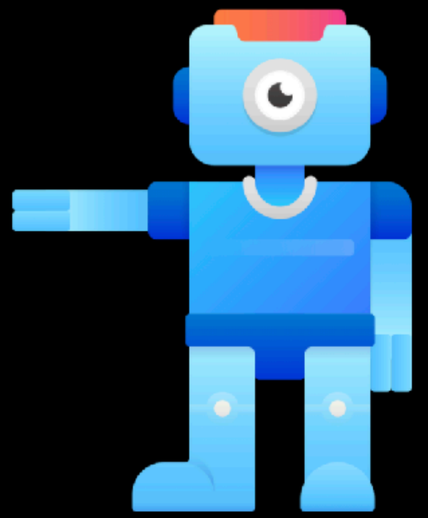


MODERN COMPUTER VISION

BY RAJEEV RATAN

Basic CNN Design Principles

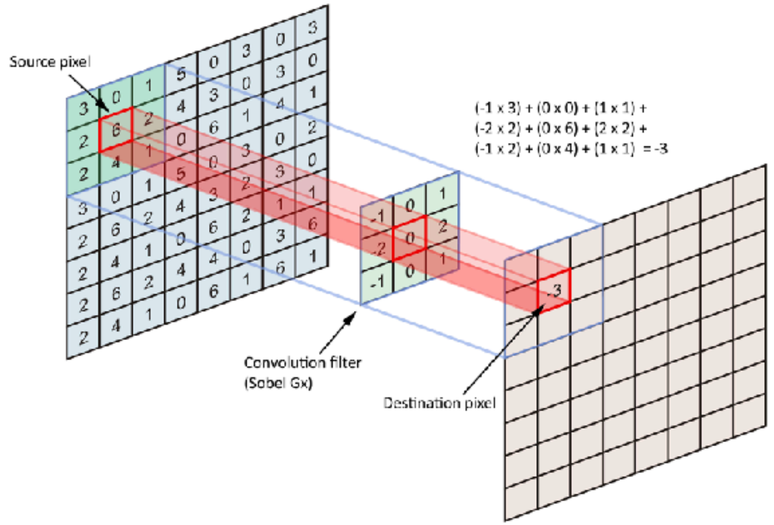
Rule of thumb guide to designing basic CNN Architectures

When Designing a CNN You're Faced with Many Questions

- How many Layers (Conv Filters)?
- How many Filters per layer?
- What size Kernel? Stride?
- Maxpool? Activation Functions?
- How long to train?

A Basic Guide to Designing Effective CNNs

Kernel or Filter Size



- Each filter is a matrix of weights that is convolved with the input image
- Each filter acts like a detector for some feature (edge, blob etc) in the image
- A smaller filters (3x3) are better at detecting local features
- Lager filters (7x7 or 9x9) are better at detecting larger features

How to decide?

- If you think what distinguishes your images are small localised features then smaller filters at better, if it's larger more global or high-level patterns then larger filters are better.
- A good practice is using smaller filters in earlier layers and gradually increase filter size in deeper layers
- Note, we use odd number sized filters so that we have source pixel (centre).
- https://datascience.stackexchange.com/questions/23183/why-convolutions-always-use-odd-numbers-as-filter-size

A Basic Guide to Designing Effective CNNs

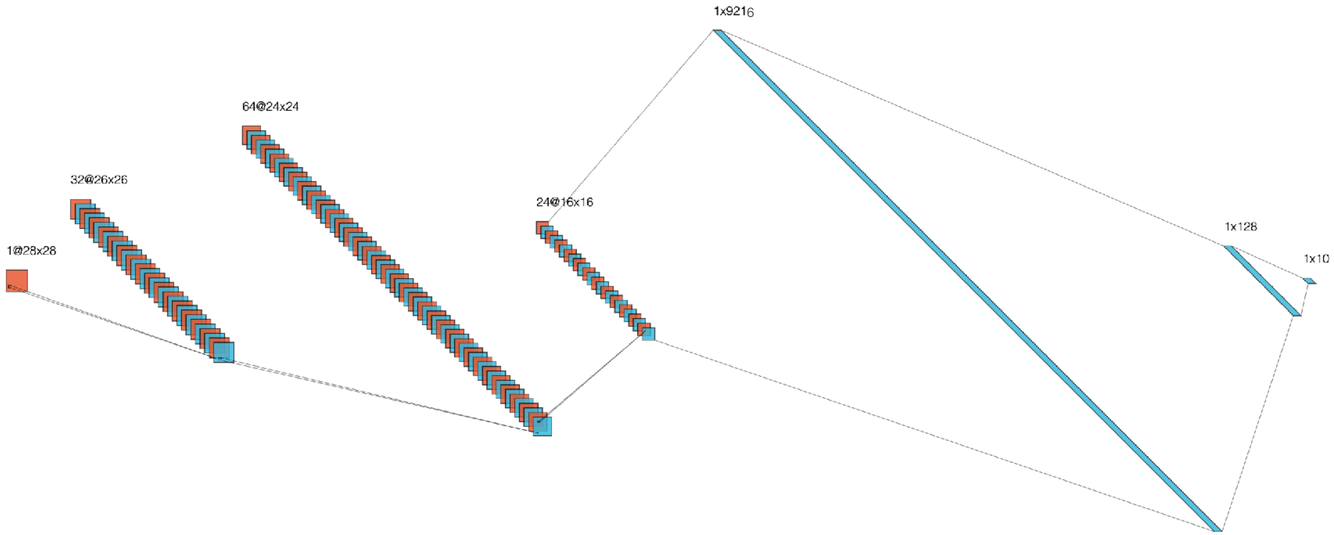
Number of Kernel/Filters and Number of Conv Layers

- Generally if your task is not too complex 2 or 3 layers would be a good starting point.
  - Complexity is determined by:
    - Number of classes
    - Similarity of classes
    - Variation and deformation in the dataset
- By convention the number of channels grow as we progress through the network (e.g. 32 -> 64 -> 128)
- Note you may see this reversed in some types of networks (like U-Net or some Siamese Network architectures).

A Basic Guide to Designing Effective CNNs

Number of Nodes in our Dense Fully Connected Layers

- The number of hidden neurons should be **between** the **size of the input layer** and the **size of the output** layer.
- The number of hidden neurons should be **2/3 the size** of the **input layer**, **plus** the size of the **output layer.**
- The number of hidden neurons should be **less than twice the size of the input layer.**



- According to Steffen B Petersen · Aalborg University
- In order to secure the ability of the network to generalise the number of nodes has to be kept as low as possible. If you have a large excess of nodes, you network becomes a memory bank that can recall the training set to perfection, but does not perform well on samples that was not part of the training set.

A Basic Guide to Designing Effective CNNs

Padding

- It’s generally good practice to use padding (zero padding) so that we retain information at the borders and don’t downsample our image too much as we propagate through the network

A Basic Guide to Designing Effective CNNs

Stride

- Normally we generally use Stride of 1, any greater would reduce the input size too much so generally we stick to 1

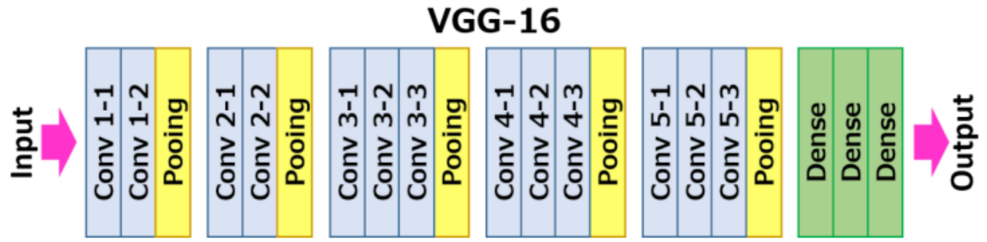
A Basic Guide to Designing Effective CNNs

Pooling and Number of Channels

- Normally we use Max Pooling, other pooling methods aren't as well suited to most CNNs
- Number of Channels choice is simple in theory, 3 for colour (RGB) or 1 for grayscale.
- However, the choice of whether to change your input image to grayscale depends on whether you think colour to adds key information in distinguishing your class

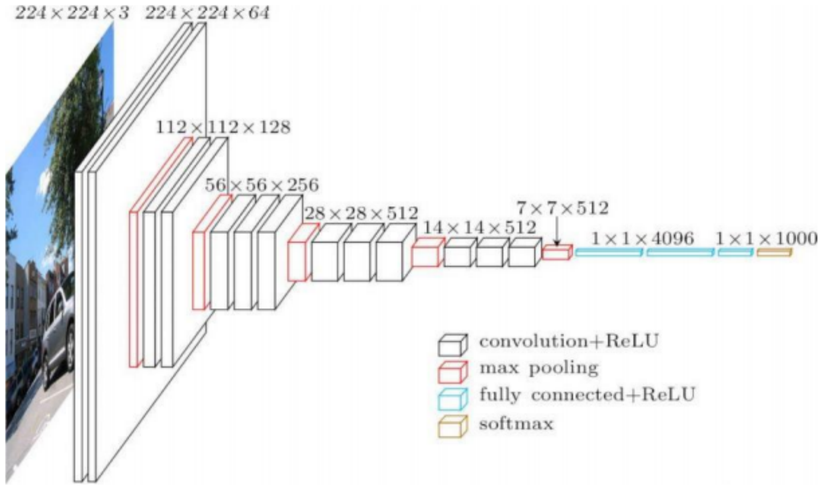
Classic CNN Architectures

- LeNET
- AlexNet
- VGG16



The Architecture

The architecture depicted below is VGG16.



Modern CNN Architectures

- ResNet
- InceptionV3
- Xception
- MobileNet
- SqueezeNet
- DenseNet
- EfficientNet
- GoogleNet

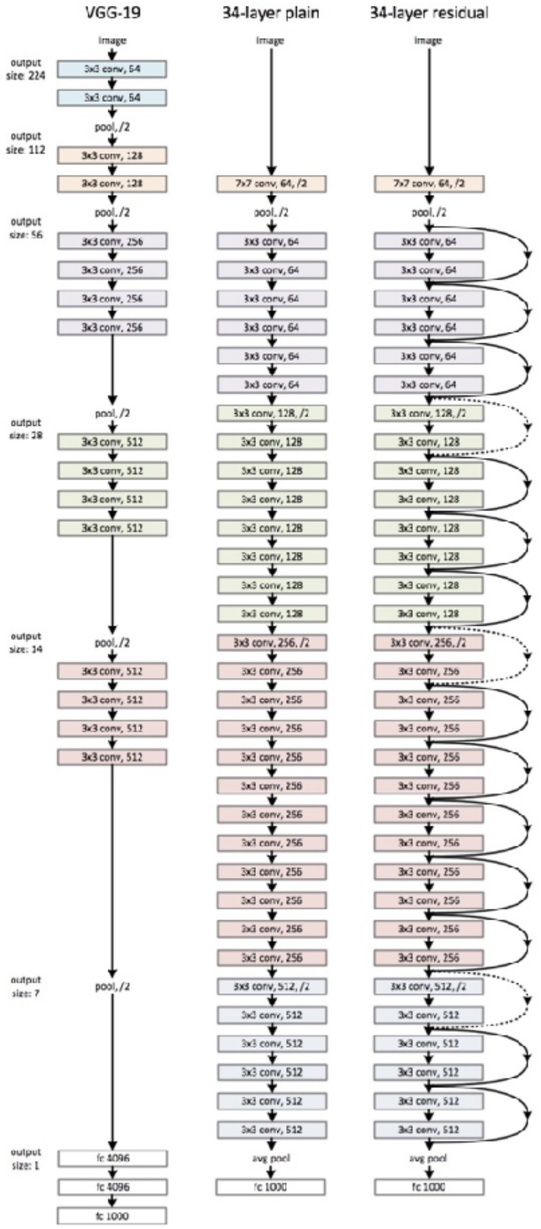
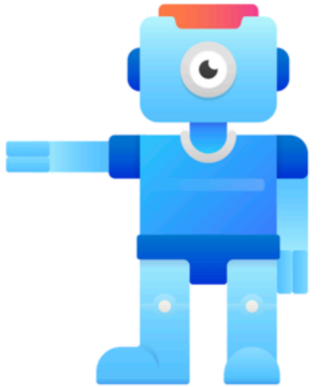


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

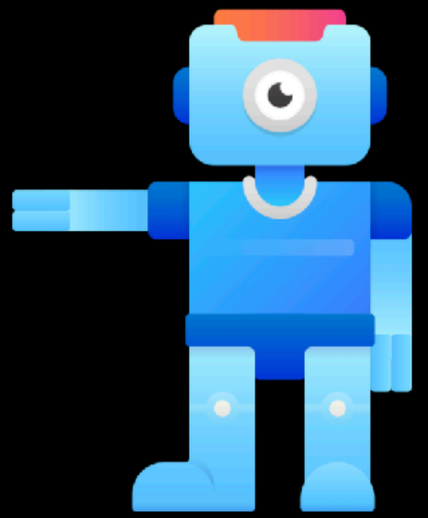


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

A look at the Evolution of CNNs



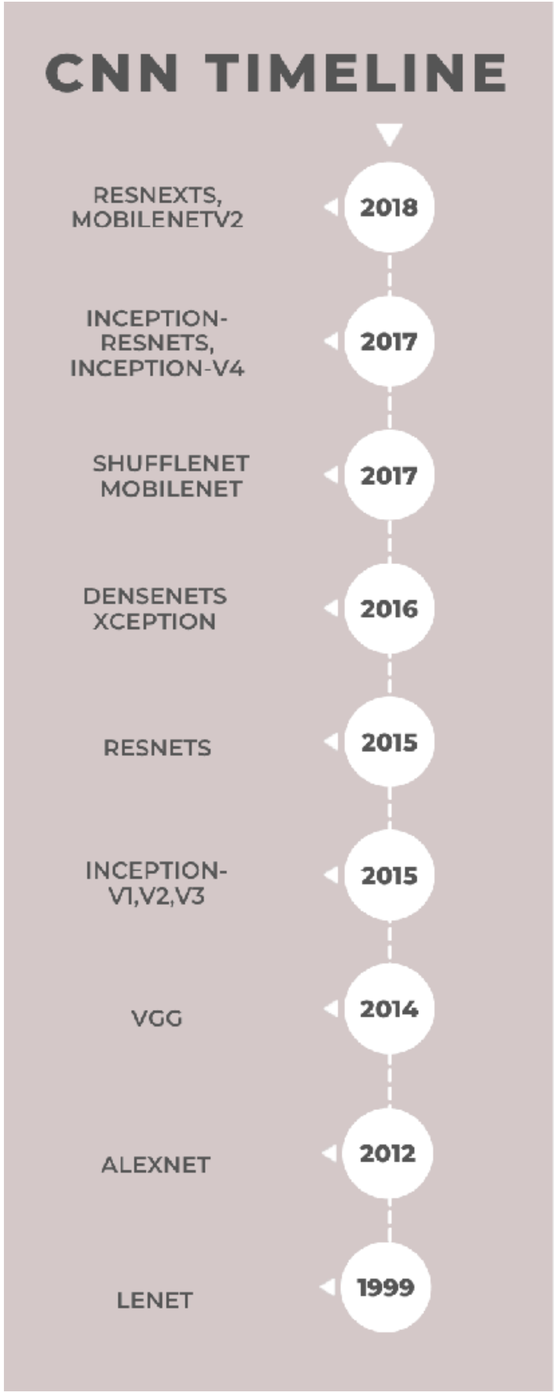
MODERN COMPUTER VISION

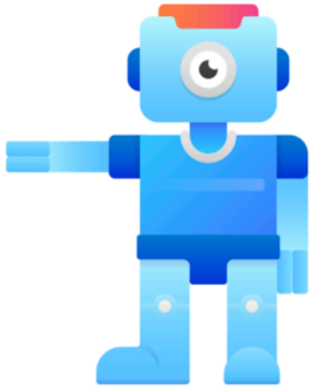
BY RAJEEV RATAN

CNN History

A brief outline of the evolution of CNN Research

CNN History


CNN TIMELINE
RESNEXTS, MOBILENETV2
2018
INCEPTION-RESNETS, INCEPTION-V4
2017
SHUFFLENET MOBILENET
2017
DENSENETS XCEPTION
2016
RESNETS
2015
INCEPTION-V1,V2,V3
2015
VGG
2014
ALEXNET
2012
LENET
1999

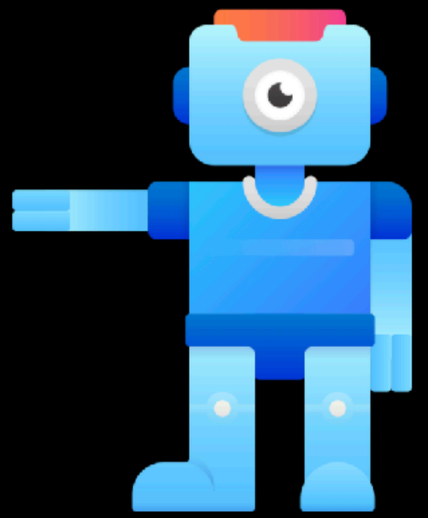


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

A look at LeNet



MODERN COMPUTER VISION

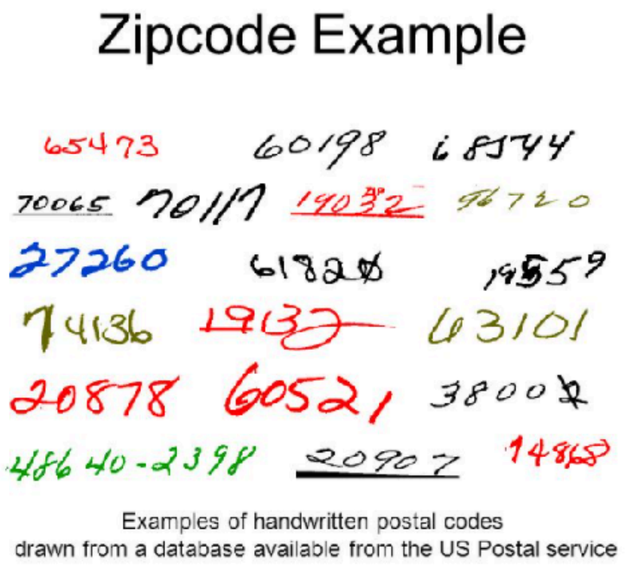
BY RAJEEV RATAN

LeNet

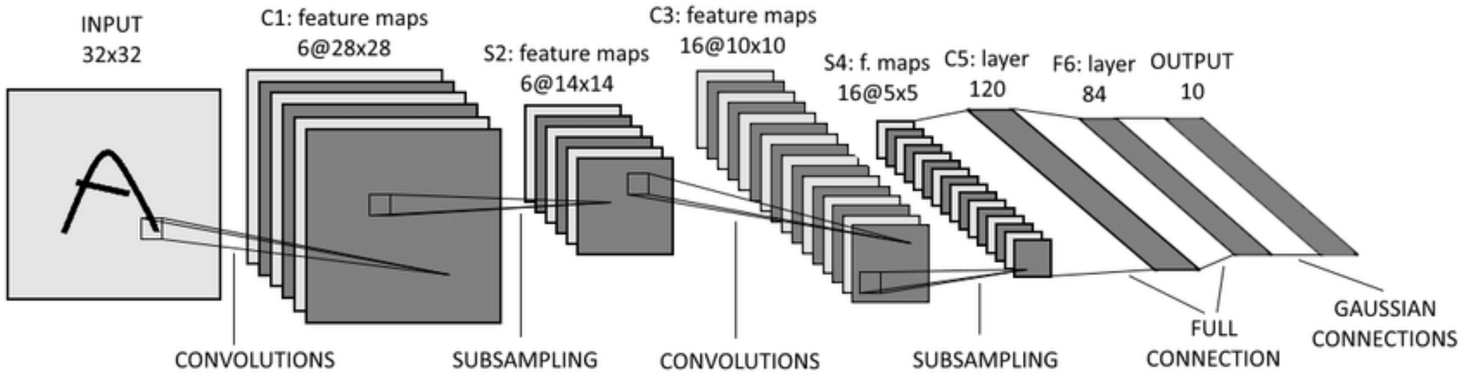
An overview of the LeNet CNN

LeNet

- LeNet was introduced in 1995 by LeCun
- It was developed for handwritten digit recognition for US zip codes (MNIST)



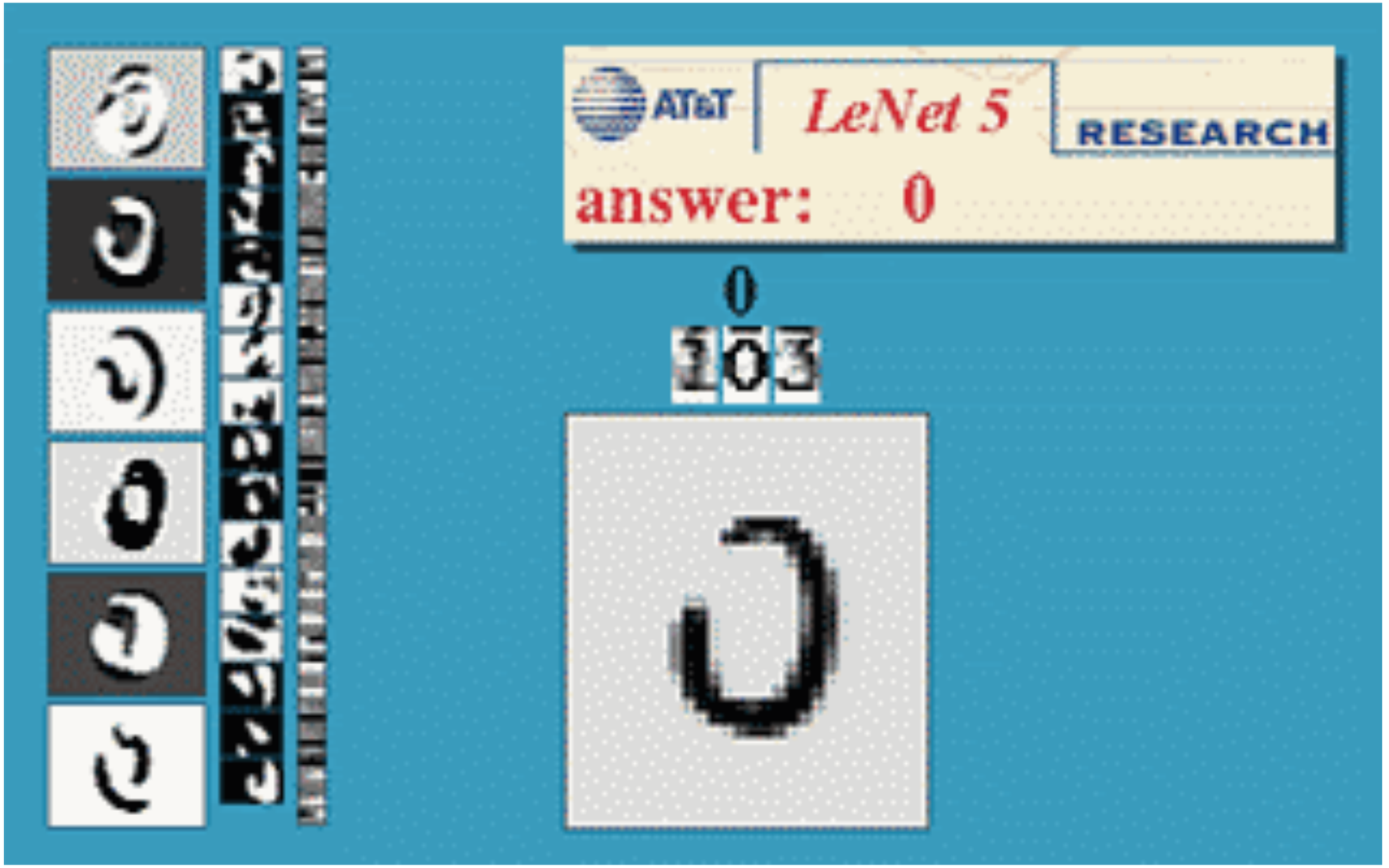
LeNet Architecture

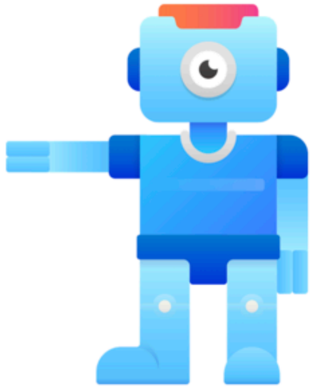


- Two convolution layers with Max Pool (subsampling of 2x2 kernel and stride 2)
- First Conv layer had 6 filters/kernels and size 5x5 (stride 1, padding 0)
- Second Conv layer had 16 filters/kernels of size 5x5 (stride 1, padding 0)
- After the 2nd Max Pool, we flatten the 5x5x16 layer into 400 neurons and connect it to the first FC layer of 120 neurons, then another FC layer of 84 neurons before reaching the final output layer (10 classes)
- LeNet can achieve 99.3% Accuracy on MNIST

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

LeNet Animation



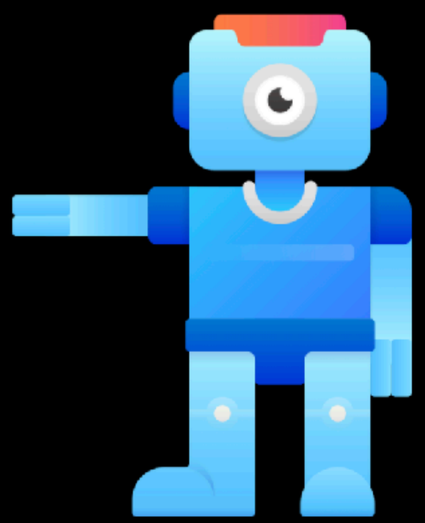


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

A look at AlexNet



MODERN COMPUTER VISION

BY RAJEEV RATAN

AlexNet

An overview of the AlexNet CNN

AlexNet

- AlexNet was introduced in 2012 by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton from the University of Toronto.
- It was the ILSVRX winner in 2012.
- It contained 8 layers with the first five being Convolutional Layers and the last 3 being FC layers.
- It has over 60 Million parameters and was trained on two GPUs for over a week!

IMAGENET **Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)**

Held in conjunction with PASCAL Visual Object Classes Challenge 2012 (VOC2012)

Back to Main page

All results

- Task 1 (classification)
- Task 2 (localization)
- Task 3 (fine-grained classification)
- Team information and abstracts

Task 1

Team name	Filename	Error (5 guesses)	Description
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422	Using only supplied training data
ISI	pred_FVs_wLACs_weighted.txt	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
ISI	pred_FVs_weighted.txt	0.26602	Weighted sum of scores from classifiers using each FV.
ISI	pred_FVs_summed.txt	0.26646	Naive sum of scores from classifiers using each FV.
ISI	pred_FVs_wLACs_summed.txt	0.26952	Naive sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.

http://www.image-net.org/challenges/LSVRC/2012/results.html

AlexNet Architecture

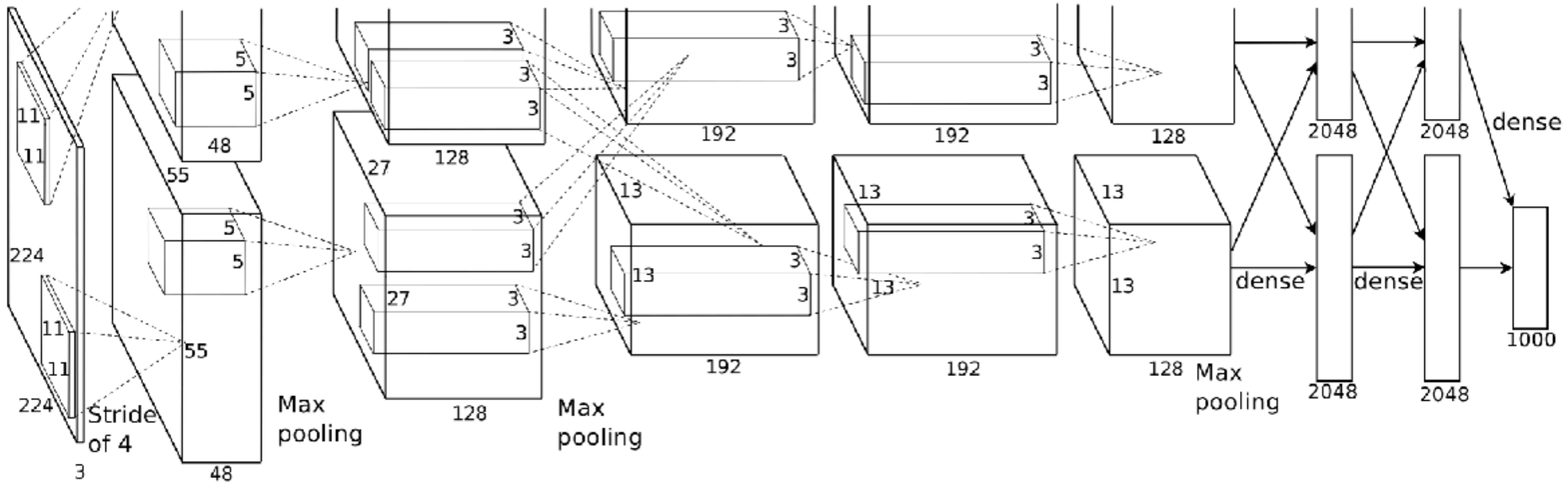
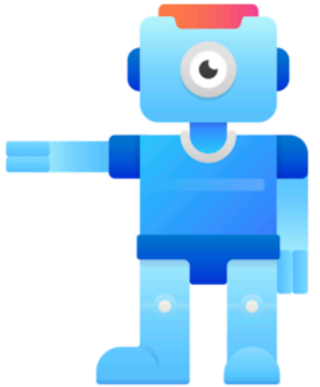


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

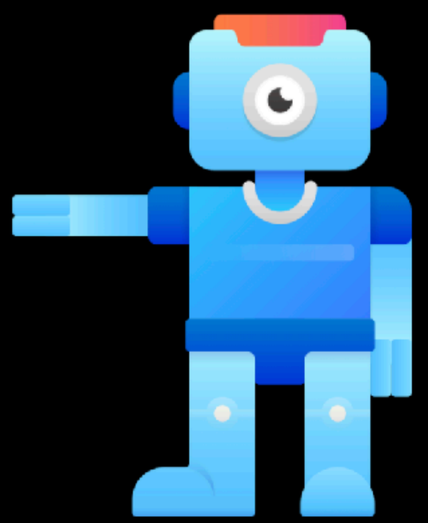


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

A look at VGGNet



MODERN COMPUTER VISION

BY RAJEEV RATAN

VGGNet

An overview of the VGGNet CNN

VGGNet

- VGGNet was first introduced in 2014 by Oxford University Researchers Karen Simonyan and Andrew Zisserman.
- It achieved 92.7% top-5 Accuracy in ImageNet (1000 classes).
- VGG16 has 13 Conv Layers with 3 FC Layers
- VGG19 has 16 Conv Layers with 3 FC Layers

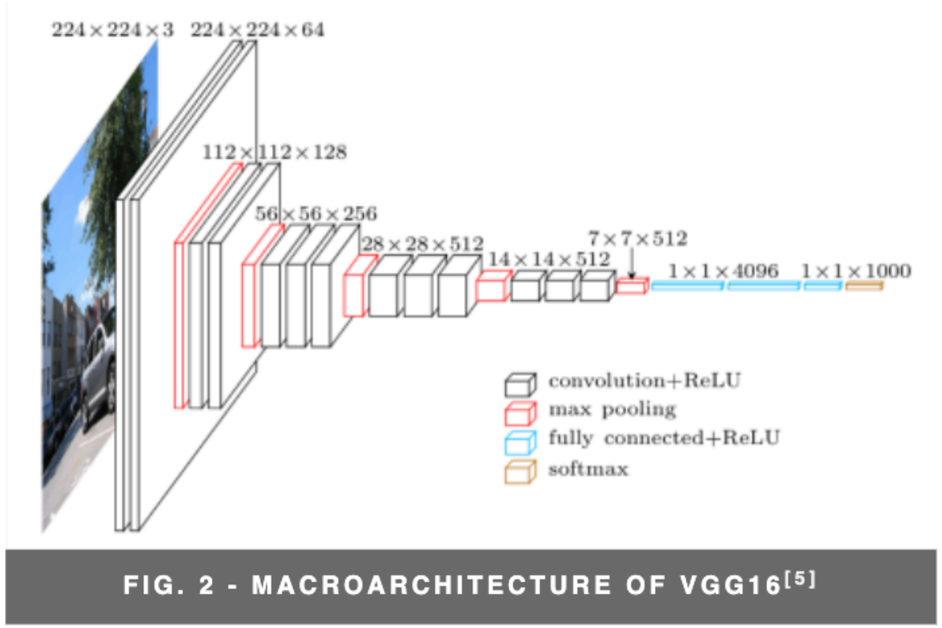
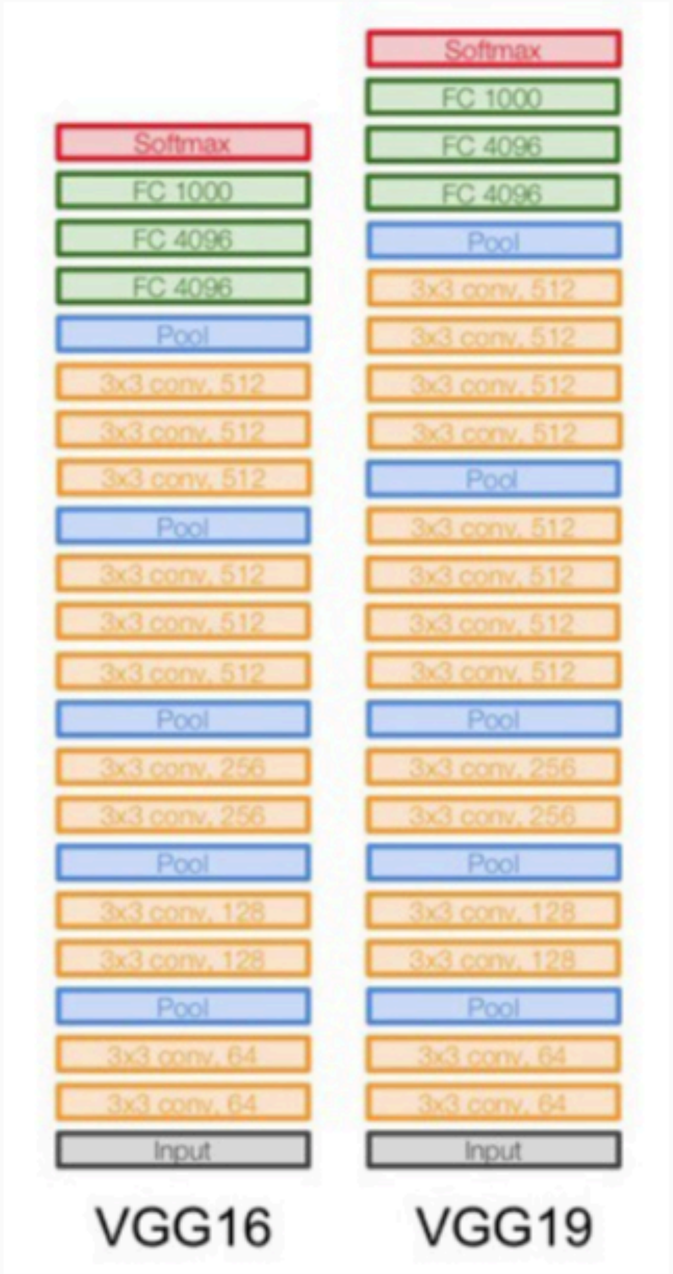


FIG. 2 - MACROARCHITECTURE OF VGG16[5]

https://www.cs.toronto.edu/~frossard/post/vgg16/

VGGNet

- VGG follows what we will now call a ‘Classical CNN’ approach where we have sequences of Conv Layers followed by a pooling layer.
- This is then repeated with increasing number of Feature Maps/Filters until we connect it to multiple FC Layers which then outputs our class probabilities using a softmax layer.



https://towardsdatascience.com/architecture-comparison-of-alexnet-vggnet-resnet-inception-densenet-beb8b116866d

VGGNet - Parameter Heavy

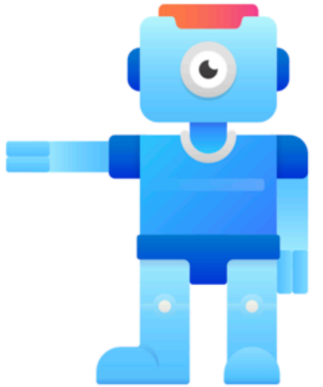
Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

