Prácticas: CIFAR-10 con WideResNet y DenseNet

Ana Valentina López Chacón

Visión por Computadora
MIARFID, UPV

Mayo, 2025

1. Objetivos

Entrenar y evaluar arquitecturas WideResNet y DenseNet para clasificación de imágenes en el conjunto de datos CIFAR-10 [1]. El enfoque principal es comparar el rendimiento de diferentes configuraciones de WideResNet, y posteriormente realizar el mismo análisis para DenseNet, reportar los mejores valores de precisión.

2. Dataset y Preprocesamiento

Se utilizó el conjunto CIFAR-10 [1], compuesto por 60,000 imágenes en color de tamaño 32×32 divididas en 10 clases, con 50,000 imágenes para entrenamiento y 10,000 para prueba. Con el fin de mejorar la generalización, se aplicaron las siguientes técnicas de aumento de datos:

- `RandomHorizontalFlip`
- `RandomAffine` (rotación de hasta 15°, traslación y escalado)
- `RandomResizedCrop` con escala (0.8, 1.0)
- Normalización mediante `ToTensor`

3. Configuración de Experimentos

Se fijaron los siguientes hiperparámetros en todos los experimentos, tras una búsqueda previa realizada exclusivamente sobre arquitecturas ResNet:

- Optimizador: Adam con tasa de aprendizaje de 0.001
- Regularización L2 (weight decay): 1^{-6}
- Tamaño del batch: 100

- Número de épocas: 100
- Scheduler: `ReduceLROnPlateau` con factor 0.1, paciencia 10, LR mínimo de 1^{-5}
- Función de pérdida: Entropía cruzada (CrossEntropy)

Todos los modelos se entrenaron con la GPU de Kaggle y se fijaron semillas para garantizar reproducibilidad.

4. WideResNet

WideResNet es una variante moderna de las redes residuales (ResNet) que busca mejorar el rendimiento al aumentar el ancho de la red en lugar de su profundidad. La arquitectura mantiene el principio fundamental de las conexiones residuales, pero con menos bloques más anchos, lo que facilita el entrenamiento y mejora la eficiencia del modelo. La arquitectura se compone de cuatro etapas principales:

- **Convolución Inicial:** una convolución 3×3 con 16 filtros y padding=1 que preserva el tamaño de la imagen de entrada (32×32).
- **Bloques Residuales:** existen tres bloques residuales, cada uno compuesto por n bloques `WideResidualBlock`, donde $n = \frac{\texttt{depth}-4}{6}$.
- **WideResidualBlock:** cada uno de estos bloques contiene:
  - BatchNorm + ReLU
  - Convolución 3×3
  - BatchNorm + ReLU
  - Convolución 3×3
  - (si aplica) conexión corta con convolución 1×1 para igualar dimensiones
- **Clasificación final:**
  - BatchNorm final + ReLU
  - Pooling global promedio (kernel 8×8)
  - Capa lineal de salida (fully connected) para 10 clases

La cantidad de filtros por etapa está controlada por el `widen_factor` (k):

Etapa	Número de filtros
Inicial	16
Bloque 1	$16 \times k$
Bloque 2	$32 \times k$
Bloque 3	$64 \times k$

5. DenseNet

DenseNet (Dense Convolutional Network) es una arquitectura que introduce conexiones densas entre capas, donde cada capa recibe como entrada los mapas de características de todas las capas anteriores. Esto permite una reutilización eficiente de características, menor número de parámetros y mejores gradientes durante el entrenamiento. La arquitectura se compone de cuatro etapas principales:

- **Convolución Inicial:** una convolución 3×3 que proyecta las imágenes RGB a un espacio de 2× `growth_rate` canales (típicamente 24 o 32).
- **Bloques Densos:** cada bloque está formado por múltiples capas `Bottleneck`, que consisten en:
  - BatchNorm + ReLU
  - Convolución 1×1 (compresión interna)
  - BatchNorm + ReLU
  - Convolución 3×3 (expansión con el `growth_rate` como número canales)
  - Concatenación con la entrada original
- **Bloques de Transición:** aplicados entre bloques densos, reducen dimensionalidad y resolución espacial:
  - BatchNorm
  - Convolución 1×1
  - Pooling promedio 2×2
- **Clasificación final:**
  - BatchNorm + ReLU
  - Pooling global promedio (adaptativo)
  - Capa lineal de salida (fully connected) para 10 clases

Los parámetros que se varían en esta arquitectura son:

- `nblocks`: número de capas `Bottleneck` por bloque denso
- `growth_rate`: número de nuevos canales añadidos por cada capa
- `reduction`: factor de compresión en bloques de transición (típicamente 0.5)

6. Resultados

Se evaluaron diferentes configuraciones de `WideResNet` variando la profundidad, entre 16, 22 y 28, y el ancho dado por el `widen_factor` (k), tomando valores de 4, 6 y 8. Todas las redes fueron entrenadas durante 100 épocas bajo la misma configuración experimental.

Modelo	Depth	k	Parámetros	Acc en Test
WideResNet-16-4	16	4	2,748,890	92.72 %
WideResNet-22-4	22	4	4,298,970	93.31 %
WideResNet-22-6	22	6	9,658,458	93.5 %
WideResNet-28-4	28	4	5,849,050	93.41 %
WideResNet-28-6	28	6	13,144,794	93.89 %
WideResNet-28-8	28	8	23,354,842	**94.1 %**

Se evaluaron tres configuraciones diferentes de DenseNet variando el número de capas por bloque y la tasa de crecimiento de características. Todos los modelos utilizaron una reducción de 0.5 y fueron entrenados con la misma política de entrenamiento que WideResNet, salvo un cambio en el tamaño de batch a 64.

Modelo	`nblocks`	`growth_rate`	Parámetros	Acc en Test
DenseNet-BC1	[6, 8, 8]	12	232,090	89.38 %
DenseNet-BC2	[8, 12, 12, 8]	12	542,692	91.94 %
DenseNet-BC3	[6, 12, 24, 16]	24	3,930,826	**93.04 %**

Los resultados muestran que al incrementar tanto la profundidad (número de capas por bloque) como la tasa de crecimiento, el modelo mejora su precisión en test.

7. Conclusiones

Para finalizar este informe podemos destacar las siguientes conclusiones:

- Se observa que aumentar la profundidad y el factor de ensanchamiento mejora sistemáticamente el rendimiento, siendo la configuración WideResNet-28-8 la que alcanza la mejor precisión. No obstante, esto conlleva un incremento significativo en el número de parámetros, por lo que existe un compromiso entre rendimiento y eficiencia.
- La DenseNet utiliza concatenaciones en lugar de sumas como la arquitectura ResNet, por lo que tiene menos parámetros a igual nivel de profundidad, lo cual mejora la eficiencia de uso de características y gradientes. El mejor resultado se obtiene en el experimento DenseNet-BC3 siendo competitivos respecto a WideResNet con menor tiempo de ejecución.

Referencias

[1] Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images.*