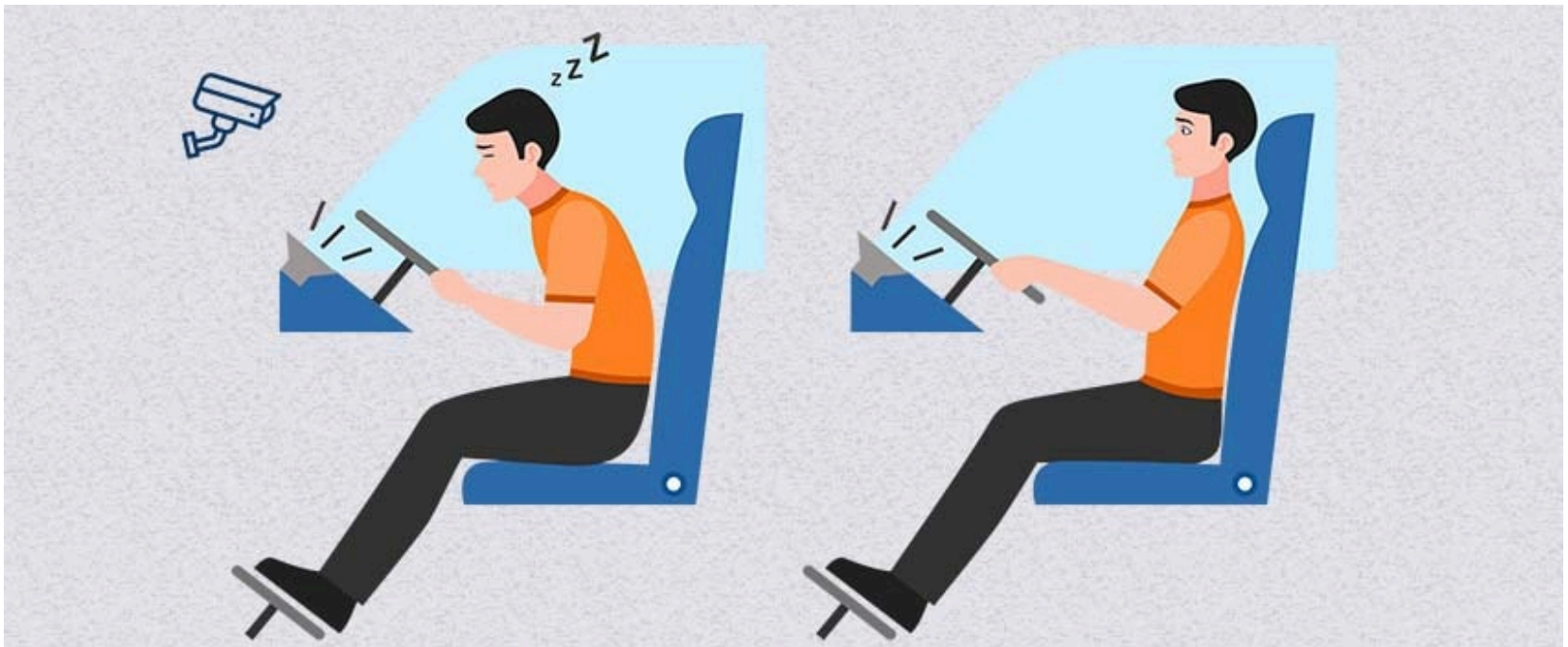
Driver Drowsiness Detection using CNN



According to a report, around 40% of road accidents that happen on highways are caused by Drowsy Driving. This project aims at detecting whether a driver is feeling drowsy or is active while driving based on whether both the eyes of the driver are closed representing drowsiness or both the eyes are open using a CNN architecture.

Importing necessary libraries..

In [1]:
```
import numpy as np
import cv2
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, Dropout, Flatten, Dense, MaxPool2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.optimizers import Adam, SGD
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
from tensorflow.keras.regularizers import l2
import tensorflow as tf
import pandas as pd
from imutils.face_utils import FaceAligner
from imutils.face_utils import rect_to_bb
import imutils
import dlib
```

Loading the dataset and the test images

You can upload your own test images also.

In [6]:
```
!wget https://cainvas-
static.s3.amazonaws.com/media/user_data/Yuvnish17/driver_drowsiness_detection_modified.zip
!unzip -qo driver_drowsiness_detection_modified.zip
```

```
--2021-07-03 15:56:18--  https://cainvas-
static.s3.amazonaws.com/media/user_data/Yuvnish17/driver_drowsiness_detection_modified.zip
Resolving cainvas-static.s3.amazonaws.com (cainvas-static.s3.amazonaws.com)... 52.219.62.88
Connecting to cainvas-static.s3.amazonaws.com (cainvas-static.s3.amazonaws.com)|52.219.62.88|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 127436131 (122M) [application/x-zip-compressed]
Saving to: ‘driver_drowsiness_detection_modified.zip’

driver_drowsiness_d 100%[===================>] 121.53M  98.2MB/s    in 1.2s

2021-07-03 15:56:19 (98.2 MB/s) - ‘driver_drowsiness_detection_modified.zip’ saved [127436131/127436131]
```

Loading the images and labels and Preprocessing the Dataset

Input Shape of Images for the CNN model - (32, 32, 3)
Dataset os split into Train Set and Test Set with test set containing 20% of the total number of images
Labels -> 0 - Closed Eye
1 - Open Eye

In [7]:
```
data = np.load('driver_drowsiness_detection/dataset_compressed.npz', allow_pickle=True)
X = data['arr_0']
Y = data['arr_1']

X = list(X)
Y = list(Y)
print(len(X))
print(len(Y))
```

```
1452
1452
```

In [8]:
```
for i in range(len(X)):
    img = X[i]
    img = cv2.resize(img, (32, 32))
    X[i] = img

print(len(X))
print(X[0].shape)
```

```
1452
(32, 32, 3)
```

In [9]:
```
label_encoder = LabelEncoder()
Y = label_encoder.fit_transform(Y)
print(Y.shape)
print(Y[0])
print(set(Y))
```

```
(1452,)
0
{0, 1}
```

In [10]:
```
X = np.array(X)
Y = np.array(Y)
print(X.shape)
print(Y.shape)
```

```
(1452, 32, 32, 3)
(1452,)
```

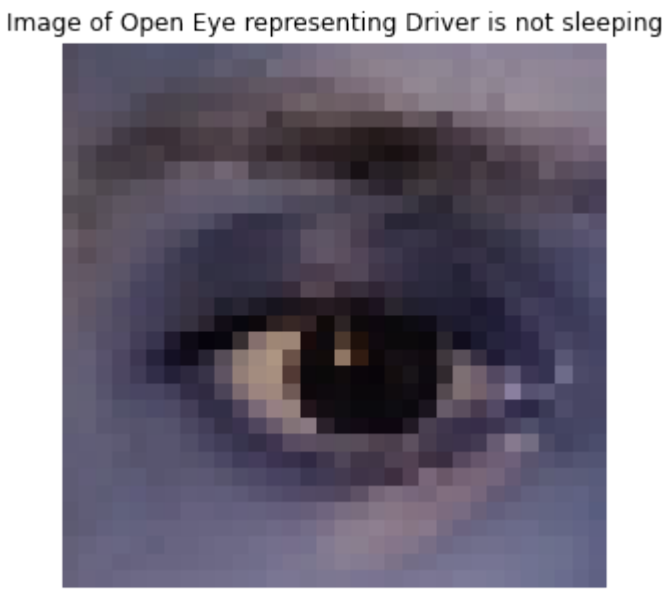
In [11]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [12]:
```
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)

Y_train = to_categorical(Y_train)
Y_test = to_categorical(Y_test)

print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(1161, 32, 32, 3)
(1161,)
(291, 32, 32, 3)
(291,)
(1161, 32, 32, 3)
(1161, 2)
(291, 32, 32, 3)
(291, 2)
```

