Machine learning engineer nanodegree

Capstone project

Facial Expression Recognition

Ola Hamdy Ahmed
November,2019

1.Definition

1.1 Project overview

Facial expression is the facial changes that reflects a person's internal emotional states, intentions and feelings. Humans usually employ different cues to express their emotions, such as facial expressions, hand gestures and voice. Facial expressions represent up to 55% of human-communications while other ways such as oral language are allocated a mere 7% of emotion expression.
Since then, Facial expression recognition (FER) has become a growing area of research with numerous real-world applications such as mental state identification, lie detection, human behaviour understanding and security.

The goal of this project is to apply deep learning techniques to recognize the key seven human emotions: anger, disgust, fear, happiness, sadness, surprise and neutrality.
The dataset used for this project is Kaggle's Facial Expression Recognition Challenge dataset. This dataset is representative because of its size, unstructured nature of faces and relatively uniform distribution of the data across the seven main human emotions.

1.2 Problem statement

One of the non-verbal communication methods by which one can understand the emotional state of a person is the expression of his face. As the technology runs our lives these days, the majority of our time is spent in interacting with computers and mobile phones in our daily lives. Therefore, Adding facial expression recognition feature to our devices to expect the user's emotional state can improve human-computer interaction. In this way, it can be used in a healthcare system to detect humans' mental state and improve it by exploring their behaviour patterns.

The proposed solution is to apply deep learning techniques that have proven to be highly successful in the field of image classification.
In this project, I am going to use Convolutional neural network algorithm with the aid of the Keras library. The input of the model will be a human face image and the model predicts the facial expression among the seven basic expressions (anger, disgust, fear, happiness, sadness, surprise and neutrality).

1.3 Metrics

Since our problem is a multi-label classification problem, I proposed using accuracy as an evaluation metrics. It is the number of correctly classified data points over the total number of observations

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

We need to compare the accuracy of training set and validation set while training, to make sure that the model generalizes well (doesn't overfit) and it uses the data in the best possible way (doesn't underfit). My goal is to create a model that doesn't suffer from neither overfitting nor underfitting with accuracy of at least 60% for both training set and validation set.

2.Analysis

2.1 Data exploration

In this project, the dataset used is Kaggle's Facial Expression Recognition Challenge dataset. This dataset is representative because of its size, unstructured nature of faces and relatively uniform distribution of the data across the seven main human emotions.

The chosen dataset should not only provide a representative number of images, but also should contain data that is uniformly distributed across the race, sex, age and ethnicity, and with a relatively even distribution across the emotions. The Kaggle dataset meets all these aspects.
The dataset labels are the seven key emotions:

- Angry
- Disgust
- Fear
- Happy
- Sad
- Surprise
- Neutral

The dataset attributes:

- 35,887 image
- Image format: 48x48 pixels
- Various individuals across the entire spectrum of: ethnicity, age, gender and race, with all these images being taken at various angles.
- The seven key emotions are relatively equally distributed with the one exception being disgust, at ~1.5%.

Figure1 Example pictures of the seven expressions of Kaggle dataset

The dataset contains 3 columns, “emotion”, “pixels” and “Usage”:

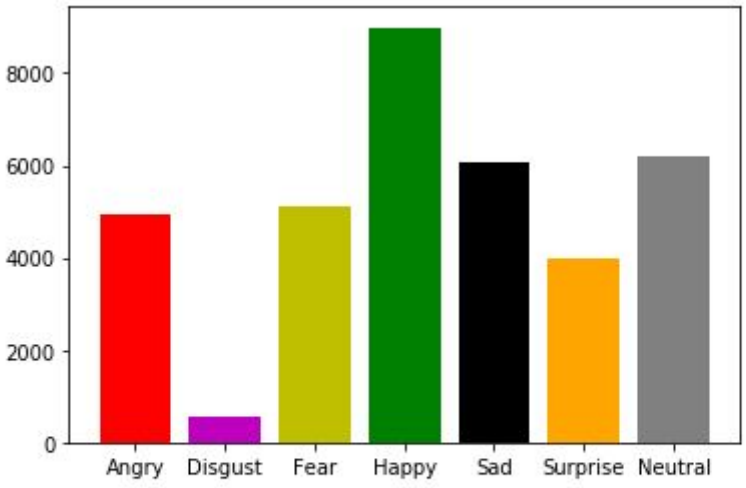
- The “emotion” column contains a numeric code from 0 to 6 that represents the emotion of the image (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).
- The “pixels” column contains a string for each image. This string consists of space-separated pixel values in row major order.
- The “Usage” column helps to split the data into 80% training set, 10% validation set and 10% test set. So, “Usage” column has three values, “Training” specifies the images of the training set, “PublicTest” specifies the images of the validation set and “PrivateTest” specifies the images of the testset.

	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

Figure 2. First five rows in the Dataset

2.2 Exploratory visualization

The count of images for each class is in the dataset is obtained and the graph is plotted as shown below. This plot is the most suitable one to start with as it is simple and easy to understand. It also gives valuable information on how the values are distributed.




```
All data:        [4953  547 5121 8989 6077 4002 6198]
Train data:      [3995  436 4097 7215 4830 3171 4965]
Validation data: [467  56 496 895 653 415 607]
```

Figure 3. Data distribution

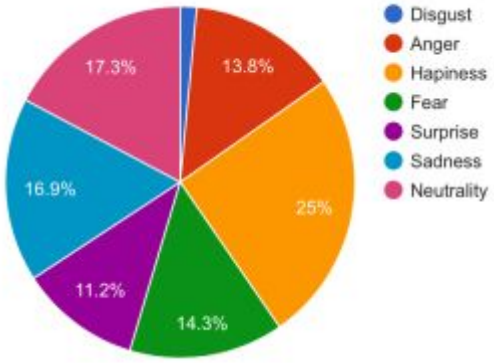


Figure 4. Dataset distribution with percentages

Here, we can see that these seven key emotions are relatively equally distributed with the one exception being disgust, at ~1.5% .

2.3 Algorithms and techniques

The main goal of this project is to build a Convolutional Neural Network (CNN) to predict the facial expression among the seven basic expressions. CNN is the state-of-art algorithm for most image processing tasks including classification. It can be thought of automatic feature extractor from the image, It works really well when the order of the features is important, and this is exactly the case with images. The numeric representation of the image pixels are our features, and the order of pixels is very significant.
The CNN approach is based on the idea that the model function properly based on a local understanding of the image. It uses fewer parameters compared to a fully connected network by reusing the same parameter numerous times. While a fully connected network generates

weights from each pixel on the image, a convolutional neural network generates just enough weights to scan a small area of the image at any given time. CNN works by applying a series of layers on the data in a sequential manner, each layer is composed by neurons that filter the data.

Each convolutional layer within a neural network should have the following attributes:

- Input is a tensor with shape (number of images) x (image width) x (image height) x (image depth).
- Convolutional kernels whose width and height are hyper-parameters, and whose depth must be equal to that of the image. Convolutional layers convolve the input and pass its result to the next layer.

Convolutional networks may also include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer.
Since our problem is to expect a facial expression in an image, we've decided to use CNN implemented in Keras library with a Tensorflow backend. We've used keras as it offers a great balance between simplicity and efficiency.

To achieve our goal, we first need to train our convolutional network with labeled images (Images that their labels are known). Before training, I intended to convert the vector-based data into image data (multi-dimensional arrays). Once I convert them to uniform sized image data then I can feed them to stack of Convolutional layers.

While training, model will come up with different set of weights during each set of iteration. In order to capture the best set of weights I have added a 'ModelCheckpoint'. This checkpoint keeps track of the best set weights (largest validation accuracy value) by storing the latest best value in a mentioned physical location in the form of 'hdf5' file. Once the model's training is over then we can simply load the best weights from the file in to the model. This handy technique is provided by keras library.
After training, the model is ready to be fedden with an input image, and it can predict its label (the facial expression).

2.4 Benchmark

Support Vector Machine SVM is used as a benchmark model. SVMs have been widely applied to machine vision fields such as character, handwriting digit and text recognition, and more recently to satellite image classification. It is interesting to compare our results with a wide-scoped machine learning algorithm like SVM.
We can expect for CNN to have higher accuracy than the general algorithm. If this assumption is confirmed, we can conclude that, when it comes to image recognition we should give preference to more specialized algorithm in that domain.

3. Methodology

3.1 Data preprocessing

For the input images:

- Convert and reshape the space-separated pixel values string intro 48x48x1 numpy array (square grayscale image) to meet the requirements of the model.
- Divide the RGB values by the maximum value 255, so that, it ranges 0-1.
- Split the dataset into training, validation and testing set.
- Apply augmentations on images of the training set like shifting, mirroring,rotating and zooming, so that the model is more generalized for more features of images.

For the labels:

- Use one-hot technique to convert the “emotion” values into categorical values such that each emotion will be represented as 1x7 vector.

3.2 Implementation

3.2.1 Loading and preprocessing the dataset

- The dataset is loaded from Kaggle’s Facial Expression Recognition Challenge.
- For the “pixels” column, It contains a string for each image. This string consists of space-separated pixel values in row major order. So it is converted to be square grayscale images with size 48x48x1
- The pixel values of the images are divided by 255 so the range of the pixels is 0-1.
- The “emotions” column consists of a numeric code from 0 to 6 that represents the emotion of the image,So these values are converted into categorical values such that each emotion will be represented as 1x7 vector.

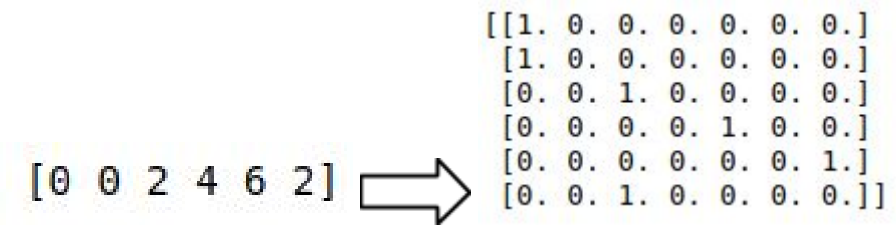


Figure 5. One-hot encoding of the first five values of labels

3.2.2 Building a model

- A simple CNN is created to classify images. It consists of six convolutional layers with three max pooling layers in between (one after each 2 ConvLayers).
- Filters were increased from 32 to 128 and dropout was added.
- Relu was used as an activation function for the hidden layers.

- Flattening layer along with two fully connected layers were added at the end of the network
- Last layer used softmax as the activation function with seven nodes for our seven labels.
- The model is compiled using ‘adam’ as the optimizer and ‘categorical_crossentropy’ as the loss function.

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 48, 48, 32)	320
conv2d_16 (Conv2D)	(None, 48, 48, 32)	9248
max_pooling2d_8 (MaxPooling2	(None, 24, 24, 32)	0
conv2d_17 (Conv2D)	(None, 24, 24, 64)	18496
conv2d_18 (Conv2D)	(None, 24, 24, 64)	36928
max_pooling2d_9 (MaxPooling2	(None, 12, 12, 64)	0
dropout_10 (Dropout)	(None, 12, 12, 64)	0
conv2d_19 (Conv2D)	(None, 12, 12, 128)	73856
conv2d_20 (Conv2D)	(None, 10, 10, 128)	147584
max_pooling2d_10 (MaxPooling	(None, 5, 5, 128)	0
dropout_11 (Dropout)	(None, 5, 5, 128)	0
flatten_3 (Flatten)	(None, 3200)	0
dense_7 (Dense)	(None, 512)	1638912
dropout_12 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 512)	262656
dropout_13 (Dropout)	(None, 512)	0
dense_9 (Dense)	(None, 7)	3591

Total params: 2,191,591
Trainable params: 2,191,591
Non-trainable params: 0

Figure 6. Model architecture

3.2.3 Training a model

- Data augmentation is applied on the images of the training set like shifting, mirroring,rotating and zooming, so that the model is more generalized for more features of images (non-overfitted).
- The model was trained on 200 epochs with a batch size of 256. In addition, training and validation datasets are provided. Therefore, While training the model is able to validate itself against validation dataset in every single iteration.
- During training, the weights that results the best validation accuracy is saved using “ModelCheckPoint”, in order to achieve best results.

3.2.4 Predicting new images

- To predict the label of an image, it is first reshaped to 48x48 grayscale image to meet the requirements of the model, and then, its pixel values are divided by 255, so that, it ranges 0-1.

3.2.5 Comparing the results

- As we mentioned, a simple SVM model is used as a benchmark model. It was first trained on the training set in its form of 1-dimensional array.
- It is then evaluated using test set.
- To predict a label of an image, it is first needed to be flattened as the model expects 1-dimensional array. So, we had to flatten our 2x2 array before feeding it to the classifier.

3.3 Refinement

- To get initial results, A simple CNN model was created and evaluated. This model had a validation and training accuracy around 30%.
- In order to improve the results, More convolutional layers are added with dropout, but the resultant model was overfitted. it had a training set accuracy around 98% and validation accuracy around 60%.
- In order to eliminate the overfitting, Data augmentation was applied on the training set like shifting, mirroring,rotating and zooming, so that the model is more generalized for more features of images.
- For tuning our hyperparameters, I've tried many combinations of the dimensions of the filters, number of batches and number of epochs till finally the fine-tuned model was selected that provided the best accuracy without neither overfitting nor underfitting. It gave us a validation accuracy of 65.98% and training accuracy of 65.5%.

4. Results

4.1 Model evaluation and validation

Since our problem is to expect a facial expression in an image, we've decided to use CNN implemented in Keras library with a Tensorflow backend. We've used keras as it offers a great balance between simplicity and efficiency.
During model creation and development, a validation set was used to evaluate our model. As we mentioned, the evaluation metrics used was "accuracy". I've compared the accuracy of training set and validation set while training, to make sure that the model generalizes well (doesn't overfit) and it uses the data in the best possible way (doesn't underfit). During training, the weights that results the best validation accuracy is saved using "ModelCheckPoint", in order to achieve best results.

After building our network and tuning our hyperparameters, We finally achieved a model that doesn't suffer from neither overfitting nor underfitting with validation set accuracy of 66.5%.
The figure below shows the change in accuracy while training with the number of epochs.
It shows us that the model generalizes well (doesn't overfit) and it uses the data in the best possible way (doesn't underfit).

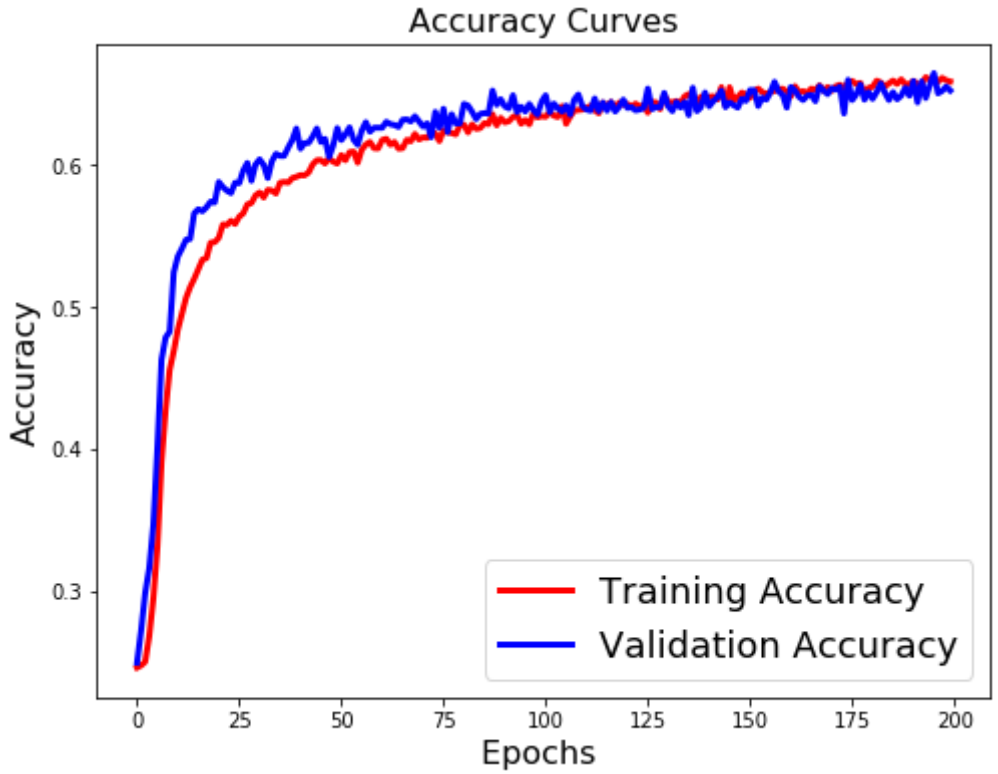


Figure 7. Training and validation accuracy curves

4.2 Justification

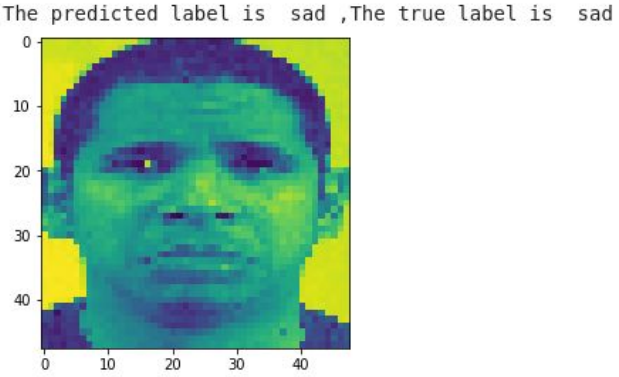
As we mentioned before, My benchmark model is a simple SVM that has an accuracy of 38.5%.
The model I created has a testing accuracy of 67.2% which is higher than the benchmark model. As we mentioned before, The testing accuracy is our main focus.
Our model seems to be able to generalize the feature detection and perform decently. I guess for this problem statement this model is a good initial solution for recognizing the facial expressions.