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					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

https://arxiv.org/pdf/1409.1556.pdf

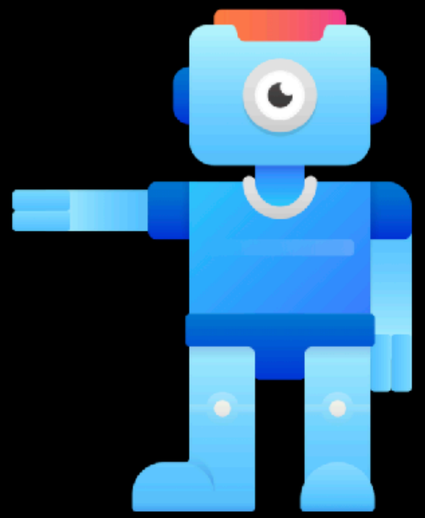


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

A look at ResNets



MODERN COMPUTER VISION

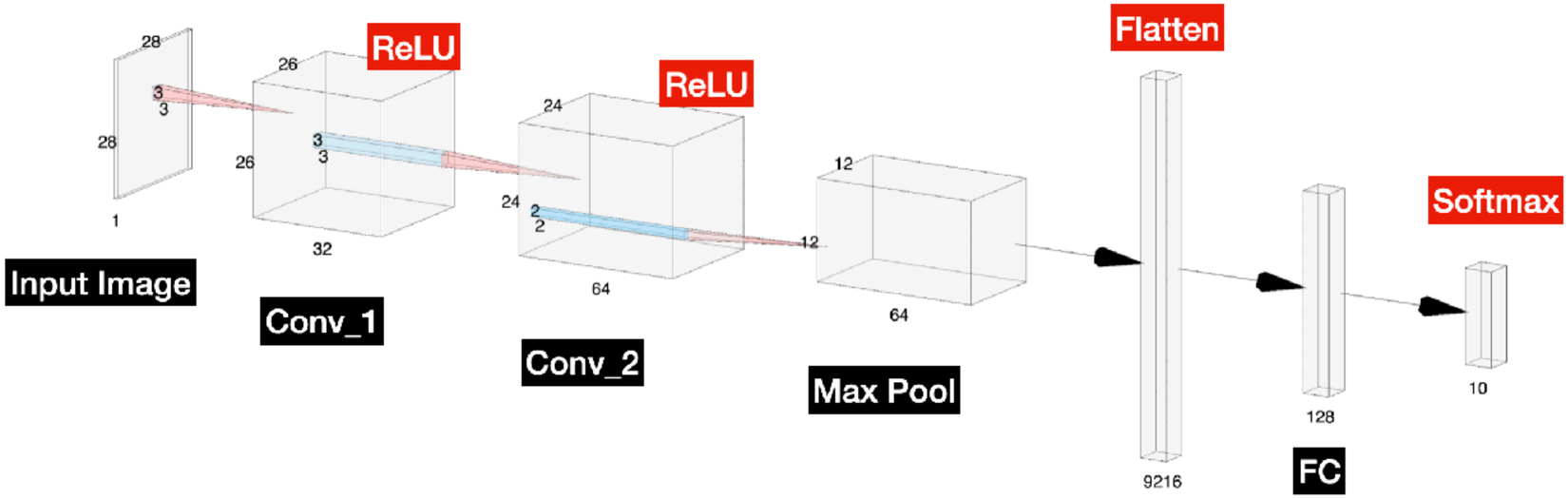
BY RAJEEV RATAN

ResNets

An overview of the ResNet or Residual Neural Networks

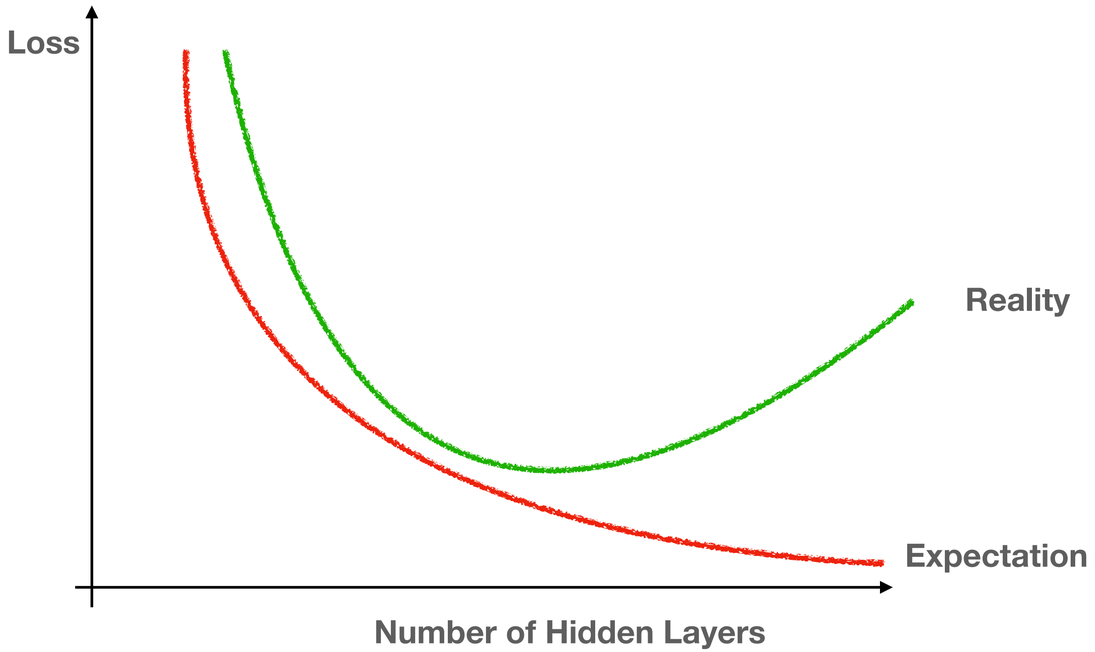
Plain or Classical CNNs

Look like this…



- A linear sequential sequence of linear operations
- Problems?

Classical CNNs Give Worse Performance if too Deep

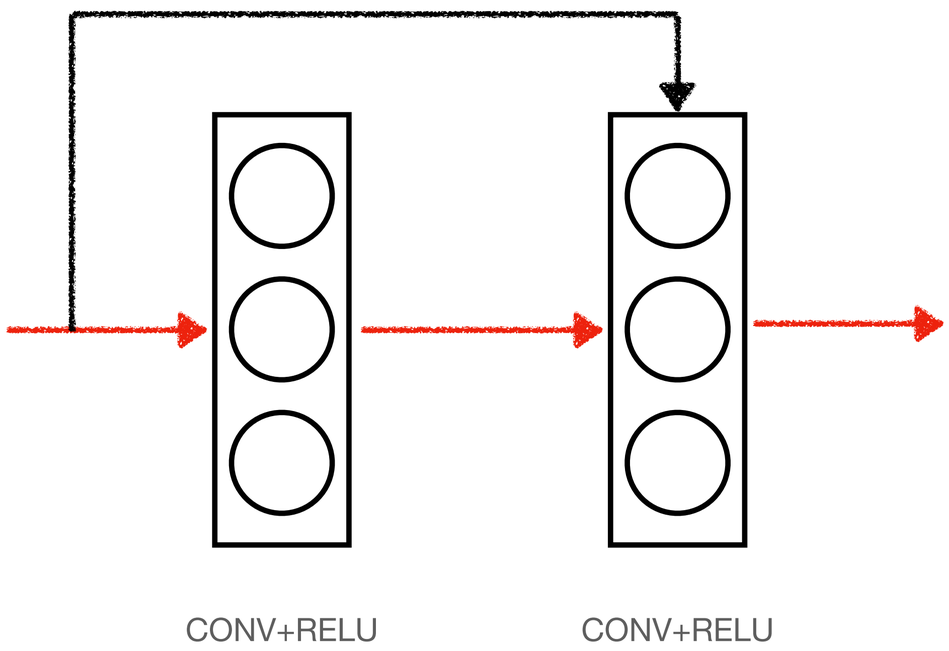


- In reality, a classical CNN architecture has lots of problems training if layers are very deep
- Often performance gets worse when number layers are used

Very Deep CNNs Experience Exploding or Vanishing Gradients

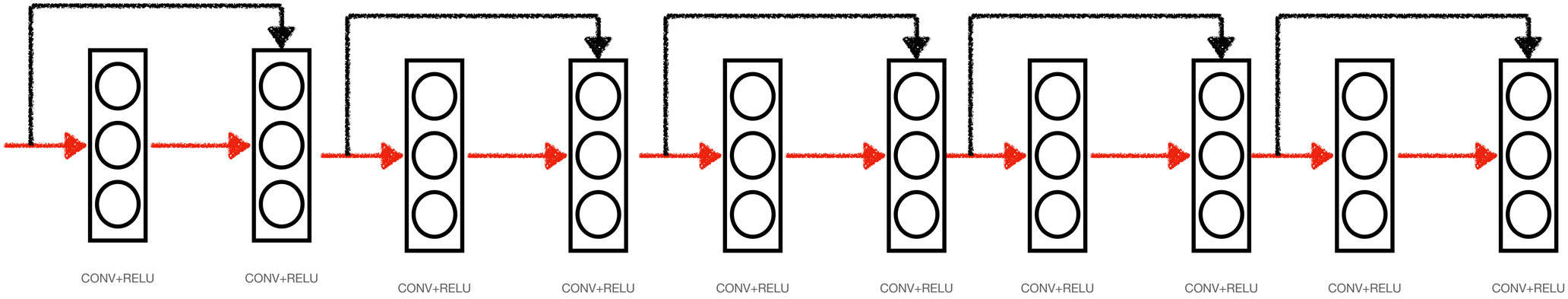
- In deep networks with N layers, N derivatives must be multiplied together to perform our gradient updates
- If a derivative is large, gradients increase exponentially or **“explode”**
- Likewise, if derivatives are small, they decrease exponentially or **“vanish”**

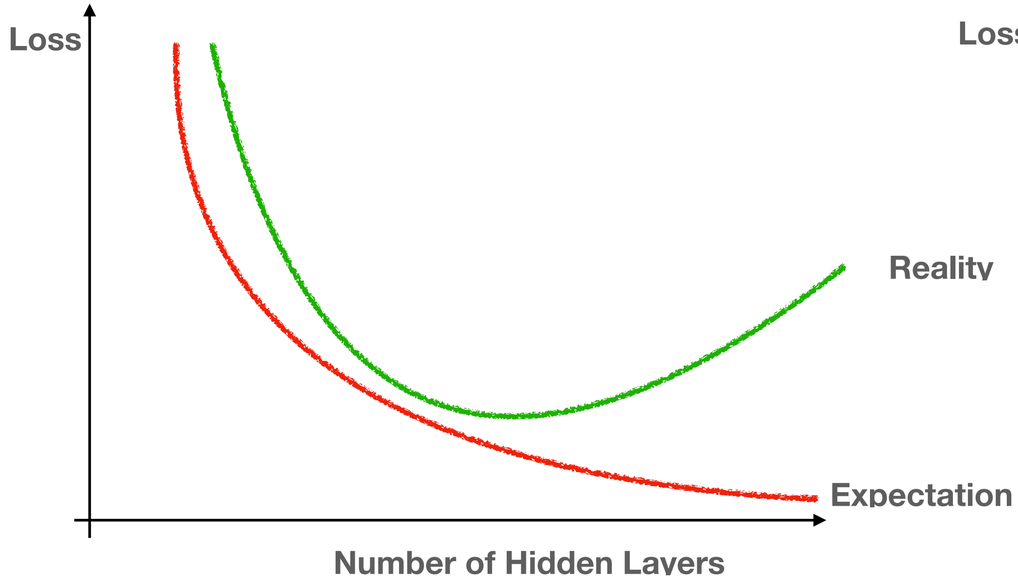
ResNet of Residual Networks Help Solve This

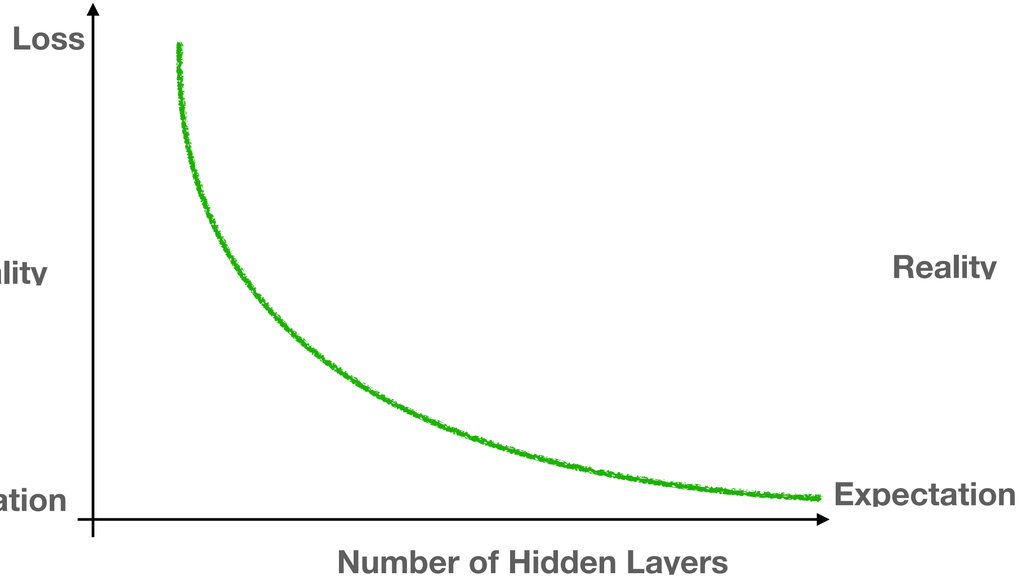


- We ‘Short Circuit’ the network by connecting the input to the previous layer to the layer ahead.

ResNet of Residual Networks Help Solve This

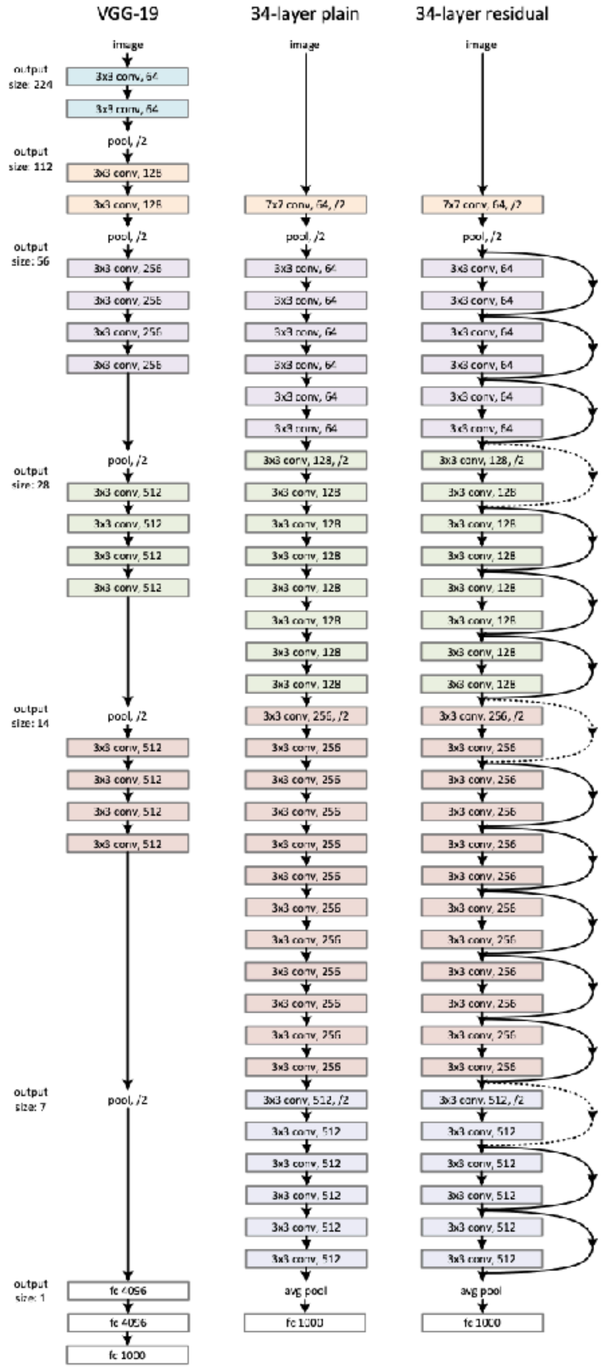


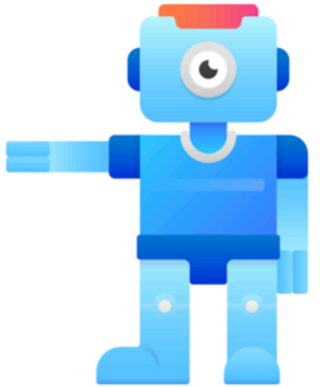




ResNets

- ResNets were introduced in 2015 by Microsoft Researchers in paper titled, Deep Residual Learning for Image Recognition from He et al.



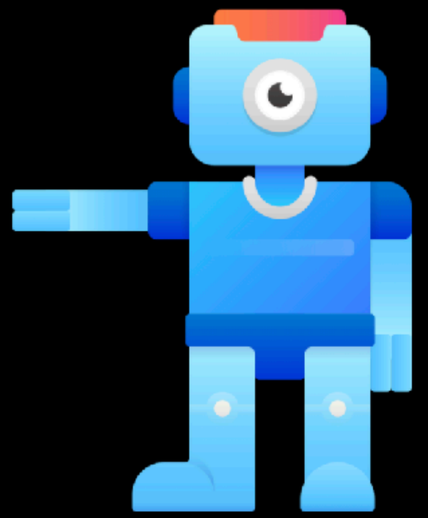


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

Why to ResNets Work?



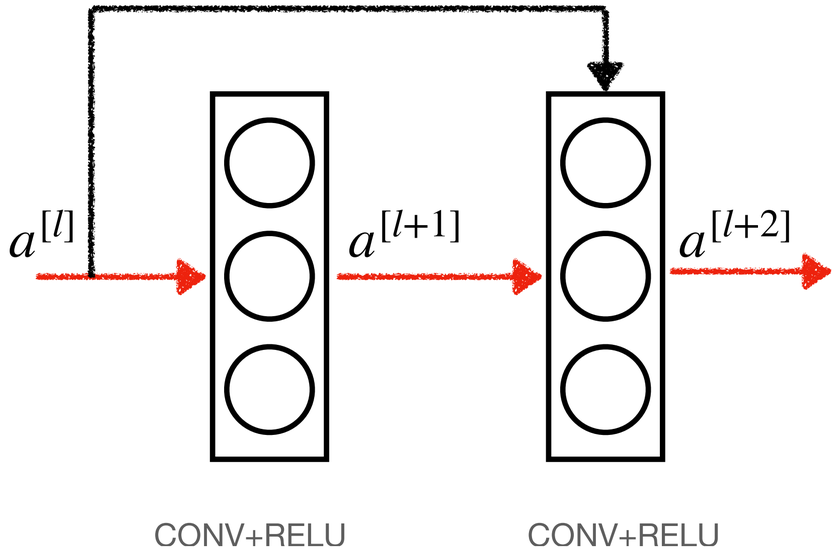
MODERN COMPUTER VISION

BY RAJEEV RATAN

Why Do ResNets Work?

