In [1]:

```
# Gerekli Kütüphaneler
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os
from google.colab import drive
```

In [2]:

```
#data çağırma
drive.mount('/content/drive')
base_dir = '/content/drive/MyDrive/data/'

# Öğrenme hızı, Epochs, batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = os.path.join(base_dir, 'test')
CATEGORIES = ["Masked", "No_Mask"]
```

```
Mounted at /content/drive
```

In [3]:

```
print("[Bilgi] Fotğraflar yükleniyor...")

data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)

        data.append(image)
        labels.append(category)
```

```
[Bilgi] Fotğraflar yükleniyor...
```

In [4]:

```
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
```

```
data = np.array(data, dtype="float32")
labels = np.array(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=42)
```

In [5]:

```
# veri büyütmek için eğitim görüntü oluşturucusunu oluşturma
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
```

In [6]:

```
# MobileNetV2 ağını yükleme, "FC katman setleri"
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128
, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobile
net_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9412608/9406464 [==============================] - 0s 0us/step
```

In [7]:

```
#temel modelin üstüne yerleştirilecek modelin başını inşa etme
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

In [8]:

```
# Model, eğiteceğimiz gerçek model olacaktır
model = Model(inputs=baseModel.input, outputs=headModel)
```

In [9]:

```
# temel modeldeki tüm katmanlar üzerinde döngü oluştuema ve ilk eğitim sürecinde güncelle
nmemeleri için onları dondurma
for layer in baseModel.layers:
    layer.trainable = False
```

In [10]:

```
# Modeli derleme
print("[bilgi] model derleniyor...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])
```

```
[bilgi] model derleniyor...
```

In [11]:

```
# Modeli eğitmek
print("[bilgi] model eğitiliyor...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
```

```
    epochs=EPOCHS)
```

```
[bilgi] model eğitiliyor...
Epoch 1/20
144/144 [==============================] - ETA: 0s - loss: 0.5911 - accuracy: 0.7121WARNI
NG:tensorflow:Your input ran out of data; interrupting training. Make sure that your data
set or generator can generate at least `steps_per_epoch * epochs` batches (in this case,
36 batches). You may need to use the repeat() function when building your dataset.
144/144 [==============================] - 238s 2s/step - loss: 0.5903 - accuracy: 0.7128
- val_loss: 0.3144 - val_accuracy: 0.8944
Epoch 2/20
144/144 [==============================] - 193s 1s/step - loss: 0.3054 - accuracy: 0.8876
Epoch 3/20
144/144 [==============================] - 192s 1s/step - loss: 0.2262 - accuracy: 0.9235
Epoch 4/20
144/144 [==============================] - 193s 1s/step - loss: 0.1969 - accuracy: 0.9332
Epoch 5/20
144/144 [==============================] - 197s 1s/step - loss: 0.1810 - accuracy: 0.9301
Epoch 6/20
144/144 [==============================] - 199s 1s/step - loss: 0.1552 - accuracy: 0.9485
Epoch 7/20
144/144 [==============================] - 193s 1s/step - loss: 0.1421 - accuracy: 0.9511
Epoch 8/20
144/144 [==============================] - 189s 1s/step - loss: 0.1314 - accuracy: 0.9504
Epoch 9/20
144/144 [==============================] - 187s 1s/step - loss: 0.1281 - accuracy: 0.9558
Epoch 10/20
144/144 [==============================] - 188s 1s/step - loss: 0.1151 - accuracy: 0.9632
Epoch 11/20
144/144 [==============================] - 193s 1s/step - loss: 0.1062 - accuracy: 0.9601
Epoch 12/20
144/144 [==============================] - 190s 1s/step - loss: 0.0988 - accuracy: 0.9673
Epoch 13/20
144/144 [==============================] - 189s 1s/step - loss: 0.1086 - accuracy: 0.9611
Epoch 14/20
144/144 [==============================] - 190s 1s/step - loss: 0.1031 - accuracy: 0.9635
Epoch 15/20
144/144 [==============================] - 191s 1s/step - loss: 0.0898 - accuracy: 0.9694
Epoch 16/20
144/144 [==============================] - 192s 1s/step - loss: 0.0897 - accuracy: 0.9683
Epoch 17/20
144/144 [==============================] - 191s 1s/step - loss: 0.0855 - accuracy: 0.9695
Epoch 18/20
144/144 [==============================] - 191s 1s/step - loss: 0.0830 - accuracy: 0.9716
Epoch 19/20
144/144 [==============================] - 194s 1s/step - loss: 0.0803 - accuracy: 0.9728
Epoch 20/20
144/144 [==============================] - 197s 1s/step - loss: 0.0767 - accuracy: 0.9763
```

In [12]:

```
# Test
print("[bilgi] deeğerlendirme...")
predIdxs = model.predict(testX, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
```

```
[bilgi] deeğerlendirme...
```

In [13]:

```
# Rapor
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))
```

```
              precision    recall  f1-score   support

      Masked       0.97      0.96      0.97       581
     No_Mask       0.96      0.97      0.97       574

    accuracy                           0.97      1155
   macro avg       0.97      0.97      0.97      1155
weighted avg       0.97      0.97      0.97      1155
```

In [14]:

```
# Eğitimiz modeli kaydetme
print("[bilgi] maske_algila modeli kaydediliyor...")
model.save("maske_algila.model", save_format="h5")
```

```
[bilgi] maske_algila modeli kaydediliyor...
```

In [16]:

```
# Eğitim kaybını ve doğruluğunu çizme
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```