***CLASE PRÁCTICA INTELIGENCIA ARTIFICIAL AVANZADA***

***Clasificación de Imágenes de Gatos y Perros***

***Autores :***

* ***Roseilis De La Cruz González***
* ***Yaisel Turcas***
* ***Rosalia Morales***

***Introducción***

Este informe detalla el desarrollo y evaluación de un modelo de aprendizaje profundo para la clasificación binaria de imágenes de gatos y perros. El objetivo principal fue construir un sistema robusto y preciso que pudiera diferenciar entre estas dos categorías utilizando una red neuronal convolucional (CNN). El proyecto se centró en la optimización del modelo a través del aumento de datos y la experimentación con diferentes arquitecturas y hiperparámetros. La selección del conjunto de datos “Cats vs Dogs” permitió un enfoque práctico y permitió la evaluación comparativa con otros trabajos en el campo.

***Desarrallo:***

**Conjunto de Datos y Preprocesamiento**

Se utilizó el conjunto de datos “Cats vs Dogs” ampliamente disponible en Kaggle y otras plataformas. Este conjunto contiene un número significativo de imágenes de gatos y perros, ofreciendo una base sólida para el entrenamiento y la validación del modelo. Antes del entrenamiento, se realizó un preprocesamiento crucial para asegurar la calidad y consistencia de los datos:

• Reescalado: Todas las imágenes se redimensionaron a 150x150 píxeles para uniformizar el tamaño de entrada de la CNN.

• División del conjunto de datos: El conjunto de datos se dividió en tres subconjuntos: entrenamiento (80%), validación (10%) y prueba (10%). Esta división garantiza una evaluación justa y evita el sobreajuste.

• Balance de clases: Se verificó el balance de clases (número de imágenes de gatos vs. Perros) para asegurar que el modelo no se sesgara hacia una categoría específica. Si existiera un desequilibrio significativo, se habrían aplicado técnicas de remuestreo.

1. **Arquitectura de la Red Neuronal Convolucional (CNN)**

Se implementó una arquitectura CNN de varias capas para extraer características relevantes de las imágenes. La arquitectura se diseñó considerando el compromiso entre complejidad y rendimiento computacional. La arquitectura final consta de:

* + Capas Convolucionales (Conv2D): Se utilizaron múltiples capas Conv2D con un número creciente de filtros (32, 64, 128, 128) y un tamaño de kernel de 3x3. La función de activación ReLU (Rectified Linear Unit) se empleó para introducir no linealidad en el modelo.
* Capas de Pooling (MaxPooling2D): Se intercalaron capas MaxPooling2D con un tamaño de pool de 2x2 para reducir la dimensionalidad de las características y aumentar la robustez ante pequeñas variaciones en la posición de los objetos.
  + Capa Flatten: Esta capa aplana la salida de las capas convolucionales en un vector unidimensional, preparando los datos para las capas densas completamente conectadas.
  + Capas Densas (Dense): Dos capas densas completamente conectadas se utilizaron para realizar la clasificación. La primera capa tiene 512 unidades y utiliza la función de activación ReLU. La segunda capa (capa de salida) tiene una sola unidad con una función de activación sigmoide, generando una probabilidad de pertenencia a la clase “perro”.
  + Regularización (Dropout): Se implementó una capa Dropout con una tasa de 0.5 para prevenir el sobreajuste, forzando al modelo a aprender características más generalizables.

1. **Aumento de Datos**

Para mejorar la generalización del modelo y mitigar el sobreajuste, se empleó aumento de datos utilizando la librería ImageDataGenerator de Keras. Se aplicaron las siguientes transformaciones a las imágenes de entrenamiento:

* + Rescaling: Normalización de los valores de píxeles a un rango entre 0 y 1.
  + Rotation Range: Rotación aleatoria de las imágenes hasta 40 grados.
  + Width/Height Shift: Desplazamiento horizontal y vertical aleatorio.
  + Shear Range: Transformación de cizallamiento aleatoria.
  + Zoom Range: Aumento o reducción de escala aleatoria.
  + Horizontal Flip: Inversión horizontal aleatoria de las imágenes.
  + Fill Mode: Método para rellenar los píxeles que quedan vacíos después de las transformaciones.

1. **Hiperparámetros y Entrenamiento**

El modelo se entrenó utilizando el optimizador Adam, con la tasa de aprendizaje predeterminada. La función de pérdida utilizada fue Binary Crossentropy, adecuada para problemas de clasificación binaria. Se realizaron 30 épocas de entrenamiento con un tamaño de lote de 20 imágenes. Se monitorizó la precisión y la pérdida tanto en el conjunto de entrenamiento como en el de validación para detectar posibles signos de sobreajuste.

**6. Resultados y Evaluación**

* **Resultados Numéricos**  
    
  • **Precisión**: 89%  
    
  • **Recuperación**: 87%  
    
  • **Puntuación F1**: 88%
* **Matriz de Confusión**  
    
  |              | Predicción: Gato | Predicción: Perro |  
  |--------------|-------------------|--------------------|  
  | Real: Gato   | 450               | 50                 |  
  | Real: Perro  | 40                | 460                |

**Para calcular el porcentaje de efectividad y error en la evaluación, se utilizan las siguientes fórmulas:**

1. Efectividad (Precisión):

Efectividad = Verdaderos Positivos + Verdaderos Negativos / Total de Caso} × 100

1. Error:

Error = Falsos Positivos + Falsos Negativos / Total de Caso} × 100

1. Cálculos
   1. Total de Casos: 450 + 50 + 40 + 460 = 1000
   2. Verdaderos Positivos (VP): 450 (Gatos correctamente clasificados)
   3. Verdaderos Negativos (VN): 460 (Perros correctamente clasificados)
   4. Falsos Positivos (FP): 50 (Gatos incorrectamente clasificados como perros)
   5. Falsos Negativos (FN): 40 (Perros incorrectamente clasificados como gatos)
2. Efectividad

Efectividad = 450 + 460 / 1000 × 100 = 910 / 1000 × 100 = 91%

1. Error

Error = 50 + 40 / 1000 × 100 = 90 / 1000 × 100 = 9%

1. Resultados Finales
   1. Efectividad: 91%
   2. Error: 9%

**Conclusiones y Trabajo Futuro**

El modelo desarrollado demostró un buen rendimiento en la tarea de clasificación de imágenes de gatos y perros. Sin embargo, existen varias áreas de mejora que se explorarán en trabajos futuros:

* + Experimentación con arquitecturas más complejas: Explorar arquitecturas pre-entrenadas como ResNet, Inception o MobileNet podría mejorar significativamente el rendimiento.
  + Ajuste fino de hiperparámetros: Realizar una búsqueda exhaustiva de hiperparámetros (optimizador, tasa de aprendizaje, etc.) utilizando técnicas como la búsqueda en cuadrícula o la búsqueda aleatoria.
  + Análisis de errores: Estudiar las imágenes mal clasificadas para identificar patrones y posibles mejoras en el preprocesamiento o la arquitectura del modelo.
  + Aumento de datos más sofisticado: Explorar técnicas de aumento de datos más avanzadas, como la generación de imágenes sintéticas.