



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
MAESTRÍA EN CIENCIAS MATEMÁTICAS

Interpretabilidad de los Ataques Adversarios

REPORTE DEL PROYECTO FINAL
- REDES NEURONALES -

Aaron Kelley

Rodrigo Fritz

20 de junio, 2021

Abstract

The increasing use of artificial neural networks (ANNs) in security areas makes it imperative to understand the way in which these attacks act and why they are so potent. Thus, in this work we attempt to shed some light on this by experimenting and analyzing the effects that attacks, defenses, and inherent properties of the networks (such as overfitting, overparameterization, and transfer function) may have on the ability of the network to resist attacks. In addition to studying the attacks via frequency analysis, we also consider them in the context of saliency (changes to which pixels have the most effect in the classification?). Our finding suggests that as image complexity increases (MNIST vs CIFAR-10), the attacks become less noticeable and difficult to interpret. We also found some supporting evidence for the hypothesis that networks that show more resistance to adversarial attacks tend to have a transfer function more similar to that of humans (band pass). However, there is still a lot to understand and develop in order to defend ANNs adequately.

Keywords: neural networks, adversarial attacks, JPEG defense, saliency, overfitting, overparameterization, transfer function, 2D FFT

Contents

1	Introducción	2
2	Método	3
2.1	Datos	3
2.1.1	MNIST	3
2.1.2	CIFAR-10	4
2.2	Ataques	6
2.2.1	FGM y FGSM	6
2.2.2	Carlini & Wagner con norma L_2	6
2.3	Defensas	8
2.3.1	Compresión JPEG	8
3	Resultados	9
3.1	Función de transferencia de las redes	9
3.2	¿Qué hace la defensa de JPEG?	12
3.3	JPEG en CIFAR-10	13
3.4	Los efectos de overfitting y overparameterization	16
3.5	Saliency	17
4	Conclusiones	20
5	Apéndice: precisión y resumen de las redes	22

1 Introducción

Las redes neuronales profundas (DNNs) han ganado una alta reputación con respecto a sus capacidades de clasificar imágenes con tanta precisión (y en algunos casos con una precisión incluso superior) como la de los humanos. Pero es cada vez más claro que las redes aprenden a clasificar de una manera muy distinta a la de los humanos. Una de las propiedades que demostró eso fue el descubrimiento de su susceptibilidad a los ataques adversarios [13]. Estos ataques se construyen agregándoles a las imágenes una pequeña perturbación, imperceptible para los humanos, pero que engaña a la red, es decir, aunque una imagen adversaria para nosotros parezca igual a la original, la red tiene una alta probabilidad de clasificar mal a la imagen adversaria. Dado que los ataques, en general, no son detectables por los humanos, se plantea la preocupación de que puedan ser usados maliciosamente; por ejemplo, en la tecnología de reconocimiento de imágenes que se utiliza en los automóviles autónomos, por eso se requiere más profundizar en el conocimiento asociado a estos ataques adversarios.

A grandes rasgos, los ataques pueden dividirse en dos categorías: dirigidos y no dirigidos. El objetivo de los ataques no dirigidos es que cambien la clasificación correcta por cualquier otra, mientras que los ataques dirigidos buscan cambiar la clasificación correcta por una clasificación específica. En el caso de los ataques no dirigidos, generalmente se usa el gradiente de manera directa: se toma el gradiente de la función de pérdida con respecto a la imagen y eso se usa para diseñar una perturbación que maximice el error de la clasificación cuando se le agrega a la imagen. Dos ejemplos de ataques no dirigidos que utilizan el gradiente directamente son el **projected gradient descent** (PGD) [8] y el **fast gradient sign method** (FGSM) [3]. Este último toma el signo de cada elemento del gradiente en lugar de usar el valor del gradiente, esto hace que sea más rápido con grandes cantidades de datos. En este trabajo utilizamos el **fast gradient method** FGM (con norma L_2), el FGSM (FGM con norma L_∞) y el ataque con norma L_2 de Carlini y Wagner (CW) para explorar los ataques no-dirigidos. El ataque de Carlini y Wager puede ser dirigido o no-dirigido.

Cabe mencionar la diferencia entre los ataques de caja blanca y caja negra. En el primero, todos los detalles de la red (pesos, arquitectura, etc.) son conocidos por el atacante. Por el contrario, en los ataques de caja blanca, los detalles están escondidos, y solo se conocen las entradas (inputs) y las salidas (outputs). Aunque parezca muy difícil, los ataques de caja negra son bastante exitosos por la facilidad de aproximar el gradiente solo por las entradas y las salidas. Los ataques que se emplean en este artículo son los de caja blanca.

Desde el descubrimiento de la vulnerabilidad que tienen las DNNs ante los ataques adversarios, se han diseñado defensas para tratar de combatir los ataques. Aunque ninguna defensa funciona para resistir completamente los ataques, muchas sí tienen efecto, y vale la pena explorar qué propiedades contribuyen a su éxito. Al igual que los ataques, las defensas también se pueden categorizar en dos modalidades. Las del primer tipo modifican el entrenamiento de la red para que la función de pérdida se vuelva más suave. Entre más suave sea esa función, más difícil será que las perturbaciones pequeñas hagan grandes cambios. Un ejemplo conocido de este tipo es el entrenamiento adversario [3, 10, 13]. Las defensas del segundo tipo no modifican el entrenamiento ni la arquitectura, sino que utilizan un preprocesamiento en las imágenes de entrada. La defensa que se utiliza en este artículo corresponde al segundo tipo y es la compresión JPEG [2].

En este trabajo estudiamos la eficacia de la compresión JPEG en contra de los ataques y estudiamos el efecto que tienen el overfitting y la overparameterization en las redes y, dado que nuestro objetivo fue encontrar una medida de interpretabilidad de los ataques adversarios, empleamos análisis de frecuencias para observar si las redes funcionan como un filtro en el sentido de teoría de control, específicamente esperando encontrar un comportamiento del tipo pasabandas, ya que esa es más o menos la manera en que nuestra mente clasifica imágenes, y analizamos la prominencia (saliency) [12] de las imágenes para saber cuáles son las regiones en las que las redes se enfocan cuando analizan las imágenes.

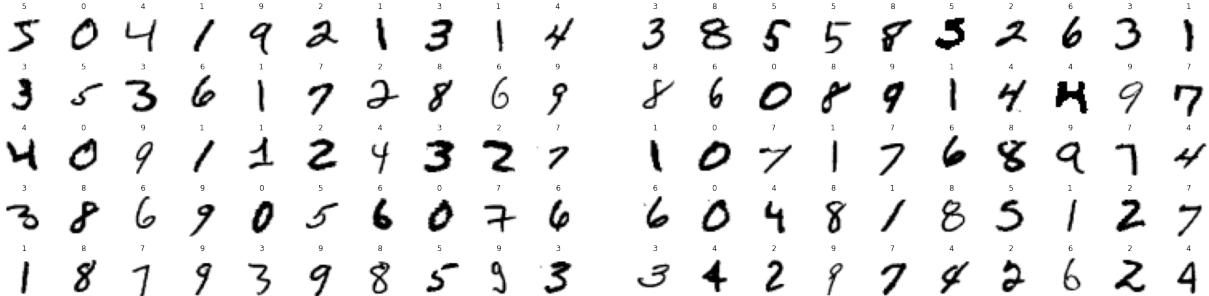
2 Método

2.1 Datos

Para entrenar nuestras redes neuronales artificiales y probar los ataques adversarios y las defensas contra ellos, se emplearon las dos bases de datos más sencillas para procesamiento de imágenes: MNIST y CIFAR-10.

2.1.1 MNIST

La base de datos MNIST es un compendio de los dígitos del 0 al 9 en letra manuscrita con 60,000 imágenes para entrenar a distintos sistemas de procesamiento de imágenes y 10,000 imágenes para evaluarlos. Se trata de un subconjunto de imágenes de un conjunto más grande que compiló el National Institute of Standards and Technology (NIST) del Departamento de Comercio los EEUU. Las siglas MNIST database significan Modified NIST database. Es una buena base de datos para probar técnicas de aprendizaje y métodos de reconocimiento de patrones con datos del mundo real empleando un esfuerzo mínimo en preprocesamiento y formato. En la Figura 1 se muestran 100 imágenes del conjunto de entrenamiento y en la Figura 2 se muestran 100 imágenes del conjunto de evaluación.

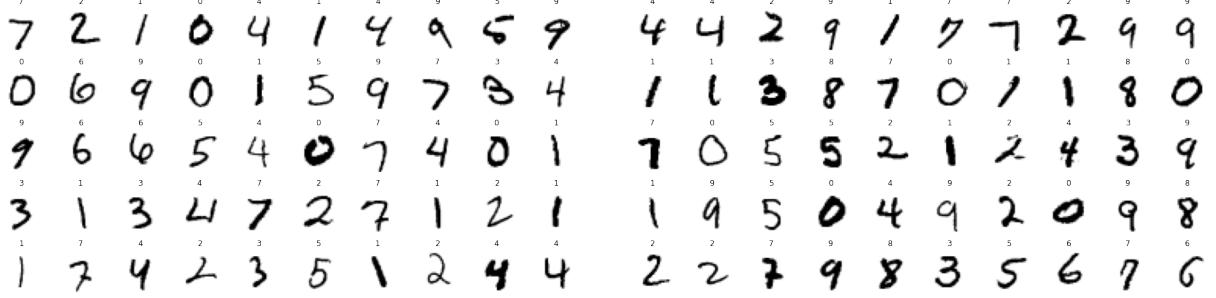


(a) Primeras 50 imágenes del conjunto `train` de MNIST con sus respectivas etiquetas.