How ResNets Solve the Vanishing Gradient Problem

ResNet Mathematics



$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]}$$ First Linear Operation

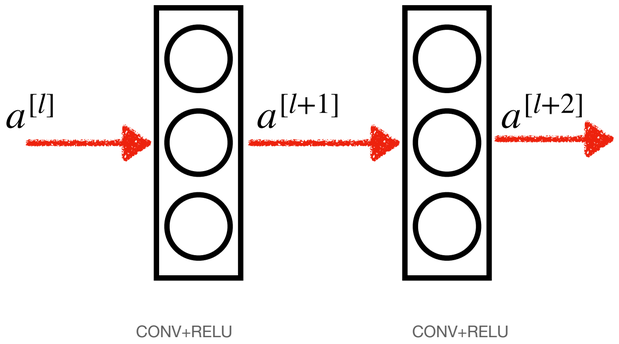
$$a^{[l+1]} = g(z^{[l+1]})$$ After ReLU operation

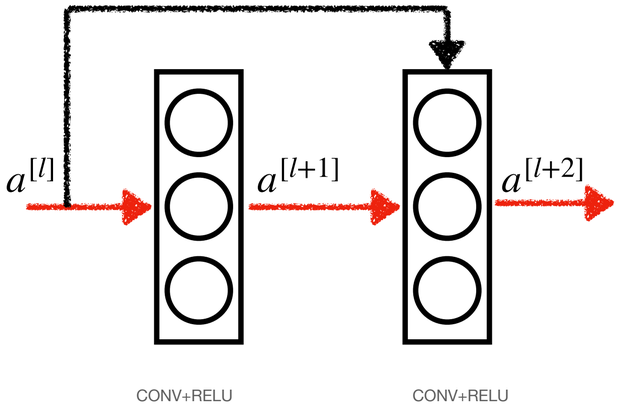
$$z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}$$ Second Linear Operation

$$a^{[l+2]} = g(z^{[l+2]})$$ Output without Short Circuit

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$ Output **with** Short Circuit

ResNet Mathematics





Output **without** Short Circuit

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$$a^{[l+2]} = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]})$$

Output **with** Short Circuit

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

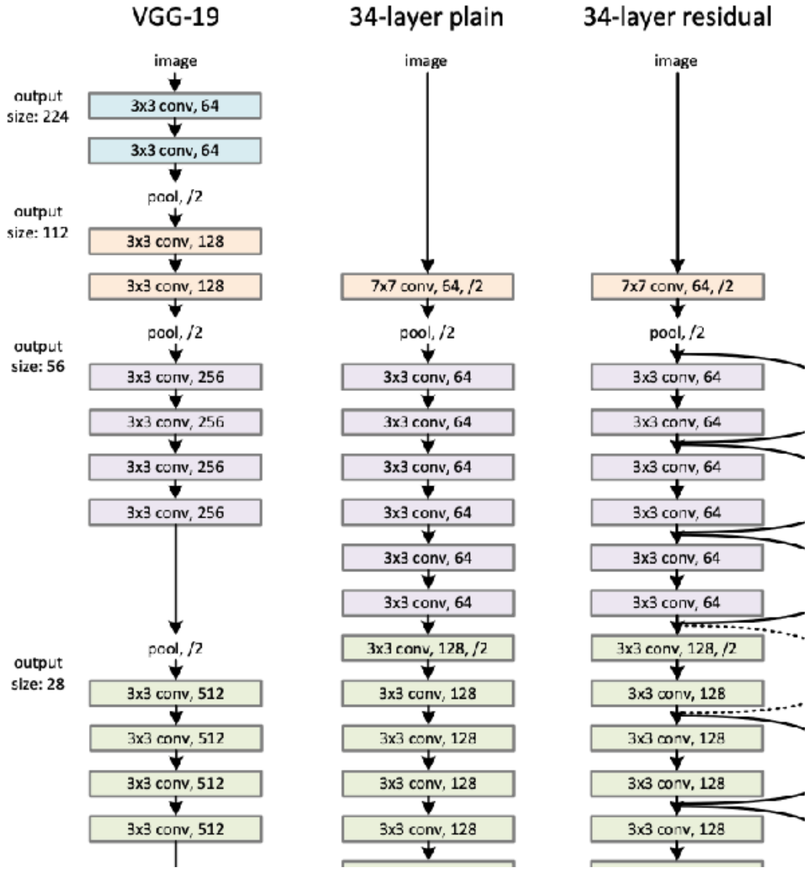
$$a^{[l+2]} = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

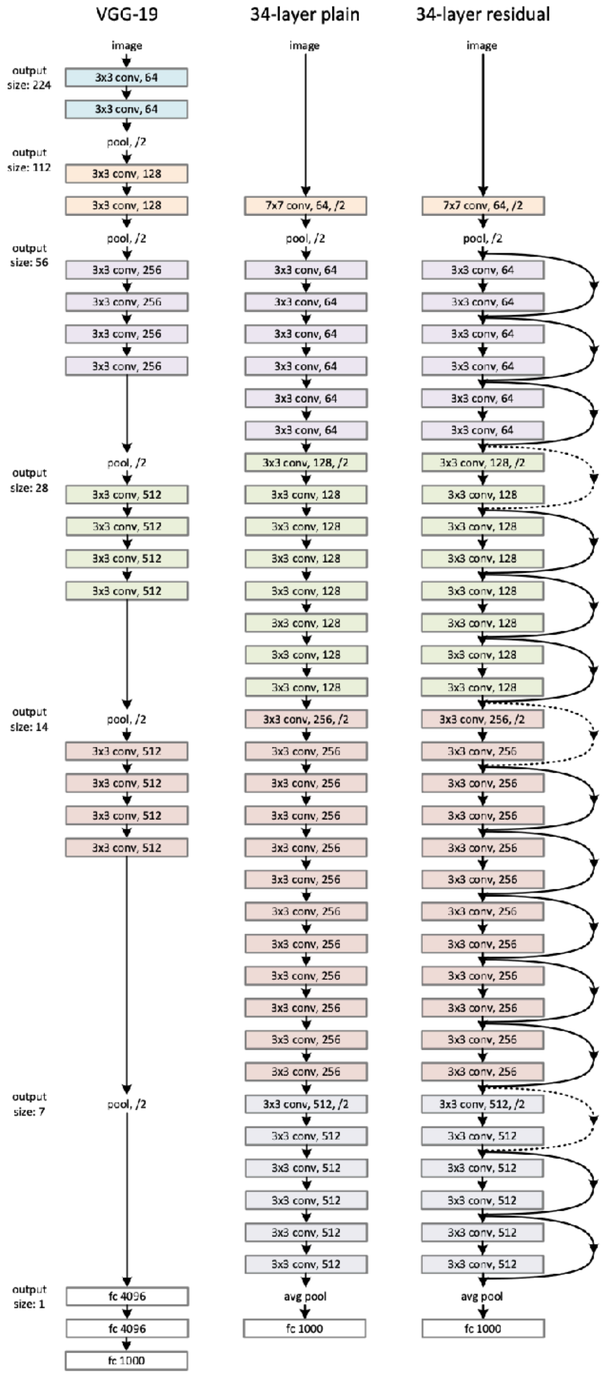
- If W or b is small or near 0, $a^{[l+2]} = g(a^{[l]})$

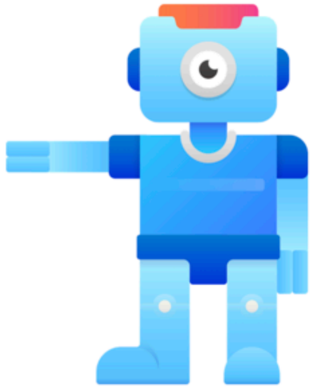
- This means it allows the our Network to learn the Identity function, thus solving the **vanishing gradient** problem

ResNet34 and ResNet50 Architectures

- ResNet34 and ResNet50 features several consecutive Conv layers of 3x3 with feature maps (64, 128, 256, 512) with bypasses every 2 Convolutions.
- Their output dimensions remain constant (padding same, stride =1)
- ResNet was built by several stacked residual units and developed with many different numbers of layers: 18, 34, 50, 101, 152, and 1202.





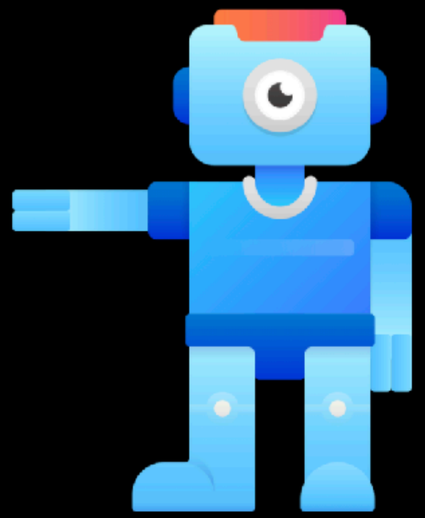


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

Take a look at MobileNet



MODERN COMPUTER VISION

BY RAJEEV RATAN

MobileNet

MobileNet a fast interference CNN modle

MobileNet

- MobileNet was a CNN architecture developed by Google in 2017.
  - https://arxiv.org/pdf/1704.04861.pdf

- MobileNet was designed to be an efficient lightweight CNN that could be used in embedded devices and mobile phones.
- Good CNNs are large (weights are 100mb+)
- Relatively slow on inference (i.e. forward propagation)

MobileNet Use Cases

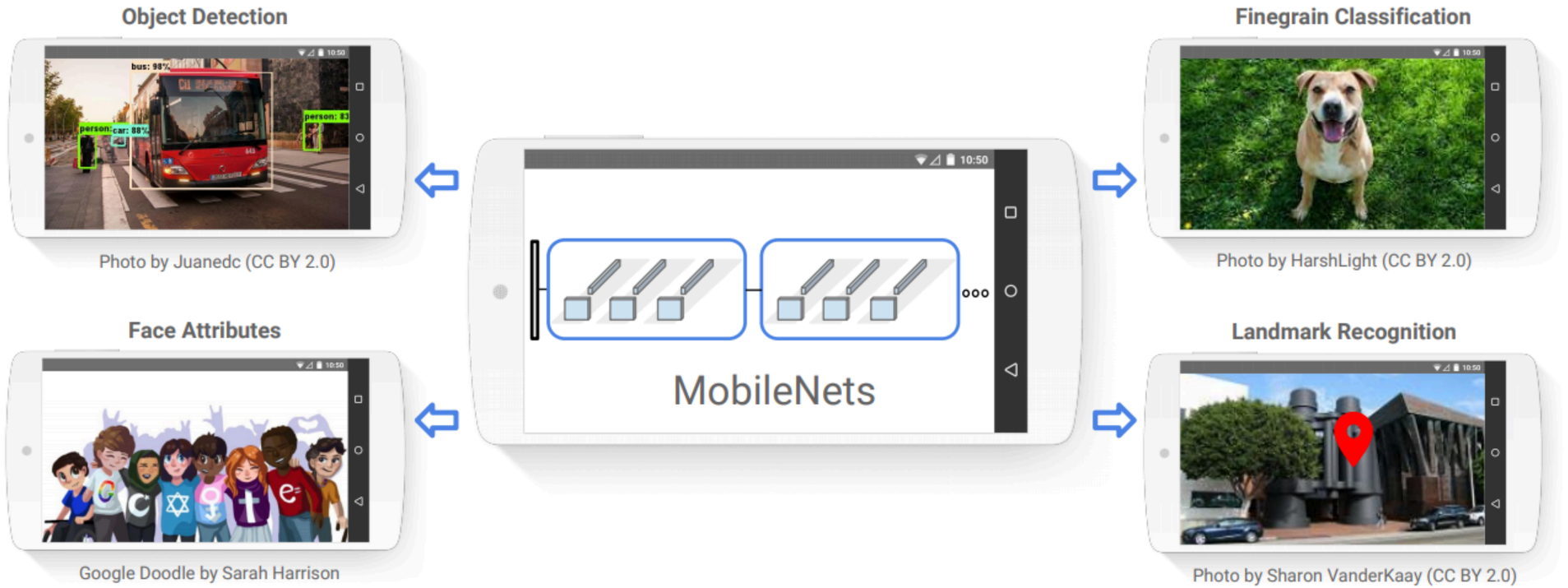


Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

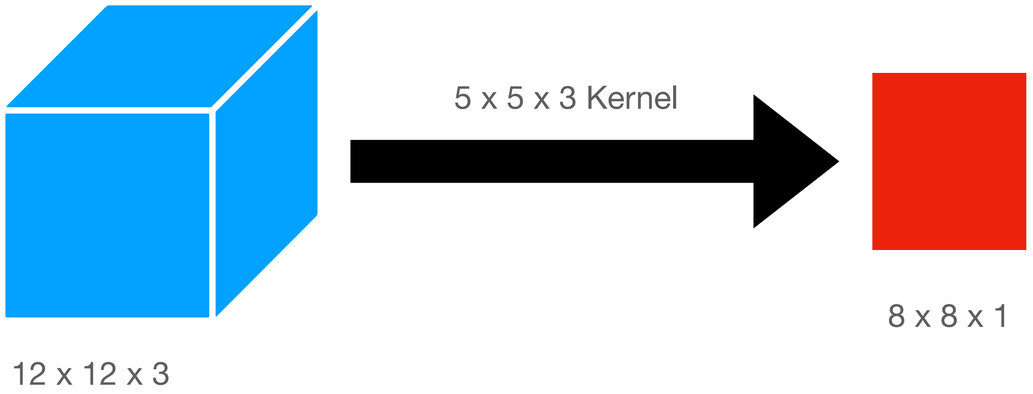
https://arxiv.org/pdf/1704.04861.pdf

Making CNNs for Mobile/Embedded Devices

- Mobile or embedded systems typically have low computational power as they are made to be cheap and power efficient.
- Using CNNs on these devices required:
  - Training smaller models
  - Compressing existing models via methods such as Pruning, Distillation or low bit networks.

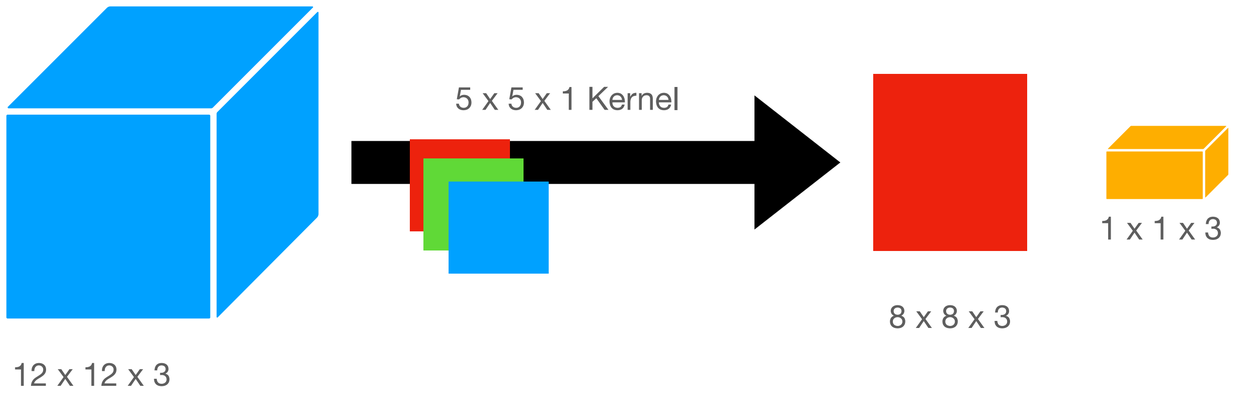
- MobileNet achieved the goal of being a good mobile phone model by using:
  - Depthwise Separable Convolutions
  - Two Hyper-Parameters

Convolution Operations in Regular CNNs



- Input image of 12x12x3 convolved with a kernel of 5x5x3 produces a feature map of size 8x8x1
- Given a stride of 1, we have to perform 5 * 5 * 3 * 64 operations
- If we had 128 filters this ends up being
- 75 * 64 * 128 = 614,400

Depth Wise Convolutions



5 x 5 x 3 * 64 = 75 * 64 = 4,800

3 x 64 x 128 = 24,576

4800 + 24,576 = 29,376 Operations

- We use 3 Filters of 5x5x1 and multiply each with a single channel from our input image (12x12x1)
- This gives us a 8x8x3 output
- **Pointwise** Convolutions are then used to get the same output shape
  - We then multiple by our output by a 1x1x3 layer
  - This is multiplied 64 times, this now gives us a 8x8x1 output
  - This is roughly 20X less Operations

Two Hyper Parameters

- MobileNet also features two hyper parameters that effectively reduce the size of the model.
  - **Width Multiplier** - This thins the model at each layer
  - **Resolution Multiplier** - Reduces the input image size and thus reduces the internal representation of every subsequent layer.

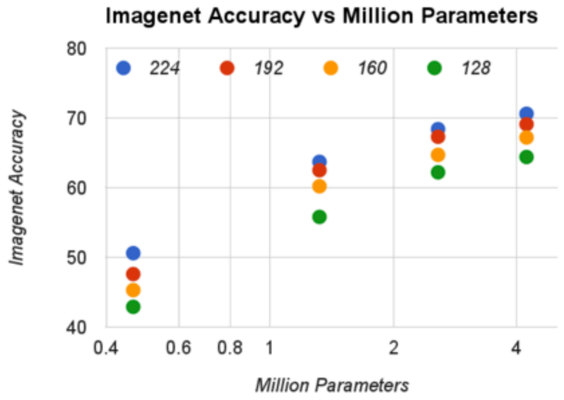


Figure 5. This figure shows the trade off between the number of parameters and accuracy on the ImageNet benchmark. The colors encode input resolutions. The number of parameters do not vary based on the input resolution.

Table 8. MobileNet Comparison to Popular Models

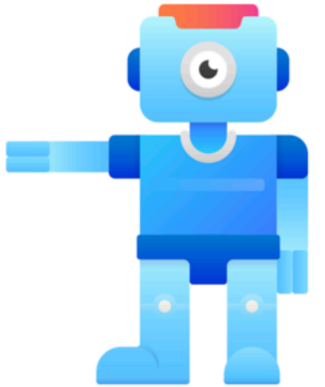
Model	ImageNet Accuracy	Million Parameters
1.0 MobileNet-224	70.6%	4.2
GoogleNet	69.8%	6.8
VGG 16	71.5%	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Parameters
0.50 MobileNet-160	60.2%	1.32
Squeezenet	57.5%	1.25
AlexNet	57.2%	60

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Parameters
Inception V3 [18]	84%	23.2
1.0 MobileNet-224	83.3%	3.3
0.75 MobileNet-224	81.9%	1.9
1.0 MobileNet-192	81.9%	3.3
0.75 MobileNet-192	80.5%	1.9

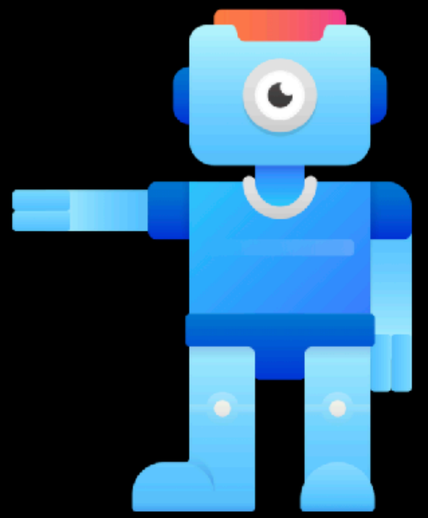


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

Take a look at Inception Network



MODERN COMPUTER VISION

BY RAJEEV RATAN

Inception Network

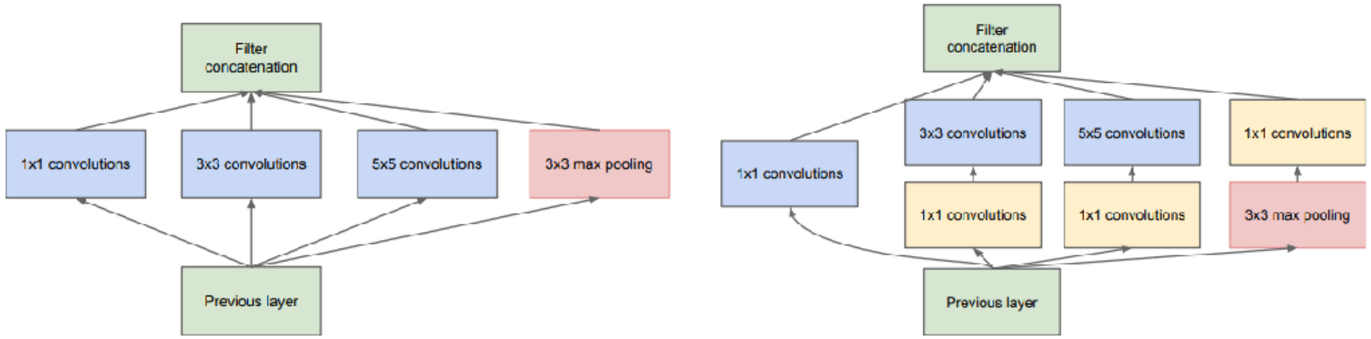
We explore the novel Inception Network or GoogleLeNet Architecture

Inception Network Motivation

- As you've seen there's a lot of parameter tweaking involved in CNNs
- Filter Sizes, stride, depth, padding, FC layers etc.
- Inception aimed to solve the Filter size selection problem.

