COLOMBIA POTENCIA DE LA VIDA

TIC

Actividad 6

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Para esta actividad vamos a trabajar con el conjunto de datos Fashion MNIST

La documentación oficial se encuentra en: https://github.com/zalandoresearch/fashion-mnist

Es un conjunto de imágenes de prendas de vestir en formato de 28x28 píxeles en escala de grises. Contiene un total de 70,000 imágenes, que están divididas en 60,000 imágenes de entrenamiento y 10,000 imágenes de prueba. Cada imagen pertenece a una de las 10 categorías de prendas diferentes, que incluyen camisetas, pantalones, suéteres, vestidos, abrigos, sandalias, camisas, zapatillas, bolsos y botines.

Cada ejemplo de entrenamiento y prueba se asigna a una de las siguientes etiquetas:

0 camiseta
1 pantalón
2 suéter
3 vestido
4 abrigo
5 sandalia
6 camisa
7 zapatilla de deporte
8 bolsa
9 Botín

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Este conjunto de datos es ampliamente utilizado en la comunidad de aprendizaje automático y visión por computadora como un reemplazo del conjunto de datos MNIST original, que contiene dígitos escritos a mano. La elección de prendas de vestir como categorías proporciona un desafío adicional para los algoritmos de clasificación de imágenes, ya que las diferencias entre las categorías pueden ser más sutiles que las diferencias entre los dígitos manuscritos.

El conjunto de datos Fashion MNIST es útil para probar y comparar diferentes algoritmos de clasificación de imágenes y para enseñar conceptos básicos de aprendizaje automático y visión por computadora. Es un recurso valioso para los investigadores, estudiantes y profesionales que trabajan en estos campos.

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Carga del conjunto de datos Fashion MNIST:

Utiliza la función load_data() del módulo Fashion mnist de Keras para cargar el conjunto de datos Fashion MNIST, que contiene imágenes de prendas de vestir y sus etiquetas asociadas.

```
import keras
# Cargue el conjunto de datos Fashion MNIST
fmnist = keras.datasets.fashion_mnist
# Cargue la división de entrenamiento y prueba del conjunto
de datos Fashion MNIST
(training_images, training_labels), (test_images,
test_labels) = fmnist.load_data()
```

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Visualización de una muestra:
Selecciona un índice específico (index) dentro del conjunto de entrenamiento para visualizar una muestra de imagen y su etiqueta correspondiente. Imprime la etiqueta y muestra la imagen utilizando plt.imshow() de Matplotlib.

```
import numpy as np
import matplotlib.pyplot as plt
# Puedes poner aquí entre 0 y 59999
index = 1
# Imprime la etiqueta y la imagen.
np.set_printoptions(linewidth=320)
print(f'Label: {training_labels[index]}')
print(f'Image:\n {training_images[index]}')
# Visualiza la imagen
plt.imshow(training_images[index])
```

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Normalización de los datos:

Normaliza los valores de píxeles de las imágenes dividiendo por 255.0, lo que asegura que los valores estén en el rango [0, 1].

```
# Normalizar los valores de píxeles del tren y probar las
imágenes.
training_images  = training_images / 255.0
test_images = test_images / 255.0
```

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Construcción del modelo:

Define un modelo secuencial utilizando keras.models.Sequential(). Este modelo consta de una capa de aplanamiento (Flatten) para convertir la imagen 2D en un vector 1D, seguida de dos capas completamente conectadas (Dense) con funciones de activación relu y softmax.

```
# Construir el modelo de clasificación.
model =
keras.models.Sequential([keras.layers.Flatten(input_shape=(28
,28)),
keras.layers.Dense(128, activation='relu'),
keras.layers.Dense(10, activation='softmax')])
```

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Compilación del modelo:

Compila el modelo utilizando model.compile(), especificando el optimizador (adam), la función de pérdida (sparse_categorical_crossentropy) y las métricas (accuracy).

```
# Compilar el modelo
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Entrenamiento del modelo:

Entrena el modelo con los datos de entrenamiento utilizando model.fit(), especificando el número de épocas.

```
# Entrenar el modelo
history = model.fit(training_images, training_labels,
epochs=10)
```

Graficar el historial de entrenamiento:

Utiliza pandas para convertir el historial de entrenamiento del modelo en un DataFrame y traza las curvas de pérdida y métricas durante el entrenamiento y la validación.

```
# Graficar el historial de entrenamiento:
pd.DataFrame(history.history).plot(grid=True)
```

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Evaluación del modelo:

Evalúa el modelo en el conjunto de entrenamiento y prueba utilizando model.evaluate(), lo que proporciona la pérdida y la precisión del modelo en los datos de prueba.

```
# Evaluar el modelo en el conjunto de entrenamiento
loss, accuracy = model.evaluate(training_images,
training_labels)
print("Pérdida en el conjunto de entrenamiento:", loss)
print("Precisión en el conjunto de entrenamiento:", accuracy)

# Evaluar el modelo con datos no vistos
loss, accuracy = model.evaluate(test_images, test_labels)
print("Pérdida en el conjunto de prueba:", loss)
print("Precisión en el conjunto de prueba:", accuracy)
```

Redes neuronales completamente conectadas aplicadas en Fashion MNIST reconocimiento de prendas de vestir

Pasos a ejecutar:

Predicción de una muestra:

Selecciona un índice específico dentro del conjunto de prueba para realizar una predicción utilizando model.predict(). Imprime la etiqueta real y la clasificación predicha.

```
#predict
index = 1
print(f'Label: {test_labels[index]}')
classification = model.predict(test_images[index:index+1])
print(f'Classification:\n {classification.reshape(-1,1)}')
```

Preguntas de comprensión

1. ¿Qué conjunto de datos se utiliza en este código y qué problema de aprendizaje automático se aborda?
2. ¿Por qué es importante normalizar los valores de píxeles de las imágenes antes de entrenar el modelo?
3. ¿Qué arquitectura de red neuronal se utiliza en este código y cuántas capas tiene?
4. ¿Cuál es la función de activación utilizada en la capa oculta y en la capa de salida de la red neuronal, y por qué se eligen esas funciones?
5. ¿Qué función de pérdida se utiliza para compilar el modelo y qué métricas se utilizan para evaluar su rendimiento?
6. ¿Cuántas épocas se utilizan para entrenar el modelo y por qué se elige ese número?
7. ¿Qué significa la función model.summary() y qué información proporciona?

Ejercicios de exploración:

Para profundizar en su comprensión y explorar más, pruebe los siguientes ejercicios:

Ejercicio 1: Ejecuta la predicción de una muestra y observe que el resultado es una lista de números.
¿Por qué crees que es así y qué representan esos números?

Ejercicio 2: Examine las capas de su modelo y experimente con diferentes valores para la capa densa como 256 neuronas. Observa cómo varían los resultados en términos de pérdida, tiempo de entrenamiento, etc.
¿Por qué crees que ocurre esto?

Ejercicio 3: ¿Qué sucedería si eliminas la capa Flatten()?
Reflexiona sobre cómo afectaría esto al procesamiento de las imágenes y por qué.

Ejercicios de exploración:

Ejercicio 4: Considera las capas finales (de salida).
¿Por qué hay 10 de ellas? ¿Qué pasaría si tuvieras una cantidad diferente a 10?
Intenta entrenar la red con un número diferente de capas finales y reflexiona sobre cómo afectaría esto a la capacidad de clasificación del modelo.

Ejercicio 5: Explora los efectos de añadir otra capa entre la capa con 256 neuronas y la capa final con 10. Observa cómo esto podría afectar la capacidad del modelo para capturar características complejas de los datos de entrada y cómo afecta el rendimiento general del modelo.

Ejercicio 6: Considera el impacto de entrenar durante más o menos épocas.
Experimenta con diferentes números de épocas y reflexiona sobre cómo esto afecta la mejora del modelo en términos de pérdida y precisión.

Ejercicio 7: Reflexiona sobre el impacto de eliminar la normalización de los datos antes del entrenamiento. Ejecuta el código sin normalización y observa cómo difieren los resultados. ¿Por qué crees que ocurre esto?

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira