RECONOCIMIENTO DE PINTURAS USANDO REDES NEURONALES CONVOLUCIONALES

LUIS FERNANDO LOYOLA CRUZ

1. Introducción

Con los avances tecnológicos de estos últimos años, no es sorpresa que la computación se pueda mezclar a áreas que, cualquier pensaría que tienen tan poco en común como lo es el arte.
Para los historiadores del arte, el reconocimiento de pinturas es un trabajo, nada trivial, que suele llevar bastante tiempo. Sin embargo con ayuda de la *Inteligencia Artificial* es posible hacer mas rápido y de alguna manera mas eficiente este trabajo.
Supongamos que encontramos una pintura de la cual no tenemos mucha información. Lo que nos gustaría saber es si perteneció a algún pintor reconocido, si es así también nos gustaría saber a que corriente pertenece o que estilo de pintura se utilizó.
Por otro lado, imaginemos que una persona cualquiera llega a una galería de artes prestigiosa ofreciendo una pintura que supuestamente el encontró y que afirma ser del famoso pintor *Pablo Picasso*, pintura por la cual este tipo esta pidiendo unos cuantos millones de dolares, lo que nos lleva a preguntarnos, ¿Será una pintura autentica? ¿O solo es una réplica?

Para ambos problemas, la solución mas obvia es llamar a un prestigioso historiador de arte que nos ayude a evaluar la calidad de la pintura y nos pueda quitar nuestras inquietudes acerca del tema, lo que posiblemente le tomé una gran cantidad de tiempo añadiendo el hecho de que el historiador nos va cobrar una cantidad considerable de dinero por su trabajo.
Ahora imaginemos que en vez de tener que llamar a un profesional en el ámbito del arte tenemos una cámara fotográfica de alta calidad, vamos y le tomamos una foto a la pintura de manera que dicha foto captura a la perfección los detalles de esta, en seguida vamos y encendemos nuestra computadora, transferimos la imagen a esta, abrimos un programa mágico que recibe como entrada una fotografía de una pintura y como salida nos dice si es falsa o no, si no lo es también nos proporciona los datos del posible autor, la corriente, el estilo, etc. todo esto en unos cuantos minutos y sin tener que sacar una moneda de nuestro bolsillo.

La idea es que a partir de una imagen que represente la digitalización de una pintura podamos obtener información acerca de esta, como podría ser el autor, la corriente, el estilo, incluso una descripción de la pintura y la probabilidad de que esta información sea correcta.

2. Redes Neuronales Convolucionales

Aunque las redes neuronales convolucionales fueron inventadas en el año 1980 por Kunihiko Fukushima[5] fue hasta el año 2012 que Dan Ciresan y algunas personas

mas vieron el potencial de estas para el procesamiento de imágenes[6].

Las redes neuronales convolucionales son muy parecidas a las redes neuronales clásicas, pues aunque ambas están compuestas de neuronas, las cuales tienen asociado un peso y un sesgo la arquitectura es un poco diferente.

La manera en como las redes neuronales convolucionales van a estar constituidas es la siguiente:

- Una capa convolucional(de ahi el nombre de la red) la cual funciona utilizando la función de convolución, una función matemática que en el contexto de visión por computadora, preserva la relación espacial entre los píxeles al aprender las características de la imagen utilizando pequeños cuadrados de datos de entrada[2].
- Una capa de reducción o de *pooling*,la cual reduce la cantidad de parámetros al quedarse con las características mas relevantes de la imagen.
- Una capa de clasificación totalmente conectada.

Dado que el problema planteado pertenece al área de *visión por computadora*, las redes neuronales convolucionales son un gran candidato frente a otras técnicas en el área de *Aprendizaje de Máquina* y *Aprendizaje Profundo*, ya que lo que necesitamos es procesar imágenes y encontrar patrones entre ellas, identificando así, después de un entrenamiento, la clase a la que cada imagen pertenece.

3. Conjunto de Datos

Los datos que se utilizaron para entrenar a la red neuronal fueron obtenidos del sitio web `Kaggle` y corresponde a un conjunto de imágenes obtenidas de `WikiArt`.

Dado que el conjunto de datos original tenia un peso aproximado de 80GB y había limitaciones en cuanto a procesamiento, velocidad de descarga y almacenamiento se optó por utilizar un conjunto de datos preprocesado. En este caso el procesamiento solo significaba un cambio de tamaño($256x256$) el cual disminuyo el tamaño del conjunto de datos en un 97.5 %.

El conjunto de datos esta constituido por **103239** imágenes, de las cuales **79422** pertenecen al conjunto de entrenamiento. Entre las características proporcionadas tenemos el nombre de la imagen, autor, estilo, genero, tamaño, titulo de la pintura, año, etc.

En cuanto a características relevantes tenemos un aproximado de 2312 artistas, 137 estilos y 43 géneros.

Por motivos experimentales y de tiempo, del conjunto de datos completo se tomaron algunas muestras relevantes. Se eligieron 10 artistas con mas de 300 muestras, entre ellos *Pablo Picasso*, *Salvador Dalí*, *Claude Monet*, *Vincent Van Gogh*, etc. De las pinturas de esos 10 artistas, se eligieron los estilos y géneros con mas muestras, obteniendo un total de 11 estilos y 16 géneros, de manera que nuestro conjunto de datos se vio reducido a un total de 4784 imágenes.

De nuestro conjunto de datos obtenido, 80 % de las imágenes pertenecen al conjunto de entrenamiento, 10 % al conjunto de validación y 10 % al conjunto de prueba.

4. Aumento de datos

Como consecuencia de la reducción de los datos y con la finalidad de tener una buena precisión fue necesario obtener mas datos, o en este caso, mas imágenes. Sin embargo, ¿Cómo conseguimos mas datos?
La respuesta a esta pregunta es la técnica *Aumento de Datos*.
El Aumentos de Datos es una técnica que consiste en aplicar una o mas transformaciones, en este caso, a una serie de imágenes con el fin de tener mas variedad de datos.

En la figura 1 podemos observar como las 3 imágenes corresponden a una pelota de tenis, con la diferencia de que en cada imagen la pelota esta en un lugar diferente, esto porque se le aplico una traslación a la imagen original

Figura 1.

Entre las técnicas de aumento de datos mas populares están:

- Rotaciones
- Traslaciones
- Cambio de tamaño
- Cropping(tomar un pedazo de la imagen)
- Flipping

De las cuales, se utilizaron solo 3 en la resolución del problema: Çroppingz "Flipping"

5. Implementación

Inspirado por [10], en lugar de tratar de crear una red neuronal convolucional desde 0, se decidió hacer la comparación entre dos arquitecturas, *Resnet18* y *AlexNet*. Para ello, se hizo uso del lenguaje de programación `Python` y del framework `Pytorch` que además de tener implementadas dichas arquitecturas, provee de algunas otras cosas como funciones de perdidas, algoritmos de optimización de funciones, transformaciones(aumento de datos), creacion de *datasets*, etc.

Además de hacer uso de las redes neuronales que Pytorch ya tiene implementadas, se hizo uso de las mismas arquitecturas entrenadas con imágenes de ImageNet, lo cual se explicará mas adelante.

Por recomendación de [4] se creo una función que inicializa los pesos de una capa de una red neuronal utilizando una técnica denominada *Inicialización de Xavier*. Por otro lado, siguiendo el caso de uso descrito en [13] se decidio utilizar como función de perdida la función de Entropia Cruzada.
Por último, después de un par de experimentos, se encontró que el algoritmo de

optimización *Adam* tenia una mejor precisión que el algoritmo *SGD* por lo que se utilizó el primero con una tasa de aprendizaje de $1e-3$.

6. Experimentos

La metodología que se utilizó para entrenar a los modelos fue la siguiente:

- Se entrenó al modelo durante un determinado número de épocas haciendo usa del conjunto de entrenamiento.
- Terminada la fase de entrenamiento, se reducia el numero de épocas a aproximadamente la mitad y se entrenaba el model de nuevo, pero haciendo uso del conjunto de validación.
- Se media la precisión del modelo con el conjunto de prueba.

Inicialmente, se creía que al ser una red neuronal convolucional el entrenamiento tenia que ser relativamente largo, por lo que se decidió entrenar al modelo durante 20 épocas en la fase de entrenamiento y 7 en la fase de validación. Sin embargo, por motivos de tiempo se decidió decrementar el número de épocas para ambos casos, en donde fue posible notar que la precisión obtenida en las 27 épocas era la misma si las reducíamos a 8, fue asi como la fase de entrenamiento duro 5 épocas y la fase de validación 3.

Aun así, después de varios experimentos modificando la tasa de aprendizaje y otras cosas como las transformaciones de las imagenes o el tamaño del lote del conjunto de entramiento y validación los resultados obtenidos no eran buenos en ninguna de las dos arquitecturas.

Investigando un poco sobre las posibles causas de que el desempeño del modelo fuesen bajas nos encontramos con una técnica llamada *Transferencia de Aprendizaje*. Esta técnica consiste en usar una red neuronal ya entrenada con un conjunto de datos para aprender un nuevo conjunto de datos, es decir, si tenemos una red neuronal que sepa clasificar coches, podemos utilizar la misma red neuronal, haciendo unos pequeños cambios descritos en [1], para poder clasificar motocicletas.

Siguiendo esta metodología, Pytorch ofrece redes neuronales ya entrenadas con un conjunto de imagenes obtenidas de `ImageNet`.

Teniendo en mente esto, la metodología que se optó por utilizar es la siguiente:

- Se hace uso de una red neuronal convolucional pre-entrenada(Resnet18 o AlexNet).
- Se cambia la última capa, la cual corresponde a la capa de clasificación para adaptarla a nuestro problema.
- Se entrena el modelo con el conjunto de entrenamiento de manera que solo se actualicen los pesos de la capa que modificamos, dejando todo lo demás intacto.
- Terminada la fase de entrenamiento, procedemos a entrenar el modelo con el conjunto de validación por un menor tiempo, dejando en esta ocasión que toda la red se modifique.
- Medimos la precisión del modelo con el conjunto de prueba.

7. Resultados

Después de una serie de experimentos con distintas épocas, distintos conjuntos de datos, distintas arquitecturas de de redes neuronales obtenemos los siguientes resultados:

Resultados					
Arquitectura	Tipo	Tiempo(minutos)	Epocas	Pre-Entrenada	Precisión
Resnet18	Artista	8	8	Si	68 %
Resnet18	Artista	21	8	No	38 %
AlexNet	Artista	3	8	Si	72 %
AlexNet	Artista	5	8	No	18 %

Inmediatamente notamos que la arquitectura AlexNet obtenía una mejor precisión en menor tiempo. Por otro lado, nos dimos cuenta que si usábamos redes que no estaban pre-entrenadas, la precisión era muy baja, por lo que posiblemente necesitarían un mayor tiempo de entrenamiento, el cual no teníamos, así que se dejaron de hacer experimentos con redes no pre-entrenadas.

Y fue asi como obtuvimos los siguientes y últimos resultados:

Resultados					
Arquitectura	Tipo	Tiempo(minutos)	Épocas	Pre-Entrenada	Precisión
Resnet18	Artista	13	15	Si	71 %
Resnet18	Estilo	6	8	Si	58 %
Resnet18	Género	5	8	Si	50 %
AlexNet	Artista	4	15	Si	74 %
AlexNet	Estilo	2	8	Si	66 %
AlexNet	Género	2	8	Si	54 %

8. Conclusiones

Observando las tablas podemos sacar varias conclusiones:

- AlexNet tiene una mejor precisión que Resnet18
- AlexNet se entrena en menor tiempo a comparación de Resnet18
- El desempeño de ambos modelos para clasificar artistas y estilos es medianamente bueno, aunque con Alexnet tenemos un mejor resultado en menor tiempo.
- En general, el desempeño de los modelos para clasificar géneros es bajo. Eso posiblemente se deba a un mal procesamiento de datos en conjunto con las transformaciones ademas la poca variedad de cada uno por clase.
- Posiblemente el cambio de tamaño antes del aumento de datos provocó una perdida de resolución que pudo tener cierta relevancia al momento de clasificar.
- En la mayoría de los casos, dada una muestra de nuestro conjunto de datos, si sucede que dada esa muestra el modelo da una clasificación errónea, la clasificación correcta se encuentra entre las 3 probabilidades que el modelo devuelve.

Referencias

[1] Sasank Chilamkurthy. *Transfer Learning Tutorial* https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

[2] Ujjwal Karn. *An Intuitive Explanation of Convolutional Neural Networks* https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[3] Andrej Karpathy *CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.github.io/

[4] Saurabh Yadav *Weight Initialization Techniques in Neural Networks* https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78

[5] Fukushima, Kunihiko (1980) *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position.* Biological Cybernetics 36 (4): 193–202.

[6] Ciresan, Dan; Ueli Meier; Jonathan Masci; Luca M. Gambardella; Jurgen Schmidhuber (2011). *Flexible, High Performance Convolutional Neural Networks for Image Classication.* Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two 2: 1237–1242.

[7] Bharath Raj. *Data Augmentation. How to use Deep Learning when you have Limited Data. Part II* https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

[8] Nitin Viswanathan. *Artist Identification with Convolutional Neural Networks* http://cs231n.stanford.edu/reports/2017/pdfs/406.pdf

[9] Adrian Lecoutre; Benjamin Negrevergne; Florian Yger. *Recognizing Art Style Automatically in painting with deep learning* https://pdfs.semanticscholar.org/6a6c/4797a6ed0e0a87f2e8cc569e3bfe6a858b3b.pdf?_ga=2.115701070.933191018.1560301983-772218503.1560301983

[10] Jennie Chen; Andrew Deng. *Comparison of Machine Learning Techniques for Artist Identification* http://cs229.stanford.edu/proj2018/report/41.pdf

[11] Wei Ren Tan; Chee Seng Chan; Hernan E. Aguirre; Kiyoshi Tanaka. *Ceci n'est pas une pipe: A Deep Convolutional Network for Fine-art Paintings Classification* http://cs-chan.com/doc/ICIP2016.pdf

[12] Samet Hicsonmez; Nermin Samet; Fadime Sener; Pinar Duygulu. *Deep networks for Recognizing styles of Artists Who illustrate children's books* https://arxiv.org/pdf/1704.03057.pdf

[13] Cross Entropy Loss https://pytorch.org/docs/stable/nn.html#crossentropyloss