Clasificación de imágenes de prendas de vestir por medio de técnicas de aprendizaje de máquina

Iván Darío González Collazos[1], Diego Andrés Baquero Tibocha[2] y Juliana Carolina Niño Valcárcel[3]

***Abstract*— Debido a la movilización de actividades de la vida cotidiana hacia el internet en la era digital, negocios como el comercio electrónico comienzan a utilizar más herramientas para satisfacer a sus clientes. La industria de la moda representa uno de los grandes rubros del sector de comercio electrónico, donde sus clientes tienen necesidades que deben ser satisfechas. Inno Clothing Sorter utiliza un modelo de entrenamiento que usa redes neuronales convolucionales, para poder ser utilizado mediante una aplicación móvil, que se comunica con el modelo mediante un RESTful API, teniendo una precisión de aproximadamente 75 % al analizar las imágenes en el momento de realizar el proceso de entrenamiento y validación del modelo de *cursiva* (ML) propuesto, el cual utilizó 50 épocas *epochs* para realizar el entrenamiento.**

I. Introducción

Dado que muchas de las actividades que realizamos diariamente se han trasladado hacia Internet, la incorporación de tecnología innovadora ha sido fundamental para la evolución de las diferentes tareas que se han involucrado en la era digital. Por ejemplo, el comercio electrónico es uno de los rubros más representativos y de mayor crecimiento de los últimos años, teniendo impacto en cómo el software es fabricado y vendido para atender las necesidades de las tiendas electrónicas.

Particularmente en la industria de la moda se presenta un desafío con las tiendas en línea: ¿cómo hacer que los computadores reconozcan las prendas en la web exactamente como lo hacen los consumidores en la vida cotidiana?. Para realizar esto, se debe aislar e identificar la ropa y los accesorios dentro de imágenes visualmente complejas como fotos personales o imágenes publicadas en Internet. El objetivo es que las personas puedan tener recomendaciones de prendas muy acordes con sus gustos, y así potecializar las ventas a través del comercio electrónico.

Por lo tanto, se necesitan técnicas avanzadas que permitan que las diferentes tiendas puedan acercarse a los clientes y solucionar sus problemas a la hora de elegir una prenda. Con el uso de una aplicación móvil y ML, el comercio electrónico de ropa puede estar más cerca de las elecciones del cliente.

II. Descripción del problema

Actualmente las personas buscan inmediatez cuando se refiere a satisfacer sus gustos y necesidades, dada la velocidad que el Internet ha logrado inyectar en la vida cotidiana de la sociedad.

Específicamente en el escenario de la compra de ropa online, es complejo hallar el producto exacto que se quiere adquirir, porque no se le ha dado la suficiente importancia al reconocimiento de todas las características que tiene la prenda y a muchos usuarios les terminan llegando productos que ni se acercan a lo que creían sería igual a la imagen modelo.

Partiendo de esto, trabajamos en el reto iMaterialist Challenge de Kaggle[4], el cual requiere desarrollar un modelo que detecte automáticamente los atributos de los productos en las fotos de prendas de vestir que vemos en Internet o que tomamos con nuestro celular y los clasifique en cuatro categorías generales: shoe, dress, pants y outerwear.

III. Solución propuesta

Para solucionar el reto de la clasificación de imágenes de prendas de vestir según su categoría, implementamos un modelo de ML junto con el desarrollo de una aplicación móvil. Esta última tiene como objetivo facilitarle al usuario el interactuar con el modelo entrenado mediante una interfaz sencilla.

III-A. DenseNet-121

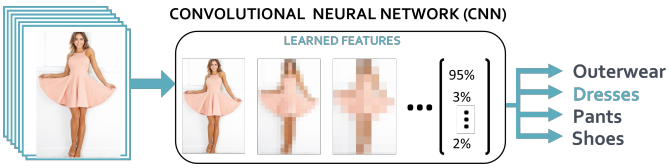

Fig. 1. Flujo de trabajo del modelo de Deep Learning. Las imágenes se envían a la Red Neuronal Convolucional (CNN), la cual es capaz de aprender las características y clasificar las prendas de forma automática.

Cuando se trabaja con CNN surgen problemas cuando se profundiza, esto se debe a que el camino para obtener información desde la capa de entrada hasta la capa de salida (y para el gradiente en la dirección opuesta) se vuelve tan grande, que puede desaparecer antes de llegar al otro lado.

