Modelo AlexNet

Arquitectura profunda

Se utilizó una red profunda con ocho capas, mucho más profunda que los modelos anteriores, lo que contribuyó a los avances en las arquitecturas de CNN.

Uso del GPU

Se aprovecharon las GPUs para acelerar el entrenamiento, mejorando significativamente el rendimiento y la eficiencia en el procesamiento de grandes conjuntos de datos.

Técnicas innovadoras

Activación ReLU: Se emplearon Unidades Lineales Rectificadas (ReLU) para un entrenamiento más rápido, un componente esencial en la optimización del aprendizaje basado en gradientes

Dropout: Se previno el sobreajuste eliminando aleatoriamente neuronas durante el entrenamiento, mejorando la robustez del modelo

Aumento de datos: Se mejoró la generalización del modelo mediante técnicas como traducciones y reflejos de imágenes, cruciales para un preprocesamiento de datos eficaz.

Datos a gran escala: Entrenado en el extenso conjunto de datos ImageNet, que contiene millones de imágenes, demostrando la importancia de conjuntos de datos extensos y diversos en el aprendizaje automático

Architectura

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-

Capas completamente conectadas y de dropout

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-
Fully Connected 1	-	-	-	-	4096	ReLU
Dropout 2	rate = 0.5	-	-	-	4096	-
Fully Connected 2	-	-	-	-	4096	ReLU
Fully Connected 3	-	-	-	-	1000	Softmax

Resumen

- Tiene 8 capas con parámetros entrenables.
- La entrada al modelo son imágenes RGB.
- Tiene 5 capas de convolución combinadas con capas de max-pooling.
- Luego tiene 3 capas completamente conectadas.
- La función de activación utilizada en todas las capas es ReLU.
- Se utilizaron dos capas de Dropout.
- La función de activación utilizada en la capa de salida es Softmax.
- El número total de parámetros en esta arquitectura es de 62.3 millones.

Inception V1

Hay cuatro canales paralelos en cada módulo Inception, y se realiza la concatenación al final del canal.

La convolución 1x1 se utiliza principalmente para reducir las dimensiones en el artículo y evitar cuellos de botella en los cálculos. También añade una pérdida softmax adicional a algunas ramas de la capa de red anterior para evitar el problema de la desaparición del gradiente.

Canales paralelos:

Convolución 1x1: Tomado de [Network in Network], el mapa de características de entrada puede reducirse en dimensiones y mejorarse sin perder demasiada información espacial de la entrada.

Convolución 1x1 seguida de convolución 3x3: La convolución 3x3 aumenta el campo receptivo del mapa de características y cambia la dimensión a través de la convolución 1x1.

Convolución 1x1 seguida de convolución 5x5: La convolución 5x5 aumenta aún más el campo receptivo del mapa de características y cambia las dimensiones a través de la convolución 1x1.

Max pooling 3x3 seguido de convolución 1x1: El autor considera que, aunque la capa de pooling perderá información espacial, ha sido aplicada con éxito en muchos campos, lo que

demuestra su efectividad. Por eso se añade un canal paralelo, y se cambia su dimensión de salida mediante la convolución 1x1

Desventajas

Los modelos más grandes que utilizan InceptionNet tienden a sobreajustarse, especialmente cuando hay un número limitado de muestras etiquetadas. El modelo puede sesgarse hacia ciertas clases que tienen un mayor volumen de etiquetas en comparación con otras.

Otra desventaja es que aumentar uniformemente el tamaño de la red incrementará el uso de recursos computacionales. Por ejemplo, en una red profunda, si dos convoluciones están encadenadas, cualquier mejora unificada de sus núcleos de convolución provocará una mayor demanda de recursos.

Inception V3

Esta arquitectura se enfoca en cómo utilizar núcleos de convolución de dos o más tamaños más pequeños para reemplazar otros, e introduce capas asimétricas; es decir, también se ha propuesto una convolución dimensional para la capa de pooling como una solución para la pérdida de información espacial. Existen ideas como el 'label-smoothing', BN-auxiliary, entre otras.

El costo computacional se reduce mientras se mejora la precisión de la red.

Principios Generales de Diseño

- **Evitar cuellos de botella**: Se debe evitar que una gran cantidad de características se comprima en las capas intermedias de la red, ya que esto puede causar pérdida de información.
- **Alta dimensionalidad:** Las características con una mayor dimensionalidad suelen converger más rápidamente durante el entrenamiento.
- **Reducción de dimensionalidad: Se** puede reducir la cantidad de cálculos necesarios mediante la reducción de la dimensionalidad de las características.
- **Equilibrio entre profundidad y ancho:** Para maximizar el rendimiento de la red, es importante aumentar tanto la profundidad como el ancho de manera proporcional.

Arquitectura

El modelo Inception V3 fue lanzado en el año 2015, tiene un total de 42 capas y una tasa de error más baja que sus predecesores. Veamos las diferentes optimizaciones que hacen que el modelo Inception V3 sea mejor.

Las principales modificaciones realizadas en el modelo Inception V3 son:

- Factorización en convoluciones más pequeñas
- Factorización espacial en convoluciones asimétricas
- Utilidad de clasificadores auxiliares
- Reducción eficiente del tamaño de la cuadrícula

¿Qué hace mejor el V3 que el V1?
Inception V3 es simplemente la versión avanzada y optimizada del modelo Inception V1. El modelo Inception V3 utilizó varias técnicas para optimizar la red y mejorar la adaptación del modelo.

- Tiene mayor eficiencia.
- Tiene una red más profunda en comparación con los modelos Inception V1 y V2, pero su velocidad no se ve comprometida.
- Es computacionalmente menos costoso.
- Utiliza clasificadores auxiliares como regularizadores.

Inception V4

Inception v4 es una arquitectura de red neuronal convolucional que se basa en iteraciones previas de la familia Inception, simplificando la arquitectura y utilizando más módulos Inception que Inception v3.

Arquitectura

- El conjunto inicial de capas, al que el artículo se refiere como el 'tallo de la arquitectura', fue modificado para hacerlo más uniforme. Estas capas se utilizan antes del bloque Inception en la arquitectura.
- Este modelo puede entrenarse sin la partición de réplicas, a diferencia de las versiones anteriores de Inception, que requerían diferentes réplicas para ajustarse a la memoria. Esta arquitectura utiliza optimización de memoria en la retropropagación para reducir el requerimiento de memoria.

¿Qué hace mejor el V4 que el V1?

- **Diseño más uniforme:** Inception V4 tiene una estructura más homogénea y simplificada en comparación con V3.
- **Bloques Inception-ResNet:** Incorpora módulos que combinan las ideas de Inception con conexiones residuales, lo que permite entrenar redes más profundas y mejorar el flujo de gradientes.
- **Mayor profundidad:** V4 es generalmente más profunda que V3, lo que potencialmente permite aprender representaciones más complejas.
- **Rendimiento mejorado:** En varios benchmarks, Inception V4 ha demostrado obtener una precisión ligeramente superior a Inception V3.
- **Eficiencia computacional:** A pesar de ser más profunda, V4 está diseñada para ser computacionalmente eficiente.
- **Mejor manejo de características en múltiples escalas:** La arquitectura refinada permite un mejor procesamiento de características a diferentes escalas en la imagen.

VGG 19

El modelo VGG19 (también conocido como VGGNet-19) tiene la misma idea básica que el modelo VGG16, con la excepción de que soporta 19 capas. Los números '16' y '19' se refieren a las capas de peso del modelo (capas convolucionales).

Arquitectura

Inputs: VGGNet acepta imágenes de 224x224 píxeles como entrada.

Convolutional Layers: Las capas convolucionales de VGG utilizan el campo receptivo más pequeño posible, de 3x3, para captar movimientos de izquierda a derecha y de arriba a abajo. Además, se utilizan filtros de convolución de 1x1 para transformar la entrada de manera lineal.

Hidden Layers: Las capas ocultas de la red VGG utilizan ReLU. La Normalización de Respuesta Local (LRN) no se suele usar en VGG, ya que aumenta el uso de memoria y el tiempo de entrenamiento

Fully Connected Layers: VGGNet contiene tres capas con conectividad total. Las dos primeras capas tienen 4096 canales cada una, mientras que la tercera capa tiene 1000 canales, con un canal para cada clase

Ventajas

Tiene un avance significativo respecto a AlexNet que es el componente es una unidad ReLU, que reduce el tiempo de entrenamiento.

