Facial Expression Recognition Using Convlutional Neural Network
A Case Study of The Relationship Between Dataset Characteristics and Network Performance

Weier Wan
weierwan@stanford.edu

Chenjie Yang
yangcj@stanford.edu

Yang Li
yang8@stanford.edu

Abstract

Convolutional neural network has been adopted to achieve state-of-the-art performance across various tasks involving high dimensional data with local correlation. In our work, we apply convolutional neural network to facial expression recognition. We trained a network using images from Kaggle facial expression challenge, and were able to achieve a 65.3% *accuracy across 7 categories. Moreover, through a qualitative analysis of the dataset including its size, dimensionality and complexity, we studied the relationship between dataset characteristics and network performance, and tried to understand why certain architecture and training approach are preferred over the others based on dataset characteristics. We made a thorough comparison among different training and fine-tuning strategies, and network with different configurations.*

1. Introduction

Understanding human communication has been at forefront of artificial intelligence research for decades. While traditional approach typically relies on verbal-based natural language processing for understanding communication, in many cases, the intended meaning of spoken words has to be understood with the knowledge of the speakers emotion, which can be most directly reflected from humans facial expression. In recent years, due to the fast growing social networks, photos and videos that include faces constitute a great proportion of visual data on the Internet. Moreover, facial expression recognition can be applied to many other interesting applications such as human behavior research and human-computer interface.

With recent success of deep convolutional neural network in tasks involving locally correlated high dimensional data, we think deep convolutional neural network would be suitable for recognizing facial expression.

There are two main objectives for this project. First, we are interested in developing some intuition on how to choose a suitable network architecture and training technique for a given dataset through a qualitative high-level analysis of the data. The dataset will be understood qualitatively through a comparison with ImageNET dataset which was used to pre-train many models. Some characteristics of dataset that we look into include the dimensionality of the features, the size and resolution of images and the diversity and similarity of data. With these characteristics, we then try to understand how the dataset can affect the choice of an optimal network architecture and training techniques.

Second, we study how different regularization and data augmentation techniques will have effects on the learning performance. To do this we first construct a relatively simple network, and add different regularization techniques step-by-step to quantify their effectiveness, and finally we will construct a deeper network based on our previous observations to achieve a high accuracy.

Section 2 of the paper will briefly talk about the background and some early approaches people used for facial expression recognition. Section 3 will discuss in details the two tasks introduced above and our approach. Our results will be presented in section 4 followed by a detailed analysis. Section 5 will conclude the paper and points out some further improvement that can be done.

2. Background

Researchers have tried different approaches to recognize facial expression from unconstrained images. In general, face related tasks (e.g. identity recognition, emotion recognition) involve three steps: face detection, face modelling and classification. Early researchers tried to extract the best high-dimensional feature representation of faces [7] [3], and find most important set of features with dimension reduction techniques such as PCA and sparse learning. People have also construct a variety of 2-D and 3-D models for face that are suitable for the task. With many recent success in high-dimensional data learning, convolutional neural network has been adopted in both face modelling [6] and recognition [4] [2]. Yaniv [6] reports that using deep learning framework, they are able to generate compact 3-D representation of face that can be generalized across various unconstrained environment.

3. Approach

One thing that we are particularly interested in is to use this facial expression recognition task as a case study to develop some understanding on the interaction between dataset and network architecture and training techniques. The questions that we want to explore include: given a task and associated dataset, what network architecture (depth of the network, filter and stride size at each layers, etc) should we choose; how to tradeoff between training speed and complexity of the network; should we use a pretrained model or train a model from scratch; if using pre-trained model, what is the best fine-tuning strategy, etc.

We want to study these design decisions based on some qualitative analysis of the dataset. The dataset we will use for training and evaluating our network is the Kaggle facial expression challenge dataset. The dataset consists of 35887 48*48 gray-scaled images of faces divided into 28709 for training and 7178 for testing. Each face corresponds to one of the seven emotion categories, namely Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral. Figure 1 shows an example from each category.

(a) Angry (b) Disgust (c) Fear

(d) Happy (e) Sad (f) Surprise

(g) Neutral

Figure 1: Sample images from each category of Kaggle dataset

To study the effects of dataset characteristics on network performance, we conduct a few experiments. First, we want to compare several fine-tuning techniques and study their tradeoffs between speed and performance. In many cases, fine-tuning only the bottom fully connected layers while leaving all the convolutional layers frozen could lead to good performance. However, it is also natural to think that refining the shallow convolutional layers can lead to extra performance gain, while trading off training speed. We will try both approaches and understand the results based on the characteristics of Kaggle images in comparison with ImageNet.

The second question we look into is that how a model trained from scratch will perform compared to the model fine-tuned from existing models. In many cases, training a deep network from scratch will take very long time to converge, so we want to see with a limited number of epochs and reasonable amount of training time, is it possible to train a deep network that can perform equally well or even better than a pre-trained model, and how this has any implication on the relationship between data complexity and network structure.

We conduct the aforementioned experiments on both pretrained AlexNet [1] and VGGNet [5] so that we could know if the properties are network-specific or solely determined by the dataset. Since we use both networks, it is natural for us to also draw some conclusion on the performance of AlexNet vs. VGGNet, with a focus on the relationship between the performance discrepancy and the characteristics of the dataset.

4. Experiment

4.1. Architectures and Training Approaches

As discussed in the previous section, to develop some qualitative understanding of the relationship between dataset and the performance of various network architectures and training approaches, we applied different training approaches on both AlexNet and VGGNet. Figure 2 summarizes the results.

Fine-tuning: Freezing conv layers vs. Tuning entire network. Since training deep network (backpropagate through all the layers) is computationally intensive, and initial large gradient signal during fine-tuning tends to overly disturb pretrained weights, people normally freeze shallower layers during fine-tuning. Results have shown that fine-tune a model pre-trained with ImageNet dataset could achieve state-of-the-art performance across various image classification tasks. The key behind fine-tuning is that images usually share similar sets of basic constructing components such as edges and simple motifs.

In our experiment, we first try to follow this typical fine-tuning method. We only re-initialize fully-connected layers while setting the learning rate of convolution layers all to zero. We compare its result to the case where we do not freeze any layers, but instead setting the learning rate of convolution layers 1/10 of fully connected layers. For more thorough comparison, we also tried fine-tuning the bottom two convolution layers while leaving top conv layers frozen.

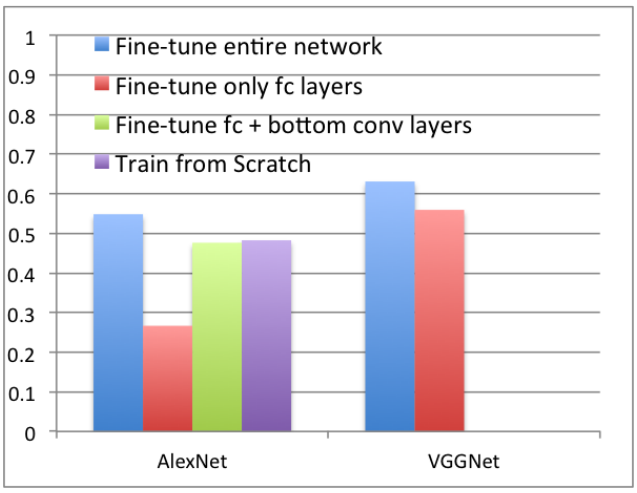


Figure 2: Summary of results obtained with AlexNet and VGGNet using different training approaches

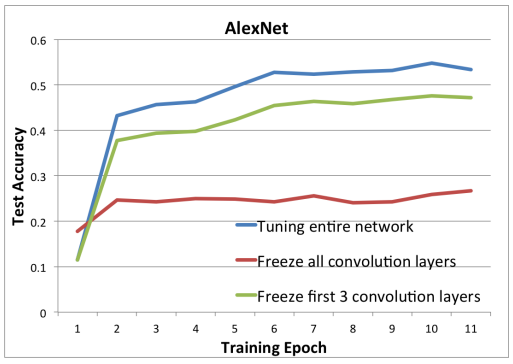


Figure 3: Results of AlexNet using different fine-tuning methods.

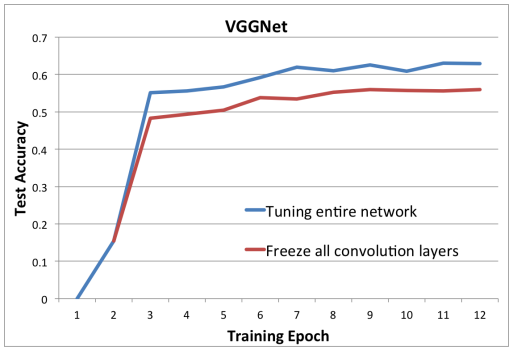


Figure 4: Results of VGGNet using different fine-tuning methods.

