Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Momento de Retroalimentación Individual: Implementación de un modelo de Deep Learning.

Carol Arrieta Moreno
A01275465

Monterrey, Nuevo León, México

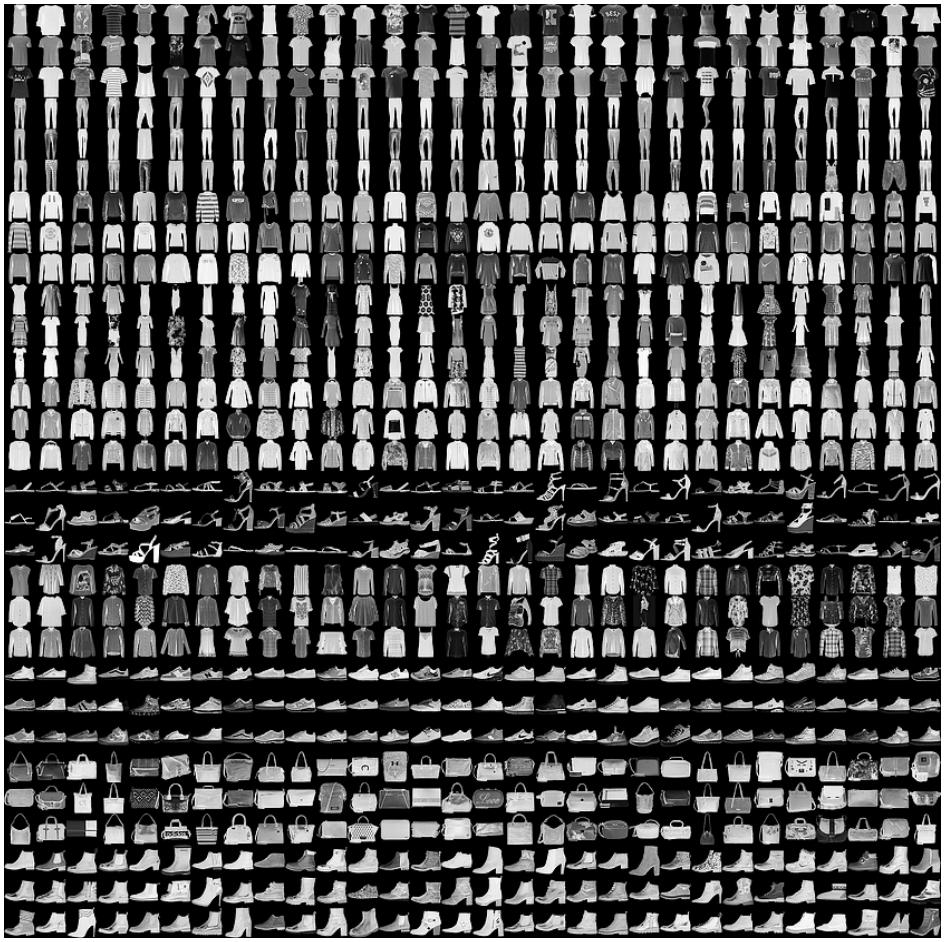
04 de Noviembre De 2023

Inteligencia artificial avanzada para la ciencia de datos II (Gpo 501)

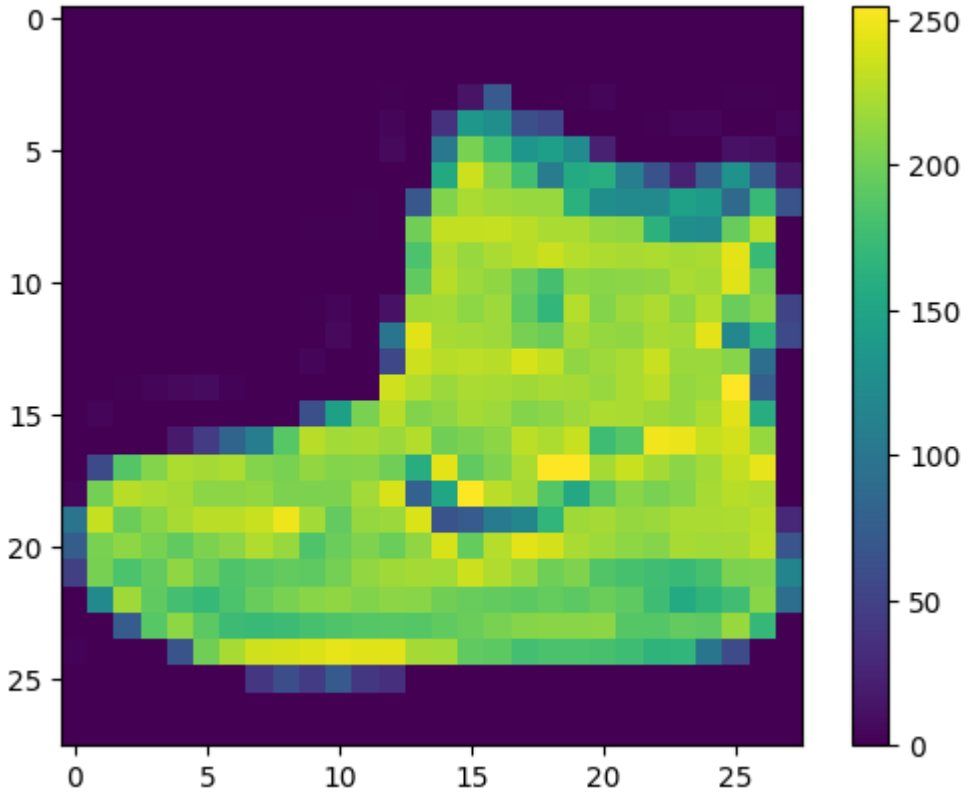
Para esta evidencia decidí realizar un modelo que ayude a diferenciar las diferentes prendas de ropa, para esto hice uso de tensorflow, así como de otras librerías como numpy y matplotlib, el set de datos que utilicé contiene más de 70,000 imágenes para 10 categorías diferentes que son las siguientes:

Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

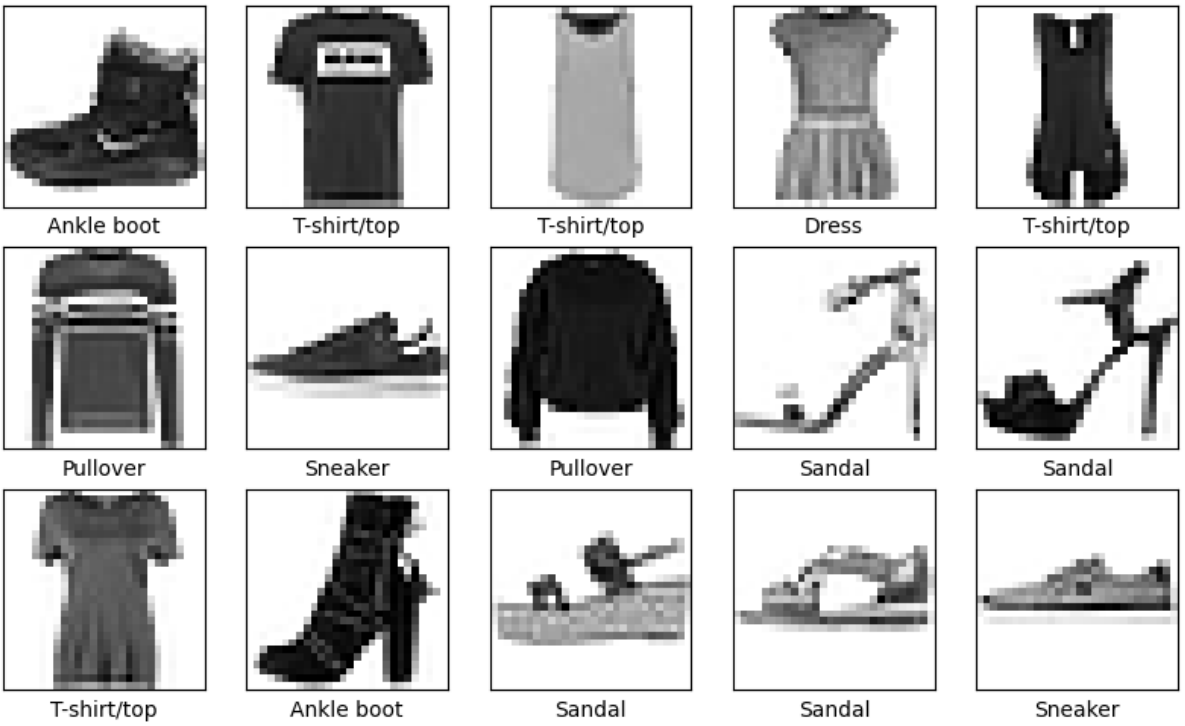
El set de datos usado contiene diferentes imágenes, a grandes rasgos se ve algo así



Lo primero que realicé fue procesar el set de datos de entrenamiento y de test para después poder entrenar la red de manera correcta, inicialmente el rango de los valores de las imágenes era el siguiente



Y para realizar la red neuronal se escalaron los valores en un rango de 0 a 1, aquí podemos ver como quedan



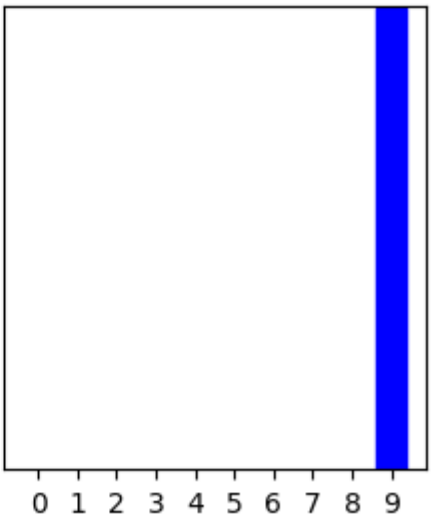
Primero se configuraron las capas en la primera capa con 28 nodos y la segunda y última con 10 nodos, para realizar el entrenamiento del modelo se usa el metodo model.fit que ajusta el modelo de acuerdo a nuestros datos de entrenamiento, mientras se va entrenando el modelo muestra la pérdida de datos que tuvo y la exactitud, evaluamos la exactitud después de realizar el entrenamiento y nos da 88%

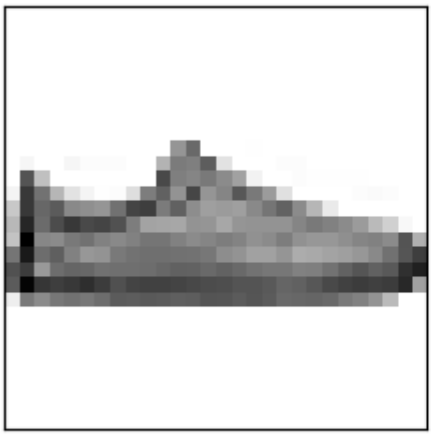
```
313/313 - 1s - loss: 0.3325 - accuracy: 0.8839 - 997ms/epoch - 3ms/step

Test accuracy: 0.883899986743927
```

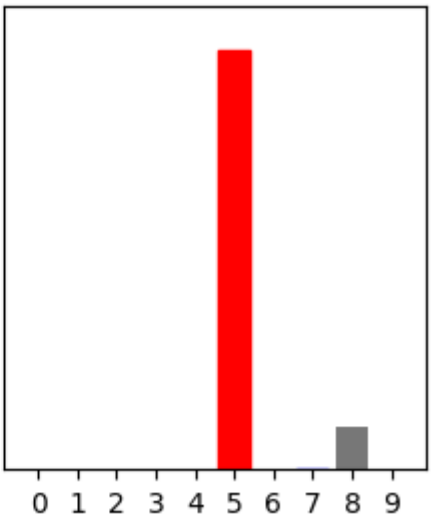
Realizamos la prueba viendo las predicciones con nuestro conjunto de datos que guardamos para esto, con lo que se vio que el conjunto de ropa dentro de los datos que tuvo la mayor cantidad de clasificaciones correctas fue de ankle boot, se graficaron algunas de las predicciones realizadas

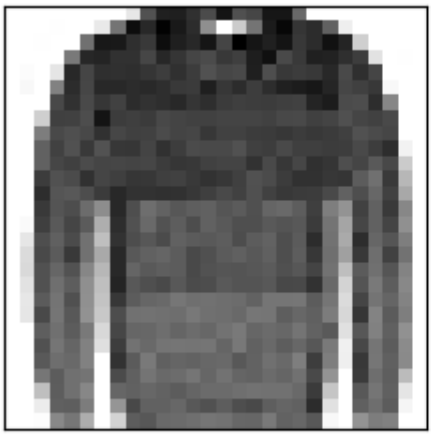
Ankle boot 100% (Ankle boot)



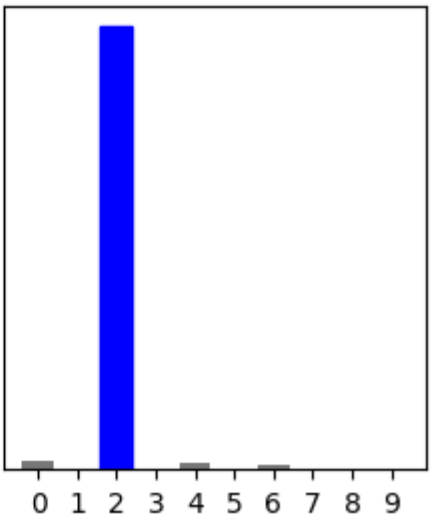


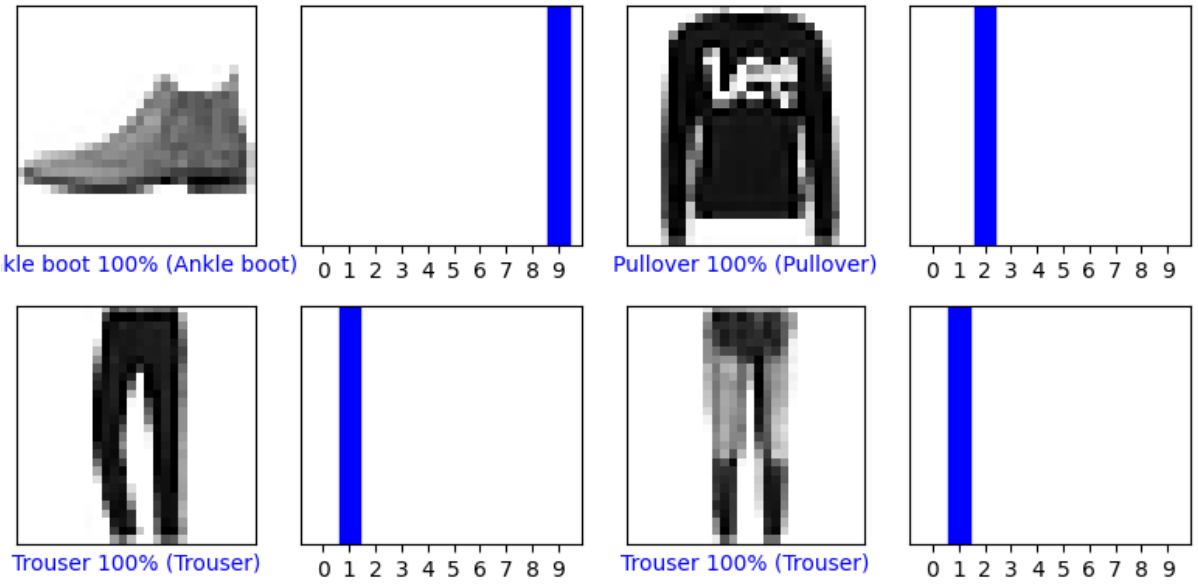
Sandal 91% (Sneaker)





Pullover 96% (Pullover)





Código

```
# TensorFlow y tf.keras
import tensorflow as tf
from tensorflow import keras

# Librerias de ayuda
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)

fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress',
'Coat','Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

train_images.shape

len(train_labels)

train_labels

test_images.shape

len(test_labels)
```

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()

train_images = train_images / 255.0

test_images = test_images / 255.0

plt.figure(figsize=(10,10))
for i in range(15):#primeros 15 imagenes
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),#si cambia este num,
linea 32 error, debido a que son los pixeles
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=15)# los epochs se pueden
modificar

test_loss, test_acc = model.evaluate(test_images,  test_labels,
verbose=2)

print('\nTest accuracy:', test_acc)

predictions = model.predict(test_images)

predictions[0]
```

```
np.argmax(predictions[0])

test_labels[0]

def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array,
true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array, true_label[i]
  plt.grid(False)
  plt.xticks(range(10))#no se pueden cambiar el rango
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')


i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
```

```
plot_value_array(i, predictions[i],  test_labels)
plt.show()


i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()


i = 20   #nueva
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()


num_rows = 2
num_cols = 2
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()


img = test_images[1]

print(img.shape)


img = (np.expand_dims(img,0))

print(img.shape)
```

```
predictions_single = model.predict(img)

print(predictions_single)



plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)

np.argmax(predictions_single[0])
```