Integrantes: Fernando Guevara – Juan Zapata

Nivel y Paralelo: Séptimo "A"

Fecha: 15/01/2024

Modulo y Docente: Inteligencia Artificial – Ing. Rubén Nogales

Tema: Proyecto Final – Reconocimiento de Placa

CREACION DEL DATASET

Para la generación del conjunto de datos, se llevo a cabo un proceso de preprocesamiento de imágenes utilizando la biblioteca OpenCV en Python. El dataset original consta imágenes de las letras y números separadas por carpetas, cada imagen tiene un tamaño de 128x128.

El preprocesamiento que se llevo acabo se realizo lo siguiente: cada imagen se redujo su tamaño de 128px a 28px, para disminuir los tiempos de entrenamiento de modelo, además, se paso la imagen a escala de grises y se invirtió los colores de la imagen para que el contenido de interés quede de fondo blanco. El resultado fue el siguiente:

Tras completar el proceso de preprocesamiento de imágenes en el conjunto de datos original, se procedió a la transformación y etiquetado de las imágenes para su posterior utilización en modelos de aprendizaje automático.

Etiquetado de Imágenes

Cada imagen en el conjunto de datos fue asignada a una etiqueta correspondiente, representando las letras y números presentes en la imagen. Esta asignación se llevó a cabo de manera cuidadosa y se refleja en un archivo CSV que actúa como un índice de asociación entre las imágenes y sus respectivas etiquetas.

Aplanado de Imágenes

Como parte del proceso de preparación para el aprendizaje automático, cada imagen fue aplanada mediante la operación "flatten". Esta operación convierte las imágenes bidimensionales en vectores unidimensionales, facilitando su almacenamiento en formato CSV y su posterior procesamiento por modelos de aprendizaje automático que esperan datos en formato de vector.

Modelo CNN

Carga y Preprocesamiento del Conjunto de Datos

El conjunto de datos se carga desde el archivo CSV (new_dataset.csv) que contiene las rutas de las imágenes (imagen) y sus respectivas etiquetas (etiqueta). Las etiquetas se convierten de texto a números mediante un LabelEncoder para facilitar el entrenamiento del modelo.

Las imágenes se cargan utilizando TensorFlow y se aplican operaciones de preprocesamiento, que incluyen la lectura del archivo, decodificación de JPEG, redimensionamiento a un tamaño específico (en este caso, 224x224 píxeles) y preprocesamiento específico de ResNet50.

Creación de Conjuntos de Datos con tf.data
Se utiliza tf.data para crear conjuntos de datos eficientes que se pueden utilizar en el entrenamiento del modelo. El conjunto de entrenamiento se mezcla, se divide en lotes de tamaño 32 y se prefetcha para una mejor eficiencia durante el entrenamiento. El conjunto de prueba se crea de manera similar pero sin mezcla.

Definición del Modelo CNN
El modelo CNN se define utilizando la API secuencial de Keras. Incluye capas convolucionales, capas de agrupación (max pooling), una capa de aplanado, capas totalmente conectadas y una capa de salida con activación softmax. La arquitectura se personaliza según el problema específico, y en este caso, se proporciona un ejemplo básico.
La arquitectura del modelo CNN que has definido utiliza la API secuencial de Keras y se compone de varias capas convolucionales, de agrupación y totalmente conectadas. Aquí está una explicación detallada de cada capa:

Capa Conv2D (32 filtros, tamaño del kernel 3x3, activación ReLU):

Esta es la primera capa convolucional que procesa la imagen de entrada.
Utiliza 32 filtros, lo que significa que la capa extraerá 32 características diferentes de la imagen.
El tamaño del kernel es 3x3, lo que indica que se utilizará una ventana de 3x3 píxeles para realizar la convolución.
La activación ReLU (Rectified Linear Unit) se aplica para introducir no linealidades en la red y permitir la convergencia más rápida durante el entrenamiento.
La forma de entrada es (224, 224, 3), indicando imágenes de tamaño 224x224 píxeles con tres canales de color (RGB).
Capa MaxPooling2D (tamaño de la ventana 2x2):

La capa de agrupación reduce la dimensionalidad de la representación y el número de parámetros en la red, ayudando a evitar el sobreajuste.
Utiliza una ventana de agrupación de 2x2 para reducir a la mitad la altura y la anchura de la representación.
Capa Conv2D (64 filtros, tamaño del kernel 3x3, activación ReLU):

Otra capa convolucional similar a la primera, pero ahora con 64 filtros.
Capa MaxPooling2D (tamaño de la ventana 2x2):

Otra capa de agrupación para reducir la dimensionalidad de la representación.
Capa Conv2D (128 filtros, tamaño del kernel 3x3, activación ReLU):

Una tercera capa convolucional, esta vez con 128 filtros.
Capa Flatten:

La capa de aplanado transforma la salida de las capas convolucionales y de agrupación en un vector unidimensional. Esto es necesario antes de conectar la capa completamente conectada.
Capa Dense (128 unidades, activación ReLU):

Una capa completamente conectada con 128 unidades y activación ReLU.
Capa Dense (Número de clases de salida, activación softmax):

La capa de salida con activación softmax produce la distribución de probabilidad sobre las clases. El número de unidades en esta capa es igual al número de clases en tu conjunto de datos, y la activación softmax garantiza que las salidas sean probabilidades válidas.

Compilación y Entrenamiento del Modelo
El modelo se compila utilizando el optimizador Adam, la función de pérdida 'sparse_categorical_crossentropy' (adecuada para clasificación con etiquetas enteras) y la métrica de precisión ('accuracy'). Posteriormente, se entrena el modelo utilizando el conjunto de entrenamiento y se valida con el conjunto de prueba.

Resultados del Entrenamiento
Durante el entrenamiento, el modelo muestra una pérdida (loss) y precisión (accuracy) para cada época. En este caso, después de 3 épocas, se obtienen los siguientes resultados:

Pérdida de entrenamiento (loss): 0.0992
Precisión de entrenamiento (accuracy): 96.96%
Pérdida de validación (val_loss): 0.2069
Precisión de validación (val_accuracy): 94.44%
Estos resultados indican que el modelo ha aprendido eficazmente en el conjunto de entrenamiento y es capaz de generalizar bien a datos no vistos en el conjunto de prueba. La pérdida y precisión de entrenamiento y validación proporcionan información sobre el rendimiento y la capacidad de generalización del modelo.

Guardar el Modelo Entrenado
Finalmente, el modelo entrenado se guarda en un archivo llamado 'modelo_cnn.h5' para su posterior uso o despliegue en aplicaciones prácticas.

Procesamiento de imágenes en tiempo real
Detección de Placas:

La cámara captura continuamente imágenes que se convierten a escala de grises (imgGray).
Se utiliza el clasificador en cascada para detectar posibles regiones que corresponden a placas de matrícula en la imagen. Estas regiones se almacenan en la variable numberPlates.
Resaltado Visual:

Para cada región detectada como posible placa ((x, y, w, h)):
Se calcula el área de la región (area = w * h).
Si el área supera un umbral mínimo (minArea), se considera una región válida y se procede a resaltarla.
Se dibuja un rectángulo alrededor de la región detectada utilizando OpenCV (cv2.rectangle).
Se agrega un texto "NumberPlate" cerca del rectángulo para indicar visualmente la región detectada (cv2.putText).

Link Proyecto

Proyecto IA