From AlexNet to DenseNet

Report for EECS 432 & 433

Fan Hu 3146124

1. **Introduction**

Since 2012, the research of computer vision has undergone tremendous changes. This change makes a profound impact on the development of computer vision in the past six years, and has promoted the development of countless innovative applications. This change is the emergence of convolutional neural network.

Since the extensive application of CNN in the computer field and in order to better understand this technology and prepare for my future development, I read a number of excellent papers from 2012 to 2016 and summarized them. Given that computer vision is so broad so I just focused on two fields, which are object classification and object detection. Besides of that, I read several other papers such as Super Resolution and image segmentation but not too much.

The purpose of this paper is to give a brief introduction, summary, and insights on these papers. I would summarize my learning at this stage to facilitate the review of knowledge later.

1. **From AlexNet to DenseNet**

2012 is an important year for Computer Vision, since a great paper appeared - Hinton’s AlexNet. It brought a huge impact to Computer Vision and totally changed the next 6 years, and it maybe even longer.

In this part, I would mostly focus on Image Classification, where CNN firstly changed. Basically, I would share my notes for AlexNet, VGGNet, GoogleNet and ResNet, which I think are four important papers or CNNs in Computer Vision. Also, I would like to share some other papers, such as MobileNet, SquzzeNet and so on.

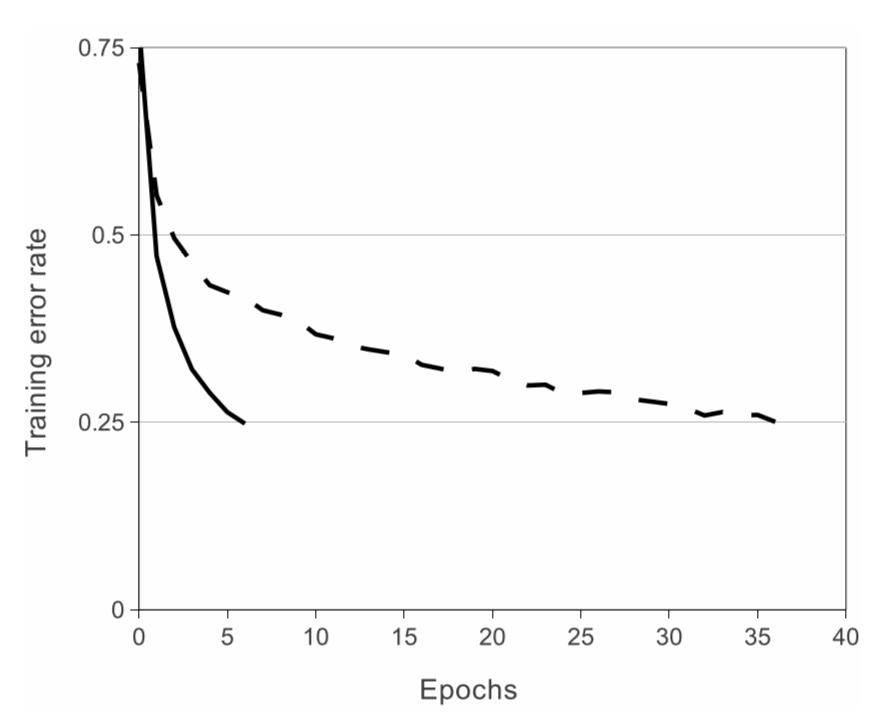
**AlexNet: Introduce CNN into image classification**

It may not be very accurate to say it is the first paper since LeNet is much earlier than AlexNet. But if we define “first” as “the first paper that realize 1000 classes image classification”, then we can say that.

Personally, I think AlexNet brings many great points into Computer Vision field: The first one introduces CNN into multiclass image classification; the first one introduces GPUs into training; using Relu, dropout and data augmentation in training. In a word, this paper points out a new way to do computer vision research and gives us a detailed guideline on how to preparing data and training them efficient. It’s performance may not very good compared with other CNNs in the next several years, but it is their “father”. More specifically, I would share some great points with more details on this paper.

**ReLu: Nonlinearity**

Linearity once killed neural network, so people now are very concerning on nonlinearity. Previously, people usually use f(x) = tanh(x) or f(x) = (1 + e−x)−1 on training. But it turns out that these would be much slower than ReLu(f(x) = max(0,x)). Figure 1 shows that a four-layer convolutional neural network with ReLUs (solid line) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons(dashed line). (Now there are some refinements of ReLu, such as Leaky ReLu and PReLu).



**Training on Multiple GPUs**

During that time, Nvidia cannot provide as powerful GPUs as today, so to train their huge datasets hinton came up with training on multiple GPUs due to the memory limit of GTX 580. Although currently Nvidia can provide you Titan with enough computation and memory, it is too expensive for some small or medium companies. So they would choose to train on multiple GPUs so that they can combine some GPUs which are not so expensive to accelerate training process. In a word, at that time multiple GPU training is for bigger networks and now is for faster training.

**Reducing Overfitting**

Since AlexNet has over 60 million parameters, it is very likely to cause overfitting. So Hinton uses two ways to reduce overfitting - data augmentation and dropout. When I read this part I was very shocked, since I never thought that they can handle this problem with less computation in this way. More importantly, I think data augmentation is really genius. We all know that translation invariant is an important feature of CNN. But it only have a little rotation invariant. So sometimes if you rotate one picture and do classification or detection you may not get what you want. But by flipping and reflection, this problem can be handled to some extent since CNN has learned how to handle rotation. Also, dropout is a important tricks to realize “combining the predictions of many different models” with much less computation. More specifically, some neurons would not contribute to the forward pass and do not participate in backpropagation. In this way, at each iteration CNN would be looked like different architecture, which would performs like combining different models. These two tricks are really important since comparing with other CNNs, AlexNet is kind of too shallow, so overfitting would be much server on these “Deep” network. But thanks to AlexNet, it points out a great way to reduce overfitting.

**Conclusion**

Generally speaking, AlexNet’s performance cannot be compared with other latter networks, even MobileNet with much less parameters(60 million vs 4.2 million), but no one would say AlexNet is not a good network. We may even can say that “ImageNet Classification with Deep Convolutional Neural Networks” is the Bible in Deep learning. Also I have to admit that I did not fully understand some parts like Local Response Normalization and Overlapping Pooling, so I will not talk about these parts.

**VGGNet: Introduce “deep” network**

One thing I need to mention is that though GoogLeNet(or Inception V1) also introduces “deep” conception, they contributes more on other fields. So VGGNet’s contribution is “deep” to some extents.

After the generation of AlexNet, more and more people bring CNN into their researches and try different tricks to improve CNNs’ performance. One example is ZFNet, which tries to shrink receptive field and gets a quite good result on ILSVRC-2013. But this kind of improvement is kind of limit. On ILSVRC-2014, there were two groups both came up with a innovative and useful method--increasing networks’ depth, which correspondingly brought VGGNet and Inception V1. I would talk about Inception families in next part, and VGGNet’s main contributions could be summarized as: deeper network with small conv layers rather than big conv. And this feature brings two great results: more nonlinearity and less parameters.

**More nonlinearity**

Instead of using one 7\*7 conv layer, VGGNet chooses using three 3\*3 layers. One reason is that all hidden layers are equipped with the rectification (ReLU) non-linearity. In this way, three ReLu is better than one since this will make the decision function more discriminative. We all know that nonlinear is better than linear. That’s why this design is good. Also, introducing 1\*1 conv also increasing nonlinearity. You may wonder 1\*1 changes nothing and it is still a linear operation, but do no forget there is a ReLu after each layer, that is why nonlinearity is introduced.Network-in-Network

**Less Parameters**

One obstacle of deeper network is it will bring more parameters. We can see that AlexNet has around 60 million parameters with just 5 conv layers and 3 FC layers. If we do not make any improvement on VGGNet with 19 layers, you can imagine how big the network would be. But actually, VGG 16 only has around 138 million parameters, which is just twice of AlexNet rather than square. There is one example in paper which I think is very vivid. Assuming that both the input and the output of a three-layer 3 × 3 convolution stack has C channels, the stack is parameterized by 3\*3^2\*C^2 = 27C^2 weights; at the same time, a single 7 × 7 conv. layer would require 7^2\*C^2 = 49\*C^2 parameters, i.e. 81% more. This can be seen as imposing a regularisation on the 7 × 7 conv. filters, forcing them to have a decomposition through the 3 × 3 filters (with non-linearity injected in between).

