清华大学自动化系

专业实践技术报告

题目： How architectures influence the network's generalization

班　　级： 自 45

姓　　名： 柳荫

学　　号： 2014011858

实践地点： Ithaca, NY, the United States

实践单位： Cornell University

实践部门： Department of Computer Science

指导教师： Gao Huang, John E. Hopcroft, Kilian Q. Weinberger

2017 年　　9 月　　10 日

Contents

Background and meaning.

In 2006, Hinton proposed a Deep Confidence Network (DBN), a deep network model. Using a greedy unsupervised training method to solve the problem and achieve good results. DBN (Deep Belief Networks) training methods to reduce the difficulty of learning parameters of hidden layers. And the training time of the algorithm is almost linear with regard to the size and depth of the network.

Different from the traditional shallow network learning, the depth of study emphasis more on the depth of the model structure, weighing the importance of feature learning, through the layer of feature transformation. Representing of sample element space feature to a new feature space in order that classification or prediction is easier. Compared with the method of constructing the characteristics of artificial rules, it is possible to use the large data to study the characteristics, and to describe the rich and internal information of the data.

Compared with the shallow model, the deep model has great potential. In the case of massive data, it is easy to achieve a higher correct rate by increasing the model. The deep model can be used for unsupervised feature extraction, direct processing of unlabeled data, and learning of structured features, so deep learning is also called Unsupervised Feature Learning. With the emergence of GPU, FPGA and other devices for high performance computing, the emergence of neural network hardware and the emergence of distributed deep learning system, the deep learning training time has been significantly reduced, so that people can simply increase the number of devices to use to enhance the speed of learning. The emergence of deep network model has made countless problems in the world solved, deep learning

has become the most popular research direction in the field of artificial intelligence.

Until 2012 Alex Krizhevsky won the ImageNet game with deep neural network, they set off a deep learning boom. As for the reason why the deep neural network had kept silent until 2012, the external reason is the rise of heterogeneous computing (mutual support from cpu and gpu), easy access to data and explosive growth of the data volumn, the internal reason is the gradually solving of the problem of the depth of deep network (the training problem of deep neural network stability). The depth and width of the network is the key to network capacity. Obviously, the large number of parameters of AlexNet is redundant in the fitting ability of functions. Afterwards, people focus on the network stability training problems (gradient vanishing and gradient explosion), on the depth of the deep neural network, the width and the kernel size. Focusing on the above problem, people have created a variety of CNN varients. There appeared ZFNet, VGGNet and GoogLeNet Inception in the following years.

In 2015，Kaiming He pointed out the two problems exist in the deep learning of newral network in his paper of ResNet: 1. Gradient vanishing ／ gradient explosion；2. When network goes deeper and deeper, the phenomenon of network degradation would come out. Network degradation means making networks deeper only by piling one convolutional layer on one another will make the performance even worse than shallow networks. Also, this isn’t due to the reason of over fitting. The problem of degradation is one of the optimization，deep model couldn’t be optimized easily. Also he thinks: the deeper model should be able to perform at least as well as the shallower model; A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

Residual Learning is an important idea in ResNet. Why do we do Residual Learning? As we all know, no matter in traditional machine learning or in deep learning, we all want the net to learn a non-linear mapping H(x). When the data pass through H， the samples of the same class get close to each other， and those of different classes get as far away as they can (minimize intra-classes variance and maximize inter-classes variance). In deep learning， we suppose that piling up of multi-nonlinear layers is able to be infinitely close to the complex function H(x) that need to be learnt. This is also the same as infinitely being close the the residual function: F(x) = H(x) – x. If H(x) is too complicated， then it will be quite hard for the deep network to directly learn the one-step mapping to H. Thus it is better to adopt the two-step learning: first learn the residual function F(x), then learn H(x) = F(x) + x via some simple mapping. With the two-step learning， learning residual function will alleviate the problem of deep model degradation. Moreover, identity mapping can make the shallow layers get access to the gradient information more easily, minimizing the bad influence of gradient vanishing.

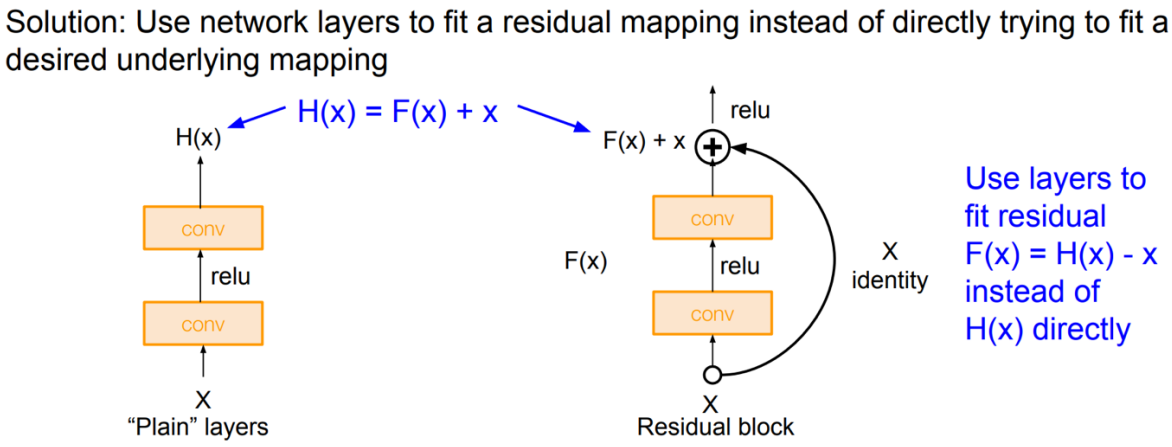


Figure 1

As for the problem of gradient vanishing and the problem of Network degradation, ResNet proposed a two-connection solution of identity mapping. The use of shortcut connection makes ResNet not have to use the auxiliary function like GoogLeNet to help the shallow network receive gradient information. The idea of residual learning has solved the problem of hard optimization of deep model， finally making the network be able to be be deep till thousands of layers. Also the famous shortcut connection further triggered the appearance of DenseNet. After the analysis of these networks, the network has been pursuing the depth of development, the depth is the final decision of whether the network effect is good or bad, because if the network is deep, then there will be an increase in the nonlinear level, resulting in that the network's ability to fit the complex function can be increased.

The graph below is copied from one presentation of Kaiming He which show the revolution of the deepening of newral networks. The y-axis is top5 accuracy.

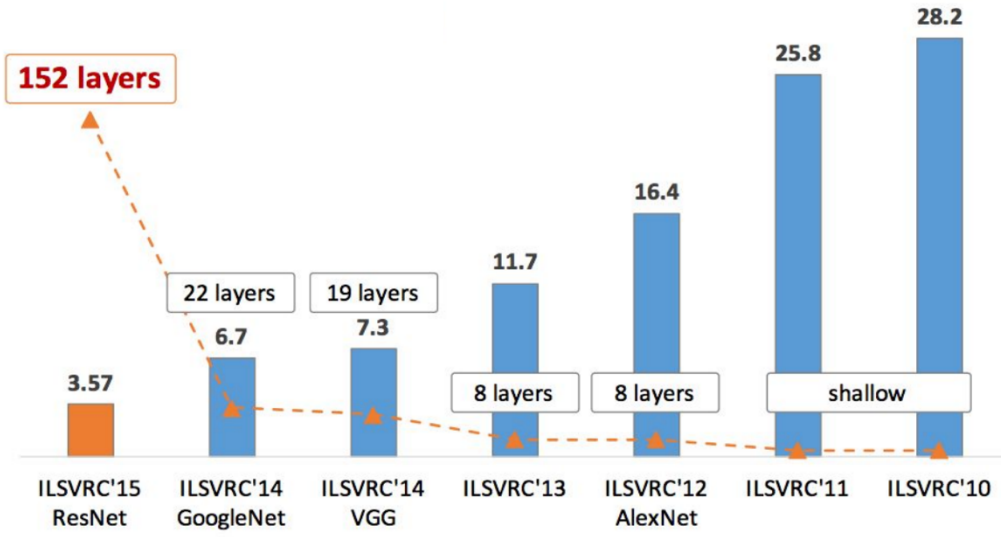


Figure 2

In 2016，Gao Huang, a post doc in Cornell university and Zhuang Liu, an undergraduate student in Tsinghua proposed the DenseNet model together，which is one more milestone in the architecture of deep neural networks. ResNet, HighwayNet and FractalNets have all succeeded by passing the deep information directly to the shallow layers via shortcut connection. Densenet further maximize the benefit of shortcut connections to the extreme. In DenseNet (more accurately in one dense block) every two layers has been linked，making every layer can use the information from all its previous layers. In doing this, DenseNet is able to efficiently mitigate the problem of gradient vanishing or degradation，making the input features of each layer various and diverse, making the calculation more efficient. The usage of shortcut connection played a role of deep supervised learning. Below is the DenseNet's connection：

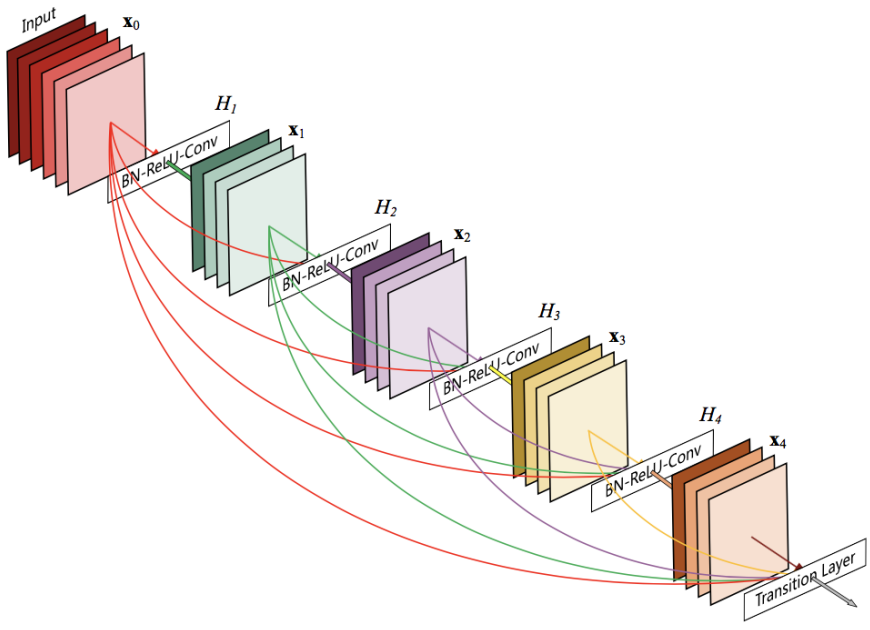


Figure 3

DenseNet mainly consists of convolutional layers， Dense Block, transition layer, and the classifier after the global average pooling。

Concatenation in Dense Block: the output of each layer will concatenate with its own input and then being passed forward to the next layer together. This makes the input characteristics of the next layer diversified and effectively improves the computation and helps the network to integrate shallow layer features to learn discriminative feature. Meanwhile， the neurons in the same Dense block are interconnected to achieve the effect of feature reused. This is why DenseNet does not need to be very wide and can achieve very good results (Note: depth and width are both key factors of the network, the network width will only achieve a certain effect after the network has already had some good performance). What's more，the reason why it doesn't use the simple add function to the features in previous different layers like being adopted in ResNet，is that the simple adding is one kind of simple behavior of feature fusion，which will badly cause the loss or disorder of information.

Compression in transition layer: DenseNet has one feature: the number of parameters in use is far less than that of ResNet. Apart from decreasing the width of network due to feature reused，it has also utilized the 1 * 1 convolution layer to compress the information in its transition layers，which has been used before in GoogLeNet inception v3. As a result, it makes the model more compact. In addition, another advantage of the reduction in the amount of parameters is to reduce the complexity of the model to prevent over-fitting phenomenon.

Deeply supervision: shortcut connections form the multi-channel model，Making the flow of information from input to output unimpeded. Gradient information can also be fed backward directly from the loss function to the the various nodes, which is a bit like that a brain directly control the behavior of different body parts.

Novel exploration

There are enough demonstrations of the performance of DenseNet in various famous dataset in the paper. This time, my main goal is to：explore how this kind of skip connections be used as a novel way of network regularization，and further how it influences the generalization ability of the whole network.

-- Weight decay experiments

Firstly, I notice that the reasons DenseNet can achieve better performance than other networks may be that they can make fully use of the feature maps produced by the shallow layers and therefore it seems that the early features are more important. This, however, doesn't mean I can obtain the similar performance by simply decreasing the network depth. What I will do, is to manage to add some constrain to the latter layers' weights and by doing so I have a preference on the earlier layers. Luckily there is one thing called 'weight decay' which can be modified to reach my goal. In the standard implementation in github, they always use the globally invariant weight decay like 1e-4 which doesn't discriminate different layers. So I manually implement my idea by doing the following thing: rewrite the regularization function and use it properly. I have tried many different kinds of 'gradual weight decay'.

```
def add_regularization(model, reg_param):
    for i in range(model.nblayer):
        weight = model.block1.layer[i].bn1.weight
        reg = reg_param
        #for j in range(0, len(weight.grad.data) - 3, 1):
        for j in range(len(weight.grad.data) - 2, -1, -1):
            # reg *= decay
            reg = reg * (j + 1) / len(weight.grad.data)
            weight.grad.data[j] += 2.0 * reg * weight.data[j]
```

Figure 4

For example, 1. the linear weight decay (as for every dense connection with a source layer and a target layer: ‘linear’ wrt target layer when fixing the source layer, ‘linear’ wrt source layer when fixing the target layer, or ‘linear’ wrt the distance between two different layers(larger the distance, larger the decay)). 2. the several non-linear form.

