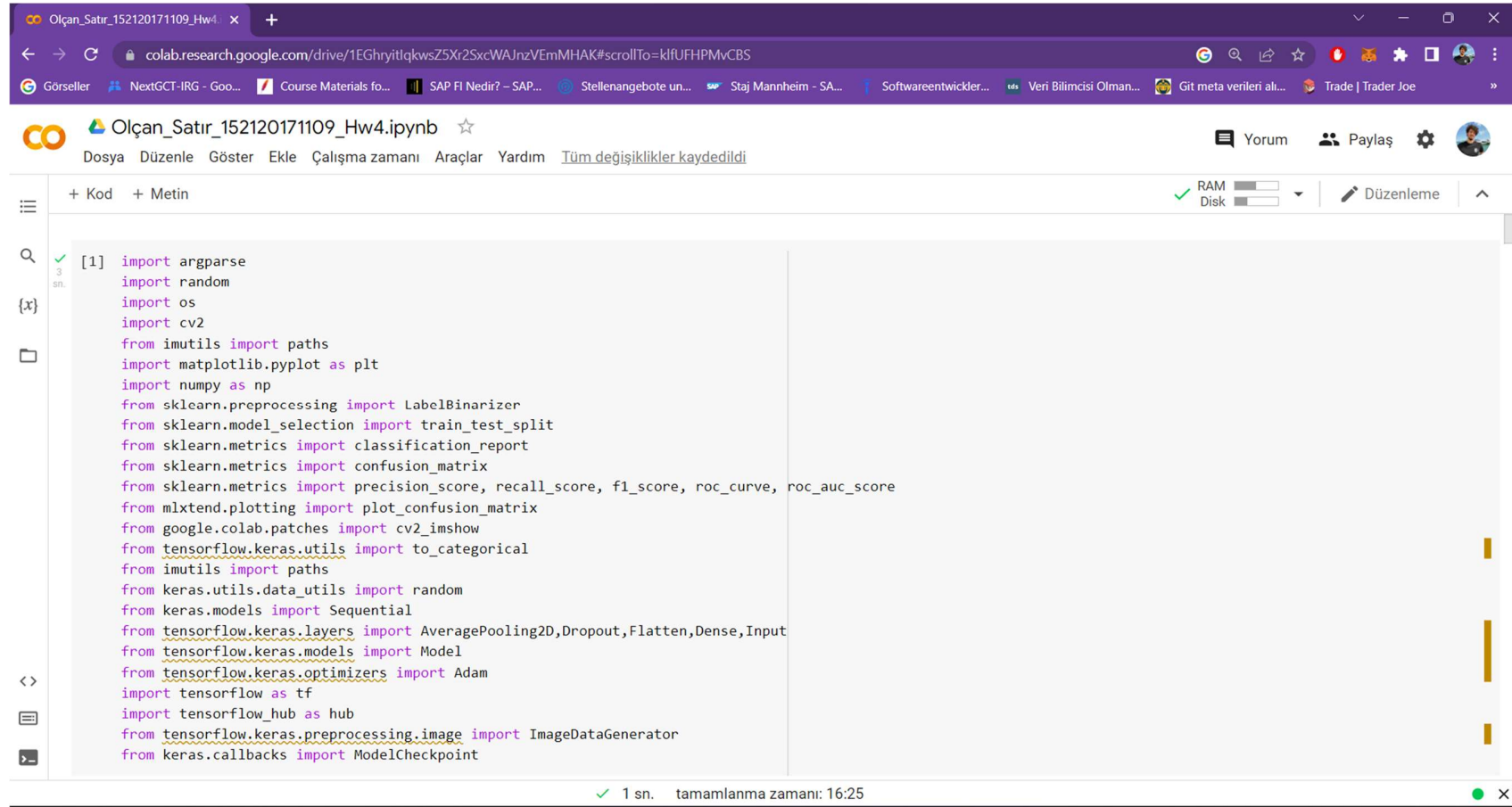


OLÇAN SATIR

152120171109

Deep Learning HW-4

We define our libraries.

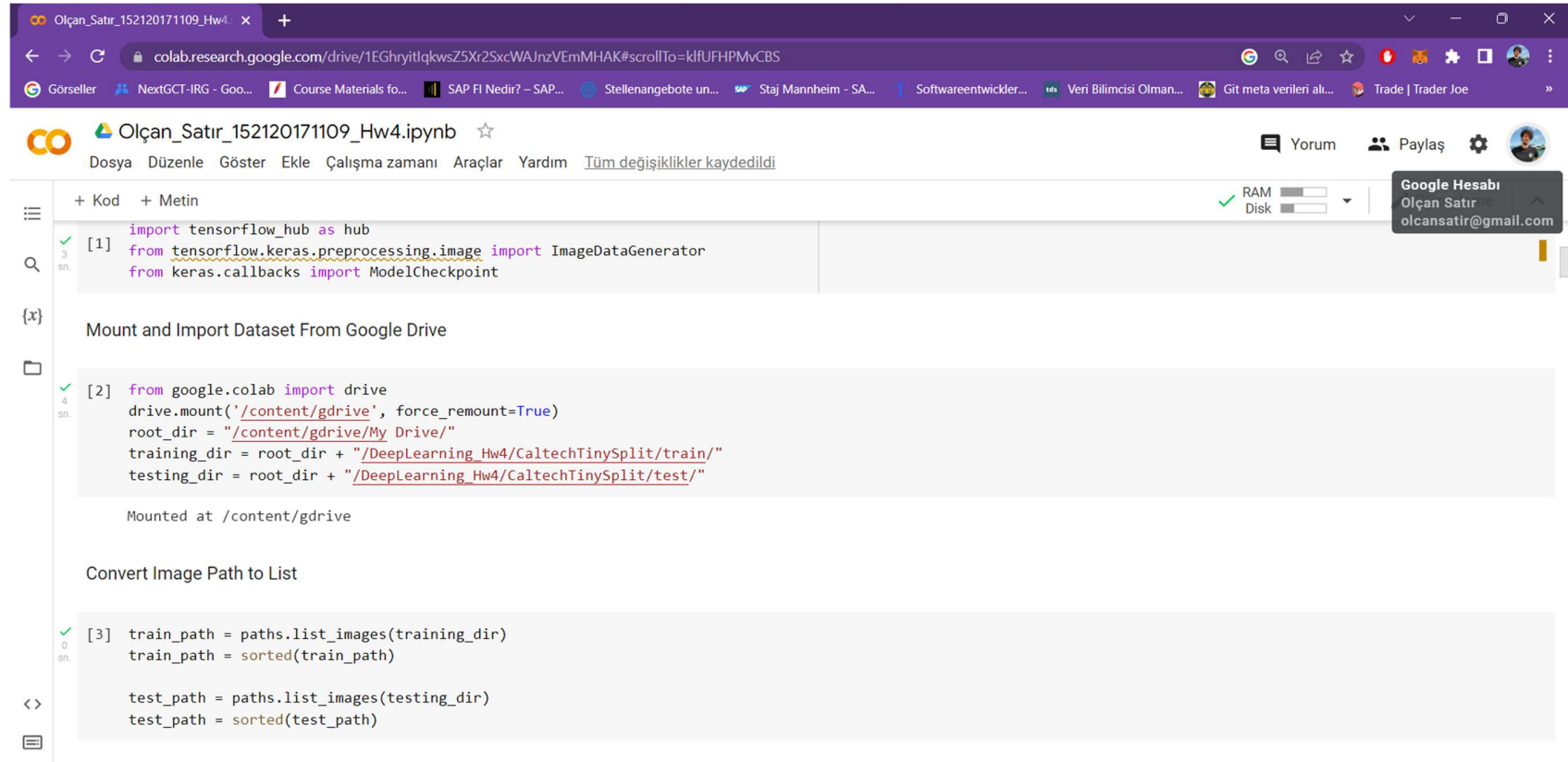


The screenshot shows a Google Colab notebook interface. The browser address bar indicates the notebook is located at colab.research.google.com/drive/1EGhryitqkwsZ5Xr2SxcWAJnzVEmMHAK#scrollTo=klfUFHPMvCBS. The notebook title is "Olçan_Satir_152120171109_Hw4.ipynb". The code cell [1] contains the following imports:

```
[1] import argparse
import random
import os
import cv2
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, roc_curve, roc_auc_score
from mlxtend.plotting import plot_confusion_matrix
from google.colab.patches import cv2_imshow
from tensorflow.keras.utils import to_categorical
from imutils import paths
from keras.utils.data_utils import random
from keras.models import Sequential
from tensorflow.keras.layers import AveragePooling2D, Dropout, Flatten, Dense, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint
```

The status bar at the bottom indicates that the code was executed successfully (green checkmark) and took 1 second to complete, with a total runtime of 16:25.

