Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Campus Estado de México

Documentación del Modelo

Inteligencia artificial avanzada para la ciencia de datos II (TC3007C Gpo. 501)

Equipo 4:
A01751655 Cortés Olvera Gabriela
A01745865 García Gómez José Ángel
A01745096 González de la Parra Pablo
A01751580 Islas Montiel Zaide
A01745371 Sánchez Bahnsen Elisa
A01382889 Ana Martínez Barbosa
A01706870 José María Ibarra Pérez
A01745158 García Sánchez Erika Marlene

Docentes:
Víctor Adrián Sosa Hernández
Julio G. Arriaga Blumenkron
Andrea Torres Calderón
Elisabetta Cresio
Luis Ángel Trejo

03 de diciembre

1. **Introducción**
2. **Modelo**
    a. **Descripción de técnica para la generación del modelo**

La creación de un modelo de clasificación de Aprendizaje Profundo es una parte fundamental de la solución propuesta para Oxxo. La deficiencia en el proceso de acomodo de los productos, despierta la necesidad de brindarle al cliente una herramienta integral que tenga la capacidad de determinar el correcto acomodo de los productos en las tiendas dependiendo de la estrategia a implementar.

Para la construcción de este modelo se prefirió utilizar la técnica de transfer learning sobre la creación de una arquitectura compuesta desde cero dadas las complejidades que trae consigo el entrenamiento de modelos de visión computadora. El método de transfer learning consiste en utilizar un modelo que fue previamente entrenado para cierto labor y reutilizarlo para una tarea distinta [1]. La ventaja de estos modelos, que ya están públicamente disponibles, es la reducción en el tiempo de entrenamiento y la mejora en el rendimiento de los mismos puesto que aunque los pesos que lo conforman ya están entrenados con ciertos valores, estos se pueden ajustar con nuestros datos sin necesidad de modificar la arquitectura de la red.

Dicho esto, la Resnet 18 fue el modelo pre entrenado que se escogió para la clasificación de los productos en los estantes de Oxxo. El nombre de la red proviene de su arquitectura, *"Residual Neural Network"*, ya que esta fue propuesta como solución al problema del desvanecimiento del gradiente de la VGGNet, arquitectura estelar en ese entonces. Esta red fue entrenada con el famoso dataset de imagenet-1k y ganó el concurso de ILSVRC & COCO en el 2015 [3]. El bloque de aprendizaje residual es una técnica con la cual se omite el entrenamiento de ciertas capas intermedias y posteriormente estas se unen con su salida. Esto ayuda principalmente a las redes con una mayor cantidad de capas profundas ya que en caso de tener alguna capa con mal rendimiento, el proceso de regularización las omitirá con el proceso de regularización [2]. Este funcionamiento se explica en la siguiente imagen (figura 1).

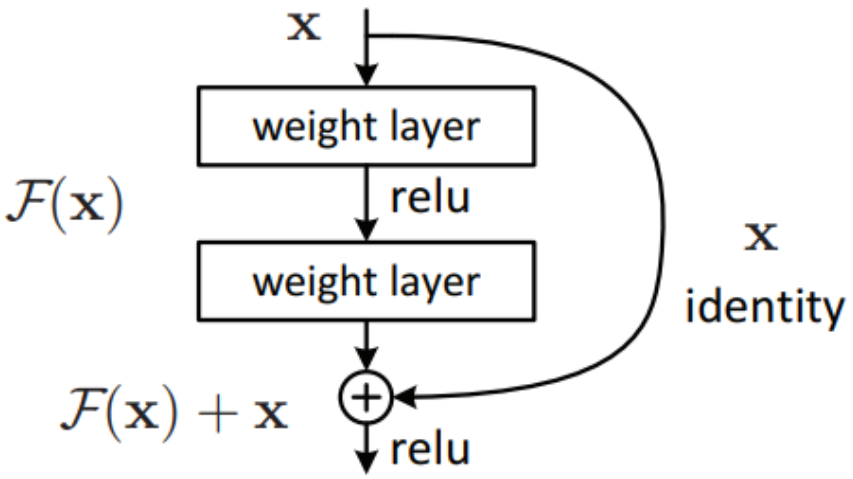

Figura 1. Bloque residual.