Visualizing images of Closed Eye and Open Eye from the Dataset

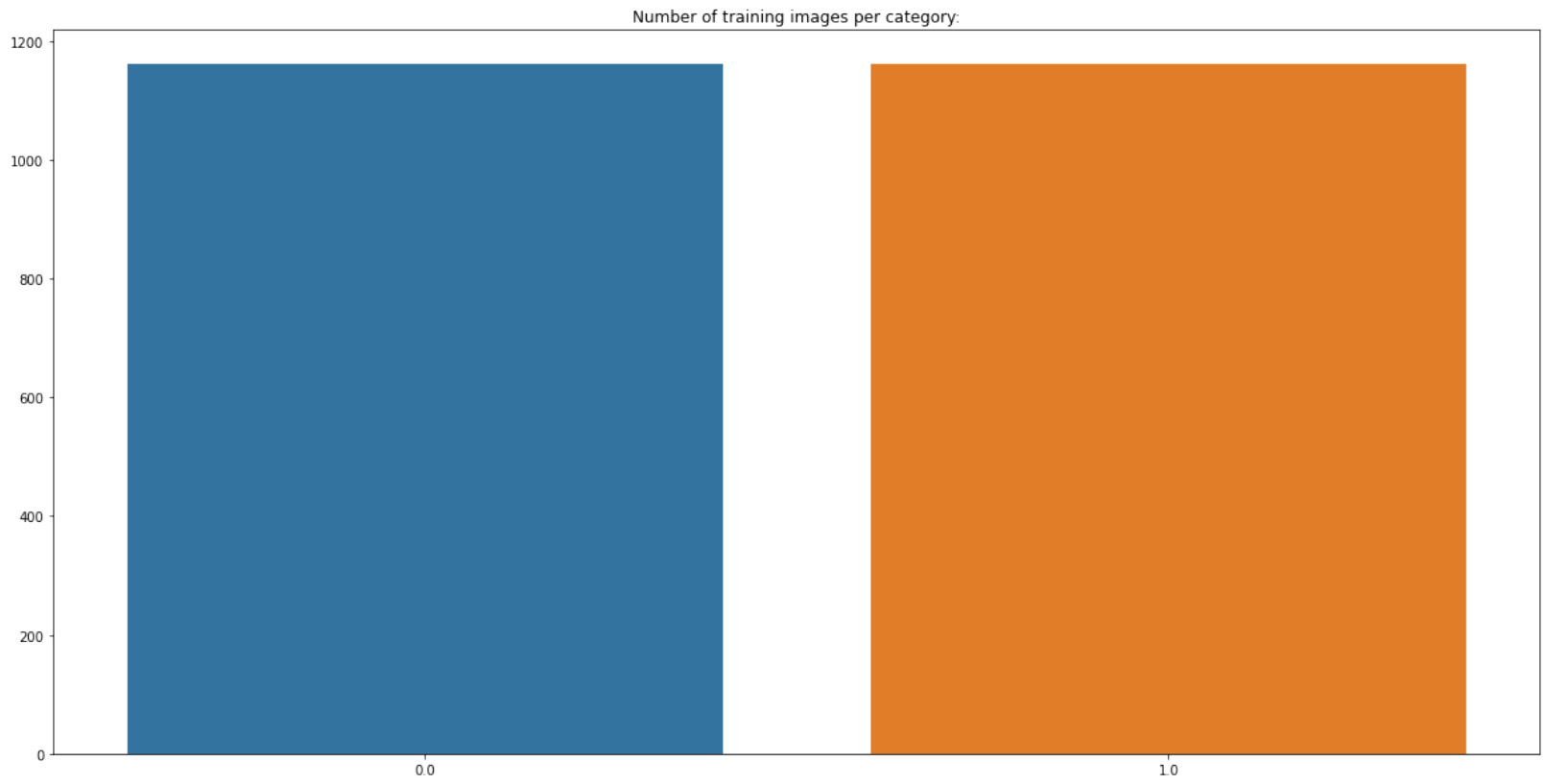
In [13]:
```
figure1 = plt.figure(figsize=(5, 5))
idx_closed = np.where(Y==0)
img_closed = X[idx_closed[0][0]]
plt.imshow(img_closed)
plt.title('Image of Closed Eye representing Driver is sleeping')
plt.axis('off')
plt.show()

figure2 = plt.figure(figsize=(5, 5))
idx_open = np.where(Y==1)
img_open = X[idx_open[0][0]]
plt.imshow(img_open)
plt.title('Image of Open Eye representing Driver is not sleeping')
plt.axis('off')
plt.show()
```



Visualizing the Data Distribution in Train and Test Set

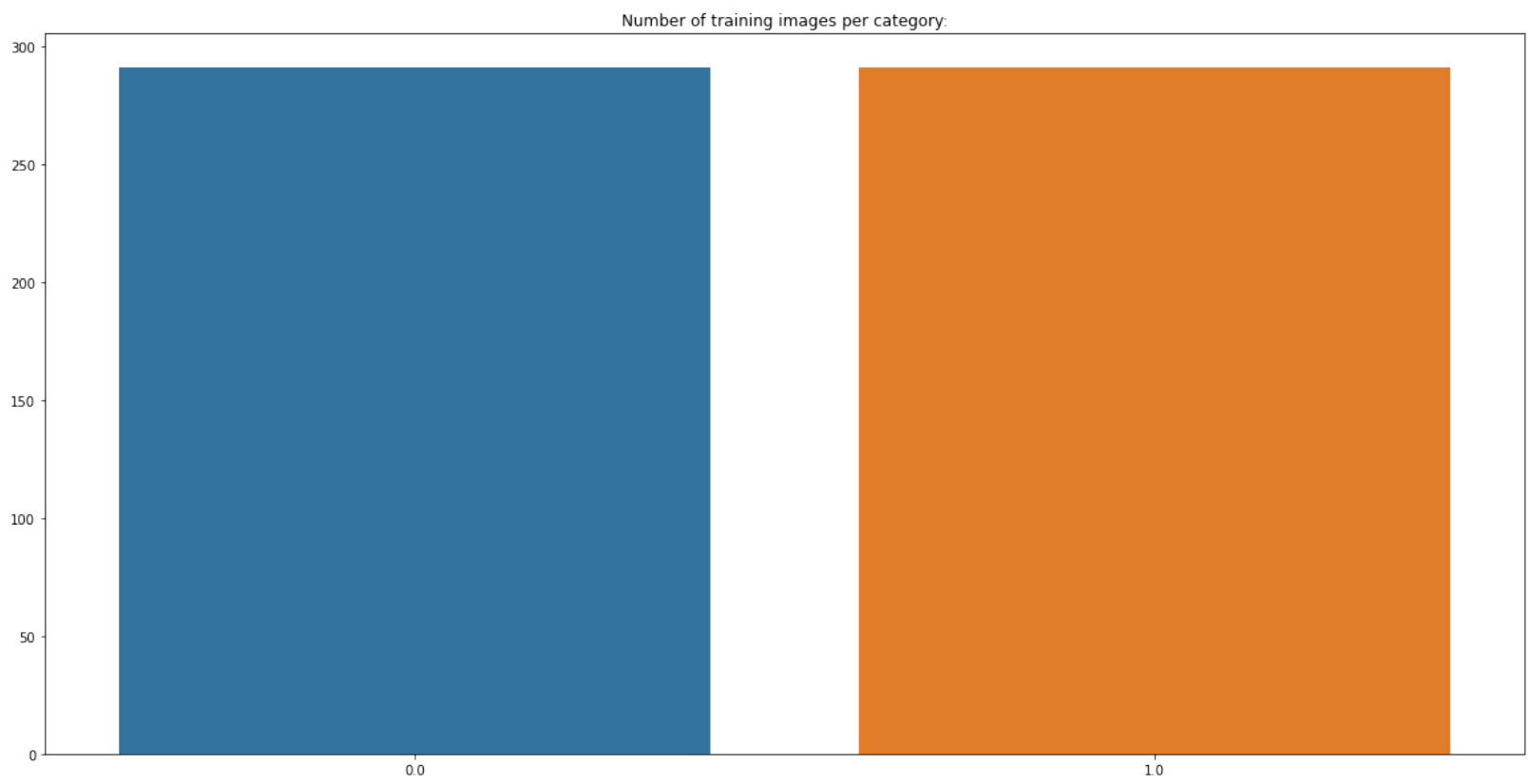
In [11]:
```
unique_train, count = np.unique(Y_train, return_counts=True)
plt.figure(figsize=(20, 10))
sns.barplot(unique_train, count).set_title("Number of training images per category:")
plt.show()
```

```
/opt/tljh/user/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following
variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



In [12]:
```
unique_test, count_test = np.unique(Y_test, return_counts=True)
plt.figure(figsize=(20, 10))
sns.barplot(unique_test, count_test).set_title("Number of training images per category:")
plt.show()
```

```
/opt/tljh/user/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following
variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



As can be seen from the bar plots that the dataset is already balanced so no need to perform dataset balancing

Defining the Model Architecture

In [13]:
```
def driver_drowsiness_detection_model(input_shape=(32, 32, 3)):
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(Conv2D(32, (3, 3), padding='same', strides=(1, 1), name='conv1', activation='relu',
                     kernel_initializer=glorot_uniform(seed=0)))
    model.add(BatchNormalization())
    model.add(Conv2D(32, (3, 3), padding='same', strides=(1, 1), name='conv2', activation='relu',
                     kernel_initializer=glorot_uniform(seed=0)))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))
    model.add(MaxPool2D((2, 2), strides=(2, 2)))

    model.add(Conv2D(64, (3, 3), padding='same', strides=(1, 1), name='conv3', activation='relu',
                     kernel_initializer=glorot_uniform(seed=0)))
    model.add(BatchNormalization())
    model.add(MaxPool2D((2, 2), strides=(2, 2)))
    model.add(Conv2D(64, (3, 3), padding='same', strides=(1, 1), name='conv4', activation='relu',
                     kernel_initializer=glorot_uniform(seed=0)))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))
    model.add(MaxPool2D((2, 2), strides=(2, 2)))

    model.add(Conv2D(64, (3, 3), padding='same', strides=(1, 1), name='conv5', activation='relu',
                     kernel_initializer=glorot_uniform(seed=0)))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), padding='same', strides=(1, 1), name='conv6', activation='relu',
                     kernel_initializer=glorot_uniform(seed=0)))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), padding='same', strides=(1, 1), name='conv7', activation='relu',
                     kernel_initializer=glorot_uniform(seed=0)))
    model.add(BatchNormalization())
    model.add(Dropout(0.4))
    model.add(MaxPool2D((2, 2), strides=(2, 2)))


    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer=glorot_uniform(seed=0), name='fc1'))
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu', kernel_initializer=glorot_uniform(seed=0), name='fc2'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax', kernel_initializer=glorot_uniform(seed=0), name='fc3'))

    optimizer = Adam(0.0001)
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

In [14]:
```
model= driver_drowsiness_detection_model(input_shape=(32, 32, 3))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
conv2 (Conv2D)               (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv3 (Conv2D)               (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv4 (Conv2D)               (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv5 (Conv2D)               (None, 4, 4, 64)          36928
_________________________________________________________________
batch_normalization_4 (Batch (None, 4, 4, 64)          256
_________________________________________________________________
conv6 (Conv2D)               (None, 4, 4, 64)          36928
_________________________________________________________________
batch_normalization_5 (Batch (None, 4, 4, 64)          256
_________________________________________________________________
conv7 (Conv2D)               (None, 4, 4, 64)          36928
_________________________________________________________________
batch_normalization_6 (Batch (None, 4, 4, 64)          256
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
fc1 (Dense)                  (None, 128)               32896
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
fc2 (Dense)                  (None, 128)               16512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
fc3 (Dense)                  (None, 2)                 258
=================================================================
Total params: 227,554
Trainable params: 226,786
Non-trainable params: 768
_________________________________________________________________
```

Training the Model

Learning Rate - 0.0001
Optimizer - Adam
Number of Epochs - 200
Batch Size = 128

In [15]:
```
aug = ImageDataGenerator(rotation_range=20, zoom_range=0.2, horizontal_flip=True)
hist = model.fit(aug.flow(X_train, Y_train, batch_size=128), batch_size=128, epochs=200, validation_data=
(X_test, Y_test))
```

```
Epoch 1/200
10/10 [==============================] - 1s 62ms/step - loss: 1.3128 - accuracy: 0.5357 - val_loss: 0.7441 -
val_accuracy: 0.4192
Epoch 2/200
10/10 [==============================] - 0s 35ms/step - loss: 1.0820 - accuracy: 0.5719 - val_loss: 0.7566 -
val_accuracy: 0.4296
Epoch 3/200
10/10 [==============================] - 0s 35ms/step - loss: 1.0270 - accuracy: 0.5461 - val_loss: 0.7175 -
val_accuracy: 0.4880
Epoch 4/200
10/10 [==============================] - 0s 35ms/step - loss: 0.9197 - accuracy: 0.5642 - val_loss: 0.6779 -
val_accuracy: 0.5464
Epoch 5/200
10/10 [==============================] - 0s 39ms/step - loss: 0.8398 - accuracy: 0.6038 - val_loss: 0.6467 -
val_accuracy: 0.6117
Epoch 6/200
10/10 [==============================] - 0s 39ms/step - loss: 0.8226 - accuracy: 0.5848 - val_loss: 0.6234 -
val_accuracy: 0.6907
Epoch 7/200
10/10 [==============================] - 0s 35ms/step - loss: 0.8189 - accuracy: 0.6012 - val_loss: 0.6081 -
val_accuracy: 0.7320
Epoch 8/200
10/10 [==============================] - 0s 35ms/step - loss: 0.7352 - accuracy: 0.6202 - val_loss: 0.5990 -
val_accuracy: 0.7904
Epoch 9/200
10/10 [==============================] - 0s 36ms/step - loss: 0.6566 - accuracy: 0.6520 - val_loss: 0.5902 -
val_accuracy: 0.7938
Epoch 10/200
10/10 [==============================] - 0s 36ms/step - loss: 0.7122 - accuracy: 0.6417 - val_loss: 0.5802 -
val_accuracy: 0.7835
Epoch 11/200
10/10 [==============================] - 0s 36ms/step - loss: 0.6710 - accuracy: 0.6503 - val_loss: 0.5703 -
val_accuracy: 0.7869
Epoch 12/200
10/10 [==============================] - 0s 36ms/step - loss: 0.7311 - accuracy: 0.6348 - val_loss: 0.5607 -
val_accuracy: 0.8007
Epoch 13/200
10/10 [==============================] - 0s 35ms/step - loss: 0.6478 - accuracy: 0.6555 - val_loss: 0.5492 -
val_accuracy: 0.8007
Epoch 14/200
10/10 [==============================] - 0s 39ms/step - loss: 0.6640 - accuracy: 0.6770 - val_loss: 0.5391 -
val_accuracy: 0.8041
Epoch 15/200
10/10 [==============================] - 0s 37ms/step - loss: 0.6336 - accuracy: 0.6882 - val_loss: 0.5273 -
val_accuracy: 0.8076
Epoch 16/200
10/10 [==============================] - 0s 36ms/step - loss: 0.6174 - accuracy: 0.6848 - val_loss: 0.5130 -
val_accuracy: 0.8144
Epoch 17/200
10/10 [==============================] - 0s 36ms/step - loss: 0.5974 - accuracy: 0.6951 - val_loss: 0.4972 -
val_accuracy: 0.8179
Epoch 18/200
10/10 [==============================] - 0s 36ms/step - loss: 0.6010 - accuracy: 0.7054 - val_loss: 0.4797 -
val_accuracy: 0.8351
Epoch 19/200
10/10 [==============================] - 0s 36ms/step - loss: 0.5604 - accuracy: 0.7140 - val_loss: 0.4619 -
val_accuracy: 0.8591
Epoch 20/200
10/10 [==============================] - 0s 36ms/step - loss: 0.6022 - accuracy: 0.7115 - val_loss: 0.4411 -
val_accuracy: 0.8660
Epoch 21/200
10/10 [==============================] - 0s 36ms/step - loss: 0.5179 - accuracy: 0.7407 - val_loss: 0.4199 -
val_accuracy: 0.8797
Epoch 22/200
10/10 [==============================] - 0s 37ms/step - loss: 0.5241 - accuracy: 0.7571 - val_loss: 0.3991 -
val_accuracy: 0.8832
Epoch 23/200
10/10 [==============================] - 0s 36ms/step - loss: 0.5258 - accuracy: 0.7485 - val_loss: 0.3811 -
val_accuracy: 0.8866
Epoch 24/200
10/10 [==============================] - 0s 36ms/step - loss: 0.5202 - accuracy: 0.7657 - val_loss: 0.3606 -
val_accuracy: 0.8969
Epoch 25/200
10/10 [==============================] - 0s 37ms/step - loss: 0.5184 - accuracy: 0.7502 - val_loss: 0.3444 -
val_accuracy: 0.9003
Epoch 26/200
10/10 [==============================] - 0s 36ms/step - loss: 0.4973 - accuracy: 0.7580 - val_loss: 0.3311 -
val_accuracy: 0.9210
Epoch 27/200
10/10 [==============================] - 0s 36ms/step - loss: 0.5014 - accuracy: 0.7692 - val_loss: 0.3167 -
val_accuracy: 0.9244
Epoch 28/200
10/10 [==============================] - 0s 36ms/step - loss: 0.4590 - accuracy: 0.7778 - val_loss: 0.3032 -
val_accuracy: 0.9278
Epoch 29/200
10/10 [==============================] - 0s 37ms/step - loss: 0.4679 - accuracy: 0.7786 - val_loss: 0.2884 -
val_accuracy: 0.9313
Epoch 30/200
```

