Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Lo que las redes neuronales no saben: descubriendo e interpretando el conocimiento aprendido en redes neuronales convolucionales

Tesis de Licenciatura en Ciencias de Datos

Gabriel Baca Rubinztain

Director: Gabriel Kreiman

Codirector: Pablo Negri

Buenos Aires, 2025

RESUMEN

Este trabajo aborda el problema de inferir el conocimiento adquirido por una red neuronal convolucional para clasificación de imágenes, sin información previa sobre sus datos de entrenamiento. En particular, se estudia si es posible determinar qué clases fueron utilizadas durante el entrenamiento bajo distintas restricciones sobre el acceso a la arquitectura, los pesos y las salidas de la red. Se definen seis variantes del problema, que representan escenarios cada vez más generales y desafiantes. Entre los métodos explorados, el análisis de la proporción de activación neuronal en capas Fully Connected combinado con clustering no supervisado mediante K-Means mostró resultados sólidos, alcanzando scores F1 superiores a 0,8 incluso en configuraciones con hasta siete clases de diez no entrenadas. La principal contribución del trabajo es el desarrollo de un framework basado en DeepDream para la extracción de conocimiento, que es agnóstico al dataset, a la arquitectura del modelo y a la métricas de similitud. Experimentos preliminares con este enfoque alcanzaron scores F1 superiores al 95 % en MNIST con redes artesanales pequeñas y mostraron desempeños comparables en CIFAR-10 utilizando una ResNet20 preentrenada.

Palabras clave: Redes neuronales, inferencia de conocimiento, análisis de activaciones, interpretabilidad, DeepDream, out-of-distribution detection, open set recognition.

ABSTRACT

This work addresses the problem of inferring the knowledge acquired by a convolutional neural network for image classification, without prior information about its training data. Specifically, it explores whether it is possible to determine which classes were used during training under various constraints on access to the network's architecture, weights, and outputs. Six problem variants are defined, each representing increasingly general and challenging scenarios. Among the explored methods, the analysis of neuron activation ratios in Fully Connected layers combined with unsupervised clustering via K-Means showed strong performance, achieving F1 scores above 0,8 even in configurations with up to seven out of ten classes untrained. The main contribution of this work is the development of a DeepDream-based framework for knowledge extraction, which is agnostic to the dataset, model architecture, and similarity metrics. Preliminary experiments using this approach achieved F1 scores above 95 % on MNIST with small handcrafted networks and showed comparable performance on CIFAR-10 using a pretrained ResNet20.

Keywords: Neural networks, knowledge inference, activation analysis, interpretability, DeepDream, out-of-distribution detection, open set recognition.

AGRADECIMIENTOS

Primero y principal, quiero agradecer a Exactas, a la Universidad de Buenos Aires y a Argentina, por haberme recibido con los brazos abiertos y con la misma calidez con la que acoge a quienes nacieron y vivieron toda su vida aquí. Estaré por siempre agradecido de haber tenido la oportunidad y el privilegio de formarme en una casa de estudios de tan alto nivel académico y humano.

A mis padres, por brindarme la formación y el apoyo incondicional que me permitió llegar hasta este punto, celebrando cada uno de mis logros como si fueran propios.

A mi hermano, por estar a mi lado desde el comienzo de este trayecto, con una compañía y compañerismo que hicieron más liviano el camino.

Al profesor Gabriel Kreiman, por introducirme al mundo de la investigación y darme la posibilidad de colaborar, aprender y construir sobre sus ideas.

Al profesor Pablo Negri, por tenderme la mano desde el primer momento y ayudarme con compromiso y claridad a lo largo de esta etapa.

Índice general

1. INTRODUCCIÓN

1.1. Motivación

En los últimos años, las redes neuronales han revolucionado diversos campos de la ciencia y la ingeniería, permitiendo avances tecnológicos que, hasta hace poco, hubieran parecido propios de la ciencia ficción. A pesar de su impacto, el funcionamiento interno de estos sistemas continúa siendo en gran medida una caja negra: si bien comprendemos matemáticamente por qué los modelos adquieren su capacidad predictiva, aún entendemos muy poco sobre los mecanismos internos que conducen a sus decisiones. Las redes modernas más utilizadas contienen miles de millones de parámetros cuya compleja interacción dificulta un entendimiento directo de su comportamiento. Todo a punta a que las redes neuronales vinieron para quedarse y es por esto que comprender cómo se organiza el conocimiento dentro de estas estructuras no es solo un desafío intelectual, sino una necesidad práctica en contextos donde la seguridad, la privacidad y la confianza en los modelos son fundamentales.

Un aspecto particularmente interesante, y aún poco explorado, es el vínculo entre el conocimiento adquirido por una red y los datos con los que fue entrenada. ¿Cuánta información del conjunto original permanece codificada en la red tras el entrenamiento? ¿Es posible inferir, a partir de su estructura interna, qué clases o ejemplos específicos fueron utilizados?

Estas preguntas abren una línea de investigación con profundas implicancias técnicas y éticas. Si bien las redes neuronales están diseñadas para generalizar, no está claro en qué medida lo logran sin conservar detalles particulares del conjunto de entrenamiento. Indagar sobre este fenómeno es clave no solo desde una perspectiva teórica, sino también por las posibles consecuencias prácticas que tiene: si las redes almacenan internamente representaciones que permiten reconstruir o identificar parte de los datos originales, podrían comprometer la confidencialidad de la información, aun sin intención explícita. Esto resulta especialmente preocupante en aplicaciones que involucran datos sensibles, como salud, finanzas o justicia, donde la exposición indirecta de información privada, por mínima que sea, puede tener consecuencias significativas.

Esta tesis se sitúa dentro del ámbito de la *interpretabilidad de redes neuronales*, un área de estudio dentro del aprendizaje profundo en donde no se busca generar modelos con mayor precisión, sino comprender qué saben las redes y cómo lo saben.

Se propone investigar si, observando únicamente la estructura interna de una red entrenada, sus pesos y sus activaciones, es posible determinar la clase de datos con los que fue entrenada.

1.2. Objetivos

El objetivo de este trabajo es encontrar herramientas, técnicas, algoritmos y métricas que permitan determinar el conocimiento que una red neuronal ha adquirido, sin conocer de antemano el conjunto de datos utilizado para su entrenamiento.

Se va a acotar la exploración al caso particular de redes neuronales convolucionales utilizadas para clasificación de imágenes, con el fin de mantener el problema dentro de un dominio bien definido y abordable.

En este contexto, se va a considerar que una red posee cierto conocimiento si es posible inferir, con un grado razonable de certeza, las clases de datos con las que fue entrenada. Como es de esperar, la dificultad del problema varía significativamente según la arquitectura de la red, el conjunto de datos y la información disponible sobre la misma. Por esta razón, el objetivo es explorar variantes del problema con distintos niveles de complejidad, avanzando progresivamente hacia instancias más generales y menos triviales.

2. MARCO TEÓRICO

2.1. Conceptos Preliminares

2.1.1. Redes Neuronales Artificiales

Las redes neuronales artificiales tienen su origen en los primeros intentos de modelar computacionalmente el funcionamiento de las neuronas biológicas. En 1943, McCulloch y Pitts [1] propusieron un modelo simplificado de neurona basado en una función booleana sobre múltiples entradas binarias. Este modelo fue extendido por Rosenblatt en 1958 [2] con la introducción del *perceptrón*, un modelo de neurona artificial que realiza una combinación lineal de las entradas y aplica una función de activación para clasificar patrones linealmente separables.

El desarrollo de redes más profundas se vio limitado por la imposibilidad de entrenar múltiples capas debido al problema del *desvanecimiento del gradiente*, en donde los gradientes calculados para el entrenamiento se vuelven extremadamente pequeños al propagarse hacia las capas más profundas de una red, dificultando la actualización efectiva de los pesos y obstaculizando el aprendizaje [3]. Este obstáculo fue parcialmente resuelto con el algoritmo de *retropropagación* (backpropagation), formalizado en los años 80 por Rumelhart, Hinton y Williams [4], y superado definitivamente en las décadas recientes gracias a avances en optimización, normalización, funciones de activación no lineales y poder computacional.

Formalmente, una red neuronal profunda puede describirse como una composición de funciones parametrizadas:

$$f(x;\theta) = f_L \circ f_{L-1} \circ \cdots \circ f_1(x),$$

donde cada f_l representa la operación realizada en la capa l, y θ agrupa todos los parámetros (pesos y sesgos) de la red. Cada capa aplica una transformación de la forma:

$$f_l(h_{l-1}) = \phi(W_l h_{l-1} + b_l),$$

donde $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$ es la matriz de pesos, $b_l \in \mathbb{R}^{d_l}$ es el vector de sesgos, ϕ es una función de activación no lineal (ReLU o tanh comúnmente), y h_{l-1} representa la activación de la capa anterior. La entrada h_0 es el dato inicial, que dependiendo el dominio donde se trabaje puede representan vectores numéricos, textos, imágenes, sonidos, entre muchas otras posibilidades. En este tipo de capas, todos los nodos de la capa anterior están conectados a cada nodo de la capa actual y es por esto que reciben el nombre de capas *Fully Connected*.

Redes Convolucionales: En tareas de clasificación de imágenes, las redes neuronales convolucionales (CNNs) han probado ser especialmente efectivas. En lugar de utilizar matrices de pesos densas, como en las llamadas capas Fully Connected que describimos arriba, las capas convolucionales utilizan filtros (*kernels*) aprendidos que explotan la estructura espacial de los datos.

Una capa convolucional aplica un conjunto de filtros $\{K_i\}$ de tamaño $k_i \times k_i$, sobre la imagen de entrada $X \in \mathbb{R}^{C \times H \times W}$, en donde C representa la cantidad de canales (3 en imágenes RGB o 1 en imágenes en escala de grises); H, W representan el alto y ancho de la imagen. La aplicación del filtro genera mapas de activación A_i:

$$A_i = K_i * X + b_i,$$

donde $*$ denota la operación de convolución, y b_i es un sesgo. La operación se realiza deslizando el filtro sobre la imagen con un paso definido (*stride*), y puede complementarse con *padding*, que consiste en añadir píxeles (generalmente ceros) alrededor de la imagen de entrada para controlar el tamaño de salida.

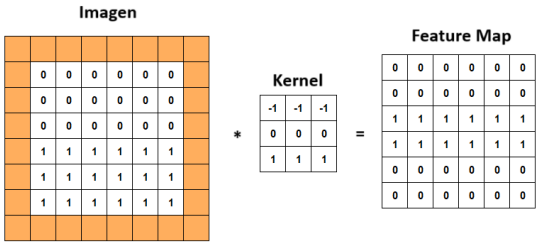


Fig. 2.1: Ejemplo de operación de convolución 2D sobre una imagen binaria. A la izquierda se observa la imagen de entrada de tamaño 8×8, extendida con padding de 1 píxel (en naranja) para preservar el tamaño espacial durante la convolución. El filtro mostrado es de 3×3. Se aplica con un stride de 1, lo que implica que el kernel se desliza un píxel por vez en cada dirección. A la derecha se muestra el mapa de activación (*feature map*) resultante, de tamaño 6×6, que refleja las regiones de la imagen donde el patrón buscado por el filtro está presente.

Estos mapas de activación se pasan por una función no lineal y, opcionalmente, una operación de *pooling* que reduce la resolución espacial, conservando la información más relevante. Dos variantes comunes de esta operación son el *max pooling* y el *average pooling*. En el caso de *max pooling*, la operación selecciona el valor máximo dentro de una ventana local (por ejemplo, 2×2) que se desliza sobre el mapa de activación, conservando únicamente la activación más fuerte en cada región [5]. Esto permite destacar las características más prominentes. Por otro lado, el *average pooling* calcula el promedio de los valores en cada ventana, produciendo una representación más suave y conservadora. Ambas técnicas ayudan a reducir la dimensionalidad y el costo computacional, manteniendo las propiedades discriminativas de las representaciones.

A medida que se apilan múltiples capas, las CNNs son capaces de aprender representaciones jerárquicas: desde bordes y texturas en las primeras capas, hasta formas y conceptos semánticos en las capas profundas [6].

Es común que las arquitecturas convolucionales utilicen múltiples capas convolucionales apiladas para extraer representaciones jerárquicas de la imagen, seguidas por una o más capas Fully Connected al final del modelo. Estas capas finales se encargan de consolidar la información extraída y realizar la predicción final, típicamente mediante una capa de clasificación suave como *SoftMax*.

Pesos vs. Activaciones: Para nuestra exploración, es fundamental distinguir las diferencias que existen entren los *pesos* W_l, b_l y las *activaciones* $h_l(x)$. Los pesos representan los parámetros aprendidos por la red durante el entrenamiento y estos permanecen fijos durante la inferencia. Las *activaciones* $h_l(x)$ son los valores que se generan al procesar un estímulo x a través de la red. A diferencia de los pesos, las activaciones son específicas de cada entrada y cambian en función del estímulo que se le presente a la red. Tanto los pesos como las activaciones reflejan patrones distintos que nos dan un vistazo adentro el conocimiento aprendidos por la red, lo cual es uno de los focos principales de este trabajo.

Proceso de Entrenamiento: El entrenamiento de una red neuronal consiste en minimizar una función de pérdida $\mathcal{L}(f(x), y)$, donde $f(x)$ es la predicción de la red e y es la etiqueta verdadera. Para ello, se utiliza el método de *descenso por gradiente estocástico* (SGD), que ajusta los pesos según:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(f(x), y),$$

donde η es la tasa de aprendizaje (*learning rate*). El gradiente se computa sobre lotes o *batches*, pequeños subconjuntos de datos del conjunto de entrenamiento. Esto permite una estimación eficiente del gradiente y mejora la convergencia.

Una extensión habitual de SGD es la incorporación de *momentum*, un término que acumula gradientes pasados para suavizar las actualizaciones y evitar oscilaciones [4]. Se introduce una variable auxiliar de velocidad v que se actualiza según:

$$v \leftarrow \mu v - \eta \nabla_\theta \mathcal{L}(f(x), y), \quad \theta \leftarrow \theta + v,$$

donde $\mu \in [0, 1)$ es el coeficiente de momentum. Este enfoque permite acelerar el descenso en direcciones coherentes del espacio de parámetros y estabilizar el entrenamiento en superficies de pérdida irregulares.

Una etapa clave antes del entrenamiento es la normalización de los datos, que implica estandarizar cada entrada para que tenga media cero y varianza unitaria. Esto acelera el entrenamiento y mejora la estabilidad numérica de las optimizaciones.

Dropout: El *dropout* es una técnica de regularización utilizada comúnmente en redes neuronales profundas para prevenir sobreajuste [7]. Consiste en apagar aleatoriamente un subconjunto de neuronas durante el entrenamiento, forzando a la red a no depender excesivamente de ninguna unidad en particular. En cada iteración del entrenamiento, se selecciona al azar una fracción de las neuronas para que no participen en la retropropagación. Esto introduce una forma de ensamblado implícito, donde múltiples subredes se entrenan simultáneamente con parámetros compartidos.

Detención Temprana: La detención temprana (*early stopping*) es una técnica de control de sobreajuste que monitorea el rendimiento del modelo sobre un conjunto de validación durante el entrenamiento. La idea central es detener el entrenamiento una vez que el modelo deja de mejorar, previniendo así que continúe ajustándose a los datos de entrenamiento

a costa de su capacidad de generalización [8]. Llamamos *paciencia* al parámetro que define la cantidad de épocas consecutivas durante las cuales se permite que la métrica de validación no mejore antes de detener el entrenamiento

2.1.2. Espacio Latente, Embeddings y Métricas de Similitud

Uno de los conceptos centrales al analizar redes neuronales es el de *espacio latente*, entendido como el espacio abstracto en el que las representaciones internas de los datos son proyectadas por la red. Cada punto en este espacio corresponde a un *embedding*, estos son representaciones vectoriales que condensan la información relevante de una entrada. Los embedding se obtienen a partir de las activaciones de capas intermedias, usualmente cercanas a la salida de la red, y permiten representar de forma compacta y estructurada los patrones que el modelo ha aprendido. Llamaremos embedding al vector de activaciones que genera una capa específica de la red al procesar una entrada, donde cada componente del vector corresponde a la activación de una neurona en dicha capa.

Las representaciones internas mediante embeddings son fundamentales en el estudio de redes neuronales artificiales, ya que encapsulan propiedades semánticas y relaciones entre los datos que son claves para la toma de decisiones del modelo [9]. En este trabajo, los embeddings juegan un rol central.

Estas representaciones internas permiten comparar los estímulos de la red ante diferentes entradas. Para ello, se utilizan métricas de similitud definidas sobre los embeddings. En este trabajo, se utilizan cuatro métricas principales descritas a continuación. En lo que sigue, x e y representan vectores de embeddings obtenidos en capas iguales de la red.

Similitud coseno:

$$\mathrm{sim}_{\cos}(x, y) = \frac{x}{\|x\|_2} \cdot \frac{y}{\|y\|_2}$$

Mide el ángulo entre los vectores normalizados. Toma valores en el intervalo $[-1, 1]$, siendo 1 cuando los vectores tienen la misma dirección. La similitud coseno resulta particularmente útil cuando interesa comparar la orientación de los vectores en el espacio latente, independientemente de su magnitud. Es adecuada en escenarios donde las activaciones pueden diferir en escala pero conservar relaciones estructurales, como ocurre frecuentemente en las capas profundas de redes neuronales.

Similitud L2:

$$\mathrm{sim}_{\mathrm{L2}}(x, y) = \frac{1}{1 + \|x - y\|_2}$$

Transforma la distancia euclidiana en una medida de similitud acotada en $(0, 1]$. Valores cercanos a 1 indican alta similitud. Esta métrica conserva la interpretación geométrica de la distancia euclidiana, pero la transforma en una escala inversa acotada para facilitar su uso como similitud. Es especialmente útil cuando se desea penalizar de forma progresiva las diferencias absolutas entre embeddings, manteniendo sensibilidad tanto a cambios globales como locales en las activaciones. A diferencia de la similitud coseno, tiene en cuenta la magnitud de los vectores.

Similitud L2 normalizada:

$$\text{sim}_{\text{L2-norm}}(x, y) = \frac{1}{1 + \left\| \frac{x}{\|x\|_2} - \frac{y}{\|y\|_2} \right\|_2}$$

Aplica distancia L2 tras normalizar ambos vectores a norma unitaria. Reduce el efecto de diferencias de escala. Al normalizar los vectores antes de calcular la distancia euclidiana, esta métrica se vuelve insensible a diferencias de escala entre embeddings. Esto permite comparar exclusivamente la forma o dirección relativa de las activaciones, lo cual resulta útil cuando se sospecha que las variaciones en norma no son informativas. La salida se transforma nuevamente en una similitud acotada en $(0, 1]$, facilitando su interpretación y comparación.

Similitud por patrón de disparo: Se binarizan los vectores según un umbral θ:

$$b_i(x) = \begin{cases} 1 & \text{si } x_i > \theta \\ 0 & \text{en otro caso.} \end{cases}$$

La similitud se define como la proporción de coincidencias entre los patrones binarios:

$$\text{sim}_{\text{firing}}(x, y) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[b_i(x) = b_i(y)]$$

donde $\mathbb{1}[\cdot]$ denota la función indicadora y n es la cantidad de componentes de los vectores de embeddings. Como se verá en el desarrollo del trabajo, esta métrica es capaz de capturar información relevante sobre los conocimientos adquiridos por la red, incluso cuando no se consideran las magnitudes de activación.

