



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
MAESTRÍA EN CIENCIAS MATEMÁTICAS

Interpretabilidad de los Ataques Adversarios

REPORTE DEL PROYECTO FINAL
- REDES NEURONALES -

Aaron Kelley

Rodrigo Fritz

18 de junio, 2021

Abstract

Abstract Goes here

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

Índice

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	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?	10
3.3. JPEG en CIFAR-10	11
3.4. Los efectos de overfitting y overparameterization	12
3.5. Saliency	14
4. Conclusión/Discusión	16
5. Apéndice A: Precisión y resumen de las redes para CIFAR-10 y MNIST	18

1. Introducción

Las redes neuronales profundas (DNNs) han ganado una alta reputación con respecto a sus capacidades de clasificar imágenes igual (o hasta superior) que los humanos. Pero en el pasado reciente, ha quedado cada vez más claro que las redes aprenden a clasificar de manera muy distinta que los humanos. Una de las propiedades que realmente demostró eso fue el descubrimiento de su susceptibilidad a los ataques adversarios [13]. Esos ataques se construyen agregándoles a las imágenes una pequeña perturbación imperceptible para los humanos que engaña a la red. Es decir, aunque una imagen adversaria para nosotros parezca igual a la original, la red se equivoca con la clasificación de la adversaria con alta probabilidad. Como los ataques pueden no ser detectables por los humanos, se plantea la preocupación 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 profundización del conocimiento asociado.

A grandes rasgos, los ataques pueden dividirse entre dos categorías: dirigidos y no-dirigidos. Primero hablemos de los ataques dirigidos; muchos usan el gradiente de manera directa. La idea es que se toma el gradiente de la función de pérdida con respecto a la imagen, y eso se usa para diseñar una perturbación que maximiza 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 saca el signo de cada elemento del gradiente en lugar de usar el gradiente verdadero; eso hace que sea más rápido con grandes cantidades de datos. En este artículo se usan tanto PGD como FGSM para explorar los ataques no-dirigidos. El objetivo de los ataques no dirigidos es que cambien la clasificación correcta a cualquier otro ataque, mientras que los ataques dirigidos tienen una clasificación deseada a la que cambian la clasificación correcta. Se examina el ataque Carlini & Wagner (CW), el cual puede actuar como 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 son 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 esa vulnerabilidad que tienen las DNN, 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 esa función, más difícil será para que las perturbaciones pequeñas hagan cambios grandes. Un ejemplo conocido de este tipo es el entrenamiento adversario [3, 11, 13]. Las defensas del segundo tipo, y la que se estudia en este artículo, 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 es compresión JPEG [2].