```
10/10 [==============================] - 0s 36ms/step - loss: 0.4664 - accuracy: 0.7933 - val_loss: 0.2724 -
val_accuracy: 0.9381
Epoch 31/200
10/10 [==============================] - 0s 36ms/step - loss: 0.4302 - accuracy: 0.7924 - val_loss: 0.2581 -
val_accuracy: 0.9416
Epoch 32/200
10/10 [==============================] - 0s 37ms/step - loss: 0.4441 - accuracy: 0.8174 - val_loss: 0.2453 -
val_accuracy: 0.9381
Epoch 33/200
10/10 [==============================] - 0s 36ms/step - loss: 0.3975 - accuracy: 0.8329 - val_loss: 0.2322 -
val_accuracy: 0.9381
Epoch 34/200
10/10 [==============================] - 0s 37ms/step - loss: 0.4253 - accuracy: 0.8243 - val_loss: 0.2171 -
val_accuracy: 0.9381
Epoch 35/200
10/10 [==============================] - 0s 36ms/step - loss: 0.4118 - accuracy: 0.8208 - val_loss: 0.2050 -
val_accuracy: 0.9381
Epoch 36/200
10/10 [==============================] - 0s 37ms/step - loss: 0.3658 - accuracy: 0.8303 - val_loss: 0.1964 -
val_accuracy: 0.9347
Epoch 37/200
10/10 [==============================] - 0s 37ms/step - loss: 0.3687 - accuracy: 0.8303 - val_loss: 0.1888 -
val_accuracy: 0.9313
Epoch 38/200
10/10 [==============================] - 0s 36ms/step - loss: 0.3588 - accuracy: 0.8519 - val_loss: 0.1797 -
val_accuracy: 0.9347
Epoch 39/200
10/10 [==============================] - 0s 36ms/step - loss: 0.3430 - accuracy: 0.8587 - val_loss: 0.1735 -
val_accuracy: 0.9381
Epoch 40/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2976 - accuracy: 0.8682 - val_loss: 0.1685 -
val_accuracy: 0.9313
Epoch 41/200
10/10 [==============================] - 0s 36ms/step - loss: 0.3316 - accuracy: 0.8579 - val_loss: 0.1632 -
val_accuracy: 0.9347
Epoch 42/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2968 - accuracy: 0.8811 - val_loss: 0.1543 -
val_accuracy: 0.9381
Epoch 43/200
10/10 [==============================] - 0s 36ms/step - loss: 0.3112 - accuracy: 0.8760 - val_loss: 0.1443 -
val_accuracy: 0.9485
Epoch 44/200
10/10 [==============================] - 0s 35ms/step - loss: 0.2778 - accuracy: 0.8880 - val_loss: 0.1405 -
val_accuracy: 0.9485
Epoch 45/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2871 - accuracy: 0.8820 - val_loss: 0.1415 -
val_accuracy: 0.9381
Epoch 46/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2912 - accuracy: 0.8889 - val_loss: 0.1260 -
val_accuracy: 0.9656
Epoch 47/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2999 - accuracy: 0.8725 - val_loss: 0.1177 -
val_accuracy: 0.9656
Epoch 48/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2901 - accuracy: 0.8811 - val_loss: 0.1133 -
val_accuracy: 0.9622
Epoch 49/200
10/10 [==============================] - 0s 37ms/step - loss: 0.2573 - accuracy: 0.9061 - val_loss: 0.1134 -
val_accuracy: 0.9622
Epoch 50/200
10/10 [==============================] - 0s 35ms/step - loss: 0.2705 - accuracy: 0.8949 - val_loss: 0.1095 -
val_accuracy: 0.9622
Epoch 51/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2763 - accuracy: 0.9001 - val_loss: 0.1096 -
val_accuracy: 0.9656
Epoch 52/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2430 - accuracy: 0.9104 - val_loss: 0.1019 -
val_accuracy: 0.9691
Epoch 53/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2337 - accuracy: 0.9078 - val_loss: 0.1000 -
val_accuracy: 0.9656
Epoch 54/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2363 - accuracy: 0.9027 - val_loss: 0.0948 -
val_accuracy: 0.9691
Epoch 55/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2478 - accuracy: 0.9165 - val_loss: 0.0898 -
val_accuracy: 0.9725
Epoch 56/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2253 - accuracy: 0.9070 - val_loss: 0.0872 -
val_accuracy: 0.9725
Epoch 57/200
10/10 [==============================] - 0s 37ms/step - loss: 0.2045 - accuracy: 0.9139 - val_loss: 0.0845 -
val_accuracy: 0.9725
Epoch 58/200
10/10 [==============================] - 0s 37ms/step - loss: 0.2179 - accuracy: 0.9147 - val_loss: 0.0821 -
val_accuracy: 0.9691
Epoch 59/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2063 - accuracy: 0.9268 - val_loss: 0.0776 -
val_accuracy: 0.9759
```

```
Epoch 60/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1910 - accuracy: 0.9251 - val_loss: 0.0774 -
val_accuracy: 0.9759
Epoch 61/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2069 - accuracy: 0.9087 - val_loss: 0.0789 -
val_accuracy: 0.9725
Epoch 62/200
10/10 [==============================] - 0s 36ms/step - loss: 0.2207 - accuracy: 0.9182 - val_loss: 0.0767 -
val_accuracy: 0.9725
Epoch 63/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1807 - accuracy: 0.9242 - val_loss: 0.0803 -
val_accuracy: 0.9725
Epoch 64/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1843 - accuracy: 0.9320 - val_loss: 0.0843 -
val_accuracy: 0.9691
Epoch 65/200
10/10 [==============================] - 0s 39ms/step - loss: 0.1745 - accuracy: 0.9388 - val_loss: 0.0890 -
val_accuracy: 0.9656
Epoch 66/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1799 - accuracy: 0.9320 - val_loss: 0.0809 -
val_accuracy: 0.9656
Epoch 67/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1914 - accuracy: 0.9285 - val_loss: 0.0788 -
val_accuracy: 0.9691
Epoch 68/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1768 - accuracy: 0.9302 - val_loss: 0.0685 -
val_accuracy: 0.9656
Epoch 69/200
10/10 [==============================] - 0s 39ms/step - loss: 0.1577 - accuracy: 0.9414 - val_loss: 0.0663 -
val_accuracy: 0.9759
Epoch 70/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1637 - accuracy: 0.9432 - val_loss: 0.0640 -
val_accuracy: 0.9759
Epoch 71/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1806 - accuracy: 0.9320 - val_loss: 0.0687 -
val_accuracy: 0.9759
Epoch 72/200
10/10 [==============================] - 0s 39ms/step - loss: 0.1754 - accuracy: 0.9294 - val_loss: 0.0897 -
val_accuracy: 0.9691
Epoch 73/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1697 - accuracy: 0.9423 - val_loss: 0.0744 -
val_accuracy: 0.9759
Epoch 74/200
10/10 [==============================] - 0s 39ms/step - loss: 0.1511 - accuracy: 0.9492 - val_loss: 0.0717 -
val_accuracy: 0.9691
Epoch 75/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1606 - accuracy: 0.9388 - val_loss: 0.0730 -
val_accuracy: 0.9691
Epoch 76/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1564 - accuracy: 0.9397 - val_loss: 0.0731 -
val_accuracy: 0.9691
Epoch 77/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1386 - accuracy: 0.9518 - val_loss: 0.0745 -
val_accuracy: 0.9622
Epoch 78/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1787 - accuracy: 0.9414 - val_loss: 0.0758 -
val_accuracy: 0.9725
Epoch 79/200
10/10 [==============================] - 0s 35ms/step - loss: 0.1468 - accuracy: 0.9535 - val_loss: 0.0742 -
val_accuracy: 0.9725
Epoch 80/200
10/10 [==============================] - 0s 39ms/step - loss: 0.1503 - accuracy: 0.9449 - val_loss: 0.0810 -
val_accuracy: 0.9725
Epoch 81/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1587 - accuracy: 0.9492 - val_loss: 0.0822 -
val_accuracy: 0.9725
Epoch 82/200
10/10 [==============================] - 0s 35ms/step - loss: 0.1370 - accuracy: 0.9483 - val_loss: 0.0816 -
val_accuracy: 0.9691
Epoch 83/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1517 - accuracy: 0.9440 - val_loss: 0.0829 -
val_accuracy: 0.9725
Epoch 84/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1514 - accuracy: 0.9440 - val_loss: 0.0955 -
val_accuracy: 0.9588
Epoch 85/200
10/10 [==============================] - 0s 35ms/step - loss: 0.1516 - accuracy: 0.9363 - val_loss: 0.0848 -
val_accuracy: 0.9691
Epoch 86/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1268 - accuracy: 0.9543 - val_loss: 0.0862 -
val_accuracy: 0.9691
Epoch 87/200
10/10 [==============================] - 0s 35ms/step - loss: 0.1279 - accuracy: 0.9518 - val_loss: 0.0752 -
val_accuracy: 0.9656
Epoch 88/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1170 - accuracy: 0.9561 - val_loss: 0.0740 -
val_accuracy: 0.9691
Epoch 89/200
10/10 [==============================] - 0s 40ms/step - loss: 0.1306 - accuracy: 0.9483 - val_loss: 0.0781 -
```

```
val_accuracy: 0.9656
Epoch 90/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1162 - accuracy: 0.9535 - val_loss: 0.0853 -
val_accuracy: 0.9691
Epoch 91/200
10/10 [==============================] - 0s 35ms/step - loss: 0.1307 - accuracy: 0.9535 - val_loss: 0.0746 -
val_accuracy: 0.9691
Epoch 92/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1218 - accuracy: 0.9595 - val_loss: 0.0618 -
val_accuracy: 0.9759
Epoch 93/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1346 - accuracy: 0.9569 - val_loss: 0.0625 -
val_accuracy: 0.9759
Epoch 94/200
10/10 [==============================] - 0s 39ms/step - loss: 0.1269 - accuracy: 0.9569 - val_loss: 0.0698 -
val_accuracy: 0.9656
Epoch 95/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1172 - accuracy: 0.9587 - val_loss: 0.0594 -
val_accuracy: 0.9759
Epoch 96/200
10/10 [==============================] - 0s 39ms/step - loss: 0.1206 - accuracy: 0.9569 - val_loss: 0.0669 -
val_accuracy: 0.9691
Epoch 97/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1250 - accuracy: 0.9483 - val_loss: 0.0695 -
val_accuracy: 0.9691
Epoch 98/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1014 - accuracy: 0.9699 - val_loss: 0.0600 -
val_accuracy: 0.9759
Epoch 99/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1229 - accuracy: 0.9569 - val_loss: 0.0635 -
val_accuracy: 0.9759
Epoch 100/200
10/10 [==============================] - 0s 35ms/step - loss: 0.0951 - accuracy: 0.9681 - val_loss: 0.0698 -
val_accuracy: 0.9725
Epoch 101/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0858 - accuracy: 0.9673 - val_loss: 0.0672 -
val_accuracy: 0.9725
Epoch 102/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0766 - accuracy: 0.9707 - val_loss: 0.0669 -
val_accuracy: 0.9725
Epoch 103/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0794 - accuracy: 0.9673 - val_loss: 0.0670 -
val_accuracy: 0.9691
Epoch 104/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1060 - accuracy: 0.9647 - val_loss: 0.0626 -
val_accuracy: 0.9691
Epoch 105/200
10/10 [==============================] - 0s 40ms/step - loss: 0.1153 - accuracy: 0.9612 - val_loss: 0.0710 -
val_accuracy: 0.9794
Epoch 106/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1148 - accuracy: 0.9595 - val_loss: 0.0701 -
val_accuracy: 0.9794
Epoch 107/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1073 - accuracy: 0.9595 - val_loss: 0.0705 -
val_accuracy: 0.9794
Epoch 108/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0861 - accuracy: 0.9638 - val_loss: 0.0686 -
val_accuracy: 0.9794
Epoch 109/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0800 - accuracy: 0.9716 - val_loss: 0.0691 -
val_accuracy: 0.9828
Epoch 110/200
10/10 [==============================] - 0s 35ms/step - loss: 0.0823 - accuracy: 0.9699 - val_loss: 0.0672 -
val_accuracy: 0.9794
Epoch 111/200
10/10 [==============================] - 0s 39ms/step - loss: 0.0861 - accuracy: 0.9716 - val_loss: 0.0681 -
val_accuracy: 0.9794
Epoch 112/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0984 - accuracy: 0.9664 - val_loss: 0.0672 -
val_accuracy: 0.9759
Epoch 113/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0905 - accuracy: 0.9664 - val_loss: 0.0785 -
val_accuracy: 0.9725
Epoch 114/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1056 - accuracy: 0.9587 - val_loss: 0.0893 -
val_accuracy: 0.9691
Epoch 115/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0725 - accuracy: 0.9742 - val_loss: 0.0798 -
val_accuracy: 0.9725
Epoch 116/200
10/10 [==============================] - 0s 36ms/step - loss: 0.1012 - accuracy: 0.9612 - val_loss: 0.0737 -
val_accuracy: 0.9759
Epoch 117/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0977 - accuracy: 0.9587 - val_loss: 0.0763 -
val_accuracy: 0.9794
Epoch 118/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0768 - accuracy: 0.9724 - val_loss: 0.0882 -
val_accuracy: 0.9725
Epoch 119/200
```

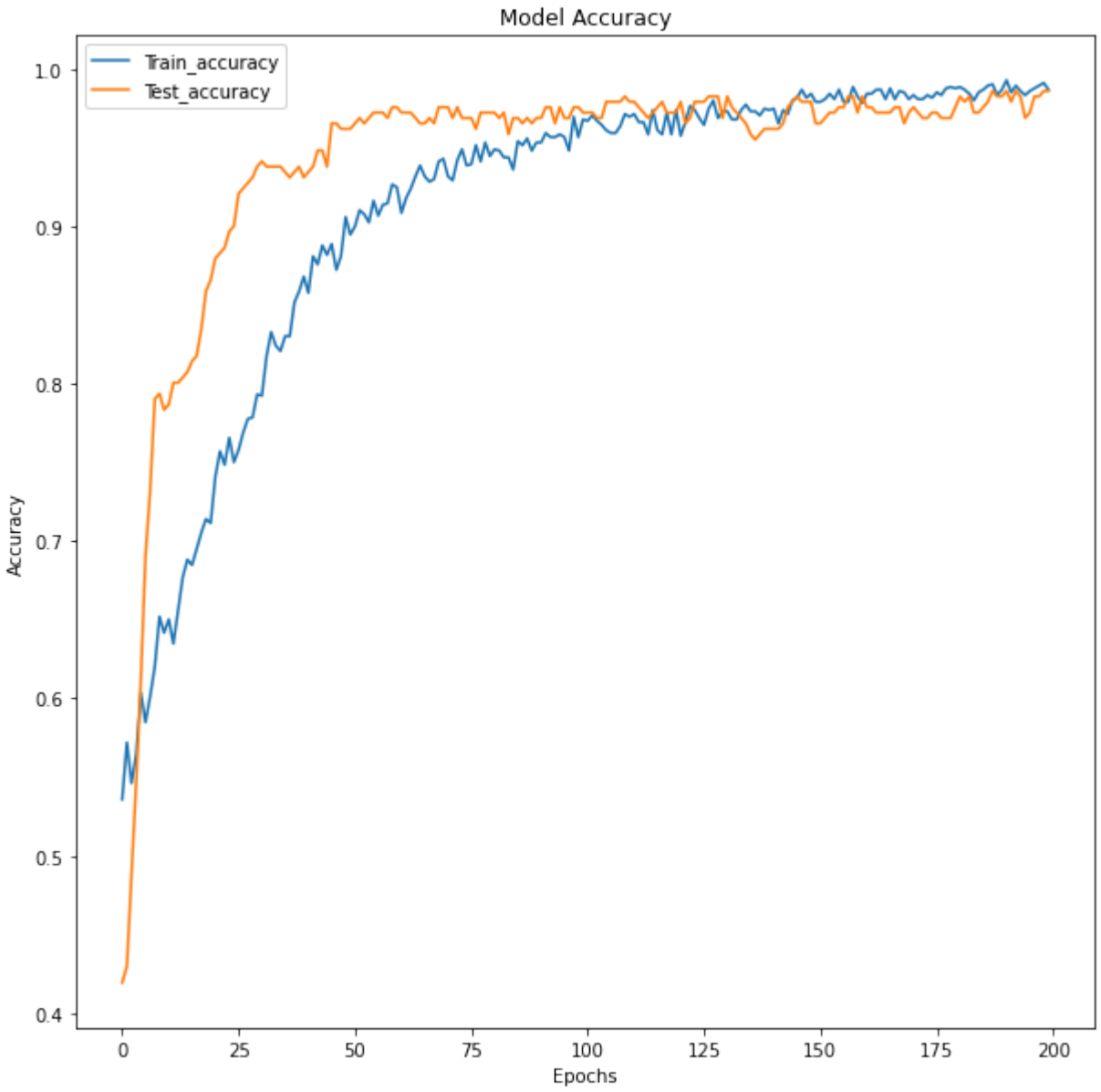
```
10/10 [==============================] - 0s 36ms/step - loss: 0.1243 - accuracy: 0.9587 - val_loss: 0.0810 -
val_accuracy: 0.9725
Epoch 120/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0814 - accuracy: 0.9742 - val_loss: 0.0868 -
val_accuracy: 0.9725
Epoch 121/200
10/10 [==============================] - 0s 37ms/step - loss: 0.1041 - accuracy: 0.9578 - val_loss: 0.0934 -
val_accuracy: 0.9794
Epoch 122/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0879 - accuracy: 0.9664 - val_loss: 0.0962 -
val_accuracy: 0.9656
Epoch 123/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0795 - accuracy: 0.9767 - val_loss: 0.0838 -
val_accuracy: 0.9691
Epoch 124/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0665 - accuracy: 0.9742 - val_loss: 0.0691 -
val_accuracy: 0.9794
Epoch 125/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0682 - accuracy: 0.9690 - val_loss: 0.0729 -
val_accuracy: 0.9794
Epoch 126/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0872 - accuracy: 0.9647 - val_loss: 0.0761 -
val_accuracy: 0.9794
Epoch 127/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0846 - accuracy: 0.9750 - val_loss: 0.0664 -
val_accuracy: 0.9828
Epoch 128/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0676 - accuracy: 0.9802 - val_loss: 0.0704 -
val_accuracy: 0.9828
Epoch 129/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0813 - accuracy: 0.9690 - val_loss: 0.0807 -
val_accuracy: 0.9828
Epoch 130/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0811 - accuracy: 0.9733 - val_loss: 0.0876 -
val_accuracy: 0.9691
Epoch 131/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0766 - accuracy: 0.9733 - val_loss: 0.0604 -
val_accuracy: 0.9828
Epoch 132/200
10/10 [==============================] - 0s 39ms/step - loss: 0.0961 - accuracy: 0.9681 - val_loss: 0.0677 -
val_accuracy: 0.9759
Epoch 133/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0843 - accuracy: 0.9681 - val_loss: 0.0988 -
val_accuracy: 0.9725
Epoch 134/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0771 - accuracy: 0.9742 - val_loss: 0.1048 -
val_accuracy: 0.9691
Epoch 135/200
10/10 [==============================] - 0s 40ms/step - loss: 0.0692 - accuracy: 0.9776 - val_loss: 0.0993 -
val_accuracy: 0.9656
Epoch 136/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0880 - accuracy: 0.9733 - val_loss: 0.1205 -
val_accuracy: 0.9588
Epoch 137/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0690 - accuracy: 0.9733 - val_loss: 0.1100 -
val_accuracy: 0.9553
Epoch 138/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0847 - accuracy: 0.9707 - val_loss: 0.1003 -
val_accuracy: 0.9588
Epoch 139/200
10/10 [==============================] - 0s 35ms/step - loss: 0.0715 - accuracy: 0.9750 - val_loss: 0.0811 -
val_accuracy: 0.9622
Epoch 140/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0700 - accuracy: 0.9742 - val_loss: 0.0743 -
val_accuracy: 0.9622
Epoch 141/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0758 - accuracy: 0.9750 - val_loss: 0.0739 -
val_accuracy: 0.9622
Epoch 142/200
10/10 [==============================] - 0s 40ms/step - loss: 0.0972 - accuracy: 0.9655 - val_loss: 0.0750 -
val_accuracy: 0.9622
Epoch 143/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0758 - accuracy: 0.9742 - val_loss: 0.0708 -
val_accuracy: 0.9656
Epoch 144/200
10/10 [==============================] - 0s 43ms/step - loss: 0.0713 - accuracy: 0.9716 - val_loss: 0.0593 -
val_accuracy: 0.9759
Epoch 145/200
10/10 [==============================] - 0s 45ms/step - loss: 0.0555 - accuracy: 0.9802 - val_loss: 0.0629 -
val_accuracy: 0.9794
Epoch 146/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0542 - accuracy: 0.9819 - val_loss: 0.0627 -
val_accuracy: 0.9828
Epoch 147/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0531 - accuracy: 0.9871 - val_loss: 0.0663 -
val_accuracy: 0.9794
Epoch 148/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0500 - accuracy: 0.9819 - val_loss: 0.0635 -
val_accuracy: 0.9794
```

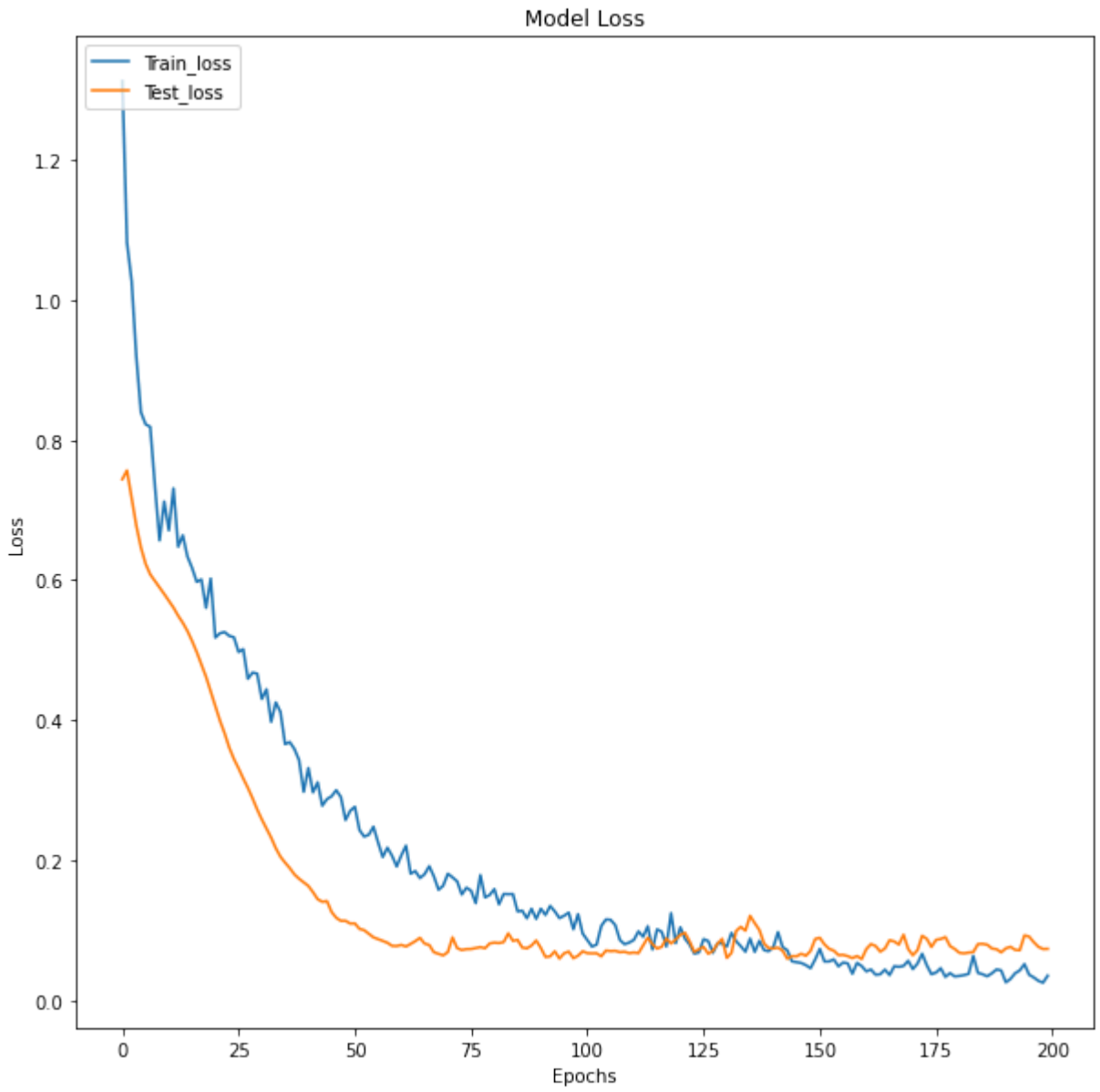
```
Epoch 149/200
10/10 [==============================] - 0s 39ms/step - loss: 0.0457 - accuracy: 0.9845 - val_loss: 0.0704 -
val_accuracy: 0.9794
Epoch 150/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0583 - accuracy: 0.9793 - val_loss: 0.0874 -
val_accuracy: 0.9656
Epoch 151/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0732 - accuracy: 0.9793 - val_loss: 0.0891 -
val_accuracy: 0.9656
Epoch 152/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0553 - accuracy: 0.9811 - val_loss: 0.0797 -
val_accuracy: 0.9691
Epoch 153/200
10/10 [==============================] - 0s 38ms/step - loss: 0.0554 - accuracy: 0.9845 - val_loss: 0.0734 -
val_accuracy: 0.9725
Epoch 154/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0583 - accuracy: 0.9811 - val_loss: 0.0708 -
val_accuracy: 0.9725
Epoch 155/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0483 - accuracy: 0.9871 - val_loss: 0.0644 -
val_accuracy: 0.9759
Epoch 156/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0538 - accuracy: 0.9793 - val_loss: 0.0647 -
val_accuracy: 0.9759
Epoch 157/200
10/10 [==============================] - 0s 39ms/step - loss: 0.0526 - accuracy: 0.9793 - val_loss: 0.0629 -
val_accuracy: 0.9828
Epoch 158/200
10/10 [==============================] - 0s 35ms/step - loss: 0.0378 - accuracy: 0.9888 - val_loss: 0.0601 -
val_accuracy: 0.9828
Epoch 159/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0533 - accuracy: 0.9828 - val_loss: 0.0630 -
val_accuracy: 0.9725
Epoch 160/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0488 - accuracy: 0.9785 - val_loss: 0.0588 -
val_accuracy: 0.9828
Epoch 161/200
10/10 [==============================] - 0s 38ms/step - loss: 0.0412 - accuracy: 0.9845 - val_loss: 0.0731 -
val_accuracy: 0.9759
Epoch 162/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0438 - accuracy: 0.9845 - val_loss: 0.0801 -
val_accuracy: 0.9759
Epoch 163/200
10/10 [==============================] - 0s 39ms/step - loss: 0.0363 - accuracy: 0.9871 - val_loss: 0.0774 -
val_accuracy: 0.9725
Epoch 164/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0371 - accuracy: 0.9871 - val_loss: 0.0693 -
val_accuracy: 0.9725
Epoch 165/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0436 - accuracy: 0.9811 - val_loss: 0.0738 -
val_accuracy: 0.9725
Epoch 166/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0358 - accuracy: 0.9879 - val_loss: 0.0857 -
val_accuracy: 0.9725
Epoch 167/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0484 - accuracy: 0.9811 - val_loss: 0.0839 -
val_accuracy: 0.9759
Epoch 168/200
10/10 [==============================] - 0s 39ms/step - loss: 0.0477 - accuracy: 0.9862 - val_loss: 0.0795 -
val_accuracy: 0.9759
Epoch 169/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0489 - accuracy: 0.9854 - val_loss: 0.0935 -
val_accuracy: 0.9656
Epoch 170/200
10/10 [==============================] - 0s 40ms/step - loss: 0.0563 - accuracy: 0.9811 - val_loss: 0.0731 -
val_accuracy: 0.9725
Epoch 171/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0445 - accuracy: 0.9836 - val_loss: 0.0642 -
val_accuracy: 0.9759
Epoch 172/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0523 - accuracy: 0.9811 - val_loss: 0.0714 -
val_accuracy: 0.9725
Epoch 173/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0664 - accuracy: 0.9811 - val_loss: 0.0920 -
val_accuracy: 0.9691
Epoch 174/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0499 - accuracy: 0.9836 - val_loss: 0.0876 -
val_accuracy: 0.9691
Epoch 175/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0371 - accuracy: 0.9819 - val_loss: 0.0760 -
val_accuracy: 0.9725
Epoch 176/200
10/10 [==============================] - 0s 40ms/step - loss: 0.0395 - accuracy: 0.9854 - val_loss: 0.0863 -
val_accuracy: 0.9725
Epoch 177/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0458 - accuracy: 0.9836 - val_loss: 0.0874 -
val_accuracy: 0.9691
Epoch 178/200
10/10 [==============================] - 0s 35ms/step - loss: 0.0333 - accuracy: 0.9879 - val_loss: 0.0901 -
```

```
val_accuracy: 0.9691
Epoch 179/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0386 - accuracy: 0.9888 - val_loss: 0.0773 -
val_accuracy: 0.9691
Epoch 180/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0340 - accuracy: 0.9879 - val_loss: 0.0732 -
val_accuracy: 0.9759
Epoch 181/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0347 - accuracy: 0.9888 - val_loss: 0.0677 -
val_accuracy: 0.9828
Epoch 182/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0358 - accuracy: 0.9871 - val_loss: 0.0669 -
val_accuracy: 0.9794
Epoch 183/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0376 - accuracy: 0.9845 - val_loss: 0.0680 -
val_accuracy: 0.9828
Epoch 184/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0633 - accuracy: 0.9802 - val_loss: 0.0687 -
val_accuracy: 0.9725
Epoch 185/200
10/10 [==============================] - 0s 40ms/step - loss: 0.0390 - accuracy: 0.9854 - val_loss: 0.0801 -
val_accuracy: 0.9725
Epoch 186/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0371 - accuracy: 0.9871 - val_loss: 0.0804 -
val_accuracy: 0.9759
Epoch 187/200
10/10 [==============================] - 0s 39ms/step - loss: 0.0342 - accuracy: 0.9897 - val_loss: 0.0792 -
val_accuracy: 0.9794
Epoch 188/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0384 - accuracy: 0.9905 - val_loss: 0.0739 -
val_accuracy: 0.9863
Epoch 189/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0440 - accuracy: 0.9836 - val_loss: 0.0728 -
val_accuracy: 0.9828
Epoch 190/200
10/10 [==============================] - 0s 37ms/step - loss: 0.0425 - accuracy: 0.9871 - val_loss: 0.0683 -
val_accuracy: 0.9828
Epoch 191/200
10/10 [==============================] - 0s 40ms/step - loss: 0.0255 - accuracy: 0.9931 - val_loss: 0.0740 -
val_accuracy: 0.9863
Epoch 192/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0305 - accuracy: 0.9854 - val_loss: 0.0763 -
val_accuracy: 0.9794
Epoch 193/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0385 - accuracy: 0.9897 - val_loss: 0.0719 -
val_accuracy: 0.9863
Epoch 194/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0425 - accuracy: 0.9862 - val_loss: 0.0714 -
val_accuracy: 0.9828
Epoch 195/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0517 - accuracy: 0.9836 - val_loss: 0.0925 -
val_accuracy: 0.9691
Epoch 196/200
10/10 [==============================] - 0s 35ms/step - loss: 0.0364 - accuracy: 0.9862 - val_loss: 0.0909 -
val_accuracy: 0.9725
Epoch 197/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0325 - accuracy: 0.9879 - val_loss: 0.0829 -
val_accuracy: 0.9828
Epoch 198/200
10/10 [==============================] - 0s 35ms/step - loss: 0.0277 - accuracy: 0.9897 - val_loss: 0.0762 -
val_accuracy: 0.9828
Epoch 199/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0250 - accuracy: 0.9914 - val_loss: 0.0731 -
val_accuracy: 0.9863
Epoch 200/200
10/10 [==============================] - 0s 36ms/step - loss: 0.0350 - accuracy: 0.9871 - val_loss: 0.0734 -
val_accuracy: 0.9863
```

Plotting Loss Values and Accuracy against Number of Epochs for Train Set and Test Set

In [16]:
```
figure = plt.figure(figsize=(10, 10))
plt.plot(hist.history['accuracy'], label='Train_accuracy')
plt.plot(hist.history['val_accuracy'], label='Test_accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc="upper left")
plt.show()

figure2 = plt.figure(figsize=(10, 10))
plt.plot(hist.history['loss'], label='Train_loss')
plt.plot(hist.history['val_loss'], label='Test_loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc="upper left")
plt.show()
```





Evaluating Model on Test Set

In [17]:
```
pred = model.evaluate(X_test, Y_test)
print(f'Test Set Accuracy: {pred[1]}')
print(f'Test Set Loss: {pred[0]}')
```

```
10/10 [==============================] - 0s 7ms/step - loss: 0.0734 - accuracy: 0.9863
Test Set Accuracy: 0.9862542748451233
Test Set Loss: 0.07337489724159241
```

Classification Report

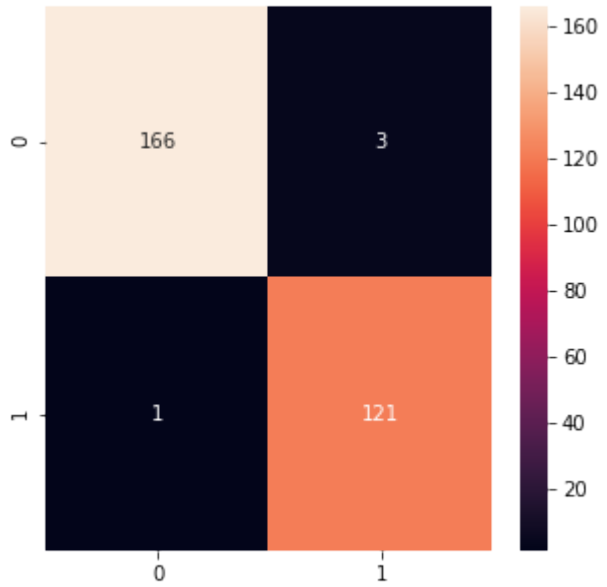
In [18]:
```
ypred = model.predict(X_test)
ypred = np.argmax(ypred, axis=1)
Y_test_pred = np.argmax(Y_test, axis=1)
print(classification_report(Y_test_pred, ypred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       169
           1       0.98      0.99      0.98       122

    accuracy                           0.99       291
   macro avg       0.98      0.99      0.99       291
weighted avg       0.99      0.99      0.99       291
```

Confusion Matrix

In [19]:
```
matrix = confusion_matrix(Y_test_pred, ypred)
df_cm = pd.DataFrame(matrix, index=[0, 1], columns=[0, 1])
figure = plt.figure(figsize=(5, 5))
sns.heatmap(df_cm, annot=True, fmt='d')
```

Out[19]: <AxesSubplot:>



Saving the Model

In [20]:
```
model.save('Driver_Drowsiness_Detection.h5')
```

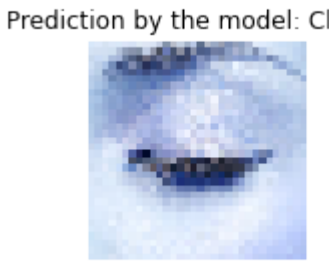
Testing Model on Images

Testing the classification performnace on random images of Open and Closed Eyes

In [21]:
```
labels = ['Closed', 'Open']
img_closed1 = cv2.imread('driver_drowsiness_detection/closed_eye.jpg')
img_closed2 = cv2.imread('driver_drowsiness_detection/closed_eye2.jpg')
img_open1 = cv2.imread('driver_drowsiness_detection/open_eye.jpg')
img_open2 = cv2.imread('driver_drowsiness_detection/open_eye2.jpg')