(b) 50 imágenes aleatorias del conjunto `train` de MNIST con sus respectivas etiquetas.

Figure 1: Imágenes del conjunto de entrenamiento (train) de MNIST

De las 10,000 imágenes de evaluación, la mitad fueron escritas por estudiantes de preparatoria y la otra mitad por empleados de la oficina de censos, mientras que de las 60,000 imágenes de entrenamiento, 58,527 de los números fueron escritos por 500 estudiantes y el resto por los empleados. El tamaño de estas imágenes

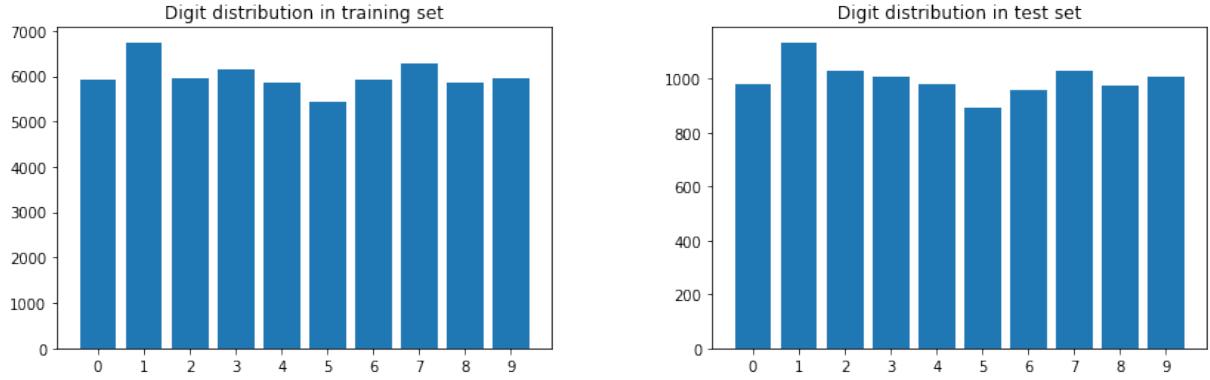


(a) Primeras 50 imágenes del conjunto `test` de MNIST con sus respectivas etiquetas.

(b) 50 imágenes aleatorias del conjunto `test` de MNIST con sus respectivas etiquetas.

Figure 2: Imágenes del conjunto de evaluación (`test`) de MNIST

en blanco y negro fue normalizado a una caja de 20×20 pixeles y se colocó su centro de masa en un campo de 28×28 [6]. Cabe recalcar que la distribución de los dígitos en MNIST no es homogénea, el número 1 es el que más se repite, como puede observarse en la Figura 3. Esto no ocurre con la base de datos de CIFAR-10, donde cada una de las 10 clases tiene exactamente el mismo número de imágenes.



(a) Distribución de los dígitos en el conjunto de entrenamiento de MNIST

(b) Distribución de los dígitos en el conjunto de evaluación de MNIST

Figure 3: Distribución de los dígitos en MNIST

2.1.2 CIFAR-10

Los grupos del MIT y la NYU recopilaron un conjunto de millones de diminutas imágenes en color de la web, se trata de un excelente conjunto de datos para el entrenamiento no supervisado de modelos generativos profundos. Se crearon dos juegos de etiquetas confiables: el conjunto CIFAR-10, que tiene 6000 ejemplos de cada una de 10 clases y el conjunto CIFAR-100, que tiene 600 ejemplos de cada una de 100 clases que no se superponen. Usando estas etiquetas, se mostró que el reconocimiento de objetos mejora significativamente al entrenar previamente una capa de características en un gran conjunto de imágenes diminutas sin etiquetar. Las siglas CIFAR vienen del Canadian Institute For Advanced Research.

Esta base de datos fue ensamblada buscando en la web imágenes de cada sustantivo en inglés no abstracto en la base de datos léxica WordNet. Utilizaron varios motores de búsqueda, incluidos Google, Flickr y

Altavista y mantuvieron aproximadamente los primeros 3000 resultados para cada término de búsqueda. Después de recopilar todas las imágenes para un término de búsqueda en particular, eliminaron duplicados perfectos e imágenes en las que una parte excesivamente grande de los píxeles eran blancos, ya que tendían a ser figuras sintéticas en lugar de imágenes naturales. El término de búsqueda utilizado para encontrar una imagen le proporciona una etiqueta aproximada, aunque es extremadamente poco confiable debido a la naturaleza de la tecnología de búsqueda de imágenes en línea. En total, el conjunto de datos contiene 80 millones de imágenes en color reducidas a 32×32 y distribuidas en 79,000 términos de búsqueda [4].

El conjunto de datos se divide en cinco lotes de entrenamiento y un lote de prueba, cada uno con 10,000 imágenes. El lote de prueba contiene exactamente 1000 imágenes seleccionadas al azar de cada clase. Los lotes de entrenamiento contienen las imágenes restantes en orden aleatorio, pero algunos lotes de entrenamiento pueden contener más imágenes de una clase que de otra. Entre ellos, los lotes de entrenamiento contienen exactamente 5000 imágenes de cada clase [5]. En la Figura 4 se muestra un ejemplo de las imágenes del conjunto de entrenamiento.



Figure 4: Imágenes aleatorias de la base de datos CIFAR-10

Las 10 clases de CIFAR-10 se encuentran en orden alfabético en la base de datos y son:

- | | | | | |
|---------------|---------|---------|----------|-----------|
| 1. airplane | 3. bird | 5. deer | 7. frog | 9. ship |
| 2. automobile | 4. cat | 6. dog | 8. horse | 10. truck |

2.2 Ataques

2.2.1 FGM y FGSM

Sean θ los parámetros de un modelo, x la entrada, y las salidas asociadas, y $J(\theta, x, y)$ la función de costo [3]. La función de costo se linealiza alrededor del valor actual de θ . Sea $\epsilon \in \mathbb{R}^+$. Definamos la imagen adversaria

$$\tilde{x} = x + \epsilon \eta_{\text{opt}}$$

Se puede definir η_{opt} por el problema de optimización

$$\eta_{\text{opt}} = \underset{\eta}{\operatorname{argmax}} \left\{ \text{grad}^\top \eta : \|\eta\|_p < \epsilon \right\}$$

Donde $p \in \mathbb{N} \cup \{\infty\}$ y $\text{grad} = \nabla_x J(\theta, x, y)$. Experimentamos con dos valores de p :

a) $p = 2$, que corresponde al Fast Gradient Method (FGM),

$$\eta_{\text{opt}} = \frac{\text{grad}}{\|\text{grad}\|}$$

b) $p = \infty$, que corresponde al Fast Gradient Sign Method (FGSM),

$$\eta_{\text{opt}} = \text{sign}(\text{grad})$$

2.2.2 Carlini & Wagner con norma L_2

Nicholas Carlini y David Wagner [1] propusieron en 2017 un método para encontrar ejemplos adversarios que tuvieran poca distorsión en la norma L^2 . Dada x , se escoge una clase t y se busca la ω que resuelva

$$\underset{\omega}{\operatorname{minimize}} \quad \left\| \frac{1}{2} (\tanh(\omega) + 1) - x \right\|^2 + c \cdot f \left(\frac{1}{2} (\tanh(\omega) + 1) \right)$$

con f definida como

$$f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$$

Esta f está basada en una función objetivo en la que se controla la confianza con la que ocurre una clasificación errónea al ajustar el valor de κ . El parámetro κ sugiere a quien esté resolviendo el problema encontrar una instancia adversaria x' que sea clasificada con mucha confianza como la clase t . Ellos toman $\kappa = 0$. $Z(x)$ es el logit para x , es decir la pre-activación de la capa de salida softmax.

En la Figura 5 se observa que los ataques con norma L_2 y norma L_∞ son prácticamente imperceptibles para nosotros, mientras que con el de norma L_0 sí se observan algunos pixeles modificados. Aunque este ataque puede ser dirigido, se usa de la forma no dirigida en este trabajo. En la Figura 6 se muestra el tipo de ruido que generan los ataques FGSM, FGM y Carlini-Wagner, los cuales son cada vez más sutiles en ese respectivo orden. Con esos tres ataques se logra que la red clasifique erróneamente un 8 por un 3.



Figure 5: Los ataques diseñados por Carlini & Wagner con cada una de las 3 normas que emplearon: L^2 , L_∞ y L_0 , mostrados en ese orden en cada columna, sobre MNIST y sobre CIFAR-10. La imagen de cada renglón es la primera de su clase en el conjunto de evaluación. Dada la etiqueta correcta l , todas las clasificaciones (erróneas) en presencia de los ataques fueron $l + 1 \pmod{10}$, i.e. fueron clasificadas con la etiqueta posterior a la correcta [1].