Here, we first do it from the folder we created for the dataset in the drive. Then we create an image list and sort it.



The screenshot shows a Google Colab notebook titled "Olçan_Satır_152120171109_Hw4.ipynb". The notebook is open in a web browser with the URL colab.research.google.com/drive/1EGhryitlqkwsZ5Xr2SxcWAJnzVEmMHAK#scrollTo=klfUFHPMvCBS. The notebook interface includes a top bar with the file name, a star icon, and buttons for "Dosya", "Düzenle", "Göster", "Ekle", "Çalışma zamanı", "Araçlar", "Yardım", and "Tüm değişiklikler kaydedildi". On the right side, there are buttons for "Yorum", "Paylaş", and a user profile icon. A "Google Hesabı" dropdown menu is visible, showing the user's name "Olçan Satır" and email "olcansatir@gmail.com". The notebook content is divided into three sections: 1. Importing libraries:

```
[1] import tensorflow_hub as hub
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint
```

 2. Mounting and importing the dataset:

```
[2] from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
root_dir = "/content/gdrive/My Drive/"
training_dir = root_dir + "/DeepLearning_Hw4/CaltechTinySplit/train/"
testing_dir = root_dir + "/DeepLearning_Hw4/CaltechTinySplit/test/"
```

 Below the code, the output shows "Mounted at /content/gdrive". 3. Converting image paths to a list:

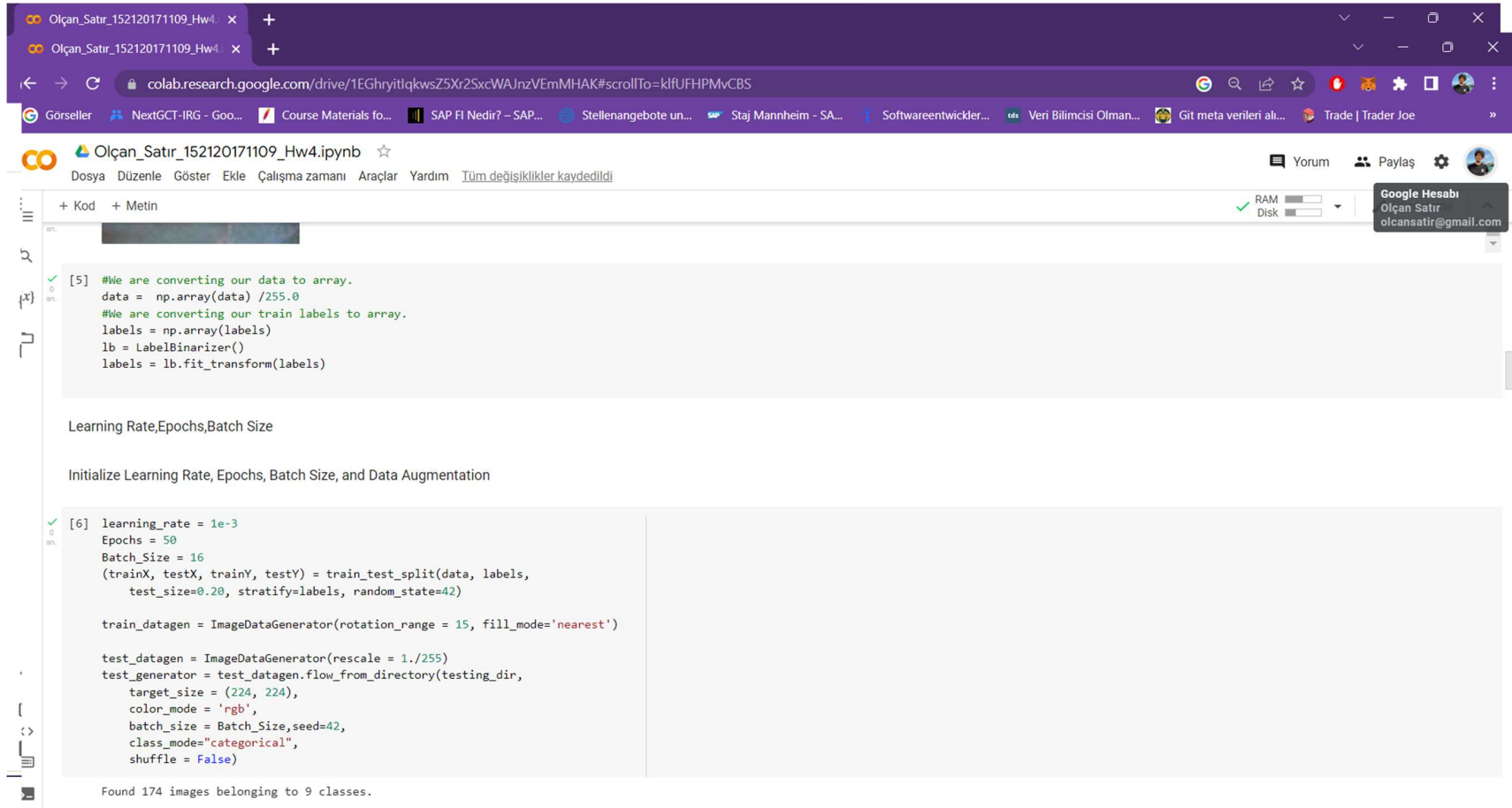
```
[3] train_path = paths.list_images(training_dir)
train_path = sorted(train_path)

test_path = paths.list_images(testing_dir)
test_path = sorted(test_path)
```

 The notebook interface also includes a left sidebar with icons for file explorer, search, and other functions. The top bar shows the current file name and a star icon. The right side of the top bar has buttons for "Yorum", "Paylaş", and a user profile icon. A "Google Hesabı" dropdown menu is visible, showing the user's name "Olçan Satır" and email "olcansatir@gmail.com".

Then we resize the train pictures as 224 by 224 by converting them from BGR to RGB by naming them as inputs and targets. In the for loop below, we print 6 random pictures to see. Here we convert the data and labels into array. Then we define the learning rate epoch number and batch size.

We define train_datagen by editing test datagen and test generator.



