Representation learning: Facial Expression Recognition Challenge

Abstract

The purpose of this document is to propose elements of representation learning. After describing the concepts of representation learning, we will present different approaches on the Facial Expression Recognition challenge. The following includes the description and visualization of the dataset that we experimented on, related works in the field, and a synthesis on the main approaches proposed by (Bengio et al., 2013). Then, we go into details of the two approaches that we implemented. Finally, we discuss the results of each approach.

1. Introduction

1.1. Motivation

Because of the fruitful information it carries, facial expression plays an important role in human-machine interfaces, behavioral science or medical applications. Recent advances in face detection, tracking, classification and recognition have shown great promises in its application, highlighted by data mining based techniques. In this project, we are interested in exploring representation learning techniques, as well as putting them to use in real world scenarios.

1.2. Dataset

The dataset we chose for conducting the project is from a Kaggle challenge titled Representation Learning: Facial Expression Recognition Challenge. As described from the official challenge website, the dataset consists of 48 48 pixel gray-scale images of faces. The heatmap of an example is shown in Figure 3. The faces have been detected automatically to center the face in each image. Relatively, the occupying areas of face among images are about the same.

The dataset contains 32000+ data, including a training set of 28 709 examples, as well as a public testing set and a private testing set. In our project, we combined them together to do pre-selection and simplification. The endgame of the challenge is to categorize each face with a label of facial expression in one of seven categories: Angry, Disgust, Fear, Happy, Sad, Surprise or Neutral.

Figure 1. Example of images : Angry (a), Disgust (b), Fear (e), Happy (c), Sad (f), Surprise (g), Neutral (d)

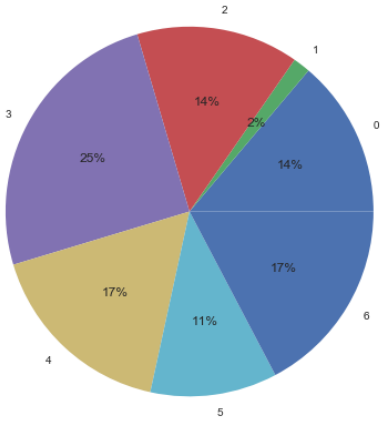

Figure 2. Class distribution : Angry (0), Disgust (1), Fear (2), Happy (3), Sad (4), Surprise (5), Neutral (6)

Figure 2 represents the distribution of the provided training set. As shown by the pie chart, the examples over all 7 classes are fairly equally homogeneous except for the 1st category which covers only 2% of the total dataset. This can be a problem in learning how to classify the 1st category, but we hope that feature learning will help compensate for the scarcity of the dataset in this particular class.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

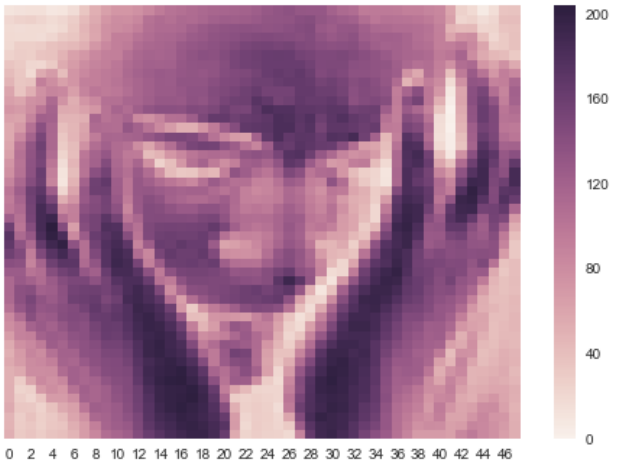

Figure 3. Heatmap of a sample in the dataset.

2. Related work

Generally speaking, the research problem of facial expression recognition can be divided into six modules (Kovac et al., 2003):

- Data acquisition
- Preprocessing
- Feature extraction
- Feature reduction
- Classification
- Post-processing

Preprocessing, feature extraction, feature reduction and classification are major steps among these 6 modules.

2.1. Preprocessing

Preprocessing generally includes illuminating equalization and face detection, dealing with the form of signal conditioning, such as noise removal, and normalization against the variation of pixel position or brightness, together with segmentation.

2.2. Feature Extraction and Reduction

Feature extraction converts pixel data into a data representation of facial components movement and distance, color and texture of face. It is a key in facial expression recognition. If features are not adequately extracted, even the best classifier model could fail to achieve accurate recognition (Chibelushi & Bourel, 2003).

Generally, there are three types of feature extraction:

- Geometric features
- Appearance features
- Hybrid features

Geometric features are facial components, including mouth, eyes, eyebrows, and nose, which are extracted to form feature vectors that represents the face geometry. Appearance features are facial appearance like skin texture, changes of the face, such as wrinkles and furrows that are used to form feature vectors to represent face appearance. Hybrid features are a combination of geometric features and appearance features.

In most cases, the process of feature extraction yields a large amount of features but only a smaller proportion of features will be selected for classification. Therefore, reducing the redundancy of the features is necessary in our study. Dictionary learning and principle component analysis (PCA) are two generally used feature reduction techniques.

2.3. Classification

Classifiers are used in the last step of facial expression analysis to classify extracted features of an image into particular category of expression. Many classifiers have been proposed for facial expression recognition such as neural network (NN), support vector machines (SVM), linear discriminant analysis (LDA), K-nearest neighbor, multinomial logistic ridge regression (MLR), hidden Markov models (HMM), tree augmented naive Bayes, among others (Chibelushi & Bourel, 2003; Shlens, 2014). Depending on whether temporal information across image frames is used, these methods can be divided into two categories: frame-based and sequence-based approaches.

3. Representation Learning

Representation learning has become an important field in the machine learning community, notably in *Deep Learning* or *Feature Learning*. It has shown remarkable successes both the academic community and in industry. Some are its applications are Speech Recognition, Signal Processing, Object Recognition, Natural Language Processing, Multi-Task and Transfer Learning and even Domain Adaptation.

As for object recognition on which our project is based, deep learning(Hinton et al., 2006)(Bengio et al., 2007) has long since surpassed the state-of-the-art SVMs (1.4% error) on the MNIST digit image classification problem.

Presently, the latest records are still held by deep networks: 0.27% error for the unconstrained version(Ciresan et al., 2012) of the task, and 0.81% error for the knowledge-free version(Rifai et al., 2011) where no image-specific prior is used (such as image deformations or convolutions).

3.1. Representation Learning Strength

Representation learning can be convenient to express many general priors about the dataset which are not necessarily task-specific but still helpful for training an efficient machine learning model. Examples of such general-purpose priors are the following:

- **Smoothness:** the most basic prior which assumes the function to be learned f such that if $x \approx y$ then $f(x) \approx f(y)$. However, this suffers from the problems of dimensionality as it assumes the target function is smooth enough, which is not necessarily true as the number of wrinkles (ups and downs of the target function) may grow exponentially with the number of relevant interacting factors in raw input space.
- **Multiple explanatory factors:** this is based on the assumption that the data is generated by different underlying features, and mostly a learned feature can be generalized to other features as well. Therefore, the objective is to recover these underlying features. This is the basic idea behind distributed representations where a reasonably-sized learned representation can capture a huge number of possible input configurations, which outperform classic algorithms using one-hot representations such as KNN and SVMs.
- **A hierarchical organization of explanatory factors:** the features can be redefined in terms of abstract concepts, which is mostly used in deep representations. However, this method, albeit its success, is often difficult to understand and implement effectively.
- **Semi-supervised learning:** representations that are useful in generating the input space can also be helpful when learning to predict the output space by sharing the statistical strength between the unsupervised and supervised learning tasks.
- **Shared factors across tasks:** features learned in a domain can be transfered to another resource-poor domain to help boost its learning. This is the main idea behind Multi-Task, Transfer Learning and Domain Adaptation.
- **Manifolds:** probability mass concentrates near regions that have a much smaller dimensionality than the original space where the data lives. This is explicitly exploited in some of the auto-encoder algorithms and other manifold-inspired algorithms.
- **Natural clustering:** different values of categorical variables such as object classes are associated with separate manifolds. This is based on the idea that humans have named categories and classes due to a statistical structure discovered by their brain and propagated by their nature, and machine learning tasks often involves in predicting such categorical variables.
- **Temporal and spatial coherence:** this representation technique assumes that the data changes overtime, so the associated representations have to change as well to adapt by penalizing changes in values over time or space.
- **Sparsity:** not all features provided in dataset are relevant. Therefore, only prominent features that are invariant to small changes of the input data should be extracted and used for learning. This can be achieved by putting a threshold on the latent variables or by using a non-linearity whose value is flat at 0.
- **Simplicity of Factor Dependencies:** not all features are independent to one another. In fact, most features are codependent in some ways that can be exploited to form better representations of the dataset.

3.2. Approaches

Three types different major approaches are discussed on the paper of Yoshua Bengio et al. (2012).

Probabilistic models. From the probabilistic modeling perspective, the question of feature learning can be interpreted as an attempt to recover a small set of latent or hidden random variables that describe a distribution over the observed data. Let $p(x, h)$ be a probabilistic model over the joint space of the latent variables, noted h, and observed data or visible variables x. Simply put, learning is achieved by estimating a set of model parameters which maximizes the regularized likelihood (probability) of the training data. In this regard, the probabilistic graphical model formalism provides two modeling paradigms: directed and undirected graphical models.

Directed graphical models separately parametrize the conditional likelihood $p(x|h)$ and the *prior* (latent variables) $p(h)$ to construct the joint distribution, $p(x, h) = p(x|h)p(h)$. An example of this decomposition is Principal Components Analysis (PCA).

Undirected graphical models, also called Markov random fields (MRFs), parametrize the joint p(x,h) through a product of unnormalized non-negative *clique potentials*:

$$p(x, h) = \frac{1}{Z_\theta} \prod_i \psi_i(x) \prod_j \eta_j(h) \prod_k \nu_k(x, h)$$

where $\psi_i(x)$, $\eta_j(h)$ and $\nu_k(x,h)$ are the clique potentials describing respectively the interactions between the visible elements, between the hidden variables (prior), and the interactions between the visible an hidden variables. The partition function Z_θ is for the distribution normalization. A particular form of this model is a called a Boltzmann distribution of which the most popular subclass is Restricted Boltzmann Machines (RBMs). This is discussed in finer details in the paper.

Reconstruction-based algorithms. The use of latent variables can become very complicated and intractable if the model (either directed or undirected graphical model) has more than a couple of interconnected layers. An alternative non-probabilistic approach is feature learning via a direct encoding, i.e., a parametric map from inputs to their representation. A method of this type is Auto-Encoders, notably Regularized Auto-Encoders.

In the auto-encoder framework, features are extracted and encoded with a specific parametrized closed form function called the **encoder**, and another closed form parametrized function called the **decoder** maps from feature space back into input space, producing a **reconstruction**. The reconstruction error is the difference between the actual input space and its reconstruction. Therefore, the optimization task consists in finding the set of parameters θ which minimize the reconstruction error.

Regularized Auto-Encoders further improve the conventional Auto-Encoder by constraining the representation. The effect of this regularization is that the auto-encoder cannot reconstruct well everything, it is trained to reconstruct well the training examples and generalization means that reconstruction error is also small on test examples. Another possible objective of the regularization is making the representation as stable as possible with respect to changes in input. There are two variants of regularized auto-encoders: *contractive auto-encoders* which reduce the number of effective degrees of freedom of the representation (around each point), and *denoising auto-encoder* which makes the whole mapping robust (insensitive to small random perturbations).

Geometry motivated manifold-learning approaches. Another important perspective on representation learning is based on the geometric notion of manifold. Its premise is the manifold hypothesis, according to which real-world data presented in high dimensional spaces are expected to concentrate in the vicinity of a manifold $\mathcal{M}$ of much lower dimensionality $d_{\mathcal{M}}$, embedded in high dimensional input space $\mathbb{R}^{d_x}$. This prior seems is particularly well suited for AI tasks involving images, sound or text whose sampled input configurations are unlike natural stimuli.

Connections between different approaches. While the three approaches are apparently disconnected, learning to draw connections between them can lead to new combined models that take advantage of the relative strengths of each paradigm. This is currently a very active area of research.

4. Proposed Approach

When we try to classify image from raw pixel, we get about 0.05 accuracy with an SVM (RBF kernel). Based on section 3, we propose two approaches to improve the performance: a standard approach which uses the Gabor filter and PCA (Probabilistic model), and a second approach based on neural networks (Reconstruction-based algorithm).

4.1. Standard Approach

We propose a method which deploys a series of techniques in 3 majors steps : feature extraction, dimensionality reduction and classification.

Feature extraction: We do feature extraction to select information from the original pixel data by choosing the Gabor filter as a holistic feature in our approach. Gabor filter is a gaussian kernel modulated by sinusoidal wave in the spatial domain.
For empirical reasons, we select 32 different Gabor filter banks with 4 scales and 8 orientations. Figure 4 shows the images of the 32 Gabor banks. Then, we apply the 32 Gabor filters to the image, and get the responses from each bank. Figure 5 shows the responses for an image.

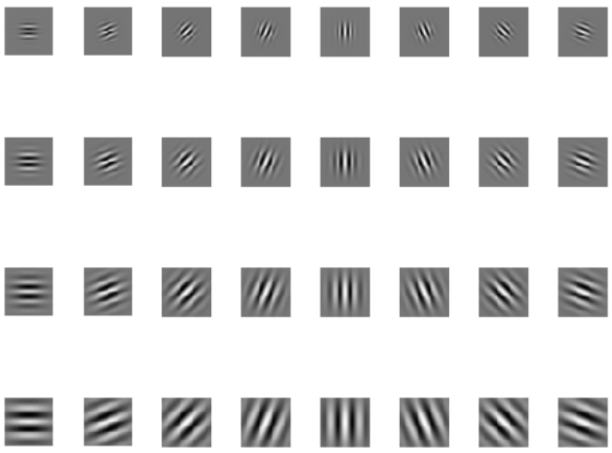

Figure 4. 32 Gabor banks with 4 scales and 8 orientations

Feature reduction: After feature extraction, the dimension of each image is transformed from 48 x 48 to 48 x 48 x 32, which amounts to 73 728 pixels/features per image. Such a large feature dimension is not suitable for direct classification. The reasons are in three-folds. First, large feature dimension incurs large computational complexity and

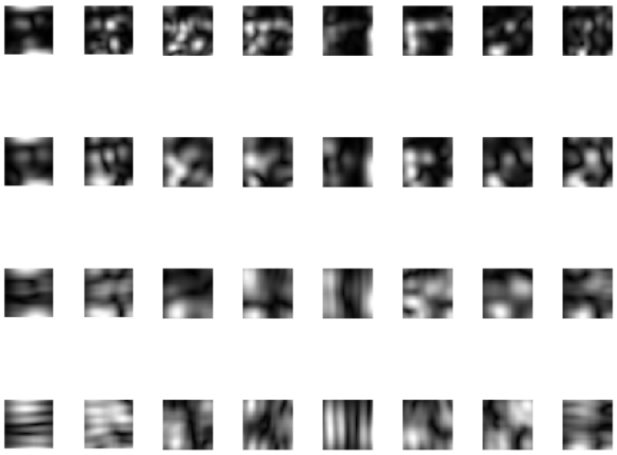

Figure 5. Responses after applying each Gabor bank

requires more computational resources for later classification. Second, to avoid over-fitting problem, large feature dimension requires large dataset to provide enough information in each dimension. Third, for classification models which rely on the calculation of distances, the distance between two samples with large dimension feature is not meaningful.

If the feature dimension for an example is huge, when there is a subtle difference between two examples, the distance would differ dramatically, and thus machine learning algorithms (SVM) would perform poorly in classification. This phenomenon is called the curse of dimensionality.

Because of these reasons, we propose to do feature reduction on the features created by Gabor filter : **Principal component analysis** (PCA). We directly apply PCA on the 48 x 48 x 32 feature dimension generated from Gabor filter, and try different numbers of components selected for classification. With this, the features would be reduced to n dimensions (parameter to be defined).

Classification: We will use SVM for classification based on the reduced features.

4.2. Neural Networks Approach

In the standard method, the most important part is feature extraction. We have to find very good features of the images to represent them. However, using neural networks, the features can be learned and extracted automatically. The inputs of the networks are the raw pixel data of each image. The PCA method is very good, but it can not be stacked into more abstract representations which can be done easily with neural networks.

In this approach, we develop two neural networks, both of them are Convolutional Neural Networks. One is a simple CNN without many layers, the other is a deep CNN. CNN has been proved to have a very good performance in image detections. We consider the structure as following:

input-M convolutional layers-N fully connected layers-softmax-output

The first part refers to M convolutional layers that can posses spatial batch normalization. The second part refers to N fully connected layers. Finally we link these two parts together to get our final CNN.

Simple CNN: The simple CNN is the first model with only 3 layers: 2 convolutional layers and 1 fully connected layer.

Deep CNN: To improve the performance, we use Deep CNN with 4 convolutional layers and 2 fully connected layers.

5. Experiments and Evaluation

5.1. Gabor filter + PCA + SVM

Based on the dataset gained from the Gabor filter, we apply PCA for feature reduction by setting the number of principal component n from 2 to 28. The accuracy in function of n is shown in Figure 6.

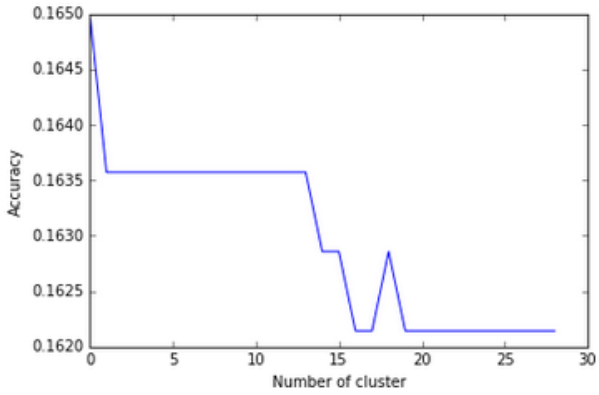

Figure 6. The accuracy of SVM after Gabor filter and PCA

We observe that the accuracy lies between 0.16 to 0.165, which is considerably low. When $n = 2$, we get the best accuracy of 0.165.

In fact, if we limit ourself to only two classes (Angry and Happy), we get an accuracy of up to 0.89. The two classes are considerably different which facilitate the classification task. We can conclude that the Gabor filter-based method provide poor feature representation on facial expression recognition task.

5.2. CNN

As mentioned above, we implement two kinds of CNNs, of which one is a simple CNN and another is a deep CNN. As expected, the results show that the deep CNN performs much better than the simple one. In this experiment, we use Keras and Theano as the backend to create our models. Also, to accelerate the computation, we make use of the GPU power made available with Theano.

Simple CNN: In the first convolutional layer, we have 32 3x3 filters, with the stride of size 1, along with batch normalization and dropout, but without max-pooling. In the second convolutional layer, we have 64 3x3 filters, with the stride of size 1, along with batch normalization and dropout also with max-pooling with a filter size 2x2. In the fully connected layer, we have a hidden layer with 512 neurons and softmax as the loss function. In all layers, we used relu as the activation function. For the training process, we used all of the images in the training set with 30 epochs and batch size of 128 and cross-validated different values of the number of hidden neurons. The final test result is 55%. We can see that after 5 epochs, the validation accuracy starts to become stable.

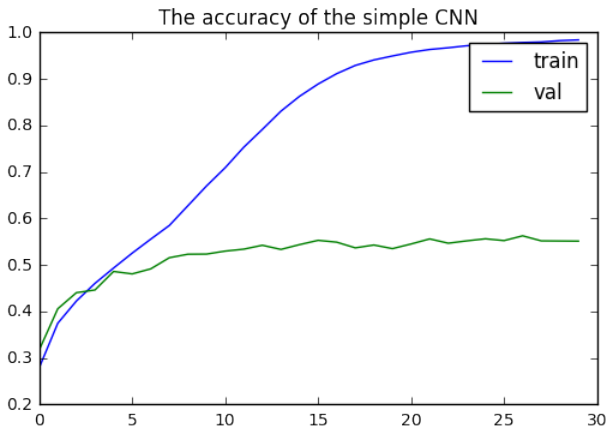

Figure 7. The accuracy of simple CNN

Deep CNN: In the first convolutional layer, we have 64 3x3 filters, the second one has 128 5x5 filters, the third one has 512 3x3 filters and the last one had 512 3x3 filters. In all the convolutional layers, we have a stride of size 1, batch normalization, dropout, max-pooling and relu as the activation function. The hidden layer in the first fully connected layers has 256 neurons and the second fully connected layer has 512 neurons. In both fully connected layers, same as in the convolutional layers, we use batch normalization, dropout and relu. Softmax is our loss funtion. In the training process, we use 35 epochs and a batch size of 128 and the input is all the images' raw data in the dataset. The final result this structure gives an accuracy of 64% on the testing set, which is much better than the simple CNN. We can also see from the Figure 8 and Figure 9 that the validation accuracy becomes stable after 10 epochs.

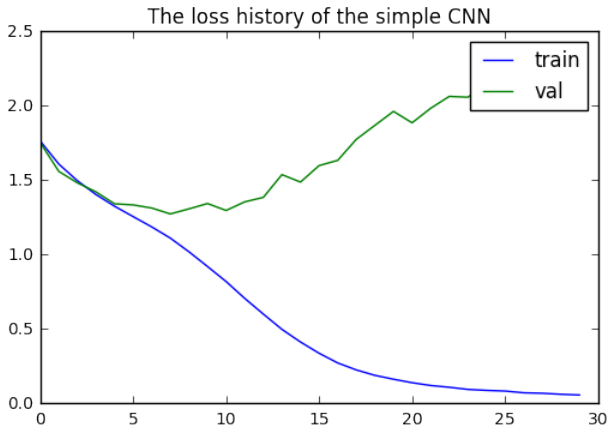

Figure 8. The loss of simple CNN

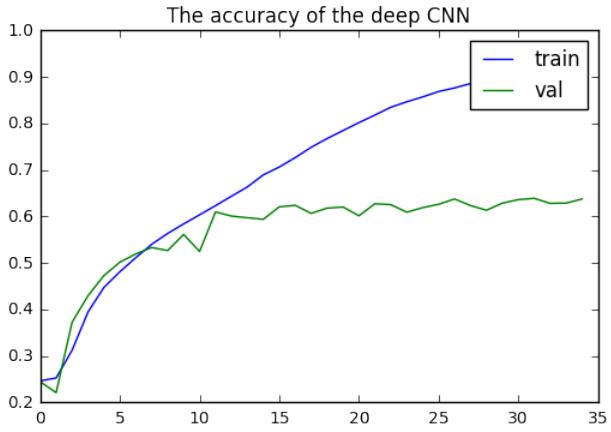

Figure 9. The accuracy of the deep CNN

Comparing the two CNNs from the figures, we can observe that the deep CNN has reduced over-fitting by adding more non-linearity usage of anti-over-fitting techniques like dropout. We try to increase the number of convolutional layers to 5, 6 and 7, but they do not perform any better than with 4 layers.

We also compute the accuracy of each model for each expression category, which is shown in Table 1. We can see from the table that classification in the Happy category has the highest accuracy score. Furthermore, deep CNN gains higher accuracy than the simple CNN on each category except the Fear category which has the same accuracy in both models. It means that for the Fear category, using a deeper structure does not yield a better performance.

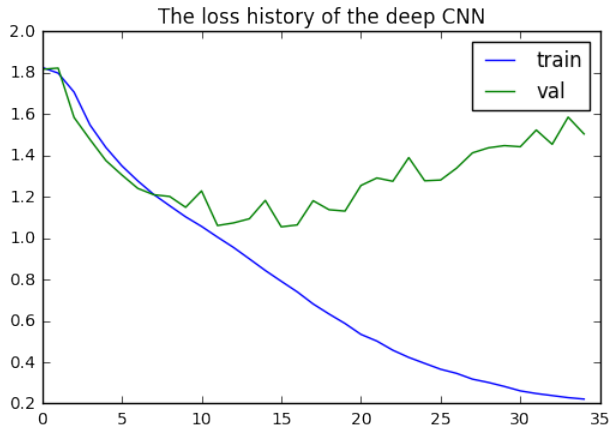

Figure 10. The loss of the deep CNN

Expression	Simple CNN	Deep CNN
Angry	0.45	0.56
Disgust	0.45	0.54
Fear	0.41	0.41
Happy	0.72	0.81
Sad	0.39	0.57
Surprise	0.74	0.79
Neutral	0.50	0.56

Table 1. The accuracy of each expression in the simple and deep CNNs

6. Conclusion

In this project, we tackle the challenge of facial expression recognition by using representation learning. We started with our motivation for this project and by doing some data visualization on the dataset to get an idea of what we are going to work on. Next, we discussed related works which consist of preprocessing raw pixel data, extracting image features, doing feature reduction and applying classifier. Then we tried two different methods on this project, one is the standard method Gabor filter + PCA + SVM, the other is CNN. From the experiments, we can conclude that CNN performs much better than the standard method. The results also show that deep CNN are capable of learning facial characteristics and improving facial expression detection. The most important point is that CNN can learn the features only using the raw data which the standard method can't.

However, the results also show that we're faced with a problem of over-fitting during training. This can be explained by the fact that our proposed CNN models are not complex enough to capture hidden information which is necessary for predicting unknown data, and they instead focus on learning (too much) from the current dataset. In the future, we would like to introduce more complex structures to solve the over-fitting problem and make a better feature representation.

References

Bengio, Yoshua, Lamblin, Pascal, Popovici, Dan, and Larochelle, Hugo. Greedy layer-wise training of deep networks. In Schölkopf, P. B., Platt, J. C., and Hoffman, T. (eds.), *Advances in Neural Information Processing Systems 19*, pp. 153–160. MIT Press, 2007. URL https://dl.acm.org/citation.cfm?id=2976476.

Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.50. URL http://dx.doi.org/10.1109/TPAMI.2013.50.

Chibelushi, Claude C. and Bourel, Fabrice. Facial expression recognition: A brief tutorial overview. 2003. URL http://lyle.smu.edu/~mhd/8331f06/CCC.pdf.

Ciresan, Dan C., Meier, Ueli, and Schmidhuber, Jürgen. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012. URL http://arxiv.org/abs/1202.2745.

Hinton, Geoffrey E., Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL http://dx.doi.org/10.1162/neco.2006.18.7.1527.

Kovac, J., Peer, P., and Solina, F. Human skin color clustering for face detection. 2:144–148 vol.2, Sept 2003. doi: 10.1109/EURCON.2003.1248169.

Rifai, Salah, Dauphin, Yann, Vincent, Pascal, Bengio, Yoshua, and Muller, Xavier. The manifold tangent classifier. In Shawe-Taylor, John, Zemel, Richard S., Bartlett, Peter L., Pereira, Fernando C. N., and Weinberger, Kilian Q. (eds.), *NIPS*, pp. 2294–2302, 2011. URL http://dblp.uni-trier.de/db/conf/nips/nips2011.html#RifaiDVBM11.

Shlens, Jonathon. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014. URL http://arxiv.org/abs/1404.1100.