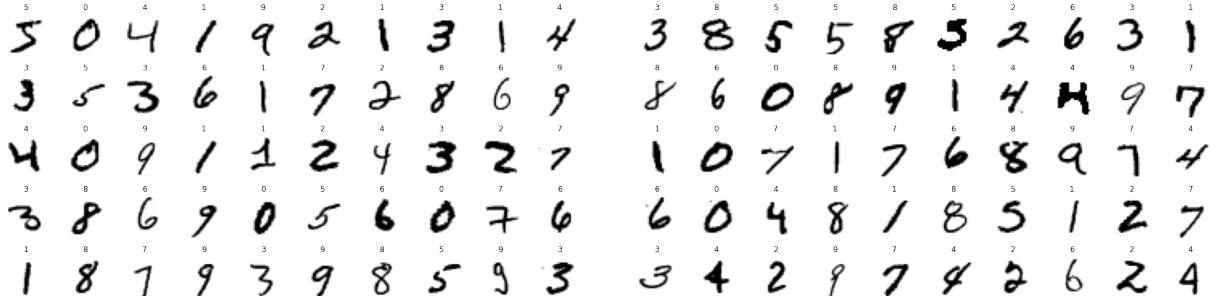
2. Método

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

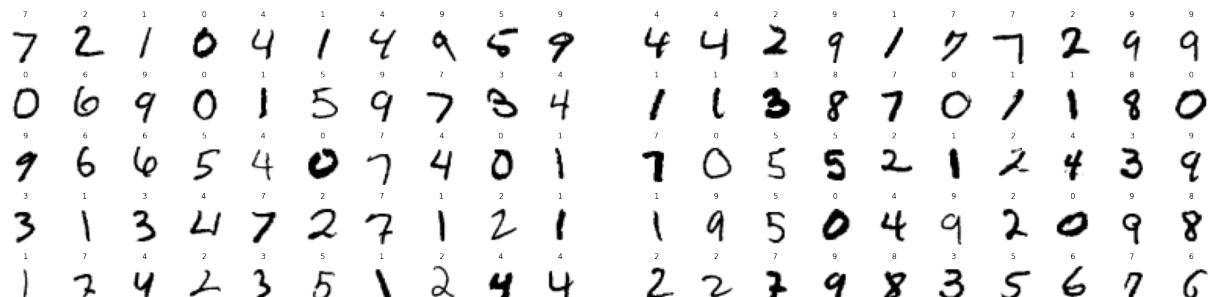
La base de datos MNIST es un compendio de los dígitos del 0 al 9 en letra manuscrita con 60,000 imágenes para entrenar a distintos sistemas de procesamiento de imágenes y 10,000 imágenes para evaluarlos. Se trata de un subconjunto de imágenes de un conjunto más grande que compiló el National Institute 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.

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



(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.

Figura 2: Imágenes del conjunto de evaluación (test) 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 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.

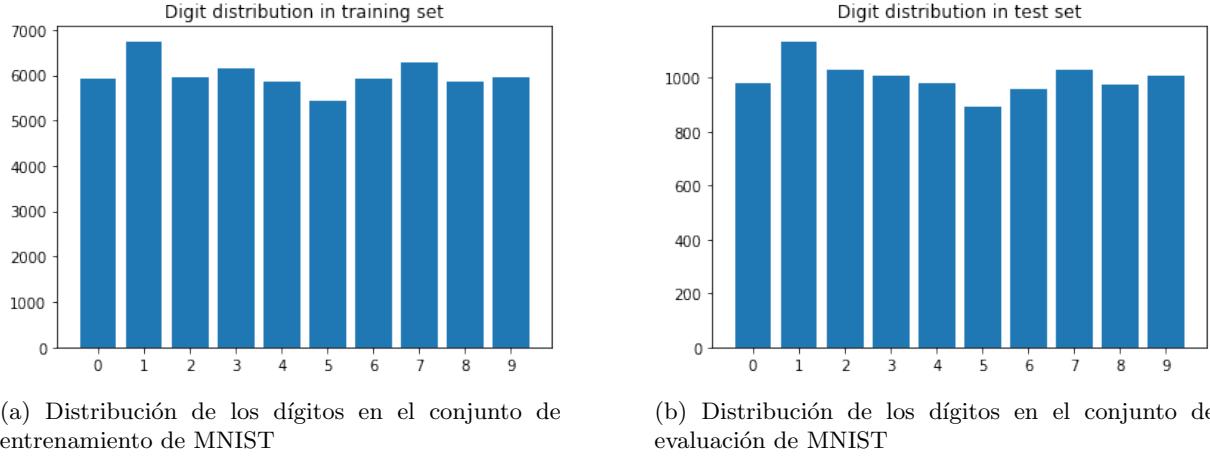


Figura 3: Distribución de los dígitos en MNIST

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.



Figura 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
2. automobile
3. bird
4. cat
5. deer
6. dog
7. frog
8. horse
9. ship
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

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 el ataque es totalmente imperceptible para nosotros. Aunque este ataque puede ser dirigido, se usa de la forma no dirigida en este reporte: 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, y con esos tres ataques se logra que la red clasifique erróneamente un 8 por un 3.

		Target Classification (L_2)									
		0	1	2	3	4	5	6	7	8	9
Source Classification	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3	3	3	3
	4	4	4	4	4	4	4	4	4	4	4
	5	5	5	5	5	5	5	5	5	5	5
	6	6	6	6	6	6	6	6	6	6	6
	7	7	7	7	7	7	7	7	7	7	7
	8	8	8	8	8	8	8	8	8	8	8
	9	9	9	9	9	9	9	9	9	9	9

Figura 5: Ataque de Carlini & Wagner con la norma L^2 sobre MNIST aplicando un ataque dirigido para cada par fuente/objetivo (source/target) [1].

