```
My dataset is available in googledrive; so I am accessing my d
rom google.colab import drive
rive.mount('/content/drive', force_remount=True)
Once this is executed, you will see your drive appearing on the
```

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

```
# Define paths to the dataset
train_dir = '/content/drive/MyDrive/horse-or-human/test'
test_dir = '/content/drive/MyDrive/horse-or-human/train'
```

```
import os
print(len(os.listdir('/content/drive/MyDrive/horse-or-human/t
print(len(os.listdir('/content/drive/MyDrive/horse-or-human/t
print(len(os.listdir('/content/drive/MyDrive/horse-or-human/t
print(len(os.listdir('/content/drive/MyDrive/horse-or-human/t
```

```
500
527
500
527
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGe

# Create ImageDataGenerator for training set
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2  # Split 20% of the images for vali
)

# Load and prepare training data
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=(256,256),
    batch_size=32,
    class_mode='binary',  # 'binary' for binary classificati
    subset='training'  # Specify 'training' for the training
)

# Create ImageDataGenerator for validation set
validation_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2  # Note: Using the same validation s
)

# Load and prepare validation data
validation_data = validation_datagen.flow_from_directory(
    train_dir,
    target_size=(256,256),
    batch_size=32,
    class_mode='binary',
    subset='validation'  # Specify 'validation' for the vali
)
```

```
Found 822 images belonging to 2 classes.
Found 205 images belonging to 2 classes.
```

```
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,Ba
```

```
# create CNN model - custom-made

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activa
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='va

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activa
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='va

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activ
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='va

model.add(Flatten())

model.add(Dense(128,activation='relu')) #feature reduction
model.add(Dense(64,activation='relu'))
model.add(Dense(1,activation='sigmoid'))  #output layer
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/conv
  super().__init__(activity_regularizer=activity_regularizer,
```

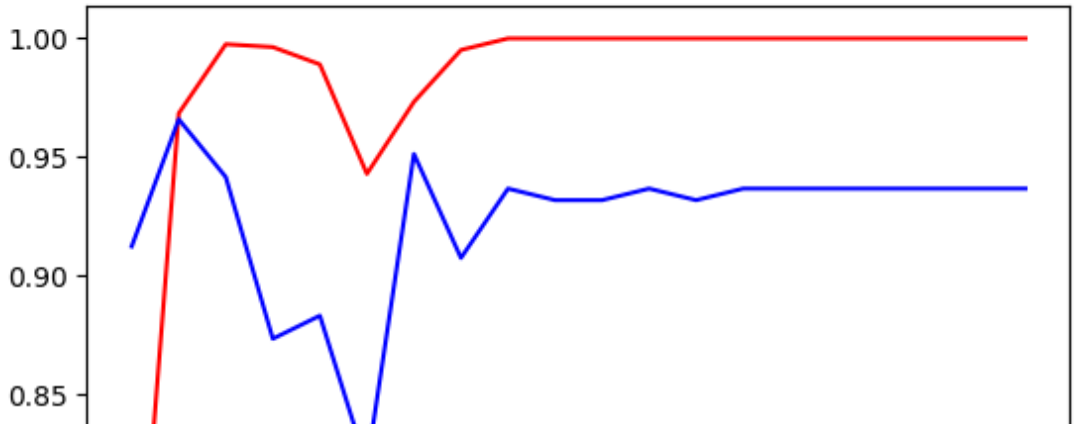
```
from keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.001),loss='bina
```

```
history = model.fit(train_data, epochs=20, validation_data=va
```

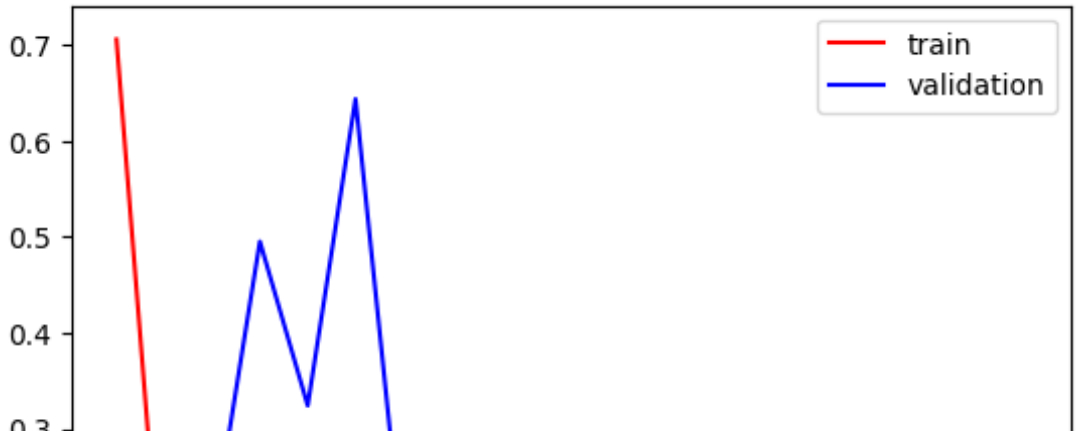
```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/da
  self._warn_if_super_not_called()
Epoch 1/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 193s 7s/step - accuracy: 0.5979 -
Epoch 2/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 138s 5s/step - accuracy: 0.9563 -
Epoch 3/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 140s 5s/step - accuracy: 0.9933 -
Epoch 4/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 139s 5s/step - accuracy: 0.9959 -
Epoch 5/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 149s 6s/step - accuracy: 0.9954 -
Epoch 6/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 139s 5s/step - accuracy: 0.9374 -
Epoch 7/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 138s 5s/step - accuracy: 0.9776 -
Epoch 8/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 148s 6s/step - accuracy: 0.9873 -
Epoch 9/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 149s 6s/step - accuracy: 1.0000 -
Epoch 10/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 149s 6s/step - accuracy: 1.0000 -
Epoch 11/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 209s 6s/step - accuracy: 1.0000 -
Epoch 12/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 139s 5s/step - accuracy: 1.0000 -
Epoch 13/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 137s 5s/step - accuracy: 1.0000 -
Epoch 14/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 137s 5s/step - accuracy: 1.0000 -
Epoch 15/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 137s 5s/step - accuracy: 1.0000 -
Epoch 16/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 139s 5s/step - accuracy: 1.0000 -
Epoch 17/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 150s 6s/step - accuracy: 1.0000 -
Epoch 18/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 150s 6s/step - accuracy: 1.0000 -
Epoch 19/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 190s 5s/step - accuracy: 1.0000 -
Epoch 20/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 152s 6s/step - accuracy: 1.0000 -
```

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'],color='red',label='trai
plt.plot(history.history['val_accuracy'],color='blue',label=
plt.legend()
plt.show()
```



```
plt.plot(history.history['loss'],color='red',label='train')
plt.plot(history.history['val_loss'],color='blue',label='val
plt.legend()
plt.show()
```




```
test_datagen = ImageDataGenerator(rescale=1./255)
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(256,256),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 1027 images belonging to 2 classes.
```

```
#predict the test data
predictions = model.predict(test_data)
```

```
33/33 ━━━━━━━━━━━━━━━━━━━━ 118s 4s/step
```

```
from sklearn.metrics import confusion_matrix, classification_

# Assuming you have ground truth labels (true_labels) and pr
true_labels = test_data.classes
predicted_labels = (predictions > 0.5).astype(int)  # Adjust

# Calculate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Print confusion matrix
print("Confusion Matrix:")
print(cm)

# Print classification report
print("Classification Report:")
print(classification_report(true_labels, predicted_labels))
```

```
Confusion Matrix:
[[247 253]
 [266 261]]
Classification Report:
              precision    recall  f1-score   support

           0       0.48      0.49      0.49       500
           1       0.51      0.50      0.50       527

    accuracy                           0.49      1027
   macro avg       0.49      0.49      0.49      1027
weighted avg       0.49      0.49      0.49      1027
```

```
# create CNN model

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activa
model.add(BatchNormalization())  # added to reduce overfittin
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='va

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activa
model.add(BatchNormalization())  # added to reduce overfittin
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='va

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activa
model.add(BatchNormalization())  # added to reduce overfittin
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='va

model.add(Flatten())

model.add(Dense(128,activation='relu')) #feature reduction
model.add(Dropout(0.1))  # added to reduce overfitting
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))  # added to reduce overfitting
model.add(Dense(1,activation='sigmoid'))  #output layer
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convo
  super().__init__(activity_regularizer=activity_regularizer,
```

```
from keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.001),loss='binar
```

```
history = model.fit(train_data, epochs=20, validation_data=va
```

```
Epoch 1/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 235s 9s/step - accuracy: 0.8428 -
Epoch 2/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 209s 8s/step - accuracy: 0.9778 -
Epoch 3/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 208s 8s/step - accuracy: 0.9895 -
Epoch 4/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 216s 8s/step - accuracy: 0.9832 -
Epoch 5/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 261s 8s/step - accuracy: 0.9872 -
Epoch 6/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 216s 8s/step - accuracy: 0.9957 -
Epoch 7/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 209s 8s/step - accuracy: 0.9931 -
Epoch 8/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 207s 8s/step - accuracy: 0.9929 -
Epoch 9/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 261s 8s/step - accuracy: 0.9981 -
Epoch 10/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 207s 8s/step - accuracy: 0.9972 -
Epoch 11/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 208s 8s/step - accuracy: 0.9887 -
Epoch 12/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 260s 8s/step - accuracy: 0.9919 -
Epoch 13/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 207s 8s/step - accuracy: 0.9850 -
Epoch 14/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 207s 8s/step - accuracy: 0.9980 -
Epoch 15/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 209s 8s/step - accuracy: 0.9970 -
Epoch 16/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 206s 8s/step - accuracy: 0.9986 -
Epoch 17/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 215s 8s/step - accuracy: 0.9976 -
Epoch 18/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 206s 8s/step - accuracy: 0.9953 -
Epoch 19/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 214s 8s/step - accuracy: 0.9968 -
Epoch 20/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 206s 8s/step - accuracy: 0.9998 -
```

```
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(256,256),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 1027 images belonging to 2 classes.
```

```
predictions = model.predict(test_data)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/da
  self._warn_if_super_not_called()
33/33 ━━━━━━━━━━━━━━━━━━━━ 58s 2s/step
```

```
from sklearn.metrics import confusion_matrix, classification_

# Assuming you have ground truth labels (true_labels) and pre
true_labels = test_data.classes
predicted_labels = (predictions > 0.5).astype(int)  # Adjust

# Calculate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Print confusion matrix
print("Confusion Matrix:")
print(cm)

# Print classification report
print("Classification Report after applying techniques to har
print(classification_report(true_labels, predicted_labels))
```

```
Confusion Matrix:
[[263 237]
 [277 250]]
Classification Report after applying techniques to handle ove
              precision    recall  f1-score   support

           0       0.49      0.53      0.51       500
           1       0.51      0.47      0.49       527

    accuracy                           0.50      1027
   macro avg       0.50      0.50      0.50      1027
weighted avg       0.50      0.50      0.50      1027
```

```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten

resnet_model = Sequential()    #ResNet50 is a pre-trained mo
pretrained_model = tf.keras.applications.ResNet50(include_to
                                                  input_shape =
                                                  pooling = 'm
                                                  weights = 'i
for layer in pretrained_model.layers:
    layer.trainable = False  #all layers in the Resnet50 is

resnet_model.add(pretrained_model)
resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation = 'relu'))
resnet_model.add(Dense(1, activation = 'sigmoid'))
```

```
Downloading data from https://storage.googleapis.com/tensorfl
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
from keras.optimizers import Adam
resnet_model.compile(optimizer=Adam(learning_rate=0.001),los
```

```
history = resnet_model.fit(train_data, epochs=20, validation_
```

```
Epoch 1/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 268s 10s/step - accuracy: 0.5358 -
Epoch 2/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 255s 10s/step - accuracy: 0.7011 -
Epoch 3/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 286s 11s/step - accuracy: 0.8359 -
Epoch 4/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 290s 10s/step - accuracy: 0.9046 -
Epoch 5/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 254s 10s/step - accuracy: 0.8924 -
Epoch 6/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 287s 11s/step - accuracy: 0.9405 -
Epoch 7/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 255s 10s/step - accuracy: 0.9153 -
Epoch 8/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 255s 10s/step - accuracy: 0.9704 -
Epoch 9/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 255s 10s/step - accuracy: 0.9342 -
Epoch 10/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 254s 10s/step - accuracy: 0.9681 -
Epoch 11/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 255s 10s/step - accuracy: 0.9707 -
Epoch 12/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 286s 11s/step - accuracy: 0.9557 -
Epoch 13/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 255s 10s/step - accuracy: 0.9535 -
Epoch 14/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 256s 10s/step - accuracy: 0.9588 -
Epoch 15/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 256s 10s/step - accuracy: 0.9565 -
Epoch 16/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 256s 10s/step - accuracy: 0.9788 -
Epoch 17/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 292s 11s/step - accuracy: 0.9793 -
Epoch 18/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 254s 10s/step - accuracy: 0.9511 -
Epoch 19/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 254s 10s/step - accuracy: 0.9780 -
Epoch 20/20
26/26 ━━━━━━━━━━━━━━━━━━━━ 256s 10s/step - accuracy: 0.9700 -
```

```
predictions = resnet_model.predict(test_data)
```

```
33/33 ━━━━━━━━━━━━━━━━━━━━ 265s 8s/step
```

```
from sklearn.metrics import confusion_matrix, classification_

# Assuming you have ground truth labels (true_labels) and pre
true_labels = test_data.classes
predicted_labels = (predictions > 0.5).astype(int)  # Adjust

# Calculate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Print confusion matrix
print("Confusion Matrix:")
print(cm)

# Print classification report
print("Classification Report after applying techniques to har
print(classification_report(true_labels, predicted_labels))
```

```
Confusion Matrix:
[[259 241]
 [312 215]]
Classification Report after applying techniques to handle ove
              precision    recall  f1-score   support

           0       0.45      0.52      0.48       500
           1       0.47      0.41      0.44       527

    accuracy                           0.46      1027
   macro avg       0.46      0.46      0.46      1027
weighted avg       0.46      0.46      0.46      1027
```