Image Recognition Models

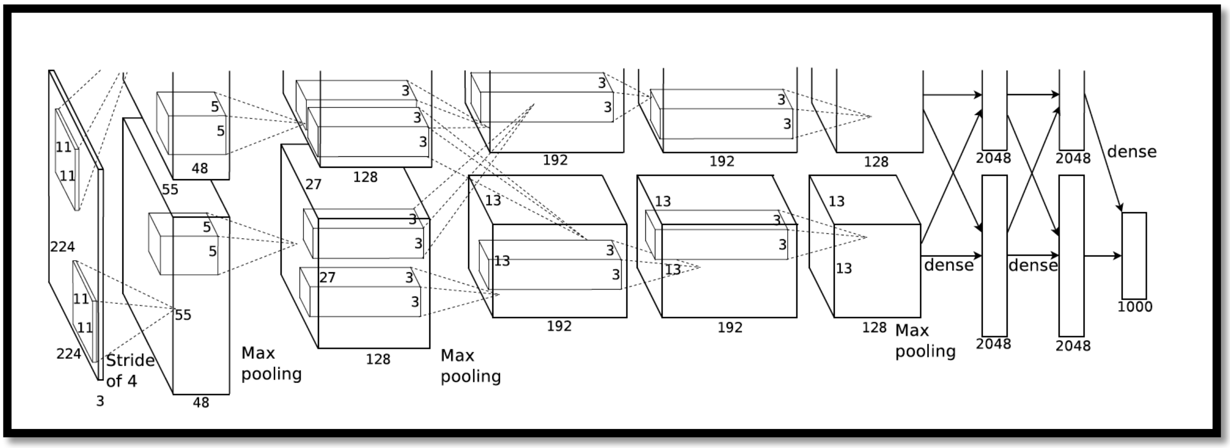
EXISTING MODEL

1. ALEXNET
2. ZFNET
3. VGGNET
4. INCEPTION
5. RESNET

ALEXNET [Krizhevsky et al. 2012]

In 2012, Alex Krizhevsky released AlexNet which was a deeper and much wider version of the LeNet and won by a large margin the difficult ImageNet competition. The success of AlexNet started a small revolution. Convolutional neural network is now the workhorse of Deep Learning, which became the new name for “large neural networks that can now solve useful tasks”.

MODEL ARCHITECTURE



FEATURES

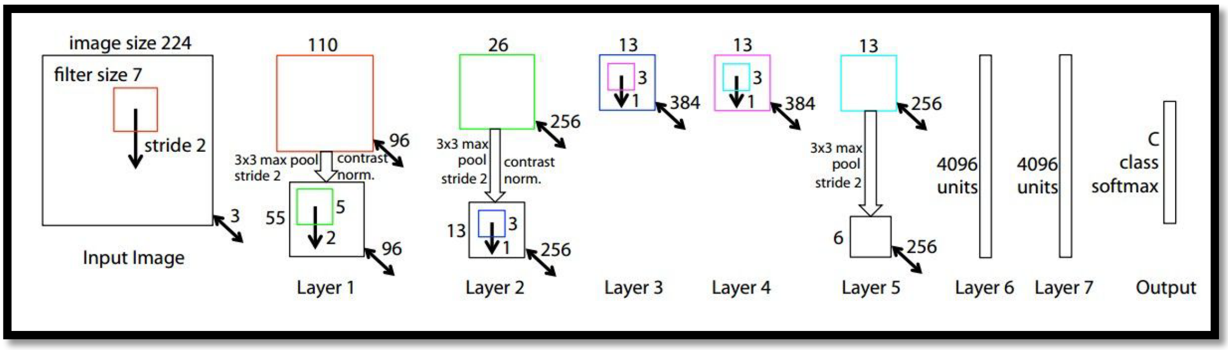
1. Relu Activation function ($f(x) = \max(0, x)$) for faster training
2. Input image size – 224X224X3
3. Convolution layer 1 – 11X11X3, 96 filters, Stride – 4
4. Convolution layer 2 – 5X5X48, 256 filters
5. Convolution layer 3 – 3X3X256, 384 filters
6. Convolution layer 4 – 3X3X192, 384 filters
7. Convolution layer 5 – 3X3X192, 256 filters
8. heavy data augmentation to reduce overfitting
9. dropout (probability – 0.5)
10. batch size 128
11. SGD Momentum – (hyper parameters – 0.9 (for momentum))
12. use of GPUs

IMAGENET ACCURACY

15.4% top 5 error

ZFNET [Zeiler and Fergus, 2013]

ARCHITECTURE



FEATURES

1. Same architecture as ALEXNET
2. Convolution layer 1 – 7X7X3, 96 filters, Stride – 2
3. Convolution layer 2 – 5X5X48, 256 filters
4. Convolution layer 3 – 3X3X256, 512 filters
5. Convolution layer 4 – 3X3X192, 1024 filters
6. Convolution layer 5 – 3X3X192, 512 filters

VGGNET

The VGG networks from Oxford were the first to use much smaller 3×3 filters in each convolutional layers and also combined them as a sequence of convolutions.

This seems to be contrary to the principles of AlexNet, where large convolutions were used to capture similar features in an image. Instead of the 9×9 or 11×11 filters of AlexNet, filters started to become smaller, too dangerously close to the 1×1 convolutions. But the great advantage of VGG was the insight that multiple 3×3 convolution in sequence can emulate the effect of larger receptive fields.

VGG used large feature sizes in many layers and thus inference was quite costly at run-time. Reducing the number of features, as done in Inception bottlenecks, will save some of the computational cost.

ARCHITECTURE

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

THE BEST MODEL ACCORDING TO RESULTS IS THE ConvNet Configuration D.

FEATURES

1. Only 3X3 filters with stride 1 and pad 1
2. Only 2X2 max pool with stride 2

IMAGENET ACCURACY

7.3% top 5 error

GOOGLENET [Szegedy et. al., 2014]

Christian Szegedy from Google begun a quest aimed at reducing the computational burden of deep neural networks, and devised the GoogLeNet the first Inception architecture.

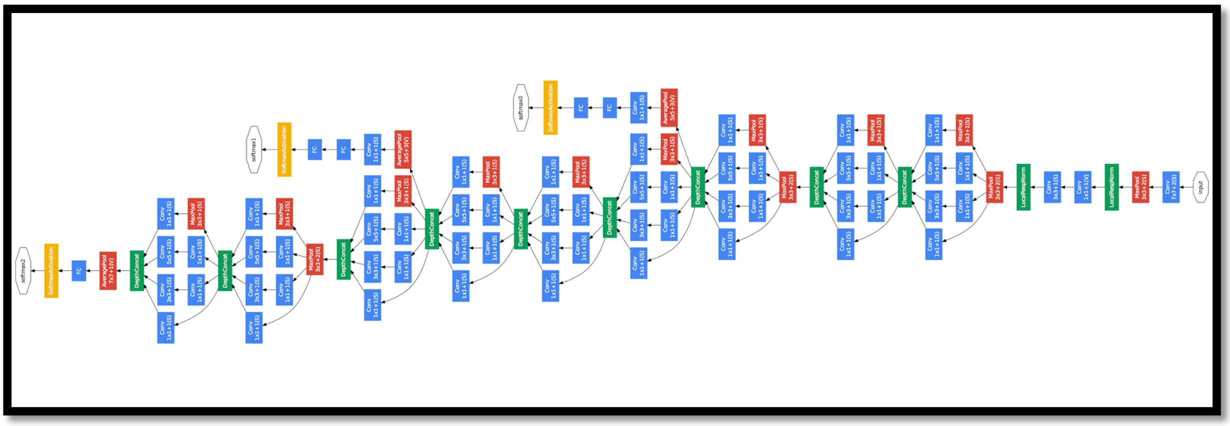
By Fall 2014, deep learning models were becoming extremely useful in categorizing the content of images and video frames. Most skeptics had given in that Deep Learning and neural nets came back to stay this time. Given the usefulness of these techniques, the internet giants like Google were very interested in efficient and large deployments of architectures on their server farms.

Christian thought a lot about ways to reduce the computational burden of deep neural nets while obtaining state-of-art performance (on ImageNet, for example). Or be able to keep the computational cost the same, while offering improved performance.

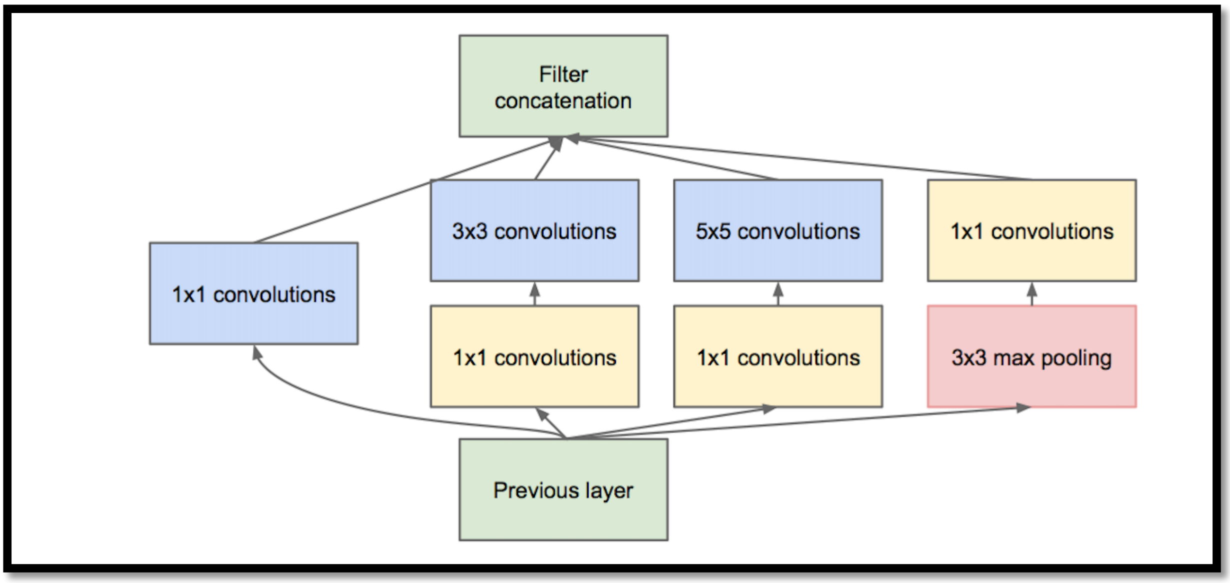
He and his team came up with the Inception module, which at a first glance is basically the parallel combination of 1×1, 3×3, and 5×5 convolutional filters. But the great insight of the inception module was the use of 1×1 convolutional blocks to reduce the number of features before the expensive parallel blocks. This is commonly referred as “bottleneck”.

GoogLeNet used a stem without inception modules as initial layers, and an average pooling plus softmax classifier. This classifier is also extremely low number of operations, compared to the ones of AlexNet and VGG. This contributed to a very efficient network design.

ARCHITECTURE



Complete architecture.



Inception module

FEATURES

1. Fully connected layers are removed.
2. Instead of one type of convolution filters introduces the inception module.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

IMAGENET ACCURACY

6.7% Top 5 Error

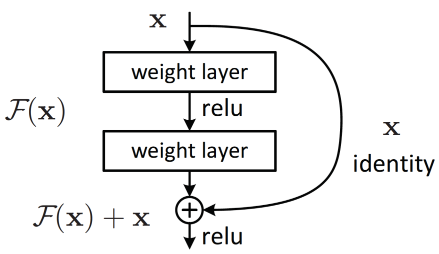
RESNET [He et al., 2015]

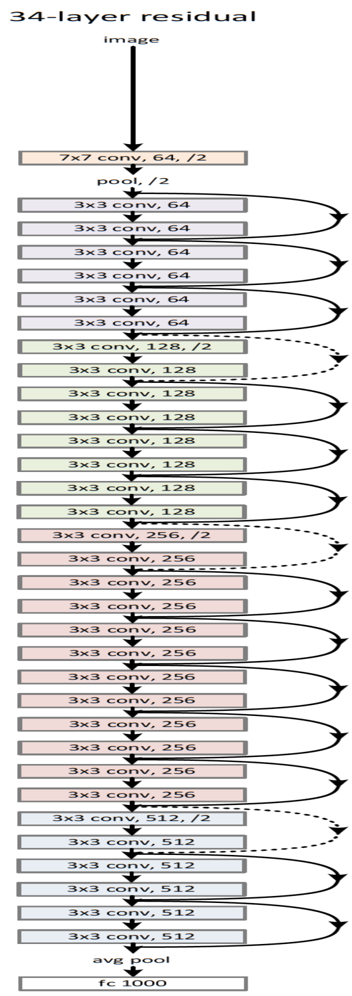
ResNet introduces a simple idea: feed the output of two successive convolutional layer AND also bypass the input to the next layers. they bypass TWO layers and the model is applied to large scales. Bypassing after 2 layers is the key intuition, as bypassing a single layer did not give much improvements.

ARCHITECTURE

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

RESNET
Building block





FEATURES

1. Introduction of Residual connections (“shortcut connections”)

IMAGENET ACCURACY

3.6% top 5 error

ACCURACY COMPARISON [Jonathan Huang et. al. 2016]

MODEL	ACCURACY %	NO. OF PARAMETERS
VGG16	71	14,714,688
INCEPTION	73.9	10,173,112
RESNET	76,4	42,605,504
INCEPTION RESNET V2	80.4	54,336,736

Experiments Conducted

1. Image classification (Own model):
    - CIFAR-10, CIFAR-100 dataset
    - CIFAR-10 accuracy – 78%, CIFAR-100 accuracy – 35%
    - Train steps – 30
2. Image Classification pre trained model (Inception Resnet V2)
    - Model trained for ImageNet dataset (1000 classes)
    - Accuracy (according to Tensorflow) : 80.4%

REFERENCES

[1] Alex Krizhevsky and Ilya Sutskever and Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, Advances in neural information processing systems, pp. 1097-1105,2012

[2] Matthew D Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks”, Computer Vision – ECCV 2014, pp 818-833, 2014

[3] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv technical report, 2014

[4] Christian Szegedy and Wei Liu, Yangqing Jia and Pierre Sermanet and Scott Reed and Dragomir Anguelov and Dumitru Erhan and Vincent Vanhoucke and Andrew Rabinovich, “Going Deeper with Convolutions”, Computer Vision and Pattern Recognition (CVPR) , 2015

[5] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, “Deep Residual Learning for Image Recognition”, Computer Vision and Pattern Recognition (CVPR), 2016

[6] Jonathan Huang and Vivek Rathod and Chen Sun and Menglong Zhu and Anoop Korattikara and Alireza Fathi and Ian Fischer and Zbigniew Wojna and Yang Song and Sergio Guadarrama and Kevin Murphy,” Speed/accuracy trade-offs for modern convolutional object detectors”, In submission to CVPR 2016, 2016.