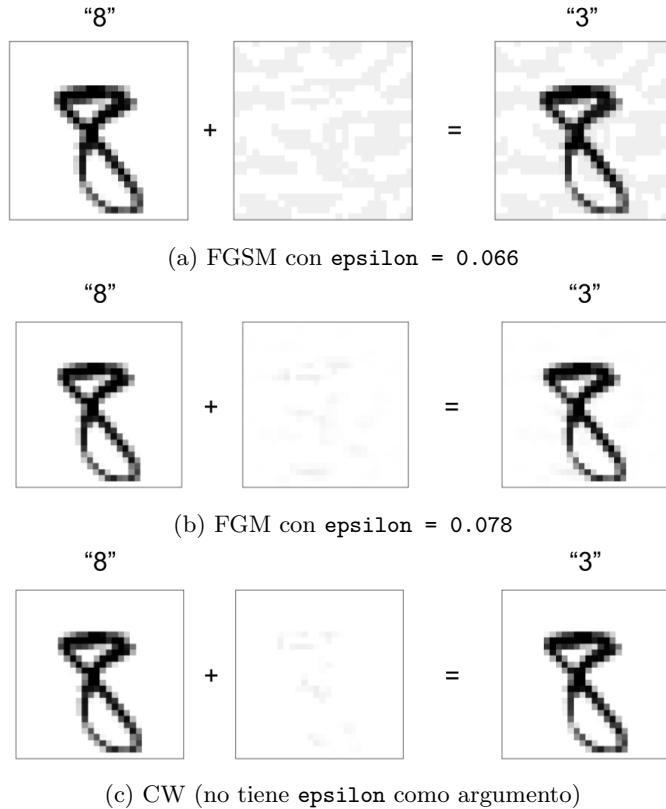


Figura 6: Ruido de los ataques FGSM, FGM y Carlini-Wagner sobre MNIST, con los cuales un 8 es clasificado erróneamente por LeNet como un 3.

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, 12]. 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.



Figura 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. De esta forma, construimos una función de transferencia calculando la respuesta de la red a ciertas frecuencias.

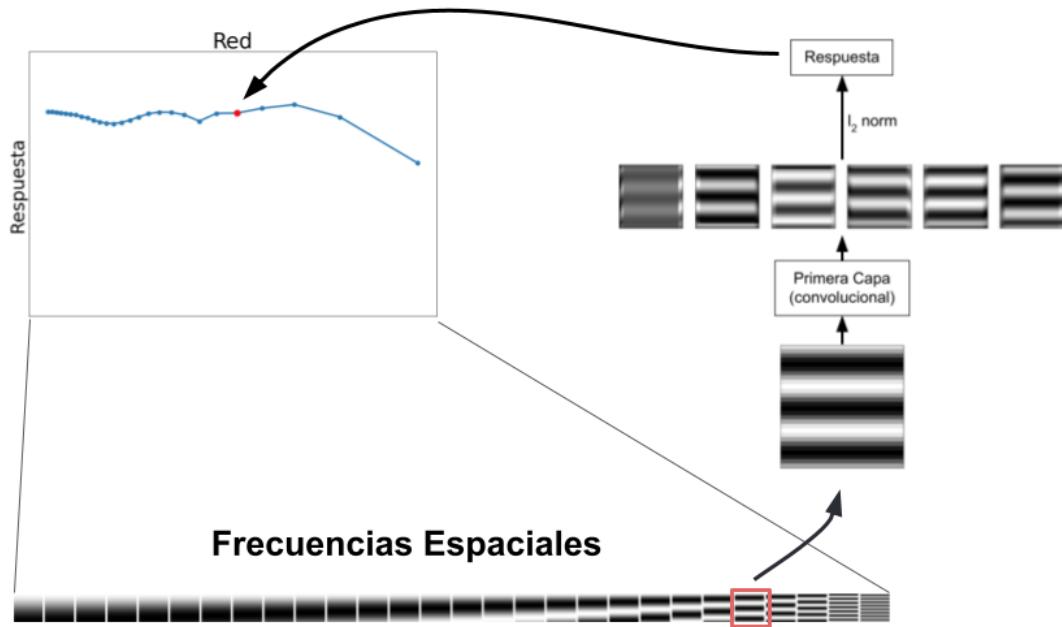


Figura 8: stuff

Así se puede ver las frecuencias a las que la red tiene mayor o menor sensibilidad.

