CS867 Computer Vision Fall 2021

Lecture 11-12

More on Convolutional Neural Network Architectures

Dr. Muhammad Moazam Fraz
Associate Professor
Department of Computing
NUST SEECS

Web : Lab

Where are we

Convolutional Neural Network

- **CNN Parameters**
- **CNN Architecture Case Studies**
  - Alex Net
  - VGG Net
  - GoogleNet
- **CNN Training**
  - Training in Batches
  - Learning Rate configuration
  - Data augmentation on the fly
  - Transfer Learning (freeze layers and train rest of layers)
  - How to see (visualize) the feature map at different layers
  - Debug CNN, how to analyze why CNN is not performing good.
  - GradCam demonstration

Today we will cover

More on CNNs

CNN Architecture Case Studies

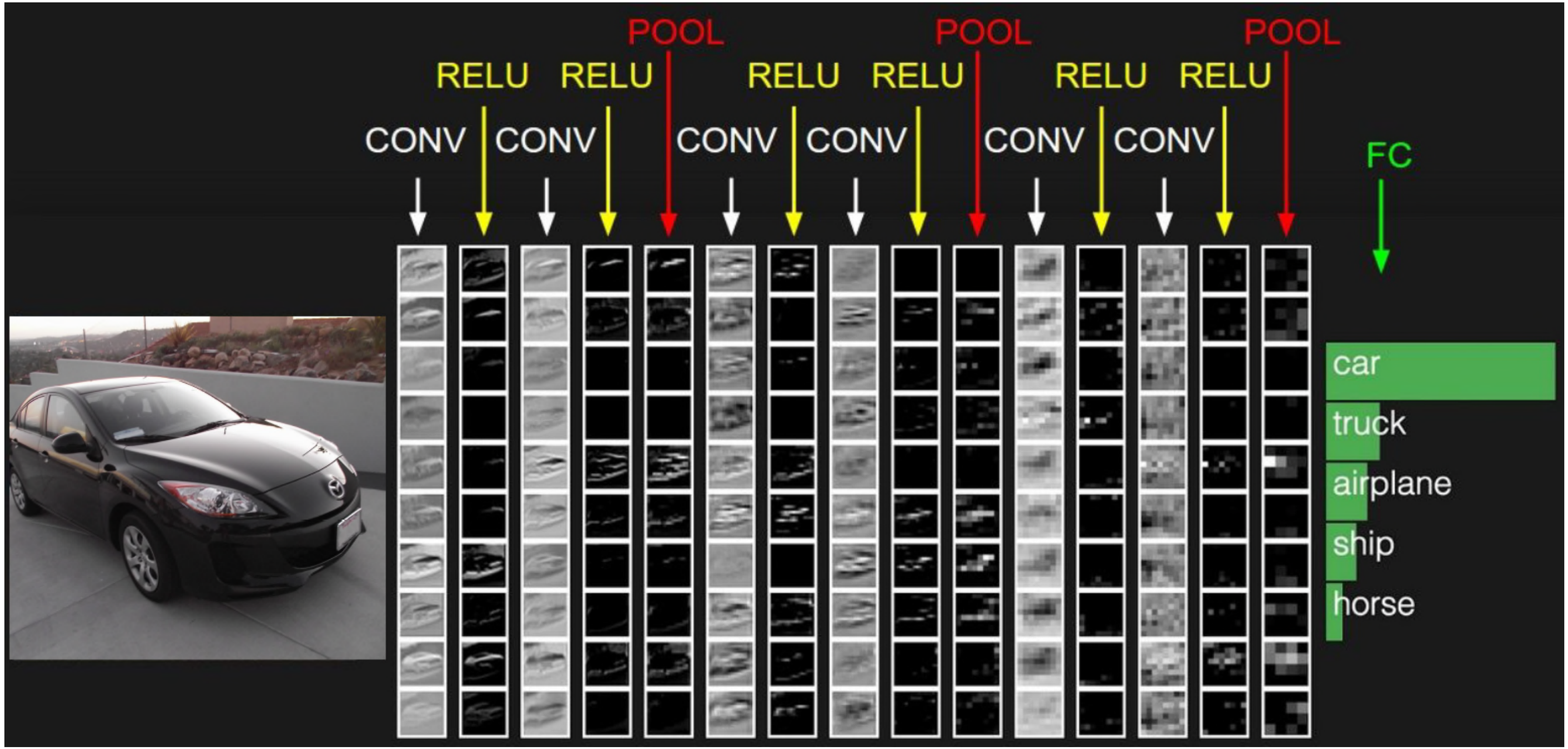
- Resnet
- ResNeXt
- Squeeze and Excitation Net
- DenseNet
- Separable Convolutions
- Neural Architecture Search
- EfficientNet

What I will not be covering

Revise it : Ref CS 878 Deep Learning

- Inductive Bias in CNNs
  - https://analyticsindiamag.com/top-5-inductive-biases-in-deep-learning-models/
  - https://samiraabnar.github.io/articles/2020-05/indist (First paragraphs)
- Bias – Variance Trade off
- Weight Initialization in CNNs
- Overfitting – under fitting
- Regularization
- Batch Normalization, Layer Normalization, Group Normalization
- Network parameter architectures search

Recap : CNN Examples



Recap

- **How do we decide the parameters for network architecture?**
  - *For less complicated situations, we can use 'trial and error'*

- **Is there any other method?**
  - 'Grid search' -> trial and error
  - 'Bayesian optimization' -> meta-learning; optimize the hyperparameters

- **General strategy:**
  - Bottleneck -> extract information (kernel) and learn to squeeze representation into a smaller number of parameters.

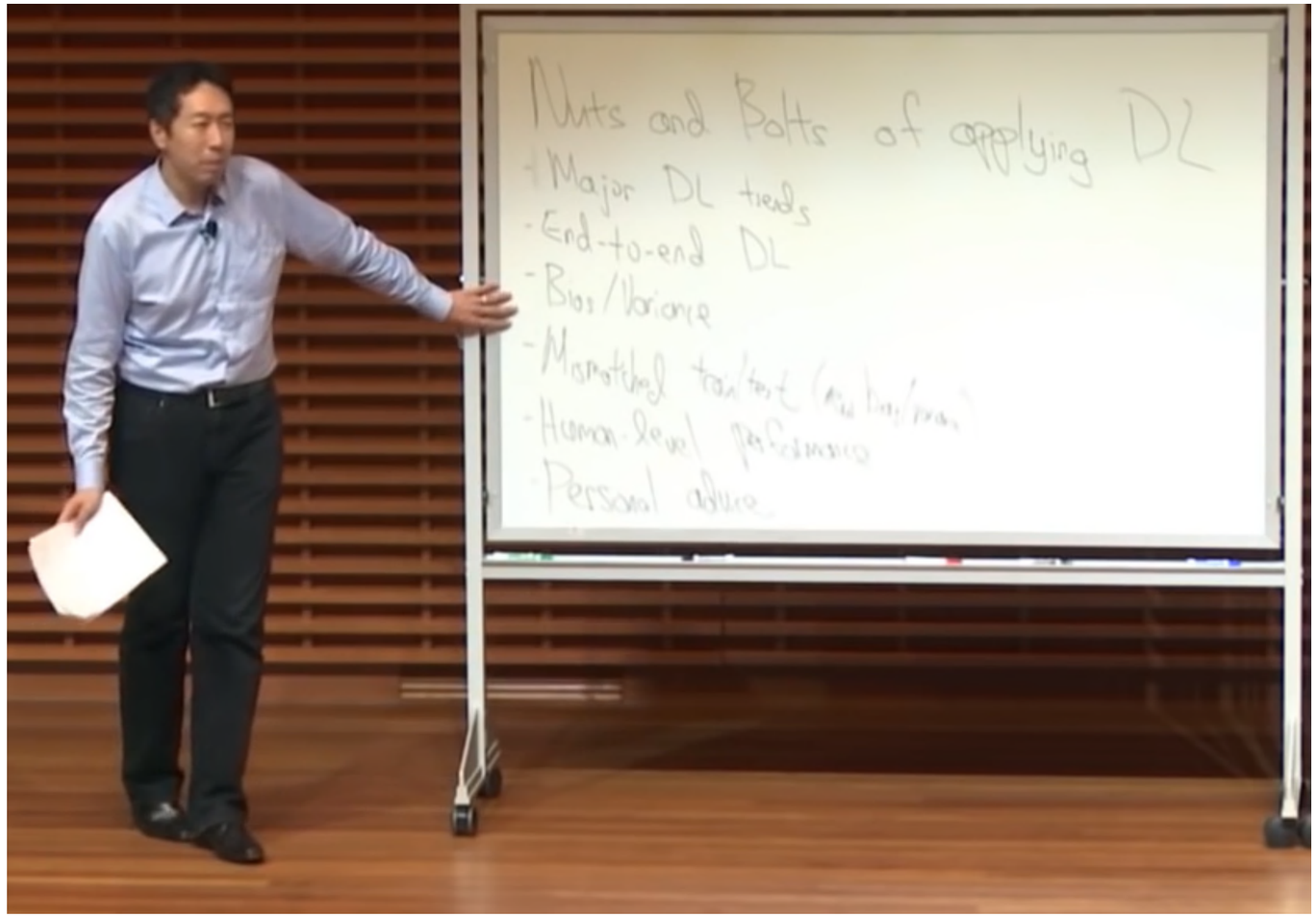
- **But – I happen to have a thousand GPUs:**
  - Neural Architecture Search!

When something is not working…

… how do I know what to do next?

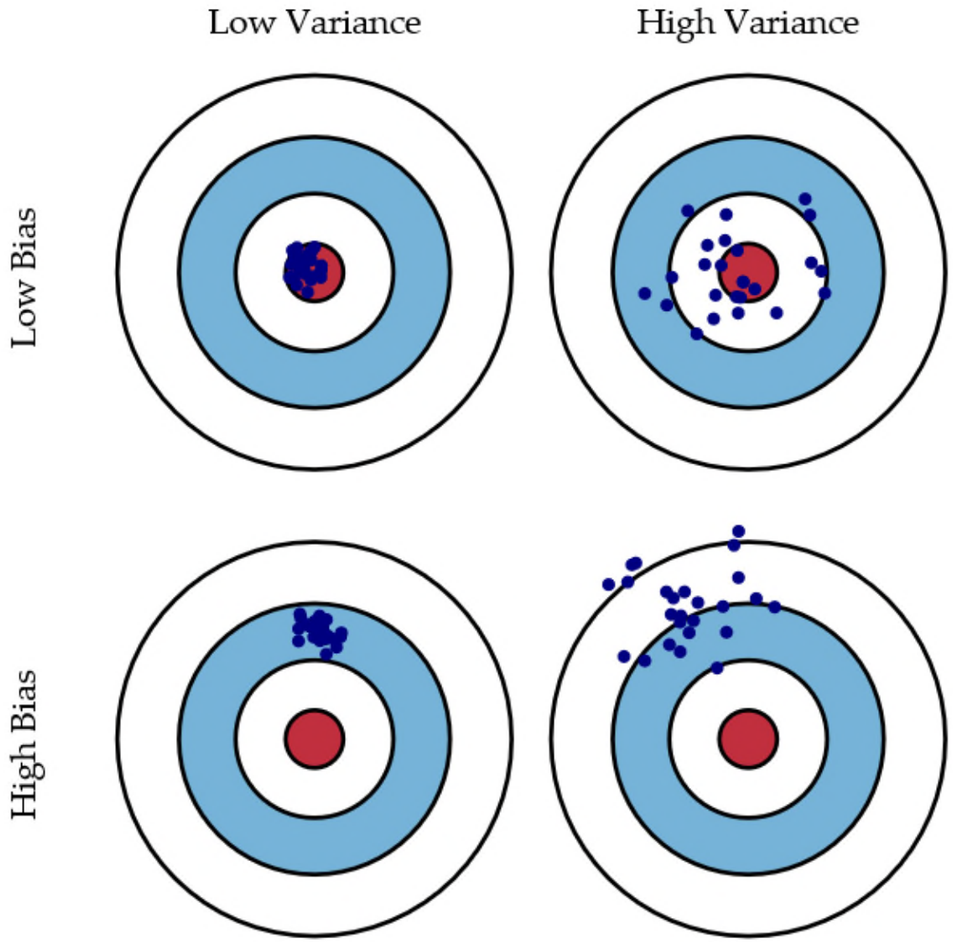
The Nuts and Bolts of Building Applications using Deep Learning

- Andrew Ng - NIPS 2016
- https://youtu.be/F1ka6a13S9I



Bias/variance trade-off

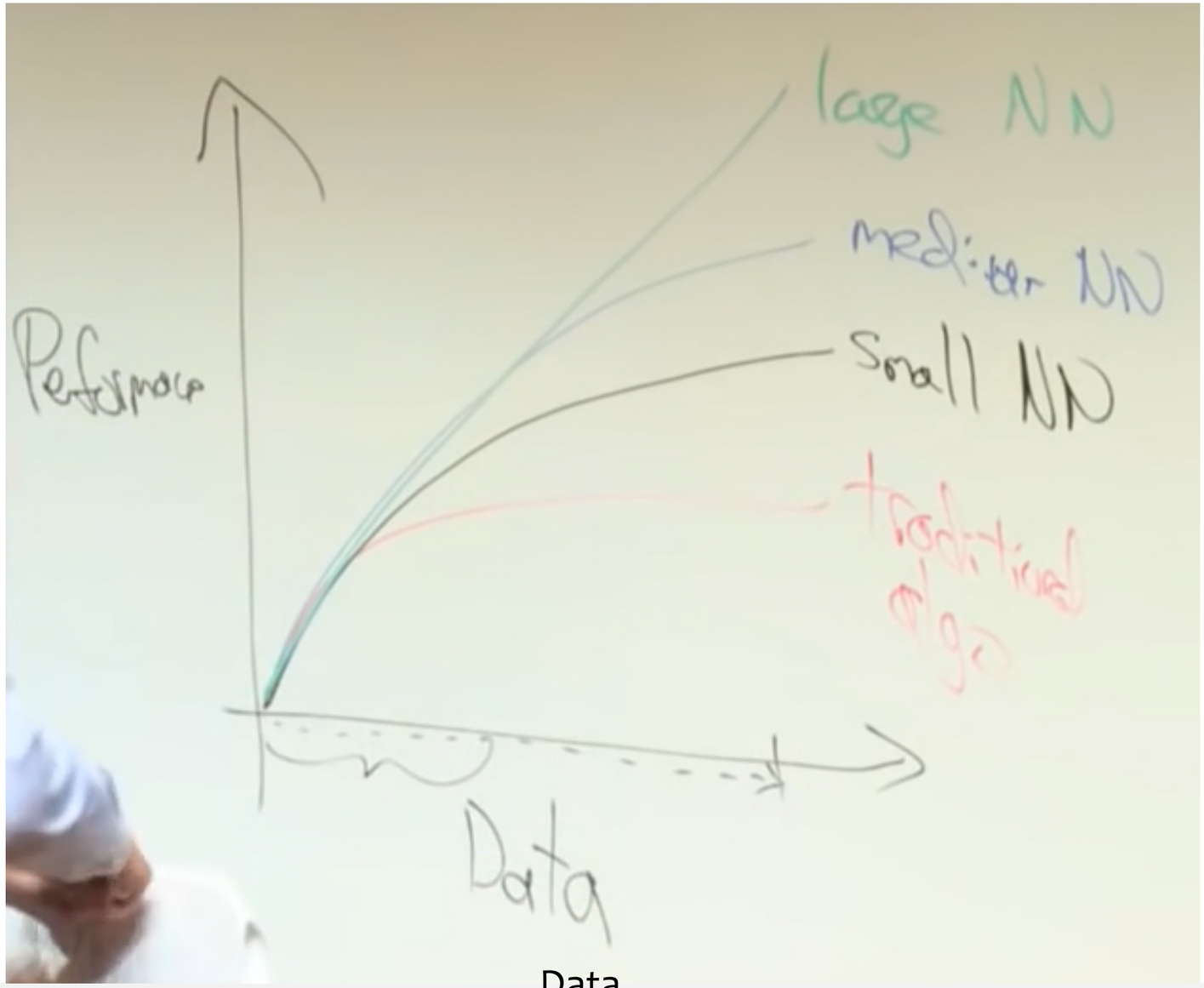
"It takes surprisingly long time to grok bias and variance deeply, but people that understand bias and variance deeply are often able to drive very rapid progress." *--Andrew Ng*



Bias = accuracy
Variance = precision

Scott Fortmann-Roe

Data


Performance
large NN
medium NN
small NN
traditional algo
Data

Data

Go collect a dataset

Most important thing:

Training data must represent target application!

Take all your data

60% training

40% testing

20% testing

20% validation (or ‘development’)

Properties

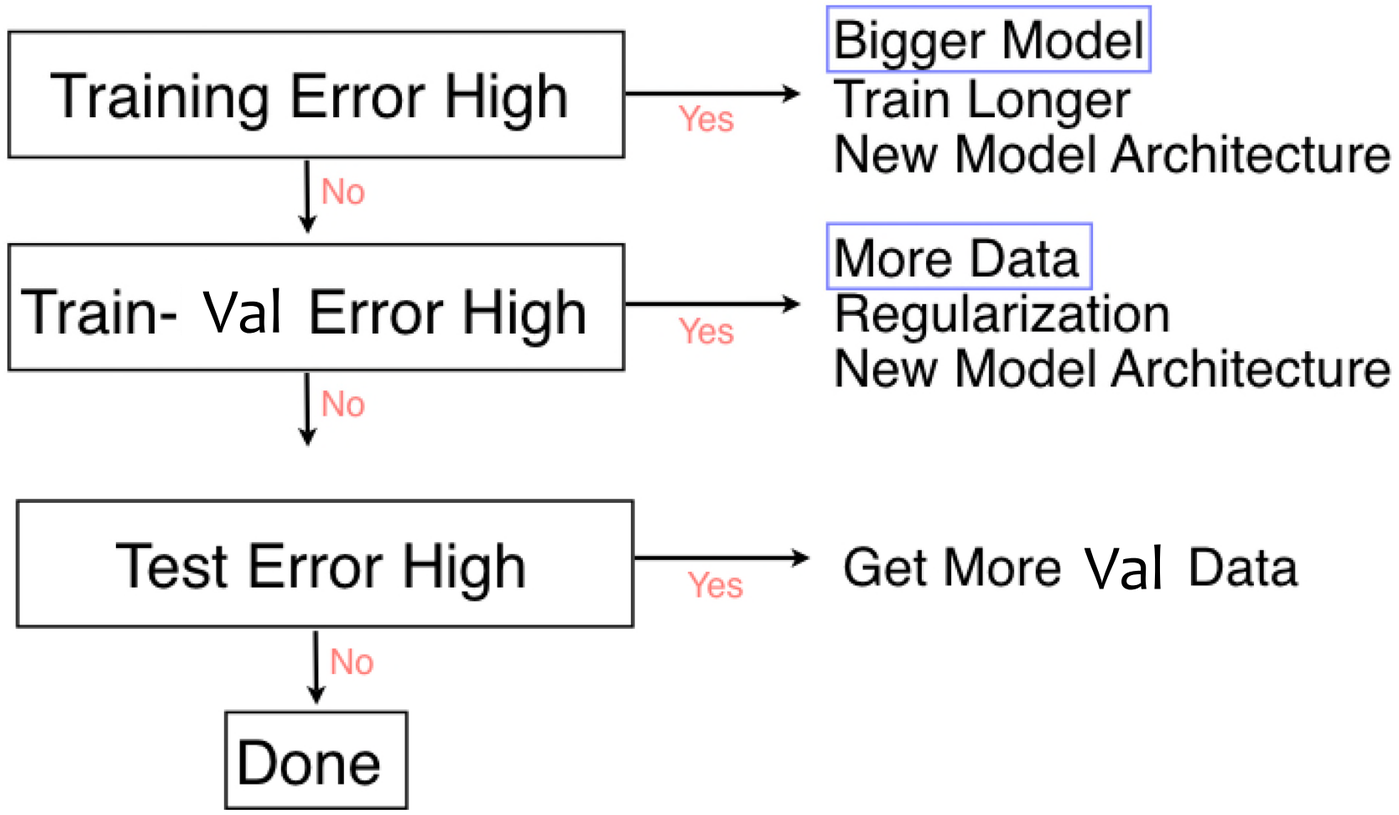
- Human level error = 1%
- Training set error = 10%
- Validation error = 10.2%
- Test error = 10.4%

"Bias"

"Variance"

Overfitting to validation

The Nuts and Bolts of Building Applications using Deep Learning



[Andrew Ng]

My Neural Network isn't working! What should I do?

http://theorangeduck.com/page/neural-network-not-working

So you're developing the next great breakthrough in deep learning but you've hit an unfortunate setback: your neural network isn't working and you have no idea what to do. You go to your boss/supervisor but they don't know either - they are just as new to all of this as you - so what now?

Well luckily for you I'm here with a list of all the things you've probably done wrong and compiled from my own experiences implementing neural networks and supervising other students with their projects:

1. You Forgot to Normalize Your Data
2. You Forgot to Check your Results
3. You Forgot to Preprocess Your Data
4. You Forgot to use any Regularization
5. You Used a too Large Batch Size
6. You Used an Incorrect Learning Rate
7. You Used the Wrong Activation Function on the Final Layer
8. Your Network contains Bad Gradients
9. You Initialized your Network Weights Incorrectly
10. You Used a Network that was too Deep
11. You Used the Wrong Number of Hidden Units

Daniel Holden

Image Classification is Challenging

Image Classification is Challenging

Image Classification is Challenging

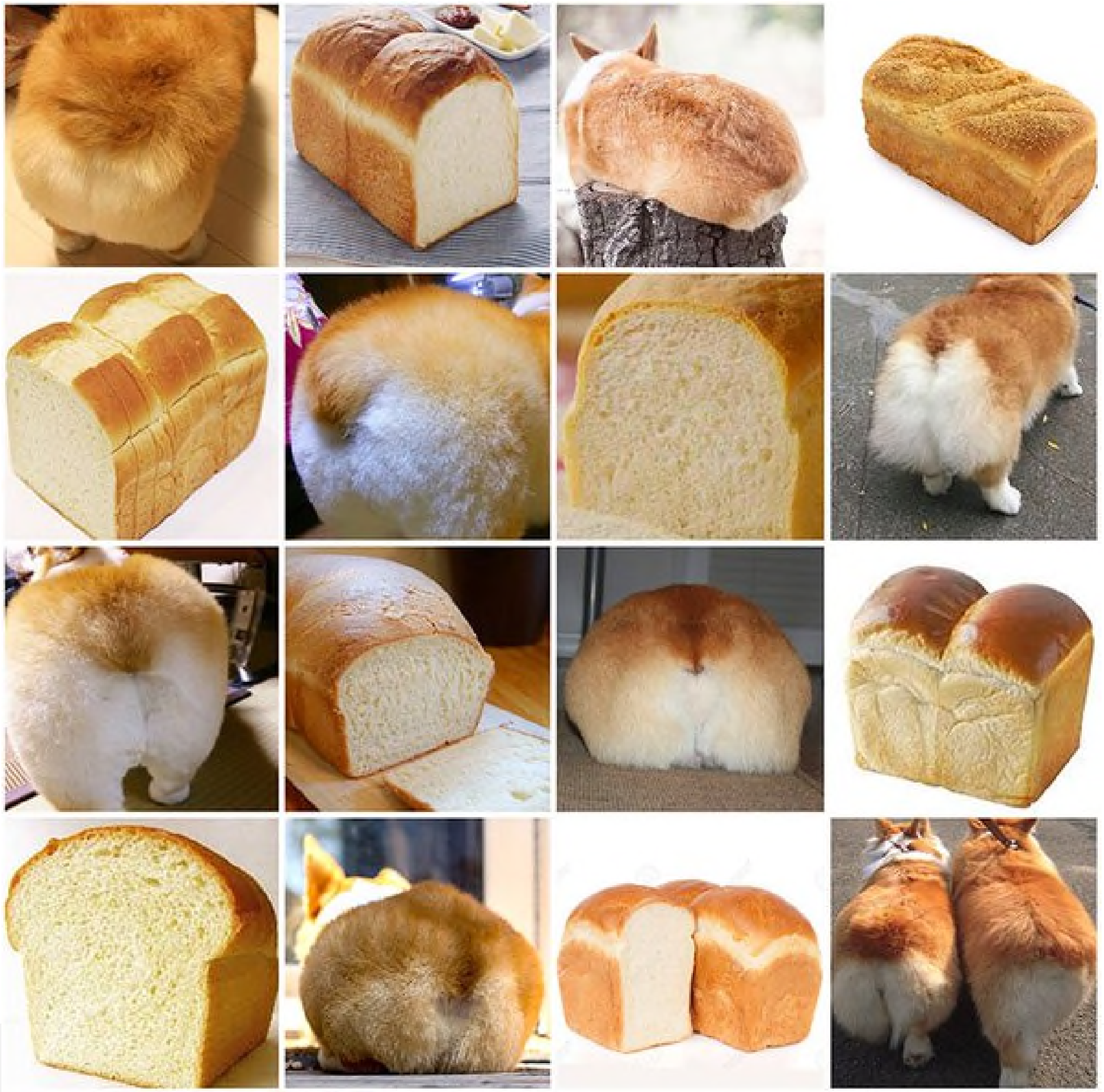


Image Classification is Challenging

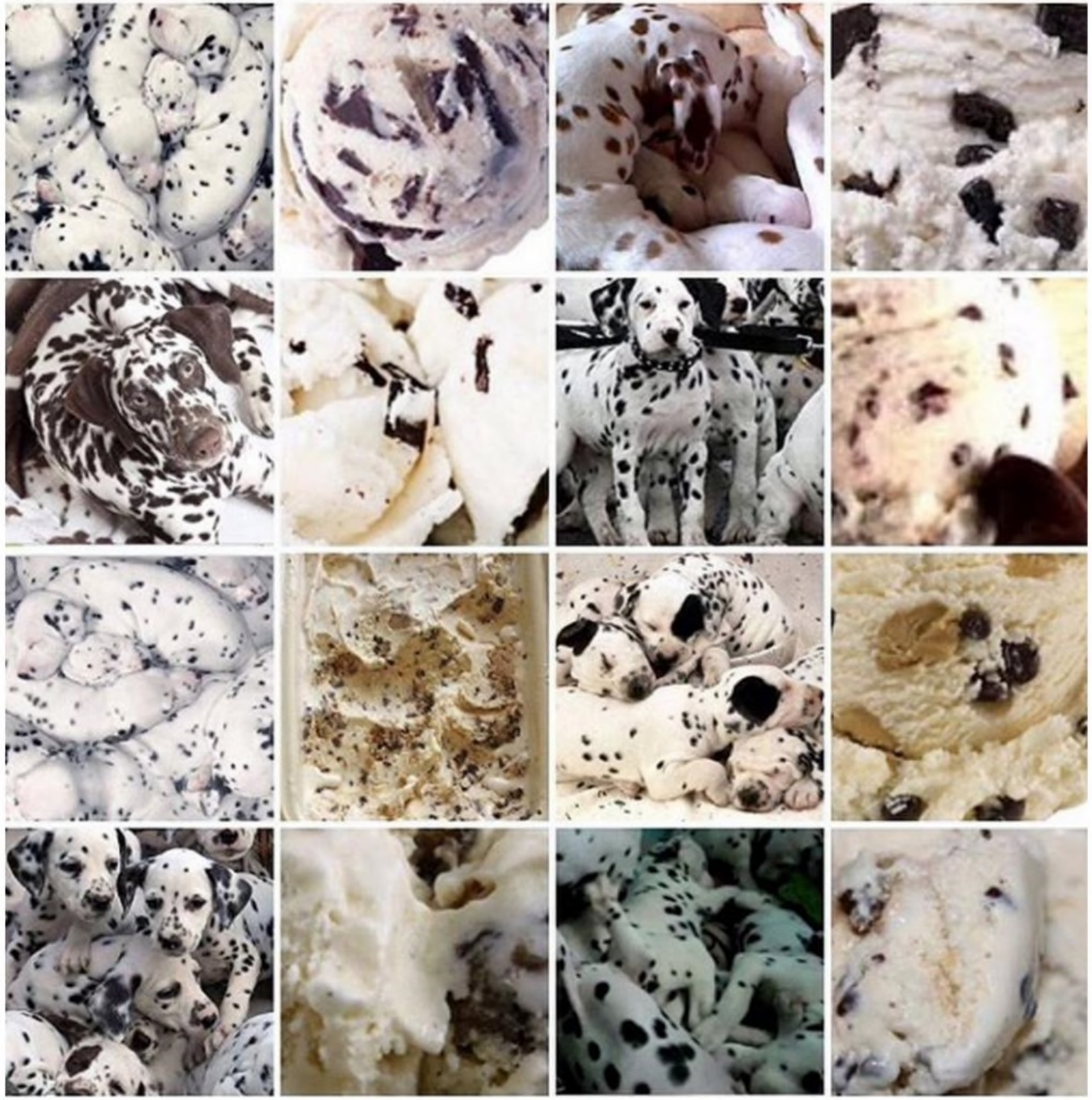


Image Classification is Challenging

Image Classification is Challenging

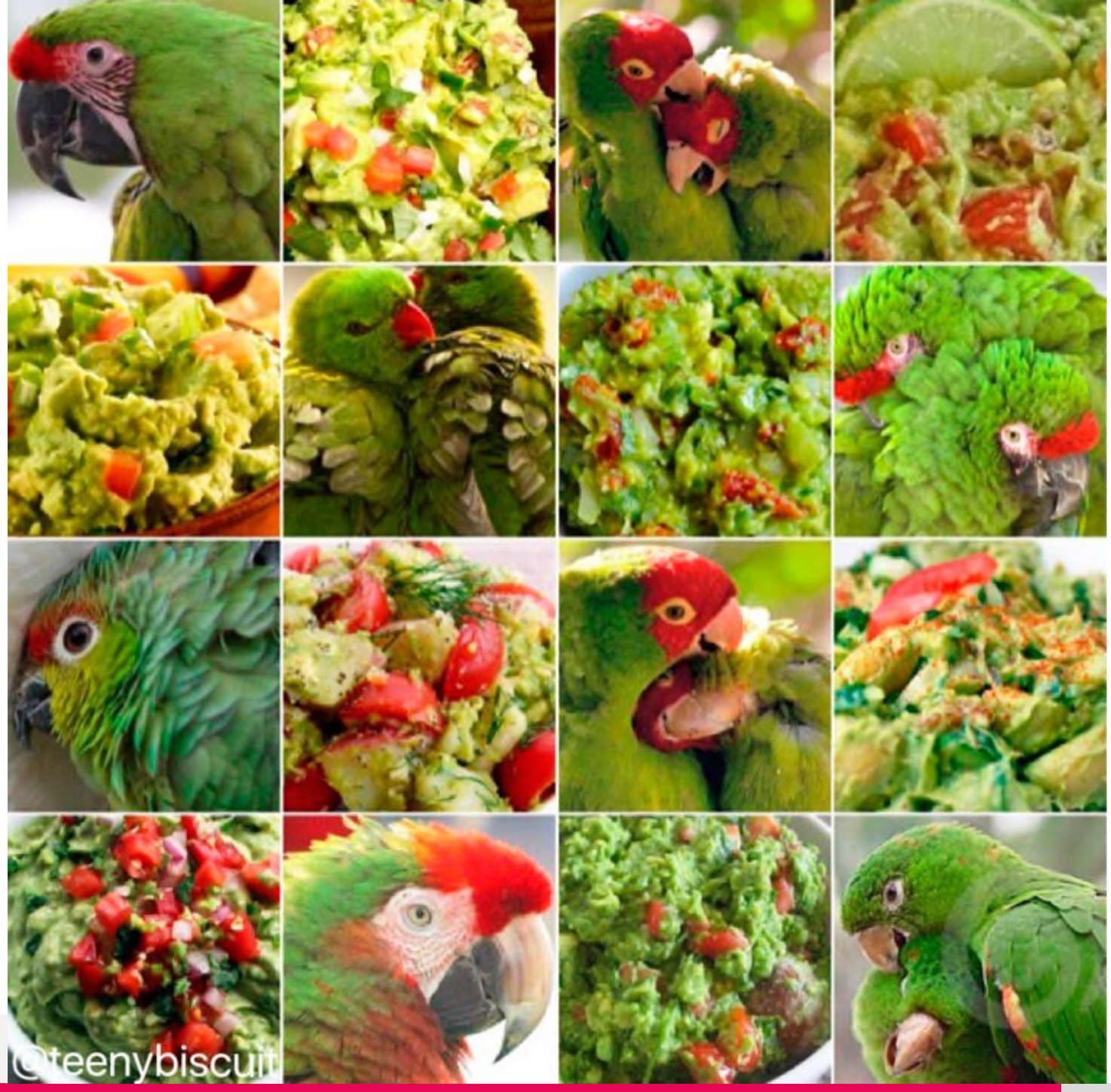


Image Classification is Challenging

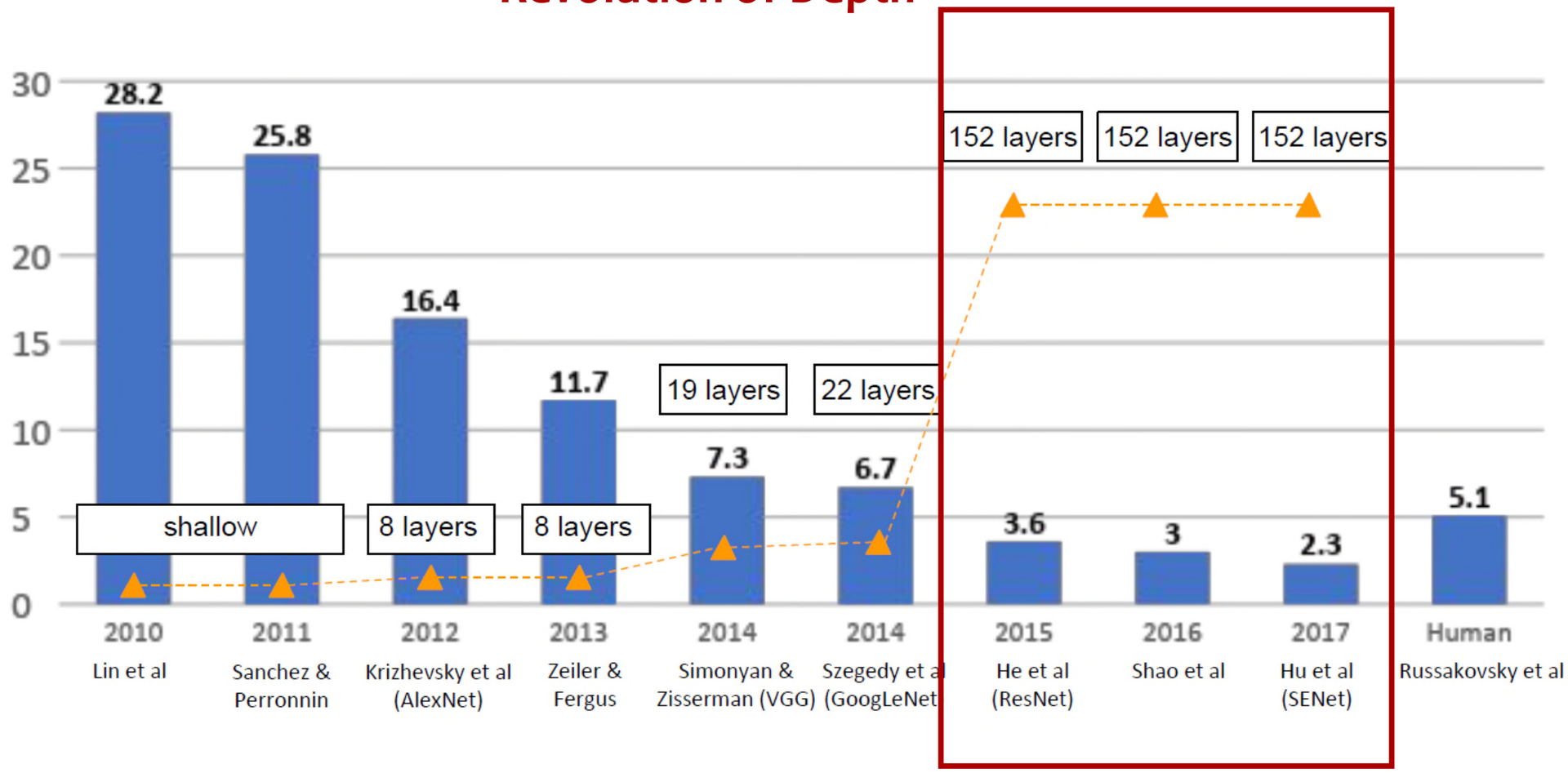
Res Net

ILSVRC 2015 Winner

Residual Blocks

More Deeper CNNs

Large Scale Visual Recognition Challenge winners



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Depth Re-Visited

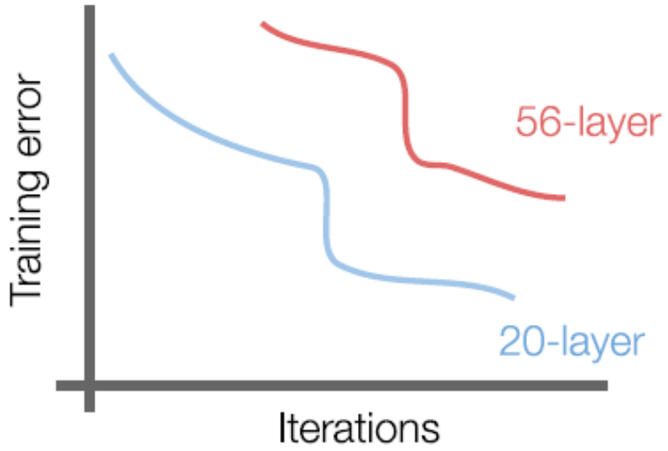
- Since AlexNet, the state-of-the-art CNN architecture is going deeper and deeper.
  - While AlexNet had only 5 convolutional layers, the VGG network and GoogleNet had 19 and 22 layers respectively.

