ISSN 1870-4069

Explorando la robustez adversaria ante el ataque adversario PGD de los modelos AlexNet, VGG y ResNet

María Fernanda Castro-Sandoval[2], Ivan Reyes-Amezcua[1], Andres Mendez-Vazquez[1]

[1] Centro de Investigación y de Estudios Avanzados, Departamento de Computación, Mexico

[2] Universidad de Guadalajara, México

{ivan.reyes, andres.mende}@cinvestav.mx, maria.castro6643@alumnos.udg.mx

Resumen. A pesar de los últimos avances en el campo de Machine Learning y de la aparente precisión y calidad de los modelos actuales de visión por computadora, estos aún preservan una vulnerabilidad, y es ante los ataques adversarios. Estos consisten en ligeras modificaciones (a menudo imperceptibles para el ojo humano) en las imágenes de entrada, capaces de provocar que el modelo haga predicciones incorrectas. En el campo de la robustez adversaria se desarrollan métodos de defensa contra estos ataques, como el entrenamiento adversario y los métodos de generación de ataques como PGD. En este trabajo empleamos estos métodos para poner a prueba la robustez adversaria que pueden alcanzar algunos modelos de referencia como AlexNet, VGG y ResNet sobre dos datasets de referencia como CIFAR-10 y CIFAR-100. Nuestros resultados arrojan que aspectos como el algoritmo de optimización y su tasa de aprendizaje juegan un papel fundamental en el desempeño de los modelos y su robustez adquirida. Además, en nuestros experimentos, el modelo con la menor capacidad fue aquél que logró la mejor precisión tras el entrenamiento adversario, mientras que aquellos con la mayor capacidad presentaron un sobreajuste y porcentajes de precisión más bajos que el modelo de menor capacidad.

Palabras clave: Entrenamiento adversario, ataque adversario, robustez adversaria, PGD, transfer learning finetuning.

Exploring Adversarial Robustness Against PGD Adversarial Attack of AlexNet, VGG, and ResNet Models

Abstract. Despite the recent advances in the field of Machine Learning and the apparent accuracy and quality of current computer vision models, they still preserve a vulnerability, and it is against adversarial attacks. These consist of slight modifications (often imperceptible to the human eye) to input images,

capable of causing the model to make incorrect predictions. In the field of adversarial robustness, defense methods against these attacks are developed, such as adversarial training and attack generation methods like PGD. In this work, we employ these methods to test the adversarial robustness that some reference models like AlexNet, VGG, and ResNet can achieve on two reference datasets like CIFAR-10 and CIFAR-100. Our results show that aspects such as the optimization algorithm and its learning rate play a fundamental role in the performance of the models and their acquired robustness. Additionally, in our experiments, the model with the lowest capacity was the one that achieved the best accuracy after adversarial training, while those with the highest capacity exhibited overfitting and lower accuracy percentages than the model with the lowest capacity.

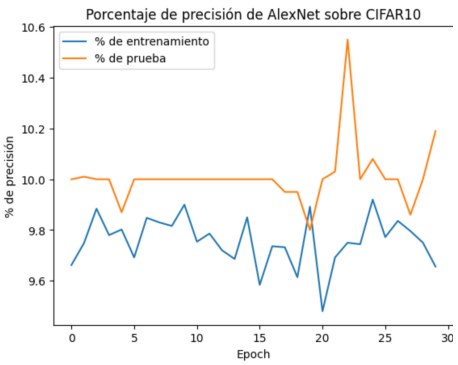
Keywords: Adversarial training, adversarial attack, adversarial robustness, PGD, transfer learning, finetuning.

1. Introducción

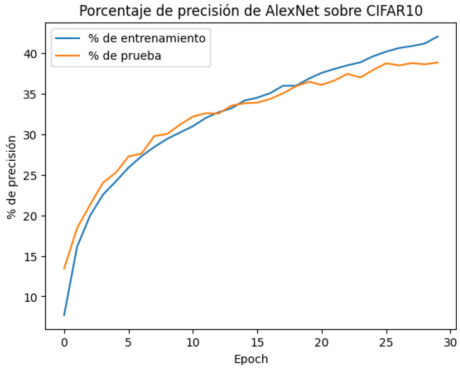
En los últimos años hemos presenciado el auge del campo del Machine Learning, al punto de posicionarse como una tecnología lider. Los modelos de redes neuronales profundas han demostrado ser bastante útiles y eficientes para una gran variedad de aplicaciones. Entre estos, los que resultan más atractivos y apasionantes son aquellos de visión por computadora, que interpretan el mundo visual a través de imágenes y videos para aprender a desempeñar diversas tareas: desde el reconocimiento y generación de imágenes hasta la detección de fraudes, diagnósticos médicos y la conducción de vehículos autónomos. Sin embargo, a pesar de los avances y de la aparente precisión y calidad de los modelos actuales de visión por computadora, existe una problemática importante: estos aún son vulnerables a los ataques adversarios.