The picture above is an example of linear decay adding. I have tried L2 regularizer as well as L1 regularizer.

However, the result is not promising. The improvement is not significant. The training result curve is almost the same from its original ones.

This is the first big experiment I have done. Although it may not be successful in some sense. At least, I have been familiar with different terminologies in Deep Network. Furthermore, I have been able to write a whole deep network using PyTorch and know what exactly each line of codes means. Additionally, many spots that need to be careful of such as the learning rate varying dynamically, the epochs, the training dataset size, and the data augmentation have all been scrutinized and I have used quite a few combinations of them. All in all, the practice is good, resulting in that I have got a good comprehension of how to establish a good working deep networks with these various tricks.

All experiments in the first experiment are implemented on Cifar10. Since the results aren’t good, I haven’t keep the graph, and I can retrain it if needed later.

■

But I also noticed in their paper Densely Connected Convolutional Networks，there is a heat map (the average absolute weight serves as a surrogate for the dependency of a convolutional layer on its preceding layers.) Red dots in position (l, s) means layer l make strong use of features maps produces s-layers before.

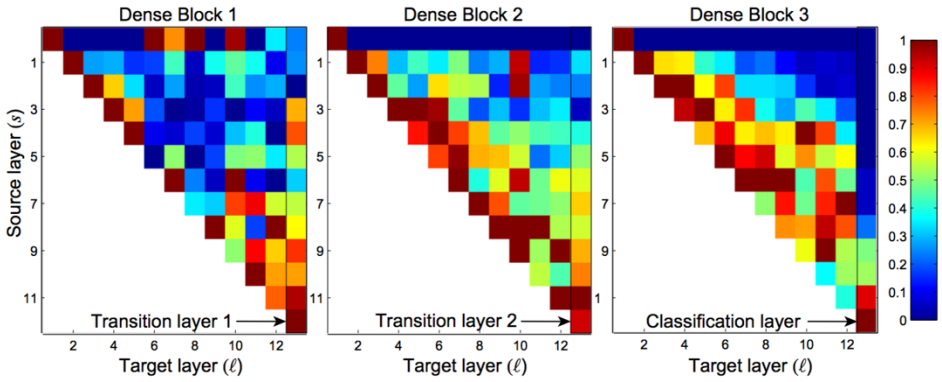


Figure 5

So, in each dense block, it looks like that the current layer use the features produced from the previous close layers more, e.g. the 8th layer use the features from 5th, 6th and 7th more than 4th, 3rd, 2nd etc. So I doubt my past thought that the shallower features weigh more importantly. Therefore, I do the totally opposite with previous – add more weight decay to the shallower weights. However, the results seem not as I hope either.

-- 1-dimension curve fitting

Next, I have done the 1-dimension experiment – the curve fitting. My goal is to use the simple networks which contains only linear connected layer and the nonlinear layers to fit the Sinc Function: sin(x) / x in range of (-15, 15). I implement 3 kinds of networks – just use their main characteristics – the plain network with layer after layer (a smooth linear model), the residual network in which I have add the skip connection every two layers and the dense network in which there is only one dense block that is between every two layers has one direct connection.

■ Dense, plain simple

In order to analyze the results more in detail, I have not only plot the final learned curve but also extract the output from each layer. For example, when I use 7-layer dense net and 7-layer plain net individually without any other data process. I try to control the number of parameters almost the identical by adjusting the width of two kinds of nets (that is the same meaning of growth-rate in the paper). When growth-rate is 2 for dense net and 4 for plain net, their parameters are 98 and 113 respectively. While the plain has more parameters, its results is bad, just as I think.

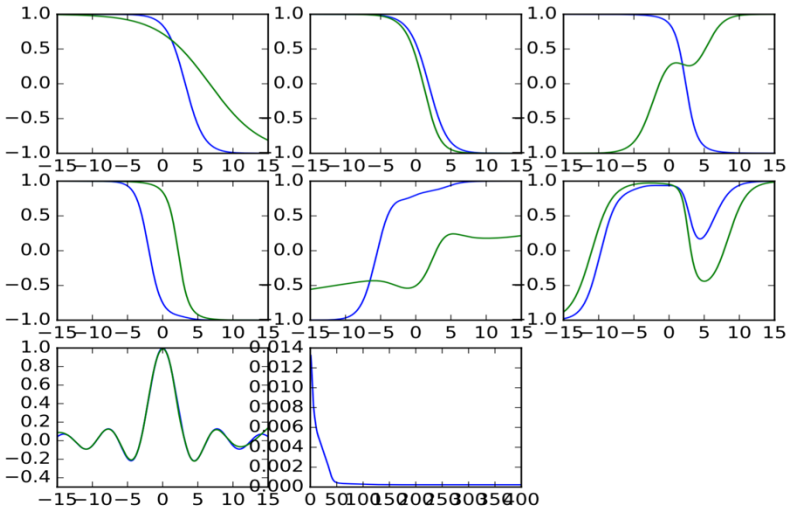


Figure 6

The above picture is the learning process of 7-layer dense net with a sample of 30 points. Each subplot is the output of each layer (as the growth-rate is 2 there are 2 curves in each plot). Also, while there is only 30 training data, it can learn pretty good. In the 7th plot, the blue one is tha standard sinc curve and the green one is its learnt one where we can see the two are almost identical. The last plot is the training loss change with the epochs.

On the contrary, as for the ordinary plain net, the result with 30 training data is as below:

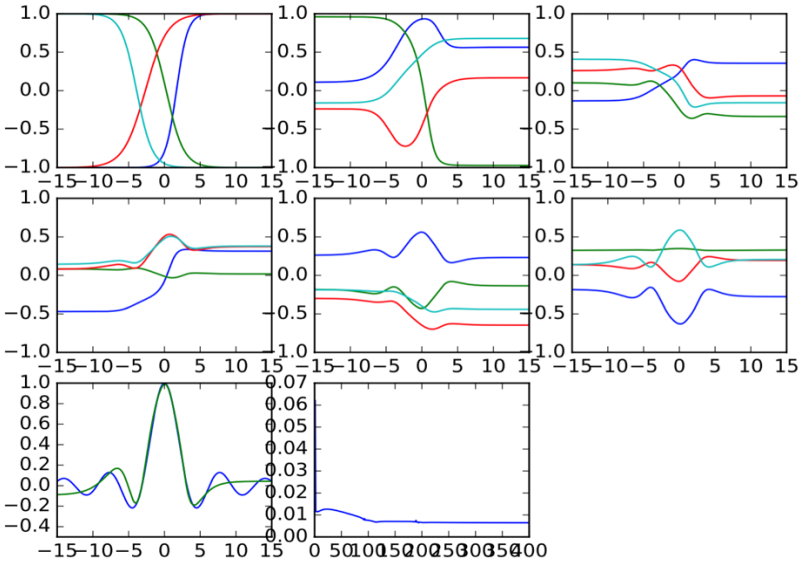


Figure 7

The output 4 curves from each layer is also as simple but the final results are bad. It couldn't learn the trivial wave. The thing that is also bad is in its training loss which drops quickly at the beginning but cannot be smaller later.

■ More data

Let us take more training data to feed the plain net to see whether this architecture can do a good job:

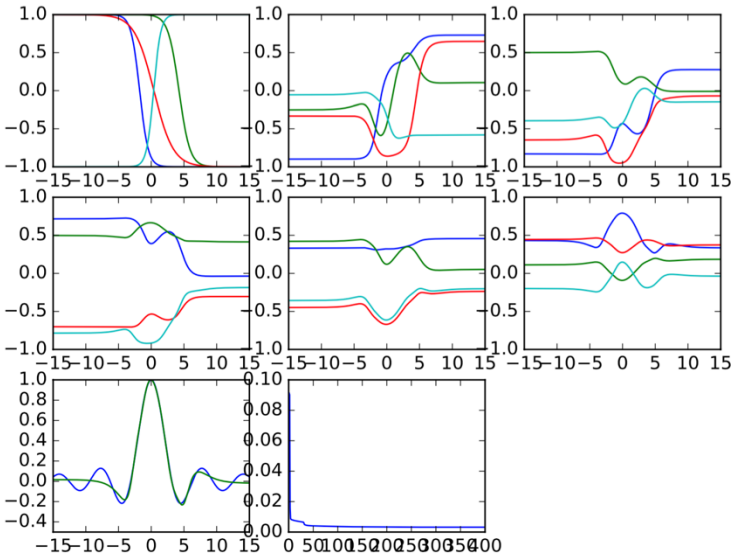


Figure 8

Above is 200 training data points. A bit better, but still couldn't fit the waves.
Next,

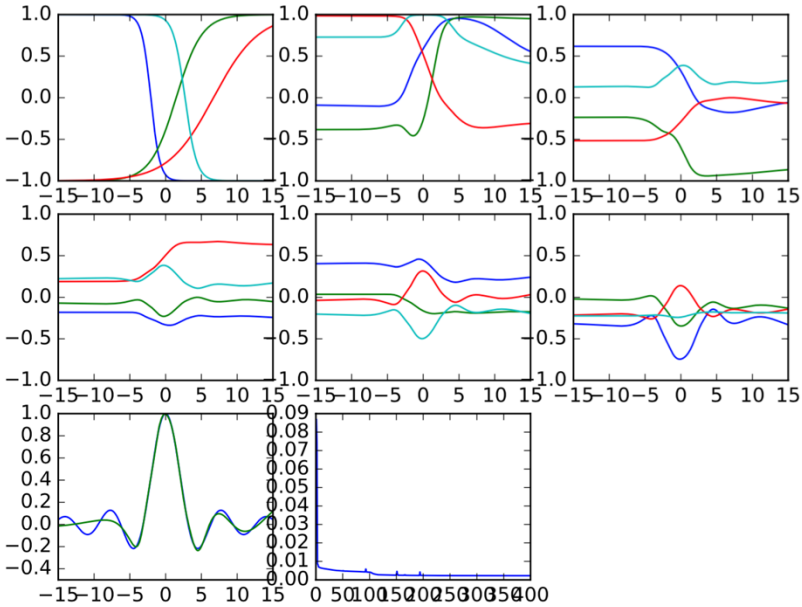
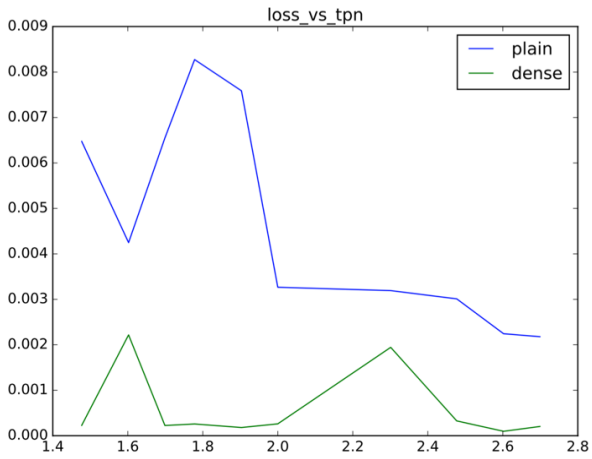


Figure 9

400 data points, fit the right waves! Good job. But it will be hard for this plain net to compete with the dense one.

■ Loss with depth

The learning ability is definitely stronger in dense net than plain one. We can prove it by plotting the loss function, the x-axis is the training data number. Below, the right one is the result of 3 growth-rate for dense and 6 for plain.



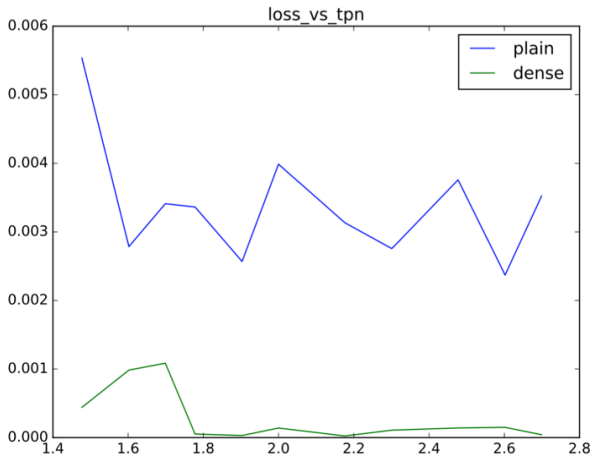


Figure 10 & Figure 11

■ Statistical results

Then, why don't I do some more interesting and convincing things? So I've done the same experiments several times and plot a statistics result.

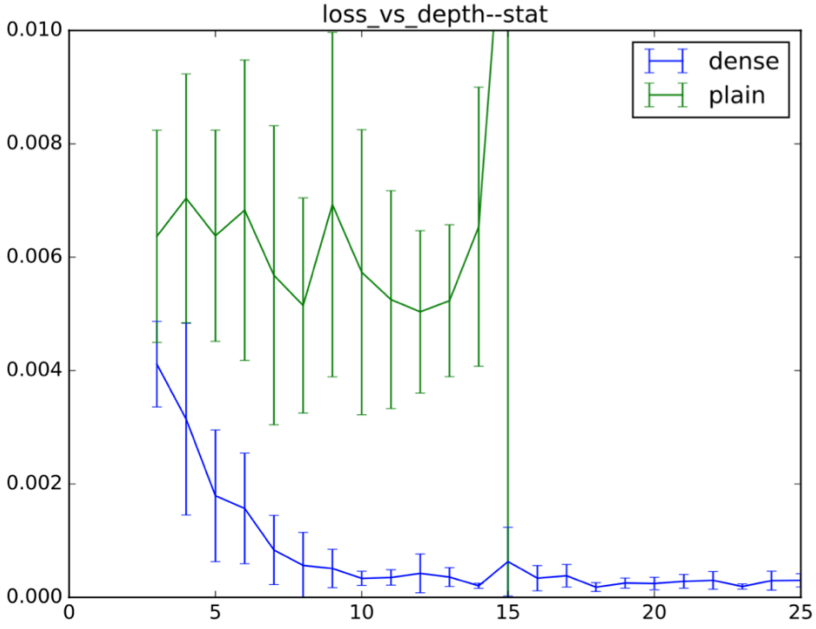


Figure 12

Above, x is the depth and y is the final loss. The middle point is the mean loss in 10 experiments, the line segments represent the deviation in these same experiments.