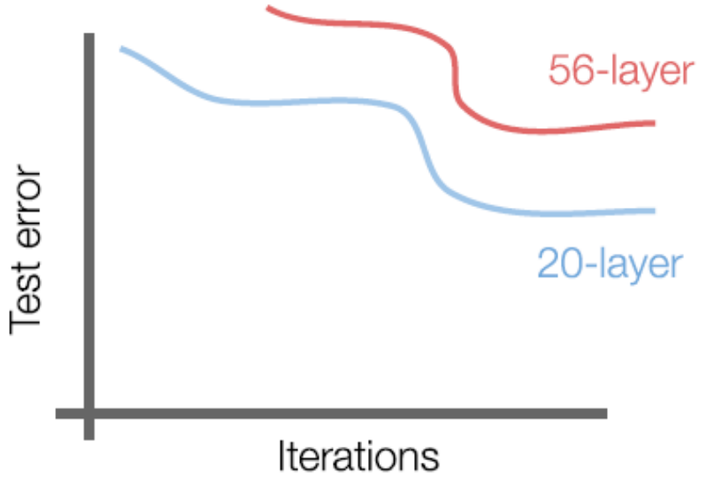
- However, increasing network depth does not work by simply stacking layers together.

- Deep networks are hard to train because of the notorious vanishing gradient problem
  - As the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small.
  - As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

Case Study: ResNet

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

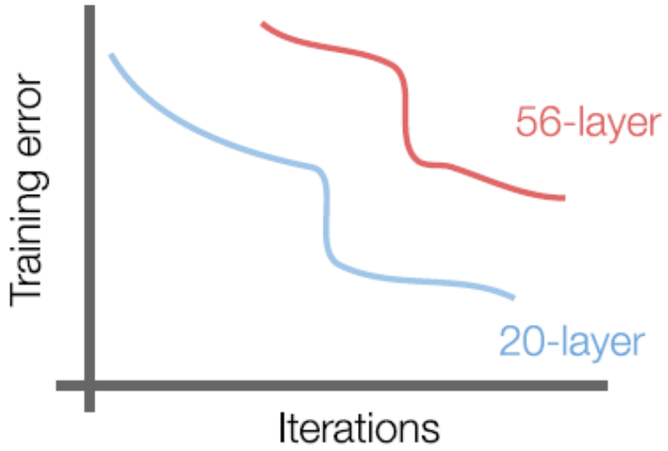


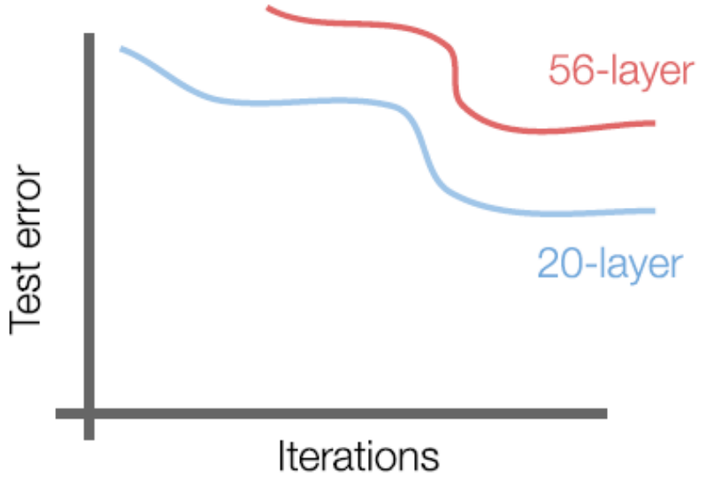


Q: What’s strange about these training and test curves?

Case Study: ResNet

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?





56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it’s not caused by overfitting!

Case Study: ResNet

Hypothesis:

The problem is an *optimization* problem, deeper models are harder to optimize typically due to problems related to vanishing gradient

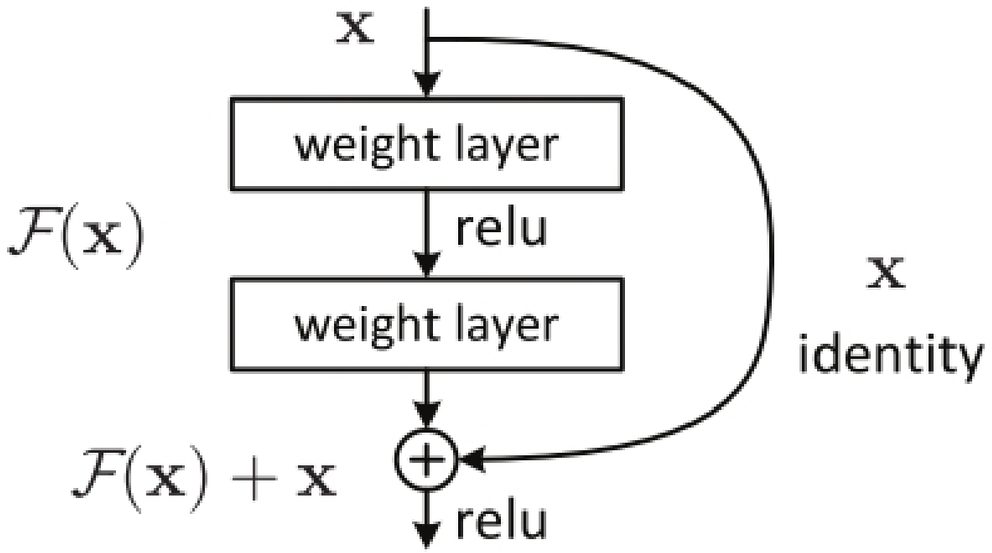
The deeper model should be able to perform at least as well as the shallower model

Inception adds an auxiliary loss in a middle layer as extra supervision

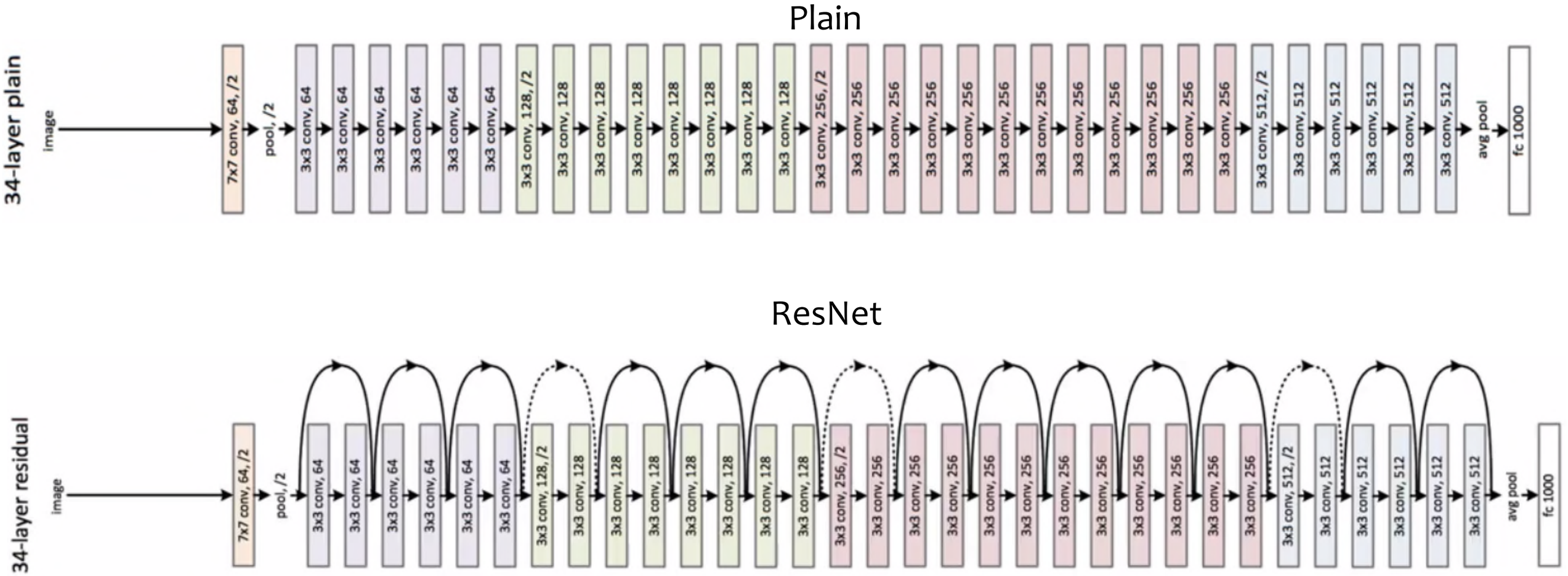
Another solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping

Identity shortcut connection

- The core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers



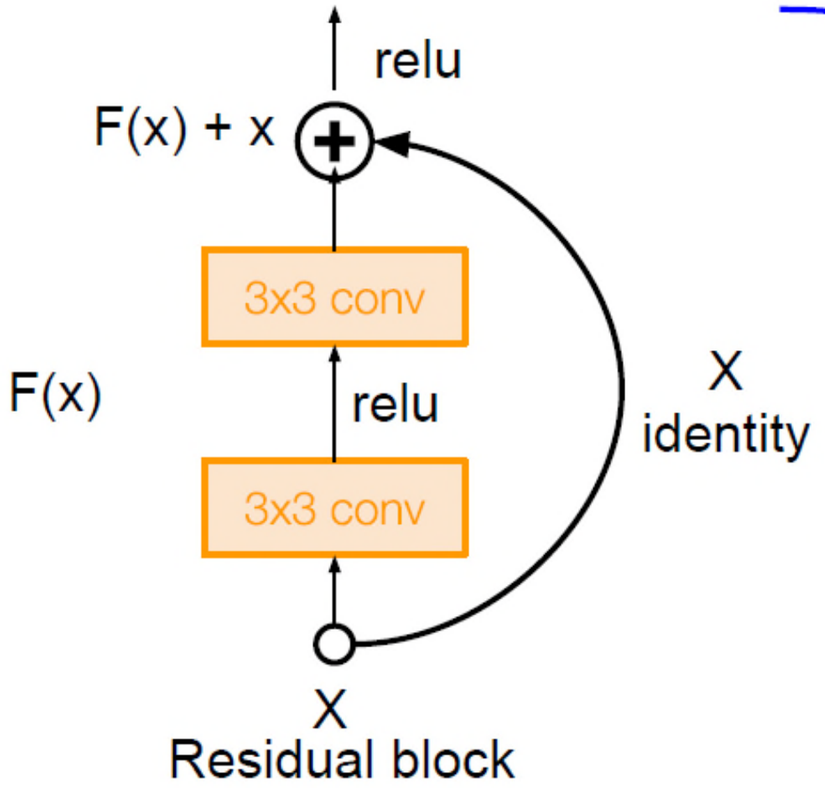
Case Study: ResNet

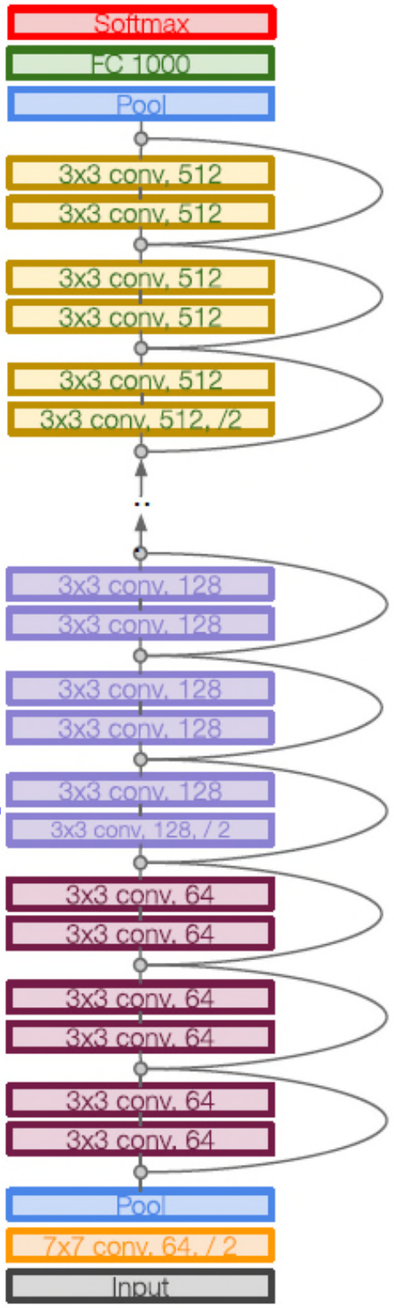


Case Study: ResNet

Full ResNet architecture:
- Stack residual blocks

- Every residual block has two 3x3 conv layers



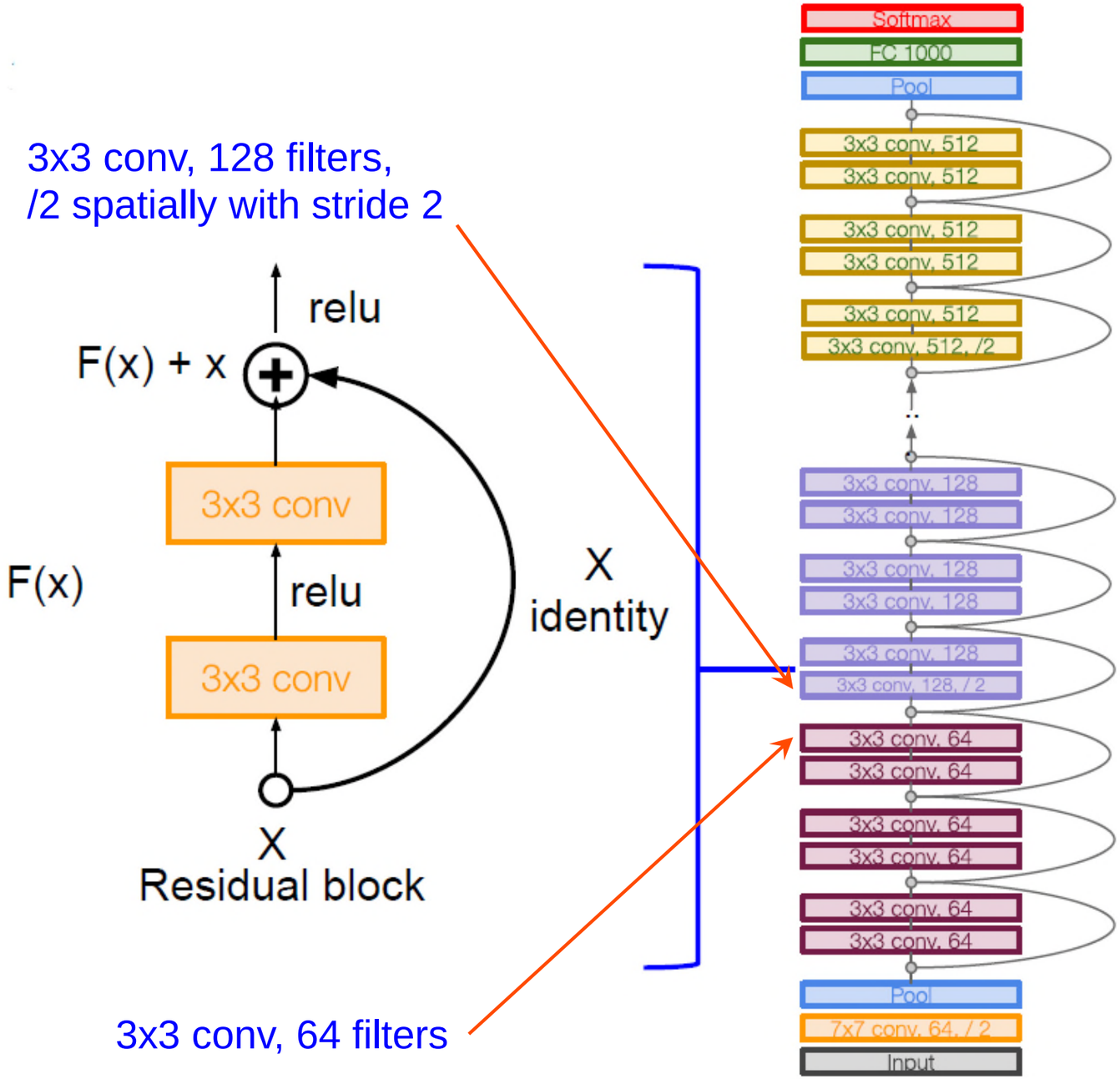


Case Study: ResNet

Full ResNet architecture:
- Stack residual blocks

- Every residual block has two 3x3 conv layers

- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)



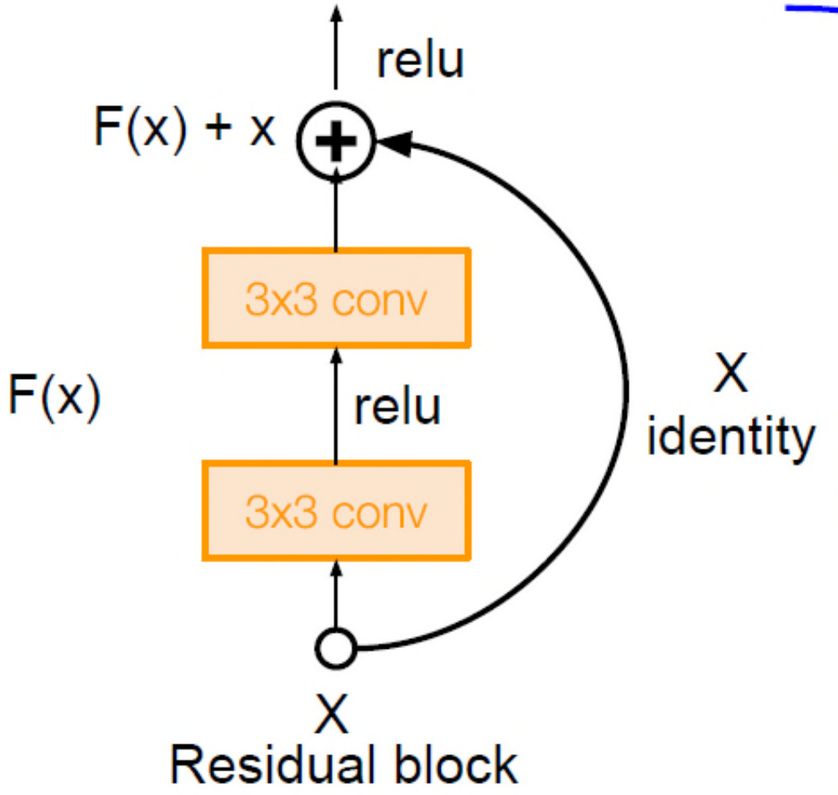
Case Study: ResNet

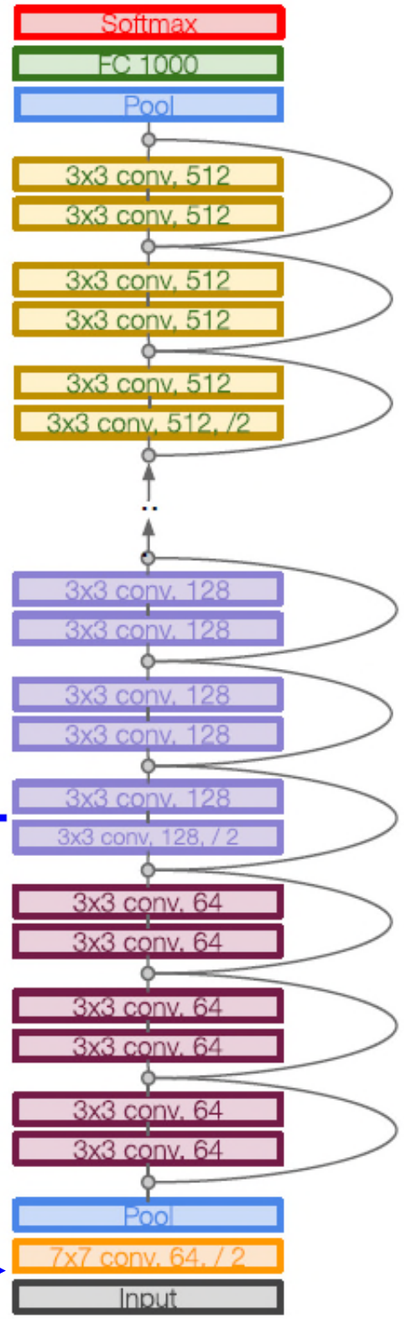
Full ResNet architecture:
- Stack residual blocks

