Logistic Regression and Softmax Regression via Gradient Descent for Emotion Recognition with PCA

Yongchuang Huang
PID: A53322973
yoh001@eng.ucsd.edu

Jingpei Lu
PID: A91033661
jil360@eng.ucsd.edu

Yiran Xu
PID: A53270264
y5xu@eng.ucsd.edu

Abstract

In this report, we use Logistic Regression and Softmax Regression both via Gradient Descent to recognize human emotions. Before implementing each model, Principal Component Analysis (PCA) is used for each image. For Logistic Regression, we achieved the 97.8% (0.044) accuracy on Happiness vs Anger using **aligned** data and 84.0% (0.120) accuracy on Fear vs Surprise; for Softmax Regression, we achieved 63.3%(0.123) average accuracy for all emotions on test dataset by training on 20 principal components.

1 Introduction

Human emotion recognition can be considered as an image classification problem. If the emotion only contains two categories, it reduces to a binary classification problem and can be solved by Logistic Regression. If the emotion contains more than two categories, however, Softmax Regression is able to solve the problem. These two models are both derived from Linear Classifiers. In order to estimate the parameters of the classifier, i.e. the weights, Gradient Descent is one simple and efficient approach.

However, as the size of datasets increases, a single batch gradient descent handling all the data might cause memory issue. Stochastic Gradient Descent (SGD) deals with one data point at one time, which mitigates the memory load caused by big datasets.

Finally, images are instinctively high-dimension data. For instance, a low-resolution image with a 200×200 size is actually a vector with a size of 40,000. This requires more data for training. To reduce the dimensionality, Eigenfaces with PCA [1] can find the most principal components of the data by emphasizing variation. By picking a few most principal components, we can reduce the dimensionality of the data and also reduce the requirement for the memory.

2 Methods

2.1 PCA for data pre-processing

Applying PCA method is to decompose the training set, obtain its principal components and finally reconstruct the data set with these components. Since the inputs are face images, i.e. N_1-by-N_2 two-dimensional arrays, there is a need to reconstruct these arrays into an $(N_1 \cdot N_2)$-by-M matrix called the training set, where M is the number of images as training samples. Let the training set be

$[\Gamma_1, \Gamma_2, ..., \Gamma_M]$ and Φ denote the centered data [1], which are related by:

$$\Phi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{1}$$

The covariance matrix, denoted by C, is AA^T, where $A = [\Gamma_1 - \Phi, \Gamma_2 - \Phi, ..., \Gamma_M - \Phi]$. The eigenfaces $u_i,\ i = 1, 2, ..., M$ is just a set of M orthonormal eigenvectors of C associated with non-zero eigenvalues. Note that $C \in \mathbb{R}^{(N_1 \cdot N_2)\times(N_1 \cdot N_2)}$, $N_1 \cdot N_2 > M$. Instead of computing C directly, in [1], an efficient way to compute the eigenvectors is done by the following:

$$A^T A v_i = \mu_i v_i \tag{2}$$

where v_i is an eigenvector for $A^T A$ and μ_i is its corresponding eigenvalue. Obviously, there are only M eigenvectors obtained from Eq. 2. Based on this equation,

$$AA^T A v_i = A\mu_i v_i \Longrightarrow C \cdot (Av_i) = \mu_i \cdot (Av_i) \tag{3}$$

So the vector Av_i is an eigenvector of the matrix C and its corresponding eigenvalue remains to μ_i. Before reconstructing the data set, it is a need to normalize these vector $Av_i,\ i = 1, 2, ..., M$, that is,

$$u_i = \frac{Av_i}{\|Av_i\|} \tag{4}$$

Finally, the follow equation is to project the training images to the k eigenfaces associated with corresponding k largest eigenvalues and $k < M$

$$\hat{\Gamma}_j = \sum_{i=1}^{k} u_i^T(\Gamma_i - \Phi)u_i + \Phi \quad j = 1, 2, ..., M \tag{5}$$

2.2 *k*-fold Cross Validation

k-Fold Cross Validation is a re-sampling procedure used to evaluate learning models on limited data samples. The main idea is to split the data into k folds and iterate different sets for validation and then average the performance over them all. The general procedure of applying k-fold Cross Validation before training in our programming is as follow:

(a) Use a function ”balanced_sample” provided to sample a balanced dataset, i.e., make the number of images in each facial expression same;

(b) Separate the image arrays and their corresponding labels, convert the image arrays into vectors and store them into two matrices, X and y. The i-th column of X is the i-th images and its corresponding label is in the i-th column of y;

(c) Shuffle the columns of two matrices randomly in the same way;

(d) Divide k folds based on the value of k and the number of columns in X or y, that is, the total number of samples used for later training;

(e) Repeat k times: choose two of the sets to be holdout set and test set, and take the remaining folds as the training set.

