Comparison of Maximum Likelihood Classification Method with Artificial Neural Network on Static Facial Expressions in the Wild Dataset

Dingyi Wang

ResearchSchool of Computer Science, Australian National University, Australia
u6471736@anu.edu.au

Abstract. Human facial expression classification is an emerging topic in both academic and industrial fields. Researchers used several methods to strive for higher accuracy, among which statistical methods and neural networks are the most widely used ones. We implement a statistical method, which is maximum likelihood classification, and a modern artificial neural network on a static facial expression database called Static Facial Expressions in the Wild (SFEW) [1], to compare their performances. The result shows both methods perform worse than the support vector machine method used in SFEW research in terms of accuracy, we conclude that maximum likelihood is not able to find the pattern on such datasets, and the plain neural network lacks enough data. We extend the neural network to a convolutional neural network (CNN) using the ResNet [20] basic blocks on the original images of SFEW and the result is comparable to the original performance. More data and training and the use of a deeper detection-based CNN classification are potential solutions for future improvements.

Keywords: Facial expression classification, maximum likelihood classification, artificial neural network, convolutional neural network

1 Introduction

Identifying what human expressions are in images and videos is an attracting topic in both research study and industry area. Researchers applied different techniques [3][4] to solve this question, and have shown that using a proper method to analyze the data is critical to the expected accuracy of the classification.

The Static Facial Expressions in the Wild (SFEW) [1] dataset we used contains 675 lines of 10 static data which is consisted with 2 high quality image features' top 5 principal component analysis (PCA) components, each along with a label indicates one of seven emotions. The distribution of the data is as Table 1 shows. These data were collected by selecting frames from movies which are closer to real world environment expressions than the photos took in the labs such as JAFFE (The Japanese Female Facial Expression) datasets [18]. With SFEW, we can define a supervised classification problem for which we would like to use these 10 PCA components as inputs to predict the emotion label we will get.

Table 1. Distribution of the SFEW dataset.

Emotion	Number
Anger	100
Disgust	75
Fear	100
Happy	100
Sad	100
Surprise	100
Neutral	100

PCA is a useful tool for data dimension reduction and feature extraction and is widely used in image-based detection/classification studies [2, 5]. By finding the basis vectors (principle components) of a rebuilt subspace of a face space which has better ability to describe the face images, PCA allows us to use these basis vectors which have maximum variance to the original space as extracted features and minimize the reconstruction error.

To see how methods differ on such a PCA-based dataset, we choose both a conventional statistic method which is maximum likelihood classification, and simple neural network method for comparison.

Maximum likelihood classification was widely used for pattern recognition in the previous years, especially for image classification in the field of remote sensing [6, 7, 8]. It is based on multivariate normal distribution theory and able to identify similarities between individual measurements and predefined standards.

Another approach is to use neural networks. By saying neural networks, we are talking about back-propagation (BP) neural networks, which are currently the most commonly used neural network [10] and has been widely in regression, classification, detection problems and etc. [11, 12, 13]. Neural networks are inspired by how human brain works, and they typically consist nodes (input nodes and neurons), synaptic connections and functional connections [10]. In the learning

phase, the connection weights are adjusted while in the working phase, new inputs are propagated on the fixed weights to get result.

A deep convolutional neural network (CNN) is also introduced and tested on the original image data of SFEW as an extension of the neural network experiment. Since AlexNet [13] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC12) on ImageNet dataset [24] in 2012, CNNs have been widely used in image classification [19, 20]. Using convolutional and pooling layers, CNNs succeed in this field with their capability of joint feature and classifier learning [21]. We are curious to see how deeper neural networks can achieve on this dataset.

We implement all the methods to demonstrate the differences between their performances on the dataset. We also use the original experimental result provided by the SFEW paper as a baseline for comparison, which applied a support vector machine method.

