Multi-label Image Classification with Deep CNN

Zijun Zhao, Dailun Li, Kaiwen Xu

November, 2021

Abstract

In this project, we investigate the performance of convolutional neural networks on the task of classifying multi-label image data. We develop a 17-layer CNN based on Alex-Net, which achieves around 94% accuracy on the test dataset. After that, we further implement some creative strategies, including data augmentation, polling and usage of unlabelled data, and also sparsely connected Conv layers to boost that score to above 97%. Finally, we conclude and discuss where we can go beyond that score.

1 Introduction

Inspired by Alex Krizhevsky et al., whose groundbreaking research draws attention to deep convolutional neural networks [1], we want to utilize the powerful tool of deep CNN to solve real-world problems. The problem is to identify a combination of one handwritten digit and one handwritten English alphabet in a 56x56 image.

2 Datasets

This dataset contains 30,000 samples. Each sample is a 56x56 image. There are 30,000 labels accordingly. Each label is a size 36 binary vector. The first 10 binary values correspond to the digits 0-9, and the last 26 correspond to alphabet a-z. We use these data to train and validate our model. (Validation/Training = 0.1) We also have 30,000 unlabelled images, whose usage will be explored in the "Creative Works" section.

3 Experimentation

3.1 Model Selection

We first examine a naive CNN model with two conv2D layers and two fully connected layers on the dataset to gain a general understanding. The structure is listed in Figure 2. This simple covnet performs well on the training set to reach an 85% accuracy after training for 50 epochs. However, its validation accuracy stops improving at only 8%, revealing that this simple network suffers from overfitting. The model only maps very basic features to the labels. We then move our attention to more complex networks.

1. Alexnet has shown a significant improvement on the training and validation accuracy, with 82% validation accuracy after 100 epochs. After that, both the training and testing accuracies start to decrease. Considering the structure of the Alexnet in Figure 1, one can confirm that the depth of layers cannot make the model expressive enough as the it fails to converge to perfect accuracy.

2. The above experiments motivate us towards VGG16, a far more in-depth model at a glance. Its structure is listed in Figure 3 The performance of vgg16 successfully meets our expectations as the model converges at an astounding speed. After 10 epochs, the training accuracy reaches 99.9%. Nonetheless, it again suffers from overfitting as the validation accuracy stops increasing at around 83%. The detailed performance is listed in Table 1.

3. With the above knowledge, we implement a more complex Alexnet to handle both the overfitting and converge speed problem. The structure is as listed in Figure4.

3.2 Design Architecture

The 16*16*56 Conv layers are labelled as type 1 Conv layers, the 64*64*28 Conv layers are labelled as type 2 Conv layers and the 256*256*14 Conv layers are labelled type 3 Conv layers. It will be listed as tuples for further convenience, such as (1, 2, 2) representing 1 type 1, 2 type 2, and 2 type 3 layers.

We set the max-pooling layers to be at the end of type 1, type2, type3 layers and the dropout rate of the fully connected layers to be 0.5. Inspired by the structure of VGG16, the later Conv layers are deeper than the

previous ones. We also set the activation function to be leaky relu, for it contains the merits of a relu function but solves the zero relu problem. We initially set the gradient descent method to be sgd. However, it lacks the ability to adjust its learning rate. So instead, we choose to use the adam gradient descent with initial learning rate 5e-4, and (beta1,beta2)=(0,9, 0,99).

3.3 Tuning and Data Results

We first set the number of sparsely connected layers (later referred to as SC layers) to be 1 and 2, to test the effectiveness of modifying the number of type 1, type 2, type 3 layers. The results are listed in Table 2 and Table 3. As the results indicated, the structure with (3, 4, 4) layers outperforms the other models with a validation accuracy of 0.958. We then stay with the above number of type 1, type 2, type 3 layers and modify the number of SC layers. The effect of such an approach is recorded in Table 4, which shows that the 2-SC-layer model remains with the highest validation accuracy. However, the performance of the 2-SC model is inferior to the 1-SC-layer model with only less than 0.05% accuracy difference. According to Occam's Razor Principle, Figure 6, 8 with 1-SC-layer and (3, 4, 4) is chosen to be the final model. The effect of scaling, data augmentation, and relabel for this model is listed in Table 5. The accuracy versus epoch graph can be found in Figure 7. The final test accuracy of this model shown on Kaggle is 0.970.