- Every residual block has two 3x3 conv layers

- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)

- Additional conv layer at the beginning





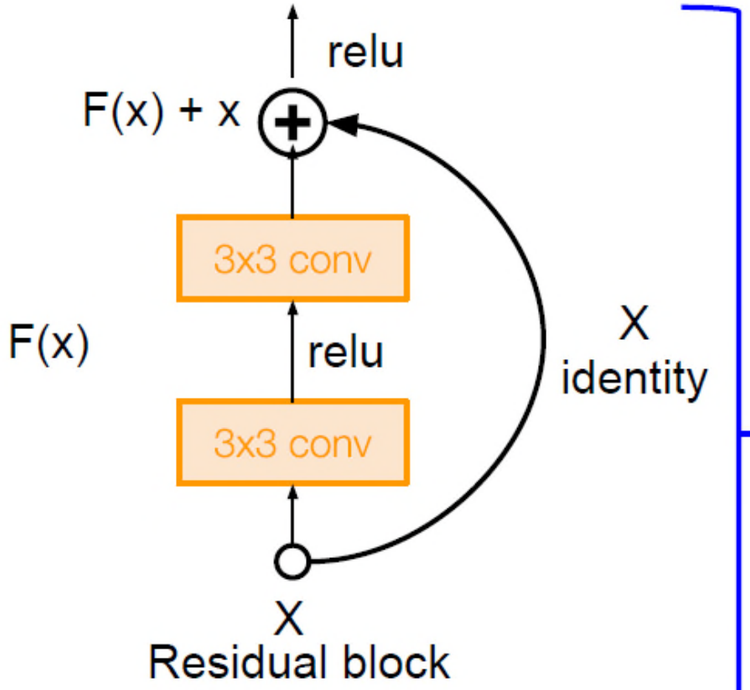
Beginning conv layer

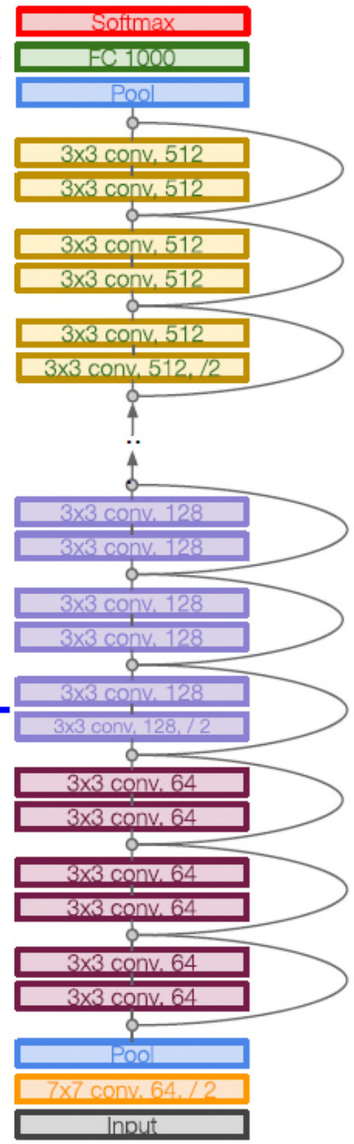
Case Study: ResNet

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)

No FC layers besides FC 1000 to output classes

Global average pooling layer after last conv layer

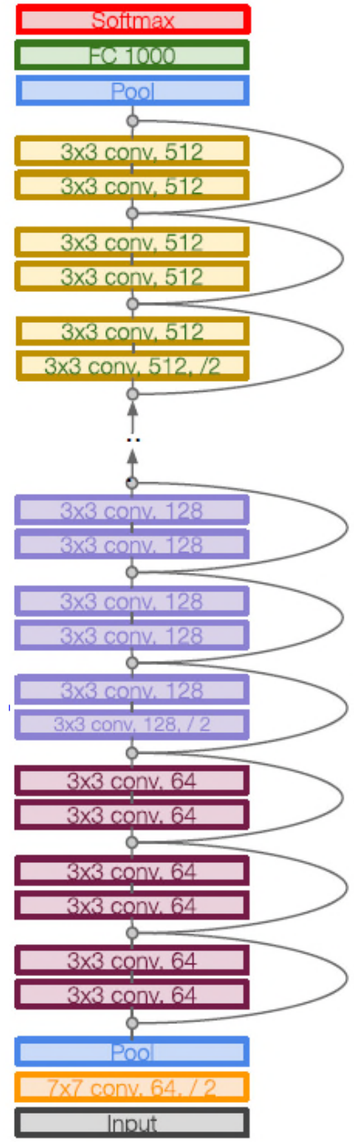




Case Study: ResNet

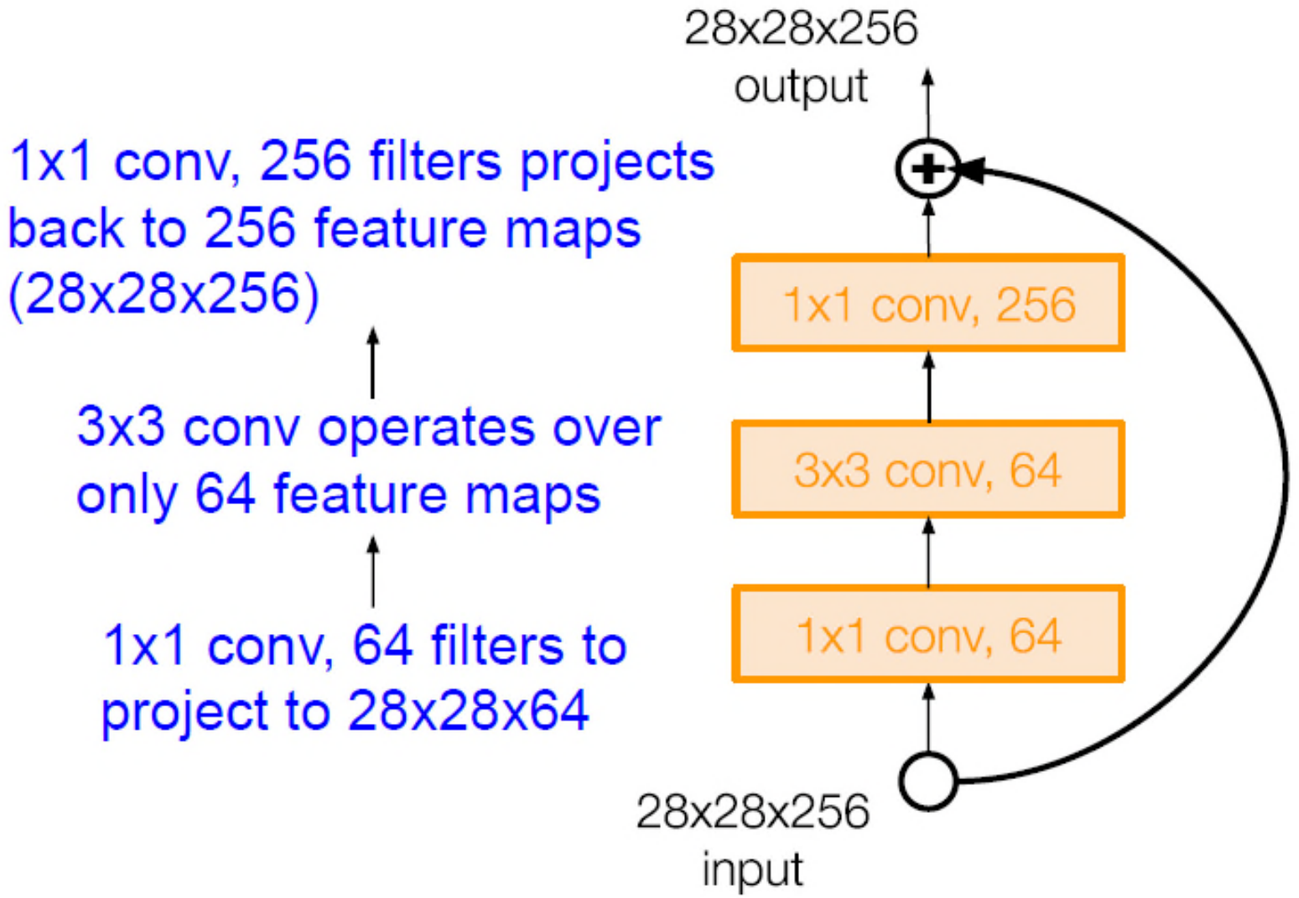
Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)

Total depths of 34, 50, 101, or 152 layers for ImageNet



Case Study: ResNet

For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



Case Study: ResNet

Training ResNet in practice:

- Batch Normalization after every CONV layer
- Xavier 2/ initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

Case Study: ResNet

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than “human performance”! (Russakovsky 2014)

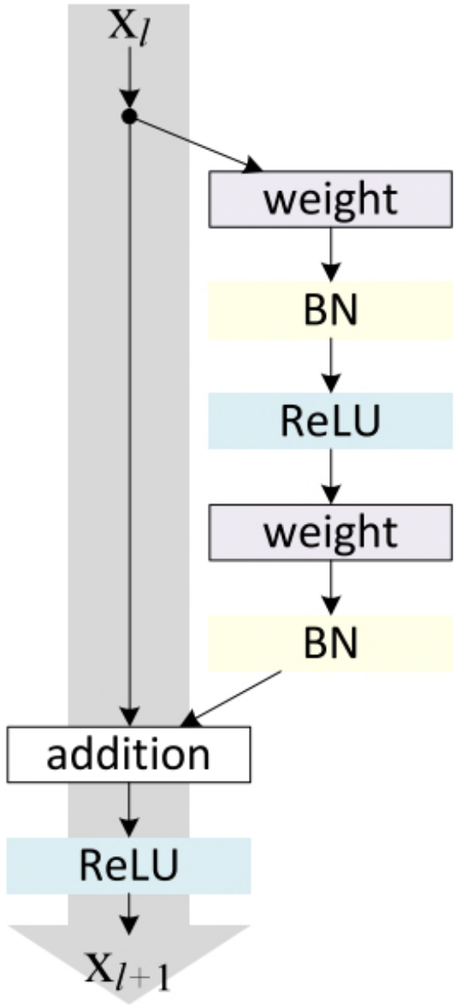
Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowing training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

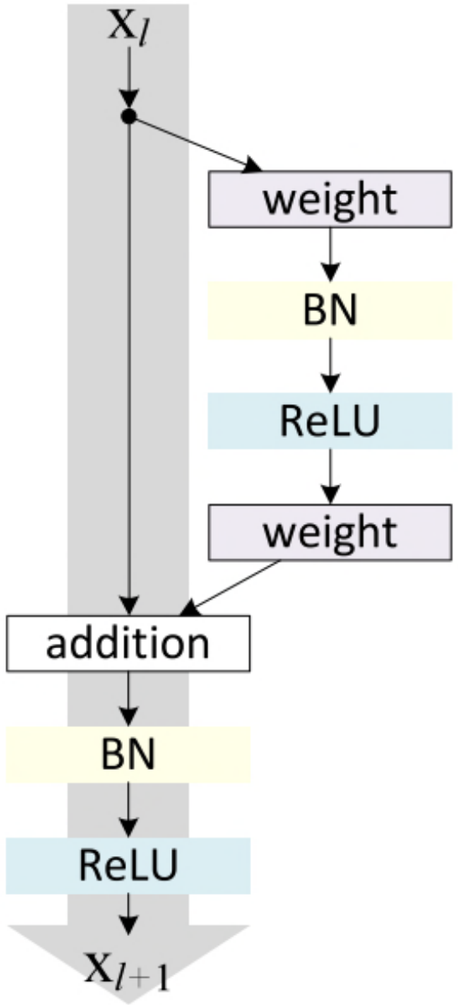
MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: “*Ultra-deep*” (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

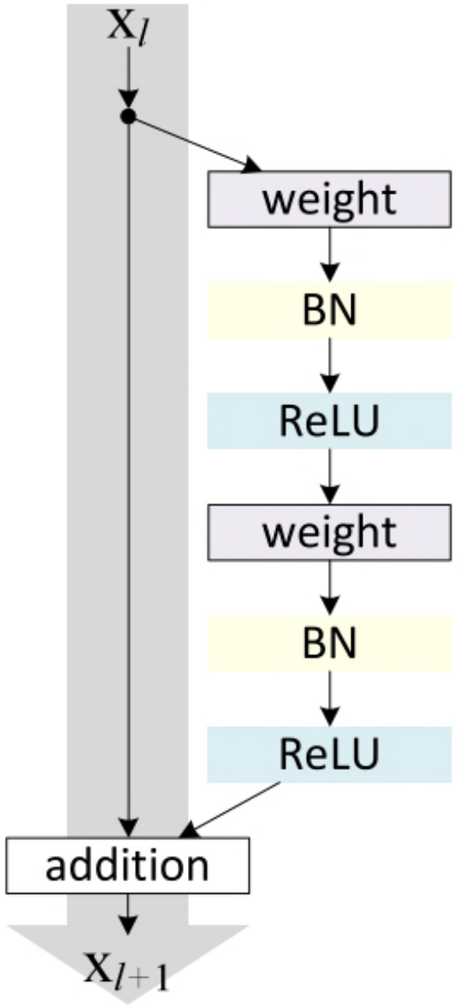
Variants of residual blocks



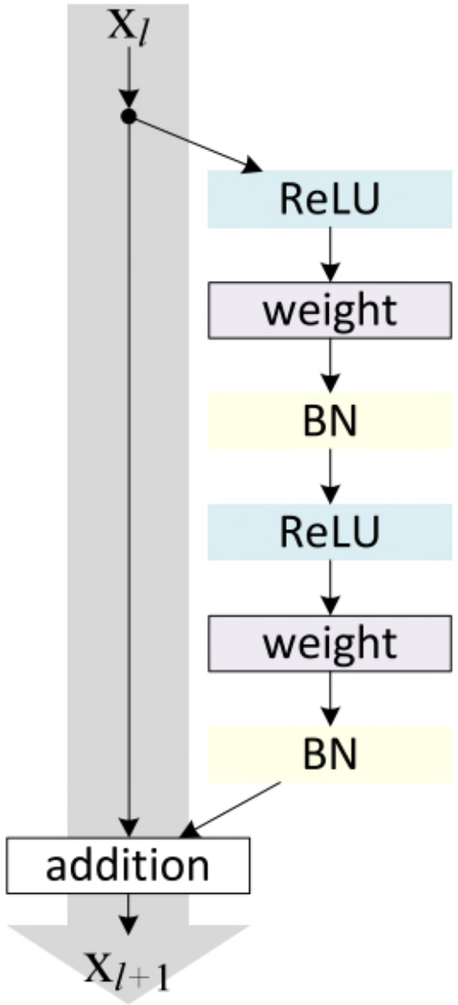
(a) original



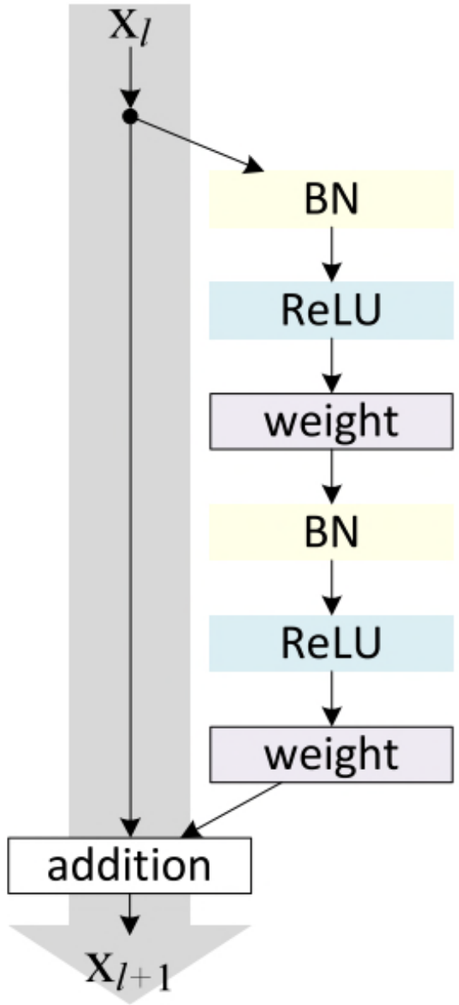
(b) BN after addition



(c) ReLU before addition



(d) ReLU-only pre-activation



(e) **full pre-activation**

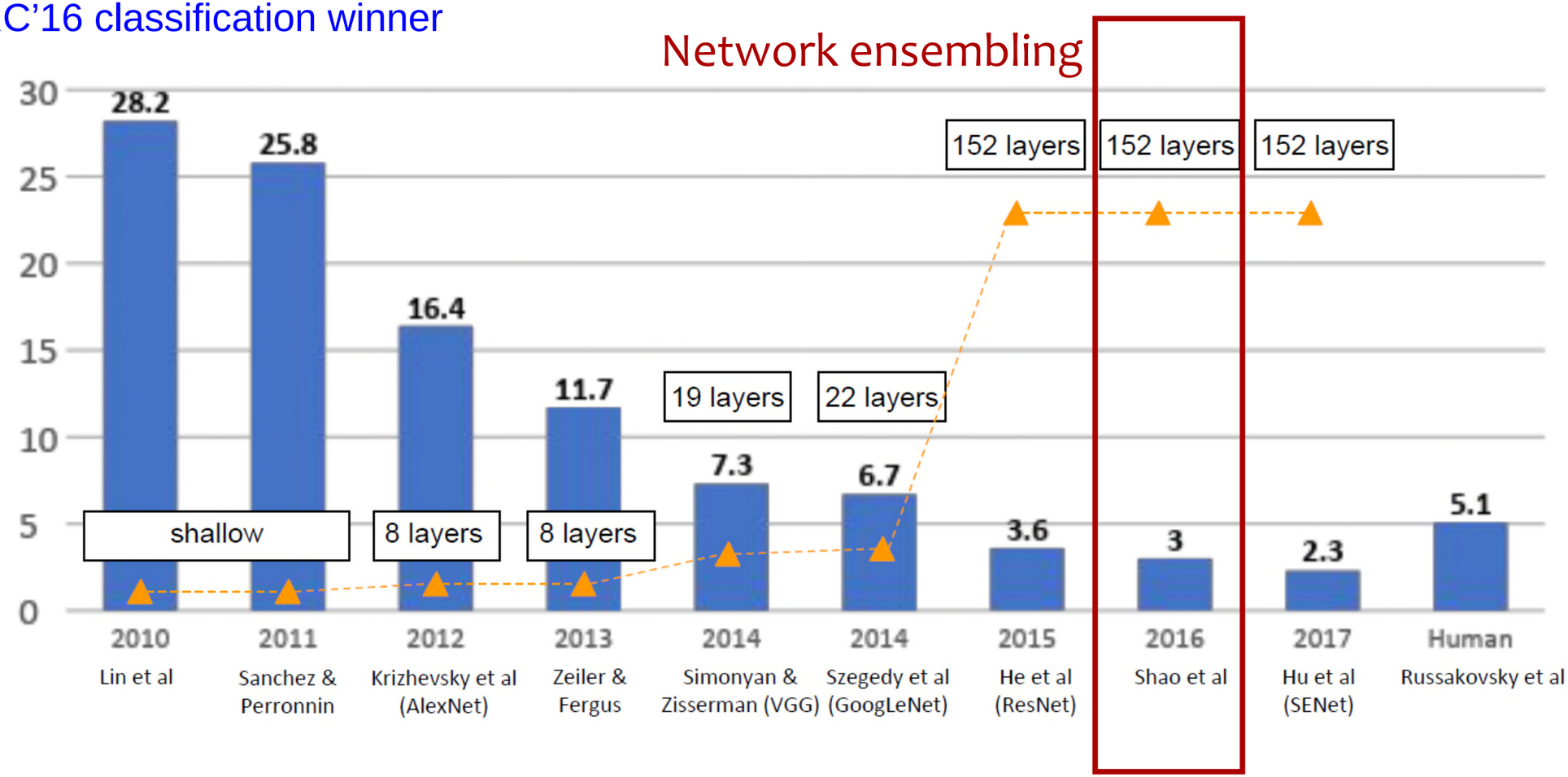
ResNeXt

ILSVRC 2016 Winner

More Deeper and More Wider CNNs

Large Scale Visual Recognition Challenge winners

- Multi-scale ensembling of Inception, Inception-Resnet, Resnet, Wide Resnet models
- ILSVRC'16 classification winner



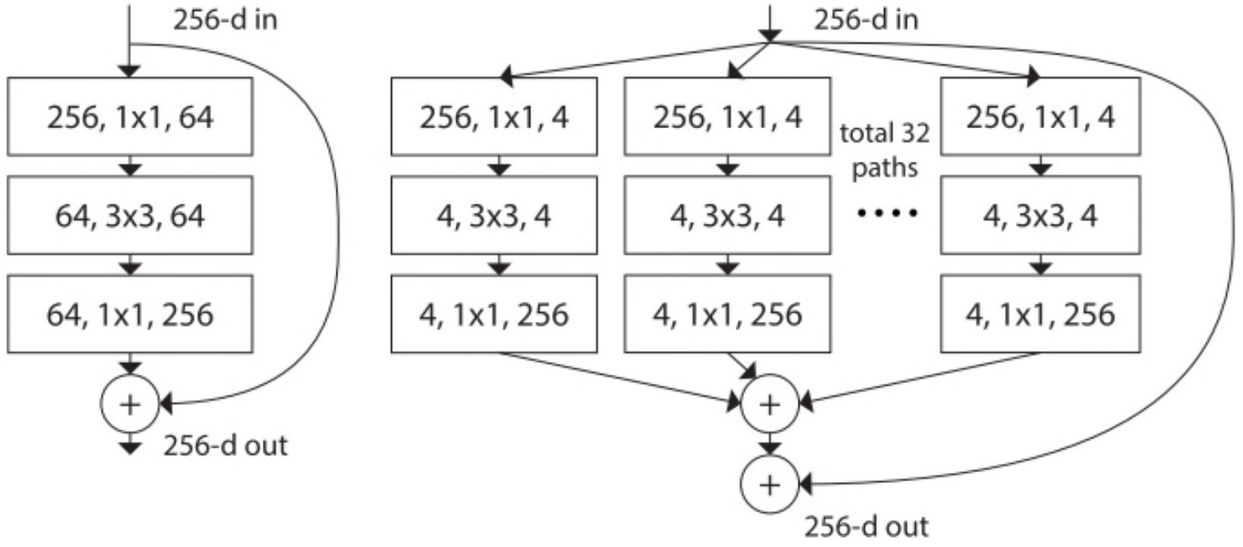
"Good Practices for Deep Feature Fusion"

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

ResNeXt

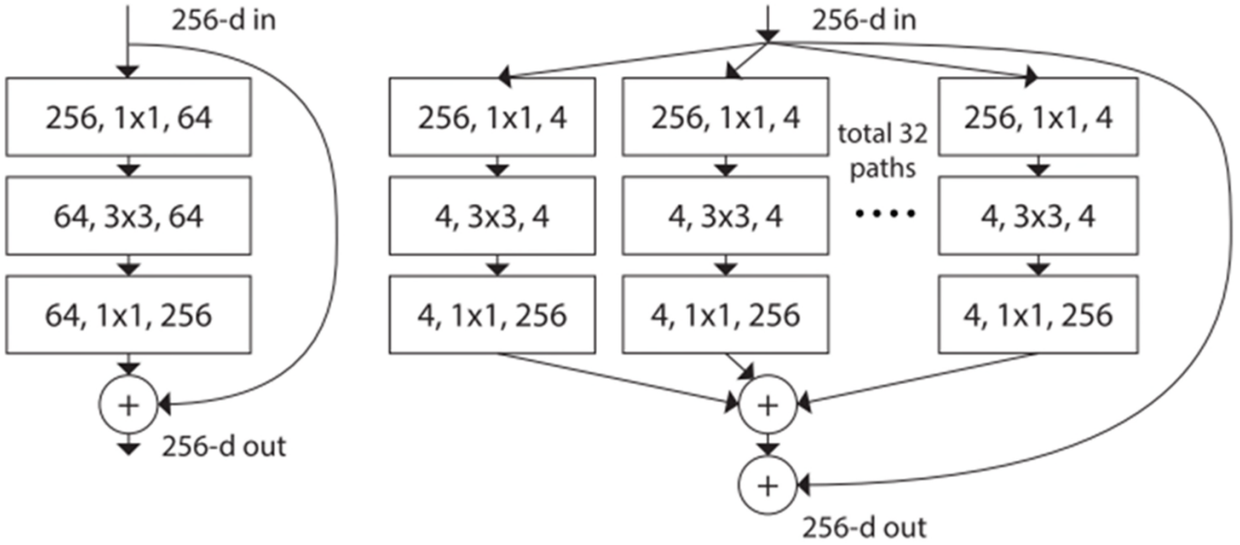
- ResNeXt won 2nd place in ILSVRC 2016 classification task and also showed performance improvements in Coco detection and ImageNet-5k set than their ResNet counter part.

- This is a very simple paper to read which introduces a new term called “cardinality”.
- The paper simply explains this term and make use of it in ResNet networks and does various ablation studies.



ResNeXt

- It follows a split-transform-aggregate strategy.
- The number of paths inside the ResNeXt block is defined as cardinality. In the diagram C=32
- All the paths contain the same topology.
- Instead of having high depth and width, Having high cardinality helps in decreasing validation error.



- Both the architectures have different width.
  - Layer-1 in ResNet has one conv layer with 64 width, while layer-1 in ResNext has 32 different conv layers with 4 width (32*4 width).
- Despite the larger overall width in ResNeXt, both the architectures have the same number of parameters(~70k)

(ResNet 256*64+3*3*64*64+64*26) (ResNeXt C*(256*d+3*3*d*d+d*256), with C=32 and d=4)

Resnet and ResNeXt Comparison

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

Inside the brackets=> residual block shape
Outside the bracket => number of stacked blocks

ResNeXt (Ablation Experiments)

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Cardinality vs width: with C increasing from 1 to 32, we can clearly see a decrease in top-1 % error rate. Therefore, Increasing the **C** by decreasing the width has improved the performance of the model.

Increasing Cardinality vs Deeper/Wider: Basically 3 cases were studied.

1) Increasing the number of layers to 200 from 101.
2) Going wider by increasing the bottleneck width.
3) Increasing cardinality by doubling C.

They have observed that increasing the C gave better performance improvements.

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

MobileNet (variants)
ShuffleNet
Xception

Depthwise Separable Convolutions

Separable convolutions

- Two main types of separable convolutions:
  - Spatial separable convolutions
  - Depthwise separable convolutions

- A spatial separable convolution simply divides a kernel into two, smaller kernels.
- The most common case would be to divide a 3x3 kernel into a 3x1 and 1x3 kernel, like so

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

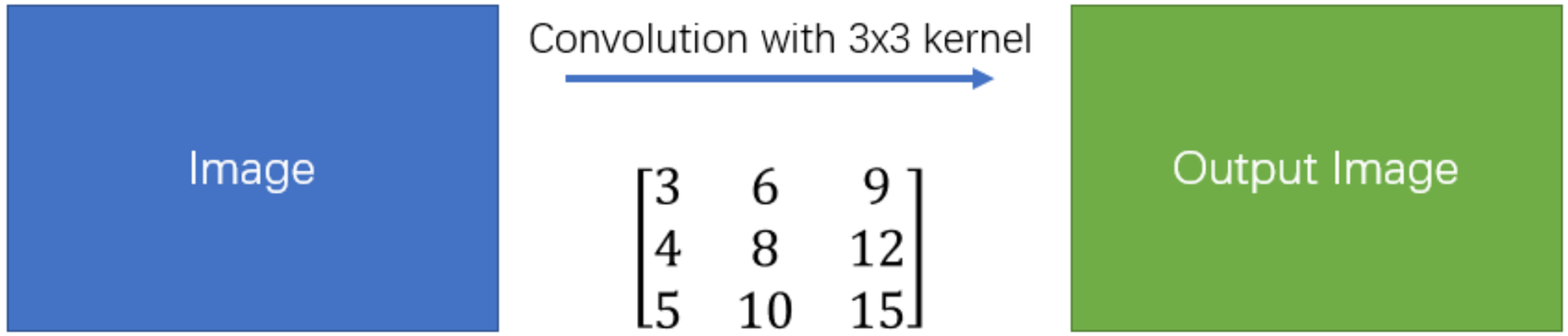
Spatial Separable Convolutions

- One of the most famous convolutions that can be separated spatially is the Sobel kernel, used to detect edges:

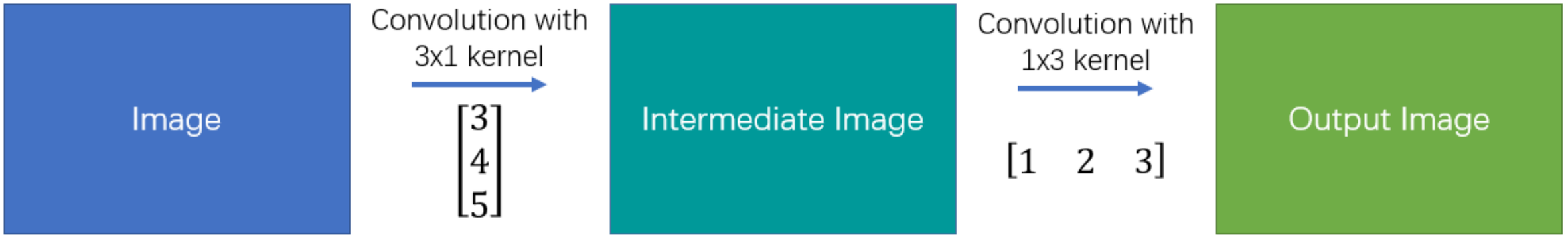
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Spatial Separable Convolutions

Simple Convolution



Spatial Separable Convolution



Depthwise Separable Convolutions

- Unlike spatial separable convolutions, depthwise separable convolutions work with kernels that cannot be “factored” into two smaller kernels.

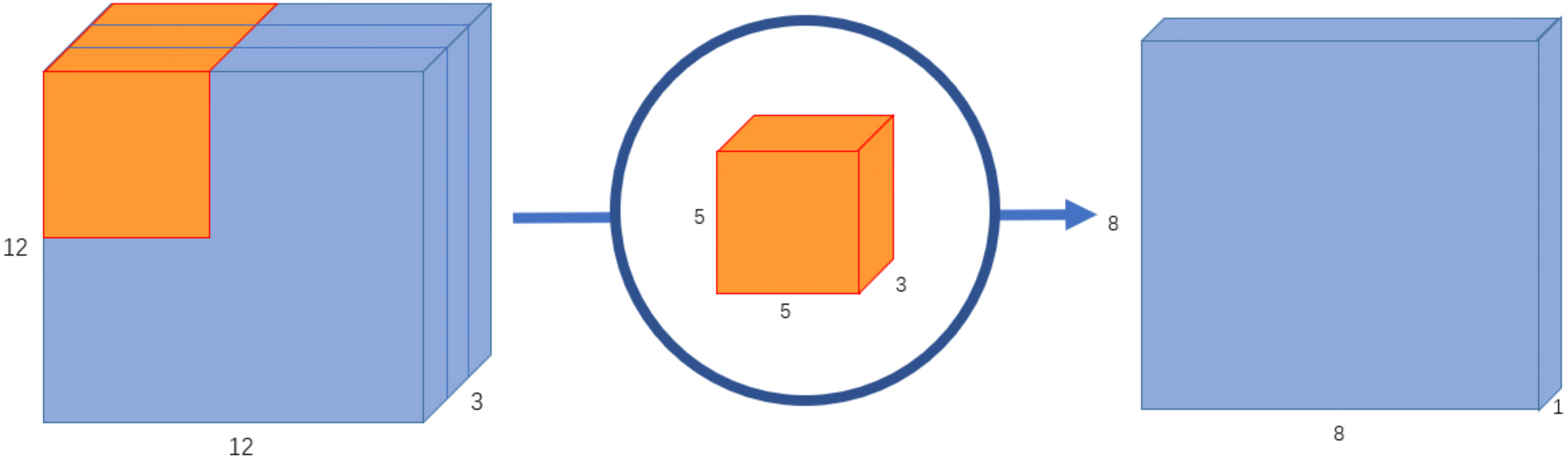
- The depthwise separable convolution is so named because it deals not just with the spatial dimensions, but with the depth dimension — the number of channels — as well

Channel Representation: An input image may have 3 channels: RGB. After a few convolutions, an image may have multiple channels.

An image with 64 channels has 64 different interpretations of that image.

Depthwise Separable Convolutions

- http://setosa.io/ev/image-kernels/

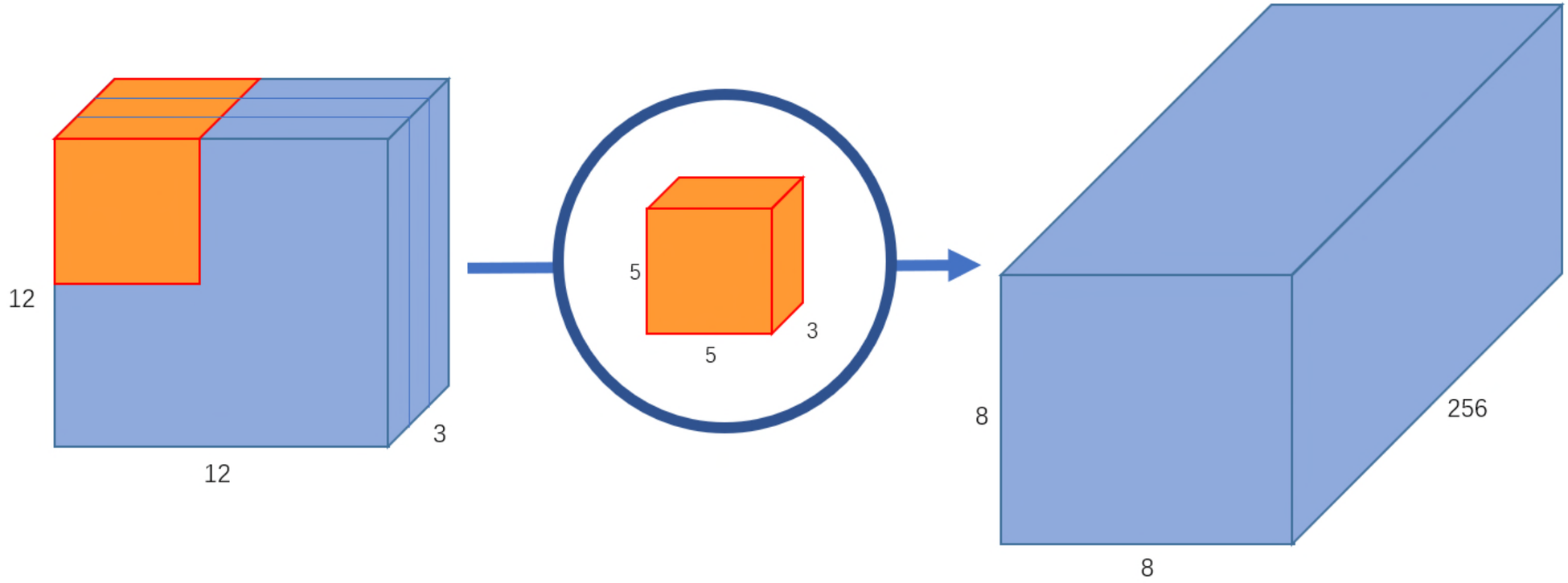


A normal convolution with 8x8x1 output

Depthwise Separable Convolutions

What if we want to increase the number of channels in our output image?
What if we want an output of size 8x8x256?

Well, we can create 256 kernels to create 256 8x8x1 images, then stack them up together to create a 8x8x256 image output.

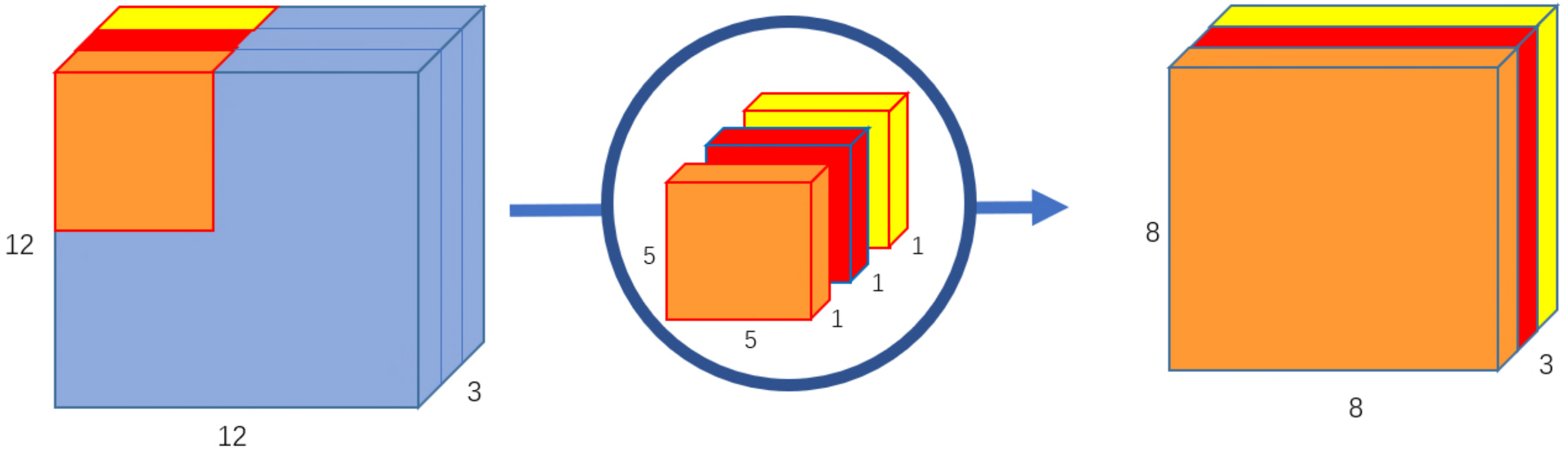


12x12x3 — (5x5x3x256) — >12x12x256
(Where 5x5x3x256 represents the height, width, number of input channels, and number of output channels of the kernel).

Depthwise Separable Convolutions

https://youtu.be/D_VJoaSew7Q

In the first part, **depthwise convolution**, we give the input image a convolution without changing the depth. We do so by using 3 kernels of shape 5x5x1.



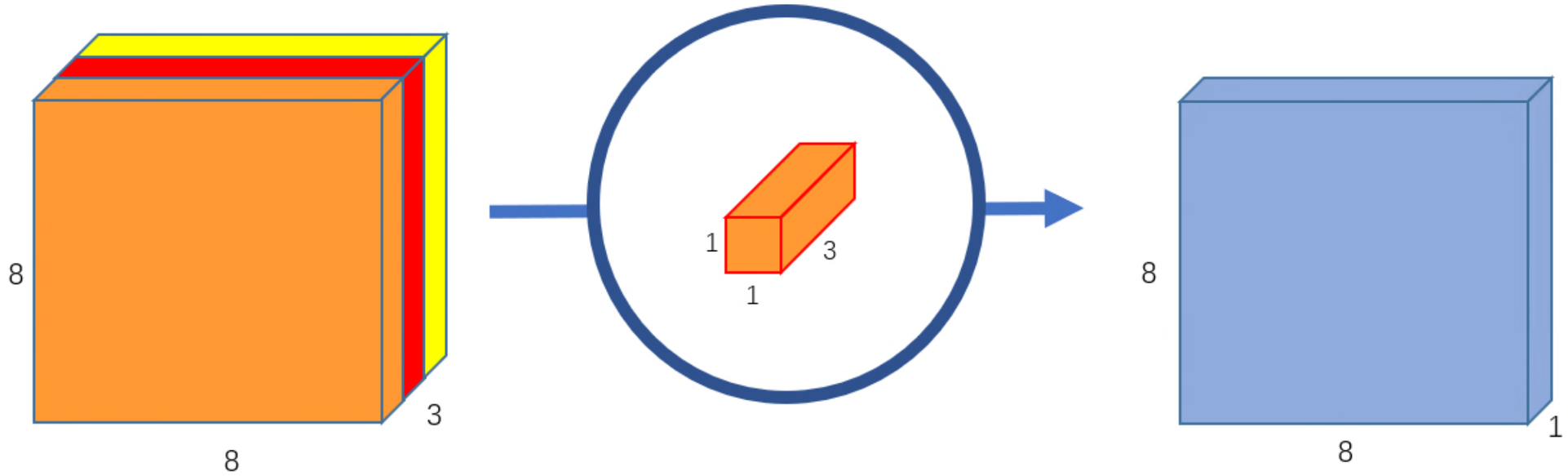
Each 5x5x1 kernel iterates 1 channel of the image (note**: 1 channel**, not all channels), getting the scalar products of every 25 pixel group, giving out a 8x8x1 image. Stacking these images together creates a 8x8x3 image.

Depthwise Separable Convolutions

- Remember, the original convolution transformed a 12x12x3 image to a 8x8x256 image.
- Currently, the depthwise convolution has transformed the 12x12x3 image to a 8x8x3 image.
  - Now, we need to increase the number of channels of each image

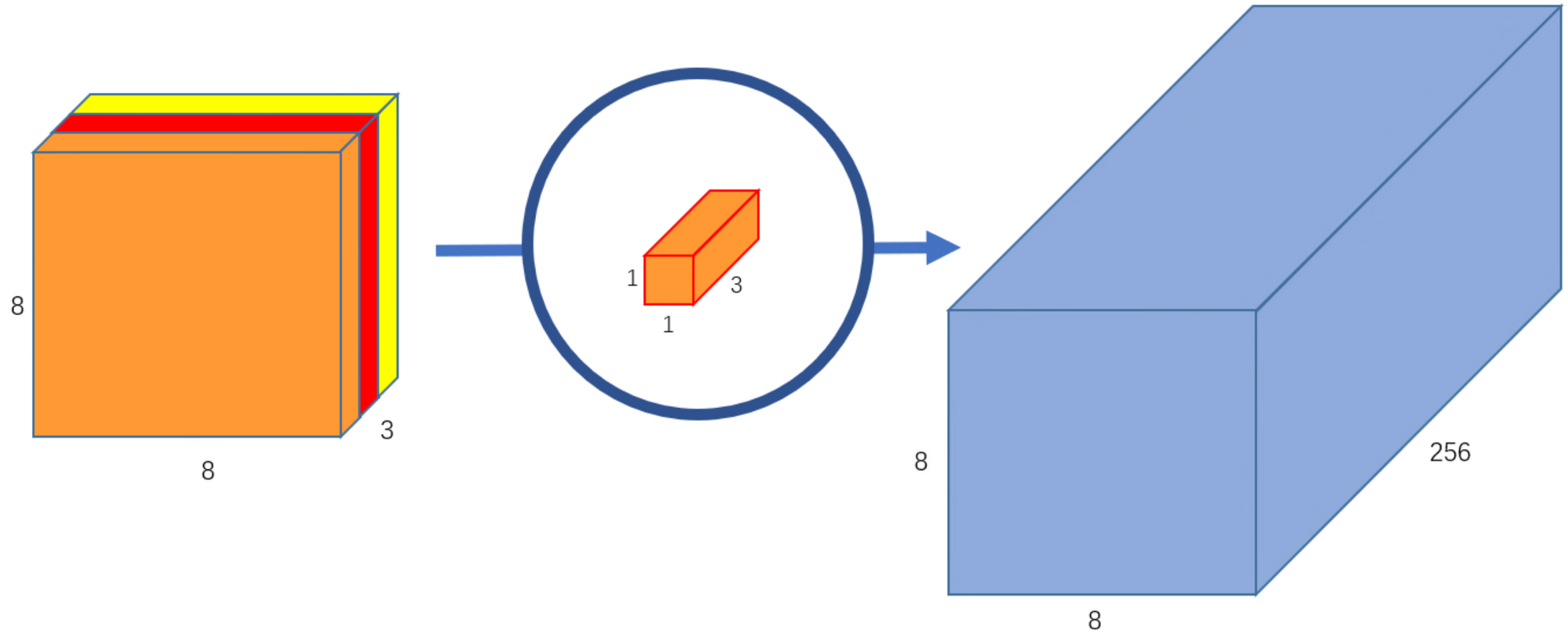
Depthwise Separable Convolutions

- The **pointwise convolution** is so named because it uses a 1x1 kernel, or a kernel that iterates through every single point.
- This kernel has a depth of however many channels the input image has;
  - in our case, 3.
- Therefore, we iterate a 1x1x3 kernel through our 8x8x3 image, to get a 8x8x1 image.



Depthwise Separable Convolutions

- We can create 256 1x1x3 kernels that output a 8x8x1 image each to get a final image of shape 8x8x256



Normal : 12x12x3 — (5x5x3x256) →12x12x256,
Separable: we can illustrate this new convolution as 12x12x3 — (5x5x1x3) — > (1x1x3x256) — >12x12x256.

Depthwise Separable Convolutions (multiplications)

- Normal Convolutions
  - There are 256 5x5x3 kernels that move 8x8 times. That's 256x3x5x5x8x8=**1,228,800** multiplications.

- Depthwise Separable Convolutions
  - In the depthwise convolution, we have 3 5x5x1 kernels that move 8x8 times. That's 3x5x5x8x8 = 4,800 multiplications.
  - In the pointwise convolution, we have 256 1x1x3 kernels that move 8x8 times. That's 256x1x1x3x8x8=49,152 multiplications.
  - Adding them up together, that's **53,952** multiplications.

MobileNet v1

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Batch Normalization (BN) and ReLU are applied after each convolution

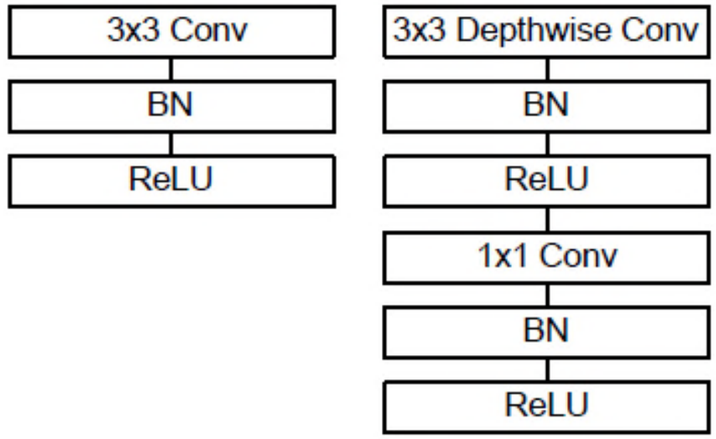


Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

MobileNet only got 1% loss in accuracy, but the Mult-Adds and parameters are reduced tremendously.

Self Study

- MobileNet, Shuffle Net, EFF Net
  - https://towardsdatascience.com/3-small-but-powerful-convolutional-networks-27ef86faa42d

- **Xception**: Deep Learning with Depthwise Separable Convolutions
  - https://arxiv.org/abs/1610.02357

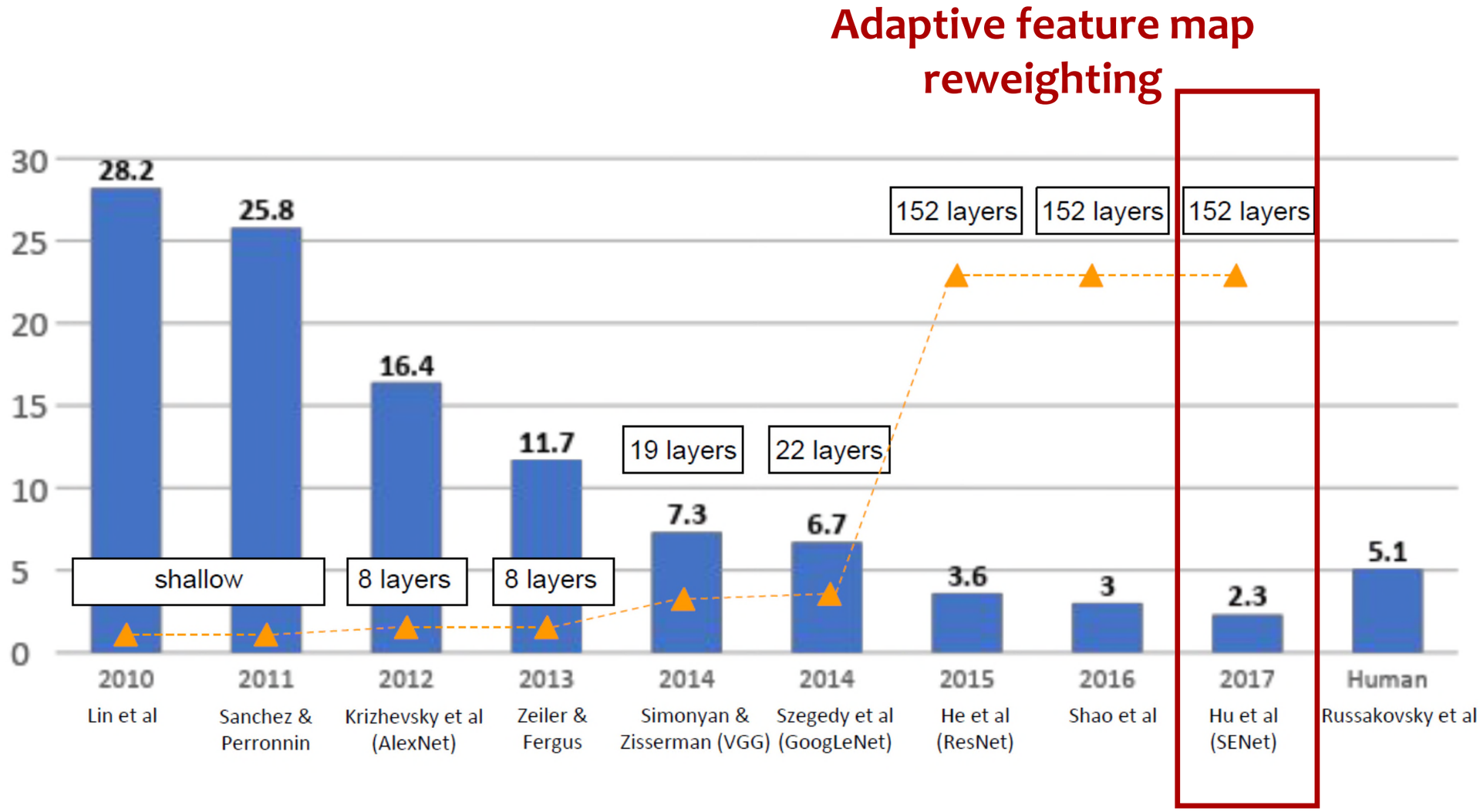
- Review: **Xception** — With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification)
  - https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

Squeeze-and-Excitation Networks (SENet)

ILSVRC 2016 Winner

Adaptive feature map reweighting

Large Scale Visual Recognition Challenge winners



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

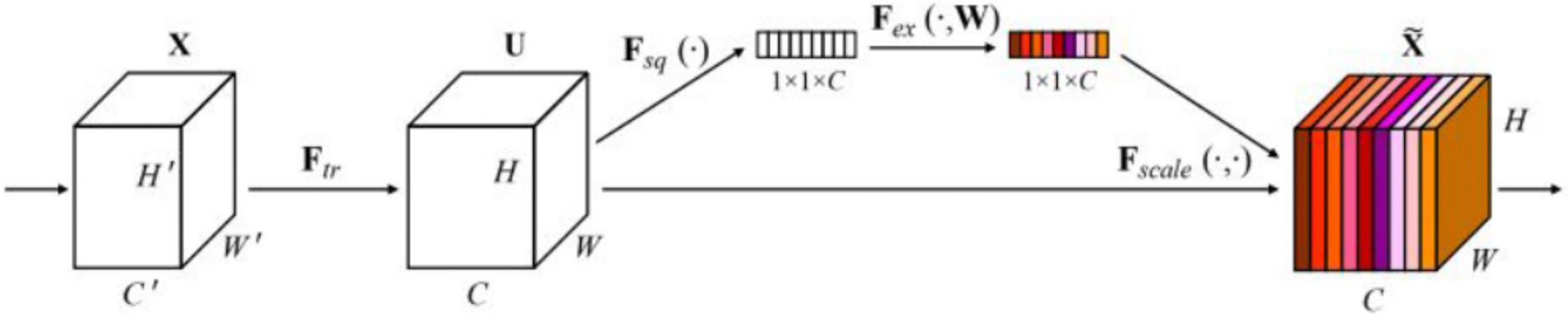
Squeeze-and-Excitation Networks (SENet)

- Let's add parameters to each channel of a convolutional block so that the network can adaptively adjust the weighting of each feature map

- All you need to understand for now is that the network weights each of its channels equally when creating the output feature maps

- SENets are all about changing this by adding a content aware mechanism to weight each channel adaptively

- In it's most basic form this could mean adding a single parameter to each channel and giving it a linear scalar how relevant each one is

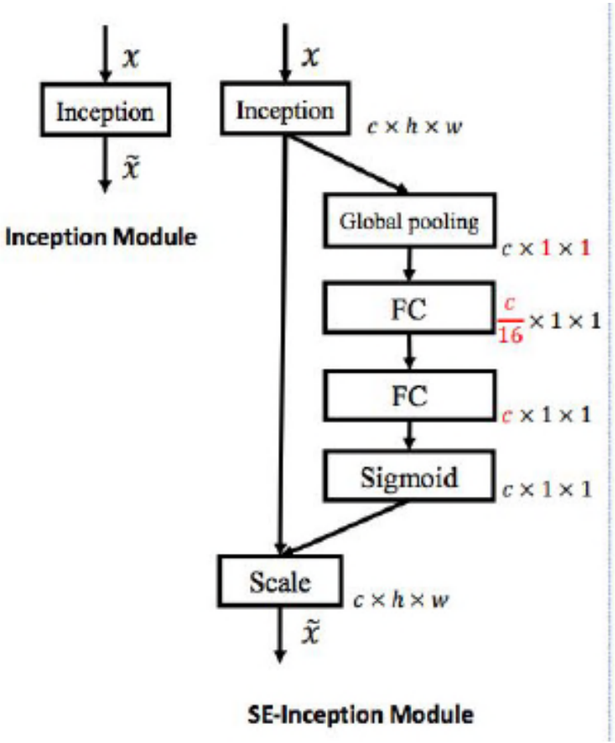


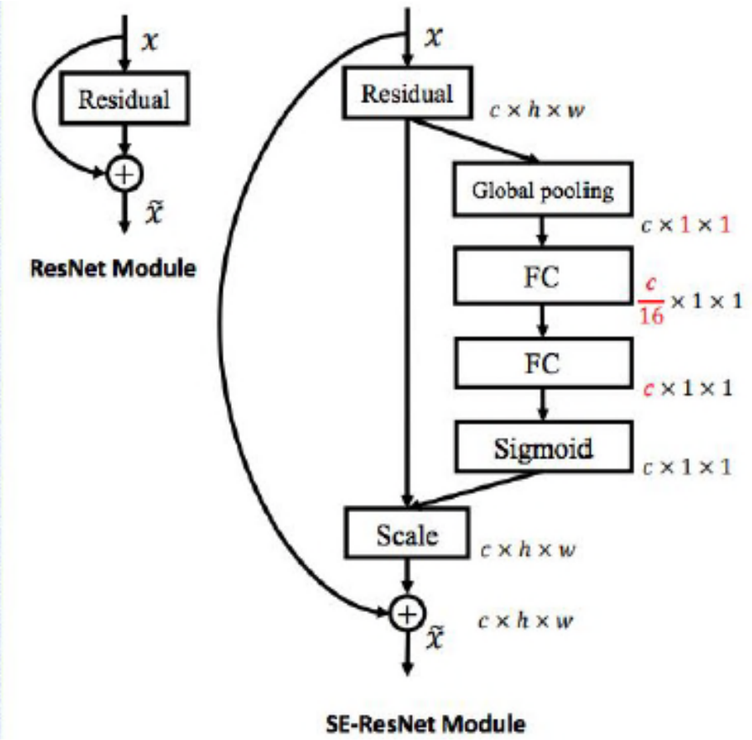
Squeeze-and-Excitation Networks (SENet)

