**CNN Emotion Classification**

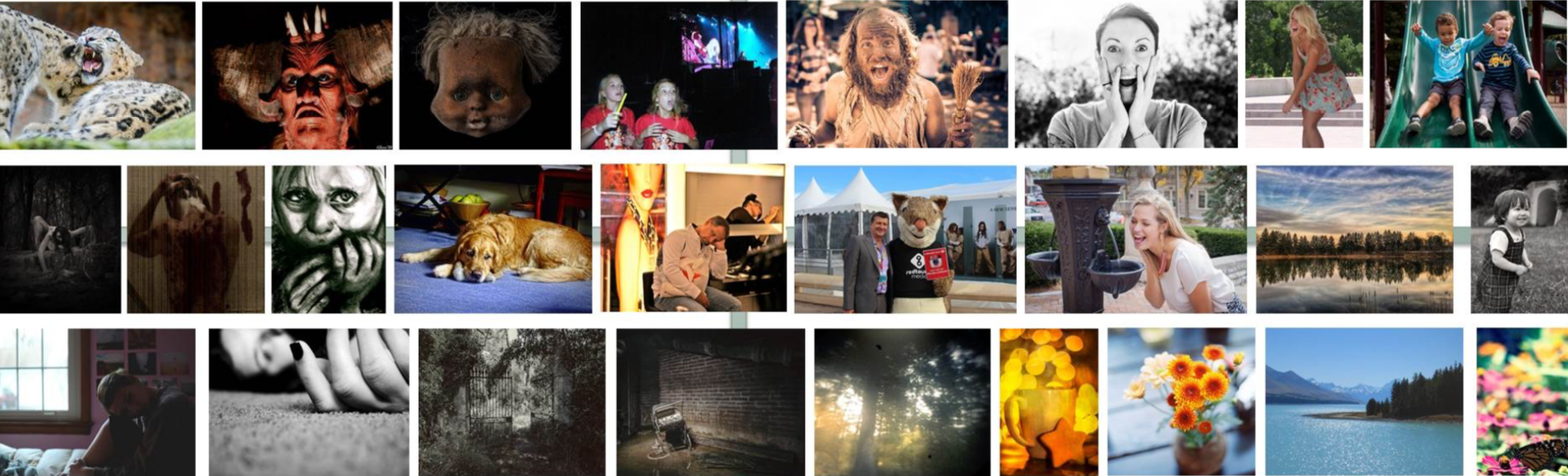
**Introduction**

Images are very powerful tools for conveying moods and emotions. Through images, people can express their feelings and communicate with other people. Many researchers have investigated in computing the image emotions by using various features extracted from images. With the recent development in the deep learning technology, CNN has been considered better at recognizing objects, faces, and actions. However, we argue that the learning process for the image emotion prediction should be different from that of image classification. This is because some images with different appearances can have same emotions, and some images with similar appearances can have different emotions.

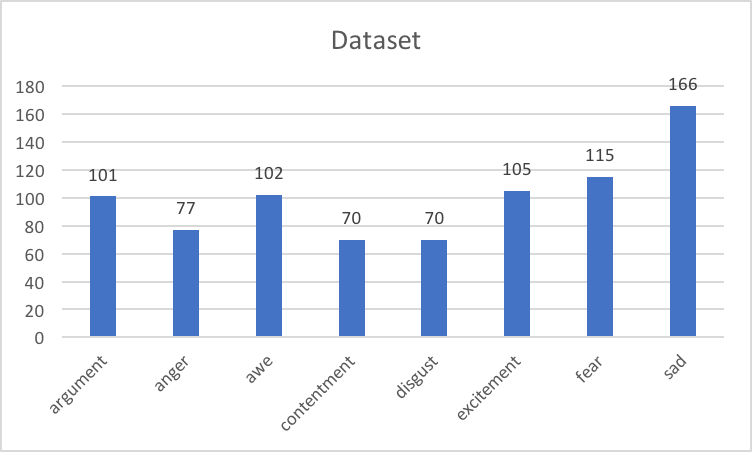
There are two different kind of features to extract for an image. Lower-level features including colors, edge, corner, blob, ridge detection and high-level feature which can only defined by human beings, including action, object and scene. For my research, I will extract both high-level and low-level feature.

**Database**

International Affective Picture System (IAPS) is being developed to provide a set of normative emotional stimuli for experimental investigations of emotion and attention. The goal is to develop a large set of standardized, emotionally-evocative, internationally-accessible, color photographs that includes contents across a wide range of semantic categories. The IAPS (pronounced eye-aps) is being developed and distributed by the Center for Emotion and Attention (CSEA) at the University of Florida.



The dataset I used contains 800 photography from different artists and the artists choose the category for the picture when uploaded. There are eight class for the whole dataset: argument, anger, awe, contentment disgust, excitement, fear and sad.



I use the Python Image Library (PIL) to preprocess and images, including resizing, graying, storing into flattened array, adding labels, normalization and shuffle dataset. And I used 80% of the dataset as training set, the left 20% to be the testing set.

**Related Work**

**TensorFlow**:

It is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. The flexible architecture allow use to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. With TensorFlow, we can easily learn and build neural network and training our dataset.

**Keras**:

It is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. There are two benefit of keras when doing research on neural networks.

1. It is user-friendly, since it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
2. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.

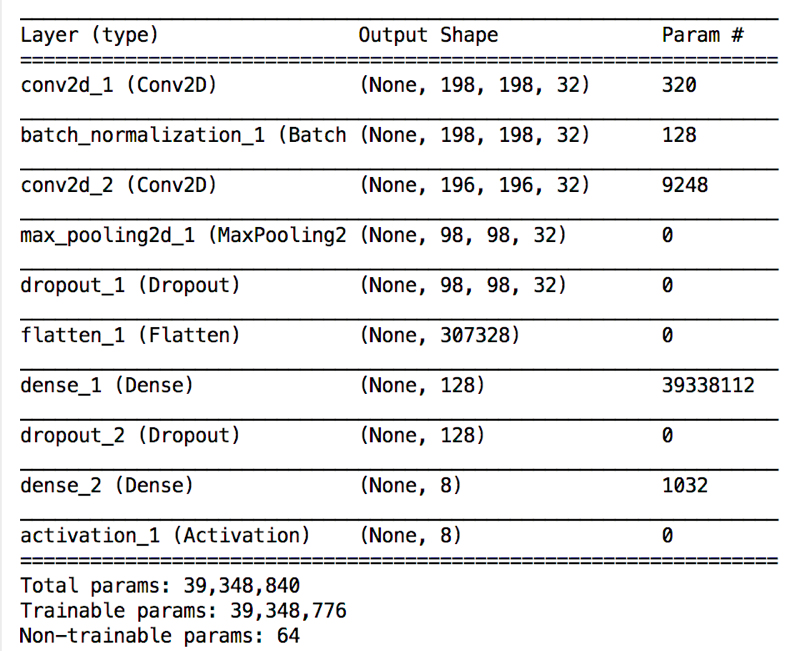
**Model**

There are several common layers that most of the CNN model will use:

1. **Convolution Layer**: The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. The CONV layer’s parameters consist of a set of learnable filters. The parameters of filter affect the learning speed and training time of the model. Using ReLu as activation function to introduce non-linearity to the system.
2. **Pooling Layer**: It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting.
3. **Dropout Layer:** In this layer, some part of the input will be randomly setting to zero to prevent the overfitting problem.
4. **Fully Connect Layer(Dense):** Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. See the *Neural Network* section of the notes for more information.
5. **Batch Normalization**: allows us to use much higher learning rates and be less careful about initialization. It also acts as a normalizer, in some cases eliminating the need for Dropout.

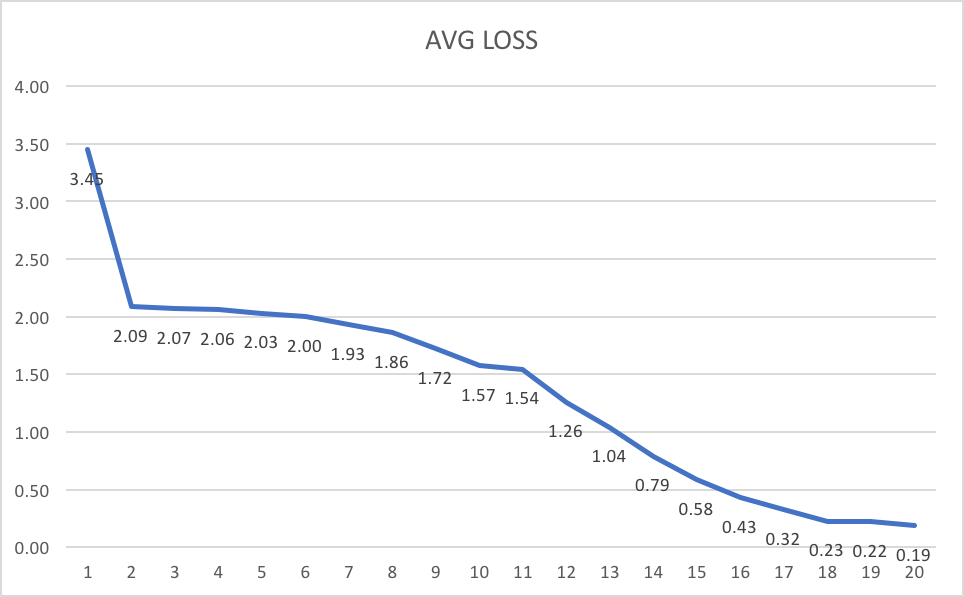
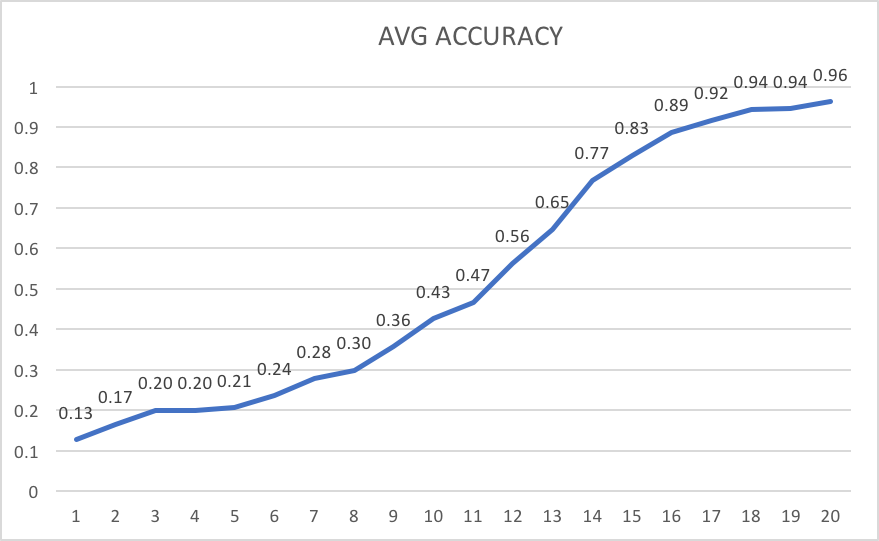
**My Model**:

I build a simple CNN model with keras as shown below. The first hidden layer is convolution layer with a 3\*3 filter, and activation function is ReLU.There are batch normalization, pooling layers and dropout layers in between, and two dense layers at the end.



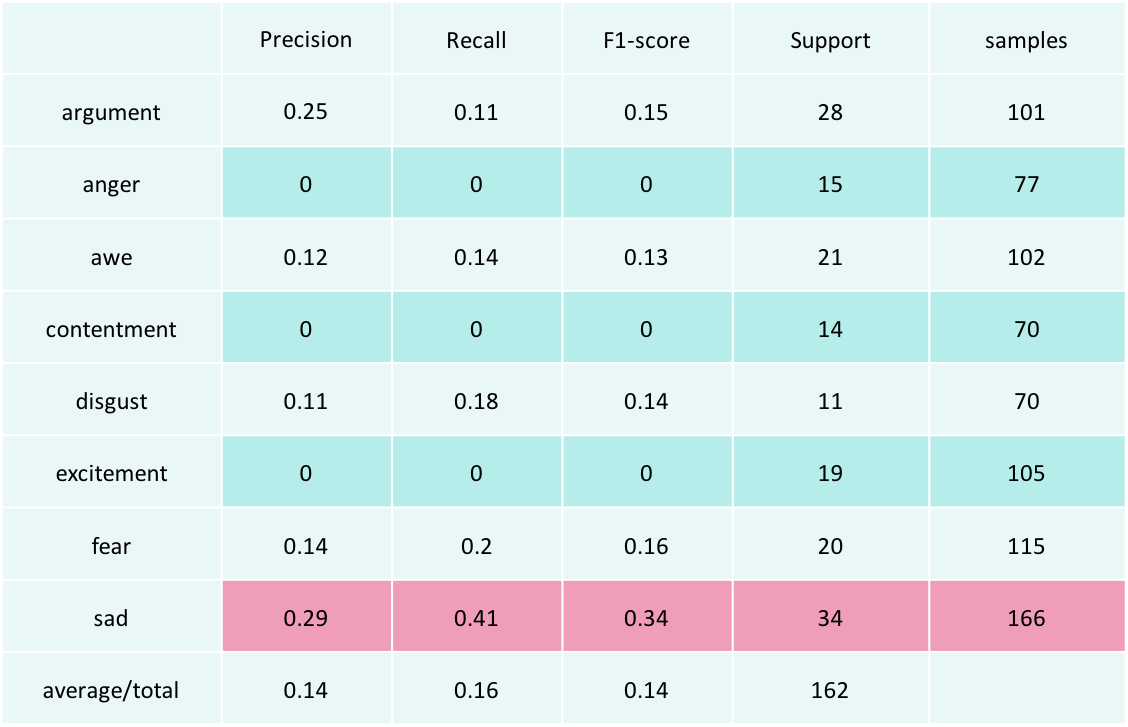
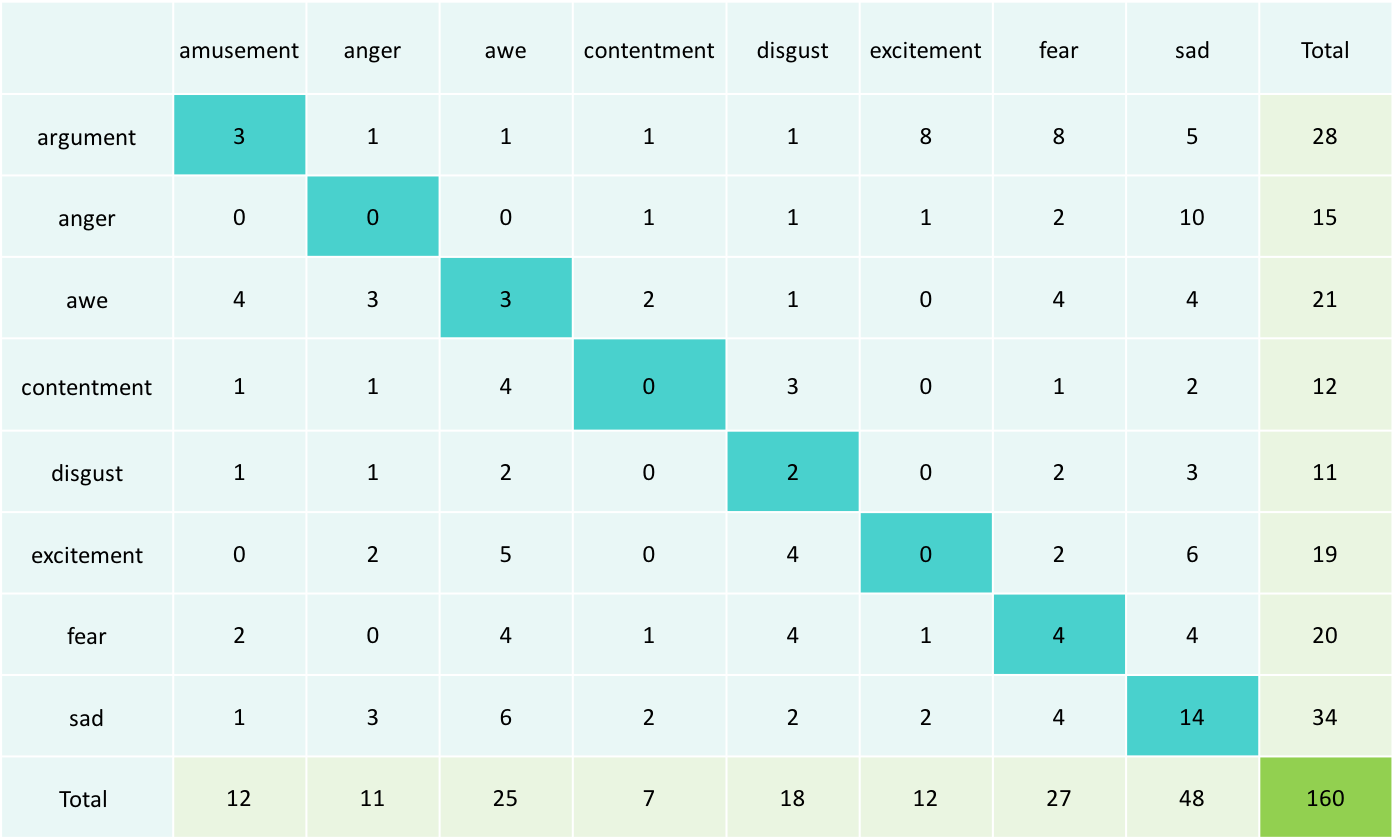
**Running Result**

The input shape of image samples is (644, 200,200,1). After running 20 epochs at first time and get the accuracy and loss values of training set as shown below:



The training process is normal, and the accuracy reach 96% at 20th epoch, and loss decrease at a normal speed, neither fast, nor too slow. So far, the model works well.

But the result of the testing set is upsetting, only 16% after 20 epochs. Here is the confusion matrix and classification report of the testing set.



As we can see from the tables, the prediction of sad class works better than other classes and reach the accuracy 40%. And for class anger, excitement and contentment, only 0 accuracy.

**Analysis**

There are several reasons for this result:

1. Dataset

In my dataset, the amount of sad class is larger than other classes, two times higher than anger, excitement and contentment classes.

In sad class images, the color is dark, which is a feature easy to capture.

The classification by human is confused. Sometimes, it is hard to which class the image is for myself. No doubt it will be hard for machine to tell the difference between those images.

Another assumption is that I didn’t shuffle the image set well, so that the three classes was not well trained.

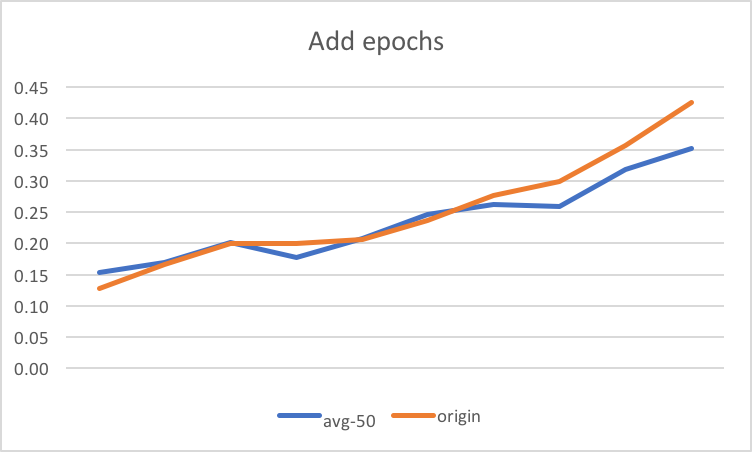
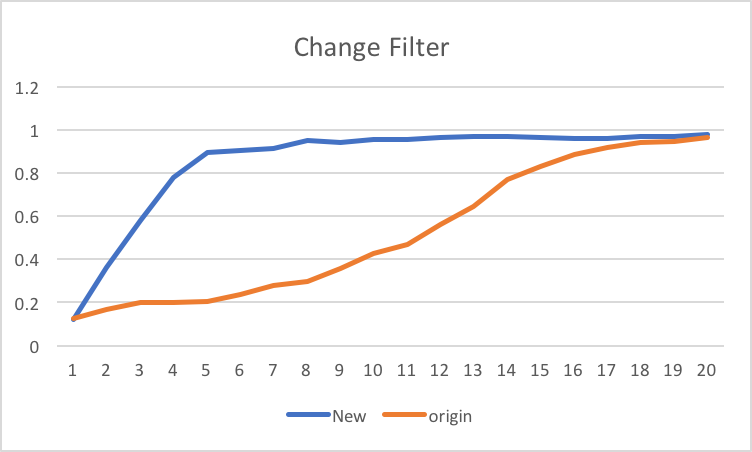
1. Model

As I shown before, my model is very simple with only 4 layers. It will be hard for the model to extract enough features to classify. I built another deep model which consist of 8 layers. But it didn’t work well and even worse than the former model, as the training accuracy only hover between 0.1 and 0.2.

1. Parameters

I also change the parameters for the filters on each layer, but it only affected the training speed until 95% accuracy. After 95%, the learning speed slow down.

Add more epochs made no big difference as well.



**Conclusion**

In my project, I presented a new emotion recognition system with a deep learning framework. The model can train the data well, reaching to 96%, but the testing accuracy only 16%. For my assumption, there are two main reasons for the problem. First, the dataset is not clear classification and easy to recognize. Second, the model is not good enough to capture all the features needed to classify, more layers are needed.

For future developing, I will apply my CNN model to fer2013 dataset, which only consist of human faces and has more than 25000 images. I will add more different layers and try different parameters.