**Conclusion**

Also there are also some other great works in this paper, such as they proves that Local Response Normalisation is not useful and actually it will increase memory consumption and computation time. Also, they do multi-scale training and which proves to be very beneficial for training. But comparing with that it brings deeper network into our consideration, these are not so outstanding. Another thing we need to notice is that this paper proves that VGGNet also performs great on other mission such as localization. This is very important since it means that if we get a great feature extractor, we can utilize it into other fields and also get great results. That is why recent year computer vision is dominated by CNNs. The last thing I want to mention is that, deeper network will make training more difficult since gradient vanishing is more likely to happen. So VGGNet’s training process is quite tricky. They trained a shallow network first, then gradually increase network’s depth. Also, they need to pay attention to the initialization of the network weights. Finally, this problem is handled by ResNet.

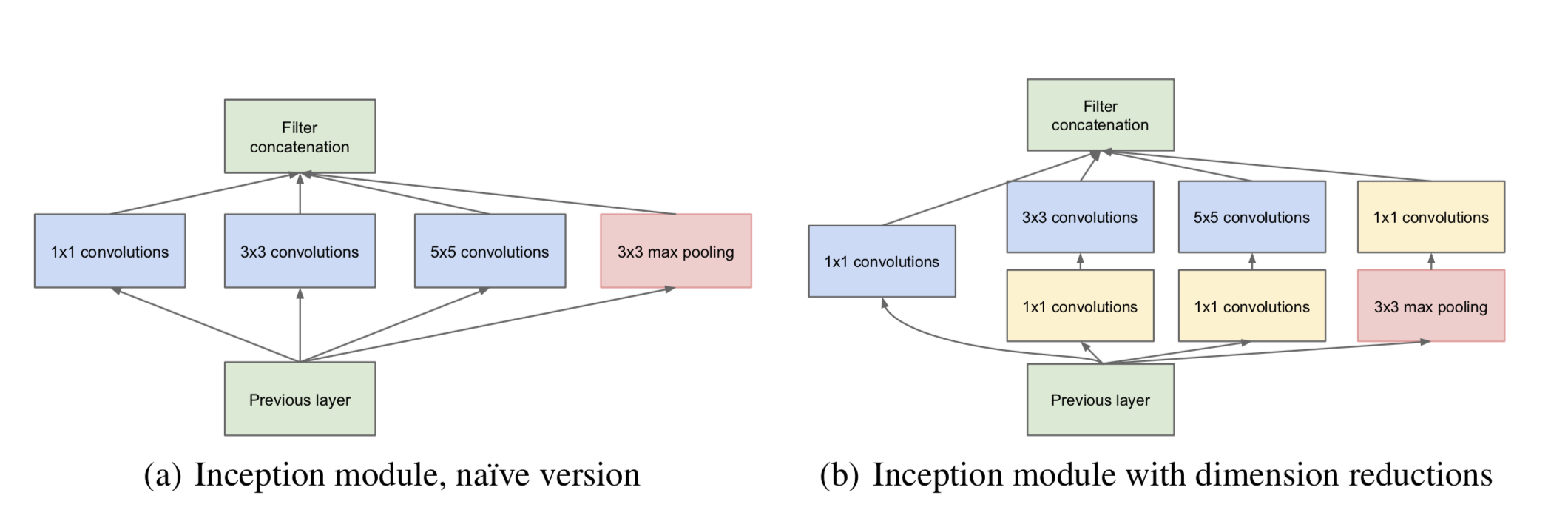
**Inceptions(v1, v2, v3, v4 and Xception):**

**V1: Both increase depth and width with 12x less parameters than AlexNet**

It seems that people noticed that increasing the size of network can dramatically improve network performance, so they start to do so. But directly increasing the size of network will inevitably cause parameters exploration, overfitting and gradient vanishing. All of them will make training difficult and hard to get a acceptable result. That’s why VGGNet replaced 5\*5 and 7\*7 convs with 3\*3 and 1\*1, also that is why Google came up with “Inception” module.

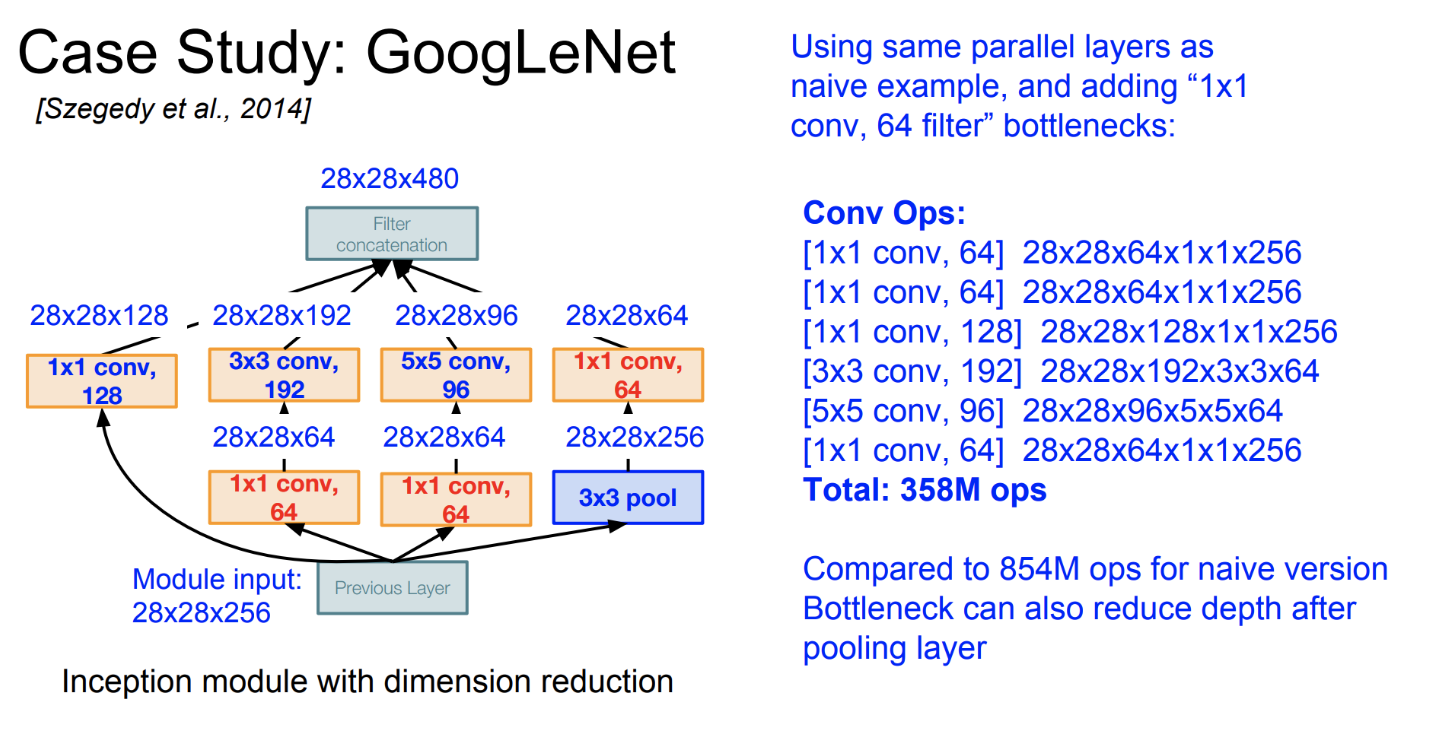
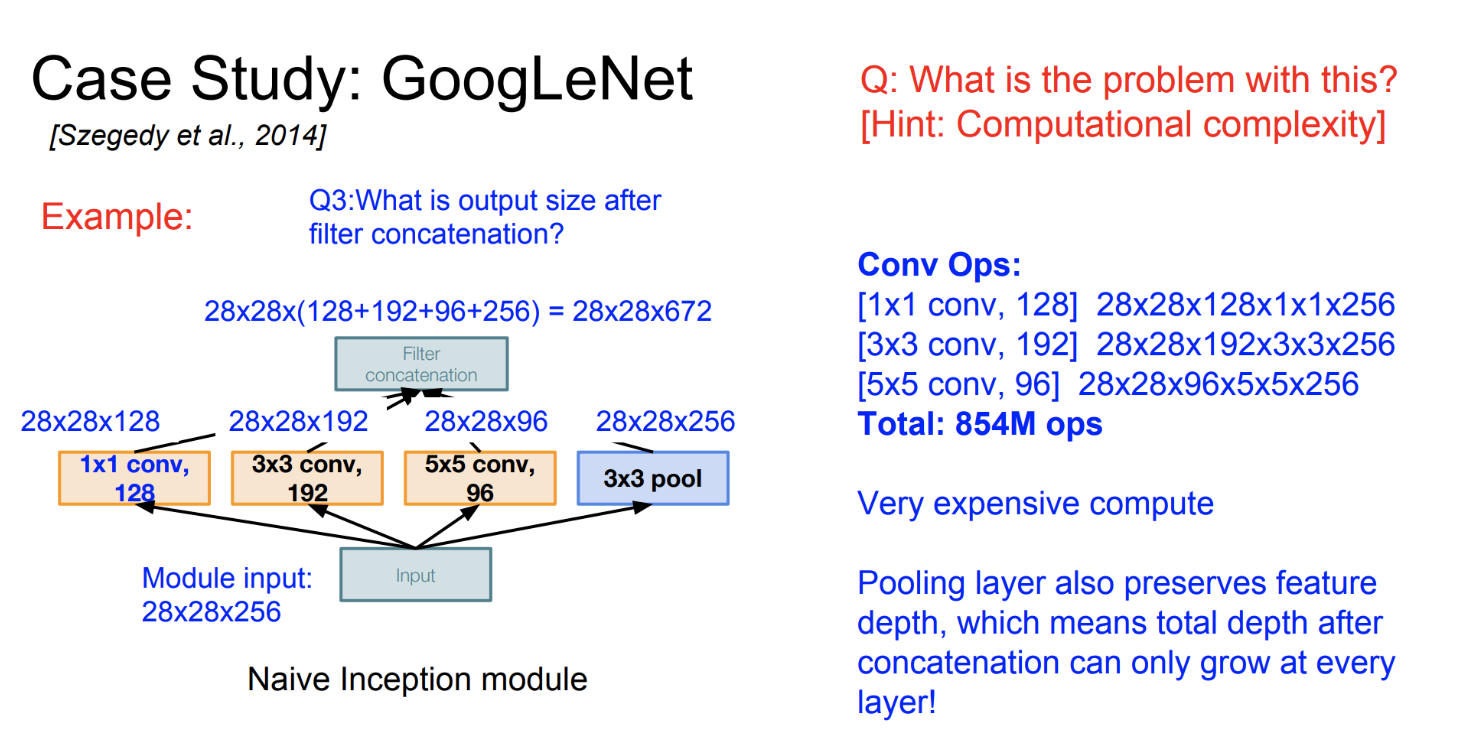
**Sparse**

There are two main ideas in this paper to improve performance. One of them has been mentioned- increasing networks size; the other is keep network as “sparse” as possible. One original solution is depicted as Figure 2 (a). This means on one specific layer, there are multiple choice for network to learn and update small part of them. For example for one kind of pictures, network may find 3\*3 and 1\*1 are more suitable and can get better result, then it will only update corresponding parameters. At the end of training it will get a relative sparse matrix. Also you can think of sparse as multiscale learning, which has been mentioned on VGGNet, while they implement this through resize pictures. I would cite one sentence in Inception V1 to better explain this. “Furthermore, the design follows the practical intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from the different scales simultaneously.”



**Reduce parameters**

It seems it will be very useful if we apply this kind of module to whole network and repeat it. But as we mentioned, this will cause parameter exploration quadratically. And this will be even server as network goes deeper. So one solution is to use 1\*1 convs, as described in Figure 2 (b). Inception V1 applied 1\*1 conv before other convs with fewer channels. Fewer channels will perform as dimension reduction, which can be explained as Figure 4 and 5. We can see that 1\*1 will not change output’s size but its channel number. That’s why we can say 1\*1 conv can perform as dimmension reduction.



**Conclusion**

Personally, I think this paper is very accurate. It goes like a record of experiment. It firstly explains what we can do if we want to improve performance, then introduce what’s the obstacle. After that it gives you a solution. Also it mentions that although Figure 2(b) might be a great solution but they did not apply this module to whole network because the limit of hardware. I just think this paper shows many details of how they design and what’s their final result.

**V2: Batch Normalization accelerate training**

Compared with V1, Inception V2 does not change too much on network structure, but it introduce “Batch Normalization” which proves that this can largely accelerate training and also eliminate dropout and initialization. I have to admit that I am not totally understand why “Batch Normalization” works, so I will cite others’ opinion here, to make a record and help me better review.

“Batch normalization is normalization strategy, which makes the distribution of layers’ input consistent at the output of layers. Citate in paper: Batch normalization eliminate the effect of internal covariant shift.(What’s internal covariate shift?\*\* The input distribution to a learning system changes during training.) Covariate shift would amplify permutation at the deeper layers. If it happens, the inputs of activation function will stay in the saturated region. In that region, the gradient will be very small and the phenomenon of vanishing gradient would happen and stop neural network to train.

Therefore, the batch normalization layer is to force the distribution of layers’ input to remain fixed over training time. Without batch normalization, sometimes the neural network used ReLU activation function still works with careful initialization and small learning rate. This is because neural network will put more effort to compensate for the changes in the distribution. Usually, it would degrade the performance of neural network.” - “<http://gwansiu.com/2017/06/06/BN/>”

**V3: General Design Principles**

Personally, this paper’s contribution are giving four general design principles and more explanations on convolution decomposition. Design principles are mostly based on their previously experiments.

**Design Principles**

1. Avoid representational bottlenecks, especially early in the network. To my understanding, this means that if you want to do dimension reduction, do not do it early, or you may lose many useful information. Also, it is not recommend to largely reducing dimension only on several layers. Intead, it is better to slowly decrease dimension through whole network. So, try to keep more information until you decide to reduce them.
2. Higher dimensional representations are easier to process locally within a network. Increasing the activations per tile in a convolutional network allows for more disentangled features. The resulting networks will train faster. To some extents we can take this as “more nonlinear activation can make training faster”
3. Spatial aggregation can be done over lower dimensional embeddings without much or any loss in representational power. This is very interesting and useful, since that is why using a 1\*1 conv can help accelerate training without reducing too much accuracy. Google team hypothesize that it is because there is a strong correlation between adjacent units. So even though we do dimension reduction and abandon many informations, useful informations are kept by adjacent units.
4. Balance the width and depth of the network. Increasing width or depth both increase the performance of network, but they will also inevitably increase computation budget. So you really need to think about it and make a balance.

**Convolution Factorization**

In this paper, they also mentioned what VGGNet has proved--Using smaller convs to replace big convs. As a later publication, this is not innovate any more. But the interesting thing is that they also come up with “Asymmetric Convolutions”. This means that instead of using 3\*3 conv, they would use 3\*1 and 1\*3. It’s obvious that you can get the same conv result with 33% less parameters, which is largely used in their latter work[MobileNet].

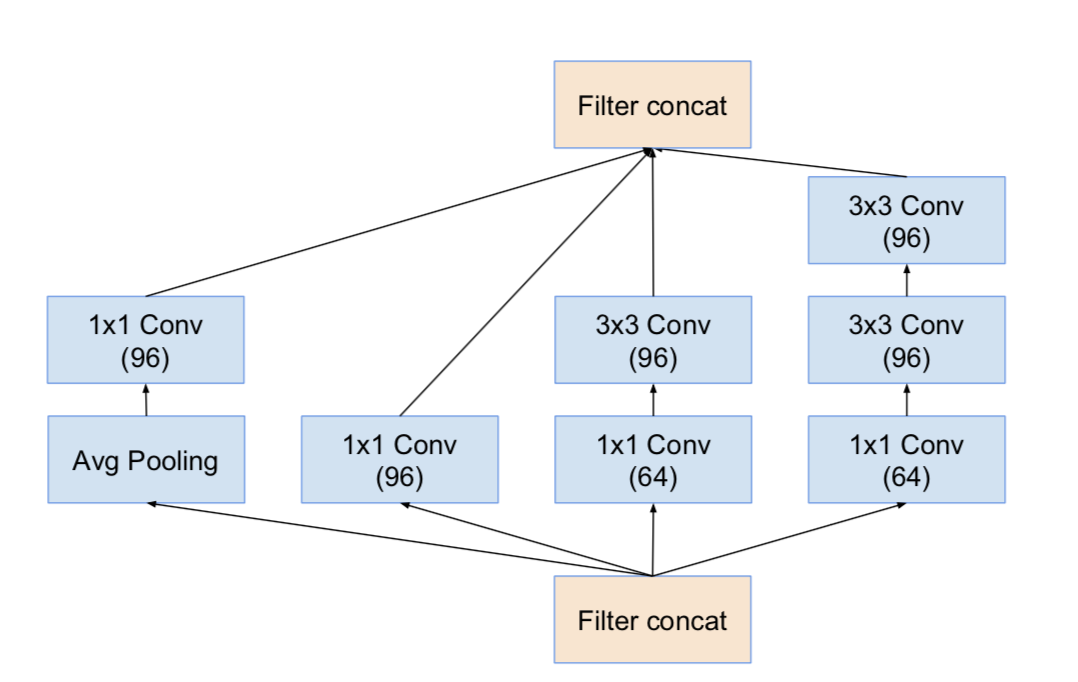
**Conclusion**

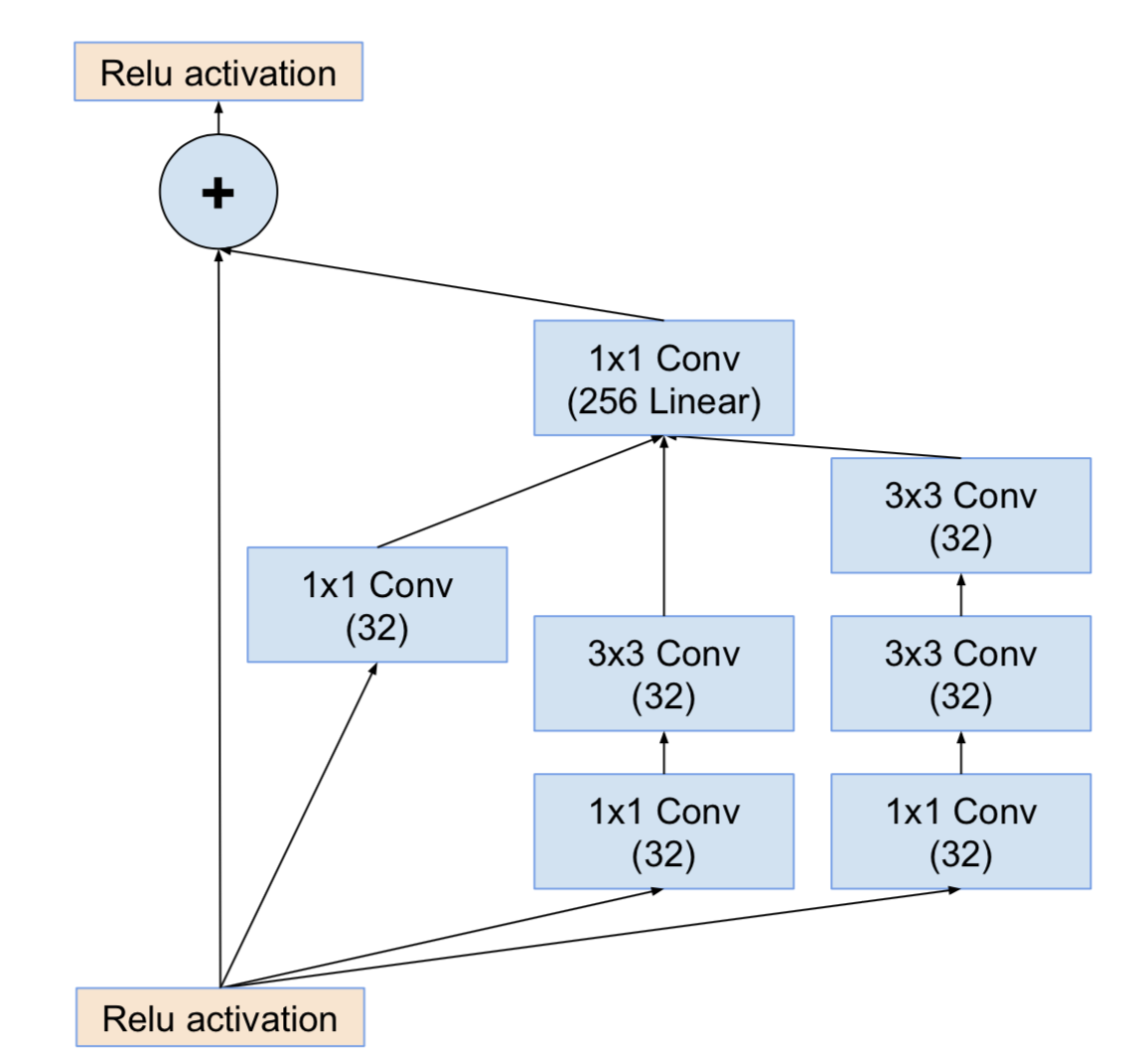
I like two parts in this paper, one is their hypothesis of why low dimension information can be abandoned without too much accuracy loss, the other is their asymmetric convs. Also, there are some parts that I am not fully understand, such as the “Utility of Auxiliary Classifiers” part. I am kind of lost and I even cannot find what “Auxiliary Classifiers” refers to in Inception V1. Apart from that, I think “Efficient Grid Size Reduction” is also a very interesting idea, although I don’t fully understand it either.

**V4: Inception with ResNet**

Inspired by Kaiming’s ResNet, Google team came up with an idea and decided to try combining Inception architecture with residual connections. And it turned out to be useful.

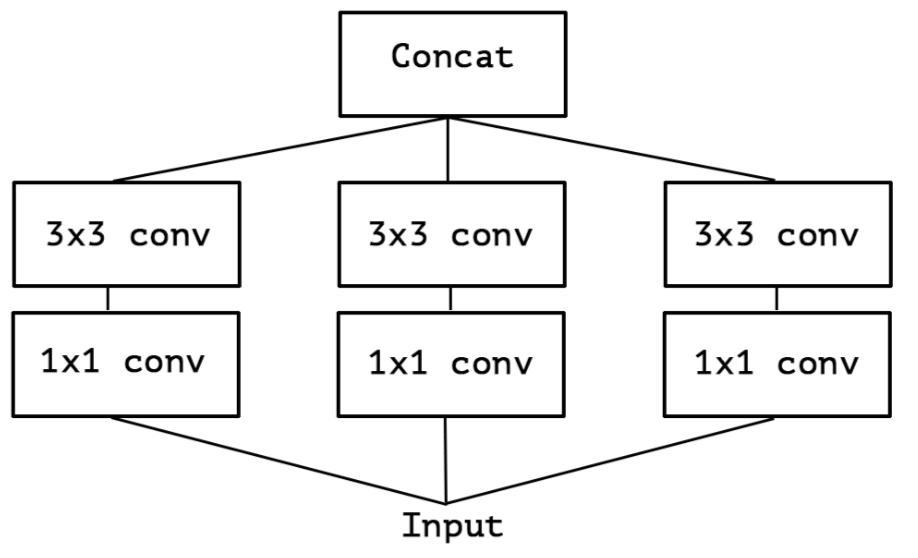
The main idea of this paper actually is very simple, google team did quite a lot of experiment and compare different modules’ performance. And their result shows that inception module with residual module converges faster, but the final error rate does not increase significantly. One part of their modules are show as Figure . We can see that the main difference between these two modules are there is a direct connection two layers, which is ResNet’s idea. I would share more details of ResNet in the following part.

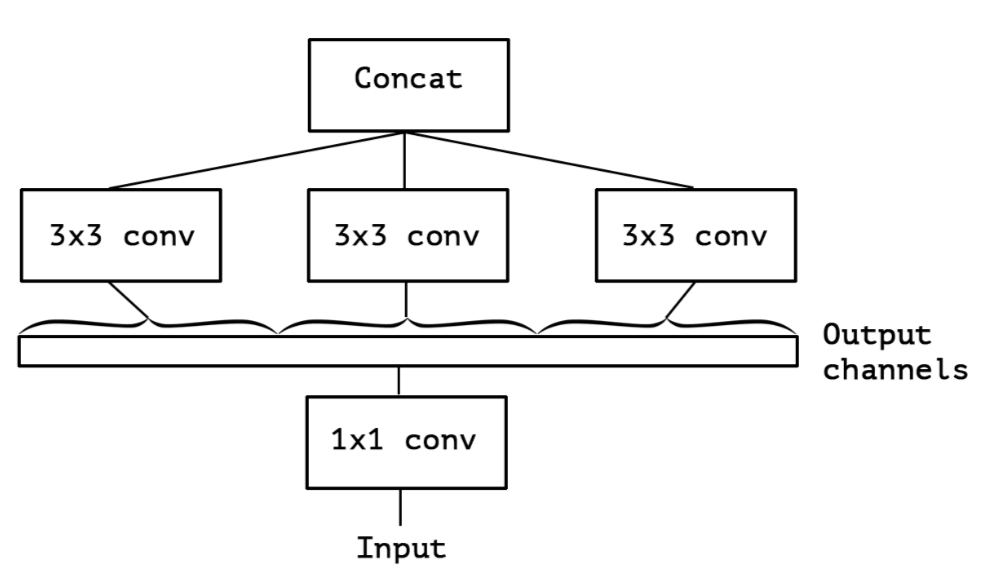




**Xception: Inception with Depthwise Separable Convolutions**

Xception is an update based on Inception-v4 and depthwise separable convolutions, its main purpose is to improve the accuracy of current network, while MobileNet focuses on compress network to make it runnable on mobile phones. As we mentioned before, InceptionNets are mostly using parallel convolution to enhance the express of network, which can be simplified as Figure. And actually, we can use an union 1\*1 layer to do convolution and each following 3\*3 takes part of its results and do another convolution. After that, using another 1\*1 conv to join these results together, which can be showed as Figure.





The main benefit of this is that it can largely reduce parameters. Since a standard convolution’s parameters can be calculated as

PNormal=I∗Dk∗Dk∗O

while a depthwise separable convolutions can be calculated as

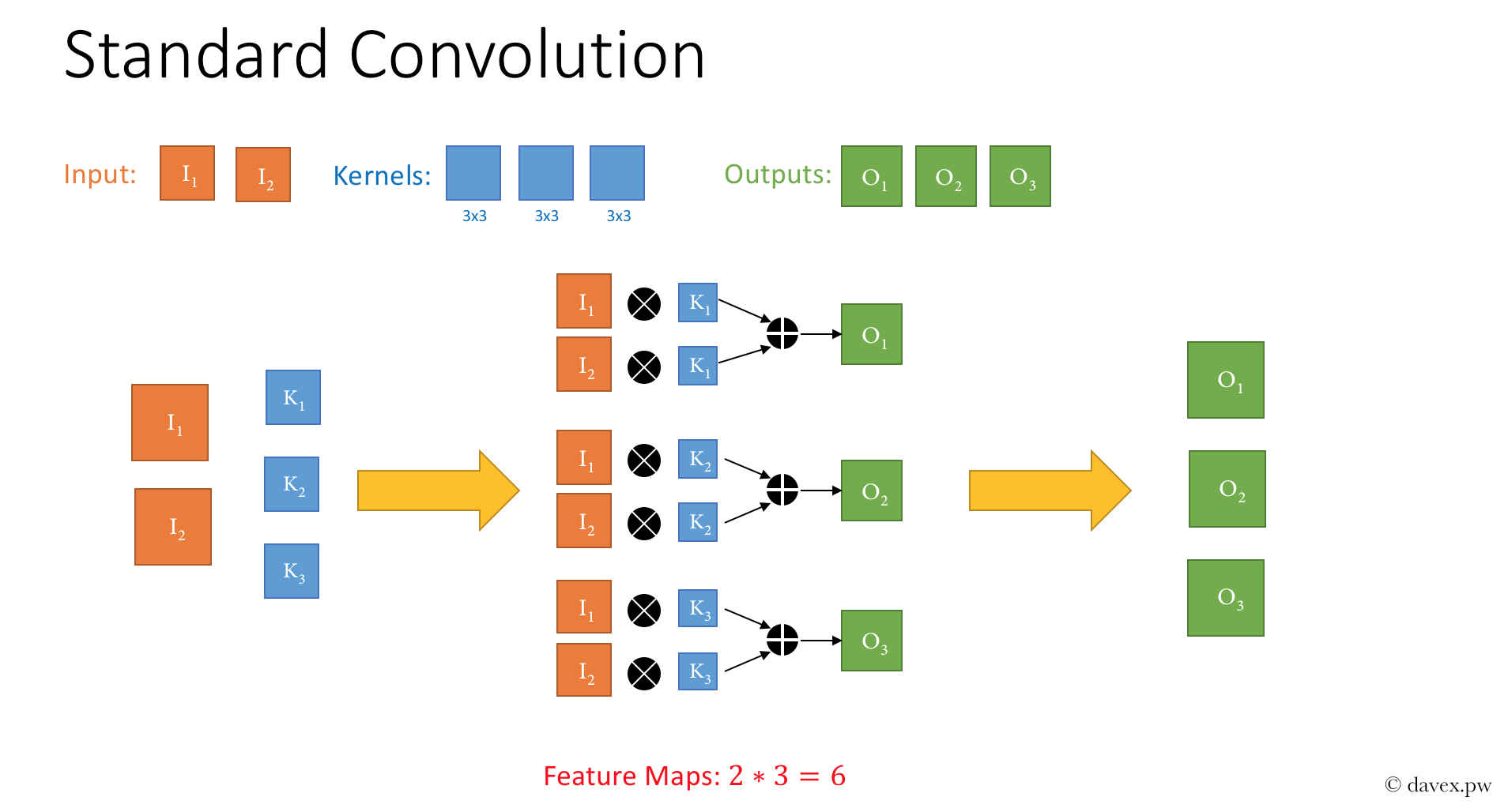
PDW=I∗Dk∗Dk+I∗O

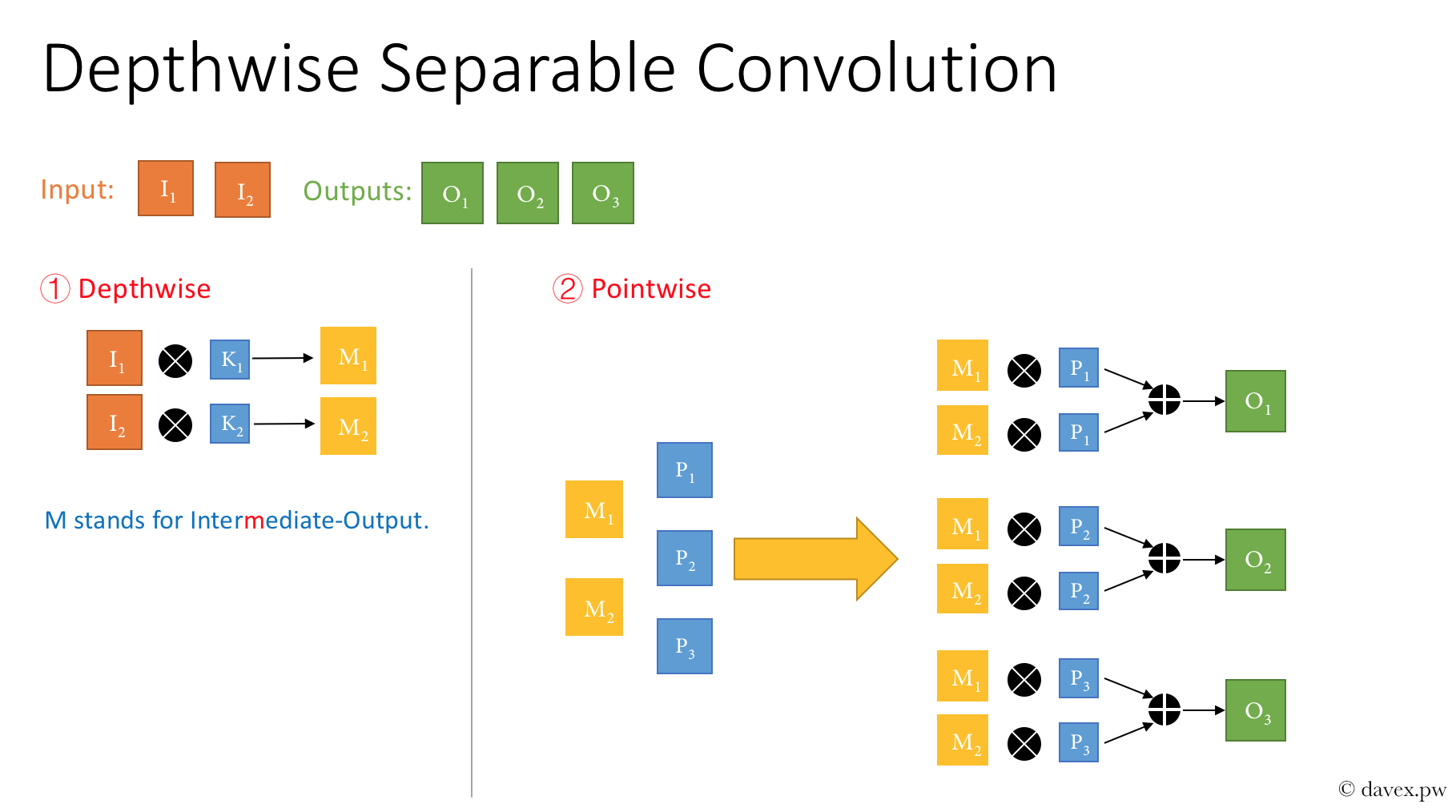
compare these two we can get

PDW/PNormal=1/O+1/Dk^2≈1/Dk^2

Where I is the number of input channels, O is the number of output channels, and Dk is the size of the standard convolution kernel.

We can see that when we use the 3x3 convolution kernel, the parameter size is approximately equal to the standard convolution kernel 1/9 , which greatly reduces the amount of parameters, thus speeding up the training. Also, I found two helpful Figures for me and I would like to share them here.





**Conslusion for Google’s work:**

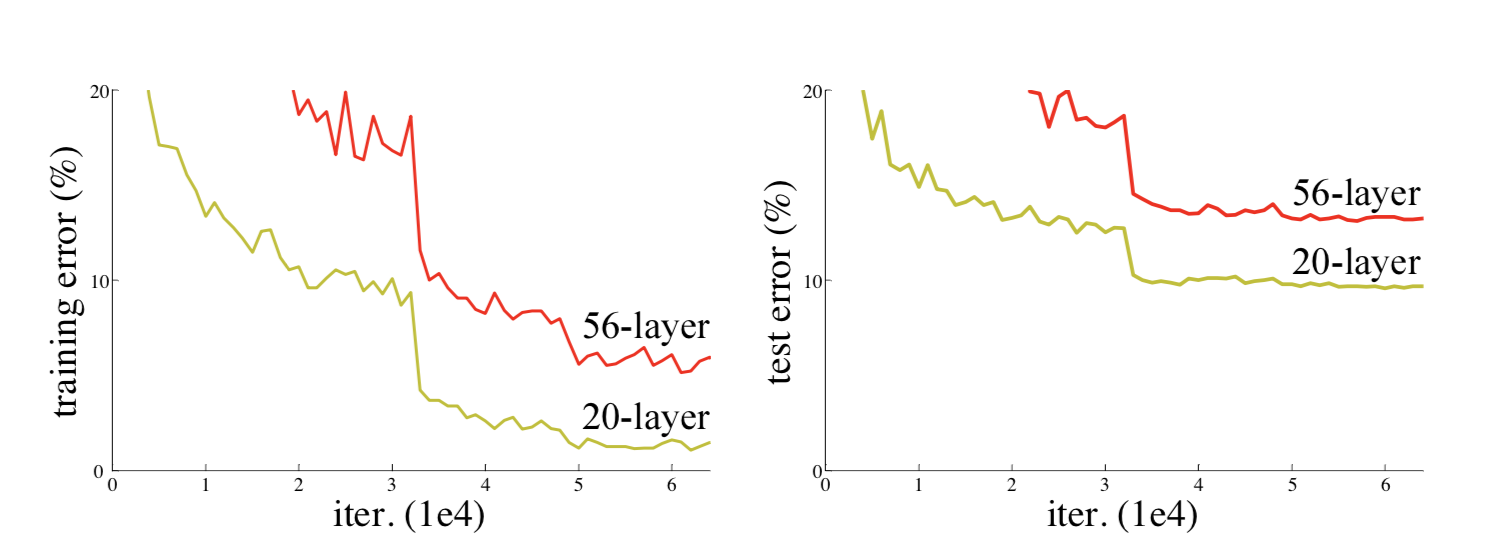
In order to increase the depth and width of the network and control parameters at the same time, inception v1 is proposed; then, through the study of the characteristics between the conv layer and the conv layer, the BN method is proposed to form Inception v2; in Inception v3, the factorization is extremely used, even on the pooling layer, which further reducing the total amount of parameters, thereby improving network performance. After completing the network redundancy, they thought of combining Inception v3 and the residual module to see what would happen, and there goes inception v4. Finally, the 1x1 con is actively played, separating cross-channel correlation and spatial correlation, and creating Xception. Google is genius!

**ResNet: Make CNN goes much deeper!**

Through AlexNet to Inception-v3, we can see that as the size of the network increases, the performance of the network will also increase significantly. But this will bring a huge computation if we do it without any planning. In addition, deep network will make it difficult to train, which we call gradient descent. Before ResNet, the deepest network I know is Inception-v2 with 22 layers, while ResNet brings a 152 layers network with lower complexity and better performance than VGG(They even built a 1202 layers network which did not perform as good as 152 layers network).

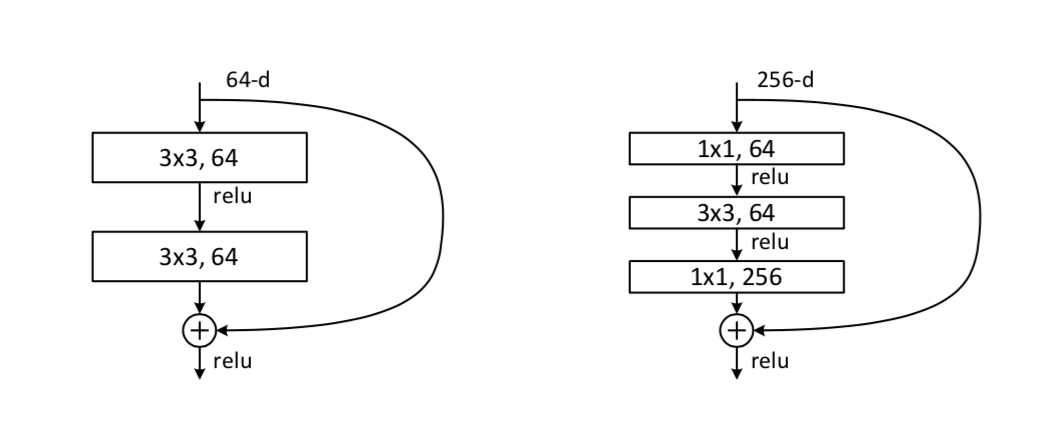
**Introduction**

We have known that it is quite common to get overfitting if our network is too big, and when we try to increase the depth of network and get a bad test result, we are likely to think that it is because of overfitting. However, it is not true. From Figure we can see that, as network goes deeper, the training error will be worse. If bad result is caused by overfitting then we are supposed to get a much better performance, which in turn indirectly shows that the reason may be the network does not converge at all. Also, Kaiming He argues that “this optimization difficulty is unlikely to be caused by vanishing gradients. These plain networks are trained with BN, which ensures forward propagated signals to have non-zero variances. ”

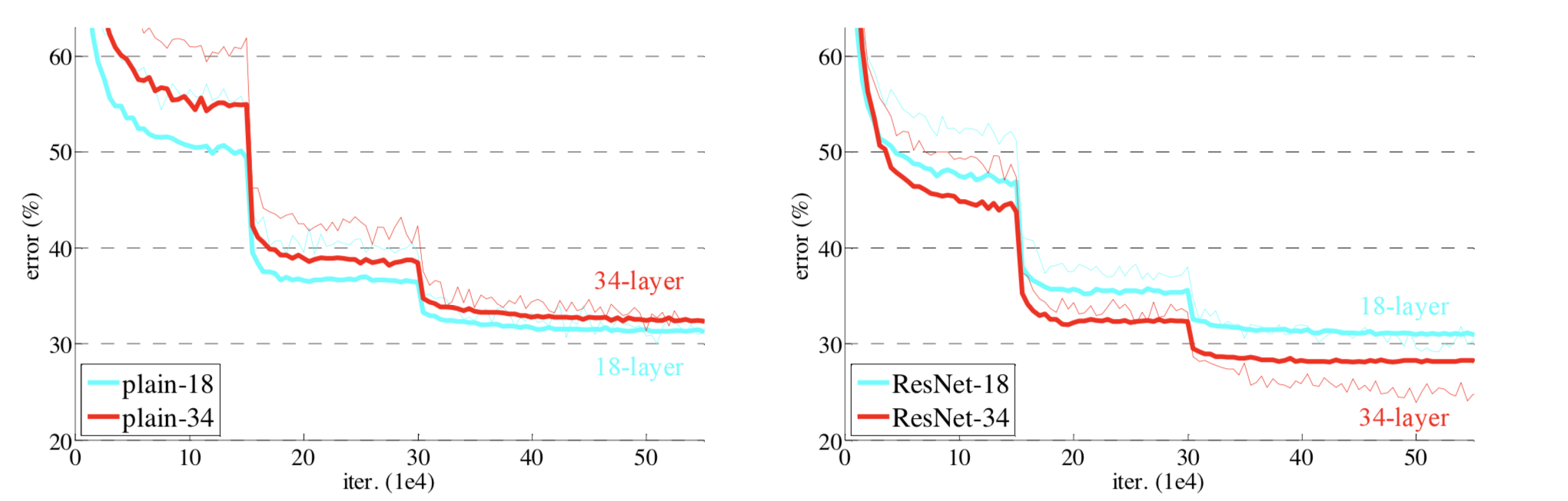


**Residual Learning**

MSRA group came up with “residual learning” inspired by VLAD’s residual vectors, which can be shown as Figure ‘s left part. Formally, denoting the desired underlying mapping as H(x), they let the stacked nonlinear layers fit another mapping of F(x) = H(x) - x. The original mapping is recast into F(x) + x. They hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. The formulation of F (x) + x can be realized by feedforward neural networks with “shortcut connections” without adding any extra parameter or computational complexity. One thing need to mention is that the final version of F(x) is fitted by three layers-first comes with a 1\*1 conv to reduce dimension, which has been talked in Inception part, then a 3\*3 to do convolution, finally using another 1\*1 to add dimension.



To my understanding, this “Residual” is like a short circuit in physics. When do backpropagation, it seems the network goes deeper, but it maybe actually run shallower, which helps network converge much faster.(Kaiming’s experiment also proves that, see Figure )



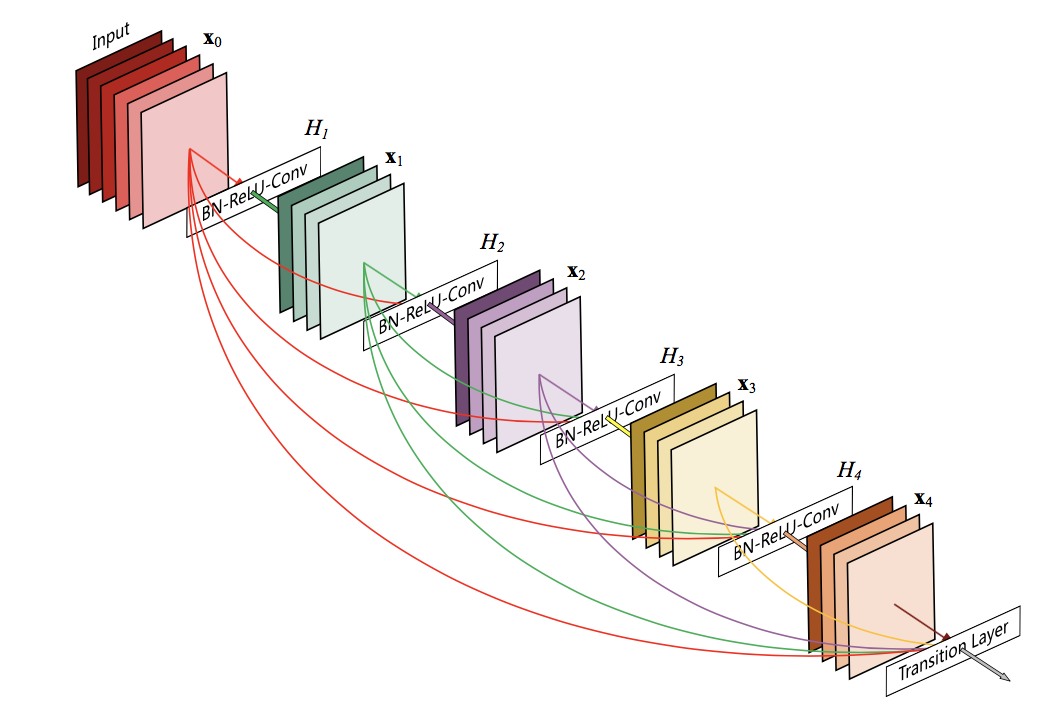
**Conclusion**

ResNet proposes a two-way method of identity mapping for gradient disappearance problems and network degradation problems. Shortcut connection helps ResNet avoid using helper function to help the shallow network receive gradient information like GoogLeNet. The Residual learning idea solves the problem of deep model optimization, and finally the network can be further deepened up to thousands of layers. And the shortcut connection scheme further inspired the birth of the subsequent DenseNet. After the analysis of these networks, the network has been pursuing in-depth development. The depth is the final decision of the network effect. Because the depth of the network means the increase of the nonlinear layer, the network's ability to fit complex functions can be increased.

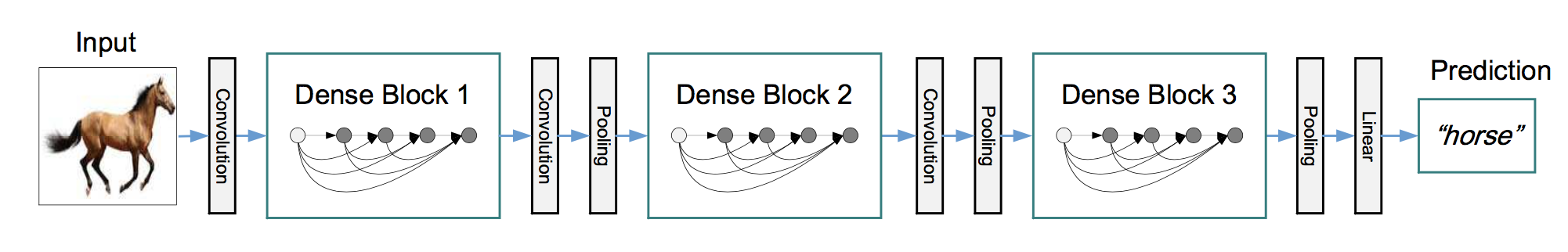
One little thing need to mention is that when to residual learning, H(x) = F(x) + x, sometimes there is a dimension change, so it is necessary to do projection shortcuts. Kaiming’s team tries three ways to do so, which are (A) zero-padding shortcuts are used for increasing dimensions, and all shortcuts are parameter-free (the same as Table 2 and Fig. 4 right); (B) projection shortcuts are used for increasing dimensions, and other shortcuts are identity; and (C) all shortcuts are projections. Though the final performance is C > B > A, but it costs too much. So it’s better to use A or B, which will not introduce too much parameters.

DenseNet: Fully Connected ResNet

Inspired by ResNet, DenseNet group decided to untilize shortcut connections wordwidely. They built module called dense block, where each layer is connected with each other. By this way,DenseNet effectively slows down the gradient disappearance and degradation behavior, diversify the input characteristics of each layer of the network to make the calculation more efficient, and the use of the shortcut connection plays the role of deep supervised learning. Figure shows how this module looks like.



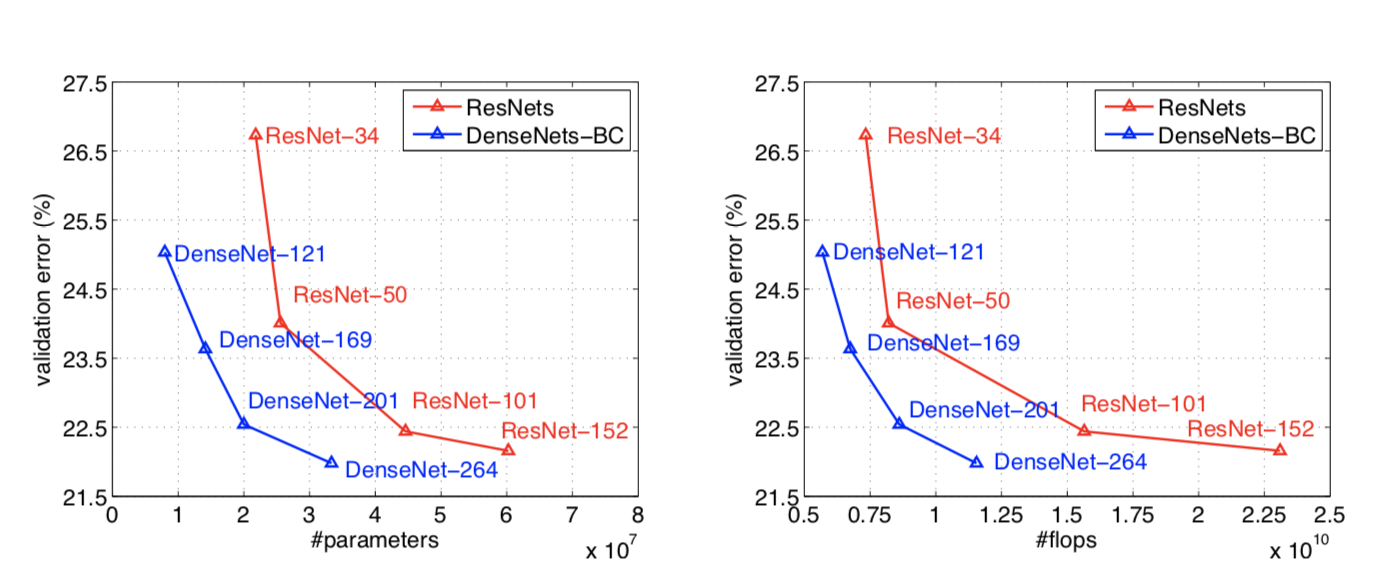
There are several dense modules in this network. Each module is connected by normal convolution layers and pooling layers. The overal structure is showed as Figure. Now we will talk more about its structure details.



1. Concatenation in Dense Block: Each layer of output is simply merged with its own output and passed to the next layer of input. This diversifies the input characteristics of the next layer, effectively improving the calculations and helping the network to integrate the shallow network features to learn the discriminative feature. At the same time, the neurons in the same Dense block are connected to each other to achieve the effect of feature reused, which is why DenseNet does not need to be wide and can achieve good results. (Note: Both depth and width are key factors in the network, and the width of the network must be functional after the network has achieved a certain effect). In addition, the reason why they do not choose to use ResNet addition method is because addition is a simple feature fusion behavior, which will result in loss or disorder of information.
2. Compression in transition layer: One characteristic of DenseNet is that the parameter usage is far less than that of ResNet. In addition to feature reused, the network width is reduced. In the transition layer, the 1x1 convolution is used for information compression. This is in Inception v3. The technique used in the model makes the model more compact. In addition, the additional benefit of reducing the amount of parameters is to reduce the complexity of the model to prevent over-fitting.
3. Deeply supervision: Shortcut connections form a multi-path model, so that the flow of information from the input to the output is unimpeded. Gradient information can also be directly fed back from the loss function directly to each node of the network. It looks like there is a brain that directly controls the behavior of the body part.

Conclusion:

DenseNet may looks similar of ResNet, however, it’s a brand new network structure, which largely decrease the depth and parameters of network, while largely improve the efficiency during training and accuracy during test. I think Figure can vividly prove how DesNet outperform ResNet both on accuracy and complexity.



1. **From R-CNN to SSD**
2. 打算最后做一个各个论文的发布时间/相互依赖关系