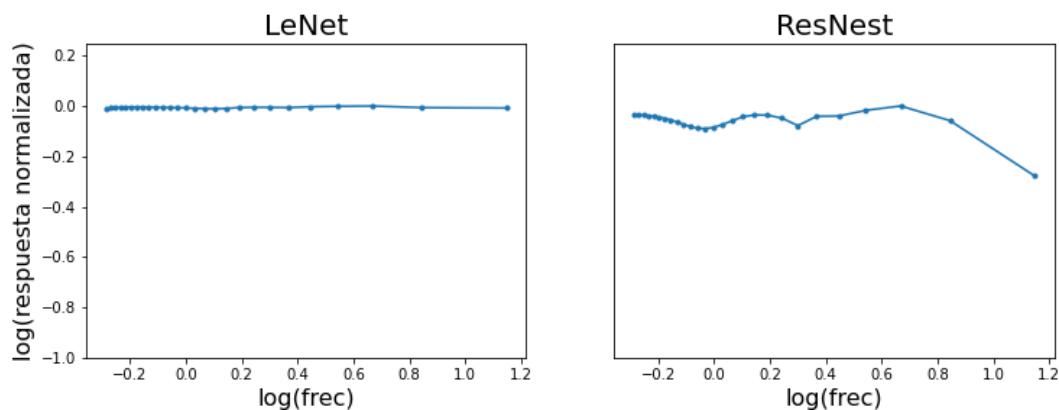


Figura 9

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 10.

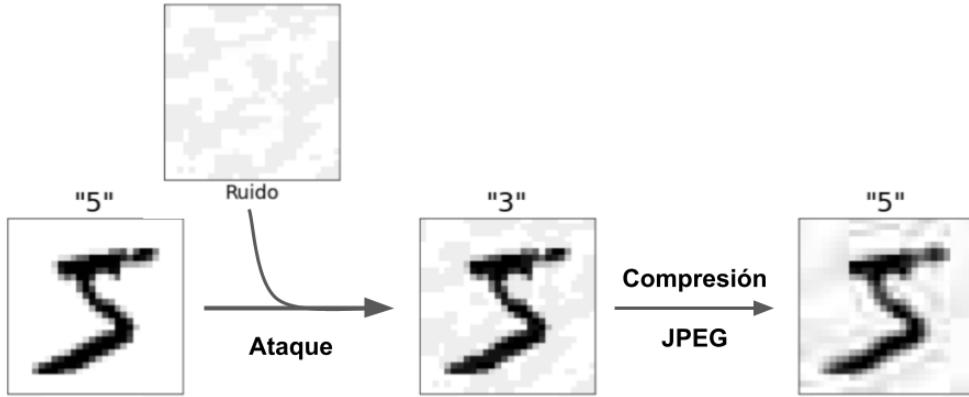


Figura 10: La defensa JPEG radica en contrarrestar el ruido agregado por el ataque adversario al remover los componentes de alta frecuencia, que son los causantes de la falla en la clasificación de la red. En este ejemplo, el dígito 5 es clasificado como un 3 con el ruido del ataque, pero al hacer una compresión JPEG, la red clasifica correctamente al dígito como un 5.

En la Figura 11 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 épsilon.

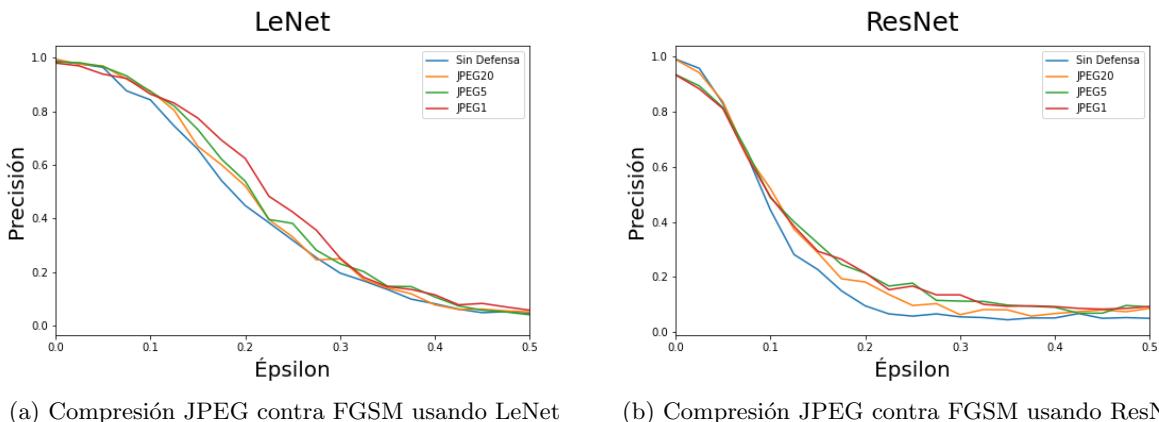


Figura 11: Resultado de la compresión JPEG como defensa ante el ataque FGSM sobre MNIST

to do:

- Carlini Wagner attack graphs

Vemos cómo cambian las funciones de transferencia de cada red cuando las imágenes pasan por compresión JPEG: frecuencia → compresión JPEG → 1^{er} capa (convolucional) → respuesta (Figura 12). Se nota que no hay mucho cambio en LeNet para ninguna calidad de compresión. Con respecto a ResNet, sí hay más modelación con unas calidades bajas (calidad = 7 – 16), pero no vimos 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.

Figura 12: 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 14), que tiene norma L_∞ , y FGM (Figura 15), 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 16a 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 16b).

En la Figura 17 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.

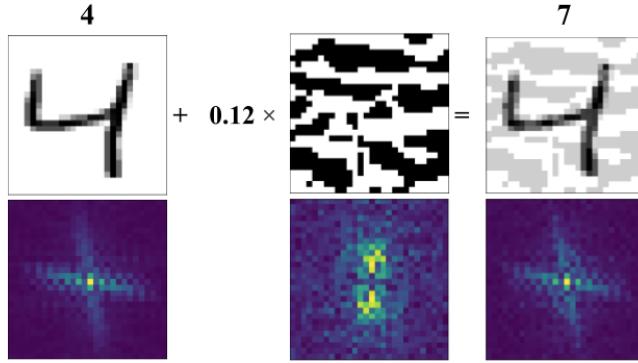


Figura 13: the noise can be seen in the fft of the adversarial image



Figura 14: FGSM sobre CIFAR-10

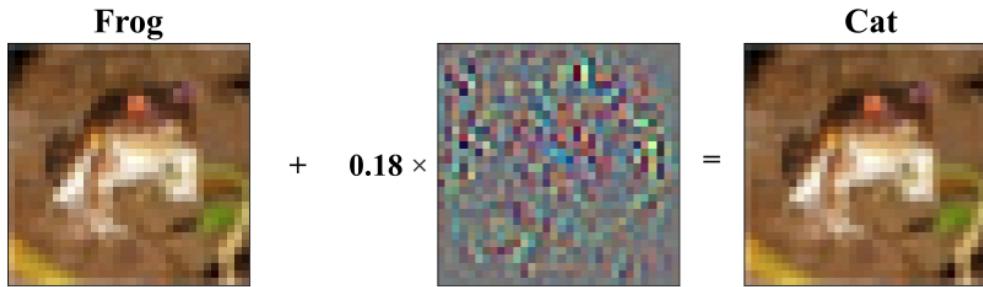
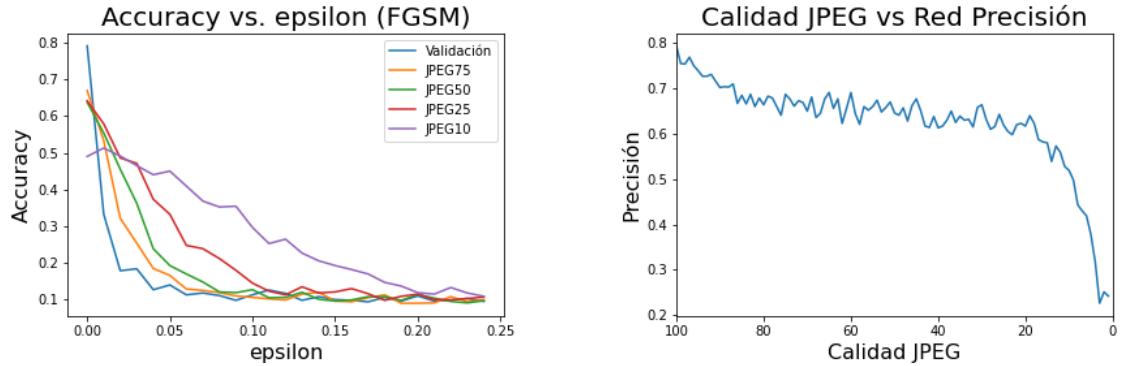


