Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Doctorado en Ciencias en Ciencias de la Computación

Ataques adversarios a sistemas de visión

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Doctor en Ciencias

Presenta:

Angello Jahir Hoyos Ibarra

Ensenada, Baja California, México

2022

Tesis defendida por

Angello Jahir Hoyos Ibarra

y aprobada por el siguiente Comité

Dr. Edgar Leonel Chávez González
Codirector de tesis

Dr. Ubaldo Ruiz López
Codirector de tesis

Dr. Jesús Favela Vara

Dr. Pedro Gilberto López Mariscal

Dr. Mariano José Juan Rivera Meraz

Dr. Pedro Gilberto López Mariscal
Coordinador del Posgrado en Ciencias de la Computación

Dr. Pedro Negrete Regagnon
Director de Estudios de Posgrado

Resumen de la tesis que presenta Angello Jahir Hoyos Ibarra como requisito parcial para la obtención del grado de Doctor en Ciencias en Ciencias de la Computación.

Ataques adversarios a sistemas de visión

Resumen aprobado por:

Dr. Edgar Leonel Chávez González	Dr. Ubaldo Ruiz López
Codirector de tesis	Codirector de tesis

Las redes neuronales profundas (DNN) pueden aprender a clasificar, sintetizar y, en general, inferir correlaciones en extensas colecciones de datos. Se han convertido en soluciones de caja negra para innumerables tareas cotidianas. Sin embargo, las DDNs son susceptibles a *ataques adversarios*, que consisten en ejemplos que contienen modificaciones imperceptibles para un observador humano los cuales están diseñados para que las DNNs cometan errores al realizar una predicción. Se observó que todas las arquitecturas basadas en DNN utilizan una codificación one-hot después de una capa softmax. La distancia de Hamming entre pares de códigos one-hot es dos, independientemente del número de clases, y se deduce que esto es fundamental para un ataque exitoso. El atacante solo requiere generar una perturbación capaz de cambiar un par de bits de información para clasificar incorrectamente una imagen. Algo similar a la transmisión de información a través de un canal ruidoso. Con la observación anterior en mente, en esta tesis se postula que el uso de códigos de corrección de errores puede mejorar la resistencia de las DNN frente a ataques adversarios. Se utilizan los códigos de Hadamard para ese fin. Los códigos Hadamard de orden n, para las clases O(n), tienen una distancia de Hamming de n/2. Por lo tanto, un ataque adversario necesitaría cambiar una mayor cantidad de bits para clasificar incorrectamente una imagen. Se probó esta hipótesis contra seis tipos de ataques de caja blanca y uno de caja negra. Los modelos con códigos Hadamard presentaron mayor resistencia que superó por mucho a las alternativas en el estado del arte. En particular, superamos a los modelos destilados, considerados como la mejor defensa adversaria en la literatura relacionada.

Palabras clave: ataques adversarios, defensa adversaria, códigos de Hadamard, clasificación de imágenes

Abstract of the thesis presented by Angello Jahir Hoyos Ibarra as a partial requirement to obtain the Doctor of Science degree in Computer Science.

Adversarial attacks to vision system

Abstract approved by:

Dr. Edgar Leonel Chávez González
Thesis Co-Director

Dr. Ubaldo Ruiz López
Thesis Co-Director

Deep neural networks (DNNs) can learn to classify, synthesize, and generally infer correlations across extensive data collections. They have become black box solutions for countless everyday tasks. However, DDNs are susceptible to *adversarial attacks*, which consist of instances that contain modifications imperceptible to a human observer, which are designed so that DNNs make mistakes when making a prediction. It was noted that all DNN-based architectures use one-hot encoding after a softmax layer. The Hamming distance between pairs of one-hot codes is two, independent of the number of classes, and it follows that this is essential for a successful attack. The attacker only requires generating a disturbance capable of changing a couple of bits of information to classify an image incorrectly. Somewhat similar to the transmission of information through a noisy channel. With the previous observation in mind, in this thesis, it is postulated that error-correcting codes can improve the resistance of DNNs against adversary attacks. Hadamard codes are used for this purpose. Hadamard codes of order n, for classes O(n), have a Hamming distance of n/2. Therefore, an adversary attack would need to change a more significant number of bits to misclassify an image. This hypothesis was tested against six types of white-box attacks and one black-box attack. The models with Hadamard codes presented more resistance that far exceeded the alternatives in the SOTA. In particular, we outperform distilled models, considered the best adversarial defense in the related literature.

Keywords: adversarial examples, adversarial defense, Hadamard codes, image classification

Dedicatoria

A lo inesperado.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de doctorado al ser el becario No. 552672.

A mis padres, en especial a mi madre, por impulsarme a ser mejor cada día.

A mis directores de tesis, el Dr. Edgar Chávez y el Dr. Ubaldo Ruiz quienes me guiaron pacientemente a lo largo de este trabajo de investigación. Les agradezco sus enseñanzas, amistad y el haber despertado en mi el interés en el área.

A los miembros de mi comité de tesis, a quienes les agradezco su tiempo, observaciones y sugerencias para mejorar mi trabajo de investigación.

A los amigos y amigas, con los que compartí muchos momentos a lo largo de mi estancia en Ensenada.

Tabla de contenido

Página

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras

Figura Página

Lista de figuras

Figura Página

Lista de figuras

Figura Página

Lista de figuras

Capítulo 1. Introducción

En los últimos años, los avances en redes neuronales profundas (DNN), el desarrollo de hardware especializado (GPUs) y la disponibilidad de una gran cantidad de datos, han permitido dar solución a una variedad de problemas. Particularmente, el conjunto de imágenes Imagenet (Deng *et al.*, 2009), junto al modelo AlexNet (Krizhevsky *et al.*, 2012), popularizaron el estudio del problema de clasificación de imágenes utilizando DNN.

A partir del modelo de Krizhevsky *et al.*, múltiples propuestas basadas en DNN como VGG (Simonyan y Zisserman, 2014) y ResNet (He *et al.*, 2016) han sido desarrolladas para dar solución al problema de clasificación de imágenes (Cheng *et al.*, 2017). Estos mismos modelos, han permitido avances en un otros problemas de visión por computadora, tales como detección de objetos (Redmon y Farhadi, 2018), segmentación de imágenes (Zhang *et al.*, 2017), estimación de pose (Toshev y Szegedy, 2014), entre otros.

1.1. Antecedentes

Pese a los avances antes mencionados, Szegedy *et al.* reportaron una vulnerabilidad en las DNN llamados ejemplos adversarios. Estos ejemplos son imágenes modificadas de tal manera que un modelo clasifique erróneamente la imagen resultante, mientras que a simple vista, una persona no nota los cambios.

(a) (b) (c)

Figura 1. Ejemplo de una imagen (a) clasificada como zorro ártico, una perturbación (b) y un ejemplo adversario (c) clasificado como araña, generado al combinar (a) y (b).

Para la generación de ejemplos adversarios, se han utilizado dos enfoques principalmente. El primero de ellos se basa en transformaciones geométricas, como la rotación y la traslación (Engstrom *et al.*, 2019). Por otra parte, el enfoque más popular consiste en crear una perturbación a partir de ruido blanco, la cual se combina con una imagen de entrada para generar un ejemplo adversario (Szegedy *et al.*, 2013), como se muestra en la Figura 1.

Recientemente, estos enfoques han sido extendidos con éxito al mundo físico. En la Figura 3(a) se pueden observar unos lentes generados en (Sharif *et al.*, 2019) que permiten engañar a un sistema de reconocimiento facial, en la Figura 3(b) se muestran unos parches creados en (Evtimov *et al.*, 2017) los cuales, son capaces de cambiar la clasificación de una señal de alto y en la Figura 2(c) se presenta una pintura creada en (Thys *et al.*, 2019) que engaña un sistema detección al evitar que una persona sea detectada.

(a) (b) (c)

Figura 2. Ejemplos adversarios en el mundo real (a) lentes adversarios (b) parches adversarios (c) pintura adversaria que evita reconocer personas. Fuente de a: (Sharif et al., 2019), b: (Evtimov et al.,2017), c: (Thys et al., 2019).

1.2. Justificación

Los sistemas de visión por computadora han pasado de ser una idea futurista a convertirse en un campo de investigación en expansión. Hoy en día, las computadoras son capaces de superar a los humanos en algunas tareas de visión, como clasificar imágenes. La forma en que estos sistemas procesan la información difiere cada vez más de cómo lo hacen los humanos.

Particularmente, los modelos de DNN reciben una imagen como entrada, la procesan a través de una serie de pasos donde primero detectan píxeles, posteriormente

bordes y contornos, hasta llegar a objetos completos y por último, producen una predicción sobre lo que están observando.

Aunque los modelos de DNN son entrenados utilizando grandes cantidades de datos, su arquitectura es fija y no madura con el tiempo como lo hace un humano. Por ejemplo, cuando vemos la imagen de un gato, lo más probable es que podamos reconocer al animal en la imagen, a pesar de que sea pelirrojo o rayado, si la imagen es en blanco y negro, o si la misma está desgastada y descolorida. Probablemente, podamos detectar a esta mascota cuando aparece acurrucada detrás de una almohada o saltando sobre un mueble. Naturalmente, hemos aprendido a identificar a un gato en casi cualquier situación. Por otro lado, aunque las DNN pueden superar a los humanos en el reconocimiento de un gato en condiciones fijas, nuevas imágenes que son un poco novedosas o ruidosas, pueden ocasionar que estos modelos fallen por completo.

En (Geirhos *et al.*, 2018), argumentan que los humanos prestan mayor atención a las formas de los objetos representados, mientras que los modelos de DNN se enfocan a las texturas de los objetos. Este hallazgo destaca el marcado contraste entre como “piensan” los humanos y las máquinas e ilustra cuán engañosas pueden ser nuestras intuiciones sobre lo que hace que la inteligencia artificial funcione.

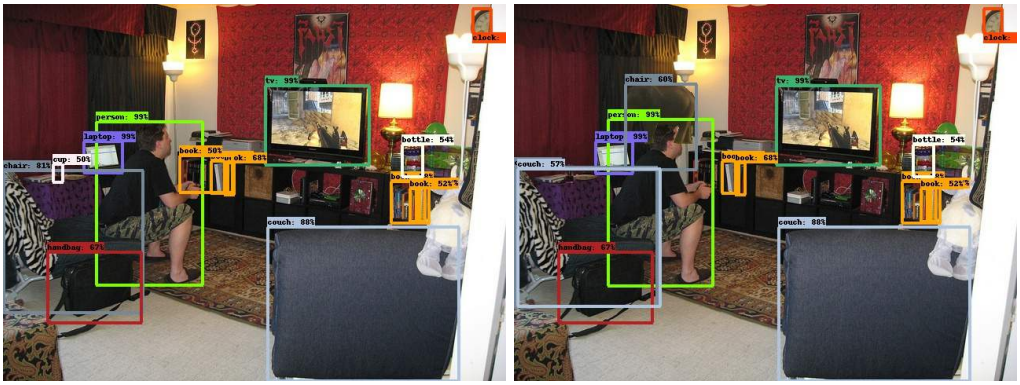
Por otro lado, este problema no se limita al procesamiento de imágenes, en (Boucher *et al.*, 2021) se demostró que pequeñas modificaciones en un texto, las cuales incluyen simplemente espacios en blanco, pueden causar estragos en la comprensión del texto por parte de un modelo de aprendizaje. Además, estas alteraciones también tienen consecuencias para los usuarios, en un ejemplo se muestra que el cambio de un solo carácter puede hacer que un usuario envíe dinero a una cuenta bancaria incorrecta.

Desde otra perspectiva, en (Reddy *et al.*, 2020) se estudia el procesamiento de imágenes cómo lo hace el ojo humano, argumentando que la diferencia entre la percepción visual en humanos y máquinas es evidente, la mayoría de los humanos procesan el mundo a través de sus ojos, mientras que las máquinas “ven” una imágen analizando una cuadrícula de números que representan el color de cada píxel de una imagen. Esto significa que tienen la misma agudeza visual en todo su campo de visión.

Dado lo anterior, se propuso entrenar un modelo con imágenes editadas para mostrar alta resolución en un solo lugar, imitando dónde podrían enfocarse el ojo humano y con una resolución decreciente que se expande hacia los límites de una imagen. Debido a que nuestros ojos se mueven para fijarse en múltiples partes de una imagen, también incluyen muchas versiones de la misma imagen con diferentes áreas de alta resolución. En sus resultados, sugirieron que los modelos entrenados con estas imágenes mejoran el rendimiento ante ejemplos adversarios sin perder precisión. Sin embargo, sus modelos no eran tan efectivos en comparación a la mejor solución no biológica, conocida como entrenamiento adversario (ver sección 2.2.2).

Pese a los problemas antes mencionados, cada vez es más frecuente el uso de modelos de DNN en aplicaciones cotidianas, por ejemplo, vehículos autónomos. El sistema de reconocimiento del vehículo recopila entradas de sensores, principalmente imágenes obtenidas por medio de cámaras, las cuales son procesadas por un modelo. Posteriormente, la salida se interpreta de acuerdo a una probabilidad de clasificación y se ejecuta la acción adecuada, como puede ser el detener un automóvil al detectar una señal de alto. No obstante, estos sistemas son vulnerables a diferentes amenazas, por ejemplo, utilizando ejemplos adversarios es posible manipular las imágenes de entrada, alterando la salida original y corromper el modelo, lo cual representa un gran peligro. Sobre todo, cuando generar y procesar un ejemplo adversario puede ser más rápido que la reacción de un humano.

Rosenfeld *et al.* presentaron las deficiencias de un algoritmo de detección de objetos Faster RCNN (Ren *et al.*, 2015) el cual clasifica objetos en una habitación, Figura 3 (a), identificando correctamente una silla, una persona, libros en un estante, entre otros objetos. Sin embargo, Rosenfeld *et al.* introdujeron un objeto anómalo a la escena al agregar la imagen de un elefante, Figura 3 (b) y la mera presencia del elefante hizo que el sistema cometiera errores al estimar a una silla como un sofá y al elefante como una silla, además que dejó de detectar otros objetos que había visto previamente.



(a) (b)

Figura 3. Se muestran las predicciones de dos imágenes donde en (a) es un escenario de una habitación normal, mientras que en (b) hay un elefante en dicha habitación, el cual es clasificado como una silla. Fuente: (Ren et al., 2015).

El caso anterior podría verse como una variante de los ejemplos adversarios, donde no se limitamos la diferencia entre la imagen original y la imagen modificada. Aún así los detectores pueden ser “ciegos” al objeto insertado, como se demuestra en la Figura 3 (b). Estos casos pueden ser inquietantes en aplicaciones como la conducción autónoma, ya que no es seguro conducir un automóvil si puede quedar ciego ante la presencia de un peatón sólo porque un animal pasó al costado de la carretera por un segundo.

A pesar de que las aplicaciones que perciben el mundo a través de cámaras y otros sensores como lo son los automóviles autónomos o sistemas de vigilancia, miden su eficiencia en función a los objetivos y capacidades alcanzadas, es posible que estas contemplen posibles amenazas. Lo anterior ha permitido que se desarrollen técnicas defensivas que frenan la efectividad de algunos ataques adversarios. A pesar del progreso reciente en el desarrollo de enfoques defensivos, aún queda un largo camino por recorrer antes de ser aceptados como soluciones infalibles. Hasta el momento, el problema de clasificar correctamente ejemplos adversarios no ha sido resuelto, las técnicas defensivas propuestas sólo proveen una protección parcial o impactan en el porcentaje de clasificación del modelo, como se reporta en Pang *et al.* (2017).

Motivado por los riesgos que representan los ejemplos adversarios y las deficiencias que presentan las estrategías defensivas actuales, en esta tesis se estudia una solución alternativa a este problema, con la cual se busca contribuir al desarrollo de aplicaciones basadas en DNN más seguras.

1.3. Hipótesis

Una convención utilizada durante el entrenamiento de una DNN, consiste en asociar cada imagen a un vector que tiene como tamaño el número total de clases del problema. En la posición correspondiente a su clase, el vector tiene como valor un uno y cero en el resto, esto es conocido como una codificación one-hot.

La codificación one-hot fue presentada en (Huffman, 1954) como una reducción de la memoria necesaria en la conmutación secuencial de circuitos, en el área de ingeniería eléctrica. Una ventaja de esta codificación es que puede binarizar las entradas categóricas (etiquetas) para que se consideren un vector en un espacio euclidiano, el cual es utilizado para calcular distancias o similitudes en muchos algoritmos de clasificación (Cassel y Lima, 2006). Debido a su simplicidad es la estrategia de codificación más popular en el área de aprendizaje de máquina (Rodríguez *et al.*, 2018).

Por su parte, un ataque adversario necesita cambiar un par de bits en la codificación one-hot para que un modelo se equivoque en la predicción de clasificación. Un ataque realiza con facilidad este cambio debido a que la distancia de Hamming entre un par de códigos one-hot es dos, independientemente del número total de clases.

Por su parte, en el problema de transmisión de información, los códigos de corrección de errores son utilizados para agregar redundancia a un mensaje por medio de una codificación. En algunos casos, esta codificación tiene una distancia de Hamming de $n/2$ para cualquier par de códigos, siendo n el tamaño del código.

A partir de las observaciones anteriores, en este trabajo se busca validar la siguiente hipótesis: Incrementar la distancia de Hamming en la codificación de clases utilizando técnicas de corrección de errores, aumentará la tolerancia de un modelo ante ataques adversarios.

1.4. Objetivos

1.4.1. Objetivo general

- Se propone utilizar códigos de corrección de errores para aumentar la distancia de Hamming entre los códigos de clasificación y aumentar la robustez de un modelo

ante ataques adversarios.

1.4.2. Objetivos específicos

- Sustituir la codificación one-hot tradicionalmente utilizada para entrenar DNN, por una codificación de Hadamard.
- Evaluar el desempeño de la codificación propuesta ante distintos ataques adversarios en comparación al mejor método defensivo mencionado en la literatura relacionada.
- Caracterizar las ventajas y desventajas de utilizar la codificación de Hadamard para el etiquetado de clases en el problema de clasificación de imágenes.

El resto de este trabajo de tesis se organiza de la siguiente manera. En el Capítulo 2, se describen los principales ataques adversarios utilizados para generar ejemplos adversarios, así como la descripción de las estrategias defensivas más efectivas de acuerdo a la literatura relacionada. En el Capítulo 3, se hace una breve descripción del problema de transmisión de información, como introducción a los códigos de corrección de errores lineales. En el Capítulo 4, se propone utilizar la codificación de Hadamard para el etiquetado de clases en el problema de clasificación de imágenes, para contrarrestar la efectividad de los ejemplos adversarios. En el Capítulo 5, se describen los experimentos realizados y se presentan los resultados obtenidos. Por último, en el Capítulo 6, se presentan las conclusiones de esta investigación y el trabajo futuro.

Capítulo 2. Trabajo relacionado

Debido a la gran cantidad de trabajos relacionados, en esta sección no se pretende realizar una revisión exhaustiva de todas las propuestas relacionadas a ataques y defensas adversarias. En cambio, se pretende dar una idea general de las principales técnicas que forman parte del estado del arte. La selección de estos trabajos se llevó acabo de acuerdo a su impacto en el área y múltiples menciones en los trabajos recopilatorios realizados por (Ozdag, 2018; Chakraborty *et al.*, 2018; Serban *et al.*, 2020).

2.1. Ataques Adversarios

De acuerdo a Szegedy *et al.*, un ejemplo de adversario se puede definir formalmente de la siguiente manera: Sea x una imagen, y la etiqueta asignada a x por un modelo de clasificación C, es decir, $y = C(x)$. Un ejemplo adversario es una imagen $x' = x + \eta$, donde η es una perturbación agregada a x, de modo que se asigna una etiqueta $\hat{y} = C(x')$ a x' donde $\hat{y} \neq y$. El objetivo principal de los ataques adversarios es aplicar una perturbación η de manera que las diferencias entre x y x' sean mínimas bajo alguna métrica de distancia y los cambios sean imperceptibles para el ojo humano.

Partiendo del modelo descrito anteriormente, un ataque adversario es capaz de generar un ejemplo adversario de diferentes maneras, según su objetivo y la cantidad de información conocida del clasificador.

En la literatura, es posible distinguir entre dos tipos de ataques adversarios, dirigidos y no dirigidos.

Los ataques dirigidos producen una clasificación incorrecta guiando la salida hacia una clase específica, por ejemplo, un ataque adversario engaña a un clasificador para predecir todos los ejemplos adversarios como una clase en particular maximizando su probabilidad.

Los ataques no dirigidos buscan asignar cualquier clase arbitraria a la salida del modelo, excepto la clase original. Estos ataques son más fáciles de implementar en comparación con los ataques dirigidos, ya que tiene más opciones para redirigir la

salida. Usualmente, los ejemplos de adversarios no dirigidos se construyen de dos formas: ejecutando varios ataques dirigidos ó eligiendo una perturbación que permita minimizar la probabilidad de clasificación en la clase original.

La cantidad de información disponible sobre el modelo clasificador divide los ataques adversarios en dos tipos. El primer escenario es conocido como caja blanca, en este el atacante tiene un conocimiento completo del modelo clasificador, incluido los datos de entrenamiento, la arquitectura del modelo, los hiperparámetros, el número de capas, las funciones de activación, los pesos del modelo, etc. Mientras que en el segundo, conocido como escenario de caja negra, el atacante solo tiene acceso a la salida del modelo.