Estos ataques consisten en modificaciones mínimas, a menudo imperceptibles para el ojo humano, en las imágenes que conforman el conjunto de datos de entrenamiento del modelo. Aunque pequeñas, estas modificaciones pueden ser capaces de causar que el modelo haga predicciones incorrectas. Dada la creciente aplicabilidad de estos modelos en nuestra vida cotidiana, esta vulnerabilidad no solo cuestiona el nivel de confianza que podemos depositar en estos modelos, sino que también puede dar lugar a problemas más comprometedores en situaciones reales. Por estas razones, el campo de la robustez adversaria, que estudia como mejorar la resistencia en la precisión de los modelos ante los ataques adversarios, se ha convertido en un área de investigación activa.

Entre las estrategias que esta ha desarrollado destaca el entrenamiento adversario, que en pocas palabras consiste en entrenar el modelo incluyendo intencionalmente ejemplos de ataques adversarios en el dataset de entrenamiento. A su vez, se han desarrollado diferentes algoritmos para generar estos ataques intencionados, como Fast Gradient Sign Method (FGSM) ó Basic Iterative Method (BIM). Pero uno de los más importantes y utilizados consiste en la aplicación del método de optimización llamado Projected Gradient Descent(PGD), que se traduce como “Descenso de Gradiente Proyectado”: un algoritmo muy útil para resolver problemas de optimización bajo restricciones.



Optimizador Adam con lr $= 0.001$:
% de entrenamiento $9.656\,\%$
% de prueba $10.19\,\%$



SGD con lr $= 0.001$ y momentum $= 0.9$:
% de entrenamiento $42.06\,\%$
% de prueba $38.85\,\%$

Fig. 1. Comparación en el aprendizaje de AlexNet al cambiar el optimizador Adam por SGD.

Así, las aportaciones de este trabajo consisten en una serie de experimentos de entrenamiento adversario con ataques generados por PGD a diferentes modelos de referencia en el área de visión por computadora: AlexNet, VGG, y ResNet. Asimismo, estos entrenamientos se llevarán a cabo sobre los datasets CIFAR-10 y CIFAR-100, también ampliamente utilizados en este campo.

2. Panorama sobre entrenamiento adversario con PGD

A pesar de que han surgido diferentes algoritmos de primer orden para generar ataques adversarios (i.e. métodos que utilizan información de primer orden sobre el modelo), en [4] se presentó evidencia de que PGD genera ataques "más fuertes" que otros algoritmos de este tipo. Por ejemplo, sus resultados arrojaron que aquellos modelos entrenados con ataques generados por FGSM presentaban un sobreajuste (overfitting) y continuaban siendo vulnerables ante ataques de PGD.

Por el contrario, si se entrena un modelo contra ataques de PGD, este también resistirá ataques generados por muchos otros métodos. Además, estos autores mostraron evidencia empírica sobre la convergencia de PGD hacia el máximo error del modelo para un dato de entrada y la similitud entre los resultados obtenidos al seleccionar distintos puntos iniciales alrededor de este punto de entrada. Finalmente, sus principales experimentos consistieron en el entrenamiento adversario de algunos modelos sobre los datasets MNIST y CIFAR-10.

En particular, sobre CIFAR-10 se entrenó un modelo ResNet y una versión amplificada de este, a la cual se le agregaron capas más amplias por un factor de 10, resultando en un modelo de 5 unidades residuales con (16, 160, 320, 640) filtros cada una. Así, al realizar el entrenamiento adversario con 20 iteraciones de PGD, ℓ_∞ y $\varepsilon = 8$, lograron un porcentaje de precisión de 43.7 % para el modelo original y 45.8 % para su versión amplificada. Una de sus conclusiones consiste precisamente en que la arquitectura de los modelos influye en gran parte sobre la robustez que pueden adquirir,

ya que la frontera de decisión que separa a los datos perturbados puede ser mucho más compleja que aquella que separa a los datos originales; así, los modelos con mayor capacidad (i.e. aquellos que poseen un mayor número de parámetros) presentan un mejor desempeño tras el entrenamiento adversario. En este trabajo vamos a manejar el planteamiento del entrenamiento adversario con PGD que proponen estos autores. Este método de defensa ha sido ampliamente utilizado en la investigación, ya que es de los pocos que entrenan a los modelos para resistir ataques más fuertes. Asimismo, varios trabajos han contribuido a mitigar su mayor desventaja: su alto costo computacional, desarrollando alternativas que logran resultados similares a un costo mucho menor, como Free Adversarial Training (entrenamiento adversario libre) [5], Fast Adversarial Training (entrenamiento adversario rápido) [8], entre otros [7].

