Learning to Recognize Facial Expressions*

Tao Li
Dept. of Computer Science
li2953@purdue.edu

Tinghan Yang
Dept. of Computer Science
yang1683@purdue.edu

Zihan Wang
Sch. of Mechanical Engineering
wang5044@purdue.edu

ABSTRACT

We study the problem of automatic facial expression recognition using machine learning. So far, we have implemented several classic algorithms discussed in the class, including Perceptron Algorithm, Adaptive Boosting (AdaBoost), and Support Vector Machine (SVM), and evaluated their performance with two resampling techniques for validation (i.e., 10-fold cross-validation and boostrapping with 10 repetitions). Moreover, we visualized the results by presenting confusion matrices. For the rest of the time of this project, we will be exploring the algorithms under different hyperparameters and experimental settings, as well as comparing them with the current state-of-the-arts.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

machine learning, facial expression recognition, cross validation

1 INTRODUCTION

Facial expressions play a key role in human communication and social interaction. Due to its importance, automatic facial expression recognition attracts increasing attention over years [3, 4, 12, 15]. In 1970s, *Ekman and Friesen* [2] defined six basic emotions based on a cross-cultural study [1], which suggests that humans perceice the same basic emotions (i.e., anger, disgust, fear, happiness, sadness, surprise) regardless of cultural backgrounds. Fig. 1 shows examples of these basic emotions from the dataset we use in this project.

Traditionally, facial expression recognition (FER) systems use handcrafted features from still facial images (e.g., local binary patterns (LBP) [15], LBP on three orthogonal planes (LBP-TOP) [18], non-negative matrix factorization (NMF) [19], and sparse learning [20]). Recently, with the advances in computing power (e.g., popularity of GPUs) and well-designed neural network architectures, machine learning algorithms especially deep learning-based methods have achieved the state-of-the-art and outperformed traditonal methods by a large margin [8, 10, 16, 17]. Although has been studied extensively, automatic expression recognition in the wild and with a high accuracy is still a challenging task given by complexity of human faces and subtlety and variability of facial expressions. We refer [11] for a more comprehensive review.

The rest of the report is organized as follows: in Section 2 we briefly review the dataset in use; we then review serveral machine learning algorithms (as specified in the project plan) in Section 3; Section 4 details experimental settings and hyperparameters, and demonstrates the experimental results, both quantitatively and qualitatively; this preliminary report is concluded in Section 5 with discussion of future works.

*Preliminary Project Report of CS 578, Fall 2020.

Figure 1: Examples of images in the FER2013 dataset [6]. Each column illustrates one of the seven facial expressions: anger, disgust, fear, happiness, sadness, surprise, and neutral. Even under the same category, the images may vary in pose, illumination, orientation, background, and so on, which make the dataset challenging.

2 DATASET

FER-2013 data was created by Pierre Luc Carrier and Aaron Courville, by using Google image search API to search for images of faces that match a set of 184 emotion-related keywords like "blissful", "enraged", etc. Together with keywords like gender, age and ethnicity, about 600 strings were used to query facial images, and the first 1000 images return for each query were kept for the next stage of processing.

The collected images were approved by human labelers who removed incorrectly labeled images, cropped to only faces by bounding box utility of OpenCV face recognition, and resized to 48 x 48 pixels greyscale images. Then a subset of the images were chosen by Mehdi Mirza and Ian Goodfellow, and the labels(categories) of the chosen images were also mapped from the fine-grained emotion keywords.

The resulting FER-2013 dataset contains 35887 images with 7 categories in total. Specifically, there are 4953 “Anger” images, 547 “Disgust” images, 5121 “Fear” images, 8989 “Happiness” images, 6077 “Sadness” images, 4002 “Surprise” images, and 6198 “Neutral” images, with label ids ranging from 0 to 6 (see Fig. 2).

Goodfellow et al. [6] have shown that the potential label errors in FER-2013 dataset do not make the classification problem significantly harder due to the experimental result that human accuracy on a small-scale dataset with 1500 images, 7 expression categories and no label error is 63-72%, which is very close to the human accuracy for FER-2013, which is 60-70%.

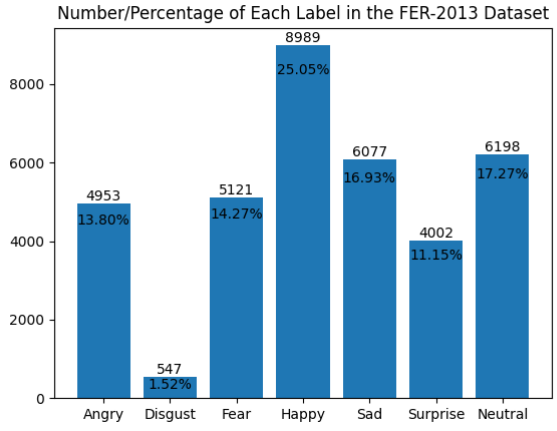


Figure 2: Distribution of labels in FER-2013 [6].

3 METHOD

In this section, we outline three classes of algorithms used in the project, i.e., Perceptron Algorithm, Adaptive Boosting (AdaBoost), and Support Vector Machine (SVM). Details of hyperparameters and experimental settings will be in Section 4.

3.1 Perceptron Algorithm

Perceptron algorithm [14] is a linear model that learns a binary classifier in a supervised manner. Formally, given input vector $\mathbf{x}$, a single-class perceptron outputs $f(\mathbf{x})$:

$$f(\mathbf{x}) = \begin{cases} 1 & \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & otherwise \end{cases} \tag{1}$$

where $\mathbf{w}$ is weight vector and b is bias. It can be naturally generalized to a multi-class classifier. For input $\mathbf{x}$ and label y of an arbitrary sample from training set, we use a feature function $f(\mathbf{x}, y)$ that maps the pair to a feature vector. Then the problem is converted to

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, f(\mathbf{x}, y) \cdot \mathbf{w}. \tag{2}$$

And the update formula for sample t becomes:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + f(\mathbf{x}, y) - f(\mathbf{x}, \hat{y}). \tag{3}$$

Note that there are plenty of variants of perceptrons. Here, we use the default setting which does not have regularization and updates only on mistakes.

3.2 Adaptive Boosting

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases, introduced in 1995 by Freund and Schapire [5].

The AdaBoost alrorithm used in this project is SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function), which is a multi-class extension of two-class AdaBoost algorithm, and does not need to be reduced to multiple two-class problems . SAMME (and its variant SAMME.R) shares the same modular boosting structure of AdaBoost, i.e. adaptively combining weak classifiers into a powerful one, but only requires the performance of each weak classifier be better than random guessing 1/K (K is number of labels) instead of 1/2. SAMME was developed by Hastie [7], and is proved to have comparable or even better performance with that of classic AdaBoost algorithms.

The main parameters to tune to obtain good results are number of weak classifiers and the complexity of weak classifiers. For the latter there are several off-the-shelf sklearn classifiers which support sample weights that can be tried, e.g. DecisionTreeClassifier, ExtraTreeClassifier, ExtraTreesClassifier, MultinomialNB, NuSVC, Perceptron, RandomForestClassifier, RidgeClassifierCV, SGDClassifier and SVC [13].

3.3 Support Vector Machine

Support Vector Machine (SVM) is kind of supervised learning algorithm for classification and regression analysis. It represents the data samples as points mapped into a kernel space, and then divides those data points by a hyperplane which aims at maximizing the gap between the plane and the data points of different categories. When a new data sample comes, the SVM model will predict its class based on the side of the hyperplane it falls into.

The hyperparameters that we are going to tune include: 1) "`C`", which is the coefficient of slack variables; 2) "`kernel`", which is the type of the kernel we use, such as "`linear`", "`polynomial`", "`rbf`", etc; 3) `tol`, which represents the stopping criterion.

4 EXPERIMENT

Selection of features and hyperparameters are critical to the performance of an algorithm. To fairly evaluate aforementioned algorithms, we conduct experiments under different settings and use cross validation techniques. In this section, we detail the settings.

4.1 Dataset Setup

As mentioned in Section 2, the original dataset has more than 35000 samples with imbalanced distributed labels. For simplicity, we pick 500 samples for each expression category, which forms a subset with a total of $500 \times 7 = 3500$ samples.

4.2 Image Encoding

In previous discussions, we assumed that the feature vector of an image is given. In practice, image coding (i.e., mapping of a 2D grayscale image to a feature vector) plays an crucial role in the performance of the algorithms. So far, we have explored two image encoding methods: raw pixels and facial landmarks.

Raw Pixels. The original images in FER2013 is in 48×48 resolution and grayscal. Each of the pixel is an integer ranging from 0 to 255. For this simpliest encoding, we flatten the 2D image matrix to 1D array and normalize it to $[0, 1]$. Then we obtain the feature vector of length 2304.

Facial Landmarks. By leveraging geometric features of faces and a pre-trained deep model, we obtain 68 facial landmarks for each image (see Fig. 3 for examples), which provide us with extra semantic information of the faces from the raw pixels representation. Specifically, we can extract certain geometric features of faces from these landmarks which, intuitively, may relate to facial expressions (e.g., the size of mouth, distance between two lips). Here, we follow [9] and convert the 68 landmarks to 12 features. Table 1 shows details of the conversion, where p_i represents the i-th landmarks and $||p_i - p_j||$ denotes the Euclidean distance between points i and j.

Figure 3: Examples of facial landmarks. The first row shows the original images in FER2013 [6]; the second row are images with facial landmarks.

Table 1: Facial landmarks to feature vector.

Feature	Description
$\|\|p_{37} - p_{40}\|\|$	Left eye height
$\|\|p_{38} - p_{42}\|\|$	Left eye width
$\|\|p_{43} - p_{46}\|\|$	Right eye height
$\|\|p_{44} - p_{48}\|\|$	Right eye width
$\|\|p_{18} - p_{22}\|\|$	Left eyebrow width
$\|\|p_{23} - p_{27}\|\|$	Right eyebrow width
$\|\|p_{49} - p_{55}\|\|$	Lip width
$\|\|p_{18} - p_{41}\|\|$	Distance between left eye and left eyebrow
$\|\|p_{48} - p_{27}\|\|$	Distance between right eye and right eyebrow
$\|\|p_{34} - p_{67}\|\|$	Distance between nose centre and lips centre
$\|\|p_{42} - p_{49}\|\|$	Distance between left eye and lips left corner
$\|\|p_{47} - p_{55}\|\|$	Distance between right eye and lips right corner

4.3 Resampling for Validation

In the experiment, we use two resampling techniques for validation: k-fold cross-validation and boostrapping.

k-fold Cross Validation. We choose $k = 10$ for k-fold cross validation, which means the size of the testing set is $3500/k = 350$, and the size of the training set is $3500 - 350 = 3150$. In order to keep the dataset balanced, we extract 450 and 50 images from each emotion to form the training and testing set respectively.

Table 2 shows the result of 3 different algorithms, perceptron, adaboost, and SVM. To be specific, the accuracy is calculated according to the formula below:

$$accuracy = \frac{\#\ of\ images\ classified\ correctly}{\#\ of\ total\ test\ images}$$

Table 2: Accuracy of the algorithms with cross validation.

Perceptron	Adaboost	SVM
0.2337	0.2557	0.3323

Table 2 shows SVM achieved better performance than perceptron and adaboost. The most possible reason is we used linear model for perceptron and adaboost, but used radial basis function kernel for SVM, which may better capture the features of emotion images. Hyperparameters of different models will be tuned in our future work.

We also plot the confusion matrices of the three algorithms in Fig. 4 (a)-(c). From this figure, we could see emotions such as "surprise" and "happy" can be recognized more accurately, while emotions like "angry" and "surprise" are hard to be classified. The detailed reasons of accuracy gaps between different emotions will be explored in the final report.

Bootstrapping. In bootstrapping, images of different emotions are sampled with replacement. In our experiment, we choose $B = 10$ and calculate the average classification accuracy of all the bootstrapping processes. Table 3 shows SVM achieved better performance than perceptron and adaboost in bootstrapping. We also plot the confusion matrices of the three algorithms in Fig. 4: (d)-(f), which are similar to the results of k-fold cross validation.

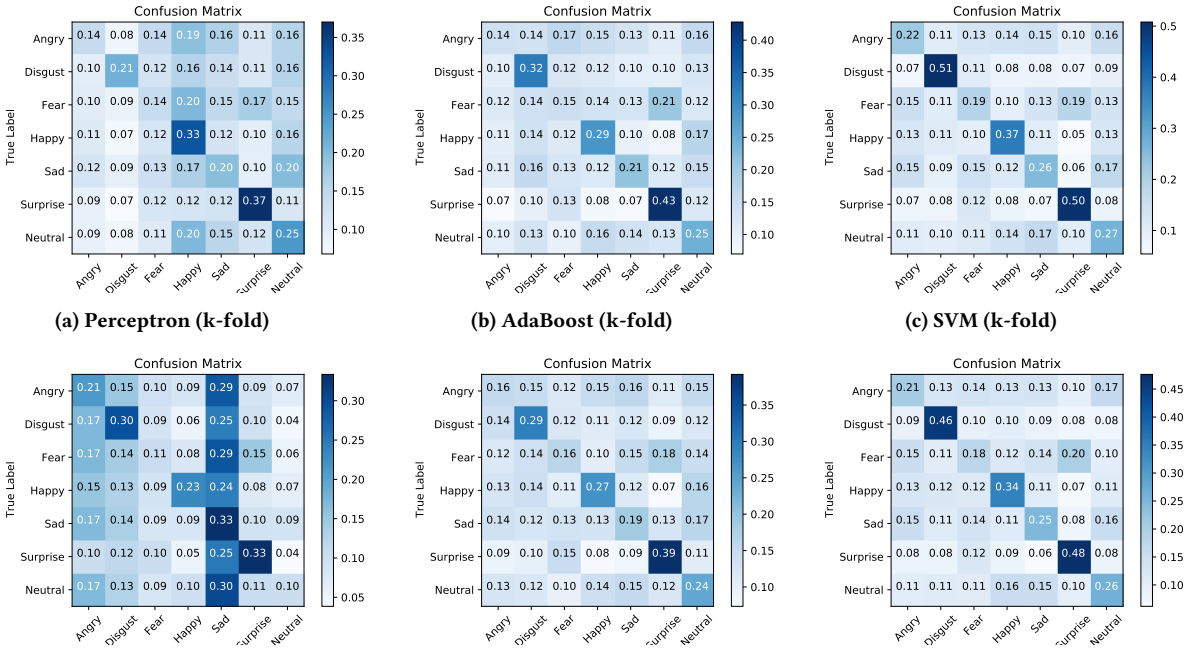
Table 3: Accuracy of the algorithms with bootstrapping.

Perceptron	Adaboost	SVM
0.23142503	0.24360149	0.31108016

5 CONCLUSION AND FUTURE WORK

So far, we have explored three mahcine learning algorithms (i.e., Perceptron, Adaptive Boosting, and Support Vector Machine) on a subset of FER2013, and evaluated their performance using two resampling techniques for validation (i.e., 10-fold cross-validation and boostrapping with 10 repetitions). We found that SVM outperformed the other two whle the Perceptron algorithm is the weakest among the three. We also tried two ways to construct the feature vector from a raw input images: raw pixels and facial landmarks, and concluded that raw pixel incorporated with hand-crafted geometric features (i.e., distances of selected semantic features calculated from the facial landmarks) achieved the best prediction.

In the future, we plan to improve in following directions (also described in the project plan): (i) exam different hyperparameter settings of the aforementioned algorithms (e.g., kernel in SVM, number of classifiers in AdaBoost); (ii) implement and compare more



(a) Perceptron (k-fold) (b) AdaBoost (k-fold) (c) SVM (k-fold)

(d) Perceptron (bootstrap) (e) AdaBoost (bootstrap) (f) SVM (bootstrap)

Figure 4: Confusion matrices of different algorithms in k-fold cross validation and bootstrapping.

machine learning algorithms (e.g., decision tree) and especially the-of-the-art methods in the literatrue (i.e., deep neural networks with various architectures); iii) explore other image encoding methods and facial features (e.g., latent face representations from pre-trained deep models); 4) explore the relationship between the size of dataset and learning accuracy of different algorithms.

REFERENCES

[1] Paul Ekman. 1994. Strong evidence for universals in facial expressions: a reply to Russell's mistaken critique. (1994).

[2] Paul Ekman and Wallace V Friesen. 1971. Constants across cultures in the face and emotion. *Journal of personality and social psychology* 17, 2 (1971), 124.

[3] Irfan A. Essa and Alex Paul Pentland. 1997. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE transactions on pattern analysis and machine intelligence* 19, 7 (1997), 757–763.

[4] Beat Fasel and Juergen Luettin. 2003. Automatic facial expression analysis: a survey. *Pattern recognition* 36, 1 (2003), 259–275.

[5] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[6] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. 2013. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*. Springer, 117–124.

[7] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class adaboost. *Statistics and its Interface* 2, 3 (2009), 349–360.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[9] Fuzail Khan. 2018. Facial expression recognition using facial landmark detection and feature extraction via neural networks. *arXiv preprint arXiv:1812.04510* (2018).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[11] Shan Li and Weihong Deng. 2020. Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing* (2020).

[12] Maja Pantic and Leon J. M. Rothkrantz. 2000. Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on pattern analysis and machine intelligence* 22, 12 (2000), 1424–1445.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[14] Frank Rosenblatt. 1957. *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory.

[15] Caifeng Shan, Shaogang Gong, and Peter W McOwan. 2009. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and vision Computing* 27, 6 (2009), 803–816.

[16] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[18] Guoying Zhao and Matti Pietikainen. 2007. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 915–928.

[19] Ruicong Zhi, Markus Flierl, Qiuqi Ruan, and W Bastiaan Kleijn. 2010. Graph-preserving sparse nonnegative matrix factorization with application to facial expression recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41, 1 (2010), 38–52.

[20] Lin Zhong, Qingshan Liu, Peng Yang, Bo Liu, Junzhou Huang, and Dimitris N Metaxas. 2012. Learning active facial patches for expression analysis. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2562–2569.