fer2013

April 27, 2023

[1]:
```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle.json
```

[2]:
```
!kaggle datasets download -d msambare/fer2013
```

```
Downloading fer2013.zip to /content
 76% 46.0M/60.3M [00:00<00:00, 94.9MB/s]
100% 60.3M/60.3M [00:00<00:00, 114MB/s]
```

[ ]:
```
!unzip fer2013.zip
```

[19]:
```
import os
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.layers import Conv2D, MaxPool2D, Dropout, BatchNormalization, Dense,␣
  ↪Flatten, GlobalAveragePooling2D
from keras.models import Sequential
from keras.applications.vgg16 import VGG16
from keras.applications.efficientnet import EfficientNetB4
from tensorflow.keras.applications.resnet50 import ResNet50
import xgboost as xgb
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
  ↪confusion_matrix
```

[5]:
```
def input_data(folder_path, output_data):
  for dirs in os.listdir(folder_path):
    class_name = dirs
    new_path = os.path.join(folder_path, class_name)
    for img in os.listdir(new_path):
      img_arr = cv.imread(os.path.join(new_path, img))
      output_data.append([img_arr, class_name])
  return output_data
```

[6]:
```
train_data = input_data("/content/train", [])
```

[7]:
```
test_data = input_data("/content/test", [])
```

[8]:
```
np.random.shuffle(train_data)
np.random.shuffle(test_data)
```

[9]:
```
train_images = []
train_labels = []
for features, labels in train_data:
  train_images.append(features)
  train_labels.append(labels)
```

[10]:
```
test_images = []
test_labels = []
for features, labels in test_data:
  test_images.append(features)
  test_labels.append(labels)
```

[11]:
```
label_enc = LabelEncoder()
train_labels = label_enc.fit_transform(train_labels)
test_labels = label_enc.transform(test_labels)
```

[12]:
```
train_images = np.array(train_images)
train_labels = np.array(train_labels)
test_images = np.array(test_images)
test_labels = np.array(test_labels)
```

[13]:
```
print(f"Shape of the train images {train_images.shape}")
print(f"Shape of the train labels {train_labels.shape}")
print(f"Shape of the test images {test_images.shape}")
print(f"Shape of the test labels {test_labels.shape}")
```

```
Shape of the train images (28709, 48, 48, 3)
Shape of the train labels (28709,)
Shape of the test images (7178, 48, 48, 3)
Shape of the test labels (7178,)
```

[14]:
```python
train_images = train_images/255
test_images = test_images/255
```

[17]:
```python
plt.figure(figsize=(15,10))
for i in range(25):
  plt.subplot(5, 5, i+1)
  plt.imshow(test_images[i])
  plt.title(f"{label_enc.inverse_transform([test_labels[i]])}")
  plt.axis("off")
```

CNN Model

[26]:
```python
model1 = Sequential()
```

[27]:
```python
model1.add(Conv2D(32, (3, 3), input_shape=(48,48,3), activation="leaky_relu"))
model1.add(MaxPool2D(2,2))
model1.add(Conv2D(64, (3, 3), activation="leaky_relu"))
model1.add(MaxPool2D(2,2))
model1.add(Conv2D(128, (3, 3), activation="leaky_relu"))
model1.add(MaxPool2D(2,2))
model1.add(Conv2D(256, (3, 3), activation="leaky_relu"))
model1.add(MaxPool2D(2,2))
```

```
model1.add(Flatten())
model1.add(Dense(512, activation="relu"))
model1.add(Dense(7, activation="softmax"))
```

[54]:
```
model1.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 46, 46, 32)        896

 max_pooling2d_8 (MaxPooling  (None, 23, 23, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 21, 21, 64)        18496

 max_pooling2d_9 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 conv2d_11 (Conv2D)          (None, 8, 8, 128)         73856

 max_pooling2d_10 (MaxPoolin  (None, 4, 4, 128)        0
 g2D)

 conv2d_12 (Conv2D)          (None, 2, 2, 256)         295168

 max_pooling2d_11 (MaxPoolin  (None, 1, 1, 256)        0
 g2D)

 flatten_2 (Flatten)         (None, 256)               0

 dense_2 (Dense)             (None, 512)               131584

 dense_3 (Dense)             (None, 7)                 3591

=================================================================
Total params: 523,591
Trainable params: 523,591
Non-trainable params: 0
_________________________________________________________________
```

[28]:
```
model1.compile(optimizer="adam", loss="sparse_categorical_crossentropy",␣
↪metrics = ['accuracy'])
```

[29]:
```
history1 = model1.fit(train_images, train_labels, validation_data=(test_images,␣
↪test_labels), epochs=30)
```