Inception Network

- The Inception V1 (aka GoogleLeNet) Network was introduced by Google (Szegedy et al) in 2014
- It achieved state of the art performance in the ImageNet (ILSVRC14) Challenge

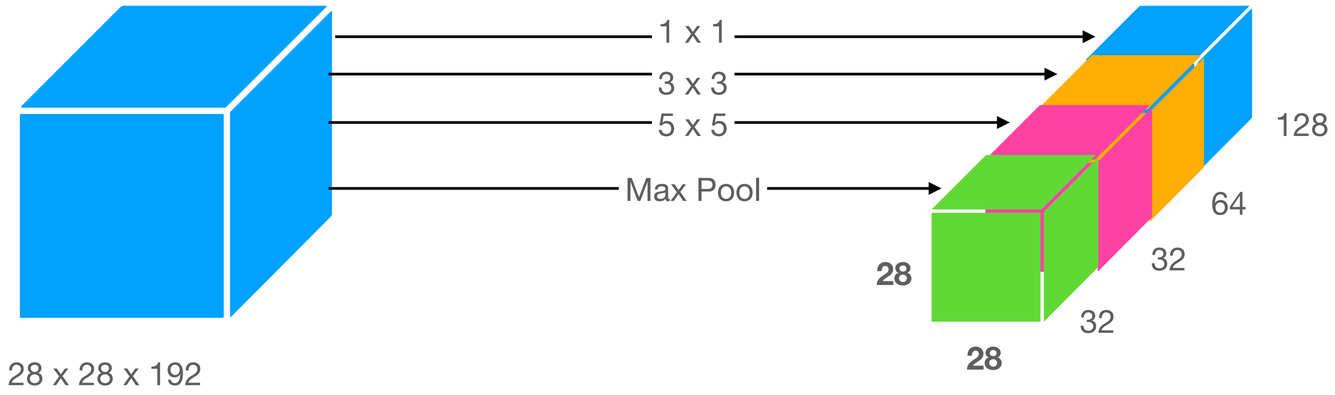


(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

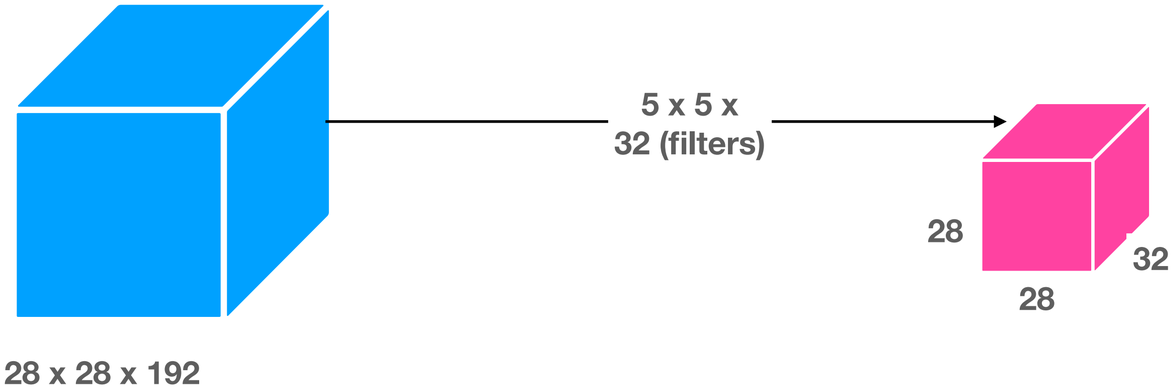
Inception Network

- The Inception Network allows us to use several different sized Conv filters



- Use ‘same’ padding and stride of 1 to maintain dimension size consistency
- We can now do all filter sizes and even a max pool and just stack them together
- This allows the network to learn a combination of high-level and low-level features

Inception Network Problem - Computation

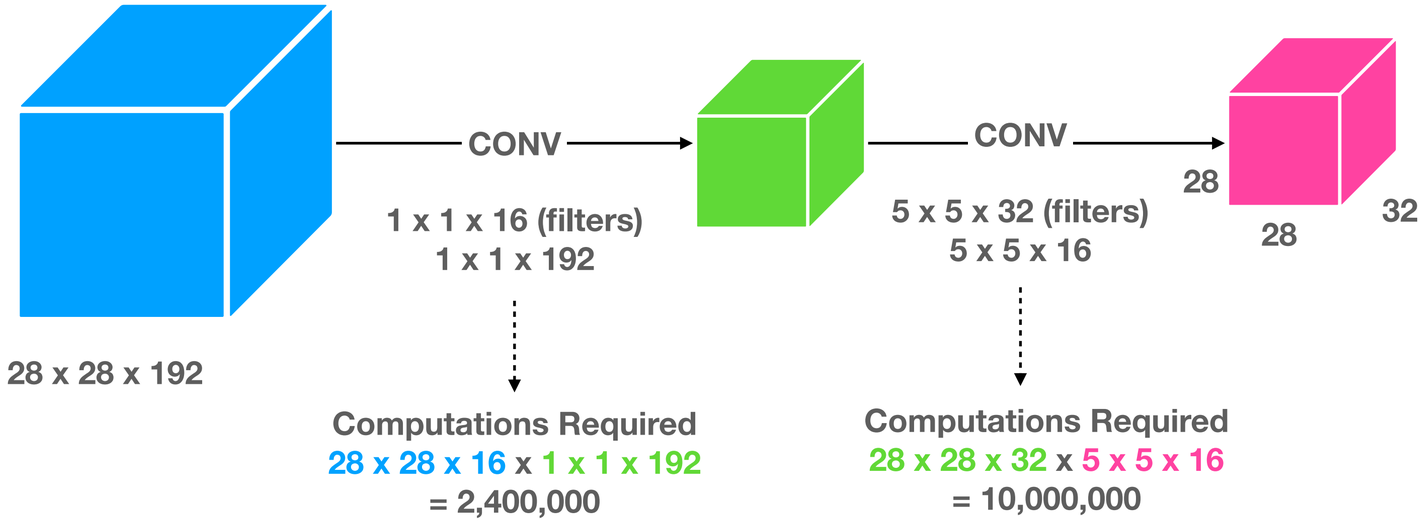


Computations Required
28 x 28 x 32 x 5 x 5 x 192
= 120,000,000

- We'll use 1x1 Convolutions to reduce the computation cost

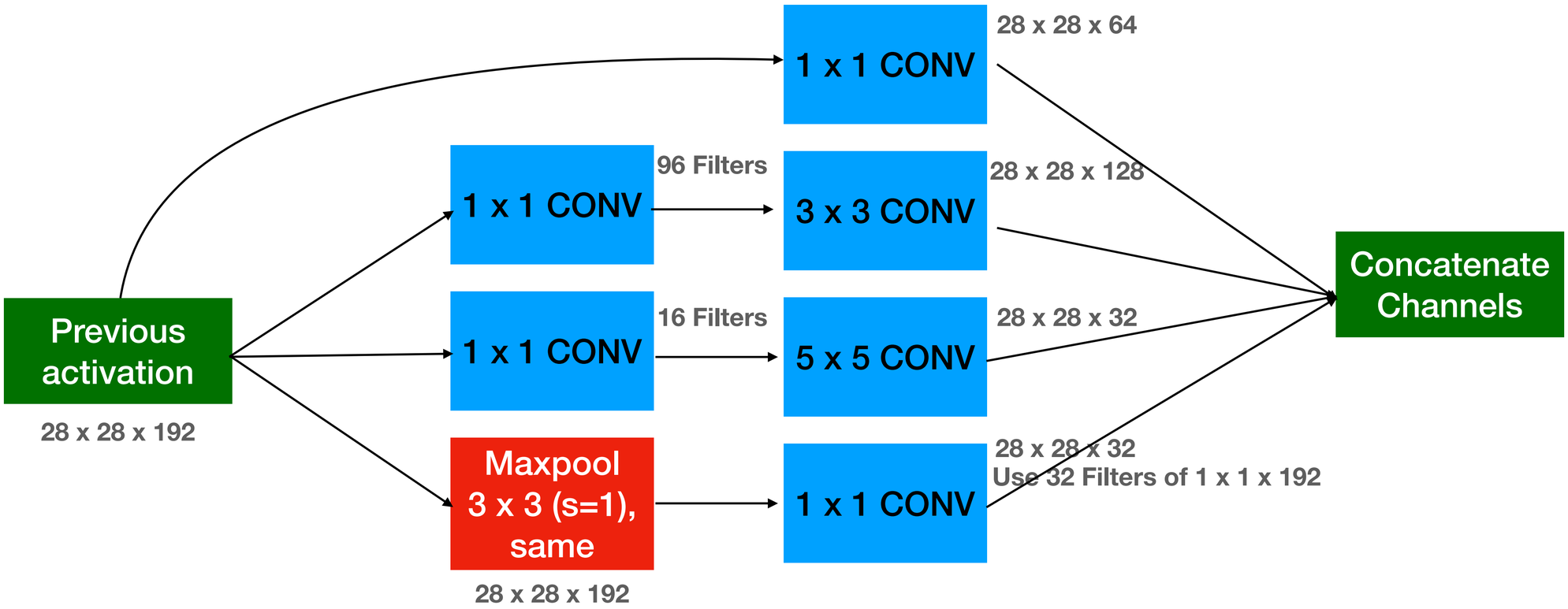
Intermediate Volumes using 1 x 1 Convolution

Using a Bottleneck Layer

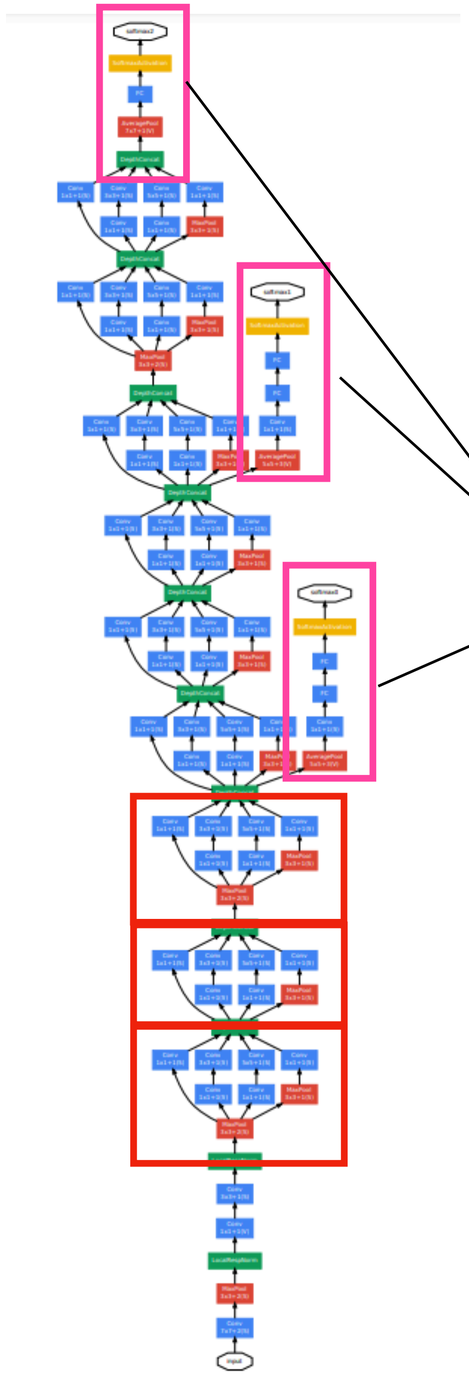


- We shrink the representation and then increase the size after
- This saves us 10X computation costs as it's now only 2.4M + 10M = **12.4M vs 120M** previously

Inception Block



Inception Design



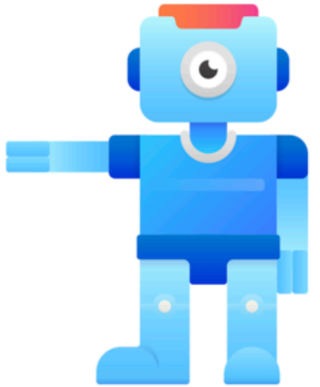
Side Branch early predictions
Has a Regularisation Effect

History of Inception Model Work

- **InceptionV1** - 2014 - Network in Network
  - https://arxiv.org/pdf/1312.4400v3.pdf
- **InceptionV2**- Going Deeper with Convolutions - 2014
  - https://arxiv.org/pdf/1409.4842v1.pdf
- **InceptionV3** - Rethinking the inception architecture for computer vision - 2015
  - https://arxiv.org/pdf/1512.00567v3.pdf
- **InceptionV4** - v4: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, Szegedy et al. (2016)
  - https://arxiv.org/pdf/1602.07261v2.pdf
- Good History here - https://nicolovaligi.com/history-inception-deep-learning-architecture.html

Fun Fact - Why is Called Inception?

- Meme was actually cited in the paper where the authors made reference to needing CNNs to be deeper, the Inception Network allows us to use deeper networks quite effectively.

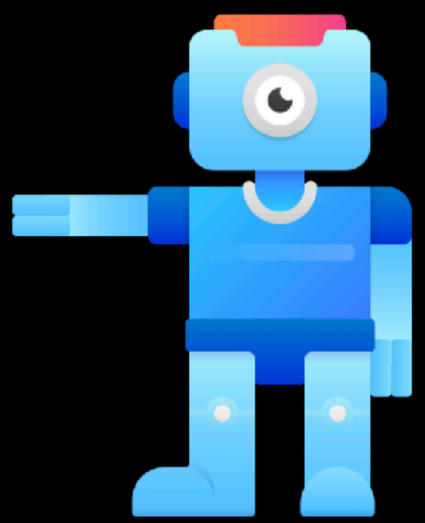


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

Let’s take a look at SqueezeNet



MODERN COMPUTER VISION

BY RAJEEV RATAN

SqueezeNet

A smaller CNN that maintains good accuracy

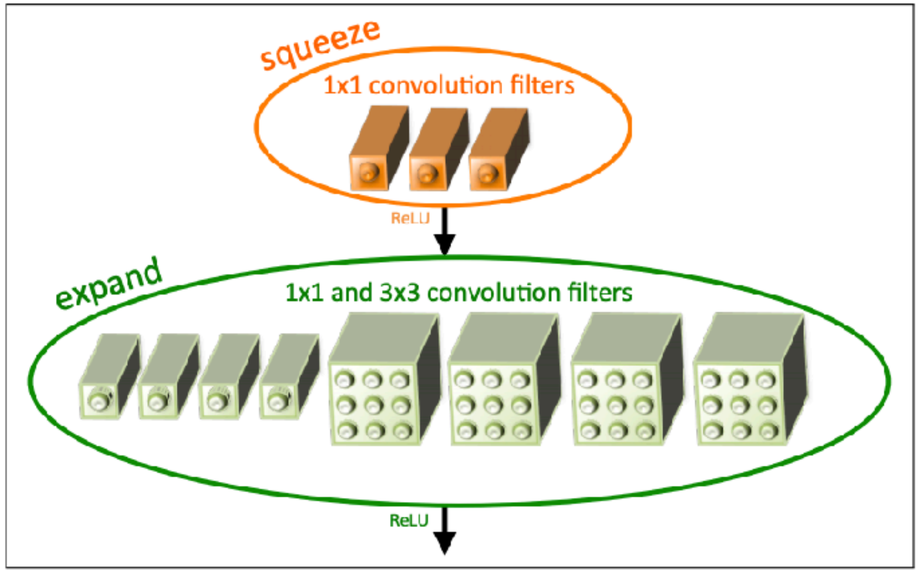
SqueezeNet

- Introduced in 2016 by researchers at the Berkeley, University of California, Stanford and DeepScale.
- Aim was to make a highly accurate by small CNN. Why?
  - Less communication across servers during distributed training
  - Smaller size allows it to be used on embedded systems like FPGAs
  - Faster to update models via the cloud (less bandwidth)
- It had 50x less parameters than AlexNet and performed 3X faster

SqueezeNet Key Takeaways

- **Replace 3x3 Filters with of 1x1** - has 9X fewer parameters than a 3x3 filter
- **Decrease the number of input channels to 3x3 filters** - Remember the quantity of parameters in a layer is (input channels * number of filters * 3 * 3).
- **Downsample later in the network so that convolution layers have larger activation maps**

Fire Module - Squeeze and Expand Layers



https://arxiv.org/pdf/1602.07360.pdf

- The Fire Module has a squeeze convolution layer (1x1 filters) that feed into an expand layer (mix of 1x1 and 3x3 convolution filters)

Fire Model - SqueezeNet's Squeeze and Expand Layers

The full SqueezeNet Architecture consists of a standalone Conv Layer followed by **8 fire** modules.

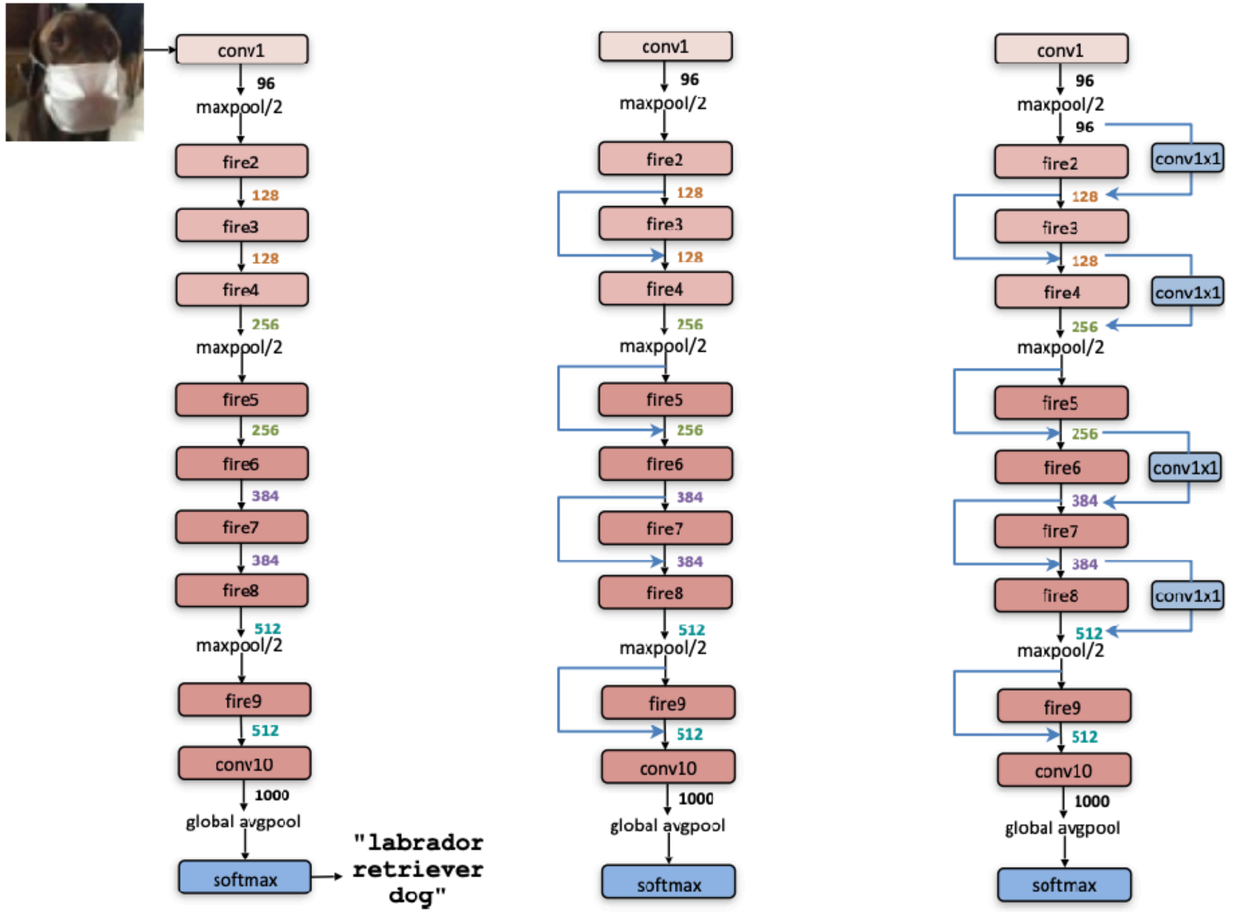
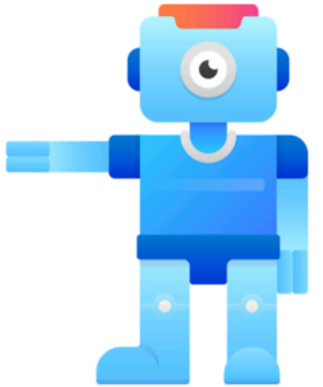


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

SqueezeNet's Performance

Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

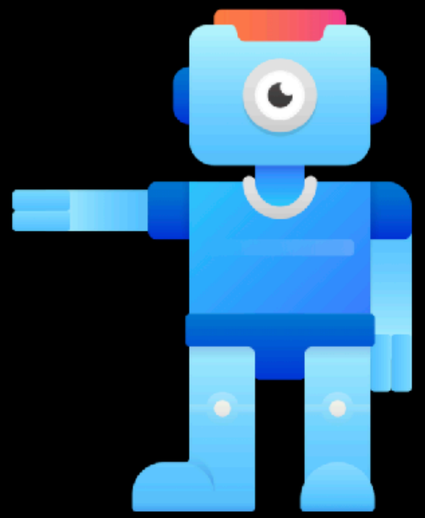


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

Let’s take a look at Goolge’s EfficentNet



MODERN COMPUTER VISION

BY RAJEEV RATAN

EfficentNet

Improving Accuracy and Efficiency through AutoML and Model Scaling

EfficientNet

- Introduced in 2019 by researchers at Google *(Mingxing Tan and Quoc V. Le)*
- Motivation behind EfficientNet:
  - CNNs are typically designed at a fixed resource cost and then scaled up (e.g. ResNet-18 to ResNet-200)
  - Scaling works by either increasing **depth** (number of layers) or **width** (number of filters)
  - This is often tedious to experiment with, requires manual tuning and results in suboptimal results
- What if there was a more principled method to scale up CNNs?

EfficientNet's Scaling Method Compound Coefficient

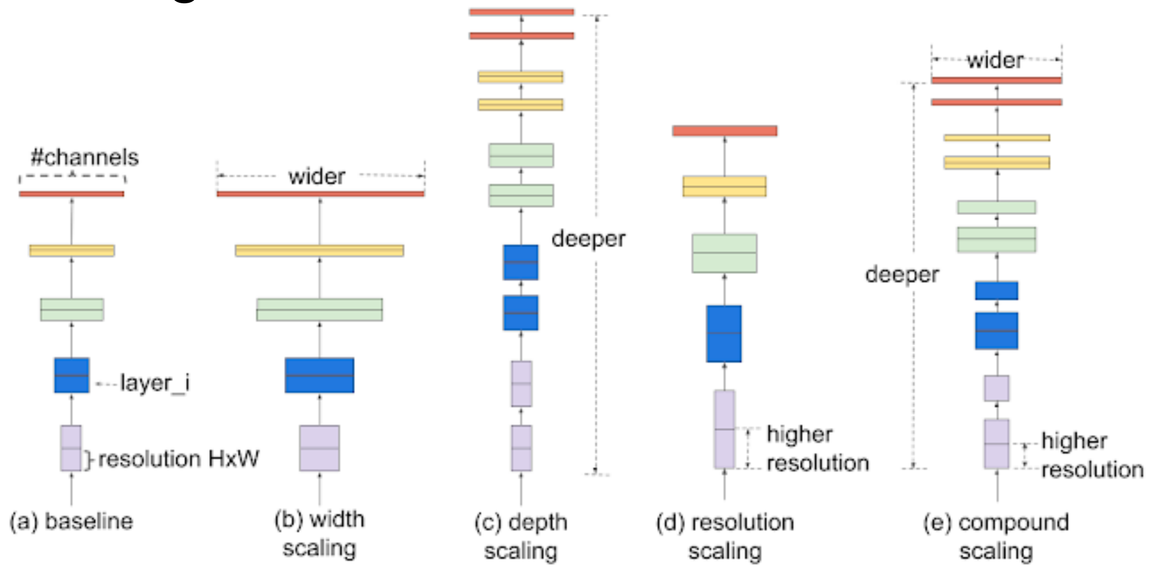
- This method uniformly scales each dimension (width, depth, resolution) with a fixed set of scaling coefficients
- It utilizes Google's new AutoML
- The EfficientNet family of models are able to surpass state-of-the-art accuracy with 10X better effiiceny

Scaling

- Researchers systematically studied the effects of scaling up different dimensions.
- It was found that balancing scaling in all dimensions resulted in the best overall performance

Grid Search

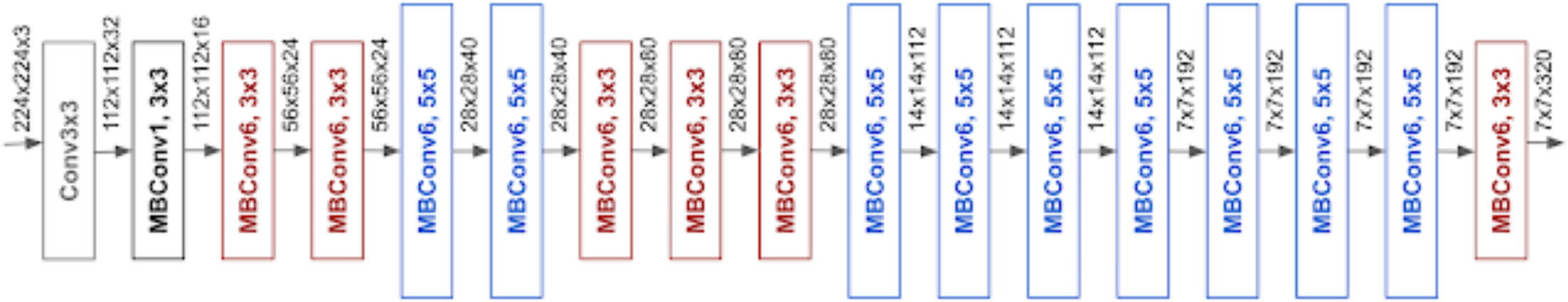
- **Grid Search** was used to find the relationship between different scaling dimensions of the baseline network under a fixed resource constraint (e.g. 2X more FLOPS).
- This determines the most **appropriate** scaling coefficient for each of the dimensions
- The coefficients are then **applied to scale up** the baseline network to the desired target model size or computational budget



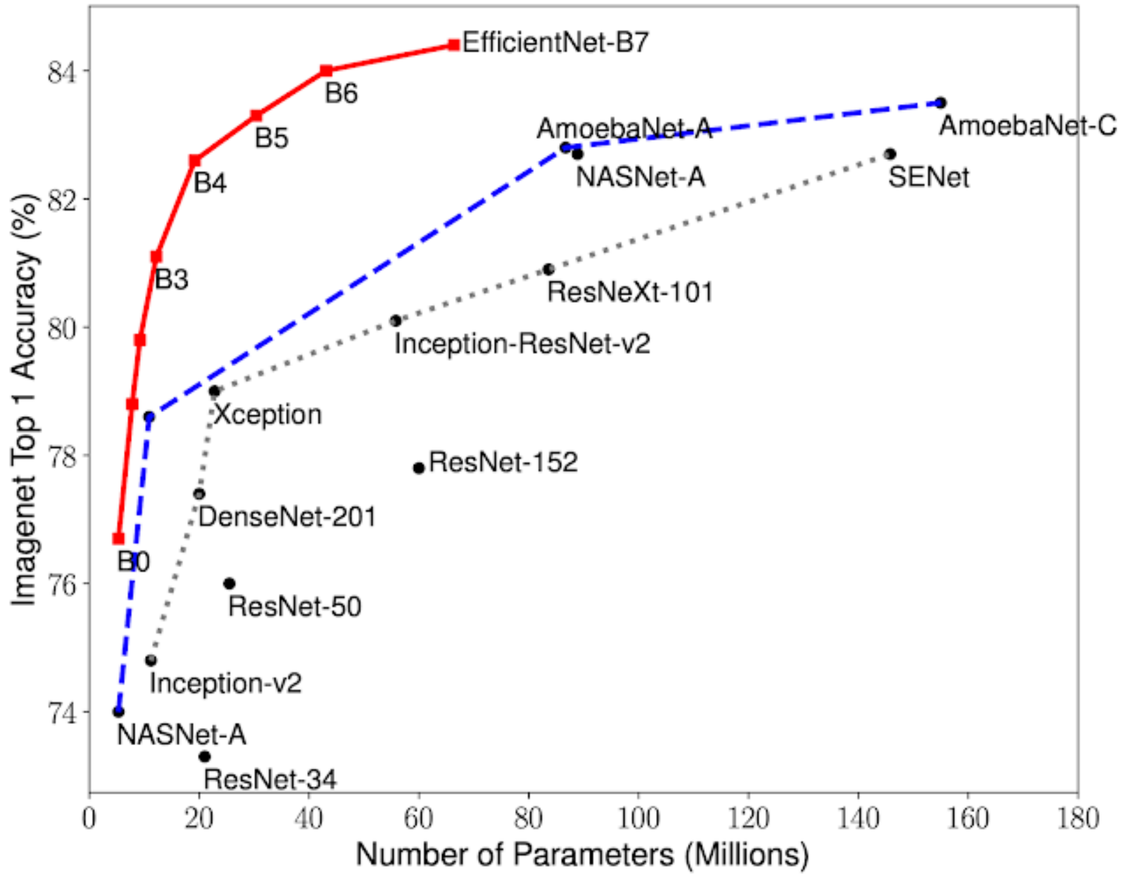
EfficientNet Architecture

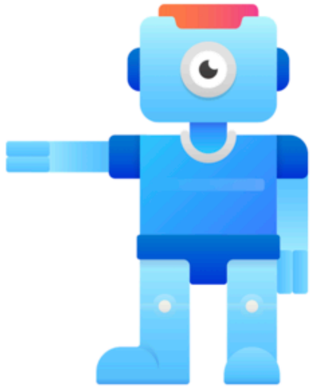
- The compound scaling methodology can be **applied to any CNN** (e.g. MobileNet attained 1.4% boost in accuracy, ResNet a 0.7%)
- The effectiveness of model scaling **depends heavily on the baseline network.**
- Researchers developed a new baseline network by performing a **neural architecture search** using AutoML's MNAS framework which optimises both accuracy and efficiency.
- Their new architecture using mobile **inverted bottleneck convolution** similar to MobileNetV2, but is slightly larger due to the increased FLOP budgets

EfficientNet Architecture



EfficientNet Performance



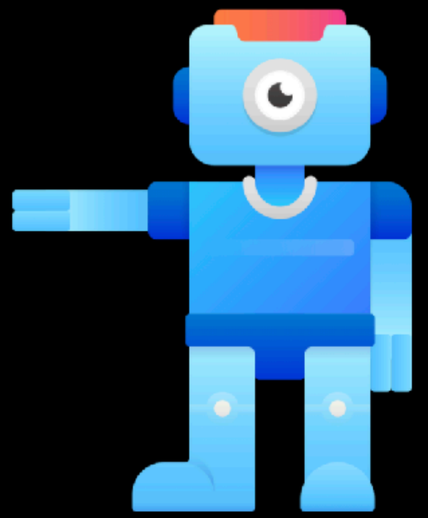


MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

Let’s take a look at DenseNets



MODERN COMPUTER VISION

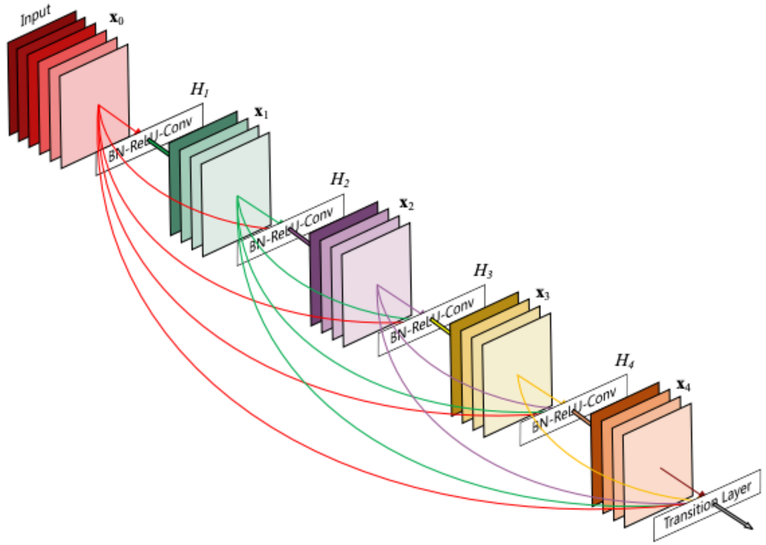
BY RAJEEV RATAN

DenseNets

ResNets but better!

Densely Connected Convolutional Neural Networks

- Introduced in 2016 jointly by Cornwell University, Tsinghua University and Facebook AI Research, DenseNet won Best Paper Award at 2017 CVPR conference.
- It was able to attain higher accuracy than ResNet with fewer parameters.

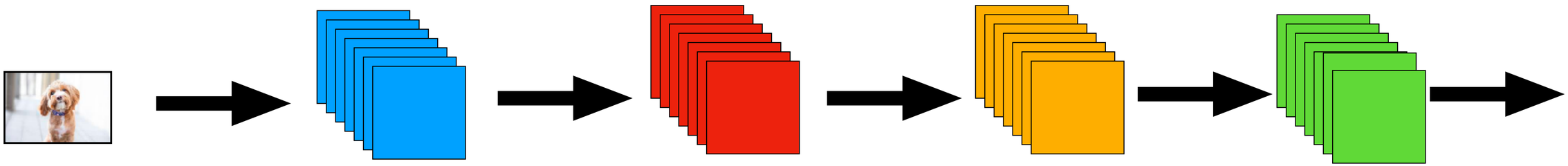


https://arxiv.org/pdf/1608.06993.pdf

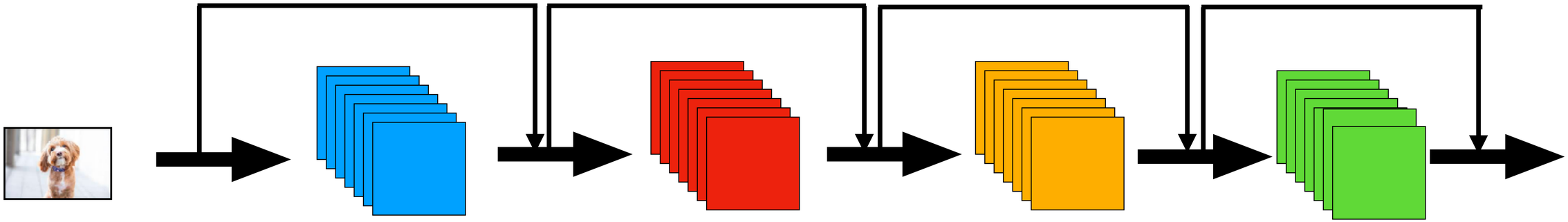
Motivation

- Training Deep CNNs is problematic due to **vanishing gradients**
- This is because the path for deep networks become so long gradients go to zero before completing the path (vanish)
- DenseNets solve this by using an ingenious concept of ‘collective knowledge’ where each layer receives info from all previous layers

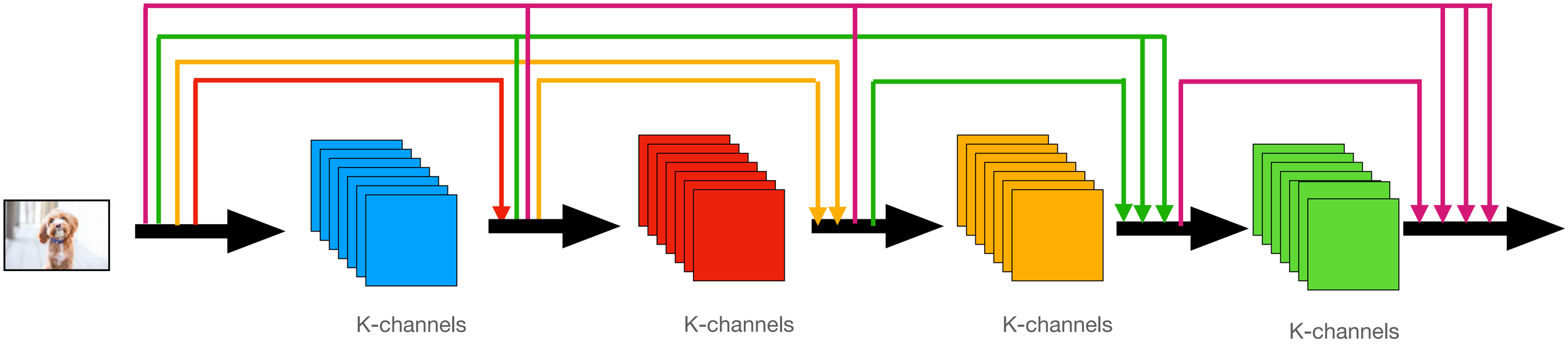
Classical Convolution Architecture



ResNet Convolution Architecture



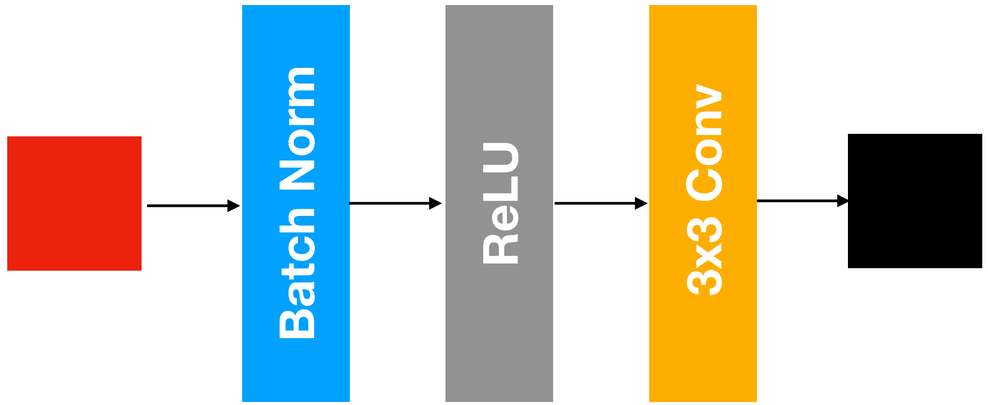
DenseNet Architecture



- Each layer receives info from all previous layers
- Growth Rate k is the additional number of channels for each layer

DenseNet Component

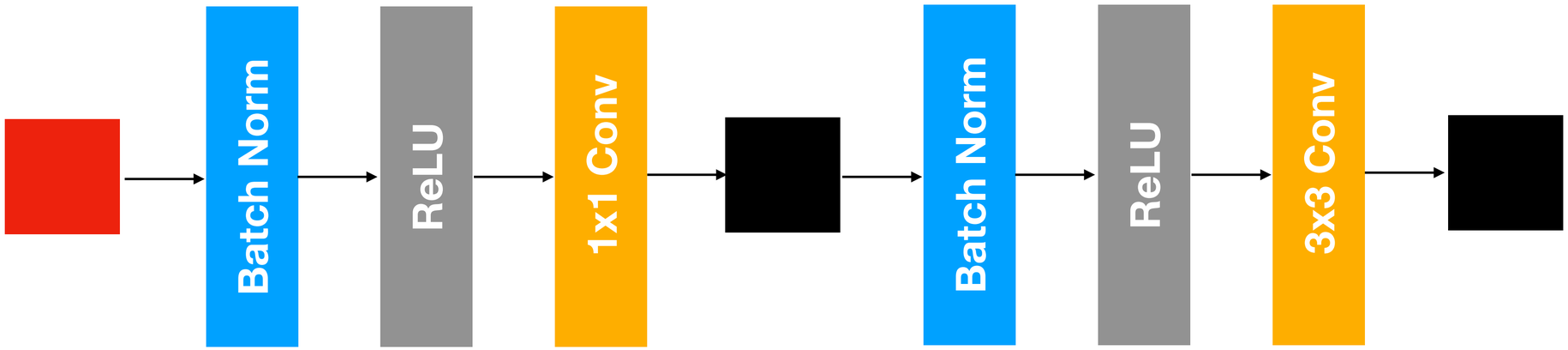
Basic DenseNet Composition Layer



- Basic DenseNet Composition Layer contains Pre-Activation Batch Norm, ReLU and then a 3x3 Conv Layer

DenseNet Component

DenseNet-B (Bottleneck Layers)



- BN-ReLU **1x1** Conv is done before BN-ReLU **3x3** Layer

DenseNet Component

Multiple Dense Blocks with Transition Layers

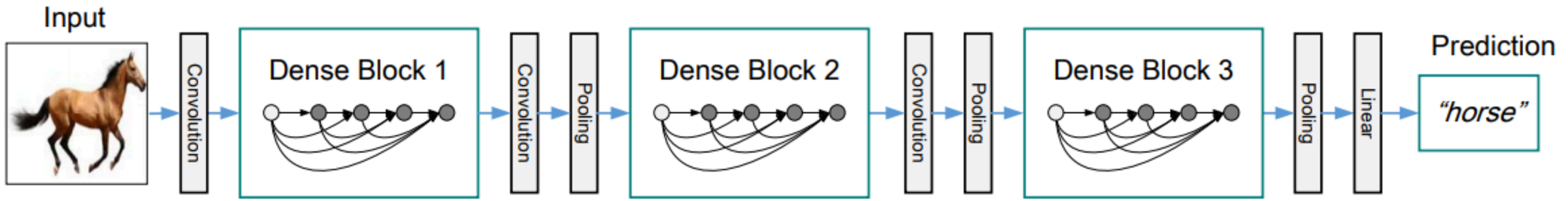


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

- 1x1 Conv is followed by 2x2 Average Pooling are used as 'Transition Layers' between two contiguous dense blocks

DenseNet Component

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

DenseNet Performance

Model	top-1	top-5
DenseNet-121	25.02 / 23.61	7.71 / 6.66
DenseNet-169	23.80 / 22.08	6.85 / 5.92
DenseNet-201	22.58 / 21.46	6.34 / 5.54
DenseNet-264	22.15 / 20.80	6.12 / 5.29

Table 3: The top-1 and top-5 error rates on the ImageNet validation set, with single-crop / 10-crop testing.

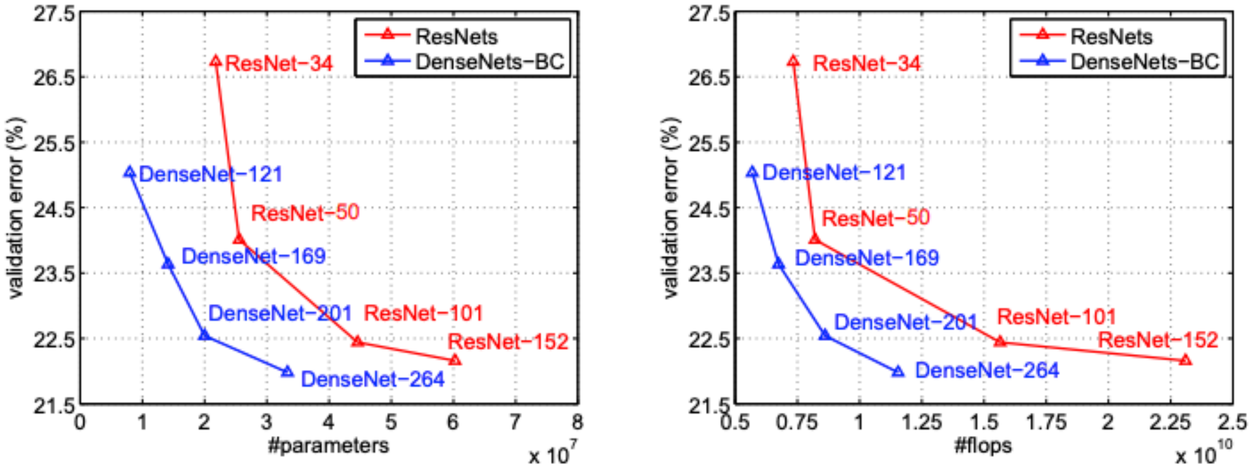
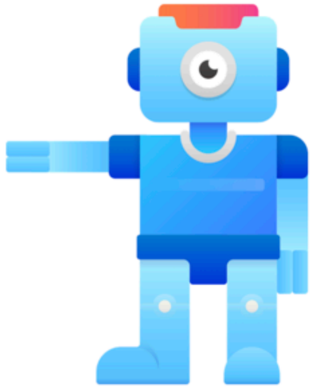


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

- DenseNet-B - DenseNets with a bottleneck layer
- DenseNet-BC - Bottleneck + Compression (C) Factor *theta* (Controls the feature map reduction)



MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

ImageNet

Examine Models

PyTorch

2. AlexNet
3. VGG
4. ResNet
5. SqueezeNet
6. DenseNet
7. Inception v3
8. GoogLeNet
9. ShuffleNet v2
10. MobileNet v2
11. ResNeXt
12. Wide ResNet
13. MNASNet

Keras

1. Xception
2. VGG16 19
3. ResNet50
4. ResNet152V2
5. InceptionV3
6. MobileNetV2
7. DenseNet
8. NASNetMobile
9. NASNetLarge
10. EfficientNet

1.Pretrained Models

2.ImageNet

3.HDF5

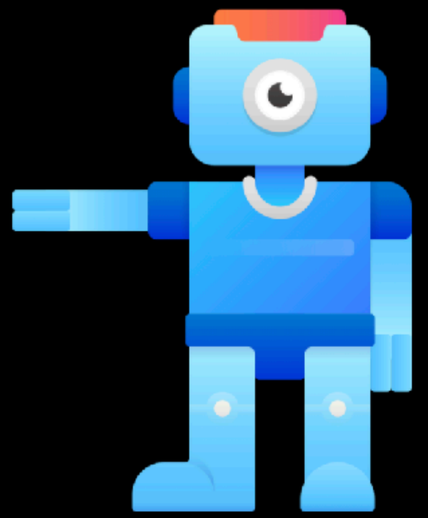
4.Rank-1 and Rank-5

5.Feature Extractors

6.Transfer Learning Theory

7.Transfer Learning Keras

8.Transfer Learning PyTorch



MODERN COMPUTER VISION

BY RAJEEV RATAN

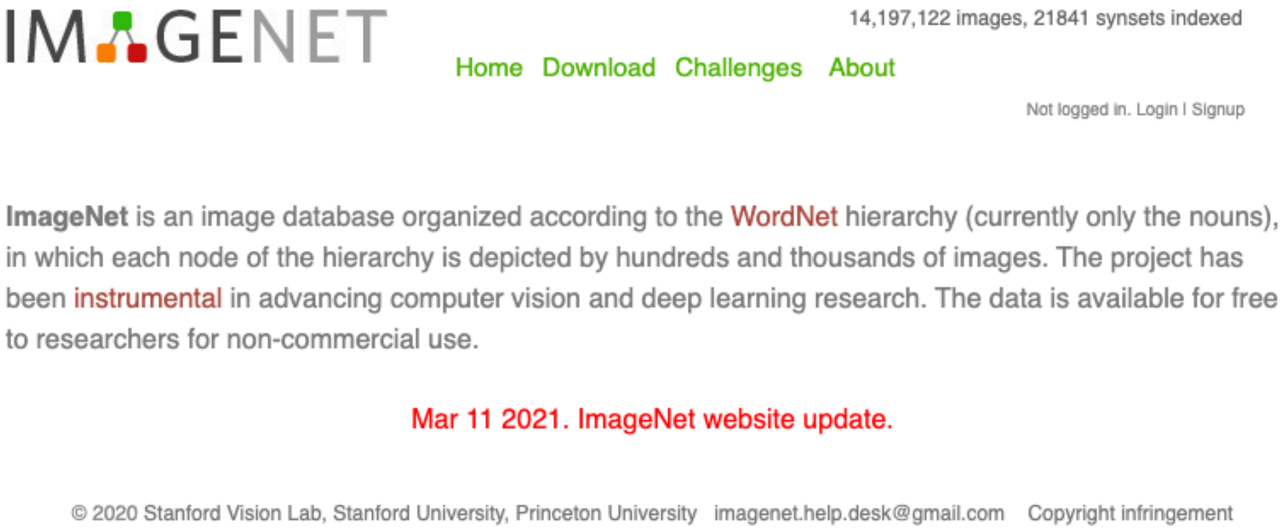
ImageNet

Why ImageNet is so important to the Computer Vision World

ImageNet

- Currently the world's largest dataset of labeled images

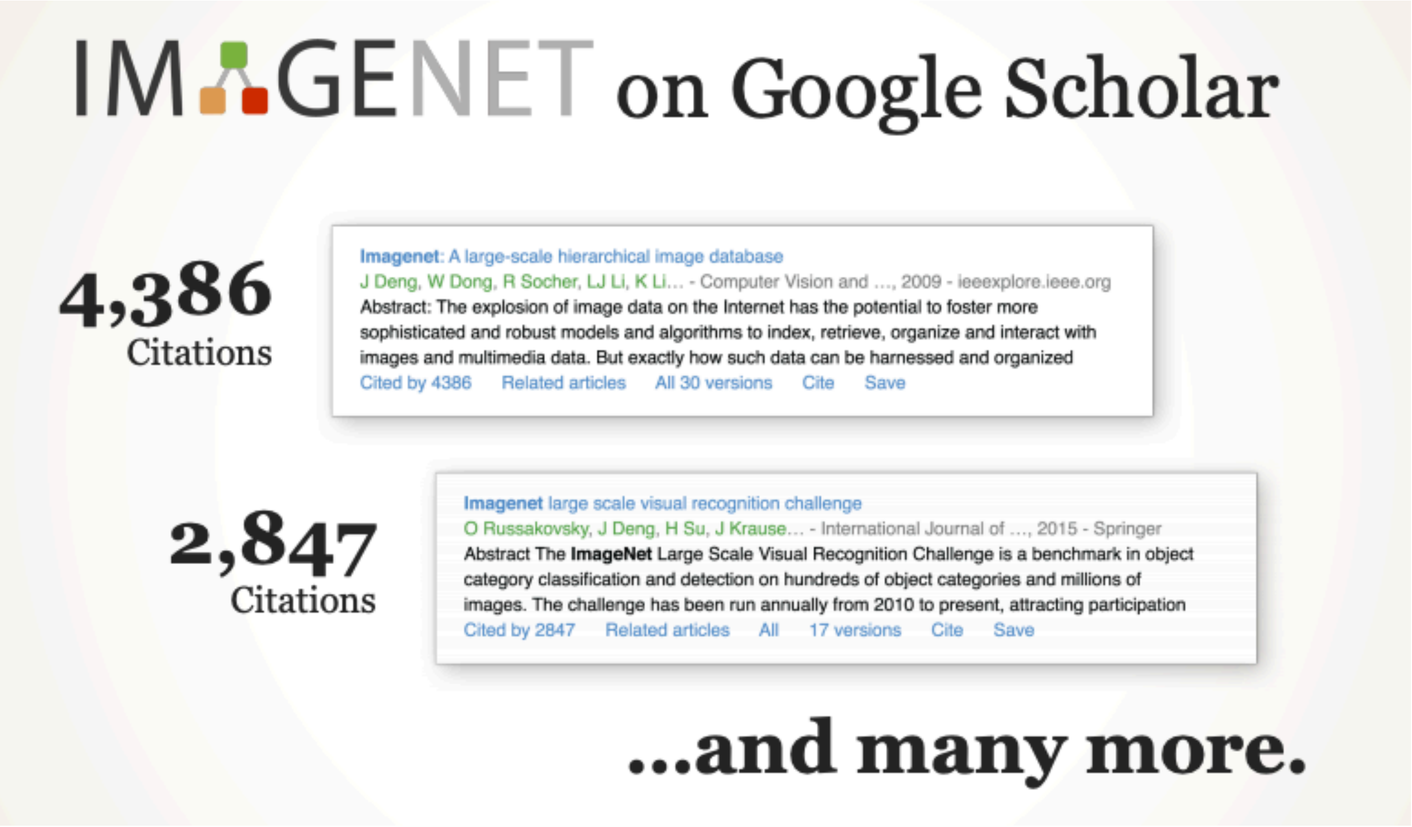
https://www.image-net.org/



ImageNet - ILSVR

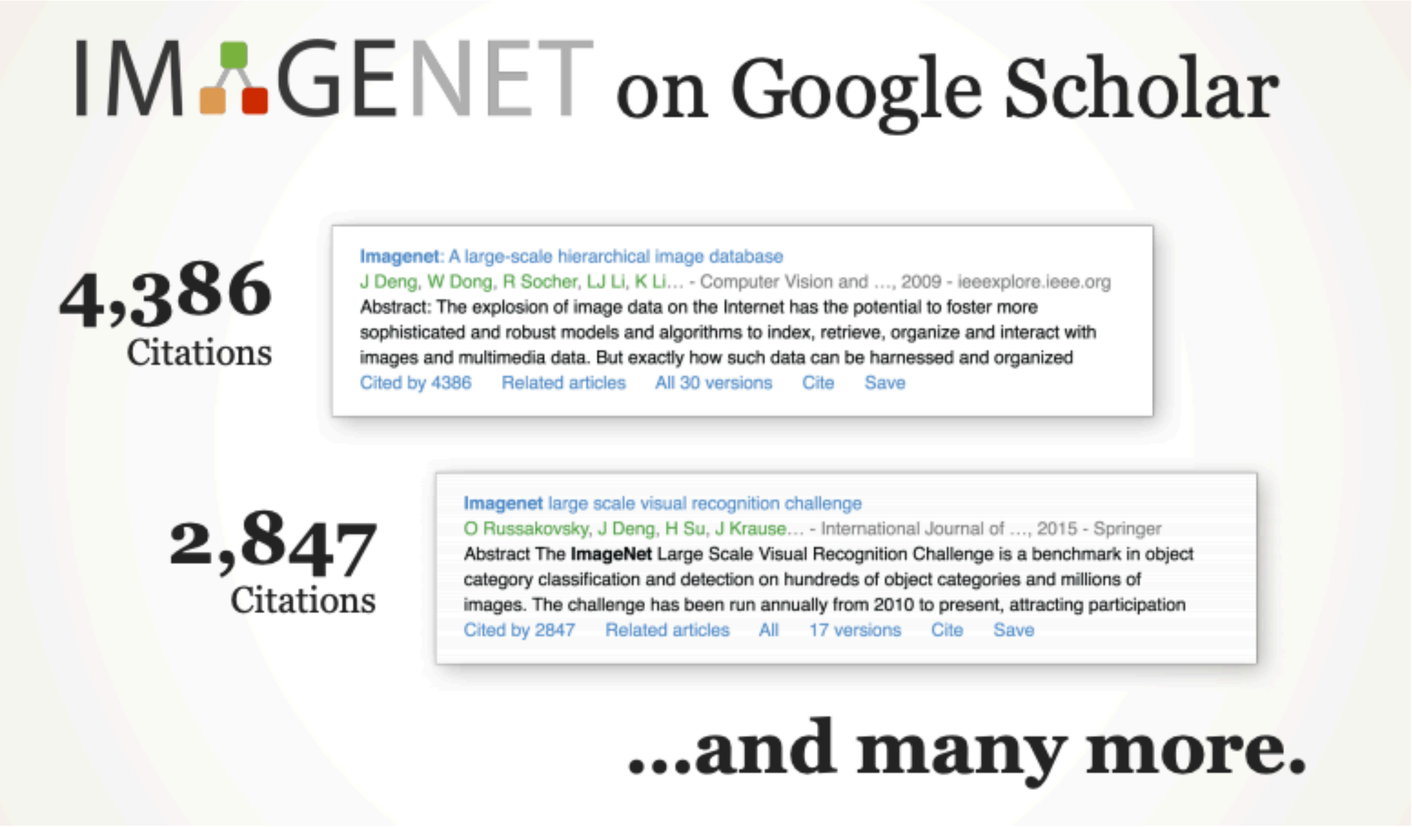
- The most highly-used subset of ImageNet is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017 image classification and localisation dataset.
- The dataset spans 1000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images. This subset is available on Kaggle.
- It is the most common benchmark used when evaluating new exotic CNN Models.

ImageNet in Research



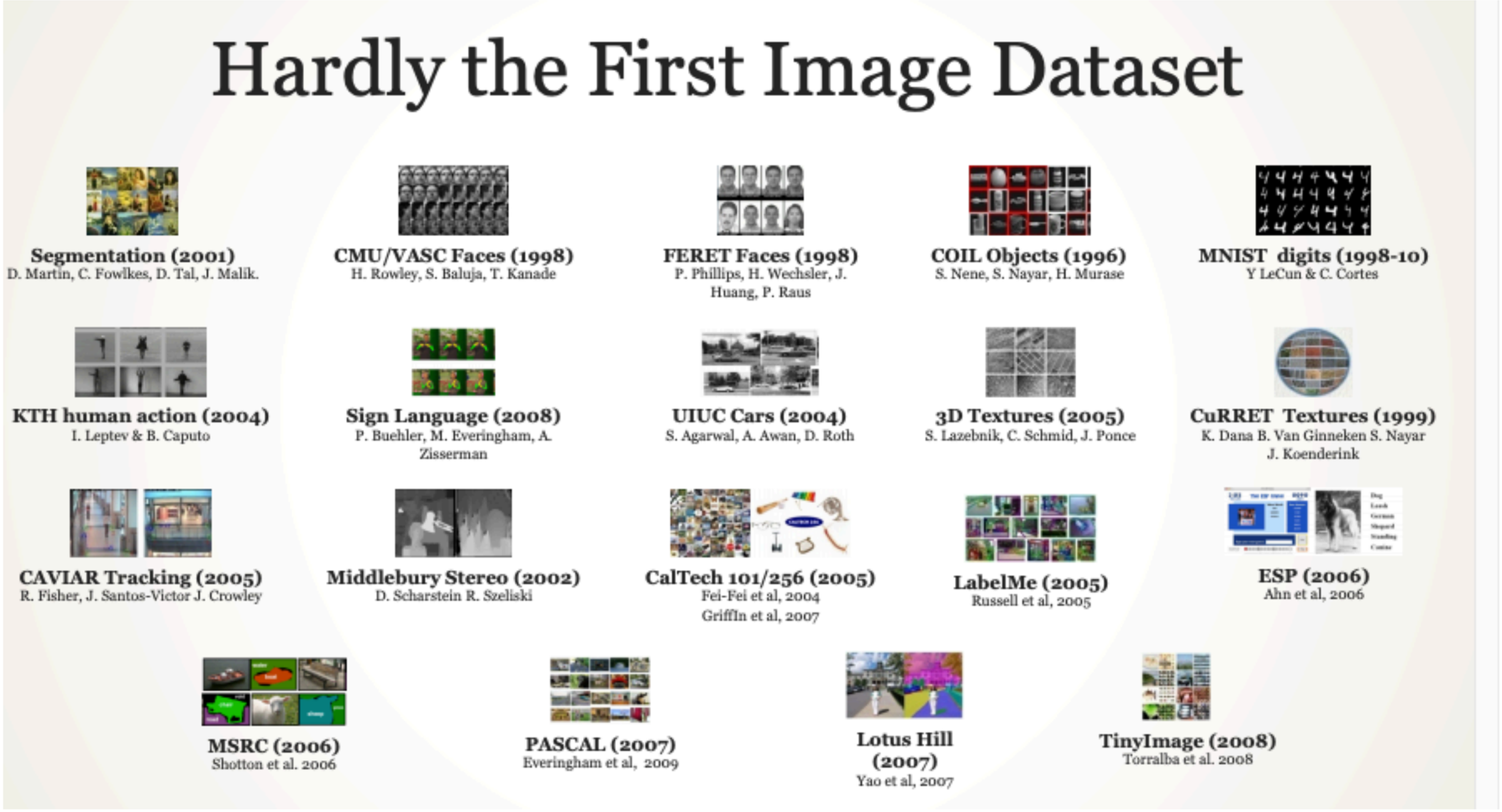
https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

ImageNet in Research



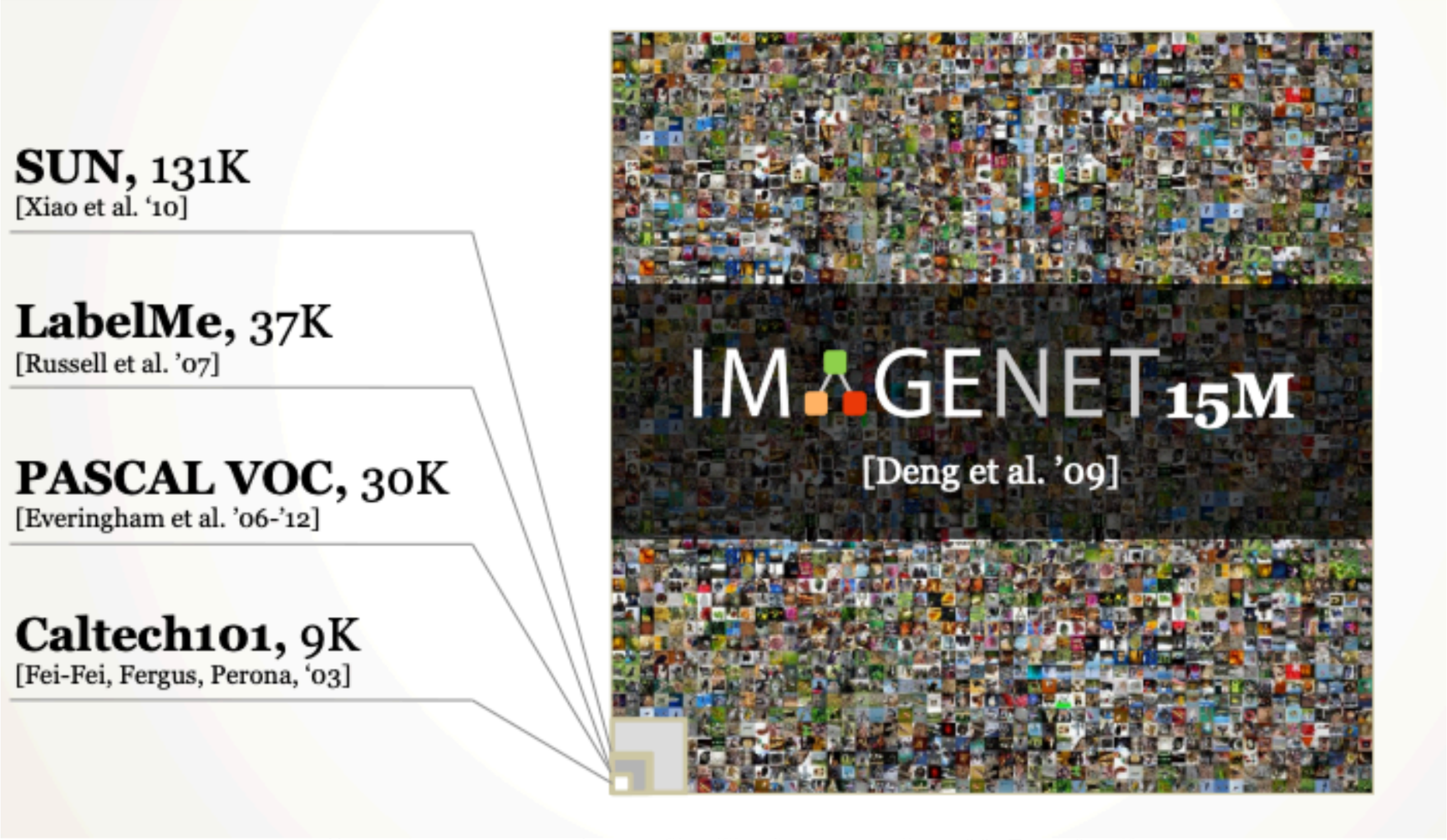
https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

ImageNet



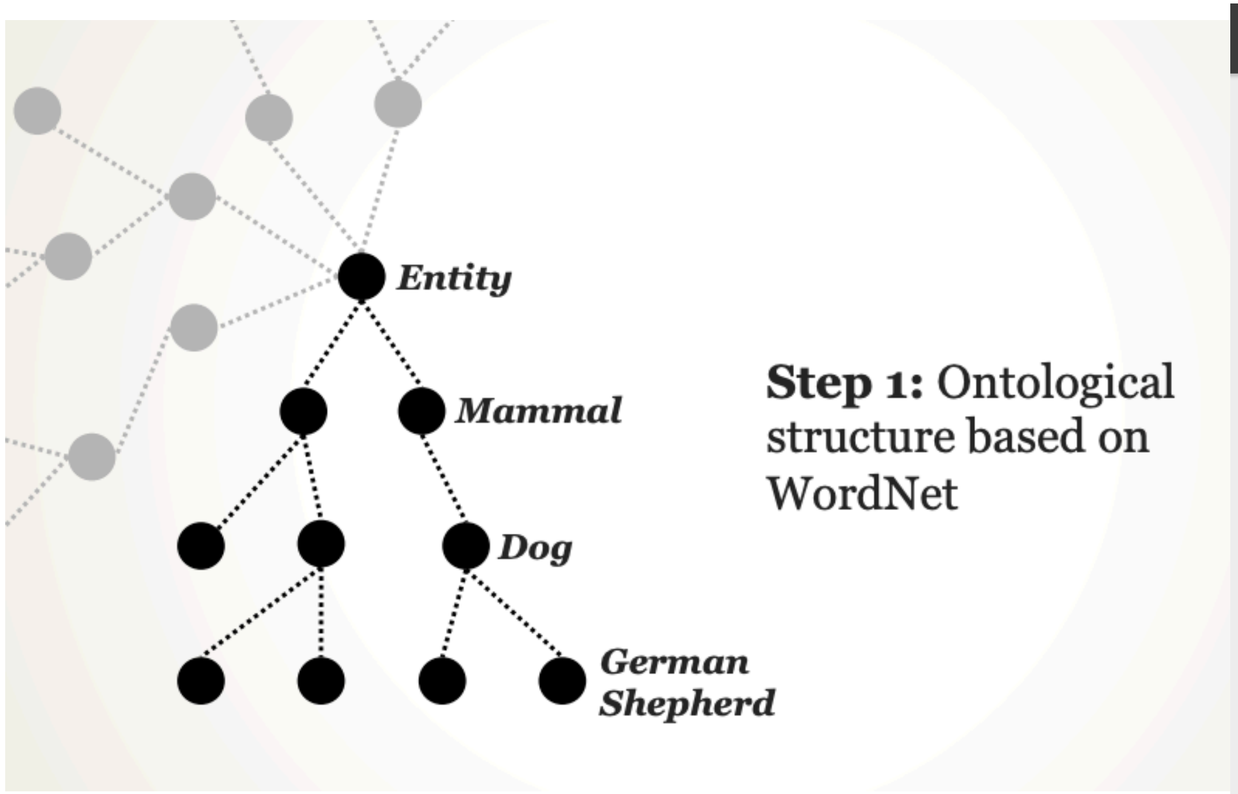
https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

ImageNet - What makes it so good? It's Size

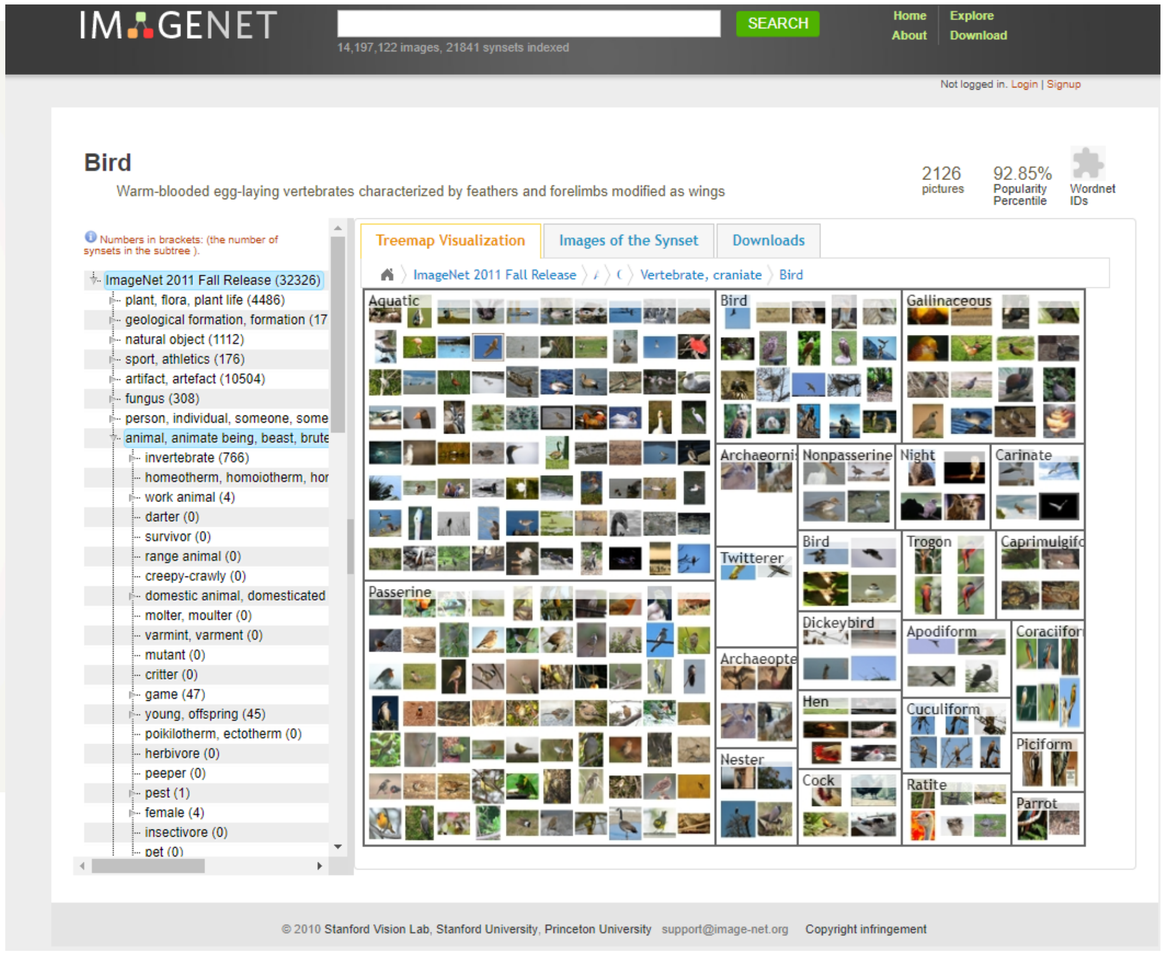


https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

ImageNet - WordNet?

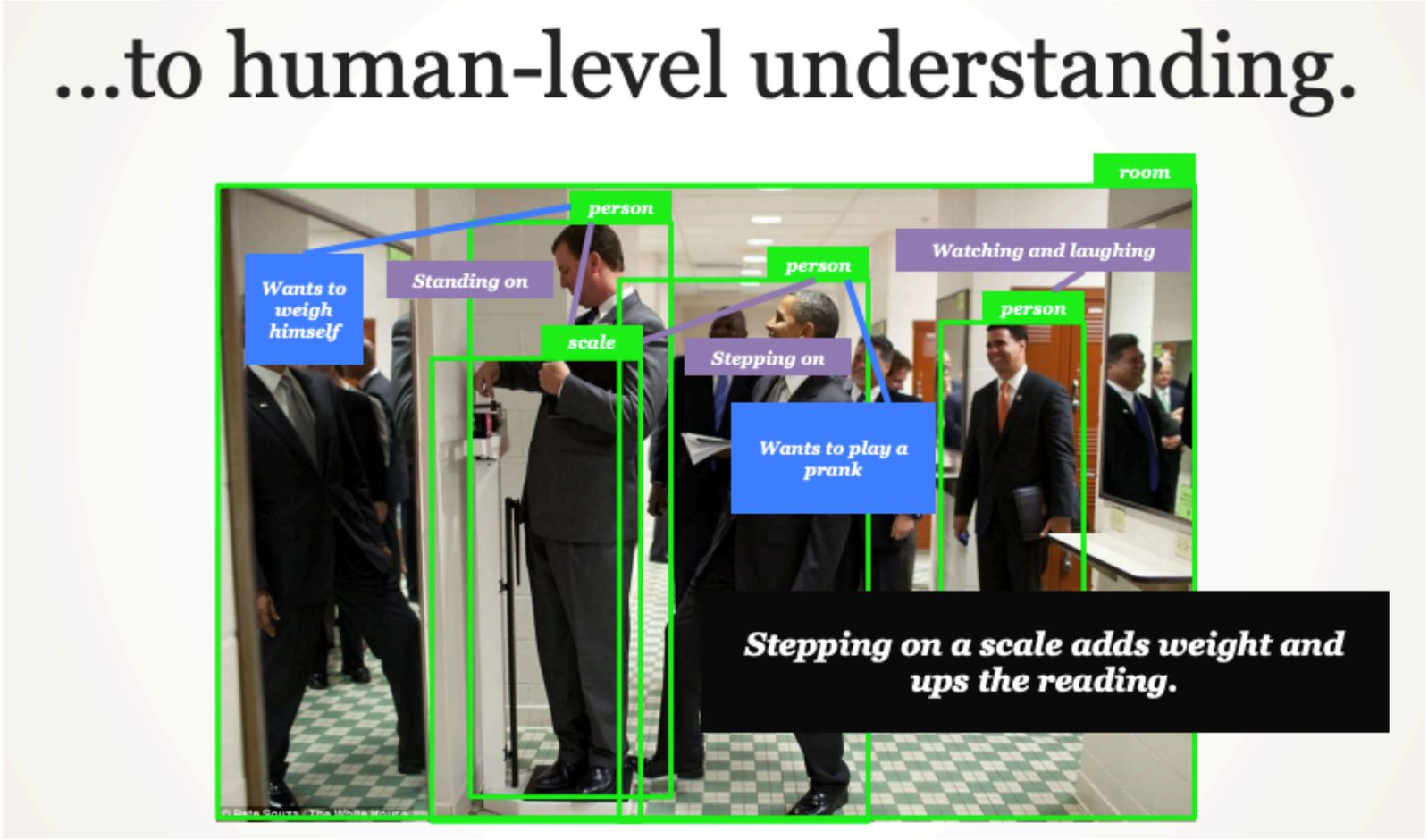


Step 1: Ontological structure based on WordNet



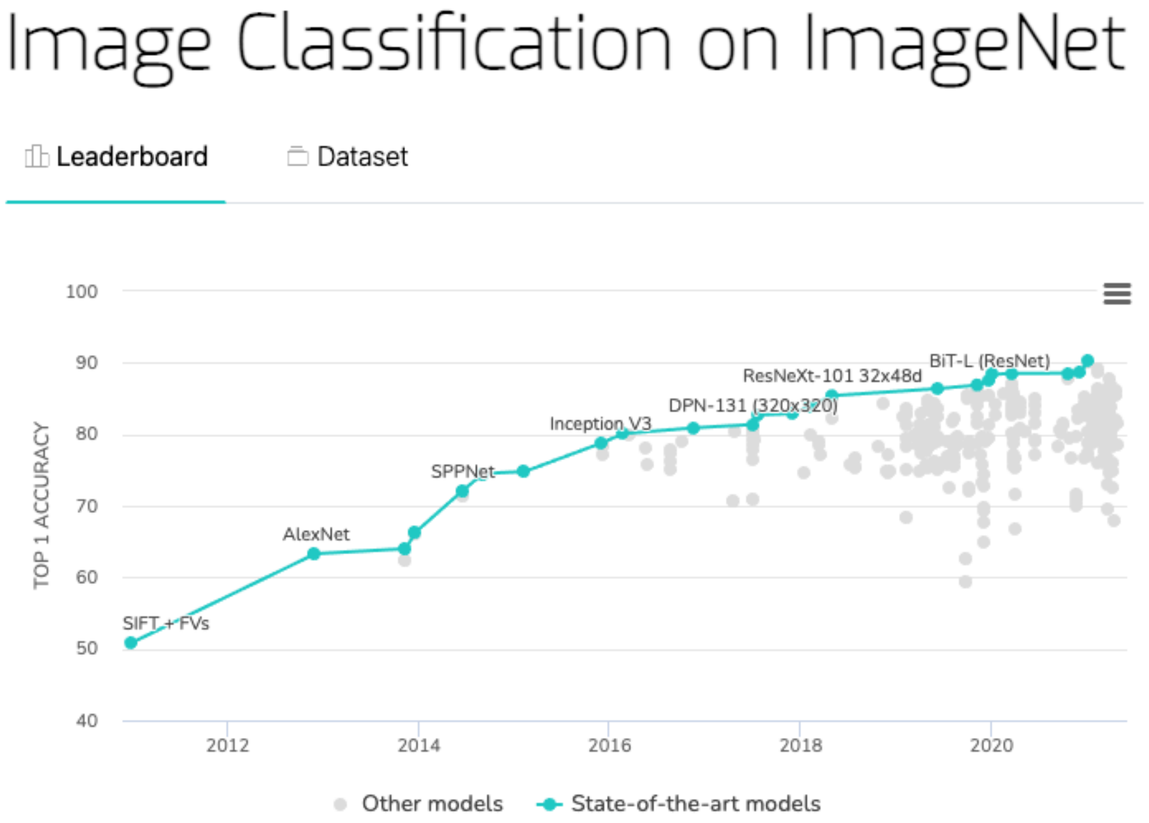
https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

ImageNet - WordNet - Aims to Give Rise to…



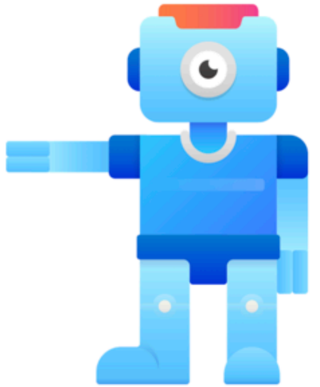
https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

ImageNet - Current State of Art Performance



RANK	MODEL	TOP 1 ACCURACY	TOP 5 ACCURACY	NUMBER OF PARAMS	EXTRA TRAINING DATA	PAPER
1	**Meta Pseudo Labels** (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels
2	**Meta Pseudo Labels** (EfficientNet-B6-Wide)	90%	98.7%	390M	✓	Meta Pseudo Labels
3	**NFNet-F4+**	89.2%		527M	✓	High-Performance Large-Scale Image Recognition Without Normalization
4	**ALIGN** (EfficientNet-L2)	88.64%	98.67%	480M	✓	Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision
5	**EfficientNet-L2-475** (SAM)	88.61%		480M	✓	Sharpness-Aware Minimization for Efficiently Improving Generalization

https://paperswithcode.com/sota/image-classification-on-imagenet



MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

Top-1 and Top-5 Accuracy