2.2. Literatura Existente

El problema abordado en este trabajo se encuentra en la intersección de distintas líneas de investigación dentro del aprendizaje profundo. El tema específico que se explora no responde a una categoría tradicional única y existe poca literatura que estudia sistemáticamente nuestro problema. Dicho esto, existen diferentes ramas de estudio que abordan conceptos y motivaciones similares. Destacamos en particular la interpretabilidad de modelos, el Open Set Recognition y la detección de datos fuera de distribución (Out-of-Distribution Detection). El estudio de estas ramas sirvió como un punto clave en este trabajo, pues proporcionaron una mirada holística al problema, sirviendo así de una buena fuente de inspiración.

En la mayoría de los enfoques presentes en la literatura se proponen modificaciones a la arquitectura del modelo o se requiere acceso a la red durante el entrenamiento para registrar métricas internas que luego sirvan como punto de comparación. En contraste, las variantes exploradas en esta tesis resultan novedosas, pues operan exclusivamente sobre redes ya entrenadas, en su estado natural, sin alterar su arquitectura y sin haber tenido acceso al proceso de entrenamiento. Esto permite estudiar el conocimiento implícito de la

red desde una perspectiva más general y realista, alineada con escenarios donde solo se dispone del modelo final.

2.2.1. Interpretabilidad

Una de las ramas más relevantes a este trabajo es la de interpretabilidad de redes neuronales. Esta área busca entender cómo funcionan los modelos de aprendizaje profundo, qué patrones detectan, y cómo y por qué toman las decisiones que toman.

Dentro de la rama de interpretabilidad se han desarrollado múltiples enfoques relacionados al análisis de las representaciones internas que genera la red y cómo esta responde a estímulos externos.

Uno de los enfoques más conocidos es el de *maximización de activaciones* que consiste en identificar o generar imágenes de entrada que maximicen la activación de ciertas neuronas, revelando los patrones que "espera ver" y abriendo así la posibilidad de interpretar su toma de decisiones. Un trabajo fundacional en esta área fue *DeepDream*. Este enfoque se basa en trabajos previos como el de Erhan et al. [10], quienes propusieron maximizar activaciones neuronales con regularización para obtener patrones representativos de alto nivel. Zeiler y Fergus [11] desarrollaron las *DeconvNets*, una técnica que proyecta activaciones intermedias de vuelta al espacio de entrada para visualizar qué partes de la imagen activan cada filtro. Yosinski et al. [12] mejoraron la claridad de las visualizaciones al incorporar regularizadores y redes previas para sintetizar imágenes que activan neuronas específicas. Finalmente, Olah et al. [13] sintetizaron y extendieron estas ideas en una presentación interactiva que combinó técnicas de optimización, ejemplos visuales y explicaciones intuitivas, consolidando herramientas claves para el estudio de la interpretabilidad. En la actualidad, diferentes técnicas han sido extendidas a ramas interdisciplinarias. Xiao y Kreiman desarrollaron XDream, un método de maximización de activaciones basado en redes generativas y algoritmos evolutivos sin necesidad de recurrir a backpropagation [14]. XDream ha sido utilizado para estudiar las preferencias neuronales en monos ante estímulos visuales [15].

Otro enfoque influyente para la interpretabilidad de redes neuronales es el de los *saliency maps*, que busca estimar la importancia de cada píxel de la imagen de entrada en la decisión final del modelo. Simonyan et al. [16] introdujeron una técnica de visualización basada en calcular el gradiente del *logit* de una clase (es decir, la salida del modelo antes de aplicar *Softmax*) con respecto a los píxeles de entrada. Este gradiente indica qué cambios en la imagen afectarían más la confianza del modelo en esa clase, permitiendo generar mapas de calor que resaltan las regiones más relevantes para la predicción.

2.2.2. Open Set Recognition

Otro campo relevante es el *Open Set Recognition* (OSR), que aborda escenarios en los cuales las clases disponibles durante el testeo no coinciden completamente con las clases presentes en el entrenamiento. En estos casos, el objetivo del modelo no solo es clasificar correctamente las clases conocidas, sino también identificar aquellas entradas que pertenecen a clases no vistas. Scheirer et al. [17] introdujeron una formulación basada en teoría de conjuntos abiertos, incorporando funciones de rechazo en modelos discriminativos. Posteriormente, Bendale y Boult [18] propusieron *OpenMax*, una extensión de redes neuronales

que ajusta la salida SoftMax para modelar mejor la incertidumbre frente a clases desconocidas. Más recientemente, Neal et al. [19] combinaron autoencoders y redes adversarias para generar imágenes sintéticas de clases no vistas, entrenando modelos más robustos ante ejemplos fuera de distribución. Aunque el objetivo de esta tesis no es detectar ejemplos individuales fuera del conjunto de entrenamiento, existe una conexión conceptual: tanto en OSR como en este trabajo, se busca razonar sobre la presencia o ausencia de clases dentro del proceso de entrenamiento.

2.2.3. Out-of-Distribution Detection

Finalmente, la detección de datos fuera de distribución (Out-of-Distribution Detection, OOD) constituye otra línea relacionada. En este campo, el objetivo es identificar entradas que no provienen de la misma distribución que los datos usados durante el entrenamiento. Hendrycks y Gimpel [20] propusieron un método simple pero efectivo que utiliza la probabilidad máxima del SoftMax como señal de confianza para distinguir entre datos conocidos y desconocidos. Posteriormente, Liang et al. [21] introdujeron ODIN, una técnica que mejora la separación entre distribuciones mediante reescalado de temperatura y pequeñas perturbaciones adversarias. Más recientemente, Mohseni et al. [22] exploraron enfoques auto-supervisados que aprovechan la estructura interna de los datos para mejorar la detección de OOD sin necesidad de etiquetas. Aunque típicamente el enfoque es en identificar muestras individuales OOD, esta tesis propone un cambio de perspectiva: en lugar de centrarnos en muestras aisladas, se busca evaluar si una clase completa está o no representada en la experiencia previa del modelo.

3. METODOLOGÍA

En la tesis se abordará el problema de forma incremental, comenzando por variantes más simples y avanzando progresivamente hacia instancias más complejas. Esta estrategia permite construir una base conceptual y experimental sólida, que sirva de soporte para enfrentar variantes más desafiantes en etapas posteriores del trabajo. En general, se busca poder generalizar resultados obtenidos a partir de casos particulares, utilizando instancias específicas como punto de partida.

El orden en que se presentan las variantes coincide con el orden cronológico en que se llevó a cabo la experimentación. En total, se explorarán seis variantes distintas del problema original, cada una incorporando nuevas restricciones.

Para cada variante, se comienza por definir con claridad el problema específico a resolver. A esta definición le sigue un marco teórico que contextualiza la problemática, incluyendo motivaciones, supuestos relevantes y enfoques previos presentes en la literatura.

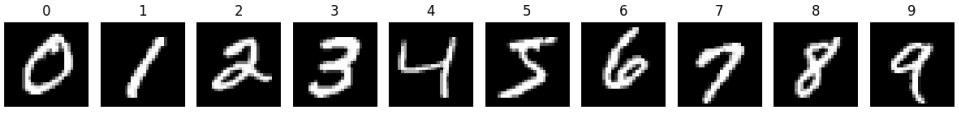
Luego de esta introducción, se describen los experimentos realizados, detallando las arquitecturas utilizadas, los datasets empleados, las métricas de evaluación y los procedimientos implementados. Se presentan los resultados obtenidos y se analiza su relevancia en el contexto de la variante estudiada.

Cada sección concluye con una breve reflexión que resume los aprendizajes obtenidos y señala cómo estos aportan al desarrollo general del trabajo o abren nuevas líneas de exploración.

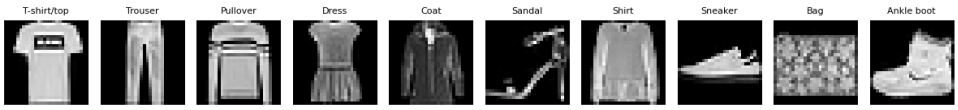
3.1. Datasets

A lo largo de este trabajo se utilizaron distintos datasets clásicos en visión por computadora, tanto por su disponibilidad como por su relevancia histórica en el estudio de redes neuronales. A continuación, se describen brevemente sus características principales:

- **MNIST**: Contiene imágenes en escala de grises de dígitos manuscritos del 0 al 9. Cada imagen tiene dimensión 28×28 píxeles. El conjunto incluye 60 000 imágenes para entrenamiento y 10 000 para testeo [5].



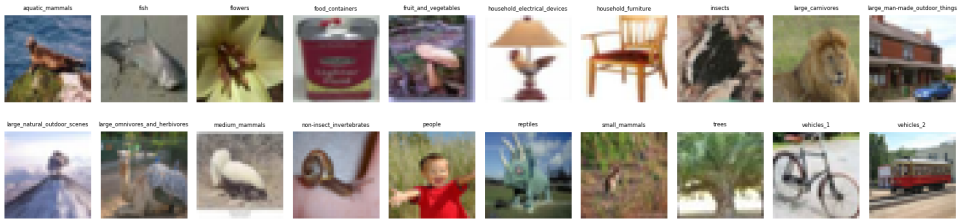
- **Fashion-MNIST**: Tiene la misma estructura que MNIST, pero reemplaza los dígitos por imágenes de prendas de ropa. También contiene 60 000 imágenes de entrenamiento y 10 000 de testeo, distribuidas en 10 clases [23].



- **KMNIST**: También con imágenes en escala de grises de 28×28 píxeles, representa caracteres selectos del silabario japonés hiragana. Posee 60 000 imágenes para entrenamiento y 10 000 para testeo, con un total de 10 clases [24].

- **CIFAR-10**: Conjunto de imágenes a color (RGB) de 32×32 píxeles, distribuidas en 10 clases que incluyen animales y objetos. Contiene 50 000 imágenes para entrenamiento y 10 000 para testeo [25].

- **CIFAR-100**: Similar a CIFAR-10 en formato y cantidad de imágenes (50 000 de entrenamiento y 10 000 de testeo), pero con 100 clases distintas organizadas en 20 superclases. A continuación se muestran algunos ejemplos por superclase [25].



La diversidad en complejidad y estructura de estos datasets permite estudiar cómo varía el conocimiento que una red es capaz de adquirir, en función del dominio, la cantidad de clases y la naturaleza visual de los datos.

3.2. Arquitecturas

NetMNIST: La arquitectura `NetMNIST` es una red neuronal convolucional pequeña, diseñada e implementada específicamente para este trabajo, con el fin de clasificar imágenes sobre el dataset MNIST. Su estructura se compone de dos capas convolucionales seguidas por dos capas Fully Connected. La primera capa convolucional aplica 10 filtros de tamaño 5×5, seguida de una operación de *max pooling* con función de activación ReLU. La segunda capa convolucional extiende la representación a 20 canales, seguida de un *dropout* espacial y otra operación de *max pooling* con ReLU. La salida es luego aplanada (flattened) y pasada por una capa Fully Connected de 50 unidades con activación ReLU, seguida de

un segundo *dropout*. Finalmente, se proyecta a una capa de 10 salidas, correspondiente a las clases del dataset, con una función *Log-SoftMax* para clasificación multiclase.

- `Conv2d(1, 10, kernel_size=5)` + ReLU + MaxPool
- `Conv2d(10, 20, kernel_size=5)` + Dropout2d + ReLU + MaxPool
- Flatten + `Linear(320, 50)` + ReLU + Dropout
- `Linear(50, 10)` + LogSoftmax

El entrenamiento se realizó utilizando el optimizador SGD con una tasa de aprendizaje de 0,01 y momentum de 0,5, entrenando en lotes. En todos los casos se implementó la técnica de detención temprana con una paciencia de 3 épocas. Esto implica que el entrenamiento se detenía si no se observaba una mejora en el conjunto de validación durante 3 épocas consecutivas.

Al tratarse de una arquitectura pequeña y computacionalmente liviana, `NetMNIST` permite entrenamientos rápidos y un análisis detallado de todas sus capas. Estas características la convierten en una herramienta ideal para exploraciones preliminares. En este trabajo, se utilizó esta arquitectura como punto de partida para desarrollar y validar los métodos propuestos, que posteriormente se buscaron generalizar en redes más profundas y datasets de mayor complejidad.

ResNet20: Para las exploraciones sobre arquitecturas más profundas, se utilizó una red `ResNet20` preentrenada sobre CIFAR-10, proveniente del repositorio de Yaofo Chen [26]. Esta arquitectura pertenece a la familia de las *Residual Networks*, introducidas por He et al. [27], que incorporan conexiones residuales entre capas para facilitar el flujo del gradiente y permitir el entrenamiento efectivo de redes profundas. En lugar de aprender directamente una transformación $\mathcal{H}(x)$, cada bloque residual aprende una función $\mathcal{F}(x)$ tal que:

$$\mathcal{H}(x) = \mathcal{F}(x) + x,$$

donde x es la entrada del bloque y $\mathcal{F}(x)$ representa la transformación aprendida (usualmente una secuencia de convoluciones, normalización y activaciones). Esta operación residual permite preservar la información original y facilita la propagación del gradiente durante el entrenamiento.

ResNet20 consta de un bloque inicial convolucional seguido por tres bloques residuales (`layer1`, `layer2`, `layer3`), cada uno compuesto por tres *BasicBlocks* con normalización por lotes y activaciones ReLU. A medida que se avanza en profundidad, se duplican los canales (de 16 a 32 y luego a 64) y se reducen las dimensiones espaciales mediante convoluciones con `stride` 2 acompañadas de operaciones `downsample`. La salida final se obtiene a través de una capa `average pool` y una capa completamente conectada de 10 unidades, una por cada clase de CIFAR-10.

En los experimentos realizados sobre ResNet20, se analizaron distintas capas de la red ordenadas de mayor a menor profundidad: `FC`, `AVGPOOL`, `LAYER3_2_BN2`, `LAYER3_2_BN1`, `LAYER3_1_BN2`, `LAYER3_1_BN1`, `LAYER3_0_BN2`, `LAYER3_0_BN1`. Las capas `FC` y `AVGPOOL` corresponden a las últimas operaciones de la red, encargadas de condensar la información

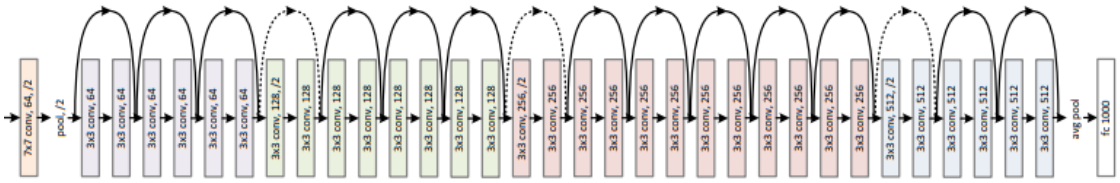


Fig. 3.1: Diagrama de ResNet34, que comparte la estructura general con ResNet20 utilizada en este trabajo. Ambas redes están compuestas por bloques residuales con conexiones de atajo, aunque ResNet34 es más profunda. Imagen adaptada de [27].

y realizar la clasificación final. Las capas restantes pertenecen al bloque `layer3`, el más profundo de los tres bloques residuales de la red. En la notación `LAYER3_i_BNj`, el número 3 indica que la capa forma parte del tercer bloque residual; el índice $i \in \{0, 1, 2\}$ representa la posición del *BasicBlock* dentro de dicho bloque, mientras que $j \in \{1, 2\}$ denota si se trata de la primera o segunda capa de normalización batch dentro del bloque.

Esta arquitectura, más profunda y expresiva que `NetMNIST`, permite evaluar la escalabilidad de los métodos desarrollados y analizar cómo se manifiestan las representaciones internas y el conocimiento adquirido en modelos de mayor complejidad.

3.3. Variantes del Problema

A continuación se detallan cada una de las variantes del problema exploradas en este trabajo.

Variante 1: En esta primera variante se plantea el problema en su forma más simple: se dispone de una red neuronal convolucional entrenada para la clasificación de dígitos del dataset MNIST, con un subconjunto de los 10 dígitos posibles. El objetivo es determinar cuáles fueron los dígitos utilizados durante su entrenamiento.

Para esta variante se tiene acceso completo a la red neuronal: se conoce su arquitectura, se dispone de sus pesos entrenados, y se pueden realizar inferencias libremente sobre cualquier conjunto de imágenes.

Variante 2: Esta variante parte del mismo escenario que la anterior: se dispone de una red neuronal entrenada con un subconjunto de dígitos de MNIST y se busca identificar cuáles fueron esos dígitos. También se cuenta con acceso completo a la red y se permite realizar inferencias.

El foco ahora no radica en simplemente resolver el problema, sino en resolverlo de forma eficiente, minimizando la cantidad de inferencias necesarias. El objetivo es encontrar una estrategia que, con la menor cantidad posible de ejemplos procesados, permita determinar con alta confianza si un dígito fue o no parte del entrenamiento.

Variante 3: En esta variante se busca nuevamente identificar qué dígitos fueron utilizados para entrenar una red neuronal sobre MNIST. A diferencia de las anteriores, no se

permite realizar inferencias: únicamente se tiene acceso a los pesos entrenados de la red, sin posibilidad de pasar imágenes ni observar activaciones.

El desafío es, entonces, inferir el subconjunto de dígitos de entrenamiento analizando únicamente la estructura interna de la red, en particular los valores numéricos de los pesos. Se parte de la hipótesis de que los dígitos que participaron del entrenamiento dejaron características distinguibles en diferentes capas de la red.

Variante 4: En esta variante también se busca determinar qué dígitos de MNIST fueron utilizados para entrenar una red neuronal, pero con una restricción distinta: no se tiene acceso a los pesos de la red, ni a su arquitectura completa. En cambio, se dispone únicamente de los valores de activación producidos al procesar imágenes.

Esto significa que sí se puede pasar imágenes por la red y observar sus activaciones internas, pero no se pueden ver ni modificar los pesos ni realizar inferencias explícitas sobre las salidas de la capa final. El problema se plantea entonces como uno de análisis de activaciones: encontrar patrones o diferencias estadísticas en las respuestas internas de la red al procesar distintos dígitos, que permitan inferir cuáles fueron efectivamente entrenados.

Variante 5: En esta variante se busca determinar qué dígitos fueron utilizados para entrenar una red neuronal sobre MNIST, pudiéndose usar los pesos, las activaciones e inferencias sobre la red, sin restricción alguna sobre capas. Esto es similar a la situación que podría llegar a suceder en la vida real, en donde se logra acceso a la arquitectura, capas internas y pesos de una red neuronal, pero sin entender a profundidad para qué sirve ni qué sabe.

Buscamos lograr inferir qué dígitos han sido usados para entrenar a la red, pudiendo usarla como sea necesario.

Variante 6: En el ecosistema actual de investigación sobre redes neuronales, suele ser requisito mínimo para publicación en revistas científicas y conferencias de alto perfil, probar los métodos estudiados en arquitecturas grandes y datasets complejos. Parte de los resultados obtenidos en variantes anteriores son agnósticos a arquitecturas y datasets y dan pie a realizar exploraciones más profundas.

Motivados por lo anterior, en esta variante buscamos explorar cómo podemos aplicar las estrategias desarrolladas en las variantes previas. ¿los métodos encontrados en variantes anteriores sirven cuando se trabaja sobre redes mucho más profundas? ¿son los resultados anteriores replicables en datasets que no sean MNIST?

Buscamos responder estas preguntas trabajando sobre redes ResNet20 que han sido preentrenadas para clasificación de imágenes en CIFAR10 [26].

4. DESARROLLO

4.1. Variante 1

Un primer acercamiento a este problema resulta de explotar el conocimiento previo que tenemos de la red: si sabemos de antemano que fue entrenada con dígitos de MNIST, entonces una forma natural de clasificar a cierto dígito como entrenado es si la red es capaz de predecirlo con un grado suficientemente alto de exactitud.