Figura 15: FGM sobre CIFAR-10

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



(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

Figura 16: Eficacia de JPEG

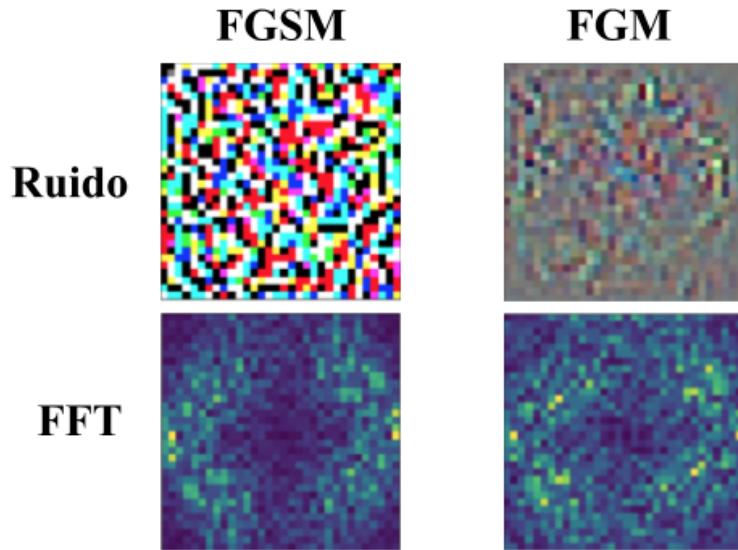


Figura 17: La FFT de los ruidos de FGSM y FGM

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 19).

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).

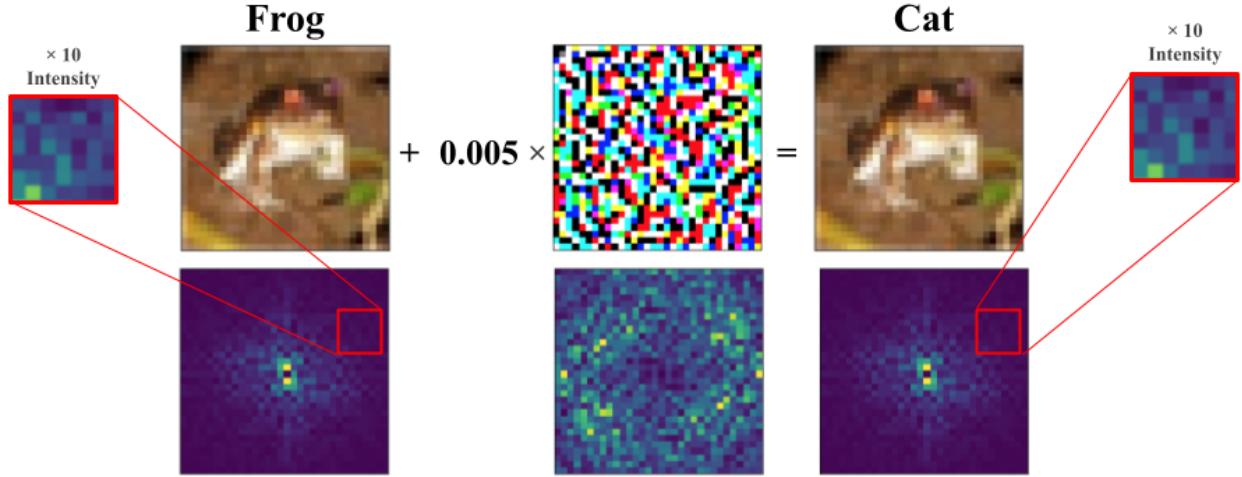


Figura 18: Comparación de la FFT de la imagen original y de la imagen con el ruido agregado por FGSM

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.

3.5. Saliency

The score of an image $S(I)$ is the output of the NN just before it gets converted to probabilities (this happens in softmax).

- Given an image I_0 and a class c , let $S_c(I_0)$ be the score of class c .
- We would like to rank the pixels of I_0 based on their influence on the score $S_c(I_0)$
- We can approximate $S_c(I)$ with a linear function in the neighbourhood of I_0 by computing the first-order Taylor expansion:

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

- where w is the derivative of S_c with respect to the image I at the point (image) I_0 .

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

- the magnitude of the derivative indicates which pixels need to be changed the least to affect the class score the most.

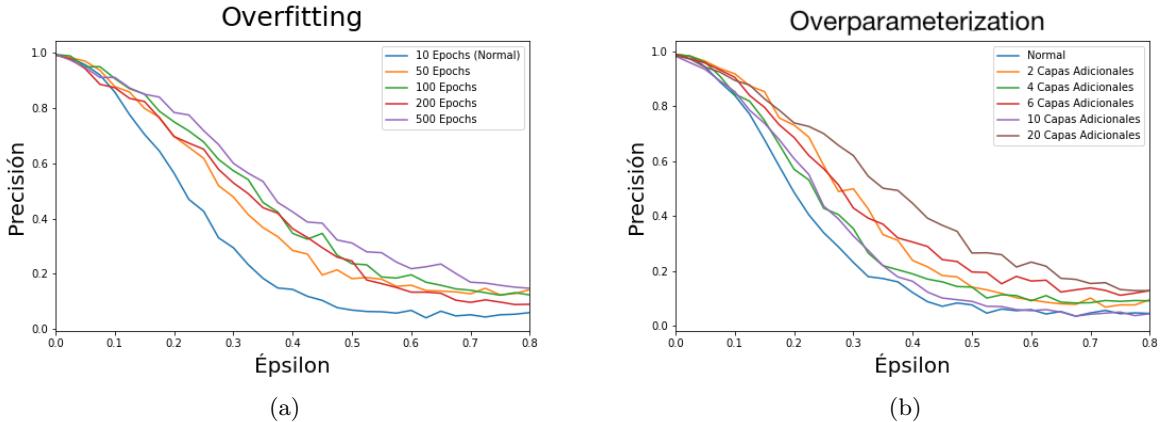


Figura 19: a) El overfitting consiste en tener exceso de épocas. Aquí se observa que conforme aumenta el número de épocas, la precisión aumenta con respecto a cada épsilon. b) Overparameterization consiste en exceso de capas. En general, aumentar el número de capas aumenta la precisión.

- Show figures from notebook of how adversarial noise attacks parts of the image that seem vulnerable (for example changing a 3→8)
- “first error” images for both cifar and mnist
- Show Gradient-based localization of both image sets and how they change after adding adversarial noise[10]
- See the saliency of an attacked image

4. Conclusión/Discusión

- Networks do seem to be sensitive to higher frequencies
- noise adds high frequency to image (but it's such low intensity...)
- JPEG defense has some effect with both data we tested
- JPEG defense doesn't really change the transfer function much
- The attacks are more effective with the color pictures, but JPEG defense is more effective as well
- seems like more pixels allow for smaller epsilons and less obvious attacks. (also I know this is mentioned in an article somewhere)

Referencias

- [1] Nicholas Carlini y David Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2017. arXiv: 1608.04644 [cs.CR].
- [2] Nilaksh Das y col. *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 y Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].
- [4] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Inf. téc. 2009.
- [5] Alex Krizhevsky, Vinod Nair y Geoffrey Hinton. *The CIFAR-10 website*. 2009. URL: %5Curl%7Bhttps://www.cs.toronto.edu/~kriz/cifar.html%7D.
- [6] Yann LeCun, Corinna Cortes y CJ Burges. «MNIST handwritten digit database». En: *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist> 2 (2010).
- [7] Xingjun Ma y col. *Understanding Adversarial Attacks on Deep Learning Based Medical Image Analysis Systems*. 2020. arXiv: 1907.10456 [cs.CV].
- [8] Aleksander Madry y col. *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] Ramprasaath R. Selvaraju y col. «Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization». En: *International Journal of Computer Vision* 128.2 (oct. de 2019), págs. 336-359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: <http://dx.doi.org/10.1007/s11263-019-01228-7>.
- [11] Uri Shaham, Yutaro Yamada y Sahand Negahban. «Understanding adversarial training: Increasing local stability of supervised models through robust optimization». En: *Neurocomputing* 307 (sep. de 2018), págs. 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>.
- [12] Uri Shaham y col. *Defending against Adversarial Images using Basis Functions Transformations*. 2018. arXiv: 1803.10840 [stat.ML].
- [13] Christian Szegedy y col. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