In this experiments I have make sure that in each depth the parameter number of two kinds of nets are similar through adjusting the growth-rate. We can see the dense net is better when deeper. And plain one will learn nothing is it is too deep. Absolutely it is due to the gradient vanishing as we will see later.

■ Noisy training data

Then I add noise to the training data that is to change a little of the value y to each input x of the sinc function. So it is more interesting to see the networks' generalization. I have chosen the growth-rate carefully for every single network setting. As can be seen below.

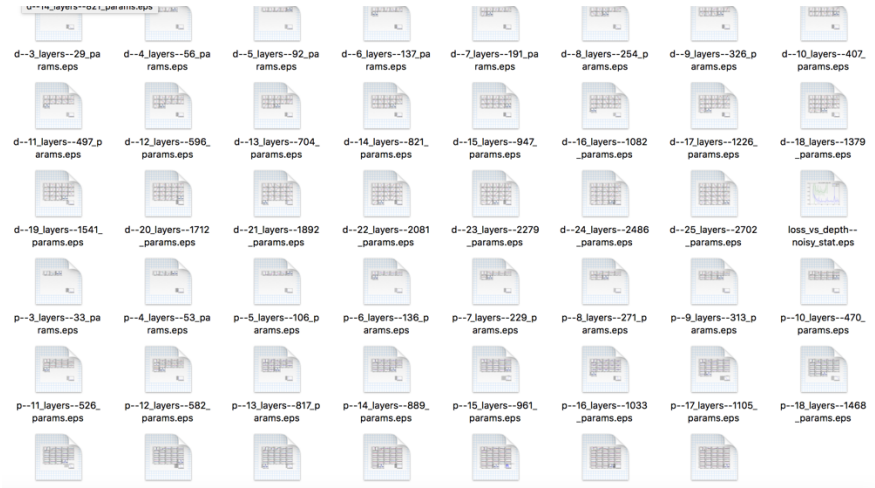


Figure 13

The results are similar as the circumstance with no noise:

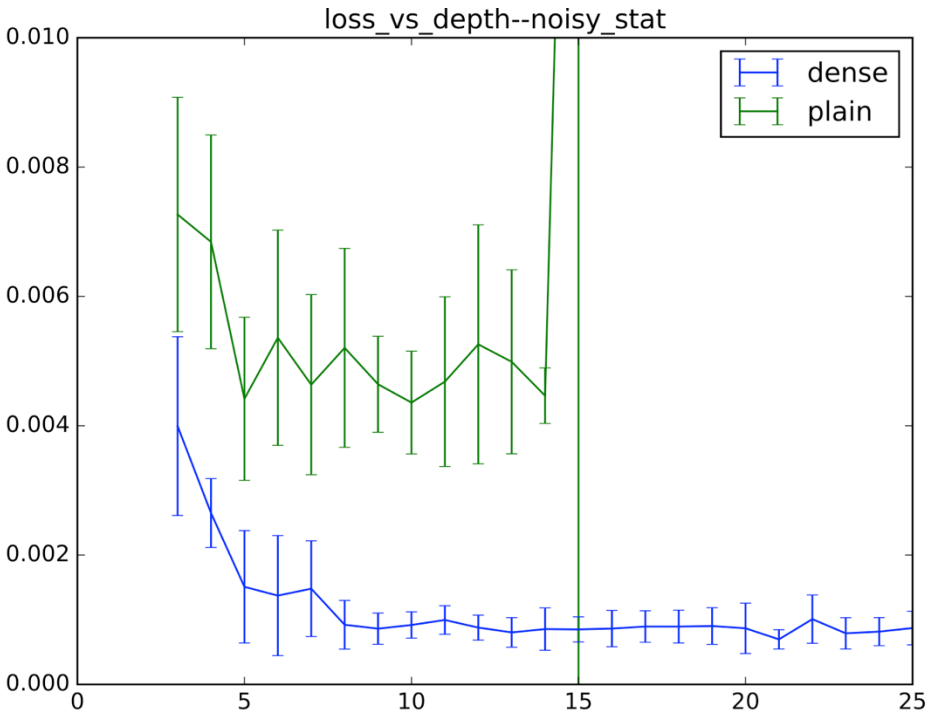


Figure 14

■ Together with ResNet

And why don't I add the ResNet to compare together? At least residual net is of best competitiveness before the dense net. So let the three net compare with each other together. Still, I choose the growth-rate carefully. See the statistic result:

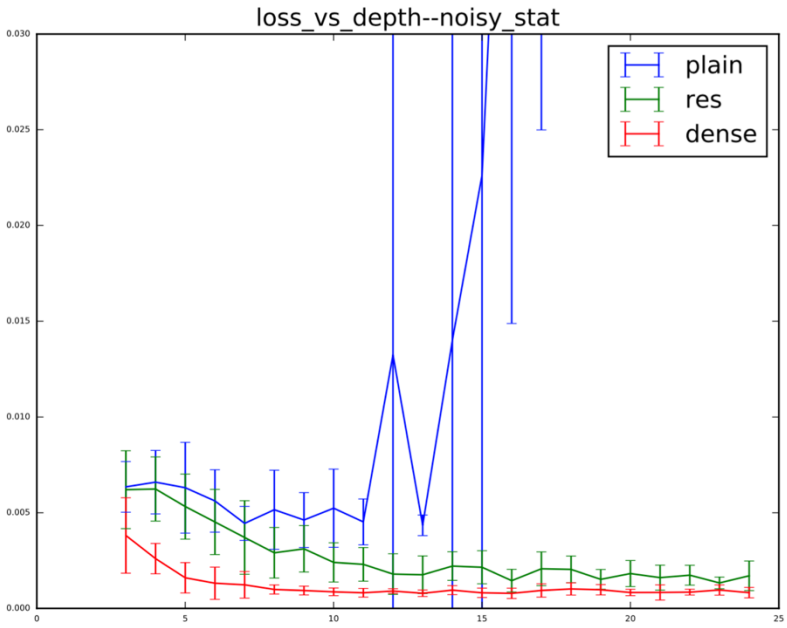


Figure 15

No matter in the mean loss or the deviation, the dense net is smaller than the other two. And It can generalize even much better than residual net relatively when the nets are shallow.

Let's have a look at like 20 layers of each net's learning detail:

First, plain net (the worst one): vanishing gradient is severe and finally it learns just a line of y = const which makes no sense at all:

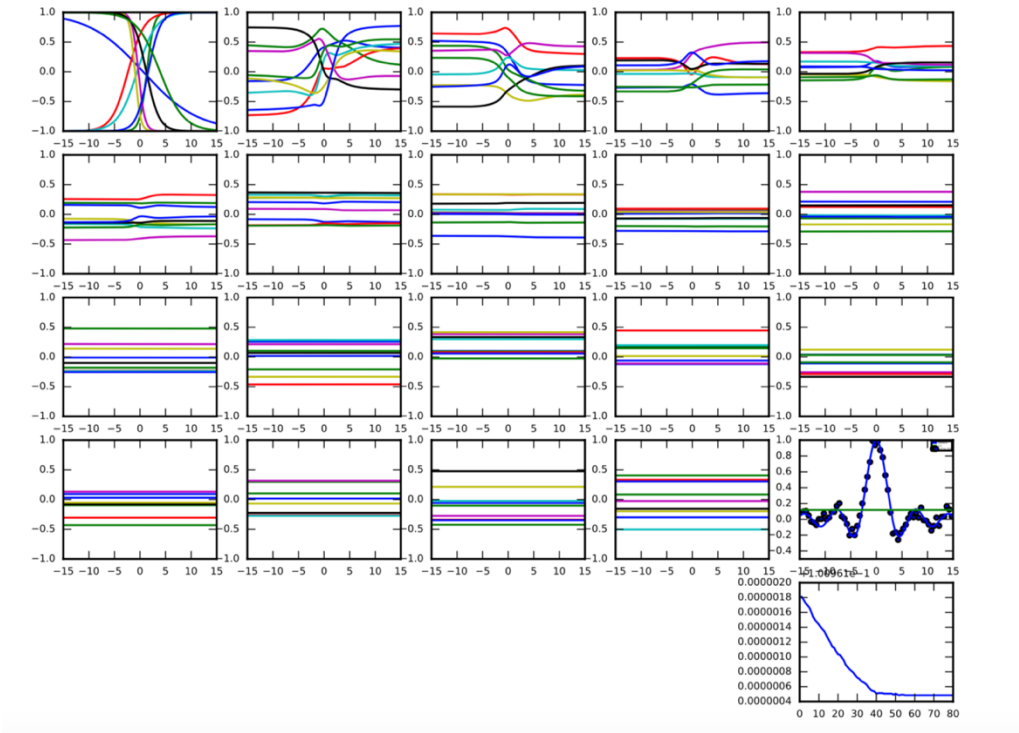


Figure 16

Then, res net: much better than plain net:

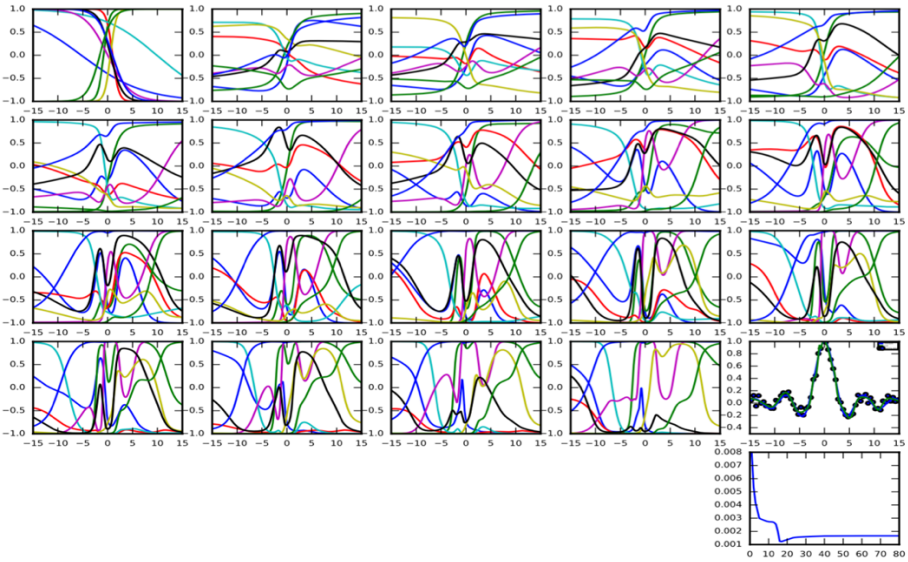


Figure 17

It can fit the curve well in the last plot but one. However, the middle output seems complicated.

Last, dense net: only 3 growth-rate each layer:

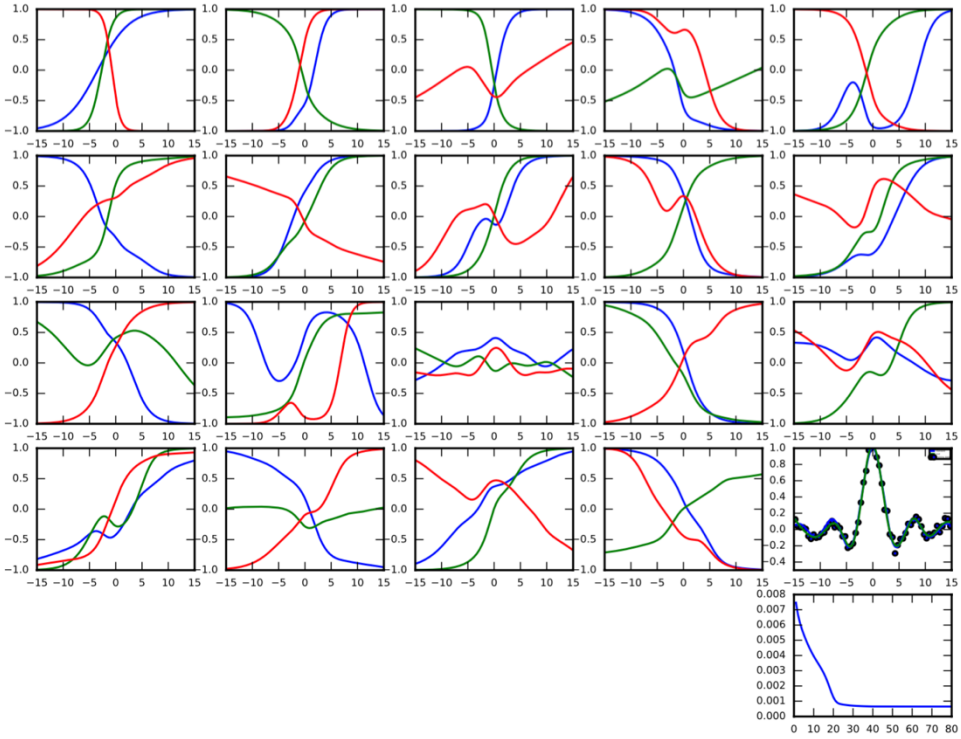


Figure 18

■ Results

Output from each layer is very simple and it can do the regression best. I think we can rethink about this interesting architecture and its tremendous learning power.

-- 2-dimension decision boundary (brief)

Actually, we have done a similar optimization experiments of the 2-dimension decision boundary learning. Still, it compares the dense net and simple plain net to learn the decision boundary. As there are only 2 classes of the points in a 2-dimension plane, this experiment is sometimes not that obvious to see the superiority of the dense net. Just show you a picture of 2 kinds of net's performance as below:

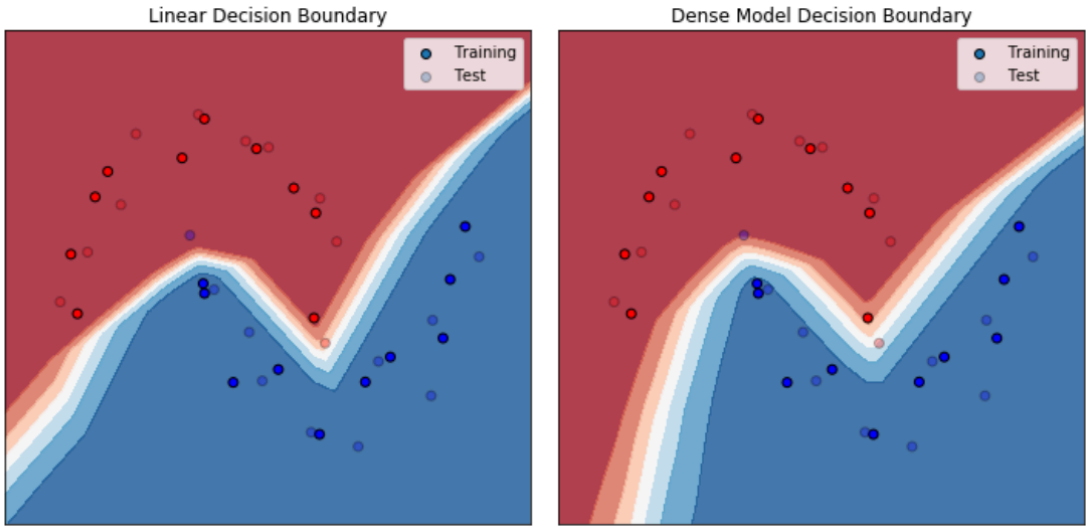


Figure 19

We will further improve this experiment and I am confident to see the distinction between two networks.

-- Test performance in bad conditions

■ -- Varying training data quantity

Finally, I have done the important test of varying the number of training data to see how the 3 kinds of neural networks generalize in lack of enough training data:

I have draw the absolute validation accuracy and relative validation accuracy respectively. The x-axis is training data size in Cifar-100:

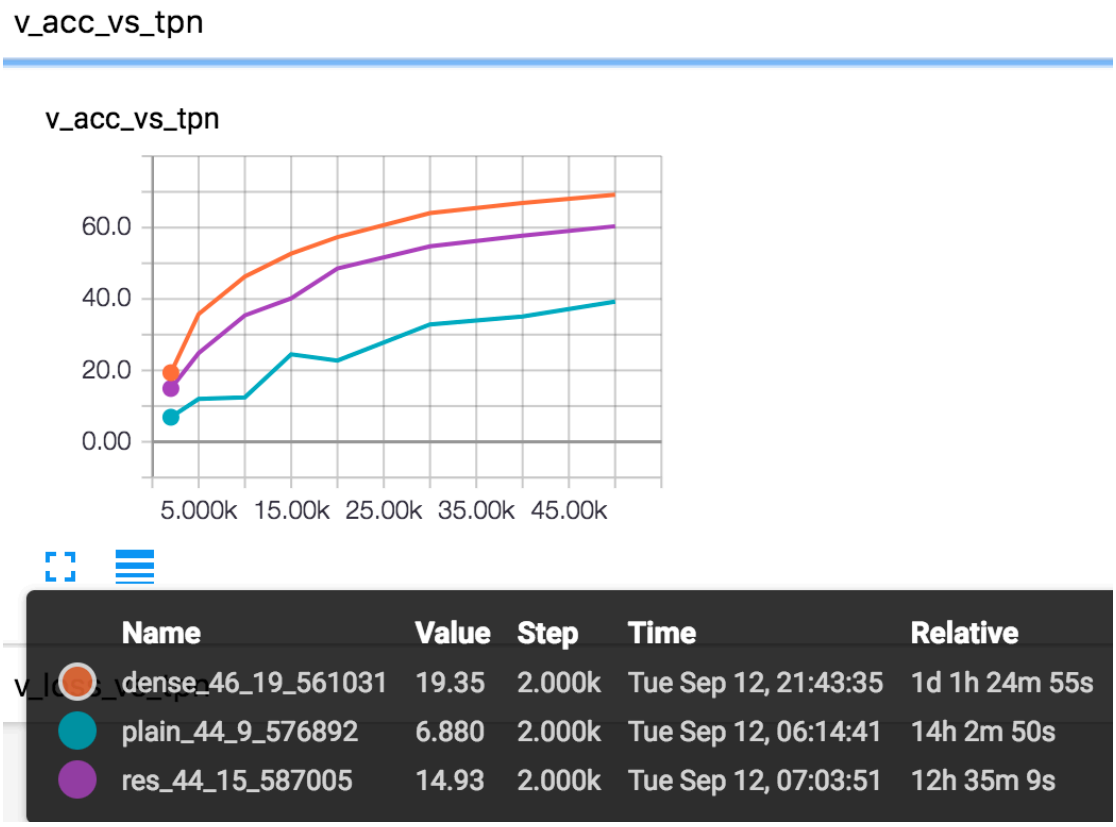


Figure 20

Of course dense net's performance is always better than the other two, no matter how small the data size is. Specially, like before, I have selected the appropriate depth and parameters for 3 networks to compare with each other. 44 or 46 layers is good since it has a not small learning capacity and you may ask why don't I choose the standard DenseNet or ResNet with like over a hundred layers. The answer is if there are more than like 60 layers, the plain net will suffer from some gradient problems as mentioned before. Also, as I will use the nearly

close layers of 3 kinds of nets to do the comparison, I can only compromise by using a like 44 or 46-layer net.

Then, from figure 20, you may not see much valuable or new discoveries because the superiority of dense net is known to all. Therefore, if you see the figure 21, you will be convinced of the advantage of dense net.

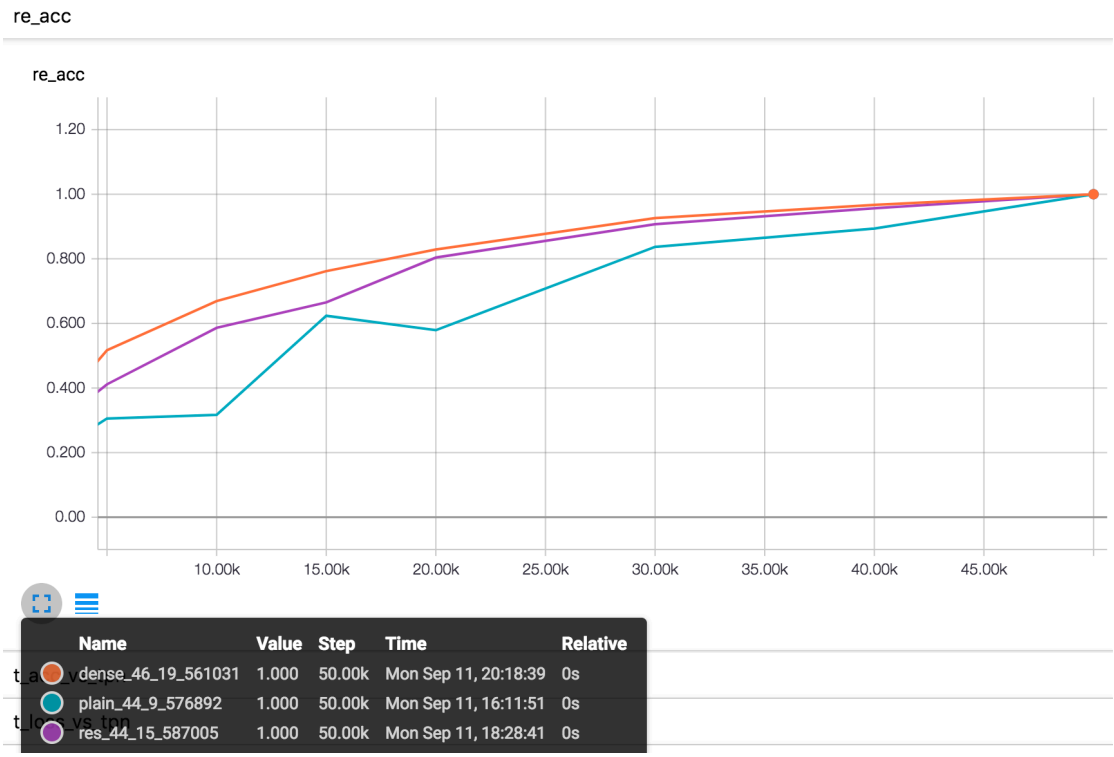


Figure 21

Yes, the figure above is the relative validation accuracy, that means: each kind of net compare only to its own best performance, which, of course, is in the condition of full 50k training data. So the rightest points of 3 nets are the same: (50k, 100%). And with the decrease of data size, the performance of dense net drop more slowly than the others. I can say that, this architecture can truly benefit the network's stability and its generalization. As a consequence, the dense connection may be an alternative to dropout as a method of regularization.

I have also drawn the generalization gap (just the validation error – training error):

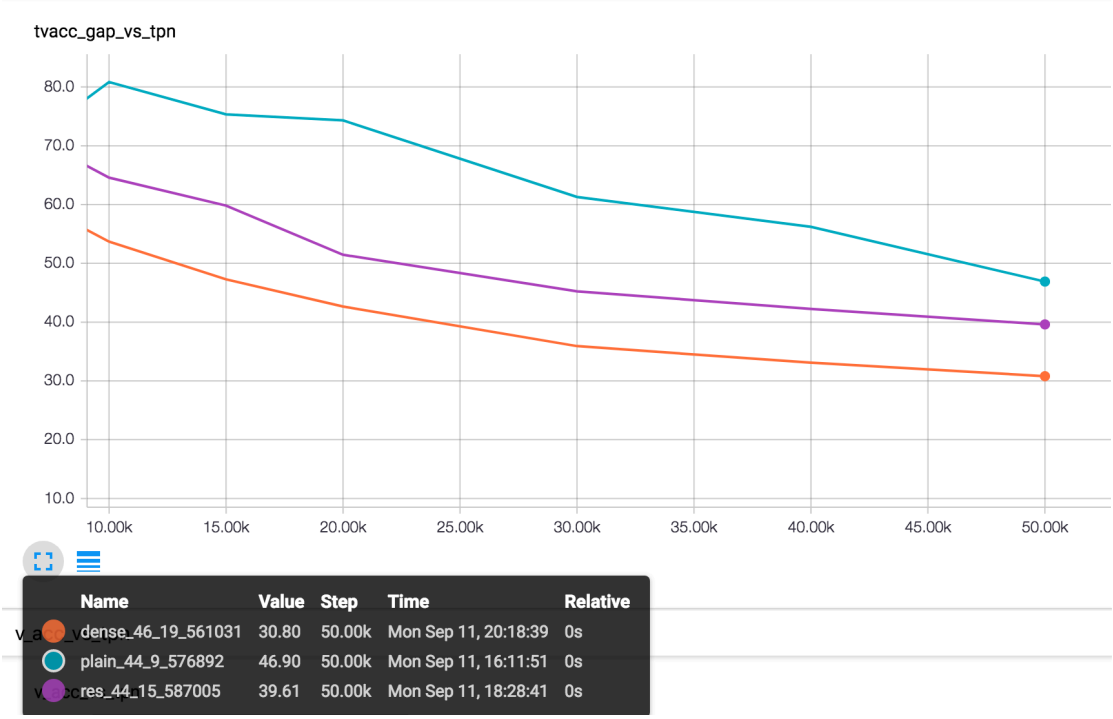


Figure 22

Also, we have seen the same results in other dataset like MNIST and Cifar-10. So the results are promising.

■ Varying 'noise'

Like decreasing the training data size, I have tried to add some noise to them, one kind of noise is to directly set some pixels in some channels as 0, that is to change the pixels of the image. The result is as the noise grows bigger, the decrease in dense net performance will be smaller than others. And the conclusion I will draw is consequently almost the same as above.

But there is another kind of 'noise' – shuffle the labels. This noise is very strong, e.g. you tell the program that the image of number '7' is a '5'. Then it is very hard to keep a good accuracy. The results (I have only do the absolute ones):

Figure 23

Dense net is always the best no matter how bad the outside conditions are (lack of training data, large noise). From the above graph, I can see that if I use the result data to redraw a relative one, dense net is still better than the two others when noise goes big.

Summary and prospect

Through this several weeks' major practice, I have felt and learnt a lot of things. From the proficiency in PyTorch framework to think of my own ideas and then to implement them, I have been polished up in my computer science skills, specifically, in deep learning analysis ability.

The DenseNet is one of the greatest deep net architecture, which, however, doesn't mean that it is perfect. I am trying to do the analysis of its hidden tricks, and trying to find out the value of 'skip connection'.

My personal work may seem trivial when being compared to Dr Gao's DenseNet, but at least I will consider my work as a guidance or a novel direction to lead others to discover more, such as my supervisor Prof Hopcroft, who will focus on the theoretical field of machine learning in the following years. I hope I have done some contribution to this field.

References:

1. G. Huang, Z. Liu, K. Weinberger, and L. Maaten. Densely connected convolutional networks. In *CVPR*, 2017.
2. K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.
3. D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Julien. A closer look at memorization in deep networks. *eprint arXiv:1706.05394*
4. C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Viyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.