Ejercicio 1: Construcción de una Red Neuronal LSTM con Keras

Objetivo:

En este ejercicio, aprenderás a implementar una red neuronal de tipo LSTM (Long Short-Term Memory) utilizando la librería Keras. Las redes LSTM son un tipo de red neuronal recurrente (RNN) especialmente adecuada para problemas de predicción de secuencias, como series temporales o procesamiento de texto. El objetivo es construir un modelo LSTM para una tarea sencilla de predicción y evaluar su rendimiento.

Descripción de la Tarea:

Realizarás los siguientes pasos:

1. Preprocesar un conjunto de datos de series temporales (como precios de acciones, datos de temperatura u otra secuencia numérica).

2. Crear un modelo LSTM utilizando Keras.

3. Entrenar el modelo con los datos de entrenamiento.

4. Evaluar el rendimiento del modelo en los datos de prueba.

5. Visualizar la pérdida a lo largo de las epocs y realizar predicciones.

Pasos:

- Paso 1: Importar las librerías necesarias y cargar el conjunto de datos.

- Paso 2: Preprocesar los datos: Normalizarlos y dividirlos en conjuntos de entrenamiento y prueba.

- Paso 3: Reestructurar los datos al formato de entrada requerido para LSTM: [muestras, pasos de tiempo, características].

- Paso 4: Definir un modelo LSTM utilizando la clase *Sequential* de Keras.

- Paso 5: Compilar el modelo con una función de pérdida y un optimizador adecuado.

- Paso 6: Entrenar el modelo con los datos de entrenamiento.

- Paso 7: Evaluar el rendimiento del modelo y visualizar los resultados.

Conjunto de Datos:

Puedes usar cualquier conjunto de datos de series temporales para este ejercicio. Un conjunto de datos sencillo de temperatura, precios de acciones u otra secuencia numérica puede cargarse utilizando

librerías como Pandas. Para simplicidad, en este ejercicio supondremos que estás usando un conjunto de datos sintético de series temporales.

Entregables Esperados:

1. Un modelo LSTM completamente implementado utilizando Keras.

2. Un gráfico que muestre la pérdida de entrenamiento y validación a lo largo del tiempo.

3. Una comparación de los valores predichos frente a los valores reales en el conjunto de prueba.

Ejercicio 2: Construcción de una Red Neuronal Convolucional (CNN) para Clasificación de Imágenes Usando Keras

Objetivo:

En este ejercicio, implementarás una Red Neuronal Convolucional (CNN) utilizando la librería Keras para clasificar imágenes del conjunto de datos CIFAR-10. Este conjunto de datos consta de 60,000 imágenes de 32x32 píxeles a color, distribuidas en 10 clases diferentes. Al finalizar el ejercicio, comprenderás cómo construir, entrenar y evaluar un modelo de CNN utilizando Keras.

Pasos:

1. Preparación del Conjunto de Datos: Cargarás el conjunto de datos CIFAR-10, que está incluido en Keras, y lo preprocesarás normalizando los valores de los píxeles y convirtiendo las etiquetas a formato categórico.

2. Arquitectura del Modelo: Crearás un modelo de CNN con múltiples capas convolucionales, capas de pooling (submuestreo) y capas fully connected. El objetivo es crear un modelo capaz de extraer jerarquías espaciales de características a partir de las imágenes.

3. Compilación del Modelo: Compilarás el modelo utilizando una función de pérdida, un optimizador y una métrica de rendimiento adecuados.

4. Entrenamiento del Modelo: Entrenarás el modelo con los datos de entrenamiento de CIFAR-10 y lo validarás utilizando una parte de los datos.

5. Evaluación del Modelo: Evaluarás el modelo en el conjunto de datos de prueba y medirás su precisión.

Requisitos:

- Entorno de Python 3.x

- Keras y TensorFlow instalados (*pip install keras tensorflow*)

- Comprensión básica de Python y Redes Neuronales

Tareas:

1. Cargar y Preprocesar el Conjunto de Datos CIFAR-10

   - Normalizar los datos de imagen escalando los valores de los píxeles entre 0 y 1.

   - Convertir las etiquetas en formato categórico para clasificación multiclase.

2. Definir la Arquitectura de la CNN

   - Utilizar capas convolucionales con activación ReLU.

   - Incluir capas de max-pooling para reducir la dimensionalidad.

   - Agregar una capa fully connected al final, seguida de la capa de salida con activación softmax para la clasificación.

3. Compilar el Modelo

   - Usar *categorical_crossentropy* como función de pérdida.

   - Utilizar un optimizador como *Adam* y realizar un seguimiento de la precisión del modelo.

4. Entrenar el Modelo

   - Entrenar la CNN con los datos de entrenamiento y validarla utilizando una parte de los datos.

   - Visualizar la pérdida y la precisión durante el entrenamiento y la validación.

5. Evaluar el Modelo

   - Medir la precisión del modelo en los datos de prueba.

Entrega:

Envía el código Python junto con un breve informe (1-2 párrafos) que resuma el rendimiento del modelo, la precisión obtenida y cualquier observación realizada durante el entrenamiento.

¡Buena suerte!