Tema: AP2_2 Tarea 6.1

Nombre del estudiante: Rommel Torres Capelo

Docente: Ing. Cristian David Muñoz

Fecha entrega: 11 de febrero de 2024

Resumen

En esta tarea se describe el uso TensorFlow con Keras. Se describe el proceso de construcción, entrenamiento y evaluación de un modelo de clasificación de imágenes utilizando una base de datos de imagenes.

Introducción

Las redes neuronales son una herramienta poderosa en el campo del aprendizaje automático y la inteligencia artificial. TensorFlow, es una de las bibliotecas más populares para la implementación de redes neuronales, junto con Keras, una interfaz de alto nivel que simplifica el proceso de construcción de modelos.

Desarrollo

El algoritmo de red neuronal de Microsoft se compone de hasta tres capas de nodos:

Nivel de entrada: los nodos de entrada definen todos los valores de atributos de entrada para el modelo de minería de datos, así como sus probabilidades.

Nivel oculto: los nodos ocultos reciben entradas de los nodos de entrada y proporcionan salidas a los nodos de salida. El nivel oculto es donde se asignan pesos a las distintas probabilidades de las entradas. Un peso describe la relevancia o importancia de una entrada determinada para el nodo oculto. Cuanto mayor sea el peso asignado a una entrada, más importante será el valor de dicha entrada. Los pesos pueden ser negativos, lo que significa que la entrada puede impedir, en lugar de activar, un resultado concreto.

Nivel de salida : los nodos de salida representan valores de atributo de predicción para el modelo de minería de datos.(kfollis, 2023)

Se deben seguir varios procesos hasta obtener los resultados esperados de desarrollo, estos procesos son:

Preparación de los Datos

El modelo de red neuronal requiere datos de entrada, debe contener una columna de clave, una o más columnas de entrada y una o más columnas de predicción.

Para este ejemplo se ha utilizado el dataset "Fashion MNIST", que contiene 60,000 imágenes de entrenamiento y 10,000 imágenes de prueba. Las imágenes tienen una resolución de 28x28 píxeles en escala de grises.

Entrenamiento del Modelo

El objetivo del modelo es encontrar los parámetros W y b que conduzcan a la menor pérdida posible. No se puedes los datos X o las etiquetas correspondientes porque son fijos, pero puedes ajustar los valores de W y b. Estos problemas de este tipo caen en el área matemática de la optimización.

El modelo se entrena utilizando el algoritmo SGD Stochastic Gradient Descent, El algoritmo SGD realiza actualiza las ponderaciones del modelo lineal por cada ejemplo de datos que encuentre.

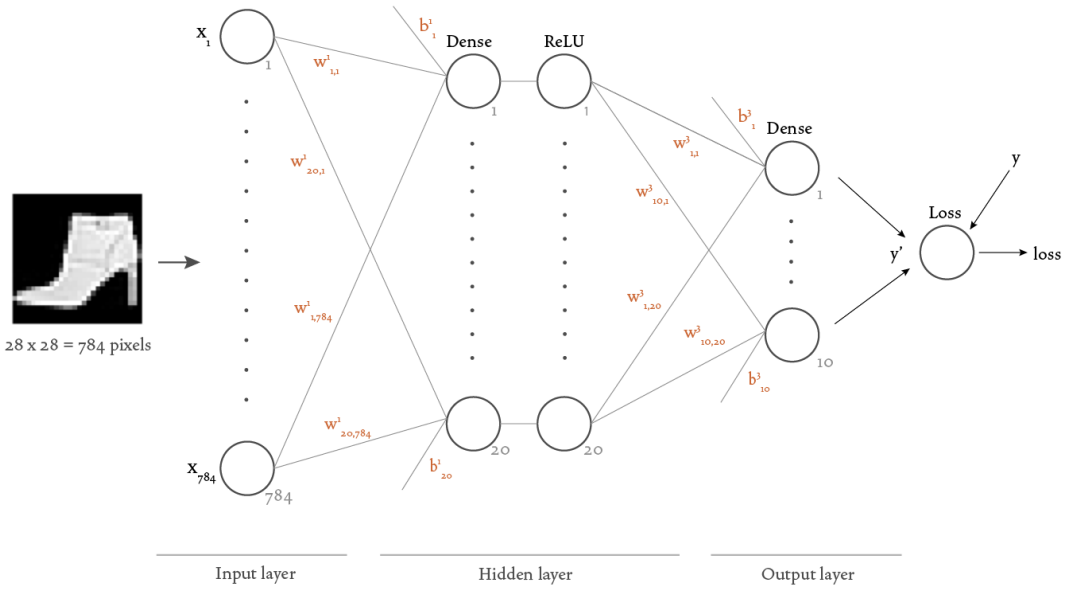

Figura 1. Entrenamiento del modelo

Para hacer una predicción, se debe llamar al método de predicción del modelo y pasar una o más imágenes.

Evaluación del Modelo

El modelo entrenado se evalúa utilizando el conjunto de datos de prueba. Se calcula la precisión del modelo, así como otras métricas de rendimiento como la pérdida.

Se adjunta el código del ejemplo:

```
import gzip
import numpy as np
import tensorflow as tf
from typing import Tuple
import requests
from PIL import Image

class NeuralNetwork(tf.keras.Model):
  def __init__(self):
    super(NeuralNetwork, self).__init__()
    self.sequence = tf.keras.Sequential([
      tf.keras.layers.Flatten(input_shape=(28, 28)),
      tf.keras.layers.Dense(20, activation='relu'),
      tf.keras.layers.Dense(10)
    ])

  def call(self, x: tf.Tensor) -> tf.Tensor:
    y_prime = self.sequence(x)
    return y  prime


labels_map = {
    0: 'T-Shirt',
    1: 'Trouser',
    2: 'Pullover',
    3: 'Dress',
    4: 'Coat',
    5: 'Sandal',
    6: 'Shirt',
    7: 'Sneaker',
    8: 'Bag',
    9: 'Ankle Boot',
  }


def read_images(path: str, image_size: int, num_items: int) -> np.ndarray:
  with gzip.open(path, 'rb') as file:
    data = np.frombuffer(file.read(), np.uint8, offset=16)
    data = data.reshape(num_items, image_size, image_size)
  return data
```

```
def read_labels(path: str, num_items: int) -> np.ndarray:
  with gzip.open(path, 'rb') as file:
    data = np.frombuffer(file.read(num  items + 8), np.uint8, offset=8)
    data = data.astype(np.int64)
  return data


def get  data(batch  size: int) -> Tuple[tf.data.Dataset, tf.data.Dataset]:
  image_size = 28
  num_train = 60000
  num_test = 10000

  training  images = read  images('data/FashionMNIST/raw/train-images-idx3-ubyte.gz', image  size,
num_train)
  test_images = read_images('data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz', image_size, num_test)
  training  labels = read  labels('data/FashionMNIST/raw/train-labels-idx1-ubyte.gz', num  train)
  test  labels = read  labels('data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz', num  test)

  # (training_images, training_labels), (test_images, test_labels) =
tf.keras.datasets.fashion  mnist.load  data()

  train_dataset = tf.data.Dataset.from_tensor_slices((training_images, training_labels))
  test_dataset = tf.data.Dataset.from_tensor_slices((test_images, test_labels))

  train  dataset = train  dataset.map(lambda image, label: (float(image) / 255.0, label))
  test  dataset = test  dataset.map(lambda image, label: (float(image) / 255.0, label))

  train_dataset = train_dataset.batch(batch_size).shuffle(500)
  test  dataset = test  dataset.batch(batch  size).shuffle(500)

  return (train_dataset, test_dataset)


def training  phase():
  learning  rate = 0.1
  batch_size = 64
  epochs = 5

  (train  dataset, test  dataset) = get  data(batch  size)

  model = NeuralNetwork()

  loss  fn = tf.keras.losses.SparseCategoricalCrossentropy(from  logits=True)
  optimizer = tf.keras.optimizers.SGD(learning_rate)
  metrics = ['accuracy']
  model.compile(optimizer, loss_fn, metrics)

  print('\nFitting:')
  model.fit(train_dataset, epochs=epochs)
```

```
  print('\nEvaluating:')
  (test_loss, test_accuracy) = model.evaluate(test_dataset)
  print(f'\nTest accuracy: {test_accuracy * 100:>0.1f}%, test loss: {test_loss:>8f}')

  model.save('outputs/model')


def inference_phase():
  print('\nPredicting:')

  model = tf.keras.models.load_model('outputs/model')

  url = 'https://raw.githubusercontent.com/MicrosoftDocs/tensorflow-learning-path/main/intro-
keras/predict-image.png'

  with Image.open(requests.get(url, stream=True).raw) as image:
    X = np.asarray(image, dtype=np.float32).reshape((-1, 28, 28)) / 255.0

  predicted_vector = model.predict(X)
  predicted_index = np.argmax(predicted_vector)
  predicted_name = labels_map[predicted_index]

  print(f'Predicted class: {predicted_name}')


training_phase()
inference_phase()
```

Link del repositorio:

https://github.com/RTorresCapelo/ITI

Conclusiones

En este ejemplo, muestra la viabilidad de utilizar TensorFlow con Keras para desarrollar modelos de redes neuronales para tareas de clasificación de imágenes.

Referencias

kfollis. (2023, octubre 31). Algoritmo de red neuronal de Microsoft. https://learn.microsoft.com/es-es/analysis-services/data-mining/microsoft-neural-network-algorithm?view=asallproducts-allversions

Parámetros de entrenamiento—Amazon Machine Learning. (s. f.). Recuperado 11 de febrero de 2024, de https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/training-parameters1.html