

---

# Class Project: Exploration on Multi-class Classification

---

**Yuqi Liu**

Department of Computer Engineering  
New York University  
Brooklyn, NY 11201  
y17344@nyu.edu

## Abstract

The state-of-the-art CNN, like models on ImageNet, which contains more than 14 million images and 20,000 categories, need precise input size( $256*256*3$ ), complex convolution layers(used for correct feature selecting) and huge fully-connected layers. While the growth of area(depth and width) does not get same linear performance boost of the network. In this project, I do experiments on the efficiency on the multi-class classifications and problem splitting.

## 1 Introduction

GitHub Repo Link: <https://github.com/y17344/Multiclass-classification>

### 1.1 Dataset and Network Construct

My project is based on CIFAR-100[3], ResNet[2] and my own-designed convolution neural networks. Cifar-100 has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs) as shown in Figure1.

### 1.2 Ideas on multi-class classification

My very first instinct was that with same construct of the network or smaller network, larger training data versus class numbers will lead to a better performance of the network. Therefore my first idea was to train a network to differ images in Cifar-100 on their superclass. Then, building small convolution networks for each superclass to identify the rest 5 different classes.

Then I decided to reduce the ResNet on its depth and width to see the performance of newly-built network and figure out the way of several networks work together to beat the standard one.

## 2 Original reference example

At first, I build a standard 34-layer ResNet. In each block of the ResNet, the output combines(the add of) outputs from one and two BatchNormed-Convolution-Layer, with blocks of [3, 4, 6, 3], and an average pool layer and a fully connected layer. The number of all layers is 34. The output channel ranges from 64 to 512(as a geometric sequence). The number of parameters is 21,328,292, the estimated total size is 104.38MB. The mini batch size is 250, optimizer is Adam and learning rate is  $1e-4$ .

| Superclass                     | Classes   |
|--------------------------------|---|
| aquatic mammals                | beaver, dolphin, otter, seal, whale                   |
| fish                           | aquarium fish, flatfish, ray, shark, trout            |
| flowers                        | orchids, poppies, roses, sunflowers, tulips           |
| food containers                | bottles, bowls, cans, cups, plates                    |
| fruit and vegetables           | apples, mushrooms, oranges, pears, sweet peppers      |
| household electrical devices   | clock, computer keyboard, lamp, telephone, television |
| household furniture            | bed, chair, couch, table, wardrobe                    |
| insects                        | bee, beetle, butterfly, caterpillar, cockroach        |
| large carnivores               | bear, leopard, lion, tiger, wolf                      |
| large man-made outdoor things  | bridge, castle, house, road, skyscraper               |
| large natural outdoor scenes   | cloud, forest, mountain, plain, sea                   |
| large omnivores and herbivores | camel, cattle, chimpanzee, elephant, kangaroo         |
| medium-sized mammals           | fox, porcupine, possum, raccoon, skunk                |
| non-insect invertebrates       | crab, lobster, snail, spider, worm                    |
| people                         | baby, boy, girl, man, woman                           |
| reptiles                       | crocodile, dinosaur, lizard, snake, turtle            |
| small mammals                  | hamster, mouse, rabbit, shrew, squirrel               |
| trees                          | maple, oak, palm, pine, willow                        |
| vehicles 1                     | bicycle, bus, motorcycle, pickup truck, train         |
| vehicles 2                     | lawn-mower, rocket, streetcar, tank, tractor          |

Figure 1: Cifar-100 classes

The total training time of this network is 43min 53s and the final accuracy of this network is **63.56%** after 10,000 iterations. The loss through training are shown in Figure2.

Though it seems that the test loss begins to increase after 4,000 iterations(data augmentation is applied), the test accuracy continuously increase in the total 10,000 iterations. It might mean we achieve better performance on normal inputs but get worse performance on ones we cannot detect correctly.

### 3 Split the task

My first idea is to split the task. Use 20 pre-labeled superclass to replace the original 100 labels. I hope the increase of proportion of training data versus class numbers will increase the accuracy of classification.

On the contrary of that, The training on superclass does not increase but decrease the accuracy. Though the the training speed is very fast: after 800 iterations, it reaches its minimum test loss as shown in the Figure3.

However, the accuracy is rather low than the original model. Its accuracy only reaches **50.93%**. This might be that

Then, I try to train 18-layer ResNet on the mini-train-set in one superclass. I try 20 different superclasses and their test set classification accuracy ranges from **50%** to **60%**. Their train accuracy grows very fast and simply reaches more than **99%**, however, the test accuracy cannot get up to **70%**.

I try to figure out whether it's an over-fitting problem and reduce ResNet to 10 layers and develop a simple batchnormed CNN, who's convolution layer ranges from 1 to 5. However, I try each of the network bu get the same problem—training set fitted while remains low test accuracy.

Therefore I check the input images. Through looking the image, I find that there might be different features of test set comparing to train set and the network cannot detect those features.

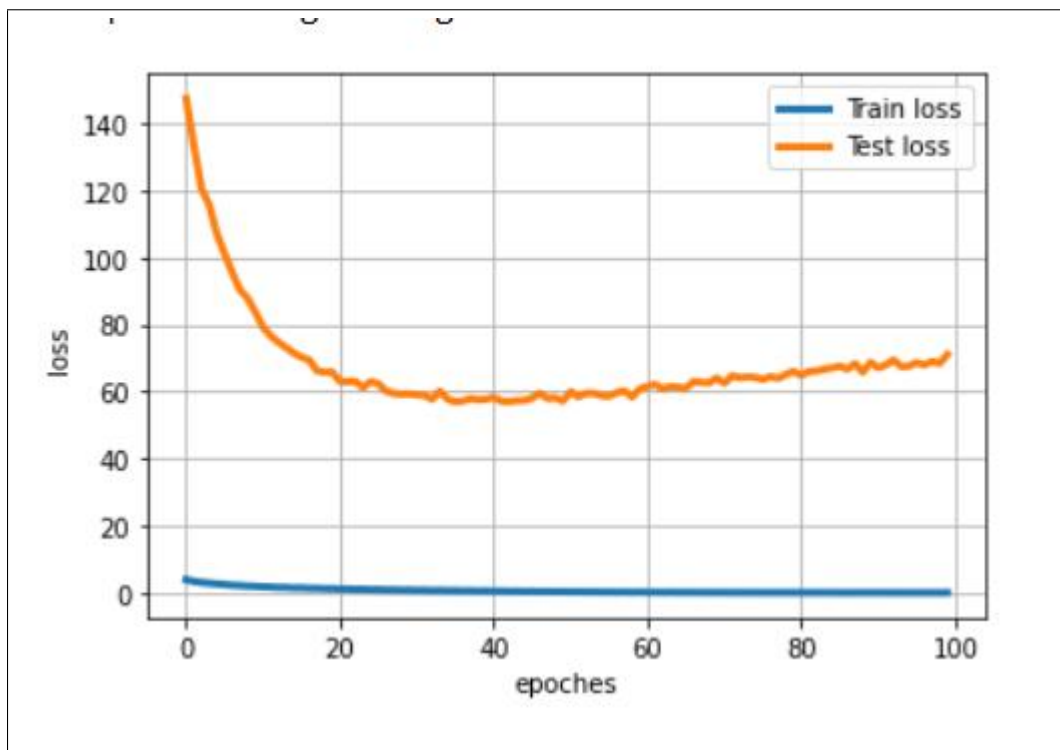


Figure 2: Loss of Standard ResNet34

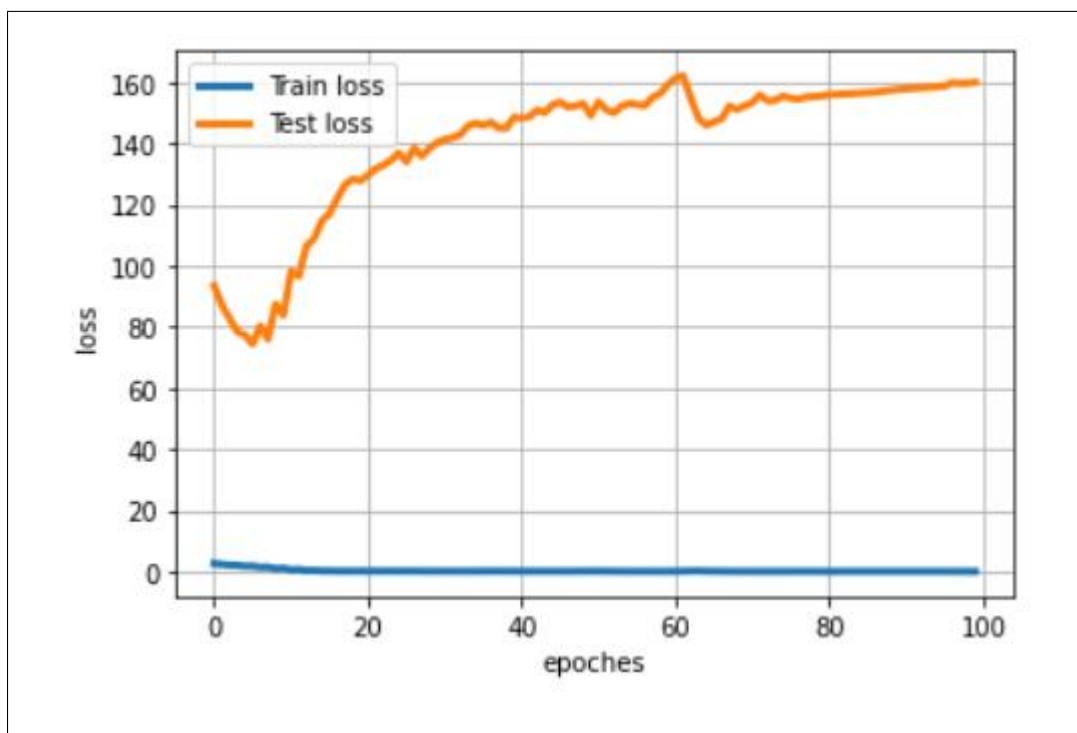


Figure 3: Loss of Standard ResNet34(20 super classes)

Due to this suggestion, I change the training process to learn from random test set(using SGD) for each 10 iterations and reach both **99%** accuracy on both set. However, it's very risky to use test set and this algorithm is not robust plus a low accuracy on overall accuracy.

What's more, I try to train the small nets on the classes they should differ and those samples not belong to this super class. With total dataset, it reach **95%** accuracy, which means it does not work accurately(all points are labeled as class 6, which means the data point does not belong to this super class).

## 4 Reduction on width

When Splitting the task fails. I look at another way: reduce the width of the network, which is, reduce the convolution channels of the network. I reduce the channel by 4 times with the same structure of the ResNet. Through this change, the number of parameters of single network reduces to 1,346,228 and the total size reduces to 10.90MB.

I train this network on Cifar-100 superclasses(20 classes in total), however, the old problem does not solved, which means this division of the data set is not efficient. Several networks are trained with a randomly dropout of 0.2.

Next, I train the network on the original data set. The result is that, network with this construction shows a very low training rate and unsatisfactory accuracy(an average of **40%**) on the total set. The combination of several small networks works poorly as well.

## 5 Reduction on depth

When we cannot go narrower, shallower might be a choice. I construct ten 10-layer ResNets with their largest convolution channels as 512. Each small net contains 4,949,412 parameters, which is approximately 1/5 of the original network(21,328,292 parameters).

With this kind of network, the training accuracy is finally around **58%**. Training time for each small network is about 8 minutes, which is about 1/5 of the original network. The loss through training is shown in Figure 4.

To enhance the performance of network set, I use static expert to combine the output of the networks. The network with least accuracy is **58.10%**, with the number of networks increase, the test accuracy shows a growth trend as shown in Figure 5.

The final accuracy of 10 networks set is **67.16%**. With similar training time, which is a combination of 5 networks, reaches an accuracy of **65.35%** which beats the original network which has **63.56%**.

When Considering training time, three small networks could reach **64.31%** accuracy and reduce the total train time and parameters by about **40%**.

## 6 Conclusion

There are still lots of work to do with this topic. In this project, the small networks are trained separately. As from Ranked-1st network on ImageNet[4], it used meta learning to train the network.

Or, the combination of small networks could use reinforcement learning[1] to build Q-map to select networks instead of static experts.

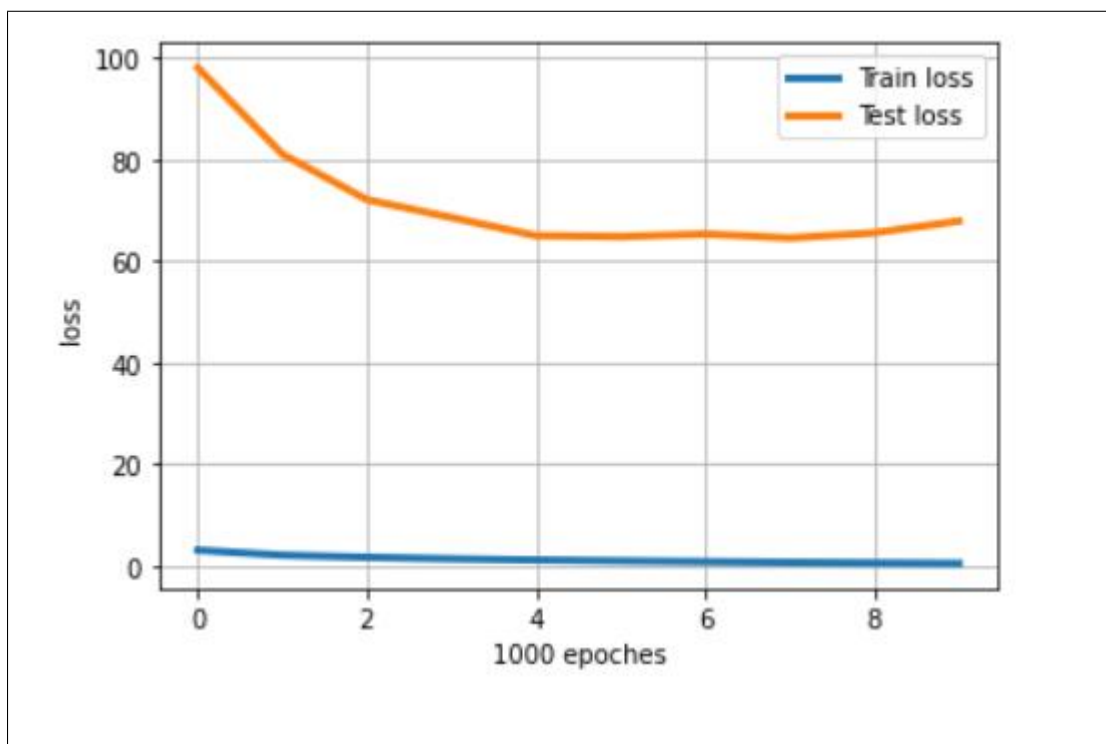


Figure 4: Loss of ResNet10

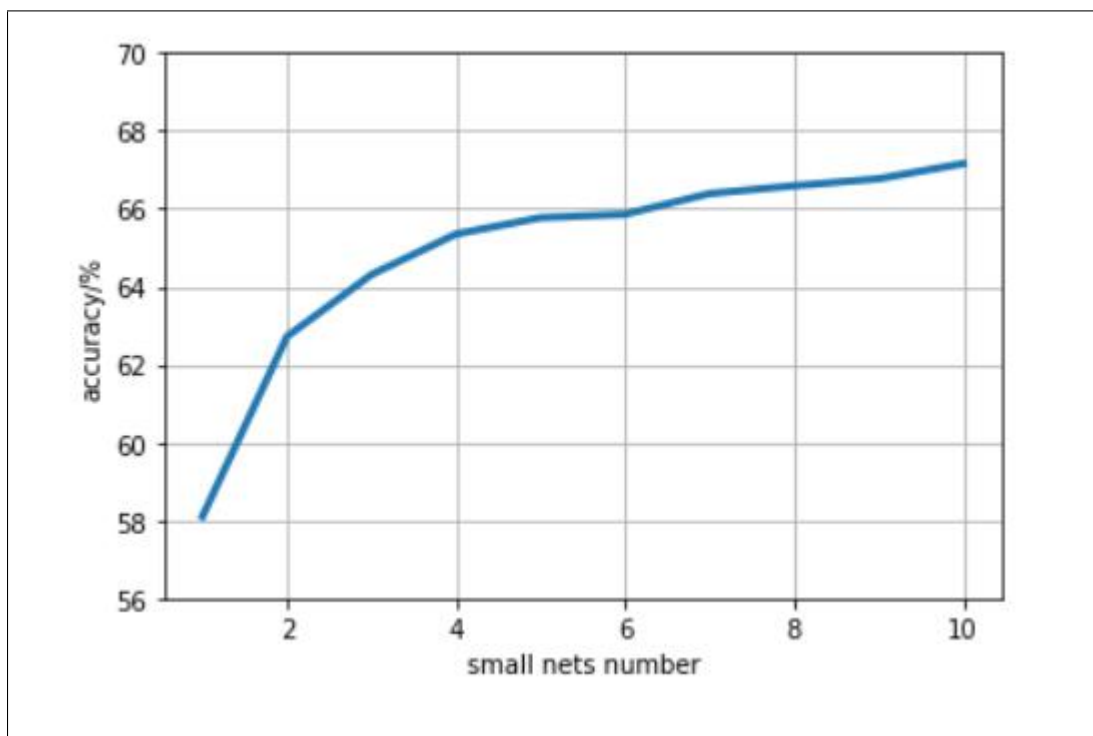


Figure 5: Loss of ResNet10

## References

- [1] Peter Dayan and Yael Niv. Reinforcement learning: the good, the bad and the ugly. *Current opinion in neurobiology*, 18(2):185–196, 2008.
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [3] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
- [4] Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V Le. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020.