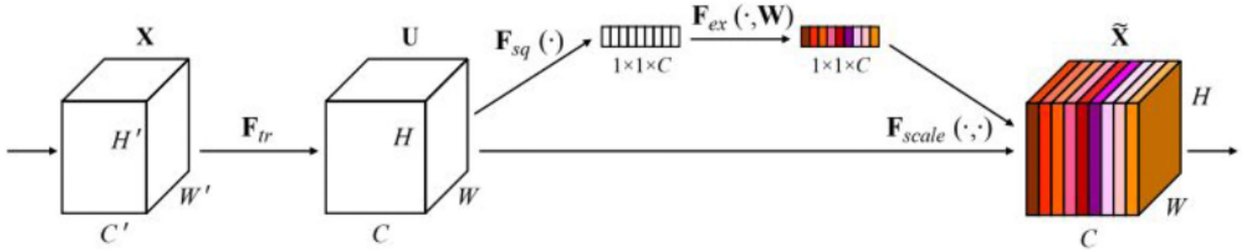
- Add a “feature recalibration” module that learns to adaptively reweight feature maps

- Global information (global avg. pooling layer) + 2 FC layers used to determine feature map weights

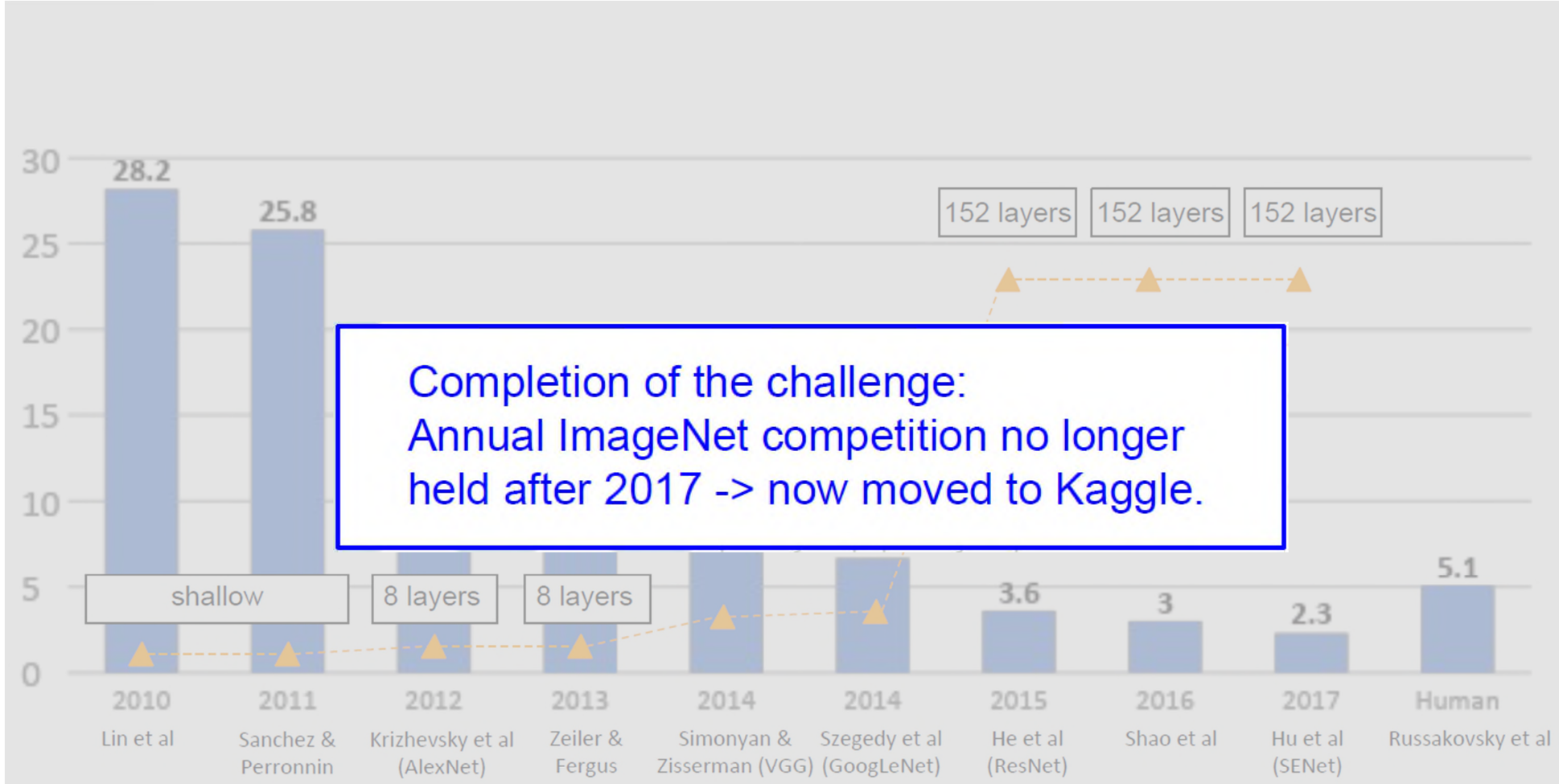
- ILSVRC’17 classification winner (using ResNeXt-152 as a base architecture)





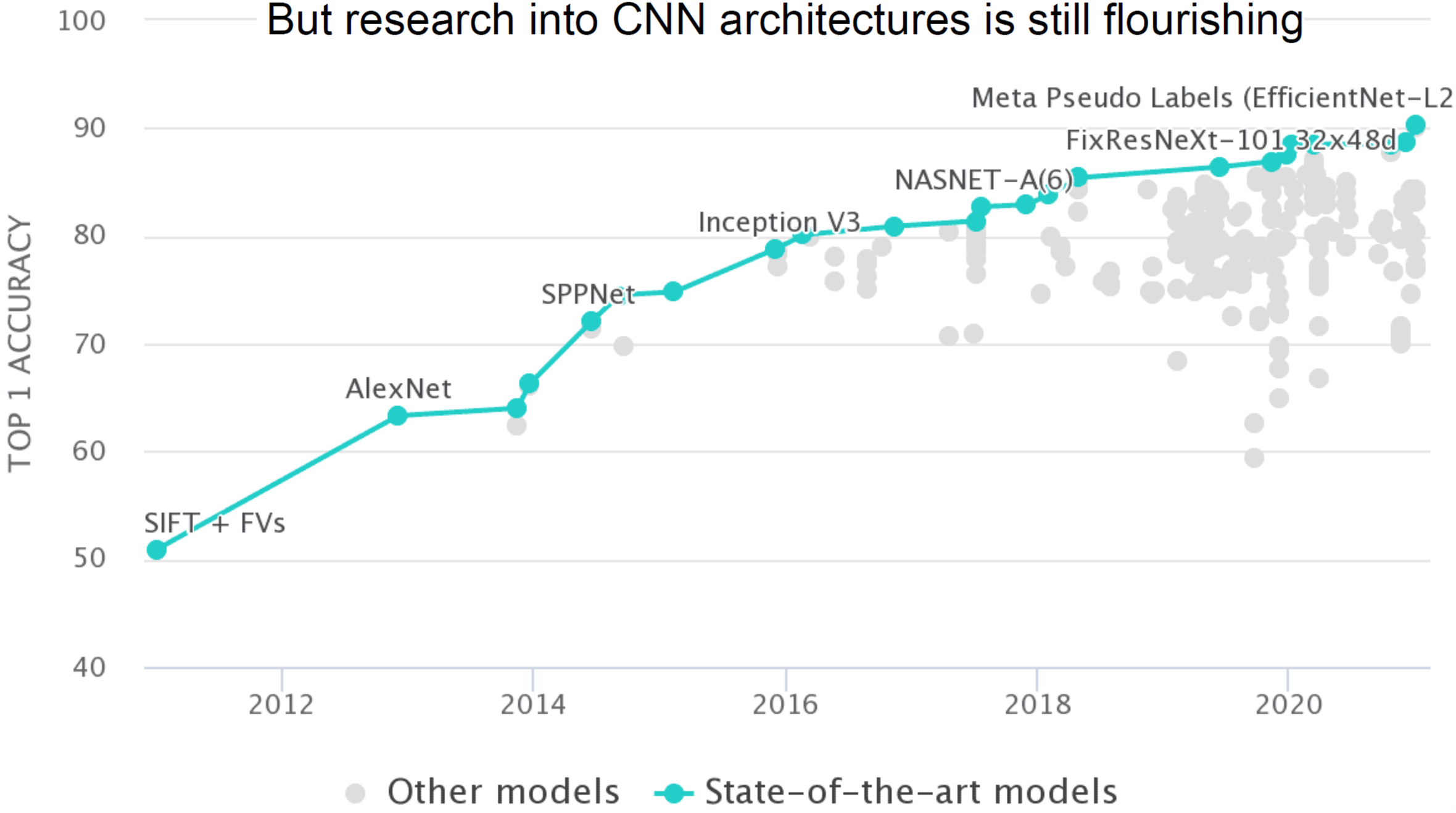


Large Scale Visual Recognition Challenge winners



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

But research into CNN architectures is still flourishing



DenseNet: Densely Connected Convolutional Networks (2017)

- Skip connections are a pretty cool idea. Why don't we just skip-connect everything?

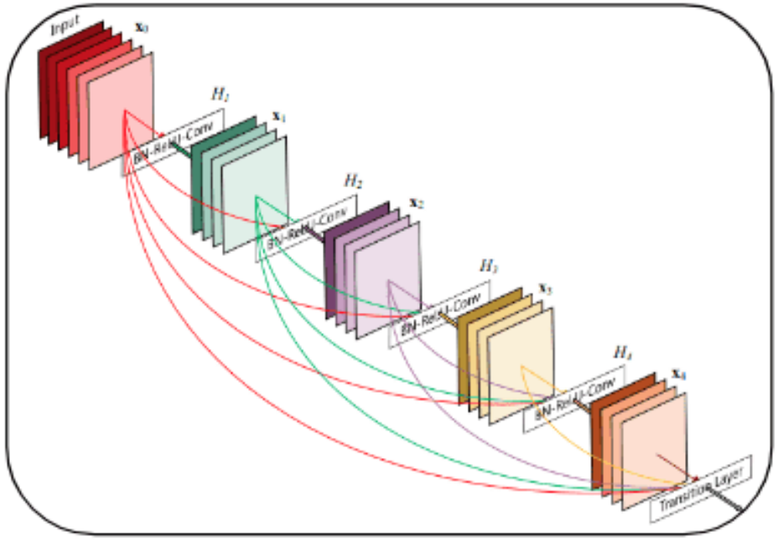
- Densenet is an example of pushing this idea into the extremity.
  - Of course, the main difference with ResNets is that we will concatenate instead of adding the feature maps.

- There are two concerns here:
  - The feature maps have to be of the same size.
  - The concatenation with all the previous feature maps may result in memory explosion.

- To address the first issue we have two solutions:
  - a) use conv layers with appropriate padding that maintain the spatial dims or
  - b) use dense skip connectivity only inside blocks called Dense Blocks.

DenseNet: Densely Connected Convolutional Networks (2017)

Dense Block 1

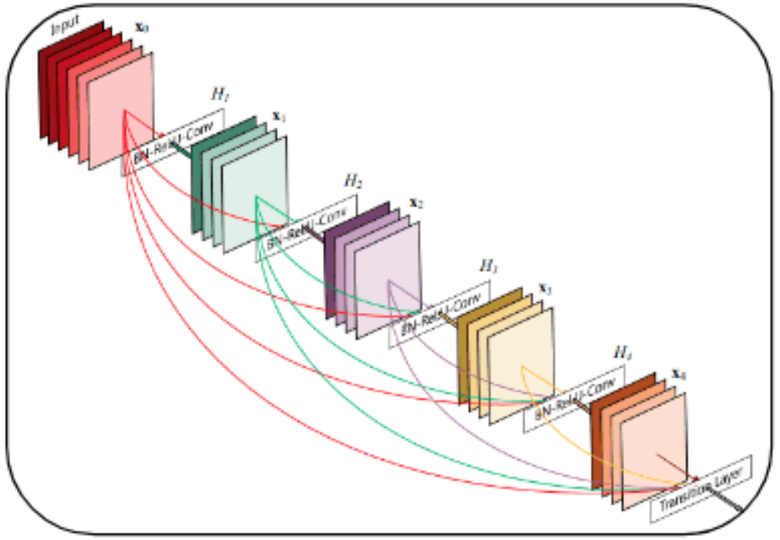


Spatial dims: 256x256
512 feature maps

Transition Layer

1×1×K convs
+
2×2 average pooling

Dense Block 2



Spatial dims: 128x128
K feature maps < 512

- The transition layer can down-sample the image dimensions with average pooling.
- To address the second concern which is memory explosion, the feature maps are reduced (kind of compressed) with 1x1 convs. Notice that we used K in the diagram, but densenet uses K= featmaps/2$K=featmaps/2$
- Furthermore, they add a dropout layer with p=0.2 after each convolutional layer when no data augmentation is used.

Neural Architecture Search (NAS)

- Although most popular and successful model architectures are designed by human experts, it doesn't mean we have explored the entire network architecture space and settled down with the best option.
- We would have a better chance to find the optimal solution if we adopt a systematic and automatic way of learning high-performance model architectures.

- Neural Architecture Search (NAS) automates network architecture engineering. It aims to learn a network topology that can achieve best performance on a certain task.

Neural Architecture Search (NAS)

- NAS as a system with three major components, which is clean & concise, and also commonly adopted in other NAS papers.
- Search space:
  - The NAS search space defines a set of operations (e.g. convolution, fully-connected, pooling) and how operations can be connected to form valid network architectures.
    - The design of search space usually involves human expertise, as well as unavoidably human biases.
- Search algorithm:
  - A NAS search algorithm samples a population of network architecture candidates.
  - It receives the child model performance metrics as rewards (e.g. high accuracy, low latency) and optimizes to generate high-performance architecture candidates.
- Evaluation strategy:
  - We need to measure, estimate, or predict the performance of a large number of proposed child models in order to obtain feedback for the search algorithm to learn.

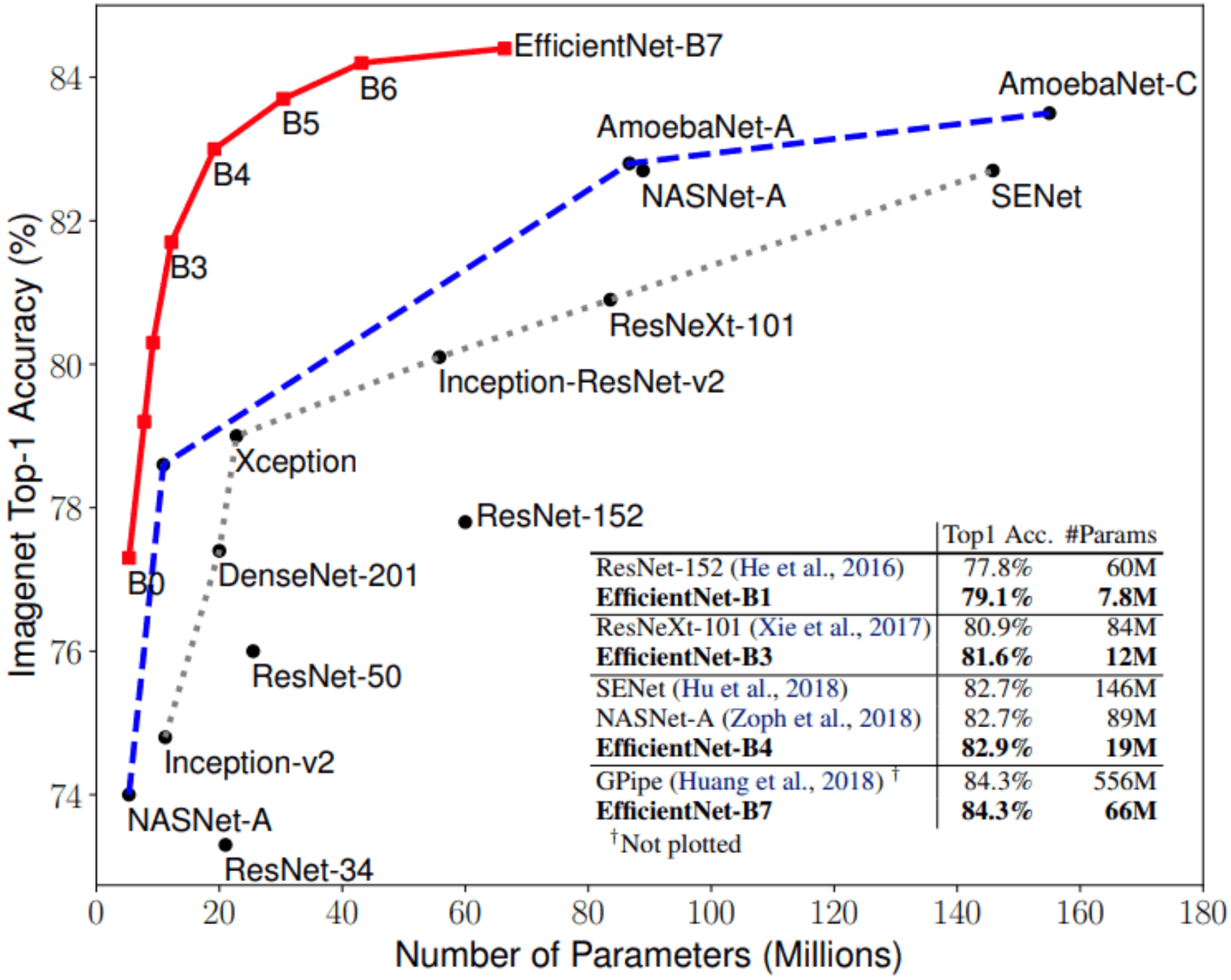
EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (2019)

EfficientNet is all about engineering and scale.

It proves that if you carefully design your architecture you can achieve top results with reasonable parameters.

It's incredible that EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152.

The graph demonstrates the ImageNet Accuracy VS model parameters.

