Transferred Deep Residual Learning Network for Classification of Static Facial Expression in the Wild: Construction, Parameter Tuning and Cross-Validation Evaluation

Gengliang Li

Research School of Computer Science,
Australian National University
u6799959@anu.edu.au

Abstract: The classification of human facial expressions using AFEW dataset with Support Vector Machine model [1] has showed us the difficulty of developing facial expressions recognition techniques that are robust enough for uncontrolled environment experiments. In this article, apart from the feed-forward neural network built in last assignment which can perform similar facial expression clustering functions using principle components of predictors calculated from AFEW dataset, a transferred deep residual learning network [9] will also be built for same task. However, the data used for training will be the original images from AFEW dataset, which are frames captured from movies. To implement model training and hyperparameter tuning, I use a training-testing process for rough selection and cross-validation for overall evaluation. The resulting 5-folder cross-validation overall accuracy for the transferred model is about twice as much as the simple feed-forward neural net and almost the same as the Support Vector Machine model [1] built by the original paper.

Keywords: Static Facial Expression Classification, Deep Transfer Learning, Deep Residual Learning Network

1 Introduction

The human facial expression analysis has attracted more and more attention of people working on computer vision since the past decade. The importance of such analysis lies in the interaction of many important applications, for example, human computer interaction, surveillance and crowd analysis.[6] Though in most scenarios we care about dynamic expression analysis, static facial expression analysis can also be vital for image-based data.

The majority of existing static facial datasets mainly focus on classifying 7 basic expressions, which are angry, disgust, fear, happy, sad, surprise and the neutral class. Some earliest of these datasets such as JAFFE [7] and PIE[8] are collected in a lab-controlled environment. The problem with these datasets is that they are not similar enough to the real-world situations. The model that works well upon these datasets may give unsatisfactory performance when dealing with real-world data from movies, short videos and live camera.

To avoid this problem, a dynamic temporal facial expressions data corpus called Acted Facial Expressions in the Wild (AFEW) [3] is created. It consists of facial expression images that are extracted from movies, so these facial expressions are close to real world expressions. To train the model for this report, I will use dataset generated by frame selection of AFEW called SFEW (Static Facial Expression in the Wild). [3] Pictures in SFEW are more difficult to deal with than those generated in labs because they may have low resolution and illumination, motion blur, non-frontal head-poses and etc. Besides, the size of the dataset is relatively small. In total, SFEW contains 700 labelled images. These characters make SFEW a more realistic dataset to train with, and also, a more challenging one.

In the last assignment, I trained a simple feed-forward neural net with one hidden layer to classify these expressions using principle components data. However, the performance of such net is not satisfactory. As a result, in this report, a deep neural network transferred from Deep Residual Learning Network [9] is implemented to improve the model performance.

Deep Residual Learning Network, or ResNet for short, is first invented by a research group lead by Dr. Kaiming He in 2015. When training a Convolutional Neural Net, instead of hoping each few stacked layers directly fit a desired underlying mapping, they explicitly let these layers fit a residual mapping. [9] And they trained their model on ImageNet dataset, [10] which is a very large labeled image dataset widely used for model training and evaluation in computer vision area nowadays. In this report, the trained ResNet model is transferred to customize our task by reinitializing the last fully connected layer. After that, the transferred model will be trained and its hyperparameters will be tuned. Finally, Cross-validation will be implemented to evaluate model's performance more objectively.

2 Method

2.1 Review of Previous Work

Given an image of facial expression, we can develop its Local Binary Patterns (LBQ) descriptor [4], which assigns binary labels to pixels by thresholding the neighborhood pixels with the central value [1] and its extension Local Phase

Quantization (LPQ), which is based on computing the short-term Fourier transform (STFT) on a local image window [1]. Also, we develop the histogram of oriented gradients (HOG) descriptor [5], which counts occurrences of gradient orientation in localized portions of an image. [1] As an extension of HOG, we will also calculate the pyramid of histogram of oriented gradients (PHOG) descriptor. [1]

In previous assignment, to train the feed-forward neural net, I first use a dataset composed of the first 5 principle components from feature LPQ as inputs and the corresponding facial expression labels as output, and then similarly another dataset consists of the first 5 principle components from feature PHOG as inputs and expression labels as outputs. As there are seven classes representing each expression of angry, disgust, fear, happy, sad, surprise and the neutral, it is a multi-classification net which has an input dimension of 5 and output dimension of 7. For each dataset, there are 100 samples for each expression class except the disgust class, so we have 675 samples for training in total, each sample consists of 5 principle components either from LPQ or PHOG.

To build a multi-classification network, first I set some initial hyperparameters, which includes batch size for loading training data into the net, learning rate for weights update, and training epochs number and hidden layer size. These hyperparameters are used for net initializations and will be adjusted later. Also, fold number of cross-validation is set as 5, which is the same choice as the SVM classifier built by previous study. [1] Adam[1] optimization algorithm is used for weights update, this is because our dataset for training is relatively small and step size of Adam update rule is invariant to the magnitude of the gradient, which helps explore more space.

Second, a function is created to realize data loading by batch size while training (part of code for this function takes reference from Lab2 script), and cross validation for model accuracy evaluation. For the cross-validation part, both the LPQ components dataset and PHOG components dataset are shuffled to make each of the expression class equally distributed in each folder as possible. Then the dataset is divided into 5 folders and for every training-validation progress four of these folders are used to train the model from scratch and after enough training the left folder is used for validation.

Finally, a confusion matrix is created to analyze the overall validation accuracy for this neural net as well as precision, recall and specificity for each expression class. By tuning hyperparameters mentioned above and repeat the cross-validation training process, I try to find the model that gives best validation accuracy.

2.2 Transfer Model Construction

2.2.1 Image Preprocessing and Data Loading

For this assignment, a new set of data will be used for training. Instead of just principle components from some image descriptor as numeric values, samples in this dataset are all real images frames selected from movies. These images are all of three channels with 576*720 dimension of pixels. When we take a look at these pictures, we can see that the edges can be removed because almost they don't contain the figure's faces. Thus, for all training, testing or validation images, they are center cropped to size 576 and resize to size 299 as a common choice for image classification. For data normalizing, the sequence of means for each channel should be abstracted from each sample and then divided by sequence of standard deviations for each channel. However, the method to calculate mean and standard deviation for each channel of an image dataset is complicated, thus here, I use the results from ImageNet data, which is (0.485, 0.456, 0.406) for mean and (0.229, 0.224, 0.225) for standard deviation of each RGB channel.

Then image data loaders should be designed for loading data by batch during training, testing and cross-validation process. For this task, as each sample data is large in size, the batch size of loading sample images into the model can be an essential hyperparameter to be tuned. Too large a batch size might lead to poor generalization and overfitting while too small a batch size might not be able to fully explore the solution space and easily stuck in local minima. To inspect the validation of our data loaders, we may visualize a batch of images.

2.2.2 Model Explanation

For this task, a transferred Deep Neural Net model of ResNet will be implemented. First published in 2015, ResNet has been a milestone in DNN model evolution timeline. Basically, it focuses on offering new solution to the old problem that while training, the vanishing gradient effect on network output makes parameters' gradients in the initial layers becomes extremely small. As a result, weights in initial layers update very slowly or remain unchanged, resulting in an increase in error.

For ResNet, instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping. Formally, denoting the desired underlying mapping as H(x), we let the stacked nonlinear layers fit another mapping of residual denoted by F(x) = H(x) – x, and the original mapping is recast into F(x)+x. [9] So as illustrated the ResNet neuron block in Figure 1, [9] the last layer of each block consists of a residual mapping F(x) and an identity mapping of x. The identity 'shortcut' connections add neither extra parameter nor computational complexity, [9] but only help jump over some layers so that a 'deep' network may not be that 'deep' as it looks like.

[1] See https://arxiv.org/pdf/1412.6980.pdf for more details

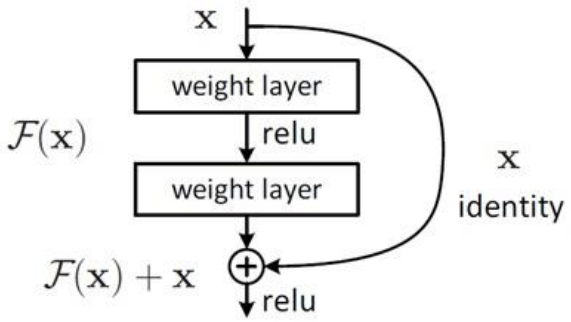

Fig. 1.: Structure of ResNet Neuron Block

There are different versions of ResNet, including ResNet-18, ResNet-34, ResNet-50, and so on. The numbers denote convolutional layers, while the architecture is the same. Here we use Resnet-18 due to our small dataset of 675 samples and limited computing capacity. The architecture of Resnet-18 is showed in the following Figure 2, with both parameters of each convolutional layer and the overall structure.

Layer Name	Output Size	ResNet-18
conv1	112 × 112 × 64	7 × 7, 64, stride 2
conv2_x	56 × 56 × 64	3 × 3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$
conv3_x	28 × 28 × 128	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$
conv4_x	14 × 14 × 256	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$
conv5_x	7 × 7 × 512	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$
average pool	1 × 1 × 512	7 × 7 average pool
fully connected	1000	512 × 1000 fully connections
softmax	1000	

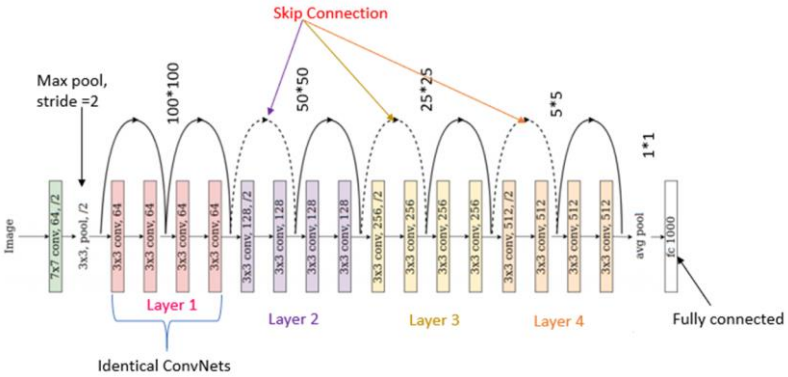

Fig.2.: Architecture of ResNet-18

To implement transfer learning to the ResNet-18 model, I use the well trained ResNet-18 model in pytorch. This model has been well trained on ImageNet, [10] which is a widely used large image dataset which containing 1.2 million images with 1000 categories. Models trained on ImageNet usually gain a good capability to extract features, and such property is vital for image classification. Thus, for this task the well trained ResNet-18 will be utilized as a fixed feature extractor, which is one of the two major formats of transfer learning.

As illustrated in above Figure 3, the final fully connected layer of ResNet-18 is of dimension 1000, so we need to modify this layer to fit the size of our classification output, which is the 7 expression varieties. Then apart from weights of this final fully connected layer, all the weights from previous convolutional, pooling layers will be freezed. This means we only train the final layer.

While training the final layer, I implement cross entropy loss as my loss function as it's a classification problem. The stochastic gradient descent optimizer is used for weights update method with initial learning rate set as 0.1 and momentum value as a hyperparameter to be tuned. Besides, the learning rate of the model is object from *torch.optim.lr_scheduler* module, which can be adjusted based on the number of epochs when training the model. To be more specific, the learning rate of each parameter group decays by multiplier gamma for every step size of epochs, where both gamma and step size are hyperparameters to be tuned. Such settings of learning rate can help us better explore the solution space at the beginning of training and avoid oscillation back and forth from the local minima. Finally, the number of training epoch is set as 50, which I think achieve a relatively good balance between underfitting and overfitting.

2.3 Model Training and Hyperparameters Tuning

For this assignment, the process of model training and hyperparameters tuning is different from what might be expected. Usually, to evaluate the performance of a set of hyperparameters, cross-validation of a dataset should be implemented. However, due to the limited time and computing capacity, iterating through all the samples in our dataset for each set of hyperparameters is difficult.

My solution is to first randomly split 80% of the whole dataset for training, and the rest 20% for testing. For each epoch, both training and testing are implemented and after all epochs, the best epoch testing accuracy will be recorded and this will be the criterion to choose hyperparameters for cross-validation. Those sets of hyperparameters which gives better testing accuracy are picked out for cross-validation evaluation. Such procedure might lead to the problem that the chosen hyperparameters are best fit for the randomly generated training-testing pair but not the whole dataset, but this is my best solution after weighing pros and cons carefully.

There are many hyperparameters that can be tuned in this model, which include: the batch size of data loaders, the training epoch number, the initial learning rate and momentum value of SGD optimizer, the step size and decays multiplier gamma… It's impossible to iterate through all these hyperparameters, thus only some of them are picked for tuning: the batch size of data loaders, momentum value of SGD optimizer and decay multiplier of learning rate schedular. And for each of these chosen hyperparameters, only several different values are implemented for trial, thus the hyperparameters tuning of this model is far from enough.

2.4 Cross-Validation Model Evaluation

As mentioned above, the sets of hyperparameters that generate the best testing accuracy are picked out for cross-validation process. This is to evaluation them more objectively. To implement a 5-folder cross-validation for a certain set of hyperparameters, the whole dataset is shuffled and split into 5 equal subsets. For each time of training-validation, four of these subsets are used for training the model (to be more precise, the last fully connected layer) from scratch and the rest subset is used for validation. Thus, by the end of this 5-folder cross-validation, every single sample of images has been validated by one of the 5 models trained under this set of hyperparameters. Such evaluation can be a true portrayal of the performance of this transferred ResNet with its hyperparameter settings on unseen data.

3 Results and Discussions

3.1 Results from Previous Work

For the seven-expression classification, the best overall accuracy achieved by simple feed-forward neural net with one hidden layer, under 5-folder cross validation, is 20.32% for LPQ principle components dataset and 20.47% for PHOG principle components dataset, with hyperparameter setting as 3 for training loading batch size, 0.03 for learning rate, 1200 for training epoch, and hidden unit number as 6. And the precision (true positive/(true positive + false positive)), recall (true positive/(true positive + false negative)), and specificity (true negative/(true negative + false positive)) for each class are shown in Figure 3 for LPQ dataset and Figure 4 for PHOG dataset. The precise values of above statics for each expression class is provided in Appendix A table 2 and table 3.

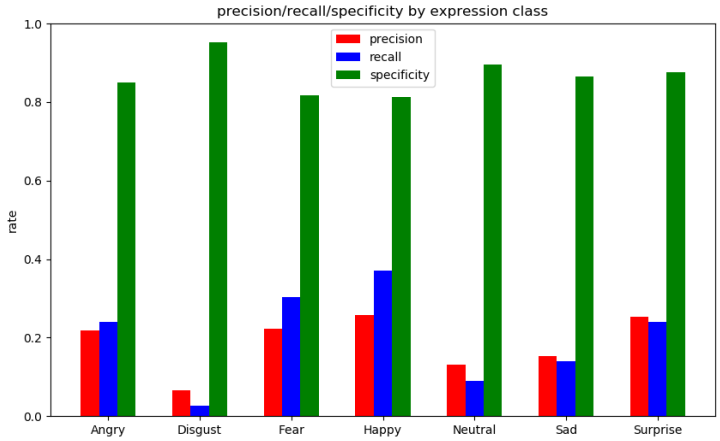

Fig. 3. The overall Precision, Recall, and Specificity for each expression class, LPQ dataset.

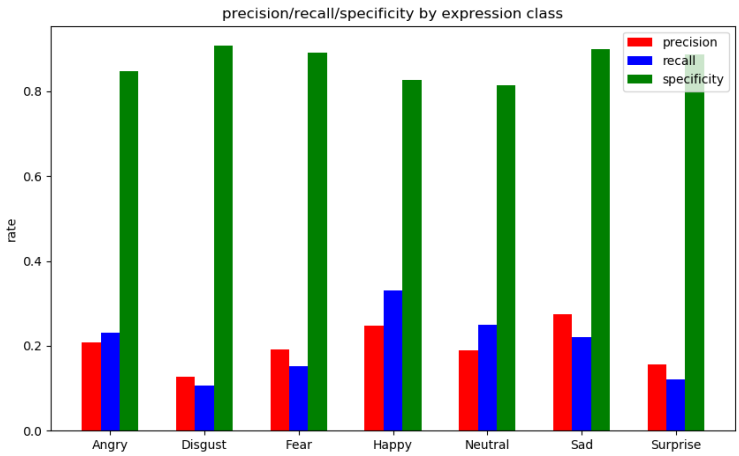

Fig. 4. The overall Precision, Recall, and Specificity for each expression class, PHOG dataset.

3.2 Sample Visualization from Data Loader

To get a sense of what samples are been used for inputs of the model, a batch images of training data loader are visualized as Figure 5. As mentioned above, these images are center cropped and resized to square 299*299 pixels and normalized before adding to data loader. Thus, to visualize them, we need to first de-normalize the values.

Fig. 5. Samples Visualization after Center Crop and Resize

3.3 Model Training and Rough Evaluation of Hyperparameters

For each hyperparameter set, a rough evaluation of its performance is implemented by the mentioned randomly split training-testing pair. The testing accuracy of each of these hyperparameter set are recorded in the following Table 1. Also, the settings of other hyperparameters are as: Training Epoch Number 50, Initial Learning Rate 0.01, and Step Size of Learning rate Schedular as 5.

Hyperparameter Set Index	Batch Size	Momentum Value of SGD Optimizer	Decay Multiplier of Learning rate Schedular.	Testing Accuracy
1	1	0.1	0.5	16.29%
2	2	0.1	0.5	37.78%
3	5	0.1	0.5	66.77%
4	10	0.1	0.5	61.58%
5	5	0.3	0.5	68.90%
6	5	0.5	0.5	71.95%
7	5	0.7	0.5	**74.69%**
8	5	0.9	0.5	**78.96%**
9	5	0.9	0.3	**75.61%**
10	5	0.9	0.7	**80.18%**

Table 1. Value of each of hyperparameter set with corresponding testing accuracy.

The loss variation plot for training-testing process of above 10 trials, with testing Precision, Recall, and Specificity for each expression class for each trial, can be found in the Appendix B Table 4.

From above table, we pick out the first three hyperparameters sets which gives the best testing accuracy, which are the 7^{th}, 8^{th}, 9^{th}, and 10^{th} set, for later cross-validation evaluation.

3.4 Cross-Validation of selected Hyperparameters

For the 7^{th} set of hyperparameters, the overall accuracy for the 5-folder cross-validation is 40.00%, with the corresponding values of precision, recall, and specificity for each class recorded in Appendix C Table 5 and visualized as follows Figure 6.

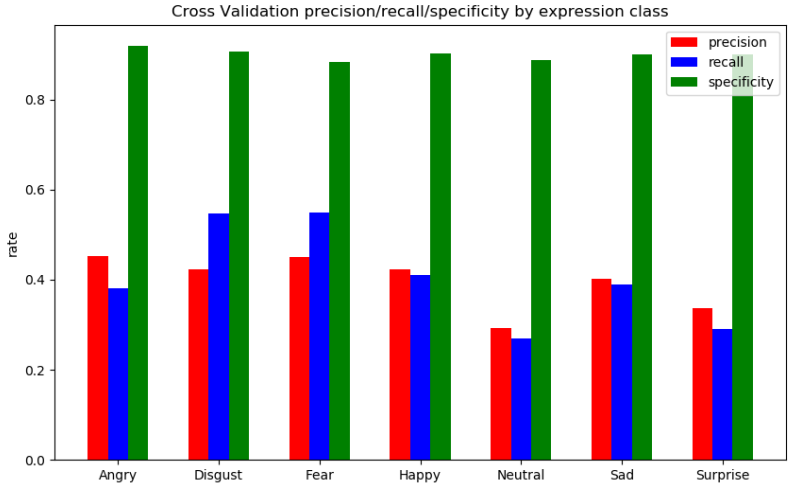

Figure. 6. precision, recall, and specificity for each class by model implementing the 7th hyperparameter set

For the 8th set of hyperparameters, the overall accuracy for the 5-folder cross-validation is 41.85%, with the corresponding values of precision, recall, and specificity for each class recorded in Appendix C Table 6 and visualized as follows Figure 7.

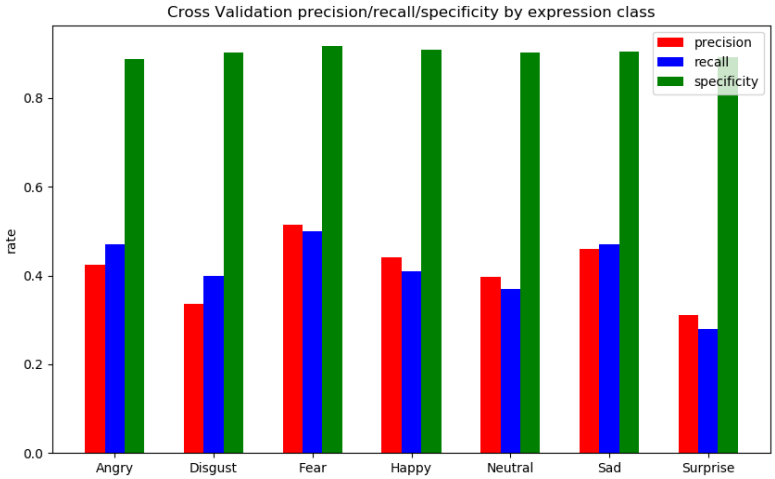

Figure. 7. precision, recall, and specificity for each class by model implementing the 8th hyperparameter set

For the 9th set of hyperparameters, the overall accuracy for the 5-folder cross-validation is 38.51%, with the corresponding values of precision, recall, and specificity for each class recorded in Appendix C Table 7 and visualized as follows Figure 8.

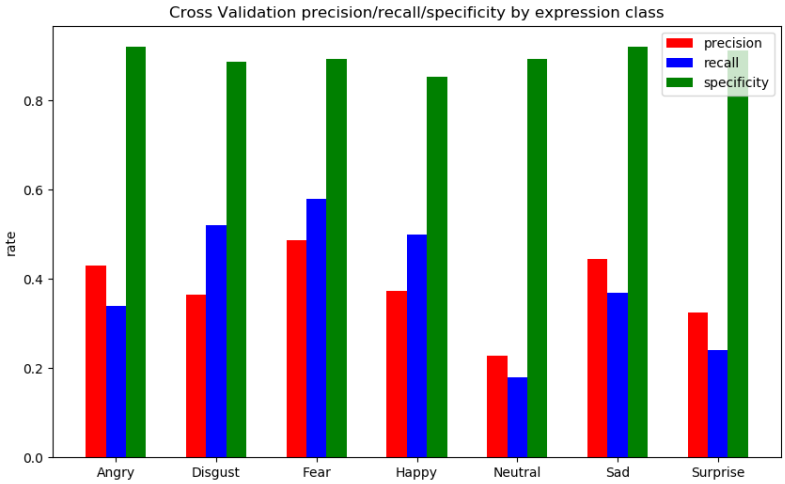

Figure. 8. precision, recall, and specificity for each class by model implementing the 9th hyperparameter set

For the 10th set of hyperparameters, the overall accuracy for the 5-folder cross-validation is 41.03%, with the corresponding values of precision, recall, and specificity for each class recorded in Appendix C Table 8 and visualized as follows Figure 9.

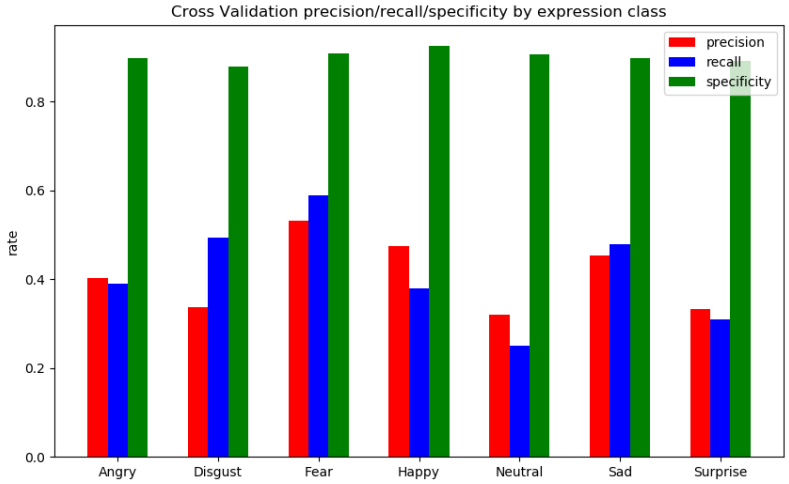

Figure. 9. precision, recall, and specificity for each class by model implementing the 10th hyperparameter set

3.5 Discussions

The results from simple feed-forward neural net achieved about half of the overall accuracy performed by SVM classifier implemented in original paper, [1] which is 43.71% for LPQ components and 46.28% for PHOG components. The precision, recall and sensitivity rates for each separate class are closer to SVM classifier, but still worse. There are several potential reasons for that. First, as mentioned in original paper [1], LPQ and PHOG might not be appropriate descriptors for this task as samples in SFEW dataset are collected in a more realistic environment and it's more challenging task to extract their features. Second, the feed-forward neural net is quite naïve and requires improvement and more complicated structures should be involved.

As for the transferred ResNet Model, we can see a significant drop of accuracy from the training-testing dataset (about 75%) to the cross-validation dataset (about 40%). The main reason for that, as mentioned before, is the hyperparameter sets selected for cross-validation are only well fit for the specific shuffled, split dataset. When the dataset is reshuffled and every single sample of whole dataset is validated, such hyperparameter set won't work as well as before. That's why we need to experience as many hyperparameters as possible when we train the model and it's also one of the directions of future work.

Finally, by taking a closer look at the dataset, there is another issue that makes this task difficult: sometimes its really hard to classify the face emotion even if the picture is clear. For example, I won't be able to classify this image into class 'fear' if I am not told the true label. This is much more difficult than distinguishing between cats and dogs for human brain, which is also kind of a feature extractor.


Figure. 10. OceansTwelve_002551440_00000064.png of class Fear

4 Conclusions and Future Work

4.1 Conclusions

The simple feed-forward neural network with principle components from descriptors LPQ and PHOG failed to show better performance than the SVM classifier developed in the cited paper [1] with regard to this expression classification task. And similar to SVM classification results, the model performs best with Happy expression class and has difficulty

handling Disgust class. Also, model trained by PHOG components have a slight edge over model trained by LPQ in terms of precision and recall.

For the transferred ResNet model, the best overall accuracy achieved is by the 8th set of hyperparameters, which is 41.85%. This accuracy is more than twice as the best result of simple feed-forward net, and the recall and precision rates of each class are more evenly distributed. These results all indicate that implementing a deep neural network and taking original images as training set will significantly enhance the model's performance with regard to this multi-classification task. The reasons include better feature extraction for deeper convolutional neural net, better weights updating algorithm brought by ResNet and more information contained in real images than in principle components. Also, such accuracy is comparable to the overall accuracy performed by SVM classifier implemented in original paper, and it's quite possible to outperform the SVM classifier if the model is improved by further work.

4.2 Future Work

To better improve the performance of this model, there are several directions of further work. First, more hyperparameters need to be tested. The hyperparameters attempted in this assignment is far from enough to exploit the real potentialities of this transferred model.

Second, we can also try different patterns of transfer learning, or different models to be transferred. There are two main patterns of transfer learning, the pattern implemented in this assignment is taking the model transferred as a fixed feature extractor by freezing the inner weights of convolutional/pooling layers while training. However, we could also switch to another transfer learning mode by fine tuning these weights. Also, there are other models that can be transferred such as VGG net or Alex net.

The third method to improve model performance might be constructing a face detector. Such detector could exclude the noise of background and focus our training on the people's face profiles.

The last and the most important issue to be considered is the generalization problem. By comparing the accuracy from 3.3 training-testing results and 3.4 cross-validation results, we can draw the conclusion that a certain set of hyperparameters that gives good performance in a specific subset can't be equally fit with other subsets. Even if we use the cross-validation for the whole SFEW dataset to select a set of hyperparameters that fit with all its subsets, how can we ensure its performance on other unseen large datasets, say for instance, SFEW 2 and SFEW 3? This is the problem faced by every deep learning task and for now, huge massive of data and faster algorithm might be the best solution.

References

1. Dhall, A., Goecke, R., Lucey, S. and Gedeon, T., 2011, November. Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. In 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops) (pp. 2106-2112). IEEE.
2. Milne, L.K., Gedeon, T.D. and Skidmore, A.K., 1995. Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood. training, 109(81), p.0.
3. A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Acted Facial Expressions in the Wild Database. In Technical Report, 2011. 1, 2
4. V. Ojansivu and J. Heikkil. Blur Insensitive Texture Classification Using Local Phase Quantization. In Proceedings of the 3rd International Conference on Image and Signal Processing, ICISP'08, pages 236–243, 2008.
5. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, CVPR'05, pages 886–893, 2005.
6. Ng, H. W., Nguyen, V. D., Vonikakis, V., & Winkler, S. (2015, November). Deep learning for emotion recognition on small datasets using transfer learning. In Proceedings of the 2015 ACM on international conference on multimodal interaction (pp. 443-449).
7. M. J. Lyons, S. Akamatsu, M. Kamachi, and 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
8. T. Sim, S. Baker, and M. Bsat. The CMU Pose, Illumination, and Expression Database. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(12):1615–1618, Dec. 2003.
9. He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
10. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.

Appendix

A. Precision, Recall and Specificity of each expression class from Previous work

	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.19	0.1	0.22	0.24	0.12	0.11	0.22
Recall	0.18	0.04	0.30	0.37	0.12	0.07	0.21
Specificity	0.86	0.95	0.81	0.79	0.85	0.91	0.87

Table 2. Simple Feed-forward Net cross-validation Precision, Recall and Specificity of each expression class, LPQ dataset.

	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.21	0.13	0.19	0.25	0.19	0.28	0.16
Recall	0.23	0.11	0.15	0.33	0.25	0.22	0.12
Specificity	0.85	0.91	0.89	0.83	0.81	0.90	0.89

Table 3. Simple Feed-forward Net cross-validation Precision, Recall and Specificity of each expression class, PHOG dataset.

B. The loss variation plot for training-testing process of 10 trials, with testing Precision, Recall, and Specificity for each expression class for each trial are all in Figures_u6799959 folder of deliverables. The file names are as follows.

Hyperparameter Set Index	Loss Variation Plot	Precision, Recall, Specificity for each expression class
1	Figure 1	Figure 2
2	Figure 3	Figure 4
3	Figure 5	Figure 6
4	Figure 7	Figure 8
5	Figure 9	Figure 10
6	Figure 11	Figure 12
7	Figure 13	Figure 14
8	Figure 15	Figure 16
9	Figure 17	Figure 18
10	Figure 19	Figure 20

Table 4.

C. Values of precision, recall, and specificity for each class, for each cross-validated hyperparameter set.

	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.43	0.41	0.42	0.41	0.31	0.40	0.35
Recall	0.38	0.57	0.57	0.40	0.29	0.39	0.32
Specificity	0.90	0.88	0.86	0.88	0.86	0.87	0.87

Table 5. 7th set of hyperparameters

	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.42	0.37	0.49	0.42	0.40	0.44	0.32
Recall	0.45	0.39	0.48	0.40	0.38	0.45	0.30
Specificity	0.88	0.89	0.90	0.89	0.89	0.89	0.88

Table 6. 8th set of hyperparameters

	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.43	0.39	0.50	0.38	0.24	0.43	0.36
Recall	0.38	0.51	0.58	0.47	0.19	0.37	0.22
Specificity	0.90	0.88	0.89	0.88	0.89	0.91	0.90

Table 7. 9th set of hyperparameters

	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.40	0.36	0.57	0.47	0.31	0.47	0.37
Recall	0.39	0.47	0.59	0.38	0.28	0.48	0.36
Specificity	0.88	0.87	0.88	0.90	0.89	0.89	0.88

Table 8. 10th set of hyperparameters