AlexNet challenge

Felipe Andres Castillo
email: 3.1416.p@ciencias.unam.mx
Diplomado Inteligencia Artificial y Ciencia de Datos

7 de marzo de 2025

Resumen

Utilizando el dataset CIFAR-10, se entrenaron y compararon modelos neuronales basados en AlexNet implementados en PyTorch y TensorFlow. Se evaluaron tres enfoques distintos: el primero empleó directamente las imágenes del dataset (32x32), lo que requirió ajustar algunos parámetros del modelo original; el segundo utilizó la arquitectura original de AlexNet, con imágenes redimensionadas a 224x224; y el tercero correspondió a un modelo preentrenado. Como era de esperar, el modelo preentrenado obtuvo los mejores resultados en la fase de prueba. Sin embargo, tanto su entrenamiento como el del modelo sin preentrenamiento fueron significativamente más largos en comparación con el modelo ajustado para imágenes de 32x32.

1. Introducción

AlexNet es una arquitectura de red neuronal convolucional (CNN) diseñada por Alex Krizhevsky en colaboración con Ilya Sutskever y Geoffrey Hinton para el concurso de ImageNet en 2012.

Su estructura está compuesta por cinco capas convolucionales, de las cuales la primera, la cuarta y la quinta están seguidas por una capa de max-pooling. Tras estas capas, se añaden tres capas densas, donde la última cuenta con 1.000 neuronas de salida responsables de la clasificación de las imágenes. Todas las neuronas, excepto las de la capa de salida, utilizan la función de activación rectificadora (ReLU).

AlexNet marcó un hito en el campo del aprendizaje profundo, al demostrar la eficacia de las redes convolucionales para la clasificación de imágenes y sus diversas aplicaciones. Además, evidenció que los modelos podían entrenarse de manera eficiente en GPU, impulsando así el desarrollo de la computación en paralelo para tareas de visión por computadora.

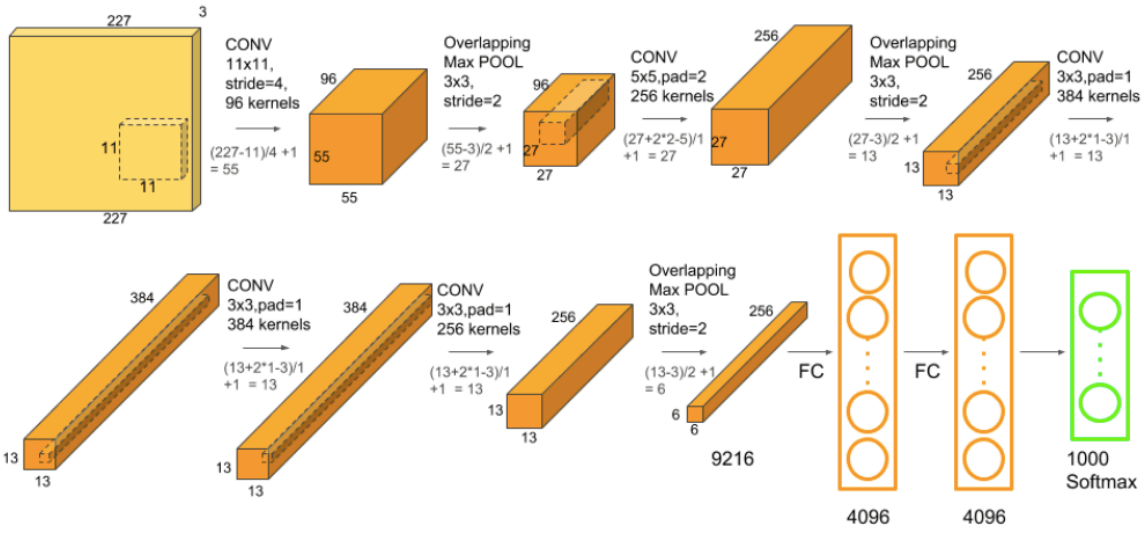

Figura 1: Arquitectura del modelo AlexNet[1]

2. Metodología

2.1. Los datos

El conjunto de datos CIFAR-10 consta de 60 000 imágenes en color de 32 x 32 en 10 clases, con 6000 imágenes por clase. Hay 50 000 imágenes de entrenamiento y 10 000 imágenes de prueba.

El conjunto de datos se divide en cinco lotes de entrenamiento y un lote de prueba, cada uno con 10 000 imágenes. El lote de prueba contiene exactamente 1000 imágenes seleccionadas aleatoriamente de cada clase. Los lotes de entrenamiento contienen las imágenes restantes en orden aleatorio, pero algunos lotes de entrenamiento pueden contener más imágenes de una clase que de otra. Entre ellos, los lotes de entrenamiento contienen exactamente 5000 imágenes de cada clase.

Figura 2

2.2. Los modelos

Para este ejercicio, se ejecutaron tres modelos distintos en PyTorch y TensorFlow:

- **AlexNet**: Consta de cinco capas convolucionales, tres capas densas, así como capas de max pooling y dropout. Requiere que las imágenes de entrada tengan un tamaño de 224x224. Este modelo no fue importado de ninguna biblioteca, sino que se definió explícitamente.

- **AlexNet modificado**: Dado que las imágenes del dataset tienen un tamaño de 32x32, se ajustaron los valores de entrada y salida de las capas densas y se eliminó el stride = 4 de la primera capa convolucional para adaptarlo a esta resolución.

- **AlexNet preentrenado**: Se utilizó la versión preentrenada disponible en las bibliotecas torchvision.models y keras.models.

Para comparar el desempeño de los modelos, todos fueron entrenados durante 15 épocas con *batches* de 64 elementos. Como función de pérdida, se empleó *CrossEntropy* en PyTorch y *Categorical CrossEntropy* en TensorFlow. La función de optimización utilizada fue *Adam*, con una tasa de aprendizaje de 0.001.

Los diagramas de los modelos se presentan en la figura 3.

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 22, 22, 96)	34,944
zero_padding2d_2 (ZeroPadding2D)	(None, 26, 26, 96)	0
max_pooling2d_6 (MaxPooling2D)	(None, 12, 12, 96)	0
conv2d_11 (Conv2D)	(None, 12, 12, 256)	614,656
max_pooling2d_7 (MaxPooling2D)	(None, 5, 5, 256)	0
conv2d_12 (Conv2D)	(None, 5, 5, 384)	885,120
conv2d_13 (Conv2D)	(None, 5, 5, 384)	1,327,488
conv2d_14 (Conv2D)	(None, 5, 5, 256)	884,992
max_pooling2d_8 (MaxPooling2D)	(None, 2, 2, 256)	0
flatten_2 (Flatten)	(None, 1024)	0
dense_6 (Dense)	(None, 512)	524,800
dropout_4 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 512)	262,656
dropout_5 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 10)	5,130

Total params: 4,539,786 (17.32 MB)
Trainable params: 4,539,786 (17.32 MB)
Non-trainable params: 0 (0.00 B)

(a) AlexNet (32x32)

Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 224, 224, 3)	0
conv2d_5 (Conv2D)	(None, 54, 54, 96)	34,944
zero_padding2d_1 (ZeroPadding2D)	(None, 58, 58, 96)	0
max_pooling2d_3 (MaxPooling2D)	(None, 28, 28, 96)	0
conv2d_6 (Conv2D)	(None, 28, 28, 256)	614,656
max_pooling2d_4 (MaxPooling2D)	(None, 13, 13, 256)	0
conv2d_7 (Conv2D)	(None, 13, 13, 384)	885,120
conv2d_8 (Conv2D)	(None, 13, 13, 384)	1,327,488
conv2d_9 (Conv2D)	(None, 13, 13, 256)	884,992
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten_1 (Flatten)	(None, 9216)	0
dense_3 (Dense)	(None, 4096)	37,752,832
dropout_2 (Dropout)	(None, 4096)	0
dense_4 (Dense)	(None, 4096)	16,781,312
dropout_3 (Dropout)	(None, 4096)	0
dense_5 (Dense)	(None, 10)	40,970

Total params: 58,322,314 (222.48 MB)
Trainable params: 58,322,314 (222.48 MB)
Non-trainable params: 0 (0.00 B)

(b) AlexNet (224x224)

Figura 3: Esquema de los modelos AlexNet utilizados.

3. Resultados y análisis

PyTorch				
Modelo	Parámetros	Tiempo de entrenamiento	Accuracy	Loss
AlexNet mod.	4,539,786	471.11 segundos	0.6370	1.052892
AlexNet	58,322,314	2810.18 segundos	0.6661	1.013621
AlexNet pre-ent.	57,044,810	2651.97 segundos	0.7809	0.689839
TensorFlow				
AlexNet mod.	4,539,786	266.15 segundos	0.6293	1.029841
AlexNet	58,322,314	1152.28 segundos	0.6651	1.019877

Tabla 1: Resultados de cada modelo para el conjunto de prueba.

Los resultados muestran que el modelo preentrenado obtuvo la mayor precisión y la menor pérdida, con un tiempo de entrenamiento similar al del modelo sin preentrenamiento. Esto se debe a que ambos comparten un número de parámetros similar, lo que resalta los beneficios del aprendizaje por transferencia.

En cuanto al modelo AlexNet (224x224), que replicaba la arquitectura original sin preentrenamiento, su desempeño se ubicó en un punto intermedio entre los otros dos modelos, sin diferencias significativas. Sin embargo, un aspecto interesante es la comparación entre PyTorch y TensorFlow, ya que ambos lograron valores de *accuracy* y *loss* casi idénticos, pero con una diferencia sustancial en el tiempo de ejecución: en TensorFlow, el entrenamiento fue más del doble de rápido. Determinar la causa exacta de esta diferencia es complejo, ya que puede estar relacionada con factores de optimización de los algoritmos o con el funcionamiento interno del backend de cada API.

Por último, el modelo de AlexNet modificado para trabajar directamente con imágenes de 32x32 obtuvo los valores más bajos de precisión y mayor pérdida, aunque no muy diferentes a los del modelo sin preentrenamiento. No obstante, su tiempo de entrenamiento fue considerablemente menor, lo que abre la posibilidad de aumentar el número de épocas con el objetivo de mejorar los resultados sin un incremento significativo en el tiempo de ejecución.

4. Conclusiones

Este desafío deja dos conclusiones importantes. En primer lugar, resalta la utilidad del aprendizaje por transferencia, ya que permite mejorar significativamente los resultados de predicción sin aumentar el tiempo de entrenamiento ni el consumo de recursos computacionales.

En segundo lugar, evidencia la importancia del uso de la GPU para entrenar modelos de este tipo. Inicialmente, se intentó procesar los datos utilizando la CPU, pero el entrenamiento tuvo que ser interrumpido debido a su extrema lentitud, con un progreso mínimo tras varios minutos de ejecución. Esto permite comprender por qué AlexNet representó un avance tan significativo en el deep learning, al demostrar la viabilidad del entrenamiento de redes neuronales en GPU.

Referencias

[1] Sutskever I. Hinton G. E. Krizhevsky A. *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems, 2012.