Modelo ResNet- 50

Arquitectura

- Capa de entrada: Imagen de entrada
- Bloque inicial:
    - Convolución 7x7, 64 filtros
    - BatchNorm
    - ReLU
    - MaxPool 3x3
- 4 etapas de bloques residuales (Bottleneck blocks):
    - Etapa 1: 3 bloques, 64 filtros
    - Etapa 2: 4 bloques, 128 filtros
    - Etapa 3: 6 bloques, 256 filtros
    - Etapa 4: 3 bloques, 512 filtros
- Global Average Pooling
- Fully Connected de 1000 unidades con Softmax

Modelo ResNetv2-50

Es una red neuronal convolucional con 50 capas. La base de datos ImageNet contiene una versión preentrenada de la red que ha sido entrenada con más de un millón de fotos. La red preentrenada puede categorizar imágenes en 1000 categorías de objetos diferentes, incluyendo varios animales, un teclado, un ratón y un lápiz.

Arquitectura

- Capa de entrada: Imagen de entrada

- Bloque inicial:
    - Convolución 7x7, 64 filtros
    - BatchNorm
    - ReLU
    - MaxPool 3x3
- 4 etapas de bloques residuales (Bottleneck blocks):
    - Etapa 1: 3 bloques, 64 filtros
    - Etapa 2: 4 bloques, 128 filtros
    - Etapa 3: 6 bloques, 256 filtros
    - Etapa 4: 3 bloques, 512 filtros
- BatchNorm
- ReLU
- Global Average Pooling
- Fully Connected de 1000 unidades con Softmax

Ventajas

Conexiones residuales: Permiten el entrenamiento de redes más profundas al mitigar el problema del desvanecimiento del gradiente

Profundidad: Con 50 capas, puede aprender representaciones complejas manteniendo un buen equilibrio entre profundidad y eficiencia computacional

Rendimiento: Ofrece una precisión de clasificación de imágenes muy alta con un coste computacional razonable.

Flexibilidad: Puede ser fácilmente adaptada para diferentes tamaños de entrada y número de clases de salida.

¿Qué hace mejor el v2 que el original?

El orden de las capas ya que la v2 utiliza pre-activación, un entrenamiento más estable y mejor rendimiento.

CLIP ResNet 50

Se caracteriza por el uso de las conexiones residuales

Arquitectura

Se basa principalmente en que es más fácil aprender una función residual que la función original y tiene como componentes:

Bloques residuales

Capas convolucionales

Capas de agrupación

Capas completamente conectadas

Ventajas

Mayor profundidad

Mayor rendimiento

Fácil entrenamiento

Desventajas

Debido a que puede llegar a tener muchos parámetros, el recurso computacional que requiere es alto.

Al igual que otras redes neuronales profundas, puede llegar a sobreajustarse si no se utilizan las técnicas de regularización adecuadas.

CLIP ResNet 101

Es un modelo capaz de realizar diversas tareas, entre ellas, la clasificación de imágenes.

Arquitectura

Tiene dos componentes principales:

Contiene 101 capas que extraen características visuales de las imágenes, por lo tanto, permite capturar patrones visuales complejos y discriminativos. También está el Encoder de texto.

Ventajas

Flexibilidad

Precisión

Deventajas

Puede ser computacionalmente costoso y puede ser sensible a pequeñas perturbaciones en las imágenes.

Tabla de comparación de modelos

Modelo	Detección de características finas	Rendimiento	Flexibilidad	Costo computacional
AlexNet				
Inception v1				
Inception v3				
Inception v4				
VGG 19				
ResNet-50				
ResNet50v2				
CLIP ResNet 50				

CLIP ResNet 101				
CLIP ResNet 50 4x				
CLIP ResNet 50 16x				
CLIP ResNet 50 V0				
MobileNetV2				

Tensorflow callbacks

Cuando hablamos de “callbacks” nos referimos a las clases y métodos que permiten realizar acciones en diferentes momentos durante el entrenamiento de un modelo como:
on_batch_begin: Se ejecuta al inicio de cada (batch) durante el entrenamiento.

on_batch_end: Se ejecuta al final de cada lote durante el entrenamiento.

on_epoch_begin: Se ejecuta al inicio de cada época durante el entrenamiento.

Args	
`epoch`	Integer, index of epoch.
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_epoch_end: Se ejecuta al final de cada época durante el entrenamiento.

Args	
`epoch`	Integer, index of epoch.
`logs`	Dict, metric results for this training epoch, and for the validation epoch if validation is performed. Validation result keys are prefixed with `val_`. For training epoch, the values of the `Model`'s metrics are returned. Example: `{'loss': 0.2, 'accuracy': 0.7}`.

on_predict_batch_begin: Se ejecuta al inicio de cada batch durante la fase de predicción

Args	
`batch`	Integer, index of batch within the current epoch.
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_predict_batch_end: Se ejecuta al final de cada batch durante la fase de predicción.

Args	
`batch`	Integer, index of batch within the current epoch.
`logs`	Dict. Aggregated metric results up until this batch.

on_predict_begin: Se ejecuta al inicio del proceso de predicción.

Args	
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_predict_end: Se ejecuta al final del proceso de predicción.

Args	
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_test_batch_begin: Se ejecuta al inicio de cada batch durante la evaluación del modelo.

Args	
`batch`	Integer, index of batch within the current epoch.
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_test_batch_end: Se ejecuta al final de cada batch durante la evaluación del modelo.

Args	
`batch`	Integer, index of batch within the current epoch.
`logs`	Dict. Aggregated metric results up until this batch.

on_test_begin: Se ejecuta al inicio de la evaluación del modelo.

Args	
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_test_end: Se ejecuta al final de la evaluación del modelo.

Args	
`logs`	Dict. Currently the output of the last call to `on_test_batch_end()` is passed to this argument for this method but that may change in the future.

on_train_batch_begin: Se ejecuta al inicio de cada batch durante el entrenamiento

Args	
`batch`	Integer, index of batch within the current epoch.
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_train_batch_end: Se ejecuta al final de cada batch durante el entrenamiento.

Args	
`batch`	Integer, index of batch within the current epoch.
`logs`	Dict. Aggregated metric results up until this batch.

on_train_begin: Se ejecuta al inicio del proceso de entrenamiento.

Args	
`logs`	Dict. Currently no data is passed to this argument for this method but that may change in the future.

on_train_end: Se ejecuta al final del proceso de entrenamiento.

Args	
`logs`	Dict. Currently the output of the last call to `on_epoch_end()` is passed to this argument for this method but that may change in the future.

set_model: Configura el modelo dentro del callback

set_params: Configura los parámetros del callback (como épocas, pasos, etc.).

MobileNetV2

Es una ligera red neuronal convolucional que está específicamente diseñado para aplicaciones de visión en dispositivos móviles y embebidos. Otro aspecto notable de este modelo es su capacidad para lograr un buen equilibrio entre el tamaño del modelo y la precisión, lo que lo hace ideal para dispositivos con recursos limitados.

Características principales

Este modelo incorpora varias características clave que contribuyen a su eficiencia y efectividad en tareas de clasificación de imágenes como la convolución separable en profundidad, residuos invertidos, diseño de cuellos de botella, cuellos de botella lineales y bloques de squeeze-and-excitation.

Arquitectura

Convolución Separable en Profundidad

Separa la convolución estándar en dos operaciones independientes: convolución en profundidad y convolución puntual, de esta manera, se reduce el costo computacional de la convolución

Residuos Invertidos

Ayuda a mejorar la precisión del modelo pues introducen una estructura de cuello de botella que expande el número de canales antes de aplicar las convoluciones separables en profundidad.

Diseño de Cuello de Botella

Reduce aún más el costo computacional utilizando convoluciones de 1×1 para reducir el número de canales antes de aplicar convoluciones separables en profundidad.

Cuellos de Botella Lineales

Al usar activaciones lineales en lugar de activaciones no lineales, el modelo preserva más información y mejora su capacidad para capturar detalles precisos.

Bloques de Squeeze-and-Excitation

Estos bloques recalibran de manera adaptativa las respuestas de las características por canal, lo que permite que el modelo se enfoque en características más informativas y suprima aquellas menos relevantes."

¿Por qué usar este modelo?

Ofrece varias ventajas pues tiene una arquitectura ligera que permite una implementación eficiente en dispositivos con un recurso computacional limitado, también gracias a su arquitectura logra una precisión competitiva en comparación con otros modelos que son más grandes y costosos computacionalmente, por último, debido a que el modelo tiene un tamaño pequeño, permite tiempos de inferencia más rápidos.