4 Extra Creative Works

4.1 Data Augmentation

Although convolutional neural networks are inherently translation-invariant because of pooling layers, they are not invariant to rotational transforms. As a result, we apply a rotation of -30°, -15°, 15°and 30°to the training set to form a much larger dataset. We train our model on the new dataset. In this way, our model learns to identify labels with different rotations.

4.2 Other Preprocess and Polling

In addition to data augmentation, we also want to utilize other preprocessing methods such as "Scaling" and "Relabelling". "Scaling" resizes the images from 56x56 to 112x112, which we assume may help the model learn more details. "Relabelling" means to convert the size 36 mutli-labels to size 260 single-labels (260 classes from "0a" to "9z"). Now, we use different combinations of preprocessing methods to train 8 separate models. Details of the techniques used in each model is shown in Table 6. Finally we let them poll to give us the final result. This improves the robustness of result.

4.3 Use Unlabelled Data

Inspired by Kihyuk Sohn et al.[2], we decide to try semi-supervised learning methods on our unlabelled data. The idea is to feed these unlabelled images to our existing model and get predictions. Use these labels as pseudo-labels of these images. Then, we augment this unlabelled images dataset by rotating them with random degrees. Finally, we retrain our model with these rotated images and their corresponding pseudo-labels. In this way, the amount of input images is further expanded, which could reduce the variance while not changing the bias. Unfortunately, when we run it, we see the accuracy does not change (see figure 10).

4.4 Confusing labels and Future work

One of the difficulties we encounter is that the data augmentation method results in a misinterpretation of the letters "x" and "t" due to the fact that an "x" looks like a "t" when rotating about 30 degrees. Another common mistake is that the digit classifier and the alphabet classifier may end up predicting the same part of the graph. For instance, in Figure 9, the correct label for this picture is "o8", whereas it is predicted as "o0" by the model, which means the circle "o" has been predicted twice. For further improvement, additional build-up algorithm to exclude this case is required.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[2] Kihyuk Sohn et al. *FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence*. 2020. arXiv: 2001.07685 [cs.LG].

5 Appendix

Table 1: Different Models with their parameters and performances

Model	# of Conv layers	# of Fc layers	gradient descent	Activation	Accuracy(Training/Testing)
Naive Convnet	2	3	SGD	Relu	0.85/0.08
Alexnet	4	3	SGD	Relu	0.96/0.82
VGG16	13	3	SGD	Relu	0.99/0.83
DeepAlex	9	3	Adam	Leaky Relu	0.999/0.94

Table 2: Deep Alex and model hyperparameter tuning with 2 sc layers

# of sc layers	# of type1 layers	# of type2 layers	# of type3 layers	Accuracy (Training/Testing)
2	1	1	2	0.999/0.936
2	1	1	3	0.999/0.944
2	1	2	2	0.999/0.956
2	1	2	3	0.999/0.943
2	2	2	4	0.998/0.950
2	2	3	3	0.999/0.955
2	2	3	4	0.998/0.953
2	3	3	5	0.999/0.958*
2	3	4	4	0.999/0.958*
2	3	4	5	0.999/0.948

Table 3: Deep Alex and model hyperparameter tuning with 1 sc layer

# of sc layers	# of type1 layers	# of type2 layers	# of type3 layers	Accuracy (Training/Validating)
1	1	1	2	0.999/0.925
1	1	1	3	0.999/0.936
1	1	2	2	0.999/0.938
1	1	2	3	0.999/0.938
1	2	2	4	0.998/0.941
1	2	3	3	0.999/0.951
1	2	3	4	0.998/0.949
1	3	3	5	0.999/0.949
1	3	4	4	0.999/0.958*
1	3	4	5	0.999/0.951

Table 4: Deep Alex and model hyperparameter tuning with 2 sc layers

# of sc layers	# of type1 layers	# of type2 layers	# of type3 layers	Accuracy (Training/Validating)
1	3	4	4	0.999/0.958
2	3	4	4	0.999/0.958
3	3	4	4	0.999/0.952
4	3	4	4	0.999/0.956

Table 5: Deep Alex and model hyperparameter tuning with 1 sc layers and (3, 4, 4)

scaled	augmented	relabeled	converge epoch	acc
Yes	No	No	26	0.996/0.956
No	No	Yes	20	0.996/0.947
Yes	No	Yes	26	0.993/0.947
No	Yes	No	28	0.999/0.971
Yes	Yes	No	30	0.999/0.968
No	Yes	Yes	7	0.985/0.961
Yes	Yes	Yes	28	0.998/0.962

Table 6: The 8 models we trained for polling

index	techniques used
1	semi-supervise
2	semi-supervise, scaling
3	semi-supervise, relabelling
4	semi-supervise, scaling, relabelling
5	data augmentation
6	data augmentation, scaling
7	data augmentation, relabelling
8	data augmentation, scaling, relabelling

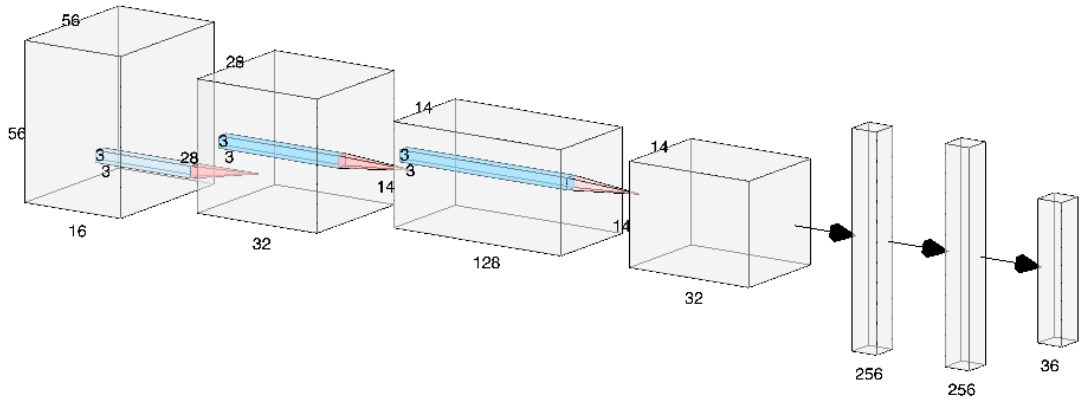

Figure 1: naive alexnet model

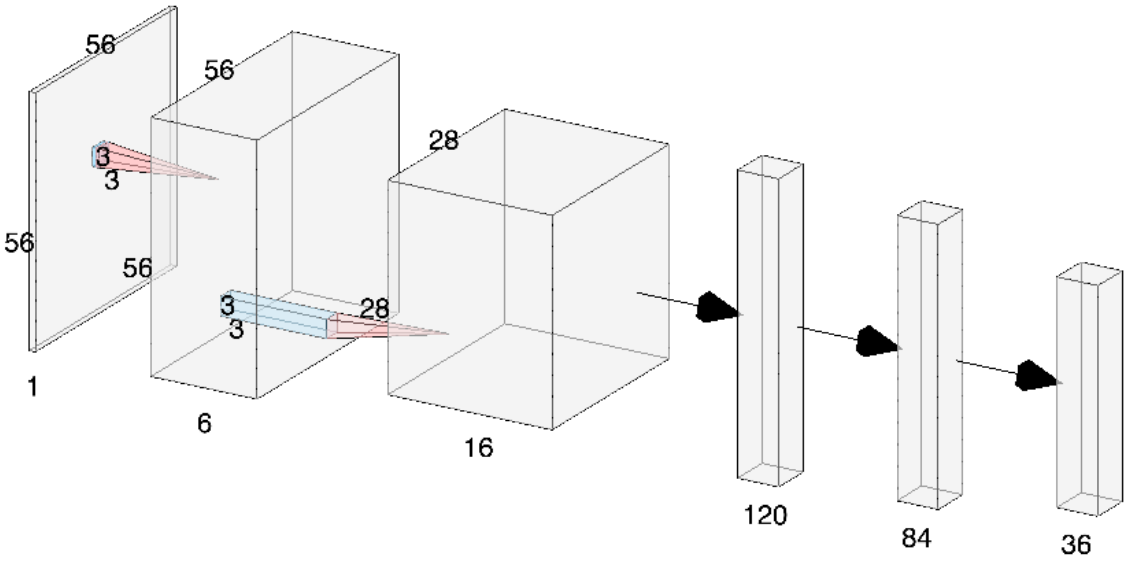

Figure 2: naive baseline model

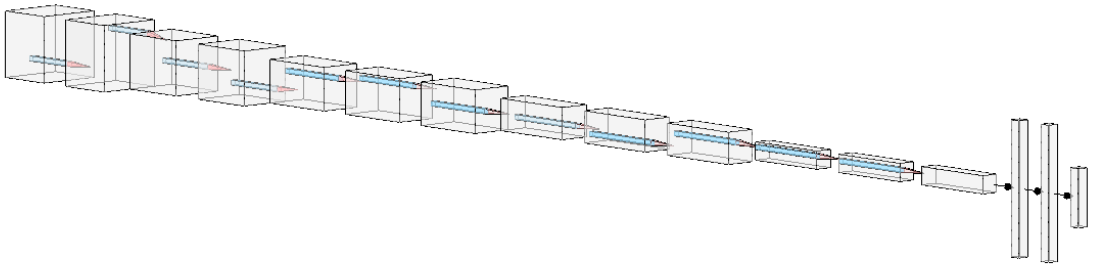

Figure 3: VGG model for comparison

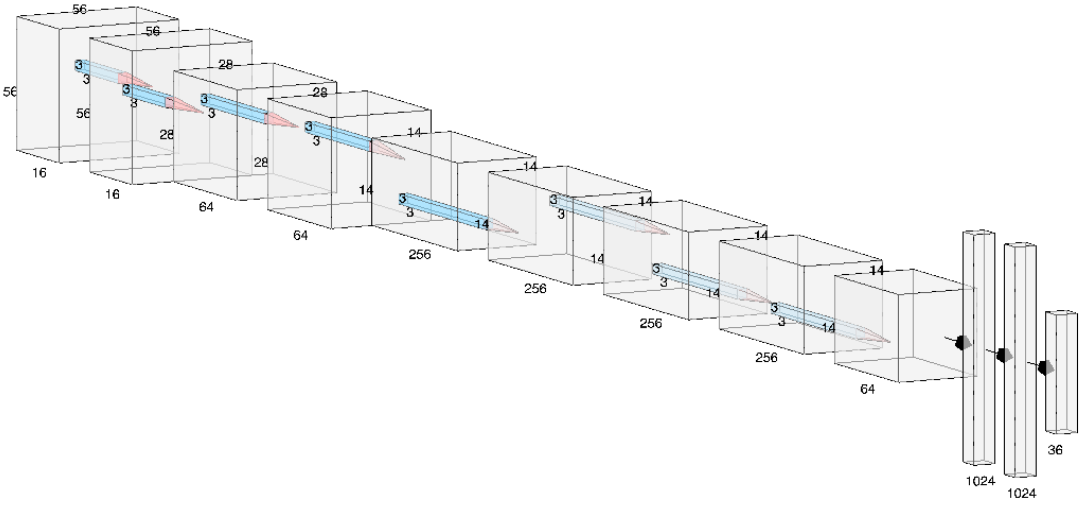

Figure 4: the developed deep alex model

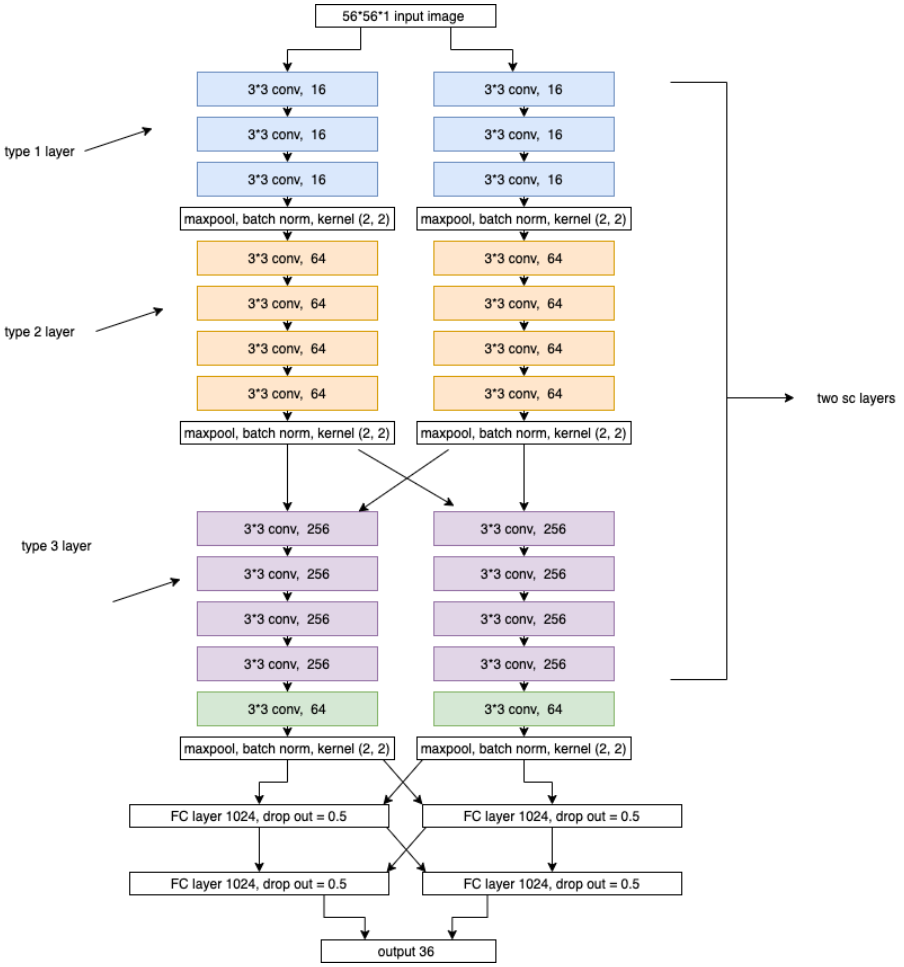

Figure 5: (3, 3, 4) with 2 sc layers deep alexnet model

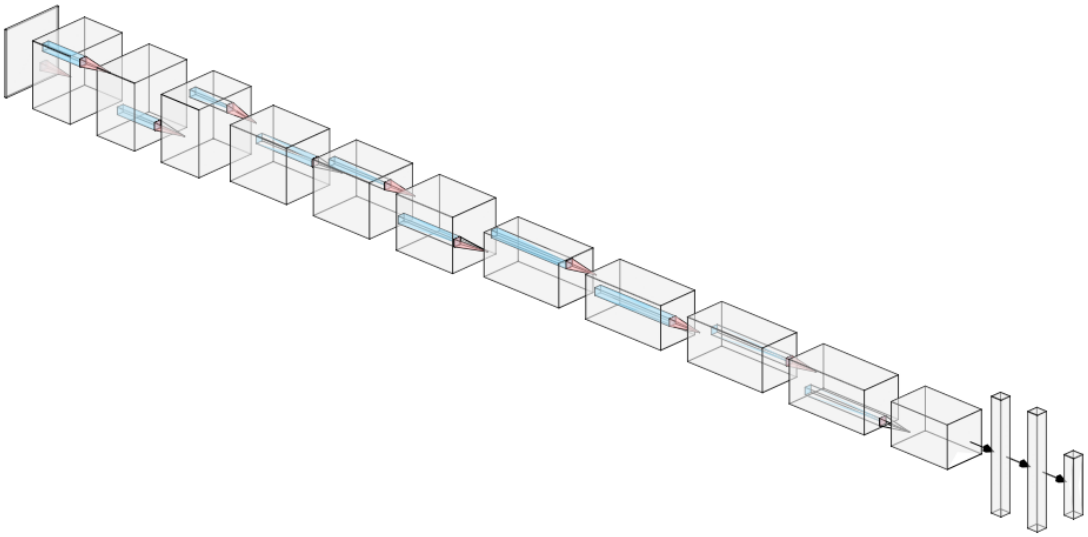

Figure 6: (3, 3, 4) with 1 sc layer deep alexnet model

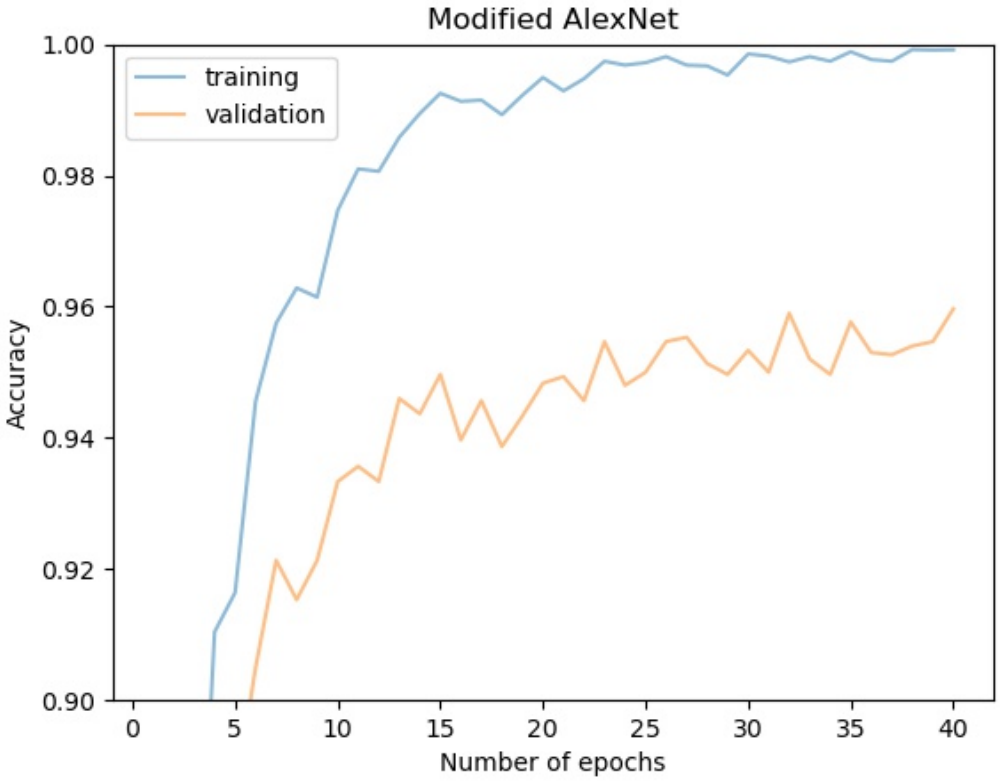

Figure 7: Final Model epoch versus accuracy graph

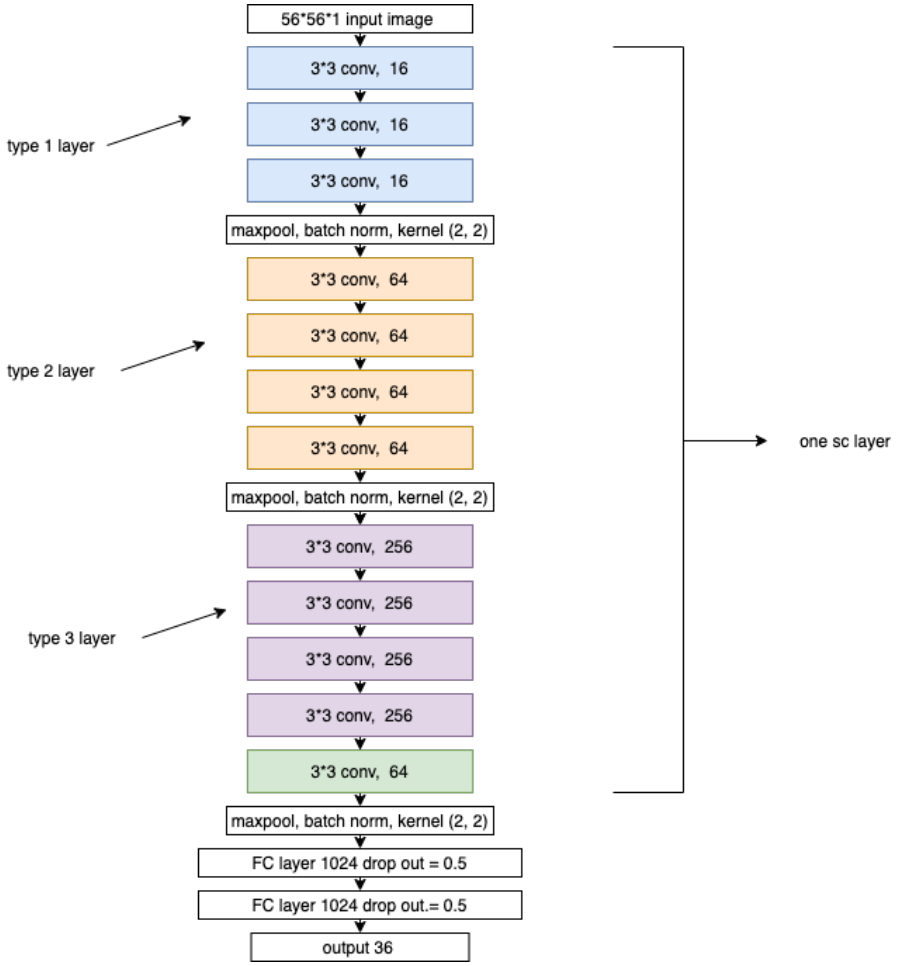

Figure 8: (3, 3, 4) with 1 sc layer deep alexnet model

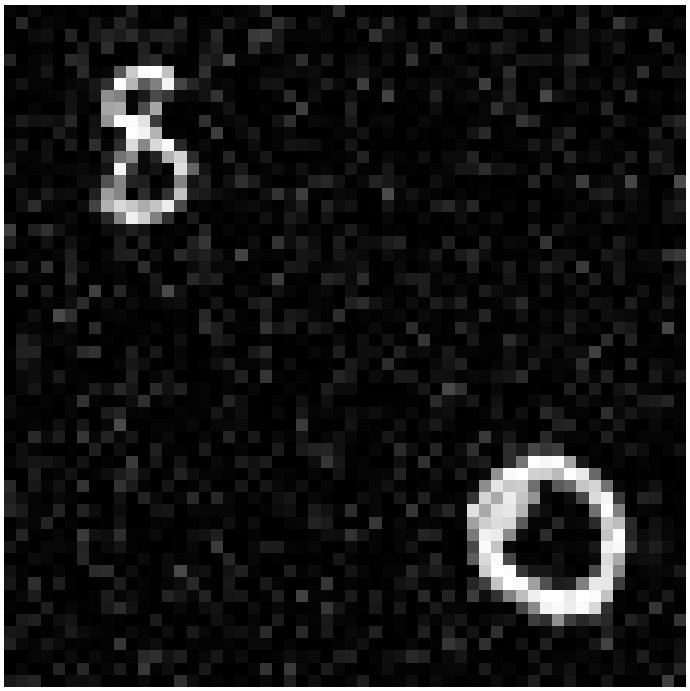

Figure 9: Correct label is "8o" but predicted as "0o"

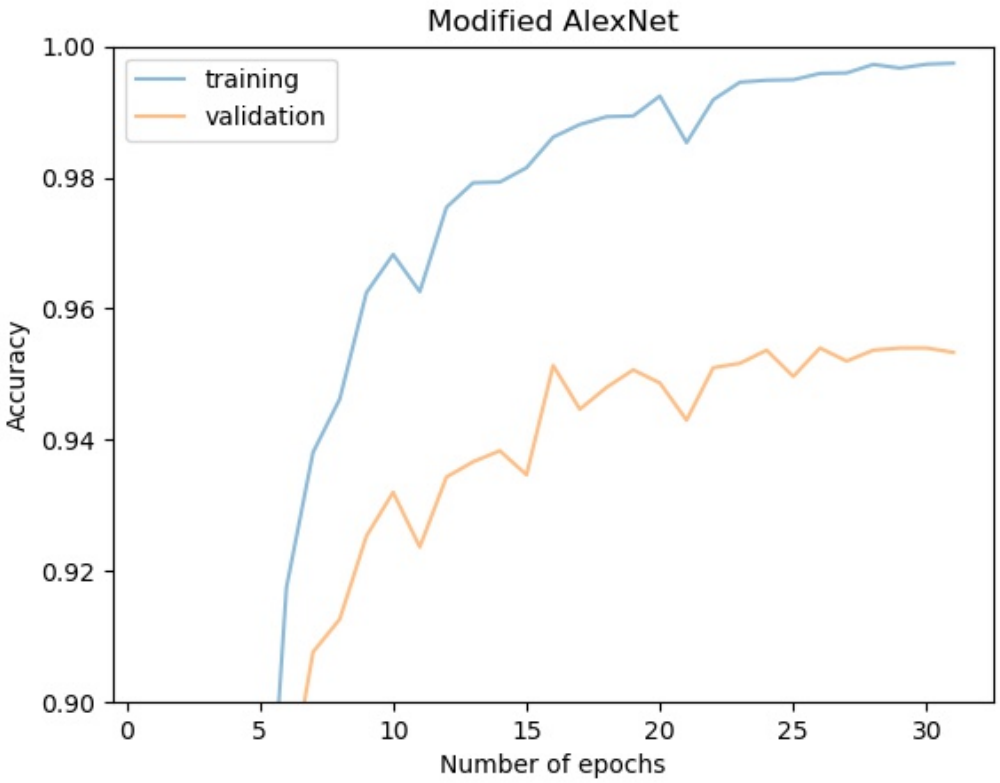

Figure 10: Semi-supervised learning accuracy curve