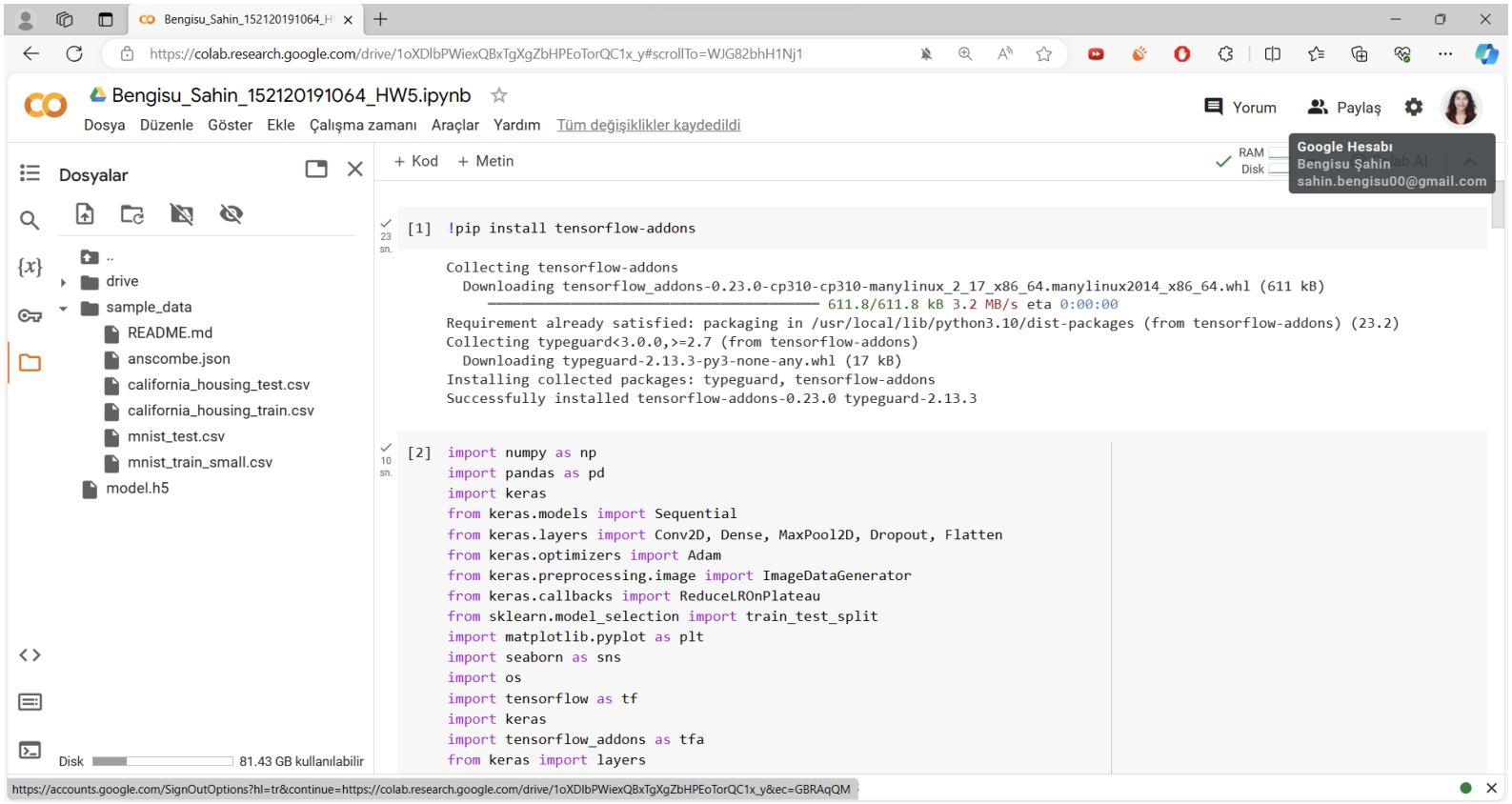
Bengisu Şahin

152120191064

INTRODUCTION to DEEP LEARNING HW5 Report

Gear Fault Datasetinin LSTM, GRU, CNN1D ya da Dense NN sınıflandırıcılarını kullanarak sınıflandırılması, model oluşturulması ve oluşturulan modelin testinin yapılması istendi. Classifier olarak GRU ve Dense classifierlarını kullanıldı ve modele layerlar eklendi.



- Gerekli kütüphaneler kuruldu ve uygulamaya import edildi. Sonrasında drive hesabımdan ödev dosyasında verilmiş olan data çekildi.Sonrasında google drive'da bulunan dataset np.load fonkisyonu ile X_trian değişkenine alındı ve transpose işlemi ile de kodda kullanıma hazır hale getirildi. Types değişkeni altında kaç çeşit sınıflandırmanın olduğu belirlendi ve sonraki adımda da bu sınıflandırmaların label isimleri oluşturuldu ve ekrana basıldı.
- Modelin eğitim adımında kullanılmak üzere Y_train ve X_train değişkenleri oluşturuldu.
- Aşağıdaki görselde görülebileceği gibi model oluşturmak için gru ve dense kullanıldı ve katmanlar eklendi. Katmanlar ekledikten sonra modelin özeti ekrana basıldı, bu şekilde modelde bulunan katmanlar, katman özellikleri, parametreler gibi özellikler görülebilir hale geldi.