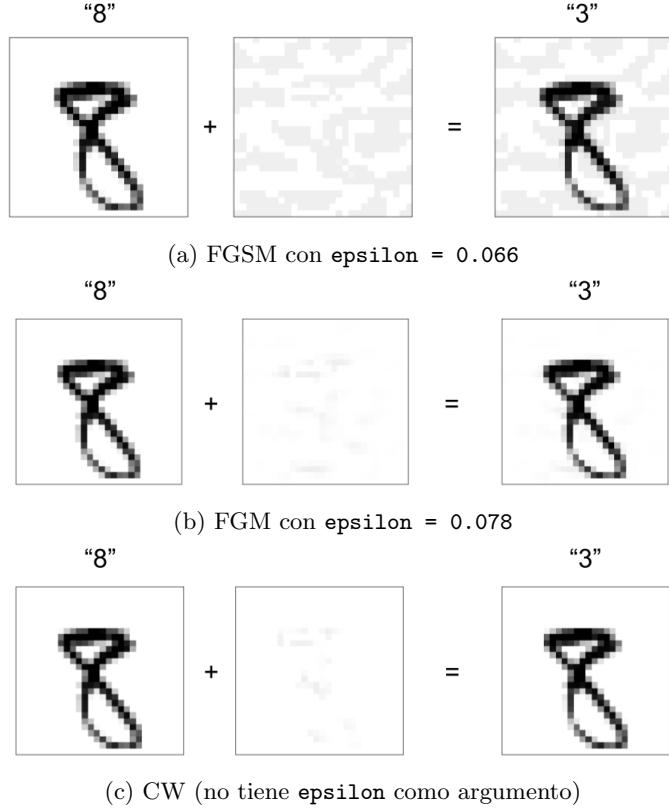


Figure 6: Ruido de los ataques FGSM, FGM y Carlini-Wagner sobre MNIST, con los cuales un 8 es clasificado erróneamente por LeNet como un 3.

2.3 Defensas

2.3.1 Compresión JPEG

Las siglas JPEG vienen de "Joint Photographic Experts Group", que es el nombre del grupo que creó el estándar en 1992, el cual es una compresión con pérdida que utiliza la transformada discreta del coseno (DCT), una técnica propuesta por Nasir Ahmed en 1972, y que normalmente elimina muchos componentes de alta frecuencia, a los que la percepción humana es menos sensible [2, 11]. Consta de los siguientes pasos:

1. Conversión de la imagen de formato RGB a formato YC_bC_r , donde el canal Y representa luminancia y los canales C_b y C_r representan crominancia. Esto está hecho porque el sistema visual humano se basa más en el contenido espacial y la agudeza que en el color para la interpretación.
2. Submuestreo espacial de los canales de crominancia en el espacio YC_bC_r : el ojo humano es mucho más sensible a los cambios de luminancia, y reducir la muestra de la información de crominancia no afecta mucho la percepción del ser humano de la imagen.
3. Dividir la imagen en bloques de 8×8 y aplicar DCT en 2D a cada bloque. Esto se hace para cada canal por separado. Este paso produce mayor compresión de los datos de la imagen.
4. Cuantificación de las amplitudes de frecuencia, lograda dividiendo cada término de frecuencia por una constante (diferente) y redondeándola al número entero más cercano. Como resultado, muchos componentes de alta frecuencia generalmente se establecen en cero y otros se reducen. La cantidad de compresión se rige por un parámetro de calidad especificado por el usuario, que define la reducción en la resolución. Aquí es donde el algoritmo JPEG logra la mayor parte de la compresión, a expensas de la calidad de la imagen. Este paso suprime más las frecuencias más altas, ya que estos coeficientes contribuyen menos a la percepción humana de la imagen.
5. Compresión sin pérdidas de los datos del bloque.

En la Figura 7 se muestran 2 ejemplos de compresión JPEG con imágenes de MNIST (Fig. 7a) y CIFAR-10 (Fig. 7b), para distintos niveles de compresión, la cual aumenta como el inverso de la calidad.

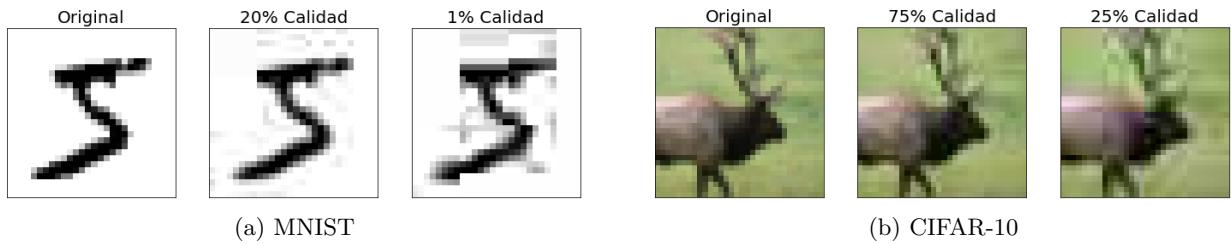


Figure 7: Distintos niveles de compresión (inverso de la calidad) JPEG

3 Resultados

3.1 Función de transferencia de las redes

Una manera útil de pensar a una red es con un punto de vista basado en teoría de control, a partir del cual construimos una función de transferencia calculando la respuesta de la red a ciertas frecuencias. Así se pueden determinar las frecuencias en las que la red tiene mayor o menor sensibilidad.

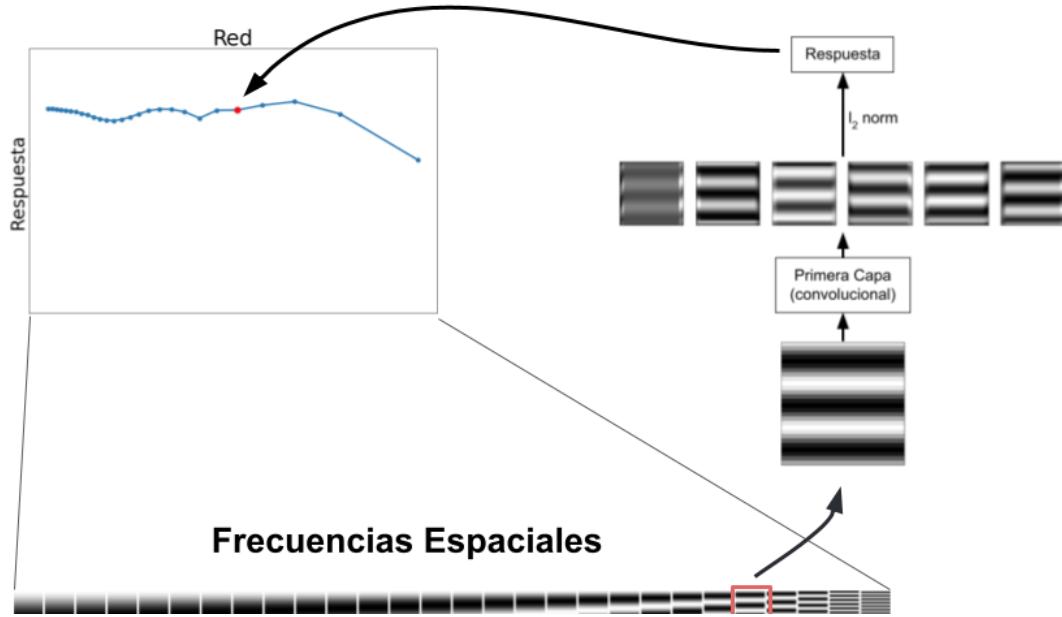


Figure 8: Diagrama de cómo se construye la función de transferencia para cierta red. Primero se generan varias frecuencias en una sola dirección (en este caso vertical) y se ingresan a la primera capa de la red. A la salida de la capa previa a la función de activación se le toma la norma L_2 , y esa es la respuesta de la red (un escalar) para tal frecuencia.

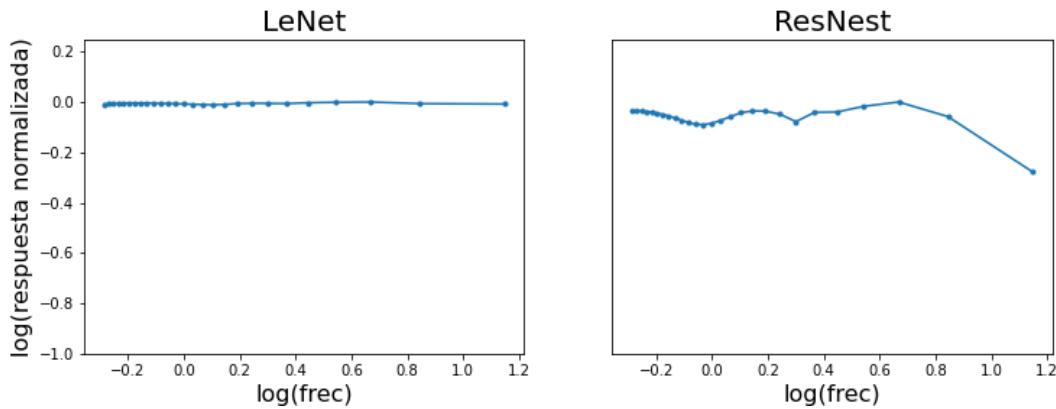


Figure 9: Funciones de transferencia para LeNet y ResNet. Las dos son más o menos igual de susceptibles a todas las frecuencias presentadas, pero ResNet sí tiene un poco de modulación negativa con cierta persistencia en las frecuencias altas y para calidades de JPEG intermedias.