3. Metodología y materiales

3.1. Noción matemática de un ataque adversario

Sea $f_\theta : X \to Y$ un modelo de visión por computadora compuesto por redes neuronales profundas con parámetros θ para un conjunto de datos etiquetados $\{X,\ Y\}$ (con $x \in \text{Mat}_{\ell \times w \times 3}(\mathbb{R})$, $\forall x \in X$; $y \in \mathbb{Z}$, $\forall y \in Y$). En este contexto, un ataque adversario para cualquier imagen de entrada $x \in X$ puede expresarse matemáticamente como:

$$x' = x + \delta, \tag{1}$$

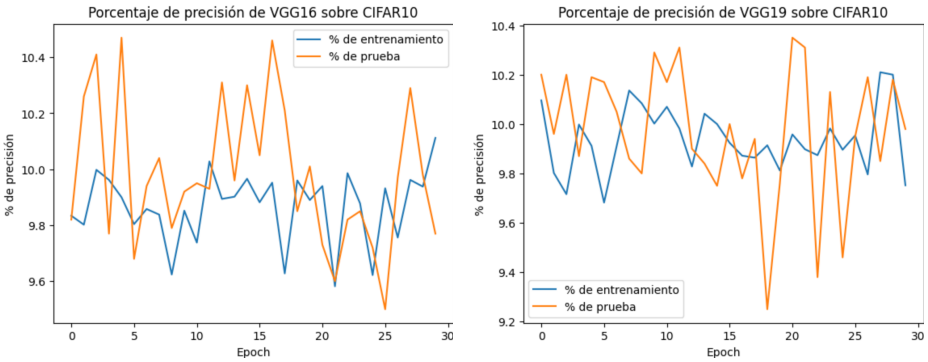
donde δ representa un tensor del mismo tamaño que x (es decir, $\delta \in \text{Mat}_{\ell \times w \times 3}(\mathbb{R})$) cuyos componentes contienen valores numéricos muy pequeños (a menudo denominados como ruido) que al sumarse a los valores de x provocan una ligera perturbación de este, misma que denotamos ahora como x'.

3.2. Entrenamiento adversario

Como se ha mencionado, el campo de investigación de la robustez adversaria estudia y desarrolla estrategias para lograr que los modelos como f_θ mantengan el mayor porcentaje posible de precisión al clasificar este tipo de ejemplos adversarios. Una de las técnicas más utilizadas para este propósito es el Entrenamiento adversario. Este tipo de entrenamiento utiliza Data Augmentation, una técnica que consiste en crear artificialmente datos nuevos a partir de los ya existentes, para aumentar la cantidad de muestras en el conjunto $\{X,\ Y\}$.

En este caso, las nuevas muestras generadas corresponden a un subconjunto de entradas del conjunto original (X) perturbadas mediante ataques adversarios como el de (1). Así, el entrenamiento adversario del modelo f_θ consiste en crear ejemplos adversarios de manera intencionada a partir de los datos de entrada X, para luego agregar estos nuevos ejemplos al mismo conjunto y agregar sus etiquetas (que corresponden a las de los mismos datos sin modificar) al conjunto Y, creando así un nuevo conjunto de datos aumentado al que denotaremos como $\{\bar{X},\ \bar{Y}\}$, sobre el cual finalmente se entrena el modelo.

Optimizador Adam con lr $= 0.001$:



% de entrenamiento 9.9 %
% de prueba 10.47 %

% de entrenamiento 9.958 %
% de prueba 10.35 %

Fig. 2. Resultados de entrenamiento adversario de VGG16 y VGG19 al emplear el optimizador Adam.

Esto se traduce en que, si se tiene la función $L(\theta; x, y)$ que calcula el error del modelo para una sola entrada $x \in X$, resultando en la función de error:

$$J(\theta; X, Y) := \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(x_i), y_i). \tag{2}$$

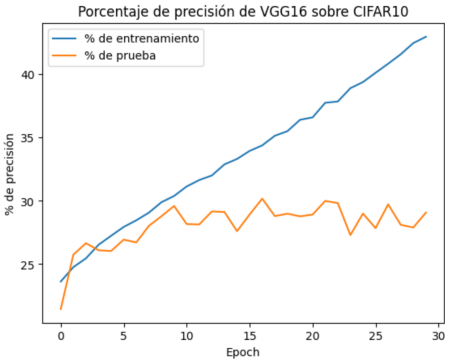
Para entrenar al modelo, entonces durante el entrenamiento adversario surge la nueva función de error:

$$\bar{J}(\theta; \bar{X}, \bar{Y}) := \frac{1}{n+m} \sum_{i=1}^{n} L(f_\theta(x_i), y_i) + \frac{1}{n+m} \sum_{i=1}^{m} L(f_\theta(x_i + \delta_i), y_i), \tag{3}$$

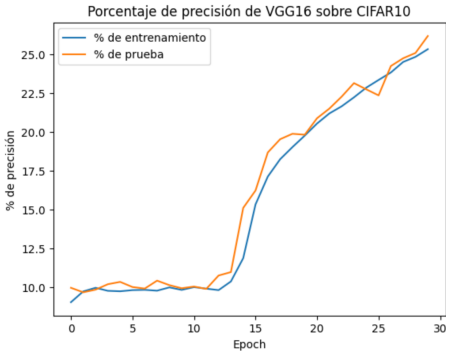
donde n denota el número de entradas en X, m la cantidad de entradas agregadas, $(x_i, y_i) \in X \times Y$ para $i = 1, 2, \ldots, n$, y cada $(x_i + \delta_i, y_i)$ $(i = 1, 2, ..., m)$ corresponde a la entrada (x_i, y_i) modificada.

3.3. Generación de ataques adversarios (PGD)

Existen diferentes técnicas para obtener los ataques adversarios de la forma (1) para aumentar el conjunto de datos $\{X,\ Y\}$. Una de las más importantes es el método PGD. Primero, se mencionó que los valores en el tensor δ son muy pequeños, pero para ser más específicos, se suele elegir a δ de manera que $||\delta||_p \leq \varepsilon$, donde $||\cdot||_p$ (a menudo también denotada por ℓ_p) representa a la norma p con $p \in [1, \infty]$ (así, cuando $p = 2$ corresponde a la norma euclidiana) y ε es un valor positivo suficientemente pequeño en el contexto del problema que se esté resolviendo. De hecho, se puede interpretar a ε como un valor numérico que mide la intensidad de los ataques adversarios, es decir, la diferencia entre una imagen modificada y aquella sin modificar (entre mayor sea ε, mayor será la diferencia).



Optimizador SGD con lr $= 0.1$:
% de entrenamiento 34.38 %
% de prueba 30.17 %



Optimizador SGD con lr $= 0.05$:
% de entrenamiento 25.312 %
% de prueba 26.16 %

Fig. 3. Diferencias en el aprendizaje de VGG16 al cambiar el parámetro lr.

Luego, el valor de ε se mantiene fijo durante todo el entrenamiento adversario, indicando la magnitud de los ataques de los que el modelo está aprendiendo a defenderse. Se pretende que el valor de ε sea lo suficientemente pequeño para que las alteraciones de las imágenes se mantengan imperceptibles para el ojo humano, pero al mismo tiempo logren confundir al modelo. Algunos valores de referencia para ε en la práctica son: $\varepsilon = 0.5$ para ℓ_2, y $\varepsilon = 8/255$, $\varepsilon = 4/255$ para ℓ_∞.

Por otra parte, es razonable pensar que si se lleva a cabo un entrenamiento adversario sobre aquellos ataques que logren la mayor disminución del rendimiento del modelo, la robustez adquirida por este será mayor. Por eso, para un $\varepsilon > 0$ y un par $(x, y) \in X \times Y$ dados, surge la idea de obtener el ataque para x de magnitud menor o igual a ε que provoque el mayor error en la predicción del modelo para esa entrada. Esto se plasma matemáticamente en el problema de optimización:

$$\delta^* = \arg\max_{\delta} L(\theta; f_\theta(x+\delta), y), \quad \text{sujeto a: } ||\delta||_p \leq \varepsilon. \tag{4}$$

Como vemos, este es un problema de optimización con una restricción, de la forma:

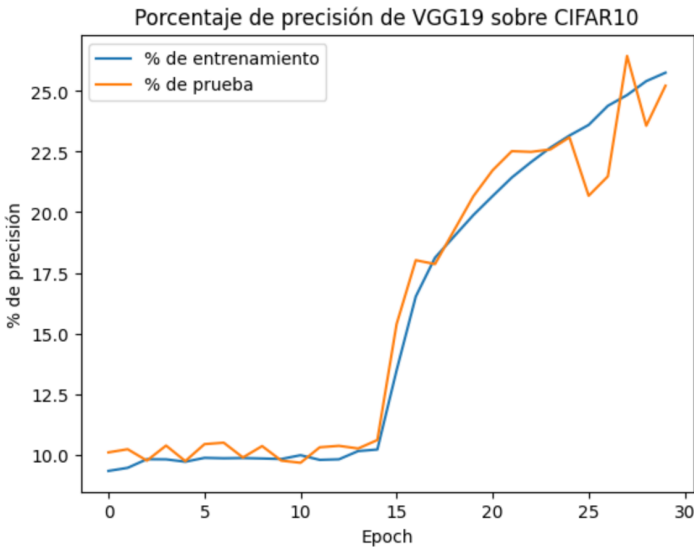
$$\arg\min_{t \in T} g(t), \tag{5}$$

donde $g = -L$, $t = \delta$ y $T = \{\delta : ||\delta||_p \leq \varepsilon\} = B_\varepsilon$, ($B_\varepsilon$ denota a la bola cerrada de radio ε centrada en cero, en $\mathbb{R}^{\ell \times w \times 3}$). Este es el tipo de problemas que resuelve el método del descenso de gradiente proyectado (PGD). Su mecanismo es muy similar al del método del descenso por gradiente, que calcula cada iteración $k+1$ de la forma:

$$t_{k+1} = t_k - \alpha_k \nabla g(t_k). \tag{6}$$

La diferencia consiste en que el método PGD proyecta el resultado de esta iteración hacia el conjunto de la restricción (T), mediante el operador de proyección:

$$\mathcal{P}_T(t_0) := \text{proj}_T(t_0) := \arg\min_{t \in T} ||t - t_0||_p. \tag{7}$$



Optimizador SGD con lr $= 0.05$:
% de entrenamiento 24.834 %
% de prueba 26.45 %

Fig. 4. Resultados del entrenamiento adversario para VGG19.

Además, PGD aplica la función:

$$\begin{aligned} \operatorname{sign}(\boldsymbol{v}) = \operatorname{sign}(v_1, v_2, \ldots, v_n) := (\operatorname{sign}(v_1), \operatorname{sign}(v_2), \ldots, \operatorname{sign}(v_n)), \\ \boldsymbol{v} \in \mathbb{R}^n,\ n \in \mathbb{N}. \end{aligned} \tag{8}$$

Al gradiente $\nabla g(t_k)$. De este modo, cada iteración $k+1$ del método del descenso de gradiente proyectado tiene la forma:

$$t_{k+1} = \mathcal{P}_T[t_k - \alpha_k \operatorname{sign}(\nabla g(t_k))]. \tag{9}$$

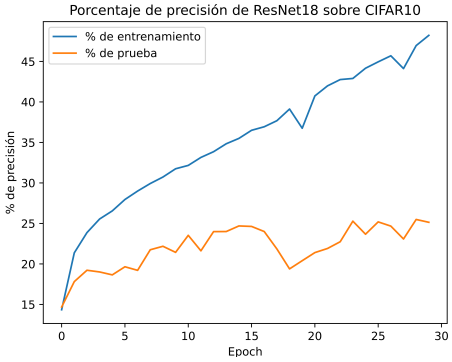
Así, para el caso que nos interesa (4), esta tiene la forma:

$$\delta_{k+1} = \mathcal{P}_{B_\varepsilon}[\delta_k + \alpha_k \operatorname{sign}(\nabla_\delta L(\theta; f_\theta(x+\delta), y))]. \tag{10}$$

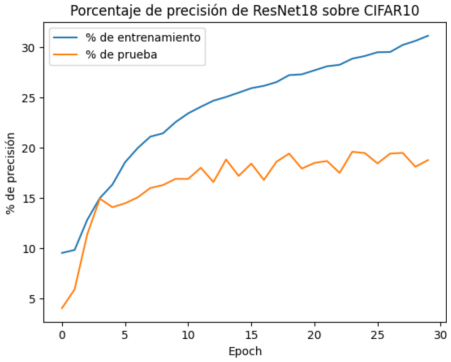
De esta manera se calcula cada iteración hasta que se cumpla algún criterio de paro, resultando en la δ^* que se buscaba en (4). En la práctica se suele elegir una cantidad fija de iteraciones, ya que de estas también puede depender la magnitud del ataque (entre más iteraciones, más cerca se puede llegar a aquel δ^* que maximiza el error del modelo). La elección del valor inicial δ_0 y cada valor α_k también varía según el problema. Una vez creados los ataques adversarios con este método, y agregados al dataset $\{X,\ Y\}$ formando uno nuevo: $\{\bar{X},\ \bar{Y}\}$, lo siguiente es entrenar al modelo f_θ sobre $\{\bar{X},\ \bar{Y}\}$, es decir, resolver el problema:

$$\theta^* := \arg\min_\theta \left(\frac{1}{n+m} \sum_{i=1}^{n} L(f_\theta(x_i), y_i) + \frac{1}{n+m} \sum_{i=1}^{m} L(f_\theta(x_i + \delta_i^*), y_i) \right). \tag{11}$$

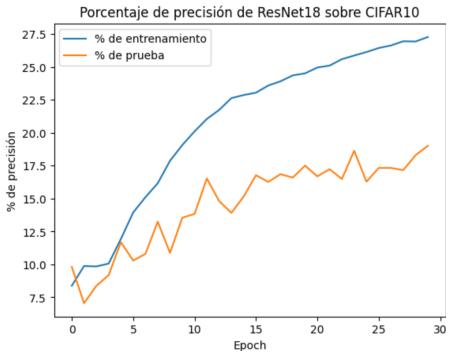
De este modo se lleva a cabo un entrenamiento adversario aplicando ataques de PGD.



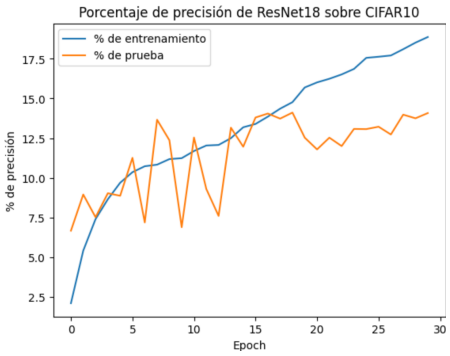
Optimizador Adam con lr $= 0.001$:
% de entrenamiento 46.956 %
% de prueba 25.49 %



Optimizador SGD con lr $= 0.1$:
% de entrenamiento 28.84 %
% de prueba 19.59 %



Optimizador SGD con lr $= 0.05$:
% de entrenamiento 27.264 %
% de prueba 19.0 %



Optimizador SGD con lr $= 0.001$:
% de entrenamiento 14.766 %
% de prueba 14.766 %

Fig. 5. Experimentos realizados para el modelo ResNet18.

3.4. Transfer learning

La cantidad de ejemplos incluidos en un dataset, así como su estructura y distribución, son aspectos fundamentales para una mayor calidad en el aprendizaje de un modelo. Sin embargo, en la práctica no siempre es sencillo reunir la suficiente cantidad de datos, ya que tanto su recolección como su almacenamiento pueden llegar a ser tareas muy costosas; sobretodo si los datos consisten en imágenes.

En el campo de la visión por computadora se han desarrollado varias bases de datos de código abierto suficientemente extensas y completas. Una de las más grandes y ampliamente utilizadas es conocida como ImageNet, aunque en realidad, la mayoría de las veces se refiere con este nombre a un "pequeño" subconjunto de ella (que aún es muy extenso) con más de 1.2 millones de imágenes clasificadas en 1,000 categorías correspondientes a distintos animales y objetos. Este subconjunto recibe el nombre de ImageNet Large Scale Visual Recognition Challenge (ILSVRC), que se traduce como "Desafío de reconocimiento visual a gran escala de ImageNet".

Su nombre se debe a que, entre 2010 y 2017, el proyecto ImageNet organizaba un concurso anual en el que se entrenaban modelos de clasificación de imágenes sobre este. Varios de los modelos ganadores de esta competencia —que entrenaremos en este trabajo– marcaron grandes pasos en la historia de este campo y se convirtieron en modelos de referencia para redes neuronales convolucionales. En la siguiente subsección se describirán más a detalle.

Además de contribuir a una buena precisión en los modelos entrenados, se sabe que el tamaño y contenido de ILSVRC es ideal para que estos mismos modelos muestren un buen desempeño sobre otros datasets con categorías similares. Sin embargo, tal cantidad de contenido también provoca que el entrenamiento sobre este dataset resulte cuando menos desafiante, además de demandar un equipo de hardware especializado.

No obstante, hoy en día se encuentran disponibles en diferentes bibliotecas de software (como Pytorch) o distintos foros y repositorios en línea los pesos que se han obtenido después de entrenar a estos modelos sobre ILSVRC. Así, es posible obtenerlos y emplearlos como pesos iniciales (en lugar de generar a estos últimos aleatoriamente) para entrenar los mismos modelos sobre nuestro dataset más pequeño pero de naturaleza similar. A esta práctica se le conoce como transfer learning (aprendizaje por transferencia), y más específicamente, corresponde al método de Finetuning. En este trabajo se aplicará esta estrategia para realizar un entrenamiento adversario de los modelos que se mencionan a continuación sobre datasets más pequeños, que a su vez se describen en la sección 3.6.

3.5. Modelos

En este caso se optó por experimentar con 3 tipos diferentes de modelos, todos ganadores del ILSVRC: en un principio AlexNet, luego los modelos VGG, y finalmente los modelos de redes neuronales residuales o ResNet. En seguida se profundiza en cada una de sus arquitecturas.

AlexNet. En 2012 se popularizaron las redes neuronales convolucionales gracias al surgimiento de AlexNet: una red de este tipo compuesta por cinco capas convolucionales y tres capas totalmente conectadas (fully connected) que ganó la competencia de ILSVRC en ese mismo año. Recibe su nombre gracias a su creador, Alex Krizhevsky [3].

VGG. Su nombre proviene de las siglas para Visual Geometry Group, que se traduce como "Grupo de Geometría Visual". Este modelo propuesto por K. Simonyan y A. Zisserman de la Universidad de Oxford ganó el segundo lugar en la competencia de ILSVRC en el año 2014 [6]. Existen dos versiones de este, conocidas como VGG16 Y VGG19. Mediante su creación se experimentó en el aspecto de la profundidad en modelos de redes neuronales convolucionales, ya que ambas versiones poseen arquitecturas más profundas que sus predecesoras, con 16 y 19 capas en total respectivamente.

ResNet. Es común que en la práctica se espere que aquellos modelos con mayor profundidad sean los que reflejen el mayor desempeño. Sin embargo, la práctica de apilar una gran cantidad de capas de manera "ingenua" en un modelo puede dar lugar a algunos inconvenientes, como el desvanecimiento del gradiente y el sobreajuste.

Tabla 1. Resultados del entrenamiento adversario de los 3 modelos ResNet sobre CIFAR-10, con el optimizador SGD y lr $= 0.05$.

	ResNet18	**ResNet50**	**ResNet101**
% de entrenamiento	27.264 %	36.658 %	35.862 %
% de prueba	19.0 %	16.16 %	25.67 %

Por esta razón, los creadores de ResNet (el equipo de Microsoft Research liderado por Kaiming He) desarrollaron una manera ingeniosa de conectar las capas de una red profunda evitando estos inconvenientes: mediante los llamados "bloques residuales". Un bloque residual consiste en un par de capas de pesos sinápticos con una función de activación ReLU, donde, además de las conexiones directas, existe una conexión atajo que traslada el valor de entrada x y lo suma al valor resultante de aplicar las capas anteriores ($\mathcal{F}(x)$). Esto favorece sobretodo a la propagación del gradiente. Hasta ahora se han desarrollado varias arquitecturas diferentes de este modelo. El ganador de la competencia ILSVRC en 2015 contiene en total 152 capas [1]. En este trabajo nos enfocaremos en las versiones: ResNet18, ResNet50 y ResNet101, que contienen 18, 50 y 101 capas de profundidad respectivamente.

3.6. Datasets

Los datasets sobre los que vamos a trabajar también son muy utilizados en el área de visión por computadora, aunque son mucho más pequeños que ImageNet. Se describen a continuación.

CIFAR-10: Su nombre corresponde a las siglas para Canadian Institute for Advanced Research, y el número 10 se debe a que posee 10 clases. En realidad es un subconjunto de otro dataset mucho más extenso, llamado 80 million tiny images (80 millones de imágenes diminutas) recolectado por Alex Krizhevsky junto con Vinod Nair y Geoffrey Hinton [2]. CIFAR-10 contiene 6,000 imágenes por cada clase, de las cuales 5,000 pertenecen al conjunto de entrenamiento y 1,000 al conjunto de prueba.

Así, en total posee 60,000 imágenes de 32×32 pixeles cada una, donde los conjuntos de entrenamiento y de prueba están formados por 50,000 y 10,000 imágenes respectivamente.

CIFAR-100: Este dataset proviene de la misma fuente que el anterior y posee el mismo tamaño; la diferencia consiste en que este contiene 100 categorías de clasificación. Así, sus 60,000 imágenes se reparten en 600 para cada clase, con 500 en el conjunto de entrenamiento y 100 en el conjunto de prueba. Sus 100 categorías se dividen en 20 grupos llamados "superclases", donde cada superclase contiene 5 clases. De este modo, a cada imagen del dataset se le asignan 2 etiquetas: una etiqueta "fina" que indica la clase a la que pertenece, y una etiqueta "gruesa" que corresponde a su superclase.

4. Planteamiento de los experimentos y resultados obtenidos

Todos los experimentos incluidos en este trabajo se han realizado con la biblioteca Pytorch y consisten en el entrenamiento adversario de versiones preentrenadas de los modelos (es decir, empleando el método de finetuning, al tomar como pesos iniciales

Tabla 2. Resultados del entrenamiento adversario de todos los modelos sobre ambos datasets.

	CIFAR-10		**CIFAR-100**	
	% de entrenamiento	% de prueba	% de entrenamiento	% de prueba
AlexNet	42.06 %	38.85 %	17.022 %	15.37 %
VGG16	25.312 %	26.16 %	14.224 %	12.23 %
VGG19	24.834 %	26.45 %	9.862 %	9.19 %
ResNet18	27.264 %	19.0 %	11.046 %	4.61 %
ResNet50	36.658 %	16.16 %	19.43 %	3.41 %
ResNet101	35.862 %	25.67 %	33.56 %	6.37 %

a aquellos obtenidos en el entrenamiento sobre ImageNet). Todos los entrenamientos constan de 30 épocas. En cada una de ellas, las imágenes del dataset se agrupan en mini-batches (en pequeños lotes) y cada uno de ellos recibe un ataque de PGD. Estos ataques se generaron mediante 10 iteraciones, utilizando un valor constante $\alpha_k = 0.01$ y partiendo de $\delta_0 = \vec{0}$. En cuanto a la magnitud del ataque, para todos los experimentos se consideró $\boldsymbol{\varepsilon = 0.5}$. Finalmente, en nuestros experimentos, los ataques adversarios tomaron el lugar de cada imagen en cada mini-batch, formando así un dataset constituido puramente por imágenes perturbadas, como se sugiere en [4].

En este nuevo dataset es donde calculamos la función de error y los pasos de optimización. Para entrenar el modelo AlexNet se tuvo que cambiar la dimensión de las imágenes, así como aplicarles cortes centrales. Además, se normalizaron los datos utilizando los valores [0.485, 0.456, 0.406] para la media y [0.229, 0.224, 0.225] para la desviación estándar. Mientras tanto, para el resto de los modelos únicamente se normalizaron los datos con valores de 0.5 para la media y la desviación estándar. En seguida se presentan varios resultados obtenidos en estos experimentos para cada dataset.

4.1. Resultados para CIFAR-10

Uno de los detalles más importantes que se notó en los experimentos fue que el aprendizaje de los modelos era muy susceptible a la arquitectura elegida para el entrenamiento. En la figura 1 se ilustra esta situación para el modelo Alexnet: en un inicio se llevó a cabo su entrenamiento adversario con el optimizador Adam, pero su aprendizaje se estancaba y sus porcentajes de precisión se mantenían al rededor del 10 %. Sin embargo, al cambiar el optimizador por Stochastic Gradient Descent (SGD) y agregar el parámetro momentum $= 0.9$, su aprendizaje se mantuvo mayormente ascendente para ambos conjuntos de datos, llegando a precisiones de alrededor del 40 %.

Lo mismo ocurrió al entrenar los modelos VGG16 y VGG19 con el optimizador Adam y el mismo valor para la tasa de aprendizaje o learning rate (lr), como se muestra en la Figura 2. Para VGG16, al igual que en el caso anterior, se optó por cambiar el optimizador por SGD. Sin embargo, aunque la curva de aprendizaje lucía mayormente creciente, los porcentajes de precisión solo llegaron a 9.724 % para el conjunto de entrenamiento y 10.11 % para el conjunto de prueba.