2.3 Logistic Regression via Gradient Descent

Logistic Regression estimates the parameters of a logistic model. A general logistic function $p(\mathbf{x}) : \mathbb{R}^d \to (0, 1)$ can be written as:

$$y = p(\mathbf{x}) = \sigma(\mathbf{w}^\top\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^\top\mathbf{x} + w_0))}, \tag{6}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weights and w_0 is the bias. Usually, the loss function for Logistic Regression is binary Cross-Entropy:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\{t^{(n)}\text{In}(y^{(n)}) + (1 - t^{(n)})\text{In}(1 - y^{(n)})\}, \tag{7}$$

where $t^{(n)}$ represents the n-th target and N is the batch size. Since we are using Gradient Descent, by taking the derivative and using Delta Rule we have

$$w_j^{i+1} = w_j^i - \alpha \frac{\partial E(\mathbf{w})}{\partial w_j} = w_j^i + \alpha \sum_{n=1}^{N} (t^{(n)} - y^{(n)}) x_j^{(n)}, \tag{8}$$

where $x_j^{(n)}$ represents the j-th dimension of the n-th data point, α is the learning rate.

Therefore, we can estimate the parameters of the model via Eq. 8 under a learning rate α.

2.4 Softmax Regression via Gradient Descent

Softmax regression is the generalization of logistic regression for multiple classes. Given an input $x^{(n)}$, softmax regression will output a vector $y^{(n)}$, where each element, $y_k^{(n)}$ represents the probability that $x^{(n)}$ is in class k. The softmax activation function can be expressed as

$$y_k^{(n)} = \frac{exp(a_k^{(n)})}{\sum_{k'} exp(a_{k'}^{(n)})} \tag{9}$$

where $a_k^{(n)}) = \omega_k^T x^n$ is called the net input to output unit y_k. For softmax regression, we use a one hot encoding of the targets. That is, the targets are a c-dimensional vector, where the k^{th} element for example n (written $t_k^{(n)}$) is 1 if the input is from category k, and 0 otherwise.

To calculate the loss, we use cross-entropy cost function for multiple categories, which is defined as

$$E(\omega) = -\sum_n \sum_{k=1}^{c} t_k^{(n)} ln(y_k^{(n)}) \tag{10}$$

We want to minimize this cost function via gradient descent. To do so, we need to derive the gradient of the cost function with respect to the parameters. We can get

$$-\frac{\partial E(\omega)}{\partial \omega_{jk}} = (t_k^{(n)} - y_k^{(n)}) x_j^{(n)} \tag{11}$$

where ω_{jk} is the weight from the j^{th} input to the k^{th} output. Then we can use the learning rule defined in equation (8) to update the weights.

3 Results and Discussion

3.1 Setup

Data Preprocessing. To do cross validation for our models, we first split our datasets into 10 folds. In every running, we picked one fold as hold-out set and one as test set. In addition to this, we implement our models on two different datasets: **resized** and **aligned**. In **resized** dataset, each image is simply resized from the original. In **aligned** dataset, however, faces are aligned as a prepocessed step. Before starting to feed data into models, PCA is used for dimensionality reduction.

Logistic Regression. For Logistic Regression, we used Batch Gradient Descent, i.e. used all data points to update the weights in each epoch. The number of epochs is 50. The learning rate $\alpha = 0.01$ for **resized** dataset and $\alpha \in \{0.5, 0.01, 1.0 \times 10^{-10}\}$ for **aligned** dataset. For PCA, 5 most principal components are used.

Softmax Regression. For Softmax Regression, we experiment with Batch Gradient Descent and Stochastic Gradient Descent. Our implementation of the gradient descent and training procedures are strictly following the algorithms giving in the instruction. The number of epochs is 50 and the learning rate is $\alpha = 0.1$ for all experiments on Softmax Regression. For most of the experiments, 20 most principal components are used. For visualizing the weights, we experimented with both 20 principal components and 40 principal components to provide more intuition.

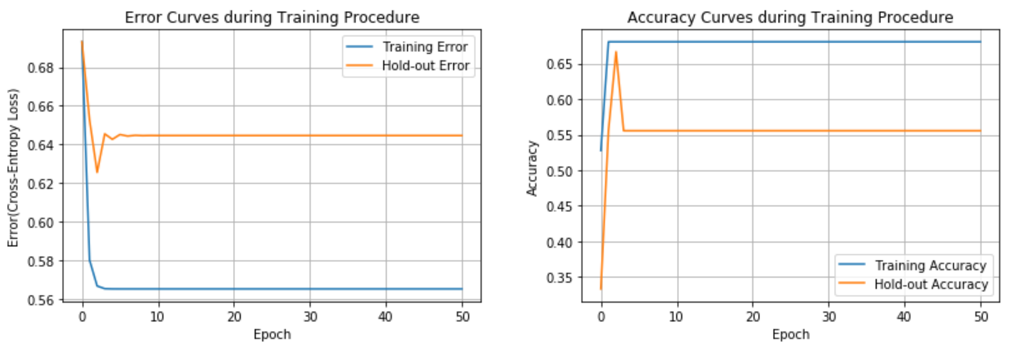


Figure 1: Error curve (left) and Accuracy curve (right) during the training procedure on **resized** data.

3.2 The Result of Logistic Regression

We report the error (loss) and accuracy on **resized** and **aligned**, respectively. For **resized** dataset, we report the error and accuracy curves of Happiness vs Anger on the training set and the hold-out set in Figure 1, as well as the final accuracy on the test set in Table 1. For **aligned** dataset, we report the results of Happiness vs Anger, and Fear vs Surprise with the best learning rate.

Table 1: Accuracy on the test sets. "H." represents "Happiness", "A." represents "Anger".

Experiments	1	2	3	4	5	6	7	8	9	10	mean	std
H. vs A. on **resized** data	0.556	-	-	-	-	-	-	-	-	-	0.556	-
H. vs A. on **aligned** ($\alpha = 0.01$)	1.000	0.889	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.889	**0.978**	**0.044**
H. vs A. on **aligned** ($\alpha = 0.5$)	0.889	0.889	1.000	0.889	1.000	1.000	0.889	1.000	0.889	0.889	0.933	0.054
H. vs A. on **aligned** ($\alpha = 10^{-10}$)	1.000	1.000	0.889	1.000	1.000	0.667	0.778	1.000	1.000	0.889	0.922	0.112
Fear vs Surprise on **aligned**	0.800	0.800	1.000	0.600	1.000	0.800	1.000	0.800	0.800	0.800	0.840	0.120

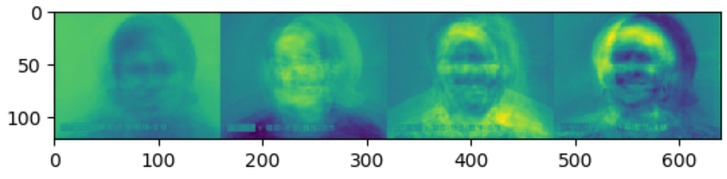
Model on Happiness vs Anger using the resized dataset. The training error and hold-out error during training procedure on **resized** data for Happiness vs Anger are shown in Figure 1. Note that the hold-out error first decreases but then rises. So does the accuracy (See Figure 1). This indicates that the model overfits after several epochs. The final accuracy on test set is 55.6%. The performance on test set is poor. This can be explained by Eigenfaces. We show the first 4 principal components in Figure 2(a). Since images are simply resized, the first 4 principal components cannot represent the faces well.

Model on Happiness vs Anger on the aligned dataset. We report the error and accuracy curves again in Figure 3 and Figure 4. Also, in order to compare the influence of different learning rates, we show the training error curves in Figure 5. The final accuracy is presented in Table 1.

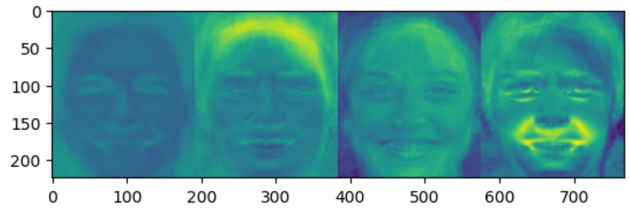
First, compared to the model using **resized** dataset, the model using **aligned** dataset achieves higher accuracy (See Table 1). This can be explained by Eigenfaces again. Shown in Figure 2, the first 4 principal components from **aligned** data focus more on the faces. Also, the 4-th principal component highlights the mouth and eyebrow section which represents the variance between "Happiness" and "Anger".

In Table 1, the best learning rate is $\alpha = 0.01$, which achieves the highest accuracy on the test set. For $\alpha = 1 \times 10^{-10}$, it leads to slow convergence for the loss (See Figure 5, the error changes slowly), and it leads to higher standard deviation compared to $\alpha = 0.01$. For $\alpha = 0.5$, although it achieves 93.3% accuracy, its loss "explodes" during the training procedure. The accuracy then is achieved only the weights in the very early time (the 1st Epoch). Thus, its accuracy on the test set is worse than $\alpha = 0.01$. This achieves an accuracy of 97.8% (0.044).

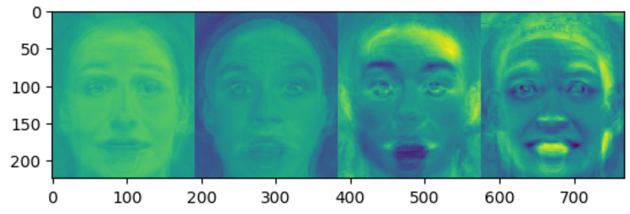
Model on Fear vs Surprise. We finally use our best learning rate, $\alpha = 0.01$, to train the model on Fear vs Surprise. The performance of the model is presented in Table 1 and the training error is shown in Figure 6. The final accuracy is 84.0% (0.120). Compared to Happiness vs Anger,



(a) First 4 principal components on **resized**: Happiness vs Anger



(b) First 4 principal components on **aligned**: Happiness vs Anger



(c) First 4 principal components on **aligned**: Fear vs Surprise

Figure 2: First 4 principal components on **resized** and **aligned**.

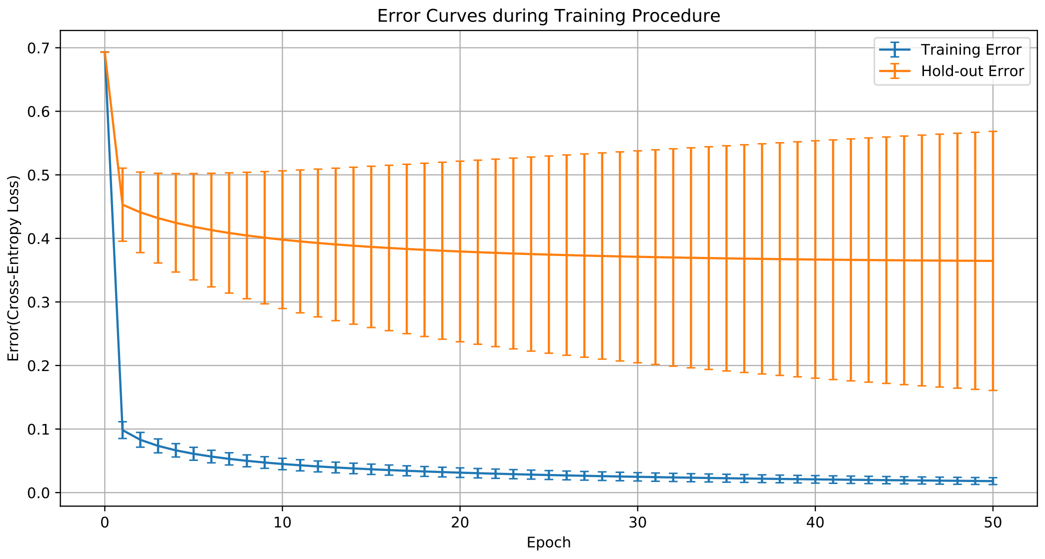


Figure 3: Training error and Hold-out error during the training procedure on **aligned** data.

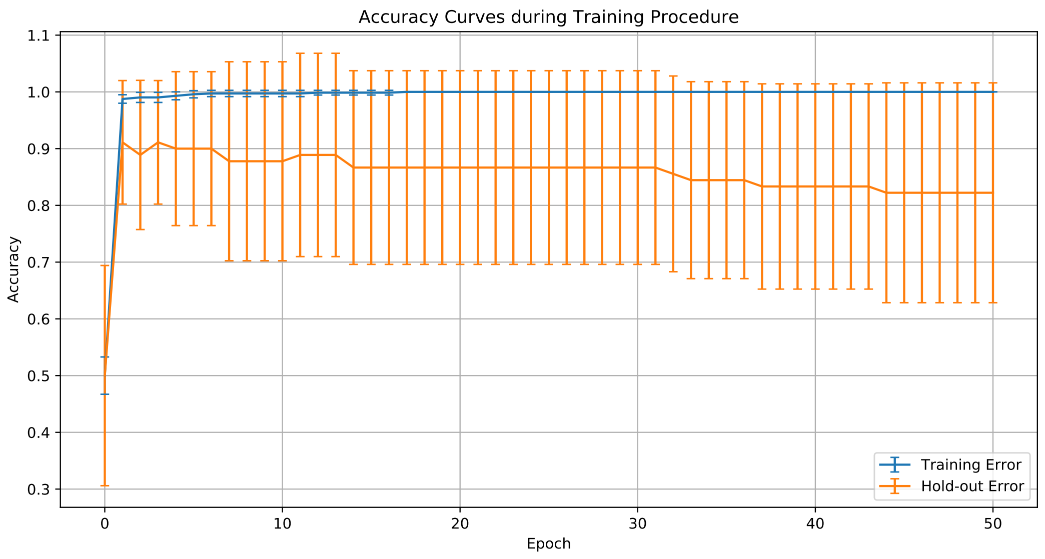


Figure 4: Training accuracy and Hold-out accuracy during the training procedure on **aligned** data.

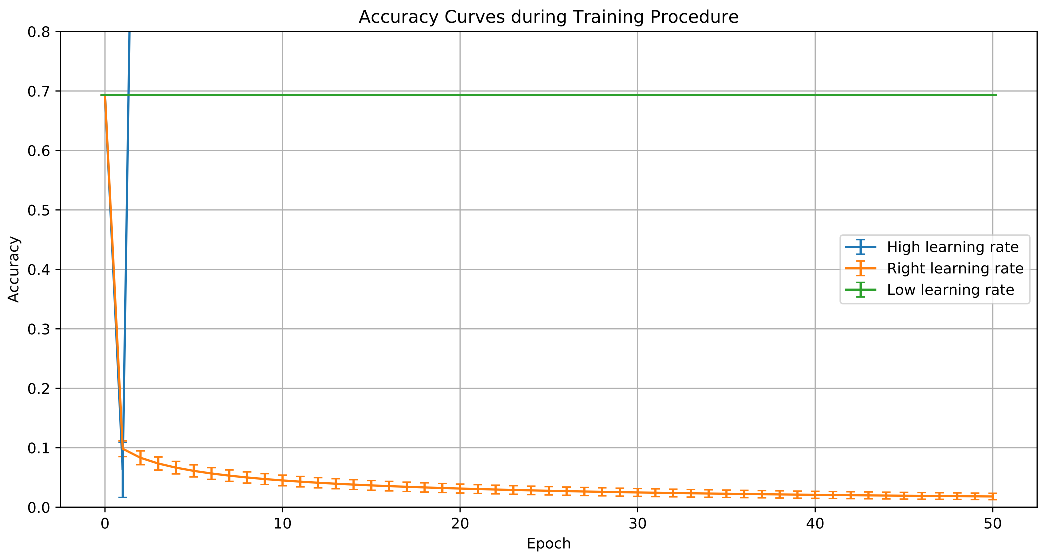


Figure 5: Three different learning rate.

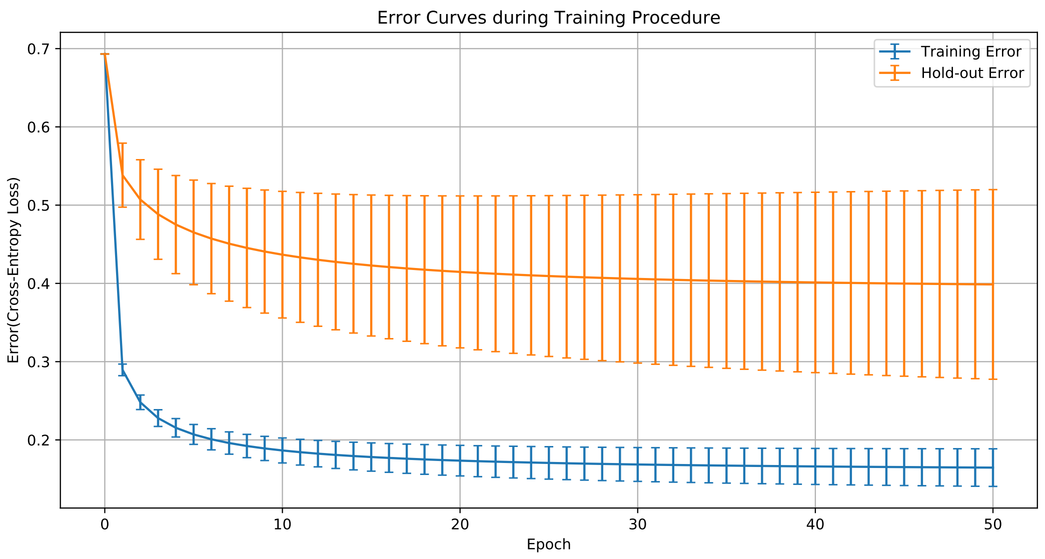


Figure 6: Training loss on Fear vs Surprise

the model on Fear vs Surprise achieves lower accuracy. This is because those two emotions are similar. In Figure 2(c), we show the first 4 principal components. Compared to Figure 2(b), the 4-th Eigenface on Fear vs Surprise highlights the mouth and eyebrow. However, it does not have a strong variance compared to Happiness vs Surprise.

3.3 The Result of Softmax Regression

Evaluate the network on all six emotions. We report the training and validation loss of the network trained using Softmax Regression with Batch Gradient Descent on Figure 7. We evaluated the model which has the smallest validation loss on the test dataset. The resulting confusion matrix is showed on the Figure 8.

Batch versus stochastic gradient descent. We experimented the Softmax Regression with both Stochastic Gradient Descent and Batch Gradient Descent. For both of experiments, the learning rate is set to 0.1. The loss curves are showed on the Figure 9. Although it is not very obvious, we can see that the network converges a little bit faster by using the Stochastic Gradient Descent. Instead of doing a big update on each epoch, Stochastic Gradient Descent updates the weights more frequently based on the loss of each sample. By doing so, the network training might be more efficient than doing the batch updates in the way such that the local minimal can be avoid.

Visualize the weights. We visualize the weights of the network trained on 20 principal components and 40 principal components. The visualizations are showed on Figure 10. We can see that by using 20 principal components, we can already see some features of different emotions. Using 40 principal components, the features of different emotions are more distinctive. That is what Softmax Regression learned from the gradient descent. The regressor adjusts its weights to learn more representative features in order to classify emotions.

Extra Credit: handle imbalanced dataset. One approach to handle the imbalanced dataset is to use resampling techniques during the training. To deal with the imbalanced dataet, we can intentionally either increasing the frequency of the minority class or decreasing the frequency of the majority class when sampling the training samples. By doing so, we can obtain a trainig dataset that is somehow balanced.

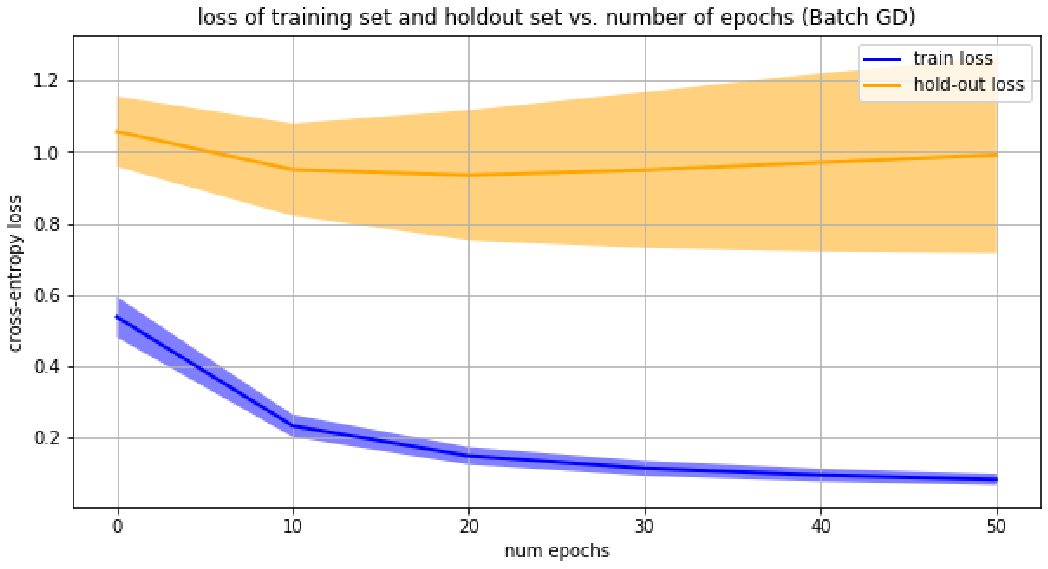
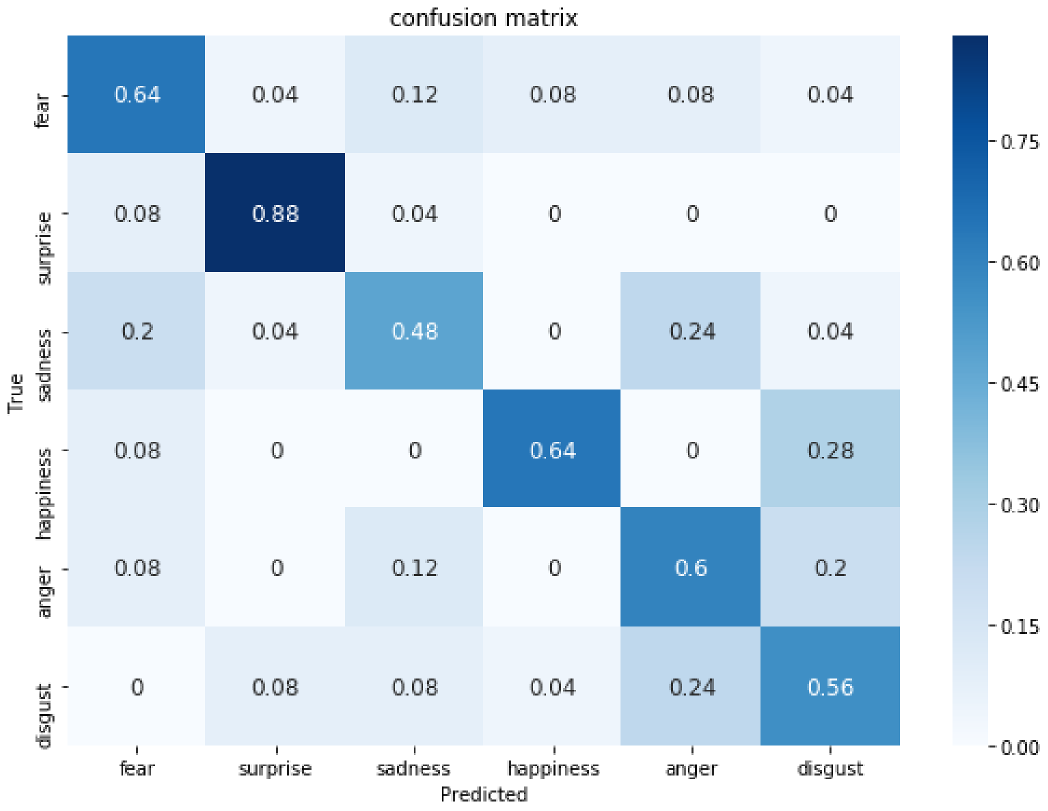


Figure 7: Loss on the training set as well as the loss on the holdout set vs. number of epochs of gradient descent. The loss is averaged over 10 runs and the shade shows the standard deviation.



True \ Predicted	fear	surprise	sadness	happiness	anger	disgust
fear	0.64	0.04	0.12	0.08	0.08	0.04
surprise	0.08	0.88	0.04	0	0	0
sadness	0.2	0.04	0.48	0	0.24	0.04
happiness	0.08	0	0	0.64	0	0.28
anger	0.08	0	0.12	0	0.6	0.2
disgust	0	0.08	0.08	0.04	0.24	0.56

Figure 8: The confusion matrix for this data based on the test set results.

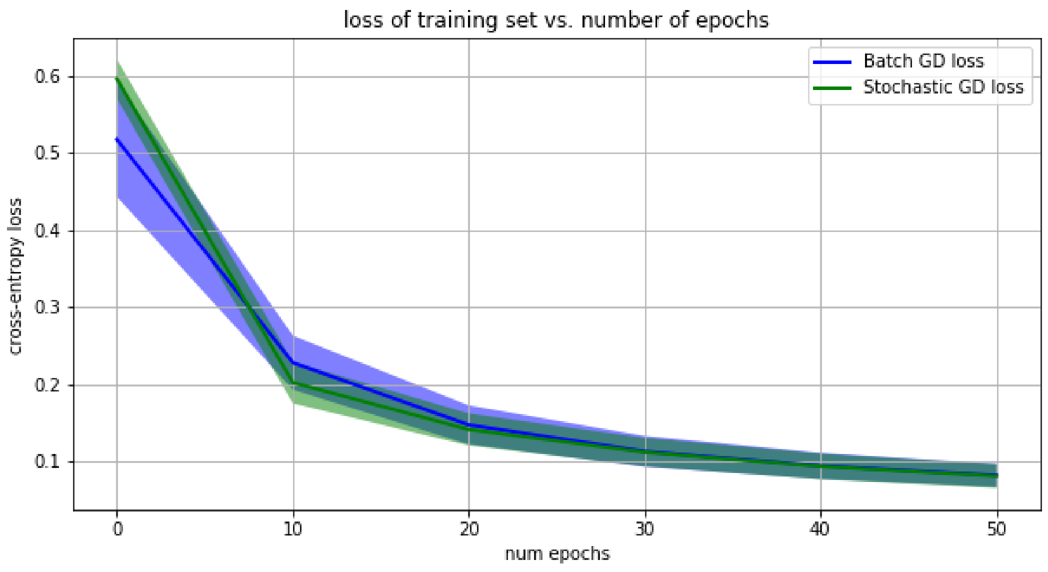


Figure 9: The training set loss over training epochs. Blue curve indicates the batch mode above, and green curve indicates stochastic gradient descent.

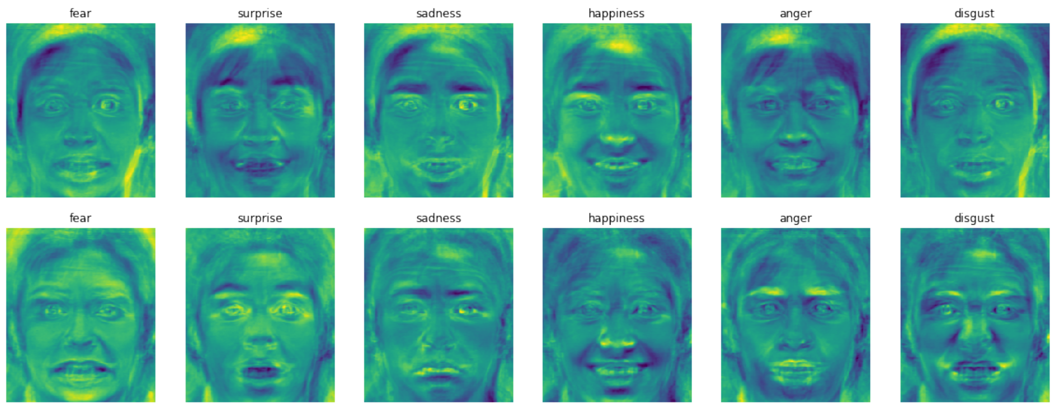


Figure 10: The visualization of the weights of the network. The first row shows the results trained on 20 principal components. The second row shows the results trained on 40 principal components.

4 Contributions

Yongchuang Huang finished the part of data preprocessing, including k-fold Cross Validation and PCA.

Yiran Xu finished the part of Logistic Regression.

Jingpei Lu finished the part of Softmax Regression.

References

[1] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1), 71-86.