En un experimento adicional, entrenamos 100 veces la misma red (LeNet) y comparamos la accuracy en cada entrenamiento como respuesta al ataque FGSM (Figura 10). Se observa que hay mucha variabilidad en el desempeño de las 100 redes generadas con la misma arquitectura. Vemos que el promedio de las funciones de transferencia de las 10 redes con el peor desempeño es mucho mas plana que el de las 10 mejores (Figuras 11 y 12). En la Figura 13 se muestra un GIF en el que se puede ver cómo cambia la función de transferencia al mover el promedio a lo largo de la distribución de las redes.

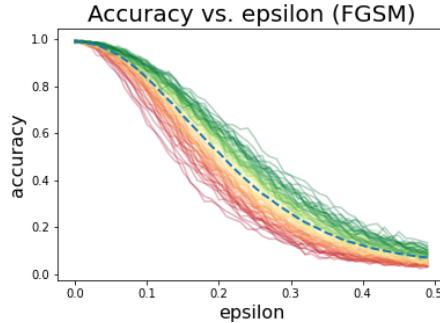


Figure 10: Accuracy de los 100 entrenamientos para LeNet sometiendo las imágenes de MNIST al ataque FGSM con distintos épsilons.

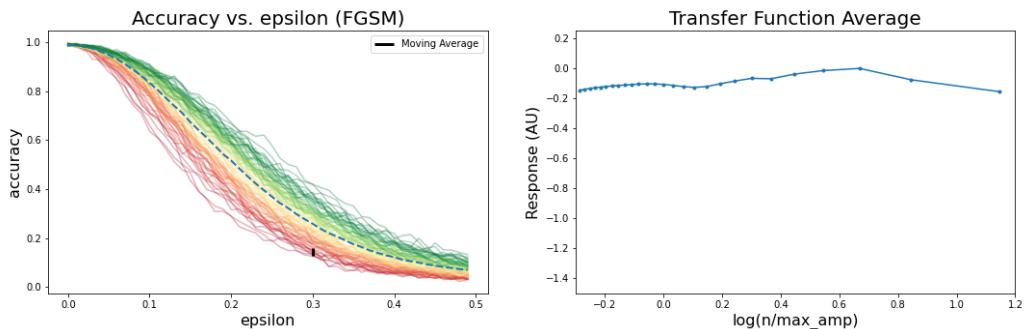


Figure 11: Promedio de las funciones de transferencia de las 10 redes con el peor desempeño (en rojo en la gráfica de la izquierda) con respecto a los ataques.

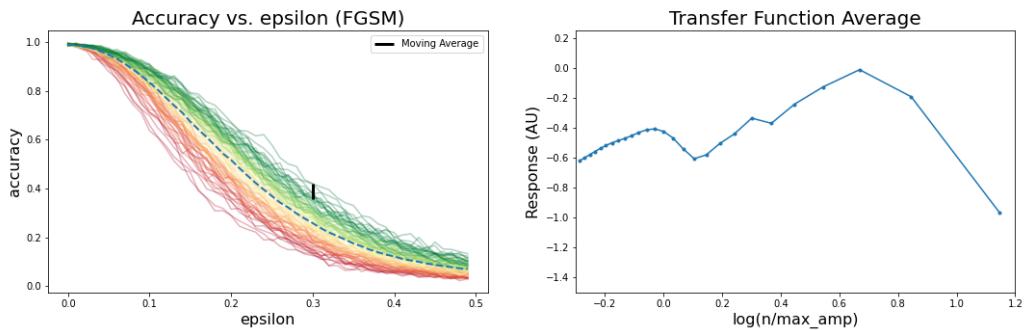


Figure 12: Promedio de las funciones de transferencia de las 10 redes con el mejor desempeño (en verde en la gráfica de la izquierda) con respecto a los ataques.

Figure 13: Este GIF muestra cómo cambia la función de transferencia conforme el promedio móvil de 10 redes avanza en accuracy en orden ascendente.

3.2 ¿Qué hace la defensa de JPEG?

Como vimos en la sección de compresión JPEG, cuando esta es aplicada a una imagen que ha sido sometida a un ataque adversario, los componentes de alta frecuencia se eliminan y, así como los humanos percibimos en menor medida dichos componentes, es posible que la red no logre distinguir el ataque perpetrado sobre la imagen, esto debido a que los ataques suelen tratarse de pequeñas cantidades de ruido o distorsión introducidos en la imagen y, al reducir la calidad de la misma, la red tiene más probabilidades de recuperar la clasificación original de la imagen. Esto se encuentra esquematizado en la Figura 14.

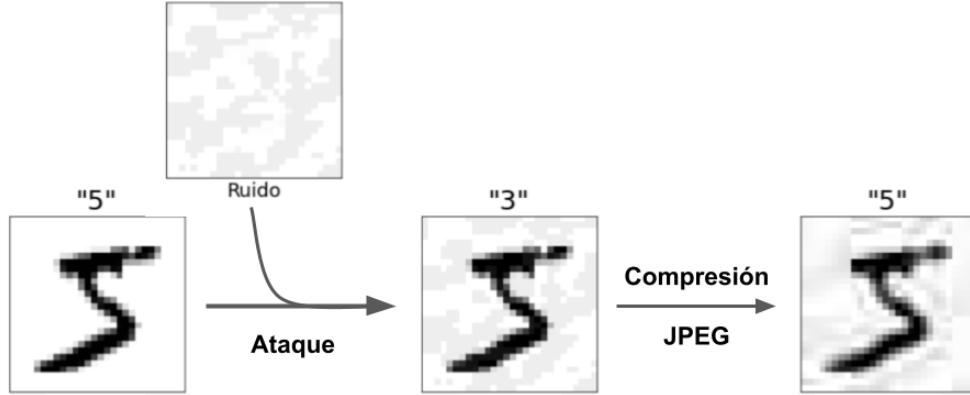


Figure 14: La defensa JPEG radica en contrarrestar el ruido agregado por el ataque adversario al remover los componentes de alta frecuencia, que son los causantes de la falla en la clasificación de la red. En este ejemplo, el dígito 5 es clasificado como un 3 con el ruido del ataque, pero al hacer una compresión JPEG, la red clasifica correctamente al dígito como un 5.

En la Figura 15 se muestra el resultado de nuestro entrenamiento de las redes LeNet y ResNet utilizando MNIST y el ataque FGSM con la norma L_∞ para distintos niveles de compresión JPEG como defensa. Se observa que, efectivamente, conforme aumenta la compresión (disminuye la calidad), la precisión (accuracy) en la clasificación de las imágenes por parte de la red es mayor. Nótese que con ResNet la precisión decae más rápidamente conforme aumenta el ϵ .

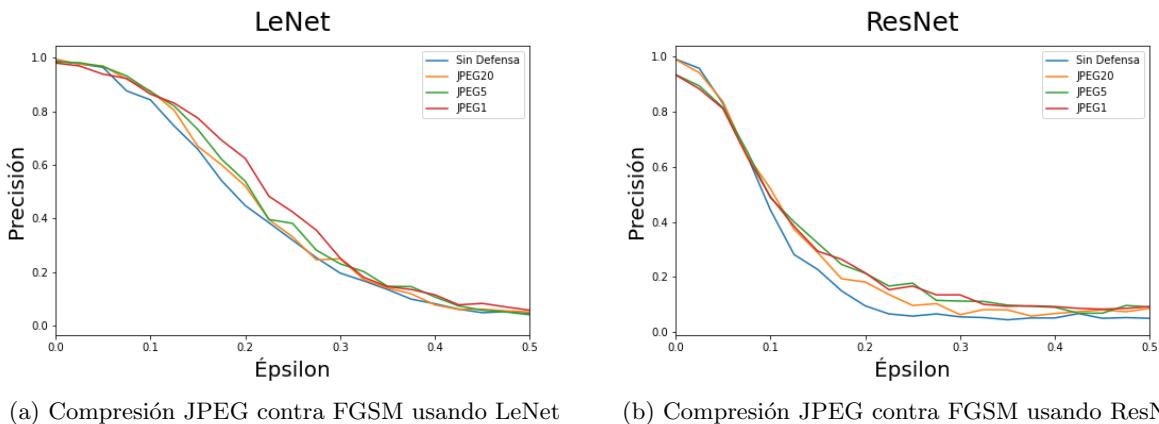


Figure 15: Resultado de la compresión JPEG como defensa ante el ataque FGSM sobre MNIST

En la Figura 16 se observa cómo cambian las funciones de transferencia de cada red cuando las imágenes pasan por distintos niveles de compresión JPEG según el siguiente esquema:

frecuencia → compresión JPEG → 1^{era} capa (convolucional) → respuesta

Se nota que no hay mucho cambio en LeNet para ninguna calidad de compresión. Con respecto a ResNet, sí hay modulación con algunas calidades bajas, más o menos en el rango de 7 a 16, pero no se observa un gran cambio en el desempeño de la red. Cabe mencionar que las calidades más bajas (1 – 5) destruyeron completamente cada frecuencia que pasamos por la red, y por eso no hay ningún cambio en la función de transferencia para esas calidades. Pensamos que eso es un artefacto del tamaño de las imágenes y no se debe tomar al pie de la letra. Asimismo, por el tamaño de las imágenes, hay baja resolución frecuencial, y eso también puede afectar los resultados. Este análisis debe de ser repetido para imágenes más grandes.

Figure 16: Estos GIFs fueron hechos de la misma manera que la Figura 9, pero las frecuencias fueron comprimidas con JPEG antes de pasarlas por la red.

3.3 JPEG en CIFAR-10

Utilizando LeNet, con los ataques FGSM (Figura 17), que tiene norma L_∞ , y FGM (Figura 18), que tiene norma L_2 , se tiene una clasificación errónea de la imagen `train[0]` del conjunto CIFAR-10 a partir del épsilon indicado en la figura. Como en el FGSM cada pixel tiene una mayor intensidad, no se requiere un épsilon tan grande como el de FGM. En la Figura 19a se observa la eficacia de la defensa JPEG contra el ataque FGSM para distintas calidades de la imagen: mientras menor es la calidad, mayor es la precisión de la red al clasificar la imagen. Nótese que hay una desventaja en usar JPEG porque disminuye la precisión de la red entre más baja sea la calidad (Figura 19b).

En la Figura 20 se muestran las transformadas rápidas de Fourier (FFTs) en 2D de los ruidos generados por el FGSM y el FGM, y se observa que ambas FFTs son muy similares a pesar de que los ruidos parecen distintos por la capa gris del FGM. Esto se debe a que, con FGM, los valores de la intensidad del ruido están en el intervalo $(-1, 1)$, pero en FGSM solo pueden ser -1 o 1 , entonces los colores son de menor intensidad en FGM: el gris en RGB es (r, g, b) cuando $r \approx g \approx b < 1$.



Figure 17: FGSM sobre CIFAR-10

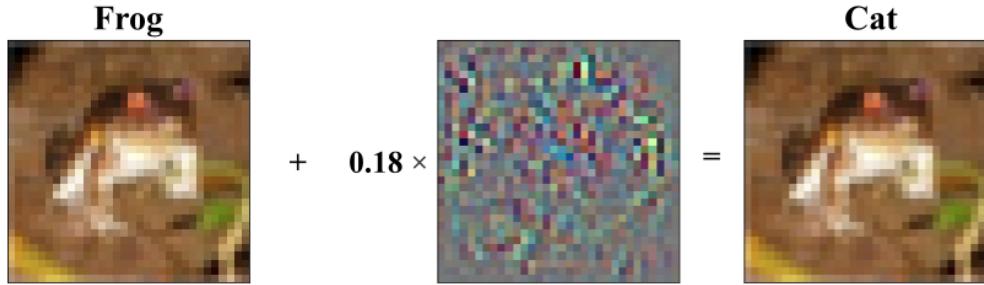
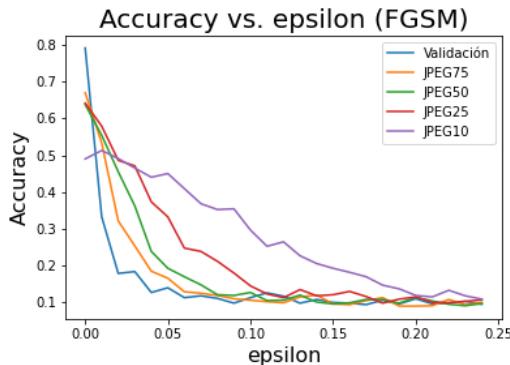
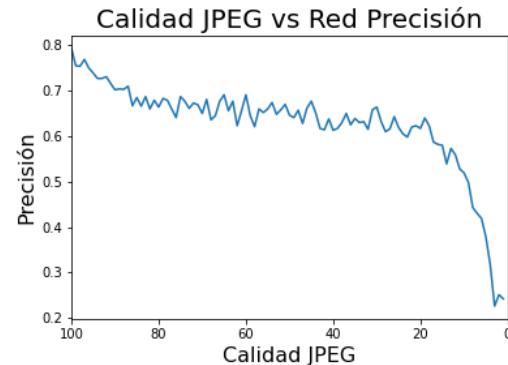


Figure 18: FGM sobre CIFAR-10



(a) Precisión de LeNet para distintos valores de la calidad de la imagen en la compresión JPEG



(b) Precisión de LeNet en función de la calidad de la imagen en la compresión JPEG

Figure 19: Eficacia de JPEG

En la Figura 21 se muestra cómo la FFT de la imagen original y la imagen atacada parecen exactamente iguales debido a que entre más pixeles y más compleja sea la imagen, la adición del ruido es más sutil, además de que el valor mínimo del épsilon adversario es menor con las imágenes de color. En cambio, con MNIST la adición del ruido sí es observable en la FFT (Figura 22).

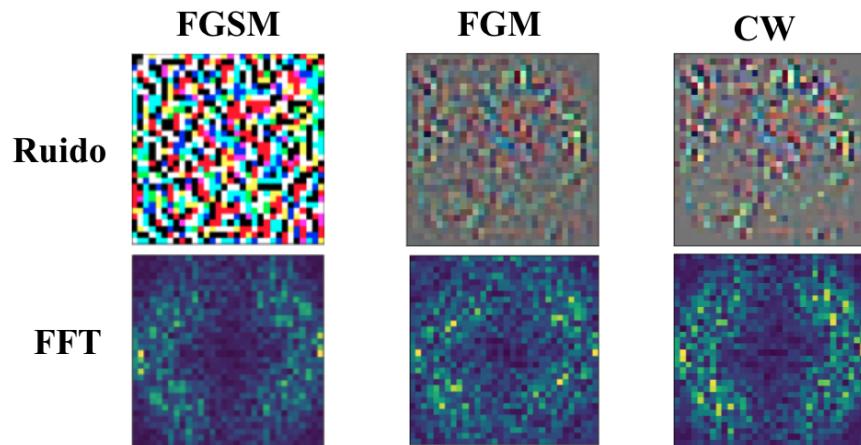


Figure 20: La FFT de los ruidos de FGSM, FGM y CW

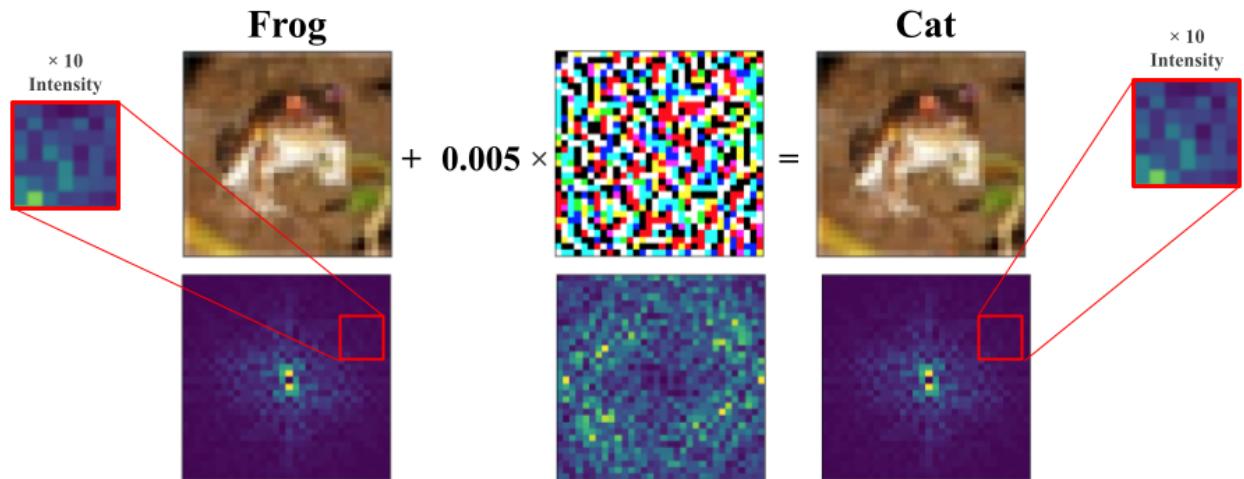


Figure 21: Comparación de la FFT de la imagen original y de la imagen con el ruido agregado por FGSM

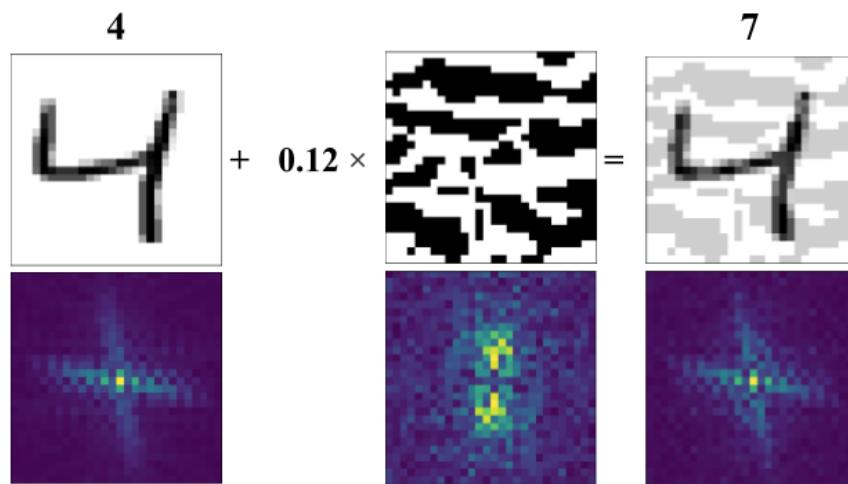


Figure 22: Con MNIST sí se observa una pequeña contribución del ruido en la FFT de la imagen adversaria.

