Lec11 - Convolutional Neural Networks

CS 783 - Visual Recognition

Vinay P. Namboodiri

IIT Kanpur

12^{th} February 2019

Contents

1. Recap
   - Architecture
   - Convolutional Layer
   - Pooling Layer
   - Contrast Normalisation Layer
2. Architecture Variants
   - AlexNet
   - VGG
   - GoogleNet
   - ResNet
3. Initialisation
   - Xavier
4. Normalisation
   - Batch Normalisation

Outline

1. Recap
   - Architecture
   - Convolutional Layer
   - Pooling Layer
   - Contrast Normalisation Layer
2. Architecture Variants
3. Initialisation
4. Normalisation

Convolutional network

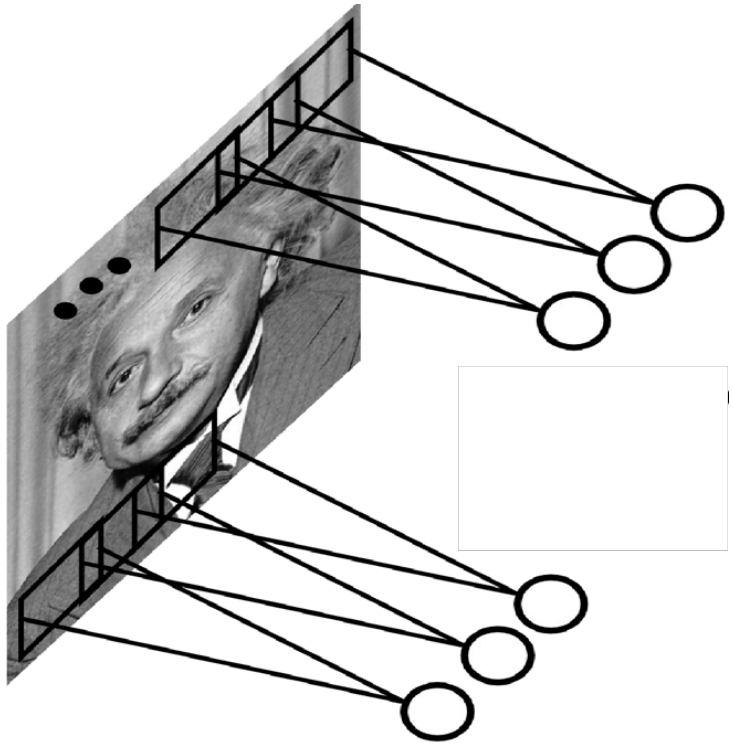
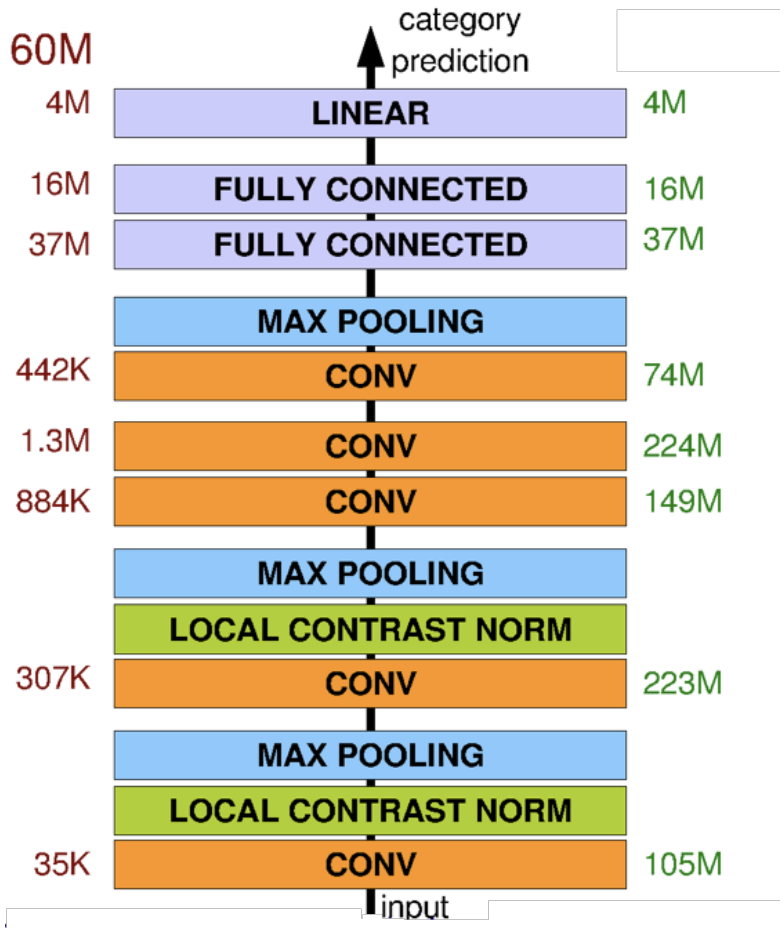


Figure: A convolutional network sharing parameters (figure by Marc Aurelio Ranzato)

- A convolutional network shares the parameter set across different locations.

Architecture of a Convolutional network



- A convolutional network comprises of three kinds of layers that are combined in a network.

Figure: Alexnet CNN architecture, fig. credit. Ranzato

Architecture of a Convolutional network

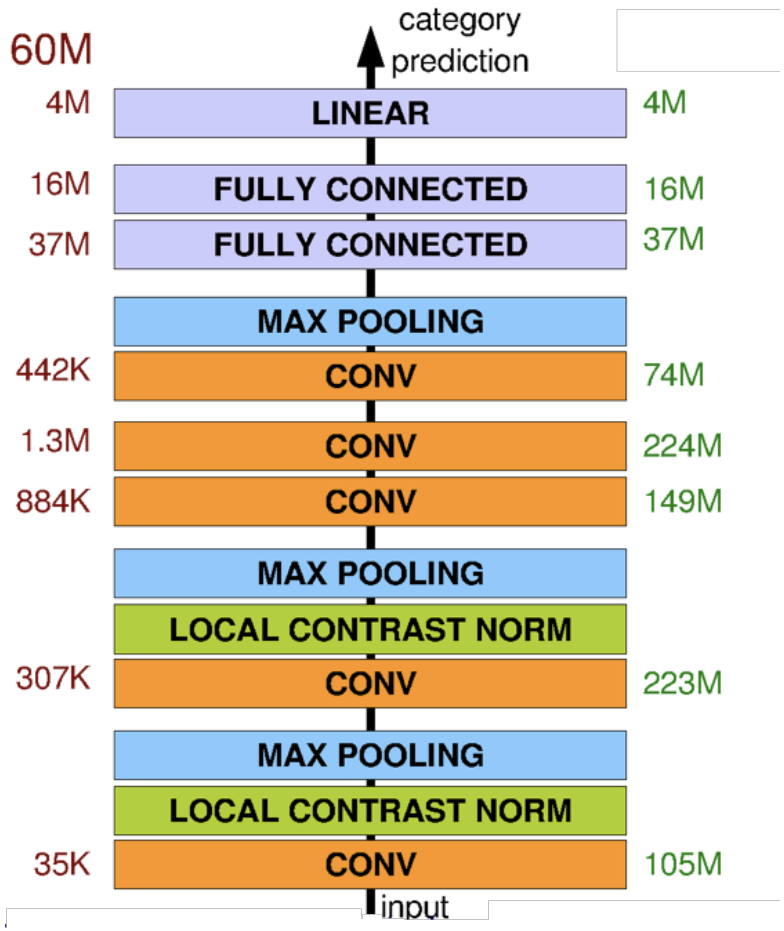


Figure: Alexnet CNN architecture, fig. credit. Ranzato

- A convolutional network comprises of three kinds of layers that are combined in a network.
  - Convolutional Layer

Architecture of a Convolutional network

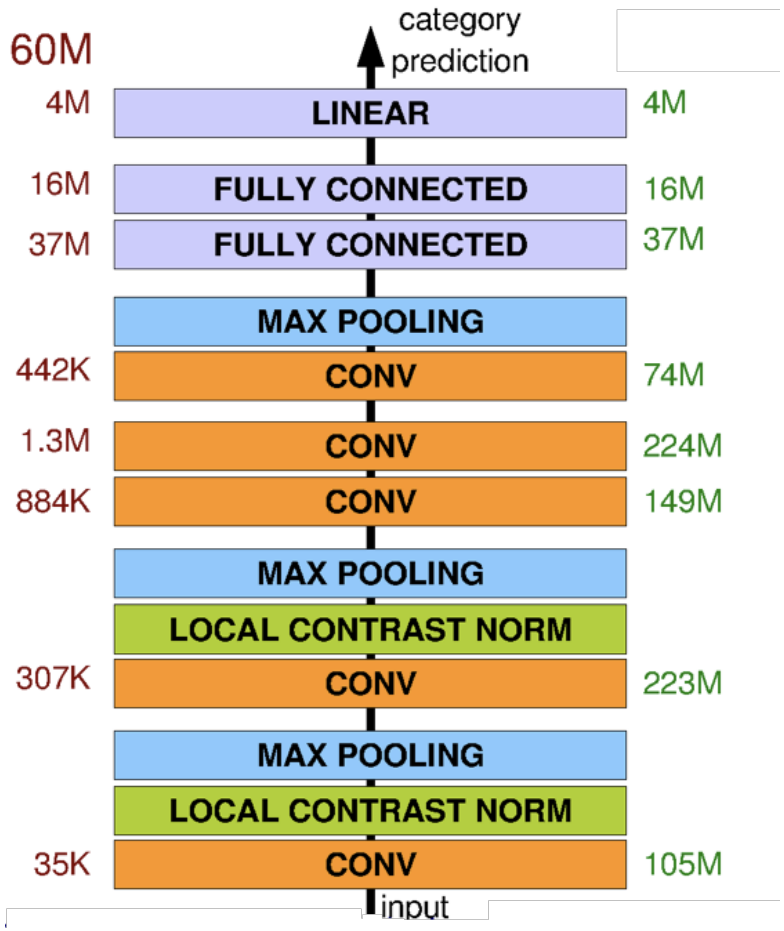


Figure: Alexnet CNN architecture, fig. credit. Ranzato

- A convolutional network comprises of three kinds of layers that are combined in a network.
  - Convolutional Layer
  - Pooling Layer

Architecture of a Convolutional network

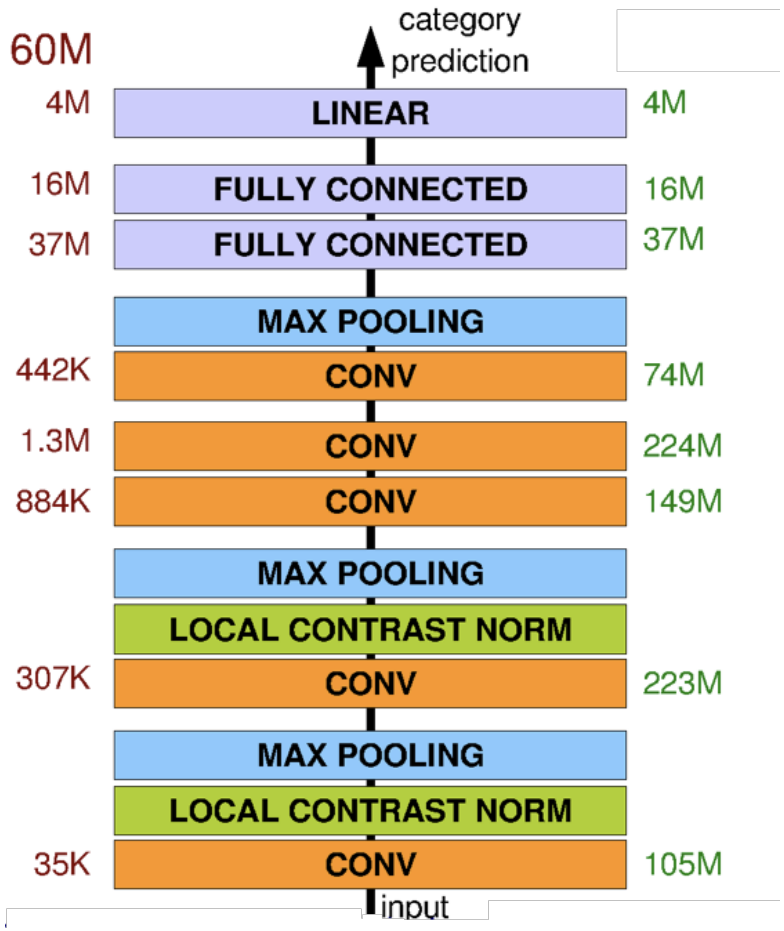


Figure: Alexnet CNN architecture, fig. credit. Ranzato

- A convolutional network comprises of three kinds of layers that are combined in a network.
  - Convolutional Layer
  - Pooling Layer
  - Normalization Layer

Architecture of a Convolutional network

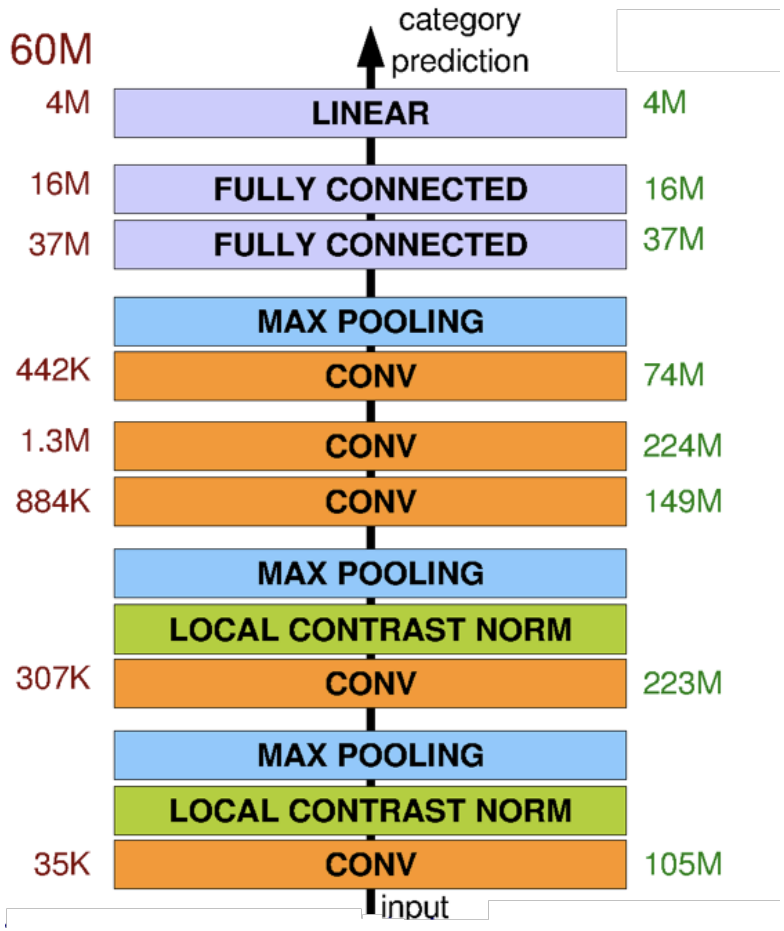


Figure: Alexnet CNN architecture, fig. credit. Ranzato

- A convolutional network comprises of three kinds of layers that are combined in a network.
  - Convolutional Layer
  - Pooling Layer
  - Normalization Layer
- We consider the role of each of these layers in detail

Convolutional Layer

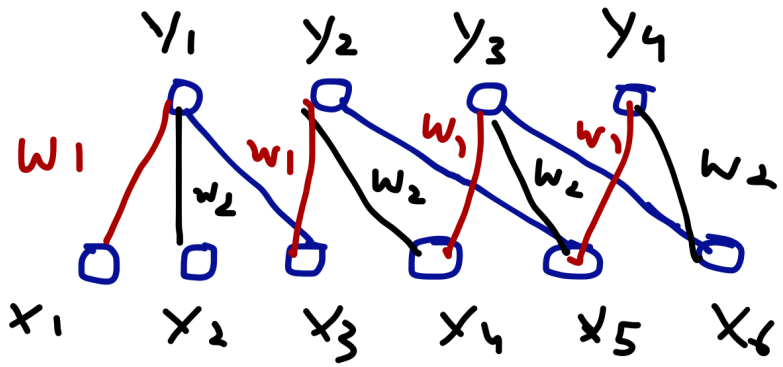


Figure: Illustration of a Convolution layer

- At its simplest form, a convolutional layer comprises a matrix that determines the weights that are used to obtain the output response given the input activations.

Convolutional Layer

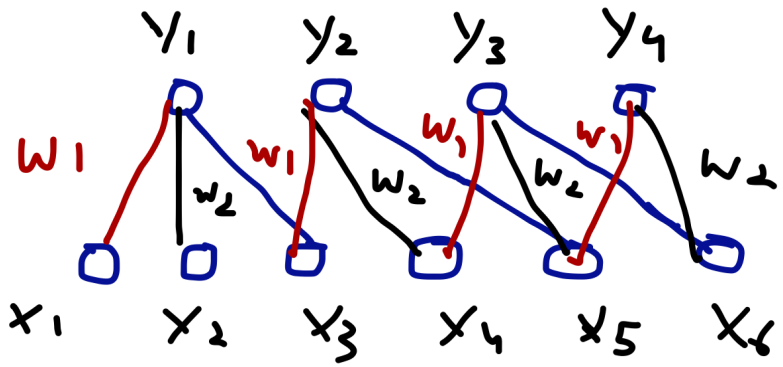


Figure: Illustration of a Convolution layer

- At its simplest form, a convolutional layer comprises a matrix that determines the weights that are used to obtain the output response given the input activations.
- These weights are learned by the network. The figure illustrates that the weights w_1, w_2, w_3 are shared between the input layer $X_1, X_2, \cdot$ and the output layer $Y_1, Y_2, \cdot$

Convolutional Layer

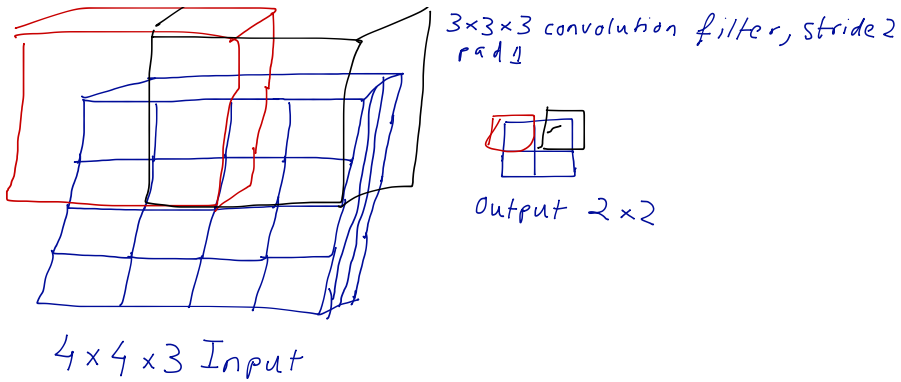


Figure: Illustration of input-output filters. Check out also the interactive demo at http://jsfiddle.net/yces4vn9/43/

- Each convolution layer converts an input 3D matrix to an output 3D matrix.

Convolutional Layer

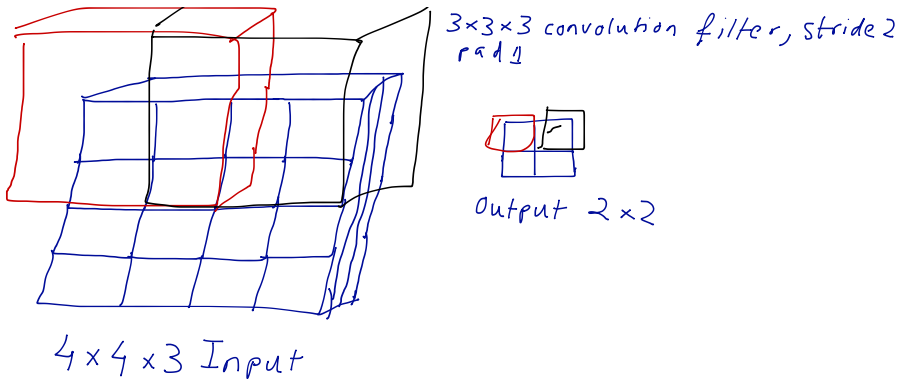


Figure: Illustration of input-output filters. Check out also the interactive demo at http://jsfiddle.net/yces4vn9/43/

- Each convolution layer converts an input 3D matrix to an output 3D matrix.
- The parameters of the layer are width w and height h of the kernel, the input channel depth d_i, the output channel depth d_o (that is obtained through d_o filters), the stride s, zero padding p and the dilation d for the filter.

Pooling Layer

The pooling layer can be either max-pooling or average pooling and are obtained by the following equation

Max-pooling:

$$h_j^n(x, y) = max_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})$$

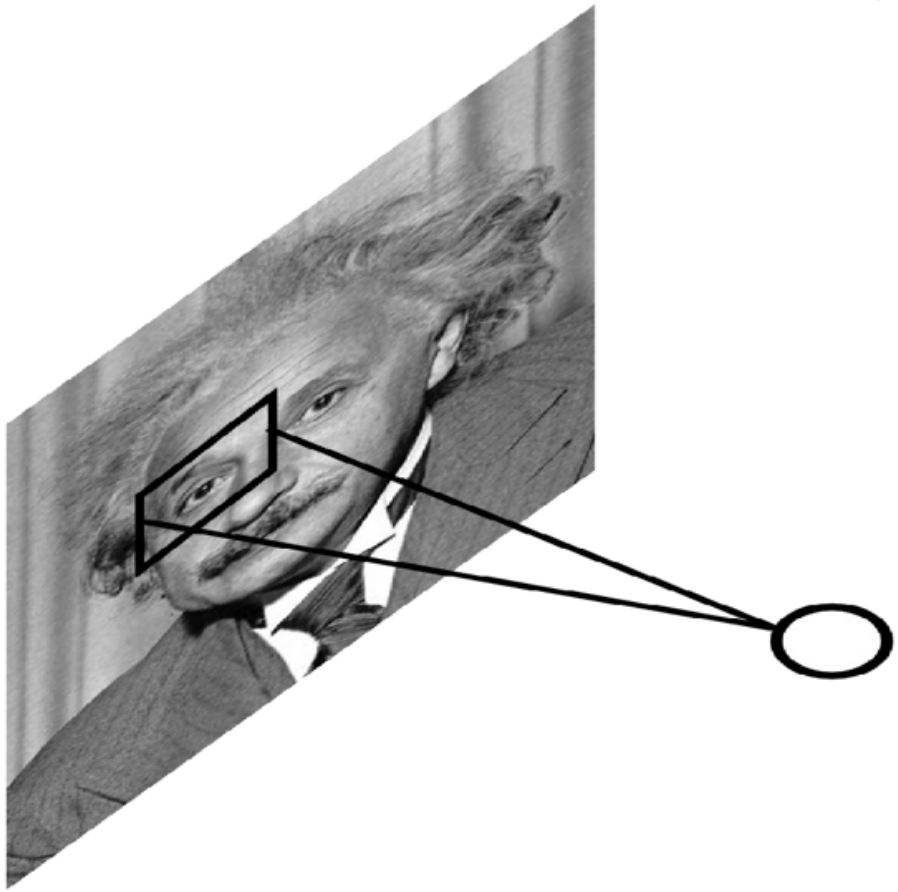
Average-pooling:

$$h_j^n(x, y) = 1/K \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})$$

If the convolutional filters have size KxK and stride 1 and pooling layers have pools of size PxP, then each unit in the pooled layer depends upon a patch of size (P+K-1)x(P+K-1). The effect can be visualized through the interactive demo available at http://jsfiddle.net/yces4vn9/43/

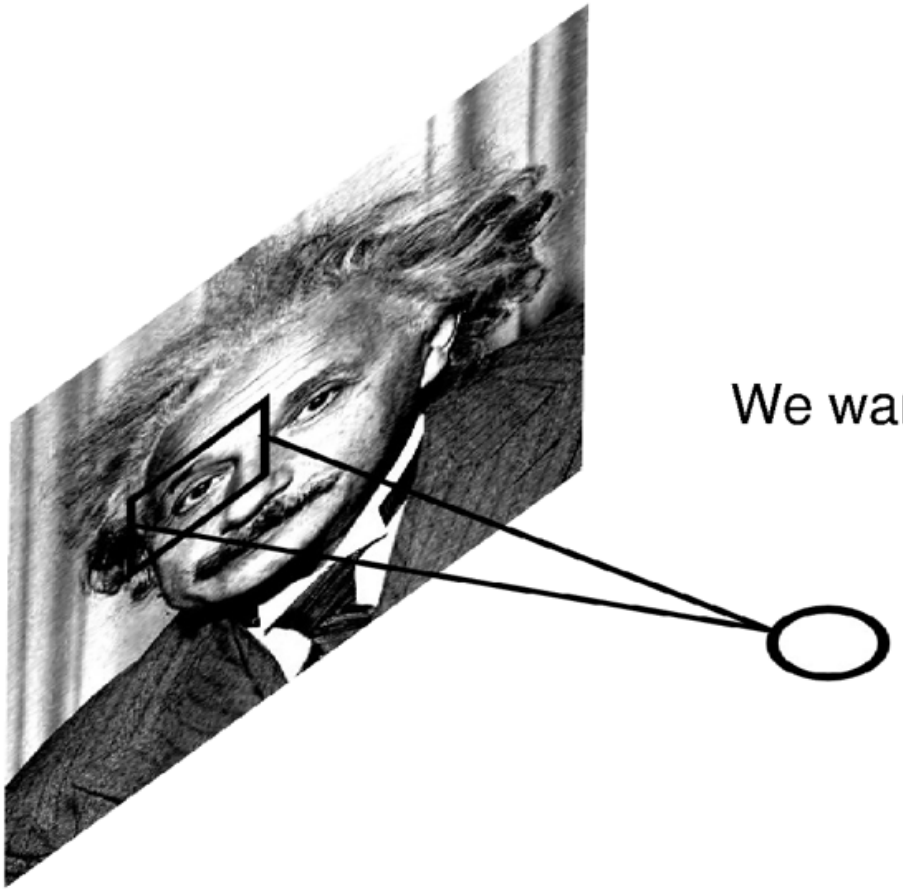
Contrast Normalisation Layer

$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{\sigma^i(N(x,y))}$$



Contrast Normalisation Layer

$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{\sigma^i(N(x,y))}$$



We want the same response.