2 Methods

2.1 Maximum Likelihood Classification

Maximum likelihood classification was widely used for pattern recognition in the previous years, especially for image classification in the field of remote sensing [6, 7, 8]. It is based on multivariate normal distribution theory and able to identify similarities between individual measurements and predefined standards.

Consider one observation x consists p variables (dimensions or channels), then by multivariate normal statistical theory, assuming equal prior probabilities, the probability of x will occur under it belongs to class k is given by [6]:

$$P(x|k) = \frac{1}{\sqrt{(2\pi)^p|\sum_k|}} \cdot e^{-\frac{1}{2}(x-\mu_k)^T{\sum_k}^{-1}(x-\mu_k)}, \quad (1)$$

where μ_k is the observed (training) mean vector of class k, and $\sum_k$ is the observed (training) variance-covariance matrix of class k. The term $(x-\mu_k)^T{\sum_k}^{-1}(x-\mu_k)$ can be considered as the Mahalanobis distance between the vector and the centroid of a specific class [9]. We can then use expression (1) to calculate the probability of x is each of k classes. As the log of the probability function is monotonic increasing, we would use the log form of expression (1), which gives us:

$$\ln\big(P(x|k)\big) = -\frac{1}{2}pln(2\pi) - \frac{1}{2}\ln|\sum_k| - \frac{1}{2}(x-\mu_k)^T{\sum_k}^{-1}(x-\mu_k), \quad (2)$$

and we can simply compare the values of this equation for each class to make decisions.

In this study, we will implement the equation (2) with a Python program, the 10 PCA components are considered as the measurements of observations and the emotion labels are the expected classes. We let the classifier fit with the train data and predict on the test data.

2.2 Neural Network

2.2.1 Simple BP Neural Network

BP neural networks usually have multiple layers of nodes, including an input layer, one or more hidden neuron layers and an output layer. The algorithm will run a cycle of forward propagation and backward propagation during the learning phase [10]. The work flow includes:

1. Initialize weights.
2. Feed input X & target output T.
3. Calculate output Y.
4. Calculate the cost with proper loss function and its gradients.
5. Back propagate the gradients using the chain rule from output layer to input layer.
6. Update weights.
7. Repeat until the end of the training.

For this experiment, we will use Python and PyTorch to construct a simple multi-layer BP neural network with 1 to 2 hidden layers, all with Sigmod activation function and Adam optimizer [14], which is a well-known gradient-based optimization of stochastic objective functions, instead of classical stochastic gradient descent (SGD) for faster learning in a relatively less training iterations to avoid overfitting. As this is a classification problem, we choose cross entropy loss as our loss function [15]. A multi-class cross entropy loss is given by:

$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \frac{e^{\hat{y}^{(k)}}}{\sum_{j=1}^{K} e^{\hat{y}^{(j)}}}, \quad (3)$$

where k is the class, $\hat{y}$ is the output and y is the target. Different numbers of neurons were applied to increase the model's capacity for higher accuracy.

2.2.2 Convolutional Neural Network

The majority parts of CNNs are layers called convolution layers which filter their layer inputs and find useful features within those inputs [22]. We customize the parameters of filters which consists neurons and apply the filter kernels to specific areas of the input data, such process is called convolution, with which we can produce feature maps. There are also pooling layers which produce summary statistic of the output from the previous convolutional layer and can help learned representation to be invariant to small translations of input. Optimization techniques such as Dropout [13] and Batch Normalisation[23] are often used to avoid overfitting or accelerate convergence.

In this experiment, we adopt the technique of ResNet [20], which is a well-known image classification network to build our CNN. The core of ResNet is to use "residual blocks" (Fig.1) to address the problem of performance decreases when the network gets deeper (also referred as "degradation"). Assume the deeper output is $\mathcal{H}(x)$, it tries to achieve $\mathcal{H}(\boldsymbol{x}) = \mathcal{F}(x) + x$ when we want to use map shallow output x to $\mathcal{H}(\boldsymbol{x})$ using mapping $\mathcal{F}(\boldsymbol{x})$, which is based on the idea that it will be easier to optimize the network to achieve at worst the same performance when the original output is optimal. ResNet implements such idea by applying "shortcut connections" for which shallow outputs are directly added to the outputs after one or more layers to perform identity mapping.

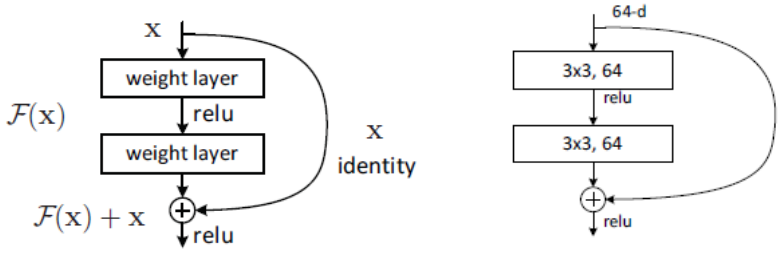

Fig. 1. Residual basic building block

Due to our limited computational resources, we implement a much shallower version of ResNet, which uses 4 basic building blocks of ResNet, and the structure of our CNN is shown in Fig.2:

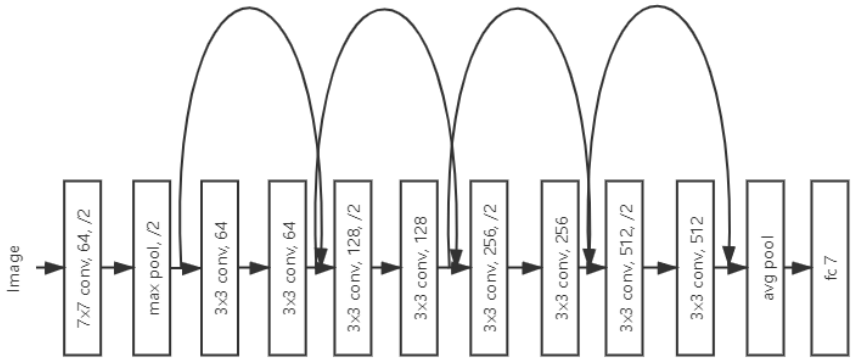

Fig. 2. CNN structure

where the numbers in the convolution filter box represent filter numbers, and "/2" represents downsampling realized by using layers with a stride of 2. The identity shortcut's downsampling is done by 1x1 convolutions with a stride of 2. After each convolution, we perform batch normalization and ReLU activation. The last layer is a fully-connected layer.

We still use a cross-entropy loss and Adam optimizer, but as the inputs and the network structure are different, we need to adjust part the training process. The original images are resized to 224x224 as inputs, and are fed into the network

with 40 training epochs and a batch size of 90 as our graphic card is not able to support computations with full batch training.

2.3 Test Method

As SFEW doesn't have separate datasets for training and testing, we would like to apply K-fold cross validation method [16] for our testing phase. It's a good way to evaluate the model's performance on unavailable data and to prove the model is not overfitting. For both classification method, we randomly divide the original dataset into 5 equal-sized partitions, of which one is used for testing, and others are used for training at a time, the final accuracy is produced by calculating the mean accuracy of these k-time experiments.

3 Result and Discussion

The k-fold accuracy result of maximum likelihood classifier (MLC) is shown in Fig.3 and Fig.4, and the experiment result of our simple neural network is shown is Table 2 and 3. To compare with, the original SFEW paper's SVM classifier got an accuracy of 43.71% with the first 5 PCA components and 46.28% with the last 5 PCA components on the seven-expression experiment [1].

Both of these implementations are not able to match the original accuracy, while the reasons could be different. The result of MLC method shows that although the accuracy of MLC on SFEW is relatively low, it maintains in a relatively consistent way both for training and testing data. We believe the reason why MLC misperformed in this experiment is that it depends more on the distribution of the training data, while the SFEW data doesn't really comply a multivariate normal distribution which is what MLC theory is based on [6], as SFEW was collected from different scenes of different movies. It is the characteristic of the method that makes MLC failed to find the pattern inside the data, and thus not capable to solve this problem.

On the other hand, the unsatisfiable result of the plain neural network has more complex reasons, although it still achieves a higher accuracythan the MLC. We can notice that throughout the experiment, while the training accuracy increases as the epochs increases, the test accuracy quickly drops when the model trained for too many epochs. Even though we add more neurons and layers to the network, the model still quickly demonstrates overfitting. A major issue of BP algorithms is the local minimum [10], but it is clear that we haven't reached that point. We argue that this is because the dataset is not big enough or complete enough for the network to find the correlation between the features and the

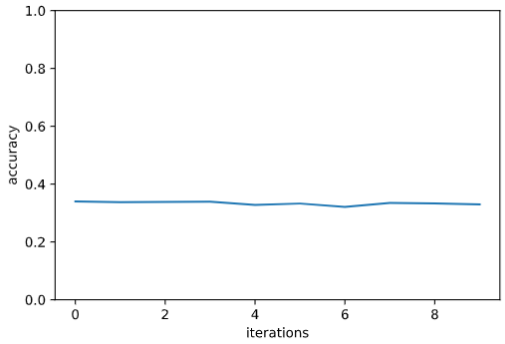

Fig. 3. Training accuracy of MLC

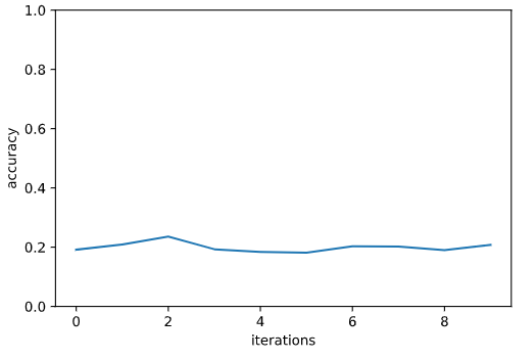

Fig. 4. Testing accuracy of MLC

Table 2. Average 5-fold loss with different hyper-parameters of plain neural network.

Settings	2000 epochs	4000 epochs	6000 epochs	8000 epochs	10000 epochs
One hidden layer with 10 neurons	1.8628	1.8135	1.7955	1.7675	1.7390
One hidden layer with 20 neurons	1.8334	1.8030	1.7800	1.7419	1.6964
Two hidden layers with 10 and 20 neurons	1.8473	1.7682	1.6890	1.5623	1.4363

Table 3. Average 5-fold accuracy with different hyper-parameters of plain neural network.

Settings	2000 epochs	4000 epochs	6000 epochs	8000 epochs	10000 epochs
One hidden layer with 10 neurons	22.38%	22.54%	25.52%	24.33%	23.28%
One hidden layer with 20 neurons	23.28%	24.63%	24.93%	25.67%	24.18%
Two hidden layers with 10 and 20 neurons	25.67%	23.54%	24.63%	23.28%	22.54%

expressions, nor to complete generalization. In contrast, it may also be another proof that SVMs have better generalization ability especially when provided with limited training data [17].

Other than the accuracy, a major difference that can be noticed is the efficiency. MLC can complete 10 iterations of both training (fitting) phase and predict phase in less than 1s on an Intel® Core™ i7-8750H platform, while our neural network requires far more than time than that of MLC especially when training for large epochs. This can be explained by how these methods works. Assuming we have N training examples x of k classes for both methods. Provided that we are using a simple 2-layer BP neural network with respectively $N1$ and $N2$ hidden neurons and k output nodes, then in one epoch of training, the computer needs to execute the following operations [10]:

1. Calculate Outputs $Y(y_1, y_2, \dots, y_k)$:

$$h_j^I = f\left(\sum_{i=0}^{N} w_{ij} x_i\right), 0 \le j \le N1 \quad \textbf{(4)}$$

$$h_k^{II} = f\left(\sum_{j=0}^{N1} w_{jk}^I h_j^I\right), 0 \le k \le N2 \quad \textbf{(5)}$$

$$y_l = \mathrm{f}\left(\sum_{k=0}^{N2} w_{kl}^{II} h_k^{II}\right), 0 \le l \le k \quad \textbf{(6)}$$

2. Calculate cost with loss function.
3. Update weights with back propagated gradients with learning rate $0 < \eta < 1$:

$$w_{kl}(t+1) = w_{kl}(t) + \eta\left(-\frac{dL}{dw_{kl}}\right) \quad \textbf{(7)}$$

$$w_{jk}(t+1) = w_{jk}(t) + \eta\left(-\frac{dL}{dw_{jk}}\right) \quad \textbf{(8)}$$

While for MLC, it only needs to calculate k means vectors and covariance matrixes in the fitting phase, and do k times of matrix multiplication $(x-\mu_k)^T {\sum_k}^{-1}(x-\mu_k)$ in the predicting phase. The computational cost gap caused the execution-time difference, which is enlarged by the number difference of training epochs.

After applying CNN on the image-based dataset, although it's still not enough to say that the network can make trustworthy predictions, we can see a noticeable increase in terms of accuracy:

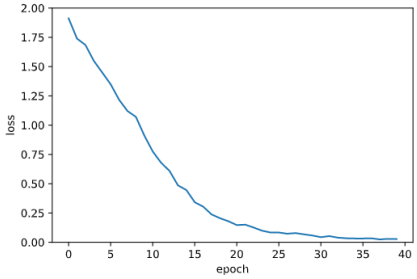

Fig. 5. Average loss of 5-fold validation process of CNN

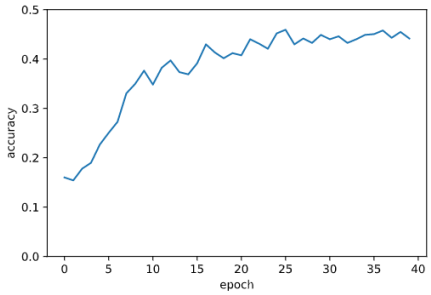

Fig. 6. Average accuracy of 5-fold validation process of CNN

Different with the plain neural network, we can see as the loss decreases, the accuracy of the CNN predictions increases for a long period of time instead of quickly falls into overfitting. Except for a much bigger network structure, the reason that CNNs can perform better in such computer vision problems is their capability of joint feature and classifier learning [21].

4 Conclusion and Future Work

Using SFEW dataset, we have implemented two kinds of methods for facial expression classification. The result demonstrated that both of the methods are not suitable for perform classification task on this dataset. We analyzed the result and proposed possible reasons: MLC failed for not being able to extract the statistical features behind the data, while neural network requires more data to complete a generalized training. Our experiment process also indicates the characteristics of these two methods, that neural networks can achieve higher accuracy with a significantly longer training time, and MLC is useful for more statistical connected and time-efficient tasks.

We also extended our experiment on neural networks by implementing a CNN to train and test on the original image-based dataset of SFEW, in which the CNN performed better in terms of accuracy despite using this relatively small dataset. It shows the ability of CNNs on pattern recognition, and demonstrated the potential of CNNs on facial expression classification tasks given enough data and computational resources.

Our experiment also has some limitations. We are not able to try out all possible hyperparameters for neural networks, and both of our NN structures are relatively simple. Further improvements can be using a larger dataset for training the neural networks, and design a more complex neural network that can first detect the faces [25] to get bounding boxes and then perform classifications on the selected areas of images.

References

1. Dhall, A., Goecke, R., Lucey, S., Gedeon, T. Static facial expressions in tough conditions: Data, evaluation protocol and benchmark. In: 1st IEEE International Workshop on Benchmarking Facial Image Analysis Technologies BeFIT, ICCV2011. (2011).
2. Sahoolizadeh, A.H., Heidari, B.Z., Dehghani, C.H.: A new face recognition method using PCA, LDA and neural network. International Journal of Computer Science and Engineering, vol.2(4), pp.218-223. (2008).
3. Zhang, T., Zheng, W., Cui, Z., Zong, Y., Yan, J. & Yan, K.: A Deep Neural Network-Driven Feature Learning Method for Multi-view Facial Expression Recognition. IEEE Transactions on Multimedia, vol.18, pp.2528-2536. (2016).
4. De Campos, C.P., Tong, Y., Ji, Q.: Constrained maximum likelihood learning of Bayesian networks for facial action recognition. In European Conference on Computer Vision, pp. 168-181. Springer, Berlin, Heidelberg. (2008).

5. Lin DT: Facial expression classification using PCA and hierarchical radial basis function network. Journal of information science and engineering, vol.22(5), pp.1033-46. (2006).
6. Strahler, A. H.: The use of prior probabilities in maximum likelihood classification of remotely sensed data. Remote Sensing of Environment, vol.10, pp.135-163. (1980).
7. Miles, N. W., Morris, G. M. Maximum-Likelihood Image Classification. Proc.SPIE. (1988).
8. Milne, L. K., Gedeon, T. D., Skidmore, A. K.: Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood. training, vol.109(81), pp.0. (1995).
9. Foody, G. M., Campbell, N. A., Trodd, N. M., Wood, T. F.: Derivation and applications of probabilistic measures of class membership from the maximum-likelihood classification. Photogrammetric engineering and remote sensing, vol.58(9), pp.1335-1341. (1992).
10. Zhang, W.: Computational ecology: Artificial neural networks and their applications. World Scientific Publishing Co Pte Ltd. (2010)
11. Specht, D. F.: A general regression neural network. IEEE transactions on neural networks, 2(6), pp.568-576. (1991).
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. (2016).
13. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems pp. 1097-1105. (2012).
14. Kingma, D. P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980. (2014).
15. Zhang, Z., Sabuncu, M.: Generalized cross entropy loss for training deep neural networks with noisy labels. In Advances in neural information processing systems, pp. 8778-8788. (2018).
16. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society: Series B (Methodological), vol.36(2), pp.111-133. (1974).
17. Shao, Y., Lunetta, R. S.: Comparison of support vector machine, neural network, and CART algorithms for the land-cover classification using limited training data points. ISPRS Journal of Photogrammetry and Remote Sensing, vol.70, pp.78-87. (2012)
18. M. J. Lyons, S. Akamatsu, M. Kamachi, J. Gyoba.: Coding facial expressions with gabor wavelets. In Proceedings of the IEEE International Conference on Automatic Face Gesture Recognition and Workshops, FG'98, (1998).
19. Christian S., Wei L., Yangqing J., Pierre S., Scott R., Dragomir A., Dumitru E., Vincent V., Andrew R.: Going deeper with convolutions. arXiv: 1409.4842. (2014).
20. Kaiming H., Xiangyu Z., Shaoqing R., Jian S.: Deep residual learning for image recognition. arXiv:1412.6980. (2014).
21. Jiuxiang G., Zhenhua W., Jason K., Lianyang M., Amir S., Bing S., Ting L., Xingxing W., Li W., Gang W., Jianfei C., Tsuhan C.: Recent Advances in Convolutional Neural Networks. arXiv: 1512.07108. (2015).
22. Iffat Z., Giounona T., Richard B., Nimesh P., Leonardo A.: Hands-On Convolutional Neural Networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python. Packt Publishing. (2018).
23. Sergey I., Christian S.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv: 1502.03167. (2015).
24. Deng, J., Dong, W., Socher, R., Li, L.-J.,Li, K., Fei-Fei, L.: ImageNet: Alarge-scale hierarchical imagedatabase. In CVPR09 (2009).
25. Kaipeng Z., Zhanpeng Z., Zhifeng L., Yu Q.: Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. arXiv:1604.02878. (2016)