3.4 Los efectos de overfitting y overparameterization

Se ha sugerido que algunas propiedades de ciertas redes pueden contribuir a su susceptibilidad a los ataques adversarios. Por ejemplo, en el contexto de las imágenes médicas y sus respectivas redes, se propone que el sobreajuste (overfitting) y la sobreparametrización (overparameterization) pueden actuar de esa manera [7]. El *overfitting* ocurre cuando el modelo clasifica bien los datos de entrenamiento, pero no se generaliza tan bien, es decir, en los datos de validación tiene peor desempeño. La *overparameterization* tiene lugar cuando hay tantos parámetros que la red es capaz de “memorizar” completamente los datos. Hicimos un pequeño análisis para probar esta hipótesis y no encontramos resultados que la apoyen. De hecho, con las redes que probamos, encontramos lo opuesto. En general, el overfitting y la overparameterization hacen que la red sea más robusta a los ataques adversarios (Figure 23).

Para explorar el overfitting, la red LeNet fue modificada al ser entrenada con 5 cantidades distintas de épocas (epochs). Mientras más épocas, mayor es la probabilidad de que la red haya sido sobreajustada (overfit). Para explorar la overparameterization, la red fue entrenada con distintas cantidades de parámetros, agregando un cierto número de capas (layers) adicionales directamente antes de la última, cada capa adicional con 256 nodos (Tabla 1).

Red	Número de parámetros
Normal	61,706
2 Capas Adicionales	150,978
4 Capas Adicionales	282,562
6 Capas Adicionales	414,146
10 Capas Adicionales	743,106
20 Capas Adicionales	1,335,234

Tabla 1: Número de parámetros con cada conjunto de capas adicionales en la red LeNet.

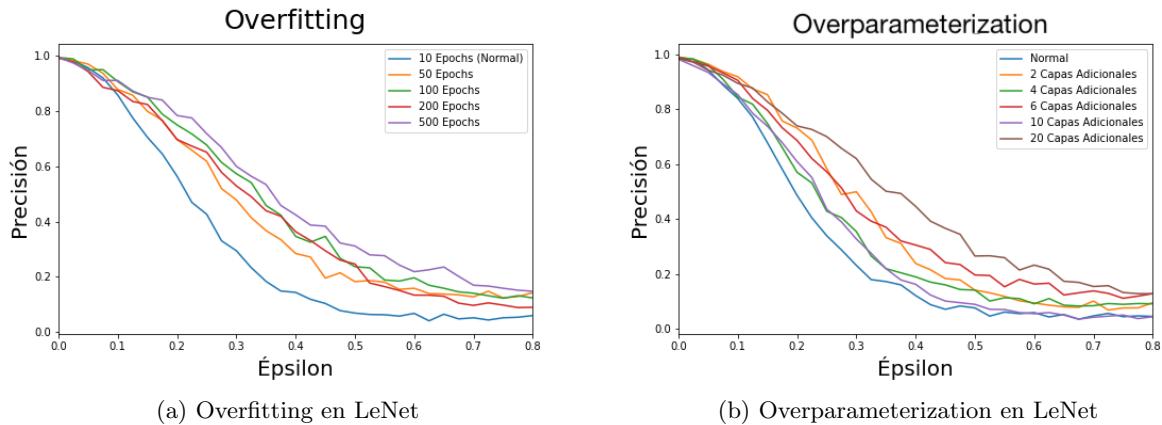


Figure 23: a) El overfitting consiste en tener exceso de épocas. Aquí se observa que conforme aumenta el número de épocas, la precisión aumenta con respecto a cada épsilon. b) La overparameterization consiste en exceso de capas. En general, aumentar el número de capas aumenta la precisión.

3.5 Saliency

Figure 24: GIF del saliency en una imagen de CIFAR-10 conforme aumenta ε de 0 a 0.099 en el FGSM

Figure 25: GIF del saliency en otra imagen de CIFAR-10 conforme aumenta ε de 0 a 0.099 en el FGSM

La puntuación (score) $S(I)$ de una imagen es la salida de la capa de la red neuronal justo antes de que se convierta en probabilidades (esto sucede en la función de activación softmax). El procedimiento para obtener la prominencia (**saliency**) de la imagen es el siguiente:

Dada una imagen I_0 y una clase c , sea $S_c(I_0)$ la puntuación de la clase c .

Nos gustaría rankear a los pixeles de I_0 con base en su influencia sobre la puntuación $S_c(I_0)$.

Podemos aproximar $S_c(I)$ con una función lineal en la vecindad de I_0 al calcular la expansión de Taylor a primer orden,

$$S_c(I) \approx w^\top I + b,$$

donde w es la derivada de S_c con respecto de la imagen I en el punto (imagen) I_0 ,

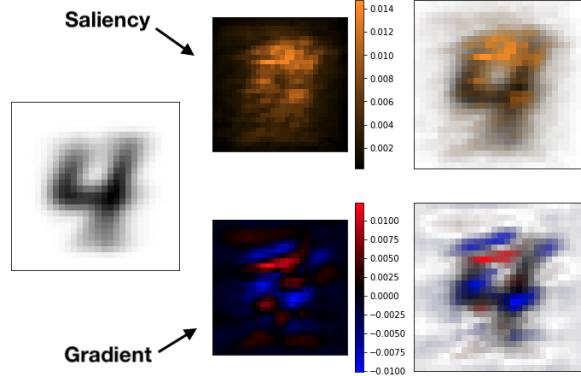
$$w = \left. \frac{\partial S_c}{\partial I} \right|_{I_0}.$$

La magnitud de la derivada nos indica cuáles son los pixeles que deben ser cambiados en la menor cantidad para afectar la puntuación de la clase en la mayor medida [12]. Es decir, la prominencia (saliency) M se define como

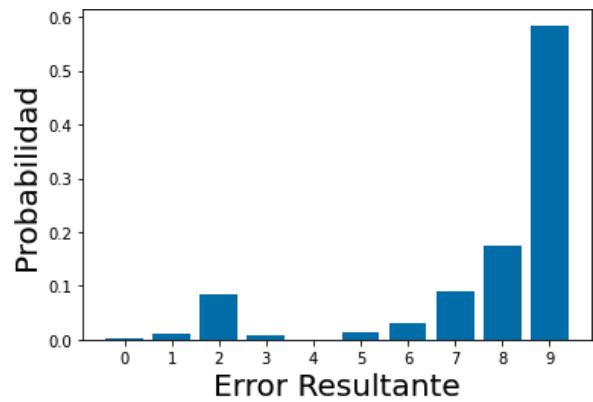
$$M_{ij} = |w_{i,j}|$$

y para datos multi-canal (por ejemplo RGB), se toma el máximo valor de los canales,

$$M_{ij} = \max_c |w_{i,j,c}|$$

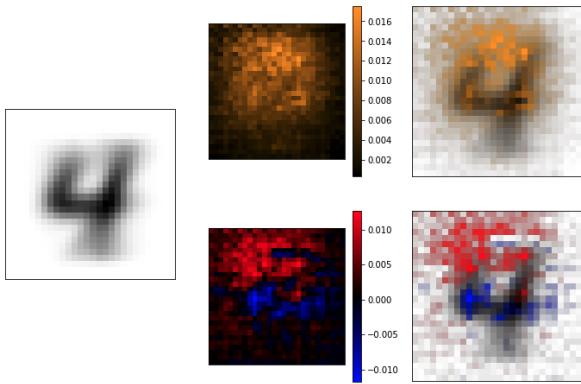


(a) Saliency y gradient del promedio de las imágenes con 4

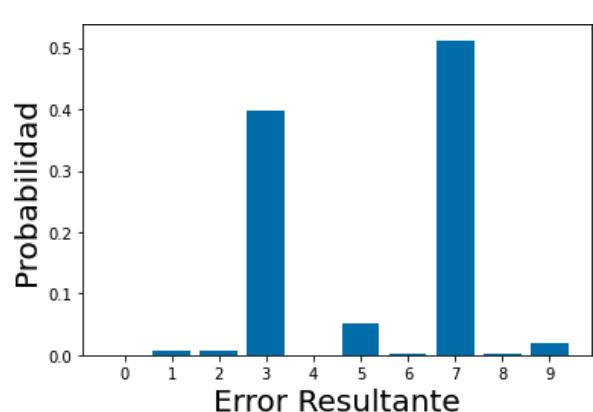


(b) Distribución de la clasificación del promedio del 4

Figure 26: Comparación del saliency y el gradient del promedio de las imágenes con 4 del conjunto `test` de MNIST junto con la distribución de la clasificación, empleando LeNet