4.1.1. Experimentación

Se propone así el siguiente esquema de predicción: sea d un dígito y M una red entrenada con ciertos dígitos del dataset MNIST en su versión de entrenamiento. Diremos que M ha sido entrenada con d si la exactitud de predicción de M sobre las muestras de d en el dataset de MNIST en su versión de testeo son mayores a un umbral θ. Debido a que redes neuronales convolucionales simples son capaces detectar dígitos con una exactitud mayor al 99 % [28], podemos fijar de forma conservadora a θ en 0,9.

Para verificar la validez de este método, se entrenaron redes NetMNIST, tomando subconjuntos con $k \in \{1, 2, ..., 10\}$ cantidad de dígitos. Para cada k, se tomaron 5 subconjuntos al azar y sin repetición de k dígitos, para un total de 50 redes.

Para cada red, se hizo inferencia sobre la totalidad de las 10000 muestras de MNIST y se evaluó la exactitud por dígito.

4.1.2. Resultados y Discusión

Este simple método de clasificación logró predecir correctamente el 100 % de los dígitos como entrenados o no entrenados en las 50 redes.

Si bien el enfoque planteado en esta variante es simple y sus buenos resultados eran, en cierta medida, esperables, su valor radica en haber demostrado que es posible explotar el comportamiento observable de la red para inferir información sobre sus datos de entrenamiento, al menos teniendo un conocimiento previo sobre ella. Esta variante resultó fundamental como punto de partida, pues permitió establecer un esquema metodológico concreto que sirvió como base para las variantes siguientes.

4.2. Variante 2

Nuevamente resulta natural aprovechar el hecho de que la red ha sido entrenada con MNIST, sin embargo, resulta poco eficiente tener que procesar 10000 muestras para poder predecir los dígitos entrenados y no entrenados, como lo hicimos en la Variante 1.

Quisiéramos, entonces, ser capaces de usar las ideas de la variante anterior pero valiéndonos de una cantidad acotada de muestras. Esta restricción no es nueva y es la misma a la que se enfrentan científicos en áreas como estudios preclínicos animales, en donde por temas éticos se busca experimentar sobre la menor cantidad de animales posible [29].

El problema de contar con pocos datos ha sido estudiado a profundidad y se han utilizado herramientas de la estadística clásica para atacarlos. De esta forma, es interesante extrapolar nuestro problema al campo de la estadística.

4.2.1. Experimentación

Dada una red M_d podemos pensarla como una variable aleatoria binomial que predice correctamente la clase de una imagen que contiene un dígito d con una probabilidad p. $M_d \sim Bi(n, p)$ en donde una realización del experimento consiste en predecir una muestra del dataset, con 1 representando una correcta predicción, y 0 una incorrecta predicción. Podemos decir que M_d ha sido entrenada para predecir el dígito d si su probabilidad p correspondiente es alta.

Si contamos con la red, podemos generar realizaciones de su experimento pasándole imágenes y evaluando la exactitud de las predicciones. Con esta data podemos tratar de estimar p. Más aún, podemos plantear tests de hipótesis para testear con un cierto nivel de certeza si p se encuentra o no, por encima de un umbral.

Se propone el test exacto binomial:

$$H_0 : p \leq \theta$$
$$H_1 : p > \theta$$

H_0 representa que la red no fue entrenada con el dígito d, mientras H_1 rechaza esto y por lo tanto asume que sí. Para n cantidad de ensayos, k cantidad de aciertos (imágenes correctamente predichas), el p-valor del test queda determinado por $P(M_d \geq k \mid p = \theta)$.

Para medir la efectividad de este método, se entrenaron 50 redes NetMNIST usando subconjuntos de $k \in \{1, 2, ..., 10\}$ dígitos. Para cada k, se tomaron 5 subconjuntos al azar y sin repetición.

Se hizo inferencia en las redes, mostrando 40 imágenes por cada dígito posible. Con estas muestras se realizó el test de hipótesis para cada dígito. En total se realizó el test 500 veces, 10 veces por cada red entrenada.

En los test se usó un nivel de significancia del 0,05, con lo cual cada dígito se consideró usado para el entrenamiento si su p-valor fue menor o igual a 0,05.

4.2.2. Resultados y Discusión

Notemos que en la experimentación no fijamos un único θ. Resultados empíricos sobre redes de ejemplo mostraron una exactitud de predicción por dígito siempre mayor a 0,9, independientemente de la cantidad de dígitos usados para el entrenamiento. Así pues, resultaría intuitivo elegir $\theta = 0{,}9$.

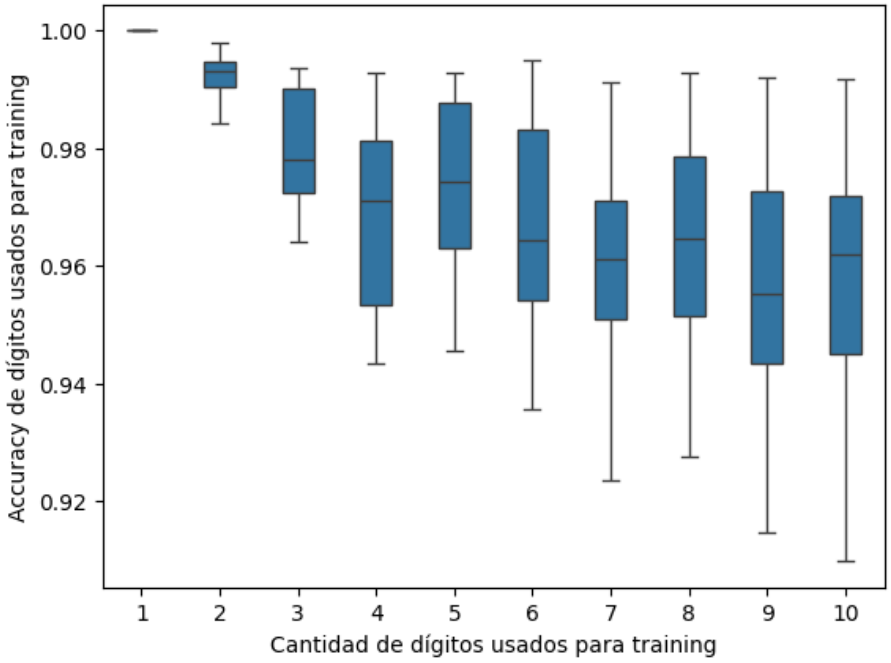


Fig. 4.1: Boxplots de la exactitud de predicción de imágenes por cantidad de dígitos usados para entrenamiento en 50 redes entrenadas con MNIST. Sólo se muestra la exactitud de los dígitos que fueron usados para el entrenamiento. Se puede observar que la mayor exactitud se obtuvo en las redes que usaron solo un dígito en su entrenamiento. A medida que aumenta la cantidad de dígitos usados, la exactitud tiende a disminuir pero siempre los valores están por encima de 0,9.

Sin embargo, en el test de hipótesis $\theta = 0{,}9$ genera un criterio excesivamente estricto. Esto se debe a que, para poder rechazar H_0 con un nivel de significancia de 0,05, se requiere que la red acierte prácticamente todos los ejemplos presentados. Por ejemplo, incluso obteniendo 39 aciertos sobre 40 muestras, el p-valor resultante es aproximadamente 0,07, lo que impide rechazar H_0. El test, en consecuencia, solo clasifica como entrenado a un dígito si alcanza 40 aciertos sobre 40, lo cual es una exigencia poco realista y reduce drásticamente el poder del test.

Para aumentar la sensibilidad, una alternativa razonable es reducir el valor de θ que se desea testear. Se exploró así, a posteriori, cómo cambiaban los resultados al variar θ entre 0,5 y 1,0

Sorpresivamente, el método propuesto tuvo una exactitud de predicción de dígitos entrenados mayor al 99 % para θ entre 0,5 y 0,7, indicando que aún usando una baja cantidad de imágenes por dígito, somos capaces de obtener altos niveles de predicción.

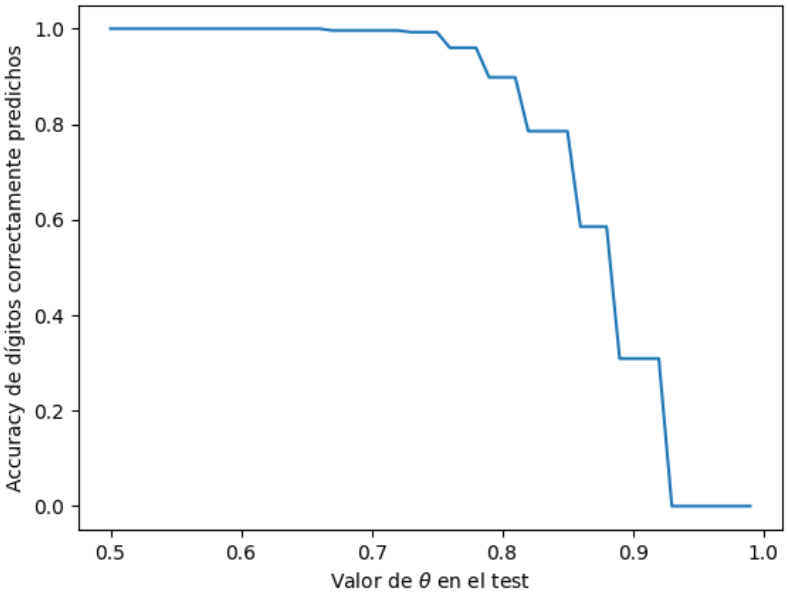


Fig. 4.2: Exactitud al predecir dígitos usados para entrenamiento en 50 redes. Se muestra el rendimiento del test propuesto al variar θ.

4.2.3. Conclusión

Si bien se logró obtener resultados positivos para nuestro objetivo, los resultados de esta experimentación deben interpretarse como un primer acercamiento al problema desde una perspectiva estadística. Aunque el test exacto de la binomial aporta un marco riguroso, se evidenció que no es una herramienta adecuada cuando se busca resolver el problema con una cantidad limitada de inferencias. Ajustar arbitrariamente el umbral θ para forzar buenos resultados va en contra del espíritu del análisis, y revela más bien una limitación del enfoque propuesto.

Además, el problema parece estar resuelto de manera trivial en muchos casos: ninguna red predijo correctamente un dígito con el que no fue entrenada, lo que permite resolver el problema simplemente observando si hay al menos una predicción correcta por dígito tras unas pocas inferencias.

Un camino más interesante para futuras exploraciones es analizar qué ocurre cuando se entrena con cantidades marginales de ciertos dígitos, sin llegar a que la red los aprenda completamente. Esto podría abrir la puerta a estrategias de detección más sutiles y realistas.

4.3. Variante 3

Es bien sabido que, en problemas de visión, las capas neuronales más profundas son capaces de codificar información más concreta y semánticamente relevante que capas superficiales e intermedias [11]. Así, un primer acercamiento a esta variante resulta de explorar los pesos en las últimas capas de la red.

4.3.1. Experimentación

En una red NetMNIST, la segunda capa Fully Connected (FC2) es la capa sobre la cual se aplica el SoftMax para la predicción. Es por esto que los pesos de FC2, gracias al mecanismo propio del backpropagation, terminan por relacionarse directamente con los dígitos aprendidos.

Análisis empíricos sobre los pesos de FC2 en redes NetMNIST entrenadas con diferentes dígitos, permitieron observar diferencias claras en los patrones de pesos ligados a dígitos entrenados. Como se observa en la Figura 4.3, los pesos relacionados a dígitos entrenados muestran mayor variabilidad. Estás diferencias permiten diferenciar "a ojo" aquellos dígitos que sí fueron entrenados.

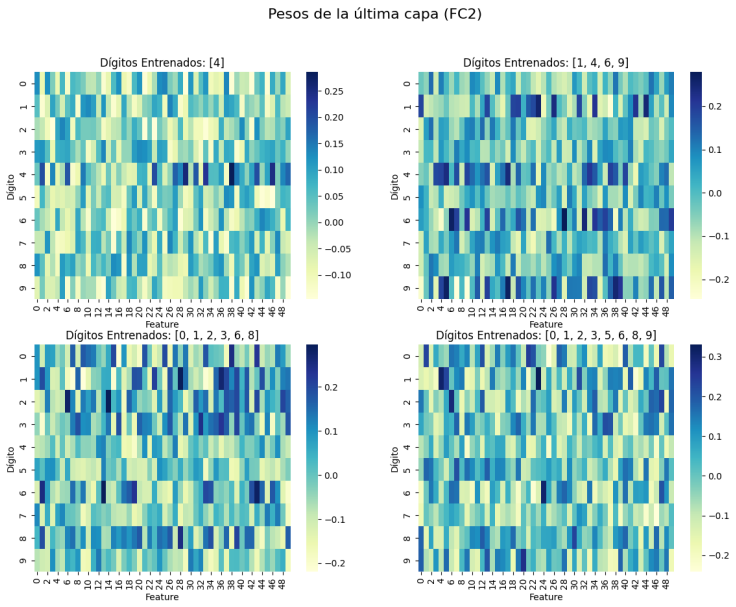


Fig. 4.3: Mapas de calor de los pesos de la última capa Fully Connected en 4 redes entrenadas usando 1, 4, 6 y 8 dígitos. Se puede observar que los pesos vinculados a dígitos entrenados tienden a tener pesos más altos, además, parecen aglomerarse pesos más altos en secciones adyacentes. En las redes entrenadas con 1 y 4 dígitos resulta clara la diferencia entre los dígitos entrenados y no entrenados, sin embargo, a medida que la cantidad de dígitos entrenados aumenta, no resulta tan clara y marcada la diferencia.

Queremos buscar una forma algorítmica de explotar estas diferencias observadas de forma automática. Para esto, se entrenaron 50 redes NetMNIST con $k \in \{1, 2, ..., 10\}$ dígitos. Para cada k, se tomaron 5 subconjuntos al azar y sin repetición de k dígitos distintos. Los pesos de las redes entrenadas fueron guardas juntos a los dígitos entrenados en cada red.

4.3.2. Resultados y Discusión

Tal como los análisis empíricos iniciales mostraron, se pudo observar que el promedio de los pesos relacionados a dígitos que fueron usados para el entrenamiento fue significativamente mayor al de los dígitos que no fueron usados para el entrenamiento. Además, los pesos en dígitos usados para entrenamiento mostraron una mayor varianza. En la Figura 4.4 se puede observar cómo cada categoría de dígito (entrenado, no entrado) tiende a formar clusters muy marcados.

Es posible crear "artesanalmente" un clasificador si se considera a un dígito como usado para el entrenamiento cuando la desviación estándar de sus pesos es mayor a 0,1. En nuestra experimentación esto logra una clasificación perfecta, indicado por el hecho de que todos los dígitos entrenados se encuentran por encima de la línea punteada en la Figura 4.4

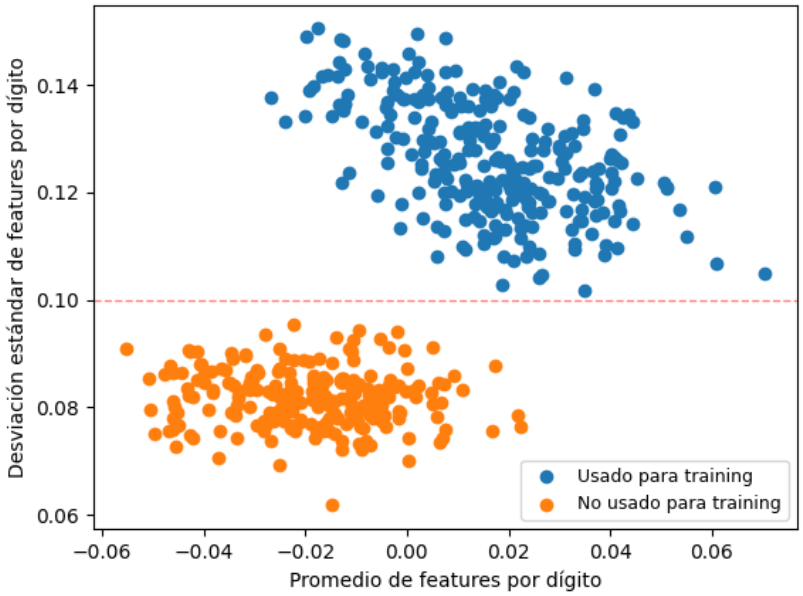


Fig. 4.4: Gráfico de dispersión del promedio de los pesos por dígito contra la desviación estándar de los pesos en dígitos usados y no usados para entrenamiento. Se puede observar que los dígitos usados para entrenamiento muestran promedios generalizadamente mayores y con desviaciones estándar más altas. Se observa un clustering muy marcado entre categorías

Se exploró utilizar el algoritmo de K-Means como forma de segmentar los dígitos en cada red. Para cada una de las 50 redes neuronales, se realizó K-Means en el espacio bidimensional de promedio y desviación estándar de sus pesos.

K-Means logró particionar correctamente todos los dígitos usados para entrenamiento en 42 de las 50 redes. Como es de esperarse el algoritmo falló en los casos donde los 10 dígitos fueron usados para entrenamiento pues K-Means requiere la cantidad de clusters como hiperparámetro. El algoritmo falló en los casos donde la cantidad de dígitos usados y no usados para entrenamiento era muy desbalanceada (como por ejemplo 9 dígitos usados para entrenamiento y 1 no usado).

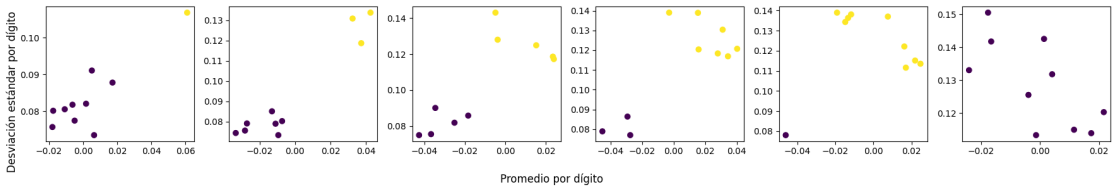


Fig. 4.5: Gráfico de dispersión del promedio de los pesos contra la desviación estándar de los pesos para dígitos usados y no usados para entrenamiento. Se puede observar que los dígitos usados para entrenamiento muestran promedios generalizadamente mayores y con desviaciones estándar más altas. Se observa un clustering muy marcado entre categorías

4.3.3. Conclusión

La exploración permitió comprobar de manera cuantitativa que existen diferencias estructurales marcadas en los pesos de salida asociados a clases entrenadas y no entrenadas, al menos en la capa FC2. En particular, se observó que los pesos correspondientes a dígitos utilizados durante el entrenamiento presentan tanto valores promedio como desviaciones estándar significativamente mayores, lo cual permite diferenciarlos con relativa facilidad.

Estos resultados validan la hipótesis de que el entrenamiento deja huellas detectables en los pesos de la red. El hecho de que sea posible realizar una clasificación perfecta de las clases entrenadas a partir de umbrales simples sobre estos estadísticos, e incluso lograr buenos resultados con métodos de clustering no supervisados como K-Means, nos muestra indicios de que el problema en general puede ser resuelto.

En cuanto a otros enfoques explorados, se intentó realizar clustering utilizando la proyección bidimensional de los pesos obtenida por PCA, pero los resultados fueron peores a los obtenidos directamente sobre las métricas de promedio y desviación estándar, lo que sugiere que estas últimas contienen de forma más explícita la información relevante para el problema.

Cómo formas de continuar el estudio de esta variante, se podría experimentar usando formas más sofisticadas de K-Means para que logre predecir los casos en donde todos o ningún dígito fue usado para entrenamiento. Se podría, por ejemplo, correr 2 K-Means con hiperparámetros de número de clusters fijados en 1 y 2 y evaluar si los centroides resultan cercanos.

