(+) + Falso(-)}$$

▷ Exactitud(accuracy):

$$Accuracy = \frac{Verdadero(+) + Verdadero(-)}{Total\,de\,prediciones}$$

▷ Medida f1(f1-score):

$$F_1 = 2 * \frac{Pr \times Rc}{Pr + Rc}$$

Los calculos se ven visualizados en los cuadros 5, 6, 7, 8, 9, 11, de acuerdo con el conjunto de umbrales(0.5, 0.6, 0.7, 0.85, 0.9, 0.95). Cabe resaltar que el valor de "support" corresponde al número de imágenes que cumplieron la condición de umbral.

3.4. Interpretabilidad

Para interpretar la manera en cómo la red neuronal identifica la clase asociada a la imagen desde un punto de vista local, se utilizó el método de LIME. El método LIME fue aplicado al modelo CNN, el cual se escogió como el mejor modelo con base en sus predicciones y tiempo computacional.

En la implementación del método explicitado, se tomó de manera aleatoria una imagen por cada clase del conjunto de datos de entrenenamiento. Cada imagen fue fragmentada de forma diferente en 77 regiones, y por cada una de estas , se generaron 700 imágenes perturbadas. La finalidad de esta operación fue identificar las 7 zonas que más contribuyeron a la probabilidad de la clase.

A continuación, en las imágenes que se relacionan se puede detectar las regiones que más incidieron en la clasificación correcta de la imagen, versus, la imagen original perturbada en regiones. Como se puede apreciar en la parte izquierda de cada imagen se encuentra visualmente pincelada con color verde en donde se resalta las zonas más influyentes en la predicción; en tanto que, en la parte derecha se muestra como fue fragmentada la imagen en regiones.

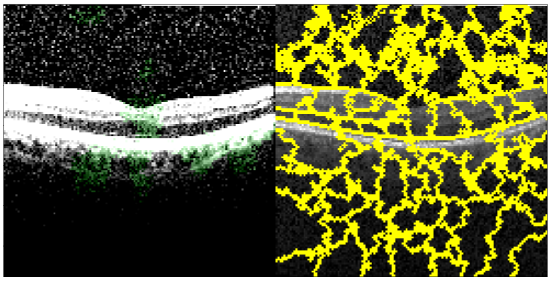


Figura 14: Clase normal.

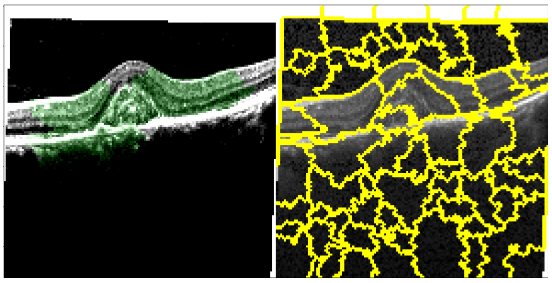


Figura 15: Clase CNV.

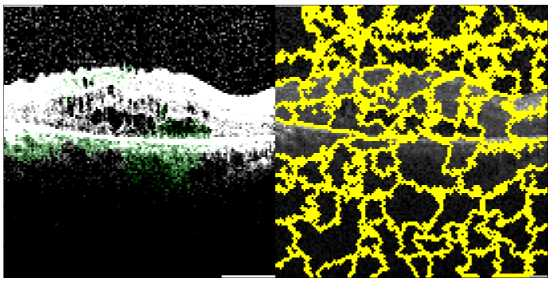


Figura 16: Clase DME.

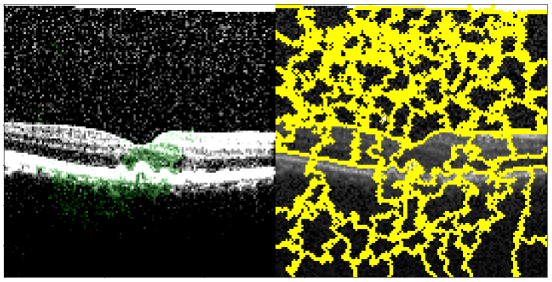


Figura 17: Clase Drusen.

3.5. Ensamble SVM

Se propuso adaptar el método de SVM al modelo CNN calculado para analizar que tan ventajoso puede llegar a ser la combinación de estas dos metodologias, con el objetivo de lograr una mejor predicción. Para ello, se modificó la estructura de la red CNN, retirando la multilayer perceptron(MLP) y colocando en su reemplazo el método de SVM.

Cuadro 5: Medidas de la tabla de confusión con umbral de 0.5.

	Normal		CNV		DME		Drusen		T = 0.5		
Arquitectura	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Support**	**Accuracy**	**f1-score**
CNN	0.9959	0.9800	0.8985	0.9920	0.9919	0.9840	0.9780	0.8991	998	0.96392	0.96394
DenseNet con S&E	0.9958	0.9677	0.9014	0.9880	0.9735	0.9920	0.9779	0.8915	997	0.95987	0.95982

Cuadro 6: Medidas de la tabla de confusión con umbral de 0.6.

	Normal		CNV		DME		Drusen		T = 0.6		
Arquitectura	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Support**	**Accuracy**	**f1-score**
CNN	0.9959	0.9798	0.9107	0.9959	0.9959	0.9918	0.9771	0.9029	977	0.96827	0.96793
DenseNet con S&E	1.0	0.9716	0.9114	0.9880	0.9725	0.9959	0.9822	0.9020	991	0.96468	0.96458

Cuadro 7: Medidas de la tabla de confusión con umbral de 0.7.

