MicroProyecto 4

Equipo: 20

Dataset: Clasificación de Aviones

Integrantes:

- Leonard David Vivas Dallos - 1070947936
- Tomás Escobar Rivera - 1036448606
- Nicolás Orozco Medina - 1011510119

1) Redes Neuronales Convolucionales (CNN):

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) son un tipo de red neuronal artificial diseñada para procesar datos con una estructura de cuadrícula, como imágenes. Estas redes están inspiradas en la organización de la corteza visual del cerebro humano, donde ciertas neuronas se activan ante regiones específicas del campo visual.

Las CNN están compuestas por varias capas de procesamiento, cada una de las cuales extrae características relevantes de los datos de entrada. Estas capas incluyen:

- **Capas convolucionales:** Aplican filtros o kernels para detectar patrones locales.
- **Capas de activación:** Introducen no linealidades en el para mejorar la capacidad de representación.
- **Capas de pooling:** Reducen la dimensionalidad, mejoran la invariancia a la traslación y reducen el costo computacional.
- **Capas totalmente conectadas:** Realizan la clasificación final combinando las características extraídas.
- **Capas de regularización:** Evitan el sobreajuste mediante técnicas como Dropout.

Una CNN procesa una imagen de entrada, pasando la misma a través de una series de capas convolucionales y de pooling, en las cuales se detectan características básicas de la imagen. Posteriormente, las capas más profundas reconocen algunas de las formas y objetos más complejos. Luego, pasan por las capas completamente conectadas que usan dichas características extraídas para realizar la tarea específica de la CNN.

Las principales aplicaciones de las CNN incluyen la clasificación de imágenes, el reconocimiento facial, la detección de objetos, el análisis médico y la visión por computadora en vehículos autónomos.

2) Conceptos de las CNN y sus capas:

- **Capa convolucional:** Es más útil en las capas iniciales e intermedias. Utiliza filtros o kernels para extraer características locales de la imagen, como bordes, texturas o formas. Estas capas son fundamentales para identificar patrones espaciales.

- **Funciones de activación:** Se usan entre capas convolucionales y densas, después de cada operación lineal para introducir no linealidad.

    - **SoftMax:** Convierte las salidas en probabilidades para problemas de clasificación.
    - **Escalón/Step:** Activa o desactiva una neurona según un umbral (poco usada por su falta de diferenciabilidad).
    - **ReLU (Rectified Linear Unit):** Reemplaza valores negativos por cero, acelerando la convergencia.
    - **LeakyReLU:** Variante de ReLU que permite pequeños valores negativos para evitar la desactivación de neuronas.
    - **Sigmoide:** Produce valores entre 0 y 1, útil para la clasificación binaria.
    - **Tangente Hiperbólica (Tanh):** Produce valores entre -1 y 1, útil para modelos con datos centrados en cero.
- **Pooling Layer:** Se implementa después de las capas convolucionales. Reduce la dimensión espacial de los mapas de características. Los métodos más comunes son Max Pooling y Average Pooling.

- **Dropout:** Se usa en capas intermedias o finales de redes densas y convolucionales durante el entrenamiento. Desactiva aleatoriamente un porcentaje de neuronas durante el entrenamiento para prevenir el sobreajuste.

- **Dense Layer:** Se implementa en las capas finales de la red neuronal. Es una capa completamente conectada, donde cada neurona se conecta con todas las neuronas de la capa anterior, útil para la toma de decisiones finales.

- **Callbacks:** Se usan durante el entrenamiento, en el proceso de optimización. Permiten intervenir en el proceso para guardar modelos, detener el entrenamiento o ajustar la tasa de aprendizaje:

    - **ModelCheckpoint:** Guarda el modelo con mejor rendimiento.
    - **EarlyStopping:** Detiene el entrenamiento si no hay mejora en la validación.
    - **TensorBoard:** Permite la visualización del proceso de entrenamiento.
    - **LearningRateScheduler:** Ajusta la tasa de aprendizaje durante el entrenamiento.
    - **ReduceLROnPlateau:** Reduce la tasa de aprendizaje si no hay mejora en la pérdida.
    - **Custom Callbacks:** Permiten implementar funciones personalizadas para controlar el entrenamiento.

- **Redes Preentrenadas:** Se utilizan como base en la arquitectura, generalmente en la extracción de características antes de la capa densa final. Modelos previamente entrenados sobre grandes conjuntos de datos y son usados como punto de partida para nuevas CNN, con el fin de aprovechar el conocimiento ya adquirido. Ejemplos:

    - **VGG16:** Arquitectura con 16 capas, utilizada en la clasificación de imágenes.
    - **ResNet:** Introduce conexiones residuales para entrenar redes muy profundas.
    - **Inception:** Utiliza módulos con convoluciones de diferentes tamaños para mejorar la precisión.
    - **MobileNet:** Optimizada para dispositivos móviles con recursos limitados.

3) Optimizadores y Algoritmo ADAM:

Los optimizadores son algoritmos que ajustan los pesos de la red para minimizar la función de pérdida y mejorar la precisión del modelo.

El algoritmo **ADAM (Adaptive Moment Estimation)** combina las ventajas de los métodos de descenso de gradiente con momento y RMSprop. Utiliza estimaciones adaptativas de primer orden (media) y segundo orden (varianza) de los gradientes.

La actualización de los pesos se realiza con los siguientes pasos de los cuales se derivan las siguientes ecuaciones:

1. Se calcula un media móvil de los gradientes (primer momento)
    - $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
2. Se calcula una media móvil del cuadrado de los gradientes (segundo momento)
    - $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
3. Corrección del sesgo.
    - $\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$
    - $\widehat{v}_t = \frac{v_t}{1-\beta_2^t}$
4. Se actualizan los parámetros del modelos usando las estimaciones corregidas.
    - $\theta_t = \theta_{t-1} - \alpha \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon}$

Donde en las fórmulas, g_t es el gradiente en el paso t, β_1 y β_2 son factores de decaimiento para las medias móviles, α es la tasa de aprendizaje y ϵ es un valor pequeño para evitar divisiones por cero.

Ahora bien, el algoritmo ADAM es adecuado para redes neuronales profundas debido a que:

- Ajusta la tasa de aprendizaje para cada parámetro.
- Es eficiente en términos de memoria.
- Converge más rápido que otros algoritmos.
- Funciona bien con grandes conjuntos de datos y redes profundas.

Gracias a estas características, ADAM se ha convertido en uno de los optimizadores más utilizados en el entrenamiento de redes neuronales.