4.4. Variante 4

Debido a que estudios preliminares realizados sobre la capa FC2 mostraron que los resultados obtenidos en la Variante 3 eran directamente replicables en esta variante, se decidió agregar como restricción adicional no poder tratar con las activaciones de FC2, sino sólo con capas menos profundas (es decir, solo hasta la capa FC1).

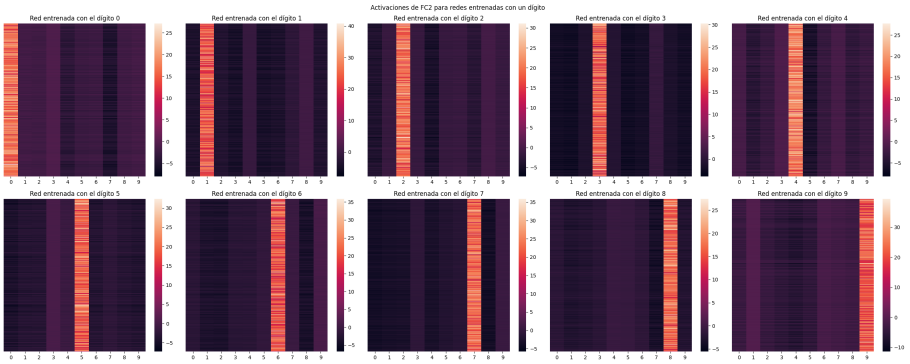


Fig. 4.6: Mapa de calor de la capa FC2 en 10 redes neuronales entrenadas con un dígito para una muestra de 1000 imágenes sobre MNIST en su versión de testeo. Se puede observar que en cada caso, el atributo correspondiente al dígito con el que se entrenó la red muestra activaciones muy superiores. Este resultado se mantiene aún para los casos en que la red se entrena con varios dígitos.

Restringirnos a no utilizar la capa FC2, que codifica la información semántica más relevante [11], y en su lugar trabajar con la capa FC1, agrega una enorme complejidad al problema. En el caso de redes NetMNIST significa pasar de tratar con vectores 10-dimensionales; a vectores 50-dimensionales. Además, se pierde la relación directa que existe entre la activación de la neurona y la selección de su clase correspondiente.

Visualizaciones de las activaciones en FC1 no permiten discernir relaciones claras.

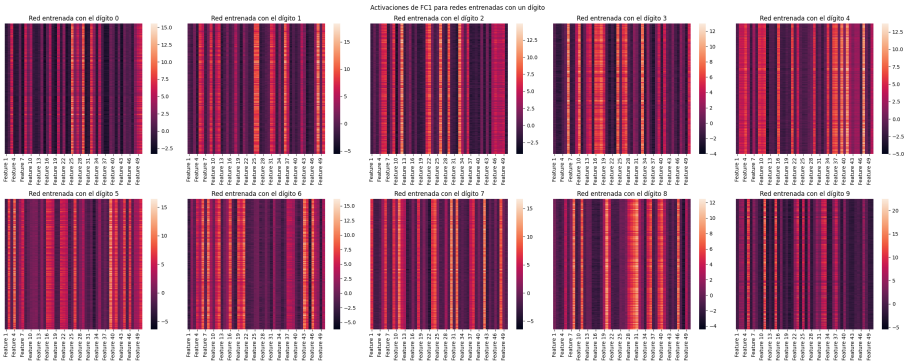


Fig. 4.7: Mapa de calor de la capa FC1 en 10 redes neuronales entrenadas con un dígito para una muestra de 1000 imágenes sobre MNIST en su versión de test. En este caso no se observa un claro patrón.

No es trivial, por lo tanto, que exista una relación directa entre las activaciones de FC1 y las clases entrenadas. Parte fundamental de la exploración es poder asegurar que tal relación existe.

Visualizaciones de proyecciones en PCA de los vectores de activación de FC1 en redes NetMNIST con todos los dígitos entrenados muestran un claro clusterizado por clase (Figura 4.8). Esto nos indica que la red aprende a transformar muestras de clases conocidas de formas similares y replicables [30].

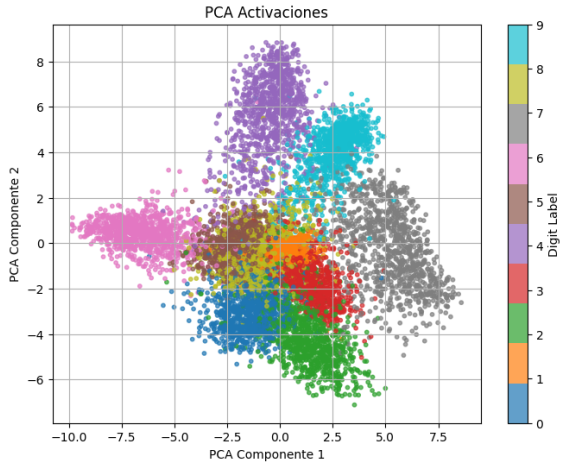


Fig. 4.8: Visualización en 2 componentes principales de las activaciones sobre todo el dataset de test de MNIST en una red NetMNIST entrenada con todos los dígitos.

Si nos fijamos en las activaciones mismas, se observa un patrón similar. La Figura 4.9 nos muestra que las activaciones de cada neurona tienden a ser similares al procesar muestras de una misma clase. Existen neuronas que generalizadamente se activan o apagan al ver muestras de una misma clase.

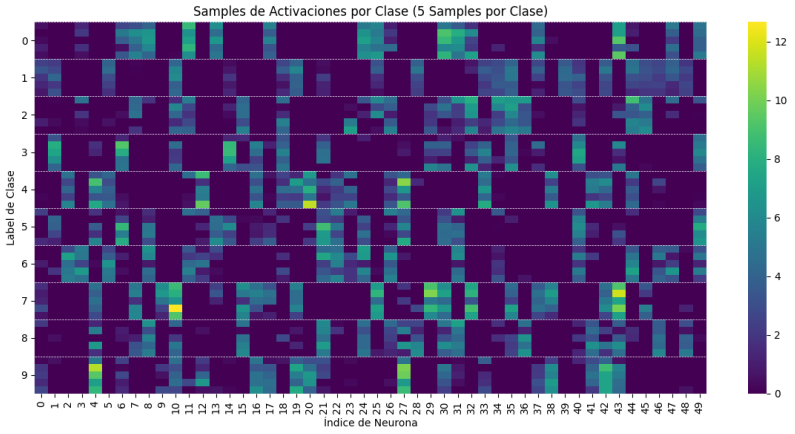


Fig. 4.9: Mapa de calor de activaciones en la capa FC1 para una red NetMNIST. Se muestran 5 imágenes de test por cada dígito (de 0 a 9), organizadas por clase en el eje vertical. El eje horizontal representa los índices de las 50 neuronas de la capa FC1. Cada celda indica la magnitud de activación de una neurona para una imagen dada. Puede observarse que distintas clases tienden a activar subconjuntos distintos de neuronas, lo que sugiere que la capa FC1 comienza a codificar representaciones discriminativas por clase.

Estos patrones evidencian una clara relación entre activación y clase. Así, resulta interesante explorar cómo cambia el comportamiento al incluir una clase no entrenada.

Al analizar las activaciones cuando se procesa una clase de control, conformada por muestras de ruido, se puede observar que las neuronas reaccionan de forma muy distinta: prácticamente ninguna neurona es excitada, y aquellas que sí, lo hacen con muy baja intensidad.

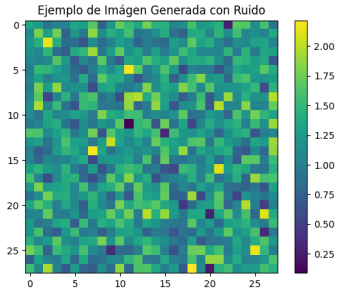


Fig. 4.10: Imagen generada con ruido, muestreando de una distribución uniforme entre 0 y 1 y posteriormente estandarizando con la media y desviación estándar de MNIST.

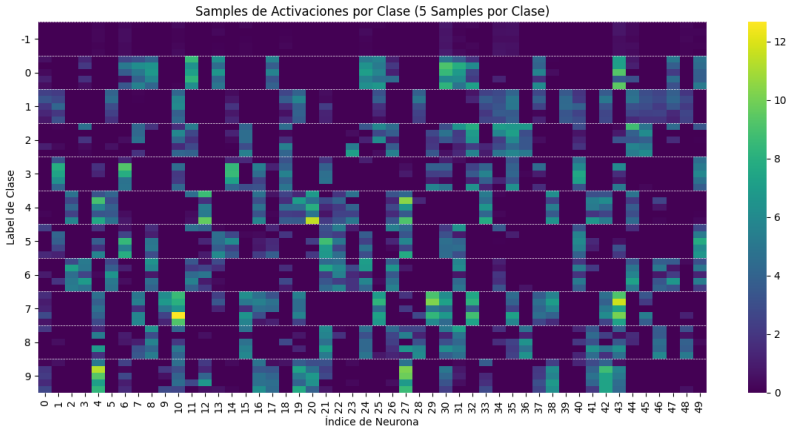


Fig. 4.11: Mapa de calor de activaciones en la capa FC1 para una red NetMNIST. La clase -1 representa muestras generadas con ruido. Se puede observar que, a diferencia de las otras clases que sí han sido entrenadas en la red, prácticamente ninguna neurona es activada.

Si bien estos resultados son prometedores, este comportamiento puede llegar a ser un efecto colateral de procesar imágenes sin estructura y vastamente distintas al dataset entrenado.

Al procesar muestras pertenecientes al dataset Fashion-MNIST en redes entrenadas para MNIST, nuevamente podemos observar que las neuronas tienden a activarse en mucha menor medida en comparación a las clases entrenadas, como se observa en la Figura 4.12

Si visualizamos la proyección bidimensional usando PCA de las activaciones de Fashion-MNIST, podemos observar que el clusterizado en gran medida desaparece: salvo por la clase 19, correspondiente a la etiqueta “bota”, las activaciones proyectadas se encuentran esparcidas sin concentraciones claras y con mucha superposición entre clases.

Tenemos ahora evidencia de que existen diferencias claras entre las activaciones de clases entrenadas y no entrenadas, al menos cuando estas representan conjuntos semánticamente diferentes. En la configuración de nuestra exploración, sin embargo, buscamos atacar el

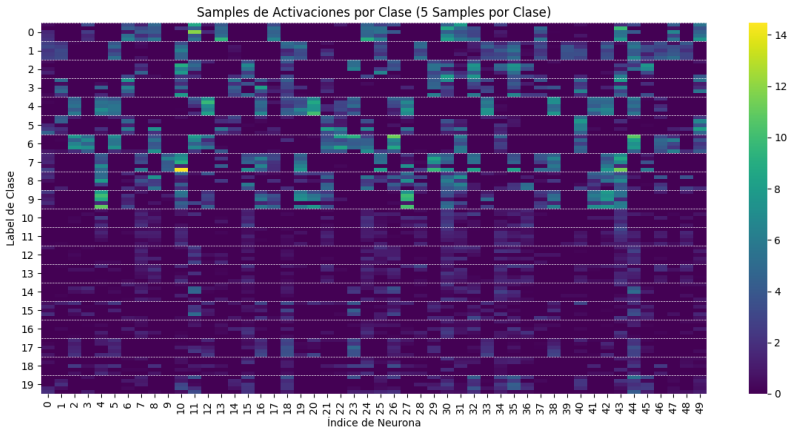


Fig. 4.12: Mapa de calor de activaciones en la capa FC1 para una red NetMNIST. Las clases del 0 al 9 corresponden a dígitos del dataset MNIST, mientras que las clases del 10 al 19 corresponden a clases del dataset Fashion-MNIST. Se observa que las activaciones tienden a formar patrones distintos entre los dos dominios, indicando que la red responde de manera diferenciada según el tipo de dataset.

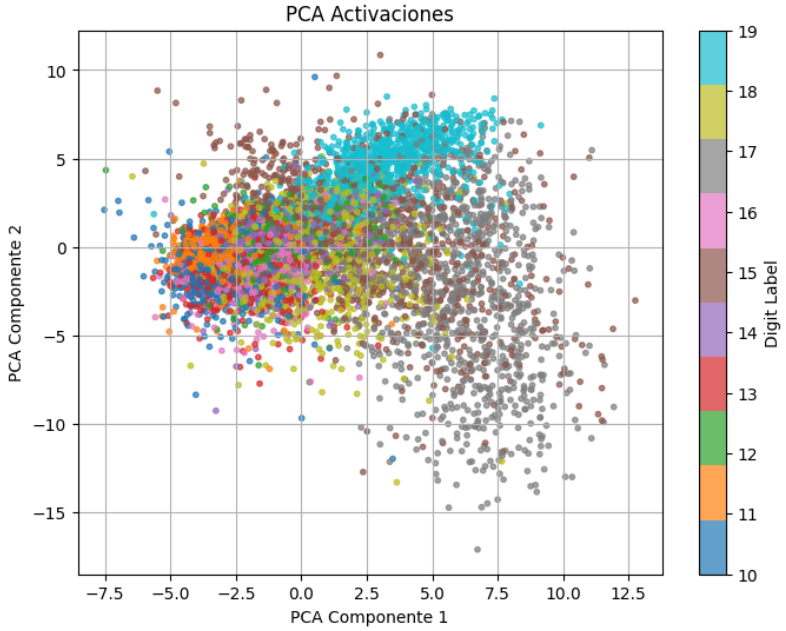


Fig. 4.13: Proyección bidimensional mediante PCA de las activaciones de la capa FC1. Se muestran las activaciones de imágenes pertenecientes a las clases 10 a 19, correspondientes al dataset Fashion-MNIST.

problema cuando la red ha sido entrenada únicamente con dígitos de MNIST y por lo tanto no existen diferencias tan drásticas entre clases entrenadas y no entrenadas.

En el caso en que la red es entrenada con todos salvo un dígito, las diferencias en activaciones son mucho menos marcadas, como muestra la Figura 4.14. Vemos así evidencia de que la red se comporta de manera distinta, y menos diferenciada, cuando la clase no entrenada es similar a las sí entrenadas.

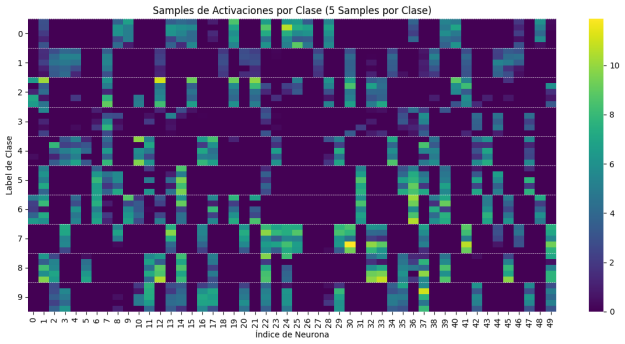


Fig. 4.14: Mapa de calor de activaciones en la capa FC1 para 5 muestras de cada clase del 0 al 9. En este caso, la red no fue entrenada con imágenes de la clase 3. Aunque puede observarse una menor activación promedio en dicha clase, las diferencias respecto de las clases entrenadas ya no son tan marcadas como antes.

Lee y Chae [31] atacaron un problema similar al que tratamos de resolver. Una métrica fundamental en su análisis fue el *Neuron Activation Ratio* (NAR), es decir la proporción de veces que una dada neurona fue activada (representado por una activación positiva gracias al ReLU). En general, se ha observado que, en ciertos escenarios, la proporción de activación puede reflejar o encapsular información funcionalmente relevante sobre la dinámica de la red o el procesamiento del estímulo [32] [33] [34]. Por lo tanto, una idea razonable es no enfocarnos en la intensidad de las activaciones, sino más bien en el NAR de las neuronas.

Sorprendentemente, al graficar las proporciones de activación se puede observar una diferencia clara en el dígito no entrenado (Figura 4.15). Las neuronas de la red, al observar muestras de una misma clase entrenada tienden a comportarse de forma homogénea: cada neurona o bien se activa siempre (reflejando una proporción alta) o nunca (reflejando una proporción nula). En el caso de clases no entrenadas, la red tiende a tener un comportamiento no uniforme, con lo cual cada neurona puede excitarse, o no, aún al procesar muestras de una misma clase. Este comportamiento lleva a un NAR con valores promedios, alejado tanto del 0 como del 1 y que es notablemente diferente a las proporciones que se obtienen en clases que sí fueron entrenadas.

Si filtramos para quedarnos únicamente con neuronas cuya proporción de activación se haya encontrado entre 0,2 y 0,8, observamos que la clase no entrenada cuenta con una cantidad de neuronas notablemente mayor que las clases entrenadas (Figura 4.16).

Habiendo podido observar diferencias claras en los valores y patrones de activación, usamos el filtrado de las proporciones de activación como las bases del método a explorar en esta variante.

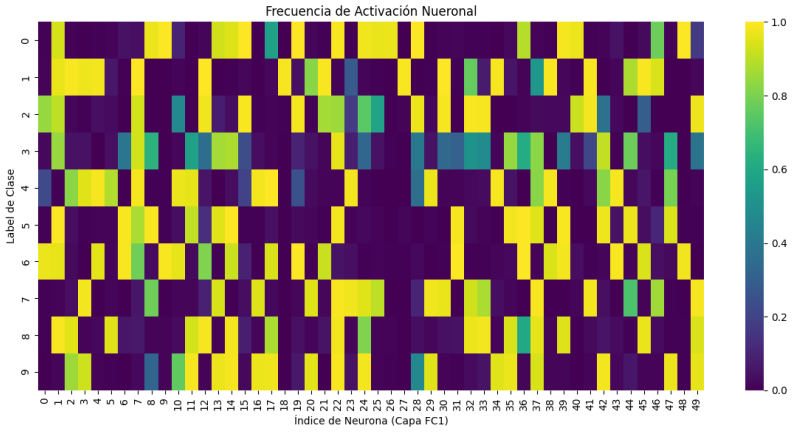


Fig. 4.15: Proporción de activación neuronal para cada clase del conjunto de test de MNIST, en una red entrenada con todos los dígitos excepto el 3. Se visualiza la proporción de imágenes que activaron cada neurona de la capa FC1. Se observa una clara diferencia en el comportamiento del dígito 3 con un proporción de activación menos extrema que la de otros dígitos sí entrenados.

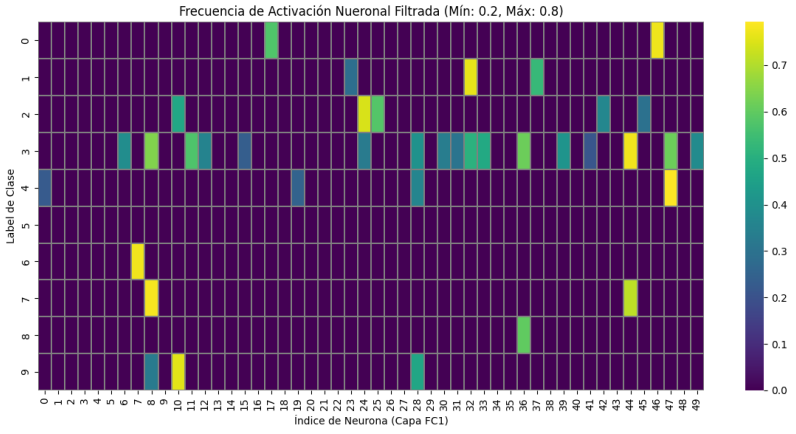


Fig. 4.16: Proporción de activación neuronal filtrada para cada clase del conjunto de test de MNIST, en una red entrenada con todos los dígitos excepto el 3. Se visualizan las neuronas cuya proporción de imágenes que activaron cada neurona en la capa FC1 estuvo entre 0,2 y 0,8. Se observa que el dígito 3 cuenta con 17 neuronas filtradas, mucho mayor que las otras clases sí entrenadas que cuentan con entre 0 y 5 neuronas filtradas.