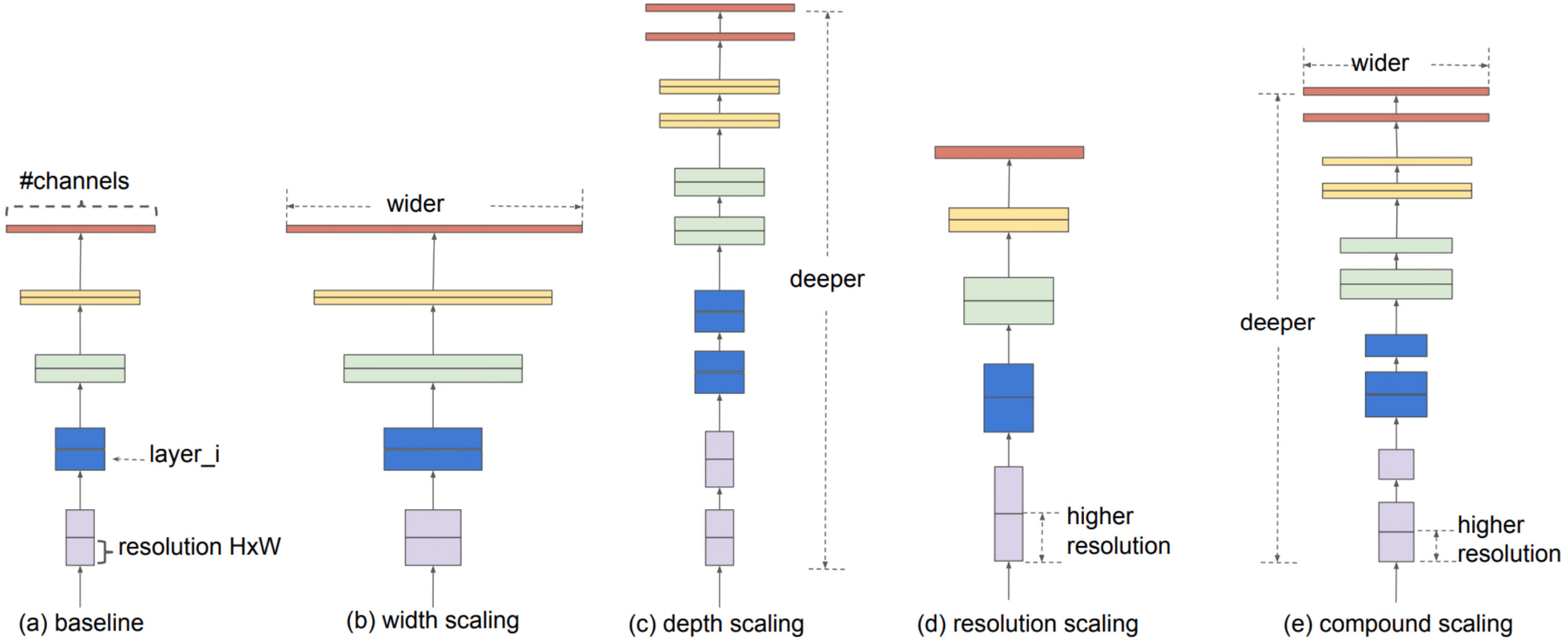


	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

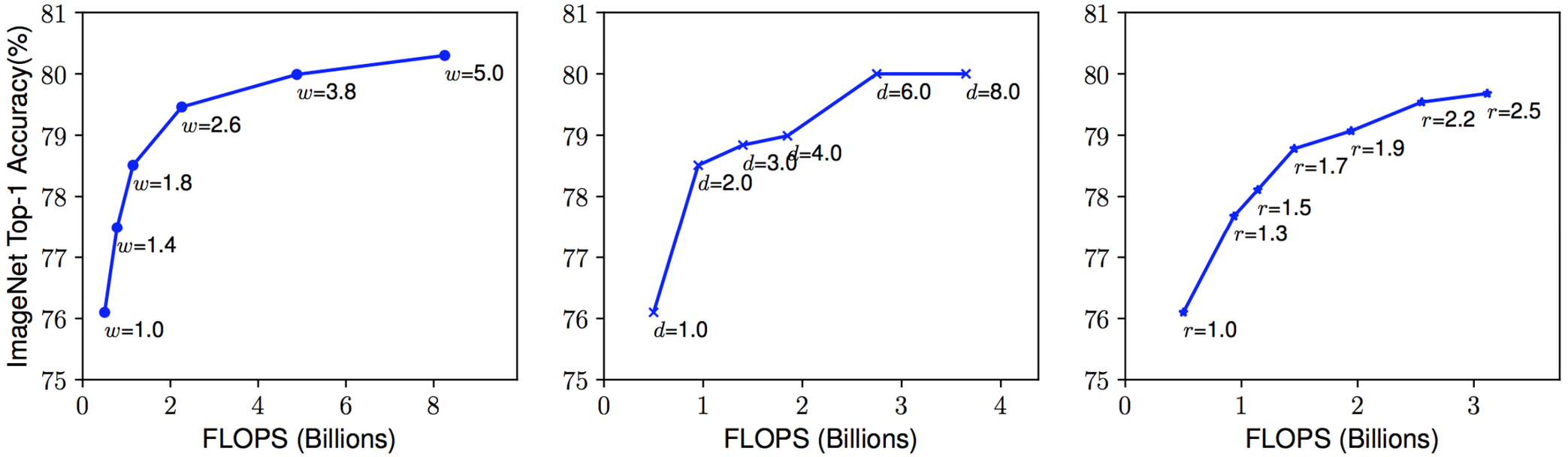
EfficientNet

- There are three scaling dimensions of a CNN:
  - **Depth** simply means how deep the networks is which is equivalent to the number of layers in it.
  - **Width** simply means how wide the network is. One measure of width, for example, is the number of channels in a Conv layer.
  - **Resolution** is simply the image resolution that is being passed to a CNN.



EfficientNet

- The above three points lead to our **first observation**:
  - Scaling up any dimension of network (width, depth or resolution) improves accuracy, but the accuracy gain diminishes for bigger models
  - Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of



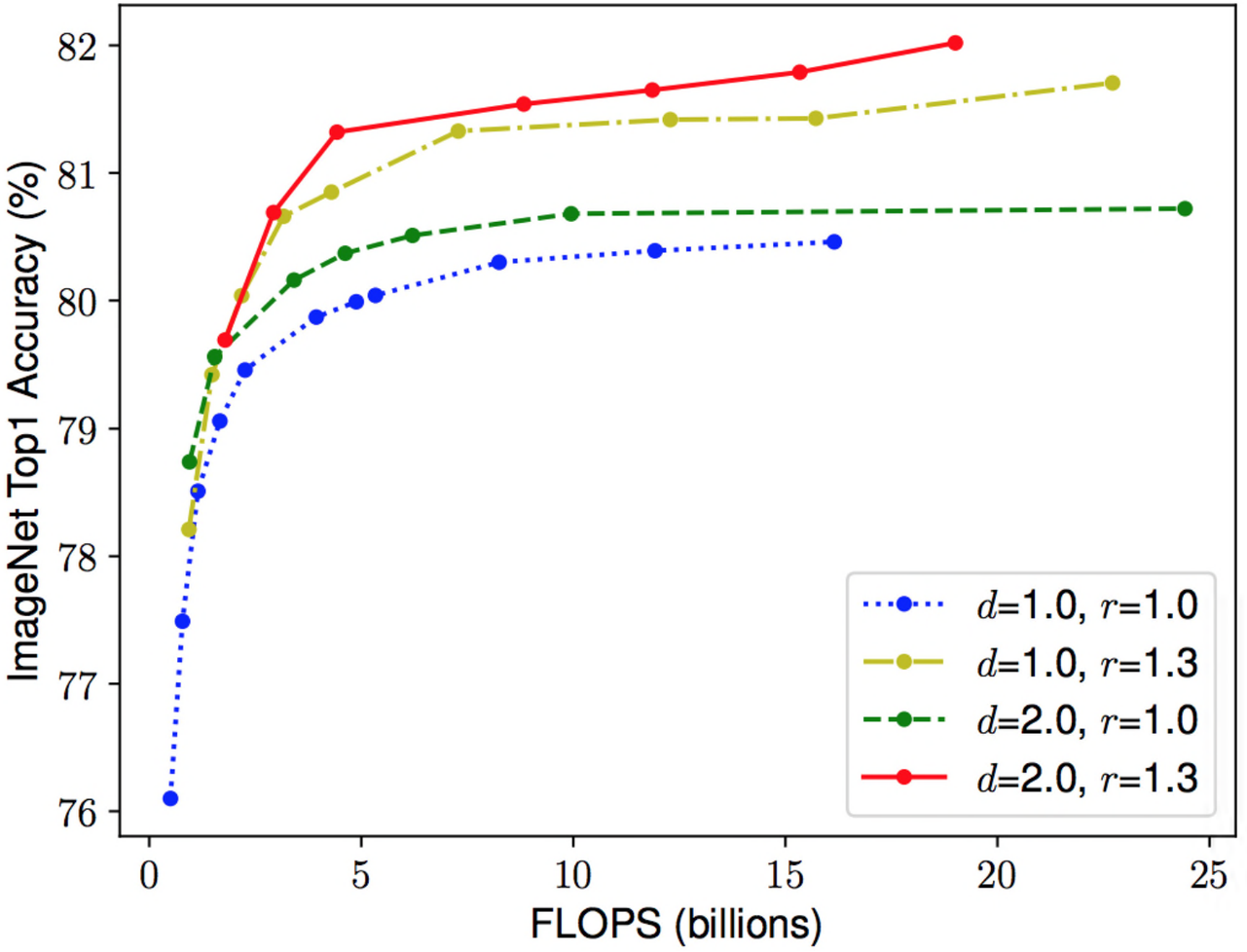
Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.

EfficientNet

- Intuition says that
  - As the resolution of the images is increased, depth and width of the network should be increased as well.
  - As the depth is increased, larger receptive fields can capture similar features that include more pixels in an image.
  - As the width is increased, more fine-grained features will be captured.

EfficientNet

- To validate this intuition,
  - The authors ran a number of experiments with different scaling values for each dimension.
  - For example, as shown in the figure below from the paper, with deeper and higher resolution, width scaling achieves much better accuracy under the same FLOPS cost.



These results lead to our ***second*** observation:

- *It is critical to balance all dimensions of a network (width, depth, and resolution) during CNNs scaling for getting improved accuracy and efficiency.*

EfficientNet

- The authors proposed a simple yet very effective scaling technique which uses a **compound coefficient ϕ** to uniformly scale network width, depth, and resolution in a principled way:

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

The different values of φ produce EfficientNets B1-B7.

EfficientNet

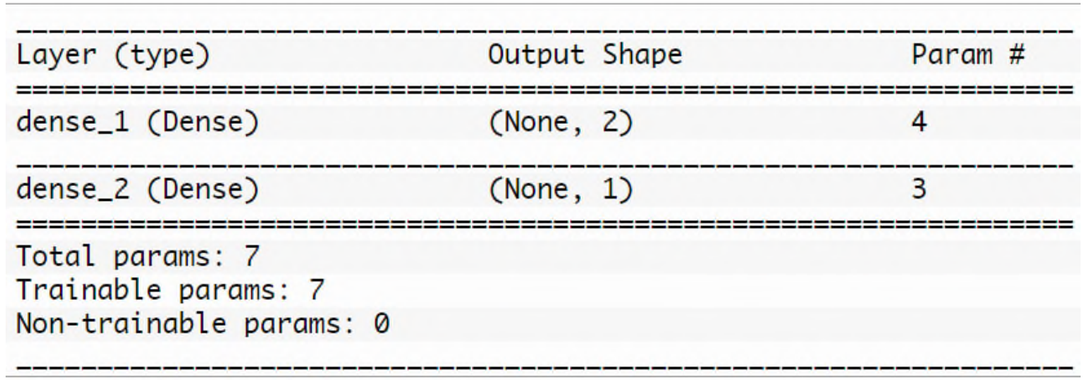
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

The MBConv block is nothing fancy but an Inverted Residual Block (used in MobileNetV2) with a Squeeze and Excite block injected sometimes.

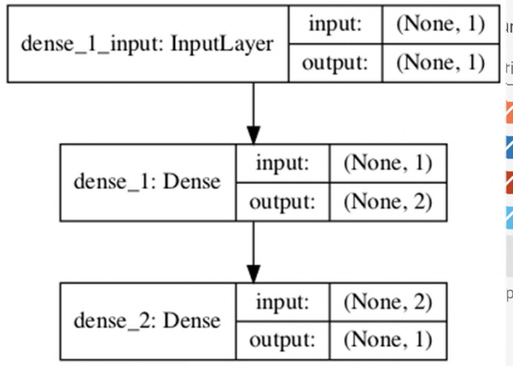
1.EfficientNet paper: https://arxiv.org/abs/1905.11946
2.Official released code: https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet

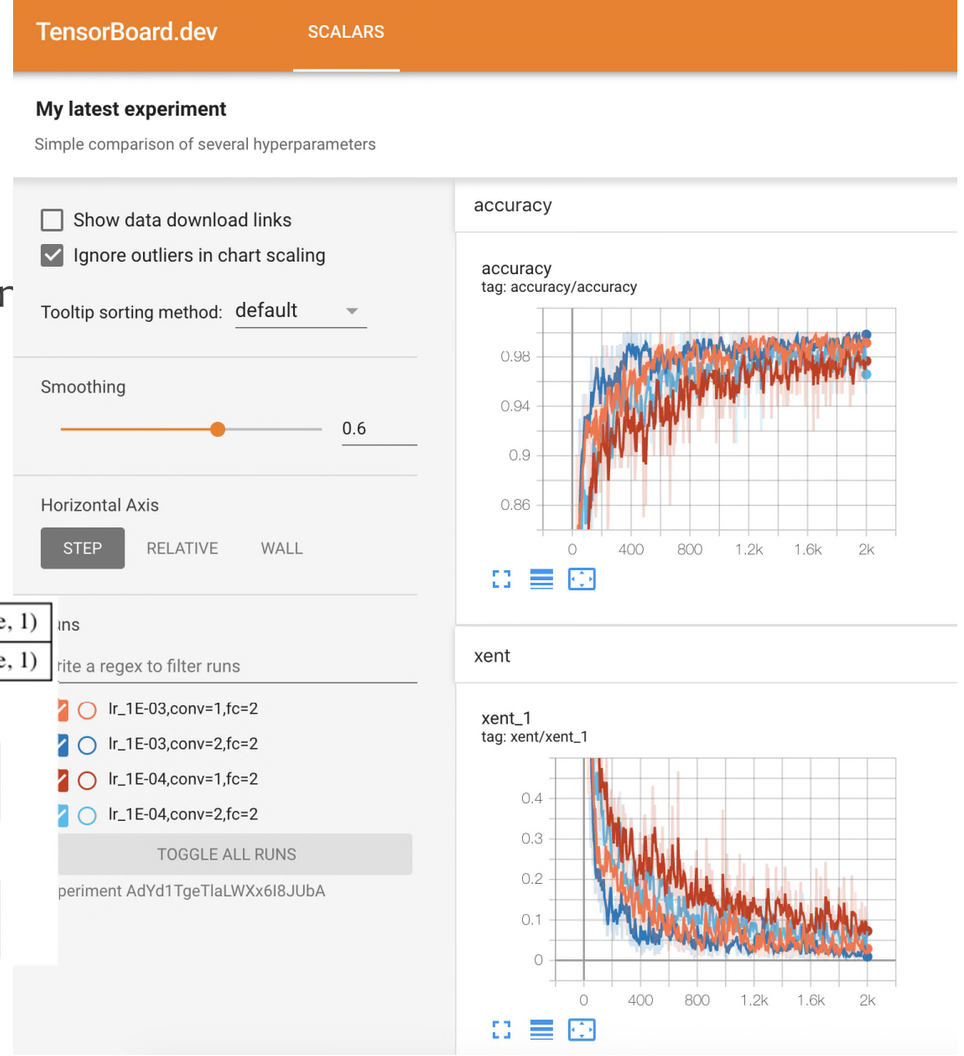
What next

- Demonstrating the capabilities of tensor board.
  - Visualizing graphs, network architectures (of different CNNs) and other stuff ir
- Graphviz graph library
  - Load ResNet, VGG, Inception and visualize using this library.

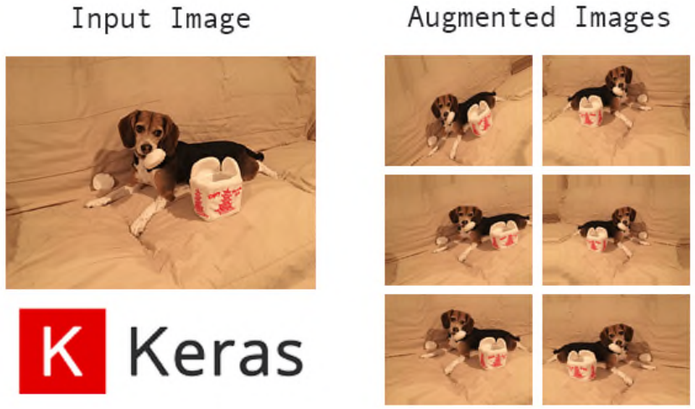


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 2)                 4
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 3
=================================================================
Total params: 7
Trainable params: 7
Non-trainable params: 0
_________________________________________________________________
```





- Using Keras Data Generators



Reading Material

- Receptive Filed
  - A guide to receptive field arithmetic for Convolutional Neural Networks

- ABOUT GoogleNet (INCEPTION)
  - **A Simple Guide to the Versions of the Inception Network**
    - https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
  - **Inception (Video)**
    - https://www.youtube.com/watch?v=VxhSouuSZDY
  - **Inception Network Building Block**
    - https://www.youtube.com/watch?v=C86ZXvgpejM&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&index=17
  - **Combining Building Blocks to build the Inception**
    - https://www.youtube.com/watch?v=KfV8CJh7hE0&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&index=18

Reading Material

- **ResNet**
  - **https://arxiv.org/abs/1512.03385**
  - An Overview of ResNet and its Variants
- **CNN Architectures: AlexNet, VGG, GoogLeNet, ResNet and more ….**
  - https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5
- **ResNeXt :**
  - https://towardsdatascience.com/review-resnext-1st-runner-up-of-ilsvrc-2016-image-classification-15d7f17b42ac
- **Squeeze-and-Excitation Networks, (paper)**
  - https://arxiv.org/abs/1709.01507
- **Squeeze-and-Excitation Networks, brief intro**
  - https://towardsdatascience.com/squeeze-and-excitation-networks-9ef5e71eacd7
  - https://medium.com/@konpat/squeeze-and-excitation-networks-hu-et-al-2017-48e691d3fe5e
- **An Introduction to different Types of Convolutions in Deep Learning**
- https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d25

Reading Material

- **Illustrated: 10 CNN Architectures (**A compiled visualisation of the common convolutional neural networks)
  - https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

- **Best deep CNN architectures and their principles: from AlexNet to EfficientNet by Nikolas Adaloglou** on2021-01-21
  - https://theaisummer.com/cnn-architectures/

- **EfficientNet:** Improving Accuracy and Efficiency through AutoML and Model Scaling
  - https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html
  - https://medium.com/@nainaakash012/efficientnet-rethinking-model-scaling-for-convolutional-neural-networks-92941c5bfb95
  - Video : https://www.youtube.com/watch?v=3svIm5UC94I
- **EfficientNet** B0 to B7
  - https://keras.io/api/applications/efficientnet/

Acknowledgements

Various contents in this presentation have been taken from different books, lecture notes and the web.
These solely belong to their owners, and are here used only for clarifying various educational concepts.
Any copyright infringement is *not* intended.