De forma general, esta red convolucional está conformada por 18 capas que incluyen capas convolucionales y bloques residuales que se conectan con una capa densa para la clasificación final. Esta última capa, a excepción del resto, no pertenece al modelo pre entrenado ya que es necesario entrenarlo con nuestros propios datos para el resultado final de cada clasificación. Es por ello que para el entrenamiento de esta última capa se utilizó una función Softmax con la cual se pudieran obtener las probabilidades de clasificación para cada clase al igual que un criterio de Cross Entropy y un optimizador Adam.

La estructura se puede visualizar gráficamente en el siguiente diagrama (figura 2). Además, es indispensable mencionar que así como este, la mayoría de los modelos pre entrenados requieren un preprocesamiento para así recibir imágenes con los 3 canales RGB y una altura y anchura no mayor a 224 px. También es necesario normalizar las imágenes con el promedio ([0.485, 0.456, 0.406]) y desviación estándar ([0.229, 0.224, 0.225]) dados por la documentación para así mantener los valores entre 0 y 1.

b. Tecnologías utilizadas y configuración.

Todo este procedimiento se llevó a cabo utilizando la paquetería de PyTorch pues sus funciones de procesamiento de imágenes al igual que los modelos pre cargados facilitaron el entrenamiento y permitieron lograr buenos resultados. PyTorch es una biblioteca de aprendizaje automático de código abierto para Python muy utilizada para desarrollar y entrenar modelos de aprendizaje profundo. Proporciona un grafo computacional flexible y dinámico. Usamos numpy, matplotlib, torchvision (que es una librería de PyTorch para tareas de visión computacional) y torchsummary (una extensión de PyTorch para poder visualizar el resumen del modelo).

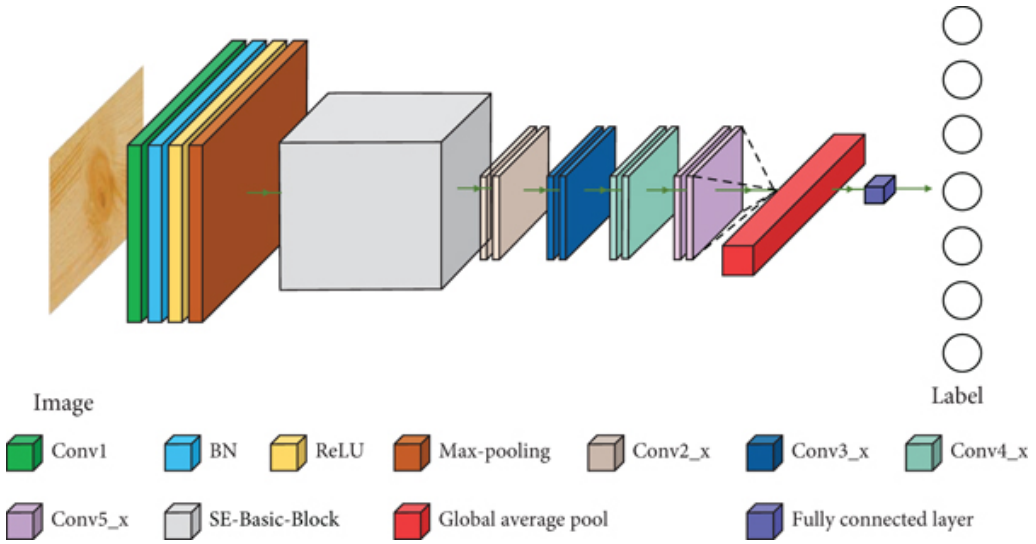

Figura 2. Arquitectura propuesta para el modelo Resnet 18.

Como ya se mencionó anteriormente una de las tecnologías que usamos es transfer learning y a través de esta se ajustaron las capas finales del modelo usando la arquitectura de ResNet-18.

Descripción y generación de bases de datos (entrenamiento, testing y validación) si aplica.

Una vez comprendida la arquitectura y elección de la Resnet 18, es importante mencionar que por cuestiones experimentales del proyecto, se entrenaron dos distintos modelos de clasificación sin realizar modificaciones en la arquitectura de los mismos. La diferencia entre estos dos modelos radica en los datos con los cuales fueron entrenados. Como bien se ha mencionado, los datos de entrenamiento constan de un set de imágenes por producto de Oxxo.

Claro está que los colores e incluso el diseño de los empaques podrían confundir al modelo y generar clasificaciones incorrectas, es por ello que antes de la selección de un modelo definitivo que incluyera imágenes de todos los productos, se hicieron múltiples pruebas con la cual se lograron identificar los productos con más clasificaciones correctas durante las pruebas. Es así como se obtuvo un modelo entrenado con imágenes de únicamente 13 productos y otro que se entrenó con las imágenes de los 36 productos. Los resultados de cada uno de estos se analizará más adelante.

Hemos capturado imágenes exhaustivas de cada uno de los productos detallados en la lista proporcionada por Oxxo. Con el objetivo de ampliar significativamente la cantidad de imágenes disponibles, hemos aplicado diversas alteraciones a dichas fotografías utilizando código especializado. Estas modificaciones abarcan desde alargamientos y recortes hasta la introducción de ruido en la parte posterior y rotaciones, entre otras técnicas. Dada la diversidad de productos necesarios para abordar las distintas variantes y ejecuciones del modelo, hemos llevado a cabo una meticulosa partición del conjunto de datos.

En el proceso de división, hemos reservado un conjunto de datos específico destinado exclusivamente al entrenamiento del modelo. Para ello, hemos seleccionado de manera aleatoria el 70% de los datos disponibles. Adicionalmente, hemos asignado el 15% del conjunto de datos para evaluar el rendimiento del modelo durante las pruebas y otro 15% para su validación. Esta estrategia de partición nos permite asegurar una representación adecuada de los datos en cada fase del desarrollo del modelo.

En nuestro enfoque iterativo para el desarrollo del modelo, hemos optado por comenzar con un conjunto inicial de 8 productos. Esta decisión nos ha permitido garantizar el buen funcionamiento del modelo incluso con una cantidad limitada de productos, antes de expandir la gama de productos involucrados en las futuras fases del proyecto.Poco a poco fuimos aumentando el número de productos que usamos para los modelos. Decidimos empezar con 9 productos, posteriormente lo aumentamos a 13, después decidimos hacer un modelo con todas las papas y al último para los 32 productos.

c. Evaluación e interpretación del performance del modelo

En la evaluación del rendimiento del modelo, se implementó un proceso de entrenamiento que abarca varias épocas. Se utilizaron métricas clave, como la pérdida (loss) y la precisión (accuracy), tanto en el conjunto de entrenamiento como en el de validación, para

evaluar la capacidad del modelo para generalizar datos no vistos. Durante la fase de entrenamiento, se llevó a cabo un pase hacia adelante, el cálculo de la pérdida y la retropropagación para ajustar los pesos del modelo.

La fase de validación proporcionó información crucial sobre la capacidad del modelo para generalizar datos no utilizados en el entrenamiento. Además, se implementó una técnica de "early stopping" para prevenir el sobreajuste, deteniendo el entrenamiento si la pérdida en el conjunto de validación dejaba de disminuir. Los resultados se monitorean a lo largo de las épocas, destacando cualquier tendencia o cambio en las métricas evaluadas. La interpretación de estos resultados proporcionó insights sobre el desempeño del modelo y, en caso de activación del "early stopping", señaló el momento en que se detuvo el entrenamiento para evitar sobreajuste.

La tabla 1 muestra los resultados obtenidos en las métricas de los modelos con el fin de la comparación de los mismos.

Modelo	Training	Validation
9 Productos	`loss: 0.0001647383364162012, accuracy: 0.9967697277341947`	`loss: 0.000575493242398932, accuracy: 0.9870689655172413`
13 Productos	`loss: 0.002543199435521087, accuracy: 0.9618892508143323`	`loss: 0.003010084775437836, accuracy: 0.952887537993921`

Tabla 1. Comparación de métricas del modelo

Como evidencia de los resultados obtenidos, los modelos entrenados exhibieron un rendimiento positivo en relación con las métricas previamente mencionadas. No obstante, se optó por considerar como referencia el modelo que abarcaba la totalidad de los productos de papas. Esta decisión se basó en la mayor amplitud de productos contenidos en dicho modelo y en los resultados exitosos que había demostrado en la aplicación. La elección se respaldó en la premisa de que la inclusión de un conjunto más amplio de productos contribuía con mayores herramientas a la aplicación.

Referencias

[1] Brownlee, J. (19 de septiembre de 2019). *A Gentle Introduction to Transfer Learning for Deep Learning*. Machine Learning Mastery. Recuperado de: https://machinelearningmastery.com/transfer-learning-for-deep-learning/
[2] Geeksforgeeks. (10 de enero de 2023). Residual Networks (ResNet) – Deep Learning. Recuperado de: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

[3] Hugging Face. (25 de septiembre de 2023). *ResNet*. Recuperado de: https://huggingface.co/microsoft/resnet-18