4.4.1. Experimentación

Se busca utilizar la proporción de activación de neuronas filtradas para predecir automáticamente clases de dígitos entrenadas y no entrenadas.

Para cada cantidad de dígitos entrenados, k, con $k \in \{1, 2, ..., 9\}$ se entrenaron 10 redes NetMNIST usando subconjuntos distintos de k de dígitos, para un total de 90 redes entrenadas.

Para determinar las predicciones de dígitos entrenados se procesó una cantidad fija de muestras tomadas aleatoriamente de MNIST en su versión testeo. Para cada dígito posible se filtraron las neuronas en FC1 para mantener únicamente aquellas neuronas cuya proporción de activación estuviera entre 0,2 y 0,8. La proporción de las activaciones se realizó en base a muestras de una misma clase.

Se usó como estadístico la cantidad de neuronas que permanecieron después de hacer el filtrado. Con esto, una red entrenada produjo 10 estadísticos, uno por cada dígito posible representando la cantidad de neuronas que no fueron filtradas. Finalmente, sobre estos 10 estadísticos se realizó K-Means fijado en 2 clusters y se consideró aquel cluster con promedio de número de neuronas mayor como el cluster de dígitos no entrenado.

Adicionalmente, se exploró variar la cantidad de muestras usadas para generar las predicciones. Se experimentó hacerlo con 10, 50, 100, 250 y 1000 muestras tomadas al azar de MNIST.

4.4.2. Resultados y Discusión

Se obtuvieron resultados muy favorables para el método propuesto. Al generar las predicciones utilizando 1000 muestras, el método alcanzó un score F1 por dígito mayor a 0,9 para hasta 6 dígitos no entrenados, indicando un excelente equilibrio entre precisión y recall (Figura 4.17); el método logró identificar correctamente la mayoría de las clases no entrenadas, cometiendo muy pocos falsos positivos. Se obtuvo una exactitud por dígito mayor a 0,9 para hasta 5 dígitos no entrenados y mayor a 0,8 para hasta 7 dígitos no entrenados.

Si bien el método empeora al dejar 8 y 9 dígitos no entrenados, resulta notable que en todos los casos alcanza una precisión mayor al 0,8, y cercana a 1 para entre 2 y 8 dígitos no entrenados. Como buscamos predecir los dígitos no entrenados, esto se traduce en que aún para redes que han sido entrenadas con un conjunto de entrenamiento poco variado la red es capaz de detectar al menos los dígitos no entrenados en casi todos los casos.

Se analizó también el promedio de dígitos correctamente predichos por modelo utilizando 1000 muestras (Figura 4.18). Para hasta 6 dígitos no entrenados, el método fue capaz de predecir correctamente en entrenados o no entrenados a más de 9 dígitos. Esto muestra la misma tendencia anterior en donde el método se degrada al dejar muchos dígitos no entrenados.

Finalmente, en los análisis variando el número de muestras usadas para la predicción, se pudo observar un score F1 mayor al 0,8 al usar a penas 100 muestras y un score mayor al 0,9 al usar más de 250 muestras (Figura 4.19). Estos resultados indican que el método propuesto es capaz de obtener resultados favorables aún usando un bajo número de muestras. Estos resultados se relacionan directamente con los análisis realizados en

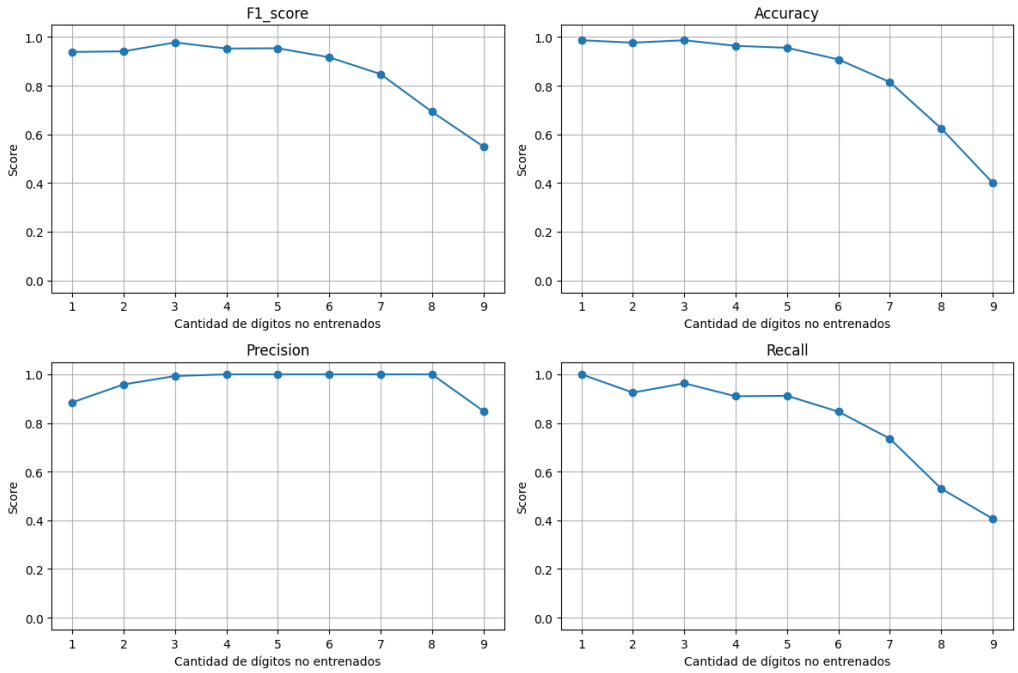


Fig. 4.17: Métricas de performance del método de filtrado por proporción de activación neuronal y K-Means al usar 1000 muestras de MNIST en su versión de testeo para generar las predicciones.

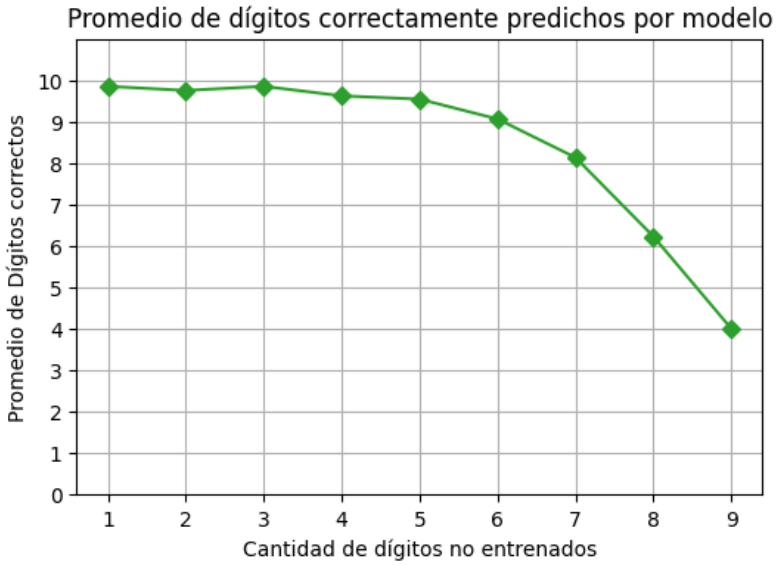


Fig. 4.18: Promedio de dígitos correctamente predichos en los 90 modelos usando el método de filtrado por proporción de activación neuronal y K-Means al tomar 1000 muestras de MNIST para generar las predicciones.

4.2 y muestran un acercamiento distinto a la problemática de usar pocos datos para la predicción.

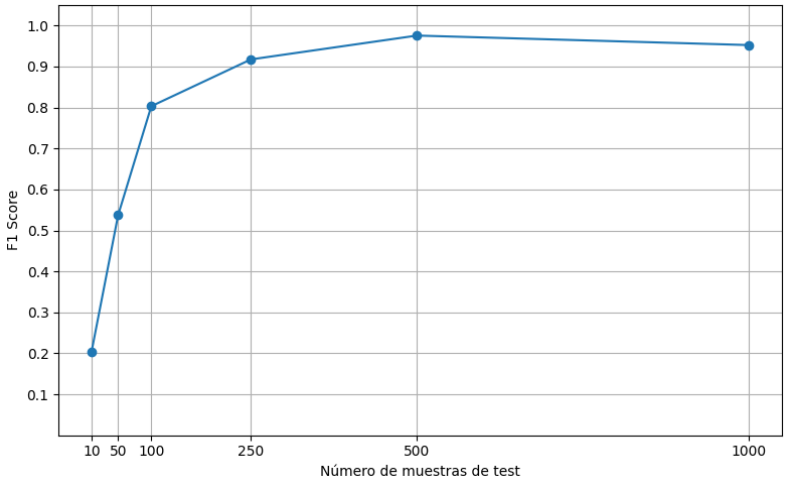


Fig. 4.19: Score F1 de dígitos correctamente predichos en los 90 modelos usando el método de filtrado por proporción de activación neuronal y K-Means al tomar una cantidad variable de muestras.

4.4.3. Conclusión

La exploración de esta variante permitió demostrar que, incluso sin acceso a los pesos ni a las salidas de la red, es posible inferir información relevante sobre las clases con las que fue entrenada una red. A través del análisis de activaciones en capas intermedias de redes NetMNIST, en particular la capa FC1, se identificaron patrones estructurales y estadísticos que permiten distinguir entre clases entrenadas y no entrenadas.

Se pudo observar que las neuronas se excitan en mucha mayor medida al procesar imágenes semánticamente relacionadas a sus clases entrenadas, mientras que las neuronas tienden a no activarse o bien hacerlo con poca intensidad al procesar clases muy distintas a las entrenadas.

En especial, se observó que la proporción de activación neuronal tiende a mostrar valores extremos (cercanos a 0 o 1) en clases entrenadas, mientras que en clases no entrenadas exhibe valores intermedios, reflejando una mayor variabilidad en el comportamiento de las neuronas. Las neuronas aprenden a comportarse de forma homogénea frente a muestras de clases entrenadas.

El método desarrollado, basado en el filtrado de neuronas según su proporción de activación y el posterior clusterizado mediante K-Means, mostró resultados altamente favorables. Con apenas 1000 muestras, se logró un score F1 superior a 0,9 para configuraciones con hasta 6 dígitos no entrenados, y una precisión consistentemente alta incluso en escenarios más adversos. Además, se comprobó que el método mantiene un desempeño razonable aun cuando se reduce significativamente el número de muestras utilizadas.

En conjunto, los resultados obtenidos en esta variante muestran que las activaciones internas de una red contienen suficiente información estructurada como para permitir inferencias sobre su conocimiento, incluso sin acceso explícito a sus parámetros.

Estos resultados introducen una estrategia novedosa que no había sido explorada previamente. La efectividad del método bajo condiciones tan restringidas sugiere un camino prometedor para investigaciones futuras, tanto en interpretabilidad como en auditoría de redes, especialmente en escenarios donde no se tiene acceso completo al modelo.

4.5. Variante 5

En 2015, Mordvintsev et al. [35] introdujeron DeepDream, una técnica de visualización en redes neuronales para clasificación de imágenes, en donde mediante la maximización de una o varias neuronas, se logran obtener patrones visuales que revelan características en una imagen que excitan particularmente a la red.

Si bien originalmente DeepDream fue concebido como una técnica artística, en tiempos recientes ha mostrado un gran poder como herramienta para la interpretabilidad. En este sentido, se busca explorar si podemos utilizar esta técnica para determinar el conocimiento aprendido de una red.

Buscamos analizar las imágenes resultantes de optimizar una neurona específica en la última capa de la red, FC2. FC2 es la capa sobre la cual se realiza el SoftMax y por lo tanto maximizar sus neuronas se corresponde a encontrar imágenes prototípicas o ejemplares de cada clase. Como ejemplo, si encontramos la imagen que maximiza la activación de la neurona en la posición 3 de FC2, responsable de clasificar una imagen como 3, esta imagen puede ser considerada como un ejemplo prototípico del 3 para la red.

Sea $A_n(x)$ el valor de activación que toma la n-ésima neurona de FC2 al procesar la imagen x. Comenzamos con x una imagen de 28×28 píxeles en blanco.

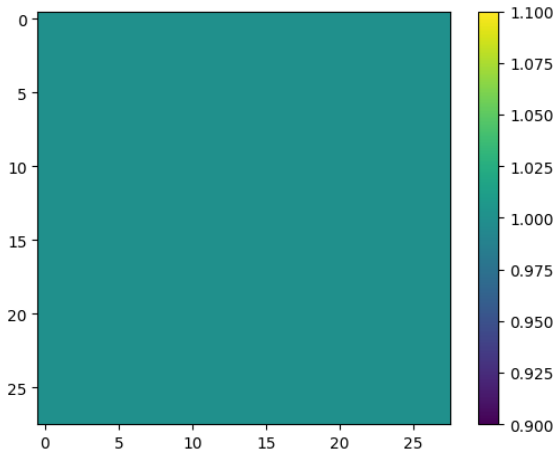


Fig. 4.20: Imagen de 28×28 píxeles de valor 1,0, reflejando una imagen en blanco.

En cada paso k, optimizamos la imagen original buscando maximizar la tercera neurona de FC2, moviéndonos en sentido del gradiente de $A_3(x_k)$. Realizamos este procedimiento 100 veces, con una tasa de aprendizaje de 0,05. El resultado es una imagen que muestra rasgos característicos del dígito 3 que parecen repetirse a través de toda la imagen (Figura 4.21).

Si bien somos capaces detectar a ojo las similitudes con el dígito 3, esto puede no ser inmediatamente obvio sin tener el conocimiento previo de cuál dígito se optimiza. Más aún, en ciertos dígitos este proceso no genera imágenes tan evidentes. Para tratar de remediar esto, podemos introducir un término regularizador que penaliza la Variación Total (VT) [36]. Maximizamos entonces $A_3(x) - \lambda_1 \cdot VT(x)$, donde λ_1 es un coeficiente de regularización que fijamos en 50.

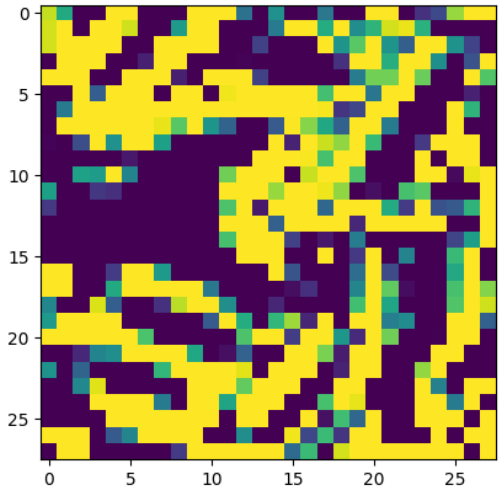


Fig. 4.21: Imagen resultante de optimizar sin regularización 100 veces la tercera neurona de FC2 con una tasa de aprendizaje de 0,05.

Al realizar la optimización 100 veces, comenzando nuevamente con una imagen en blanco y con la misma tasa de aprendizaje, observamos una imagen con bordes suaves y sin tanto ruido, en donde el 3 se distingue con mayor facilidad.

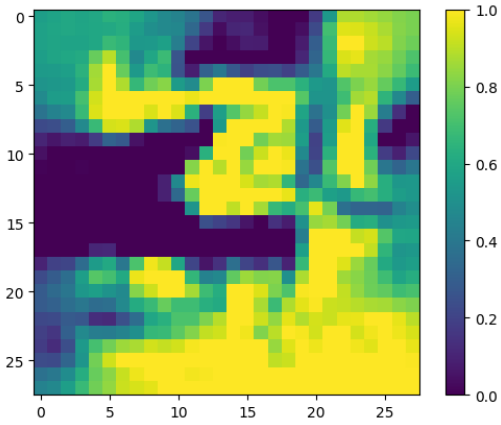


Fig. 4.22: Imagen resultante de optimizar con regularización de Variación Total 100 veces la tercera neurona de FC2 con una tasa de aprendizaje de 0,05.

Podemos ir un paso más allá y usar el hecho de que en MNIST las imágenes tienden a tener una gran de cantidad de píxeles apagados (representados por color negro en los lugares fuera del trazo del dígito). En base a esto, podemos agregar un nuevo termino regularizador que penalice la norma $L2$ de la imagen, buscando priorizar imágenes que tengan menos cantidad de píxeles encendidos. Esta regularización tiende a generar imágenes más suaves globalmente y llevar valores pequeños a 0, como observamos en la Figura 4.23 [37].

Maximizamos ahora $A_3(x) - \lambda_1 \cdot VT(x) - \lambda_2 \cdot ||x||_2^2$ con $\lambda_1 = 50, \lambda_2 = 10$.

Obtenemos una imagen mucho más clara del dígito, en donde el ruido ha disminuido considerablemente y la imagen se asemeja a muestras reales del dataset.

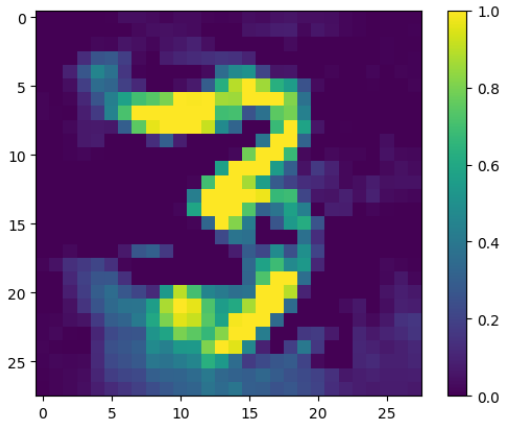


Fig. 4.23: Imagen resultante de optimizar con regularización de Variación Total y norma $L2$ 100 veces la tercera neurona de FC2 con una tasa de aprendizaje de 0,05.

Finalmente, podemos agregar un posprocesado aplicando una máscara gaussiana con $\sigma = 7$, que atenúa los valores del centro de la imagen y reduce los valores en los bordes. Este procesado tiene sentido pues en las muestras de MNIST, los píxeles no nulos tienden a encontrarse en el centro de la imagen.

Una *máscara gaussiana* consiste en una matriz de pesos definidos por la función de densidad de una distribución normal bivariada:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Esta máscara se aplica multiplicando punto a punto los valores de la imagen, lo que genera un efecto de "enfoque central", suavizando progresivamente las regiones extremas.

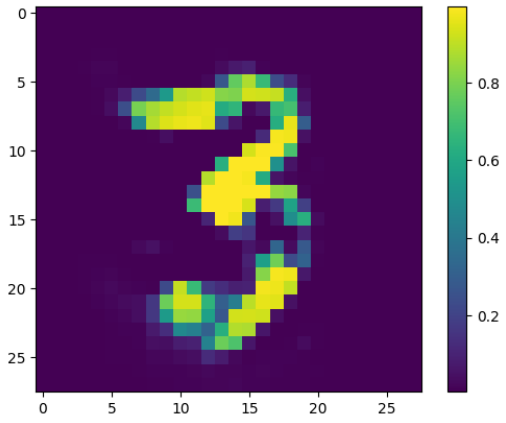


Fig. 4.24: Imagen resultante de optimizar con regularización de Variación Total y norma $L2$ 100 veces la tercera neurona de FC2 con una tasa de aprendizaje de 0,05. El resultado de la optimización es posprocesado con una máscara gaussiana.

El resultado es una imagen clara y de trazado mayormente continuo, sin ruido por fuera del trazo. Esta optimización genera una imagen prácticamente indistinguible de muestras

reales del dataset.

Lo que hemos logrado es realmente notable. Hemos comenzado con una red de la cual conocíamos únicamente que ha sido entrenada con MNIST y hemos podido generar un ejemplo realista, prototípico de una de sus clases entrenadas. Con este método hemos podido extraer el conocimiento aprendido de la red usando pocas suposiciones previas.

Este método funciona cuando optimizamos cada una de las neuronas que clasifican los diferentes dígitos de MNIST y permite recrear imágenes que al, ojo humano, nos permiten inferir con qué dígitos ha sido entrenada la red.

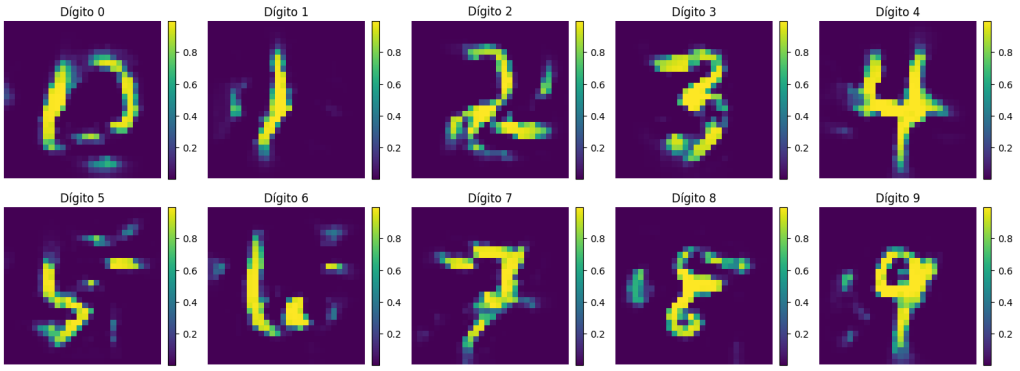


Fig. 4.25: Imágenes resultantes de aplicar el método con regularizaciones de VT, norma $L2$ y posprocesado en cada una de las neuronas de FC2 en una red NetMNIST que ha sido entrenada con todos los dígitos. Se parte de una imagen en blanco y se itera 100 veces con una tasa de aprendizaje de $0,05$. El resultado son imágenes similares a los dígitos reales que pueden ser interpretados y entendidos por el ojo humano. En algunos casos se observan pequeñas secciones con ruido que no ha podido ser eliminado con los términos regularizadores ni el posprocesado.

Como nuestro objetivo original es identificar dígitos no entrenados, un siguiente paso es tratar de aplicar el método de maximización de activaciones en redes que no han sido entrenadas con todos los dígitos.

Se exploró aplicar el método con una optimización más laxa, regularizando únicamente la Variación Total y no la norma $L2$. Al aplicarlo en una red que fue entrenada con todos los dígitos, excepto el 6, se puede observar una recreación fiel de cada dígito, a excepción del 6, en donde se observa ruido indistinguible en el centro de la imagen.

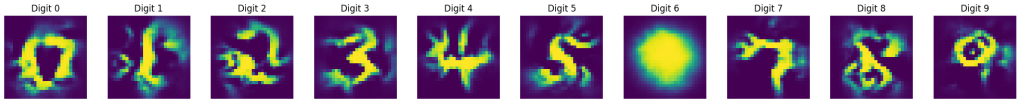


Fig. 4.26: Imágenes resultantes de aplicar el método con regularizaciones de VT y posprocesado en una red NetMNIST que ha sido entrenada con todos los dígitos salvo el 6.

Que los bordes sean blancos tiene sentido debido al posprocesado. El hecho de que los píxeles activos no tengan forma alguna reflejan lo que se esperaría: como la red nunca vio imágenes que tuvieran al 6 como etiqueta, nunca aprendió a distinguir sus rasgos característicos; no aprendió a alinear sus pesos de forma tal que sus neuronas en FC2 reaccionen favorablemente a imágenes del 6.

De alguna forma, este método resuelve nuestro problema en su totalidad, y deja en evidencia clara el conocimiento que ha sido aprendido por la red.

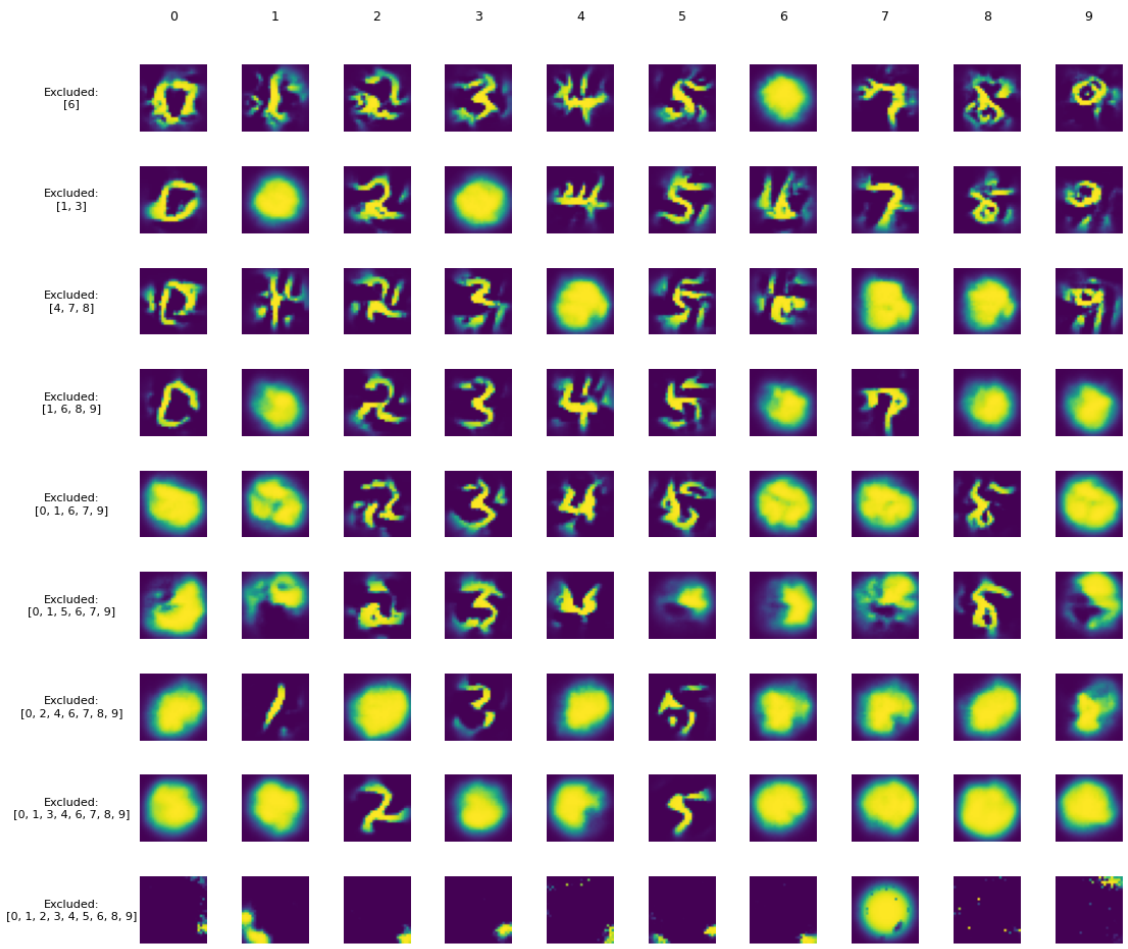


Fig. 4.27: Resultados de aplicar el método en 9 redes distintas que han sido entrenadas usando $k \in \{1, 2, ..., 9\}$ dígitos. Las filas representan redes distintas, mientras que las columnas representan el dígito optimizado. En cada caso se puede diferenciar claramente los dígitos no entrenados pues el resultado son imágenes con manchas deformes e irregulares en el centro, a diferencia de imágenes con estructura clara en el caso de dígitos entrenados.

En la Figura 4.27 podemos observar cómo, a ojo, somos capaces de fácilmente detectar los dígitos no entrenados, aún el caso en que 9 de 10 dígitos se han dejado sin entrenar. A diferencia de los resultados encontrados en la Variante 4, este método funciona igual de bien cuando aumenta la cantidad de dígitos no entrenados.

Es notable la última fila, representando los resultados en una red que ha sido entrenada únicamente con el 7. En esta red, la optimización del dígito 7 arroja una mancha similar a la obtenida para dígitos no entrenados en las otras redes. Si bien la notable distinción con todos los otros dígitos nos permiten inferir que el 7 es el dígito entrenado, la reconstrucción

obtenida por el método no genera una imagen con estructura clara. Esto es sumamente interesante pues nos abre la puerta a interpretar el conocimiento específico adquirido: dado que en su entrenamiento, la red nunca se enfrentó a alguna imagen distinta al 7, nunca fue necesario aprender a distinguir estructuras semánticas; la red considera a cualquier imagen con pixeles encendidos como un 7.

Si bien el método parece ser muy bueno al usarlo con nuestra interpretación humana propia, en muchos casos es deseable poder encontrar métodos algorítmicos que puedan ser puestos a prueba de forma automática. En esta línea, queremos explorar cómo una red procesa internamente las imágenes optimizadas y cómo se compara con el procesado de muestras reales.

Enfocamos la exploración en la capa FC1, pues esta tiene embeddings de mayor dimensión que FC2 y que pueden potencialmente codificar información semántica más rica.

La idea es retroalimentar la red con la imagen obtenida al optimizar la activación de alguna neurona en FC2 y capturar el embedding de activación producido en FC1. Posteriormente comparamos los embeddings del ejemplo de clase generado con los embeddings obtenidos al procesar muestras reales de diferentes clases y datasets.

Partimos de optimizar la neurona número 4, bajo las mismas condiciones que antes.

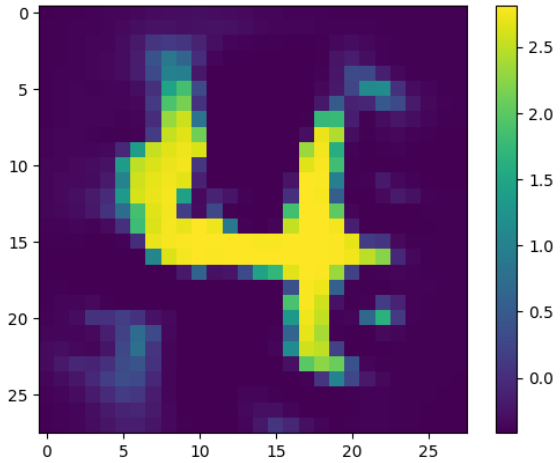


Fig. 4.28: Imagen obtenida al optimizar 100 veces la neurona del 4 en FC2, con regularización de VT y con posprocesado.

Posteriormente tomamos aleatoriamente 100 muestras por clase en cada una de las clases de los datasets MNIST, Fashion-MNIST y KMNIST, para un total de 3000 muestras.

Las 3000 muestras son alimentadas a la red y los embeddings obtenidos en FC1 guardados. Finalmente guardamos las similitudes coseno resultantes de comparar los embeddings del ejemplo prototípico del 4, con cada unos de los otros 3000.

Como se esperaría, la similitud mediana correspondiente a la clase del 4 es la mayor, con una clara diferencia por sobre todas las otras clases (Figura 4.29).

Para validar la robustez del método, podemos comparar las similitudes obtenidas al procesar un ejemplo adversario, en donde la imagen obtenida en la optimización es distinta a

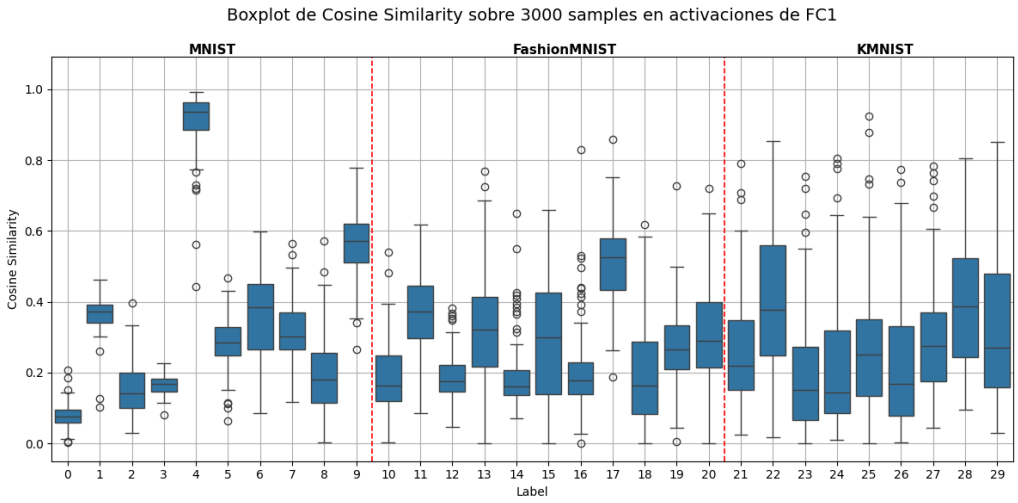


Fig. 4.29: Boxplot de las similitudes coseno entre embeddings en FC1 del ejemplo de 4 prototípico y 3000 muestras de MNIST, Fashion-MNIST y KMNIST. La mayor similitud es la de la clase del 4. Las similitudes con KMNIST tienen una varianza generalizadamente mayor.

las encontradas en el dataset y no tan fácilmente distinguible al ojo humano.

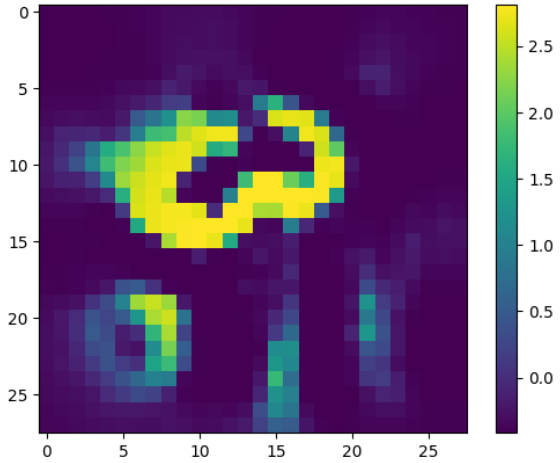


Fig. 4.30: Imagen obtenida al optimizar 100 veces la neurona del 9 en FC2, con regularización de VT y con posprocesado. En esta red particular, se obtiene una imagen que no es inmediatamente distinguible como un 9.

Aún en este caso, la similitud con la clase del 9 es la mayor con diferencia (Figura 4.31).

Esto nos indica una característica notable en nuestro método: si bien en la mayoría de los casos somos capaces de inferir resultados apelando al ojo humano, las imágenes generadas por el método no sólo maximizan las activaciones, sino que además llevan a la red a comportarse de forma similar a cuando procesa muestras reales de la clase optimizada. En otras palabras, aunque una imagen generada no sea visualmente similar a muestras de la clase, esta es procesada e interpretada por la red de la misma forma que otras muestras reales. El método está logrando extraer de la red una imagen que la misma "entiende"

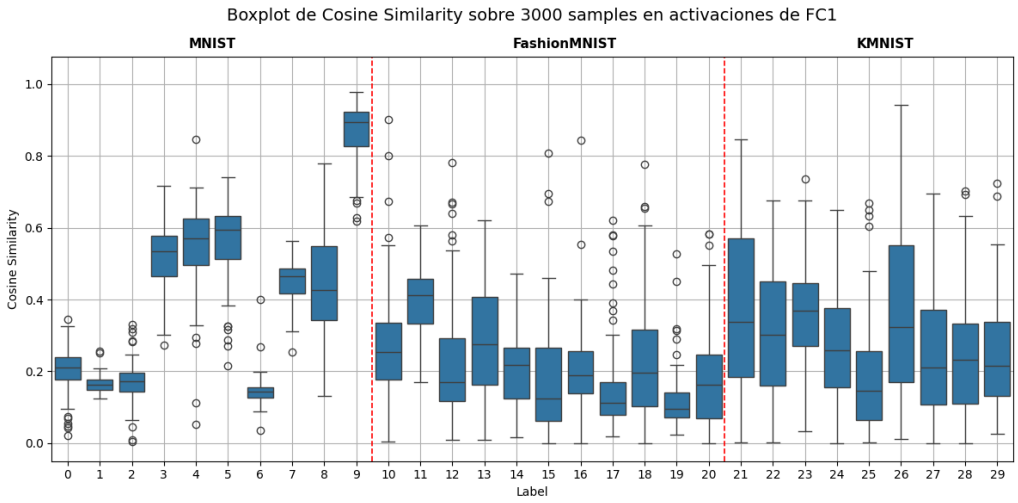


Fig. 4.31: Boxplot de las similitudes coseno entre embeddings en FC1 del ejemplo de 9 prototípico y 3000 muestras de MNIST, Fashion-MNIST y KMNIST. Si bien el prototipo no es fácilmente interpretado como un 9, las similitudes con la clase del 9 son las mayores.

como un 9, independientemente si para nosotros lo es. Esto nos abre las puertas explorar a el conocimiento de la red de forma automática, como veremos a continuación.

4.5.1. Experimentación

Vamos a evaluar la exactitud de nuestro método, esta vez sobre un caso que podría ser más natural en la vida real. Vamos a experimentar con redes NetMNIST en donde la capa FC2 se ha reducido para tener $e \in \{1, 2, ..., 9\}$ neuronas y por lo tanto son entrenadas para clasificar e dígitos.

En total se entrenaron 90 redes NetMNIST reducidas, 10 para cada e posible. En cada caso los dígitos entrenados se tomaron de forma aleatoria y sin repetición. Para cada red se realizó una asignación aleatoria entre dígito y neurona en FC2. El mapeo dígito-neurona fue guardado y utilizado posteriormente para medir los resultados predichos.

En cada caso, se realizó el método descrito, con regularización de Variación Total con coeficiente $\lambda_1 = 50$ y posprocesado con máscara gaussiana de $\sigma = 10$, sin regularización $L2$. Cada prototipo fue generando en 100 iteraciones con una tasa de aprendizaje de 0,05.

Para determinar los dígitos entrenados, se optimizaron las neuronas de la red y se compararon las activaciones sobre todos las muestras de MNIST en su versión de testeo. En cada neurona, se tomó como el dígito entrenado a aquella clase cuya similaridad coseno fuera la de mayor mediana. Así, se tomo como dígitos no entrenados a los dígitos que no fueron asignados para ninguna neurona.

4.5.2. Resultados y Discusión

Se obtuvieron resultados sumamente favorables para el método propuesto. El método logró un score F1 de clasificación mayor al $0,9$ para cada cantidad de dígitos no entrenados y un score mayor al 0,95 cuando se entrenó más de un dígito.

Resultados similares se observaron tanto para la exactitud como la sensibilidad (*recall*). Similarmente, se obtuvo una precisión mayor al $0,9$ en todos los casos, indicando que la red detecta los dígitos no entrenados con mucha eficacia.

