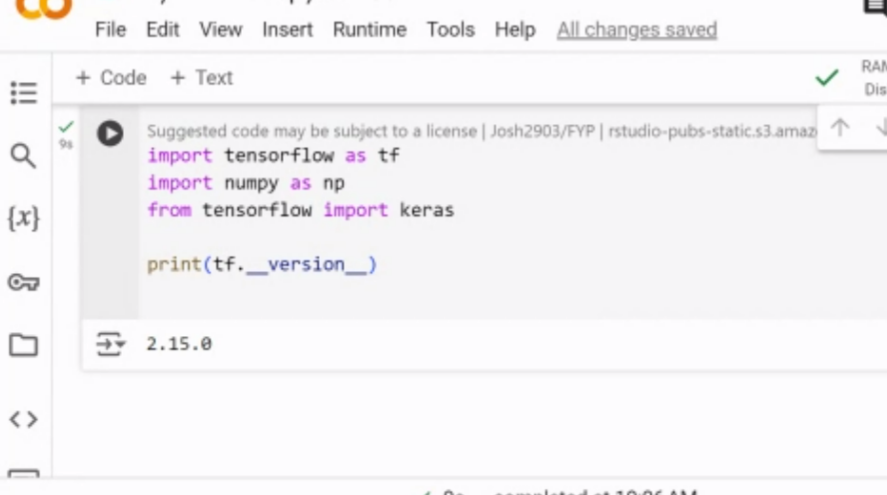
Day 1

Tensor Flow

Keras

Colab





My file: 





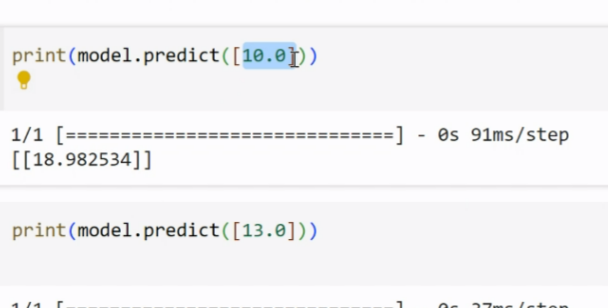
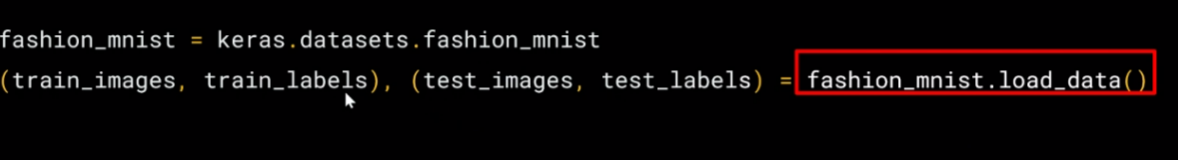


Image classification





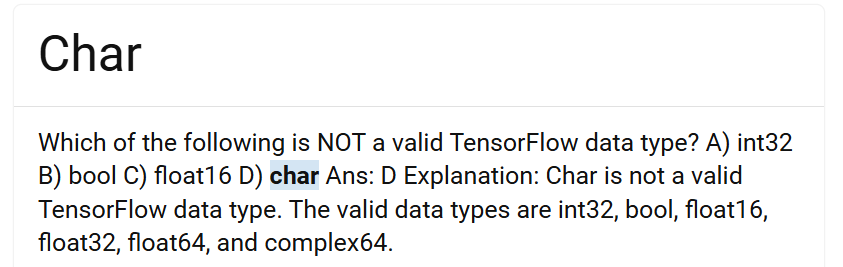
import numpy as np  
import matplotlib.pyplot as plt

index = 0  
np.set\_printoptions(linewidth=320)  
print(f'LABEL:{training\_labels[index]}')  
print(f'\nIMAGE PIXEL ARRAY:\n{training\_images[index]}')  
plt.imshow(training\_images[index])  
training\_images = training\_images / 255.0  
test\_images = test\_images / 255.0

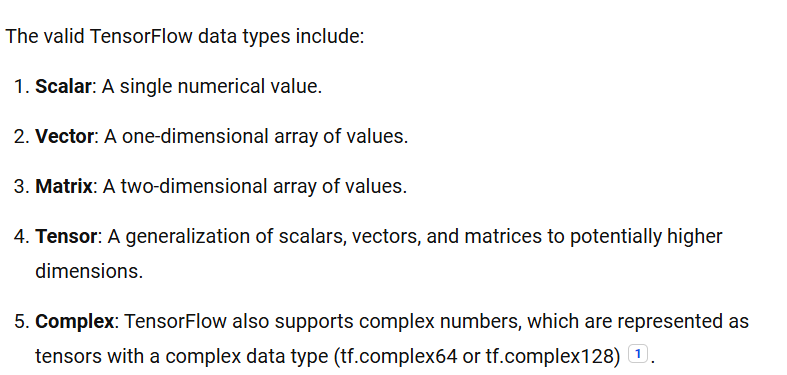
import tensorflow as tf  
print(tf.**version**)   
fmnist= tf.keras.datasets.fashion\_mnist  
(training\_images, training\_labels), (test\_images, test\_labels) = fmnist.load\_data()  
import numpy as np  
import matplotlib.pyplot as plt  
index = 0  
np.set\_printoptions(linewidth=320)  
print(f'LABEL:{training\_labels[index]}')  
print(f'\nIMAGE PIXEL ARRAY:\n{training\_images[index]}')

plt.imshow(training\_images[index])  
training\_images = training\_images / 255.0  
test\_images = test\_images / 255.0

Second Poll

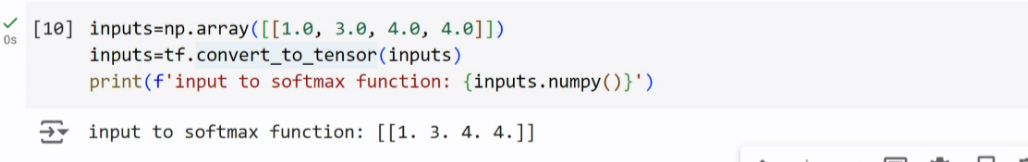


[TensorFlow MCQs and Answers With Explanation (freshersnow.com)](https://www.freshersnow.com/tensorflow-mcqs-and-answers-with-explanation/#:~:text=Which%20of%20the%20following%20is%20NOT%20a%20valid,are%20int32%2C%20bool%2C%20float16%2C%20float32%2C%20float64%2C%20and%20complex64.)



**DAY 2**

model=tf.keras.models.Sequential([tf.keras.layers.flatten(),  
tf.keras.layers.Dense(128,activation=tf.nn.relu(),)]  
tf.keras.layers.Dense(10,activation=tf.nn.softmax())])

****

inputs=np.array([1.0,3.0,4.0,4.0])  
inputs=tf.convert\_to\_tensor(inputs)  
print(f'input to softmax:{inputs.numpy()}')

outputs=tf.keras.activations.softmax(inputs)  
print(f'output of softmax function: {outputs.numpy()}')

model.compile(optimizer=tf.optimizers.Adam(),  
              loss='sparse\_categorical\_crossentropy',  
              metrics=['accuracy'])

model.fit(training\_images, training\_labels, epochs=5)

**Evaluating model on unseen DATA**

#Load the Fashion MNIST dataset  
fmnist = tf.keras.datasets.fashion\_mnist

# Load the training and test split of the Fashion MNIST dataset

(training\_images, training\_labels), (test\_images, test\_labels) = fmnist.load\_data()  
##print a training image (both as an image and a numpy array), and a training label to see:  
  
import numpy as np  
import matplotlib.pyplot as plt

# You can put between 0 to 59999 here

index = 0

# Set number of characters per row when printing

np.set\_printoptions(linewidth=320)

# Print the label and image

print(f'LABEL: {training\_labels[index]}')  
print(f'\nIMAGE PIXEL ARRAY:\n {training\_images[index]}')

# Visualize the image

plt.imshow(training\_images[index])

# Normalize the pixel values of the train and test images

training\_images = training\_images / 255.0  
test\_images = test\_images / 255.0

# Build the classification model

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),  
tf.keras.layers.Dense(128, activation=tf.nn.relu),  
tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

# Declare sample inputs and convert to a tensor

inputs = np.array([[1.0, 3.0, 4.0, 2.0]])  
inputs = tf.convert\_to\_tensor(inputs)  
print(f'input to softmax function: {inputs.numpy()}')

# Feed the inputs to a softmax activation function

outputs = tf.keras.activations.softmax(inputs)  
print(f'output of softmax function: {outputs.numpy()}')

# Get the sum of all values after the softmax

sum = tf.reduce\_sum(outputs)  
print(f'sum of outputs: {sum}')

# Get the index with highest value

prediction = np.argmax(outputs)  
print(f'class with highest probability: {prediction}')  
  
##compiling it with an optimizer and loss function  
model.compile(optimizer = tf.optimizers.Adam(),  
loss = 'sparse\_categorical\_crossentropy',  
metrics=['accuracy'])  
  
## train it by calling [model.fit](https://model.fit)()  
model.fit(training\_images, training\_labels, epochs=5).

# Evaluate the model on unseen data

model.evaluate(test\_images, test\_labels)

**QUES:** Let's now look at the layers in your model. Experiment with different values for the dense layer with 512 neurons. What different results do you get for loss, training time etc? Why do you think that's the case?

**Call back:**

import tensorflow as tf

# Instantiate the dataset API

fmnist = tf.keras.datasets.fashion\_mnist

# Load the dataset

(x\_train, y\_train),(x\_test, y\_test) = fmnist.load\_data()

# Normalize the pixel values

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# Creating a Callback class

class myCallback(tf.keras.callbacks.Callback):  
def on\_epoch\_end(self, epoch, logs={}):  
'''  
Halts the training after reaching 60 percent accuracy  
  
Args:  
epoch (integer) - index of epoch (required but unused in the function definition below)  
logs (dict) - metric results from the training epoch  
'''  
  
# Check accuracy  
if(logs.get('loss') < 0.4):  
  
# Stop if threshold is met  
print("\nLoss is lower than 0.4 so cancelling training!")  
self.model.stop\_training = True

# Instantiate class

callbacks = myCallback()

# Define and compile the model

# Define the model

model = tf.keras.models.Sequential([  
tf.keras.layers.Flatten(input\_shape=(28, 28)),  
tf.keras.layers.Dense(512, activation=tf.nn.relu),  
tf.keras.layers.Dense(10, activation=tf.nn.softmax)  
])

# Compile the model

model.compile(optimizer=tf.optimizers.Adam(),  
loss='sparse\_categorical\_crossentropy',  
metrics=['accuracy'])  
#Train the model

# Train the model with a callback

model.fit(x\_train, y\_train, epochs=10, callbacks=[callbacks])

Optional Challenge: Modify the code to make the training stop when the accuracy metric exceeds 60%.