FINAL PROJECT REPORT

Atul Kumar Tiwari
2018CSB1077
Navtejpreet Singh
2018CSB1107
Indian Institute of Technology,
Ropar

Abstract

Machines are adapting to the society and are nowadays being used in many different areas. As the capabilities and power of modern day machines saw a significant increase, so did the dependencies. Data scientists are working day and night to enable machines to perceive the information they gather from the environment in a more meaningful manner. These information include speech recognition, face recognition, self-driving cars, emotion detector and many more such tasks.

1 Introduction

One such task is emotion detection using machines. The idea of futuristic robots starts by, imagination of them perceiving visual and audio information as humans do. Facial and emotion detection enable machines to learn to interact socially with humans. Which can be achieved by data sciences and then applied to various fields like e-learning, healthcare, housekeeping etc.

2 Our Approach

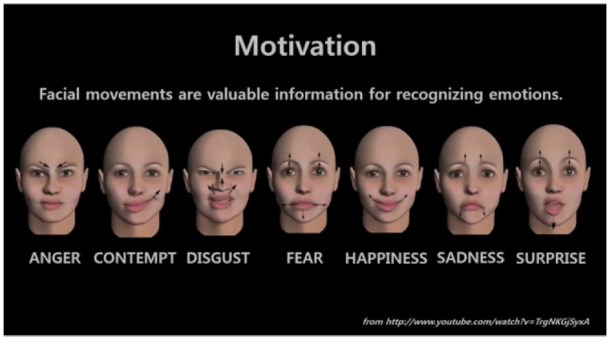
2.1 Dataset

We are using modified FER2013 dataset consisting of around 28709 images distributed across seven emotion labels. The images are 48*48 grayscale cropped images. The original dataset 7 class labels or emotions. We reduced the number of class labels for our purpose mainly due to uneven unbalanced dataset i.e. as number of pictures per emotion were unequal and we removed the ones which had too less pictures. Finally, for our purpose we worked with following class labels:

- Class 0 => Angry
- Class 1 => Surprise
- Class 2 => Neutral
- Class 3 => Happy
- Class 4 => Sad

Surprise(1)

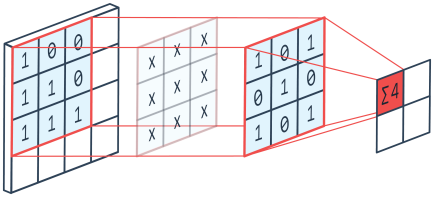
Happy(3)

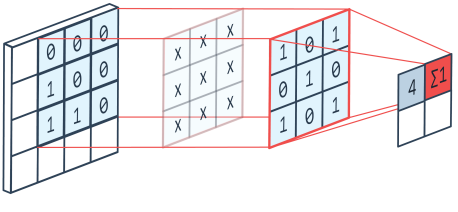


As illustrated above, facial movements provide a vital information for recognizing emotions in humans. eg. anger is easily recognized with a frown, surprise by an open mouth and raised eyebrows, etc.

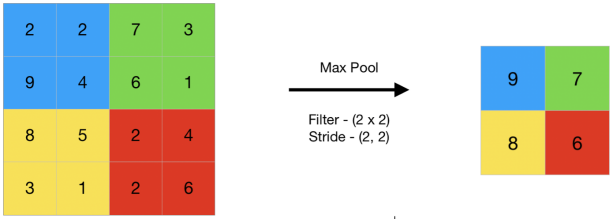
2.2 Methodology

In this project we use a deep learning technique called Convolutional Neural Networks (CNNs) to establish a classification model that combines feature extraction with classification. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex.When programming a CNN, the input is a matrix/tensor with shape (number of images) x (image width) x (image height) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map width) x (feature map height) x (feature map channels).

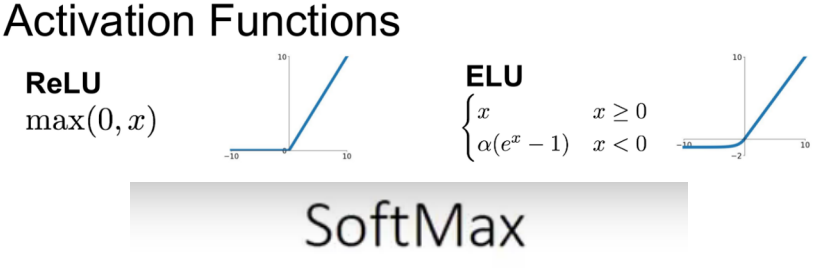




The convulational layers become complex as we go deep i.e. let's say initial layers detect vertical edges and horizontal edges in the images whereas the next convulational layer detects shapes such as oval, rectangle etc. And as we proceed more deeper convulational layer detect components such as eyes, nose and lips.
We aim to apply multiple layer of CNN with pooling and activation layers stacked between them.



The aim of these pooling layer is to reduce the dimensionality of the data and thus streamlining the underlying calculations. An added benefit of max pooling is that it forces the network to focus on a few neurons instead of all of them which has a regularizing effect on the network, making it less likely to over-fit the training data and hopefully generalize well.
Activation functions play important role in neural networks. When a matrix passes through a 'relu' layer all the negative values in the matrix take a value of zero.



$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

The final layers of our model are 'fully connected' layers, here the input representation is flattened into a feature vector and passed through a network of neurons using these, we take vote for our classes and the given example is classified.

2.3 Implementation

For implementing CNN in python we use "keras" module which is inbuilt in the "tensorflow" library. Our Dataset "FER 2013" contains images with seven emotions, for simplicity we will consider only 5 of them namely: "angry","surprise","neutral","happy" and "sad".

```
model = keras.models.Sequential()

model.add(Conv2D(nfeatures,kernel_size = (3,3),activation = 'elu', input_shape = (ferheight,ferwidth,1) ))
model.add(Conv2D(nfeatures,kernel_size = (3,3),activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))


model.add(Conv2D(2*nfeatures,kernel_size = (3,3),activation = 'relu'))
model.add(Conv2D(2*nfeatures,kernel_size = (3,3),activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))


model.add(Conv2D(4*nfeatures,kernel_size = (3,3),activation = 'relu'))
model.add(Conv2D(4*nfeatures,kernel_size = (3,3),activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Flatten())


model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(nlabels, activation='softmax'))
```

Here we employed introduced actiavtion function 'relu' and 'elu' to bring non-linearity in the data for better prediction results. These are pretty standard activation functions for this purpose. Softmax activation function is applied in the final layer after the dense layer for multiclass classification.
Also note kernel size applied here is small i.e. (3,3) as we wanted to capture small patterns from the initial input image. We increase the number of applied kernels or nfeatures as we proceed deeper into the network.
Note: As padding is off by deafult we see that facial dimension i.e. (width,height) is converted to (width-2,height-2) due to kernel size of (3,3)
Dropout layer were introduced to randomly drop out weights from the learning to avoid overfitting on training data. Here, hyperparameter 0.5 denotes that any parameter is dropped with a probability of 0.5. Training the model:-

```
Train on 24176 samples, validate on 3037 samples
Epoch 1/30
24176/24176 [==============================] - 435s 18ms/step - loss: 1.7169 - accuracy: 0.3160 - val_loss: 1.4389 - val_accura
cy: 0.3823
Epoch 2/30
24176/24176 [==============================] - 377s 16ms/step - loss: 1.4197 - accuracy: 0.3815 - val_loss: 1.3676 - val_accura
cy: 0.4215
Epoch 3/30
24176/24176 [==============================] - 370s 15ms/step - loss: 1.3477 - accuracy: 0.4284 - val_loss: 1.3214 - val_accura
cy: 0.4699
Epoch 4/30
24176/24176 [==============================] - 372s 15ms/step - loss: 1.3004 - accuracy: 0.4568 - val_loss: 1.3079 - val_accura
cy: 0.4524
Epoch 5/30
24176/24176 [==============================] - 376s 16ms/step - loss: 1.2693 - accuracy: 0.4671 - val_loss: 1.2456 - val_accura
cy: 0.4979
Epoch 6/30
```

After multiple convolutional layers in series with pooling and dropout operations, the image representation is converted (.Flatten()) into a feature vector that is passed into a Multi-Layer Perceptron, which merely is a neural network with at least three layers. This is referred

to as a Fully-Connected Layer. In the last step we use "softmax" activation function to predict the output.
Following is the model summary of the implementation with all layers.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 46, 46, 64)	640
conv2d_2 (Conv2D)	(None, 44, 44, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 22, 22, 64)	0
dropout_1 (Dropout)	(None, 22, 22, 64)	0
conv2d_3 (Conv2D)	(None, 20, 20, 128)	73856
conv2d_4 (Conv2D)	(None, 18, 18, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 9, 9, 128)	0
dropout_2 (Dropout)	(None, 9, 9, 128)	0
conv2d_5 (Conv2D)	(None, 7, 7, 256)	295168
conv2d_6 (Conv2D)	(None, 5, 5, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 2, 2, 256)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 1024)	1049600
dropout_3 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 5)	5125

Total params: 2,198,981
Trainable params: 2,198,981
Non-trainable params: 0

3 Results

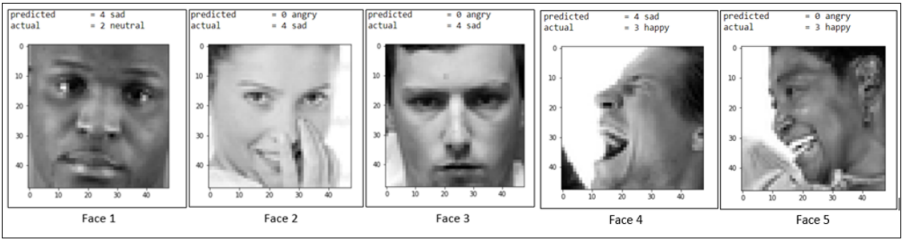
Our Final model (trained for 30 epochs) gives an accuracy of 53.19 percent.

```
accuracy on unseen test data is :  0.5319361277445109
Classification report :                precision    recall  f1-score   support

           0       0.54      0.28      0.37       491
           1       0.94      0.41      0.57       416
           2       0.39      0.64      0.48       626
           3       0.68      0.75      0.71       879
           4       0.41      0.40      0.40       594

    accuracy                           0.53      3006
   macro avg       0.59      0.49      0.51      3006
weighted avg       0.58      0.53      0.53      3006
```

```
confusion matrix :
 [[137   4 159  87 104]
 [ 22 169 122  63  40]
 [ 32   1 398  80 115]
 [ 19   4 111 658  87]
 [ 43   1 233  80 237]]
```

3.1 Looking at the mistakes

Let's have a look at some of the wrongly classified results images.



For Face 1 its really difficult to see whether the person is sad on neutral, even humans might fail in recognizing the correct emotion. Similar is the case with Face 3 as well. We could easily consider the emotion shown by person to be either sad or angry.
For Face 2 as the hand is masking some of the area of the subject's face thus model is unable to correctly predict her emotion.
For Face 4 and Face 5 side profile of the face are generating incorrect results. Side profile of the faces have fewer details for the model to comprehend the emotion.

4 Conclusion

Our model achieved satisfactory accuracy of 53.19% on unseen data. This might not seem like a lot. Emotions are abstract quantities and are sometimes very difficult for even humans to understand fully. Two different individuals might look at a image and see different emotions. So, the human level performance on this task must be <100% and the highest known performance on this task is around 70%.

5 Literature Survey

There are already many developments in the field of computer vision and data science. These aim to solve and automate many day to day tasks.
For data preprocessing, help was taken from this source. https://www.kaggle.com/kakaloveyou/fer2013-preprocessing Original dataset had a list of space separated pixel values with emotion labels. As this was irregular data was to be reshaped to regularize all the datapoints.
One such implementation is by Daniel Llatas Spiers. https://uu.diva-portal.org/smash/get/diva2:952138/FULLTEXT01.pdf. He gave beautiful insights on the model he developed. He used artificial neural networks and convolutional neural networks to predict facial emotion. Dataset he used for it were and Extended Cohn-Kanade[CK+] (which included 593 sequence of 123 subject and were recorded in a lab following a set of instruction) and Affectiva-MIT Facial Expression Data [AM-FED] (which encompases 242 facial videos of subject captured in the wild). He managed to get an test accuracy of around 70%.

6 Dependencies

- pip install numpy
- pip install pandas
- pip install tensorflow
- pip install keras
- pip install opencv-python

References

https://www.kaggle.com/kakaloveyou/fer2013-preprocessing
https://towardsdatascience.com/emotion-detection-a-machine-learn
gi=5f609e477554
https://en.wikipedia.org/wiki/Convolutional_neural_network
https://towardsdatascience.com/building-a-convolutional-neural-n