Por esta razón, se optó por elegir una tasa de aprendizaje de mayor magnitud, en este caso 0.1. Aunque los resultados mejoraron, ocurrió un sobreajuste del modelo. Así, en seguida se probó con una tasa de aprendizaje ligeramente más pequeña: 0.05. Sorprendentemente, este cambio eliminó el sobreajuste y mantuvo el aprendizaje mayormente ascendente, aunque bajando un poco los porcentajes de precisión, como se muestra en la Figura 3.

No obstante, consideramos satisfactorios a estos últimos resultados obtenidos para VGG16, por lo que se decidió emplear la misma arquitectura de entrenamiento para VGG19 (optimizador SGD con lr $= 0.05$). Como vemos en la Figura 4, para este último modelo se obtuvieron resultados bastante similares a los de VGG16. Luego, para el caso de ResNet18, se realizaron experimentos con las 4 arquitecturas que arrojaron resultados satisfactorios en los modelos anteriores (Figura 5). Sin embargo, como podemos observar, en cada uno de ellos ocurrió un sobreajuste del modelo.

El experimento donde el sobreajuste fue menor es aquél donde se empleó el optimizador SGD con lr $= 0.05$ (el experimento con lr $= 0.001$ fue descartado debido a que refleja un aprendizaje lento, en el cual, quizá con un mayor número de épocas, se llegue a un sobreajuste similar al del resto de los casos). Por esta razón, para los modelos ResNet50 y ResNet101 se llevó a cabo un solo experimento con SGD y lr $= 0.05$. En la tabla 1 se presenta una comparación entre los porcentajes obtenidos por los 3 modelos ResNet con esta arquitectura de entrenamiento. En ella podemos notar que el sobreajuste se presenta en los 3 modelos, siendo ResNet50 el más afectado por este fenómeno.

4.2. Resultados para CIFAR-100

Dada la similitud entre ambos datasets, para realizar los entrenamientos de los modelos sobre CIFAR-100 se optó por emplear las mismas arquitecturas de entrenamiento que resultaron adecuadas para el dataset anterior. Así, todos los modelos fueron optimizados por SGD, y los valores de lr se mantuvieron para cada modelo: AlexNet con 0.001 y momentum $= 0.9$; y el resto de los modelos con 0.05. En la tabla 2 se presenta en resumen todos los porcentajes de precisión obtenidos en los experimentos anterior mencionados sobre CIFAR-100, junto con todos aquellos obtenidos sobre CIFAR-10 a modo de comparación. Nótese que los modelos ResNet aún conservan el sobreajuste sobre CIFAR-100, además de que, en general, las precisiones de todos los modelos son más bajas para este último dataset.

5. Discusión

De todos nuestros resultados podemos concluir que el algoritmo de optimización empleado durante el entrenamiento adversario juega un papel fundamental en el desempeño de cada modelo. En particular, el algoritmo Adam resultó inadecuado para todos los casos, mientras que PGD demostró levantar las curvas de aprendizaje y una posible convergencia (coincidiendo con el método propuesto en [4]). Asimismo, el valor de la tasa de aprendizaje también resultó ser muy relevante para los resultados, siendo 0.05 el más adecuado para la gran mayoría de los experimentos.

El sobreajuste de los modelos ResNet puede deberse a la profundidad y complejidad de sus estructuras, en comparación con AlexNet: el modelo con menor cantidad de capas que los demás y al mismo tiempo aquél que arrojó los mejores resultados. Asimismo, resulta razonable que las precisiones obtenidas sobre el dataset CIFAR-100 sean significativamente menores, además de haber obtenido un sobreajuste mucho más pronunciado en los modelos ResNet. Esto se debe a que, al tener un dataset con la misma cantidad de imágenes, pero un número de clases 10 veces más grande, la tarea de clasificación se torna mucho más complicada de llevar a cabo. Una complementación a este trabajo podría consistir en la realización de los mismos experimentos pero con una mayor cantidad de épocas, con el objetivo de observar la convergencia del aprendizaje. Asimismo, surge el interés de estudiar la relación entre la capacidad de los modelos y su desempeño tras el entrenamiento adversario, dado que en este caso una red AlexNet logró obtener un mejor aprendizaje que un modelo ResNet.

Referencias

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
2. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
3. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM, vol. 60, no. 6, pp. 84–90 (2017) doi: 10.1145/3065386
4. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv, vol. 1050, no. 9 (2017)
5. Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., Goldstein, T.: Adversarial training for free! Advances in neural information processing systems, vol. 32 (2019)
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv, (2014)
7. Sriramanan, G., Addepalli, S., Baburaj, A.: Towards efficient and effective adversarial training. Advances in Neural Information Processing Systems, vol. 34, pp. 11821–11833 (2021)
8. Wong, E., Rice, L., Kolter, J. Z.: Fast is better than free: Revisiting adversarial training. arXiv, (2020) doi: 10.48550/arXiv.2001.03994