img_closed1 = cv2.resize(img_closed1, (32, 32))
img_closed2 = cv2.resize(img_closed2, (32, 32))
img_open1 = cv2.resize(img_open1, (32, 32))
img_open2 = cv2.resize(img_open2, (32, 32))

img_closed1 = np.array(img_closed1)
img_closed2 = np.array(img_closed2)
img_open1 = np.array(img_open1)
img_open2 = np.array(img_open2)

img_closed1 = np.expand_dims(img_closed1, axis=0)
img_closed2 = np.expand_dims(img_closed2, axis=0)
img_open1 = np.expand_dims(img_open1, axis=0)
img_open2 = np.expand_dims(img_open2, axis=0)
```

In [22]:
```
ypred_closed1 = model.predict(img_closed1)
ypred_closed2 = model.predict(img_closed2)
ypred_open1 = model.predict(img_open1)
ypred_open2 = model.predict(img_open2)
```

In [23]:
```
figure = plt.figure(figsize=(2, 2))
img_closed1 = np.squeeze(img_closed1, axis=0)
plt.imshow(img_closed1)
plt.axis('off')
plt.title(f'Prediction by the model: {labels[np.argmax(ypred_closed1[0], axis=0)]}')
plt.show()
```



In [24]:
```
figure = plt.figure(figsize=(2, 2))
img_closed2 = np.squeeze(img_closed2, axis=0)
plt.imshow(img_closed2)
plt.axis('off')
plt.title(f'Prediction by the model: {labels[np.argmax(ypred_closed2[0], axis=0)]}')
plt.show()
```

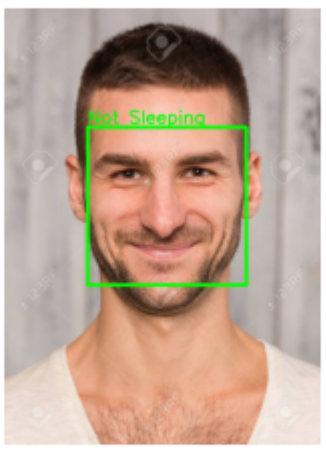
In [25]:
```
figure = plt.figure(figsize=(2, 2))
img_open1 = np.squeeze(img_open1, axis=0)
plt.imshow(img_open1)
plt.axis('off')
plt.title(f'Prediction by the model: {labels[np.argmax(ypred_open1[0], axis=0)]}')
plt.show()
```

In [26]:
```
figure = plt.figure(figsize=(2, 2))
img_open2 = np.squeeze(img_open2, axis=0)
plt.imshow(img_open2)
plt.axis('off')
plt.title(f'Prediction by the model: {labels[np.argmax(ypred_open2[0], axis=0)]}')
plt.show()
```

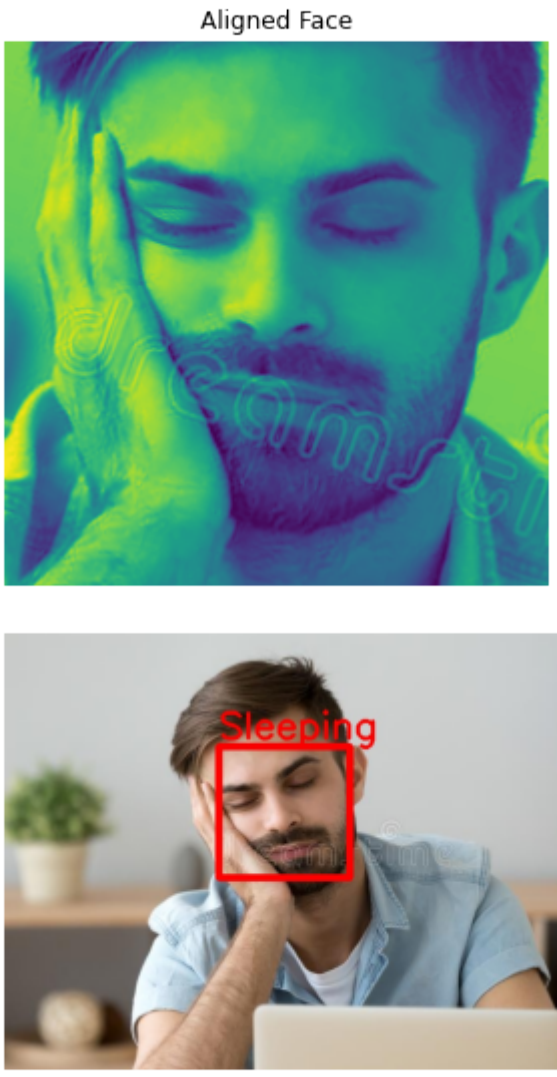
Creating Pipeline for making predictions on full face images

In [27]:
```
def full_face_detection_pipeline(input_image_path):
    face_cascade = cv2.CascadeClassifier('driver_drowsiness_detection/haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier('driver_drowsiness_detection/haarcascade_eye.xml')
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor('driver_drowsiness_detection/shape_predictor_68_face_landmarks.dat')
    fa = FaceAligner(predictor, desiredFaceWidth=256)
    test_image = cv2.imread(input_image_path)
    test_image = imutils.resize(test_image, width=800)
    test_image_gray = cv2.cvtColor(test_image, cv2.COLOR_BGR2GRAY)
    rects = detector(test_image_gray, 2)
    for rect in rects:
        (x, y, w, h) = rect_to_bb(rect)
        faceOrig = imutils.resize(test_image[y:y+h, x:x+w], width=256)
        faceAligned = fa.align(test_image, test_image_gray, rect)
        faceAligned_gray = cv2.cvtColor(faceAligned, cv2.COLOR_BGR2GRAY)
        plt.imshow(faceAligned_gray)
        plt.axis('off')
        plt.title('Aligned Face')
        plt.show()
        eyes = eye_cascade.detectMultiScale(faceAligned_gray, 1.1, 4)
        predictions = []
        for (ex, ey, ew, eh) in eyes:
            eye = faceAligned[ey:ey+eh, ex:ex+ew]
#             cv2.rectangle(test_image, (x+ex, y+ey), (x+ex+ew, y+ey+eh), (0, 0, 255), 8)
            eye = cv2.resize(eye, (32, 32))
            eye = np.array(eye)
            eye = np.expand_dims(eye, axis=0)
            ypred = model.predict(eye)
            ypred = np.argmax(ypred[0], axis=0)
            predictions.append(ypred)
        if all(i==0 for i in predictions):
            cv2.rectangle(test_image, (x, y), (x+w, y+h), (0, 0, 255), 8)
            cv2.putText(test_image, 'Sleeping', (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 3)
        else:
            cv2.rectangle(test_image, (x, y), (x+w, y+h), (0, 255, 0), 8)
            cv2.putText(test_image, 'Not Sleeping', (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, 0), 3)
    output_path = 'driver_drowsiness_detection/test_image_prediction.jpg'
    cv2.imwrite(output_path, test_image)
    return output_path
```

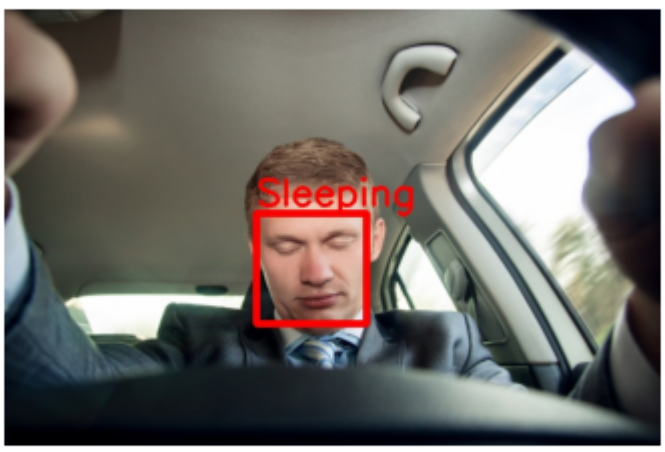
In [28]:
```
figure = plt.figure(figsize=(5, 5))
predicted_image = cv2.imread(full_face_detection_pipeline('driver_drowsiness_detection/active_person.jpg'))
predicted_image = cv2.cvtColor(predicted_image, cv2.COLOR_BGR2RGB)
plt.imshow(predicted_image)
plt.axis('off')
plt.show()
```



In [29]:
```
figure = plt.figure(figsize=(5, 5))
predicted_image = cv2.imread(full_face_detection_pipeline('driver_drowsiness_detection/drowsy_person.jpg'))
predicted_image = cv2.cvtColor(predicted_image, cv2.COLOR_BGR2RGB)
plt.imshow(predicted_image)
plt.axis('off')
plt.show()
```



In [30]:
```
figure = plt.figure(figsize=(5, 5))
predicted_image = cv2.imread(full_face_detection_pipeline('driver_drowsiness_detection/sleepy-driver.jpg'))
predicted_image = cv2.cvtColor(predicted_image, cv2.COLOR_BGR2RGB)
plt.imshow(predicted_image)
plt.axis('off')
plt.show()
```



In [ ]: