Distinctiveness Pruning Based on Simple Neural Network

Boyuan Zheng

Research School of Computer Science, Australian National University

Abstract This report based on SFEW dataset and discussed distinctiveness pruning on a 4-layer convolutional neural network (CNN) combining with 4 fully connected network and the performance comparison between CNN, principal component analysis (PCA) and residual network (ResNet). Based on our experiment, we found our CNN model perform better in this emotion classification problem. Furthermore, the distinctiveness pruning could accelerate the learning process and promote the model to converge more quickly. This report also includes our experiment process and some problem we met during the experiment and the way we solved it.

Keywords: Distinctiveness pruning, Convolutional neural network, Neural network

1 Introduction

In order to restore real-word facial emotion detect environment, we select SFEW (Static Facial Expressions in the Wild) Dataset as data source. SFEW Dataset provides more realistic and natural emotions and covers facial expressions with various head poses, large age coverage and so on (Dhall et al., 2011).The dataset is composed of 675 images coming from different movies and classified into 7 directories representing 7 kinds of emotion: angry, disgust, fear, happy, sad, surprise and one neutral emotion. Many researches related to emotion classification and CNN have been done in recent years. Cao et al. (2019) used the EEG signal and CNN to classify the emotions; Abdul Qayyum et al. used a unique CNN to classify the emotions during the speech. Based on their research and SFEW dataset, we try to find out the questions:

How much improvement could CNN provide comparing with principal component analysis (PCA)?

How much improvement could distinctiveness pruning provide?

Performance comparison between residual network (ResNet)by implementing transfer learning

2 Method

The network architecture used in this report is a 4-layer convolutional neural network (CNN) combining with 4 fully connected network. In order to figure out best hyper parameters, suitable

activation function and optimizer, we adopt 5-fold cross validation. The pruning method is inspired by T.D. Gedeon, which prunes the “least different hidden neuron” (Gedeon T., 1995).

2.1 Data input and pre-processing

In data input part, we use os and tqdm these two libraries to load the images from various directories. The colourful images with the size of 576*720 are labelled by their directories, e.g. the images from the first directory would be labelled as 0. Then we pre-process the images into 227*227 size in order to reduce the computational cost and maintain the information at the same time. For later comparison part, we rescale the image size to 224*224 to fit the input of ResNet.

2.2 Define neural network

Convolutional neural network is efficient to extract features and usually used to deal with image inputs. To define a good convolutional neural network, the most important part is to determine the hyper parameter. we initially set the hyper parameter as: learning rate = 0.01, epoch = 300, batch size = 8 and a CNN which could extract 512 features with 6*6 size. After experiment, we found it is inappropriate and unnecessary to use these huge number of features and the learning rate is still too large to train a model, reducing the feature size could enlarge the batch size. In this case, we finally set the hyper parameter as: learning rate = 0.0006, epoch = 50, batch size = 32 and a CNN which could extract 64 features with 6*6 size (see table 1).

	Initial setting (output channel, output size, filter size, stride, padding)	Final setting (same as initial setting format)
Conv_1	(64, 55, 11, 4, 0)	(16, 55, 11, 4, 0)
Conv_2	(128, 26, 5, 2, 0)	(32, 26, 5, 2, 0)
Conv_3	(256, 24, 3, 1, 0)	(64, 24, 3, 1, 0)
	Max pooling: (256, 12, 2, 2, 0)	Max pooling: (64, 12, 2, 2, 0)
Conv_4	(512, 12, 3, 1, 1)	(64, 12, 3, 1, 1)
	Max pooling: (512, 6, 2, 2, 0)	Max pooling: (64, 6, 2, 2, 0)
Fc_1	512*6*6 to 4096	64*6*6 to 512
Fc_2	4096 to 2048	512 to 128
Fc_3	2048 to 512	128 to 64
Fc_4	512 to 7	64 to 7

Table 1: Hyper parameter setting

This changes also accelerate the training speed significantly. With less epoch and less inputs towards the fully connection layer, the total time to run a 5-fold cross validation reduces by at least 10 times and the time spend on each epoch reduces to about 2 seconds. As for choosing the activation function and optimizer, we initially used ReLU activation function with Adam optimizer, it could get reasonable result without distinctiveness pruning but the angel between each weight vector doesn’t change very much which is not suitable to implement distinctiveness pruning. In this case, we finally set the activation function and optimizer as Sigmoid and Adam.

2.3 implement pruning method

Pruning plays a signification role in neural network. Since extra neurons would contain nearly duplicated functionality as other existing neuron and slow down the speed of the network by increasing the network size. (Gedeon T., 1995) PyTorch already provide a dropout function to drop

some neurons according to customized dropout possibility. But this dropout function isn't smart enough since it might drop some functionally distinctive neurons. In this case, T.D. Gedeon presented a new kind of pruning method based on distinctiveness theory (Gedeon T., 1995). The main idea is calculating the distinctiveness of weight vector angle for each neuron and based on angle threshold to determine whether or not to remove that neuron. Like T.D. Gedeon (1995) said, distinctiveness pruning could be somehow "computationally relatively complex". Adding distinctiveness pruning would significantly increase the running time but this pruning method could dynamically eliminate the redundant neuron and help the model converge more quickly. There are 3 possible weights matrix provided in the article: input weights matrix, hidden to output weight matrix and full weights matrix. This report is focusing on using hidden to output weight matrix to calculate the distinctiveness angle.

we firstly add a mask in network class to filter the output like existing dropout function, but we found this method is not enough. Then we add a function to set pruned neurons' columns of weight matrix to zero as well as their gradient. In this part, we meet an error "RuntimeError: Leaf variable has been moved into the graph interior", which means some in-place operation occur and change the leaf node to non-leaf node. we fixed this problem by using ".detach().fill_(0)" instead of simply assigning the value.

2.4 evaluation

In evaluation part, we initially separated the input data into 2 parts, 80% for training and 20% for testing, then using confusion matrix to do evaluation. This method isn't reliable and has uncertainty, this would make the hyper parameter evaluation struggle. we modified the problem by implementing 5-fold cross validation. Although it needs much more time to run, its outcome would be more reliable and convincing.

3 Results and Discussion

After determined the parameter and network design, let's go back to the previous questions.

I. CNN vs PCA

we started with PCA method firstly, we used each top 5 principal components from 2 descriptors, LPQ and PHOG. And then implement a basic neural network with 450 training epochs to do classification (see Figure 2, x axis reflects epoch, y axis reflects loss).

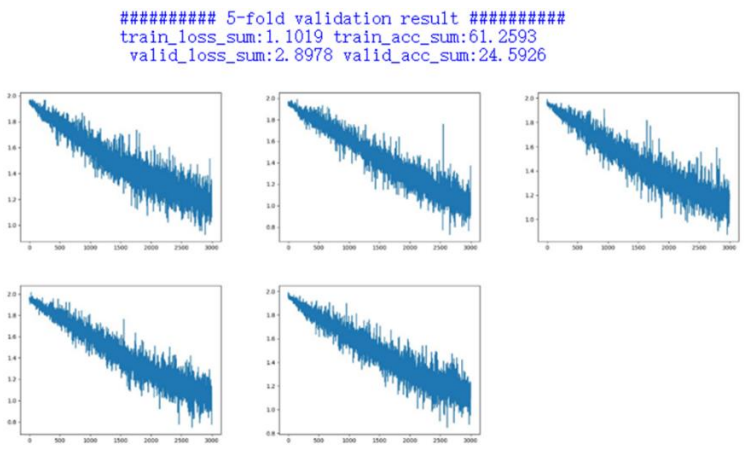
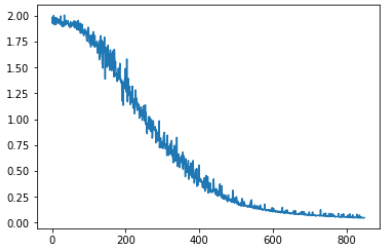


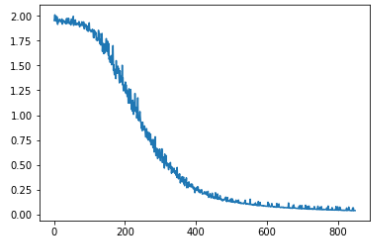
Figure 2: Result of PCA neural network without pruning

Then we used a 4-layer CNN combining with 4 fully connected network without implementing other methods, part of the result shows in Figure 3 (x axis reflects epoch, y axis reflects loss).

```
Epoch [1/50], Loss: 33.2095, Accuracy: 12.00 %
Epoch [6/50], Loss: 31.0687, Accuracy: 32.00 %
Epoch [11/50], Loss: 24.5657, Accuracy: 49.00 %
Epoch [16/50], Loss: 16.2981, Accuracy: 70.00 %
Epoch [21/50], Loss: 10.1902, Accuracy: 93.00 %
Epoch [26/50], Loss: 5.5390, Accuracy: 99.00 %
Epoch [31/50], Loss: 3.0984, Accuracy: 99.00 %
Epoch [36/50], Loss: 1.9514, Accuracy: 99.00 %
Epoch [41/50], Loss: 1.3908, Accuracy: 99.00 %
Epoch [46/50], Loss: 1.0572, Accuracy: 99.00 %
validation Accuracy: 37.00 %
```




```
Epoch [1/50], Loss: 33.3380, Accuracy: 12.00 %
Epoch [6/50], Loss: 32.0258, Accuracy: 30.00 %
Epoch [11/50], Loss: 24.3119, Accuracy: 67.00 %
Epoch [16/50], Loss: 13.6880, Accuracy: 90.00 %
Epoch [21/50], Loss: 6.7712, Accuracy: 98.00 %
Epoch [26/50], Loss: 3.3671, Accuracy: 99.00 %
Epoch [31/50], Loss: 2.0832, Accuracy: 99.00 %
Epoch [36/50], Loss: 1.4418, Accuracy: 99.00 %
Epoch [41/50], Loss: 1.0734, Accuracy: 99.00 %
Epoch [46/50], Loss: 0.8431, Accuracy: 99.00 %
validation Accuracy: 38.00 %
```




```
########## 5-fold validation result ##########
valid_loss_sum:13.2105 valid_acc_sum:35.0000
```

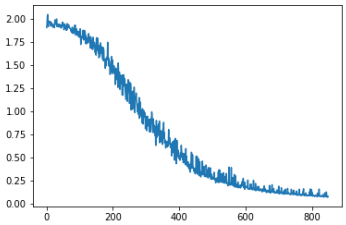
Figure 3: Result of CNN without pruning

By comparing Figure 2 and Figure 3, we can conclude that CNN uses less training epoch to train the network but CNN needs more time to run, and for this particular case, CNN might be better to classify the emotions, since the model learns relatively quicker and validation accuracy is relatively higher.

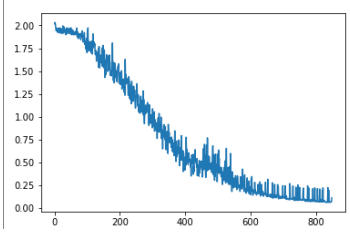
II. CNN with pruning vs. CNN without pruning

In this section, we aim to compare the CNN with pruning and the CNN without pruning. In order to interpret the significance of distinctiveness pruning, we changed the angel threshold from 15° to 25°. Because of the significant amount of calculation (there are 512 hidden neurons in first hidden layer and they would generate 130,816 angel combinations), we only display the result of CNN with pruning on second hidden layer (see Figure 4, x axis reflects epoch, y axis reflects loss), and due to the same reason, we didn't plan to present the graph about the angle changing tendency.

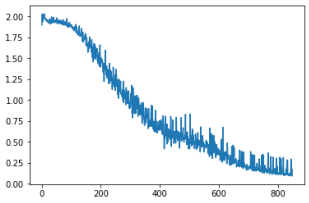
```
Epoch [36/50], Loss: 3.0711, Accuracy: 99.00 %
minimum angle between vectors: 29.261536961855178
maximum angle between vectors: 145.95609334746214
No neuron is pruned!
minimum angle between vectors: 28.597638647496787
maximum angle between vectors: 146.74921847891946
No neuron is pruned!
Epoch [41/50], Loss: 2.1979, Accuracy: 99.00 %
minimum angle between vectors: 28.08270233763635
maximum angle between vectors: 147.3986953058122
No neuron is pruned!
Epoch [46/50], Loss: 1.6754, Accuracy: 99.00 %
minimum angle between vectors: 27.573524359634654
maximum angle between vectors: 148.0392554185968
No neuron is pruned!
minimum angle between vectors: 27.163110800814344
maximum angle between vectors: 148.65073072759083
No neuron is pruned!
validation Accuracy: 38.00 %
```



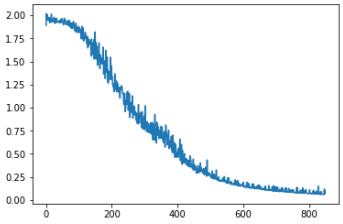

```
No neuron is pruned!
Epoch [36/50], Loss: 3.4517, Accuracy: 99.00 %
minimum angle between vectors: 26.495040619825122
maximum angle between vectors: 152.0607687337037
pruned neuron(s): {152, 157, 190, 231}
minimum angle between vectors: 26.1070290743351
maximum angle between vectors: 152.52562083317574
pruned neuron(s): {231, 152, 157, 190}
Epoch [41/50], Loss: 2.2755, Accuracy: 99.00 %
minimum angle between vectors: 25.865391101481446
maximum angle between vectors: 152.83211694661398
pruned neuron(s): {231, 152, 157, 190}
Epoch [46/50], Loss: 1.6964, Accuracy: 99.00 %
minimum angle between vectors: 25.769683644494414
maximum angle between vectors: 153.26034643867942
pruned neuron(s): {231, 152, 157, 190}
minimum angle between vectors: 25.70773868293969
maximum angle between vectors: 153.5555354639147
pruned neuron(s): {231, 152, 157, 190}
validation Accuracy: 40.00 %
```



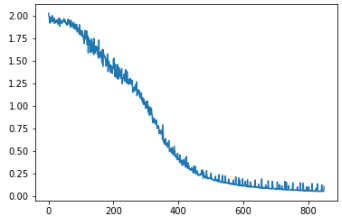

```
No neuron is pruned!
minimum angle between vectors: 25.52451100845975
maximum angle between vectors: 146.03017867207146
No neuron is pruned!
Epoch [36/50], Loss: 6.4931, Accuracy: 97.00 %
minimum angle between vectors: 25.428148933677644
maximum angle between vectors: 147.77488289681696
pruned neuron(s): {21}
minimum angle between vectors: 25.02874012314579
maximum angle between vectors: 149.7014442022151
pruned neuron(s): {88, 21}
Epoch [41/50], Loss: 3.8774, Accuracy: 99.00 %
minimum angle between vectors: 25.109702086449435
maximum angle between vectors: 150.7513491641408
pruned neuron(s): {88, 77, 21}
Epoch [46/50], Loss: 2.8775, Accuracy: 99.00 %
minimum angle between vectors: 25.25005874450288
maximum angle between vectors: 151.46188632701288
pruned neuron(s): {68, 21, 88, 58, 77}
minimum angle between vectors: 25.2589999092647
maximum angle between vectors: 151.9555608756191
pruned neuron(s): {68, 105, 77, 84, 21, 53, 88, 58, 91}
validation Accuracy: 37.00 %
```



```
maximum angle between vectors: 147.64240228135768
No neuron is pruned!
minimum angle between vectors: 26.699055611297844
maximum angle between vectors: 148.74495701397439
No neuron is pruned!
Epoch [36/50], Loss: 2.6249, Accuracy: 99.00 %
minimum angle between vectors: 25.787736909795544
maximum angle between vectors: 149.50805165354924
No neuron is pruned!
minimum angle between vectors: 25.04250761745501
maximum angle between vectors: 150.1617777334098
No neuron is pruned!
Epoch [41/50], Loss: 1.8829, Accuracy: 99.00 %
minimum angle between vectors: 25.99605663487425
maximum angle between vectors: 150.6110611002492
pruned neuron(s): {103}
Epoch [46/50], Loss: 1.4630, Accuracy: 99.00 %
minimum angle between vectors: 25.492528593990578
maximum angle between vectors: 151.48422590304997
pruned neuron(s): {93, 103}
minimum angle between vectors: 25.15972873240117
maximum angle between vectors: 152.91277338548045
pruned neuron(s): {88, 93, 103}
validation Accuracy: 38.00 %
```



```
minimum angle between vectors: 26.9433609646478
maximum angle between vectors: 149.53053177360994
No neuron is pruned!
Epoch [36/50], Loss: 2.0301, Accuracy: 99.00 %
minimum angle between vectors: 26.2227202678078
maximum angle between vectors: 150.2373938262232
No neuron is pruned!
minimum angle between vectors: 25.612082885328192
maximum angle between vectors: 150.78738650915173
No neuron is pruned!
Epoch [41/50], Loss: 1.4896, Accuracy: 99.00 %
minimum angle between vectors: 25.100735370364134
maximum angle between vectors: 151.26220644600045
No neuron is pruned!
Epoch [46/50], Loss: 1.1677, Accuracy: 99.00 %
minimum angle between vectors: 26.210119383282663
maximum angle between vectors: 150.01361358723406
pruned neuron(s): {109, 95}
minimum angle between vectors: 26.699116418037224
maximum angle between vectors: 151.6600051253184
pruned neuron(s): {109, 22, 95}
validation Accuracy: 37.00 %
```



```
########## 5-fold validation result ##########
valid_loss_sum:12.9578 valid_acc_sum:38.0000
```

Figure 4: Result of CNN with pruning

Although the result could vary from time to time, the above result shows that the result of CNN with pruning would get slightly better result than ordinary CNN. And one thing we found really interesting is the pruning process usually generates a turning point in loss graph while ordinary CNN tends to be smoother. We think the turning point might interpret the pruning process accelerates the convergence of the loss graph and show the significance improvement by pruning the similar function neurons. This finding might support T.D. Gedeon's theory about distinctiveness pruning. Comparing with T.D. Gedeon's result, this report finds the same conclusion. While the pruned neuron isn't too much, the result would be similar to the neural network which haven't done pruning. With less hidden neurons, the network could learn faster and save significant time for restarting the training process from scratch ((Gedeon T., 1995).

III. CNN from scratch vs. ResNet by transfer learning

Transfer learning is like "buying the suitable shoes" and training the network to fit the particular situation. There are mainly 2 approaches to adopt transfer learning: Finetuning the network; use the network as fixed feature extractor. In this report, we use finetuning as an example. Since the computational limitation, we only experiment the ResNet with 18, 34, 50 layers (Chilamkurthy, 2020), the result shows in Table 2 (result image in Appendix).

	CNN	ResNet18	ResNet34	ResNet50
Training accuracy	99%	99%	99%	99%
Testing accuracy	35%	33%	31%	34%

Table 2: comparison between CNN and ResNet

It is very strange that the transfer learning performs worse than CNN from scratch. We think the possible reason might be the SFEW dataset is composed of the more real-world image, so that it might contains more real-world information as noise. In this case, these models might not perform enough well as we expected.

4 Conclusion and Future Work

This report discussed distinctiveness pruning on a 4-layer CNN combining with 4 fully connected network and the performance comparison between various network. Based on experiment we found: PCA with ordinary neural network couldn't perform as well as CNN on this emotion classification problem, this situation might be caused by PCA provided limit number of features; distinctiveness pruning performs well in this particular case and it usually generate a turning point in loss graph; by comparing the CNN with 3 kinds of ResNet, we think the SFEW dataset highlight the real-world situation so that transfer learning perform not enough well in this task.

The result in this report exists high degree of overfitting problem, future research could implement methods like regional separation to solve the overfitting problem. Besides this, pruning makes improvement on network learning speed but couldn't improve the performance, the future work could focus on modifying neuron's behaviour by modifying their distinctiveness angle instead of dropping neurons. Maybe when the neuron's vector angle reaches the threshold, future method could turn similar neuron to other direction.

5 Reference

Gedeon, T., 1995. Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems.

Dhall, A., Goecke, R., Lucey, S., Gedeon, T., 2011. Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. Proceedings of the IEEE International Conference on Computer Vision.

Cao, G., Ma, Y., Meng, X., Gao, Y. and Meng, M., 2019. Emotion Recognition Based On CNN. 2019 Chinese Control Conference (CCC).

Abdul Qayyum, A., Arefeen, A. and Shahnaz, C., 2019. Convolutional Neural Network (CNN) Based Speech-Emotion Recognition. 2019 IEEE International Conference on Signal Processing, Information, Communication & Systems (SPICSCON).

Chilamkurthy, S., 2020. Transfer Learning For Computer Vision Tutorial — Pytorch Tutorials 1.5.0 Documentation. [online] Pytorch.org. Available at: <https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html?highlight=transfer> [Accessed 28 May 2020].

6 Appendix

ResNet18:

```
Epoch 23/24
----------
train Loss: 0.2624 Acc: 0.9982
val Loss: 1.9391 Acc: 0.3277

Epoch 24/24
----------
train Loss: 0.2521 Acc: 1.0000
val Loss: 1.9468 Acc: 0.3109

Training complete in 2m 43s
Best val Acc: 0.327731
```

ResNet34:

```
Epoch 23/24
----------
train Loss: 0.1289 Acc: 0.9943
val Loss: 1.8879 Acc: 0.2867

Epoch 24/24
----------
train Loss: 0.1275 Acc: 0.9981
val Loss: 1.8653 Acc: 0.3000

Training complete in 3m 40s
Best val Acc: 0.306667
```

ResNet50:

```
Epoch 23/24
----------
train Loss: 0.3146 Acc: 1.0000
val Loss: 1.8638 Acc: 0.3007

Epoch 24/24
----------
train Loss: 0.3090 Acc: 0.9962
val Loss: 1.8632 Acc: 0.3077

Training complete in 5m 21s
Best val Acc: 0.335664
```