(a) Saliency y gradient del promedio de las imágenes con 4

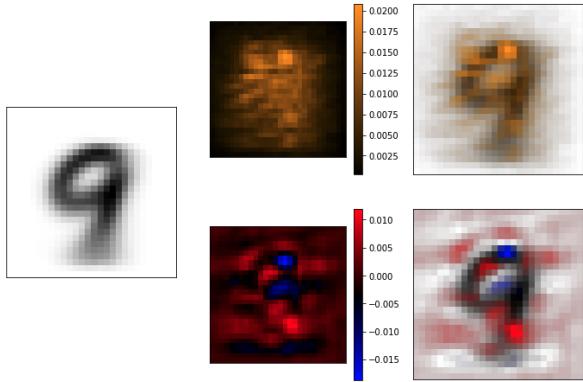


(b) Distribución de la clasificación del promedio del 4

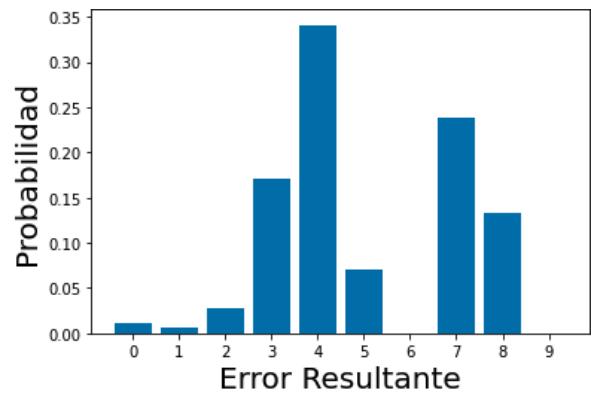
Figure 27: Comparación del saliency y el gradient del promedio de las imágenes con 4 del conjunto `test` de MNIST junto con la distribución de la clasificación, empleando ResNet

En las figuras 24 y 25 se muestran dos GIFs con la respuesta del saliency conforme aumenta ε de 0.000 a 0.099 en el FGSM. Se observa que los máximos del saliency no parecen tener una ubicación particularmente significativa para el reconocimiento de la imagen, lo cual resulta contrario a lo esperado.

En las figuras 26, 27, 28 y 29 se muestra el saliency y el gradiente de los promedios de todas las imágenes con 4 y 9 de MNIST, junto con la probabilidad de distribución de la clasificación de cada uno. En las figuras 26 y 28 se empleó LeNet y en las 27 y 29 se empleó ResNet. Resulta interesante que en el caso del 4, LeNet lo clasificó mayormente como un 9, mientras que ResNet lo clasificó en primer lugar como 7 y en segundo lugar como 3, estos tres dígitos como opciones razonables para que una persona confunda al 4, pero en especial el 9, por lo que concluimos que, en este caso, LeNet tuvo un mejor desempeño que ResNet. Se observa además que en los máximos de saliency se encuentran en la parte superior del 4, que es justo la región en la cual, si cubriéramos ese hueco, el 4 se convertiría en un 9. En el caso del gradiente, tanto con LeNet como con ResNet, los máximos también se concentran en la parte superior del 4, donde de nuevo LeNet tiene un patrón más definido. Los mínimos del gradiente con LeNet se ubican en las esquinas del cuadrado que conforma la

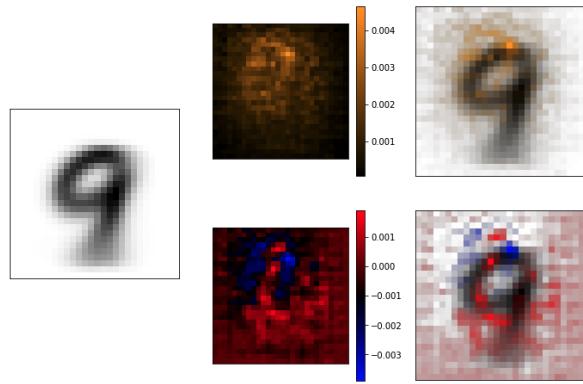


(a) Saliency y gradient del promedio de las imágenes con 9

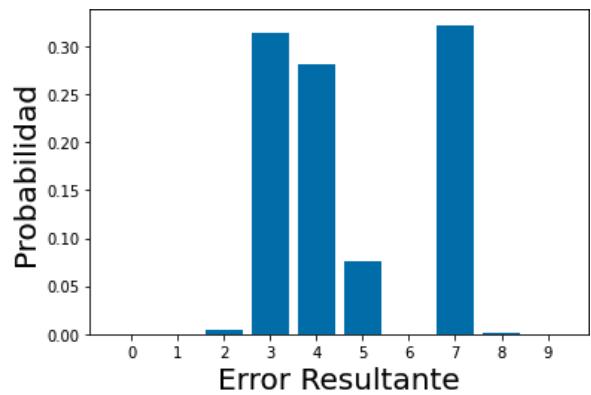


(b) Distribución de la clasificación del promedio del 9

Figure 28: Comparación del saliency y el gradient del promedio de las imágenes con 9 del conjunto `test` de MNIST junto con la distribución de la clasificación, empleando LeNet



(a) Saliency y gradient del promedio de las imágenes con 9



(b) Distribución de la clasificación del promedio del 9

Figure 29: Comparación del saliency y el gradient del promedio de las imágenes con 9 del conjunto `test` de MNIST junto con la distribución de la clasificación, empleando ResNet

mitad superior del 4, mientras que con ResNet se ubican principalmente en la barra de en medio del 4, lo cual concuerda con la imagen del promedio de todos los dígitos 4 de MNIST.

En el caso del 9, se observa justo lo mismo, que el saliency se concentra en la parte superior del 9, que es la región en la cual, al remover esa parte, el 9 se convertiría en un 4. Esto ocurre tanto con LeNet como con ResNet, sin embargo, LeNet efectivamente lo clasificó en una clara mayoría como un 4, de nuevo con un mejor desempeño que ResNet, la cual lo clasificó mayormente como 7, seguido de 3 (al igual que en la clasificación del 4), seguido ahora sí por el 4. Es interesante que con el 9 hubo mayor concordancia entre LeNet y ResNet que con el 4, pues LeNet también tuvo clasificaciones de 7, 3 y 5, donde el 8 resulta ser la única variante. En cuanto al gradiente, el máximo con LeNet es claramente la parte inferior del 9, mientras que con ResNet los máximos se distribuyen alrededor del centro del 9, lo cual no parece ofrecer información importante y además no coincide con lo que se observa en el gradiente del 4.

4 Conclusiones

En este trabajo se emplearon y analizaron 3 redes neuronales: LeNet, ResNet y una específica para las imágenes de CIFAR-10, donde LeNet fue ligeramente modificada al agregar capas extra para estudiar la overparameterization, y se emplearon los datos de MNIST y CIFAR-10, en el contexto de las imágenes adversarias, con lo que probamos 3 ataques: FGSM, FGM y CW, y una defensa: la compresión JPEG. Vimos que esta defensa efectivamente aumentó la resistencia de la red a los ataques adversarios, pero no por mucho. También existe un compromiso entre la precisión de la red y la calidad de compresión, pues la defensa JPEG tuvo mas éxito con los datos de CIFAR-10 que con los de MNIST, pero también tiene la desventaja de bajar la precisión de la red con rapidez al disminuir la calidad de la imagen con la compresión JPEG, es decir, sin el ruido del ataque adversario.

Las redes que se entrenaron con MNIST tienen una función de transferencia bastante plana, lo cual significa que son igual de sensibles a las frecuencias altas y a las bajas. También vimos que el ruido adversario tiene más frecuencias altas presentes. Entonces una hipótesis interesante es que una red que sea menos sensible a las frecuencias altas será más resistente a los ataques adversarios. No obstante, con las imágenes de CIFAR-10, vimos que el ruido es tan sutil que ni siquiera aparece en la FFT de la imagen adversaria. Parece que entre más compleja sea la imagen (más pixeles), más sutil puede ser el ataque. Al agregar compresión JPEG como una primera capa antes de obtener las funciones de transferencia, no hubo una gran diferencia en la curva resultante. Eso podría haber sido un artefacto del pequeño tamaño de los datos de MNIST. Falta repetir el análisis de las funciones de transferencia con la red de CIFAR-10.

Contrario a lo reportado en la literatura, en el contexto de imágenes médicas [7], observamos que nuestras redes con overfitting y overparameterization tienen mayor resistencia a los ataques adversarios. Esto podría deberse a la complejidad de las imágenes médicas, pero habría que continuar con el análisis en esa dirección.

El análisis de saliency con los datos de MNIST demostró que con datos así de sencillos, en general, las partes que son más atacadas por el ruido adversario son las partes que al ser modificadas transforman a un número por otro, por ejemplo, al quitar la parte de arriba a la izquierda de un 9 se obtiene un 3. También se nota una diferencia entre el saliency de la red con topología lineal (LeNet) y la red con topología no lineal (ResNet). Hay más regiones con intensidad alta en el gradiente de ResNet que en el de LeNet. Esto podría ayudar a explicar por qué la red con topología no lineal es más susceptible a los ataques.

El análisis de saliency en los datos de CIFAR-10 no es tan intuitivo, probablemente debido a que las imágenes son un poco más complejas que las de MNIST, pero sí se observa una variación en la distribución de saliency conforme aumenta el ruido adversario.

References

- [1] Nicholas Carlini and David Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2017. arXiv: 1608.04644 [cs.CR].
- [2] Nilaksh Das et al. *Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression*. 2017. arXiv: 1705.02900 [cs.CV].
- [3] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].
- [4] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.
- [5] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *The CIFAR-10 website*. 2009. URL: %5Curl%7Bhttps://www.cs.toronto.edu/~kriz/cifar.html%7D.
- [6] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist> 2 (2010).
- [7] Xingjun Ma et al. *Understanding Adversarial Attacks on Deep Learning Based Medical Image Analysis Systems*. 2020. arXiv: 1907.10456 [cs.CV].
- [8] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML].
- [9] Fares Sayah. *Cifar-10 Image Classification using CNNs*. 2020. URL: %5Curl%7Bhttps://www.kaggle.com/faressayah/cifar-10-image-classification-using-cnns-88%7D.
- [10] Uri Shaham, Yutaro Yamada, and Sahand Negahban. “Understanding adversarial training: Increasing local stability of supervised models through robust optimization”. In: *Neurocomputing* 307 (Sept. 2018), pp. 195–204. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2018.04.027. URL: <http://dx.doi.org/10.1016/j.neucom.2018.04.027>.
- [11] Uri Shaham et al. *Defending against Adversarial Images using Basis Functions Transformations*. 2018. arXiv: 1803.10840 [stat.ML].
- [12] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: 1312.6034 [cs.CV].
- [13] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

5 Apéndice: precisión y resumen de las redes

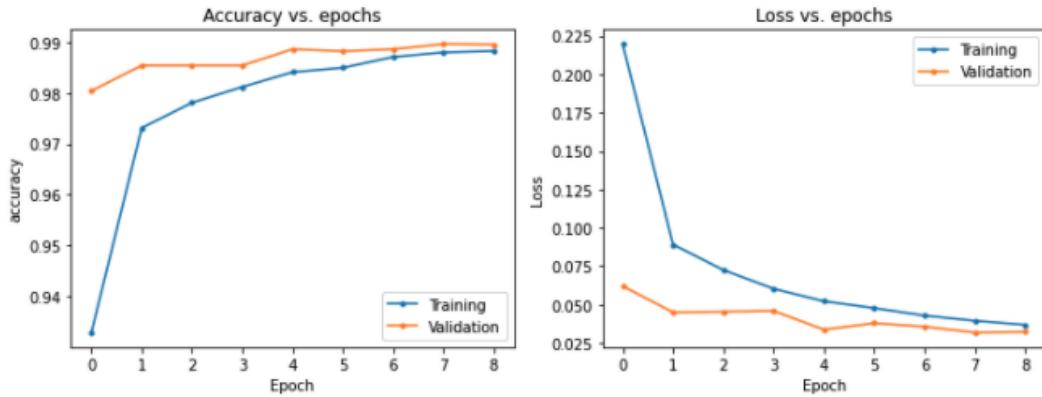


Figure 30: Precisión y pérdida de LeNet para MNIST

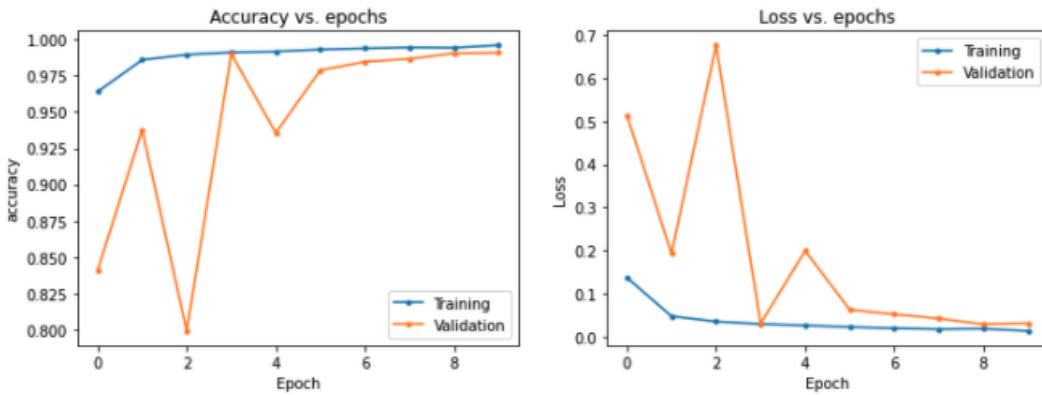


Figure 31: Precisión y pérdida de ResNet para MNIST

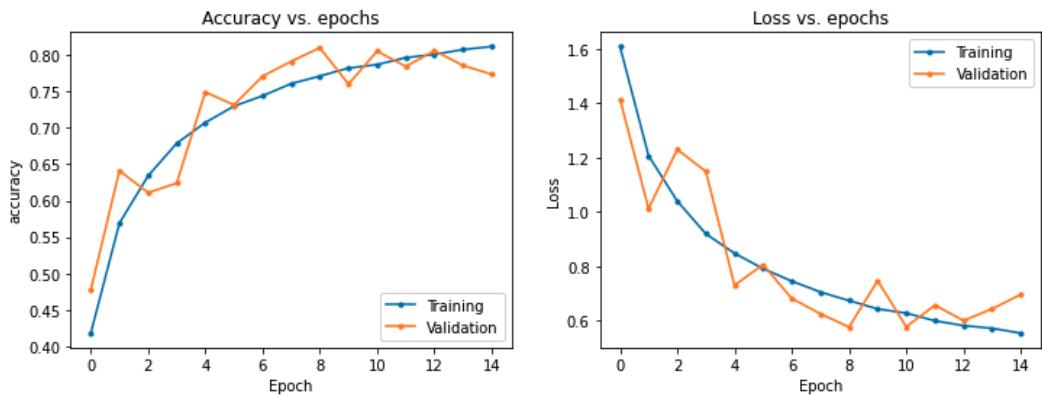


Figure 32: Precisión y pérdida de la red para CIFAR-10

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 28, 28, 6)	156
activation_30 (Activation)	(None, 28, 28, 6)	0
average_pooling2d_3 (Average)	(None, 14, 14, 6)	0
conv2d_7 (Conv2D)	(None, 10, 10, 16)	2416
activation_31 (Activation)	(None, 10, 10, 16)	0
max_pooling2d_3 (MaxPooling2)	(None, 5, 5, 16)	0
flatten_3 (Flatten)	(None, 400)	0
dense_24 (Dense)	(None, 120)	48120
activation_32 (Activation)	(None, 120)	0
dropout_3 (Dropout)	(None, 120)	0
dense_25 (Dense)	(None, 84)	10164
activation_33 (Activation)	(None, 84)	0
dense_26 (Dense)	(None, 10)	850
activation_34 (Activation)	(None, 10)	0
<hr/>		
Total params:	61,706	
Trainable params:	61,706	
Non-trainable params:	0	

Figure 33: Resumen de LeNet para MNIST

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d (Conv2D)	(None, 28, 28, 64)	3200
batch_normalization (BatchNo)	(None, 28, 28, 64)	256
activation (Activation)	(None, 28, 28, 64)	0
max_pooling2d (MaxPooling2D)	(None, 9, 9, 64)	0
identity_block (IdentityBloc)	(None, 9, 9, 64)	74368
identity_block_1 (IdentityBl)	(None, 9, 9, 64)	74368
global_average_pooling2d (G1)	(None, 64)	0
dense (Dense)	(None, 10)	650
<hr/>		
Total params:	152,842	
Trainable params:	152,202	
Non-trainable params:	640	

Figure 34: Resumen de ResNet para MNIST

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization_2 (Batch Normalization)	(None, 32, 32, 32)	128
conv2d_6 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_3 (Batch Normalization)	(None, 32, 32, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv2d_7 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_4 (Batch Normalization)	(None, 16, 16, 64)	256
conv2d_8 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_5 (Batch Normalization)	(None, 16, 16, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_2 (Dropout)	(None, 8, 8, 64)	0
conv2d_9 (Conv2D)	(None, 8, 8, 128)	73856
batch_normalization_6 (Batch Normalization)	(None, 8, 8, 128)	512
conv2d_10 (Conv2D)	(None, 8, 8, 128)	147584
batch_normalization_7 (Batch Normalization)	(None, 8, 8, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_3 (Dropout)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_3 (Dense)	(None, 128)	262272
dropout_4 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 10)	1290
<hr/>		
Total params: 552,362		
Trainable params: 551,466		
Non-trainable params: 896		

Figure 35: Resumen de la red específica para CIFAR-10 [9]