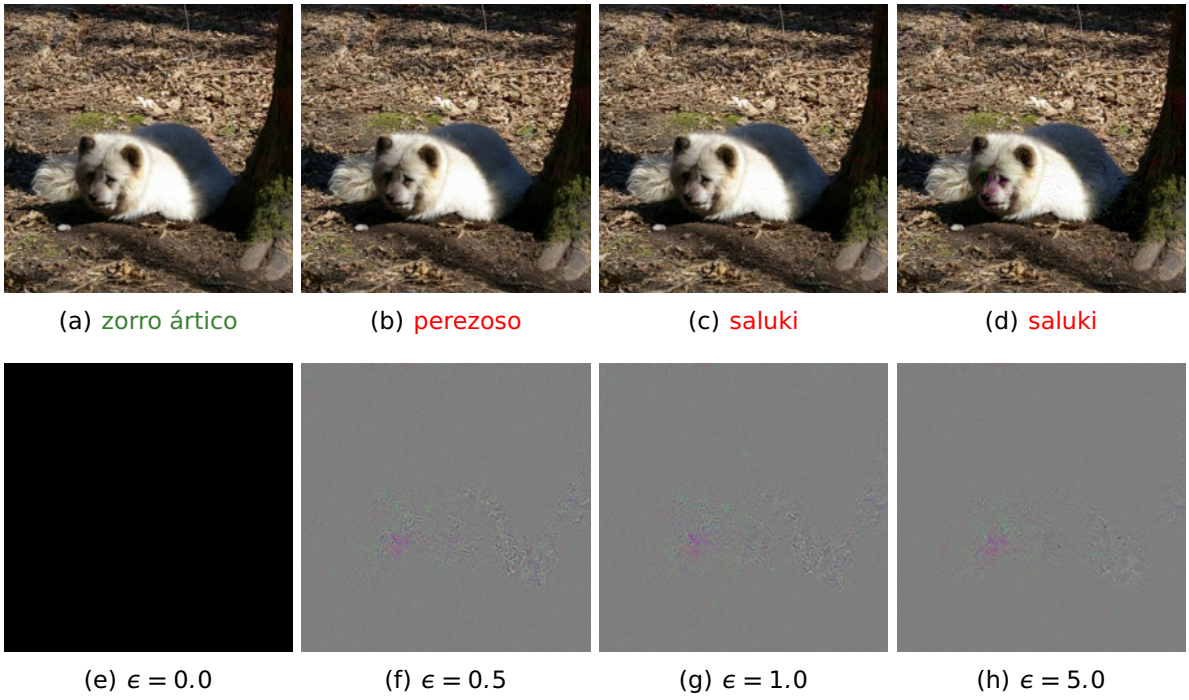
En el escenario de caja blanca, los ataques basados en ruido son los más comunes, se generan perturbaciones a partir del ruido blanco, como se muestra en la Figura 1. Estos ataques se basan en una estrategia de optimización que generalmente es precisa para encontrar perturbaciones mínimas o aproximaciones de estas. Algunos ejemplos de ataques de caja blanca no dirigidos son:

2.1.1. Fast Gradient Sign Method (FGSM)

El ataque más conocido en la literatura es el propuesto por Goodfellow *et al.*, este ataque se genera al sumar perturbaciones en la dirección del gradiente de pérdida para una clase de entrada; al moverse en la dirección del gradiente da como resultado maximizar la función de pérdida. Formalmente, los autores definen la perturbación η resultante de este ataque como:

$$\eta = \epsilon \, sign(\nabla_x \, J(\Theta, x, y)) \tag{1}$$

Donde θ son los parámetros del modelo, x es la entrada, y es la salida esperada del modelo y $J(\Theta, x, y)$ es la función de costo utilizada para entrenar una red neuronal. El valor de ϵ está relacionado con la sensibilidad de un observador humano para detectar el ataque. En la siguiente figura se pueden observar ejemplos adversarios generados con distintos valores de épsilon:



(a) zorro ártico (b) perezoso (c) saluki (d) saluki

(e) $\epsilon = 0.0$ (f) $\epsilon = 0.5$ (g) $\epsilon = 1.0$ (h) $\epsilon = 5.0$

Figura 4. En la fila superior se muestra una imagen (a) correctamente clasificada como zorro ártico, tres ejemplos adversarios generados con el ataque FGSM (b), (c) y (d). En la fila inferior se muestran las respectivas perturbaciones (e), (f), (g) y (h) de cada imagen superior.

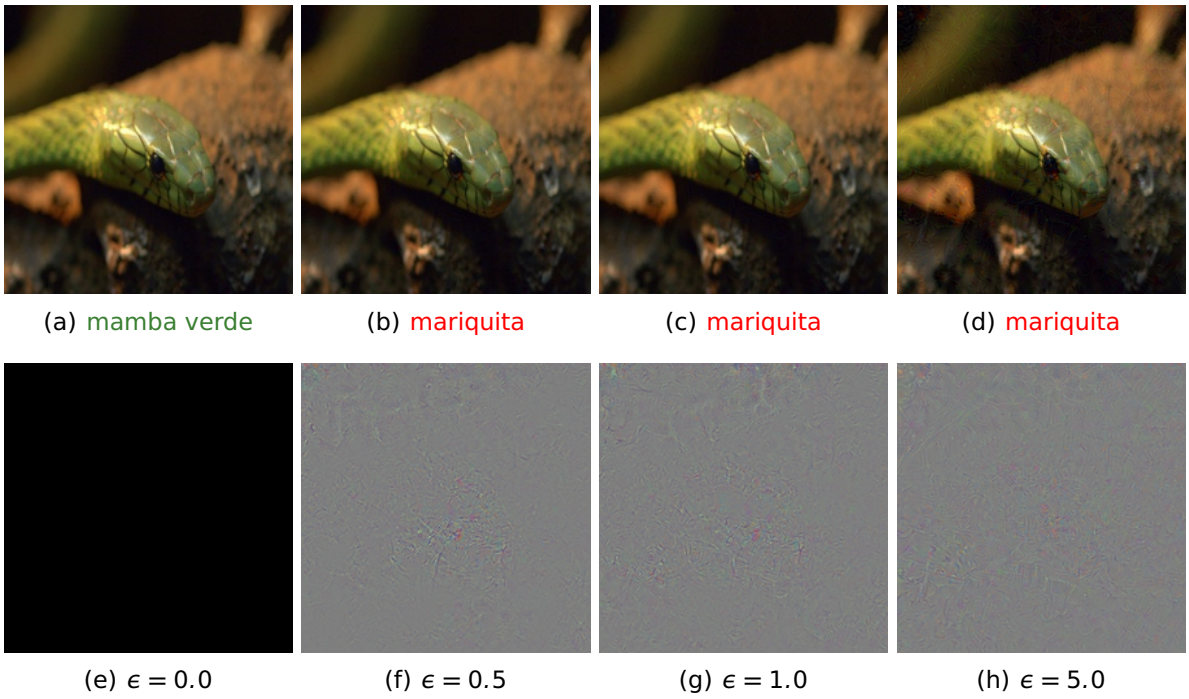
2.1.2. Basic Iterative Method (BIM)

Los métodos de un solo paso para generar ejemplos adversarios, como el ataque FGSM, se basan en encontrar la perturbación de una aproximación lineal de la función de costo. Kurakin *et al.* propusieron aplicar una versión iterativa de FGSM para aumentar la efectividad del ataque utilizando una función de recorte denominada $Clip_{X,\epsilon}(A)$ en donde los valores de los píxeles de A se acotan de tal manera que $A_{i,j}$ está recortado en un rango $[X_{i,j} - \epsilon, X_{i,j} + \epsilon]$, de esta manera, en cada iteración se ignoran los cambios grandes en los píxeles de la perturbación. Formalmente:

$$\begin{aligned} x'_0 &= x, \\ x'_{n+1} &= Clip_{X,\epsilon}(x'_n + \alpha\, sign(\nabla_x J(\Theta, x, y))). \end{aligned} \tag{2}$$

En la ecuación anterior, podemos observar que el ataque FGSM se encuentra dentro de la función $Clip_{X,\epsilon}$, con la diferencia del parámetro α, el cual se define como $\alpha = 1$ y se incrementa en uno en cada iteración, de acuerdo con los autores de este trabajo. En

la siguiente figura se pueden observar ejemplos adversarios generados con distintos valores de épsilon:



(a) mamba verde (b) mariquita (c) mariquita (d) mariquita

(e) $\epsilon = 0.0$ (f) $\epsilon = 0.5$ (g) $\epsilon = 1.0$ (h) $\epsilon = 5.0$

Figura 5. En la fila superior se muestra una imagen (a) correctamente clasificada como mamba verde, tres ejemplos adversarios generados con el ataque BIM (b), (c) y (d). En la fila inferior se muestran las respectivas perturbaciones (e), (f), (g) y (h) de cada imagen superior.

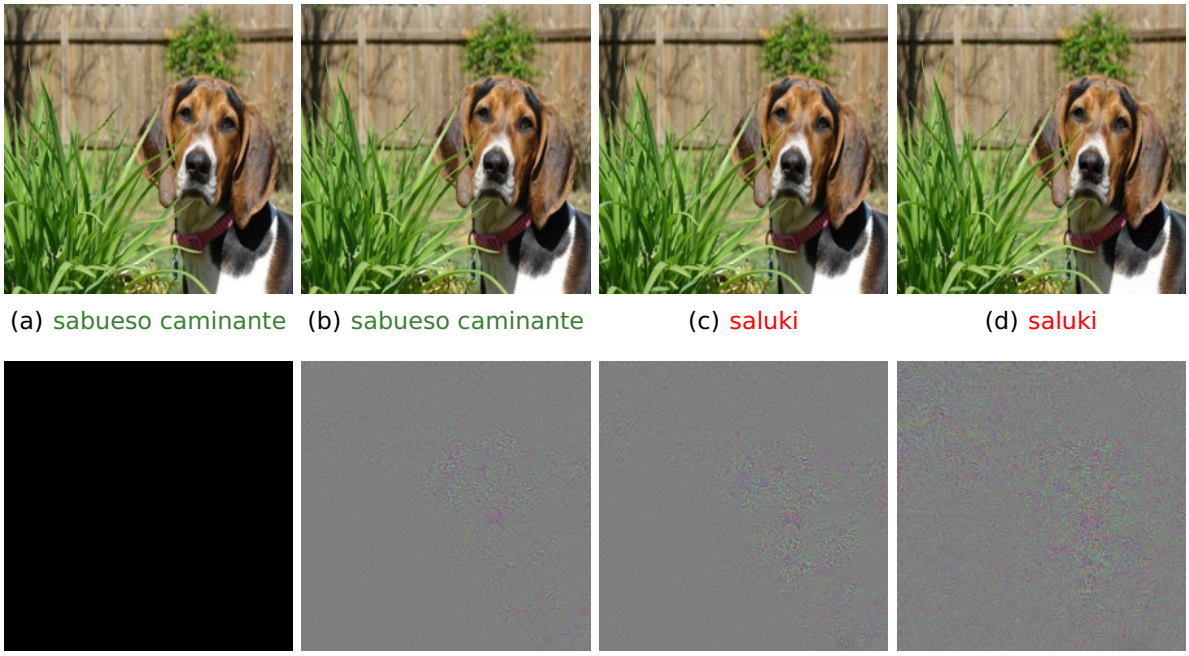
2.1.3. Projected Gradient Descent Attack (PGD)

Mądry *et al.* (2017) propusieron una variante del ataque FGSM en la que se aplica iterativamente una proyección de gradiente a partir de una serie de perturbaciones $S \subseteq \mathbb{R}^d$, las cuales se encuentran delimitadas por una norma de distancia para cada entrada x. En la clasificación de imágenes, se elige un conjunto S que capture la similitud perceptiva entre imágenes. Formalmente:

$$\begin{aligned} x'_0 &= x, \\ x'_{n+1} &= \prod_{x+S} (x'_n + \alpha \, sign(\nabla_x J(\Theta, x, y))). \end{aligned} \tag{3}$$

Es importante mencionar que el conjunto S de posibles perturbaciones, depende completamente del modelo clasificador con el que se esté trabajando, ya que este

cambia el límite de decisión del conjunto y ante un modelo de clasificación robusto, el límite de decisión del problema cambia a uno más complejo, por lo que en este caso, será necesario un mayor tiempo de cómputo. A continuación, se muestran algunos ejemplos adversarios generados con PGD:



(a) sabueso caminante (b) sabueso caminante (c) saluki (d) saluki

(e) $\epsilon = 0.0$ (f) $\epsilon = 0.5$ (g) $\epsilon = 1.0$ (h) $\epsilon = 5.0$

Figura 6. En la fila superior se muestra una imagen (a) correctamente clasificada como sabueso caminante, tres ejemplos adversarios generados con el ataque PGD (b), (c) y (d). En la fila inferior se muestran las respectivas perturbaciones (e), (f), (g) y (h) de cada imagen superior.

2.1.4. Virtual Adversarial Attack (VA)

En este ataque, Miyato *et al.* (2015) definen un término de regularización llamado suavidad distributiva local (LDS), como el negativo de la sensibilidad de la distribución del modelo $C(y|x, \theta)$ con respecto a una perturbación. Esta sensibilidad es medida de acuerdo a la divergencia Kullback–Leibler (KL), es decir, se recompensa la suavidad de la distribución del modelo con respecto a una entrada x.

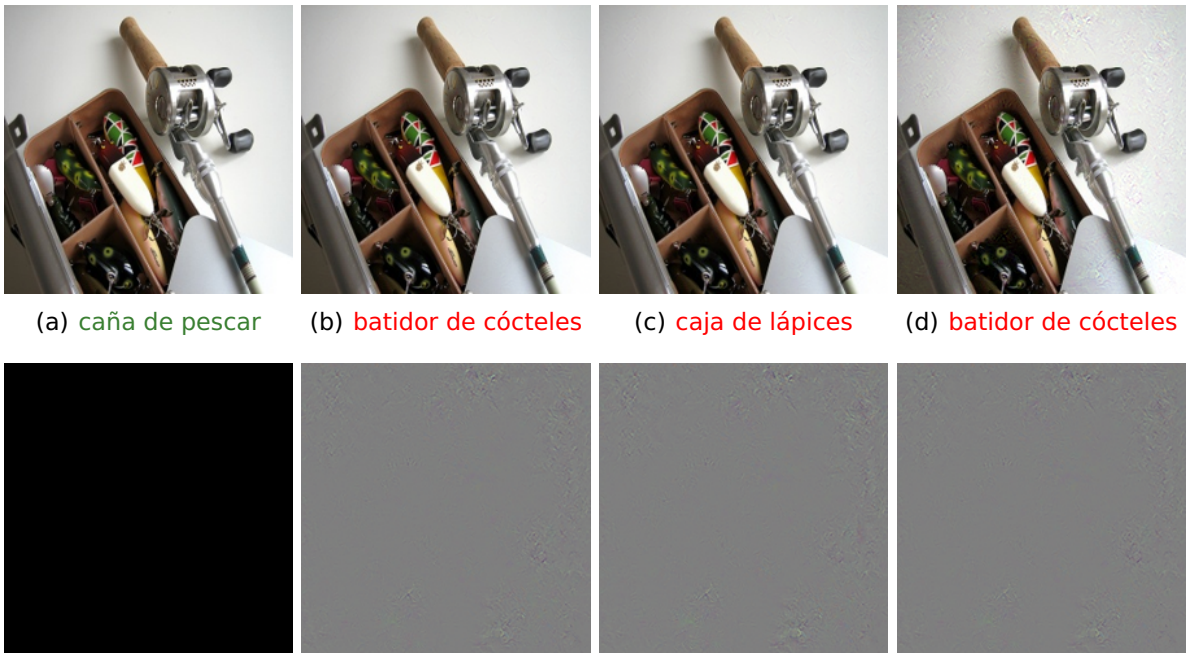
Por lo tanto, en cada iteración LDS está midiendo la robustez del modelo ante un perturbación en dirección local y "virtual". Por su parte, $KL[p||q]$ denota la divergencia KL entre las distribuciones de entrenamiento p y q. Además con el hiperparámetro $\epsilon > 0$ se define:

$$\triangle KL(r, x, \theta) \equiv KL[p(y \mid x, \theta) \,||\, q(y \mid x + r, \theta)] \tag{4}$$

$$r_{v-adv} \equiv \underset{r}{argmax}\{\triangle KL(r, x, \theta);\ ||r||_2 \leq \epsilon\} \tag{5}$$

Donde r_{v-adv} corresponde a una perturbación virtual. Por lo que LSD para una entrada x, se define como:

$$LDS(x, \theta) \equiv -\triangle KL(r_{v-adv}, x, \theta) \tag{6}$$



(a) caña de pescar (b) batidor de cócteles (c) caja de lápices (d) batidor de cócteles

(e) $\epsilon = 0.0$ (f) $\epsilon = 0.5$ (g) $\epsilon = 1.0$ (h) $\epsilon = 5.0$

Figura 7. En la fila superior se muestra una imagen (a) correctamente clasificada como caña de pescar, tres ejemplos adversarios generados con el ataque VA (b), (c) y (d). En la fila inferior se muestran las respectivas perturbaciones (e), (f), (g) y (h) de cada imagen superior.

2.1.5. Carlini and Wagner Attack (C&W)

El ataque C&W actualmente es considerado uno de los ataques más efectivos en la literatura relacionada. Los autores Carlini y Wagner proponen una alternativa del

modelo de amenaza descrito al principio de esta sección, en donde:

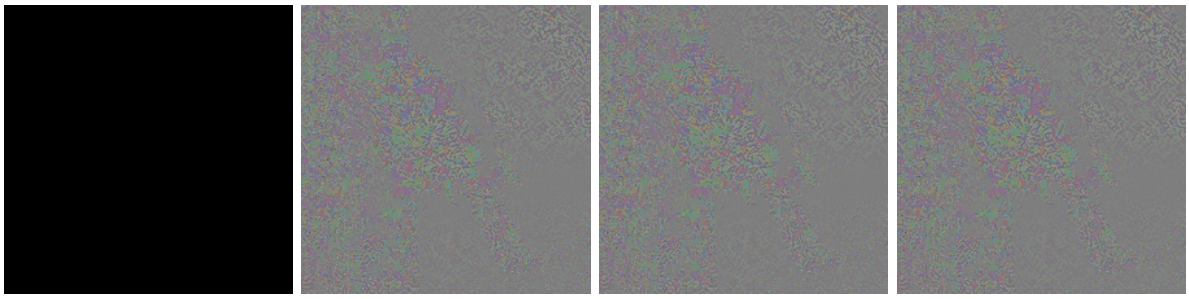
$$\begin{aligned} \min_{x'} & ||x' - x||_p \\ s.t.\ & C(x') = \hat{y} \\ & C(x) = y \\ & y \neq \hat{y} \end{aligned} \tag{7}$$

Y se define una nueva función objetivo g, tal que:

$$\min_{\eta} ||\eta||_p + c \cdot g(x + \eta) \tag{8}$$

donde c es una constante, $g(x') \geq 0$ si y sólo si $C(x') = \hat{y}$. Los autores definen siete funciones candidatas g, para optimizar la distancia entre la imagen de entrada x y el ejemplo adversario x', así como el valor de ϵ.

(a) pinzón de casa (b) campana (c) campana (d) campana



(e) $\epsilon = 0.0$ (f) $\epsilon = 0.5$ (g) $\epsilon = 1.0$ (h) $\epsilon = 5.0$

Figura 8. En la fila superior se muestra una imagen (a) correctamente clasificada como pinzón de casa, tres ejemplos adversarios generados con el ataque C&W (b), (c) y (d). En la fila inferior se muestran las respectivas perturbaciones (e), (f), (g) y (h) de cada imagen superior.

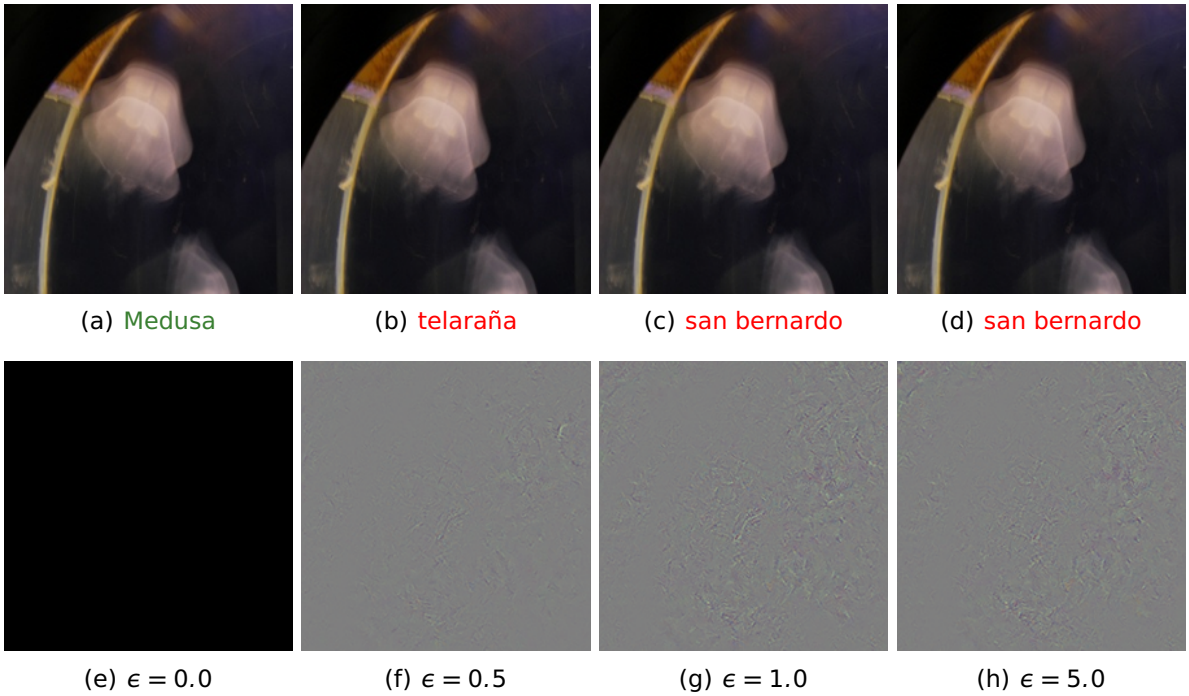
2.1.6. Decoupled Direction and Norm Attack (DDN)

Rony *et al.* (2019) propusieron un ataque donde los parámetros son ajustables en cada iteración en función de si la imagen perturbada es adversaria o no. Este ataque surgió como mejora del ataque C&W donde la principal dificultad es elegir el parámetro c en la Eq. (8), ya que si c es pequeño, el ejemplo no será adversario y si es grande, dará como resultado un ejemplo adversario muy ruidoso. Esto puede ser problemático cuando se genera un ataque con un número limitado de iteraciones.

Para evitar el caso anterior, el ataque DDN parte de una imagen x y refine iterativamente una perturbación η_k. En la iteración k, si el ejemplo actual $x'_k = x + \eta_k$ no es adversario, se considera una norma mayor $\epsilon_{k+1} = (1+\gamma)\epsilon_k$. De lo contrario, si el ejemplo es adversario, se considera $\epsilon_{k+1} = (1-\gamma)\epsilon_k$ más pequeño. Siendo γ un factor de cambio en cada iteración. Formalmente,

$$\eta_k = \eta_{k-1} + g \tag{9}$$

$$x'_k = x + \epsilon_k \frac{\eta_k}{|| \eta_k ||_2} \tag{10}$$



(a) Medusa (b) telaraña (c) san bernardo (d) san bernardo

(e) $\epsilon = 0.0$ (f) $\epsilon = 0.5$ (g) $\epsilon = 1.0$ (h) $\epsilon = 5.0$

Figura 9. En la fila superior se muestra una imagen (a) correctamente clasificada como medusa, tres ejemplos adversarios generados con el ataque DDN (b), (c) y (d). En la fila inferior se muestran las respectivas perturbaciones (e), (f), (g) y (h) de cada imagen superior.

2.1.7. Universal Adversarial Perturbations

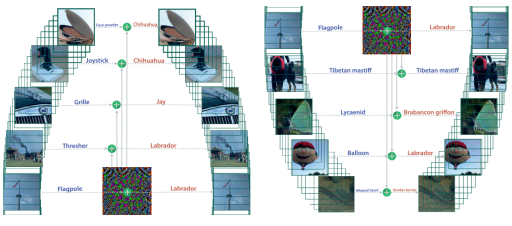
Los ataques descritos previamente sólo generan una perturbación para una imagen de entrada y esta misma rara vez puede ser utilizada en una imagen distinta. Sin embargo, en (Moosavi-Dezfooli *et al.*, 2017) se presenta un ataque capaz de generar una perturbación universal, es decir, una perturbación que se pueda aplicar a cualquier imagen para generar un ejemplo adversario.

Para generar una perturbación universal, como se muestra en la Figura 10, Moosavi-Dezfooli *et al.* proponen un algoritmo iterativo que construye gradualmente dicha perturbación sobre un conjunto de datos determinado. En cada iteración i, se busca una perturbación mínima μ_i, la cual envía a una imagen de entrada x_i al límite de decisión del clasificador. Posteriormente, se calcula y se agrega la instancia actual de la perturbación universal tomando en cuenta las siguientes restricciones:

$$||\mu||_p < \epsilon \tag{11}$$

$$\mathop{\mathbb{P}}_{x \sim X}(C(x+\mu) \neq C(x)) > 1-\delta \tag{12}$$

Donde, μ es una perturbación universal, *epsilon* el tamaño de la perturbación, $\mathbb{P}$ corresponde a un subconjunto de datos en X, C es un clasificador y δ cuantifica la tasa de engaño deseada para todas las imágenes de una distribución. Dependiendo del valor de δ es posible que se requieran múltiples iteraciones sobre un conjunto de imágenes, para mejorar la calidad de la perturbación. Al final de cada iteración, este método logra encontrar perturbaciones independientes que causan clasificaciones erróneas con un alto grado de confianza.



(a) (b)

Figura 10. Para (a) y (b) se muestra a la izquierda la imágenes originales con sus respectivas etiquetas. En el centro inferior de (a) y centro superior de (b) se muestra una perturbación universal. Y al lado derecho de (a) y (b) se muestran los ejemplos adversarios generados con sus etiquetas estimadas. Fuente: (Moosavi et al., 2017).

En el escenario de los ataques de caja negra, el modelo no está expuesto de forma transparente, de tal manera que el atacante solo conoce la salida del modelo, ya sea la etiqueta de clasificación o la distribución de probabilidades y utiliza esta información en iteraciones pasadas para analizar la vulnerabilidad del modelo. Por este motivo, los ataques de caja negra son considerados difíciles, ya que requieren un mayor número de iteraciones para generar una perturbación, en comparación a los ataques de caja blanca. En la siguiente subsección se presenta un ejemplo de un ataque de caja negra.

2.1.8. Salt and Pepper noise Attack(S&P)

Este ataque prueba la solidez de un modelo al agregar ruido de sal y pimienta distribuido en una imagen de entrada; como sugiere el nombre, el ataque genera una perturbación donde una fracción de píxeles, establecida por ϵ, se establece aleatoriamente como píxeles de color blanco (sal) en regiones oscuras y píxeles de color negro (pimienta) en las regiones claras, como se muestra en la Figura 11. En otras palabras, para una serie de píxeles (i, j, k) en una imagen de entrada RGB, una perturbación es calculada por la función $Noise(i, j, k, \epsilon)$, donde *Noise* selecciona si un pixel cambiará a un valor máximo (blanco) o mínimo (negro) y ϵ es la densidad total de ruido en una imagen.

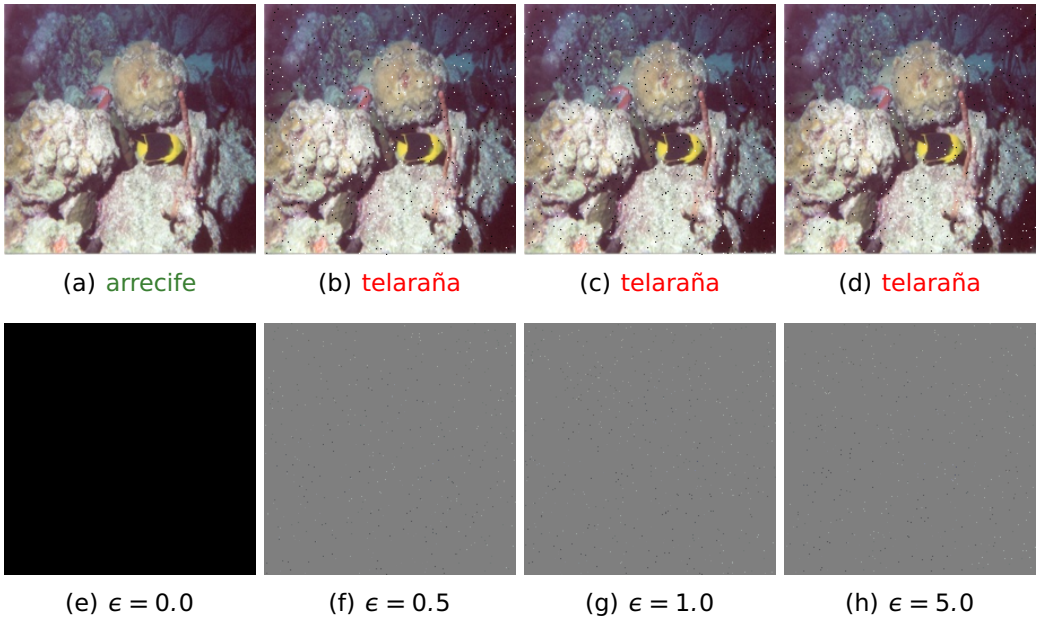
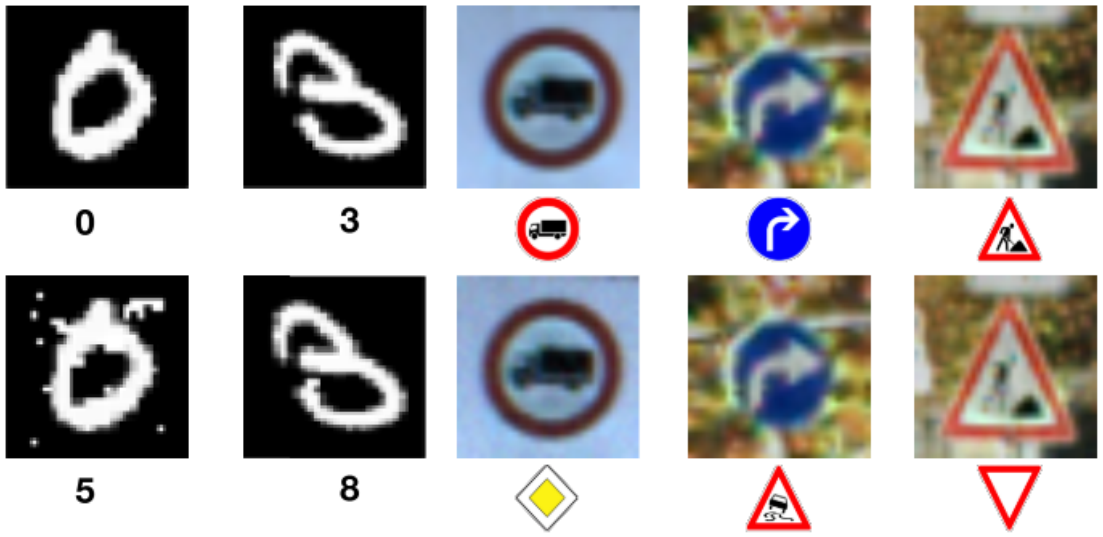


Figura 11. En la fila superior se muestra una imagen (a) correctamente clasificada como arrecife, tres ejemplos adversarios generados con el ataque S&P (b), (c) y (d). En la fila inferior se muestran las respectivas perturbaciones (e), (f), (g) y (h) de cada imagen superior.

2.1.9. Practical Black-Box Attack

Con el objetivo de generar ejemplos adversarios para un modelo objetivo utilizando solamente las etiquetas producidas por el mismo en (Papernot *et al.*, 2017) se desarrolla un ataque práctico de caja negra, a partir de un modelo sustituto, el cual es entrenado con un subconjunto de las imágenes de entrenamiento original y utilizando como etiquetas las proporcionadas por el modelo objetivo.

Posteriormente, los ejemplos adversarios se generan utilizando FGSM (ver sección 2.1.1) en el modelo sustituto y estos se transfieren al modelo objetivo, siendo esta propiedad de transferencia la principal aportación de este trabajo. Al evaluar su técnica, los autores atacan múltiples modelos de DNN utilizados en los conjuntos de imágenes de dígitos escritos a mano MNIST y señalamientos de transito alemanes GTSRD, generando clasificaciones erróneas en la mayoría de los casos. Algunos ejemplos de este ataque se pueden observar en la Figura 12.



(a) MNIST Dataset (b) GTSRD Dataset

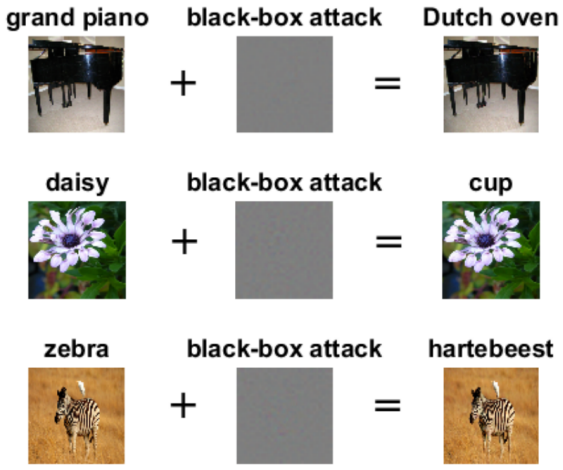
Figura 12. Se muestra ejemplos de Practical Black-Box Attack en el conjunto de datos MNIST(a) y el conjunto de datos GTSRD (b). En la parte superior se muestran imágenes originales con sus respectivas etiquetas y en la inferior se muestran los ejemplos adversarios generados con sus etiquetas estimadas. Fuente: (Papernot et al., 2017)

2.1.10. Zeroth Order Optimization (ZOO)

En (Chen *et al.*, 2017) se propone un ataque de optimización de orden cero (ZOO), el cual utiliza un modelo de optimización sin derivadas. Este método, es capaz de estimar el gradiente de una perturbación considerando el valor de una función objetivo g en dos puntos vecinos que corresponden a sumar o restar una pequeña perturbación, tal que

$$g(x') = max(\max_{i \neq l}(log[f(x)]_i) - (log[f(x)]_l, -k) \tag{13}$$

donde $[f(x)]_i$ representa la probabilidad de predicción de que la entrada x, pertenezca a la clase i, por su parte k es un parámetro de ajuste para la transferencia de ataques. Al emplear esta estimación, el ataque ZOO no necesita acceder al modelo de aprendizaje que está siendo atacado. Sin embargo, se requiere un costoso cálculo para consultar y estimar los gradientes. En sus resultados experimentales, Chen *et al.* demostraron que ZOO logró superar los ataques de caja negra que entrenan artificialmente un modelo sustituto propuesto por Papernot et al. y un rendimiento comparable al ataque de Carlini & Warner. En la Figura 13 se muestran algunos ejemplos de este ataque.



(a)

Figura 13. Se muestra ejemplos del ataque Zeroth Order Optimization. En la columna izquierda se muestran imágenes originales con sus respectivas etiquetas, en la columna central se muestra la perturbación generada por ZOO y en la columna derecha se muestran los ejemplos adversarios al combinar las dos columnas anteriores con sus etiquetas estimadas. Fuente: (Chen et al., 2017)

2.1.11. One Pixel Attack

Un caso extremo al generar un ataque adversario es cuando sólo se cambia el valor de un píxel en una imagen para engañar a un clasificador, cómo se muestra en la Figura 14. Para obtener estos ejemplos adversarios, Su *et al.* utilizaron un concepto de evolución diferencial, el cual busca optimizar dos valores, las dimensiones que necesitan ser perturbadas y la intensidad correspondiente de la modificación para cada dimensión.

Para una imagen de entrada, se crea un conjunto de 400 vectores en $\mathbb{R}^5$ de modo que cada vector contiene coordenadas (x, y) y valores RGB para un píxel candidato arbitrario. Posteriormente, los elementos de los vectores se modifican aleatoriamente para crear hijos de modo que un hijo compita con su padre en la siguiente iteración, mientras que la etiqueta estimada por el modelo se utiliza como criterio de selección. El último hijo superviviente se utiliza para alterar el píxel de la imagen de entrada.

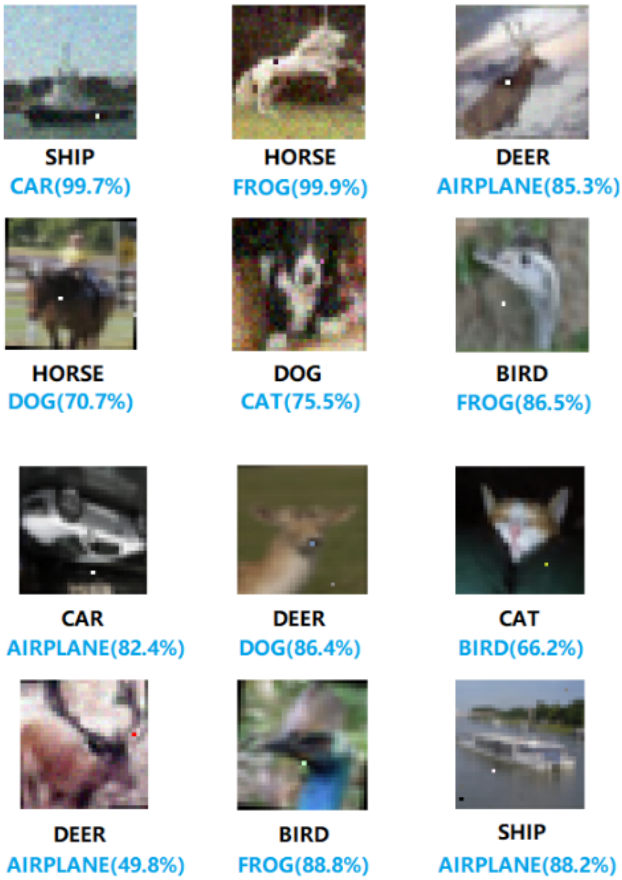


Figura 14. Se muestran ejemplos adversarios generados con One Pixel Attack en imágenes del conjunto de imágenes CIFAR-10. Para cada imagen, la etiqueta original se muestra de color negro y en color rojo la etiqueta estimada. Fuente: (Su et al., 2019)

2.1.12. Ataques adversarios físicos

Los ataques adversarios descritos en esta sección muestran cómo los sistemas de aprendizaje fallan en una prueba de visión que un niño podría realizar con facilidad. Sin embargo, existen otros ataques que burlan estos sistemas utilizando enfoques diferentes a los mencionados previamente; aunque los siguientes ataques no forman parte del trabajo realizado en esta tesis es importante conocer que existen estrategias novedosas para generar ejemplos adversarios.

Mientras que se ha reiterado que un ejemplo adversario consta de una modificación mínima en una imagen de entrada, en (Athalye *et al.*, 2018) llevan una "imagen adversaria" al mundo físico por medio de una tortuga impresa en 3D, la cual, contiene patrones capaces de engañar a un clasificador basado en DNN, específicamente el modelo Inception V3 desarrollado en (Szegedy *et al.*, 2016). Y aunque para el ojo humano simplemente se trata de una tortuga, este modelo estima que se trata de un rifle en casi cualquier posición posible, como se puede ver en la Figura 15.

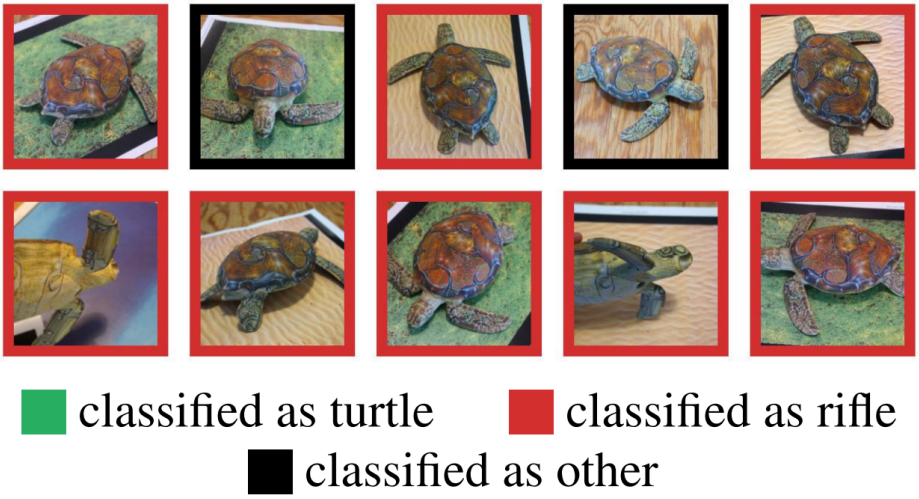


Figura 15. Se muestran una figura adversaria en 3D de una tortuga, la cual es clasificada como un rifle en casi cualquier posición que esta se encuentre. Fuente: (Athalye et al., 2018)

Para desarrollar esta figura adversaria en 3D, Athalye *et al.* presentan un algoritmo llamado Expectation Over Transformation (EOT), el cual permite la construcción de ejemplos que siguen siendo adversarios sobre una distribución de transformación T elegida.

En el enfoque clásico, los ejemplos adversarios se generan maximizando una probabilidad logarítmica de una clase objetivo y_t sobre un radio ϵ alrededor de la imagen original x, tal que

$$\underset{x'}{arg\,max} \quad log\ f(x'|y_t) \tag{14}$$

$$s.t. \quad ||x' - x||_p < \epsilon \tag{15}$$

Por su parte, en lugar de optimizar la verosimilitud logarítmica de un solo ejemplo, EOT utiliza una distribución T elegida de funciones de transformación t, tomando una entrada x' controlada por el ejemplo adversario "verdadero" de entrada $t(x')$ estimado por un clasificador, es decir, se elige el ejemplo adversario optimizado a partir de las funciones de tranformación t aplicadas. Además, en vez de utilizar una norma de $x' - x$ para restringir el espacio de solución, EOT busca restringir la distancia efectiva esperada d entre las entradas adversarias y originales, la cual define como

$$\underset{x'}{arg\,max} \quad \mathbb{E}_{t\sim T}\left[log\ f(y_t|t(x')\right] \tag{16}$$

$$s.t. \quad \mathbb{E}_{t\sim T}\left[d(t(x'), t(x))\right] < \epsilon \tag{17}$$

En la práctica, la distribución T modela distorsiones perceptivas como rotaciones aleatorias, traslaciones o adición de ruido. Sin embargo, el método generaliza más allá de simples transformaciones; Las transformaciones en T pueden realizar operaciones como la representación 3D de una textura.

Cómo una expansión del método anterior, en (Brown *et al.*, 2017) se presentan los parches adversarios capaces de engañar efectivamente a un modelo de red neuronal en un escenario de la vida cotidiana, sin modificar directamente el objeto que está siendo atacado. Por lo tanto, es posible imprimir un parche y colocarlo cerca de un objeto que se desee engañar, como se muestra en la Figura 16, donde el parche se genera con la intención que el modelo estime una clase en específico, para este caso la clase tostadora, lo que puede ser muy peligroso.

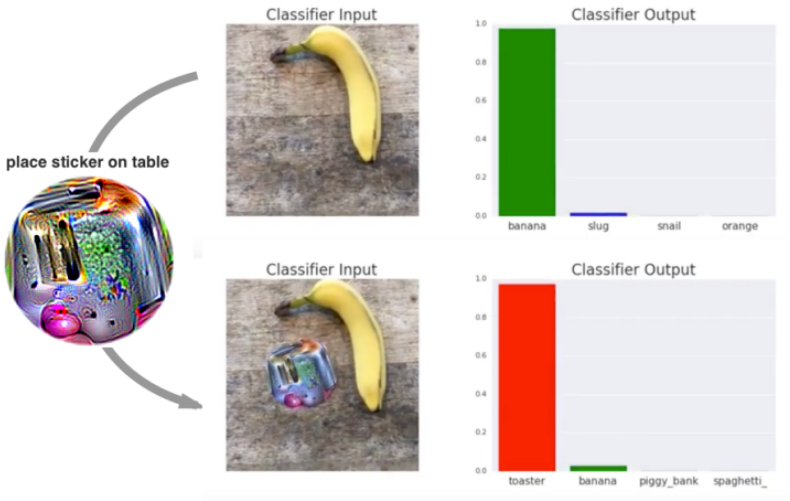


Figura 16. Se muestran como la imagen de una banana (parte superior) cambia su clasificación al colocar un parche adversario (parte inferior) y esta nueva imagen es clasificada como una tostadora. Fuente: (Brown et al., 2017)

Para generar un parche adversario independiente de una imagen en específico, un parche inicial es entrenado sobre varias imágenes, aplicando traslaciones, escalas y rotaciones aleatorias en el parche. En particular, para una imagen dada x, un parche p, con ubicación l y transformaciones t (rotaciones o escalado), se define un operador de aplicación de parche $A(p, x, l, t)$, que primero aplica las transformaciones t al parche p, y luego aplica el parche transformado p a la imagen x en una ubicación l. Concretamente, para obtener un parche adversario $\hat{p}$, se utiliza una variante de EOT, descrita previamente, donde un $\hat{p}$ es generado al optimizar la función objetivo

$$\hat{p} = \arg\max_{p} \mathbb{E}_{x\sim X, t\sim T, l\sim L}[\log f(y_t|A(p, x, l, t)] \tag{18}$$

Los ataques adversarios mencionados en esta sección no son, en la actualidad, un peligro significativo para el público. Son prácticos, sí, pero en circunstancias limitadas. Además, aunque la visión por computadora se implementa con mayor frecuencia en el mundo real, todavía no se depende tanto de ella como para que una persona mal intencionada con una impresora 3D pueda causar estragos. Su importancia radica en que ejemplifican cuán frágiles pueden ser algunos sistemas de aprendizaje. Además, si no se da solución a estos problemas ahora, podrían generar problemas mucho más importantes en el futuro.

En la literatura relacionada, es posible observar que existen un mayor número de trabajos sobre ataques de caja blanca en comparación con ataques de caja negra y que más ataques utilizan perturbaciones basadas en ruido que transformaciones geométricas. Pero es importante tener en cuenta que en escenarios es más realista, como un automóvil autónomo, el atacante no tiene pleno conocimiento del sistema que está siendo atacado o si algún mecanismo de una defensa ha sido implementado.

2.2. Defensas Adversarias

Los ejemplos adversarios muestran que las DNN se pueden engañar fácilmente. En la literatura reciente se pueden encontrar defensas para contrarrestar ejemplos adversarios. Sin embargo, la mayoría de las estrategias de defensa actuales no se adaptan a todos los tipos de ataques adversarios existentes, ya que un método puede ser muy efectivo ante un ataque en particular y ser vulnerable ante otro ataque.

Las estrategias de defensa, de manera similar a los ataques adversarios, pueden ser englobadas en dos tipos. El primer tipo se les conoce como *guardias* porque no imponen ninguna restricción al modelo solo toman precauciones. El segundo tipo de defensa actúa directamente sobre un modelo de clasificación modificando su arquitectura, datos de entrenamiento o función de pérdida y son conocidas como *defensas por diseño*. En las siguientes subsecciones se describen las defensas adversarias más representativas de cada tipo, resaltando sus ventajas y desventas.

2.2.1. Guardianes adversarios

Por lo regular, los guardianes adversarios (también conocidos como detectores adversarios) modifican el proceso de clasificación al agregar un paso de preprocesamiento el cual permite detectar ejemplos adversarios ó reducir su impacto al realizar una transformación en la entrada.

Los detectores adversarios, generalmente, se basan en clasificadores binarios para identificar los datos de entrada como legítimos ó adversarios. En (Grosse *et al.*, 2017), se propone un modelo de detección basado en pruebas estadísticas para diferenciar entre imágenes de entrada normales y ejemplos adversarios. Grosse *et al.*, afirman

que las pruebas estadísticas de discrepancia media máxima (MMD) y distancia de energía (ED), permiten detectar que las imágenes generadas con ataques FGSM se encuentran fuera de la distribución de los datos de entrenamiento.

Otros detectores como (Metzen *et al.*, 2017; Li y Li, 2017) proponen entrenar modelos binarios basados en características extraídas de distintas capas de una DNN. Estos detectores muestran un buen resultado contra ataques como FGSM, pero el rendimiento del detector disminuye contra ataques iterativos BIM o más complejos como C&W.

En general, los detectores de adversarios son adecuados para problemas en los que descartar una clasificación no es una misión crítica. Sin embargo, en algunas aplicaciones como un vehículo autónomo no es posible negarse a clasificar una imagen, ya que esta puede ser una señal de tráfico importante. Además, aunque los detectores presentados previamente utilizan características muy distintas, todos ellos se ven limitados o no detectan ataques adaptativos como VA, DDN o C&W.

Por su parte, las transformaciones en la entrada, como guardián defensivo, muestran un gran potencial debido a la facilidad de implementación. Se piensa que algunas transformaciones restringen el espacio de los ejemplos adversarios, por lo que disminuyen su impacto. Particularmente, Guo *et al.* sugieren el uso de la minimización de varianza total y el acolchado de imágenes[1], como paso previo de un clasificador. En la Figura 17 se pueden observar ejemplos de estas transformaciones.

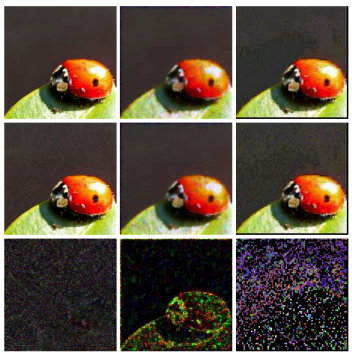


Figura 17. Muestra las transformaciones donde, de izquierda a derecha, las columnas corresponden a: sin transformación, minimización de la varianza total y acolchado de imágenes. De arriba a abajo, las filas corresponden a: la imagen original, la imagen adversaria generada con BIM y la diferencia entre las dos imágenes de arriba.

[1] El acolchonado de imágenes (Efros y Freeman, 2001) es una técnica no paramétrica que sintetiza imágenes juntando pequeños parches y calcula los cortes mínimos en una región superpuesta para eliminar los bordes.

Las transformaciones en una imagen conducen a gradientes estocásticos, que hacen que los ataques basados en análisis de sensibilidad, cómo el FGSM, BIM y PGD, pierdan efectividad. Sin embargo, cuando se prueban contra ataques que utilizan una estrategia de optimización, como C&W , las defensas basadas en transformaciones fallan. Además, las transformaciones pueden provocar efectos secundarios, por ejemplo, comprimir una imagen de alta resolución puede llevar a una pérdida de precisión.

2.2.2. Defensas por diseño

Mientras que los guardianes defensivos son considerados reactivos, las estrategias defensivas por diseño son proactivas ya que tratan de anticipar los ataques, por tal motivo, estas defensas requieren reentrenamiento de los modelos realizando alguna modificación en su arquitectura, los datos de entrenamiento, función de pérdida o una combinación de estas.

Una solución natural, propuesta en (Goodfellow *et al.*, 2014; Mądry *et al.*, 2017), consiste en aumentar la robustez de un modelo al agregar ejemplos adversarios en el conjunto de datos de entrenamiento. Este entrenamiento adversario es un enfoque de fuerza bruta donde el objetivo es asegurándose que un modelo predecirá con la misma clase imágenes originales y perturbadas. Sin embargo, el entrenamiento adversario solo es útil contra el ataque para el que fue entrenado. Por otro lado, en (Tramèr *et al.*, 2020) se demostró que el entrenamiento adversario pierde efectividad fácilmente ante ataques de caja negra.

La defensa por diseño más confiable, es el destilado de DNN. El término destilado fue propuesto originalmente en (Hinton *et al.*, 2015) como una forma de transferir conocimiento de una red neuronal a un modelo más pequeño. Por su parte en (Papernot *et al.*, 2016) propusieron el uso de destilado como un mecanismo de defensa contra ejemplos adversarios, donde a diferencia del enfoque tradicional, se mantienen la misma arquitectura.

El procedimiento de entrenamiento destilado se describe a continuación y se ilustra en la Figura 18:

1. Supongamos que X es un conjunto de entrenamiento con sus etiquetas de clase.

Específicamente, sea $x \in X$ una muestra y $y(x)$ denota su etiqueta de clase. La etiqueta $y(x)$ generalmente se codifica como un vector one-hot.

2. Dado el conjunto de entrenamiento $(x, y(x))$ para $x \in X$, se entrena un modelo de red neuronal f con una capa de salida softmax a temperatura de destilado $T \geq 1$. Tal que,

$$f_i(X) = \frac{e^{\frac{z_i(x)}{T}}}{\sum_{i=1}^{|Y|} e^{\frac{z_i(x)}{T}}} \tag{19}$$

   Por tanto, $f(x)$ es el vector de probabilidad sobre la clase de todas las etiquetas posibles.

3. Se crea un nuevo conjunto de entrenamiento considerando muestras de la forma $(x, f(x))$ para $x \in X$. En vez de usar la etiqueta de clase $y(x)$ para x, se usa la etiqueta suavizada $f(x)$ para codificar las probabilidades de predicción del modelo sobre la etiqueta de clasificación.

4. Posteriormente, se emplea el nuevo conjunto de entrenamiento $(x, f(x))$ para $x \in X$ para entrenar un modelo f_d, con la misma arquitectura que f, y la misma temperatura T utilizada en la primera etapa de entrenamiento.

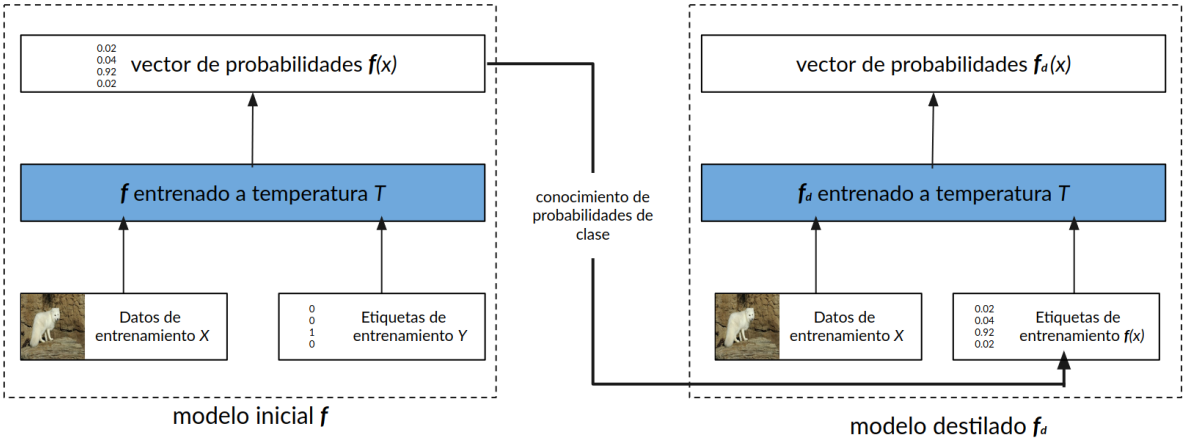


Figura 18. Muestra el procedimiento de destilado de DNN como sistema de defensa ante ejemplosadversarios. Fuente: (Papernotet al., 2016).

Entrenar un modelo con etiquetas suavizadas $f(x)$ evita que un modelo se ajuste demasiado a los datos de entrenamiento y contribuye a una mejor generalización entorno a estos. Un inconveniente de esta estrategia es que el proceso de entrenamiento

requiere el doble de tiempo, un ciclo de entrenamiento clásico para ajustar la arquitectura al conjunto de datos y poder obtener las etiquetas suavizadas para poder realizar un segundo ciclo de entrenamiento. Otro problema con este método es encontrar el valor correcto del parámetro de temperatura T; cuanto mayor sea la temperatura, más ambigua será su distribución de probabilidad, es decir, todas las probabilidades de la salida $f(x)$ estarán cercanas a $1/N$. En contraste, con una temperatura muy pequeña, la distribución de probabilidad estará más concentrada, por ejemplo, solo un valor del vector de salida $f(x)$ cercano a 1 y los valores restantes estarán cerca de 0, similar el etiquetado de clases original one-hot.

En general, la mayoría de las defensas propuestas dan lugar a una falsa sensación de seguridad asumiendo que un atacante no sabe que una defensa está en uso. Además, existe una confrontación persistente entres ataques y defensas, es decir, mientras que una defensa es desarrollada con el objetivo de contrarrestar un ataque en particular, un nuevo ataque adversario es desarrollado con el propósito de volver obsoleta a esa defensa. Particularmente, el entrenamiento adversario fue desarrollado con el objetivo de contrarestar el ataque FGSM y aunque este entrenamiento logró su cometido, bajo ciertos escenarios, es ineficaz ante el ataque C&W, ante este último el procedimiento de destilado es efectivo, pero poco puede hacer ante ataques de caja negra. Por tal motivo, se requiere una estrategia defensiva para abordar este problema desde una perspectiva diferente.

En la literatura, el número de estrategias para generar ejemplos de adversarios es abrumadoramente mayor en comparación con las técnicas de defensa. En este trabajo, se considera que la razón principal radica en la débil codificación del etiquetado de clases. Específicamente, en la codificación one-hot, que típicamente se utiliza en el entrenamiento de DNN para que un modelo aprenda una asociación entre imágenes y etiquetas. Aunque esta codificación es una convención utilizada en el área de aprendizaje de maquina, existen otras que pueden ser utilizadas para realizar la asociación anterior, en el siguiente capítulo se describen algunas alternativas a la codificación one-hot.

Capítulo 3. Códigos de Corrección de Errores Lineales

Es posible relacionar la vulnerabilidad de la codificación one-hot en DNN con el problema de transmisión de información a través de un canal de comunicación, donde es necesario determinar que mensaje fue enviado sobre la base de lo que se recibe, debido a la posibilidad de que se corrompa la información durante la transmisión.

En (Shannon, 1948) se identificó la capacidad de transmisión de un canal y se demostró que es posible una comunicación arbitrariamente confiable por debajo de la capacidad del canal, es decir, los resultados de Shannon garantizan que un mensaje se pueden codificar, de tal manera que los datos alterados durante una transmisión se pueden decodificar con cierto grado de precisión.

Una característica común en los canales de comunicación es el envío de mensajes hacia un receptor. Por ejemplo, para un disco compacto, el mensaje puede ser una imagen, audio o datos que se colocan en él, el canal es el disco en sí y es “ruidoso” en el sentido de que lo que se recibe no siempre es lo mismo que lo que se envió. En este caso, el ruido en un disco puede ser causado por rayones.

Una forma de evitar la perdida de información durante el proceso de transmisión de un mensaje a través de un canal de comunicación se ilustra en la Figura 19. Partiendo de un mensaje inicial x, este pasa a ser codificado utilizando un código c, esta codificación permite aumentar la confiabilidad de transmisión ante la posibilidad que se presenten ruido e. El teorema de Shannon garantiza que la transmisión anterior se cumplirá con cierto porcentaje de éxito, aunque debido al ruido jamás será 100 %, por lo que es probable recibir un mensaje y distinto al inicial. Sin embargo, una codificación permite detectar y corregir los errores que se presenten y una vez decodificado, es posible generar una estimación $\hat{x}$ del mensaje original, donde se espera que $\hat{x}$ sea lo más parecido a x.

La demostración del teorema en (Shannon, 1948) es probabilística y no constructiva, es decir, no se produjeron códigos específicos que brinden una precisión deseada para un canal determinado, sólo se garantiza su existencia. El objetivo de investigación en el área de la teoría de codificación consiste en producir códigos que cumplan las condiciones del teorema de Shannon. Desde su surgimiento, múltiples códigos de corrección y detección de errores han sido desarrollados.

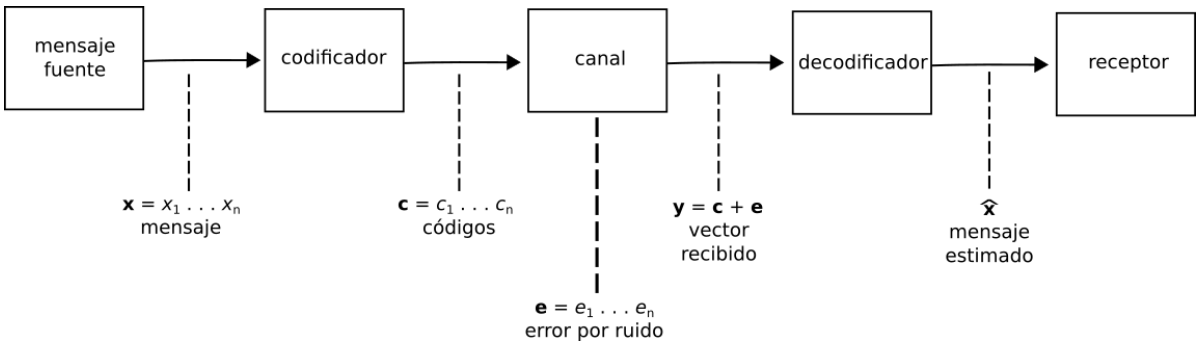


Figura 19. Muestra el procedimiento de codificar y decodificación de un mensaje a través de un canal propenso al ruido.

La forma más sencilla de codificar información en presencia de ruido es repetir cada símbolo del mensaje un número fijo de veces. Supongamos que nuestra información son los elementos $\{0, 1\}$ y repetimos cada símbolo n veces. Si tenemos, n = 7, entonces al querer enviar un 0 enviamos 0000000 y para enviar un 1 enviamos 1111111. Para este ejemplo, si se cometen como máximo tres errores en la transmisión, al decodificar por “mayoría de votos”, entonces podemos determinar correctamente el símbolo enviado 0 ó 1.

Para el ejemplo anterior, el código c consta de dos palabras de código 0 = 0000000 y 1 = 1111111 y se denomina como un código lineal binario de longitud n. Este código puede corregir hasta $e = \lfloor (n-1)/2 \rfloor$ errores, es decir, si se cometen como máximo e errores en un vector recibido y, el código original enviado se puede recuperar.

Formalmente, un código lineal de longitud n y rango k se define como un subespacio lineal C con dimensión k del espacio vectorial $\mathbb{F}_q^n$, donde $\mathbb{F}_q$ corresponde a un espacio finito con q elementos. Un código se denomina q-ario, si $q = 2$ o $q = 3$ y este es conocido como un código binario o un código ternario respectivamente. Por su parte, los vectores $[n, k]$ en C se les conoce como palabras de código o simplemente códigos.

Los códigos lineales son los más estudiados en teoría de codificación, siendo los códigos de (Hamming, 1950) y (Reed y Solomon, 1960) algunos de los más usados debido a su estructura algebraica, facilidad de construcción, codificación y decodificación. La idea central de los códigos lineales consiste en aumentar la distancia de Hamming entre los vectores de código para realizar la recuperación de mensajes con

mayor confiabilidad. Siendo la distancia de Hamming $d(x, y)$ entre dos vectores de código x, y, el número de valores en los que x y y difieren. En la siguiente sección se presentan los códigos elegidos para este trabajo.

Particularmente, los códigos Reed-Solomon son aplicados en áreas como almacenamiento de información (CDs y DVDs), telefonía móvil, códigos QR y en comunicaciones por satélite. Estos códigos se especifican como $RS(n, k)$ con símbolos de s bits, esto significa que un codificador toma k símbolos de los s bit y añade símbolos de paridad para hacer una palabra de código de n símbolos. Existen $n - k$ símbolos de paridad de s bits cada uno. Un decodificador puede corregir hasta t símbolos que contienen errores en una palabra de código, donde $2t = n - k$.

Como componente clave en los discos compactos, la codificación Reed-Solomon fue el primer uso de la codificación de corrección de errores en un producto de consumo producido en masa. En el CD, se utiliza un esquema denominado codificación Reed-Solomon entrecruzada (CIRC) que consta de un código interno $RS(32, 28)$ relativamente débil, ya que puede corregir errores de 2 bytes en bloques de 32 bytes; lo que se traduce en corregir errores de hasta 4000 bits, aproximadamente 2.5 mm en la superficie de un disco. Por su parte, los DVDs usan un esquema similar, pero con un código interno $RS(208, 192)$. En otra aplicación, los códigos $RS(1, 5)$ fueron utilizados para transmitir a la tierra fotografías de marte tomadas por la nave espacial Mariner 9.

Es posible notar que la importancia de los códigos de corrección de errores lineales radica en incluir información redundante, la suficiente como para permitirle a un receptor deducir cual fue la información que se transmitió y por lo tanto, el receptor tiene la capacidad para corregir un número limitado de errores. Sin embargo, encontrar un algoritmo de decodificación eficiente es uno de los problemas fundamentales en el área de teoría de la codificación, siendo los códigos lineales la mejor solución conocida, ya que permiten codificar y decodificar información de manera más eficaz que otros códigos. En la siguiente sección se describen los códigos lineales utilizados en este trabajo.

3.1. Códigos de Hadamard

Primeramente es necesaria una breve revisión de los códigos binarios de Reed-Muller(Reed, 1953; Muller, 1954), ya que la definición de estos códigos es utilizada para la construcción de los códigos de Hamming, Reed-Solomon, entre otros (Huffman y Pless, 2010).

La definición recursiva para construir códigos binarios Reed-Muller (RM) de tamaño 2^m es la siguiente, para un entero positivo m y un entero no negativo r tal que $r \leq m$, existe el código binario RM de orden r-ésimo. Dado por

$$R(r,m) = \{(u, u+v) | u \in R(r, m-1), v \in R(r-1, m-1)\} \tag{20}$$

Observe que los códigos $R(0,m)$ y $R(m,m)$ son triviales: el código de orden 0 : $R(0,m)$ corresponde al código de repetición binario de tamaño 2^m con base 1, mientras que el código RM de orden m : $R(m,m)$ es el espacio completo de posibles códigos.

En el siguiente ejemplo, tenemos $m = 4$ y $r = 2$, por lo que el tamaño de cada vector será $n = 2^m = 16$ y un total de $k(r,m) = 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r} = 11$ vectores de código RM. Para cada vector v_i, donde $1 \leq i \leq m$, es una tupla binaria que consiste de 2^{m-1+1} tuplas alternadas de cero y uno. Para este ejemplo, tenemos

$$RM(2,4) = \begin{bmatrix} v_0 = 1111111111111111 \\ v_1 = 0101010101010101 \\ v_2 = 0011001100110011 \\ v_3 = 0000111100001111 \\ v_4 = 0000000011111111 \\ v_1v_2 = 0001000100010001 \\ v_1v_3 = 0000010100000101 \\ v_1v_4 = 0000000001010101 \\ v_2v_3 = 0000001100000011 \\ v_2v_4 = 0000000000110011 \\ v_3v_4 = 0000000000001111 \end{bmatrix} \tag{21}$$

En este trabajo, se utilizan los códigos de (Hadamard, 1893), también conocidos como Walsh-Hadamard para algunos autores. Estos códigos son un caso especial de los códigos Reed-Muller, ya que corresponden a su primer orden. La forma más común de representar los códigos de Hadamard es con una matriz H de tamaño $n\,x\,n$ en la que todos sus valores correspondan a valores ± 1 y sus filas son ortogonales. Se inicia a partir de una matriz $H_1 = [1]$ como una matriz de códigos de Hadamard de orden 1. Mientras que una matriz de códigos de Hadamard de orden 2 corresponde a:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{22}$$

Para construir una matriz de códigos de Hadamard de tamaño $n = 2^k$, es posible utilizar el producto tensorial de la siguiente manera, sea $A = [a_{i,j}]$ una matriz de Hadamard de tamaño $n\,x\,n$ y sea $B = [b_{k,l}]$ una matriz de Hadamard de tamaño mxm. Entonces, el producto tensorial de A y B es una matriz de Hadamard de tamaño $nm\,x\,nm$, donde:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,n}B \\ & & \vdots & \\ a_{n,1}B & a_{n,2}B & \dots & a_{n,n}B \end{bmatrix} \tag{23}$$

Siguiendo la definición anterior, tenemos que para $H_4 = H_2 \otimes H_2$:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{24}$$

En la Eq. 24 es posible observar lo siguiente:

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} \tag{25}$$

donde H_2 corresponde a la Eq. 22. Por lo que una alternativa recursiva propuesta por (Sylvester, 1867) para la construcción de una matriz de códigos de Hadamard H_{2^k} es la siguiente:

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix} \tag{26}$$

Aunque en su definición, los códigos de Hadamard se construyen con valores de ± 1, en problemas de transmisión de información, es común sus sustituir los valores de -1 por 0, si se aplica esta sustitución en la matriz H_4 se obtiene la siguiente matriz binaria de Hadamard C_4:

$$C_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \tag{27}$$

En el ejemplo anterior, es posible observar que entre cualquier par de vectores fila o columna se tienen una distancia de Hamming de 2. Para toda toda matriz binaria de Hadamard C_n, donde $n = 2^k$, cualquier par de vectores está a una distancia de 2^{k-1}. En general, la distancia relativa entre diferentes vectores es $n/2$, donde n corresponde a la longitud del vector de código y es posible recuperar como máximo $n/4 - 1$ errores relativos.

Para ilustrar el proceso de transmisión y recuperación de errores, se presenta un ejemplo considerando 16 bits para la longitud de cada código, donde el número de errores como máximo es $16/4 - 1 = 3$, para que sea posible recuperar el código original. A continuación, se muestra la matriz de códigos binarios C_{16} construida utilizando el enfoque de Hadamard explicado anteriormente por la Eq. 26:

$$C_{16} = \begin{bmatrix} c_1 = 1111111111111111 \\ c_2 = 1010101010101010 \\ c_3 = 1100110011001100 \\ c_4 = 1001100110011001 \\ c_5 = 1111000011110000 \\ c_6 = 1010010110100101 \\ c_7 = 1100001111000011 \\ c_8 = 1001011010010110 \\ c_9 = 1111111100000000 \\ c_{10} = 1010101001010101 \\ c_{11} = 1100110000110011 \\ c_{12} = 1001100101100110 \\ c_{13} = 1111000000001111 \\ c_{14} = 1010010101011010 \\ c_{15} = 1100001100111100 \\ c_{16} = 1001011001101001 \end{bmatrix} \tag{28}$$

Para este ejemplo, suponga que se transmite un mensaje correspondiente al código $c_6 = 1010010110100101$ y, debido al ruido, se recibe un vector con tres errores $s = 1011010100101101$. Al calcular la distancias de Hamming entre el mensaje recibido y cada uno de los códigos en C_{16} tenemos

código	distancia	
1011010100101101		
1111111111111111	7	
1010101010101010	9	
1100110011001100	9	
1001100110011001	7	
1111000011110000	9	
1010010110100101	3	<–
1100001111000011	11	
1001011010010110	9	
1111111100000000	7	
1010101001010101	9	
1100110000110011	9	
1001100101100110	7	
1111000000001111	5	
1010010101011010	7	
1100001100111100	7	
1001011001101001	5	

donde el código de menor distancia, corresponde al código transmitido inicialmente.

Es posible recuperar un número mayor de errores aumentando la longitud del código. Por ejemplo, tres errores es el límite para un código de longitud 16 y siete errores para códigos de longitud 32. Mientras que el mecanismo de decodificación consiste en simplemente encontrar el código con la distancia mínima en la matriz de códigos binarios de Hadamard.

Desde la perspectiva de DNN en un problema de clasificación de imágenes, un ataque adversario tiene como objetivo realizar cambios mínimos en una imagen de entrada. Estos cambios se pueden considerar ruido durante la transmisión de un mensaje; el receptor, en este caso un clasificador, necesita decodificar la clase de la imagen de entrada. Mientras que el atacante necesita cambiar solo un bit de información en el vector de salida, siendo este es el umbral más bajo, para cambiar la clasificación de

una clase y generar con éxito el ejemplo adversario.

En el problema de transmisión de información a través de un canal ruidoso, presentado al inicio de este capítulo, agregar redundancia a un mensaje aumenta la confiabilidad de transmisión. Es posible intentar hacer exactamente lo mismo como defensa contra ejemplos adversarios. La idea general consiste en aumentar la distancia de Hamming entre los códigos de clase, ya que la distancia entre cualquier par de códigos one-hot diferentes es dos independientemente del número de clases. Mientras que al utilizar códigos de corrección de errores, en particular códigos de Hadamard, permite aumenta la distancia de Hamming entre vectores a $n/2$ siendo n el número de clases; por lo que un ataque adversario necesitaría cambiar $n/4$ bits para clasificar incorrectamente una imagen, lo que se traduce en generar una perturbación más severa y visible para obtener un ejemplo adversario exitoso.

Capítulo 4. Metodología

En el capítulo 2, se mencionó que los ataques adversarios aún se consideran un problema abierto, ya que las estrategias defensivas diseñadas para contrarestarlos se especializan en contener a uno ó pocos ataques a la vez. Por otra parte, la robustez de algunas soluciones defensivas, como los modelos destilados, dependen de algún parámetro (ver Sección 2.2).

Adicionalmente, el etiquetado one-hot es una regla del aprendizaje supervisado en DNN y aunque es ampliamente utilizado, este etiquetado no considera el dominio del problema; es una convención que aún no ha sido cuestionada dado que presenta buenos resultados.

4.1. Propuesta

El objetivo de esta tesis consiste en implementar un etiquetado de clases basado en códigos Hadamard (ver Figura 20) al entrenar un modelo de DNN, como estrategia defensiva ante ataques adversarios. Cabe mencionar, que en (Yang *et al.*, 2015) proponen el uso de códigos de corrección de errores para mejorar las características discriminatorias en DNN. Sin embargo, los autores no estudian el problema de los ejemplos adversarios.

class	one-hot codification	Hadamard codification
0	1 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1	0 1 0 0 0 0 0 0 0 0	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
2	0 0 1 0 0 0 0 0 0 0	1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
3	0 0 0 1 0 0 0 0 0 0	1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1
4	0 0 0 0 1 0 0 0 0 0	1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
5	0 0 0 0 0 1 0 0 0 0	1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1
6	0 0 0 0 0 0 1 0 0 0	1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
7	0 0 0 0 0 0 0 1 0 0	1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0
8	0 0 0 0 0 0 0 0 1 0	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
9	0 0 0 0 0 0 0 0 0 1	1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1

Figura 20. Muestra la codificación one-hot y Hadamard de las clases 10 correspondientes al dataset MNIST.

Para la implementación de una codificación robusta de clases, se realiza el siguiente proceso de entrenamiento. Dado cualquier modelo de DNN, se elimina la capa de salida softmax y se sustituye por una capa densa, cuyo vector de salida correspondiente en tamaño a los códigos que se estén utilizando. En caso de utilizar un modelo preentrenado, se recomienda congelar los pesos intermedios del modelo, conocidos como deep features y únicamente entrenar la parte del clasificador del modelo.

Posteriormente, el proceso de entrenamiento es parecido al realizado con una codificación one-hot. Para cada una de las clases del problema se asigna un código de Hadamard como se muestra en la Figura 20, el cual es asociado a cada una de las imágenes de dicha clase.

En un modelo de clasificación que utiliza la codificación one-hot, la predicción de una clase corresponde a la posición con valor máximo en el vector de salida. Mientras que una predicción en el modelo propuesto corresponde al código de menor distancia entre la salida del modelo y la matriz de códigos de Hadamard utilizados, similar al método de decodificación de mensajes mostrado en el ejemplo al final del Capítulo 3.

Generalmente, la salida de un modelo es un vector de números reales positivos, como se muestra en la Figura 21 (a). Con el cambio de codificación, aunque se espera que la salida sea un vector binario, al obtenerse de una capa densa, es probable que este también corresponda a un vector de números reales; de tal manera que para realizar una predicción se utiliza el error cuadrático medio (MSE por sus siglas en inglés), para calcular la distancia entre el vector de salida del modelo y los códigos de Hadamard correspondientes a cada clase. De forma similar, durante el proceso de entrenamiento, MSE se utiliza como función de pérdida.

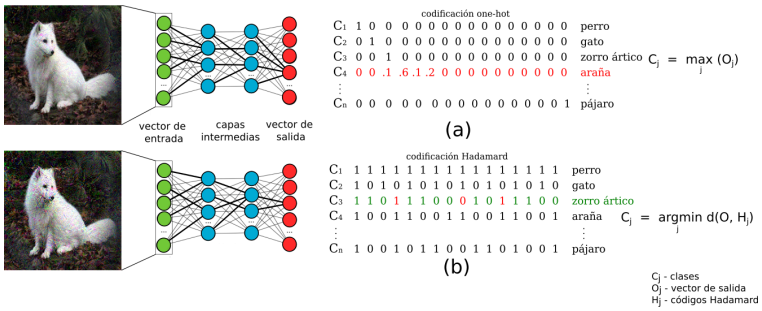


Figura 21. Muestra en la parte superior la salida al evaluar un ejemplo adversario con una DNN utilizando la codificación one-hot y en la parte inferior la salida al evaluar un ejemplo adversario con una DNN que utiliza la codificación de Hadamard.

Independientemente del cambio de codificación, se espera que un ataque genere un ejemplo adversario capaz de intercambiar uno ó más valores en el vector de salida, como se muestra en la Figura 21 (b). Si el número de valores que se cambian es menor a $n/4$, siendo n el tamaño del vector de salida, un ejemplo adversario es correctamente clasificado, ya que la distancia de menor valor corresponde al código con el que fue etiquetado. Por el contrario, un ejemplo adversario es clasificado incorrectamente si la distancia entre el vector de salida y el código correspondiente a la clase original es igual o mayor $n/4$.

En el siguiente capítulo se describen los experimentos realizados para caracterizar la robustez de la codificación propuesta ante ejemplos adversarios, así como algunas variantes de la misma.

Capítulo 5. Resultados

Con el objetivo de evaluar la robustez de la clasificación al utilizar códigos de Hadamard para el etiquetado de clases en comparación a una codificación tradicional, en este capítulo, se presentan distintos experimentos que dan validez al método propuesto. Cada uno de ellos se comparan de acuerdo a una métrica de precisión Top-1 en un conjunto de validación, es decir, la razón entre las predicciones correctas y el número total de predicciones realizadas para un conjunto de imágenes que no fueron consideradas durante el proceso de entrenamiento.

En cada experimento se utiliza un modelo base, el cual no aplica ninguna estrategia de defensa y es utilizado como referencia para mostrar cómo un ataque adversario afecta la precisión de clasificación a medida que se aumenta su intensidad. Adicionalmente, para realizar una comparación contra la mejor estrategia defensiva conocida, se incluyen variantes destiladas del modelo base, donde cada variante fue entrenada con un valor de temperatura T igual a 1, 5, 10 y 25. Por último, se incluye una variante del modelo base utilizando la codificación de Hadamard.

Para generar los ejemplos adversarios necesarios para cada experimento se utilizó la librería Foolbox (Rauber *et al.*, 2020). Se decidió usar esta librería ya que cuenta con la implementación de los principales ataques mencionados en la literatura relacionada. Cabe mencionar que cada ataque se generó utilizando sus parámetros predeterminados. El único parámetro que se modificó fue el valor de percepción de la pertubación ϵ, donde se utilizaron valores dentro del intervalo $[0, 1]$. Es importante recordar que un ejemplo adversario cumplirá su objetivo de clasificar incorrectamente una imagen a medida que se aumenta el valor de ϵ, sin embargo, esto también aumenta la posibilidad de que un observador humano descubra el engaño.

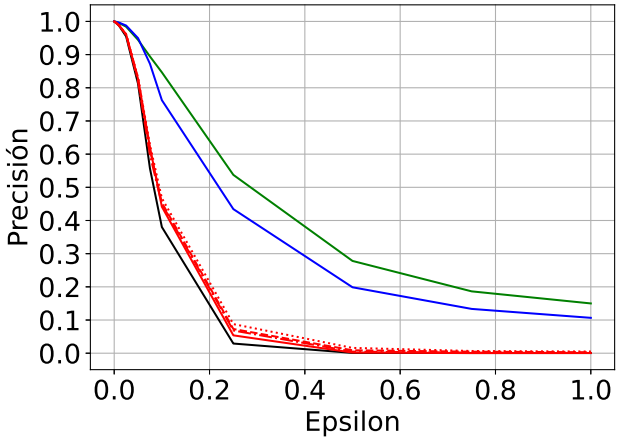
5.1. Experimentos MNIST

Para el primer experimento, se eligió utilizar MNIST (Deng, 2012), este conjunto de imágenes de dígitos escritos a mano contiene 60,000 imágenes de entrenamiento y 10,000 imágenes de validación. Como modelo base se utilizó LeNet5 (LeCun *et al.*, 2015), ya que ha sido utilizado con éxito para dar solución a este problema de clasificación. Las variantes destiladas de modelo LeNet5 corresponden a los valores de

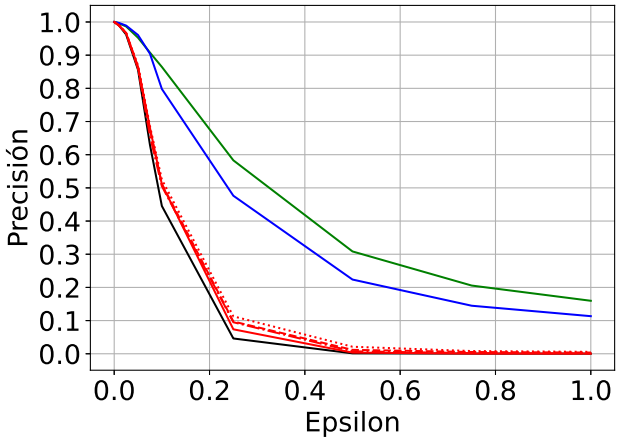
temperatura T antes mencionados. Mientras que las variantes entrenadas con una códificación de Hadamard consistieron en códigos con valores de $[0, 1]$ de una longitud 10 y 16, esto debido a que al construir los códigos de Hadamard se especifica un tamaño correspondiente a 2^k. Sin embargo, MNIST cuenta con 10 clases, por lo que se decidió entrenar un modelo con el tamaño de códigos originales (16) y otro con el tamaño de códigos de acuerdo al número de clases (10).

Por último, los ataques adversarios utilizados en este experimento fueron los ataques de caja blanca: Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM) y Projected Gradient Descent Attack (PGD), debido a que son los más mencionados en la literatura relacionada. Para cada imagen en el conjunto de validación se generó un ejemplo adversario utilizando uno de los ataques anteriores con la intención de afectar solamente al modelo encargado de la evaluación. Los resultados correspondientes a este primer experimento se muestran en la Figura 22.

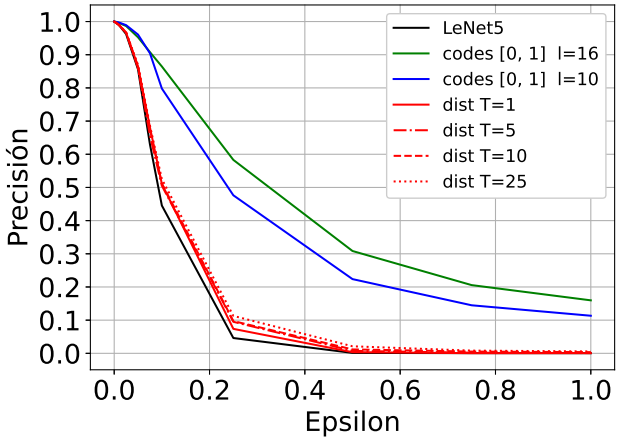
En los resultados mostrados en la figura anterior, es importante observar que todos los modelos tienen un porcentaje de clasificación similar cuando no se presentan ataques ($\epsilon = 0$) y este porcentaje disminuye a medida que el valor de ϵ aumenta. Sin embargo, los modelos que utilizan una codificación de Hadamard presentan una mayor precisión de clasificación en cada ataque generado, a pesar de aumentar el valor de ϵ. En algunos casos, se obtiene una precisión superior al $\sim 40\,\%$ en comparación al modelo base y modelos destilados. Además, es posible observar que cuando se entrena con códigos de Hamming en su longitud original, se presenta una mayor tolerancia ante ataques adversarios, en comparación al caso en que los códigos son recortados. Esto se debe a que al recortar los códigos de Hamming, disminuye la distancia entre cualquier par de códigos facilitando el trabajo de un ataque adversario para cambiar la clase de una imagen.



(a) FGSM



(b) BIM



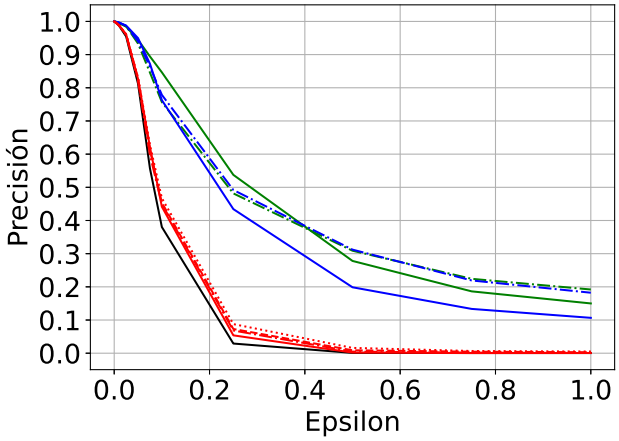
(c) PGD

Figura 22. Precisión Top-1 en el conjunto de validación del conjunto de datos MNIST utilizando el modelo de clasificación LeNet5 bajo tres ataques diferentes: (a) Fast Gradient Sign Method (FGSM), (b) Basic Iterative Method (BIM) y (c) Projected Gradient Descent Attack (PGD). Se muestran los resultados de un modelo de referencia (negro), modelos utilizando destilación (rojo) y modelos utilizando códigos de Hadamard (verde y azul). En el eje x se indica el valor de épsilon, relacionado con la sensibilidad de un observador humano para detectar el ataque.

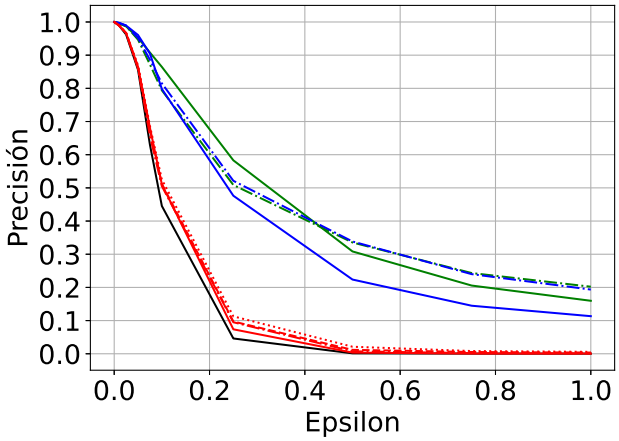
En el capítulo anterior se mencionó que la salida de un modelo es un vector de números reales. Por tal motivo, en los modelos que utilizan la codificación de Hadamard se emplea MSE para realizar el cálculo distancia entre la salida y los códigos de clase correspondientes. Ahora bien, al observar que se tiene una mayor tolerancia a ataques al aumentar la longitud del vector de salida, se decidió entrenar un par de modelos incrementando la distancia intrínseca de los códigos de Hadamard utilizando valores de $[-5, 5]$ en lugar de $[0, 1]$ y de igual manera que en el primer experimento, esta modificación se probó con códigos de longitud 10 y 16. Los resultados obtenidos se pueden observar en la Figura 23.

En los resultados anteriores, parece suficiente utilizar códigos en su longitud original, dado que este modelo presenta una mayor precisión cuando el valor de $\epsilon \leq 0.4$. Sin embargo, es posible notar que el uso de códigos de Hadamard con valores de $[-5, 5]$ da como resultado una mayor robustez ante ataques adversarios en comparación con los códigos con valores $[0, 1]$ a partir de un valor de $\epsilon \geq 0.5$. Note que para este conjunto de datos es mejor extender el rango de los códigos, así como su longitud.

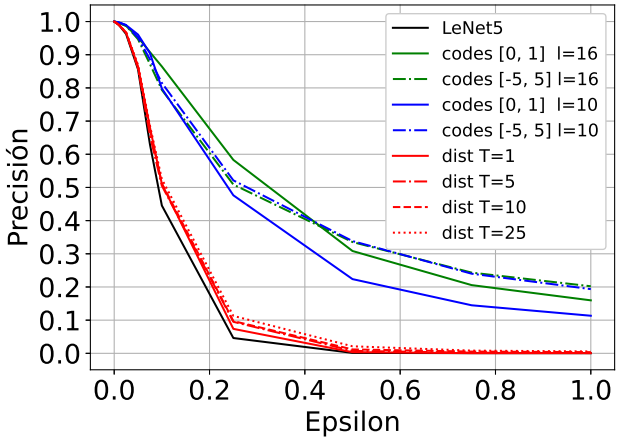
En la Figura 24, se muestra la precisión para el conjunto de entrenamiento y validación en cada época de entrenamiento. Es posible observar que el modelo base y el modelo entrenado con una codificación de Hadamard, consiguen una precisión muy parecida después de un determinado número de épocas, mostrando experimentalmente, que ambos modelos convergen a un ritmo similar, es decir, para el conjunto de imágenes MNIST, no se requiere ciclos de entrenamiento adicionales cuando se realiza el cambio de codificación.



(a) FGSM



(b) BIM



(c) PGD

Figura 23. Precisión Top-1 en el conjunto de validación del conjunto de datos MNIST utilizando el modelo de clasificación LeNet5 bajo tres ataques diferentes: (a) Fast Gradient Sign Method (FGSM), (b) Basic Iterative Method (BIM)) y (c) Projected Gradient Descent Attack (PGD). Se muestran los resultados de un modelo de referencia (negro), modelos utilizando destilación (rojo) y modelos utilizando códigos de Hadamard (verde y azul), con una línea continua los modelos que utilizaron códigos [0, 1] y con línea punteada los modelos que utilizaron códigos [-5, 5]. En el eje x se indica el valor de épsilon, relacionado con la sensibilidad de un observador humano para detectar el ataque.

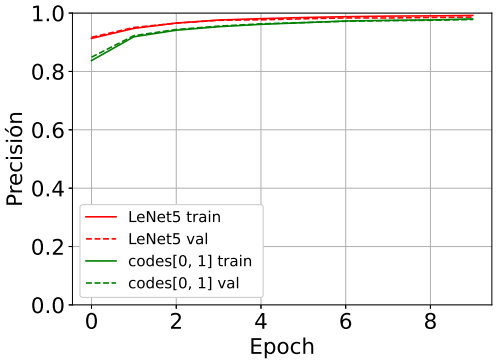
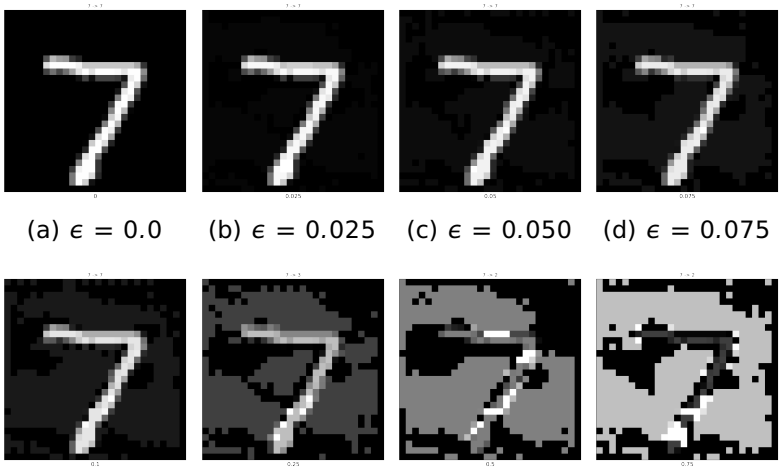


Figura 24. Muestra la precisión durante cada época de entrenamiento para el conjunto de datos MNIST usando LeNet5. Los resultados del modelo de base utilizando codificación one-hot se presenta en color rojo y los resultados del modelo utilizando códigos de Hadamard en color verde.

Aunque el conjunto de imágenes MNIST fue conveniente para comprobar rápidamente la hipótesis de este trabajo, presenta un inconveniente. Dado que el tamaño de las imágenes es muy pequeño (28 x 28 píxeles), no es necesario incrementar en gran medida el tamaño de la perturbación ϵ para que un modelo falle en sus predicciones, esto se puede observar fácilmente en las Figuras 22 y 23. En la Figura 25 se muestran ejemplos de una imagen y sus respectivos ejemplos adversarios, observe que es posible identificar rápidamente los píxeles que fueron modificados, lo cual quebranta la definición de un ejemplo adversario. Por tal motivo, se decidió realizar una segunda serie de experimentos utilizando un conjunto de datos que no presente este problema.



(a) $\epsilon = 0.0$ (b) $\epsilon = 0.025$ (c) $\epsilon = 0.050$ (d) $\epsilon = 0.075$

(e) $\epsilon = 0.10$ (f) $\epsilon = 0.25$ (g) $\epsilon = 0.50$ (h) $\epsilon = 0.75$

Figura 25. Muestra una imagen del conjunto de validación de MNIST sin ataque (a) y esta misma como un ejemplo adversario generada con el ataque FGSM variando la intensidad de perturbación ϵ (b-h).

5.2. Experimentos Mini-Imagenet

Con el objetivo de estudiar un escenario más desafiante para la codificación propuesta, en la segunda serie de experimentos se utilizó el conjunto de imágenes Mini-Imagenet (Vinyals *et al.*, 2016), el cual cuenta con 82,000 imágenes de entrenamiento y 3,200 imágenes de validación, todas ellas dividas entre 64 clases. Con la intención de que la perturbación de un ataque sea lo menos notoria posible, se decidió utilizar cada imagen en una resolución de 224x224 píxeles, a diferencia de la configuración tradicional de Mini-Imagenet, donde cada imagen tiene un tamaño de 82x82 píxeles.

Para estos experimentos, se utilizaron los modelos preentrenados ResNet18, ResNet50 y ResNet101, los cuales fueron especializados en Mini-Imagenet, es decir, solamente se entrenaron las últimas capas correspondientes al clasificador del modelo, esto con la intención de ahorrar tiempo de entrenamiento. Para cada experimento presentado, un modelo ResNet es considerado el modelo base, es decir, no emplea ninguna estrategia defensiva y es utilizado como referencia para medir el impacto de un ataque. Cada modelo base tiene variantes destiladas con temperatura T igual a 1, 5, 10 y 25. Además, se consideran dos variante del modelo base utilizando codificación de Hadamard, en la que cada código tiene una longitud 64 con valores de $[0, 1]$ y $[-25, 25]$. Estos modelos fueron evaluados ante seis ataques de caja blanca y un ataque de caja negra.

5.2.1. Ataques caja blanca

Los ataques de caja blanca utilizados en esta sección fueron: Basic Iterative Method (BIM), Decoupled Direction and Norm Attack (DDN), Fast Gradient Sign Method (FGSM), Projected Gradient Descent Attack (PGD), Virtual Adversarial Attack (VA) y Carlini y Wagner Attack (C&W). Para cada ataque se generó un ejemplo adversario a partir de las imágenes de validación utilizando la librería Foolbox, con el valores de intensidad de perturbación ϵ entre $[0, 1]$. Al igual que los experimentos anteriores, cada ejemplo adversario fue generado con la intención de atacar al modelo que realiza la evaluación.

Los resultados de la precisión Top-1 en el conjunto de validación de Mini-Imagenet se pueden observar en la Figura 26 para el modelo ResNet18, la Figura 27 para el

modelo ResNet50 y la Figura 28 para el modelo ResNet101. Algo importante que se debe tener en cuenta, es que para los experimentos de esta sección los modelos destilados muestran una menor precisión inicial en comparación al resto de los modelos; este comportamiento está relacionado con una especialización del modelo durante el primer ciclo de entrenamiento, lo cual se analiza a detalle en (Cho y Hariharan, 2019).

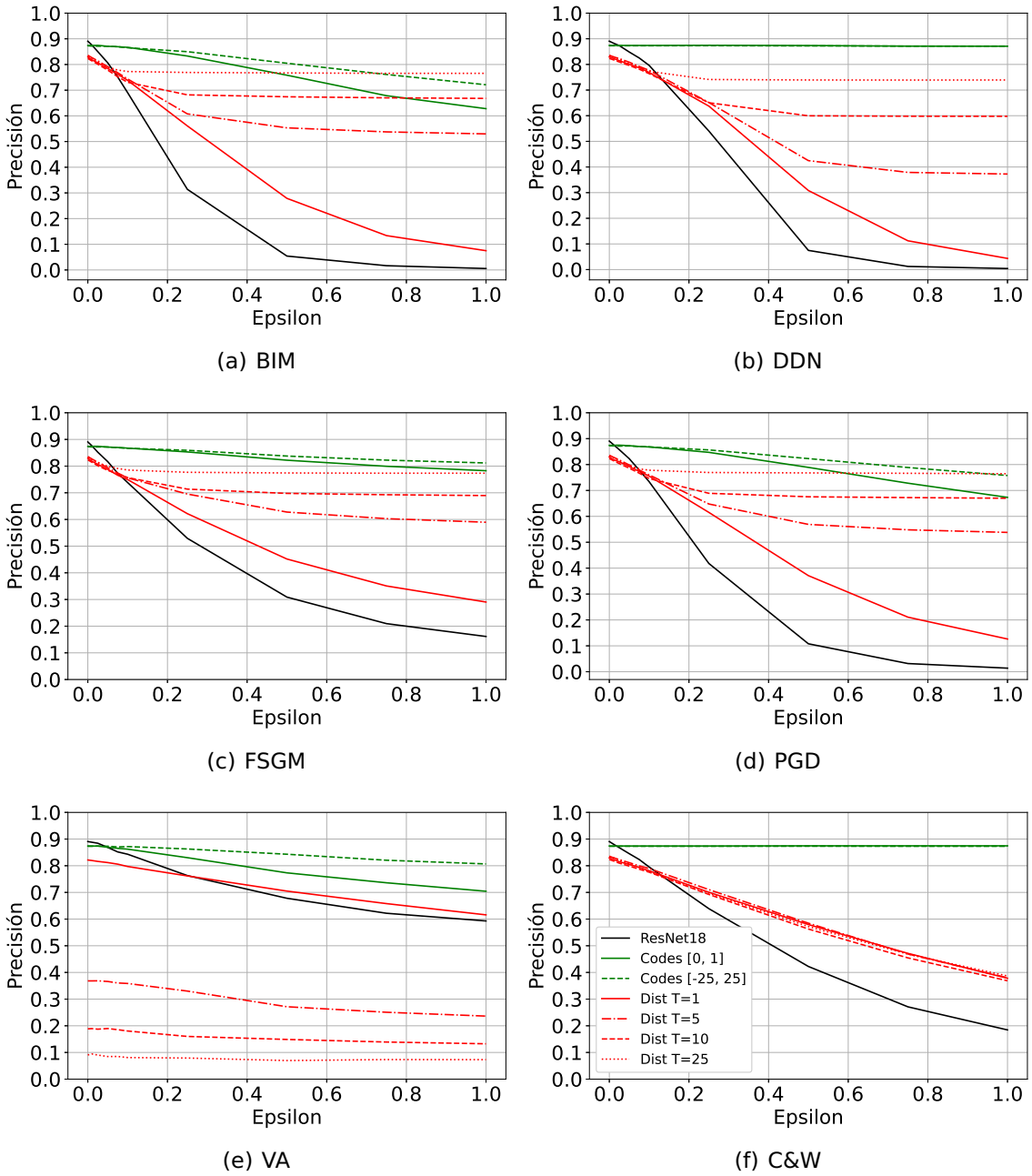


Figura 26. Precisión Top-1 en el conjunto de validación del conjunto de datos Mini-Imagenet utilizando el modelo de clasificación ResNet18 bajo seis ataques diferentes: (a) Basic Iterative Method (BIM), (b) Decoupled Direction and Norm Attack (DDN), (c) Fast Gradient Sign Method (FGSM), (d) Projected Gradient Descent Attack (PGD), (e) Virtual Adversarial Attack y (f) Carlini y Wagner Attack (C&W). Se muestran los resultados de clasificación del modelo de referencia (negro), modelos utilizando destilación (rojo) y modelos utilizando códigos de Hadamard (verde). En el eje x se indica el valor de épsilon, relacionado con la sensibilidad de un observador humano para detectar el ataque.

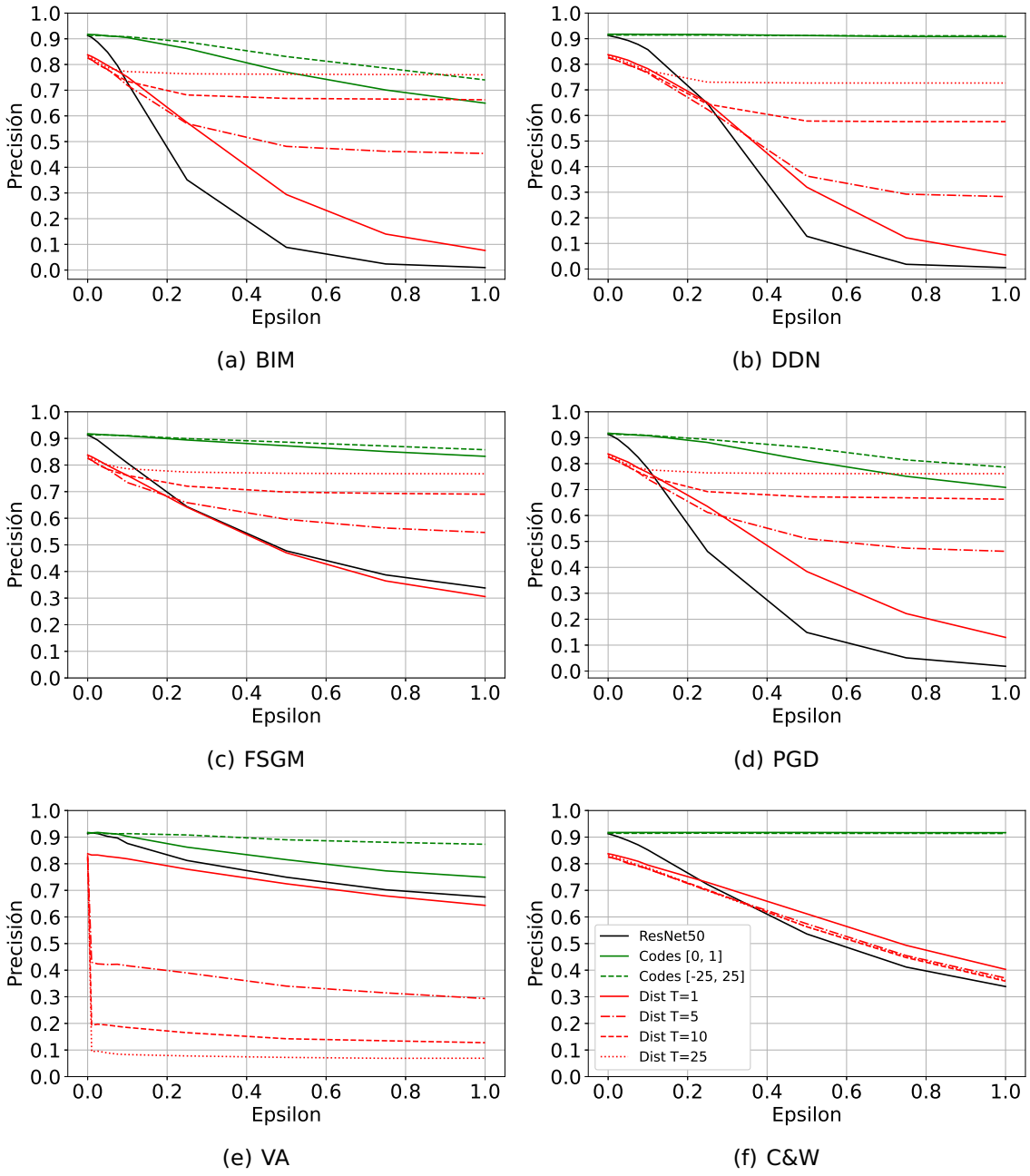


Figura 27. Precisión Top-1 en el conjunto de validación del conjunto de datos Mini-Imagenet utilizando el modelo de clasificación ResNet50 bajo seis ataques diferentes: (a) Basic Iterative Method (BIM), (b) Decoupled Direction and Norm Attack (DDN), (c) Fast Gradient Sign Method (FGSM), (d) Projected Gradient Descent Attack (PGD), (e) Virtual Adversarial Attack y (f) Carlini y Wagner Attack (C&W). Se muestran los resultados de clasificación del modelo de referencia (negro), modelos utilizando destilación (rojo) y modelos utilizando códigos de Hadamard (verde). En el eje x se indica el valor de épsilon, relacionado con la sensibilidad de un observador humano para detectar el ataque.

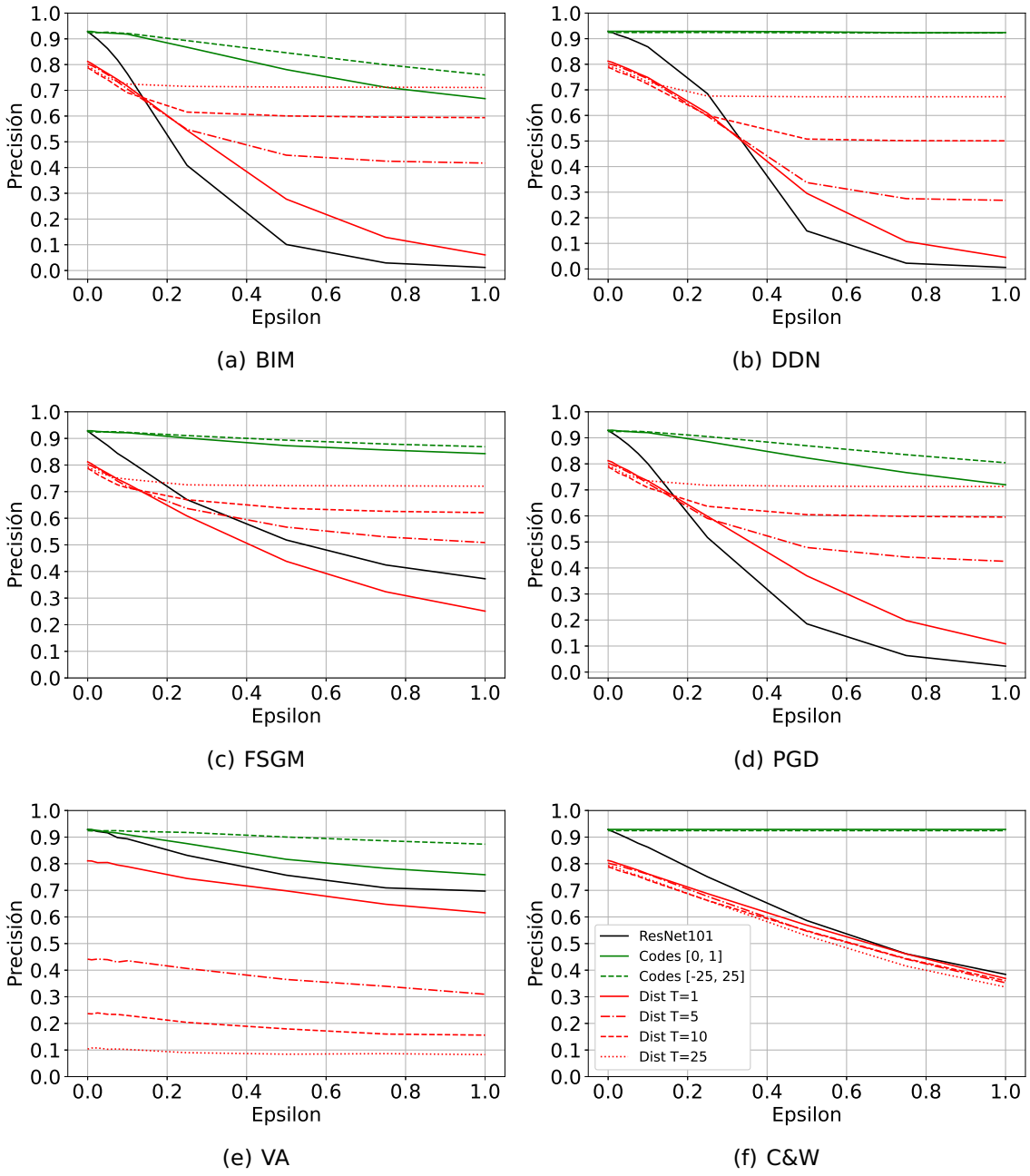


Figura 28. Precisión Top-1 en el conjunto de validación del conjunto de datos Mini-Imagenet utilizando el modelo de clasificación ResNet101 bajo seis ataques diferentes: (a) Basic Iterative Method (BIM), (b) Decoupled Direction and Norm Attack (DDN), (c) Fast Gradient Sign Method (FGSM), (d) Projected Gradient Descent Attack (PGD), (e) Virtual Adversarial Attack y (f) Carlini y Wagner Attack (C&W). Se muestran los resultados de clasificación del modelo de referencia (negro), modelos utilizando destilación (rojo) y modelos utilizando códigos de Hadamard (verde). En el eje x se indica el valor de épsilon, relacionado con la sensibilidad de un observador humano para detectar el ataque.

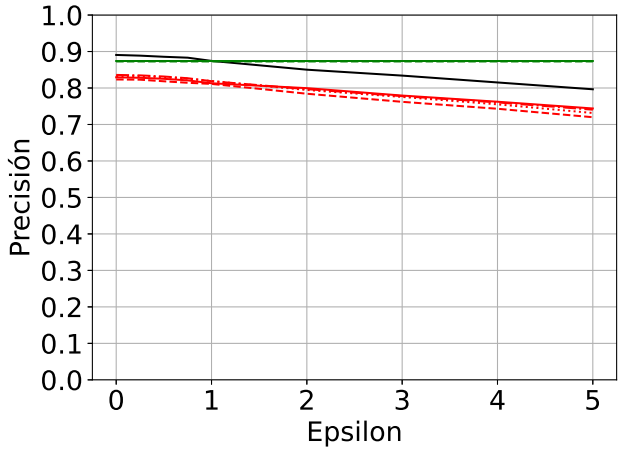
En cada una de las figuras anteriores, se puede observar que en los ataques DDN, FGSM, VA y C&W, los modelos entrenados con códigos Hadamard con valores de $[0, 1]$ y $[-25, 25]$ presentan una precisión top-1 superior en comparación con el modelo base y los modelos destilados para cada valor de ϵ utilizado.

En el caso del ataque BIM, los modelos ResNet18 y ResNet50 entrenados con códigos de Hadamard $[-25, 25]$ presenta una precisión mayor a cualquier otro modelo cuando los valores de $\epsilon < 0.5$. Cuando el valor de épsilon sobrepasa este umbral, el modelo propuesto es superado en precisión por los modelos destilados con $T = 10, 25$. Sin embargo, el modelo ResNet101 entrenado con códigos de Hadamard con valores de $[-25, 25]$ presenta una precisión de clasificación superior al modelo base y a cualquier modelo destilado para todos los valores de ϵ, esto se adjudica a la complejidad del modelo ya que cuenta con un mayor número de capas.

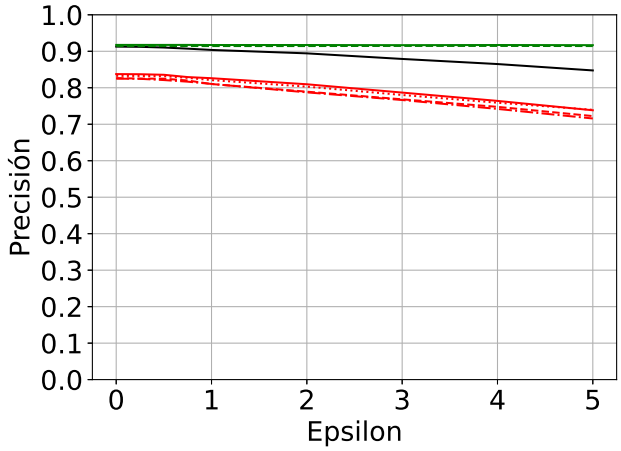
En el ataque PGD, ocurre algo similar al ataque previo. En el modelos ResNet18, al utilizar la codificación de Hadamard $[-25, 25]$ se tiene una precisión mayor para los valores de $\epsilon \leq 0.8$ y sobrepasando ese umbral se obtiene una precisión muy similar a la del modelo destilado con $T = 25$. En el caso de los modelos ResNet50 y ResNet101 entrenados con una códificación de Hadamard $[-25, 25]$, se obtiene una precisión superior en comparación con el modelo base y las variantes destiladas en todos los valores de épsilon utilizados.

5.2.2. Ataques caja negra

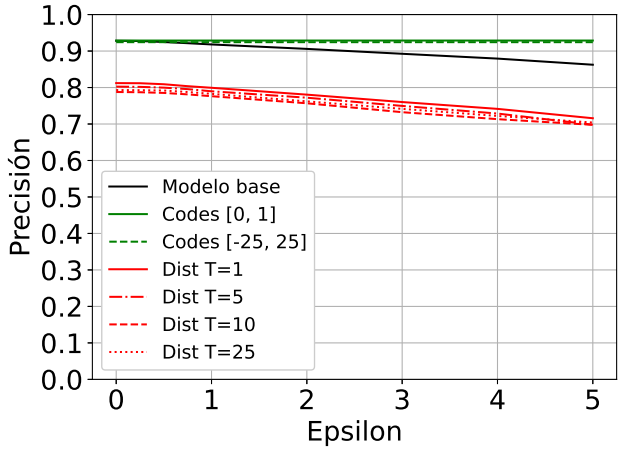
Los resultados correspondientes a la precisión en el conjunto de validación de Mini-Imagenet utilizando el ataque de caja negra llamado Salt and Pepper Noise Attack (S&P) se pueden observar en la Figura 29. Los ejemplos adversarios generados con este ataque fueron creados con valores de épsilon dentro del intervalo $[0, 5]$. Al igual que los experimentos anteriores, se utilizaron los modelos base ResNet18, ResNet50 y ResNet101. Para cada modelo base se entrenó una variante destilada con temperatura T igual a 1, 5, 10 y 25. Además, se consideran dos variantes del modelo base utilizando codificación de Hadamard con valores de $[0, 1]$ y $[-25, 25]$.



(a) ResNet18



(b) ResNet50



(c) ResNet101

Figura 29. Precisión Top-1 en el conjunto de validación del conjunto de datos Mini-Imagenet utilizando el modelo de clasificación ResNet18 (a), ResNet50 (b) y ResNet101 (c), bajo el ataque de caja negra Salt & Pepper (S&P). Se muestran los resultados de un modelo de referencia (negro), modelos utilizando destilación (rojo) y modelos utilizando códigos de Hadamard (verde). En el eje x se indica el valor de épsilon, relacionado con la sensibilidad de un observador humano para detectar el ataque.

En la figura anterior es posible observar una caída en la precisión de la clasificación tanto del modelo base como de los modelos destilados, por lo que es posible inferir que, independientemente del valor de temperatura T de los modelos destilados, no producen resistencia alguna ante este ataque de caja negra. Por otro lado, los modelos entrenados con una codificación de Hadamard no presentan disminución de la precisión para ningún valor épsilon que fue utilizado. Por último, para este ataque no se presenta diferencia cuantitativa en la precisión obtenida con los modelos que utilizan códigos de Hadamard con valores de $[0, 1]$ ó $[-25, 25]$.

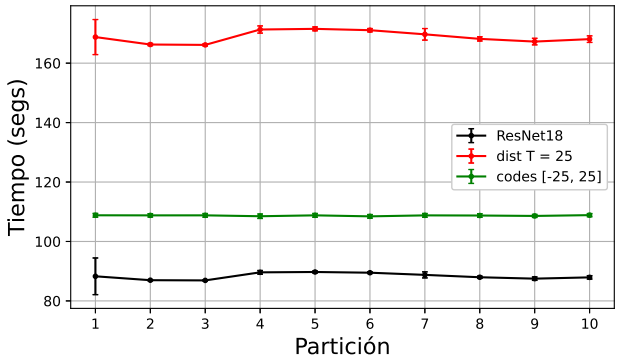
5.3. Análisis estadístico del proceso de entrenamiento

En las secciones anteriores se mostró experimentalmente la eficacia de la codificación de Hadamard como sistema defensivo ante distintos ataques adversarios. En esta sección se caracterizan las ventajas que presenta el modelo propuesto en cuanto al tiempo de entrenamiento requerido y la precisión de la clasificación. Para ello, se realizaron una serie de experimentos de acuerdo con la siguiente metodología. El conjunto de entrenamiento se dividió en 10 particiones y de manera alternada se tomaron 9 particiones como conjunto de entrenamiento y la partición restante fue utilizada como conjunto de validación. Debido a que la intención de estos experimentos es caracterizar su eficiencia durante el proceso de entrenamiento, no hizo uso de algún ataque adversario.

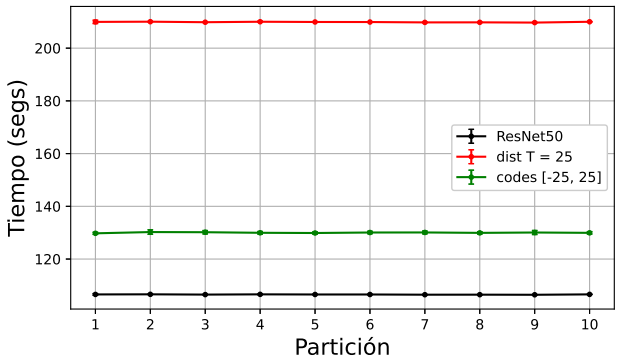
Los modelos base que se consideraron para los experimentos de esta sección fueron ResNet18, RestNet50 y ResNet101. Para cada modelo base se entrenó una variante destilada con temperatura $T = 25$ y otra con una codificación de Hadamard con valores de $[-25, 25]$, ya que estás fueron las que presentaron mejor rendimiento en la sección anterior.

En la Figura 30, se pueden observar los resultados correspondientes al tiempo total promedio en segundos de cada partición. Este tiempo corresponde a 25 épocas de entrenamiento y validación para cada partición. Para cada gráfica es posible percatarse que el modelo propuesto requiere un mayor tiempo de entrenamiento en comparación con el modelo base. Esto se debe a que la validación por parte del modelo propuesto, en la cual se calcula las distancias entre una predicción y los códigos utilizados, es

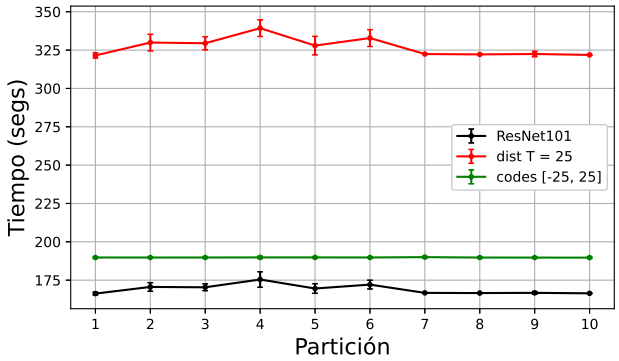
más costosa que el proceso de validación tradicional. Sin embargo, un modelo destilado necesita 1.8 veces el tiempo requerido por el modelo base, mientras que el modelo propuesto únicamente requiere 1.2 veces el tiempo que le toma al modelo base, esto se debe a que cada partición requiere realizar un procesamiento de entrenamiento "doble"; un ciclo de entrenamiento para obtener las etiquetas de clasificación suavizadas y un ciclo adicional para especializar el modelo utilizado.



(a) ResNet18



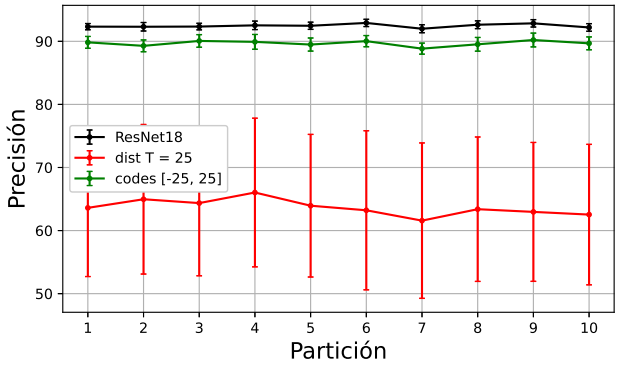
(b) ResNet50



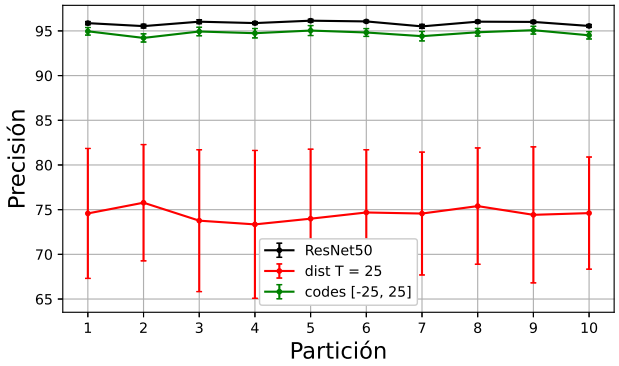
(c) ResNet101

Figura 30. Promedio del tiempo de entrenamiento de subconjuntos de Mini-Imagenet utilizando el modelo de clasificación ResNet18 (a), ResNet50 (b) y ResNet101 (c). En el eje x se indica el número de la partición, mientras que en el eje y se muestra el tiempo promedio en segundos.

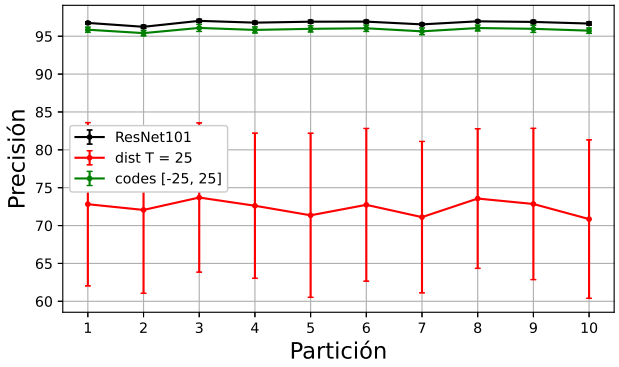
Otra ventaja que presenta el modelo propuesto se puede observar en la Figura 31, la precisión Top-1 promedio obtenida en cada partición de validación. Observe que la diferencia en la precisión del modelo base y del modelo propuesto es de $\sim 3\,\%$. Mientras que los modelos destilados presentan una diferencia promedio de $\sim 24\,\%$ con relación al modelo base y una desviación estándar promedio de $\sim 9.65\,\%$ para cada modelo ResNet utilizado, esto debido al reentrenamiento de destilado. La disminución en la precisión por parte del modelo destilado se asocia al número de épocas de entrenamiento, ya que el valor más bajo fue obtenido al inicio del entrenamiento mientras que el valor más alto se obtuvo en las épocas finales de entrenamiento.



(a) ResNet18



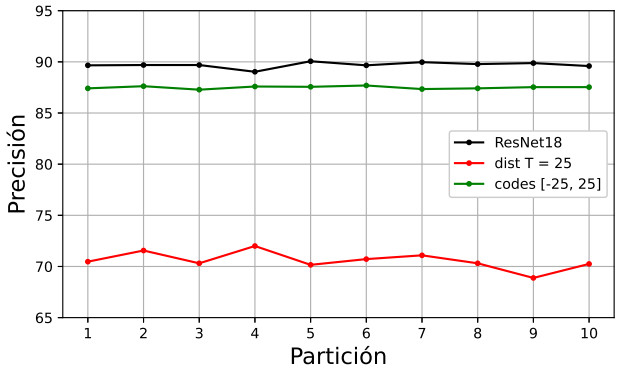
(b) ResNet50



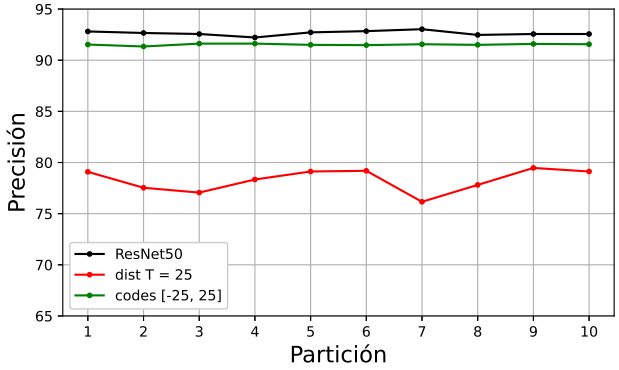
(c) ResNet101

Figura 31. Precisión Top-1 de subconjuntos de validación de Mini-Imagenet utilizando el modelo de clasificación ResNet18 (a), ResNet50 (b) y ResNet101 (c). En el eje x se indica el número de la partición.

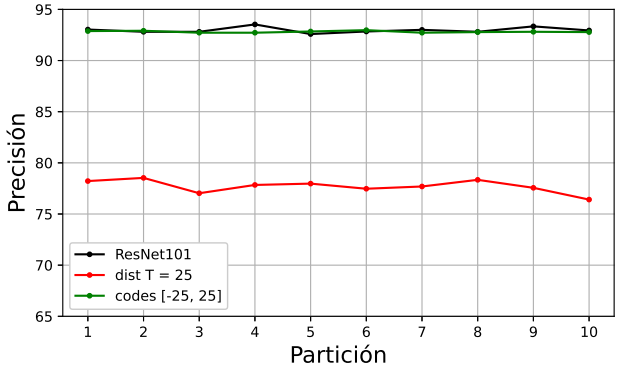
En la Figura 32 se puede observar la precisión Top-1 obtenida en cada partición para el conjunto de prueba, es decir, el conjunto original de validación de Mini-Imagenet. La mayor diferencia de precisión entre el modelo base y el modelo propuesto es de $\sim 3\,\%$. Por parte de los modelos destilados se tiene una diferencia promedio del $\sim 16\,\%$ en comparación con los modelos base.



(a) ResNet18



(b) ResNet50



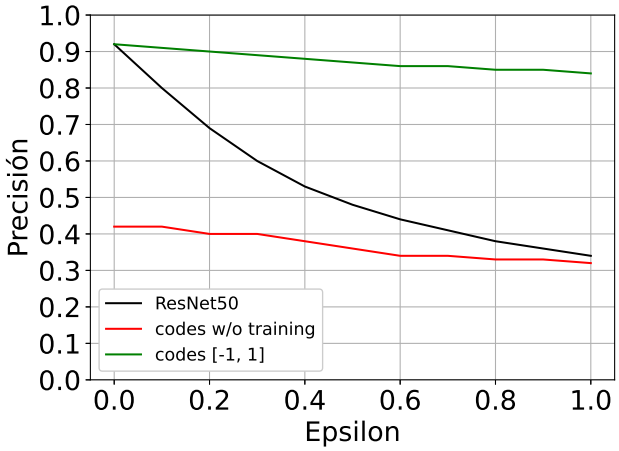
(c) ResNet101

Figura 32. Precisión Top-1 del conjunto de prueba de Mini-Imagenet utilizando el modelo de clasificación ResNet18 (a), ResNet50 (b) y ResNet101 (c). En el eje x se indica el número de la partición.

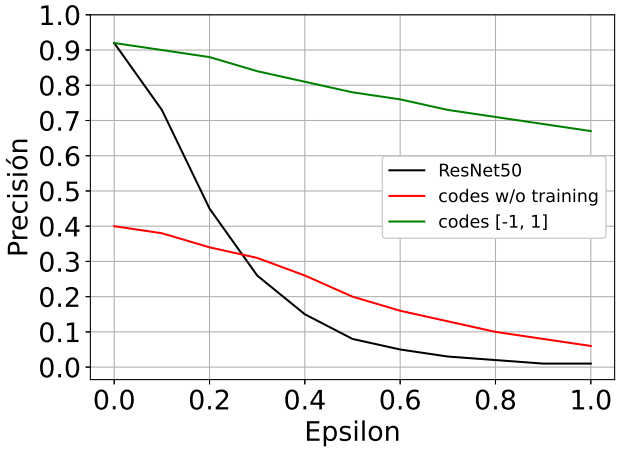
Por último, es posible cuestionar que un reentrenamiento con un cambio de codificación no es necesario, esto debido a que un modelo está aprendiendo, aparentemente, una multiplicación de matrices entre los códigos one-hot y una matriz de Hadamard.

Con el propósito de comprobar lo anterior, se realizó el siguiente experimento, para un modelo clasificador base ResNet50 f y una matriz de Hadamard H, tal que el i-ésimo renglón representa la codificación para la i-ésima clase. La salida del modelo base para una imagen entrada x está dada por $z = f(x)$ y una predicción y' corresponde a $y' = zH$, es decir, se está obligando que y' corresponda a una codificación de Hadamard, sin necesidad de entrenarlo directamente, mientras que z se está decodificando como una salida one-hot. Siguiendo este enfoque, se evaluó la robustez de este modelo ante los ataques adversarios de caja blanca FGSM, BIM y PGD, en comparación con un modelo base y un modelo entrenado con la codificación de Hadamard, para el conjunto de prueba del conjunto de imágenes Mini-Imagenet.

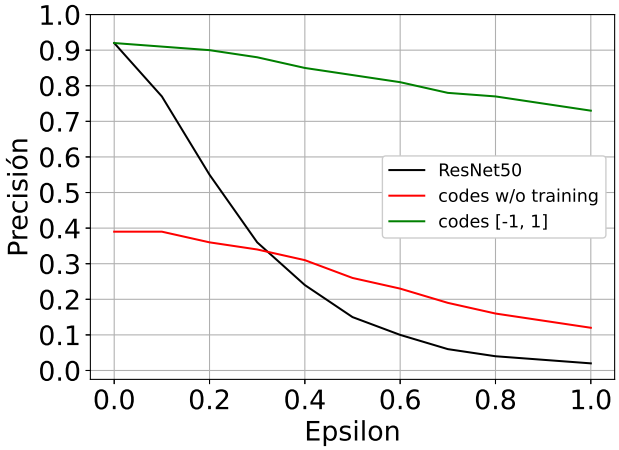
Los resultados obtenidos al realizar el experimento anterior se pueden observar en la Figura 33. Se muestra la precisión top-1 en el conjunto de prueba, donde la línea de color negro representa el modelo base ResNet50, la línea de color rojo representa la codificación obtenida sin entrenamiento, es decir, $y' = (1/n)zH$ y la línea de color verde representa el modelo base entrenado con la codificación de Hadamard. Mientras que en el eje x se muestra la intensidad ϵ utilizada para cada ataque, los cuales son considerados muy básicos en la literatura, por lo que un buen sistema de defensa adversaria debería ser capaz de presentar buena robustez ante ellos. Sin embargo, es posible observar que aunque la codificación obtenida sin entrenamiento presenta cierta resistencia ante ataques adversarios, el porcentaje de clasificación se encuentra por debajo del modelo base, en la mayoría de los casos y por debajo del 50 % de clasificación. Por tal motivo, se justifica la necesidad del reentrenamiento de un modelo base con una codificación de Hadamard como metodología defensiva ante ataques adversarios.



(a) FGSM



(b) BIM



(c) PGD

Figura 33. Precisión Top-1 en el conjunto de validación del conjunto de datos Mini-Imagenet utilizando el modelo de clasificación ResNet50 ante tres ataques diferentes: (a) Fast Gradient Sign Method (FGSM), (b) Basic Iterative Method (BIM)) y (c) Projected Gradient Descent Attack (PGD). Se muestran los resultados de un modelo de referencia (negro), utilizando una multiplicación de matrices (rojo) y utilizando códigos de Hadamard (verde). En el eje x se indica el valor de épsilon, relacionado con la sensibilidad de un observador humano para detectar el ataque.

Capítulo 6. Conclusiones y trabajo futuro

6.1. Discusión

En esta tesis, se ataca el problema de clasificación de ejemplos adversarios para los conjuntos de imágenes MNIST y Mini-Imagenet. Para lograrlo, se propone utilizar una codificación basada en códigos de corrección de errores lineales, específicamente, se abandona la codificación tradicional one-hot y se utilizan códigos de Hadamard, el motivo de este cambio de codificación es sencillo, la distancia de Hamming entre cualquier par de códigos one-hot es dos, mientras que la códificación de Hadamard presentan una distancia de Hamming de $n/2$ entre cualquier par de códigos, siendo n el tamaño del código.

Se diseñaron una serie de experimentos en los cuales se estudió la robustez de clasificación por parte del modelo LeNet5 para MNIST ante tres ataques de caja blanca (FGSM, BIM, PGD) y los modelos ResNet18, ResNet50 y ResNet101 para Mini-Imagenet ante seis ataques de caja blanca (FGSM, BIM, PGD, DDN, VA, C&W) y un ataque de caja negra (S&P). Los modelos utilizados para clasificar cada conjunto de imágenes se evaluaron utilizando una versión base, sin una estrategia defensiva, variantes destiladas, consideradas la mejor defensa adversaria y variantes utilizando la codificación propuesta. Finalmente, se realizó un análisis estadístico del tiempo de entrenamiento y precisión de la clasificación de estos modelos, sin la presencia de ejemplos adversarios, esto permitió caracterizar cuantitativamente las ventajas de la metodología propuesta.

Adicionalmente, los resultados experimentales de esta tesis permiten considerar que independientemente del conjunto de imágenes, la codificación de Hadamard presenta mejores resultados de clasificación ante ejemplos adversarios en comparación a un modelo base o un modelo destilado, lo que permite inferir que este comportamiento se mantendrá con un conjunto de imágenes distinto. Así mismo, debido a que los ataques adversarios comparten ciertas estrategias para afectar un modelo, por ejemplo los ataques BIM y PGD son variantes de FSGM, es posible especular que utilizar los códigos de Hadamard en la codificación de clases, puede resultar un excelente recurso defensivo ante otros ataques adversarios que no fueron utilizados en esta tesis.

6.2. Conclusiones

El trabajo realizado en esta tesis permite concluir lo siguiente:

- Utilizar códigos de corrección de errores, en particular los códigos de Hadamard, para el etiquetado de clases hace más robusto los modelos de redes neuronales ante múltiples ataques adversarios de caja blanca y caja negra, realizando solamente un ciclo de entrenamiento.
- Para los ataques de caja blanca FGSM, BIM, PGD, DDN, VA y C&W, se observó una disminución en la precisión de clasificación del $\sim 90\,\%$ en un modelo base ResNet a medida que se aumentaba la intensidad del ataque ϵ, mientras que para modelo propuesto con códigos de Hadamard con valores de $[0, 1]$ la peor caída fue de $\sim 23\,\%$ y la variante de codificación de Hadamard con valores de $[-25, 25]$ presentó su peor disminución de $\sim 11\,\%$. Por su parte, aunque el mejor modelo destilado $T = 25$ sólo presentó una caída del $\sim 8\,\%$ en la mayoría de estos ataques, el porcentaje de clasificación es menor al presentado por el modelo propuesto.
- Para el ataque de caja negra S&P, se observó una disminución en la precisión de clasificación de $\sim 5\,\%$ en los modelos base ResNet y hasta $\sim 9\,\%$ en los modelos destilados; mientras que en los modelos propuestos utilizando una códificación de Hadamard con valores de $[0, 1]$ y $[-25, 25]$ la caída en la precisión de clasificación fue de $\sim 1\,\%$ en el peor caso.
- En el caso particular de los ataques de caja blanca FSGM, BIM, PGD y VA, se observó una robustez promedio mayor de al menos $2\,\%$ al utilizar códigos de Hadamard con valores de $[-25, 25]$ en comparación con utilizar códigos con valores de $[0, 1]$ en los modelos base ResNet.
- Específicamente, al comparar el rendimiento de la defensa adversaria de destilado con el enfoque propuesto, este último presenta dos ventajas: se requiere menor tiempo de entrenamiento cuando se compara con un ciclo de entrenamiento tradicional y también presenta un mejor porcentaje de clasificación en imágenes adversarias y sin presencia de ataques.

6.3. Trabajo futuro

Los posibles temas a estudiar a partir de este trabajo son:

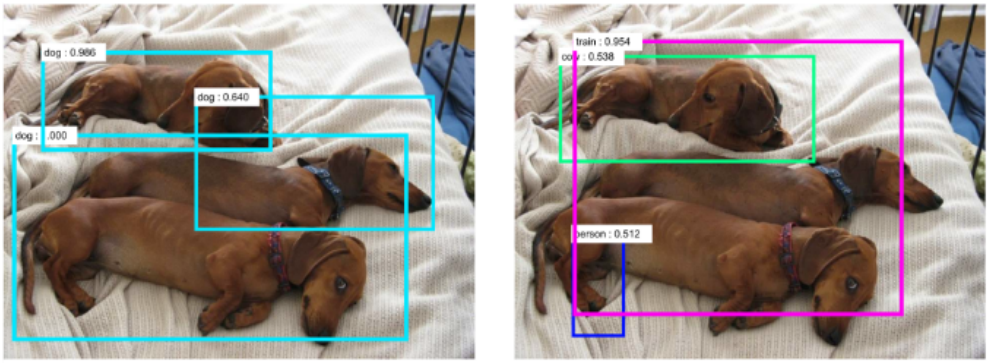
- En los experimentos realizados, los códigos de Hadamard mostraron ser buena estrategia contra los ejemplos adversarios ante múltiples ataques. Sin embargo, es de interés explorar el desempeño de otros códigos de corrección de errores lineales como los códigos de Hamming (Hamming, 1950) o los códigos Reed-Solomon (Reed, 1953), por mencionar algunos. Estudiar estos u otros códigos podrían mostrar algunas ventajas en comparación a los códigos de Hadamard, ya sea una mayor robustez ante algún ataque en específico, superar los resultados obtenidos en esta tesis o consolidar los resultados obtenidos al mostrar que los códigos de Hadamard son la mejor opción para el cambio de codificación entre múltiples códigos de corrección de errores lineales.

- Adicionalmente, el uso de códigos de corrección de errores lineales pueden ser de utilidad en otros problemas. El problema conocido como Learning to Hash (Wang *et al.*, 2017), busca crear una representación binaria de las capas intermedias de una DNN, conocidas como deep features. Al utilizar los códigos de corrección de errores lineales, ya sean los códigos de Hadamard u otros, se obtiene una representación binaria de clases que puede ser utilizada en el problema de navegación multimedia (Ahmad y Conci, 2019). En (Quiroz, 2021) se mostró la viabilidad de utilizar los códigos de Hadamard en este escenario con resultados satisfactorios en los conjuntos de imágenes ImageNet (Deng *et al.*, 2009) y Places365 (Zhou *et al.*, 2017). De igual manera, es posible realizar un estudio más amplio al considerar distintas codificaciones o conjunto de imágenes.

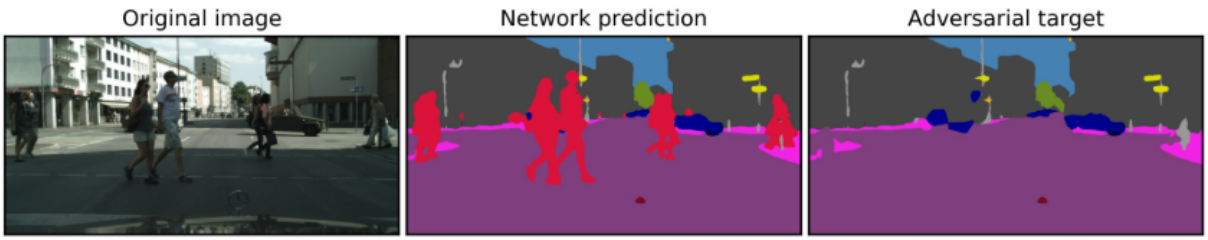
- Una extensión natural de este trabajo de tesis podría ser diseñar un ataque adversario, ya sea de caja blanca o caja negra, especializado en atacar modelos entrenados en una codificación de Hadamard. Como se mencionó en la Sección 2.2, así como existen estrategias defensivas especializadas para contrarrestar algún ataque en particular, es posible desarrollar un ataque con el único objetivo de afectar una defensa en concreto y al ser la única defensa adversaria que utiliza este cambio de codificación, hasta el momento, se podría diseñar un ataque

adversario que considere la distancia de Hamming en la codificación de clasificación, dado que está es la principal ventaja de la metodología propuesta.

- De igual manera, los ejemplos adversarios no se limitan al problema de clasificación de imágenes, existen ataques adversarios especializados en los problemas de segmentación de imágenes (Hendrik Metzen *et al.*, 2017) y detección de objetos (Xie *et al.*, 2017), algunos ejemplos adversarios de estos ataques se pueden observar en la Figura 34. Cabe mencionar que, de manera general, las soluciones a estos problemas están basadas en la clasificación de una parte de una imagen o cada pixel, respectivamente, donde cada uno de estos es representado utilizando la codificación one-hot. Por tal motivo, otra extensión del trabajo propuesto en esta tesis puede ser llevarlo a estos problemas.



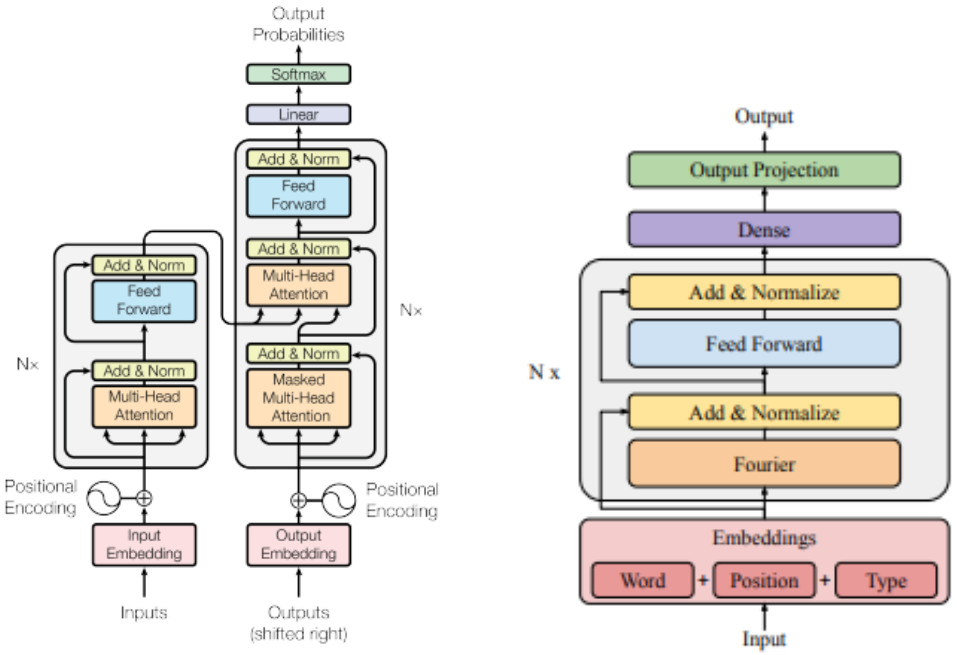
(a)



(b)

Figura 34. (a) Ejemplo de una imagen adversaria en el problema de detección de objetos. (b) Ejemplo de una imagen adversaria en el problema de detección de objetos. Fuente: (a) (Hendrik Metzenet al., 2017), (b) (Xieet al., 2017)

- En los últimos años, la arquitectura del modelo transformadores (Vaswani *et al.*, 2017), mostrada en la Figura 35 (a), ha potenciado los avances en el área de procesamiento de lenguaje natural con su principio “atención es todo lo que necesitas”. Sin embargo, recientemente el trabajo FNet (Lee-Thorp *et al.*, 2021), demostraron que las los transformadores se pueden acelerar, con costos de precisión limitados, reemplazando las subcapas de atención con transformaciones lineales simples, particularmente con la transformada de Fourier, como se muestra en la Figura 35 (b). Ahora bien, los transformadores han sido probados con exito recientemente en el área de visión por computadora (Dosovitskiy *et al.*, 2020), por lo que es de interés probar esta variante de FNet en el problema de clasificación de imágenes y así a su vez experimentar el reemplazo de las subcapas de atención con una matriz de Hadamard, esperando un resultado similar o superior al mostrado por FNet.



(a) (b)

Figura 35. (a) La arquitectura del modelo transformadores. (b) La arquitectura del modelo FNet. Fuente: (a) (Vaswaniet al., 2017), (b) (Lee-Thorpet al., 2021)

Literatura citada

Ahmad, K. y Conci, N. (2019). How deep features have improved event recognition in multimedia: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, **15**(2): 1–27.

Athalye, A., Engstrom, L., Ilyas, A., y Kwok, K. (2018). Synthesizing robust adversarial examples. En: *International conference on machine learning*. PMLR, pp. 284–293.

Boucher, N., Shumailov, I., Anderson, R., y Papernot, N. (2021). Bad characters: Imperceptible nlp attacks. *Recuperado en 26-10-2021 de https://arxiv.org/abs/2106.09898*.

Brown, T. B., Mané, D., Roy, A., Abadi, M., y Gilmer, J. (2017). Adversarial patch. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1712.09665*.

Carlini, N. y Wagner, D. (2017). Towards evaluating the robustness of neural networks. En: *2017 IEEE Symposium on Security and Privacy*. IEEE, pp. 39–57.

Cassel, M. y Lima, F. (2006). Evaluating one-hot encoding finite state machines for seu reliability in sram-based fpgas. En: *12th IEEE International On-Line Testing Symposium (IOLTS'06)*. IEEE, pp. 6–pp.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., y Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1810.00069*.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., y Hsieh, C.-J. (2017). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. En: *Proceedings of the 10th ACM workshop on artificial intelligence and security*. pp. 15–26.

Cheng, Y., Wang, D., Zhou, P., y Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1710.09282*.

Cho, J. H. y Hariharan, B. (2019). On the efficacy of knowledge distillation. En: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, y Li Fei-Fei (2009). Imagenet: A large-scale hierarchical image database. En: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 248–255.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. En: *2009 IEEE conference on computer vision and pattern recognition*. IEEE, pp. 248–255.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, **29**(6): 141–142.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., *et al.* (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *Recuperado en 26-10-2021 de https://arxiv.org/abs/2010.11929*.

Efros, A. A. y Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. En: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. pp. 341–346.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., y Madry, A. (2019). Exploring the landscape of spatial robustness. En: *International Conference on Machine Learning*. PMLR, pp. 1802–1811.

Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., y Song, D. (2017). Robust physical-world attacks on machine learning models. *Recuperado en 26-10-2021 de https://s3.observador.pt/wp-content/uploads/2017/08/08133934/1707-08945.pdf*.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., y Brendel, W. (2018). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1811.12231*.

Goodfellow, I. J., Shlens, J., y Szegedy, C. (2014). Explaining and harnessing adversarial examples. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1412.6572*.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., y McDaniel, P. (2017). On the (statistical) detection of adversarial examples. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1702.06280*.

Guo, C., Rana, M., Cisse, M., y Van Der Maaten, L. (2017). Countering adversarial images using input transformations. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1711.00117*.

Hadamard, J. (1893). Résolution d'une question relative aux determinants. *Bull. des sciences math.*, **2**: 240–246.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell system technical journal*, **29**(2): 147–160.

He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 770–778.

Hendrik Metzen, J., Chaithanya Kumar, M., Brox, T., y Fischer, V. (2017). Universal adversarial perturbations against semantic image segmentation. En: *Proceedings of the IEEE international conference on computer vision*. pp. 2755–2764.

Hinton, G., Vinyals, O., Dean, J., *et al.* (2015). Distilling the knowledge in a neural network. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1503.02531*.

Huffman, D. A. (1954). The synthesis of sequential switching circuits. *Journal of the franklin Institute*, **257**(3): 161–190.

Huffman, W. C. y Pless, V. (2010). *Fundamentals of error-correcting codes*. Cambridge University Press.

Jang, U., Wu, X., y Jha, S. (2017). Objective metrics and gradient descent algorithms for adversarial examples in machine learning. En: *Proceedings of the 33rd Annual Computer Security Applications Conference*. pp. 262–277.

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, **25**: 1097–1105.

Kurakin, A., Goodfellow, I., y Bengio, S. (2016). Adversarial machine learning at scale. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1611.01236*.

LeCun, Y. *et al.* (2015). Lenet-5, convolutional neural networks. *Recuperado en 26-10-2021 de http://yann.lecun.com/exdb/lenet*, **20**(5): 14.

Lee-Thorp, J., Ainslie, J., Eckstein, I., y Ontanon, S. (2021). Fnet: Mixing tokens with fourier transforms. *Recuperado en 21-01-2022 de https://arxiv.org/abs/2105.03824*.

Li, X. y Li, F. (2017). Adversarial examples detection in deep networks with convolutional filter statistics. En: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 5764–5772.

MacWilliams, F. J. y Sloane, N. J. A. (1977). *The theory of error correcting codes*, Vol. 16. Elsevier.

Mądry, A., Makelov, A., Schmidt, L., Tsipras, D., y Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *stat*, **1050**: 9.

Metzen, J. H., Genewein, T., Fischer, V., y Bischoff, B. (2017). On detecting adversarial perturbations. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1702.04267*.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., y Ishii, S. (2015). Distributional smoothing with virtual adversarial training. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1507.00677*.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., y Frossard, P. (2017). Universal adversarial perturbations. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1765–1773.

Muller, D. E. (1954). Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE professional group on electronic computers*, (3): 6–12.

Ozdag, M. (2018). Adversarial attacks and defenses against deep neural networks: A survey. *Procedia Computer Science*, **140**: 152–161.

Pang, T., Du, C., Dong, Y., y Zhu, J. (2017). Towards robust detection of adversarial examples. *arXiv preprint arXiv:1706.00633*.

Papernot, N., McDaniel, P., Wu, X., Jha, S., y Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. En: *2016 IEEE symposium on security and privacy (SP)*. IEEE, pp. 582–597.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., y Swami, A. (2017). Practical black-box attacks against machine learning. En: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. pp. 506–519.

Quiroz, B. (2021). *Recuperación de información utilizando códigos de Hadamard y el grafo de semi-espacios proximales*. Tesis de maestría, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California. xi, 101 hojas.

Rauber, J., Zimmermann, R., Bethge, M., y Brendel, W. (2020). Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, **5**(53): 2607.

Reddy, M. V., Banburski, A., Pant, N., y Poggio, T. (2020). Biologically inspired mechanisms for adversarial robustness. *Recuperado en 04-11-2021 de https://arxiv.org/abs/2006.16427*.

Redmon, J. y Farhadi, A. (2018). Yolov3: An incremental improvement. *Recuperado en 21-10-2021 de https://arxiv.org/abs/1804.02767*.

Reed, I. S. (1953). A class of multiple-error-correcting codes and the decoding scheme. Reporte técnico, Massachusetts Inst of Tech Lexington Lincoln Lab.

Reed, I. S. y Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, **8**(2): 300–304.

Ren, S., He, K., Girshick, R., y Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, **28**.

Rodríguez, P., Bautista, M. A., Gonzalez, J., y Escalera, S. (2018). Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, **75**: 21–31.

Rony, J., Hafemann, L. G., Oliveira, L. S., Ben Ayed, I., Sabourin, R., y Granger, E. (2019). Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **2019-June**: 4317–4325.

Rosenfeld, A., Zemel, R., y Tsotsos, J. K. (2018). The elephant in the room. *Recuperado en 04-11-2021 de https://arxiv.org/abs/1808.03305*.

Serban, A., Poll, E., y Visser, J. (2020). Adversarial examples on object recognition: A comprehensive survey. *ACM Computing Surveys*, **53**(3).

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, **27**(3): 379–423.

Sharif, M., Bhagavatula, S., Bauer, L., y Reiter, M. K. (2019). A general framework for adversarial examples with objectives. *ACM Trans. Priv. Secur.*, **22**(3).

Simonyan, K. y Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *Recuperado en 21-10-2021 de https://arxiv.org/abs/1409.1556*.

Su, J., Vargas, D. V., y Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, **23**(5): 828–841.

Sylvester, J. J. (1867). Lx. thoughts on inverse orthogonal matrices, simultaneous sign-successions, and tessellated pavements in two or more colours, with applications to newton's rule, ornamental tile-work, and the theory of numbers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **34**(232): 461–475.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., y Fergus, R. (2013). Intriguing properties of neural networks. *Recuperado en 21-10-2021 de https://arxiv.org/abs/1312.6199*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., y Wojna, Z. (2016). Rethinking the inception architecture for computer vision. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 2818–2826.

Thys, S., Van Ranst, W., y Goedemé, T. (2019). Fooling automated surveillance cameras: adversarial patches to attack person detection. En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.

Toshev, A. y Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1653–1660.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., y McDaniel, P. (2020). Ensemble adversarial training: Attacks and defenses.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., y Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, **30**.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., y Wierstra, D. (2016). Matching networks for one shot learning. *Recuperado en 26-10-2021 de https://arxiv.org/abs/1606.04080*.

Wang, J., Zhang, T., Sebe, N., Shen, H. T., *et al.* (2017). A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, **40**(4): 769–790.

Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., y Yuille, A. (2017). Adversarial examples for semantic segmentation and object detection. En: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 1369–1378.

Yang, S., Luo, P., Loy, C. C., Shum, K. W., y Tang, X. (2015). Deep representation learning with target coding. En: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29.

Zhang, Q., Cui, Z., Niu, X., Geng, S., y Qiao, Y. (2017). Image segmentation with pyramid dilated convolution based on resnet and u-net. En: *International Conference on Neural Information Processing*. Springer, pp. 364–372.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., y Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, **40**(6): 1452–1464.