UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Anashua Krittika Dastidar
Teaching Assistant

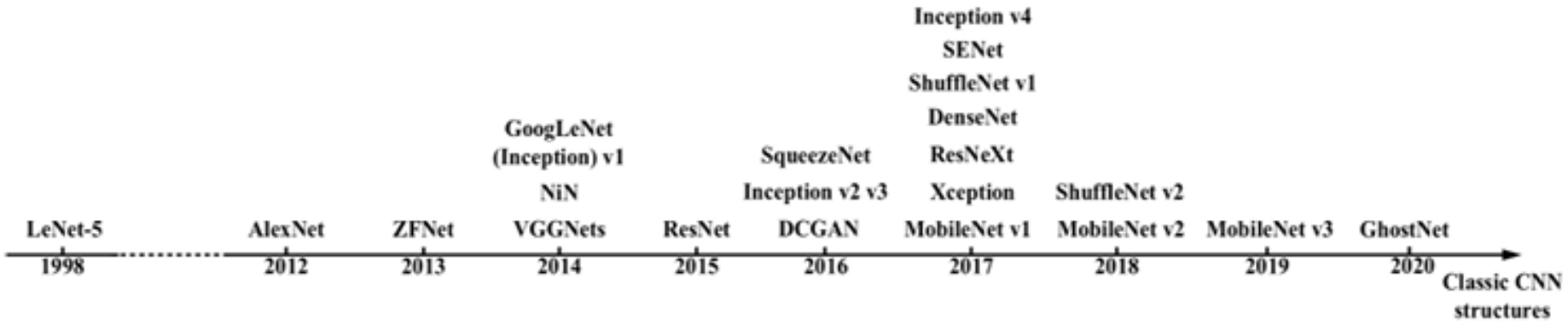
Introduction

A convolutional neural network (CNN) is one of the most significant networks in the deep learning field. Since CNN made impressive achievements in many areas, including but not limited to computer vision and natural language processing, it attracted much attention from both industry and academia in the past few years. In this lecture we shall start by discussing some historically famous CNN models and move up to state of the art complex CNN architectures.

Looking to study CNN's from a historical perspective :

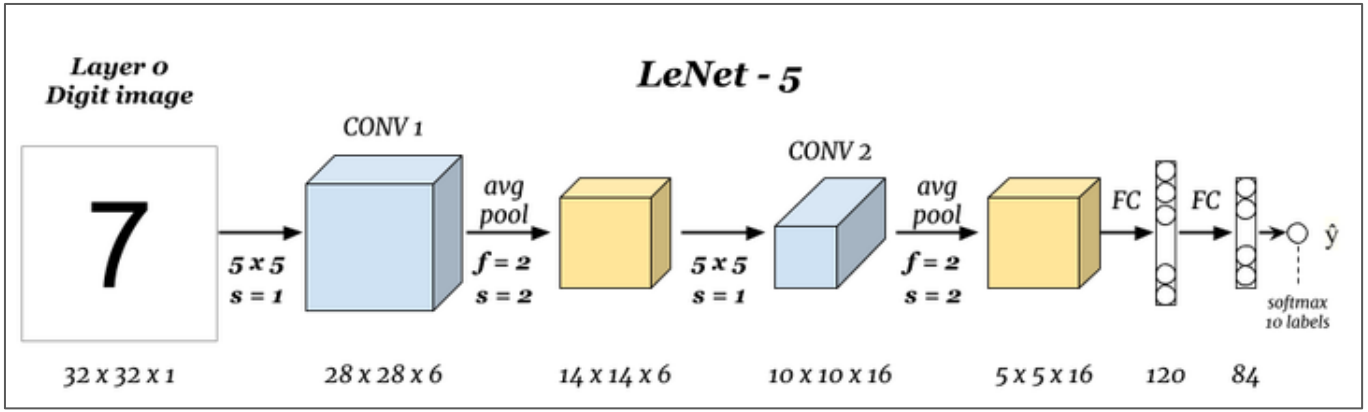
- Waibel et al. [1] proposed a time-delay neural network for speech recognition, which can be viewed as a 1-D CNN.
- Then, Zhang [2] proposed the first 2-D CNN—shift-invariant ANN.
- LeCun et al. [3] also constructed a CNN for handwritten zip code recognition and first used the term "convolution," which is the original version of LeNet.

A Timeline of CNN Architectures ….



LeNet

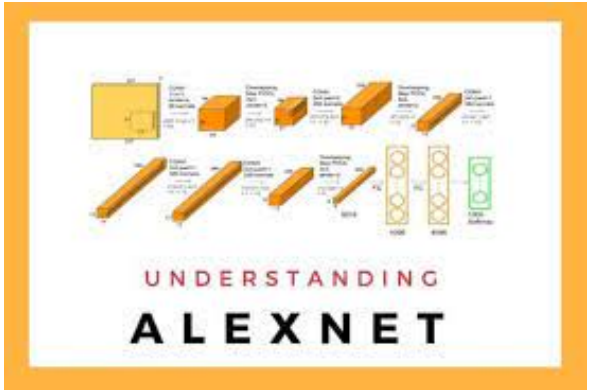
- LeNet5[3] architecture was introduced in 1998. Due to its historical importance, it is known as the first CNN model.
- It has also made great function in MNIST handwritten digital ID patterns.
- LeNet model, which is usually composed of 5 layers, accepts grayscale images with 32 x 32 x 1 as input.
- The inputs are transferred to the Conv layer and then to sub-sampling.
- Afterward, there are other Conv layers, followed by a pooling layer, and at the end of the architecture, FC layers including the output at the last layer are defined.



- This model was the first CNN architecture reducing the number of parameters and capable of automatically learning features from raw pixels. This model was introduced to identify digital manuscript patterns and postal codes in post offices

AlexNet

- AlexNet [4] was the pioneering deep architecture, with top-5 test accuracy of 84.6% on ImageNet data(15 million labeled high resolution images in over 22,000 categories).
- It utilized those data augmentation methods that are comprised of image translation, patch extractions(random cropping), colour jittering and horizontal reflection.
- This CNN model implements dropout layers with a particular end goal to battle the issue of overfitting to the training data.
- It is trained by batch stochastic gradient descent, along with particular values for weight decay and momentum. The activation function used was ReLU.
- They train on ImageNet 2011 dataset and finetune on ImageNet 2012 dataset.
- They introduced ideas like local response normalization which are quiet out of practice today as batch normalization is preferred.
- It was trained on two GTX 580 GPUs for 5–6 days and contains five convolutional layers, one max pooling, ReLU as non-linearities, three FC layers and dropout.



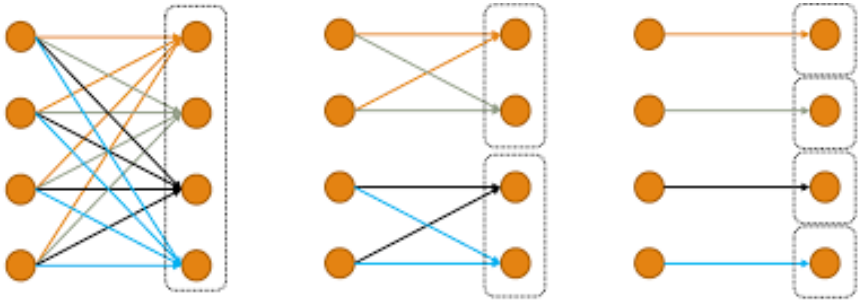
Please Note ! that however these things are pretty common to do today when AlexNet paper came out they were not the norm.

The creators of AlexNet pioneered the idea of Group convolution to train on multiple GPUs, let's see how ..

What is Group Convolution?

Usually convolution filters are applied on an image layer by layer as we have seen till now in LeNet . However to learn more features and train deeper networks we could create two models that train and backpropagate in parallel. This method of using different set of convolution filter groups on the same image is called “grouped convolution”.
As we have learnt in previous courses parallelism in training is two ways:

1. Data parallelism: where we split the dataset into chunks and then we train on each chunk. Intuitively, each chunk can be understood as a mini batch used in mini batch gradient descent. Smaller the chunks, more data parallelism we can squeeze out of it. However, smaller chunks also mean that we are doing more like stochastic than batch gradient descent for training. This would result in slower and sometimes poorer convergence.



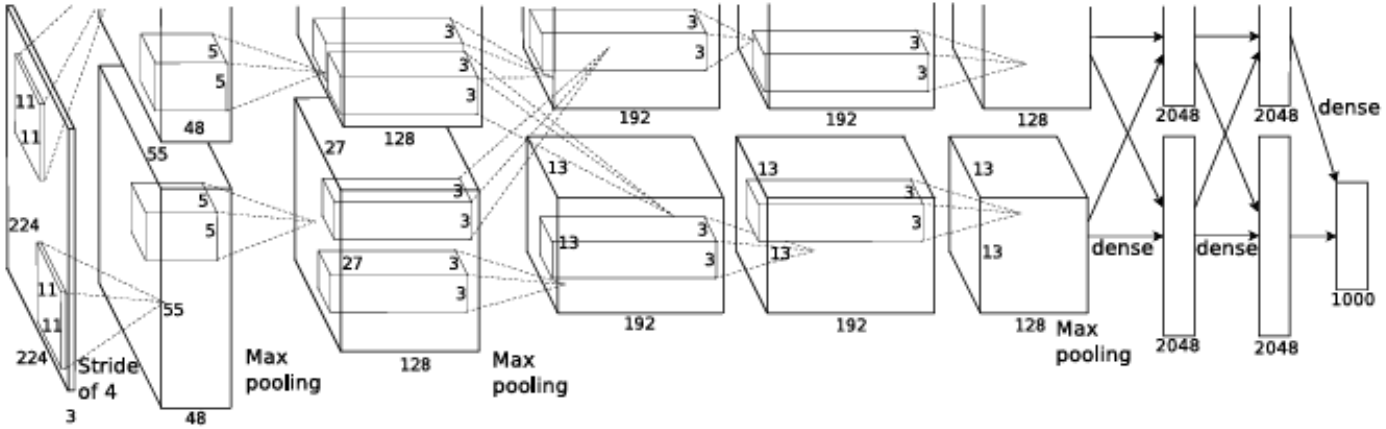
2. Model parallelism: here we try to parallelize the model such that we can take in as much as data as possible. Grouped convolutions enable efficient model parallelism, so much so that Alexnet was trained on GPUs with only 3GB RAM.

For more details read this article.

Group Convolution in AlexNet

- A single GTX 580 GPU had only 3GB of memory, which limits the maximum size of the networks that can be trained on it. It turns out that 1.2 million training examples are enough to train networks which are too big to fit on one GPU. Therefore they spread the net across two GPUs.

- This parallelization scheme puts half the neurons on each GPU However the GPUs communicate only in certain layers , for example, the kernels of layer 3 take input from all kernel maps in layer 2. However, kernels in layer 4 take input only from those kernel maps in layer 3 which reside on the same GPU.



- Choosing the pattern of connectivity is a problem for cross-validation, but this allowed them to precisely tune the amount of communication until it is an acceptable fraction of the amount of computation.

Group Convolution in AlexNet

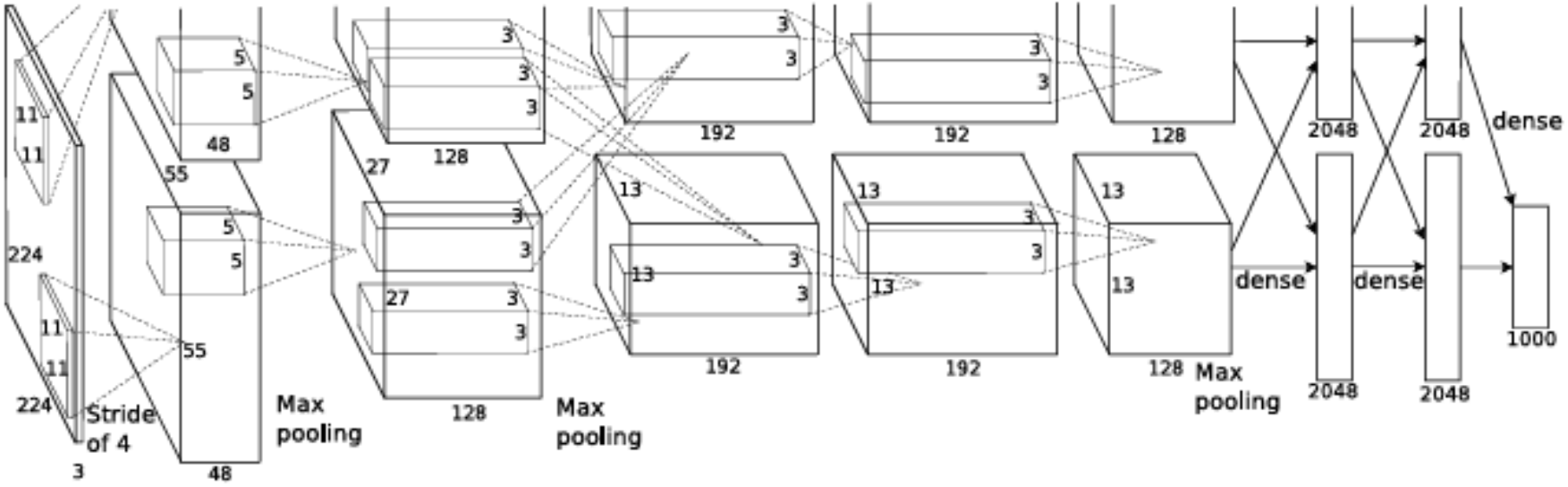
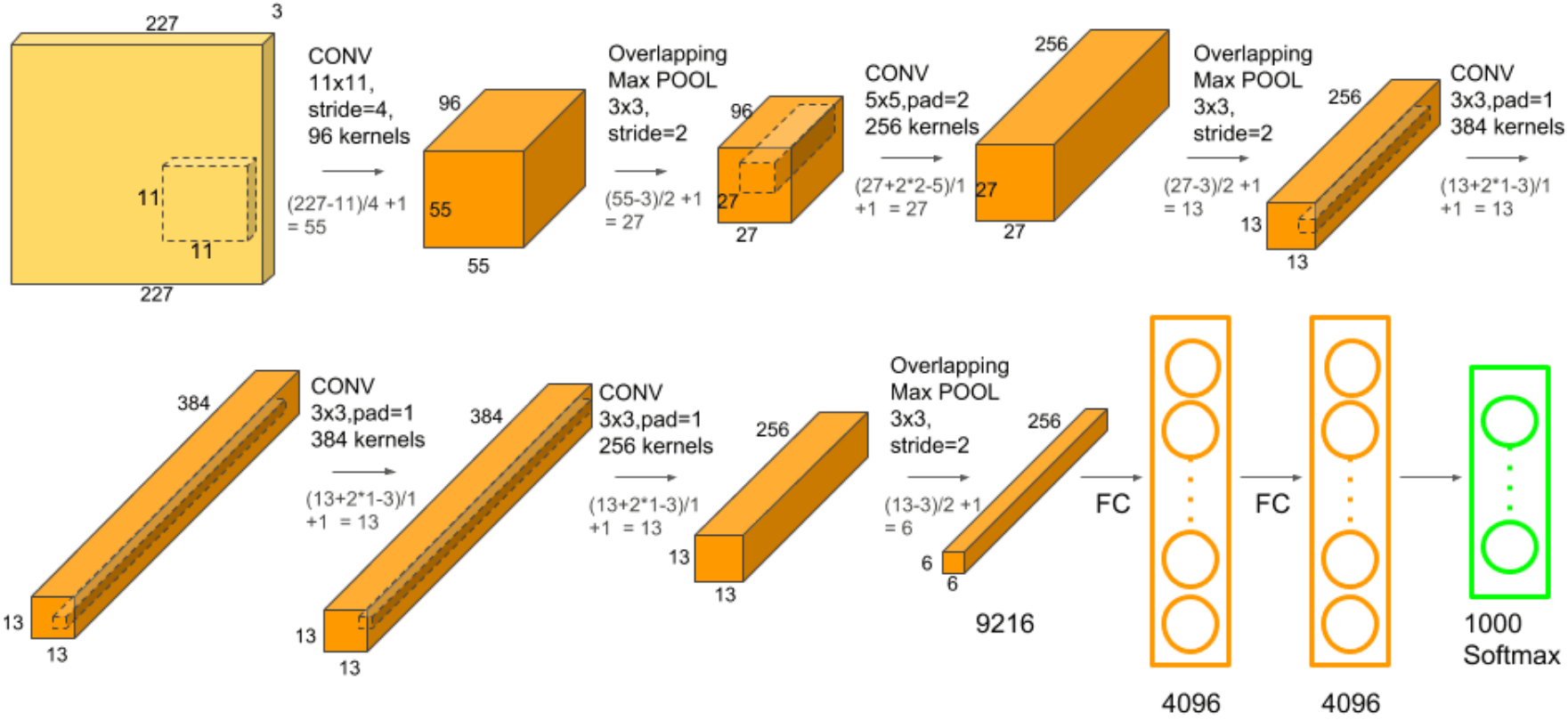


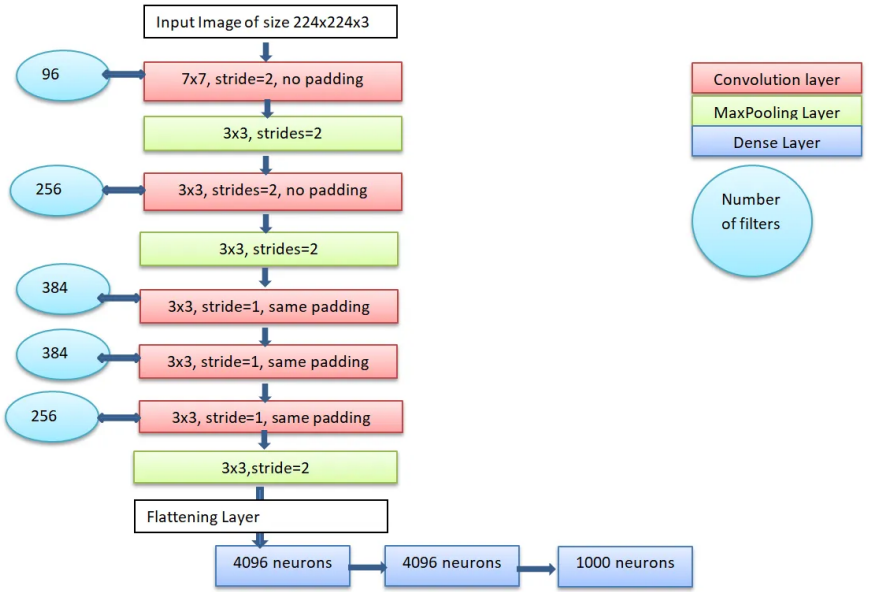
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

AlexNet Architecture



ZFNet

- The paper Visualizing and Understanding Convolutional Networks[5] introduces the notion of Deconvnet which enables us to visualize each layer.
- By visualizing each layer, we can get more insight about what the model is learning and thus, make some adjustments to make it more optimize
- That's how ZFnet was created, an AlexNet fine-tuned version based on visualization results.



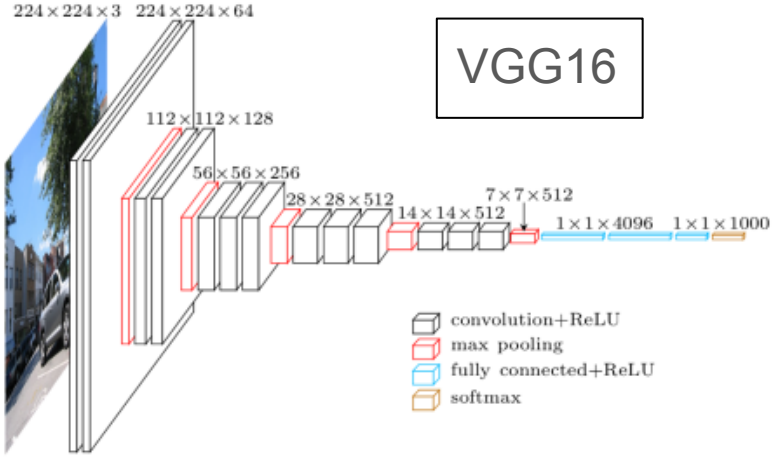
Differences with AlexNet:

AlexNet consists of eight layers, five convolutional layers followed by three fully connected layers. ZFNet retained basic architecture of AlexNet but made some architectural adjustments,particularly in the first few layers.

- AlexNet used 11x11,5x5 and 3x3 filter sizes while ZFNet used 7x7 filter size in the first layer only and 3x3 in the latter layers only.
- There is stride of 4 in the first layer of AlexNet while in ZFNet there is stride of 2 used.
- AlexNet used Local Response Normalization while ZFNet used Local Contrast Normalization

VGGNet

- The VGG model, or VGGNet, that supports 16 layers is also referred to as VGG16, which is a convolutional neural network model proposed in the research paper titled, “Very Deep Convolutional Networks for Large-Scale Image Recognition.”
- The VGG16 model achieves almost 92.7% top-5 test accuracy in ImageNet.
- The previous CNNs like AlexNet had a ton of hyperparameters which were filter sizes and sizes of the max pool layers, moreover the stride value for each layer was also a hyperparameter.
- This network simplifies it by having only 3x3 kernels for the convolution layer with a stride of 1 and 2x2 for the max pool layers with a stride of 2.
- The VGG16 model was trained using Nvidia Titan Black GPUs for multiple weeks.



VGG16

- The concept of the VGG19 model (also VGGNet-19) is the same as the VGG16 except that it supports 19 layers.

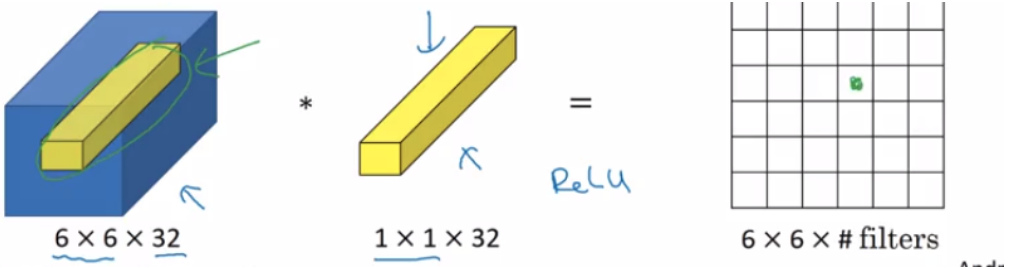
Inception Net

Fun fact, this meme was referenced in the first inception net paper.

The origin of the name 'Inception Network' is very interesting since it comes from the famous movie Inception, directed by Christopher Nolan.

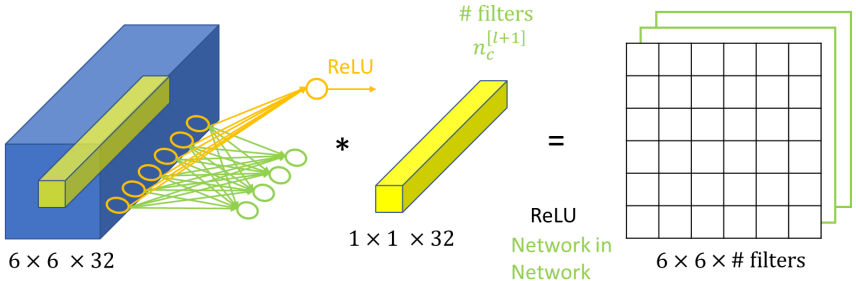
In the movie, we have dreams within dreams. In an inception network, we have networks within networks.

What is an 1x1 convolution?



- A 1x1 convolution is nothing but the element wise product of the 32 numbers on the left and the 32 numbers in each filter. With a ReLU nonlinearity applied to the result.
- If we have F no of filters the resultant is of dimensions 6 x 6 x F as indicated in the image

- Coming back to the 1 filter case it can be visualised as a single node taking in the weighted sum of 32 inputs and applying a ReLU nonlinearity to it.
- Thus using F filters is equivalent to a fully connected layer with 32 nodes connected to F nodes in the next layer.
- This idea is also known as ‘Network in Network”which was introduced by researchers in [first inception net paper]

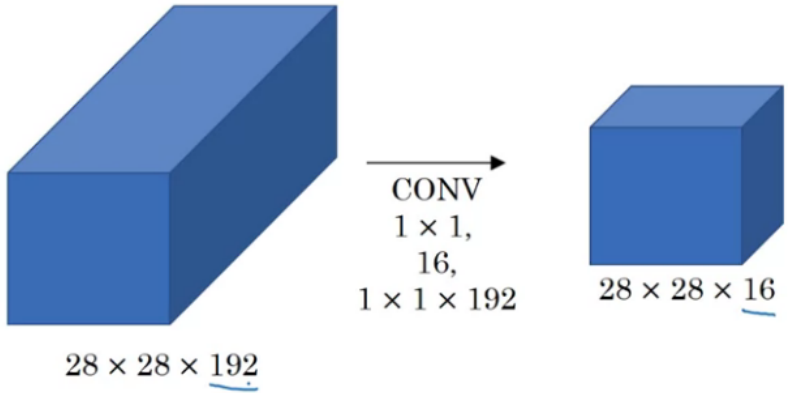


Doing this adds more nonlinearity to the network allowing the network to learn a more complex function.

How is an 1x1 convolution useful?

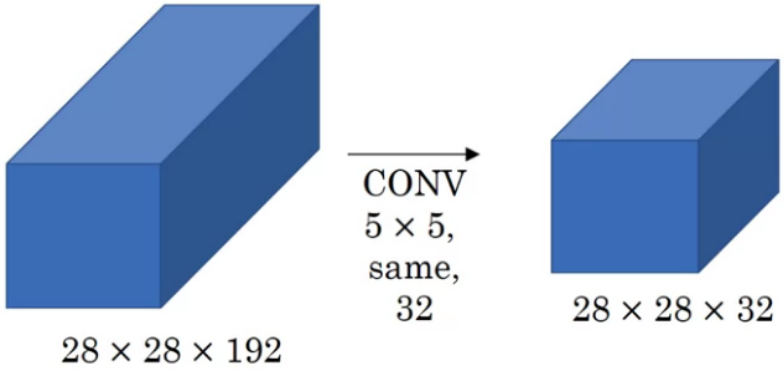
Say , we have a 28 x 28 x 192 dimension volume and we wish to shrink the no of channels to 32 . One way to do this is we could use 32 filters of dimension 1 x 1 x 192 this would output a 28 x 28 x 32 volume.

Thus 1 x 1 convolutions can be used to shrink the no of channels.

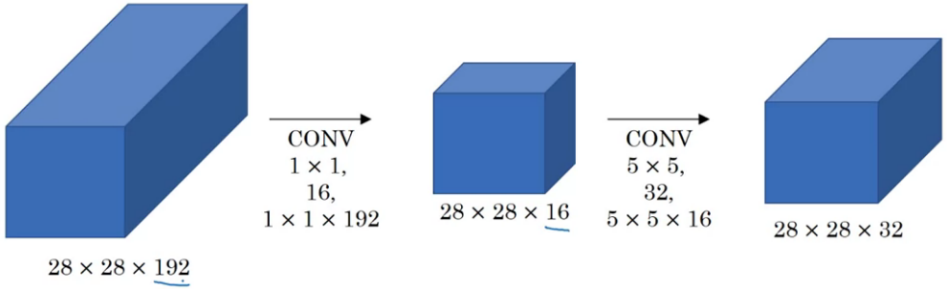


How is an 1x1 convolution useful?

A 1 X 1 convolution can also help us reduce the computational cost



Say we have a 28 x 28 x 192 volume which is passed through a convolution layer of 5 x5 kernels with 32 filters thus the no of total multiplications for this one layer is :
Total Computations = 28 x 28 x 192 x 32 == 120 Million multiplications . Which is a highly expensive operation!



Now if we use 1 x 1 convolutions to reduce the volume to a 28 x 28 x 16 and then use the 5 x 5 convolution, the no of multiplications are :

Total computations =
28 x 28 x 192 x 16 + 28x28x16x32 == 2.8 Million multiplications.

Thus this highly reduces the computational cost

Inception Net v1 or Google LeNet

The researchers behind google's Inception Net or Google LeNet came up with a module which combined various filters at the same level making the network wider rather than deeper. As they observed that objects in images can have very large variation.

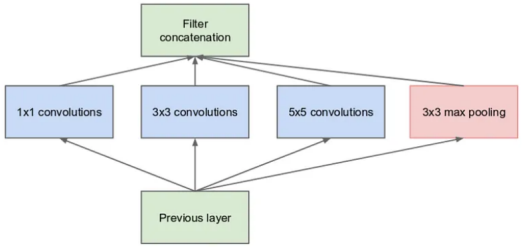
Ex: The three images below

From left: A dog occupying most of the image, a dog occupying a part of it, and a dog occupying very little space (Images obtained from Unsplash).

Because of this huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough. A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally.

What is an Inception Module?

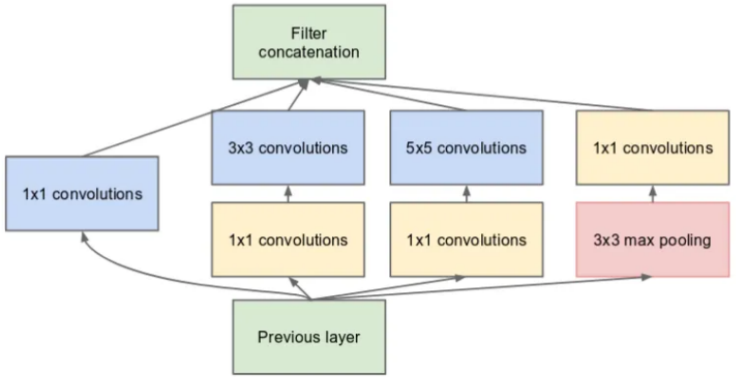
The image is the “naive” inception module. It performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.



(a) Inception module, naïve version

The naive inception module. (Source: Inception v1)

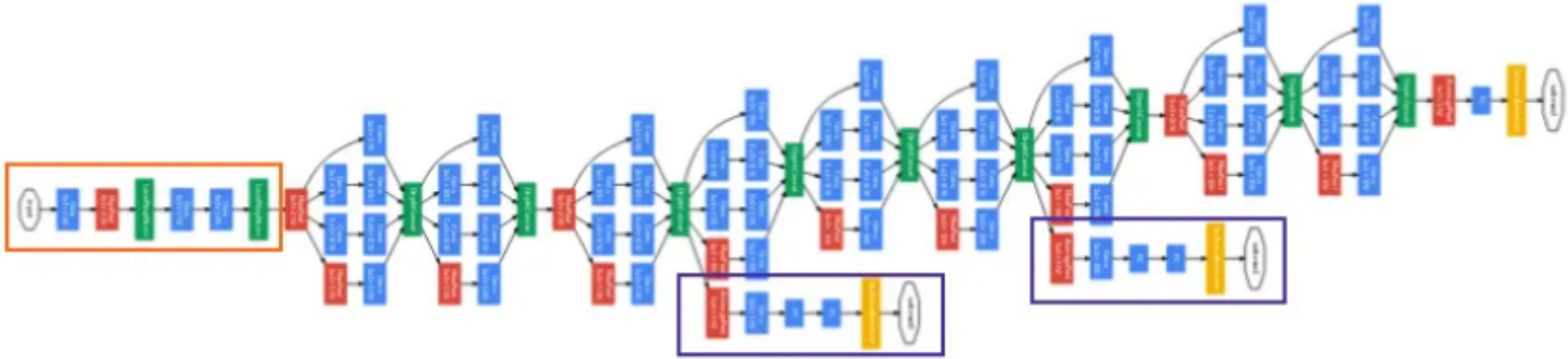
As stated before, deep neural networks are computationally expensive. To make it cheaper, the authors limit the number of input channels by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions.Do note that however, the 1x1 convolution is introduced after the max pooling layer, rather than before.



(b) Inception module with dimension reductions

Inception module with dimension reduction. (Source: Inception v1)

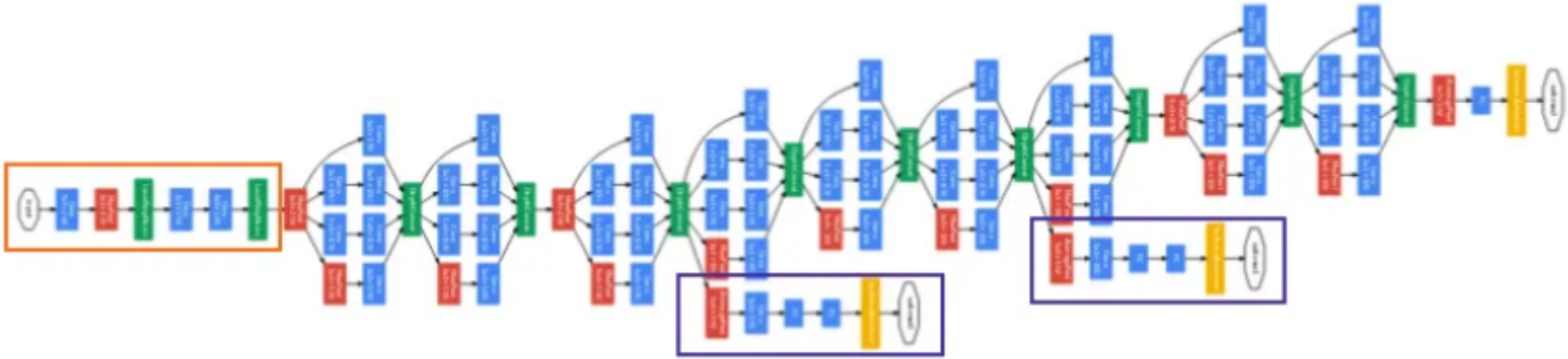
Using the dimension reduced inception module, a neural network architecture was built. This was popularly known as GoogLeNet (Inception v1). The architecture is shown below:



GoogLeNet. The orange box is the **stem**, which has some preliminary convolutions. The purple boxes are **auxiliary classifiers**. The wide parts are the inception modules. (Source: Inception v1)

An Auxiliary classifier are fully connected layers followed by a softmax layer which are used to make predictions using the outputs of middle layers in the network .

This has a regularizing effect on the network is is useful in preventing overfitting.



Inception v2

PREMISE:

- The intuition was that, neural networks perform better when convolutions didn't alter the dimensions of the input drastically. Reducing the dimensions too much may cause loss of information, known as a "representational bottleneck".
- Using smart factorization methods, convolutions can be made more efficient in terms of computational complexity.

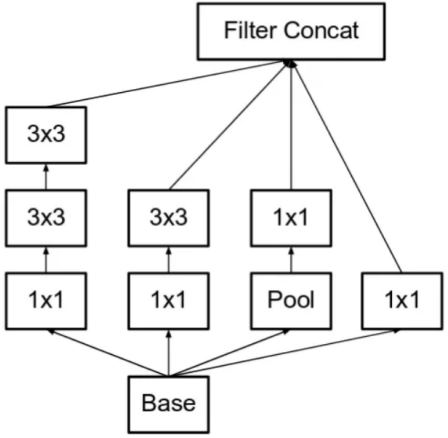
SOLUTION:

A. Factorize 5x5 convolution to two 3x3 convolution operations to improve computational speed. A 5x5 convolution is 2.78 times more expensive than a 3x3 convolution. So stacking two 3x3 convolutions in fact leads to a boost in performance.

B. Moreover, they factorize convolutions of filter size nxn to a combination of 1xn and nx1 convolutions. For example, a 3x3 convolution is equivalent to first performing a 1x3 convolution, and then performing a 3x1 convolution on its output. They found this method to be 33% more cheaper than the single 3x3 convolution.

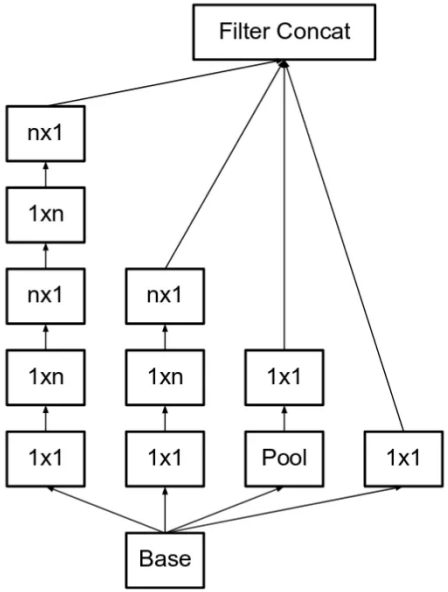
C. The filter banks in the module were expanded (made wider instead of deeper) to remove the representational bottleneck.

Inception v2



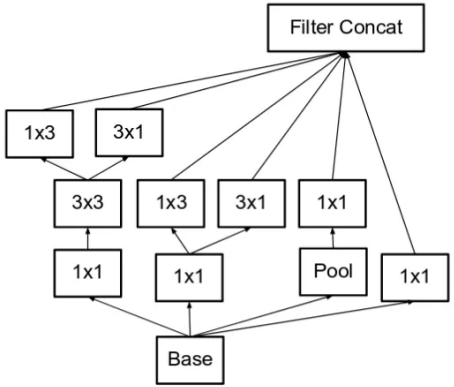
The left-most 5×5 convolution of the old inception module, is now represented as two 3×3 convolutions. (Source: Incpetion v2)

A



Here, put **n=3** to obtain the equivalent of the previous image. The left-most 5×5 convolution can be represented as two 3×3 convolutions, which inturn are represented as 1×3 and 3×1 in series. (Source: Incpetion v2)

B



Making the inception module wider. This type is equivalent to the module shown above. (Source: Incpetion v2)

C

Inception v3

PREMISE:

- The authors noted that the auxiliary classifiers didn't contribute much until near the end of the training process, when accuracies were nearing saturation. They argued that they function as regularizes, especially if they have BatchNorm or Dropout operations.
- Possibilities to improve on the Inception v2 without drastically changing the modules were to be investigated.

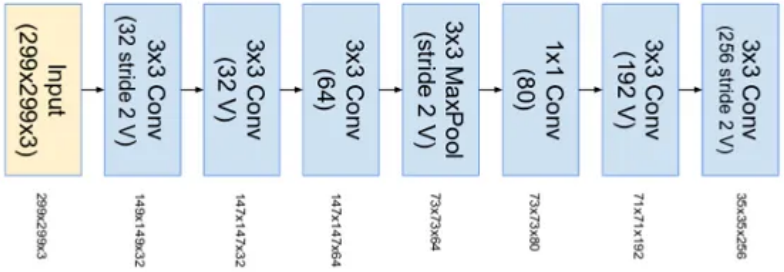
SOLUTION: Inception Net v3 incorporated all of the above upgrades stated for Inception v2, and in addition used the following:

1. RMSProp Optimizer.
2. Factorized 7x7 convolutions.
3. BatchNorm in the Auxiliary Classifiers.
4. Label Smoothing (A type of regularizing component added to the loss formula that prevents the network from becoming too confident about a class. Prevents overfitting).

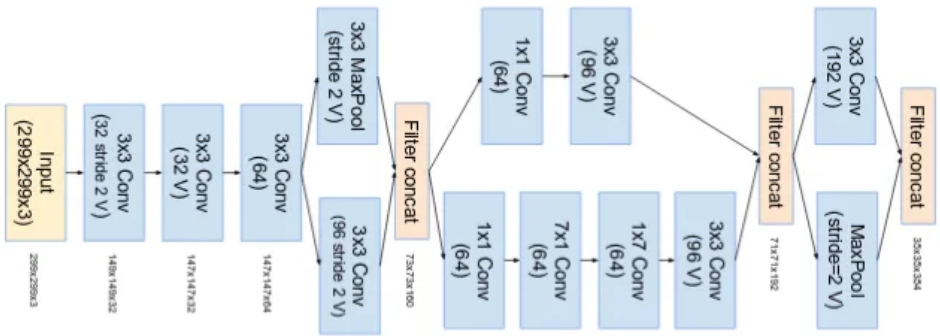
Inception v4

PREMISE: Make the modules more uniform. The authors also noticed that some of the modules were more complicated than necessary. This can enable us to boost performance by adding more of these uniform modules.

SOLUTION: The “stem” of Inception v4 was modified. The stem here, refers to the initial set of operations performed before introducing the Inception blocks.



Stem in v1



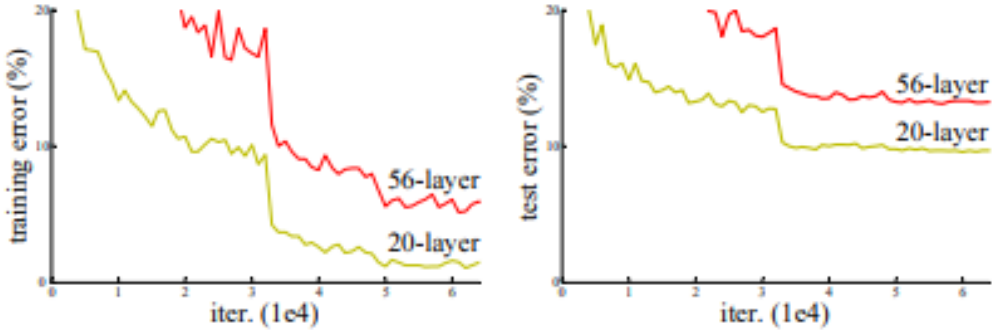
Stem in v4

ResNet

After the AlexNet model was introduced future researchers started building bigger CNNs for increased accuracy.Thus the question arises that ;

Is learning better networks is driven by depth in layers?

However , when deeper networks are able to start converging, a degradation problem was exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is not caused by overfitting, and adding more layers to a suitably deep model leads to higher training error. Thus in the following example a 59 layer network has much higher test error than the 20 layer network.



Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error.

ResNets

- Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution by construction to the deeper model: the added layers are identity mapping, and the other layers are copied from the learned shallower model.
- The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart.
- But experiments show that our current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution

Thus here x is the required deviation which the model must learn and if there is nothing to be learnt then F(x) +x is the identity function.

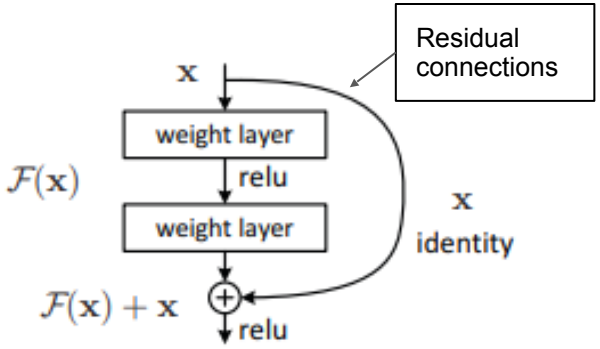
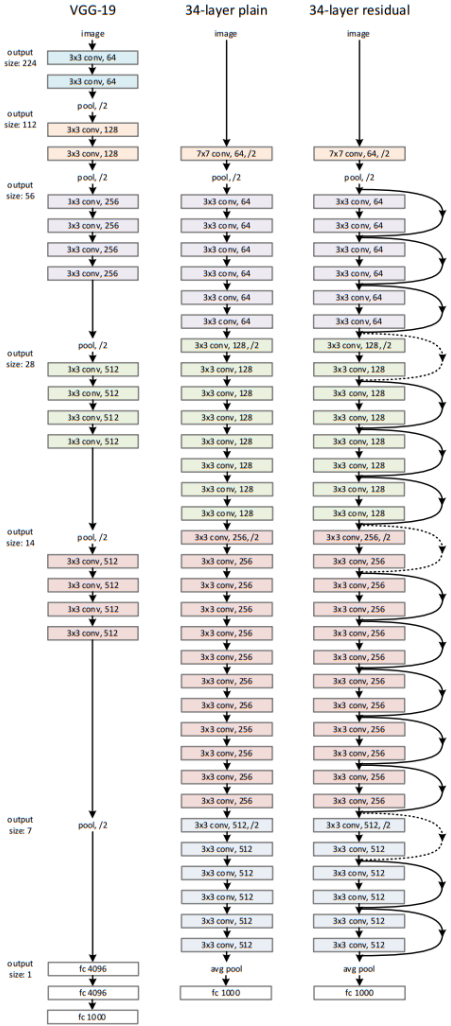


Figure 2. Residual learning: a building block.

Thus to have a NN by default learn the identity function rather than pass the data through more weight layers , the authors proposed adding skip connections in the network.



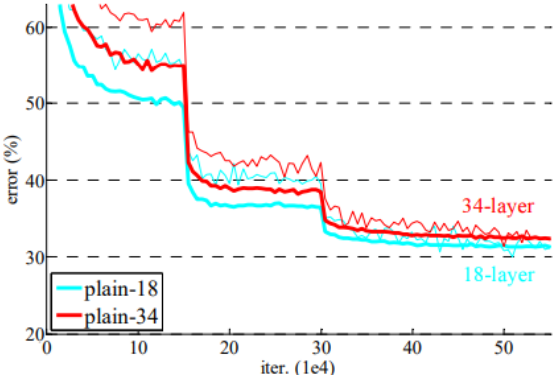
Thus we compare the following network architectures for ImageNet.

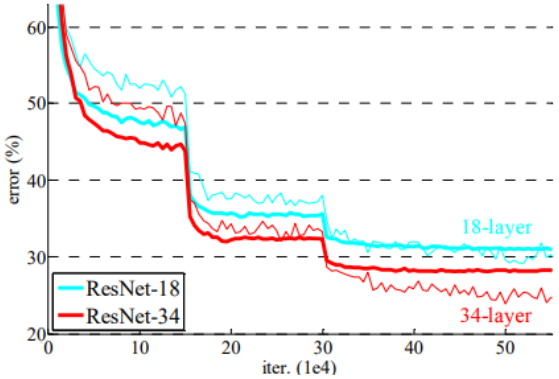
Left: the VGG-19 model [41] (19.6 billion FLOPs) as a reference.

Middle: a plain network with 34 parameter layers (3.6 billion FLOPs).

Right: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions

And the results ….





Here we see that training and validation error are higher in the larger network without residual connections.

However we see that in a network with residual connections we can train a 34 layer network with a much lower error.

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

Using residual connections helps deeper networks as the error is much lesser for the 34 layer network than the 18 layer network.

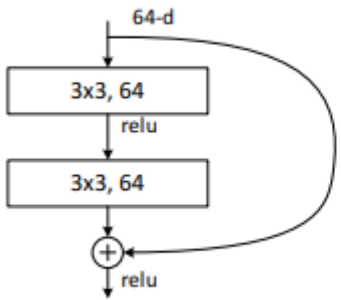
ResNets

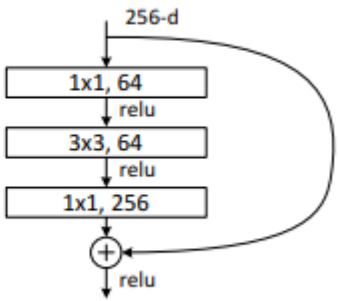
The authors thus built a family of models as the following :

- ResNet-50: With 50 layers
- ResNet-101 : With 101 layers
- ResNet-152 : With 152 layers

At the time the ResNet 152 model performed the best on the ImageNet dataset and they were declared the winners of Image Net 2015. Proving that deeper is sometimes indeed better!

They also introduced the idea of bottleneck for deeper networks i.e for each residual function F, we use a stack of 3 layers instead of 2 (Fig). The three layers are 1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions.





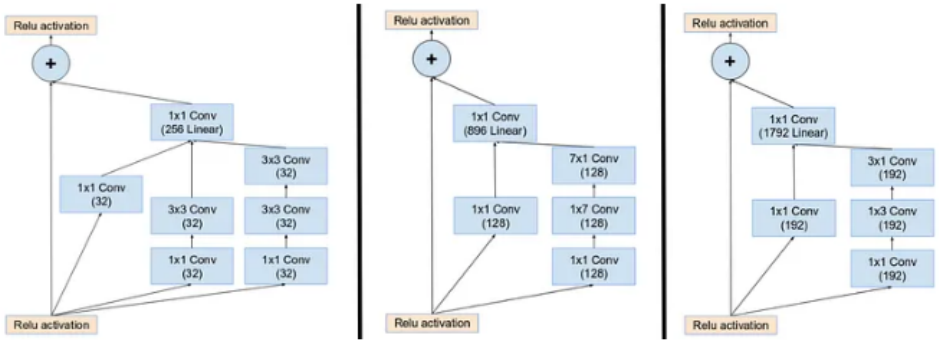
As deeper networks are computationally expensive , so to save computational power they project 256 down to 64 and then the complexity of the second layer is similar to that of a 64 dimension and then project upwards again.

Inception-ResNet

Inspired by the performance of the ResNet, a hybrid inception module was proposed. There are two sub-versions of Inception ResNet, namely v1 and v2. Before we checkout the salient features, let us look at the minor differences between these two sub-versions.

- Inception-ResNet v1 has a computational cost that is similar to that of Inception v3.
- Inception-ResNet v2 has a computational cost that is similar to that of Inception v4.
- They have different stems, as illustrated in the Inception v4 section.

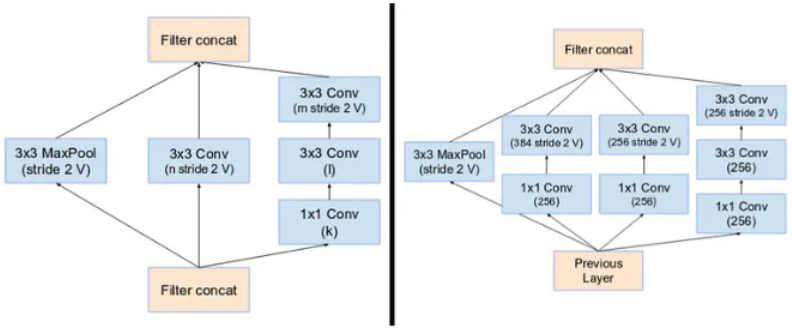
For residual addition to work, the input and output after convolution must have the same dimensions. Hence, we use 1x1 convolutions after the original convolutions, to match the depth sizes



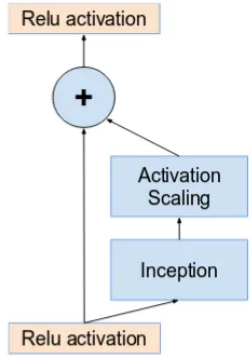
(From left) Inception modules A,B,C in an Inception ResNet. Note how the pooling layer was replaced by the residual connection, and also the additional 1×1 convolution before addition. (Source: Inception v4)

Inception-ResNet

The pooling operation inside the main inception modules were replaced in favor of the residual connections. However, you can still find those operations in the reduction blocks. Reduction block A is same as that of Inception v4.



(From Left) Reduction Block A (35×35 to 17×17 size reduction) and Reduction Block B (17×17 to 8×8 size reduction). Refer to the paper for the exact hyper-parameter setting (V,l,k). (Source: Inception v4)



Activations are scaled by a constant to prevent the network from dying. (Source: Inception v4)

Networks with residual units deeper in the architecture caused the network to “die” if the number of filters exceeded 1000. Hence, to increase stability, the authors scaled the residual activations by a value around 0.1 to 0.3.

Efficient Net(B0-B7)

The process of scaling up ConvNets has never been well understood , however there are many ways to do it . The most common of the ways is to scale up one of the following parameters width , depth or resolution. But scaling up arbitrarily requires tedious manual tuning and often leads to suboptimal results.

The key contributions of the Efficient Net paper is to show that it is critical to balance width , depth and image resolution . They prove that such balance can simply be achieved scaling each factor by a common ratio.This method is called Compound scaling.

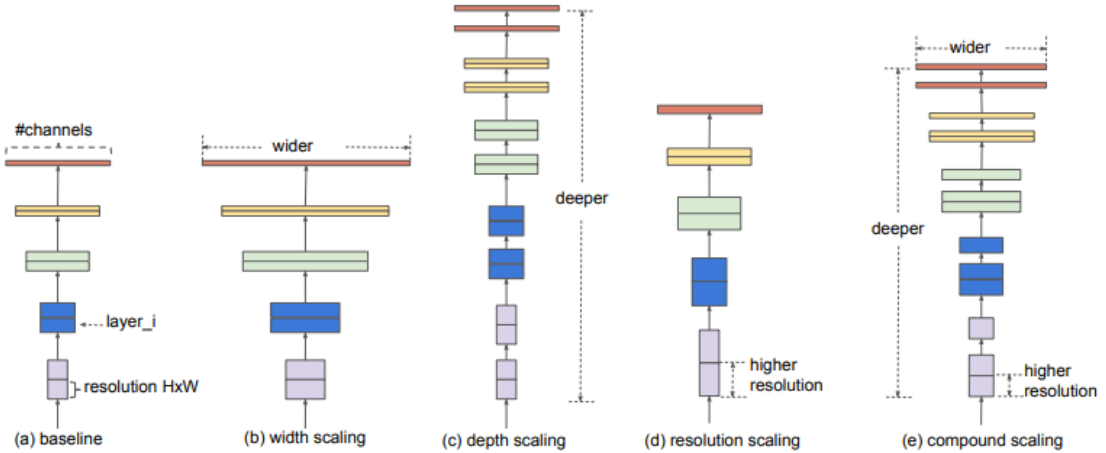


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

They demonstrate this using the MobileNet and ResNet architectures. They also develop a baseline network and scale it up to obtain the family of Efficient Net models (B0-B7)

Compound Scaling

The Compound scaling method, which uses a compound coefficient φ to uniformly scales network width, depth, and resolution in a principled way:

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

where α, β, γ are constants that can be determined by a small grid search. Intuitively, φ is a user-specified coefficient that controls how many more resources are available for model scaling, while α, β, γ specify how to assign these extra resources to network width, depth, and resolution respectively.

Efficient Net B0 (The baseline network)

At the core of Efficient Net, the mobile inverted bottleneck convolution layer is used along with the squeeze and excitation layers. The following table shows the architecture of Efficient B0:

STAGE i	OPERATOR F_i	RESOLUTION H_i X W_i	# CHANNELS C_i	# LAYERS L_i
1	Conv 3×3	224 x 224	32	1
2	MBConv1, k-3×3	112 x 112	16	1
3	MBConv6, k-3×3	112 x 112	24	2
4	MBConv6, k-5×5	56 x 56	40	2
5	MBConv6, k-3×3	28 x 28	80	3
6	MBConv6, k-5×5	14 x 14	112	3
7	MBConv6, k-5×5	14 x 14	192	4
8	MBConv6, k-3×3	7 x 7	320	1
9	Conv 1×1 & Pooling & FC	7 x 7	1280	1

The creators of Efficient Net started to scale Efficient Net B0 with the help of their compound scaling method. They applied the grid search technique to get :

$\alpha = 1.2, \beta = 1.1, \gamma = 1.15, \Phi = 1.$

Afterward, they fixed the scaling coefficients and scaled Efficient Net B0 to Efficient Net B7.

Conclusion

This lecture aims to summarize the most popular CNN architectures , however there are a plethora of other CNN architectures which the students may wish to explore . A comprehensive list is provided in the research papers listed in reference [9] and [10].

Reading the research papers cited in the references section is also encouraged.

References

[1] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, “Phoneme recognition using time-delay neural networks,” IEEE Trans. Acoust., Speech, Signal Process., vol. 37, no. 3, pp. 328–339, Mar. 1989.
[2] W. Zhang, “Shift-invariant pattern recognition neural network and its optical architecture,” in Proc. Annu. Conf. Jpn. Soc. Appl. Phys., 1988, p. 734, paper 6P-M-14.
[3] Y. LeCun et al., “Backpropagation applied to handwritten zip code recognition,” Neural Comput., vol. 1, no. 4, pp. 541–551, Dec. 1989
[4]Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
[5] M. D. Zeiler and R. Fergus, ‘Visualizing and Understanding Convolutional Networks’, *arXiv [cs.CV]*. 2013.
[6]Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
[7]He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
[8]https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
[9]A. Dhillon and G. K. Verma, “Convolutional neural network: A review of models, methodologies and applications to object detection,” Prog. Artif. Intell., vol. 9, no. 2, pp. 85–112, Jun. 2020.
[10]Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999-7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.
[11]https://youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&si=yKZ9uniVHpq2EzMq
[12]Tan, M., & Le, Q. v. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*.

UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Anashua Krittika Dastidar
Teaching Assistant