5. Apéndice A: Precisión y resumen de las redes para CIFAR-10 y MNIST

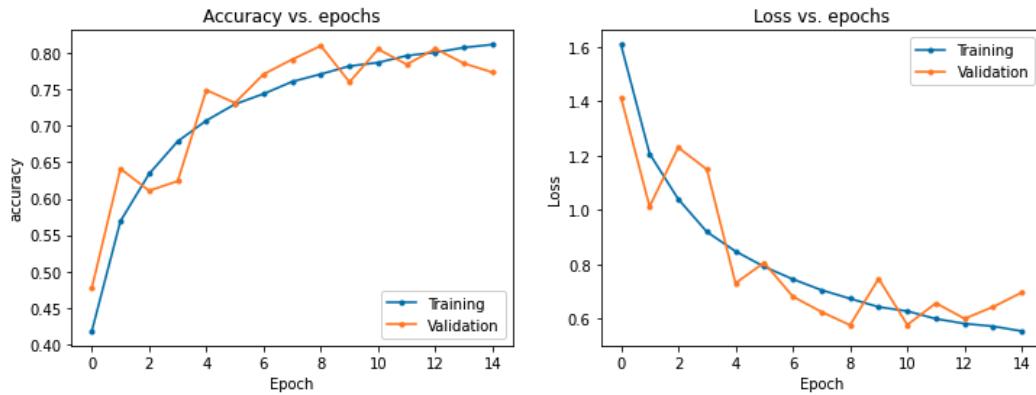


Figura 20: Precisión y pérdida de la red para CIFAR-10

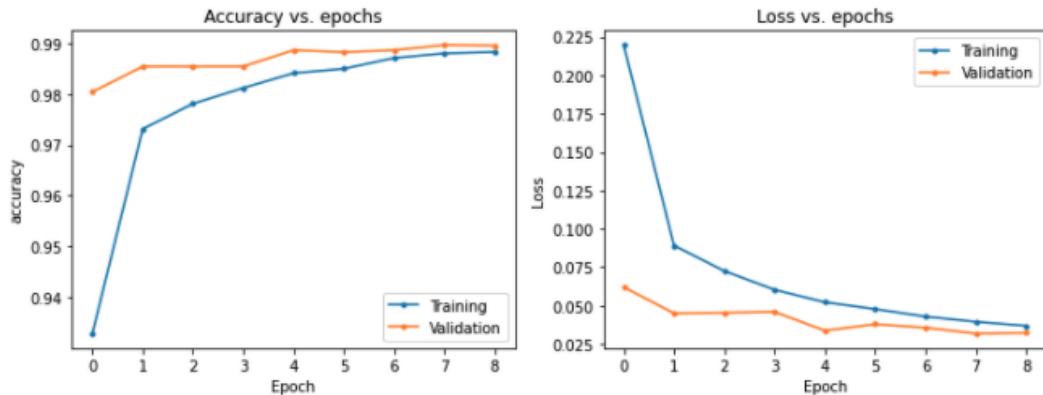


Figura 21: Precisión y pérdida de LeNet para MNIST

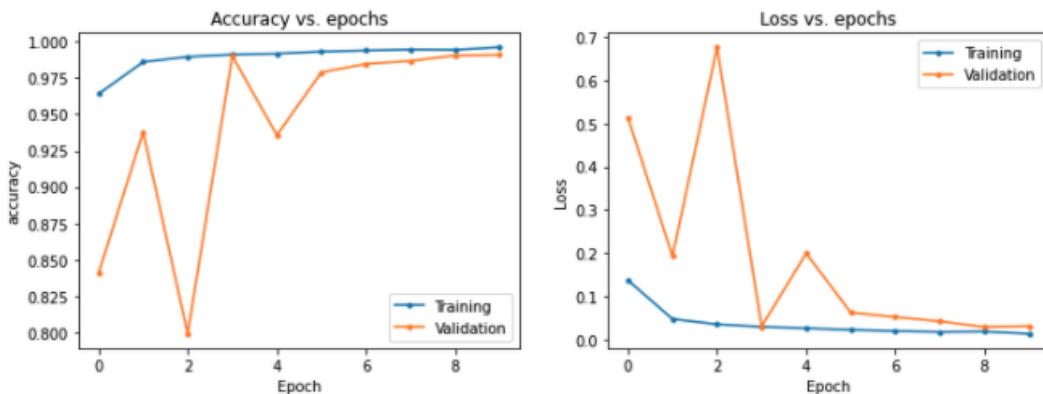


Figura 22: Precisión y pérdida de ResNet para MNIST

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	

Figura 23: Resumen de la red específica para CIFAR-10 [9]

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	

Figura 24: 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	

Figura 25: Resumen de ResNet para MNIST