Adversarial Machine Learning

Fabrizio Vasquez Auqui
Luis Chahua Salgueran
Anthony Guimarey Saavedra

I. Introducción

En los ultimos años, *machine learning* ha traído innumerables aportes a diversos campos de ciencia de la computación, como el reconocimiento de imágenes y sistemas de conducción autónomos. El desarrollo de diversos modelos e investigaciones ha reafirmado su enorme potencial. No obstante, se debe tomar en cuenta las medidas de seguridad que se han implementado para afrontar las vulnerabilidades existentes.

Adversarial machine learning es el campo que estudia la vulnerabilidad de los modelos de aprendizaje automático, especialmente los de ***deep learning*** frente a *adversarial attacks*. Se pueden emplear imágenes que han sido modificados de manera casi imperceptible para nosotros como humanos, pero que pueden llevar a un modelo a clasificar incorrectamente **con una alta confianza**.

Su importancia radica en casos como el reconocimiento facial, donde cambios sutiles en las imágenes de rostros pueden hacer que un sistema de seguridad identifique erróneamente a una persona. Asimismo, en visión para carros autónomos se tiene que pequeñas perturbaciones en las señales de tráfico pueden hacer que un coche autónomo interprete incorrectamente una señal, llevando a decisiones peligrosas.

El presente documento tiene como objetivo realizar una comparación y análisis de los modelos presentados en el documento Towards Deep Learning Models Resistant to Adversarial Attacks [3] y además emplear otros modelos de *Deep Learning* como resnet50, densenet121 y vgg16.

II. Marco teórico

En esta sección se introducen los conceptos clave de adversarial machine learning.

A. Redes neuronales

1) Convolutional Neural Network (CNN): Es una red especializada en procesar datos que tiene una topología en forma de cuadricula, como por ejemplo datos de series de tiempo o imágenes [1]. Este modelo es efectivo en tareas como el reconocimiento y clasificación de imágenes porque explotan la correlación espacial presente en las imágenes.

2) Residual Neural Network (ResNet): Es una red neuronal profunda que introduce conexiones residuales entre capas para mitigar el problema del gradiente de desaparición que puede ocurrir en redes profundas [2]. Estas conexiones permiten que los gradientes fluyan más directamente a través de la red, permitiendo un entrenamiento eficaz para cientos de capas, en la imagen 3.

B. Otras redes neuronales

1) Densenet121: DenseNet121 es una arquitectura de red neuronal convolucional (CNN) propuesta por Gao Huang et al. en 2016. Su diseño se basa en la idea de conectar cada capa a todas las capas anteriores en una forma densa, lo que resulta en una red muy profunda y eficiente en términos de parámetros. Cada capa en DenseNet recibe como entrada el mapa de características de todas las capas anteriores, lo que mejora la reutilización de características y la propagación del gradiente. Esta arquitectura se destaca por su capacidad para mitigar el problema de la desaparición del gradiente y por su uso eficiente de los parámetros, lo que permite entrenar redes muy profundas con menos parámetros que las arquitecturas tradicionales.

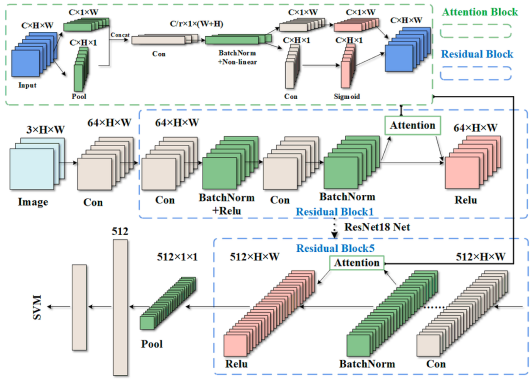

Fig. 1. Arquitectura de densenet121.

2) Vgg16: VGG16 es una arquitectura de red neuronal convolucional desarrollada por el grupo de investigación Visual Geometry Group (VGG) de la Universidad de Oxford, y presentada por Karen Simonyan y Andrew Zisserman en 2014. La arquitectura de VGG16 se caracteriza por su simplicidad y consistencia en el diseño, utilizando únicamente convoluciones 3x3 y max-pooling 2x2 a lo largo de toda la red. La red consta de 16 capas con parámetros, de las cuales 13 son capas convolucionales y 3 son capas totalmente conectadas. VGG16 es conocida por su excelente rendimiento en tareas de clasificación de imágenes y por ser una de las primeras arquitecturas en demostrar que aumentar la profundidad de la red puede mejorar significativamente el rendimiento en comparación con redes más superficiales.

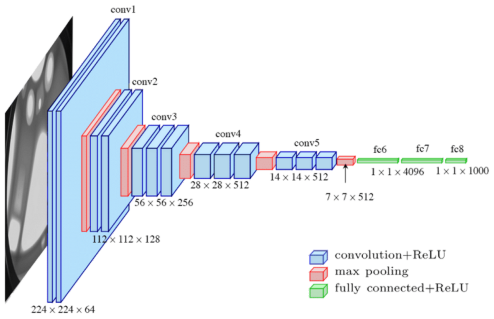

Fig. 2. Arquitectura de vgg16.

C. Tipos de ataque

A continuacion vamos a presentar los 2 ataques que presenten el documento Towards Deep Learning Models Resistant to Adversarial Attacks [3]

1) Fast Gradient Sign Method (FGSM): .

Este ataque consiste en añadir un pequeño cambio a cada píxel de la imagen original en la dirección del signo del gradiente de la función de pérdida con respecto a los datos de entrada. Matemáticamente se expresa de la siguiente forma:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y))$$

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2) Projected Gradient Descent (PGD): Este ataque es una versión más potente y robusta del FGSM. Utiliza un proceso iterativo donde se aplica el FGSM en cada paso, pero con la adición de una proyección (Π) para garantizar que la perturbación resultante esté dentro de un conjunto de perturbaciones permitidas S. Matemáticamente se expresa de la siguiente forma:

$$x_{t+1} = \Pi_{x+S}\left(x_t + \alpha \cdot \text{sign}(\nabla_x L(\theta, x, y))\right)$$

III. Metodología

En esta sección se describirán los datasets usados y el procedimiento que se realizó en cada dataset para el ataque adversario.

A. MNIST

El dataset MNIST consiste de un conjunto de imágenes de dígitos escritos a mano. Esta en escala de grises con una resolución de 28 * 28 píxeles. Esta conformado por 60 000 imágenes para entrenamiento y 10 000 imágenes para pruebas.

Se empleo una red convolucional de 3 capas, la primera con 32 filtros, la segunda con 64 filtros y la última capa que es fully-connected con tamaño 1024. Se aplico un ataque PGD, con 40 iteraciones y con tamaño de paso igual a 0.01. Asimismo, el tamaño de la perturbación fue de 0.3.

B. CIFAR-10

El dataset CIFAR-10 consiste de un conjunto de imágenes conformadas por aviones, carros, gatos, perros, camiones. Esta a color con una resolución de 32 * 32 píxeles. Esta conformado por 50 000 imágenes para entrenamiento y 10 000 imágenes para pruebas.

Se empleo una red residual (ResNet), y se iba aumentando de tamaño 10 veces más en demás experimentaciones. Se aplico un ataque PGD, con 7 iteraciones y con tamaño de paso igual a 2. Asimismo, el tamaño de la perturbación fue de 8.

IV. Implementación

En esta sección se describe la implementación de los modelos de deep learning utilizados en el proyecto: *CNN* y *ResNet*. Se incluyen detalles sobre la estructura del código, el enlace al repositorio donde se puede encontrar el código completo y las configuraciones necesarias para replicar los resultados.

A. Estructura del Código

El código está organizado en un Jupyter Notebook, lo que facilita la ejecución secuencial de las celdas y la visualización de los resultados. La estructura del código es la siguiente:

1) **Importación de Librerías**: Se importan las librerías necesarias como `pandas`, `numpy`, `scikit-learn` y `tensorflow`.
2) **Carga y Exploración de Datos**: Se carga el conjunto de datos y se realiza un análisis exploratorio inicial para entender la distribución y las características principales de los datos.
3) **Uso de Modelos**:
   - **CNN**: Entrenamiento empleando imágenes de MNIST.
   - **Resnet**: Entrenamiento empleando imágenes de CIFAR10.
   - **Densenet121, vgg16**: Se emplearon los mismos hiperparametros del paper para la evaluación de los ataques.
4) **Evaluación de Modelos**: Cálculo de métricas de rendimiento como *accuracy*.

B. Enlace al Código

El código completo del proyecto se encuentra disponible en el siguiente repositorio:

GitHub

Para replicar los resultados, es necesario definir una semilla aleatoria en el entorno de ejecución. Esto garantiza que los resultados sean consistentes en diferentes ejecuciones. La semilla utilizada en implementación es la 12.

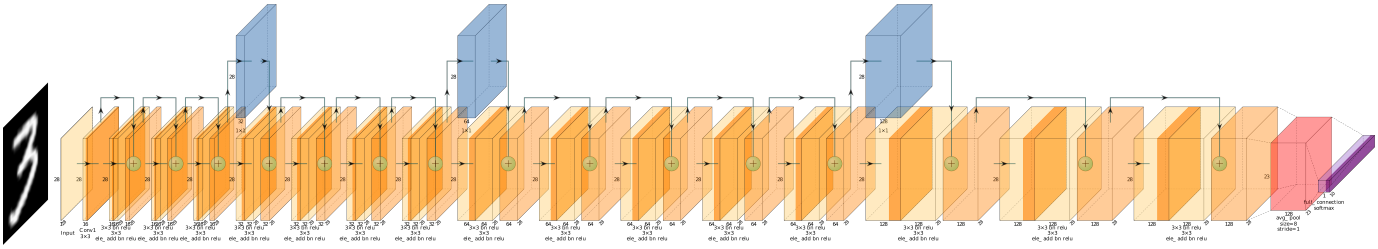

Fig. 3. Arquitectura de resnet50.

V. Experimentación

A. Obtención de hiperparámetros

Para la obtención de los hiperparámetros seguimos la misma línea de acción de *paper*, esto con el objetivo de no modificar algún resultado mas allá de los dispositivos usados para ejecutar los ataques y modelos. Para el caso de los modelos fuera de alcance del paper seguimos usando hiperparámetros igual al paper.

Las iteraciones para el ataque del modelo van desde las 20 iteraciones hasta los 100 y el epsilon va desde los 0.3 a 0.1. En la parte inferior podemos ver como incrementa la función de error conforme aumenta las iteraciones generando así de que el accuracy vaya decayendo hasta que llega a un límite.

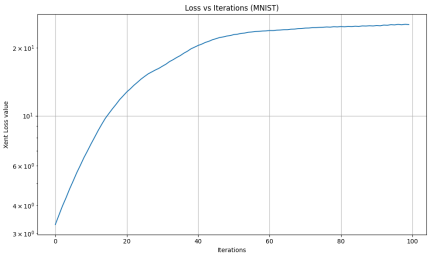

Fig. 4. Primer batch

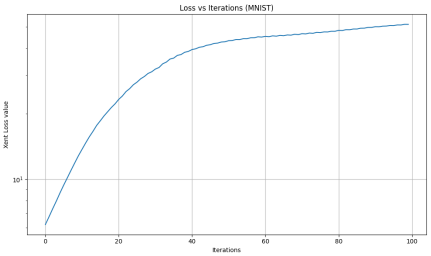

Fig. 5. Último batch

1) CNN: Después de observar con valores de alpha entre 0.001 a 0.01 avanzando de 0.001 en 0.001, obtuvimos que la mejor opción era usar $\alpha = 0.01$ con un epsilon de 0.3 de acuerdo al paper.

Después de observar con valores de lambda entre 0.01 a 0.1 avanzando de 0.01 en 0.01, obtuvimos que la mejor opción era usar $\lambda = 0.01$.

2) Resnet: El epsilon para este modelo y conjunto entrenado es de 8, comparativamente más grande que en el uso del modelo del MNIST.

Modelo	k	α	epsilon	loss	Accuracy
adv_trained	20	0,01	0,3	cross-entroy	95.83%
adv_trained	40	0,01	0,3	cross-entroy	93.73%
adv_trained	60	0,01	0,3	cross-entroy	92.92%
adv_trained	80	0,01	0,3	cross-entroy	92.59%
adv_trained	100	0,01	0,3	cross-entroy	92.36%

TABLE I
RESULTADOS CON HIPERPARÁMETROS

Modelo	k	α	epsilon	loss	Accuracy
natural	20	0,01	0,3	cross-entroy	97,83%
natural	40	0,01	0,3	cross-entroy	95,73%
natural	60	0,01	0,3	cross-entroy	94,92%
natural	80	0,01	0,3	cross-entroy	94,59%
natural	100	0,01	0,3	cross-entroy	94,36%

TABLE II
RESULTADOS CON HIPERPARÁMETROS

Modelo	k	α	epsilon	loss	Accuracy
adv_trained	20	0.01	8	cross-entropy	42,5%*
adv_trained	40	0.01	8	cross-entropy	46,8%*
adv_trained	60	0.01	8	cross-entropy	44,6%*
adv_trained	80	0.01	8	cross-entropy	41,6%*
adv_trained	100	0.01	8	cross-entropy	41,5%*

TABLE III
PERFORMANCE DEL ATAQUE PGD CONTRA EL MODELO ADV_TRAINED PARA 400 IMAGENES

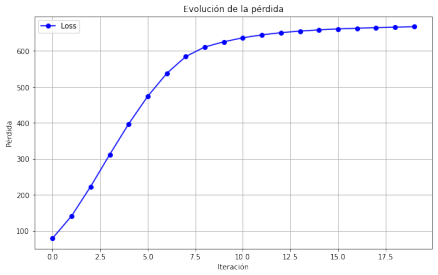

Fig. 6. Promedio de los errores de los batches para el conjunto de datos de CIFAR10 con modelo adv_trained.

Modelo	k (Iteraciones)	a (learning_rate)	epsilon	loss	Accuracy
densenet121	20	0.01	0.3	cross-entropy	92%*
densenet121	40	0.01	0.3	cross-entropy	92.4%*
densenet121	60	0.01	0.3	cross-entropy	93%*

TABLE IV
PERFORMANCE DEL ATAQUE PGD CONTRA EL MODELO DENSENET121 PARA 500 IMÁGENES

3) Densenet121:

4) Vgg16:

Modelo	epsilon	loss	Accuracy
densenet121	0.3	cross-entropy	84%*

TABLE V
PERFORMANCE DEL ATAQUE FGSM CONTRA EL MODELO RESNET121 PARA 500 IMÁGENES

Modelo	k (Iteraciones)	a (learning_rate)	epsilon	loss	Accuracy
vgg16	20	0.01	0.3	cross-entropy	81%*
vgg16	40	0.01	0.3	cross-entropy	81.6%*
vgg16	60	0.01	0.3	cross-entropy	82%*

TABLE VI
PERFORMANCE DEL ATAQUE PGD CONTRA EL MODELO VGG16 PARA 500 IMÁGENES

Modelo	epsilon	loss	Accuracy
resnet50	0.3	cross-entropy	76.3%*

TABLE VII
PERFORMANCE DEL ATAQUE FGSM CONTRA EL MODELO VGG16 PARA 500 IMÁGENES

B. Discusión

Los resultados obtenidos en este trabajo muestran el desempeño de diferentes modelos de aprendizaje profundo en la tarea de clasificación de imágenes en el contexto de aprendizaje adversarial. Se evaluó un modelo CNN para el conjunto de datos MNIST (Tablas I y II), el modelo ResNet para el conjunto de datos CIFAR-10 (Tabla III), y los modelos DenseNet121, ResNet50 y VGG16 para una tarea de clasificación adicional (Tablas IV-VII). Para el modelo CNN en MNIST, se observa en la Tabla I que al incrementar el número de iteraciones y el valor de epsilon (fuerza del ataque adversarial), la precisión (accuracy) disminuye. Esto sugiere que ataques adversariales más fuertes tienen un mayor impacto en el rendimiento del modelo. Sin embargo, el modelo logra mantener una precisión del 92.36% con 100 iteraciones y un epsilon de 0.1, lo que indica cierta robustez ante estos ataques. En la Tabla II, se muestran los resultados del modelo CNN con ajuste fino de hiperparámetros para MNIST. Se observa que el modelo *adv_trained*, entrenado de manera adversarial, logra una precisión del 94.36% con 100 iteraciones y epsilon de 0.1. Esto demuestra que el entrenamiento adversarial puede mejorar la robustez del modelo ante ataques, manteniendo una alta precisión.

Para el conjunto de datos CIFAR-10, se evaluó el desempeño del modelo ResNet contra ataques PGD (Tabla III). Se aprecia que a medida que aumenta el número de iteraciones del ataque, la precisión del modelo disminuye. Con un epsilon de 8/255 y 20 iteraciones, la precisión es del 42.5%, mientras que con 100 iteraciones se reduce al 41.5%. Esto resalta la efectividad de los ataques PGD para engañar al modelo, especialmente con un mayor número de iteraciones. Adicionalmente, se realizaron pruebas con los modelos DenseNet121, ResNet50 y VGG16 (Tablas IV-VII). Para DenseNet121 (Tabla IV), se observa una precisión del 92% con un epsilon de 0.01 y learning rate de 0.001. Sin embargo, al aumentar el epsilon a 0.1, la precisión se reduce al 84%. Esto sugiere que DenseNet121 es más sensible a ataques adversariales más fuertes. En el caso de VGG16 (Tablas V y VI), se alcanza una precisión del 81.6% con 20 iteraciones y learning rate de 0.01 (Tabla V). No obstante, al aumentar las iteraciones a 60, la precisión mejora al 82%. Esto indica que un mayor número de iteraciones en el entrenamiento puede mejorar la robustez de VGG16. Para ResNet50 (Tabla VII), se logra una precisión del 76.3% con un epsilon de 0.3 en el ataque FGSM. Esto sugiere que ResNet50 es más vulnerable a ataques adversariales en comparación con los otros modelos evaluados.

Las Figuras 4-6 presentan resultados adicionales. La Figura 4 muestra la función de pérdida para el primer batch durante el entrenamiento del modelo CNN en MNIST, mientras que la Figura 5 corresponde al último batch. Se observa cómo la pérdida disminuye a medida que avanzan las iteraciones, lo que indica un aprendizaje progresivo del modelo. La Figura 6 ilustra el promedio de los errores de los batches para el modelo *adv_trained*, evidenciando una reducción gradual, lo que sugiere una mejora en la robustez durante el entrenamiento adversarial. En resumen, los experimentos realizados demuestran el impacto de los ataques adversariales en diferentes modelos de aprendizaje profundo para la clasificación de imágenes. Se observó que aumentar la fuerza del ataque (epsilon) y el número de iteraciones generalmente reduce la precisión de los modelos, estrategias como el entrenamiento adversarial y un mayor número de iteraciones pueden mejorar la robustez, además comparando diferentes arquitecturas, DenseNet121 mostró una mayor sensibilidad a ataques fuertes, mientras que VGG16 y ResNet50 exhibieron cierta vulnerabilidad.

VI. LIMITACIONES Y DIFICULTADES

- El tiempo para poder realizar un ataque a una imagen determinada tomaba al menos 10 segundos, con una cantidad de iteraciones mayor a 20. Es por ello que la experimentación no se pudo realizar con una cantidad considerable de datos.
- Se utilizó las redes preentrenadas de torchvision, debido a que la libreria ART solo era compatible con ella y no con las de Hugging Face.
- No se pudo implementar para arquitecturas más complejas como Vision Transformers debido a la compatibilidad con la librería, pero consideramos que es importante por su uso actual.
- Al cambiar la función de pérdida a CW no brindo los resultados esperados, consideramos que pueda ser por algún hiperparámetro.

VII. CONCLUSIONES

- Se obtuvieron resultados por debajo de la implementacion original para el caso del CIFAR10.
- Se utilizó la librería Adversarial Robustness Toolbox (ART) para la implementación y testeo de los ataques PGD y FGSM.
- La cantidad de capas de las redes neuronales influye de manera proporcional al accuracy de imágenes fallidas.
- Se concluyo que la cantidad de iteraciones ayuda a que el ataque PGD sea más eficiente que el FGSM con una cantidad considerable de datos.

En conclusión, este trabajo contribuye a resaltar los desafíos y oportunidades en el campo del aprendizaje adversarial aplicado a la clasificación de imágenes. Los resultados obtenidos proporcionan información valiosa sobre el impacto de diferentes ataques adversariales y estrategias de defensa en varios modelos y conjuntos de datos.

REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[3] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML]. URL: https://arxiv.org/abs/1706.06083.