Outline

1 Recap

2 Architecture Variants
- AlexNet
- VGG
- GoogleNet
- ResNet

3 Initialisation

4 Normalisation

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96
  11x11 filters at stride 4, pad
  0

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96
  11x11 filters at stride 4, pad
  0
- (27x27x96) MAX POOL1:
  3x3 filters at stride 2

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96
  11x11 filters at stride 4, pad
  0
- (27x27x96) MAX POOL1:
  3x3 filters at stride 2
- (27x27x96) NORM1:
  Normalization layer

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96
  11x11 filters at stride 4, pad
  0
- (27x27x96) MAX POOL1:
  3x3 filters at stride 2
- (27x27x96) NORM1:
  Normalization layer
- (27x27x256) CONV2: 256
  5x5 filters at stride 1, pad 2

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96
  11x11 filters at stride 4, pad
  0
- (27x27x96) MAX POOL1:
  3x3 filters at stride 2
- (27x27x96) NORM1:
  Normalization layer
- (27x27x256) CONV2: 256
  5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2:
  3x3 filters at stride 2

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96
  11x11 filters at stride 4, pad
  0
- (27x27x96) MAX POOL1:
  3x3 filters at stride 2
- (27x27x96) NORM1:
  Normalization layer
- (27x27x256) CONV2: 256
  5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2:
  3x3 filters at stride 2
- (13x13x256)] NORM2:

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96
  11x11 filters at stride 4, pad 0
- (27x27x96) MAX POOL1:
  3x3 filters at stride 2
- (27x27x96) NORM1:
  Normalization layer
- (27x27x256) CONV2: 256
  5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2:
  3x3 filters at stride 2
- (13x13x256)] NORM2:
  Normalization layer
- (13x13x384) CONV3: 384
  3x3 filters at stride 1, pad 1

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96 11x11 filters at stride 4, pad 0
- (27x27x96) MAX POOL1: 3x3 filters at stride 2
- (27x27x96) NORM1: Normalization layer
- (27x27x256) CONV2: 256 5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2: 3x3 filters at stride 2
- (13x13x256)] NORM2:
- (13x13x384) CONV3: 384 3x3 filters at stride 1, pad 1
- (13x13x384) CONV4: 384 3x3 filters at stride 1, pad 1

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96 11x11 filters at stride 4, pad 0
- (27x27x96) MAX POOL1: 3x3 filters at stride 2
- (27x27x96) NORM1: Normalization layer
- (27x27x256) CONV2: 256 5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2: 3x3 filters at stride 2
- (13x13x256)] NORM2:
- (13x13x384) CONV3: 384 3x3 filters at stride 1, pad 1
- (13x13x384) CONV4: 384 3x3 filters at stride 1, pad 1
- (13x13x256) CONV5: 256 3x3 filters at stride 1, pad 1

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96 11x11 filters at stride 4, pad 0
- (27x27x96) MAX POOL1: 3x3 filters at stride 2
- (27x27x96) NORM1: Normalization layer
- (27x27x256) CONV2: 256 5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2: 3x3 filters at stride 2
- (13x13x256)] NORM2:
- (13x13x384) CONV3: 384 3x3 filters at stride 1, pad 1
- (13x13x384) CONV4: 384 3x3 filters at stride 1, pad 1
- (13x13x256) CONV5: 256 3x3 filters at stride 1, pad 1
- (6x6x256) MAX POOL3: 3x3 filters at stride 2

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96 11x11 filters at stride 4, pad 0
- (27x27x96) MAX POOL1: 3x3 filters at stride 2
- (27x27x96) NORM1: Normalization layer
- (27x27x256) CONV2: 256 5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2: 3x3 filters at stride 2
- (13x13x256)] NORM2:
- (13x13x384) CONV3: 384 3x3 filters at stride 1, pad 1
- (13x13x384) CONV4: 384 3x3 filters at stride 1, pad 1
- (13x13x256) CONV5: 256 3x3 filters at stride 1, pad 1
- (6x6x256) MAX POOL3: 3x3 filters at stride 2
- (4096) FC6: 4096 neurons

Alex Net Architecture

The Alex Net Architecture in detail is as follows

- (227x227x3) INPUT
- (55x55x96) CONV1: 96 11x11 filters at stride 4, pad 0
- (27x27x96) MAX POOL1: 3x3 filters at stride 2
- (27x27x96) NORM1: Normalization layer
- (27x27x256) CONV2: 256 5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2: 3x3 filters at stride 2
- (13x13x256)] NORM2:
- (13x13x384) CONV3: 384 3x3 filters at stride 1, pad 1
- (13x13x384) CONV4: 384 3x3 filters at stride 1, pad 1
- (13x13x256) CONV5: 256 3x3 filters at stride 1, pad 1
- (6x6x256) MAX POOL3: 3x3 filters at stride 2
- (4096) FC6: 4096 neurons
- (4096) FC7: 4096 neurons

Alex Net Architecture

The Alex Net Architecture in detail is as follows

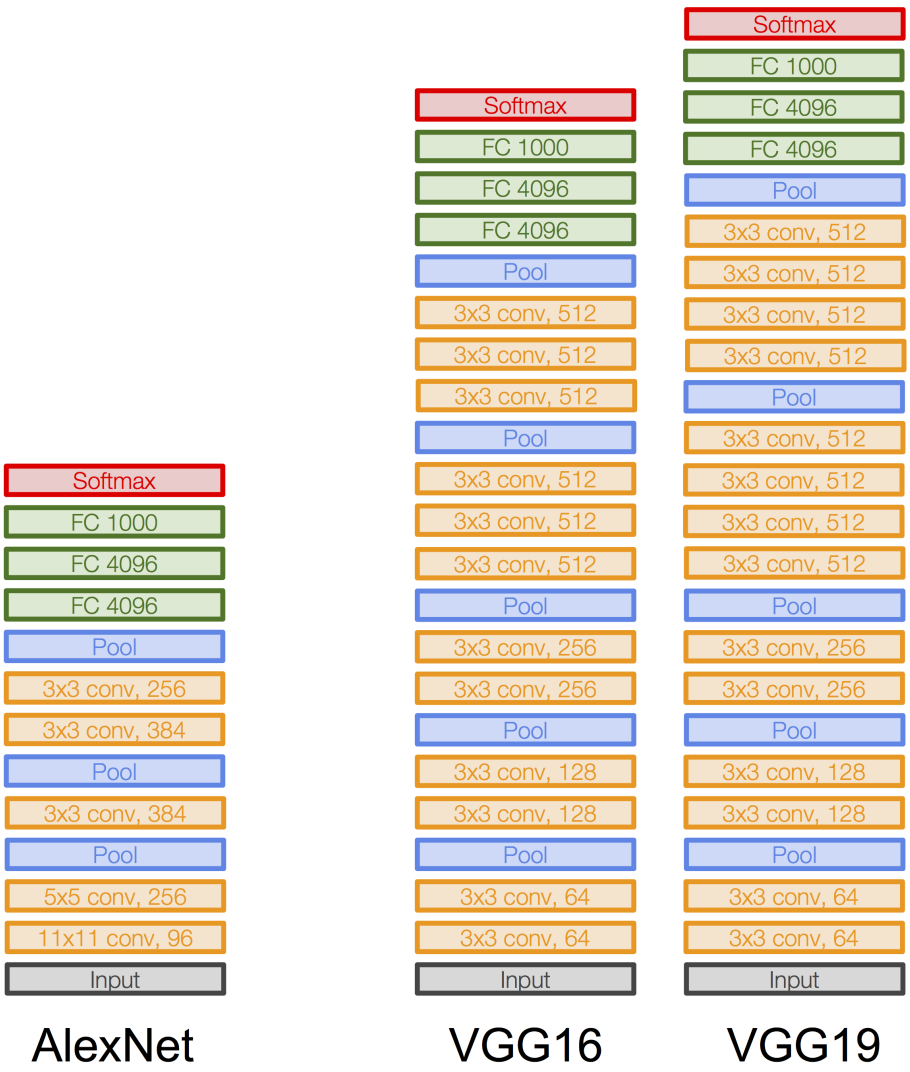
- (227x227x3) INPUT
- (55x55x96) CONV1: 96 11x11 filters at stride 4, pad 0
- (27x27x96) MAX POOL1: 3x3 filters at stride 2
- (27x27x96) NORM1: Normalization layer
- (27x27x256) CONV2: 256 5x5 filters at stride 1, pad 2
- (13x13x256) MAX POOL2: 3x3 filters at stride 2
- (13x13x256)] NORM2:
- (13x13x384) CONV3: 384 3x3 filters at stride 1, pad 1
- (13x13x384) CONV4: 384 3x3 filters at stride 1, pad 1
- (13x13x256) CONV5: 256 3x3 filters at stride 1, pad 1
- (6x6x256) MAX POOL3: 3x3 filters at stride 2
- (4096) FC6: 4096 neurons
- (4096) FC7: 4096 neurons
- (1000) FC8: 1000 neurons (class scores)

Other Architectures

There has been since Alexnet a number of architectures with increased depth

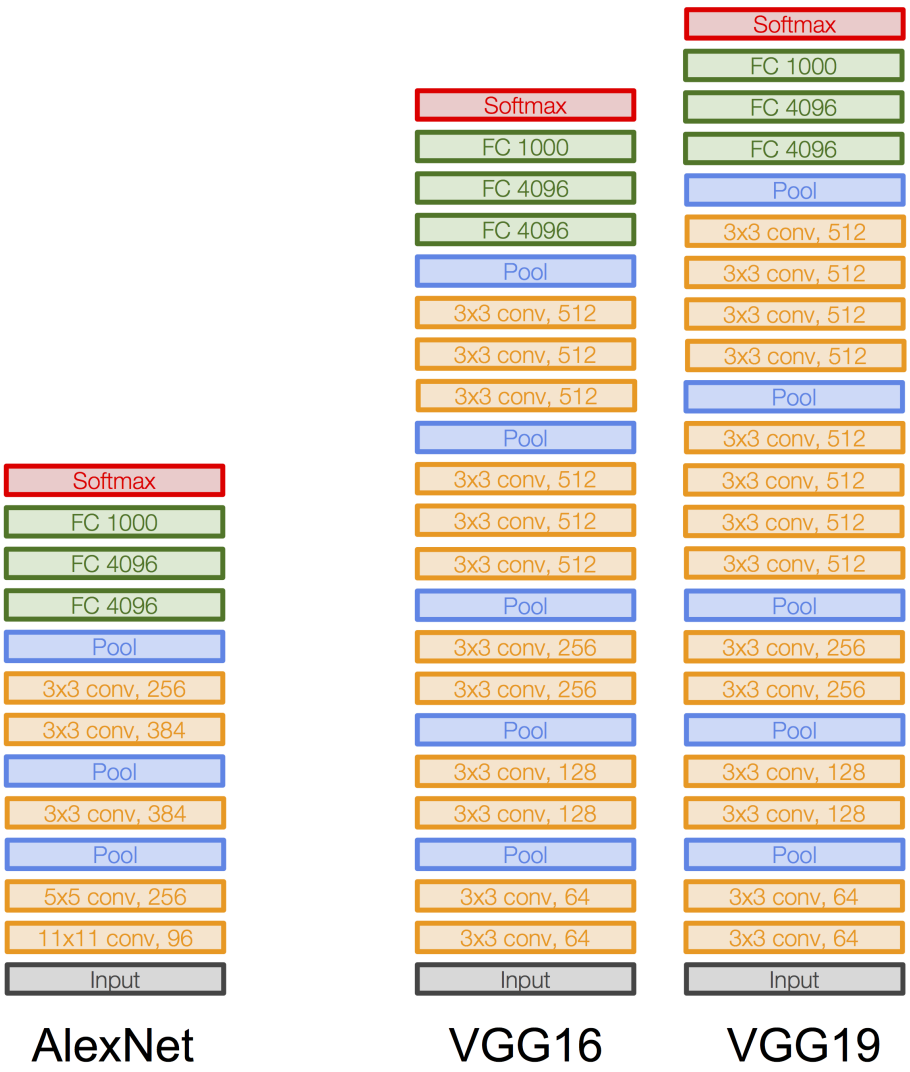
ILSVRC Year	Architecture	Depth	Top 5% Error
ILSVRC10	Shallow	-	28.2
ILSVRC11	Shallow	-	25.8
ILSVRC12	AlexNet	8 layers	16.4
ILSVRC 13	- Clarifai	8 layers	11.7
ILSVRC14	VGG	19 layers	7.3
ILSVRC14	GoogleNet	22 layers	6.7
ILSVRC15	ResNet	152 layers	3.57

VGG Deep Net



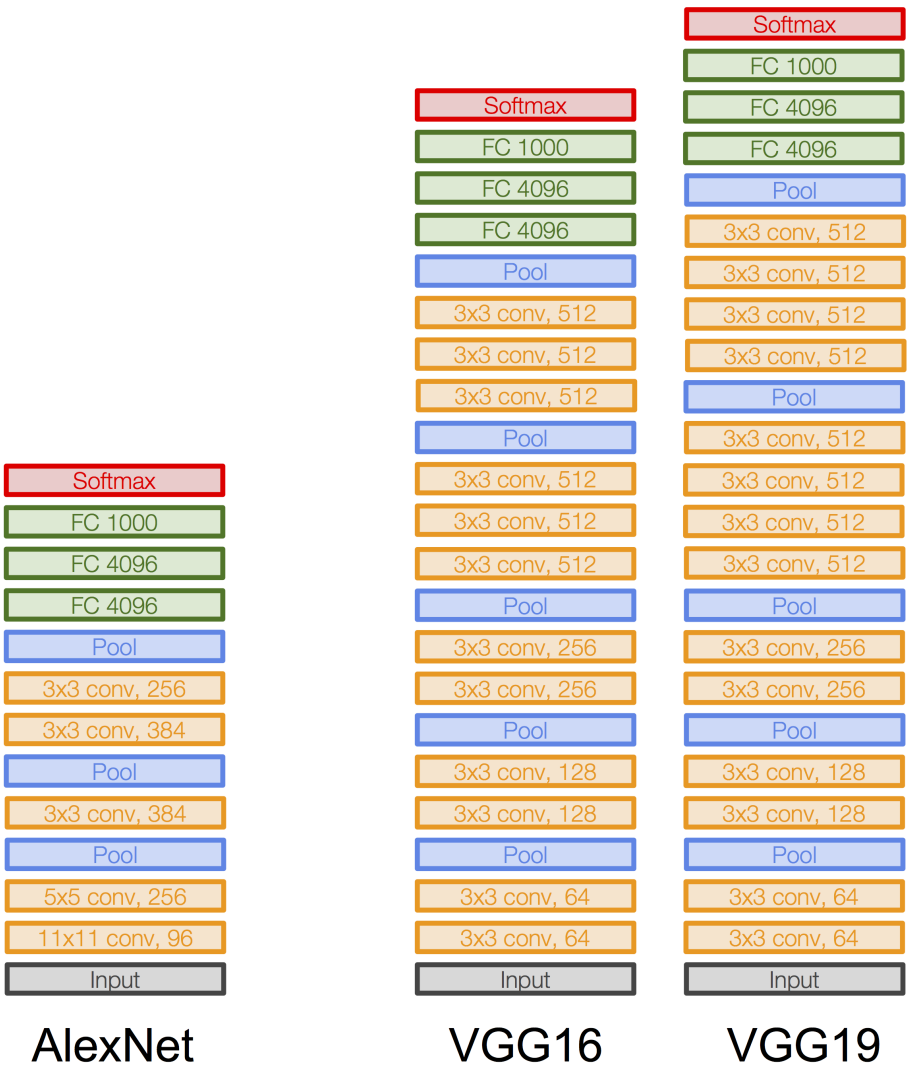
- VGG Net (by Simonyan and Zisserman) had few changes from the original AlexNet.

VGG Deep Net



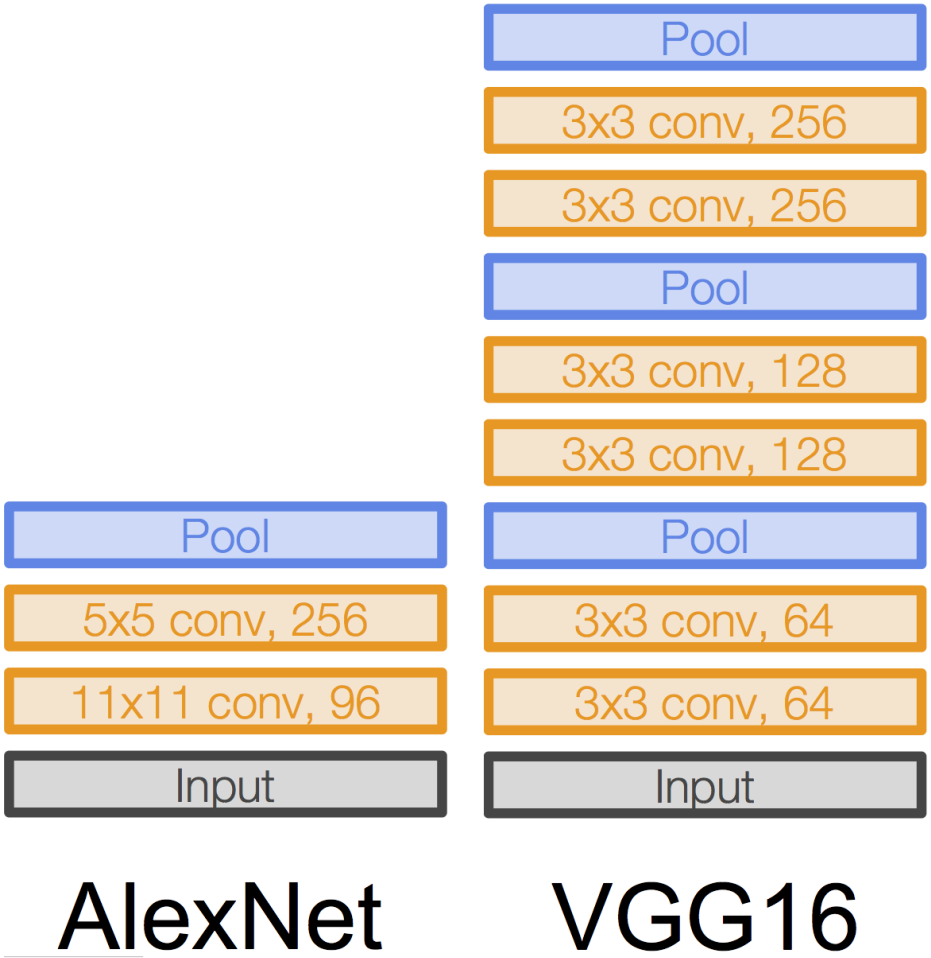
- VGG Net (by Simonyan and Zisserman) had few changes from the original AlexNet.
- The main change was that instead of using broad filters (for instance 11x11) used by AlexNet, they used smaller convolutional filters of (3x3)

VGG Deep Net



- VGG Net (by Simonyan and Zisserman) had few changes from the original AlexNet.
- The main change was that instead of using broad filters (for instance 11x11) used by AlexNet, they used smaller convolutional filters of (3x3)
- They obtained a deeper net using this strategy

VGG Deep Net



- The reason for this change is, the effective receptive field of a 7x7 convolutional filter can be obtained by stacking multiple convolutional filters with receptive field size of 3x3

Figure: VGG Net- a closer look at the

VGG Deep Net

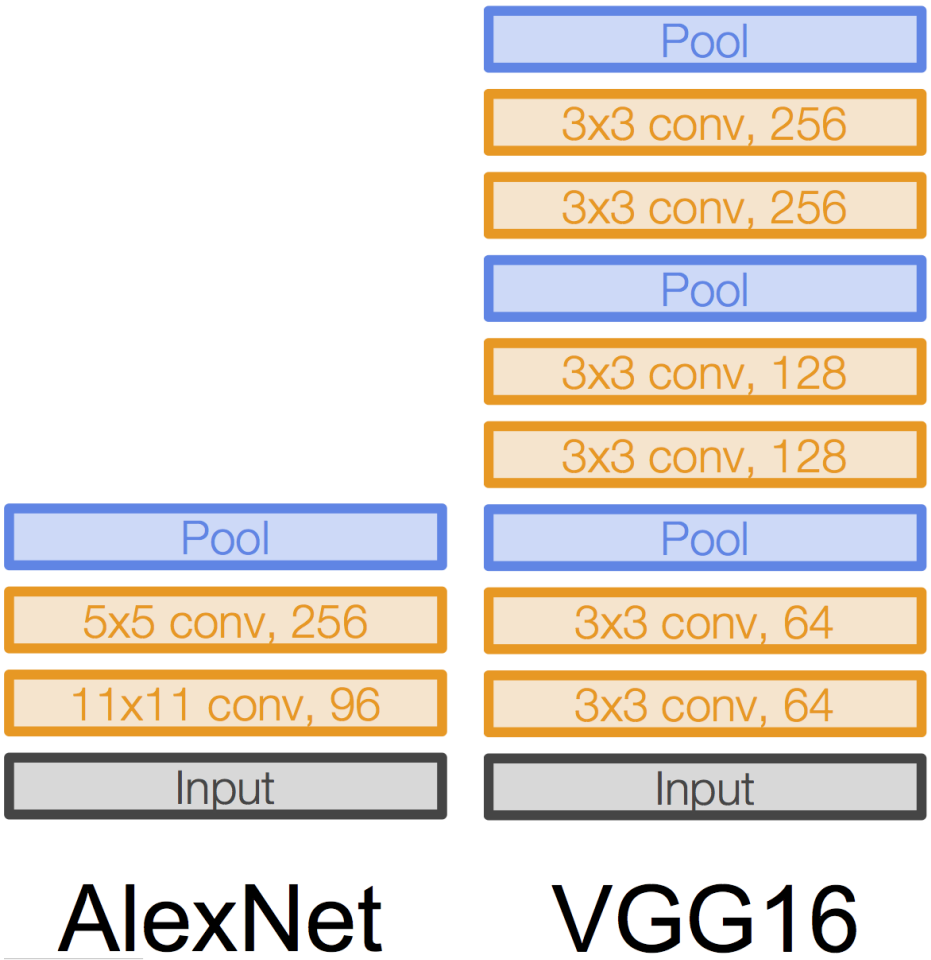


Figure: VGG Net- a closer look at the

- The reason for this change is, the effective receptive field of a 7x7 convolutional filter can be obtained by stacking multiple convolutional filters with receptive field size of 3x3
- The deeper network obtained by stacking multiple convolutional filters also has a smaller number of parameters

VGG Deep Net

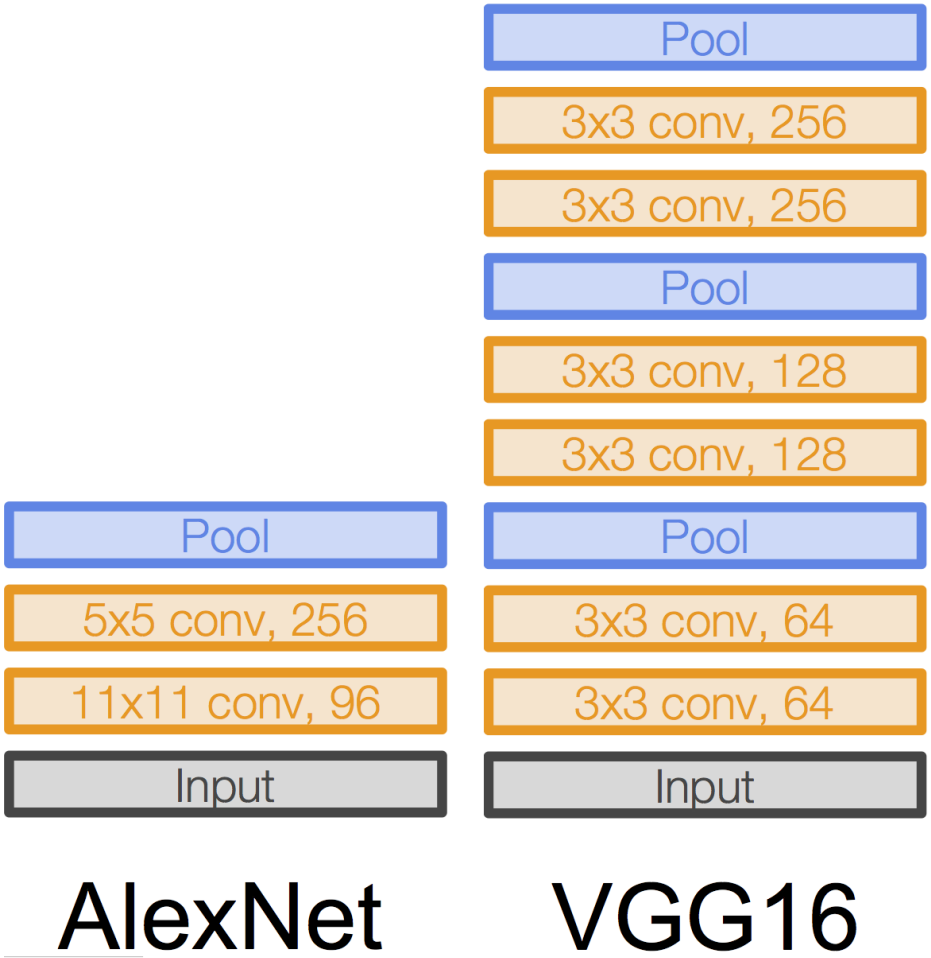


Figure: VGG Net- a closer look at the

- The reason for this change is, the effective receptive field of a 7x7 convolutional filter can be obtained by stacking multiple convolutional filters with receptive field size of 3x3
- The deeper network obtained by stacking multiple convolutional filters also has a smaller number of parameters
- The deeper network however has more non-linearities

GoogleNet Inception Network

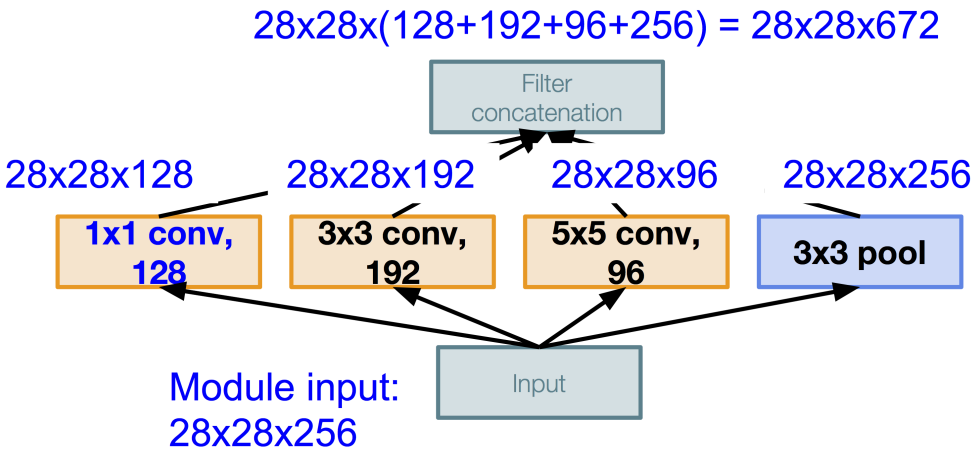


Figure: GoogleNet- a closer look at the initial layers, figure courtesy cs231n Stanford

- The main idea of the inception network architecture is to obtain the optimal local sparse structure by using multiple dense components.

GoogleNet Inception Network

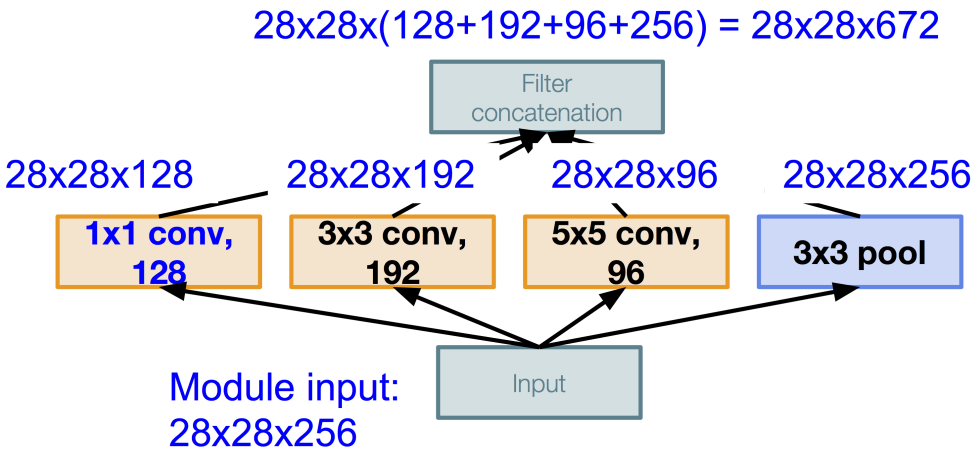


Figure: GoogleNet- a closer look at the initial layers, figure courtesy cs231n Stanford

- The main idea of the inception network architecture is to obtain the optimal local sparse structure by using multiple dense components.
- This is obtained by examining the network using multiple scales locally at a block

GoogleNet Inception Network

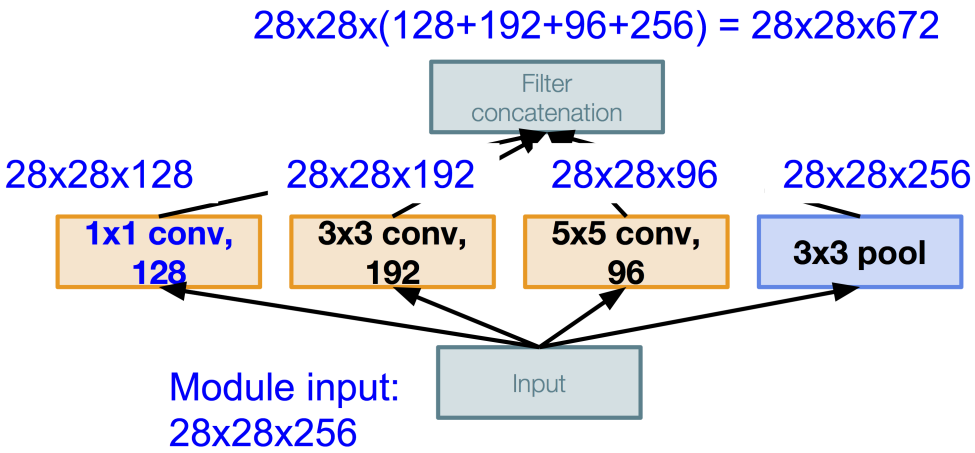


Figure: GoogleNet- a closer look at the initial layers, figure courtesy cs231n Stanford

- The main idea of the inception network architecture is to obtain the optimal local sparse structure by using multiple dense components.
- This is obtained by examining the network using multiple scales locally at a block
- The deep network is then obtained by stacking these blocks

GoogleNet Inception Network

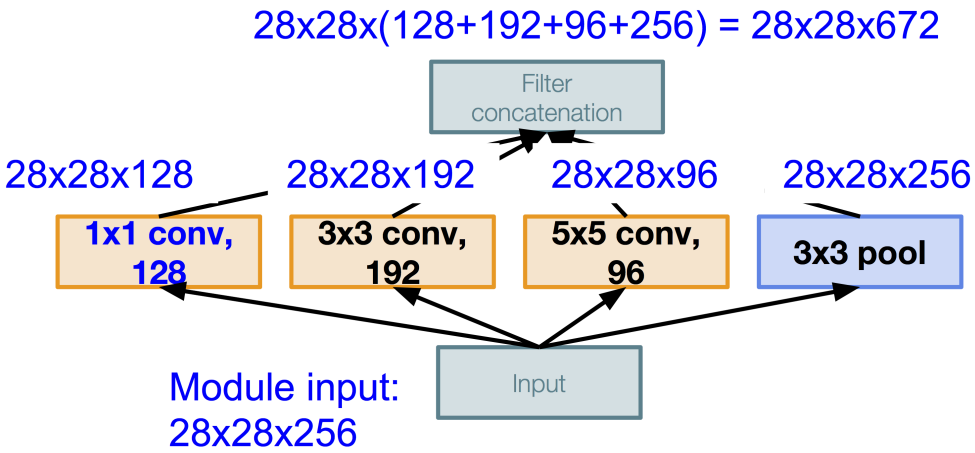


Figure: GoogleNet- inception block, figure courtesy cs231n Stanford

- The main idea of the inception network architecture is to obtain the optimal local sparse structure by using multiple dense components.

GoogleNet Inception Network

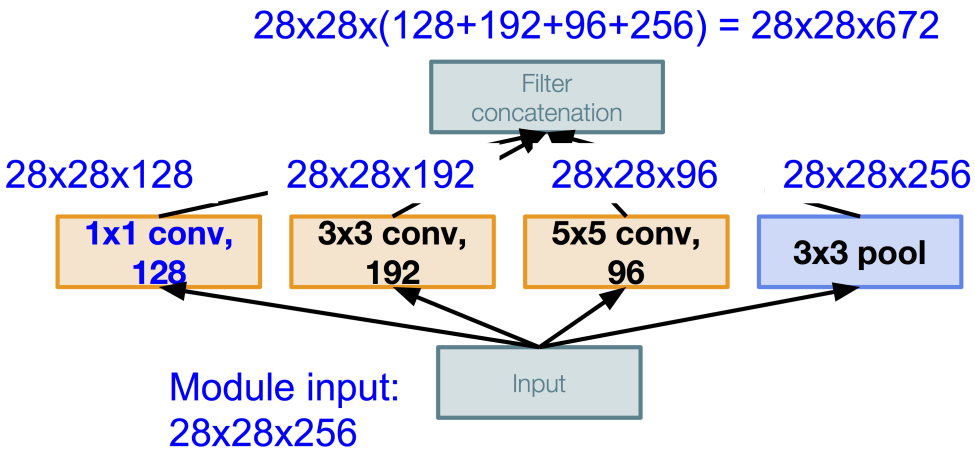


Figure: GoogleNet- inception block, figure courtesy cs231n Stanford

- The main idea of the inception network architecture is to obtain the optimal local sparse structure by using multiple dense components.
- This is obtained by examining the network using multiple scales locally at a block

GoogleNet Inception Network

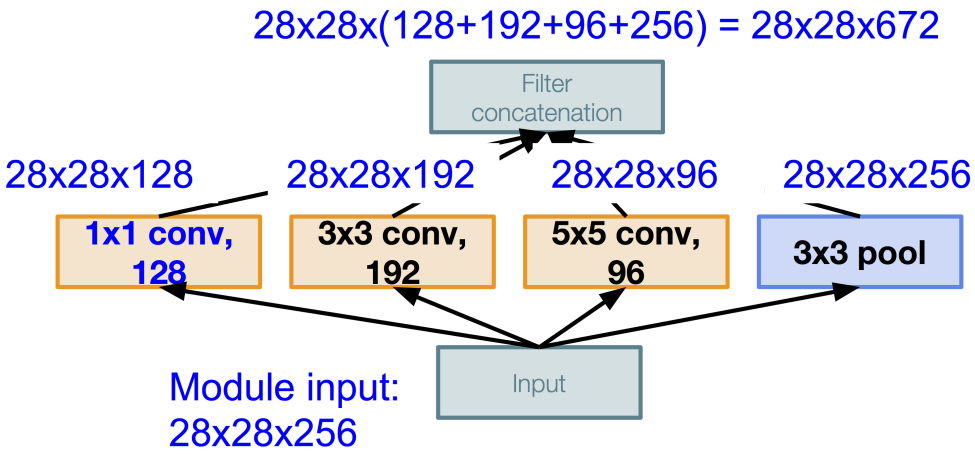


Figure: GoogleNet- inception block, figure courtesy cs231n Stanford

- The main idea of the inception network architecture is to obtain the optimal local sparse structure by using multiple dense components.
- This is obtained by examining the network using multiple scales locally at a block
- The deep network is then obtained by stacking these blocks

GoogleNet Inception Network

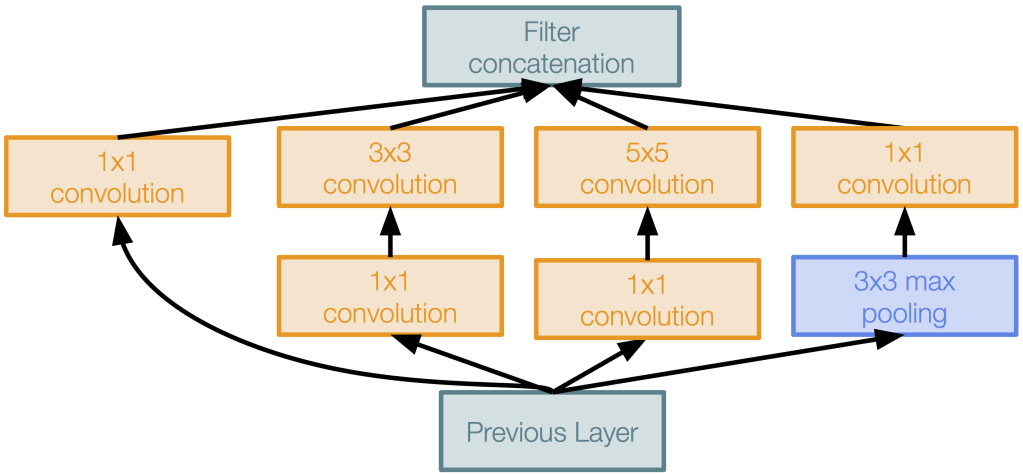


Figure: GoogleNet- inception block with dimensionality reduction, figure courtesy cs231n Stanford

- The resulting filters obtained by concatenating the output is quite large

GoogleNet Inception Network

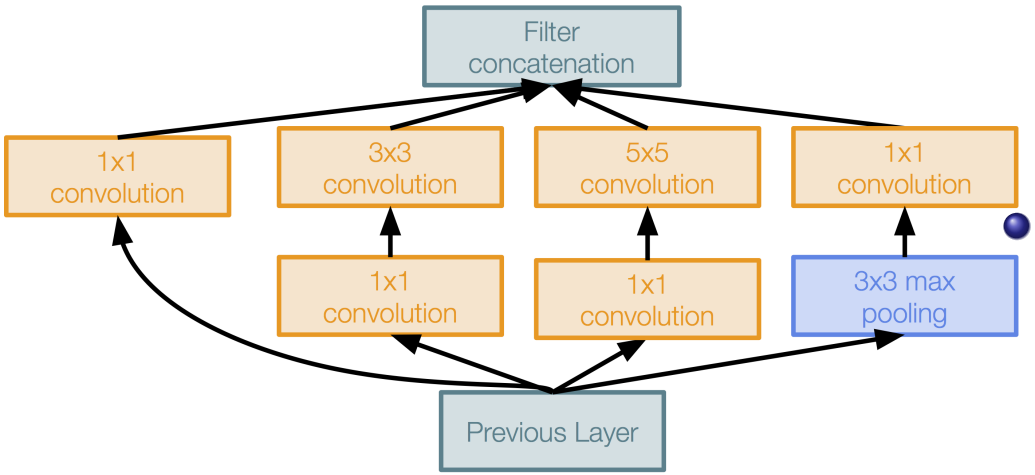


Figure: GoogleNet- inception block with dimensionality reduction, figure courtesy cs231n Stanford

- The resulting filters obtained by concatenating the output is quite large
- The design adopted in the inception architecture uses 1x1 convolutional blocks to reduce the size of the network

GoogleNet Inception Network

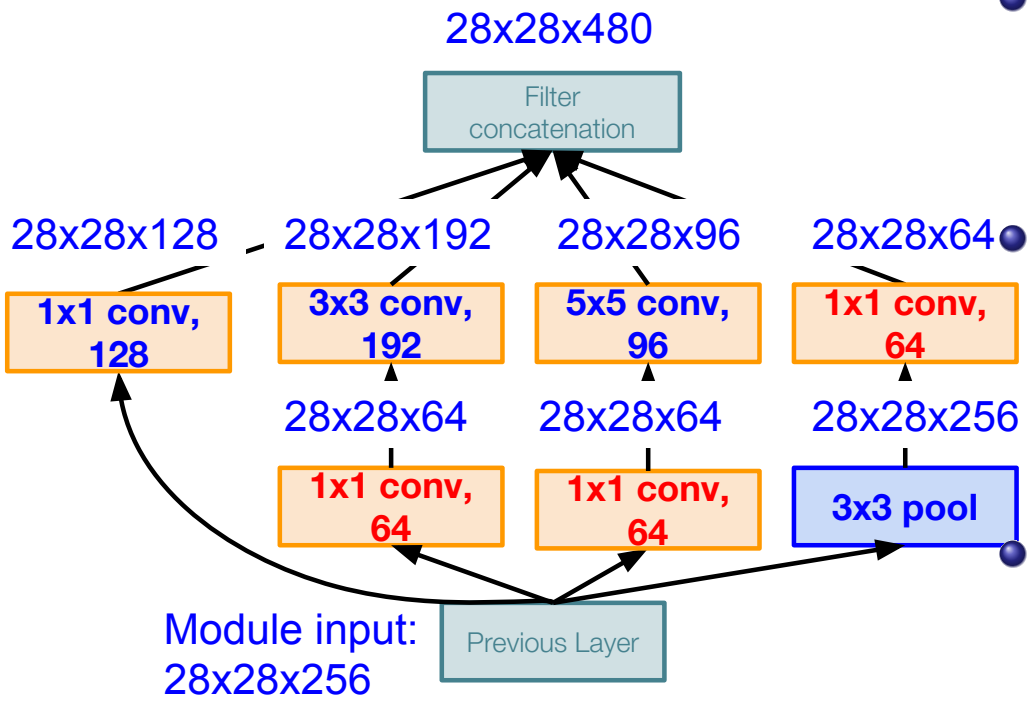
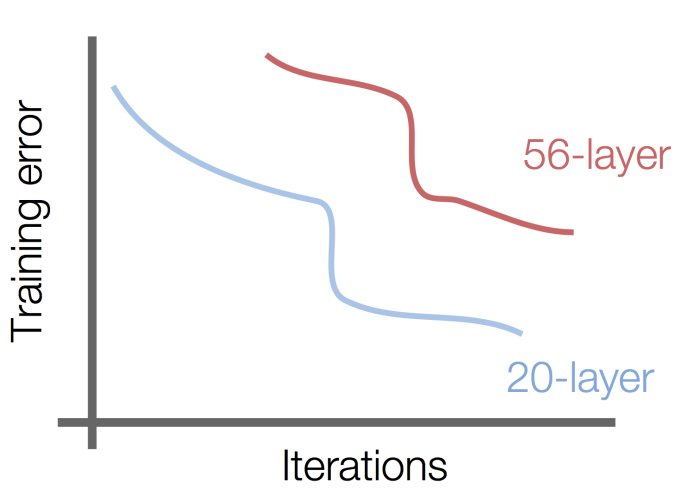


Figure: GoogleNet- inception block with dimensionality reduction, figure courtesy cs231n Stanford

- The resulting filters obtained by concatenating the output is quite large
- The design adopted in the inception architecture uses 1x1 convolutional blocks to reduce the size of the network
- The resulting network has a smaller output dimension as can be observed by looking at the detailed resultant numbers

ResNet Architecture - He et al

He et al investigated the question as to what would happen in a 'plain' convolutional network if we continued stacking layers and made it deeper.



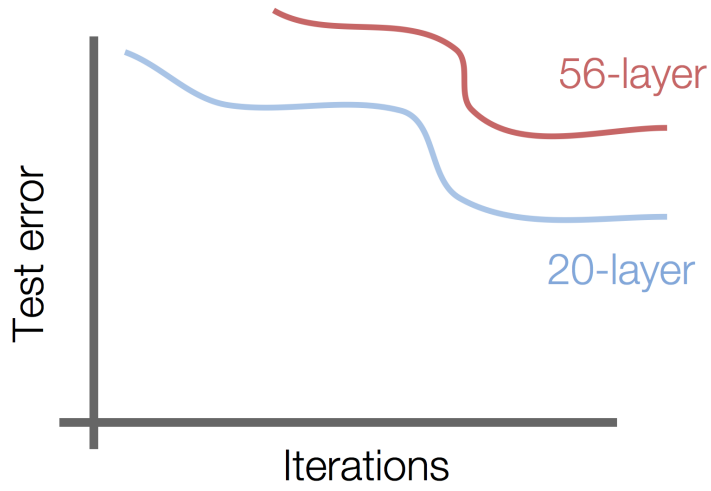


Figure: Evaluation of deeper plain networks in CIFAR dataset by He et al

ResNet Architecture - He et al

He et al investigated the question as to what would happen in a 'plain' convolutional network if we continued stacking layers and made it deeper.

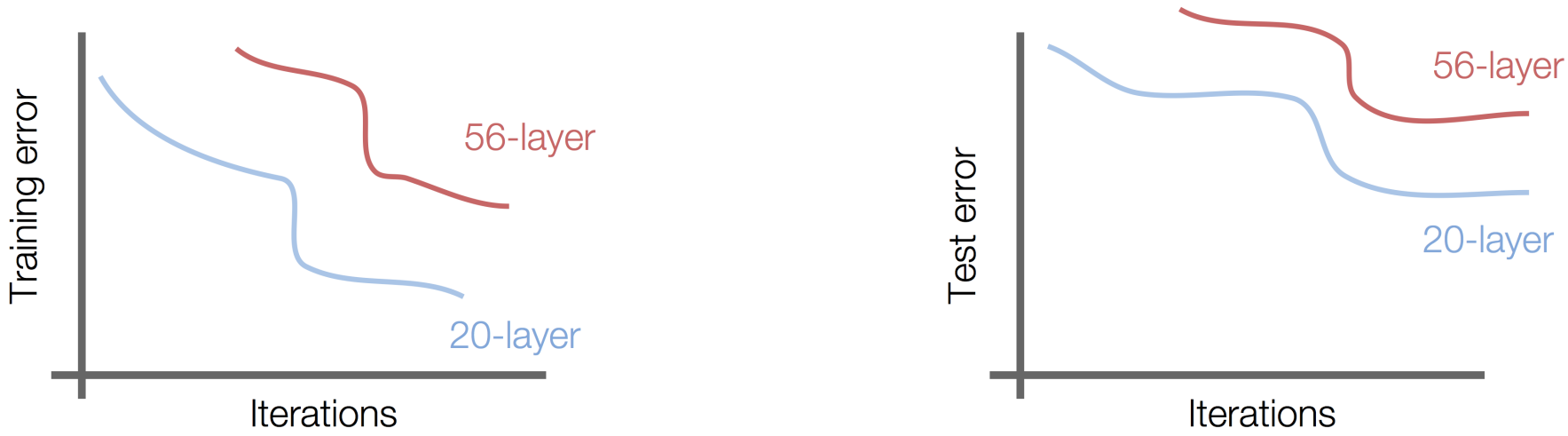
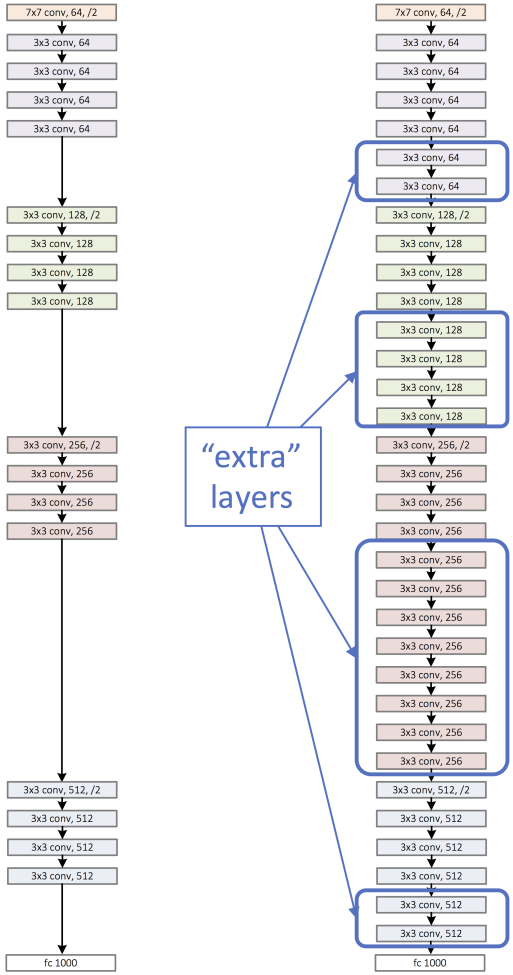


Figure: Evaluation of deeper plain networks in CIFAR dataset by He et al

- They observed that both the training and test error increased.

ResNet Architecture - He et al

a shallower model (18 layers)

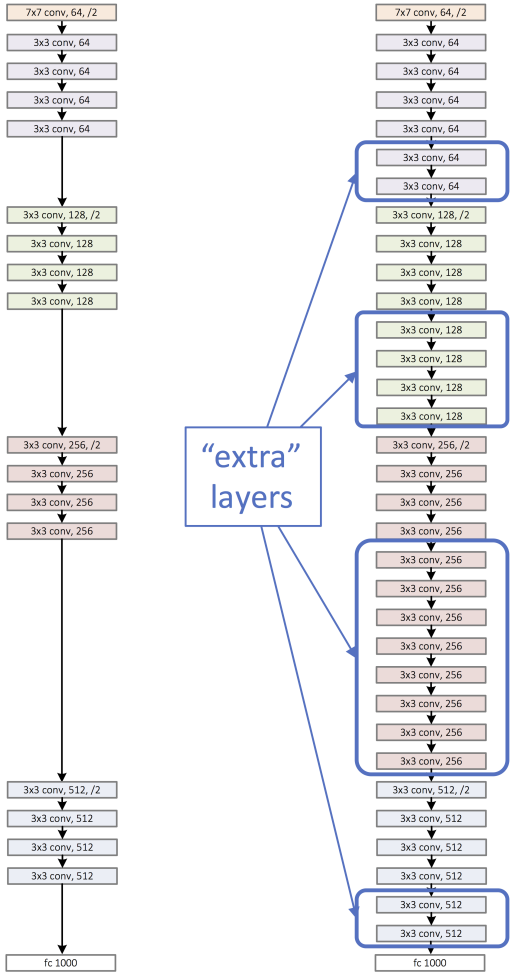


- This result was counter intuitive to the authors as they expected that a deeper model should be able to model the problem better.

Figure: Fig. by Kaiming He

ResNet Architecture - He et al

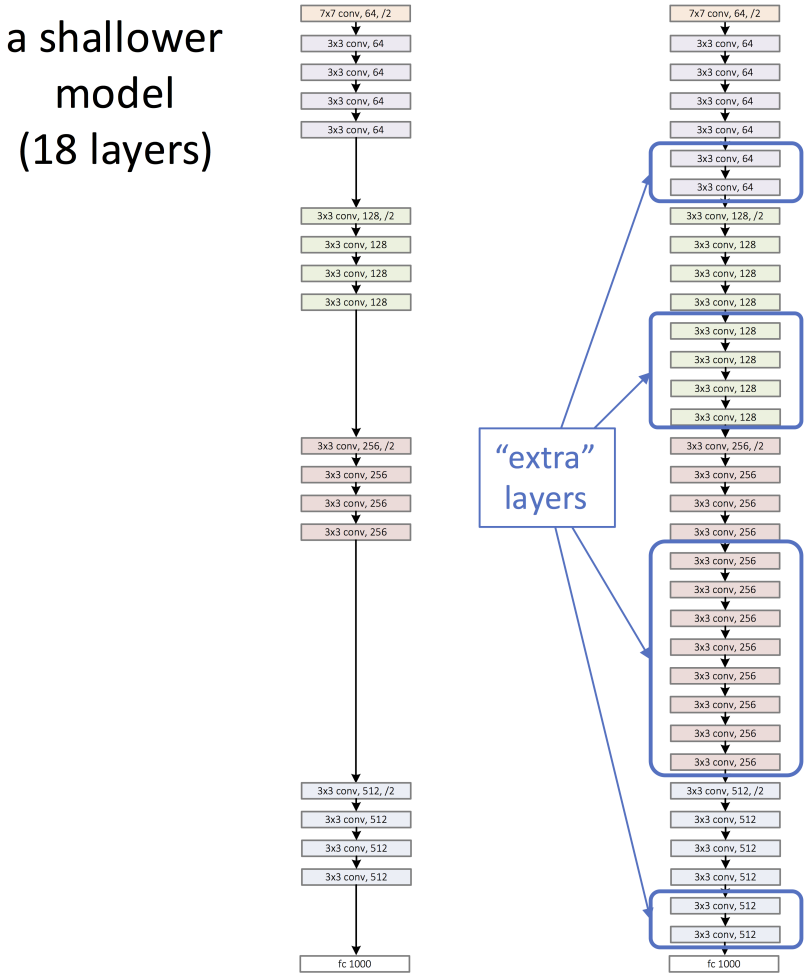
a shallower model (18 layers)



- This result was counter intuitive to the authors as they expected that a deeper model should be able to model the problem better.
- For the deeper network, one can obtain a better output just by setting the additional layers to identity

Figure: Fig. by Kaiming He

ResNet Architecture - He et al



- This result was counter intuitive to the authors as they expected that a deeper model should be able to model the problem better.
- For the deeper network, one can obtain a better output just by setting the additional layers to identity
- This gave them the idea that the network that is being learnt should learn to model just the additional residue

Figure: Fig. by Kaiming He

ResNet Architecture - He et al

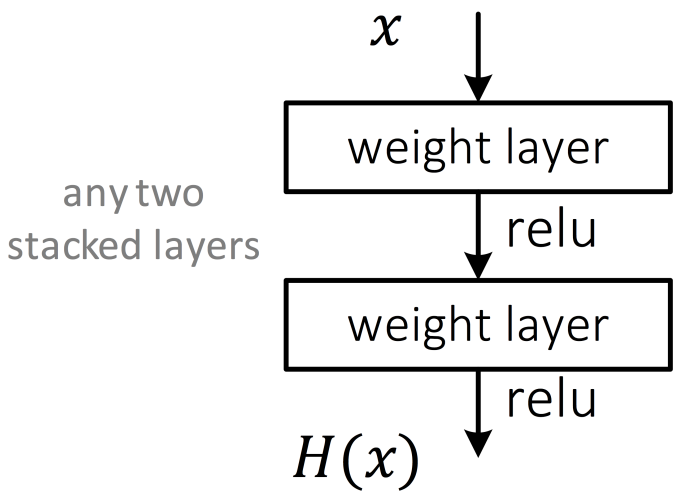


Figure: Illustration of idea by Kaiming He

- The main idea is as follows, normally given a stack of two layers that is learning a function $H(x)$ given an input x

ResNet Architecture - He et al

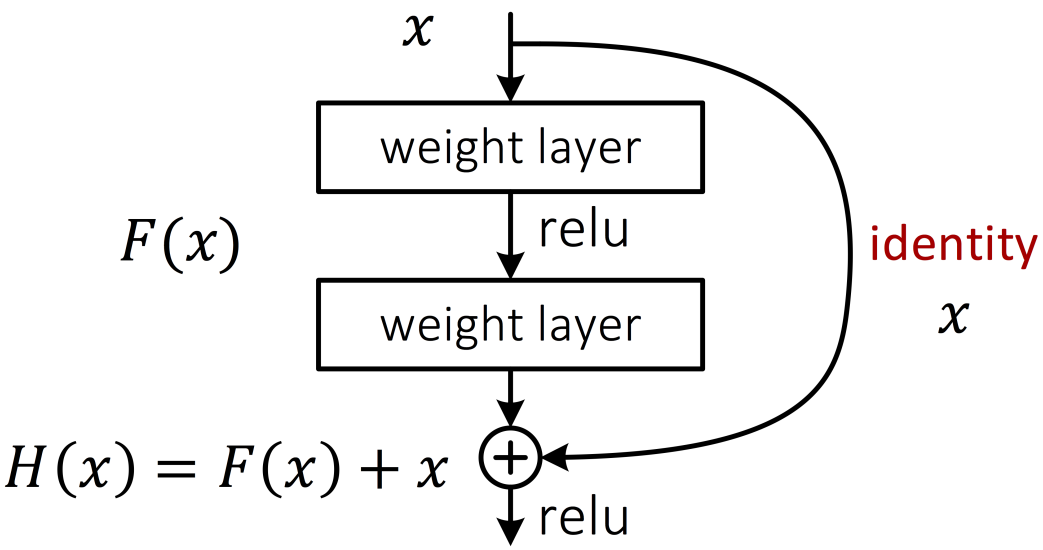


Figure: Illustration of idea by Kaiming He

- The main idea is as follows, normally given a stack of two layers that is learning a function $H(x)$ given an input x
- The idea is that the stack of layers could be used to model the residue function $F(x)$ where $F(x)$ could learn the identity function. The resultant function would be $H(x) = F(x) + x$ and the stack of layers learns the residue function $H(x) - x$.

Results of ResNet Architecture - He et al,

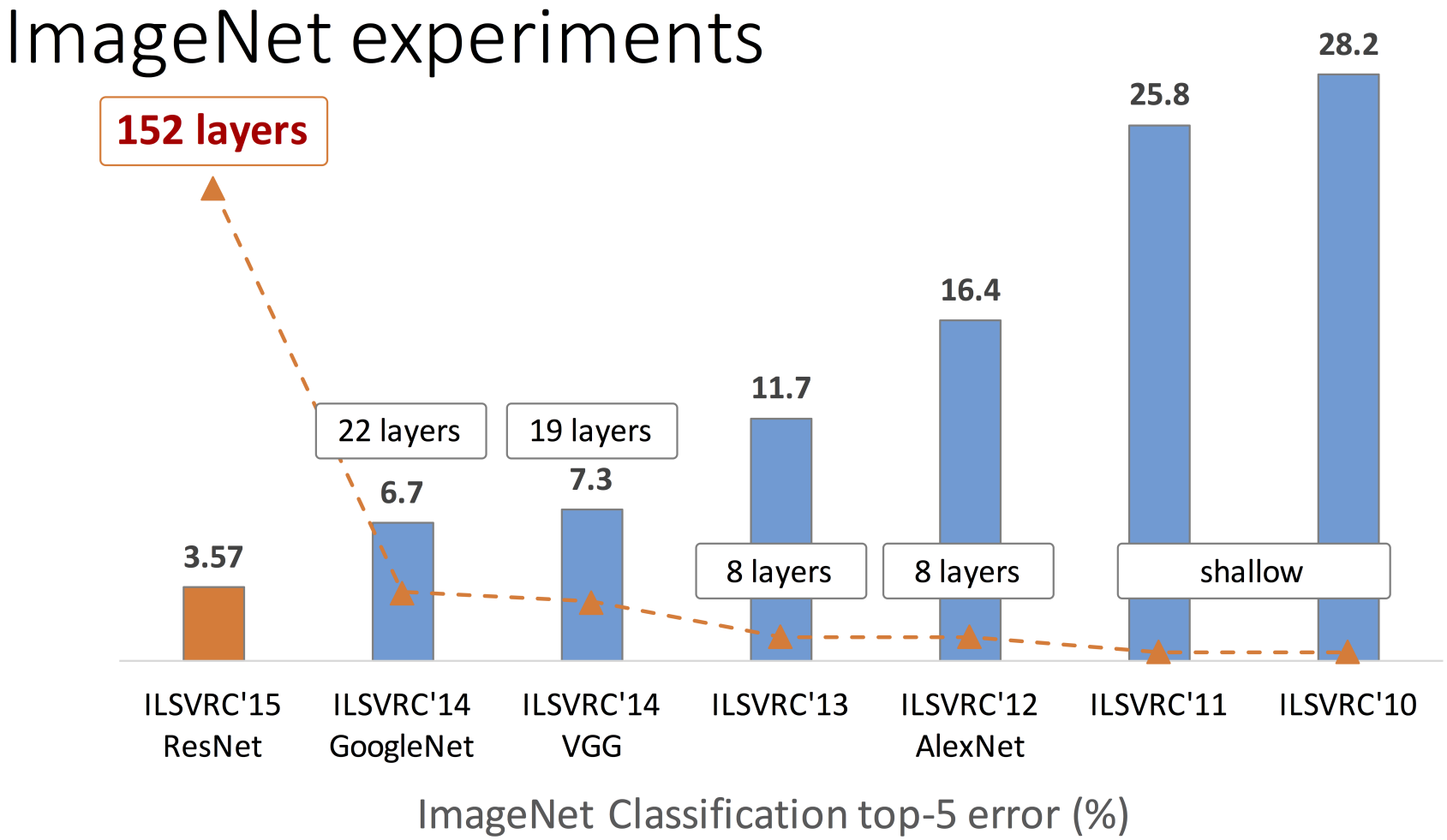


Figure: Results of Resnet 152 layer network as against other networks on ImageNet challenge. Figure by Kaiming He

Results of ResNet Architecture - He et al,

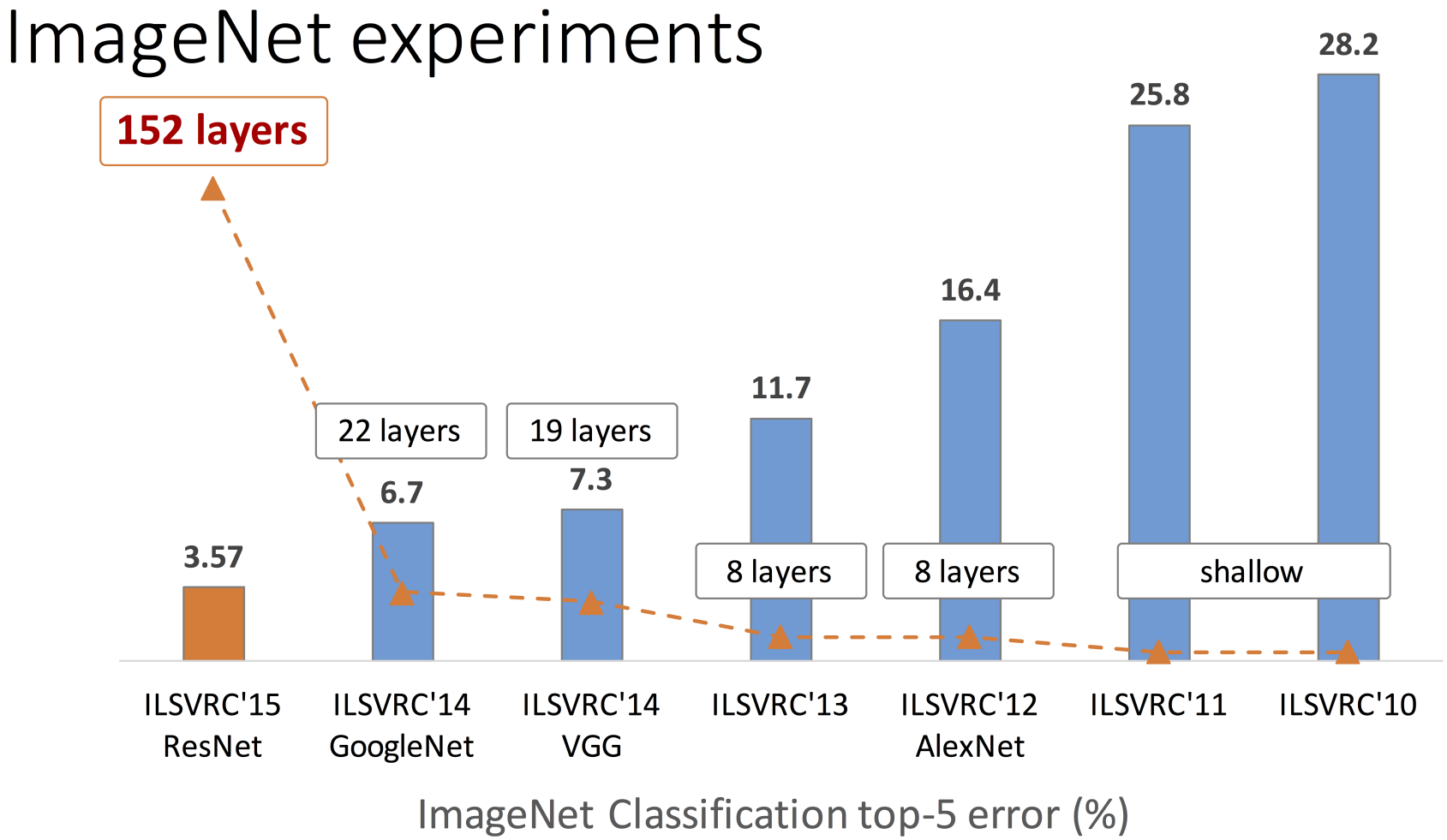


Figure: Results of Resnet 152 layer network as against other networks on ImageNet challenge. Figure by Kaiming He

Results of ResNet Architecture - He et al,

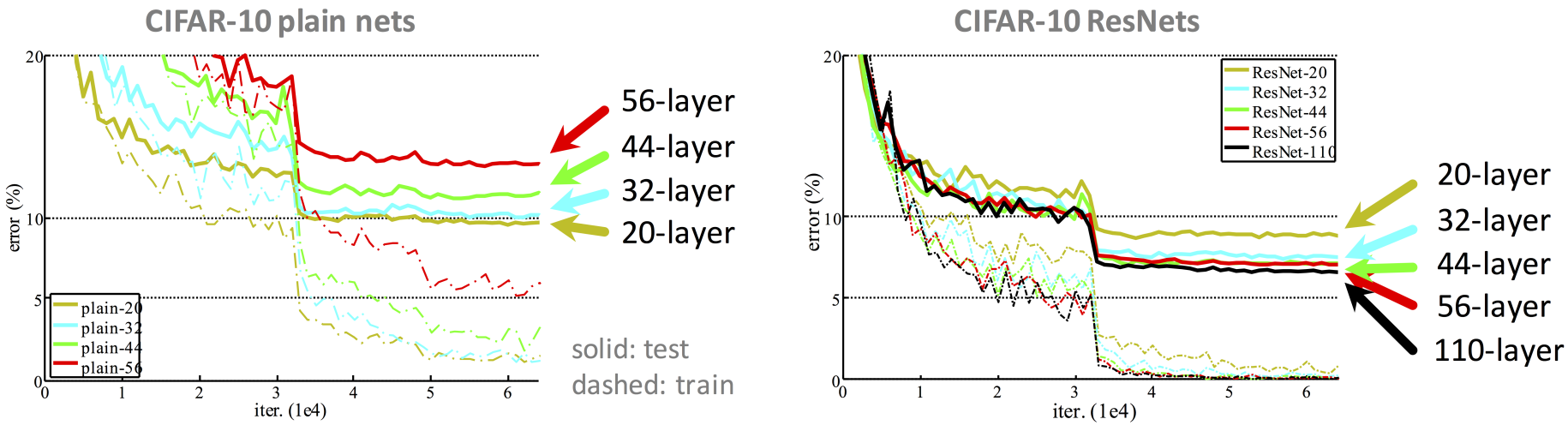


Figure: Results of Resnet as against a plain network on Cifar 10 dataset. Figure by Kaiming He

Outline

1 Recap

2 Architecture Variants

3 Initialisation
- Xavier

4 Normalisation

Xavier Initialisation

One important point while training networks is to obtain a proper initialisation. Towards this end, a solution was proposed by Xavier Glorot and Bengio (AISTATS 2010).

- The approach is as follows
  Consider a network that obtains output Y from a set of weights W and input X through the following equation

$$Y = \sum_{i=1}^{n} W_i X_i \tag{1}$$

Xavier Initialisation

One important point while training networks is to obtain a proper initialisation. Towards this end, a solution was proposed by Xavier Glorot and Bengio (AISTATS 2010).

- The approach is as follows
  Consider a network that obtains output Y from a set of weights W and input X through the following equation

$$Y = \sum_{i=1}^{n} W_i X_i \tag{1}$$

- Then the variance of $W_i X_i$ is obtained as

$$\text{Var}(W_i X_i) = E(X_i)^2 \text{Var}(W_i) + E(W_i)^2 \text{Var}(X_i) + \text{Var}(W_i) Var(X_i) \tag{2}$$

Xavier Initialisation

- Assuming the input X and the weights W to have a zero mean the resulting variance is obtained as

$$\text{Var}(W_i X_i) = \text{Var}(W_i) Var(X_i) \tag{3}$$

Xavier Initialisation

- Assuming the input X and the weights W to have a zero mean the resulting variance is obtained as

$$\text{Var}(W_i X_i) = \text{Var}(W_i)\,Var(X_i) \tag{3}$$

- Further assuming that the the X_i and W_i are independent and identically distributed

$$\text{Var}(Y) = \text{Var}(\sum_{i=1}^{n} W_i X_i). \tag{4}$$

That gives us

$$\text{Var}(Y) = n\text{Var}(W_i)\text{Var}(X_i) \tag{5}$$

Xavier Initialisation

- For training the network, we would want the input and output to have similar variance. Therefore, we obtain that

$$\text{Var}(W_i) = \frac{1}{n_{\text{input}}} \tag{6}$$

Xavier Initialisation

- For training the network, we would want the input and output to have similar variance. Therefore, we obtain that

$$\text{Var}(W_i) = \frac{1}{n_{\text{input}}} \tag{6}$$

- A similar analysis is applicable while going through backpropagation where we would want the backpropagated gradient to also be the same, resulting in

$$\text{Var}(W_i) = \frac{1}{n_{\text{output}}} \tag{7}$$

Xavier Initialisation

- For training the network, we would want the input and output to have similar variance. Therefore, we obtain that

$$\text{Var}(W_i) = \frac{1}{n_{\text{input}}} \tag{6}$$

- A similar analysis is applicable while going through backpropagation where we would want the backpropagated gradient to also be the same, resulting in

$$\text{Var}(W_i) = \frac{1}{n_{\text{output}}} \tag{7}$$

- Averaging the constraints from the forward and backward propagation on the weights, we obtain that the weights should be initialised through a zero mean Gaussian with the variance being

$$\text{Var}(W_i) = \frac{2}{n_{\text{input}} + n_{\text{output}}} \tag{8}$$

Outline

1 Recap

2 Architecture Variants

3 Initialisation

4 Normalisation
- Batch Normalisation

Batch Normalisation

- A commonly used strategy that is used while training is to apply Batch normalisation. Batch normalisation has been suggested in the work by Ioffe and Szegedy (ICML 2015)

Batch Normalisation

- A commonly used strategy that is used while training is to apply Batch normalisation. Batch normalisation has been suggested in the work by Ioffe and Szegedy (ICML 2015)
- It has been observed to greatly accelerate training while reducing the sensitivity to initialisation

Batch Normalisation

- A commonly used strategy that is used while training is to apply Batch normalisation. Batch normalisation has been suggested in the work by Ioffe and Szegedy (ICML 2015)
- It has been observed to greatly accelerate training while reducing the sensitivity to initialisation
- The basic idea is that the input X in each mini batch is normalised as follows:

$$\hat{X} = \frac{X - \mu}{\sigma} \tag{9}$$

where μ is the mean of X in mini-batch and σ is the standard deviation of X in a mini-batch

Batch Normalisation

- A commonly used strategy that is used while training is to apply Batch normalisation. Batch normalisation has been suggested in the work by Ioffe and Szegedy (ICML 2015)
- It has been observed to greatly accelerate training while reducing the sensitivity to initialisation
- The basic idea is that the input X in each mini batch is normalised as follows:

$$\hat{X} = \frac{X - \mu}{\sigma} \tag{9}$$

where μ is the mean of X in mini-batch and σ is the standard deviation of X in a mini-batch

- The resultant output of the batch-normalisation Y is then obtained as

$$Y = \gamma\hat{X} + \beta \tag{10}$$

where γ and β are learnt variables.

Batch Normalisation

- The resultant output of the batch-normalisation Y is then obtained as

$$Y = \gamma \hat{X} + \beta \tag{11}$$

  where γ and β are learnt variables.
- The idea is that the network is allowed to remap the variance and mean of the distribution to optimise the output. In case the unit variance and zero mean is appropriate for the network then it will learn $\gamma = 1$ and $\beta = 0$. Whereas if the original variance and mean are appropriate the network can undo the normalisation.

Results of Batch Normalisation

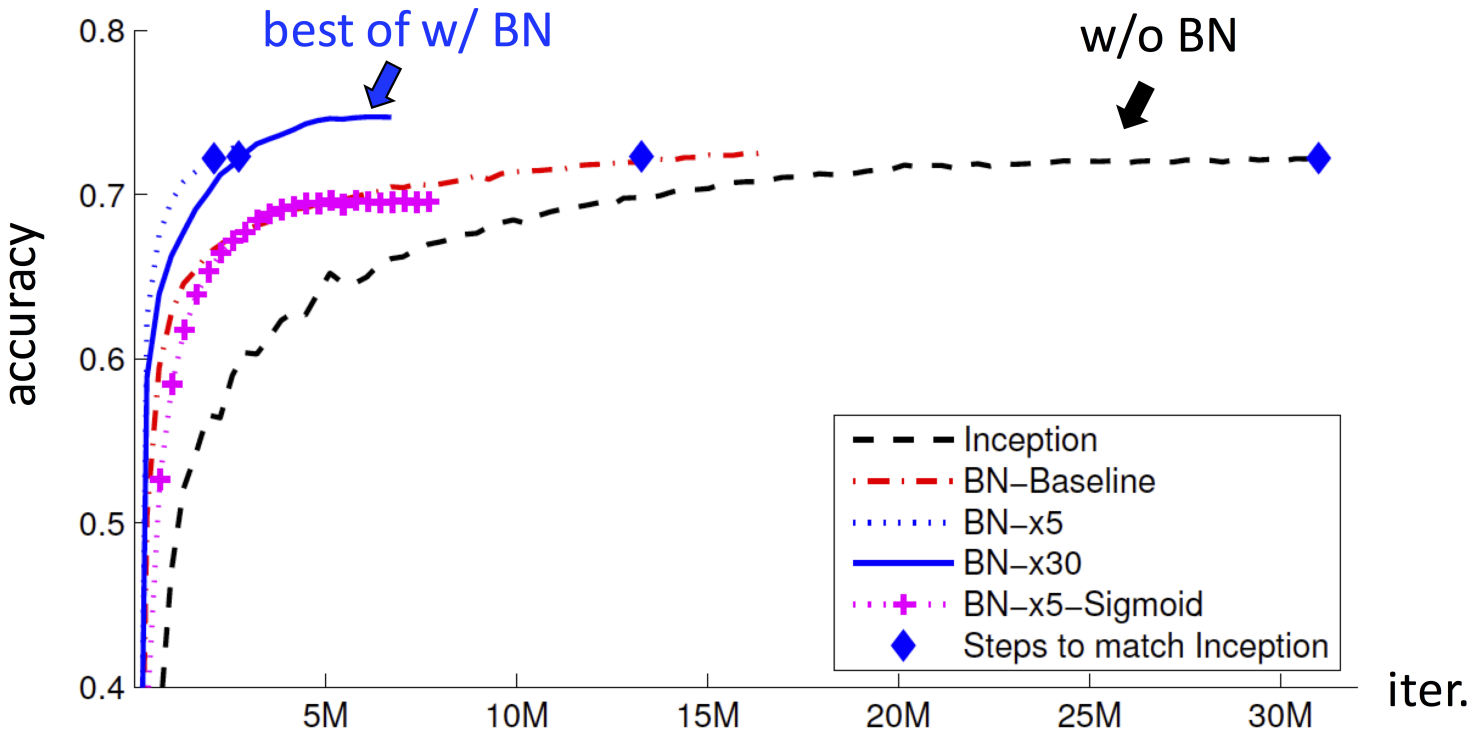


Figure: Results of Batch Normalisation. Figure by Ioffe and Szegedy

The End