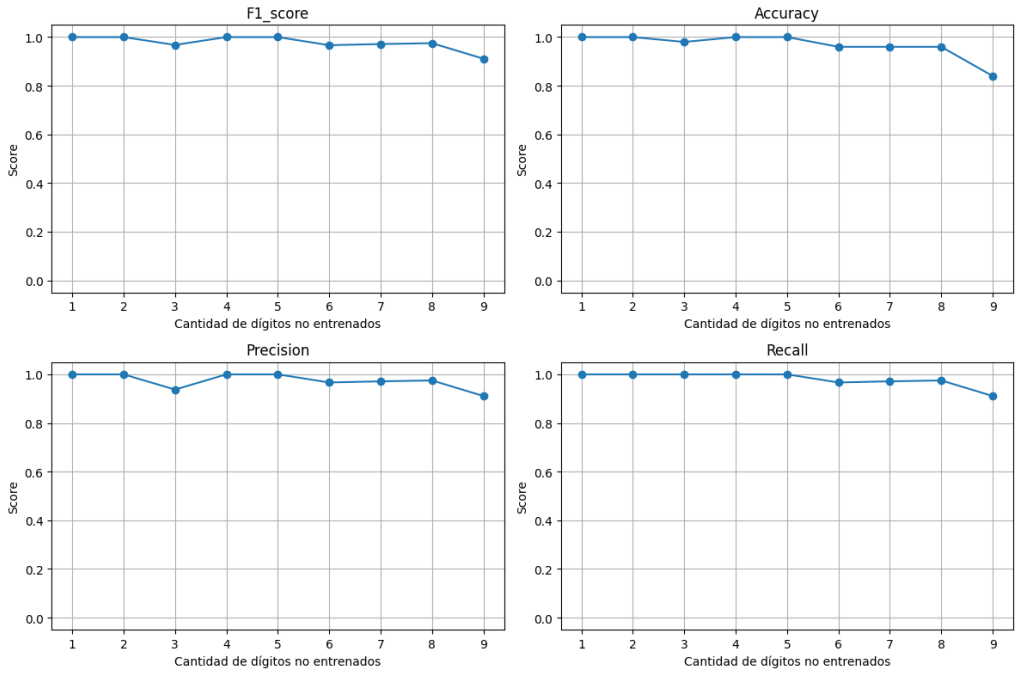


Fig. 4.32: Métricas del método usando DeepDream y la similaridad coseno sobre la totalidad de las muestras de MNIST en su versión de testeo para predecir dígitos en redes NetMNIST reducidas.

Si comparamos el promedio de dígitos correctamente predichos por modelo, observamos que la red fue capaz de predecir, en promedio, más de 9 dígitos correctamente en los casos en la que la red fue entrenada con más de un dígito. En los casos en que la red fue entrenada con un dígito, el método predijo en promedio más de 8 dígitos correctamente.

Los resultados del método mejoran a los obtenidos en la Variante 4 en todas las métricas, aunque debe considerarse que ambos métodos se evaluaron sobre arquitecturas distintas (NetMNIST reducida y NetMNIST).

Los resultados obtenidos destacan, además de la efectividad del método, que este es en general invariante a la cantidad de dígitos entrenados, mostrando resultados similares al aumentar la cantidad de dígitos no entrenados, aunque degradándose levemente cuando solo 1 dígito es usado para el entrenamiento. Estos resultados parecen estar alineados a los encontrados en la Figura 4.27, en donde buenos prototipos pueden ser generados siempre que se utilice mas de un dígito para el entrenamiento, pero se llegan a malas representaciones cuando sólo un dígito es utilizado para el entrenamiento.

4.5.3. Conclusión

Los resultados obtenidos en esta variante muestran que es posible inferir con gran exactitud los dígitos con los que una red ha sido entrenada, utilizando únicamente su estructura interna y el método de maximización de activaciones conocido como DeepDream. A

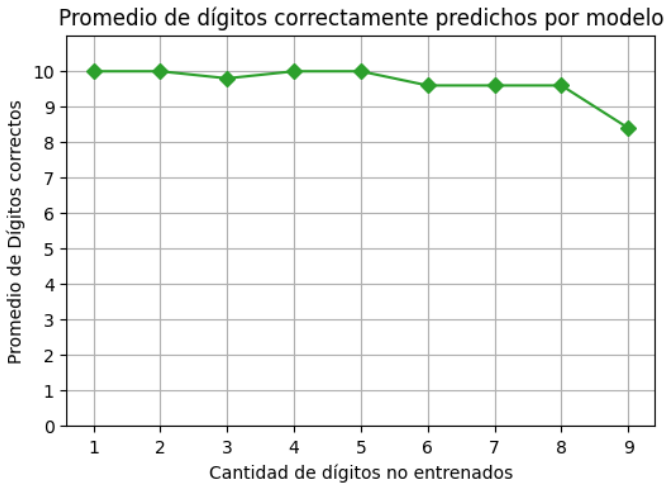


Fig. 4.33: Promedio de dígitos correctamente predichos en los 90 modelos NetMNIST reducido usando el método sobre la totalidad de MNIST.

través de la incorporación de regularizaciones simples y un posprocesado, logramos generar imágenes prototípicas por clase que, incluso en ausencia de conocimiento previo, permiten identificar visual y algoritmicamente el conocimiento aprendido por la red.

La principal fortaleza del método reside en su capacidad para funcionar de forma estable aún cuando aumenta la cantidad de dígitos no entrenados, mostrando una degradación mínima incluso en los casos más extremos. Esto sugiere que el enfoque no depende fuertemente de las condiciones particulares del entrenamiento, sino que aprovecha propiedades estructurales generales de las redes.

Además, la similitud utilizando embeddings internos refuerza la idea de que las imágenes generadas no sólo son visualmente interpretables, sino también funcionalmente coherentes con el comportamiento de la red. En otras palabras, aunque el resultado no siempre sea reconocible para un humano, la red las procesa como ejemplos legítimos de su clase y que abren la posibilidad de compararse con muestras reales, dentro y fuera del dataset entrenado.

Estos resultados posicionan al método como una estrategia sólida para la extracción de conocimiento en redes de clasificación, en donde no solo somos capaces de predecir clases no entrenadas, sino que además podemos observar cómo la red interpreta y procesa internamente sus clases aprendidas.

Si bien en esta variante exploramos redes artesanales como NetMNIST y datasets simples como MNIST y sus variantes, la realidad es que el método aquí encontrado es completamente agnóstico a arquitectura y dataset de entrenamiento con lo cual se abren nuevas vías de exploración para extender el enfoque hacia arquitecturas más complejas o dominios con estructuras semánticas menos evidentes.

4.6. Variante 6

Una forma intuitiva de comenzar la exploración es tratar de replicar los resultados obtenidos en la Variante 5, pero utilizando CIFAR10 en ResNet20. CIFAR10, a diferencia de MNIST, es un dataset con imágenes RGB de 3 canales. Así pues, resulta más natural comenzar la optimización neuronal con imágenes generadas por ruido muestreado de variables aleatorias uniformes. Con el fin de alimentar a la red con imágenes en el rango esperado, y así producir mejores resultados, podemos estandarizar la imagen con la media y desvío estándar de CIFAR10.

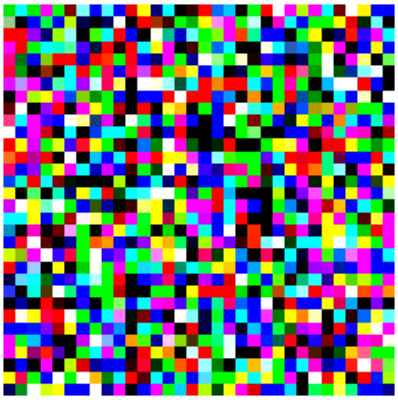


Fig. 4.34: Imagen resultante de generar una imagen RGB de 32×32 píxeles muestreando una variable aleatoria uniforme en el rango $(0, 1)$ y posteriormente estandarizando con la media y desvío estándar de CIFAR10.

Podemos aplicar la optimización descrita en la Variante 5, buscando maximizar alguna de las neuronas de la capa FC, la última capa antes del SoftMax en ResNet20. Aplicamos la optimización durante 300 iteraciones, regularizando la Variación Total con un coeficiente esta vez de 300. Como al trabajar con CIFAR10, y no MNIST, las imágenes no tienen la condición de pixeles nulos en los bordes, ni de baja intensidad de píxeles a través de la imágenes, ya no tiene sentido agregar el regularizador de la norma $L2$, ni de aplicar el posprocesado gaussiano.

Realizamos este método sobre la neurona relacionada con la clase Avión en CIFAR10.

Desafortunadamente, la imagen optimizada ya no refleja claramente la clase relacionada a su neurona. A ojo, no somos capaces de detectar de qué tipo de objeto se trata. Se observan líneas verticales y diagonales, y colores específicos en diferentes secciones de la imagen, como se observa en la Figura 4.35.

La buena noticia es que la líneas observadas reflejan formas características en las imágenes de aviones dentro del dataset.

En efecto, si bien nuestro método no es capaz de generar reproducciones fieles e interpretables como muestras reales dentro del dataset entrenado, este sí es capaz de capturar formas que nos brindan un entendimiento general sobre qué tipo de rasgos excitan a sus neuronas.

Otro ejemplo interesante es el que sucede con la clase gato. Al analizar la imagen optimizada podemos observar curvas rojas repartidas a través de todo el espacio. Estas curvas

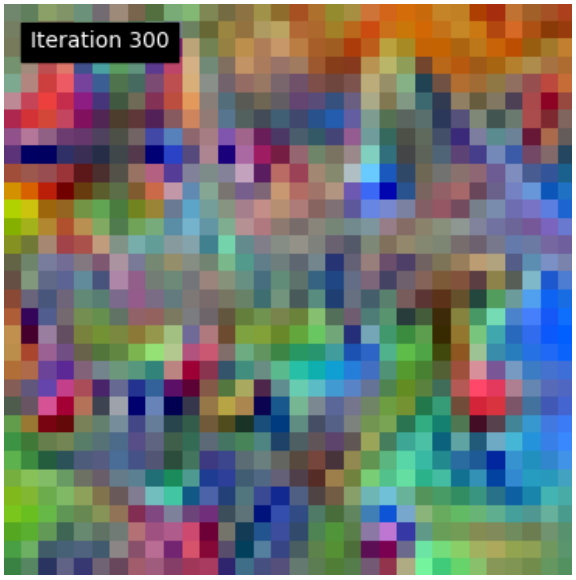


Fig. 4.35: Imagen resultante de optimizar la neurona relacionada a la clase Avión en CIFAR10 con regularización de Variación Total, partiendo de una imagen aleatoria durante 300 iteraciones.

Fig. 4.36: Ejemplo de imagen de Avión en CIFAR10. Se observan bordes rectos horizontales y diagonales, similares a los encontrados en la imagen optimizada.

parecen ser las siluetas de la cara de un gato, con los bordes superiores de la curva representando sus orejas. Esto nos indica que la red ha aprendido a relacionar imágenes de gatos con sus distintivas orejas. Más aún, al procesar una imagen, la red busca caras de gatos y sus orejas por toda la imagen; estos son las rasgos que excitan a la neurona en FC.

Fig. 4.37: Ejemplo de imagen de Gato en CIFAR10 junto con la imagen resultante de optimizar la neurona que clasifica gatos. En la imagen optimizada, se puede observar siluetas rojas que parecen representar las caras de un gato con sus orejas.

Algo similar sucede con la clase Auto, en donde se pueden observar formas circulares a lo largo de la imagen. Estas formas representan las ruedas de los autos y nos indican que la red busca ruedas por toda la imagen como forma de clasificarla, o no, en auto.

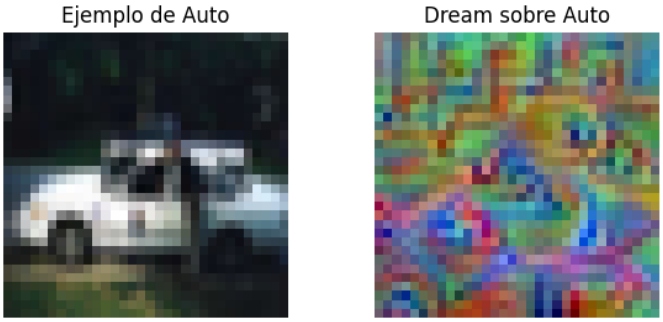


Fig. 4.38: Ejemplo de imagen de Auto en CIFAR10 junto con la imagen resultante de optimizar la neurona que clasifica autos. En la imagen optimizada, se puede observar siluetas circulares que parecen representar las ruedas de un auto.

Hemos podido observar que nuestro método de optimización funciona de forma distinta cuando trabajamos sobre CIFAR10. Hacer DeepDream sobre las neuronas de la última capa ya no genera ejemplos prototípicos de muestras reales en el dataset. En vez, captura formas y patrones comunes en muestras de una misma clase y los coloca a través de toda la imagen. Esto nos permite interpretar qué tipo rasgos excitan particularmente a la neurona y teorizar sobre el hecho de que imágenes que contengan varias caras de gatos o varias ruedas excitarán en mayor medida a sus neuronas relacionadas.

Si bien las imágenes que obtenemos ya no son fácilmente interpretables sin un conocimiento previo, resta estudiar si las redes son capaces de hacerlo.

Tomamos 100 muestras al azar y sin repetición por cada una de las 10 clases en CIFAR10 en su versión de testeo. Procesamos las muestras capturando las activaciones en la capa

AVGPOOL de ResNet20. AVGPOOL es la capa inmediatamente previa a la capa FC y contiene 64 neuronas. Posteriormente optimizamos con nuestro método la neurona relacionada a la clase Camión en FC durante 300 iteraciones con coeficiente de Variación Total en 300. Procesamos la imagen optimizada y capturamos sus activaciones en AVGPOOL. Finalmente comparamos los embeddings resultantes de la imagen optimizada contra las otras 1000 muestras capturadas utilizando similitud coseno.

Si nos fijamos en las 5 muestras con mayor similaridad, podemos observar que todas pertenecen a la clase Camión. Resulta interesante que estas 5 muestras exhiben imágenes de camiones considerablemente distintas entre sí; se observan camiones que apuntan en diferentes direcciones, con ruedas más y menos visibles y de diferentes colores.

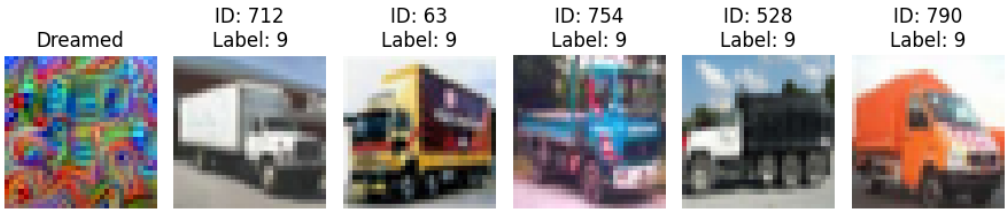


Fig. 4.39: Imagen optimizada de la clase Camión junto a las 5 muestras con mayor similaridad coseno en los embeddings de la capa AVGPOOL en 1000 muestras tomadas de todas las clases de CIFAR10.

Si comparamos todas las similitudes obtenidas observamos que las relacionadas a la clase Camión (etiqueta 9) son con diferencia aquellas que tienen mayor mediana. Se distingue también la clase de la etiqueta 1 con la segunda mediana más alta; la etiqueta 1 se relaciona con la clase Auto y está fuertemente vinculada semánticamente con la clase Camión.

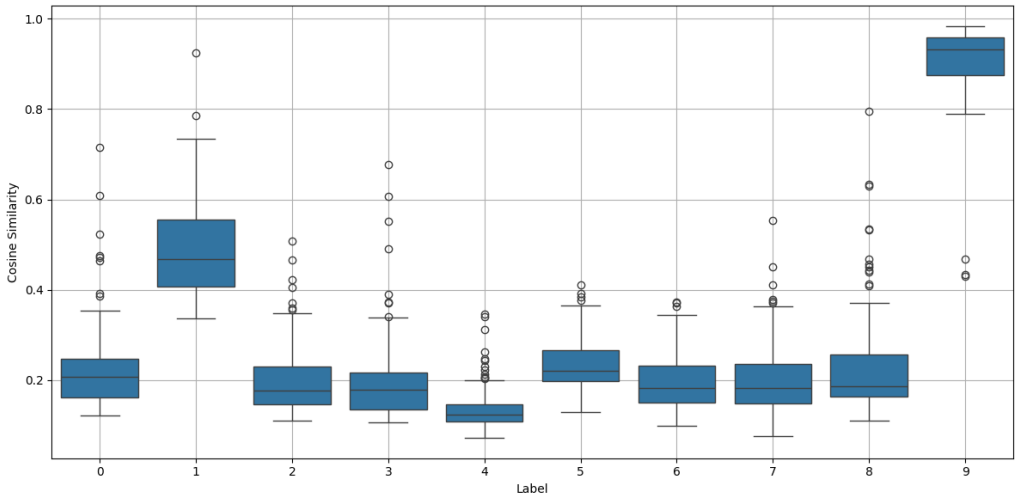


Fig. 4.40: Boxplot de las similaridades obtenidas por cada clase de CIFAR10 contra una imagen optimizada a partir de la neurona Camión en la capa FC. Las similaridades son realizadas en base a los embeddings de la capa AVGPOOL. La clase 9, relacionada a Camión, presenta con diferencia la similitud mediana más alta.

Observamos así resultados parecidos a los que habíamos obtenido con MNIST en redes mucho más pequeñas. Podemos observar que las redes son capaces de discernir las imágenes optimizadas aún cuando estas no son interpretables al ojo humano y sin embargo logran procesarlas de forma similar a las que procesa imágenes reales del dataset. Además de todo esto, la red es capaz de preservar relaciones semánticas entre clases similares.

Podemos ir un paso más allá e involucrar clases que no pertenecen al dataset original. Podemos tomar CIFAR100 que contiene 100 categorías distintas a las de CIFAR10 pero cuyas muestras, al igual que CIFAR10, contienen imágenes de 32×32 píxeles RGB.

Repetimos el mismo ejercicio pero esta vez analizando las similitudes obtenidas dentro las 110 clases (10 de CIFAR10 + 100 de CIFAR100), tomando para la comparación 100 muestras por cada clase y comparamos con los embeddings en AVGPOOL.

Realizamos la optimización nuevamente, esta vez sobre la clase Rana bajo las mismas condiciones que antes.

Fig. 4.41: Imagen resultante de optimizar la neurona relacionada a la clase Rana en CIFAR10 con regularización de Variación Total, partiendo de una imagen aleatoria durante 300 iteraciones.

Nuevamente somos capaces de encontrar con facilidad la clase optimizada. Aún comparando con muestras de 110 clases, la similitud mediana de los embeddings de la clase Rana es muy superior a todas las otras (Figura 4.42). Es notable que existen clases con cierta afinidad a Rana dentro de CIFAR100, como Tortuga o Caracol, pero estas no parecen afectar la efectividad del método.

En definitiva, parece ser que nuestro método resulta apropiado como forma de extraer el conocimiento aprendido por la red, permitiendo compararlo con datos reales. Hemos así establecido un método que permite la búsqueda de conocimiento mediante la comparación de muestras perteneciente a clases semánticas. Notemos que ciertas decisiones en nuestra exploración, como la métrica de similitud y la capa capturada, fueron tomadas de forma arbitraria y en realidad abren la posibilidad de poner a prueba rigurosa el método encontrado.

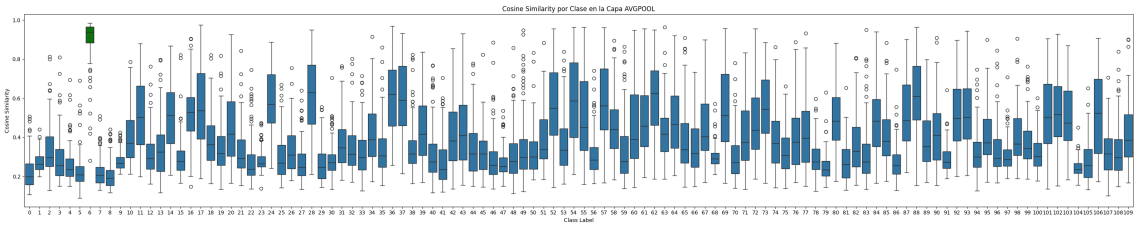


Fig. 4.42: Boxplot de similitud coseno entre la imagen optimizada de Rana en ResNet20 sobre muestras de 110 clases de CIFAR10 y CIFAR100. Las clases del 0 al 9 representan las etiquetas usuales de CIFAR10, las clases del 10 al 109 representan las etiquetas de CIFAR100 resultante de sumar 10 a sus etiquetas usuales. Se observa que la clase 6, relacionada con Rana, es la de similitud mediana mayor.

4.6.1. Experimentación

Buscamos hacer una exploración profunda de las capacidades de nuestro método. Para esto, exploramos cómo varían los resultados al utilizar activaciones de diferentes capas de la red y distintas métricas de similitud.

Las métricas exploradas fueron: similitud coseno, similitud $L2$ normalizada e, inspirados en los resultados de la Variante 4, la similitud por patrones de activación.

Las capas exploradas van de más a menos profundas y fueron:

- FC
- AVGPOOL
- LAYER3_2_BN2
- LAYER3_2_BN1
- LAYER3_1_BN2
- LAYER3_1_BN1
- LAYER3_0_BN2
- LAYER3_0_BN1

Las activaciones de las capas FC y AVGPOOL son unidimensionales por diseño; son vectores en $\mathbb{R}^{10}$ y $\mathbb{R}^{64}$, respectivamente. El resto de las capas mencionadas son bidimensionales con lo cual generan activaciones pertenecientes al espacio $\mathbb{R}^{H \times W}$ con H, W variables en cada capa. Para poder hacer comparaciones vectoriales, las capas bidimensionales fueron aplanadas (flattened), convirtiendo sus activaciones en vectores unidimensionales pertenecientes a $\mathbb{R}^{H \cdot W}$.

Debido al requerimiento de cómputo necesario para entrenar redes ResNet20, trabajamos sobre una única red preentrenada con todas las clases de CIFAR10.

Trabajamos con dos datasets: CIFAR10 y CIFAR100. De cada una de las 110 categorías conjuntas, se tomaron 200 muestras al azar y sin repetición, para un total de 22000 muestras.

Para cada una las neuronas en FC, vinculadas a clases de CIFAR10, se realizó el método de maximización neuronal durante 600 iteraciones, usando regularización de Variación Total con coeficiente de 300. Así, se obtuvieron en total 10 imágenes prototípicas de las clases de CIFAR10, una por cada clase.

Para cada métrica de similitud con las activaciones de cada capa, se realizó la comparación de las activaciones de las 22000 muestras contra las activaciones producidas por las 10 imágenes prototípicas. De entre las 110 clases posibles, una clase se consideró utilizada para el entrenamiento si fue aquella con mayor mediana de similitud al compararse con alguna de las imágenes prototípicas. Se evaluó la cantidad de clases correctamente predichas como entrenadas y también la cantidad de veces que una cierta clase entrenada estuvo dentro del top 3 similitudes de mayor mediana.

4.6.2. Resultados y Discusión

Se pudieron observar distinciones bien marcadas en las capacidades predictivas de cada capa. Tanto la similaridad coseno como la similaridad $L2$ fueron capaces de predecir correctamente 9 de 10 clases entrenadas al utilizar las primeras 3 capas. El poder predictivo comienza a decaer a medida que se utilizan capas menos profundas. Estos resultados permiten validar el hecho de que las capas más profundas tienden a contener mayor carga semántica en sus embeddings [13].

La similaridad por patrones de activación tiene un mal desempeño en las primeras dos capas, pudiendo predecir correctamente menos de 5 clases entrenadas. Es interesante que en la tercera capa logra predecir correctamente todas las clases entrenadas y en las siguientes dos capas predice tan solo un poco peor que las otras métricas. El mal poder predictivo en las capas iniciales puede ser explicado por la baja cantidad de neuronas: 10 y 64 neuronas en la primera y segunda capa, respectivamente; A diferencia de la tercera capa que cuenta con 4096 neuronas.

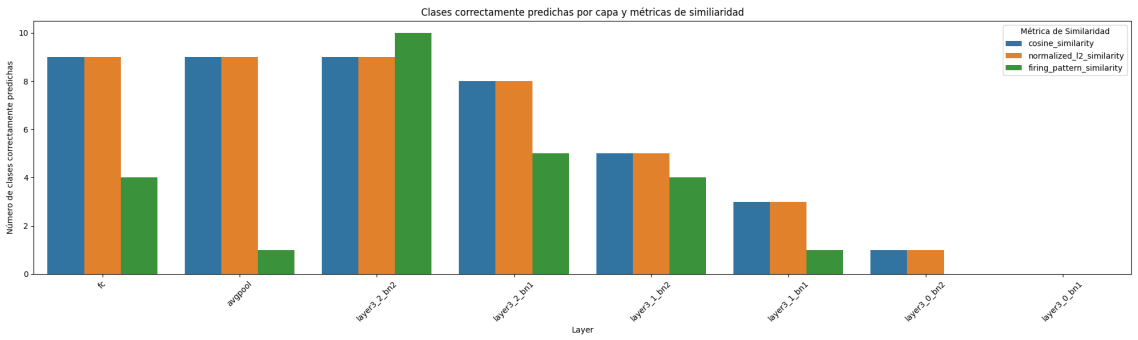


Fig. 4.43: Número de clases correctamente predichas según la capa utilizada y la métrica de similitud empleada. Se comparan las métricas de similaridad coseno, similaridad $L2$ y similaridad por patrones de activación para distintas capas de la red ResNet20.

En cuanto al criterio Top 3, se observa un comportamiento similar al del Top 1, pero con una mejora generalizada en el número de clases correctamente identificadas. Tanto la similaridad coseno como la similaridad $L2$ alcanzan nuevamente su mejor desempeño en las capas más profundas (FC, AVGPOOL y LAYER3_2_BN2), prediciendo correctamente

9 de las 10 clases entrenadas. A diferencia del Top 1, la similaridad por patrones de activación también muestra un desempeño más robusto, logrando predecir 5 clases en FC, 3 en AVGPOOL, y alcanzando un máximo de 10 clases correctamente predichas en LAYER3_2_BN2.

Este aumento de rendimiento al considerar el Top 3 indica que, aunque la predicción más probable (Top 1) pueda fallar debido a patrones adversarios en las 110 clases, las representaciones generadas por estas capas aún contienen suficiente información para ubicar la clase correcta dentro de las tres más similares. También se observa que el desempeño de las capas menos profundas sigue siendo limitado, con muy pocas clases correctamente predichas, incluso bajo este criterio más laxo.

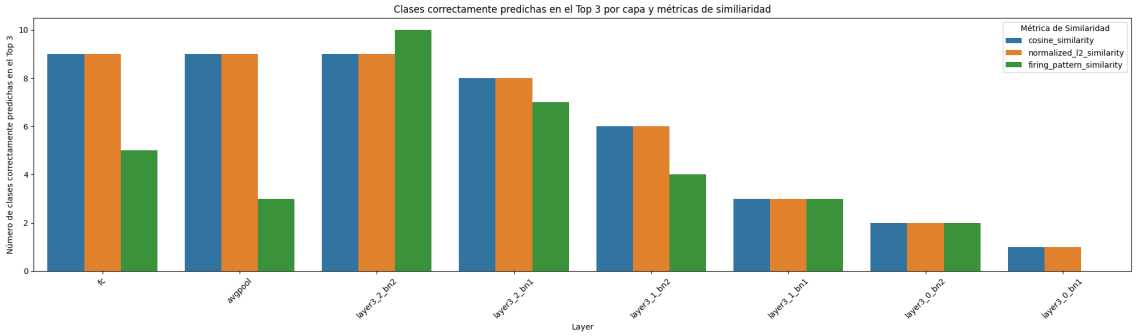


Fig. 4.44: Número de clases correctamente predichas en el Top 3 según la capa utilizada y la métrica de similitud empleada. Se comparan las métricas de similaridad coseno, similaridad $L2$ y similaridad por patrones de activación para distintas capas de la red ResNet20.

4.6.3. Conclusión

Los resultados obtenidos en esta variante demuestran que el método de maximización neuronal puede escalarse efectivamente a arquitecturas más profundas como ResNet20 y a datasets más complejos como CIFAR10 y CIFAR100. A pesar de que las imágenes generadas ya no son tan fácilmente interpretables para el ojo humano como en el caso de MNIST, la red sigue siendo capaz de procesarlas de forma coherente, reconociéndolas como representativas de las clases correspondientes.

Además, observamos que las similitudes en los espacios de activación permiten no solo identificar correctamente las clases entrenadas, sino también preservar relaciones semánticas entre clases afines. La efectividad del método se mantuvo incluso al ampliar el universo de comparación a 110 clases, lo cual valida su robustez y potencial.

Lo más relevante, sin embargo, es que esta exploración abre un abanico de nuevas oportunidades: evaluar otras arquitecturas, experimentar con diferentes métricas de similitud, modificar la capa de captura de activaciones o incluso aplicar este enfoque a dominios completamente distintos. Este trabajo constituye apenas un punto de partida hacia un entendimiento más profundo y sistemático del conocimiento almacenado en redes neuronales modernas.

5. CONCLUSIONES FINALES

5.1. Conclusiones

A lo largo de este trabajo se han mostrado diferentes técnicas para determinar el conjunto de entrenamiento de redes neuronales convolucionales, partiendo de distintos componentes de su arquitectura interna y de suposiciones sobre las clases de entrenamiento usadas.

Se ha podido corroborar que los pesos de la usual última capa Fully Connected presente en redes de clasificación exhibe un comportamiento característico y diferenciado cuando se trata de datos de entrenamiento. En el caso en que la red posee una arquitectura para clasificación de n clases, pero ha sido entrenada con solo $k < n$ de ellas, los pesos de neuronas vinculadas a las clases no entrenadas, como es esperable, exhiben pesos cercanos a 0, muy distintos a los pesos de clases entrenadas. Si se tuviera información sobre el mapeo existente entre las clases de datos reales y las neuronas en las última capa, este análisis encontraría las clases no entrenadas con un alto grado de precisión. Experimentos basados en estas ideas lograron clasificar clases entrenadas y no entrenadas con exactitud de 1 indicando una clasificación perfecta

Algo similar a lo anterior sucede en el caso de las activaciones en la última capa, producto justamente de los bajos pesos. En este caso, las activaciones de las neuronas en la capa son notablemente menores al procesar clases no entrenadas y permiten así una fácil clasificación. Este comportamiento, sin embargo, es muy distinto al observar los pesos y activaciones de la penúltima capa, aún si esta es también Fully Connected. Por más de que la penúltima capa contiene sustancial información semántica sobre la decisiones de la red, no se encontró una forma directa de diferenciar comportamientos de clases entrenadas y no entrenadas en base a únicamente las activaciones. En estos casos, se encontró que la proporción de neuronas activadas (mayores a 0) en muestras de una misma clase es un estadístico informativo para la tarea de clasificación de clases no entrenadas. Se pudo observar que la proporción de neuronas activadas en muestras de clases entrenadas exhibe comportamientos mucho más extremos, tomando valores generalizadamente cercanos a 0 o a 1. Por otro lado, en el caso de las clases no entrenadas, las neuronas tienen comportamientos menos homogéneos, llevando así a proporciones lejos del 0 y del 1. Un método basado en la utilización de K-Means sobre la cantidad de neuronas cuyas proporciones de activación estuvieran entre 0,2 y 0,8 logró resultados muy positivos, teniendo un score F1 mayor a 0,8 para hasta 7 de 10 clases no entrenadas.

Se exploró también la clasificación mediante métodos de maximización de las activaciones de neuronas en la última capa Fully Connected, en particular con una versión de DeepDream creada para esta investigación que incorpora regularizaciones sobre la Variación Total y la norma $L2$ de las imágenes creadas. En el caso en que la arquitectura de la red poseía más neuronas para clasificación que clases entrenadas, el método permitió determinar la clases no entrenadas con un altísimo grado de eficacia.

Exploraciones sobre la maximización de activaciones permitieron mostrar que, independientemente de la fidelidad de las imágenes obtenidas como resultado de la optimización,

estas son interpretadas por la red como si pertenecieran a la clase optimizada, generando así embeddings similares a los generados al procesar muestras reales. Esto llevó a la creación de un nuevo método para encontrar clases entrenadas basadas en la comparación de muestras mediante métricas de similitud entre embeddings generados por optimización y muestras de clases pertenecientes a un espacio de búsqueda. Se pudo observar que en el caso en que las clases entrenadas pertenecían al espacio de búsqueda, las similitudes de sus embeddings eran mucho mayores que las otras clases, permitiendo así clasificar a la clase entrenada como aquella con mayor similitud. Este método obtuvo muy buenos resultados que pudieron ser validados en arquitecturas artesanales pequeñas y en arquitecturas estándares en la industria como ResNet20.

5.2. Contribuciones y Próximos Pasos

Los resultados encontrados en las seis variantes exploradas en este trabajo nos permiten afirmar que en efecto existen formas de extraer el conocimiento adquirido de redes neuronales. Las ideas y técnicas desarrolladas son en su mayor parte novedosas y abren líneas de investigación que puede seguir siendo exploradas.

En particular, la utilización de la proporción de activación como forma de determinar clases no entrenadas, había sido utilizada en el pasado únicamente como un discriminador supervisado, que requería de un calibrado y acceso a la red durante el entrenamiento [31]. En este trabajo hemos podido observar comportamientos más generales que apuntan a que este estadístico puede ser usado en ambientes no supervisados. Un estudio más a profundidad de este método, en redes más grandes por ejemplo, podría revelar atributos aún más determinantes.

Sin duda el mayor aporte de este trabajo resulta de utilizar DeepDream y sus embeddings generados como forma de exponer el aprendizaje interno, desconocido, de la red y poder compararlo con clases cualesquiera por fuera de la red. Los resultados obtenidos hasta ahora son sumamente prometedores y sin duda deben ser explorados con mayor profundidad. La exploración de las seis variantes en este trabajo culminaron en el desarrollo de un framework de evaluación de conocimiento que es agnóstico a la red, al dataset utilizado y las métricas de comparación.

Si bien en la Variante 6 estudiamos algunos de los parámetros posibles, la realidad es que todavía queda mucho por hacer y explorar. Una idea inmediata es utilizar más métricas y explorar más capas de la red. Otra posibilidad es comparar los resultados en redes más grandes como ResNet en sus variantes 32, 44 y 56. Estudios preliminares en redes más grandes mostraron resultados prometedores.

Otra forma de expandir el framework de investigación es mezclar los embeddings de varias capas para hacer las comparaciones, fundamentado por el hecho de que los embeddings de cada capa encapsulan información semántica potencialmente distinta y por lo tanto incluir a diferentes capas en la comparación puede llevar a comparaciones más fieles. Nuevamente resultados preliminares de esto fueron prometedores en métricas ligadas a la estructura de la capa, como por ejemplo la similitud por patrones de activación.

Una idea que surgió durante el desarrollo del trabajo, pero que todavía no fue explorada, es variar la función objetivo y las regularizaciones a la hora de optimizar. En particular, incluir como objetivo no sólo la maximización de una neurona de clase sino además buscar

la minimización de las activaciones de las otras neuronas. Otra idea es agregar un término regularizador que premie la generación de imágenes similares a las del espacio real o bien que premie la generación de embeddings similares.

Finalmente, se puede explorar un análisis de robustez del método en donde se comparen los resultados al buscar sobre muestras de las clases que han sido alterados visualmente con respecto a los datos originales entrenados. Se podría explorar modificaciones como cambio de brillo, agregado de ruido o cortes aleatorios.

Queda todavía mucho por hacer y espero realmente poder continuar con las bases que sentaron estos 7 meses de exploración y 6 años de estudio.

Bibliografía

[1] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[2] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[3] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org, 2013.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 262–270, 2015.

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[8] Lars Prechelt. Early stopping—but when? In *Neural Networks: Tricks of the Trade*, pages 55–69. Springer, 1998.

[9] Lutfi Senel, Ihsan Utlu, Veysel Yucesoy, Aykut Koc, and Tolga Cukur. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, PP, 11 2017.

[10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 2009.

[11] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[12] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, 31st International Conference on Machine Learning (ICML)*, Lille, France, 2015. Copyright 2015 by the authors.

[13] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[14] Will Xiao and Gabriel Kreiman. Xdream: Finding preferred stimuli for visual neurons using generative networks and gradient-free optimization. *PLOS Computational Biology*, 16:1–15, 06 2020.

[15] Carlos R. Ponce, Will Xiao, Peter F. Schade, Till S. Hartmann, Gabriel Kreiman, and Margaret S. Livingstone. Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences. *Cell*, 177(4):999–1009.e10, 2019.

[16] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

[17] Walter J Scheirer, Anderson de Rezende Rocha, Atul K Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[18] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2016.

[19] Lauren Neal, Matthew Olson, Alan Fern, and Weng-Keen Li. Open set recognition using counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628, 2018.

[20] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2017.

[21] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[22] Soroush Mohseni, Megha Pitale, Vivek Yadawa, and Zhangyang Wang. Self-supervised learning for generalizable out-of-distribution detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(4):5216–5223, 2020.

[23] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints*, page arXiv:1708.07747, August 2017.

[24] Tarin Clanuwat, Mikel Bober-Irizar, and Asanobu Kitamoto. Deep learning for classical japanese literature. In *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[25] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Technical Report, https://www.cs.toronto.edu/ kriz/cifar.html.

[26] Yaofo Chen. Pytorch cifar models. `https://github.com/chenyaofo/pytorch-cifar-models`. Accessed: 2025-5-17.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[28] Sanghyeon An, Minjun Lee, Sanglee Park, Heerin Yang, and Jungmin So. An ensemble of simple convolutional neural network models for mnist digit recognition. *arXiv preprint arXiv:2008.10400*, 2020.

[29] Charity J Morgan. Use of proper statistical techniques for research studies with small samples. *American Journal of Physiology-Lung Cellular and Molecular Physiology*, 313(5):L873–L877, 2017.

[30] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[31] Young-Woo Lee and Heung-Seok Chae. Identification of untrained class data using neuron clusters. *Neural Computing and Applications*, 35:10801–10819, 2023.

[32] Tushar Ganguli and Edwin K. P. Chong. Activation-based pruning of neural networks. *Algorithms*, 17(1):48, 2024.

[33] Gabriel Kreiman, Christof Koch, and Itzhak Fried. Category-specific visual responses of single neurons in the human medial temporal lobe. *Nature Neuroscience*, 3(9):946–953, 2000.

[34] Camille Gollety. Neuronal correlates of rapid learning in human visual memory tasks. *Kreiman Lab Publications*, 2022. Disponible en: `https://klab.tch.harvard.edu/publications/PDFs/gk8105.pdf`.

[35] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`, 2015. Accessed: 2025-06-01.

[36] Antonin Chambolle. An introduction to total variation for image analysis, 2004. Lecture notes from the Summer School on Mathematical Image Processing, Como.

[37] Nacira Diffellah, Tewfik Bekkouche, and Rabah Hamdini. Image denoising algorithms using norm minimization techniques. In *Proceedings of the 2nd International Conference on Data Science and Intelligent Applications (IcoDSIA)*, volume 2904 of *CEUR Workshop Proceedings*, pages 170–178. CEUR-WS, 2021.

[38] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *CoRR*, abs/1602.03616, 2016.

[39] Alexander Mordvintsev, Chris Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog*, 2015.

[40] Alexandra Bardon, Will Xiao, Carlos R. Ponce, Margaret S. Livingstone, and Gabriel Kreiman. Face neurons encode nonsemantic features. *Proceedings of the National Academy of Sciences*, 119(16):e2118705119, 2022.

[41] Gabriel Kreiman. Chapter seven - what do neurons really want? the role of semantics in cortical representations. In Kara D. Federmeier and Diane M. Beck, editors, *Knowledge and Vision*, volume 70 of *Psychology of Learning and Motivation*, pages 195–221. Academic Press, 2019.