	Normal		CNV		DME		Drusen		T = 0.7		
Arquitectura	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Support**	**Accuracy**	**f1-score**
CNN	0.9958	0.9877	0.9312	0.9959	0.9958	0.9917	0.9858	0.9247	957	0.97596	0.97560
DenseNet con S&E	1.0	0.9874	0.9179	0.9959	0.9879	0.9959	0.9907	0.9067	969	0.97213	0.97196

Cuadro 8: Medidas de la tabla de confusión con umbral de 0.85.

	Normal		CNV		DME		Drusen		T = 0.85		
Arquitectura	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Support**	**Accuracy**	**f1-score**
CNN	1.0	0.9873	0.9473	0.9957	0.9957	0.9915	0.9843	0.9450	909	0.9812	0.98063
DenseNet con S&E	1.0	0.9912	0.9372	0.9958	0.9917	1.0	0.9948	0.9234	919	0.9793	0.97873

Cuadro 9: Medidas de la tabla de confusión con umbral de 0.9.

	Normal		CNV		DME		Drusen		T = 0.90		
Arquitectura	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Support**	**Accuracy**	**f1-score**
CNN	1.0	0.9956	0.9663	0.9956	0.9957	0.9957	0.9944	0.9621	878	0.9886	0.9880
DenseNet con S&E	1.0	0.9955	0.9473	0.9957	0.9958	1.0	0.9946	0.9346	899	0.9833	0.9825

Cuadro 10: Medidas de la tabla de confusión con umbral de 0.95.

	Normal		CNV		DME		Drusen		T = 0.95		
Arquitectura	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Support**	**Accuracy**	**f1-score**
CNN	1.0	0.9946	0.9696	0.9955	0.9955	0.9955	0.9938	0.9640	806	0.9888	0.9885
DenseNet con S&E	1.0	0.9950	0.9730	0.9954	0.9935	1.0	0.9935	0.9627	814	0.9901	0.9893

Cuadro 11: Medidas de la tabla de confusión CNN+SVM

	Normal		CNV		DME		Drusen				
Arquitectura	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Pr**	**Rc**	**Support**	**Accuracy**	**f1-score**
CNN+SVM	0.22	0.66	0.21	0.21	0.00	0.00	0.00	0.00	1000	0.22	0.13

Debido al costo computational, se procedió a escoger una muestra de cada clase del conjunto de datos de entrenamiento, el cual está representado en el siguiente cuadro:

Clase	**Muestra**
Normal	10220
CNV	7440
DME	2082
Drusen	1720

Se propuso para el SVM un kernel gaussiano y una validación cruzada k=5. Con lo anterior, se buscó que los márgenes del hiperplano de separación fueran los óptimos para una buena clasificación.

Conforme a lo explicitado anteriormente, se obtuvo como resultado la siguiente matriz de confusión:

	Normal	CNV	DME	Drusen
Normal	165	85	0	0
CNV	198	52	0	0
DME	190	60	0	0
Drusen	199	51	0	0

De acuerdo con la matriz de confusión, se calcularon diferentes medidas, contenidas en el cuadro 11.

4. Conclusiones

En el desarrollo de la investigación se pudo evidenciar desde el punto de vista téorico y aplicado que, estos modelos de redes neuronales han demostrado alta capacidad predictiva en la clasificación de imágenes.

En el análisis de las redes neuronales prospuestas, se puede concluir lo siguiente:

- De acuerdo con los modelos arrojados, se puede afirmar que, el mejor modelo de acuerdo al principio de parsimonia es la arquitectura CNN, con un total de 155'000 parámetros.
- Con base en la medida f1-score, se puede afirmar que, el mejor modelo predictivo corresponde a la arquitectura DenseNet con módulo de atención por bloque Squeeze and Exitation, con un f1-score de 0.9893.
- Una posible razón a la mala actuación que presentó la arquitectura CNV+SVM se debió a que en la arquitectura CNN, en la parte de la multilayer perceptron, se extraen los patrones clave de las imágenes para su correcta clasificación.
- La arquitectura CNN logró obtener un costo computacional menor con respecto a los otros modelos, registrando un tiempo de 4 horas y 38 minutos.

- A pesar de tener pocos datos para la clase Drusen con respecto a las demás clases, se logró obtener una buena presición de 0.9858.

Referencias

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A., and Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292.

Browning, D. J., Stewart, M. W., and Lee, C. (2018). Diabetic macular edema: evidence-based management. *Indian journal of ophthalmology*, 66(12):1737–1738.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Denker, J. S. and LeCun, Y. (1991). Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859.

Goodfellow, I., Bengio, Y., and Courville, A. (2016a). *Deep learning*. MIT press.

Goodfellow, I., Bengio, Y., and Courville, A. (2016b). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7133–7134.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jaiswal, T. and Jaiswal, S. (2019). Deep learning in medicine.

Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., et al. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA.

Ravì, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., and Yang, G.-Z. (2016). Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, 21(1):4–21.

Razzak, M. I., Naz, S., and Zaib, A. (2018). Deep learning for medical image processing: Overview, challenges and the future. In *Classification in BioApps*, pages 323–350. Springer.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?.^{ex}plaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, page 4.

Sarkar, D. (2018). A comprehensive hands-on guide to transfer learning with real-world applications in deep learning.

Williams, M., Craig, D., Passmore, P., and Silvestri, G. (2009). Retinal drusen: harbingers of age, safe havens for trouble. *Age and ageing*, 38(6):648–649.

Woo, S., Park, J., Lee, J.-Y., and So Kweon, I. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Yeo, N. J. Y., Chan, E. J. J., and Cheung, C. (2019). Choroidal neovascularization: Mechanisms of endothelial dysfunction. *Frontiers in Pharmacology*, 10:1–2.

ZhangLabData (2018). Large dataset of labeled optical coherence tomography (oct) and chest x-ray images.