The results in Figure 3 and Figure 4 show that freezing convolution layers results in much worse performance, which suggests that the features selected by the pre-trained model do not represent the dataset well. We think this performance discrepancy is not only due to the fact that the Kaggle dataset content is very different from ImageNet, but more fundamentally because the size of each image in Kaggle is only 48*48 whereas the shallow layers of the model trained on ImageNet are optimized for recognizing finer features. For example, pixel-wise, the transition of edges in Kaggle is more abrupt due to low resolution. Therefore, tuning the entire network could help these shallow features to be better extracted.

Fine-tuning vs. Training from Scratch. Regardless of the training speed, fine-tuning can sometimes give better results than training a network from scratch, especially when the dataset is small and the network is deep. For small dataset, a deep network is usually able to over-fit to the training data, but fail to extract a good set of features that can be generalized across a wider range. Also, compared to shallow network, a deep network needs more training epochs to converge, each epoch also takes longer time, therefore training a deep network would require much longer period.

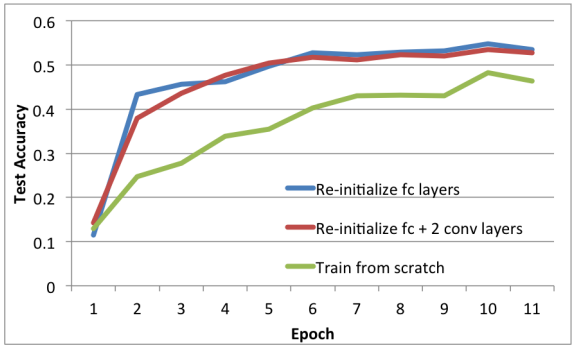


Figure 5: Test Accuracy of AlexNet fine-tuned using pre-trained model vs. trained from scratch

However, if the dataset is simple, the network can converge faster. Considering the case of a linearly separable dataset on a 2D plane, even a deep network can converge almost instantly. In this case, the network will learn to extract a more efficient set of features for the dataset than the features for ImageNet. This can partially explain why training a network from scratch for Kaggle can achieve similar accuracy to fine-tuning in reasonable number of epochs (Figure 5 and 6): The Kaggle dataset is constituted of centered and well-clipped gray-scale faces. The variation across images is small compared to ImageNet. Therefore we could expect the principal components of Kaggle dataset dominant and able to capture most information contained in an image. The dimensionality can then be largely reduced. During training, the network should need much less time

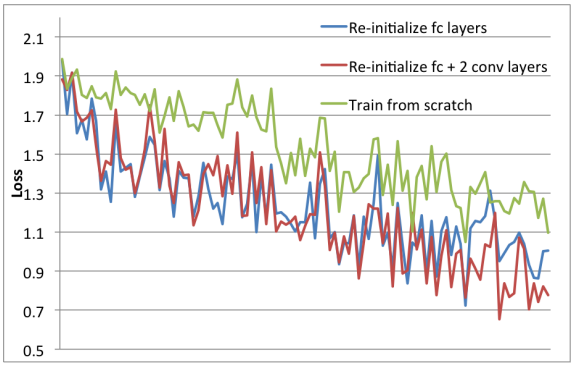


Figure 6: Loss history of AlexNet fine-tuned using pre-trained model vs. trained from scratch

to figure out an important set of features that can classify the images well. Indeed, figure 5 shows that even though the dataset is small, the network actually converges pretty fast, and could achieve a performance closed to the best accuracy obtained using fine-tuning. We could observe that the accuracy curve is still going upwards. With more training epochs, we would expect its accuracy to reach or even beat that of the fine-tuning.

One challenge of training a deep network is that at the beginning of training, if the initial bias and learning rate is not carelly selected, neurons at one layer may completely jump into the dead region. We encountered this problem while training both AlexNet and VGGNet. Interestingly, when this happens, we found the accuracy is always stuck at 25% independent of network and learning rate.

Besides, in the above two comparisons, we ignore training speed when evaluating performance. In practice, fine-tuning deeper into the network would require more computation. So there obviously exists a tradeoff between the accuracy and training speed.

AlexNet vs. VGGNet. Both AlexNet and VGGNet were winning architectures for ImageNet competition. Though both networks use similar conv-relu-pool-fc structure, they are different in many aspects. The most obvious difference is their depth. AlexNet has 8 layers while VGGNet has 16. However, we think this is not the main cause of the performance discrepancy. As discussed above, since Kaggle dataset has small variance and diversity, a deeper network may not give too much advantage in representational power. In fact, we train an 8-layer network that has similar architecture as 16-layer VGGNet. It can achieve an 62.8% accuracy on test set, which is closed to 63.1% accuracy of VGGNet.

Instead, the major difference between AlexNet and VGGNet that leads to the performance discrepancy is that AlexNet use large filter and stride size at the initial layer whereas VGGNet preserve the dimension of the input image. Preserving dimensionality may not lead to huge benefits for larger images, but can be crucial for small images (48*48) such as Kaggle. This also explains why using our 8-layer network could give much better results than an 8-layer AlexNet. Also, from the results in Table 1, we find VGGNet tends to overfit training data more than shallower network. This is expected because a deeper network would more easily over-fit to a small training set.

4.2. Regularization and Accuracy

	Training Accuracy	Testing Accuracy
AlexNet	61.7%	54.8%
VGGNet	89.4%	63.1%
8-layer-net	70.7%	62.8%
11-layer-net	74.1%	65.3%

Table 1: Summary of best results using different architectures

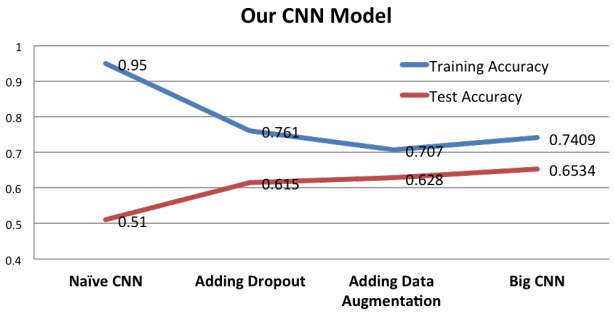


Figure 7: Improvement of regularization and accuracy with reguarization techniques

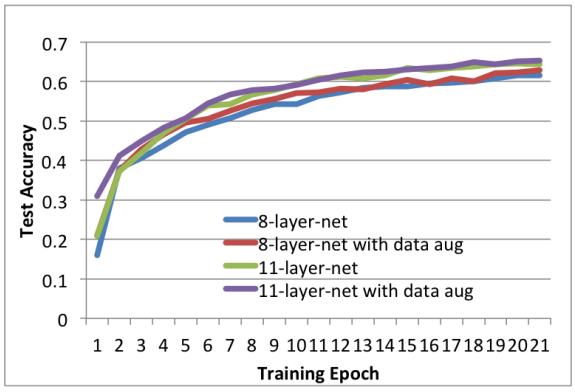


Figure 8: Test Accuracy of 8-layer and 11-layer network

To achieve highest accuracy, we first construct a relatively shallow network and use it to quantify the effective-

ness of various regularization and data augmentation techniques in reducing over-fitting and improving test accuracy. The architecture of this 8-layer network is shown in Table 2.

input data(48x48 grey scale image)
Data Augmentation
CONV $3 \times 3 \times 32$, RELU, BATCH NORM
CONV $3 \times 3 \times 32$, RELU, BATCH NORM
MAXPOOL 2×2
CONV $3 \times 3 \times 64$, RELU, BATCH NORM
CONV $3 \times 3 \times 64$, RELU, BATCH NORM
MAXPOOL 2×2
CONV $3 \times 3 \times 128$, RELU, BATCH NORM
CONV $3 \times 3 \times 128$, RELU, BATCH NORM
MAXPOOL 2×2
FC 512, RELU, BATCH NORM
FC 7
SOFTMAX

Table 2: Architecture of our 8-layer network

At the beginning of our experiment, this 8-layer network gives us 51.0% accuracy on test set, while the training accuracy arises to 95% quickly. We tried various regularization techniques to reduce the over-fitting. The most obvious way is to increase the regularization factor. However we found it hard to find a good parameter. Sometimes closing the gap between training and testing accuracy would also result in worse test time performance. To achieve better regularization for deep network, more fine-grained regularization factors might be needed, rather than a global regularization factor.

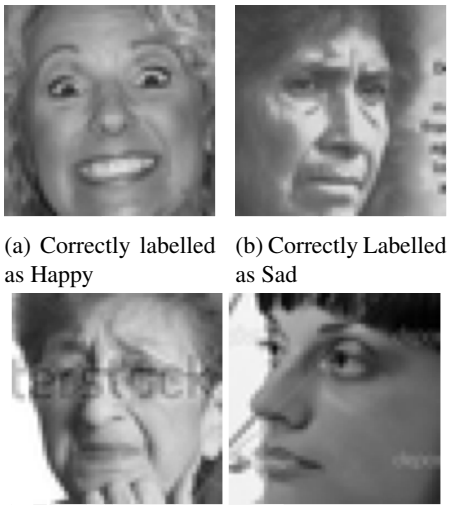
After adding dropout to our network, the overfitting issue is much improved. The gap between training and testing dataset is reduced to less than 15%. To further improve the performance, we tried a few data-augmentation techniques. For example, we randomly select some images to flip horizontally before feeding into the network. It helps to further reduce the gap to around 8%. We also tried adding random Gaussian noise and random cropping, but do not obtain much improvement.

Finally, we construct a deeper 11-layer network with slight modified architecture compared to previous 8-layer network. Here instead of inserting dropout layer right after convolution layer, we add it after the max-pooling layer. The final architecture is shown below. Our best model achieved 65.34% accuracy on test set and 74.09% on training set.

Figure 9 shows a few examples of both correctly classified examples and mis-classified examples.

input data(48x48 grey scale image)
Data Augmentation
CONV $3 \times 3 \times 64$, RELU, BATCH NORM
CONV $3 \times 3 \times 64$, RELU, BATCH NORM
MAXPOOL 2×2
CONV $3 \times 3 \times 128$, RELU, BATCH NORM
CONV $3 \times 3 \times 128$, RELU, BATCH NORM
MAXPOOL 2×2, DROPOUT 0.2
CONV $3 \times 3 \times 256$, RELU, BATCH NORM
CONV $3 \times 3 \times 256$, RELU, BATCH NORM
MAXPOOL 2×2, DROPOUT 0.25
CONV $3 \times 3 \times 512$, RELU, BATCH NORM
CONV $3 \times 3 \times 512$, RELU, BATCH NORM
MAXPOOL 2×2, DROPOUT 0.25
FC 1024, BATCH NORM, RELU, DROPOUT 0.45
FC 1024, BATCH NORM, RELU, DROPOUT 0.45
FC 7
SOFTMAX

Table 3: Architecture of our 11-layer network



(a) Correctly labelled as Happy (b) Correctly Labelled as Sad

(c) Misclassified Sad as Neutral (d) Misclassified Neutral as Fear

Figure 9: Examples of correctly and mistakenly labeled images

5. Conclusion

In this project, we used a convolutional neural network on Kaggle facial expression recognition challenge and were able to achieve an accuracy of 65.3% on the test set, which could rank top 5 among the 56 teams participating in the challenge (held in 2013). We showed that by incorporating various regularization techniques such as dropout and data-

augmentations, the over-fitting issue can be largely suppressed.

We showed that through a high-level qualitative understanding of the dataset characteristics such as its size, dimensionality and diversity, we were able to develop some intuition to explain the performance discrepancy between different architectures and training approaches. First, we found that during fine-tuning, freezing more layers would result in performance degrading significantly compared to fine-tuning weights across all layers. The intuition behind is that the early layers of pre-trained model are optimized for recognizing finer features of relatively large images in ImageNet, whereas the 48*48 Kaggle images have much lower resolution. Second, because the Kaggle dataset has relatively low variance across images, and thus requiring lower representational power for the network, the experiment shows that it takes a reasonable amount of time to train an AlexNet from scratch to reach similar accuracy as fine-tuning pre-trained model. Finally, we found that the performance of an 8-layer network that we constructed is closed to that of VGGNet, but much higher than a similar 8-layer AlexNet. And we conclude that preserving the size of feature maps at early layers is crucial for images with small size.

To further improve the performance, an unsupervised learning phase might be added before the supervised learning. This technique is particularly useful when the dataset is small. In fact, the winning team of Kaggle facial expression challenge in 2013 use a RBM (Restricted Boltzmann Machine) before the supervised learning phase and achieved an accuracy closed to 70% on the testing dataset.

References

[1] G. E. H. Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information and Processing Systems (NIPS)*, 2012.

[2] Y.-H. Byeon and K.-C. Kwak. Facial expression recognition using 3d convolutional neural network. *International Journal of Advanced Computer Science and Applications*, 5(12), 2014.

[3] C. C. Chibelushi and F. Bourel. Facial expression recognition: A brief tutorial overview. 2003.

[4] e. a. Samira Ebrahimi Kahou. Combining modality specific deep neural networks for emotion recognition in video. *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 543–550, 2013.

[5] Z. A. Simonyan, Karen. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[6] e. a. Yaniv Taigman. Deepface: Closing the gap to human-level performance in face verification. *Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014.

[7] Z. Zhang. Feature-based facial expression recognition: Sensitivity analysis and experiments with a multi-layer perceptron. *International Journal of Pattern Recognition and Artificial Interlligence*.