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Olçan_Satir_152120171109_Hw4'. The notebook title is 'Olçan_Satir_152120171109_Hw4.ipynb'. The code is written in a Jupyter-style cell with a light blue background. The code in cell [5] converts data and labels to arrays and applies a LabelBinarizer. The code in cell [6] sets learning rate, epochs, batch size, and creates ImageDataGenerators for training and testing. The output of cell [6] shows the number of images found for each class.

```
[5] #We are converting our data to array.
data = np.array(data) /255.0
#We are converting our train labels to array.
labels = np.array(labels)
lb = LabelBinarizer()
labels = lb.fit_transform(labels)

Learning Rate,Epochs,Batch Size

Initialize Learning Rate, Epochs, Batch Size, and Data Augmentation

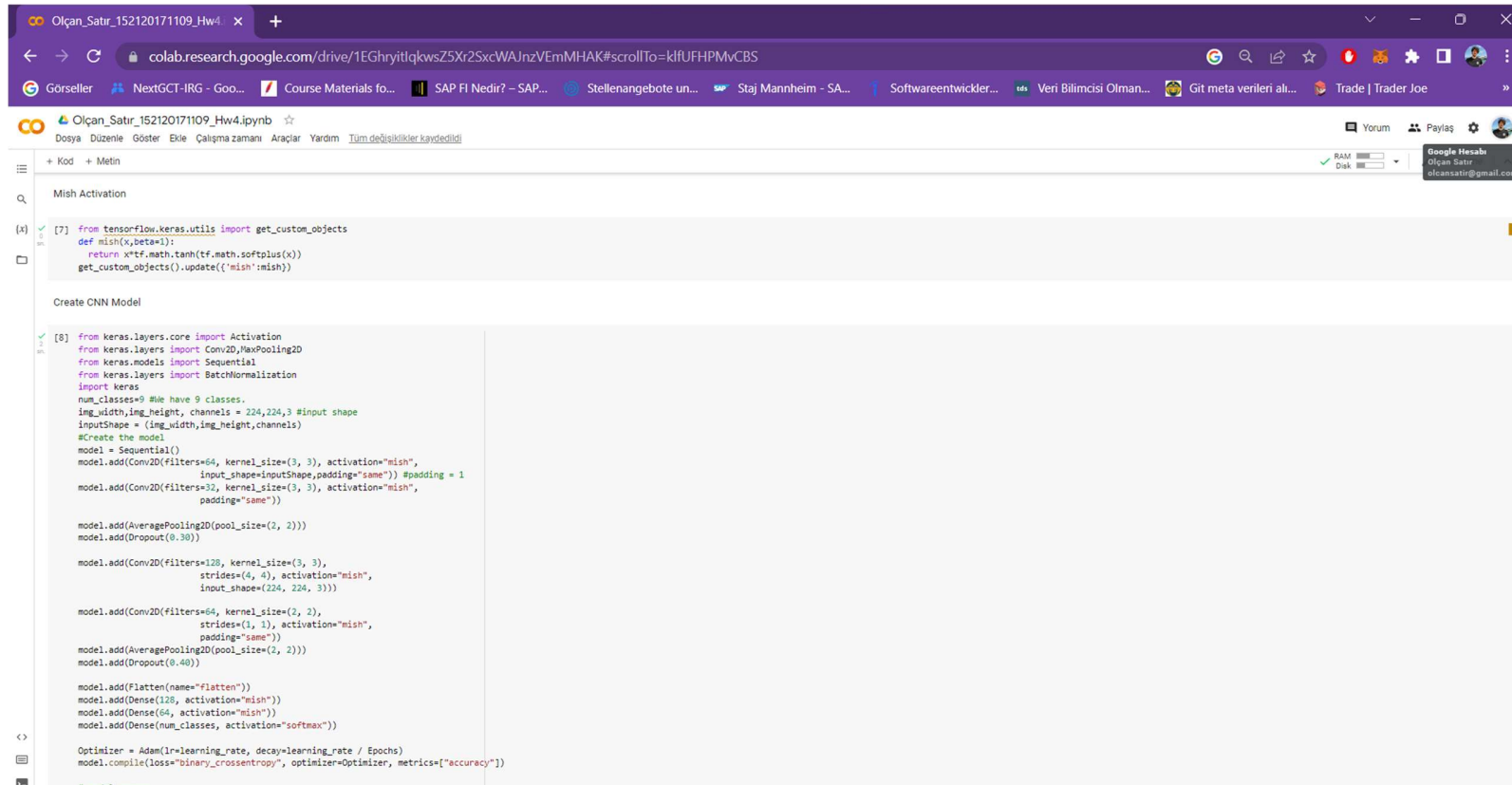
[6] learning_rate = 1e-3
Epochs = 50
Batch_Size = 16
(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=42)

train_datagen = ImageDataGenerator(rotation_range = 15, fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale = 1./255)
test_generator = test_datagen.flow_from_directory(testing_dir,
    target_size = (224, 224),
    color_mode = 'rgb',
    batch_size = Batch_Size,seed=42,
    class_mode="categorical",
    shuffle = False)

Found 174 images belonging to 9 classes.
```

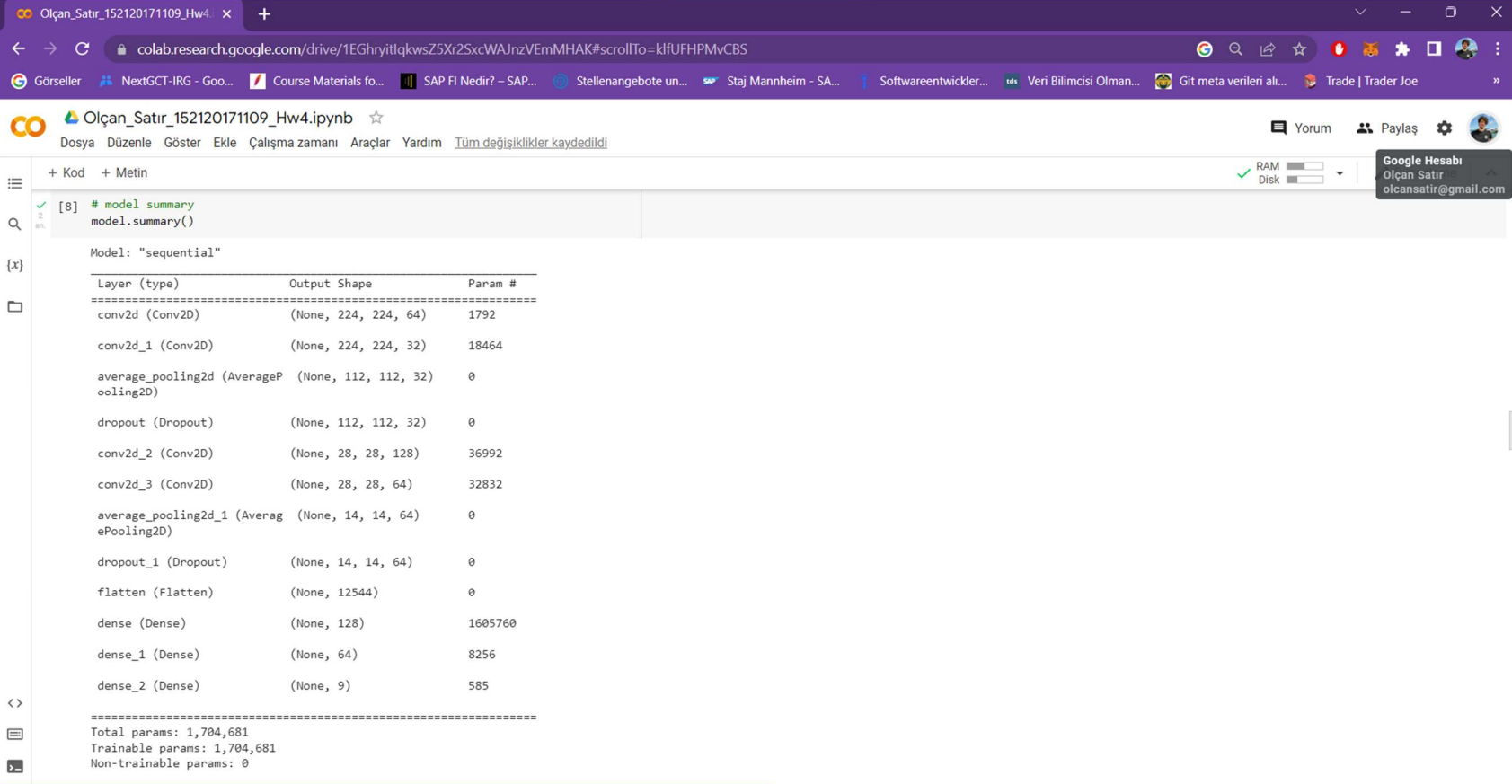
Then we create it as a function to use the mish activation function. Then we create our model. We make conv 2 times while creating the model. In the first conv, we use a 3 by 3 filter and our filter number is 64 padding once and we choose mish as the activation function. Then, in the 2nd conv, we filter 32 times with a 3 x 3 filter, our activation function is the same, we mish and we do padding once again. Then we halve our output size with Average Pooling. We use a dropout to prevent overfitting. Then we do the 3rd conv operation and apply a 3 x 3 filter 128 times. Our stride is 4 and our activation function is mish. Then, for the 4th conv, our filter size is 2x2 and we filter 64 times. Then we reduce our output size by doing AveragePooling again. Then we add a dropout again and convert our output in matrix format to vector with 2 fully connected operations. Afterwards, we create our model by giving our num_classes as 9 since there are 9 classes in total. We use Adam as the optimizer and binary crossentropy as the loss function.



The screenshot shows a Google Colab notebook with two code cells. The first cell defines the Mish activation function, and the second cell creates a CNN model with the following specifications:

- Imports: `tensorflow.keras.utils` for `get_custom_objects`, `keras.layers` for `Conv2D`, `MaxPooling2D`, `AveragePooling2D`, `Dropout`, `Flatten`, `Dense`, and `keras.models` for `Sequential`.
- Model Configuration:
 - `num_classes=9` (9 classes).
 - `img_width, img_height, channels = 224, 224, 3` (input shape).
 - `input_shape = (img_width, img_height, channels)`.
 - Layer 1: `Conv2D(filters=64, kernel_size=(3, 3), activation='mish', input_shape=input_shape, padding='same')` with `padding = 1`.
 - Layer 2: `Conv2D(filters=32, kernel_size=(3, 3), activation='mish', padding='same')`.
 - Layer 3: `AveragePooling2D(pool_size=(2, 2))`.
 - Layer 4: `Dropout(0.30)`.
 - Layer 5: `Conv2D(filters=128, kernel_size=(3, 3), strides=(4, 4), activation='mish', input_shape=(224, 224, 3))`.
 - Layer 6: `Conv2D(filters=64, kernel_size=(2, 2), strides=(1, 1), activation='mish', padding='same')`.
 - Layer 7: `AveragePooling2D(pool_size=(2, 2))`.
 - Layer 8: `Dropout(0.40)`.
 - Layer 9: `Flatten(name='flatten')`.
 - Layer 10: `Dense(128, activation='mish')`.
 - Layer 11: `Dense(64, activation='mish')`.
 - Layer 12: `Dense(num_classes, activation='softmax')`.
- Compilation: `model.compile(loss='binary_crossentropy', optimizer=Optimizer, metrics=['accuracy'])`.

Here we summarize our model.

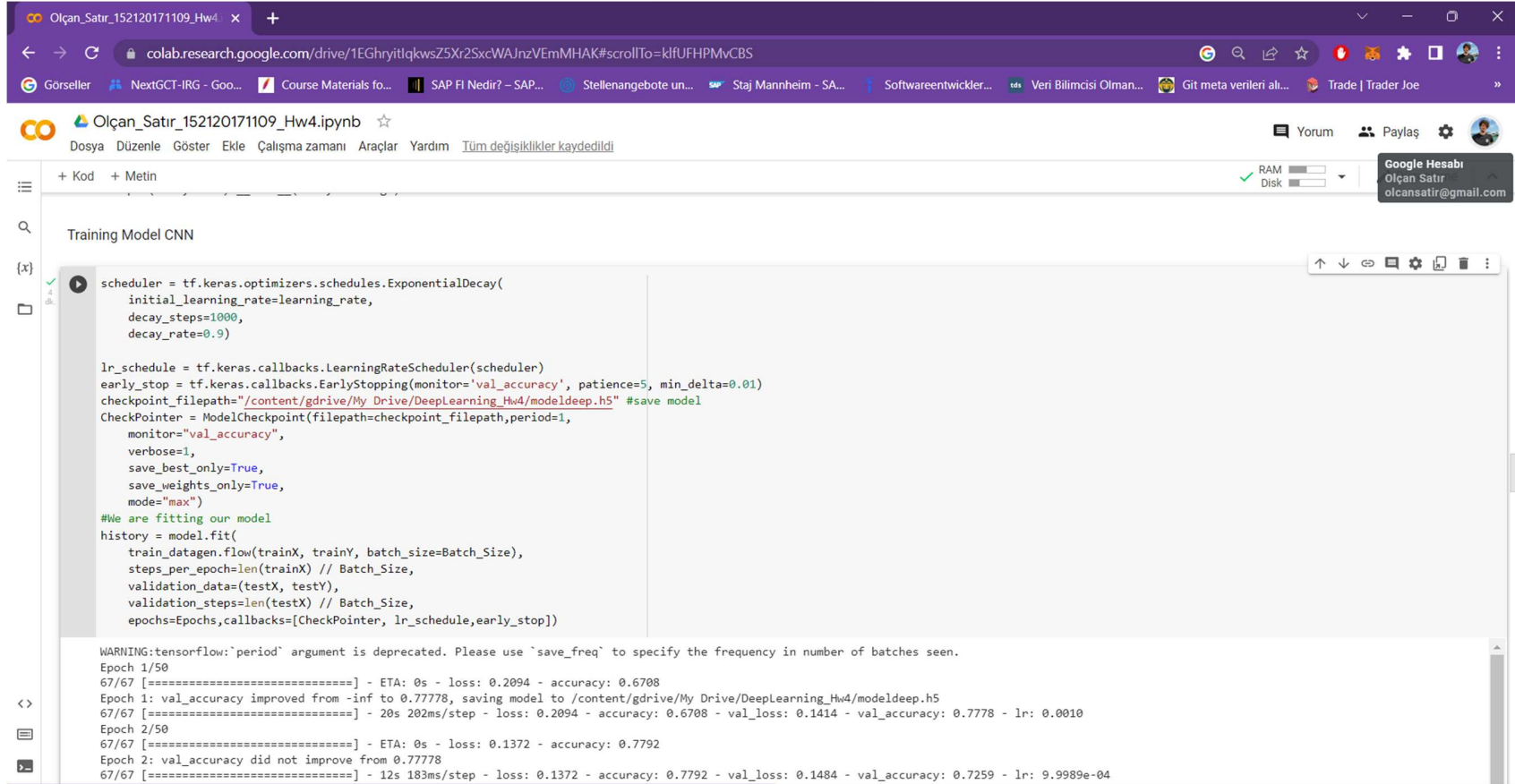


The screenshot shows a Google Colab notebook interface. The browser address bar indicates the notebook is located at `colab.research.google.com/drive/1EGhryitlqkwsZ5Xr2SxcWAJnzVEmMHAK#scrollTo=klfUFHPMvCBS`. The notebook title is "Olçan_Satır_152120171109_Hw4.ipynb". The code cell contains the command `model.summary()`, which has been executed. The output displays a detailed summary of the model's architecture, including the layer type, output shape, and the number of parameters for each layer. The total number of parameters is 1,704,681, with 1,704,681 trainable parameters and 0 non-trainable parameters.

```
[8]: # model summary
model.summary()

Model: "sequential"
=====
Layer (type)                Output Shape              Param #
-----
conv2d (Conv2D)              (None, 224, 224, 64)      1792
conv2d_1 (Conv2D)            (None, 224, 224, 32)      18464
average_pooling2d (AverageP (None, 112, 112, 32)      0
ooling2d)
dropout (Dropout)            (None, 112, 112, 32)      0
conv2d_2 (Conv2D)            (None, 28, 28, 128)       36992
conv2d_3 (Conv2D)            (None, 28, 28, 64)        32832
average_pooling2d_1 (Averag (None, 14, 14, 64)        0
ePooling2d)
dropout_1 (Dropout)          (None, 14, 14, 64)        0
flatten (Flatten)            (None, 12544)              0
dense (Dense)                 (None, 128)                1605760
dense_1 (Dense)               (None, 64)                 8256
dense_2 (Dense)               (None, 9)                  585
=====
Total params: 1,704,681
Trainable params: 1,704,681
Non-trainable params: 0
```

Here we use a CheckPointer to save our model and define an early stop, if the accuracy value of our model gives close values 5 times in a row, it saves the model. So here, our model stopped at the 16th epoch before 50 epochs were defined. Then we save our model by fitting.



```
Olcan_Satir_152120171109_Hw4.ipynb
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi

+ Kod + Metin
RAM
Disk

Google Hesabı
Olcan Satir
olcansatir@gmail.com

Training Model CNN

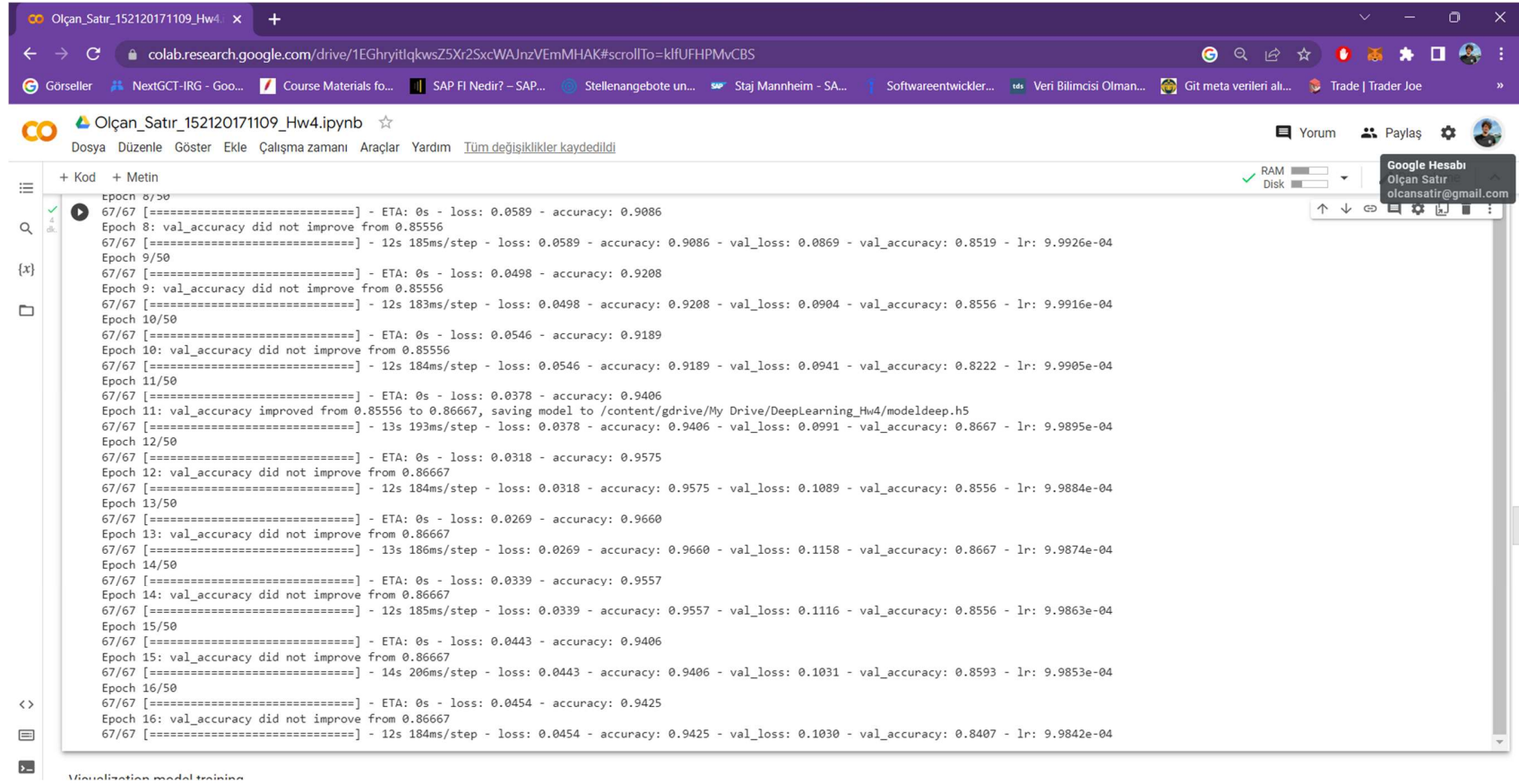
scheduler = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=learning_rate,
    decay_steps=1000,
    decay_rate=0.9)

lr_schedule = tf.keras.callbacks.LearningRateScheduler(scheduler)
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=5, min_delta=0.01)
checkpoint_filepath="/content/gdrive/My Drive/DeepLearning_Hw4/modeldeep.h5" #save model
Checkpoint = ModelCheckpoint(filepath=checkpoint_filepath,period=1,
    monitor="val_accuracy",
    verbose=1,
    save_best_only=True,
    save_weights_only=True,
    mode="max")

#We are fitting our model
history = model.fit(
    train_datagen.flow(trainX, trainY, batch_size=Batch_Size),
    steps_per_epoch=len(trainX) // Batch_Size,
    validation_data=(testX, testY),
    validation_steps=len(testX) // Batch_Size,
    epochs=Epochs,callbacks=[Checkpoint, lr_schedule,early_stop])

WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify the frequency in number of batches seen.
Epoch 1/50
67/67 [=====] - ETA: 0s - loss: 0.2094 - accuracy: 0.6708
Epoch 1: val_accuracy improved from -inf to 0.77778, saving model to /content/gdrive/My Drive/DeepLearning_Hw4/modeldeep.h5
67/67 [=====] - 20s 202ms/step - loss: 0.2094 - accuracy: 0.6708 - val_loss: 0.1414 - val_accuracy: 0.7778 - lr: 0.0010
Epoch 2/50
67/67 [=====] - ETA: 0s - loss: 0.1372 - accuracy: 0.7792
Epoch 2: val_accuracy did not improve from 0.77778
67/67 [=====] - 12s 183ms/step - loss: 0.1372 - accuracy: 0.7792 - val_loss: 0.1484 - val_accuracy: 0.7259 - lr: 9.9989e-04
```

Accuracy and loss values of our model in each epoch.



```
tpocn 0/50
67/67 [=====] - ETA: 0s - loss: 0.0589 - accuracy: 0.9086
Epoch 8: val_accuracy did not improve from 0.85556
67/67 [=====] - 12s 185ms/step - loss: 0.0589 - accuracy: 0.9086 - val_loss: 0.0869 - val_accuracy: 0.8519 - lr: 9.9926e-04
Epoch 9/50
67/67 [=====] - ETA: 0s - loss: 0.0498 - accuracy: 0.9208
Epoch 9: val_accuracy did not improve from 0.85556
67/67 [=====] - 12s 183ms/step - loss: 0.0498 - accuracy: 0.9208 - val_loss: 0.0904 - val_accuracy: 0.8556 - lr: 9.9916e-04
Epoch 10/50
67/67 [=====] - ETA: 0s - loss: 0.0546 - accuracy: 0.9189
Epoch 10: val_accuracy did not improve from 0.85556
67/67 [=====] - 12s 184ms/step - loss: 0.0546 - accuracy: 0.9189 - val_loss: 0.0941 - val_accuracy: 0.8222 - lr: 9.9905e-04
Epoch 11/50
67/67 [=====] - ETA: 0s - loss: 0.0378 - accuracy: 0.9406
Epoch 11: val_accuracy improved from 0.85556 to 0.86667, saving model to /content/gdrive/My Drive/DeepLearning_Hw4/modeldeep.h5
67/67 [=====] - 13s 193ms/step - loss: 0.0378 - accuracy: 0.9406 - val_loss: 0.0991 - val_accuracy: 0.8667 - lr: 9.9895e-04
Epoch 12/50
67/67 [=====] - ETA: 0s - loss: 0.0318 - accuracy: 0.9575
Epoch 12: val_accuracy did not improve from 0.86667
67/67 [=====] - 12s 184ms/step - loss: 0.0318 - accuracy: 0.9575 - val_loss: 0.1089 - val_accuracy: 0.8556 - lr: 9.9884e-04
Epoch 13/50
67/67 [=====] - ETA: 0s - loss: 0.0269 - accuracy: 0.9660
Epoch 13: val_accuracy did not improve from 0.86667
67/67 [=====] - 13s 186ms/step - loss: 0.0269 - accuracy: 0.9660 - val_loss: 0.1158 - val_accuracy: 0.8667 - lr: 9.9874e-04
Epoch 14/50
67/67 [=====] - ETA: 0s - loss: 0.0339 - accuracy: 0.9557
Epoch 14: val_accuracy did not improve from 0.86667
67/67 [=====] - 12s 185ms/step - loss: 0.0339 - accuracy: 0.9557 - val_loss: 0.1116 - val_accuracy: 0.8556 - lr: 9.9863e-04
Epoch 15/50
67/67 [=====] - ETA: 0s - loss: 0.0443 - accuracy: 0.9406
Epoch 15: val_accuracy did not improve from 0.86667
67/67 [=====] - 14s 206ms/step - loss: 0.0443 - accuracy: 0.9406 - val_loss: 0.1031 - val_accuracy: 0.8593 - lr: 9.9853e-04
Epoch 16/50
67/67 [=====] - ETA: 0s - loss: 0.0454 - accuracy: 0.9425
Epoch 16: val_accuracy did not improve from 0.86667
67/67 [=====] - 12s 184ms/step - loss: 0.0454 - accuracy: 0.9425 - val_loss: 0.1030 - val_accuracy: 0.8407 - lr: 9.9842e-04
```


Olçan_Satır_152120171109_Hw4

colab.research.google.com/drive/1EGhryitlqkwsZ5Xr2SxcWAJnzVEmMHAK#scrollTo=kIfUFHPMvCBS

Görseller NextGCT-IRG - Goo... Course Materials fo... SAP FI Nedir? - SAP... Stellanangebote un... Staj Mannheim - SA... Softwareentwickler... Veri Bilimcisi Olman... Git meta verileri ali... Trade | Trader Joe

Olçan_Satır_152120171109_Hw4.ipynb

Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi

Yorum Paylaş

Google Hesabı Olçan Satır olcansatir@gmail.com

RAM Disk

Visualization model training

```
[10] xlabel = 'Epoch'
      legends = ['Training', 'Validation']

      ylim_pad = [0.01, 0.1]
      plt.figure(figsize=(20, 8))

      # Plot training & validation Accuracy values
      y1 = history.history['accuracy']
      y2 = history.history['val_accuracy']

      min_y = min(min(y1), min(y2))-ylim_pad[0]
      max_y = max(max(y1), max(y2))+ylim_pad[0]

      plt.subplot(121)
      plt.plot(y1)
      plt.plot(y2)
      plt.title('Model Accuracy', fontsize=15)
      plt.xlabel(xlabel, fontsize=15)
      plt.ylabel('Accuracy', fontsize=15)
      plt.ylim(min_y, max_y)
      plt.legend(legends, loc='upper left')
      plt.tight_layout()

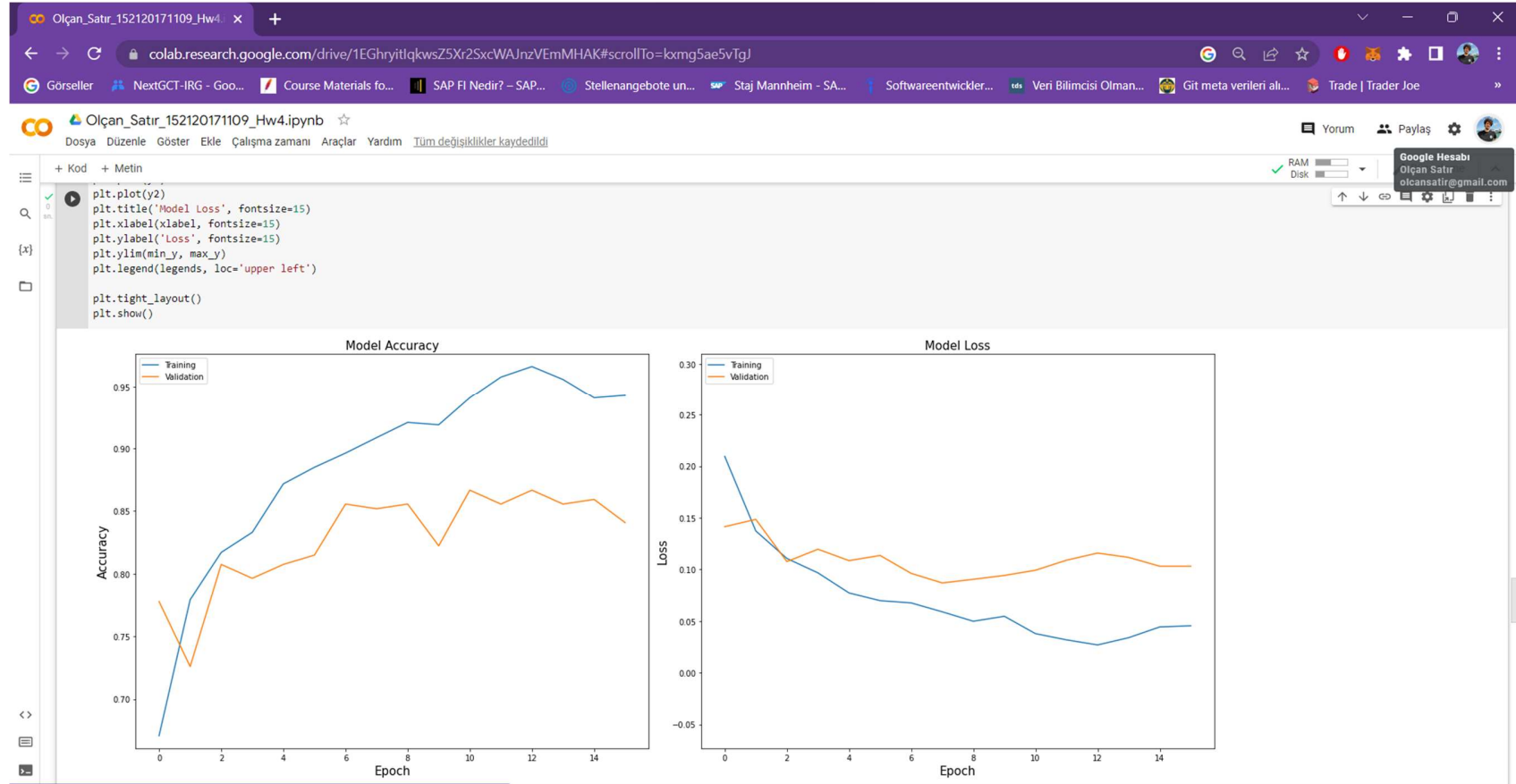
      # Plot training & validation loss values
      y1 = history.history['loss']
      y2 = history.history['val_loss']

      min_y = min(min(y1), min(y2))-ylim_pad[1]
      max_y = max(max(y1), max(y2))+ylim_pad[1]

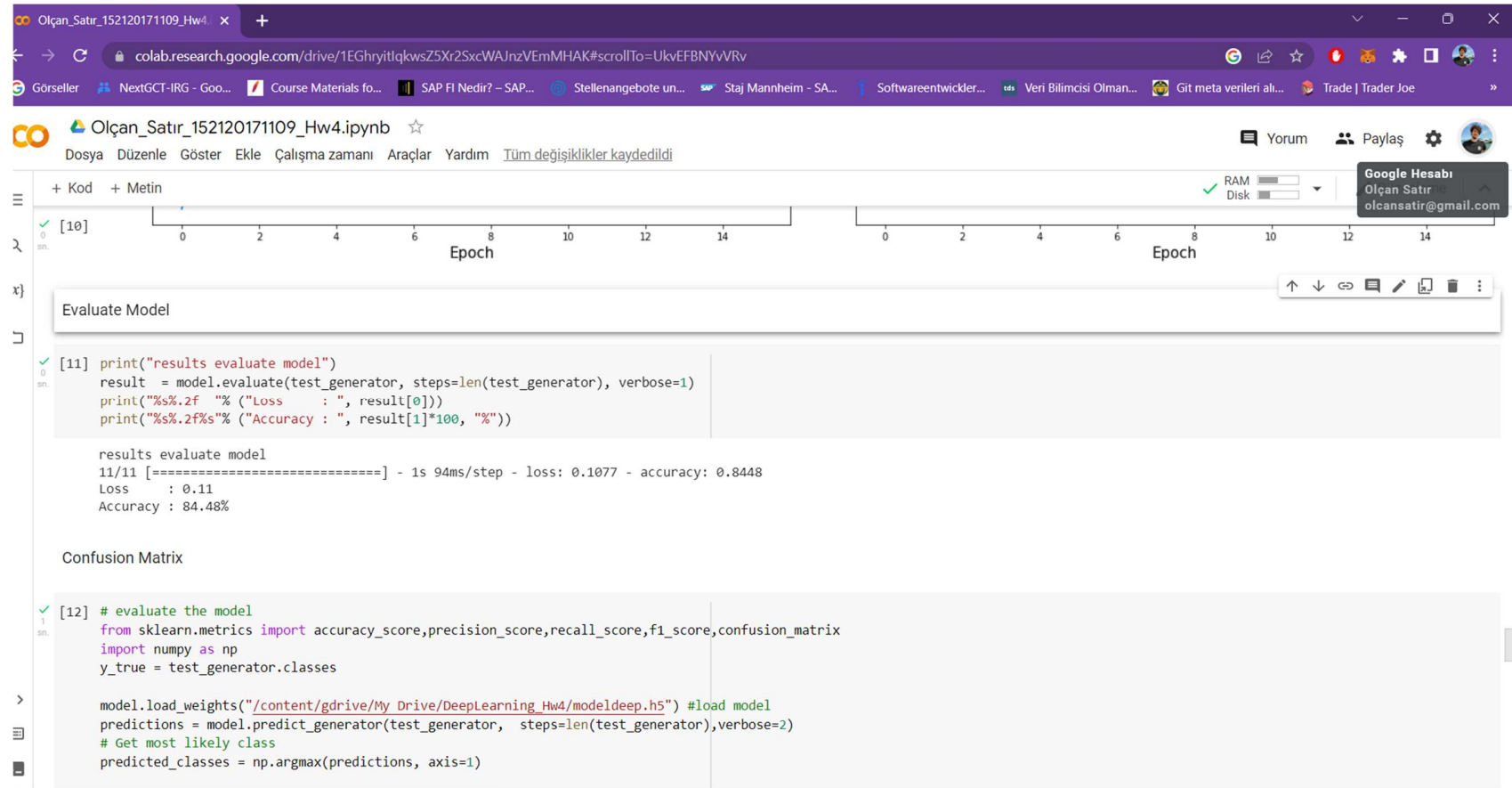
      plt.subplot(122)
      plt.plot(y1)
```

https://accounts.google.com/SignOutOptions?hl=tr&continue=https://colab.research.google... 1 sn. tamamlanma zamanı: 16:25

Here we plot the accuracy and loss values of our model for training and validation.



Then we evaluate our model and print the truth value.



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `colab.research.google.com/drive/1EGhryitlqkwsZ5Xr2SxcWAJnzVEmMHAK#scrollTo=UkvEFBNyVVRv`. The notebook title is "Olçan_Satir_152120171109_Hw4.ipynb". The interface includes a top bar with navigation icons and a right sidebar with a "Google Hesabı" (Google Account) dropdown showing "Olçan Satir" and "olcansatir@gmail.com".

The notebook content is divided into two main sections. The first section, titled "Evaluate Model", contains a code cell [11] that evaluates the model. The output of this cell shows the following results:

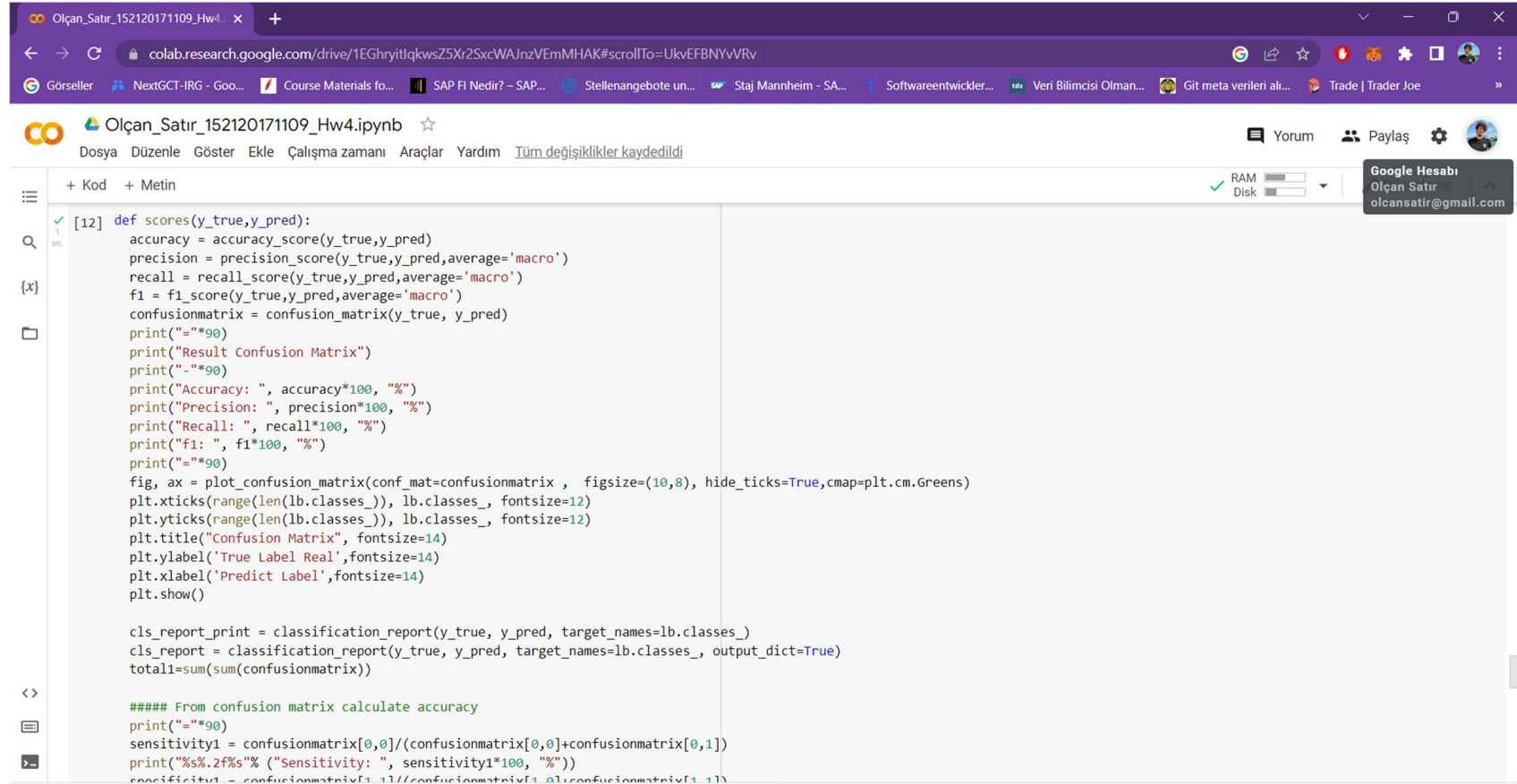
```
results evaluate model
11/11 [=====] - 1s 94ms/step - loss: 0.1077 - accuracy: 0.8448
Loss      : 0.11
Accuracy  : 84.48%
```

The second section, titled "Confusion Matrix", contains a code cell [12] that imports necessary metrics and loads the model weights. The code in this cell is as follows:

```
[12] # evaluate the model
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import numpy as np
y_true = test_generator.classes

model.load_weights("/content/gdrive/My Drive/DeepLearning_Hw4/modeldeep.h5") #load model
predictions = model.predict_generator(test_generator, steps=len(test_generator), verbose=2)
# Get most likely class
predicted_classes = np.argmax(predictions, axis=1)
```

Here we create and print the confusion matrix of our model, and then print the F1 score values at the bottom. Our model correctly predicted 44 of the faces in our test data and correctly predicted 77 of the motorbikes.



```
[12] def scores(y_true,y_pred):
    accuracy = accuracy_score(y_true,y_pred)
    precision = precision_score(y_true,y_pred,average='macro')
    recall = recall_score(y_true,y_pred,average='macro')
    f1 = f1_score(y_true,y_pred,average='macro')
    confusionmatrix = confusion_matrix(y_true, y_pred)
    print("*90")
    print("Result Confusion Matrix")
    print("-"*90)
    print("Accuracy: ", accuracy*100, "%")
    print("Precision: ", precision*100, "%")
    print("Recall: ", recall*100, "%")
    print("f1: ", f1*100, "%")
    print("-"*90)
    fig, ax = plot_confusion_matrix(conf_mat=confusionmatrix, figsize=(10,8), hide_ticks=True,cmap=plt.cm.Greens)
    plt.xticks(range(len(lb.classes_)), lb.classes_, fontsize=12)
    plt.yticks(range(len(lb.classes_)), lb.classes_, fontsize=12)
    plt.title("Confusion Matrix", fontsize=14)
    plt.ylabel('True Label Real',fontsize=14)
    plt.xlabel('Predict Label',fontsize=14)
    plt.show()

    cls_report_print = classification_report(y_true, y_pred, target_names=lb.classes_)
    cls_report = classification_report(y_true, y_pred, target_names=lb.classes_, output_dict=True)
    total=sum(sum(confusionmatrix))

    ##### From confusion matrix calculate accuracy
    print("*90")
    sensitivity1 = confusionmatrix[0,0]/(confusionmatrix[0,0]+confusionmatrix[0,1])
    print("%s%.2f%%" % ("Sensitivity: ", sensitivity1*100, "%"))
    specificity1 = confusionmatrix[1,1]/(confusionmatrix[1,1]+confusionmatrix[1,0])
```

Olçan_Satır_152120171109_Hw4

colab.research.google.com/drive/1EGhyitqlkwsZ5Xr2SxcWAJnzVEmMHAK#scrollTo=lkuEF6livZ6L

GorsellerNextGCT-IRG - Goo...Course Materials fo...SAP FI Nedir? - SAP...Stellenangebote un...Staj Mannheim - SA...Softwareentwickler...Veri Bilimcisi Olman...Git meta verileri ali...Trade | Trader Joe

Olçan_Satır_152120171109_Hw4.ipynb

DosyaDüzenleGösterEkleÇalışma zamanıAraçlarYardımTüm değişiklikler kaydedildi

+ Kod+ Metin

```
##### From confusion matrix calculate accuracy
print("="*90)
sensitivity1 = confusionmatrix[0,0]/(confusionmatrix[0,0]+confusionmatrix[0,1])
print("%s%.2f%%" % ("Sensitivity: ", sensitivity1*100, "%"))
specificity1 = confusionmatrix[1,1]/(confusionmatrix[1,0]+confusionmatrix[1,1])
print("%s%.2f%%" % ("Specificity: ", specificity1*100, "%"))

print("\n")
print("="*90)
print("="*90)
print(cls_report_print)
print("="*90)

scores(y_true,predicted_classes)

Recall: 69.91426611796983 %
f1: 71.38758594235347 %

=====

Confusion Matrix

Faces 41 0 0 0 0 0 0 0 0 0
Motorbikes 1 77 0 0 1 0 1 1 0
camera 0 2 1 1 0 0 1 0 0
cannon 0 2 0 0 0 0 0 0 0
cellphone 0 0 0 0 0 0 0 0 0
flamingo 2 0 0 1 0 3 1 1 0
hawksbill 0 0 0 0 0 0 0 0 0
ibis 1 0 0 0 0 1 2 4 0
pizza 2 0 0 0 0 0 0 0 0
```

Google HesabıOlçan Satırolcansatir@gmail.com

Olçan_Satır_152120171109_Hw4

colab.research.google.com/drive/1EGhryitlqkwsZ5Xr2SxcWAJnzVEmMHAK#scrollTo=IkuEF6liVZ6L

GörsellerNextGCT-IRG - Goo...Course Materials fo...SAP FI Nedir? - SAP...Stellenangebote un...Staj Mannheim - SA...Softwareentwickler...Veri Bilimcisi Olman...Git meta verileri al...Trade | Trader Joe

Olçan_Satır_152120171109_Hw4.ipynb

DosyaDüzenleGösterEkleÇalışma zamanıAraçlarYardımTüm değişiklikler kaydedildi

+ Kod + Metin

RAMDisk

Google HesabıOlçan Satırolcansatir@gmail.com

1 an

{x}

<>

ibis

100001240

200000000

FacesMotorbikescamera cannoncellphoneflamingohawksbillibispizza

Predict Label

Sensitivity: 100.00%

Specificity: 98.72%

precisionrecallf1-score support

Faces0.881.000.9444

Motorbikes0.950.950.9581

camera1.000.200.335

cannon0.600.600.605

cellphone0.881.000.937

flamingo0.750.380.508

hawksbill0.671.000.8010

ibis0.670.500.578

pizza1.000.670.806

accuracy0.88174

macro avg0.820.700.71174

weighted avg0.880.880.87174

amanı: 16:25