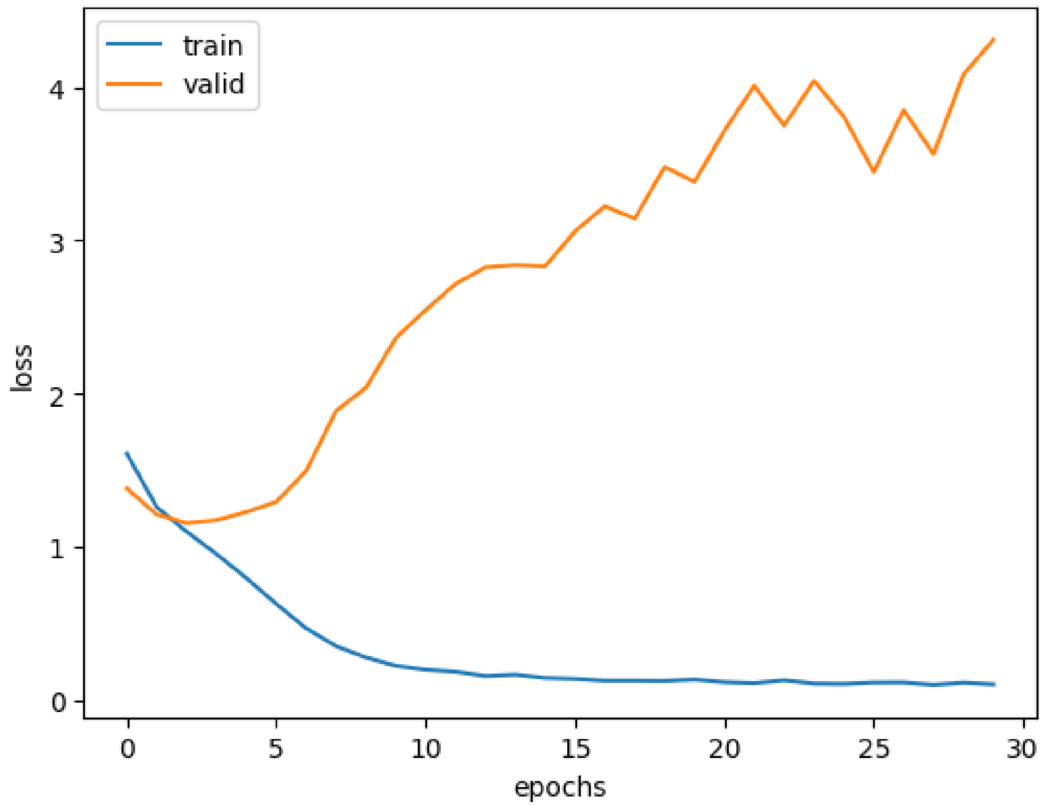
```
Epoch 1/30
898/898 [==============================] - 9s 8ms/step - loss: 1.6083 -
accuracy: 0.3585 - val_loss: 1.3817 - val_accuracy: 0.4709
Epoch 2/30
898/898 [==============================] - 5s 6ms/step - loss: 1.2594 -
accuracy: 0.5186 - val_loss: 1.2106 - val_accuracy: 0.5404
Epoch 3/30
898/898 [==============================] - 6s 6ms/step - loss: 1.0990 -
accuracy: 0.5833 - val_loss: 1.1550 - val_accuracy: 0.5626
Epoch 4/30
898/898 [==============================] - 5s 6ms/step - loss: 0.9530 -
accuracy: 0.6392 - val_loss: 1.1733 - val_accuracy: 0.5639
Epoch 5/30
898/898 [==============================] - 6s 6ms/step - loss: 0.7959 -
accuracy: 0.7029 - val_loss: 1.2274 - val_accuracy: 0.5766
Epoch 6/30
898/898 [==============================] - 5s 6ms/step - loss: 0.6272 -
accuracy: 0.7686 - val_loss: 1.2931 - val_accuracy: 0.5812
Epoch 7/30
898/898 [==============================] - 6s 6ms/step - loss: 0.4681 -
accuracy: 0.8288 - val_loss: 1.4943 - val_accuracy: 0.5724
Epoch 8/30
898/898 [==============================] - 5s 6ms/step - loss: 0.3515 -
accuracy: 0.8708 - val_loss: 1.8871 - val_accuracy: 0.5659
Epoch 9/30
898/898 [==============================] - 6s 7ms/step - loss: 0.2775 -
accuracy: 0.9010 - val_loss: 2.0380 - val_accuracy: 0.5607
Epoch 10/30
898/898 [==============================] - 5s 6ms/step - loss: 0.2231 -
accuracy: 0.9215 - val_loss: 2.3632 - val_accuracy: 0.5702
Epoch 11/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1971 -
accuracy: 0.9298 - val_loss: 2.5455 - val_accuracy: 0.5652
Epoch 12/30
898/898 [==============================] - 6s 6ms/step - loss: 0.1841 -
accuracy: 0.9368 - val_loss: 2.7169 - val_accuracy: 0.5681
Epoch 13/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1562 -
accuracy: 0.9466 - val_loss: 2.8244 - val_accuracy: 0.5571
Epoch 14/30
898/898 [==============================] - 6s 6ms/step - loss: 0.1648 -
accuracy: 0.9453 - val_loss: 2.8391 - val_accuracy: 0.5634
Epoch 15/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1431 -
accuracy: 0.9527 - val_loss: 2.8315 - val_accuracy: 0.5571
Epoch 16/30
898/898 [==============================] - 6s 6ms/step - loss: 0.1375 -
accuracy: 0.9539 - val_loss: 3.0607 - val_accuracy: 0.5617
```

```
Epoch 17/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1251 -
accuracy: 0.9598 - val_loss: 3.2240 - val_accuracy: 0.5666
Epoch 18/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1252 -
accuracy: 0.9590 - val_loss: 3.1433 - val_accuracy: 0.5723
Epoch 19/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1236 -
accuracy: 0.9600 - val_loss: 3.4812 - val_accuracy: 0.5736
Epoch 20/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1328 -
accuracy: 0.9571 - val_loss: 3.3832 - val_accuracy: 0.5729
Epoch 21/30
898/898 [==============================] - 6s 6ms/step - loss: 0.1155 -
accuracy: 0.9627 - val_loss: 3.7177 - val_accuracy: 0.5585
Epoch 22/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1088 -
accuracy: 0.9640 - val_loss: 4.0101 - val_accuracy: 0.5651
Epoch 23/30
898/898 [==============================] - 6s 6ms/step - loss: 0.1280 -
accuracy: 0.9592 - val_loss: 3.7505 - val_accuracy: 0.5665
Epoch 24/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1065 -
accuracy: 0.9661 - val_loss: 4.0427 - val_accuracy: 0.5705
Epoch 25/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1044 -
accuracy: 0.9679 - val_loss: 3.8090 - val_accuracy: 0.5649
Epoch 26/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1129 -
accuracy: 0.9628 - val_loss: 3.4479 - val_accuracy: 0.5617
Epoch 27/30
898/898 [==============================] - 5s 6ms/step - loss: 0.1135 -
accuracy: 0.9647 - val_loss: 3.8520 - val_accuracy: 0.5663
Epoch 28/30
898/898 [==============================] - 6s 6ms/step - loss: 0.0980 -
accuracy: 0.9698 - val_loss: 3.5648 - val_accuracy: 0.5634
Epoch 29/30
898/898 [==============================] - 8s 8ms/step - loss: 0.1123 -
accuracy: 0.9656 - val_loss: 4.0841 - val_accuracy: 0.5667
Epoch 30/30
898/898 [==============================] - 9s 10ms/step - loss: 0.1017 -
accuracy: 0.9682 - val_loss: 4.3113 - val_accuracy: 0.5715
```

[31]:
```
plt.plot(history1.history["loss"])
plt.plot(history1.history["val_loss"])
plt.xlabel("epochs")
plt.ylabel("loss")
```

```
plt.legend(["train","valid"])
```

[31]: <matplotlib.legend.Legend at 0x7fc4b88452e0>



[32]:
```
y_pred1 = model1.predict(test_images)
```

```
225/225 [==============================] - 1s 2ms/step
```

[33]:
```
print(classification_report(y_pred1.argmax(axis=1), test_labels))
```

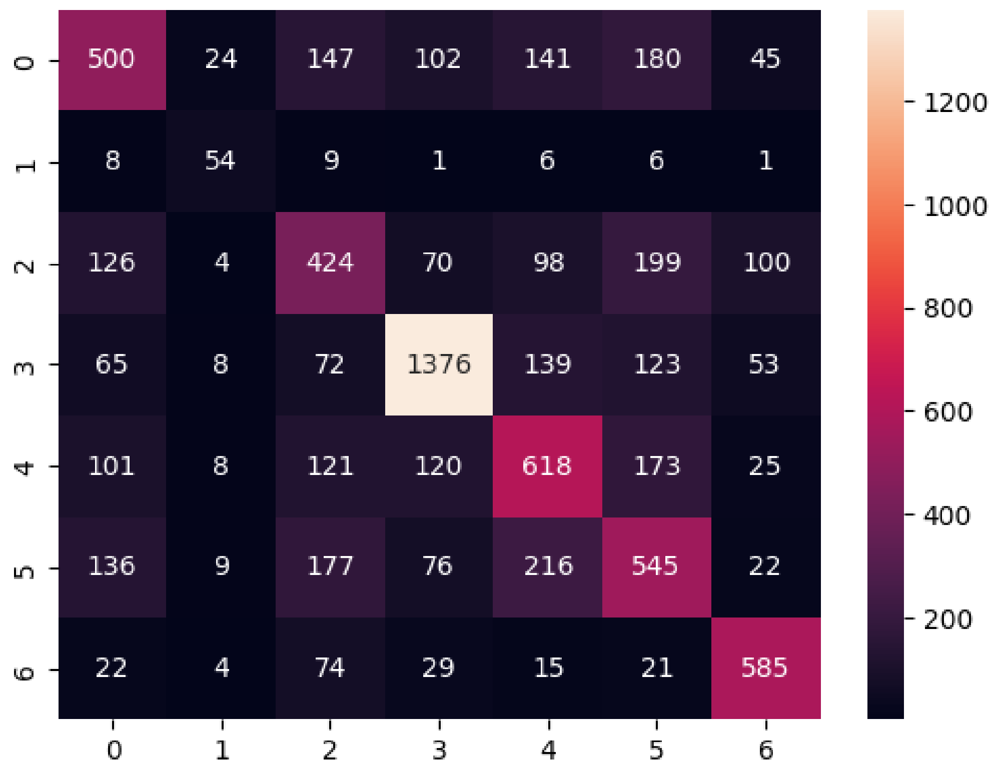
```
              precision    recall  f1-score   support

           0       0.52      0.44      0.48      1139
           1       0.49      0.64      0.55        85
           2       0.41      0.42      0.41      1021
           3       0.78      0.75      0.76      1836
           4       0.50      0.53      0.52      1166
           5       0.44      0.46      0.45      1181
           6       0.70      0.78      0.74       750

    accuracy                           0.57      7178
   macro avg       0.55      0.57      0.56      7178
```

```
weighted avg       0.57      0.57      0.57      7178
```

[37]:
```
sns.heatmap(confusion_matrix(y_pred1.argmax(axis=1), test_labels), fmt='g',␣
↪annot=True)
plt.show()
```



CNN Model with Dropout, Global Average Pooling2D and BatchNormalization layer

[38]:
```
model2 = Sequential()
```

[39]:
```
model2.add(Conv2D(32, (3, 3), input_shape=(48,48,3), activation="leaky_relu"))
model2.add(BatchNormalization())
model2.add(Dropout(0.1))
model2.add(MaxPool2D(2,2))
model2.add(Conv2D(64, (3, 3), activation="leaky_relu"))
model2.add(BatchNormalization())
model2.add(Dropout(0.2))
model2.add(MaxPool2D(2,2))
model2.add(Conv2D(128, (3, 3), activation="leaky_relu"))
model2.add(BatchNormalization())
model2.add(Dropout(0.3))
```

```
model2.add(MaxPool2D(2,2))
model2.add(Conv2D(256, (3, 3), activation="leaky_relu"))
model2.add(MaxPool2D(2,2))
model2.add(BatchNormalization())
model2.add(GlobalAveragePooling2D())
model2.add(Dense(512, activation="relu"))
model2.add(BatchNormalization())
model2.add(Dense(7, activation="softmax"))
```

[55]:
```
model2.summary()
```

```
Model: "sequential_3"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_13 (Conv2D)          (None, 46, 46, 32)        896

 batch_normalization (BatchN  (None, 46, 46, 32)       128
 ormalization)

 dropout (Dropout)           (None, 46, 46, 32)        0

 max_pooling2d_12 (MaxPoolin  (None, 23, 23, 32)       0
 g2D)

 conv2d_14 (Conv2D)          (None, 21, 21, 64)        18496

 batch_normalization_1 (Batc  (None, 21, 21, 64)       256
 hNormalization)

 dropout_1 (Dropout)         (None, 21, 21, 64)        0

 max_pooling2d_13 (MaxPoolin  (None, 10, 10, 64)       0
 g2D)

 conv2d_15 (Conv2D)          (None, 8, 8, 128)         73856

 batch_normalization_2 (Batc  (None, 8, 8, 128)        512
 hNormalization)

 dropout_2 (Dropout)         (None, 8, 8, 128)         0

 max_pooling2d_14 (MaxPoolin  (None, 4, 4, 128)        0
 g2D)

 conv2d_16 (Conv2D)          (None, 2, 2, 256)         295168
```

```
 max_pooling2d_15 (MaxPoolin  (None, 1, 1, 256)        0
 g2D)

 batch_normalization_3 (Batc  (None, 1, 1, 256)        1024
 hNormalization)

 global_average_pooling2d (G  (None, 256)              0
 lobalAveragePooling2D)

 dense_4 (Dense)             (None, 512)               131584

 batch_normalization_4 (Batc  (None, 512)              2048
 hNormalization)

 dense_5 (Dense)             (None, 7)                 3591

=================================================================
Total params: 527,559
Trainable params: 525,575
Non-trainable params: 1,984
_________________________________________________________________
```

[40]:
```
model2.compile(optimizer='adam', loss="sparse_categorical_crossentropy",
 ↪metrics=['accuracy'])
```

[41]:
```
history2 = model2.fit(train_images, train_labels, validation_data=(test_images,
 ↪test_labels), epochs=30)
```

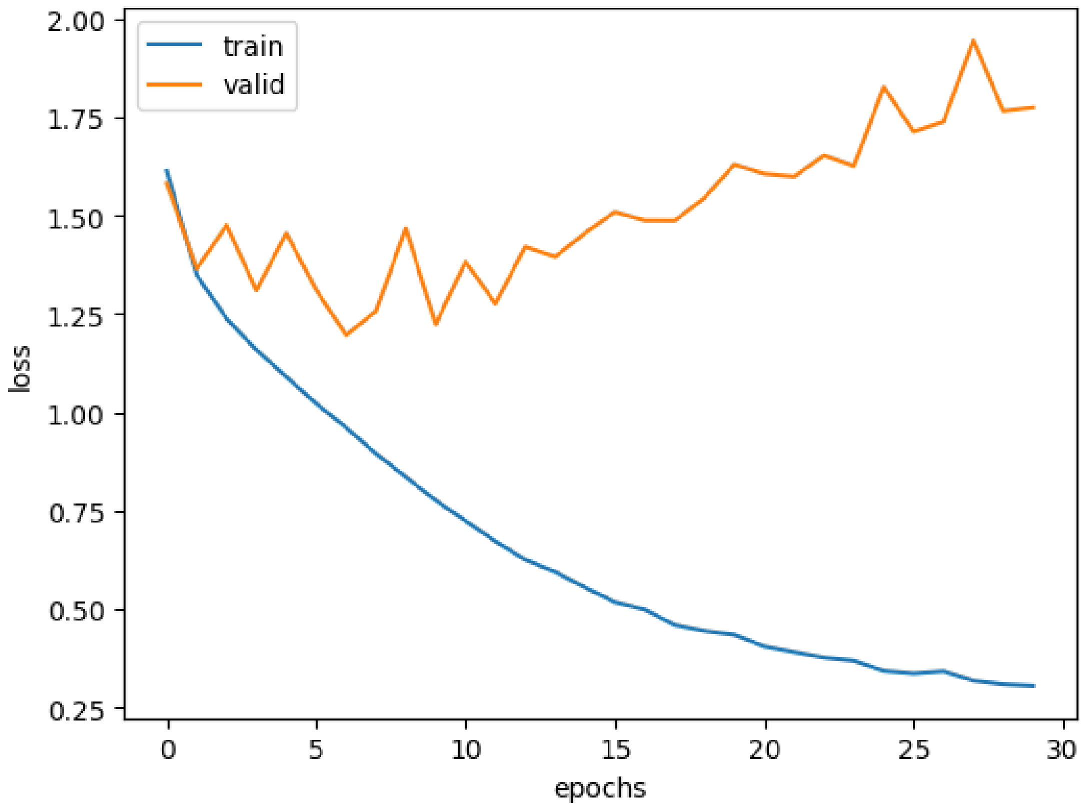
```
Epoch 1/30
898/898 [==============================] - 14s 10ms/step - loss: 1.6135 -
accuracy: 0.3894 - val_loss: 1.5822 - val_accuracy: 0.4030
Epoch 2/30
898/898 [==============================] - 9s 10ms/step - loss: 1.3493 -
accuracy: 0.4911 - val_loss: 1.3644 - val_accuracy: 0.4674
Epoch 3/30
898/898 [==============================] - 9s 10ms/step - loss: 1.2384 -
accuracy: 0.5365 - val_loss: 1.4760 - val_accuracy: 0.4695
Epoch 4/30
898/898 [==============================] - 8s 9ms/step - loss: 1.1588 -
accuracy: 0.5645 - val_loss: 1.3102 - val_accuracy: 0.5163
Epoch 5/30
898/898 [==============================] - 9s 11ms/step - loss: 1.0906 -
accuracy: 0.5911 - val_loss: 1.4555 - val_accuracy: 0.4503
Epoch 6/30
898/898 [==============================] - 9s 10ms/step - loss: 1.0224 -
accuracy: 0.6156 - val_loss: 1.3114 - val_accuracy: 0.5109
Epoch 7/30
898/898 [==============================] - 9s 9ms/step - loss: 0.9615 -
```

```
accuracy: 0.6385 - val_loss: 1.1964 - val_accuracy: 0.5620
Epoch 8/30
898/898 [==============================] - 8s 9ms/step - loss: 0.8951 -
accuracy: 0.6659 - val_loss: 1.2569 - val_accuracy: 0.5444
Epoch 9/30
898/898 [==============================] - 9s 10ms/step - loss: 0.8359 -
accuracy: 0.6886 - val_loss: 1.4681 - val_accuracy: 0.4941
Epoch 10/30
898/898 [==============================] - 9s 10ms/step - loss: 0.7763 -
accuracy: 0.7098 - val_loss: 1.2236 - val_accuracy: 0.5860
Epoch 11/30
898/898 [==============================] - 9s 10ms/step - loss: 0.7243 -
accuracy: 0.7297 - val_loss: 1.3832 - val_accuracy: 0.5627
Epoch 12/30
898/898 [==============================] - 9s 10ms/step - loss: 0.6722 -
accuracy: 0.7470 - val_loss: 1.2763 - val_accuracy: 0.5872
Epoch 13/30
898/898 [==============================] - 9s 10ms/step - loss: 0.6255 -
accuracy: 0.7688 - val_loss: 1.4210 - val_accuracy: 0.5644
Epoch 14/30
898/898 [==============================] - 9s 10ms/step - loss: 0.5943 -
accuracy: 0.7818 - val_loss: 1.3956 - val_accuracy: 0.5756
Epoch 15/30
898/898 [==============================] - 9s 10ms/step - loss: 0.5550 -
accuracy: 0.7958 - val_loss: 1.4557 - val_accuracy: 0.5780
Epoch 16/30
898/898 [==============================] - 9s 10ms/step - loss: 0.5178 -
accuracy: 0.8093 - val_loss: 1.5094 - val_accuracy: 0.5502
Epoch 17/30
898/898 [==============================] - 10s 11ms/step - loss: 0.4985 -
accuracy: 0.8155 - val_loss: 1.4877 - val_accuracy: 0.5701
Epoch 18/30
898/898 [==============================] - 10s 11ms/step - loss: 0.4597 -
accuracy: 0.8312 - val_loss: 1.4873 - val_accuracy: 0.5802
Epoch 19/30
898/898 [==============================] - 9s 10ms/step - loss: 0.4444 -
accuracy: 0.8374 - val_loss: 1.5455 - val_accuracy: 0.5885
Epoch 20/30
898/898 [==============================] - 10s 11ms/step - loss: 0.4346 -
accuracy: 0.8402 - val_loss: 1.6302 - val_accuracy: 0.5868
Epoch 21/30
898/898 [==============================] - 9s 10ms/step - loss: 0.4050 -
accuracy: 0.8508 - val_loss: 1.6066 - val_accuracy: 0.5846
Epoch 22/30
898/898 [==============================] - 9s 10ms/step - loss: 0.3907 -
accuracy: 0.8591 - val_loss: 1.5995 - val_accuracy: 0.5933
Epoch 23/30
898/898 [==============================] - 8s 9ms/step - loss: 0.3766 -
```

```
accuracy: 0.8634 - val_loss: 1.6534 - val_accuracy: 0.5743
Epoch 24/30
898/898 [==============================] - 9s 10ms/step - loss: 0.3686 -
accuracy: 0.8647 - val_loss: 1.6265 - val_accuracy: 0.5851
Epoch 25/30
898/898 [==============================] - 9s 10ms/step - loss: 0.3427 -
accuracy: 0.8740 - val_loss: 1.8270 - val_accuracy: 0.5871
Epoch 26/30
898/898 [==============================] - 9s 10ms/step - loss: 0.3363 -
accuracy: 0.8770 - val_loss: 1.7140 - val_accuracy: 0.5907
Epoch 27/30
898/898 [==============================] - 10s 11ms/step - loss: 0.3417 -
accuracy: 0.8749 - val_loss: 1.7387 - val_accuracy: 0.5981
Epoch 28/30
898/898 [==============================] - 10s 11ms/step - loss: 0.3181 -
accuracy: 0.8845 - val_loss: 1.9459 - val_accuracy: 0.5690
Epoch 29/30
898/898 [==============================] - 10s 11ms/step - loss: 0.3087 -
accuracy: 0.8898 - val_loss: 1.7662 - val_accuracy: 0.5855
Epoch 30/30
898/898 [==============================] - 14s 15ms/step - loss: 0.3047 -
accuracy: 0.8900 - val_loss: 1.7754 - val_accuracy: 0.5896
```

[42]:
```
plt.plot(history2.history["loss"])
plt.plot(history2.history["val_loss"])
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(["train","valid"])
```

[42]:
```
<matplotlib.legend.Legend at 0x7fc424199e50>
```



[43]:
```
y_pred2 = model2.predict(test_images)
```

```
225/225 [==============================] - 1s 4ms/step
```

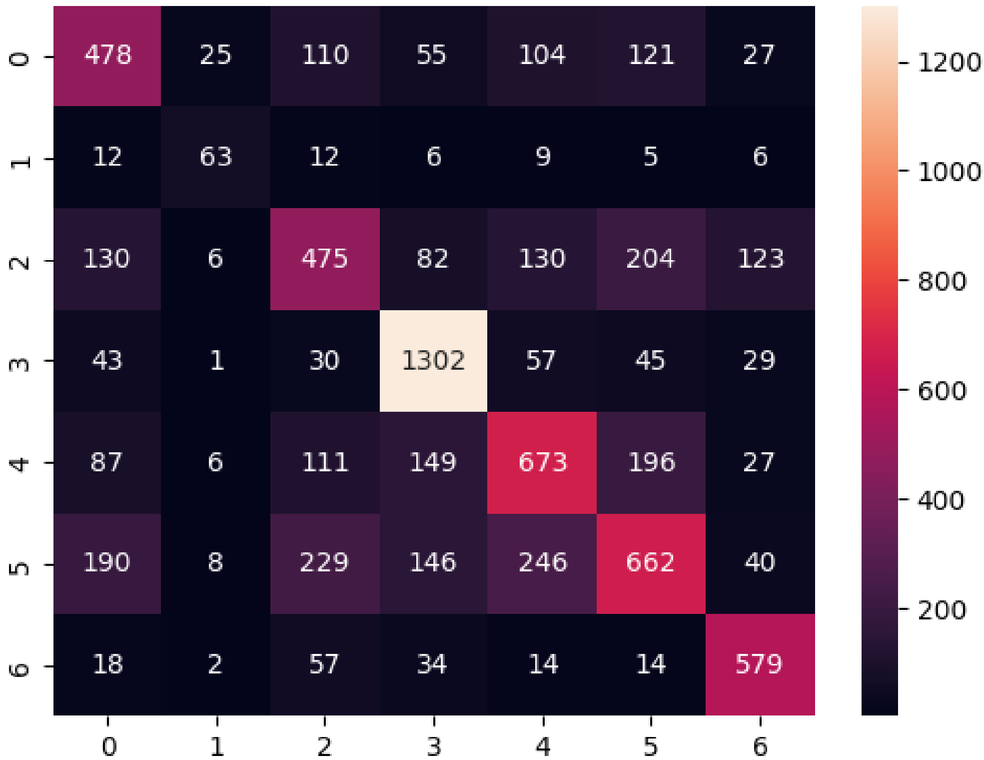
[44]:
```
print(classification_report(y_pred2.argmax(axis=1), test_labels))
```

```
              precision    recall  f1-score   support

           0       0.50      0.52      0.51       920
           1       0.57      0.56      0.56       113
           2       0.46      0.41      0.44      1150
           3       0.73      0.86      0.79      1507
           4       0.55      0.54      0.54      1249
           5       0.53      0.44      0.48      1521
           6       0.70      0.81      0.75       718

    accuracy                           0.59      7178
   macro avg       0.58      0.59      0.58      7178
weighted avg       0.58      0.59      0.58      7178
```

[45]:
```
sns.heatmap(confusion_matrix(y_pred2.argmax(axis=1), test_labels), fmt='g',␣
 ↪annot=True)
plt.show()
```



CNN Model with Dropout and GlobalAveragePooling2D layer

[62]:
```
model3 = Sequential()
```

[63]:
```
model3.add(Conv2D(32, (3, 3), input_shape=(48,48,3), activation="leaky_relu",␣
 ↪padding="same"))
model3.add(Conv2D(32, (3, 3),activation="leaky_relu", padding="same"))
model3.add(Dropout(0.1))
model3.add(MaxPool2D(2,2))
model3.add(Conv2D(64, (3, 3), activation="leaky_relu", padding="same"))
model3.add(Dropout(0.2))
model3.add(MaxPool2D(2,2))
model3.add(Conv2D(128, (3, 3), activation="leaky_relu", padding="same"))
model3.add(Dropout(0.3))
model3.add(MaxPool2D(2,2))
model3.add(Conv2D(256, (3, 3), activation="leaky_relu", padding="same"))
model3.add(MaxPool2D(2,2))
model3.add(Dropout(0.3))
```

```
model3.add(GlobalAveragePooling2D())
model3.add(Dense(512, activation="relu"))
model3.add(Dense(7, activation="softmax"))
```

[64]:
```
model3.summary()
```

```
Model: "sequential_8"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_42 (Conv2D)          (None, 48, 48, 32)        896

 conv2d_43 (Conv2D)          (None, 48, 48, 32)        9248

 dropout_17 (Dropout)        (None, 48, 48, 32)        0

 max_pooling2d_32 (MaxPoolin  (None, 24, 24, 32)       0
 g2D)

 conv2d_44 (Conv2D)          (None, 24, 24, 64)        18496

 dropout_18 (Dropout)        (None, 24, 24, 64)        0

 max_pooling2d_33 (MaxPoolin  (None, 12, 12, 64)       0
 g2D)

 conv2d_45 (Conv2D)          (None, 12, 12, 128)       73856

 dropout_19 (Dropout)        (None, 12, 12, 128)       0

 max_pooling2d_34 (MaxPoolin  (None, 6, 6, 128)        0
 g2D)

 conv2d_46 (Conv2D)          (None, 6, 6, 256)         295168

 max_pooling2d_35 (MaxPoolin  (None, 3, 3, 256)        0
 g2D)

 dropout_20 (Dropout)        (None, 3, 3, 256)         0

 global_average_pooling2d_3   (None, 256)              0
 (GlobalAveragePooling2D)

 dense_10 (Dense)            (None, 512)               131584

 dense_11 (Dense)            (None, 7)                 3591
```

```
=================================================================
Total params: 532,839
Trainable params: 532,839
Non-trainable params: 0
_________________________________________________________________
```

[65]:
```
model3.compile(optimizer='adam', loss="sparse_categorical_crossentropy",␣
↪metrics=['accuracy'])
```

[66]:
```
history3 = model3.fit(train_images, train_labels, validation_data =␣
↪(test_images, test_labels), epochs=30)
```

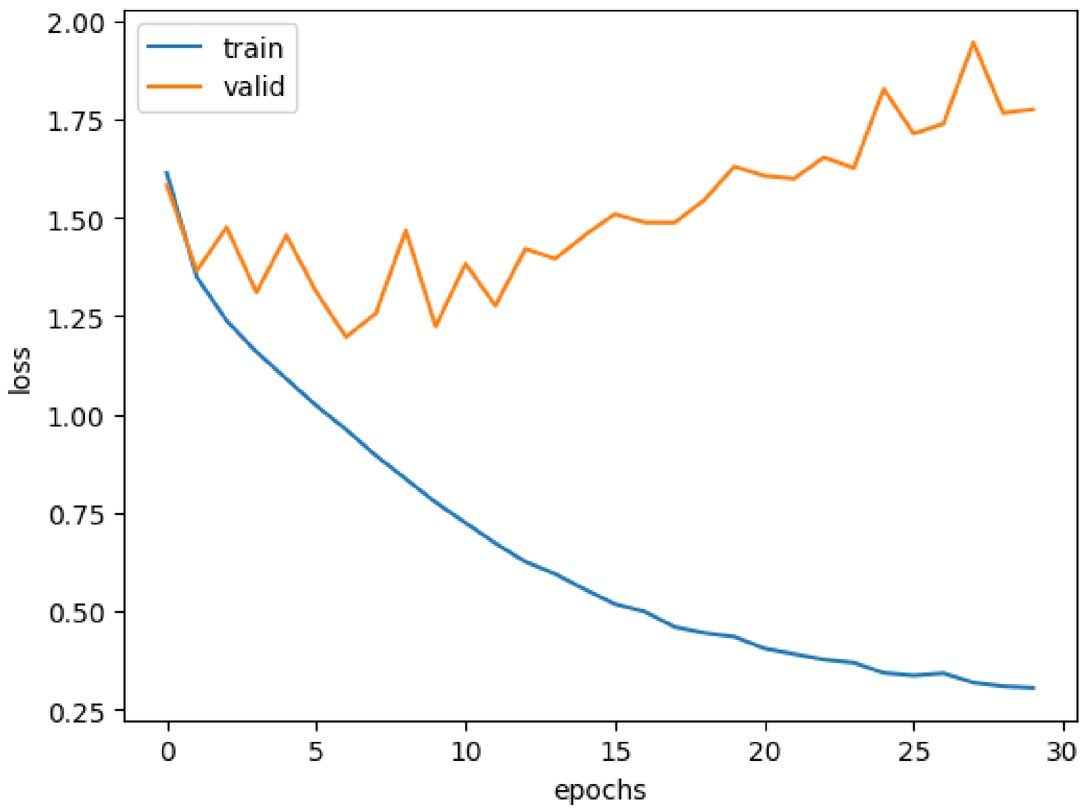
```
Epoch 1/30
898/898 [==============================] - 14s 12ms/step - loss: 1.7980 -
accuracy: 0.2588 - val_loss: 1.7406 - val_accuracy: 0.2856
Epoch 2/30
898/898 [==============================] - 9s 10ms/step - loss: 1.5542 -
accuracy: 0.3918 - val_loss: 1.4392 - val_accuracy: 0.4570
Epoch 3/30
898/898 [==============================] - 10s 11ms/step - loss: 1.3331 -
accuracy: 0.4868 - val_loss: 1.3010 - val_accuracy: 0.5001
Epoch 4/30
898/898 [==============================] - 9s 10ms/step - loss: 1.2165 -
accuracy: 0.5331 - val_loss: 1.2635 - val_accuracy: 0.5293
Epoch 5/30
898/898 [==============================] - 10s 11ms/step - loss: 1.1348 -
accuracy: 0.5624 - val_loss: 1.2208 - val_accuracy: 0.5412
Epoch 6/30
898/898 [==============================] - 9s 10ms/step - loss: 1.0856 -
accuracy: 0.5869 - val_loss: 1.3450 - val_accuracy: 0.5287
Epoch 7/30
898/898 [==============================] - 10s 11ms/step - loss: 1.0335 -
accuracy: 0.6034 - val_loss: 1.1944 - val_accuracy: 0.5683
Epoch 8/30
898/898 [==============================] - 10s 11ms/step - loss: 0.9938 -
accuracy: 0.6234 - val_loss: 1.1370 - val_accuracy: 0.5800
Epoch 9/30
898/898 [==============================] - 10s 11ms/step - loss: 0.9577 -
accuracy: 0.6353 - val_loss: 1.2025 - val_accuracy: 0.5665
Epoch 10/30
898/898 [==============================] - 10s 11ms/step - loss: 0.9276 -
accuracy: 0.6456 - val_loss: 1.1428 - val_accuracy: 0.5853
Epoch 11/30
898/898 [==============================] - 10s 11ms/step - loss: 0.8878 -
accuracy: 0.6641 - val_loss: 1.2033 - val_accuracy: 0.5791
Epoch 12/30
898/898 [==============================] - 10s 11ms/step - loss: 0.8676 -
accuracy: 0.6724 - val_loss: 1.1935 - val_accuracy: 0.5972
```

```
Epoch 13/30
898/898 [==============================] - 9s 10ms/step - loss: 0.8433 -
accuracy: 0.6815 - val_loss: 1.1968 - val_accuracy: 0.5940
Epoch 14/30
898/898 [==============================] - 10s 11ms/step - loss: 0.8120 -
accuracy: 0.6932 - val_loss: 1.1252 - val_accuracy: 0.6055
Epoch 15/30
898/898 [==============================] - 9s 11ms/step - loss: 0.7882 -
accuracy: 0.7036 - val_loss: 1.2851 - val_accuracy: 0.5750
Epoch 16/30
898/898 [==============================] - 9s 11ms/step - loss: 0.7660 -
accuracy: 0.7114 - val_loss: 1.1333 - val_accuracy: 0.6080
Epoch 17/30
898/898 [==============================] - 9s 10ms/step - loss: 0.7499 -
accuracy: 0.7165 - val_loss: 1.2586 - val_accuracy: 0.5953
Epoch 18/30
898/898 [==============================] - 10s 11ms/step - loss: 0.7342 -
accuracy: 0.7218 - val_loss: 1.2454 - val_accuracy: 0.6110
Epoch 19/30
898/898 [==============================] - 10s 11ms/step - loss: 0.7198 -
accuracy: 0.7275 - val_loss: 1.3128 - val_accuracy: 0.5926
Epoch 20/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6998 -
accuracy: 0.7351 - val_loss: 1.2665 - val_accuracy: 0.6037
Epoch 21/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6881 -
accuracy: 0.7390 - val_loss: 1.2841 - val_accuracy: 0.5993
Epoch 22/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6683 -
accuracy: 0.7483 - val_loss: 1.3442 - val_accuracy: 0.6142
Epoch 23/30
898/898 [==============================] - 9s 10ms/step - loss: 0.6638 -
accuracy: 0.7482 - val_loss: 1.4800 - val_accuracy: 0.5939
Epoch 24/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6528 -
accuracy: 0.7557 - val_loss: 1.2864 - val_accuracy: 0.6162
Epoch 25/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6314 -
accuracy: 0.7619 - val_loss: 1.6107 - val_accuracy: 0.5717
Epoch 26/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6252 -
accuracy: 0.7646 - val_loss: 1.4124 - val_accuracy: 0.5903
Epoch 27/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6186 -
accuracy: 0.7683 - val_loss: 1.3777 - val_accuracy: 0.6103
Epoch 28/30
898/898 [==============================] - 10s 11ms/step - loss: 0.6033 -
accuracy: 0.7747 - val_loss: 1.5117 - val_accuracy: 0.6163
```

```
Epoch 29/30
898/898 [==============================] - 10s 11ms/step - loss: 0.5950 -
accuracy: 0.7748 - val_loss: 1.3413 - val_accuracy: 0.6154
Epoch 30/30
898/898 [==============================] - 10s 11ms/step - loss: 0.5897 -
accuracy: 0.7807 - val_loss: 1.4010 - val_accuracy: 0.6106
```

[68]:
```
plt.plot(history2.history["loss"])
plt.plot(history2.history["val_loss"])
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(["train","valid"])
```

[68]:
```
<matplotlib.legend.Legend at 0x7fc4215bc760>
```



[67]:
```
y_pred3 = model3.predict(test_images)
```

```
225/225 [==============================] - 1s 4ms/step
```

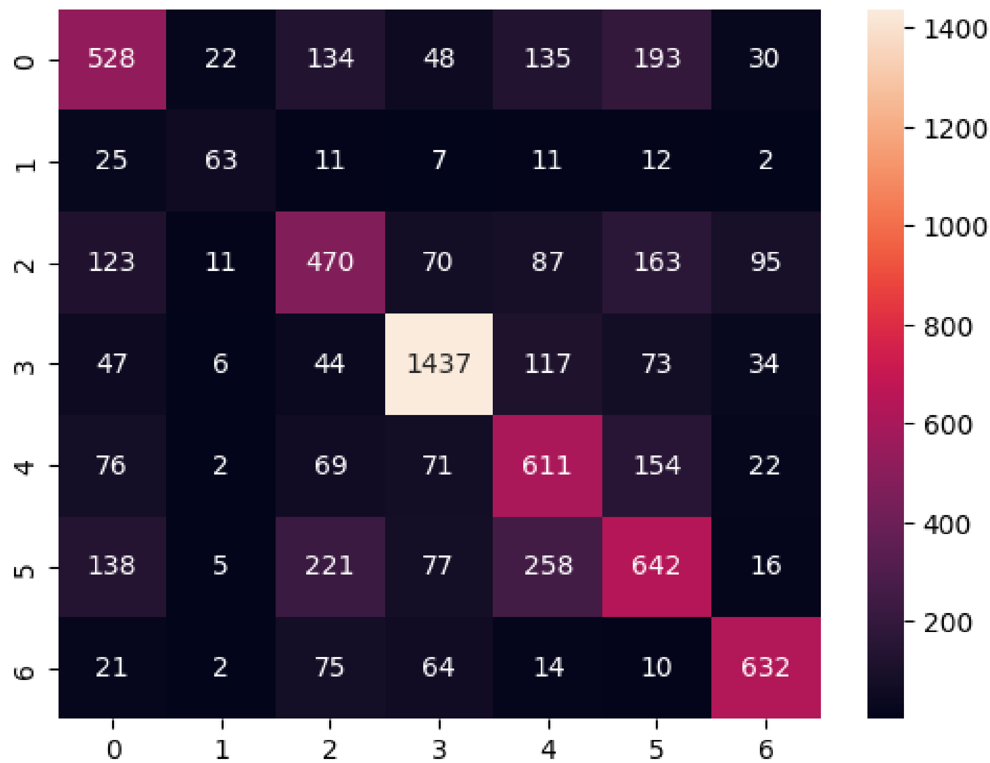
[69]:
```
print(classification_report(y_pred3.argmax(axis=1), test_labels))
```

```
              precision    recall  f1-score   support
```

```
           0       0.55      0.48      0.52      1090
           1       0.57      0.48      0.52       131
           2       0.46      0.46      0.46      1019
           3       0.81      0.82      0.81      1758
           4       0.50      0.61      0.55      1005
           5       0.51      0.47      0.49      1357
           6       0.76      0.77      0.77       818

    accuracy                           0.61      7178
   macro avg       0.59      0.59      0.59      7178
weighted avg       0.61      0.61      0.61      7178
```

[70]:
```
sns.heatmap(confusion_matrix(y_pred3.argmax(axis=1), test_labels), fmt='g',
    annot=True)
plt.show()
```



[ ]: