ISSN 1870-4069

Filtros Espaciales aprendibles en CNNs: Análisis de filtros de Gabor hacia un entrenamiento más eficiente

Carlos Orozco-Solis[1], Alfonso Rojas-Domínguez[1], Héctor Puga[1], Manuel Ornelas Rodríguez[1] Martín Carpio[1], Valentín Calzada-Ledesma[2]

[1] Tecnológico Nacional de México, Campus León, México

[2] Tecnológico Nacional de México, campus Purísima del Rincón, México

alfonso.rojas@gmail.com

Resumen. La integración de filtros de Gabor en redes convolucionales (CNNs) ha sido objeto de estudio debido a su capacidad para capturar características relevantes en imágenes. En este trabajo, se investiga el impacto de estos filtros en el rendimiento y la estabilidad del entrenamiento de una CNN. Se realizan dos experimentos utilizando bloques GCC (Gabor Convolutional) y CCC (Convolutional), variando la entrenabilidad y la inicialización de los bloques. Se analiza la convergencia de los parámetros de los filtros de Gabor a lo largo del entrenamiento y se emplea un enfoque de clusterización para identificar patrones de comportamiento. Los resultados muestran que la inclusión de filtros de Gabor no degrada significativamente el desempeño de la red y que estos filtros se estabilizan rápidamente, lo que permitiría reducir el tiempo de entrenamiento de las redes. Se presenta un análisis detallado de los comportamientos de los filtros y se discute cómo diferentes configuraciones de los bloques mencionados afectan la precisión de la red. Estos hallazgos proporcionan información valiosa para la optimización de redes neuronales convolucionales que integren filtros espaciales entrenables.

Palabras clave: Filtros entrenables, filtros de Gabor, redes convolucionales.

Learnable Spatial Filters in CNNs: Analysis of Gabor Filters Towards a More Efficient Training

Abstract. The integration of Gabor filters in convolutional neural networks (CNNs) has been the subject of study due to their ability to capture relevant features in images. This work investigates the impact of these filters on the performance and stability of CNN training. Two experiments are conducted using GCC (Gabor Convolutional) and CCC (Convolutional) blocks, varying the trainability and initialization of the blocks. The convergence of Gabor filter

parameters during training is analyzed, and a clustering approach is employed to identify behavior patterns. The results show that the inclusion of Gabor filters does not significantly degrade the network's performance and that these filters stabilize quickly, potentially reducing network training time. A detailed analysis of filter behaviors is presented, and the discussion covers how different block configurations affect network accuracy. These findings provide valuable insights for the optimization of convolutional neural networks that integrate trainable spatial filters.

Keywords: Trainable spatial filters, Gabor filters, CNNs.

1. Introducción

Las Redes Neuronales Convolucionales (CNNs por sus siglas en inglés) [8] poseen una poderosa capacidad de aprendizaje, que, a través de la implementación de múltiples etapas de extracción de características, les permite aprender automáticamente representaciones complejas de datos [4], particularmente en imágenes digitales. Antes de la popularización de las CNNs, el proceso de extracción de características en imágenes se realizaba mediante filtros espaciales, los cuales eran diseñados a mano por un experto en el campo.

Éstos ofrecen una gran flexibilidad y adaptabilidad, ya que pueden detectar una amplia gama de características, tales como bordes, esquinas, texturas, entre otros. Sin embargo, existe un problema fundamental, y es que para extraer las características principales de una o un conjunto de imágenes, es necesario tener un conocimiento a priori del dominio del problema para elegir cuidadosamente los filtros necesarios para resaltar las características deseadas, este proceso en sí mismo puede ser muy laborioso y exhaustivo, ya que implica ajustar manualmente los parámetros de cada filtro.

Las CNNs automatizan el proceso de extracción de características a través de la aplicación de capas convolucionales, cuyos parámetros se establecen automáticamente mediante un proceso de entrenamiento. En la literatura se ha observado que tras entrenar una CNN (como AlexNet [6]), los filtros convolucionales de la primera capa tienden hacia la creación de filtros espaciales, similares a los filtros de Gabor [10], los cuales son efectivos para extraer atributos simples, tales como bordes, esquinas y texturas. Asimismo, en [12] se afirma que los pesos de capas iniciales de diversos modelos de imágenes tienden a converger hacia filtros de Gabor, entre otros, y que las características universales observadas con más frecuencia en modelos de imágenes son funciones de base canónica 2D de Fourier, como filtros de Gabor o wavelets.

Es decir, los atributos que son aprendidos en las primeras capas convolucionales pueden ser obtenidos mediante filtros espaciales. Así, en la literatura se propuso reemplazar algunos filtros convolucionales con filtros espaciales, conduciendo al desarrollo de varias estrategias revisadas a profundidad en la Sección 2. Destaca un tipo de arquitectura llamada GaborNet la cual implementa filtros de Gabor en la primera capa convolucional, que llamaremos la capa de Gabor. Aunque la GaborNet ha sido utilizada en la literatura, la falta de un análisis exhaustivo de sus parámetros limita su comprensión y la capacidad para maximizar su rendimiento.

Tabla 1. Hiperparámetros de los experimentos.

Parámetro	Valor	Descripción
Epochs	100	Número de iteraciones del entrenamiento.
Batch size	1024	Tamaño del lote de entrenamiento.
Learning rate	0.001	Taza de aprendizaje para el entrenamiento.
Executions	35	Número de repeticiones del experimento.
Optimizer	Adam	Algoritmo de optimización utilizado.
Loss	Cross-entropy	Función de pérdida para optimizar el modelo.

Hasta donde sabemos, en la literatura no hay un análisis que permita comprender del todo dicha arquitectura. Es por ello que el presente trabajo se enfoca en llenar este vacío, mediante un análisis detallado de los parámetros de la capa de Gabor. Este análisis revela hallazgos significativos para mejorar la eficacia y eficiencia de las GaborNet. Por ejemplo, se observa que el diseño del filtro se decide de manera temprana durante las primeras iteraciones del proceso de entrenamiento, lo que sugiere la importancia de una inicialización adecuada. Éste y otros hallazgos avanzan hacia una base sólida para futuras investigaciones en el diseño y optimización de CNNs que incorporan filtros espaciales, y destacan la importancia de un análisis detallado de los parámetros para maximizar el rendimiento de estos modelos en aplicaciones prácticas.

2. Trabajos relacionados

Para aprovechar los beneficios de los filtros espaciales, algunos autores han propuesto aplicarlos sobre imágenes de entrenamiento antes de pasarlas a una CNN. En [7], aplican filtros espaciales para detectar bordes y esquinas en imágenes faciales, mejorando la precisión obtenida por una CNN en 8.5 %. En [3], proponen usan filtros espaciales para la clasificación de dígitos escritos a mano, logrando resultados competitivos contra LeNet [8], que obtiene 99.05 % de clasificación correcta, mientras que la propuesta alcanza 99.16 %. La integración de filtros espaciales en CNNs es una tendencia reciente que ha demostrado mejorar el rendimiento en algunos casos.

Por ejemplo, la red GCN [10] utiliza un banco de filtros de Gabor con U orientaciones y V escalas, logrando 99.37 % de precisión en MNIST, comparable al 99.43 % de la Red de Respuesta Orientada [16]. Ambas redes utilizan 0.25 y 0.49 millones de parámetros, respectivamente. En CIFAR10/100, la GCN alcanza 96.12 % y 79.87 %, respectivamente, reduciendo a la mitad los parámetros requeridos en comparación con la Wide Residual Network (WRN) [15], que logra 96.00 % y 80.95 %.

Para integrar aún más los filtros espaciales en las CNNs, se ha propuesto que la primera capa convolucional esté compuesta exclusivamente de filtros espaciales, ajustados por la red de la misma manera que las capas convolucionales convencionales. Por ejemplo, GaborNet [1] y PCFNet [11] emplean este enfoque. En ambos trabajos, se observa que la ventaja de rendimiento entre el uso de filtros espaciales en la primera capa o no usarlos disminuye a medida que aumenta el tamaño del conjunto de datos de entrenamiento.

Tabla 2. Configuración de los experimentos y resultados.

	Bloque GCC	**Bloque CCC**	**Accuracy**
Exp.	**(Gabor)**	**(Convolución)**	**en la Prueba**
1	Entrenable	Entrenable	77.55 %
2.a	Entrenable	Ausente	61.17 %
2.b	Pre-entrenado	Entrenable	67.32 %

Esto se demuestra en pruebas realizadas en una CNN de Filtros Predefinidos (PCF) [11], que implementa filtros espaciales como Gabor (Ga), Sobel (So) y Schmid (Sc) en su capa inicial. Utilizando solo el 5 % del conjunto de datos CIFAR10, PCF-GaSc-ResNet18 logró una precisión del 66.32 %, superando a ResNet18 (con un 62.05 %). De manera similar, utilizando solo el 20 % del conjunto de datos CIFAR100, PCF-GaSc-WRN168 alcanzó un 55.15 %, superando a WRN168 (con un 53.27 %). Sin embargo, al evaluar los conjuntos de datos CIFAR10/100 completos, no se observaron diferencias significativas. En [1], se comparan una CNN simple y AlexNet con sus respectivas variantes de Gabor. La Gabor-CNN supera a la CNN en 6 % de precisión en el problema "Dogs vs Cats". En el conjunto de datos "AffectNet", la diferencia es de 3 %, y en "ImageNet", no se observa una diferencia significativa.

3. Metodología

3.1. Redes neuronales convolucionales

Una Red Neuronal Convolucional (CNN) es un tipo de red neuronal profunda diseñada para procesar datos organizados en una cuadrícula, típicamente imágenes. Las CNNs aprenden representaciones jerárquicas de datos a través de sus capas convolucionales [9]. Las representaciones aprendidas luego se emplean en tareas de análisis de imágenes como clasificación, segmentación, reconocimiento de identidad, etc. Una CNN está organizada en varias capas, cada una de las cuales contiene múltiples filtros o kernels convolucionales.

Estos kernels se aplican a las imágenes de entrada mediante convolución, una operación que implica el desplazamiento de una ventana sobre la imagen (1). Este proceso facilita la extracción de características al utilizar un conjunto específico de pesos que se multiplican por los elementos correspondientes del campo receptivo [2]. La operación de convolución puede expresarse de la siguiente manera:

$$g(x,y) = w * f(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i,j) f(x-i, y-j), \tag{1}$$

donde $g(x,y)$ es la imagen filtrada, $f(x,y)$ es la imagen original, w es el kernel convolucional, $\sum_{i=-a}^{a} \sum_{j=-b}^{b}$ denota una suma doble sobre todas las posiciones del kernel, los índices i y j representan coordenadas en el kernel, y a y b definen el tamaño del núcleo (normalmente $a = b$).

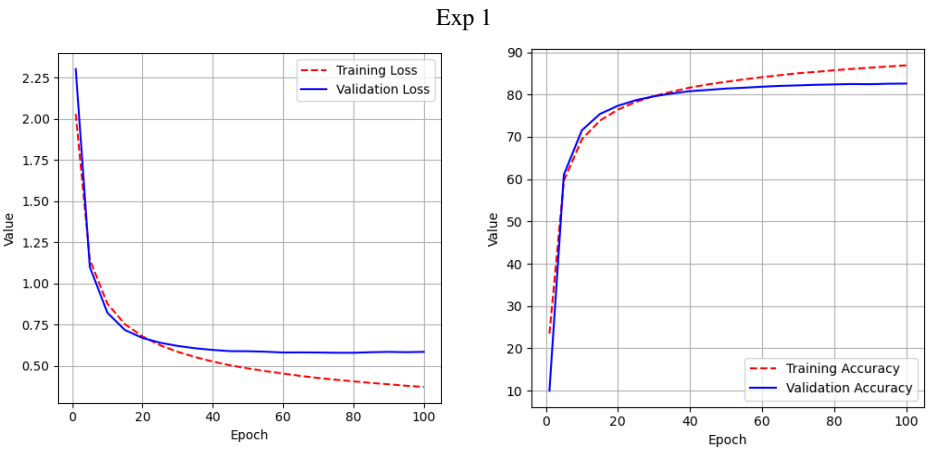

Fig. 1. Curvas de pérdida (Izq.) y precisión (Der.) durante el entrenamiento y validación del modelo - experimento 1.

3.2. Arquitectura

Este estudio emplea dos variantes de la misma arquitectura. La primera variante consiste en dos bloques, GCC y CCC: el bloque GCC incluye una capa Gabor con 40 filtros de tamaño 7 y dos capas convolucionales con 64 y 128 filtros de tamaño 3, respectivamente. El bloque CCC consta de tres capas convolucionales con 256 filtros de tamaño 3. En la segunda variante, el bloque GCC permanece igual, pero el bloque CCC se reemplaza por una operación de MaxPooling, lo que permite que las dimensiones de los mapas de activación se ajusten a la primera capa Fully Connected (FC). En total, la red contiene tres capas FC.

Si se utilizara una primera capa convolucional tradicional con 40 filtros w, cada uno de tamaño 7×7, ello implicaría un total de 49 parámetros que la red debería entrenar por cada filtro es decir, $40 \times 49 = 1960$ parámetros en total. Dada la tendencia de estos filtros a converger hacia patrones espaciales similares a funciones de Gabor, reemplazamos los filtros w con funciones de Gabor, que son sinusoides complejas moduladas por una envolvente Gaussiana [1]:

$$g(x, y, w, \theta, \psi, \sigma) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma^2}\right) \exp\left(i(wx' + \sigma)\right), \tag{2}$$

$$x' = \quad x\cos(\theta) + y\sin(\theta), \tag{3}$$

$$y' = -x\cos(\theta) + y\sin(\theta). \tag{4}$$

La ecuación (2) se puede expresar en sus partes real e imaginaria; en este trabajo, utilizamos la parte real de la función Gabor:

$$g(x, y, w, \theta, \psi, \sigma) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma^2}\right) \cos(wx' + \psi), \tag{5}$$

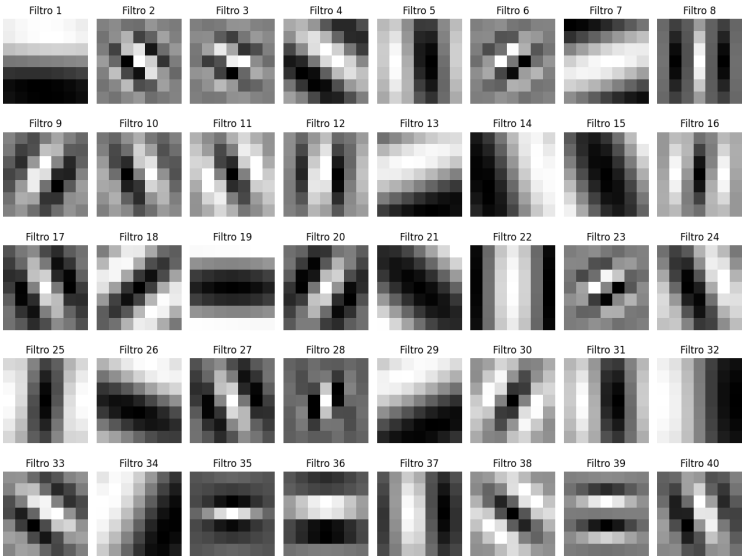

Fig. 2. Filtros de la capa Gabor entrenados - experimento 1.

donde (x, y) denota la posición del píxel en el dominio espacial, w representa la frecuencia angular central de una onda plana sinusoidal, θ indica la rotación en sentido contrario a las agujas del reloj de la función Gaussiana (es decir, la orientación del filtro de Gabor), σ representa la desviación estándar de la Gaussiana, y ψ controla la fase de las oscilaciones sinusoidales dentro del filtro de Gabor.

Establecemos $\sigma \approx \pi/w$ para definir la relación entre σ y w como se describe en [13]. Al combinar múltiples filtros de Gabor con diferentes orientaciones y frecuencias, se forma un banco de filtros. La capa Gabor es una capa convolucional especializada, diseñada para aplicar un banco de filtros de Gabor a través del cual los 49 parámetros para un filtro ahora se reducen a solo 4 parámetros: $(w, \theta, \psi, \sigma)$ ($40 \times 4 = 160$ parámetros en total para la red). Integrar esta capa de Gabor en la CNN produce la arquitectura GaborNet [1]. Debido a que modificamos la GaborNet original, y para evitar confusiones, nos referimos a nuestra propia implementación como **GaborNet2**.

3.3. Implementación

Para desarrollar **GaborNet2** en Python, utilizamos la librería PyTorch, que proporciona una interfaz dinámica para construir y entrenar modelos de aprendizaje profundo. Con esta herramienta, podemos implementar de manera efectiva una arquitectura de red neuronal convolucional (CNN) en la que se integra una capa con filtros de Gabor. La arquitectura base de la CNN en PyTorch se define creando una clase que hereda de `nn.Module`. En el constructor de esta clase se definen las capas convolucionales (`nn.Conv2d`), de agrupación (`nn.MaxPool2d`), de normalización (`nn.BatchNorm2d`), y de activación (`nn.ReLU`, `nn.Sigmoid`, etc.). Una característica notable de PyTorch es la integración de varios optimizadores, como el descenso de gradiente estocástico (SGD), Adam, RMSprop o Adagrad.

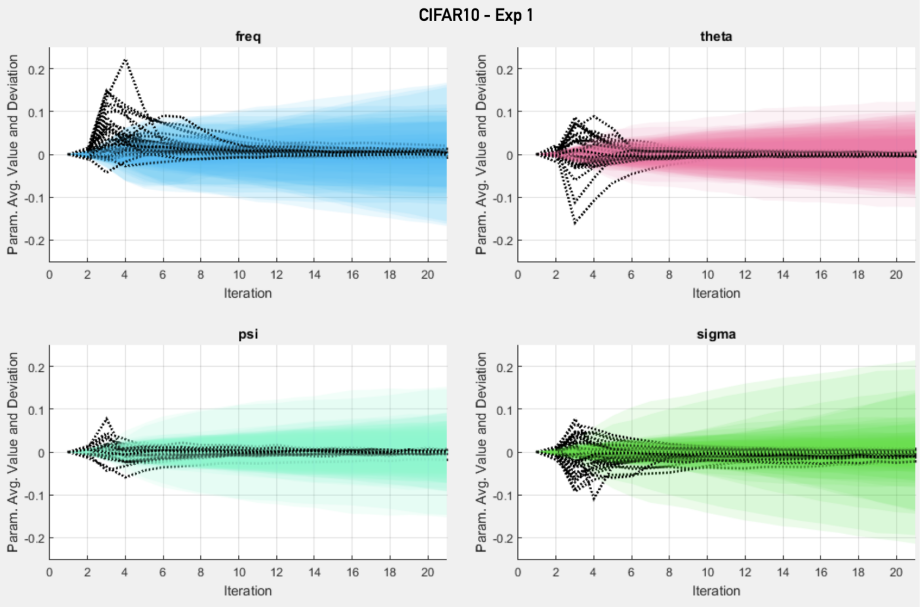

Fig. 3. Evolución de parámetros de la capa Gabor durante el entrenamiento - experimento 1. Se muestran simultáneamente las curvas y regiones sombreadas de las 35 repeticiones sombreadas.

Estos optimizadores ajustan los pesos de la red para minimizar una función de pérdida mediante la actualización iterativa de los pesos siguiendo el descenso de gradientes calculados automáticamente. Por esta razón, la capa de Gabor se diseñó basándose en la capa tradicional `nn.Conv2d`.

La capa convolucional `nn.Conv2d` es una subclase de `_ConvNd`, por lo que la capa personalizada `GaborConv2d` también hereda de ésta. En el constructor de esta capa personalizada, se definen 4 parámetros principales de los filtros de Gabor: frecuencia espacial, orientación, desviación estándar de la Gaussiana y fase. Estos parámetros son entrenables y se actualizan mediante SGD (con el optimizador Adam) de la misma manera que los pesos de una capa tradicional. Una vez definida la capa `GaborConv2d`, es fácil reemplazar la primera capa convolucional de la arquitectura base de la CNN con esta capa personalizada. Al hacerlo, la CNN utilizará filtros de Gabor en lugar de filtros convencionales en su capa inicial.

4. Configuración experimental

En esta sección, se detallan los parámetros y la configuración utilizados para evaluar la convergencia de los filtros de **GaborNet2**. Para comprender mejor el comportamiento de la red, se llevaron a cabo dos experimentos principales utilizando variantes de la arquitectura. Los hiperparámetros usados se muestran en el Cuadro 1 y la configuración de experimentos en el Cuadro 2. El Experimento 1 corresponde a la utilización de la primera variante de la arquitectura, entrenando la red de manera convencional.

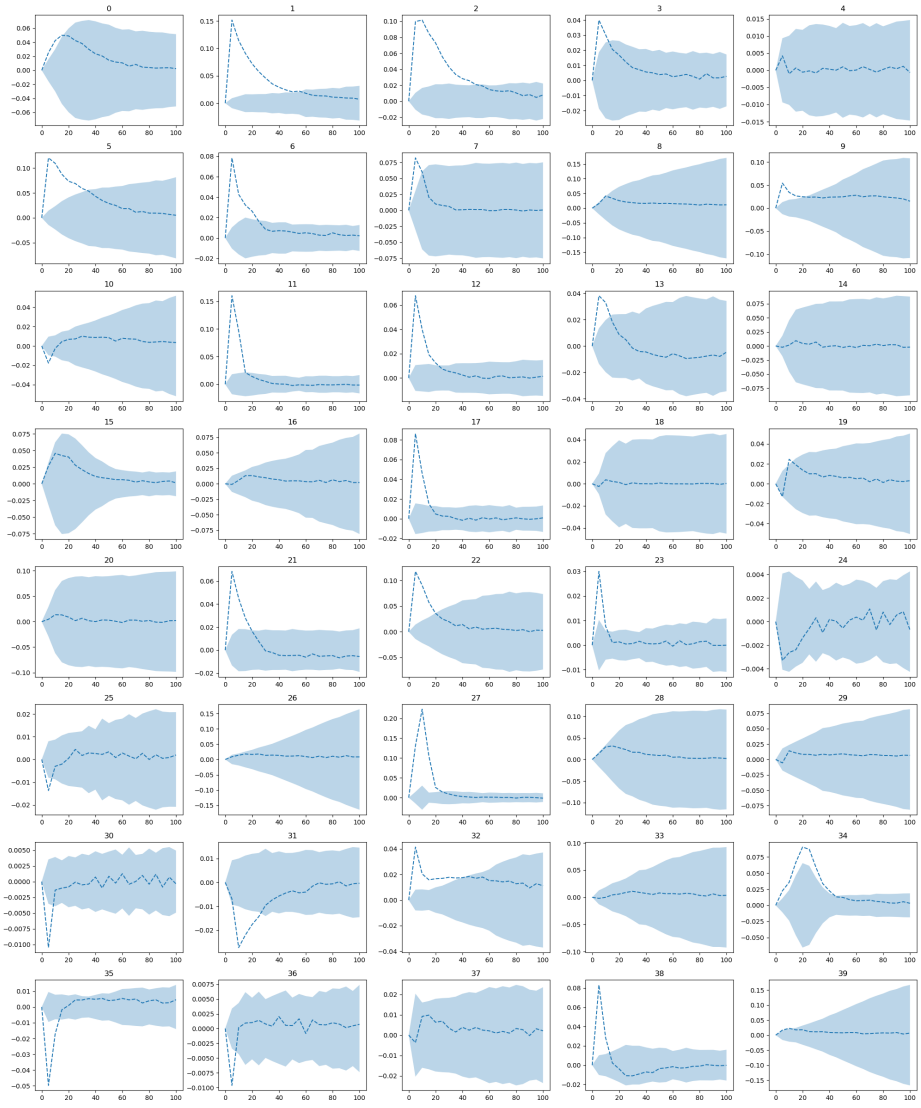

Fig. 4. Evolución de los 40 valores de frecuencia w - experimento 1.

En el Experimento 2, se separó el entrenamiento en dos etapas para obtener un análisis más detallado. En la etapa a), se empleó la segunda variante de la arquitectura para dar mayor énfasis a la capa de Gabor y observar de manera más precisa el comportamiento de los parámetros (w, θ, ψ, σ) durante el entrenamiento. Una vez finalizado este proceso, se guardaron los pesos obtenidos; posteriormente, en la etapa b) se retomó la primera variante de la arquitectura, cargando los pesos resultantes de la etapa a) y congelando el primer bloque de la red (para no perder los pesos ya entrenados hasta ese punto).

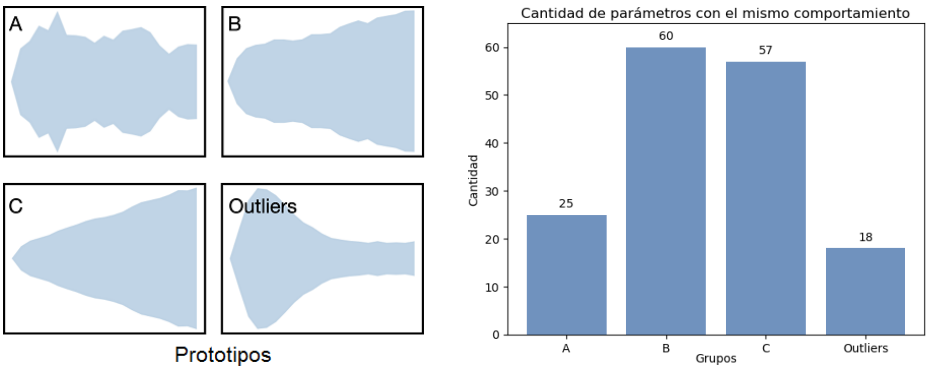

Fig. 5. Representantes (Izq.) y su frecuencia (Der.) - experimento 1.

Luego se continuó el entrenamiento de la red hasta finalizar con las épocas de entrenamiento. Con el Experimento 2 buscamos promover la mejor convergencia de los filtros de la capa Gabor, al entrenarlos de manera aislada de las otras capas convolucionales, y además buscamos determinar si el rendimiento obtenido es comparable al del Experimento 1 (donde se entrenaron ambos bloques de la red simultáneamente). Es importante destacar que cada experimento se repitió 35 veces utilizando la misma inicialización, garantizando así que cada filtro tuviera los mismos valores en la época 0, para las 35 repeticiones.

Parámetros del Filtro de Gabor: La inicialización de los 4 parámetros $(w, \theta, \psi, \sigma)$ de la capa de Gabor se basa en [13]:

$$w_n = \frac{\pi}{2} \cdot 2^{\frac{-(n-1)}{2}}, \tag{6}$$

$n \in \{1, ..., 5\}$; $\theta_m = \frac{\pi}{8} \cdot m - 1$, $m \in \{1, ..., 8\}$; $\psi \sim \mathcal{U}(0, \pi)$; y $\sigma \approx \frac{\pi}{w}$. Esta configuración resulta en un banco de filtros de Gabor con 5 escalas y 8 orientaciones diferentes. Esta diversidad en escalas y orientaciones permite una gran adaptabilidad para una amplia gama de características en las imágenes.

Dataset: CIFAR-10 es un conjunto de datos ampliamente utilizado en el campo del aprendizaje automático para entrenar y probar modelos de clasificación de imágenes. El conjunto de datos consta de 60,000 imágenes en color de 32×32 píxeles, distribuidas uniformemente en 10 clases (6,000 por clase) [5]. Para este trabajo, las imágenes se convirtieron a escala de grises y se separaron aleatoriamente en tres conjuntos: 40,000 imágenes para entrenamiento, 10,000 para validación y 10,000 para prueba.

5. Resultados

En la Fig. 1 se pueden observar las curvas de pérdida de entrenamiento y pérdida de validación para el Experimento 1. La curva de pérdida de entrenamiento (linea punteada roja) muestra una disminución constante a lo largo de las 100 épocas, mientras que la curva de precisión aumenta de manera continua.

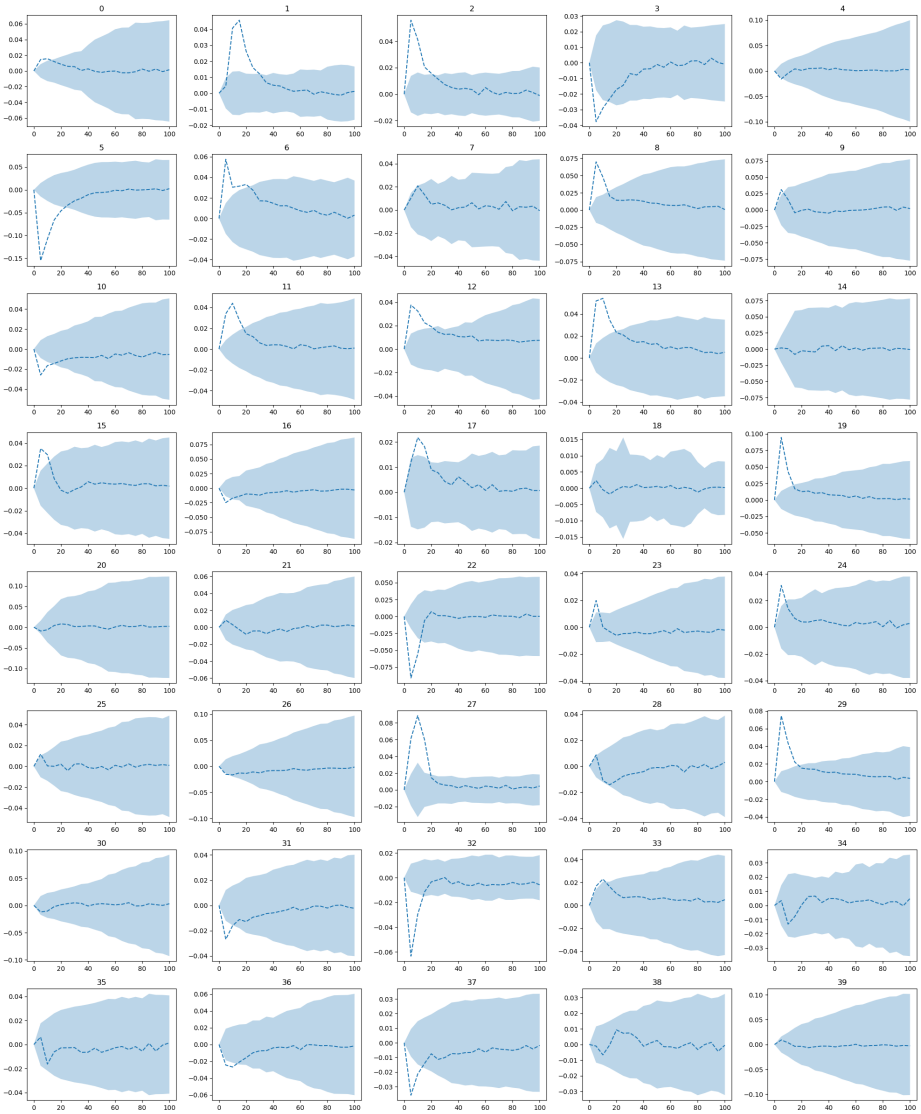

Fig. 6. Evolución de los 40 valores de orientación θ - experimento 1.

Esto sugiere un buen proceso de entrenamiento y que los filtros de la capa Gabor se integran adecuadamente con el resto de la arquitectura de la red. Los filtros de la capa Gabor entrenados se muestran en la Fig. 2. Para evaluar la convergencia de los filtros, se guardaron los valores de los parámetros $(w, \theta, \psi, \sigma)$ de la capa Gabor cada 5 épocas, durante el entrenamiento, así como los valores iniciales. El resultado se presenta en la Fig. 3, donde las líneas punteadas muestran la diferencia de la media (sobre las 35 repeticiones realizadas) de cada parámetro en esa época con respecto a su media en la época anterior.

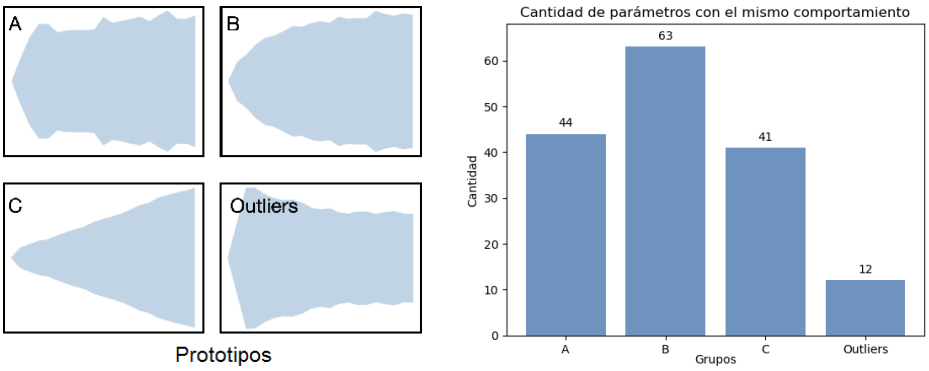

Fig. 7. Representantes (Izq.) y su frecuencia (Der.) - experimento 2.

Cuando esa línea sube, indica que en esa época el parámetro aumentó su valor con respecto a la época anterior; por el contrario, si la línea baja, indica que el valor del parámetro correspondiente disminuyó. Las regiones sombreadas representan las desviaciones estándar de cada parámetro. Esta representación visual nos permite analizar la estabilidad y convergencia de los filtros a lo largo del entrenamiento de manera precisa y detallada. En la Fig. 3, se puede apreciar que en conjunto los parámetros de la capa Gabor se estabilizan alrededor de la época 50 para el Experimento 1, esto sugiere que el entrenamiento de la capa Gabor puede detenerse antes de las 100 épocas, ya que antes de eso se llega a un punto en el que los parámetros ya no cambian significativamente.

Por otro lado, para observar la convergencia de los parámetros, es crucial centrarse en las desviaciones estándar, pues éstas reflejan la variación de los valores de los parámetros por cada época. Dado que en la Fig. 3 los filtros están superpuestos, dificultando su análisis, se muestran los filtros por separado en la Fig. 4 (para frequencia) y la Fig. 6 (para θ). Por restricciones de espacio evitamos mostrar las gráficas de los otros dos parámetros.

Analizar individualmente 160 gráficas de comportamiento (40 filtros y 4 parámetros) es muy laborioso e ineficiente, por ello, se decidió agrupar los filtros con comportamientos similares mediante un proceso de clusterización. Este enfoque permitió identificar patrones comunes entre los filtros y simplificar su análisis e interpretación de resultados. Para el proceso de clusterización se consideró solamente la información de desviaciones estándar, se aislaron los comportamientos atípicos (outliers) que pudieran sesgar los resultados y se identificaron interdependencias entre las variables del conjunto de datos.

Dadas la presencia de fuertes interdependencias, se consideró útil normalizar los datos y aplicar un Análisis de Componentes Principales (PCA) para reducir la información redundante y extraer las características más relevantes. Se concluyó que cuatro componentes son suficientes para explicar la mayor parte de la varianza. En seguida, para determinar el número óptimo de clusters, se utilizó la técnica llamada Gap Statistic, que considera la diferencia entre la suma de las dispersiones intraclase y la suma de las dispersiones interclase [14]. Se evaluó un rango de dos a seis clusters, e inicialmente se identificó el número óptimo de clusters igual a seis.

Posteriormente se aplicó el algoritmo K-means para clusterizar los datos en función de los 4 componentes principales seleccionados. Sin embargo, al analizar visualmente los resultados, se observó que algunos clusters aún mostraban gran similitud entre ellos, y tras unir los clusters similares se obtuvo como resultado final tres grupos distintos, además de los conjuntos de outliers. El proceso descrito arriba condujo a la obtención de grupos homogéneos y representativos de los comportamientos de los parámetros durante el entrenamiento. El mismo proceso se realizó de manera independiente sobre los datos del Experimento 1 y del Experimento 2.

La Fig. 5, ilustra los comportamientos representativos de cada grupo (A, B y C), así como la frecuencia de ocurrencia de los filtros, además del conjunto de outliers, para el Experimento 1. En seguida se describen brevemente cada grupo de comportamientos obtenido. El grupo A se caracteriza por tener oscilaciones pronunciadas, indicando que no hay una convergencia del parámetro a lo largo del entrenamiento. Los grupos B y C son relativamente similares, pues ambos muestran que la dispersión de los valores de los parámetros aumenta conforme avanza el entrenamiento (esto es contrario a la convergencia esperada).

Sin embargo, en el grupo B, a diferencia del C, se observan ligeras oscilaciones. En cuanto a los casos atípicos, se destaca el mostrado en la Fig. 5, donde se observa la convergencia del parámetro (en las primeras épocas la desviación incrementa hasta alcanzar un máximo y luego disminuye gradualmente); éste es el comportamiento deseado, pues muestra que el parámetro alcanzó un valor muy similar en las 35 repeticiones del experimento, reflejando estabilidad y convergencia efectiva en este caso específico.

También en la Fig. 5 se muestra la frecuencia de ocurrencia de los filtros con comportamiento dentro de cada grupo. Los outliers tienen la menor cantidad (18), de los cuales solo 2 corresponden al caso mencionado anteriormente. Los grupos B y C tienen más del doble de filtros que la clase A, es decir, en 73 % de los filtros la desviación estándar aumenta con las épocas. Este hallazgo sugiere que en cada una de las 35 repeticiones puede encontrarse un valor óptimo diferente para los parámetros de un filtro, resultando beneficioso para la red en general.

Para el Experimento 2, como ilustra la Fig. 7, los comportamientos de los parámetros pueden resumirse de forma muy similar a la del Experimento 1, indicando consistencia en los grupos identificados previamente como representativos del comportamiento de los parámetros durante el entrenamiento de la red. Destacando nuevamente los casos atípicos, en la Fig. 7 se presenta otro ejemplo en el que el parámetro muestra convergencia a lo largo del entrenamiento, aunque en este caso, el descenso de la desviación estándar no es tan pronunciado como en el ejemplo dado para el Experimento 1.

Al igual que antes, se muestra la ocurrencia de comportamientos por cada grupo, y se destaca que la mayor parte de los filtros muestran oscilaciones. De entre ellos, 44 no tienen una clara tendencia (clase A) y 63 muestran un comportamiento creciente además de oscilaciones (clase B). Finalmente, en el Cuadro 2, se presenta un resumen de los resultados obtenidos en los dos experimentos realizados. En el Experimento 1, donde se emplearon bloques GCC y CCC entrenables, se alcanzó un accuracy de 77.55 % en los datos de prueba.

Por otro lado, en el Experimento 2.a, se mantuvo el bloque GCC entrenable pero se eliminó el bloque CCC, lo que resultó en un accuracy de 61.17 %. En el Experimento 2.b, se utilizó un bloque GCC pre-entrenado y un bloque CCC entrenable, logrando un accuracy del 67.32 %. Estos resultados muestran claramente cómo diferentes configuraciones de bloques afectan el rendimiento de la red en términos de precisión en la clasificación de datos de prueba.

6. Conclusión

Los resultados obtenidos en este estudio destacan que la inclusión de filtros de Gabor en la arquitectura de una red neuronal, no conllevan a una degradación significativa del rendimiento de la red. Por el contrario, se observa que estos filtros se estabilizan rápidamente durante el proceso de entrenamiento, alcanzando una convergencia efectiva en aproximadamente 50 épocas. Este hallazgo podría ser prometedor en términos de eficiencia y eficacia para el desarrollo de modelos de aprendizaje profundo. Sin embargo, es necesario realizar más estudios para comprobar si este comportamiento es replicable en otros escenarios y contextos.

El análisis detallado de la convergencia de los filtros de Gabor se llevó a cabo mediante un enfoque de clusterización, permitiendo agrupar los filtros con comportamientos similares durante el proceso entrenamiento, simplificando así la interpretación de los resultados y proporcionando una visión más clara de la estabilidad de los filtros, esto a través de los diferentes contextos y condiciones de entrenamiento establecidos en nuestro diseño de experimentos.

Los resultados de la clusterización revelaron la existencia de tres grupos distintos de comportamientos de los filtros de Gabor, así como casos atípicos, que representan situaciones particulares en la convergencia de los parámetros. Aproximadamente el 69 % (73 % Experimento 1 y 65 % Experimento 2) de los filtros exhiben un aumento en la desviación estándar a lo largo del entrenamiento, indicando variaciones significativas en los valores de los parámetros. Esta observación es crucial, ya que sugiere que en cada repetición del experimento es posible encontrar un valor óptimo diferente para los parámetros de un filtro, lo cual podría ser beneficioso para la red, al poder explorar una gama más amplia de configuraciones para su entrenamiento.

Estos hallazgos brindan una comprensión más profunda del cómo los filtros de Gabor interactúan con una CNN y cómo su estabilidad influye en el rendimiento global. La repetibilidad de comportamientos y patrones de convergencia en diferentes experimentos sugiere que los efectos observados en los filtros de Gabor son consistentes y reproducibles, lo cual es fundamental para la validez y futura aplicabilidad de estos resultados, así como aplicaciones prácticas en aprendizaje automático y visión artificial.

7. Trabajo futuro

Se planea realizar futuras investigaciones para explorar la generalización de los resultados obtenidos en este estudio mediante la evaluación en diferentes conjuntos de datos. Al aumentar la diversidad de los datos de entrada, se respaldarán las conclusiones de este trabajo en distintos escenarios y dominios.

Además, se tiene previsto investigar el rendimiento de la capa Gabor en diferentes arquitecturas. Comparar varias arquitecturas permitirá determinar la escalabilidad y la adaptabilidad de los filtros Gabor en las CNNs. También se contempla la posibilidad de probar otros filtros espaciales además del filtro Gabor, lo cual ampliará la comprensión sobre qué filtros mejoran el rendimiento de las CNNs. Estas investigaciones planificadas contribuirán a enriquecer y fortalecer aún más los hallazgos presentados en este estudio.

Agradecimientos. Este trabajo fue apoyado por el Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) de México, a través de la Beca de Posgrado 824517 (C. Orozco) y el proyecto CÁTEDRAS-2598 (A. Rojas).

Referencias

1. Alekseev, A., Bobe, A.: Gabornet: Gabor filters with learnable parameters in deep convolutional neural network. In: International Conference on Engineering and Telecommunication, pp. 1–4 (2019) doi: 10.1109/EnT47717.2019.9030571
2. Bouvrie, J.: Notes on convolutional neural networks (2006)
3. Calderon, A., Roa, S., Victorino, J.: Handwritten digit recognition using convolutional neural netorks and Gabor filters. Proceedings of the International Congress on Computational Intelligence, pp. 1–9 (2003)
4. Khan, A., Sohail, A., Zahoora, U., Qureshi, A. S.: A survey of the recent architectures of deep convolutional neural networks. Artificial intelligence review, vol. 53, pp. 5455–5516 (2020) doi: 10.1007/s10462-020-09825-6
5. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Toronto (2009)
6. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, vol. 25 (2012) doi: 10.1145/3065386
7. Kwolek, B.: Face detection using convolutional neural networks and Gabor filters. In: International Conference on Artificial Neural Networks, pp. 551–556 (2005) doi: 10.1007/11550822_86
8. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D.: Backpropagation applied to handwritten zip code recognition. Neural computation, vol. 1, no. 4, pp. 541–551 (1989) doi: 10.1162/neco.1989.1.4.541
9. Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J.: A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems, vol. 33, no. 12, pp. 6999–7019 (2021) doi: 10.1109/TNNLS.2021.3084827
10. Luan, S., Chen, C., Zhang, B., Han, J., Liu, J.: Gabor convolutional networks. IEEE Transactions on Image Processing, vol. 27, no. 9, pp. 4357–4366 (2018) doi: 10.1109/TIP.2018.2835143
11. Ma, Y., Luo, Y., Yang, Z.: PCFnet: Deep neural network with predefined convolutional filters. Neurocomputing, vol. 382, pp. 32–39 (2020) doi: 10.1016/j.neucom.2019.11.075
12. Marchetti, G. L., Hillar, C., Kragic, D., Sanborn, S.: Harmonics of learning: Universal fourier features emerge in invariant networks. In: The Thirty Seventh Annual Conference on Learning Theory, pp. 3775–3797 (2023)
13. Meshgini, S., Aghagolzadeh, A., Seyedarabi, H.: Face recognition using gabor filter bank, kernel principle component analysis and support vector machine. International Journal of Computer Theory and Engineering, vol. 4, no. 5, pp. 767 (2012)

14. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 63, no. 2, pp. 411–423 (2001) doi: 10.1111/1467-9868.00293
15. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv (2016)
16. Zhou, Y., Ye, Q., Qiu, Q., Jiao, J.: Oriented response networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 519–528 (2017)