Facial Expression Recognition with Convolutional Networks

DSC672 Project Report

Zhong Zhong

Facial Expression Recognition with Convolutional Networks

Zhong Zhong, zzhong4@depaul.edu

November 24, 2020

Abstract

Researchers have been studying facial expression recognition for a long time, because it is important in a broad of fields, such as robotics, medical treatment, and visual-interactive games. Recently, deep learning models, especially Convolutional neural networks (CNNs), have become the dominant machine learning approach for image recognition tasks [11]. With the use of CNN models, computer vision tasks as facial recognition can often achieve state-of-the-art (SOTA) accuracy. In this project, a dataset, Facial Expression Recognition challenge (FRE2013), which contains seven main human facial expressions, Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral, is applied on different CNN models that we build from scratch based on the principles of previous SOTAs. To further study how the CNN models work, we also include activation map visualizations for the convolutional network model.

1 Introduction

Human facial expressions play a important role in human information exchange process. With the fast development of artificial intelligence, it is enormously beneficial for Artificial Intelligence devices or programs to be able to recognize human's facial expressions. Such AI products can be applied to a broad of different domains, such as robotics, medical treatment, and visual-interactive games.

Deep convolutional networks have recently achieved outstanding performance in various different visual tasks. It has been proved that CNNs are more robust to the nature of image data compared to conventional machine learning models. We believe that SOTA CNNs will achieve well performance on facial expression recognition as well. Therefore, in this project, we present three different approaches based on Convolutional network for the facial expression recognition task, FRE2013.

There has been a variety of SOTA CNNs presented. Many of them tried to modify the earlier SOTAs in order to improve the overall accuracy. On common strategy to improve architecture accuracy is to add more layers to the model. We embrace this method and construct a deeper network on our first baseline LeNet-like model. Another strategy is to use special layer module designs, such as inception module, shortcut connection, and dense connection. To further improve model accuracy, we also try to use shortcut connection in our project.

2 Related Work

2.1 Convolutional Networks

Ever since AlexNet [1] after 2012, convolutional Networks, such as AlexNet, VGGNets [2], Inceptions [3], ResNets [4], MobileNet [5] and most recently purely supervised network EfficientNets [6], are the dominate machine learning methods for visual tasks. They all achieved state-of-the-art performance on the benchmarked image classification task, ImageNet Classification challenge. Also, they have achieved one after another state-of-the-art performance on other visual tasks, such as image segmentation, object detection and so on.

Later models were built on top of earlier models and improved using different useful methods. One common method that researchers used is to add the more layers or more feature maps to the architecture, for example, VGGNets outperformed AlexNet by adding up to 11 more convolution layers. Another example is MobileNets largely improved the efficient and accuracy compared to models with similar size by adding Width Multiplier and Resolution Multiplier that can adjust the number of channels and layers in the model. Another common method to improve architecture is to use skip connections/shortcut connection. ResNets first introduced the simple identity connection which popularized this method. Skip connections can be seen in almost all different architecture after ResNets.

2.2 Facial Expression Recognition

Several works [7] [8] [9] have studied the Facial Expression Recognition Challenge dataset. All these works adopted convolutional network as their methodology. At first, a base small CNN model was built, and modifications such as increasing the depth and the width were applied on top of the base model to improve the model performance. [9] tried to use transfer learning by both fine-tuning AlexNet and VGGNet and training from scratch, but they found these two models trained on ImageNet dataset performed unexpectedly due to the much smaller size (48*48) and less variance compared to the ImageNet dataset.

3 Data

The dataset was obtained from Kaggle Facial Expression Recognition (FER2013) Challenge by Microsoft. It consists of 35,887, 48x48 pixels gray-scaled images of different human faces, in which each face corresponds to one of the seven main human facial expressions, namely Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral. The images are processed in such a way that the faces are centered, Figure 1 shows an example of each expressions. There are 28,709 training images and 3,589 test images. After loading the raw images, we normalize them by re-scaling to the range between 0 and 1. Further, in order to increase the variance of the data and alleviate the over-fitting problem, three different data augmentations, random flipping, rotating and zooming are used.

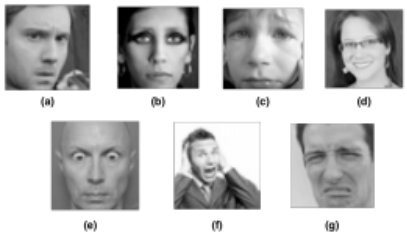

Figure 1: This is the example of facial expressions from FER2013.

4 Method

Recognizing facial expression images is essentially an image classification task. Therefore, we believe a SOTA image classification CNN architecture can also perform well on facial expression classification. We tried different CNN SOTA models, including LeNet5, VGGNets, and ResNets. Because images in FER2013 are smaller than that in ImageNet dataset used to train those SOTA, we customed above architectures to create three new versions, namely, LeNet-like, VGGNet-like, and ResNet-like. However, all the customed versions share some characteristics. First for each convolutional layer, the filter size is 3 and the stride is 2. Second, each convolutional and fully connected layer, except the output layer, follow with Batch Normalization in order to speed up the training process and add more regularization . (See Appendix A for model details)

4.1 LeNet-like CNN model

The plain baseline mode is inspired by the principle of LeNet5 [10]. In the LeNet5 model, the input data has a size of 28x28, its size is very close to the image size in our dataset, FER2013. Although the FER2013 facial recognition task is more complicated task than the handwritten digit recognition in LeNet5. We think it is a good start point to gradually build more complex and better models.

The LeNet-like network consists of five weighted layers, three convolutional layer and two fully connected layers. Max pooling layer is used after every convolutional layer in order to perform downsampling. The network ends with a 7-way fully connected layer with softmax. (see Figure 8)

4.2 VGGNet-like CNN model

VGGNets outperform AlexNet by simply using small size of filters and strides and more importantly increasing the number of layers in the network. I add more layers and feature maps to the LeNet-like CNN model and make it a much deeper 10-layer network (see Figure 9).

4.3 ResNet-like CNN model

According to ResNets, identity connections (Figure 2) can not only alleviate the gradient vanishing problem, but also improve the overall architecture perfor-

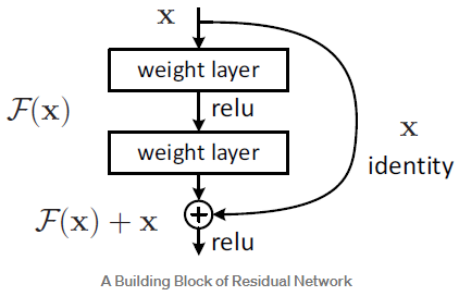

Figure 2: This is the shortcut connection in ResNet.

mance. Based on the above VGGNet-like model, we insert shortcut connections which turn the network into its counterpart residual version (see Figure 10). In order to match the input size of the shortcut with its output, a projection (1x1 convolution) is used to increase the number of feature maps.

5 Experiment Results

5.1 Data augmentation

The LeNet5-like model first appears to overfit the dataset at first. Figure 3exhibits the results of this model. this 5-layer LeNet-like network achieves 51.0 percent accuracy on test set, the training accuracy, however, arises to 80 percent very quickly. To reduce the significant overfitting problem, we tried a few data-augmentation techniques. For example, we randomly select some images to flip horizontally before feeding into the network and another example is to rotation the image before feeding into the model. By doing these data augmentation, the total images entered to the model will be significantly increased, thus the model have more data samples to train. This data augmentation technique largely reduces the gap between the train and test accuracy. (Figure 4)

5.2 Shallow vs Deeper Models

To compare the performance of the shallow model (LeNet-like) with the deeper model (VGGNet-like), we plotted the loss history and the obtained accuracy in LeNet5-like network and VGGNet-like network. Figures 5 exhibits the results.

The most obvious difference between LeNet5-like network and VGGNet-like network is their depth. LeNet5-like network has 5 layers, whereas VGGNet-like has 10. The key difference leads to the performance discrepancy between these two models. As seen in 5, the deep network enabled us to increase the validation accuracy by 3-4 percent.

5.3 Shortcut Connections

To achieve better accuracy, we insert shortcut connections to the VGGNet-like model. An extra 1x1 convolutional layer is also added to the shortcut inputs to

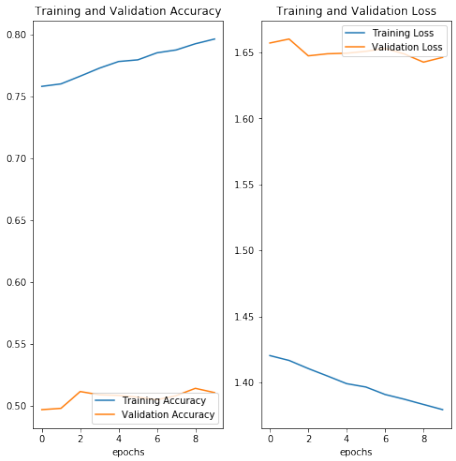

Figure 3: This is results of LeNet-like model without regularization techniques

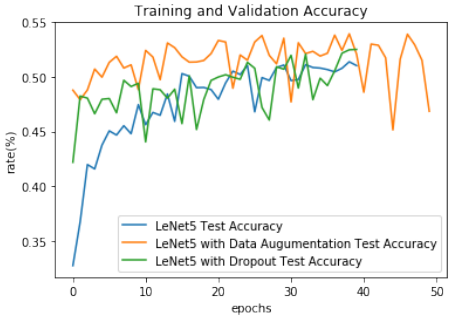

Figure 4: This is the results of LeNet-like model with regularization techniques

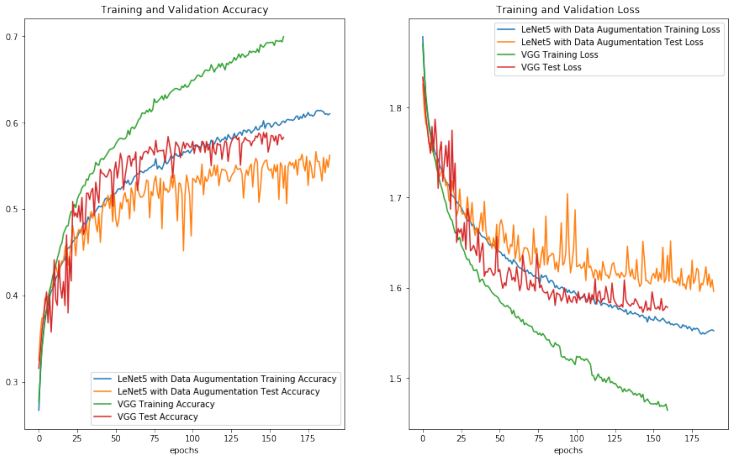

Figure 5: This is results of LeNet-like model vs VGGNet-like model

Figure 6: This is results of VGGNet-like vs ResNet-like

match the output size. Figure 6 and table exhibit the results.

We have the major observations from Table 2 and Fig. 4. First, the VGGNet-like with shortcut connections is better than its original format without shortcut connections, as it is seen in table, compared to its plain counterpart, the model with shortcut connection increases the accuracy by more than 2 percent. Second, we also note that the model with shortcut connections converges faster (Figure 6 Right vs Left). The model eases the optimization by providing faster convergence.

5.4 Visualize the CONVNets

Visualizing the output of a model is way to understand how machine learning models work, Same to CNN models. To study how the deep CNN model can classify the input image, we visualize the intermediate layers activations of the model.

The activation map is shown in Figure 7. The top layer maintains the shape of the original image even though several feature maps are not activated and left blank. At this stage, the layer activations keep almost all of the information present in the input picture. And, as it goes deeper, the layer activations become increasingly abstract and less visually interpretable. According to [12] This is because deeper layers begin to encode higher-level feature such as single nose, mouth, or eyes. Higher presentations carry increasingly less information about the visual contents of the image, but including more information related to the class the original image.

By visulizing the layer activations, we learned that early layers learn simple features, whereas deeper layers appear to capture more complex features that is useful for classifying the image categories. This characteristic of convolutional layers actually explain adding more layers to a model will help improve model performance.

6 Conclusion

In this project, we developed three different CNNs for a facial expression recognition problem and evaluated their performances using visualization techniques. We found that both using deep models and adopting shortcut connection can

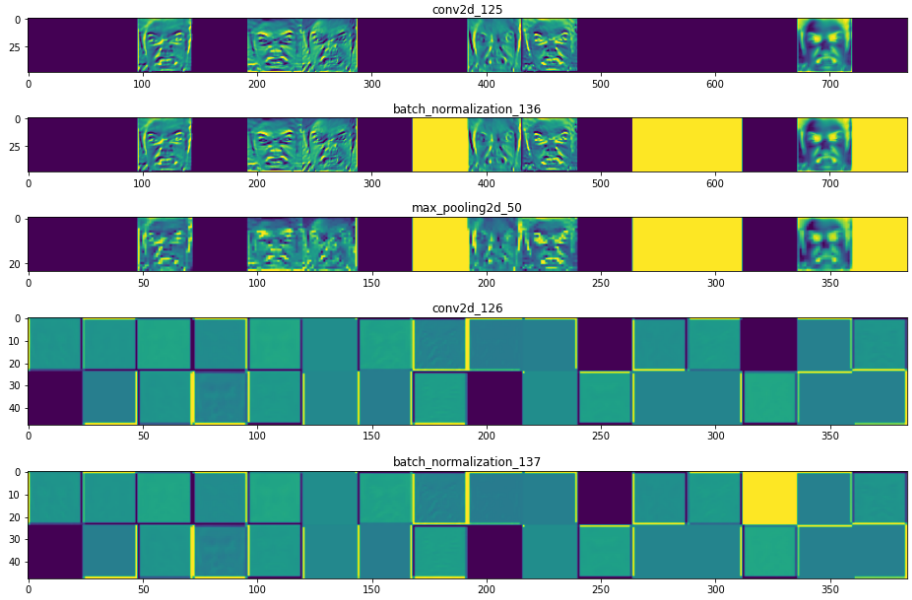

Figure 7: Visualization of intermediate layer activations

Model	Test Accuracy(percent)
LeNet5-like	51
LeNet-like with data Augmentation	53
VGG-like	56
ResNet-like	58

Table 1: Test Accuracy.

improve CNN performance. Due to limited time and computational resource, the models trained in this projects achieved as high as 58 percent test accuracy (see Table 1). For feature works, more complex can be applied to this dataset to have better performance. Also, more data needs to be collected in order to train much larger models.

References

[1] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

[2] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[3] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[4] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[5] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[6] Tan, M., Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946.

[7] Shima, A., Fazel, A. (2016). Convolutional neural networks for facial expression recognition. ArXiv2016, 3.

[8] Yu, Z., Zhang, C. (2015, November). Image based static facial expression recognition with multiple deep network learning. In Proceedings of the 2015 ACM on international conference on multimodal interaction (pp. 435-442).

[9] Wan, W., Yang, C., Yang, L. (2017) Facial Expression Recognition Using Convlutional Neural Network A Case Study of The Relationship Between Dataset Characteristics and Network Performance.

[10] LeCun, Y. (2015). LeNet-5, convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet, 20(5), 14.

[11] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[12] Chollet, F. (2018). Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH Co. KG.

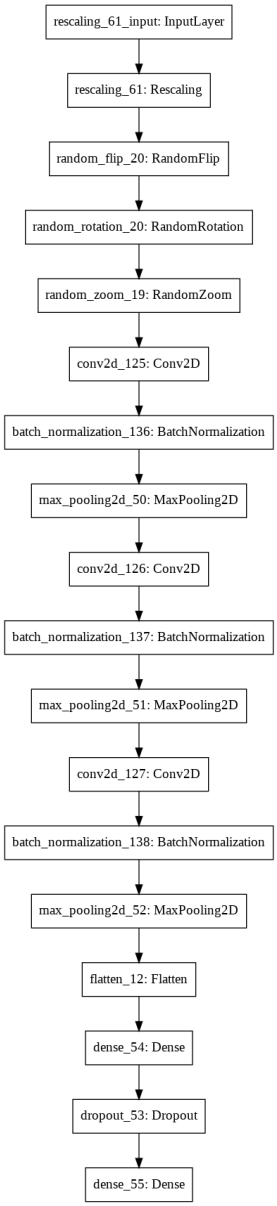

Figure 8: LeNet-like Architecture

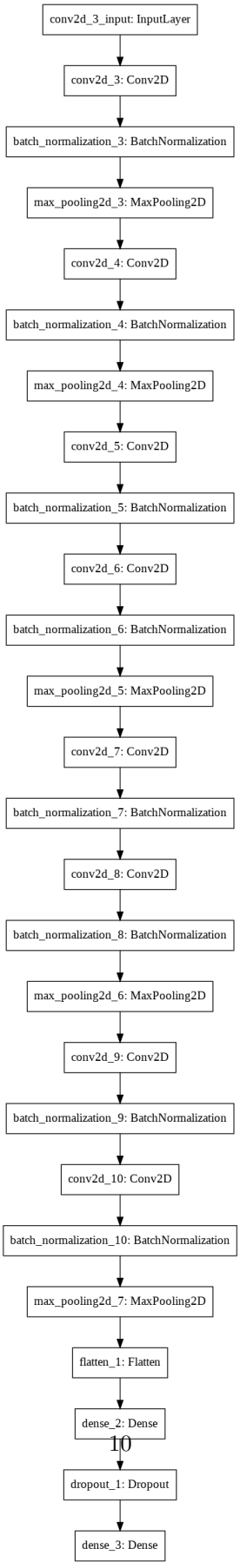

Figure 9: VGGNet-like Architecture

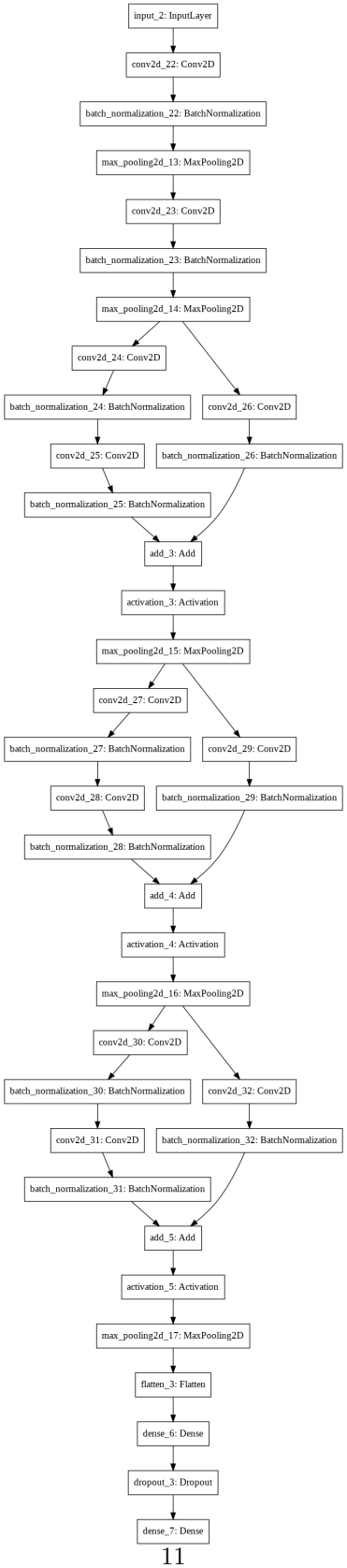

Figure 10: ResNet-like Architecture