HW3

November 19, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
import cv2
import statistics as sts
import warnings
warnings.filterwarnings('ignore')

from time import time
```

[2]:
```
train_df = pd.read_csv('/content/drive/MyDrive/Homework3/Q1_Train_Data.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Homework3/Q1_Test_Data.csv')
validation_df = pd.read_csv('/content/drive/MyDrive/Homework3/Q1_Validation_Data.
 ↪csv')
```

[3]:
```
emotions = {'angry':0, 'disgust':1, 'fear':2, 'happy':3,
            'sad':4, 'surprise':5, 'neutral':6}
```

[4]:
```
train_angry_df = train_df.loc[train_df['emotion'] == 0]
train_angry_df = train_angry_df.reset_index(drop = True)

train_disgust_df = train_df.loc[train_df['emotion'] == 1]
train_disgust_df = train_disgust_df.reset_index(drop = True)

train_fear_df = train_df.loc[train_df['emotion'] == 2]
train_fear_df = train_fear_df.reset_index(drop = True)

train_happy_df = train_df.loc[train_df['emotion'] == 3]
train_happy_df = train_happy_df.reset_index(drop = True)

train_sad_df = train_df.loc[train_df['emotion'] == 4]
train_sad_df = train_sad_df.reset_index(drop = True)

train_surprise_df = train_df.loc[train_df['emotion'] == 5]
train_surprise_df = train_surprise_df.reset_index(drop = True)
```

```
train_neutral_df = train_df.loc[train_df['emotion'] == 6]
train_neutral_df = train_neutral_df.reset_index(drop = True)
```

[5]:
```
images = [train_angry_df['pixels'][0], train_angry_df['pixels'][1],
          train_disgust_df['pixels'][0], train_disgust_df['pixels'][1],
          train_fear_df['pixels'][0], train_fear_df['pixels'][1],
          train_happy_df['pixels'][0], train_happy_df['pixels'][1],
          train_sad_df['pixels'][0], train_sad_df['pixels'][1],
          train_surprise_df['pixels'][0], train_surprise_df['pixels'][1],
          train_neutral_df['pixels'][0], train_neutral_df['pixels'][1]]
```

[6]:
```
def process_image(image_str):
  img_flat = image_str.split()
  img_flat = list(map(lambda x: float(x), img_flat))

  mean = sts.mean(img_flat)
  stdev = sts.stdev(img_flat)

  img_arr = np.asarray(img_flat)
  img_arr = (img_arr - mean)/ max(stdev, 1/np.sqrt(48*48))
  img = img_arr.reshape(48, 48)

  return img
```

[7]:
```
# reading images
images = [process_image(image) for image in images]
```

[8]:
```
fig = plt.figure(figsize=(6, 24))

# setting values to rows and column variables
rows = 7
columns = 2

for i, image in enumerate(images):
  fig.add_subplot(rows, columns, i+1)
  plt.imshow(image, cmap = 'gray', aspect = 'auto')
  plt.axis('off')
  if i%2 == 0:
    emotion = list(emotions.keys())[i//2]
  plt.title('{}'.format(emotion))
```

angry
angry
disgust
disgust
fear
fear
happy
happy
sad
sad
surprise
surprise
neutral
neutral

a. Two images per class is displayed above.

```
[9]: print("Number of samples for angry emotion = {}". format(len(train_angry_df)))
print("Number of samples for disgust emotion = {}".␣
 ↪format(len(train_disgust_df)))
print("Number of samples for fear emotion = {}". format(len(train_fear_df)))
print("Number of samples for happy emotion = {}". format(len(train_happy_df)))
print("Number of samples for sad emotion = {}". format(len(train_sad_df)))
print("Number of samples for surprise emotion = {}".␣
 ↪format(len(train_surprise_df)))
print("Number of samples for neutral emotion = {}".␣
 ↪format(len(train_neutral_df)))
```

```
Number of samples for angry emotion = 3995
Number of samples for disgust emotion = 436
Number of samples for fear emotion = 4097
Number of samples for happy emotion = 7215
Number of samples for sad emotion = 4830
Number of samples for surprise emotion = 3171
Number of samples for neutral emotion = 4965
```

b. number of samples per emotion is listed below

- Angry: 3995
- Disgust: 436
- Fear: 4097
- Happy: 7215
- Sad: 4830
- Surprise: 3171
- Neutral: 4965

```
[10]: X_train = np.asarray([process_image(X) for X in list(train_df['pixels'])])
X_train.shape
```

```
[10]: (28709, 48, 48)
```

```
[11]: y_train = [tf.keras.utils.to_categorical(y, num_classes = 7, dtype = "float32")␣
 ↪for y in list(train_df['emotion'])]
y_train = np.asarray(y_train)
y_train.shape
```

```
[11]: (28709, 7)
```

```
[12]: X_val =  np.asarray([process_image(X) for X in list(validation_df['pixels'])])
X_val.shape
```

```
[12]: (3589, 48, 48)
```

[13]:
```
y_val = [tf.keras.utils.to_categorical(y, num_classes = 7, dtype = "float32")␣
 ↪for y in list(validation_df['emotion'])]
y_val = np.asarray(y_val)
y_val.shape
```

[13]:
```
(3589, 7)
```

[14]:
```
X_test =  np.asarray([process_image(X) for X in list(test_df['pixels'])])
X_test.shape
```

[14]:
```
(3589, 48, 48)
```

[15]:
```
y_test = [tf.keras.utils.to_categorical(y, num_classes = 7, dtype = "float32")␣
 ↪for y in list(test_df['emotion'])]
y_test = np.asarray(y_test)
y_test.shape
```

[15]:
```
(3589, 7)
```

[16]:
```
X_train_fnn = X_train.reshape(-1, 2304)
X_val_fnn = X_val.reshape(-1, 2304)
X_test_fnn = X_test.reshape(-1, 2304)
```

[17]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.callbacks import ModelCheckpoint
from keras import regularizers
```

[18]:
```
# Neural network
fnn_model1 = Sequential()
fnn_model1.add(Dense(128, input_dim = 2304, kernel_regularizer = regularizers.
 ↪l2(l2=0.01), activation = 'relu'))
fnn_model1.add(Dense(64, kernel_regularizer = regularizers.l2(l2=0.01),␣
 ↪activation= 'relu'))
fnn_model1.add(Dense(32, kernel_regularizer = regularizers.l2(l2=0.01),␣
 ↪activation = 'relu'))
fnn_model1.add(Dense(7, activation = 'softmax'))
```

[ ]:
```
fnn_checkpoint1 = ModelCheckpoint("best_fnn_model1.hdf5", monitor = 'val_loss',␣
 ↪verbose = 1,
                                  save_best_only = True, mode =' min', period =␣
 ↪1)
```

[20]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
```

```
fnn_model1.compile(loss='categorical_crossentropy', optimizer = optimizer,␣
 ↪metrics = ['accuracy'])
```

[21]:
```
start = time()

history_fnn1 = fnn_model1.fit(X_train_fnn, y_train,
                    validation_data = (X_val_fnn, y_val),
                    epochs = 20, batch_size = 512,
                    validation_batch_size = 512,
                    callbacks=[fnn_checkpoint1])

time_to_run = time() - start
```

```
Epoch 1/20
56/57 [============================>.] - ETA: 0s - loss: 4.1236 - accuracy:
0.3170
Epoch 00001: val_loss improved from inf to 2.95675, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 2s 8ms/step - loss: 4.1221 - accuracy:
0.3170 - val_loss: 2.9567 - val_accuracy: 0.3667
Epoch 2/20
54/57 [===========================>..] - ETA: 0s - loss: 2.5211 - accuracy:
0.3809
Epoch 00002: val_loss improved from 2.95675 to 2.23093, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 2.5107 - accuracy:
0.3796 - val_loss: 2.2309 - val_accuracy: 0.3656
Epoch 3/20
51/57 [=========================>...] - ETA: 0s - loss: 2.0808 - accuracy:
0.3901
Epoch 00003: val_loss improved from 2.23093 to 1.98270, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 2.0691 - accuracy:
0.3911 - val_loss: 1.9827 - val_accuracy: 0.3806
Epoch 4/20
52/57 [==========================>...] - ETA: 0s - loss: 1.8992 - accuracy:
0.3990
Epoch 00004: val_loss improved from 1.98270 to 1.86366, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.8957 - accuracy:
0.3994 - val_loss: 1.8637 - val_accuracy: 0.3823
Epoch 5/20
52/57 [==========================>...] - ETA: 0s - loss: 1.8076 - accuracy:
0.4049
Epoch 00005: val_loss improved from 1.86366 to 1.81721, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.8066 - accuracy:
0.4042 - val_loss: 1.8172 - val_accuracy: 0.3734
```

```
Epoch 6/20
52/57 [==========================>...] - ETA: 0s - loss: 1.7684 - accuracy:
0.4051
Epoch 00006: val_loss improved from 1.81721 to 1.77925, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.7666 - accuracy:
0.4046 - val_loss: 1.7793 - val_accuracy: 0.3817
Epoch 7/20
53/57 [==========================>...] - ETA: 0s - loss: 1.7301 - accuracy:
0.4117
Epoch 00007: val_loss improved from 1.77925 to 1.74664, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.7299 - accuracy:
0.4116 - val_loss: 1.7466 - val_accuracy: 0.3867
Epoch 8/20
55/57 [===========================>..] - ETA: 0s - loss: 1.7086 - accuracy:
0.4118
Epoch 00008: val_loss improved from 1.74664 to 1.74087, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.7085 - accuracy:
0.4122 - val_loss: 1.7409 - val_accuracy: 0.3884
Epoch 9/20
51/57 [=========================>...] - ETA: 0s - loss: 1.6937 - accuracy:
0.4155
Epoch 00009: val_loss did not improve from 1.74087
57/57 [==============================] - 0s 5ms/step - loss: 1.6933 - accuracy:
0.4146 - val_loss: 1.7767 - val_accuracy: 0.3689
Epoch 10/20
52/57 [==========================>...] - ETA: 0s - loss: 1.6916 - accuracy:
0.4090
Epoch 00010: val_loss improved from 1.74087 to 1.70722, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6897 - accuracy:
0.4089 - val_loss: 1.7072 - val_accuracy: 0.3865
Epoch 11/20
51/57 [=========================>...] - ETA: 0s - loss: 1.6612 - accuracy:
0.4220
Epoch 00011: val_loss improved from 1.70722 to 1.69714, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6608 - accuracy:
0.4219 - val_loss: 1.6971 - val_accuracy: 0.3962
Epoch 12/20
52/57 [==========================>...] - ETA: 0s - loss: 1.6609 - accuracy:
0.4176
Epoch 00012: val_loss did not improve from 1.69714
57/57 [==============================] - 0s 5ms/step - loss: 1.6617 - accuracy:
0.4167 - val_loss: 1.7053 - val_accuracy: 0.3862
Epoch 13/20
```

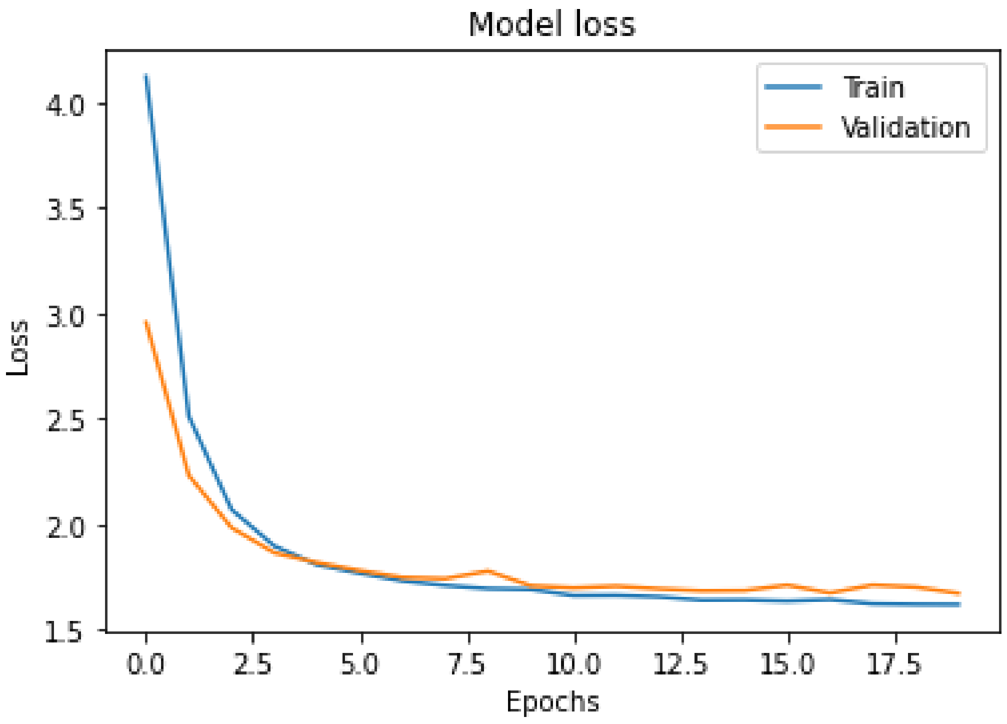
```
55/57 [===========================>..] - ETA: 0s - loss: 1.6528 - accuracy:
0.4180
Epoch 00013: val_loss improved from 1.69714 to 1.69240, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6544 - accuracy:
0.4172 - val_loss: 1.6924 - val_accuracy: 0.3979
Epoch 14/20
54/57 [===========================>..] - ETA: 0s - loss: 1.6391 - accuracy:
0.4269
Epoch 00014: val_loss improved from 1.69240 to 1.68309, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6390 - accuracy:
0.4275 - val_loss: 1.6831 - val_accuracy: 0.3951
Epoch 15/20
54/57 [===========================>..] - ETA: 0s - loss: 1.6412 - accuracy:
0.4211
Epoch 00015: val_loss did not improve from 1.68309
57/57 [==============================] - 0s 5ms/step - loss: 1.6399 - accuracy:
0.4221 - val_loss: 1.6844 - val_accuracy: 0.4060
Epoch 16/20
54/57 [===========================>..] - ETA: 0s - loss: 1.6352 - accuracy:
0.4274
Epoch 00016: val_loss did not improve from 1.68309
57/57 [==============================] - 0s 5ms/step - loss: 1.6344 - accuracy:
0.4275 - val_loss: 1.7100 - val_accuracy: 0.3962
Epoch 17/20
53/57 [==========================>...] - ETA: 0s - loss: 1.6431 - accuracy:
0.4217
Epoch 00017: val_loss improved from 1.68309 to 1.67300, saving model to
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6415 - accuracy:
0.4220 - val_loss: 1.6730 - val_accuracy: 0.3959
Epoch 18/20
53/57 [==========================>...] - ETA: 0s - loss: 1.6216 - accuracy:
0.4290
Epoch 00018: val_loss did not improve from 1.67300
57/57 [==============================] - 0s 5ms/step - loss: 1.6218 - accuracy:
0.4290 - val_loss: 1.7105 - val_accuracy: 0.3887
Epoch 19/20
54/57 [===========================>..] - ETA: 0s - loss: 1.6201 - accuracy:
0.4284
Epoch 00019: val_loss did not improve from 1.67300
57/57 [==============================] - 0s 5ms/step - loss: 1.6189 - accuracy:
0.4288 - val_loss: 1.7008 - val_accuracy: 0.3923
Epoch 20/20
51/57 [=========================>...] - ETA: 0s - loss: 1.6189 - accuracy:
0.4333
Epoch 00020: val_loss improved from 1.67300 to 1.67140, saving model to
```

```
best_fnn_model1.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6178 - accuracy:
0.4335 - val_loss: 1.6714 - val_accuracy: 0.3940
```

[22]:
```
fnn_model1.summary()

print(time_to_run)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               295040

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 7)                 231

=================================================================
Total params: 305,607
Trainable params: 305,607
Non-trainable params: 0
_________________________________________________________________
8.261720895767212
```

[23]:
```
plt.plot(history_fnn1.history['loss'])
plt.plot(history_fnn1.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



[24]:
```
# Neural network
fnn_model2 = Sequential()
fnn_model2.add(Dense(256, input_dim = 2304, activation = 'relu'))
fnn_model2.add(Dropout(0.5))
fnn_model2.add(Dense(64, activation= 'relu'))
fnn_model2.add(Dropout(0.2))
fnn_model2.add(Dense(7, activation = 'softmax'))
```

[25]:
```
fnn_checkpoint2 = ModelCheckpoint("best_fnn_model2.hdf5", monitor = 'val_loss',␣
 ↪verbose = 1,
                                  save_best_only = True, mode =' min', period =␣
 ↪1)
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to
specify the frequency in number of batches seen.
WARNING:tensorflow:ModelCheckpoint mode  min is unknown, fallback to auto mode.
```

[26]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.002)

fnn_model2.compile(loss='categorical_crossentropy', optimizer = optimizer,␣
 ↪metrics = ['accuracy'])
```

[27]:
```
start = time()
```

```
history_fnn2 = fnn_model2.fit(X_train_fnn, y_train,
                    validation_data = (X_val_fnn, y_val),
                    epochs = 20, batch_size = 512,
                    validation_batch_size = 512,
                    callbacks=[fnn_checkpoint2])

time_to_run = time() - start
```

```
Epoch 1/20
52/57 [==========================>...] - ETA: 0s - loss: 2.0057 - accuracy:
0.2462
Epoch 00001: val_loss improved from inf to 1.72485, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 1s 8ms/step - loss: 1.9897 - accuracy:
0.2488 - val_loss: 1.7249 - val_accuracy: 0.3232
Epoch 2/20
56/57 [============================>.] - ETA: 0s - loss: 1.7348 - accuracy:
0.3036
Epoch 00002: val_loss improved from 1.72485 to 1.66313, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.7345 - accuracy:
0.3037 - val_loss: 1.6631 - val_accuracy: 0.3394
Epoch 3/20
55/57 [===========================>..] - ETA: 0s - loss: 1.6920 - accuracy:
0.3300
Epoch 00003: val_loss improved from 1.66313 to 1.65258, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6927 - accuracy:
0.3294 - val_loss: 1.6526 - val_accuracy: 0.3605
Epoch 4/20
55/57 [===========================>..] - ETA: 0s - loss: 1.6689 - accuracy:
0.3397
Epoch 00004: val_loss improved from 1.65258 to 1.62031, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6676 - accuracy:
0.3407 - val_loss: 1.6203 - val_accuracy: 0.3670
Epoch 5/20
55/57 [===========================>..] - ETA: 0s - loss: 1.6406 - accuracy:
0.3520
Epoch 00005: val_loss improved from 1.62031 to 1.59883, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6400 - accuracy:
0.3523 - val_loss: 1.5988 - val_accuracy: 0.3748
Epoch 6/20
52/57 [==========================>...] - ETA: 0s - loss: 1.6127 - accuracy:
0.3602
Epoch 00006: val_loss improved from 1.59883 to 1.59079, saving model to
best_fnn_model2.hdf5
```

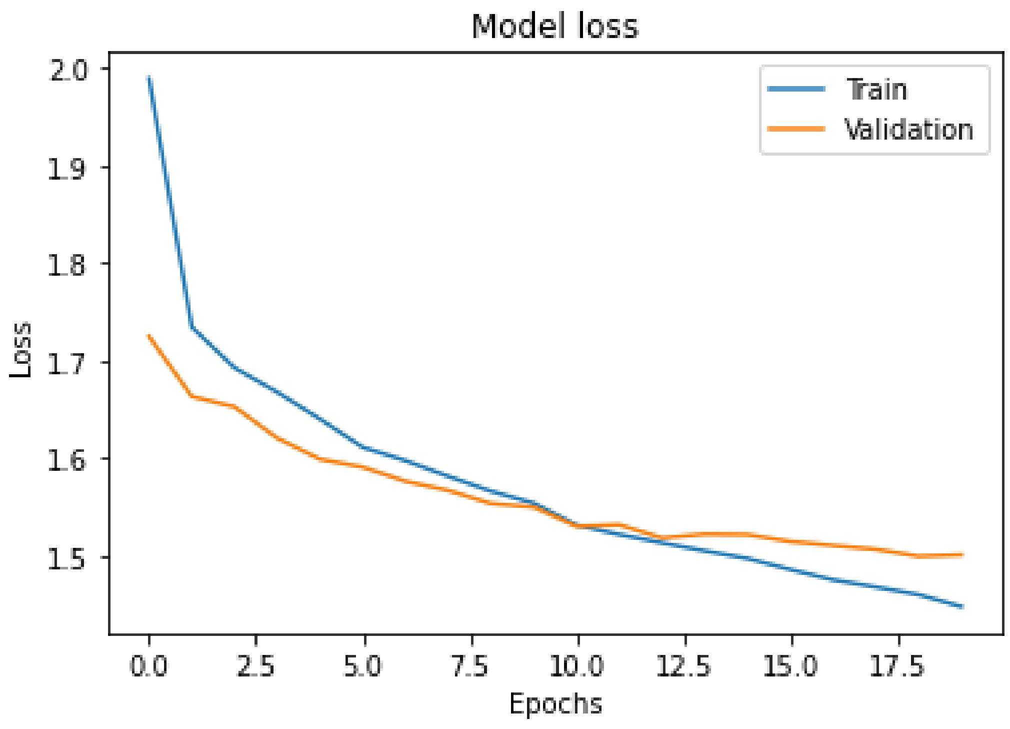
```
57/57 [==============================] - 0s 5ms/step - loss: 1.6109 - accuracy:
0.3601 - val_loss: 1.5908 - val_accuracy: 0.3661
Epoch 7/20
56/57 [============================>.] - ETA: 0s - loss: 1.5974 - accuracy:
0.3667
Epoch 00007: val_loss improved from 1.59079 to 1.57615, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.5973 - accuracy:
0.3667 - val_loss: 1.5761 - val_accuracy: 0.3828
Epoch 8/20
56/57 [============================>.] - ETA: 0s - loss: 1.5814 - accuracy:
0.3744
Epoch 00008: val_loss improved from 1.57615 to 1.56677, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.5813 - accuracy:
0.3745 - val_loss: 1.5668 - val_accuracy: 0.3926
Epoch 9/20
56/57 [============================>.] - ETA: 0s - loss: 1.5661 - accuracy:
0.3793
Epoch 00009: val_loss improved from 1.56677 to 1.55329, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.5660 - accuracy:
0.3793 - val_loss: 1.5533 - val_accuracy: 0.3906
Epoch 10/20
56/57 [============================>.] - ETA: 0s - loss: 1.5536 - accuracy:
0.3891
Epoch 00010: val_loss improved from 1.55329 to 1.54969, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.5537 - accuracy:
0.3891 - val_loss: 1.5497 - val_accuracy: 0.4110
Epoch 11/20
55/57 [===========================>..] - ETA: 0s - loss: 1.5313 - accuracy:
0.3970
Epoch 00011: val_loss improved from 1.54969 to 1.52988, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.5312 - accuracy:
0.3970 - val_loss: 1.5299 - val_accuracy: 0.4015
Epoch 12/20
54/57 [===========================>..] - ETA: 0s - loss: 1.5213 - accuracy:
0.3971
Epoch 00012: val_loss did not improve from 1.52988
57/57 [==============================] - 0s 5ms/step - loss: 1.5215 - accuracy:
0.3966 - val_loss: 1.5315 - val_accuracy: 0.3976
Epoch 13/20
52/57 [==========================>...] - ETA: 0s - loss: 1.5145 - accuracy:
0.3987
Epoch 00013: val_loss improved from 1.52988 to 1.51797, saving model to
best_fnn_model2.hdf5
```

```
57/57 [==============================] - 0s 5ms/step - loss: 1.5129 - accuracy:
0.3992 - val_loss: 1.5180 - val_accuracy: 0.4037
Epoch 14/20
51/57 [=========================>...] - ETA: 0s - loss: 1.5059 - accuracy:
0.4092
Epoch 00014: val_loss did not improve from 1.51797
57/57 [==============================] - 0s 5ms/step - loss: 1.5046 - accuracy:
0.4090 - val_loss: 1.5218 - val_accuracy: 0.4126
Epoch 15/20
54/57 [===========================>..] - ETA: 0s - loss: 1.4967 - accuracy:
0.4095
Epoch 00015: val_loss did not improve from 1.51797
57/57 [==============================] - 0s 5ms/step - loss: 1.4968 - accuracy:
0.4099 - val_loss: 1.5213 - val_accuracy: 0.4026
Epoch 16/20
57/57 [==============================] - ETA: 0s - loss: 1.4855 - accuracy:
0.4121
Epoch 00016: val_loss improved from 1.51797 to 1.51432, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.4855 - accuracy:
0.4121 - val_loss: 1.5143 - val_accuracy: 0.4085
Epoch 17/20
56/57 [============================>.] - ETA: 0s - loss: 1.4747 - accuracy:
0.4156
Epoch 00017: val_loss improved from 1.51432 to 1.51024, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.4749 - accuracy:
0.4156 - val_loss: 1.5102 - val_accuracy: 0.4060
Epoch 18/20
56/57 [============================>.] - ETA: 0s - loss: 1.4675 - accuracy:
0.4237
Epoch 00018: val_loss improved from 1.51024 to 1.50626, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.4675 - accuracy:
0.4237 - val_loss: 1.5063 - val_accuracy: 0.3979
Epoch 19/20
55/57 [===========================>..] - ETA: 0s - loss: 1.4610 - accuracy:
0.4236
Epoch 00019: val_loss improved from 1.50626 to 1.49913, saving model to
best_fnn_model2.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.4596 - accuracy:
0.4246 - val_loss: 1.4991 - val_accuracy: 0.4107
Epoch 20/20
55/57 [===========================>..] - ETA: 0s - loss: 1.4471 - accuracy:
0.4292
Epoch 00020: val_loss did not improve from 1.49913
57/57 [==============================] - 0s 5ms/step - loss: 1.4474 - accuracy:
0.4294 - val_loss: 1.5005 - val_accuracy: 0.4140
```

[28]:
```
fnn_model2.summary()
print(time_to_run)
```

```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 256)               590080

 dropout (Dropout)           (None, 256)               0

 dense_5 (Dense)             (None, 64)                16448

 dropout_1 (Dropout)         (None, 64)                0

 dense_6 (Dense)             (None, 7)                 455

=================================================================
Total params: 606,983
Trainable params: 606,983
Non-trainable params: 0
_________________________________________________________________
10.956292867660522
```

[29]:
```
plt.plot(history_fnn2.history['loss'])
plt.plot(history_fnn2.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



[30]:
```
# Neural network
fnn_model3 = Sequential()
fnn_model3.add(Dense(512, input_dim = 2304, activation ='sigmoid'))
fnn_model3.add(Dense(64, activation = 'sigmoid'))
fnn_model3.add(Dropout(0.4))
fnn_model3.add(Dense(7, activation = 'softmax'))
```

[31]:
```
fnn_checkpoint3 = ModelCheckpoint("best_fnn_model3.hdf5", monitor = 'val_loss',␣
 ↪verbose = 1,
                                  save_best_only = True, mode =' min', period =␣
 ↪1)
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to
specify the frequency in number of batches seen.
WARNING:tensorflow:ModelCheckpoint mode  min is unknown, fallback to auto mode.
```

[32]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.0005)

fnn_model3.compile(loss='categorical_crossentropy', optimizer = optimizer,␣
 ↪metrics = ['accuracy'])
```

[33]:
```
start = time()

history_fnn3 = fnn_model3.fit(X_train_fnn, y_train,
                    validation_data = (X_val_fnn, y_val),
                    epochs = 20, batch_size = 512,
                    validation_batch_size = 512,
                    callbacks=[fnn_checkpoint3])

time_to_run = time() - start
```

```
Epoch 1/20
52/57 [==========================>...] - ETA: 0s - loss: 1.8477 - accuracy:
0.2559
Epoch 00001: val_loss improved from inf to 1.68260, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 1s 8ms/step - loss: 1.8389 - accuracy:
0.2594 - val_loss: 1.6826 - val_accuracy: 0.3541
Epoch 2/20
52/57 [==========================>...] - ETA: 0s - loss: 1.7005 - accuracy:
0.3314
Epoch 00002: val_loss improved from 1.68260 to 1.63446, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6987 - accuracy:
0.3332 - val_loss: 1.6345 - val_accuracy: 0.3717
Epoch 3/20
54/57 [===========================>..] - ETA: 0s - loss: 1.6589 - accuracy:
0.3526
Epoch 00003: val_loss improved from 1.63446 to 1.61218, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.6579 - accuracy:
0.3538 - val_loss: 1.6122 - val_accuracy: 0.3753
Epoch 4/20
53/57 [==========================>...] - ETA: 0s - loss: 1.6246 - accuracy:
0.3691
Epoch 00004: val_loss improved from 1.61218 to 1.58756, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.6249 - accuracy:
0.3679 - val_loss: 1.5876 - val_accuracy: 0.3848
Epoch 5/20
51/57 [=========================>...] - ETA: 0s - loss: 1.5987 - accuracy:
0.3782
Epoch 00005: val_loss improved from 1.58756 to 1.57362, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.5974 - accuracy:
0.3792 - val_loss: 1.5736 - val_accuracy: 0.3951
Epoch 6/20
50/57 [=========================>...] - ETA: 0s - loss: 1.5718 - accuracy:
0.3876
Epoch 00006: val_loss improved from 1.57362 to 1.55999, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.5723 - accuracy:
```

```
0.3882 - val_loss: 1.5600 - val_accuracy: 0.3940
Epoch 7/20
53/57 [==========================>...] - ETA: 0s - loss: 1.5552 - accuracy:
0.3947
Epoch 00007: val_loss improved from 1.55999 to 1.54675, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 5ms/step - loss: 1.5555 - accuracy:
0.3951 - val_loss: 1.5467 - val_accuracy: 0.3982
Epoch 8/20
51/57 [=========================>...] - ETA: 0s - loss: 1.5263 - accuracy:
0.4096
Epoch 00008: val_loss improved from 1.54675 to 1.54018, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.5285 - accuracy:
0.4081 - val_loss: 1.5402 - val_accuracy: 0.4029
Epoch 9/20
51/57 [=========================>...] - ETA: 0s - loss: 1.5102 - accuracy:
0.4207
Epoch 00009: val_loss improved from 1.54018 to 1.52771, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.5084 - accuracy:
0.4212 - val_loss: 1.5277 - val_accuracy: 0.4037
Epoch 10/20
52/57 [==========================>...] - ETA: 0s - loss: 1.4898 - accuracy:
0.4272
Epoch 00010: val_loss improved from 1.52771 to 1.52365, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.4901 - accuracy:
0.4263 - val_loss: 1.5237 - val_accuracy: 0.4029
Epoch 11/20
50/57 [=========================>...] - ETA: 0s - loss: 1.4687 - accuracy:
0.4368
Epoch 00011: val_loss improved from 1.52365 to 1.50973, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.4678 - accuracy:
0.4373 - val_loss: 1.5097 - val_accuracy: 0.4093
Epoch 12/20
50/57 [=========================>...] - ETA: 0s - loss: 1.4422 - accuracy:
0.4473
Epoch 00012: val_loss improved from 1.50973 to 1.50197, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.4415 - accuracy:
0.4483 - val_loss: 1.5020 - val_accuracy: 0.4179
Epoch 13/20
52/57 [==========================>...] - ETA: 0s - loss: 1.4184 - accuracy:
0.4546
Epoch 00013: val_loss improved from 1.50197 to 1.49981, saving model to
best_fnn_model3.hdf5
```

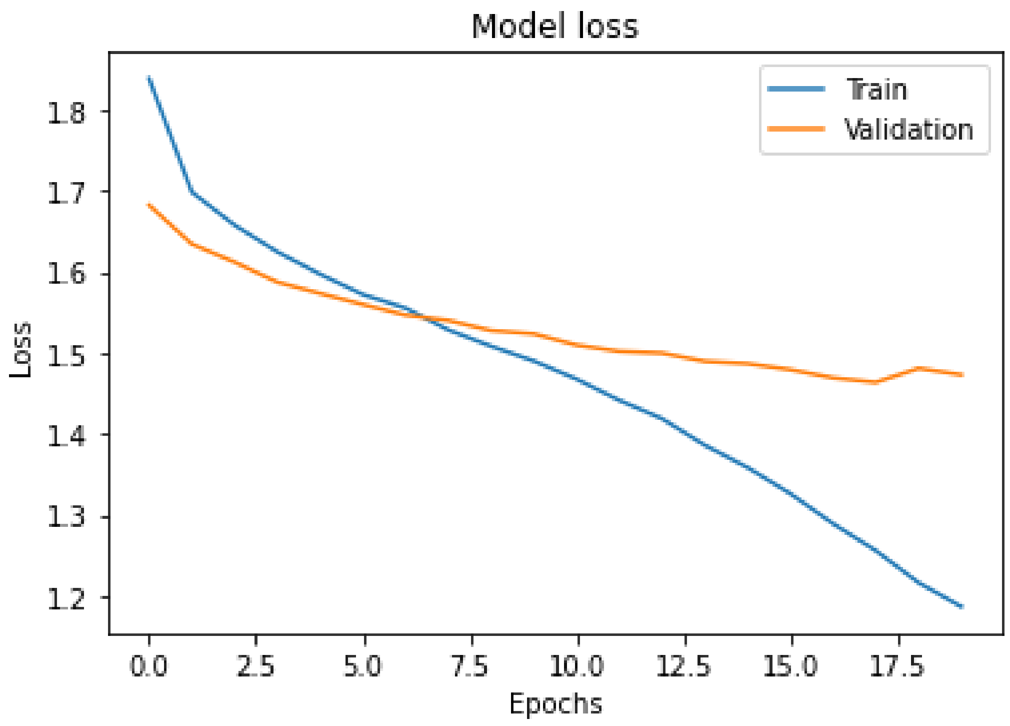
```
57/57 [==============================] - 0s 5ms/step - loss: 1.4188 - accuracy:
0.4560 - val_loss: 1.4998 - val_accuracy: 0.4149
Epoch 14/20
50/57 [=========================>...] - ETA: 0s - loss: 1.3876 - accuracy:
0.4762
Epoch 00014: val_loss improved from 1.49981 to 1.48964, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.3860 - accuracy:
0.4770 - val_loss: 1.4896 - val_accuracy: 0.4255
Epoch 15/20
49/57 [========================>...] - ETA: 0s - loss: 1.3595 - accuracy:
0.4860
Epoch 00015: val_loss improved from 1.48964 to 1.48674, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.3587 - accuracy:
0.4870 - val_loss: 1.4867 - val_accuracy: 0.4213
Epoch 16/20
52/57 [==========================>...] - ETA: 0s - loss: 1.3271 - accuracy:
0.5047
Epoch 00016: val_loss improved from 1.48674 to 1.47938, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.3264 - accuracy:
0.5048 - val_loss: 1.4794 - val_accuracy: 0.4299
Epoch 17/20
51/57 [=========================>...] - ETA: 0s - loss: 1.2889 - accuracy:
0.5183
Epoch 00017: val_loss improved from 1.47938 to 1.46924, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 0s 6ms/step - loss: 1.2893 - accuracy:
0.5176 - val_loss: 1.4692 - val_accuracy: 0.4391
Epoch 18/20
50/57 [=========================>...] - ETA: 0s - loss: 1.2578 - accuracy:
0.5339
Epoch 00018: val_loss improved from 1.46924 to 1.46370, saving model to
best_fnn_model3.hdf5
57/57 [==============================] - 1s 14ms/step - loss: 1.2560 - accuracy:
0.5348 - val_loss: 1.4637 - val_accuracy: 0.4344
Epoch 19/20
49/57 [========================>...] - ETA: 0s - loss: 1.2160 - accuracy:
0.5506
Epoch 00019: val_loss did not improve from 1.46370
57/57 [==============================] - 0s 5ms/step - loss: 1.2163 - accuracy:
0.5509 - val_loss: 1.4809 - val_accuracy: 0.4285
Epoch 20/20
51/57 [=========================>...] - ETA: 0s - loss: 1.1869 - accuracy:
0.5656
Epoch 00020: val_loss did not improve from 1.46370
57/57 [==============================] - 0s 5ms/step - loss: 1.1867 - accuracy:
```

```
0.5653 - val_loss: 1.4731 - val_accuracy: 0.4411
```

[34]:
```
fnn_model3.summary()
print(time_to_run)
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_7 (Dense)             (None, 512)               1180160

 dense_8 (Dense)             (None, 64)                32832

 dropout_2 (Dropout)         (None, 64)                0

 dense_9 (Dense)             (None, 7)                 455

=================================================================
Total params: 1,213,447
Trainable params: 1,213,447
Non-trainable params: 0
_________________________________________________________________
7.610275983810425
```

[35]:
```
plt.plot(history_fnn3.history['loss'])
plt.plot(history_fnn3.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



c(i). Details of the 3 different FNN models are listed below:

#Details on FNN Model 1:

Number of hidden layers: 2

Number of neurons in 4 layers: 128, 64, 32, 7

Total parameters: 305607

Optimizer used: Adam with learning rate of 0.001

Regulerizer used: l2

Dropout: N/A

Activation function for input and hidden layers: ReLU

Activation function for output layer: Softmax

Training accuracy (for the best model): 43.35%

Validation accuracy (for the best model): 39.40%

Training time: 8.26 s

#Details on FNN Model 2:

Number of hidden layers: 1

Number of neurons in 3 layers: 256, 64, 7

Total parameters: 606983

Optimizer used: Adam with learning rate of 0.002

Regulerizer used: N/A

Dropout: 0.5 after 1st layer, 0.2 after 2nd layer

Activation function for input and hidden layers: ReLU

Activation function for output layer: Softmax

Training accuracy (for the best model): 42.46%

Validation accuracy (for the best model): 41.07%

Training time: 10.95 s

#Details on FNN Model 3:

Number of hidden layers: 1

Number of neurons in 3 layers: 512, 64, 7

Total parameters: 1213447

Optimizer used: Adam with learning rate of 0.0005

Regulerizer used: N/A

Dropout: 0.4 after 2nd layer

Activation function for input and hidden layers: Sigmoid

Activation function for output layer: Softmax

Training accuracy (for the best model): 53.48%

Validation accuracy (for the best model): 43.44%

Training time: 7.61 s

We get a very clear idea about how different models are performing with this training set from the listing above. Even though the model 3 achieves the best training and validation accuracy for the best weights saved during training process, it has an overfitting tendency as per the loss curve. On the other hand model 2 performed similarly on the validation set and has less overfitting tendency overall. So, model 2 can be safely chosen as the best model and used for testing.

[36]:
```
best_fnn_model = tf.keras.models.load_model("best_fnn_model3.hdf5")
score = best_fnn_model.evaluate(X_test_fnn, y_test, verbose=0)
print("%s: %.2f%%" % (best_fnn_model.metrics_names[1], score[1]*100))
```

```
accuracy: 43.69%
```

c(ii). Emotion classification accuracy on the testing set = 43.69%

[37]:
```
from keras.layers import Activation, Flatten, Conv2D, MaxPooling2D
from keras.layers import BatchNormalization
from keras.utils import np_utils
```

[38]:
```
cnn_model1 = Sequential()

cnn_model1.add(Conv2D(filters=96, input_shape=(48, 48, 1), kernel_regularizer =␣
 ↪regularizers.l2(l2=0.01),
                      kernel_size=(11,11), strides=(4,4), padding='same'))
cnn_model1.add(BatchNormalization())
cnn_model1.add(Activation('relu'))
cnn_model1.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

cnn_model1.add(Conv2D(filters=256, kernel_size=(5, 5), kernel_regularizer =␣
 ↪regularizers.l2(l2=0.01),
                      strides=(1,1), padding='same'))
cnn_model1.add(BatchNormalization())
cnn_model1.add(Activation('relu'))
cnn_model1.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

cnn_model1.add(Conv2D(filters=384, kernel_size=(3,3), kernel_regularizer =␣
 ↪regularizers.l2(l2=0.01),
                      strides=(1,1), padding='same'))
cnn_model1.add(BatchNormalization())
cnn_model1.add(Activation('relu'))

cnn_model1.add(Flatten())

cnn_model1.add(Dense(4096, input_shape=(32,32,3,)))
cnn_model1.add(BatchNormalization())
cnn_model1.add(Activation('relu'))
cnn_model1.add(Dropout(0.4))

cnn_model1.add(Dense(1000))
cnn_model1.add(BatchNormalization())
cnn_model1.add(Activation('relu'))
cnn_model1.add(Dropout(0.4))

cnn_model1.add(Dense(7))
cnn_model1.add(BatchNormalization())
cnn_model1.add(Activation('softmax'))


cnn_model1.summary()
```

```
Model: "sequential_3"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 12, 12, 96)        11712

 batch_normalization (BatchN  (None, 12, 12, 96)       384
 ormalization)

 activation (Activation)     (None, 12, 12, 96)        0

 max_pooling2d (MaxPooling2D  (None, 6, 6, 96)         0
 )

 conv2d_1 (Conv2D)           (None, 6, 6, 256)         614656

 batch_normalization_1 (Batc  (None, 6, 6, 256)        1024
 hNormalization)

 activation_1 (Activation)   (None, 6, 6, 256)         0

 max_pooling2d_1 (MaxPooling  (None, 3, 3, 256)        0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 384)         885120

 batch_normalization_2 (Batc  (None, 3, 3, 384)        1536
 hNormalization)

 activation_2 (Activation)   (None, 3, 3, 384)         0

 flatten (Flatten)           (None, 3456)              0

 dense_10 (Dense)            (None, 4096)              14159872

 batch_normalization_3 (Batc  (None, 4096)             16384
 hNormalization)

 activation_3 (Activation)   (None, 4096)              0

 dropout_3 (Dropout)         (None, 4096)              0

 dense_11 (Dense)            (None, 1000)              4097000

 batch_normalization_4 (Batc  (None, 1000)             4000
 hNormalization)

 activation_4 (Activation)   (None, 1000)              0
 dropout_4 (Dropout)         (None, 1000)              0

 dense_12 (Dense)            (None, 7)                 7007

 batch_normalization_5 (Batc  (None, 7)                28
 hNormalization)

 activation_5 (Activation)   (None, 7)                 0

=================================================================
Total params: 19,798,723
Trainable params: 19,787,045
Non-trainable params: 11,678
_________________________________________________________________
```

[39]:
```
cnn_checkpoint1 = ModelCheckpoint("best_cnn_model1.hdf5", monitor = 'val_loss',␣
 ↪verbose = 1,
    save_best_only = True, mode =' min', period = 1)
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to
specify the frequency in number of batches seen.
WARNING:tensorflow:ModelCheckpoint mode  min is unknown, fallback to auto mode.
```

[40]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.0005)

cnn_model1.compile(loss='categorical_crossentropy', optimizer = optimizer,␣
 ↪metrics = ['accuracy'])
```

[41]:
```
start = time()

history_cnn1 = cnn_model1.fit(X_train, y_train,
                    validation_data = (X_val, y_val),
                    epochs = 20, batch_size = 512,
                    validation_batch_size = 512,
                    callbacks=[cnn_checkpoint1])

time_to_run = time() - start
```

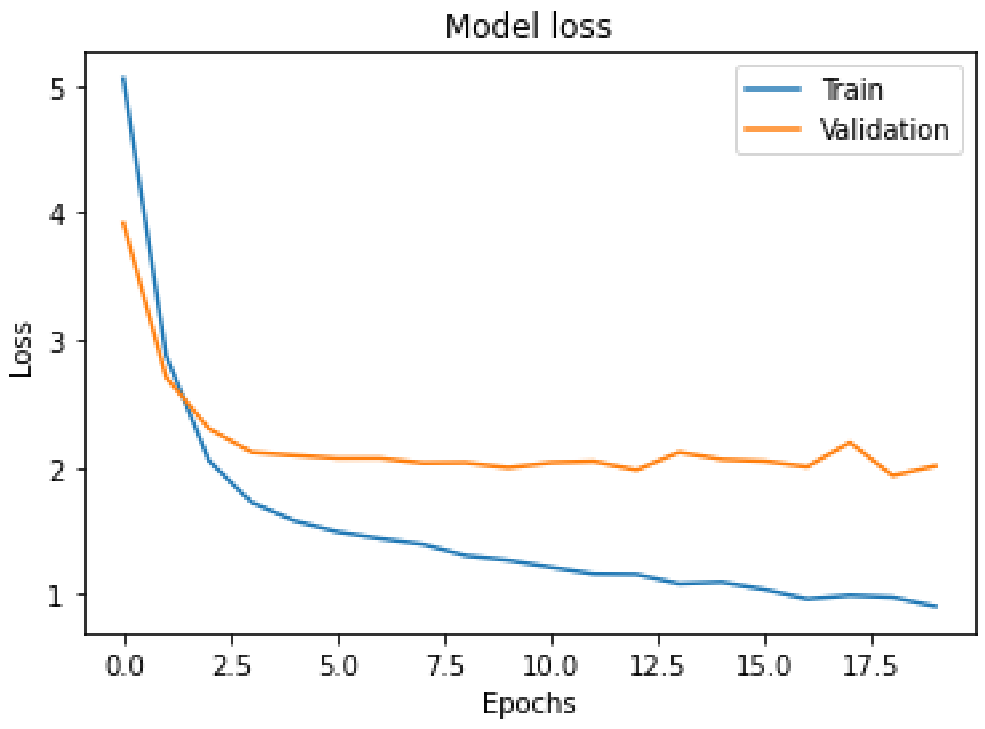
```
Epoch 1/20
57/57 [==============================] - ETA: 0s - loss: 5.0546 - accuracy:
0.3671
Epoch 00001: val_loss improved from inf to 3.91944, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 18s 44ms/step - loss: 5.0546 -
accuracy: 0.3671 - val_loss: 3.9194 - val_accuracy: 0.1744
Epoch 2/20
55/57 [===========================>..] - ETA: 0s - loss: 2.8840 - accuracy:
0.4586
Epoch 00002: val_loss improved from 3.91944 to 2.70535, saving model to
```

```
best_cnn_model1.hdf5
57/57 [==============================] - 2s 40ms/step - loss: 2.8750 - accuracy:
0.4583 - val_loss: 2.7053 - val_accuracy: 0.3282
Epoch 3/20
55/57 [===========================>..] - ETA: 0s - loss: 2.0535 - accuracy:
0.4964
Epoch 00003: val_loss improved from 2.70535 to 2.30108, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 4s 78ms/step - loss: 2.0510 - accuracy:
0.4958 - val_loss: 2.3011 - val_accuracy: 0.2558
Epoch 4/20
55/57 [===========================>..] - ETA: 0s - loss: 1.7227 - accuracy:
0.5410
Epoch 00004: val_loss improved from 2.30108 to 2.11499, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 2s 39ms/step - loss: 1.7216 - accuracy:
0.5409 - val_loss: 2.1150 - val_accuracy: 0.3706
Epoch 5/20
55/57 [===========================>..] - ETA: 0s - loss: 1.5765 - accuracy:
0.5695
Epoch 00005: val_loss improved from 2.11499 to 2.09084, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 2s 39ms/step - loss: 1.5769 - accuracy:
0.5688 - val_loss: 2.0908 - val_accuracy: 0.3943
Epoch 6/20
55/57 [===========================>..] - ETA: 0s - loss: 1.4895 - accuracy:
0.6037
Epoch 00006: val_loss improved from 2.09084 to 2.06818, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 3s 60ms/step - loss: 1.4900 - accuracy:
0.6034 - val_loss: 2.0682 - val_accuracy: 0.3714
Epoch 7/20
55/57 [===========================>..] - ETA: 0s - loss: 1.4363 - accuracy:
0.6302
Epoch 00007: val_loss did not improve from 2.06818
57/57 [==============================] - 2s 32ms/step - loss: 1.4373 - accuracy:
0.6295 - val_loss: 2.0689 - val_accuracy: 0.3658
Epoch 8/20
55/57 [===========================>..] - ETA: 0s - loss: 1.3921 - accuracy:
0.6560
Epoch 00008: val_loss improved from 2.06818 to 2.03060, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 2s 39ms/step - loss: 1.3923 - accuracy:
0.6560 - val_loss: 2.0306 - val_accuracy: 0.3892
Epoch 9/20
55/57 [===========================>..] - ETA: 0s - loss: 1.3024 - accuracy:
0.6950
Epoch 00009: val_loss did not improve from 2.03060
```

```
57/57 [==============================] - 2s 32ms/step - loss: 1.3027 - accuracy:
0.6947 - val_loss: 2.0339 - val_accuracy: 0.3717
Epoch 10/20
55/57 [============================>..] - ETA: 0s - loss: 1.2644 - accuracy:
0.7202
Epoch 00010: val_loss improved from 2.03060 to 1.99577, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 2s 40ms/step - loss: 1.2663 - accuracy:
0.7196 - val_loss: 1.9958 - val_accuracy: 0.4377
Epoch 11/20
55/57 [============================>..] - ETA: 0s - loss: 1.2107 - accuracy:
0.7450
Epoch 00011: val_loss did not improve from 1.99577
57/57 [==============================] - 2s 33ms/step - loss: 1.2110 - accuracy:
0.7449 - val_loss: 2.0342 - val_accuracy: 0.3982
Epoch 12/20
55/57 [============================>..] - ETA: 0s - loss: 1.1583 - accuracy:
0.7686
Epoch 00012: val_loss did not improve from 1.99577
57/57 [==============================] - 2s 32ms/step - loss: 1.1590 - accuracy:
0.7683 - val_loss: 2.0444 - val_accuracy: 0.3720
Epoch 13/20
55/57 [============================>..] - ETA: 0s - loss: 1.1548 - accuracy:
0.7748
Epoch 00013: val_loss improved from 1.99577 to 1.97698, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 2s 40ms/step - loss: 1.1551 - accuracy:
0.7749 - val_loss: 1.9770 - val_accuracy: 0.4316
Epoch 14/20
55/57 [============================>..] - ETA: 0s - loss: 1.0806 - accuracy:
0.8058
Epoch 00014: val_loss did not improve from 1.97698
57/57 [==============================] - 2s 32ms/step - loss: 1.0807 - accuracy:
0.8059 - val_loss: 2.1152 - val_accuracy: 0.3243
Epoch 15/20
55/57 [============================>..] - ETA: 0s - loss: 1.0940 - accuracy:
0.8010
Epoch 00015: val_loss did not improve from 1.97698
57/57 [==============================] - 2s 32ms/step - loss: 1.0932 - accuracy:
0.8014 - val_loss: 2.0595 - val_accuracy: 0.4060
Epoch 16/20
55/57 [============================>..] - ETA: 0s - loss: 1.0378 - accuracy:
0.8272
Epoch 00016: val_loss did not improve from 1.97698
57/57 [==============================] - 2s 32ms/step - loss: 1.0367 - accuracy:
0.8276 - val_loss: 2.0457 - val_accuracy: 0.3848
Epoch 17/20
55/57 [============================>..] - ETA: 0s - loss: 0.9604 - accuracy:
```

```
0.8545
Epoch 00017: val_loss did not improve from 1.97698
57/57 [==============================] - 2s 32ms/step - loss: 0.9606 - accuracy:
0.8542 - val_loss: 2.0032 - val_accuracy: 0.4503
Epoch 18/20
55/57 [===========================>..] - ETA: 0s - loss: 0.9898 - accuracy:
0.8447
Epoch 00018: val_loss did not improve from 1.97698
57/57 [==============================] - 2s 32ms/step - loss: 0.9897 - accuracy:
0.8446 - val_loss: 2.1920 - val_accuracy: 0.3422
Epoch 19/20
55/57 [===========================>..] - ETA: 0s - loss: 0.9753 - accuracy:
0.8522
Epoch 00019: val_loss improved from 1.97698 to 1.93276, saving model to
best_cnn_model1.hdf5
57/57 [==============================] - 2s 39ms/step - loss: 0.9750 - accuracy:
0.8523 - val_loss: 1.9328 - val_accuracy: 0.4645
Epoch 20/20
55/57 [===========================>..] - ETA: 0s - loss: 0.9036 - accuracy:
0.8762
Epoch 00020: val_loss did not improve from 1.93276
57/57 [==============================] - 2s 32ms/step - loss: 0.9028 - accuracy:
0.8766 - val_loss: 2.0102 - val_accuracy: 0.4257
```

[42]:
```
plt.plot(history_cnn1.history['loss'])
plt.plot(history_cnn1.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()

print(time_to_run)
```



```
83.49928736686707
```

[43]:
```
cnn_model2 = Sequential()

cnn_model2.add(Conv2D(filters=96, input_shape=(48, 48, 1), kernel_regularizer =␣
 ↪regularizers.l2(l2=0.01),
                      kernel_size=(11,11), strides=(4,4), padding='same'))
cnn_model2.add(Activation('relu'))
cnn_model2.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

cnn_model2.add(Conv2D(filters=48, kernel_size=(3,3), kernel_regularizer =␣
 ↪regularizers.l2(l2=0.01),
                      strides=(1,1), padding='same'))
cnn_model2.add(Activation('relu'))

cnn_model2.add(Flatten())

cnn_model2.add(Dense(1048, input_shape=(32,32,3,)))
cnn_model2.add(Activation('sigmoid'))
cnn_model2.add(Dropout(0.4))

cnn_model2.add(Dense(128))
cnn_model2.add(Activation('sigmoid'))
```

```
cnn_model2.add(Dropout(0.4))

cnn_model2.add(Dense(7))
cnn_model2.add(Activation('softmax'))


cnn_model2.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 12, 12, 96)        11712

 activation_6 (Activation)   (None, 12, 12, 96)        0

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 96)         0
 2D)

 conv2d_4 (Conv2D)           (None, 6, 6, 48)          41520

 activation_7 (Activation)   (None, 6, 6, 48)          0

 flatten_1 (Flatten)         (None, 1728)              0

 dense_13 (Dense)            (None, 1048)              1811992

 activation_8 (Activation)   (None, 1048)              0

 dropout_5 (Dropout)         (None, 1048)              0

 dense_14 (Dense)            (None, 128)               134272

 activation_9 (Activation)   (None, 128)               0

 dropout_6 (Dropout)         (None, 128)               0

 dense_15 (Dense)            (None, 7)                 903

 activation_10 (Activation)  (None, 7)                 0

=================================================================
Total params: 2,000,399
Trainable params: 2,000,399
Non-trainable params: 0
_________________________________________________________________
```

[44]:
```
cnn_checkpoint2 = ModelCheckpoint("best_cnn_model2.hdf5", monitor = 'val_loss',␣
 ↪verbose = 1,
    save_best_only = True, mode =' min', period = 1)
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to
specify the frequency in number of batches seen.
WARNING:tensorflow:ModelCheckpoint mode  min is unknown, fallback to auto mode.
```

[45]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.0005)

cnn_model2.compile(loss='categorical_crossentropy', optimizer = optimizer,␣
 ↪metrics = ['accuracy'])
```

[46]:
```
start = time()

history_cnn2 = cnn_model2.fit(X_train, y_train,
                    validation_data = (X_val, y_val),
                    epochs = 20, batch_size = 512,
                    validation_batch_size = 512,
                    callbacks=[cnn_checkpoint2])

time_to_run = time() - start
```

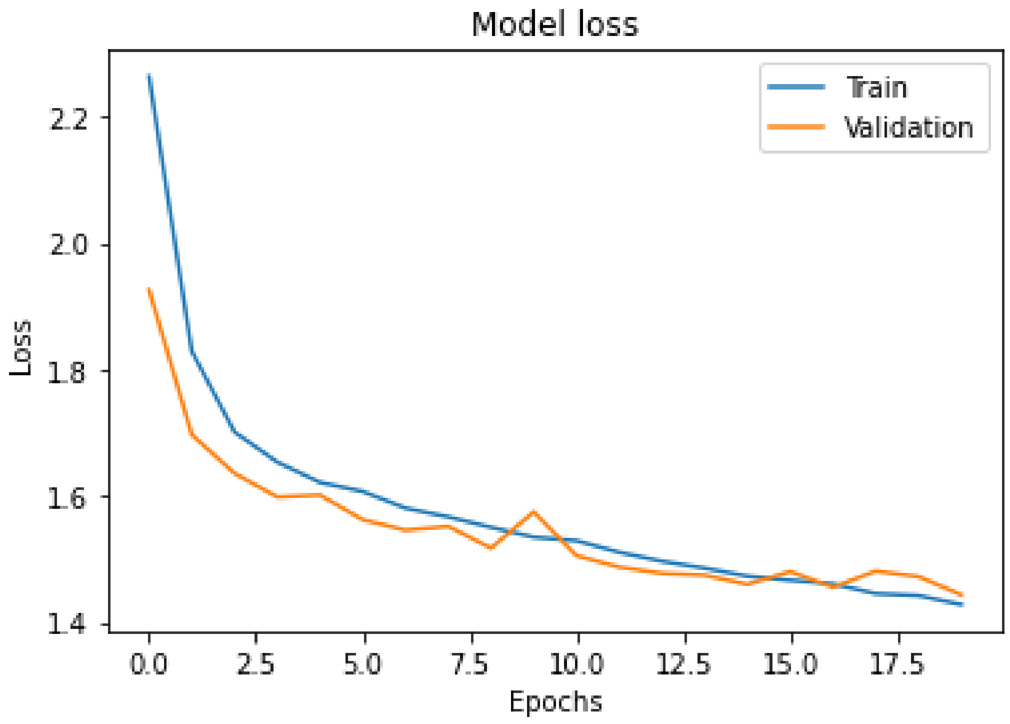
```
Epoch 1/20
56/57 [============================>.] - ETA: 0s - loss: 2.2635 - accuracy:
0.2295
Epoch 00001: val_loss improved from inf to 1.92585, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 14ms/step - loss: 2.2631 - accuracy:
0.2294 - val_loss: 1.9259 - val_accuracy: 0.3115
Epoch 2/20
53/57 [==========================>...] - ETA: 0s - loss: 1.8330 - accuracy:
0.3173
Epoch 00002: val_loss improved from 1.92585 to 1.69652, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.8285 - accuracy:
0.3187 - val_loss: 1.6965 - val_accuracy: 0.3795
Epoch 3/20
57/57 [==============================] - ETA: 0s - loss: 1.7008 - accuracy:
0.3550
Epoch 00003: val_loss improved from 1.69652 to 1.63573, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.7008 - accuracy:
0.3550 - val_loss: 1.6357 - val_accuracy: 0.3982
Epoch 4/20
51/57 [=========================>...] - ETA: 0s - loss: 1.6549 - accuracy:
0.3718
Epoch 00004: val_loss improved from 1.63573 to 1.59834, saving model to
```

```
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.6535 - accuracy:
0.3719 - val_loss: 1.5983 - val_accuracy: 0.4035
Epoch 5/20
55/57 [===========================>..] - ETA: 0s - loss: 1.6211 - accuracy:
0.3871
Epoch 00005: val_loss did not improve from 1.59834
57/57 [==============================] - 1s 10ms/step - loss: 1.6213 - accuracy:
0.3872 - val_loss: 1.6014 - val_accuracy: 0.4015
Epoch 6/20
54/57 [===========================>..] - ETA: 0s - loss: 1.6076 - accuracy:
0.3934
Epoch 00006: val_loss improved from 1.59834 to 1.56238, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.6070 - accuracy:
0.3936 - val_loss: 1.5624 - val_accuracy: 0.4146
Epoch 7/20
52/57 [==========================>...] - ETA: 0s - loss: 1.5796 - accuracy:
0.4080
Epoch 00007: val_loss improved from 1.56238 to 1.54572, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.5806 - accuracy:
0.4077 - val_loss: 1.5457 - val_accuracy: 0.4257
Epoch 8/20
56/57 [============================>.] - ETA: 0s - loss: 1.5671 - accuracy:
0.4122
Epoch 00008: val_loss did not improve from 1.54572
57/57 [==============================] - 1s 9ms/step - loss: 1.5668 - accuracy:
0.4123 - val_loss: 1.5517 - val_accuracy: 0.4205
Epoch 9/20
54/57 [===========================>..] - ETA: 0s - loss: 1.5505 - accuracy:
0.4245
Epoch 00009: val_loss improved from 1.54572 to 1.51739, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.5503 - accuracy:
0.4244 - val_loss: 1.5174 - val_accuracy: 0.4361
Epoch 10/20
56/57 [============================>.] - ETA: 0s - loss: 1.5345 - accuracy:
0.4299
Epoch 00010: val_loss did not improve from 1.51739
57/57 [==============================] - 1s 9ms/step - loss: 1.5347 - accuracy:
0.4298 - val_loss: 1.5746 - val_accuracy: 0.4060
Epoch 11/20
54/57 [===========================>..] - ETA: 0s - loss: 1.5281 - accuracy:
0.4331
Epoch 00011: val_loss improved from 1.51739 to 1.50540, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.5290 - accuracy:
```

```
0.4327 - val_loss: 1.5054 - val_accuracy: 0.4439
Epoch 12/20
51/57 [=========================>...] - ETA: 0s - loss: 1.5099 - accuracy:
0.4408
Epoch 00012: val_loss improved from 1.50540 to 1.48726, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.5108 - accuracy:
0.4409 - val_loss: 1.4873 - val_accuracy: 0.4458
Epoch 13/20
52/57 [==========================>...] - ETA: 0s - loss: 1.4965 - accuracy:
0.4497
Epoch 00013: val_loss improved from 1.48726 to 1.47732, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.4965 - accuracy:
0.4496 - val_loss: 1.4773 - val_accuracy: 0.4581
Epoch 14/20
51/57 [=========================>...] - ETA: 0s - loss: 1.4832 - accuracy:
0.4546
Epoch 00014: val_loss improved from 1.47732 to 1.47441, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.4855 - accuracy:
0.4527 - val_loss: 1.4744 - val_accuracy: 0.4581
Epoch 15/20
53/57 [==========================>...] - ETA: 0s - loss: 1.4725 - accuracy:
0.4605
Epoch 00015: val_loss improved from 1.47441 to 1.45998, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.4729 - accuracy:
0.4599 - val_loss: 1.4600 - val_accuracy: 0.4695
Epoch 16/20
51/57 [=========================>...] - ETA: 0s - loss: 1.4690 - accuracy:
0.4630
Epoch 00016: val_loss did not improve from 1.45998
57/57 [==============================] - 1s 9ms/step - loss: 1.4664 - accuracy:
0.4649 - val_loss: 1.4801 - val_accuracy: 0.4542
Epoch 17/20
53/57 [==========================>...] - ETA: 0s - loss: 1.4608 - accuracy:
0.4636
Epoch 00017: val_loss improved from 1.45998 to 1.45538, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.4608 - accuracy:
0.4635 - val_loss: 1.4554 - val_accuracy: 0.4692
Epoch 18/20
53/57 [==========================>...] - ETA: 0s - loss: 1.4450 - accuracy:
0.4740
Epoch 00018: val_loss did not improve from 1.45538
57/57 [==============================] - 1s 9ms/step - loss: 1.4453 - accuracy:
0.4735 - val_loss: 1.4808 - val_accuracy: 0.4595
```

```
Epoch 19/20
51/57 [=========================>...] - ETA: 0s - loss: 1.4402 - accuracy:
0.4741
Epoch 00019: val_loss did not improve from 1.45538
57/57 [==============================] - 1s 9ms/step - loss: 1.4421 - accuracy:
0.4733 - val_loss: 1.4723 - val_accuracy: 0.4578
Epoch 20/20
51/57 [=========================>...] - ETA: 0s - loss: 1.4258 - accuracy:
0.4845
Epoch 00020: val_loss improved from 1.45538 to 1.44347, saving model to
best_cnn_model2.hdf5
57/57 [==============================] - 1s 11ms/step - loss: 1.4280 - accuracy:
0.4821 - val_loss: 1.4435 - val_accuracy: 0.4734
```

[47]:
```
plt.plot(history_cnn2.history['loss'])
plt.plot(history_cnn2.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()

print(time_to_run)
```



```
12.479339838027954
```

[48]:
```
cnn_model3 = Sequential()

cnn_model3.add(Conv2D(filters=48, input_shape=(48, 48, 1), kernel_regularizer =␣
 ↪regularizers.l2(l2=0.01),
                      kernel_size=(5,5), strides=(2,2), padding='same'))
cnn_model3.add(Activation('relu'))
cnn_model3.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

cnn_model3.add(Conv2D(filters=16, kernel_size=(3,3), kernel_regularizer =␣
 ↪regularizers.l2(l2=0.01),
                      strides=(1,1), padding='same'))
cnn_model3.add(Activation('relu'))
cnn_model3.add(Dropout(0.4))

cnn_model3.add(Flatten())

cnn_model3.add(Dense(256))
cnn_model3.add(Activation('relu'))
cnn_model3.add(Dropout(0.2))

cnn_model3.add(Dense(64))
cnn_model3.add(Activation('relu'))
cnn_model3.add(Dropout(0.2))

cnn_model3.add(Dense(7))
cnn_model3.add(Activation('softmax'))

cnn_model3.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 24, 24, 48)        1248

 activation_11 (Activation)  (None, 24, 24, 48)        0

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 48)       0
 2D)

 conv2d_6 (Conv2D)           (None, 12, 12, 16)        6928

 activation_12 (Activation)  (None, 12, 12, 16)        0

 dropout_7 (Dropout)         (None, 12, 12, 16)        0
 flatten_2 (Flatten)         (None, 2304)              0

 dense_16 (Dense)            (None, 256)               590080

 activation_13 (Activation)  (None, 256)               0

 dropout_8 (Dropout)         (None, 256)               0

 dense_17 (Dense)            (None, 64)                16448

 activation_14 (Activation)  (None, 64)                0

 dropout_9 (Dropout)         (None, 64)                0

 dense_18 (Dense)            (None, 7)                 455

 activation_15 (Activation)  (None, 7)                 0

=================================================================
Total params: 615,159
Trainable params: 615,159
Non-trainable params: 0
_________________________________________________________________
```

[49]:
```
cnn_checkpoint3 = ModelCheckpoint("best_cnn_model3.hdf5", monitor = 'val_loss',␣
 ↪verbose = 1,
    save_best_only = True, mode =' min', period = 1)
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to
specify the frequency in number of batches seen.
WARNING:tensorflow:ModelCheckpoint mode  min is unknown, fallback to auto mode.
```

[50]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.0005)

cnn_model3.compile(loss='categorical_crossentropy', optimizer = optimizer,␣
 ↪metrics = ['accuracy'])
```

[51]:
```
start = time()

history_cnn3 = cnn_model3.fit(X_train, y_train,
                    validation_data = (X_val, y_val),
                    epochs = 20, batch_size = 512,
                    validation_batch_size = 512,
                    callbacks=[cnn_checkpoint3])

time_to_run = time() - start
```

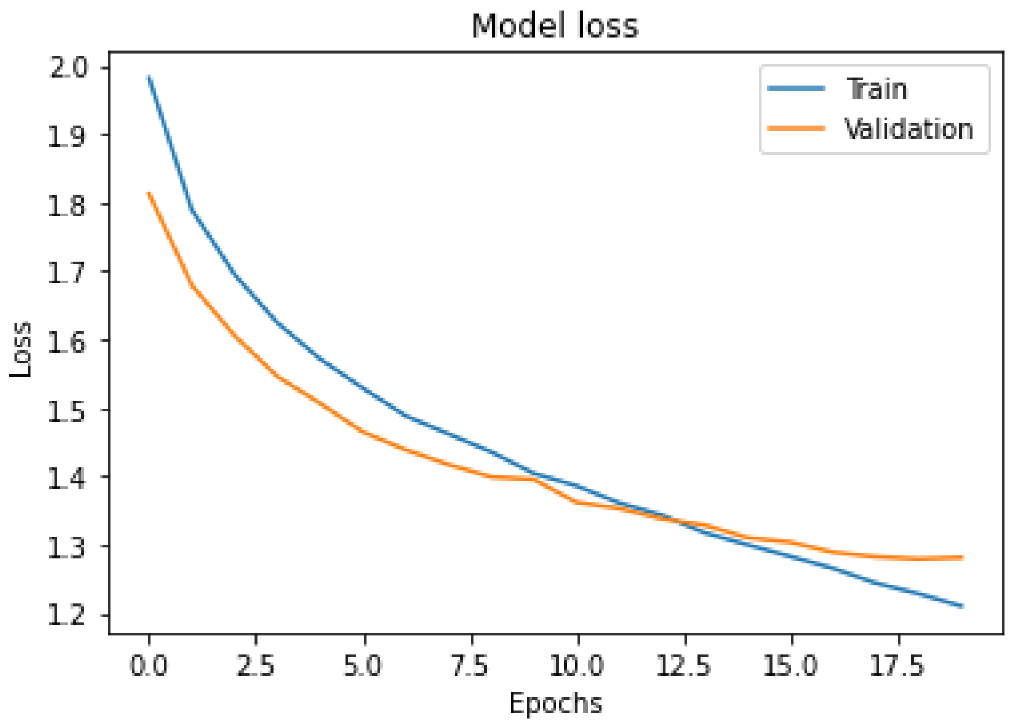
```
Epoch 1/20
56/57 [============================>.] - ETA: 0s - loss: 1.9816 - accuracy:
```

```
0.2802
Epoch 00001: val_loss improved from inf to 1.81245, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 13ms/step - loss: 1.9816 - accuracy:
0.2803 - val_loss: 1.8124 - val_accuracy: 0.3658
Epoch 2/20
55/57 [===========================>..] - ETA: 0s - loss: 1.7908 - accuracy:
0.3574
Epoch 00002: val_loss improved from 1.81245 to 1.67925, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.7891 - accuracy:
0.3581 - val_loss: 1.6792 - val_accuracy: 0.4009
Epoch 3/20
50/57 [=========================>...] - ETA: 0s - loss: 1.7001 - accuracy:
0.3847
Epoch 00003: val_loss improved from 1.67925 to 1.60565, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.6949 - accuracy:
0.3852 - val_loss: 1.6057 - val_accuracy: 0.4163
Epoch 4/20
57/57 [==============================] - ETA: 0s - loss: 1.6246 - accuracy:
0.4069
Epoch 00004: val_loss improved from 1.60565 to 1.54609, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.6246 - accuracy:
0.4069 - val_loss: 1.5461 - val_accuracy: 0.4363
Epoch 5/20
57/57 [==============================] - ETA: 0s - loss: 1.5717 - accuracy:
0.4266
Epoch 00005: val_loss improved from 1.54609 to 1.50678, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.5717 - accuracy:
0.4266 - val_loss: 1.5068 - val_accuracy: 0.4570
Epoch 6/20
50/57 [=========================>...] - ETA: 0s - loss: 1.5294 - accuracy:
0.4420
Epoch 00006: val_loss improved from 1.50678 to 1.46498, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.5291 - accuracy:
0.4409 - val_loss: 1.4650 - val_accuracy: 0.4639
Epoch 7/20
56/57 [============================>.] - ETA: 0s - loss: 1.4886 - accuracy:
0.4576
Epoch 00007: val_loss improved from 1.46498 to 1.43873, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.4884 - accuracy:
0.4577 - val_loss: 1.4387 - val_accuracy: 0.4820
Epoch 8/20
```

```
56/57 [============================>.] - ETA: 0s - loss: 1.4621 - accuracy:
0.4677
Epoch 00008: val_loss improved from 1.43873 to 1.41734, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.4621 - accuracy:
0.4676 - val_loss: 1.4173 - val_accuracy: 0.4756
Epoch 9/20
57/57 [==============================] - ETA: 0s - loss: 1.4358 - accuracy:
0.4759
Epoch 00009: val_loss improved from 1.41734 to 1.39918, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.4358 - accuracy:
0.4759 - val_loss: 1.3992 - val_accuracy: 0.4921
Epoch 10/20
56/57 [============================>.] - ETA: 0s - loss: 1.4046 - accuracy:
0.4881
Epoch 00010: val_loss improved from 1.39918 to 1.39554, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.4043 - accuracy:
0.4882 - val_loss: 1.3955 - val_accuracy: 0.4859
Epoch 11/20
57/57 [==============================] - ETA: 0s - loss: 1.3861 - accuracy:
0.4956
Epoch 00011: val_loss improved from 1.39554 to 1.36196, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.3861 - accuracy:
0.4956 - val_loss: 1.3620 - val_accuracy: 0.4976
Epoch 12/20
57/57 [==============================] - ETA: 0s - loss: 1.3611 - accuracy:
0.5041
Epoch 00012: val_loss improved from 1.36196 to 1.35316, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.3611 - accuracy:
0.5041 - val_loss: 1.3532 - val_accuracy: 0.5035
Epoch 13/20
57/57 [==============================] - ETA: 0s - loss: 1.3433 - accuracy:
0.5116
Epoch 00013: val_loss improved from 1.35316 to 1.33796, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.3433 - accuracy:
0.5116 - val_loss: 1.3380 - val_accuracy: 0.5071
Epoch 14/20
56/57 [============================>.] - ETA: 0s - loss: 1.3171 - accuracy:
0.5206
Epoch 00014: val_loss improved from 1.33796 to 1.32859, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.3174 - accuracy:
0.5205 - val_loss: 1.3286 - val_accuracy: 0.5046
```

```
Epoch 15/20
56/57 [============================>.] - ETA: 0s - loss: 1.3002 - accuracy:
0.5305
Epoch 00015: val_loss improved from 1.32859 to 1.31030, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.3000 - accuracy:
0.5305 - val_loss: 1.3103 - val_accuracy: 0.5247
Epoch 16/20
57/57 [==============================] - ETA: 0s - loss: 1.2830 - accuracy:
0.5368
Epoch 00016: val_loss improved from 1.31030 to 1.30378, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.2830 - accuracy:
0.5368 - val_loss: 1.3038 - val_accuracy: 0.5274
Epoch 17/20
50/57 [=========================>...] - ETA: 0s - loss: 1.2673 - accuracy:
0.5423
Epoch 00017: val_loss improved from 1.30378 to 1.28909, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.2656 - accuracy:
0.5428 - val_loss: 1.2891 - val_accuracy: 0.5274
Epoch 18/20
57/57 [==============================] - ETA: 0s - loss: 1.2440 - accuracy:
0.5547
Epoch 00018: val_loss improved from 1.28909 to 1.28253, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 10ms/step - loss: 1.2440 - accuracy:
0.5547 - val_loss: 1.2825 - val_accuracy: 0.5369
Epoch 19/20
57/57 [==============================] - ETA: 0s - loss: 1.2285 - accuracy:
0.5625
Epoch 00019: val_loss improved from 1.28253 to 1.27936, saving model to
best_cnn_model3.hdf5
57/57 [==============================] - 1s 9ms/step - loss: 1.2285 - accuracy:
0.5625 - val_loss: 1.2794 - val_accuracy: 0.5366
Epoch 20/20
56/57 [============================>.] - ETA: 0s - loss: 1.2105 - accuracy:
0.5698
Epoch 00020: val_loss did not improve from 1.27936
57/57 [==============================] - 1s 9ms/step - loss: 1.2105 - accuracy:
0.5696 - val_loss: 1.2813 - val_accuracy: 0.5339
```

[52]:
```
plt.plot(history_cnn3.history['loss'])
plt.plot(history_cnn3.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()

print(time_to_run)
```



```
11.894754886627197
```

d(i). Details of the 3 different CNN models are listed below:

#Details on CNN Model 1:

Number of conv layers: 3

Number of pooling layers: 2

Total parameters: 19798723

Optimizer used: Adam with learning rate of 0.0005

Regulerizer used: l2 regulerizer used with conv layers

Dropout: 0.4 after dense layers

Activation function for input and hidden layers: ReLU

Activation function for output layer: Softmax

Training accuracy (for the best model): 85.23%

Validation accuracy (for the best model): 46.45%

Training time: 83.49 s

#Details on CNN Model 2:

Number of conv layers: 2

Number of pooling layers: 1

Total parameters: 20003999

Optimizer used: Adam with learning rate of 0.0005

Regulerizer used: l2 regulerizer used with conv layers

Dropout: 0.4 after dense layers

Activation function for input and hidden layers: ReLU for conv layers & Sigmoid for dense layers

Activation function for output layer: Softmax

Training accuracy (for the best model): 48.21%

Validation accuracy (for the best model): 47.34%

Training time: 12.48 s

#Details on FNN Model 3:

Number of conv layers: 2

Number of pooling layers: 1

Total parameters: 615159

Optimizer used: Adam with learning rate of 0.0005

Regulerizer used: l2 regulerizer used with conv layers

Dropout: 0.4 after 2nd conv layers, 0.2 after dense layers

Activation function for input and hidden layers: ReLU

Activation function for output layer: Softmax

Training accuracy (for the best model): 56.25%

Validation accuracy (for the best model): 53.66%

Training time: 11.89 s

Analayzing the performance of the 3 models on train and validation sets, it can be concluded that the 3rd model performs well and we should use it for testing. One notable thing about the CNN models are they have higher number of parameters and runtime compared to the FNN model and their results are improved by roughly 10%.

```
[53]: best_cnn_model = tf.keras.models.load_model("best_cnn_model3.hdf5")
      score = best_cnn_model.evaluate(X_test, y_test, verbose=0)
      print("%s: %.2f%%" % (best_cnn_model.metrics_names[1], score[1]*100))
```

```
accuracy: 54.00%
```

d(ii). Emotion classification accuracy on the testing set = 54.00%

```
[54]: !pip install -q -U keras-tuner
```

```
     || 98 kB 5.1 MB/s eta 0:00:011
```

```
[55]: def build_model(hp):

          model = keras.Sequential([

          keras.layers.Conv2D(
              filters = hp.Int('conv_1_filter', min_value=32, max_value=128, step=16),
              kernel_size = hp.Choice('conv_1_kernel', values = [3,5]),
              activation = hp.Choice('conv_1_activition', values = ['relu','sigmoid']),
              input_shape = (48,48,1),
              padding = 'same'),

          keras.layers.MaxPooling2D(
              pool_size = hp.Choice('pool_1_kernel', values = [2,4]),
              strides = (2,2),
              padding ='same'),

          keras.layers.Conv2D(
              filters = hp.Int('conv_2_filter', min_value=32, max_value=64, step=16),
              kernel_size = hp.Choice('conv_2_kernel', values = [3,5]),
              activation = 'relu',
              padding = 'same'),

          keras.layers.MaxPooling2D(
              pool_size = hp.Choice('pool_2_kernel', values = [2,4]),
              strides = (2,2),
              padding = 'same'),

          keras.layers.Flatten(),

          keras.layers.Dense(
              units = hp.Int('dense_1_units', min_value=64, max_value=512, step=16),
              activation = hp.Choice('dense_1_activition', values =␣
       ↪['relu','sigmoid'])),

          keras.layers.Dense(7, activation='softmax')
```

```
    ])


    model.compile(optimizer=tf.keras.optimizers.Adam(hp.Choice('learning_rate',␣
 ↪values=[1e-2, 1e-3])),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

[56]:
```
#importing random search
from keras_tuner import BayesianOptimization
#creating randomsearch object
tuner = BayesianOptimization(build_model,
                    objective='val_loss',
                    max_trials = 5,
                    overwrite = True)
# search best parameter
tuner.search(X_train, y_train, epochs=10, batch_size = 512,␣
 ↪validation_data=(X_val, y_val))
```

```
Trial 5 Complete [00h 00m 28s]
val_loss: 1.2554805278778076

Best val_loss So Far: 1.2032557725906372
Total elapsed time: 00h 01m 47s
INFO:tensorflow:Oracle triggered exit
```

[57]:
```
model = tuner.get_best_models(num_models=1)[0]
print (model.summary())
# Evaluate the best model.
loss, accuracy = model.evaluate(X_test, y_test)
print('loss:', loss)
print('accuracy:', accuracy*100)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        832

 max_pooling2d (MaxPooling2D  (None, 24, 24, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 64)       0
 2D)
 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 64)                589888

 dense_1 (Dense)             (None, 7)                 455

=================================================================
Total params: 609,671
Trainable params: 609,671
Non-trainable params: 0
_________________________________________________________________
None
113/113 [==============================] - 1s 3ms/step - loss: 1.1790 -
accuracy: 0.5573
loss: 1.1789838075637817
accuracy: 55.72583079338074
```

e. Emotion classification accuracy on the testing set = 55.73%

[149]:
```
from tensorflow.keras.applications import VGG16
vgg_model = VGG16(weights = 'imagenet', include_top = False, input_shape = (48,␣
 ↪48, 3))
```

[151]:
```
# Freeze bottom layers
for layer in vgg_model.layers[:17]:
    layer.trainable = False

for i, layer in enumerate(vgg_model.layers):
    print(i, layer.name, layer.trainable)
```

```
0 input_10 False
1 block1_conv1 False
2 block1_conv2 False
3 block1_pool False
4 block2_conv1 False
5 block2_conv2 False
6 block2_pool False
7 block3_conv1 False
8 block3_conv2 False
9 block3_conv3 False
10 block3_pool False
11 block4_conv1 False
12 block4_conv2 False
13 block4_conv3 False
14 block4_pool False
15 block5_conv1 False
16 block5_conv2 False
17 block5_conv3 True
```

```
18 block5_pool True
```

[152]:
```
x = vgg_model.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(7, activation='softmax')(x)

transfer_model = tf.keras.Model(inputs = vgg_model.input, outputs = x)
```

[154]:
```
checkpoint3 = ModelCheckpoint('vgg16_finetune.h15',
                              monitor= 'val_loss',
                              mode= 'min',
                              save_best_only = True,
                              verbose= 1)
```

[155]:
```
from tensorflow.keras.optimizers import Adam
learning_rate= 0.005
transfer_model.compile(loss="categorical_crossentropy",␣
 ↪optimizer=Adam(lr=learning_rate), metrics=["accuracy"])
```

[148]:
```
X_train_trn = np.stack((X_train, X_train, X_train), axis = 3)
X_test_trn = np.stack((X_test, X_test, X_test), axis = 3)
X_val_trn = np.stack((X_val, X_val, X_val), axis = 3)
```

[156]:
```
transfer_history = transfer_model.fit(
    X_train_trn, y_train,
    batch_size = 512,
    epochs=20,
    validation_data=(X_val_trn, y_val),
    callbacks = [checkpoint3])
```

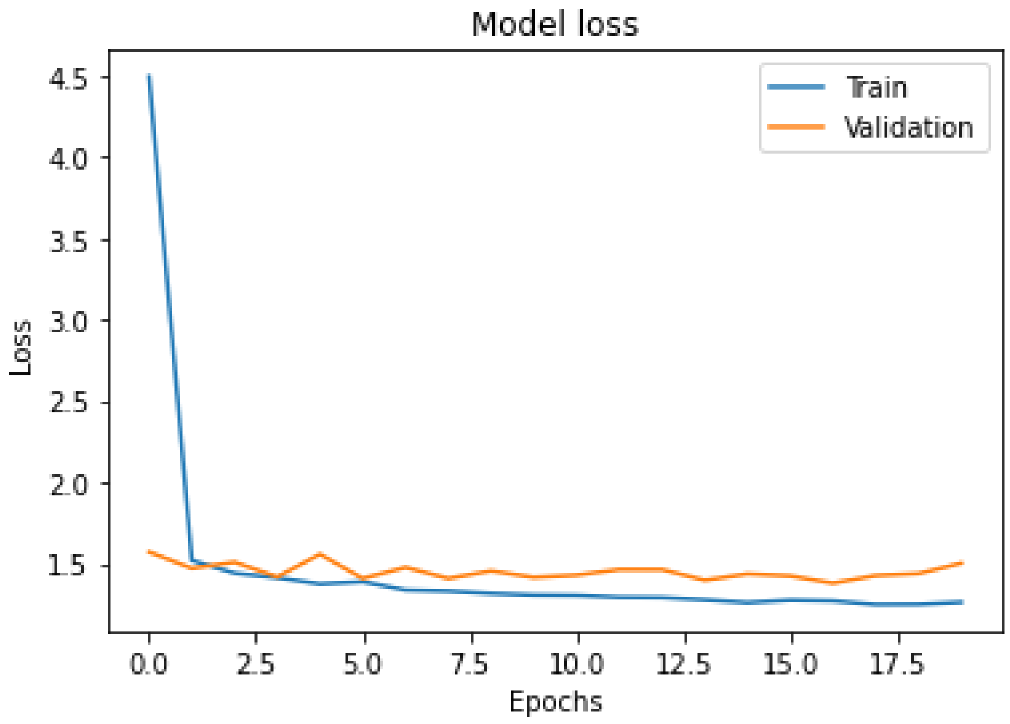
```
Epoch 1/20
56/57 [============================>.] - ETA: 0s - loss: 4.4964 - accuracy:
0.2909
Epoch 00001: val_loss improved from inf to 1.57159, saving model to
vgg16_finetune.h15
INFO:tensorflow:Assets written to: vgg16_finetune.h15/assets
57/57 [==============================] - 38s 667ms/step - loss: 4.4926 -
accuracy: 0.2910 - val_loss: 1.5716 - val_accuracy: 0.3756
Epoch 2/20
56/57 [============================>.] - ETA: 0s - loss: 1.5201 - accuracy:
0.4092
Epoch 00002: val_loss improved from 1.57159 to 1.47074, saving model to
vgg16_finetune.h15
INFO:tensorflow:Assets written to: vgg16_finetune.h15/assets
57/57 [==============================] - 6s 105ms/step - loss: 1.5196 -
accuracy: 0.4094 - val_loss: 1.4707 - val_accuracy: 0.4416
```

```
Epoch 3/20
56/57 [============================>.] - ETA: 0s - loss: 1.4410 - accuracy:
0.4490
Epoch 00003: val_loss did not improve from 1.47074
57/57 [==============================] - 4s 68ms/step - loss: 1.4408 - accuracy:
0.4490 - val_loss: 1.5088 - val_accuracy: 0.4269
Epoch 4/20
56/57 [============================>.] - ETA: 0s - loss: 1.4129 - accuracy:
0.4651
Epoch 00004: val_loss improved from 1.47074 to 1.41811, saving model to
vgg16_finetune.h15
INFO:tensorflow:Assets written to: vgg16_finetune.h15/assets
57/57 [==============================] - 6s 105ms/step - loss: 1.4129 -
accuracy: 0.4649 - val_loss: 1.4181 - val_accuracy: 0.4561
Epoch 5/20
56/57 [============================>.] - ETA: 0s - loss: 1.3745 - accuracy:
0.4818
Epoch 00005: val_loss did not improve from 1.41811
57/57 [==============================] - 4s 68ms/step - loss: 1.3747 - accuracy:
0.4817 - val_loss: 1.5577 - val_accuracy: 0.3965
Epoch 6/20
56/57 [============================>.] - ETA: 0s - loss: 1.3882 - accuracy:
0.4760
Epoch 00006: val_loss improved from 1.41811 to 1.40841, saving model to
vgg16_finetune.h15
INFO:tensorflow:Assets written to: vgg16_finetune.h15/assets
57/57 [==============================] - 6s 110ms/step - loss: 1.3881 -
accuracy: 0.4762 - val_loss: 1.4084 - val_accuracy: 0.4617
Epoch 7/20
56/57 [============================>.] - ETA: 0s - loss: 1.3371 - accuracy:
0.4948
Epoch 00007: val_loss did not improve from 1.40841
57/57 [==============================] - 4s 68ms/step - loss: 1.3370 - accuracy:
0.4949 - val_loss: 1.4766 - val_accuracy: 0.4466
Epoch 8/20
56/57 [============================>.] - ETA: 0s - loss: 1.3305 - accuracy:
0.4985
Epoch 00008: val_loss did not improve from 1.40841
57/57 [==============================] - 4s 67ms/step - loss: 1.3306 - accuracy:
0.4985 - val_loss: 1.4085 - val_accuracy: 0.4597
Epoch 9/20
56/57 [============================>.] - ETA: 0s - loss: 1.3161 - accuracy:
0.5055
Epoch 00009: val_loss did not improve from 1.40841
57/57 [==============================] - 4s 68ms/step - loss: 1.3161 - accuracy:
0.5053 - val_loss: 1.4546 - val_accuracy: 0.4380
Epoch 10/20
56/57 [============================>.] - ETA: 0s - loss: 1.3068 - accuracy:
```

```
0.5071
Epoch 00010: val_loss did not improve from 1.40841
57/57 [==============================] - 4s 68ms/step - loss: 1.3062 - accuracy:
0.5073 - val_loss: 1.4152 - val_accuracy: 0.4734
Epoch 11/20
56/57 [============================>.] - ETA: 0s - loss: 1.3039 - accuracy:
0.5108
Epoch 00011: val_loss did not improve from 1.40841
57/57 [==============================] - 4s 68ms/step - loss: 1.3040 - accuracy:
0.5108 - val_loss: 1.4287 - val_accuracy: 0.4556
Epoch 12/20
56/57 [============================>.] - ETA: 0s - loss: 1.2923 - accuracy:
0.5149
Epoch 00012: val_loss did not improve from 1.40841
57/57 [==============================] - 4s 68ms/step - loss: 1.2924 - accuracy:
0.5149 - val_loss: 1.4634 - val_accuracy: 0.4427
Epoch 13/20
56/57 [============================>.] - ETA: 0s - loss: 1.2908 - accuracy:
0.5166
Epoch 00013: val_loss did not improve from 1.40841
57/57 [==============================] - 4s 68ms/step - loss: 1.2908 - accuracy:
0.5166 - val_loss: 1.4630 - val_accuracy: 0.4508
Epoch 14/20
56/57 [============================>.] - ETA: 0s - loss: 1.2772 - accuracy:
0.5227
Epoch 00014: val_loss improved from 1.40841 to 1.39755, saving model to
vgg16_finetune.h15
INFO:tensorflow:Assets written to: vgg16_finetune.h15/assets
57/57 [==============================] - 6s 105ms/step - loss: 1.2773 -
accuracy: 0.5228 - val_loss: 1.3975 - val_accuracy: 0.4706
Epoch 15/20
56/57 [============================>.] - ETA: 0s - loss: 1.2601 - accuracy:
0.5300
Epoch 00015: val_loss did not improve from 1.39755
57/57 [==============================] - 4s 68ms/step - loss: 1.2602 - accuracy:
0.5299 - val_loss: 1.4362 - val_accuracy: 0.4542
Epoch 16/20
56/57 [============================>.] - ETA: 0s - loss: 1.2756 - accuracy:
0.5264
Epoch 00016: val_loss did not improve from 1.39755
57/57 [==============================] - 4s 68ms/step - loss: 1.2755 - accuracy:
0.5265 - val_loss: 1.4241 - val_accuracy: 0.4544
Epoch 17/20
56/57 [============================>.] - ETA: 0s - loss: 1.2711 - accuracy:
0.5254
Epoch 00017: val_loss improved from 1.39755 to 1.37925, saving model to
vgg16_finetune.h15
INFO:tensorflow:Assets written to: vgg16_finetune.h15/assets
```

```
57/57 [==============================] - 6s 112ms/step - loss: 1.2708 -
accuracy: 0.5255 - val_loss: 1.3792 - val_accuracy: 0.4806
Epoch 18/20
56/57 [============================>.] - ETA: 0s - loss: 1.2476 - accuracy:
0.5375
Epoch 00018: val_loss did not improve from 1.37925
57/57 [==============================] - 4s 68ms/step - loss: 1.2478 - accuracy:
0.5375 - val_loss: 1.4262 - val_accuracy: 0.4653
Epoch 19/20
56/57 [============================>.] - ETA: 0s - loss: 1.2486 - accuracy:
0.5344
Epoch 00019: val_loss did not improve from 1.37925
57/57 [==============================] - 4s 68ms/step - loss: 1.2490 - accuracy:
0.5342 - val_loss: 1.4384 - val_accuracy: 0.4542
Epoch 20/20
56/57 [============================>.] - ETA: 0s - loss: 1.2628 - accuracy:
0.5299
Epoch 00020: val_loss did not improve from 1.37925
57/57 [==============================] - 4s 68ms/step - loss: 1.2626 - accuracy:
0.5300 - val_loss: 1.5022 - val_accuracy: 0.4363
```

[157]:
```
plt.plot(transfer_history.history['loss'])
plt.plot(transfer_history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



[158]:
```
best_trn_model = tf.keras.models.load_model("vgg16_finetune.h15")
score = best_trn_model.evaluate(X_test_trn, y_test, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))
```

```
accuracy: 48.59%
```

(f). I have used VGG-16 pretrained model and fine-tuned it with the training data. I started with pretrained imagenet weights for training and froze all the conv blocks except the last conv and pool layers and fine tuned the weights for those layers and the dense layers at the top. The accuracy of the best model is 48.59% on the test set.

[159]:
```
X_train_aug = np.expand_dims(X_train, axis=-1)
X_val_aug = np.expand_dims(X_val, axis=-1)
X_test_aug = np.expand_dims(X_test, axis=-1)
```

[204]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    fill_mode='nearest',
    rotation_range=20,
    shear_range=20,
    zoom_range=0.1,
    horizontal_flip = True)

valid_datagen = ImageDataGenerator()
```

```
train_generator = train_datagen.flow(
    X_train_aug, y_train,
    batch_size = 512)

validation_generator = valid_datagen.flow(
    X_val_aug, y_val,
    batch_size = 256)
```

[205]:
```
n_rows=3
n_cols=5
n_images = n_rows * n_cols

plt.figure(figsize=(n_cols*4, n_rows*3))
for image_index in range(n_images):
    image, label = next(train_generator)
    img = image[0].squeeze()
    plt.subplot(n_rows, n_cols, image_index+1)
    plt.axis('off')
    plt.imshow(img, cmap = 'gray')
```

[ ]:
```
checkpoint_4 = ModelCheckpoint("best_cnn_aug_model.hdf5", monitor = 'val_loss',␣
 ↪verbose = 1,
    save_best_only = True, mode =' min', period = 1)
```

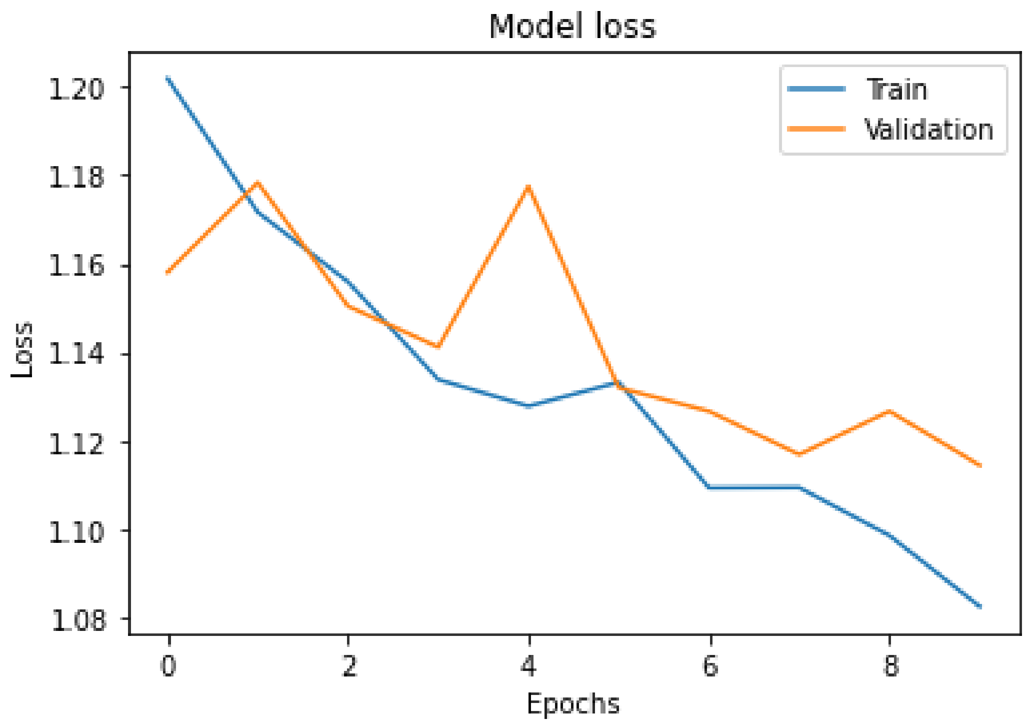
[207]:
```
history = model.fit_generator(train_generator,
                              validation_data=validation_generator,
                              epochs = 10,
                              callbacks=[checkpoint_4])
```

```
Epoch 1/10
57/57 [==============================] - ETA: 0s - loss: 1.2019 - accuracy:
0.5464
Epoch 00001: val_loss improved from inf to 1.15808, saving model to
best_cnn_aug_model.hdf5
57/57 [==============================] - 8s 141ms/step - loss: 1.2019 -
accuracy: 0.5464 - val_loss: 1.1581 - val_accuracy: 0.5620
Epoch 2/10
57/57 [==============================] - ETA: 0s - loss: 1.1716 - accuracy:
0.5588
Epoch 00002: val_loss did not improve from 1.15808
57/57 [==============================] - 8s 135ms/step - loss: 1.1716 -
accuracy: 0.5588 - val_loss: 1.1784 - val_accuracy: 0.5556
Epoch 3/10
57/57 [==============================] - ETA: 0s - loss: 1.1559 - accuracy:
0.5671
Epoch 00003: val_loss improved from 1.15808 to 1.15041, saving model to
best_cnn_aug_model.hdf5
57/57 [==============================] - 8s 139ms/step - loss: 1.1559 -
accuracy: 0.5671 - val_loss: 1.1504 - val_accuracy: 0.5662
Epoch 4/10
57/57 [==============================] - ETA: 0s - loss: 1.1338 - accuracy:
0.5732
Epoch 00004: val_loss improved from 1.15041 to 1.14113, saving model to
best_cnn_aug_model.hdf5
57/57 [==============================] - 8s 137ms/step - loss: 1.1338 -
accuracy: 0.5732 - val_loss: 1.1411 - val_accuracy: 0.5717
Epoch 5/10
57/57 [==============================] - ETA: 0s - loss: 1.1277 - accuracy:
0.5782
Epoch 00005: val_loss did not improve from 1.14113
57/57 [==============================] - 8s 136ms/step - loss: 1.1277 -
accuracy: 0.5782 - val_loss: 1.1775 - val_accuracy: 0.5637
Epoch 6/10
57/57 [==============================] - ETA: 0s - loss: 1.1332 - accuracy:
0.5722
Epoch 00006: val_loss improved from 1.14113 to 1.13193, saving model to
best_cnn_aug_model.hdf5
57/57 [==============================] - 8s 136ms/step - loss: 1.1332 -
accuracy: 0.5722 - val_loss: 1.1319 - val_accuracy: 0.5759
Epoch 7/10
57/57 [==============================] - ETA: 0s - loss: 1.1093 - accuracy:
0.5811
Epoch 00007: val_loss improved from 1.13193 to 1.12660, saving model to
best_cnn_aug_model.hdf5
57/57 [==============================] - 8s 138ms/step - loss: 1.1093 -
accuracy: 0.5811 - val_loss: 1.1266 - val_accuracy: 0.5748
Epoch 8/10
```

```
57/57 [==============================] - ETA: 0s - loss: 1.1094 - accuracy:
0.5833
Epoch 00008: val_loss improved from 1.12660 to 1.11680, saving model to
best_cnn_aug_model.hdf5
57/57 [==============================] - 8s 138ms/step - loss: 1.1094 -
accuracy: 0.5833 - val_loss: 1.1168 - val_accuracy: 0.5768
Epoch 9/10
57/57 [==============================] - ETA: 0s - loss: 1.0985 - accuracy:
0.5890
Epoch 00009: val_loss did not improve from 1.11680
57/57 [==============================] - 8s 137ms/step - loss: 1.0985 -
accuracy: 0.5890 - val_loss: 1.1266 - val_accuracy: 0.5823
Epoch 10/10
57/57 [==============================] - ETA: 0s - loss: 1.0824 - accuracy:
0.5934
Epoch 00010: val_loss improved from 1.11680 to 1.11436, saving model to
best_cnn_aug_model.hdf5
57/57 [==============================] - 8s 137ms/step - loss: 1.0824 -
accuracy: 0.5934 - val_loss: 1.1144 - val_accuracy: 0.5885
```

[208]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



[209]:
```
model = tf.keras.models.load_model("best_cnn_aug_model.hdf5")
score = model.evaluate(X_test, y_test, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))
```

```
accuracy: 58.26%
```

g. I have augmented the model using image data generator. For data augmentation, I have used rotation, shear, zoom and horizontal flip. Some of the augmented images are also displayed in a cell above. I have also trained a cnn model (model obtained from previous Bayesian optimization step) using augmented data. The accuracy on the test set is 58.26%.