- Yardımcı kod dosyalarındaki kodlar kullanılarak checkpoint, early stop gibi özellikler ayarlandı. Sonrasında da model.fit fonksiyonu kullanılarak modelin eğitimi gerçekleştirildi. Epoch değeri olarak 50 verildi.

```
adam = Adam(lr=1e-3)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=adam)
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.le
```

```
[18] # Set a learning rate annealer
     model_name_save= '/content/hw5_model.hdf5'
     checkpoint = tf.keras.callbacks.ModelCheckpoint(model_name_save, save_freq='epoch', monitor='val_accuracy', verbose=1, save_best_only=T
     early_stop = tf.keras.callbacks.EarlyStopping(  monitor='val_accuracy', patience=12, verbose=1,  mode='max',restore_best_weights=False)
     reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(  monitor='val_accuracy', factor=0.3, patience=7,  min_lr=1e-5, verbose=1,  mode='max'
```

```
[19] X_train.shape

     (598, 1, 3600)
```

```
[20] model.fit(X_train, Y_train, validation_data=(X_val, Y_val), epochs=50, batch_size=64)

     10/10 [==============================] - 0s 13ms/step - loss: 0.0355 - accuracy: 0.9983 - val_loss: 0.2237 - val_accuracy: 0.9533
     Epoch 23/50
     10/10 [==============================] - 0s 12ms/step - loss: 0.0322 - accuracy: 0.9983 - val_loss: 0.2210 - val_accuracy: 0.9533
     Epoch 24/50
     10/10 [==============================] - 0s 13ms/step - loss: 0.0295 - accuracy: 0.9983 - val_loss: 0.2190 - val_accuracy: 0.9533
     Epoch 25/50
     10/10 [==============================] - 0s 12ms/step - loss: 0.0270 - accuracy: 0.9983 - val_loss: 0.2174 - val_accuracy: 0.9533
     Epoch 26/50
     10/10 [==============================] - 0s 13ms/step - loss: 0.0248 - accuracy: 1.0000 - val_loss: 0.2150 - val_accuracy: 0.9533
     Epoch 27/50
```

- Ardından metrik ölçümlerinde kullanılmak üzere Y_pred değişkeni oluşturuluyor, bu değişken oluşturulurken model.predict fonksiyonu kullanılıyor. Öncelikle pred isimli değişkene değer atanıyor ve alınan değerleri sonrasında ölçümlerde kullanmak için belirli adımlardan geçiriyoruz. Flatten yaparak tek boyutlu dizi haline getiriyor. 0 ve 1 olarak ayarlamamız da metrik ölçümlerinde float değerlerde hata vermesi binary değer istemesinden kaynaklıdır.Sonrasında modeli save fonksiyonu ile kaydediyoruz. Ve Modelin test işlemini model.evaluate fonksiyonuna X_test ve Y_test değişkenlerini göndererek yapıyoruz ve son olarak da modelin accuracy değerini hesaplayıp ekrana bastırıyoruz. Şekil 7'de de görüldüğü üzere modelin accuracy değeri %95.74.

Predict Test Data

```
[21] pred = model.predict(X_test, batch_size=1)
     pred = pred.flatten()
     print(pred.round(2))
     Y_pred = np.where(pred > 0.05, 1, 0)
     print(Y_pred)
```

```
188/188 [==============================] - 1s 2ms/step
[0.   0.   0.   ... 0.17 0.03 0.14]
[0 0 0 ... 1 0 1]
```

```
[22] model.save("model.h5")
     new_model = tf.keras.models.load_model('model.h5')
     model.summary()
     score = model.evaluate(X_test, Y_test, batch_size=16)
     print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format
  saving_api.save_model(
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 32)             348864

 dense (Dense)               (None, 1, 8)              264

 dense_1 (Dense)             (None, 1, 9)              81

=================================================================
Total params: 349209 (1.33 MB)
Trainable params: 349209 (1.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
12/12 [==============================] - 0s 14ms/step - loss: 0.1556 - accuracy: 0.9574
accuracy: 95.74%
```

- Son olarak Y_test ve Y_pred değişkenlerinin shapelerinin aynı olmasını sağlıyoruz. accuracy_score ile accuracy değerini oluşturup ekrana print ile bastırıyoruz, confusion_matrix ile de confusion matrixini oluşturuyoruz ve plt ile de ekrana grafik olarak bastırıyoruz.

```
print(Y_test.shape)
Y_true = Y_test.reshape(1692)
print(Y_true.shape)
print(Y_pred.shape)

print('accuracy score:' + str(accuracy_score(Y_true, Y_pred)))

A=confusion_matrix(Y_true, Y_pred)
print('confusion matrix:\n' ,A)

df_cm = pd.DataFrame(A, index = [i for i in "AB"], columns = [i for i in "AB"])
plt.figure(figsize = (2,2))
sn.heatmap(df_cm, annot=True)
```

```
(188, 1, 9)
(1692,)
(1692,)
accuracy score:0.9710401891252955
confusion matrix:
 [[1457   47]
 [   2  186]]
<Axes: >
```