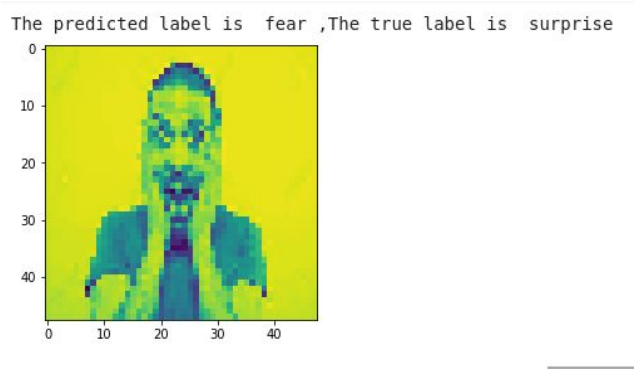
	Training accuracy	Testing accuracy
CNN	0.6553	0.6720
SVM	0.3770	0.3683

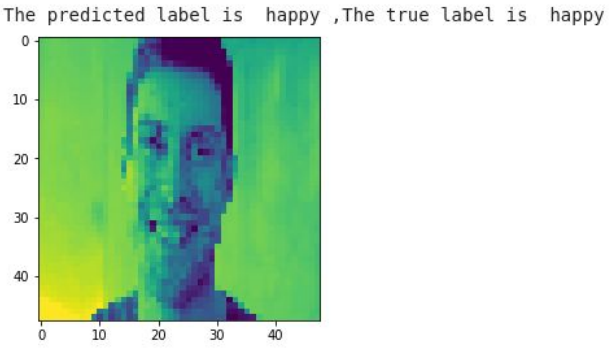
5. Conclusion

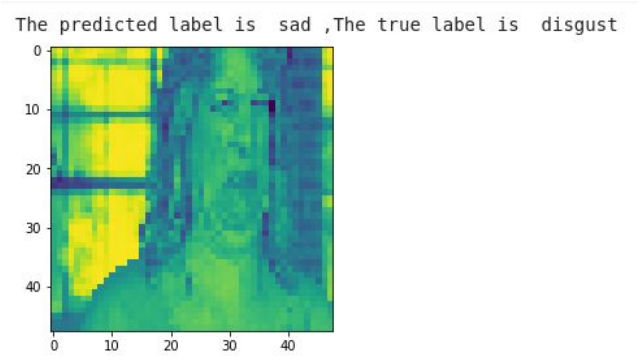
5.1 Free-Form Visualization

The final step was to predict the facial expressions of some images using the final model.









The facial emotions are actually confusable and not an easy problem. For the ‘sad’ and ‘angry’ , ‘fear’ and ‘surprise’, they are close and somebody can just got confused and could be unable to assign an exact true label. So, from the above images we can tell that the results are acceptable,and the model has performed significantly well.

5.2 Reflection

Main project steps:

1. Problem statement: An initial problem is identified and a relevant datasets were obtained.
2. Data analysis: Loading the dataset file and reading it.
3. Data preprocessing: Preprocessing dataset to meet the model requirements.
4. Dataset splitting: Splitting dataset into training set, validation set and testing set.
5. Model building: Building a CNN model and tune its parameters to achieve the best results.
6. Benchmark model: Building SVM model and training it.

Our problem was to recognize facial expression of an image. This feature can be added to our devices to expect the user's emotional state can improve human-computer interaction. In this way, it can be used in a healthcare system to detect humans' mental state and improve it by exploring their behaviour patterns.

This was a very challenging project, because the scope of it was really wide and required a lot of research, study and practice. I have used a lot of concepts acquired during the course, but also had to learn new technologies and techniques.
I particularly liked working with Convolutional Neural Networks. The concept of CNN is not an easy one to understand, however, Keras and Tensorflow help us a lot by creating a nice and easy to use API that shades the major complexities of CNN while still maintaining a high level of accuracy.
One of the most challenging parts of the project was to create the model and tune its hyperparameters. Since we're dealing with a large dataset, the training process takes much time, so trying different combinations of hyperparameters and evaluating it took so much time. After achieving a model with fairly good results and acceptable accuracy, the interesting part came. It was interesting to predict facial expressions of personal images and see its results. When it comes to our benchmark, it was interesting to evaluate it and compare its results to our model. We could prove that when it comes to image recognition problems, we should give preference to CNNs that are more specialized in that domain.

5.3 Improvements

Possible improvements that can be applied:

- We can try to use more values in hyperparameters to find better models.
- We can use transfer learning, There are many pre-trained models that could be used like VGG-16 and ResNet50.
- We can use ensemble learning with many models, It can achieve better results.
- The final model could be used as a real time application.

In future, I would like to use ensemble learning, as it gave the best results in this problem.