Entonces, dada la relación entre el peso y la precisión de los algoritmos vista en la Figura 2, se implementó la arquitectura DenseNet-121 en el modelo propuesto, el cual simplifica el patrón de conectividad entre capas; los desarrolladores de esta arquitectura resuelven el problema asegurando el flujo máximo de información (y gradiente) y para esto conectan cada capa directamente entre sí.

[1]Estudiante de Ingeniería de Sistemas y Computación. Universidad Nacional de Colombia. Contacto: ivdgonzalezco@unal.edu.co
[2]Estudiante de Ingeniería de Sistemas y Computación. Universidad Nacional de Colombia. Contacto: dabaquerot@unal.edu.co
[3]Estudiante de Ingeniería de Sistemas y Computación. Universidad Nacional de Colombia. Contacto: jucninova@unal.edu.co

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Fig. 2. Tabla comparativa entre los algoritmos para entrenamientos de imágenes.

En lugar de obtener, un cierto poder de representación de arquitecturas extremadamente profundas o amplias, DenseNet explota el potencial de la red mediante la reutilización de características y específicamente DenseNet-121 es la arquitectura más simple entre las diseñadas sobre el conjunto de datos de ImageNet.[5]

La arquitectura de DenseNet se compone de la siguiente manera, tambien vista en la Figura 3:

- Una capa de convolución con kernel 7x7.
- Una capa de agrupación con kernel 3x3.
- Un bloque denso, el cual tiene kernels de convolución 1x1 y 3x3, estos multiplicados por 6.
- Una capa de transición, que se compone de una capa de convolución con kernel 1x1 y una capa de agrupamiento 2x2.
- Un bloque denso, el cual tiene kernels de convolución 1x1 y 3x3, estos multiplicados por 12.
- Una capa de transición, que se compone de una capa de convolución con kernel 1x1 y una capa de agrupamiento 2x2.
- Un bloque denso, el cual tiene kernels de convolución 1x1 y 3x3, estos multiplicados por 24.
- Una capa de transición, que se compone de una capa de convolución con kernel 1x1 y una capa de agrupamiento 2x2.
- Un bloque denso, el cual tiene kernels de convolución 1x1 y 3x3, estos multiplicados por 16.
- Una capa de clasificación que se compone de una capa de agrupamiento 7x7 y la conexion total de red.

III-B. RMSProps

Este es el método de optimización basado en métodos de gradiente cuadrado, el cual es una mejora de RProps, donde utiliza pesos aleatorios y propagación en reversa (*backpropagation*) para validar el acierto de cada clase, moviendo el promedio del gradiente cuadrado para mejorar la precisión del algoritmo. Para este caso, en particular se

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Fig. 3. Arquitectura de DenseNet.

utilizó el parámetro weight=None para cada una de las clases, inicializando los pesos de forma aleatoria..

III-C. Aplicación

La aplicación móvil tiene como nombre Sorter App, la cual fue desarrollada en Xamarin. Por otro lado, accede a servicios de un *backend* desarrollado en C♯ con .NET Core y desplegado en una solución de computación en la nube; a través de peticiones *http* y un protocolo de intercambio de paquetes basado en *REST*.

Adicionalmente, el modelo fue desarrollado con Python. Posteriormente, se exportó y se desplegó en un servidor a través de *Flask*, donde se expuso un servicio que recibe una imagen y retorna una cadena de texto con la clase con más alto porcetaje de acierto, según el modelo.

IV. Desarrollo

El desarrollo de la aplicación se basó en el desafío expuesto en Kaggle[4], donde se hace uso del *dataset* de imágenes dado por ellos, para obtener el modelo entrenado que sea capaz de clasificar las diferentes imágenes.

Primero, se hace la descarga y preprocesamiento de los datos, donde se usa *numpy* para crear los arreglos necesarios para empezar el particionamiento y el entrenamiento del modelo. Se realiza un particionamiento de 80 % de entrenamiento, 10 % de prueba y 10 % de validación. Se utilizan 4 clases para la clasficación de las prendas de vertir, las cuales son: shoe, dress, pants y outerwear.

Después, se utiliza el modelo para clasificación multiclase DenseNet-121 de Keras para realizar el entrenamiento. Luego, para utilizar este modelo se hace uso del one hot encoder, el cual crea un arreglo denso que será usada por el CNN.

Por otro lado, el clasificador es entrenado haciendo uso de 50 *epochs* y un *batch size* de 16. El entrenamiento de lleva a cabo usando GPU (Nvidia GeForce GTX 1080 Ti), dadas las caracteristicas que se necesitan para el funcionamiento de la CNN usando DenseNet-121.

Por último, se exporta el modelo entrenado a un archivo .h5 y se depliega a través de *Flask* un servicio basado en RESTful API, para que este pueda pueda recibir llamadas desde la aplicación móvil, la cual envia una foto y recibe una cadena de texto que le dice que tipo de prenda que posiblemente le fue enviada, como se evidencia en la Figura 4.

[4]https://www.kaggle.com/c/imaterialist-challenge-FGVC2017

Fig. 4. Evidencia de la respuesta recibida por el backend.

V. Resultados

Después del entrenamiento del modelo, la exportación, puesta en marcha del RESTful API y las pruebas realizadas a través de la aplicación móvil, se evidencia que se tiene una precisión (*accuracy*) de aproximadamente 75 %.

Adicionalmente, es necesario mencionar que el desempeño de la red neuronal, según la Figura 5, va mejorando entre más iteraciones realice, dado que se obtiene menos pérdida en el caso del entrenamiento, mientras que la validación oscila entre una perdida de 1 y 2, teniendo una perdida mínima cercana en el epoch 20.

Finalmente, gracias a que la aplicación móvil tiene que hacer uso de la RESTful API, las respuestas entre el servidor y el cliente depende de la calidad de la conexión a internet que se tenga en el dispositivo donde este instalada Inno Clothing Sorter.

VI. Conclusiones

- El uso de redes neuronales convolucionales para la clasificación de imágenes puede llegar a ser demorado, pero con el uso de GPUs se optimiza el tiempo de entrenamiento de los modelos. Sin embargo, el modelo ya entrenado puede ser exportado y el procesamiento sólo se realizará una vez.
- La articulación del modelamiento de clasificadores junto con los servicios web hace una combinación que permite acercar a un usuario final con los beneficios del ML y del aprendizaje profundo.

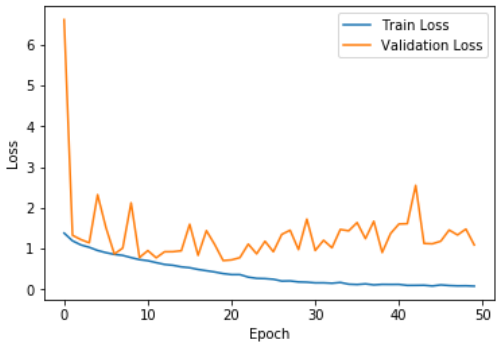

Fig. 5. Perdida en el entrenamiento y la validación utilizando redes neuronales.

- Entre más epochs se usen en la red neuronal para realizar el entrenamiento, mejores resultados de precisión y rendimiento se puede obtener del modelo entrenado, lo cual llevaría a una mejor predicción de la prenda de ropa que se suba mediante la aplicación.
- En el momento en el que se realice un modelo de ML, es necesario tener en cuenta la cantidad de información y la capacidad computacional, para encontrar un balance entre precisión y costo computacional.

References

[1] Keras Documentation (Octubre 2018), Applications. Recuperado de https://keras.io/applications/.

[2] SK Learn (Diciembre 2018), sklearn.preprocessing.OneHotEncoder. Disponible en https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.

[3] Kaggle (Noviembre 2018) Keras - DenseNet121 - Multi-Label Baseline. Recuperado de https://www.kaggle.com/phmagic/keras-densenet121-multi-label-baseline.

[4] Kaggle (Junio 2017) iMaterialist Challenge at FGVC 2017. https://www.kaggle.com/c/imaterialist-challenge-FGVC2017

[5] Understanding and visualizing DenseNets. Pablo Ruiz Ruiz. (Octubre 2018). Recuperado de https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

[6] Huang, G., Liu, Z., Maaten, L., Weinberger, K. (2018, Enero 28). Densely Connected Convolutional Networks. Disponible en https://arxiv.org/pdf/1608.06993.pdf