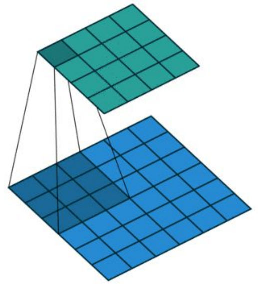
Deep Learning for ECE EECE-580G

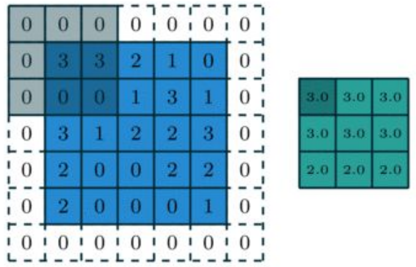
Advanced CNNs

Last time

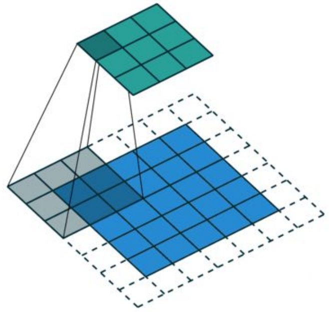
CNNs ingredients



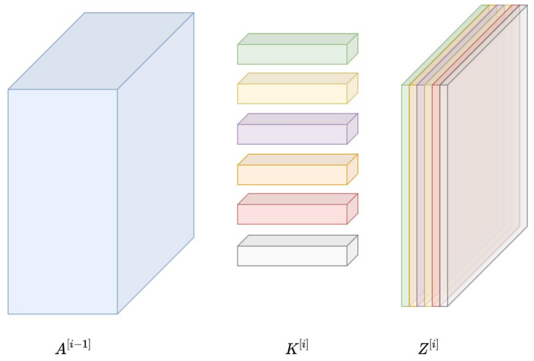
conv2d



pool with stride



conv2d with stride



conv2d on volumes

Batch Norm update (one layer)

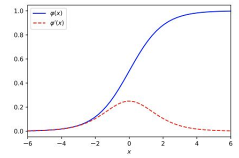
B minibatch of size m, $B = \{x^{(1)}, \dots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^d$

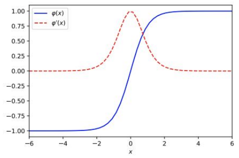
$$\mu^{(B)} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \sigma^{(B)2} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu^{(B)})^2$$

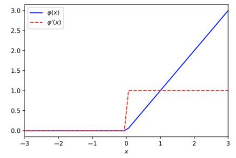
normalize $$\hat{x}_k^{(i)} = \frac{x_k^{(i)} - \mu_k^{(B)}}{\sqrt{\sigma_k^{(B)2} + \epsilon}}$$

output $$y_k^{(i)} = \gamma_k \hat{x}_k^{(i)} + \beta_k$$ γ, β trainable variables

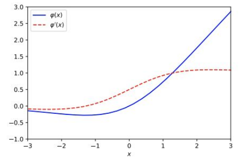
Batch Norm

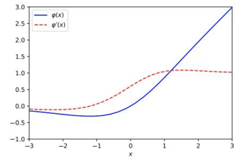






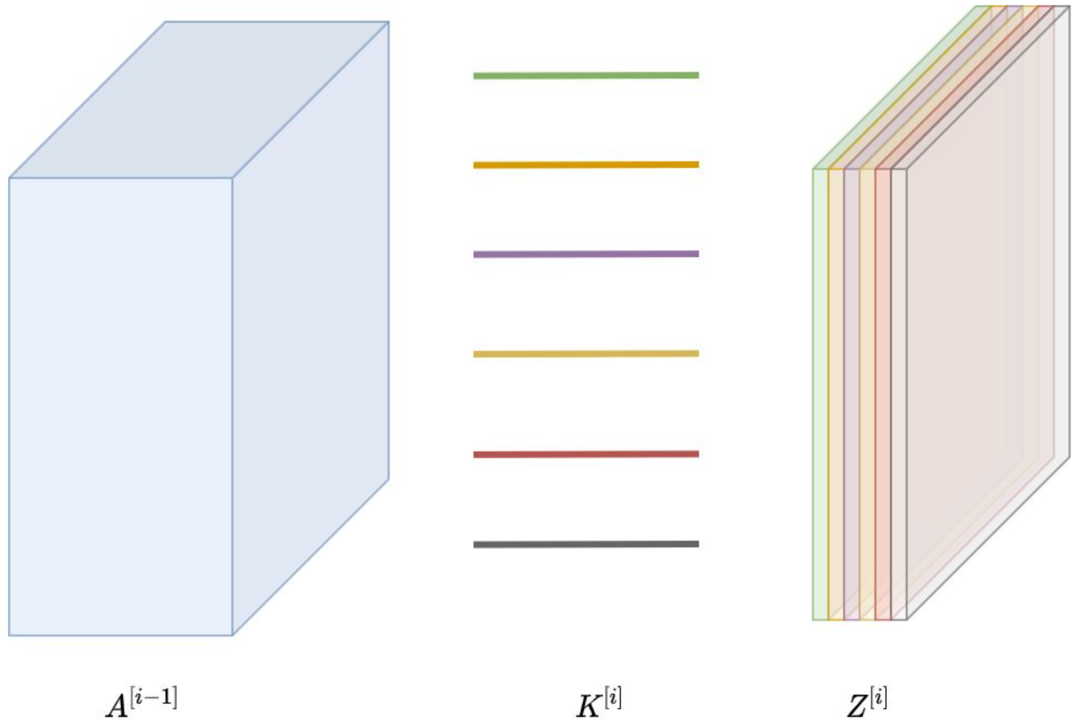
Activation functions





1x1 convolution

- Does not mix spatial information
- Only mixes channels
- Each 1x1 conv return a weighted avg of the input channels
- Useful when need to decrease/increase the depth independently of the spatial info
- Can you see that a 1x1 conv on a 1x1xD image is equivalent to flatten ↠ FC? >>> **Piazza**



The “conv” layer

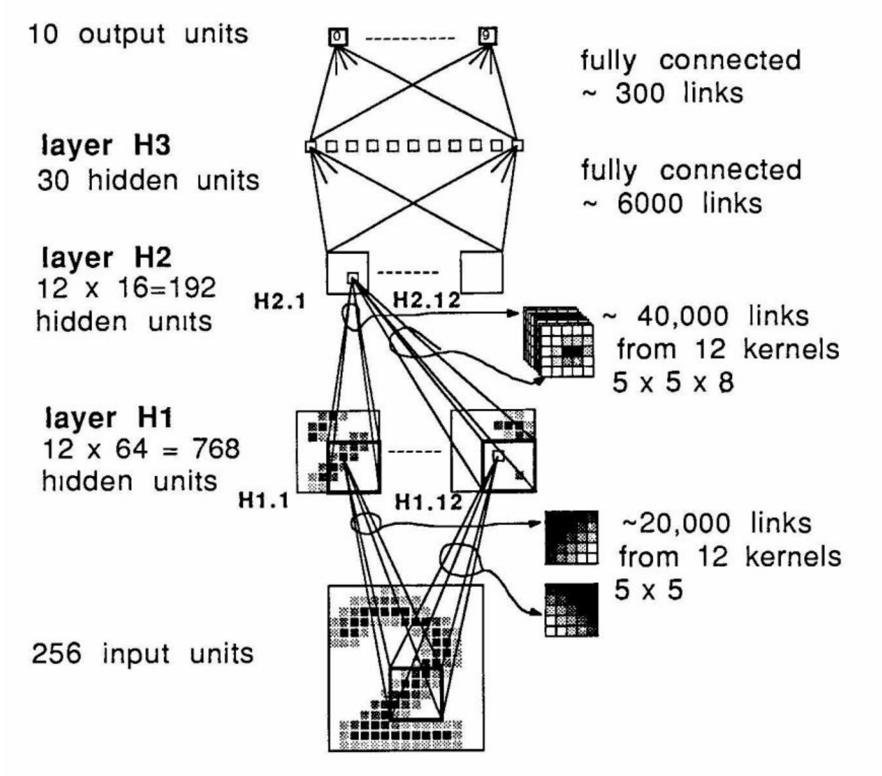
- Usually = **conv - BN - activation**
- When followed by BN no need to use the bias in conv layer

Early CNNs

First CNNs (1989)

- Yann LeCun - Bell Labs
- Trained for 3 weeks
- Activation = tanh
- Strided convolutions at each layer

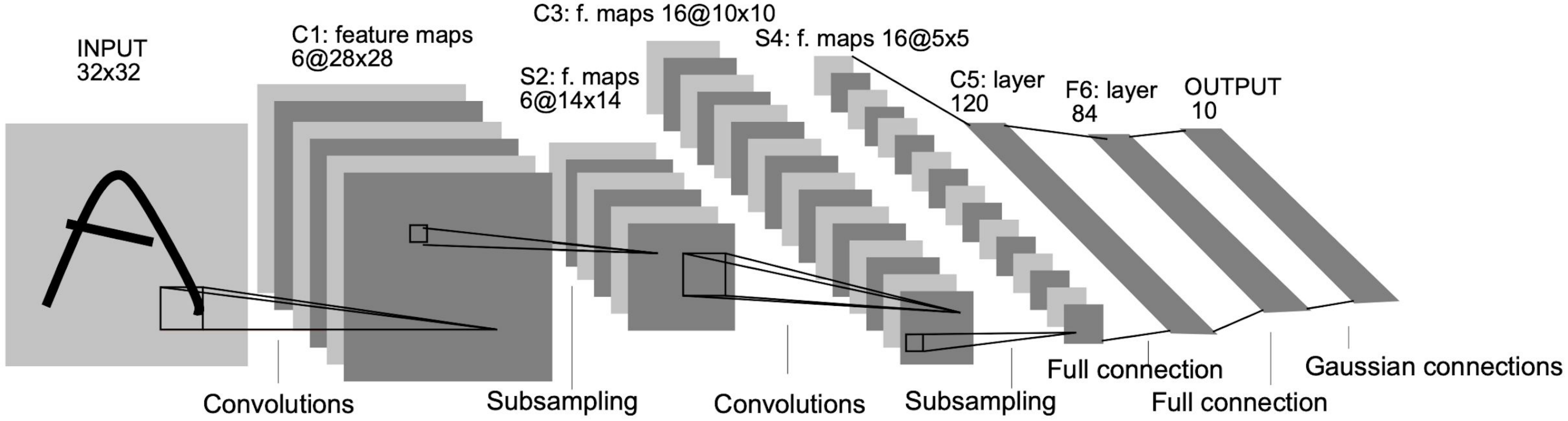
Source: NYU.edu



10 output units
fully connected ~ 300 links
layer H3 30 hidden units
fully connected ~ 6000 links
layer H2 12 x 16=192 hidden units
H2.1
H2.12
~ 40,000 links from 12 kernels 5 x 5 x 8
layer H1 12 x 64 = 768 hidden units
H1.1
H1.12
~20,000 links from 12 kernels 5 x 5
256 input units

LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." Neural computation 1.4 (1989): 541-551.

LeNet-5 (1998)



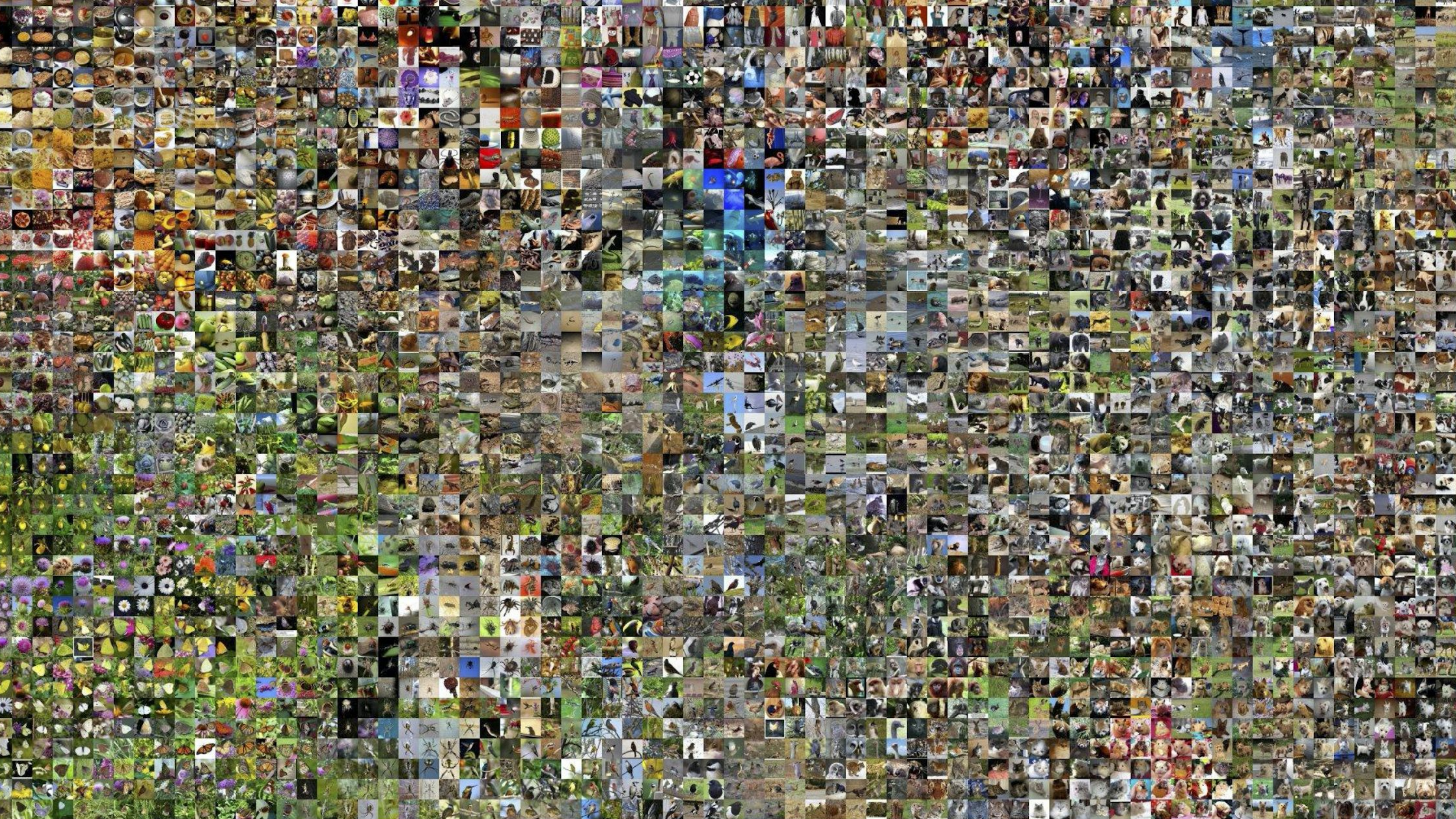
[LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.]

Missing Ingredient: Hardware...

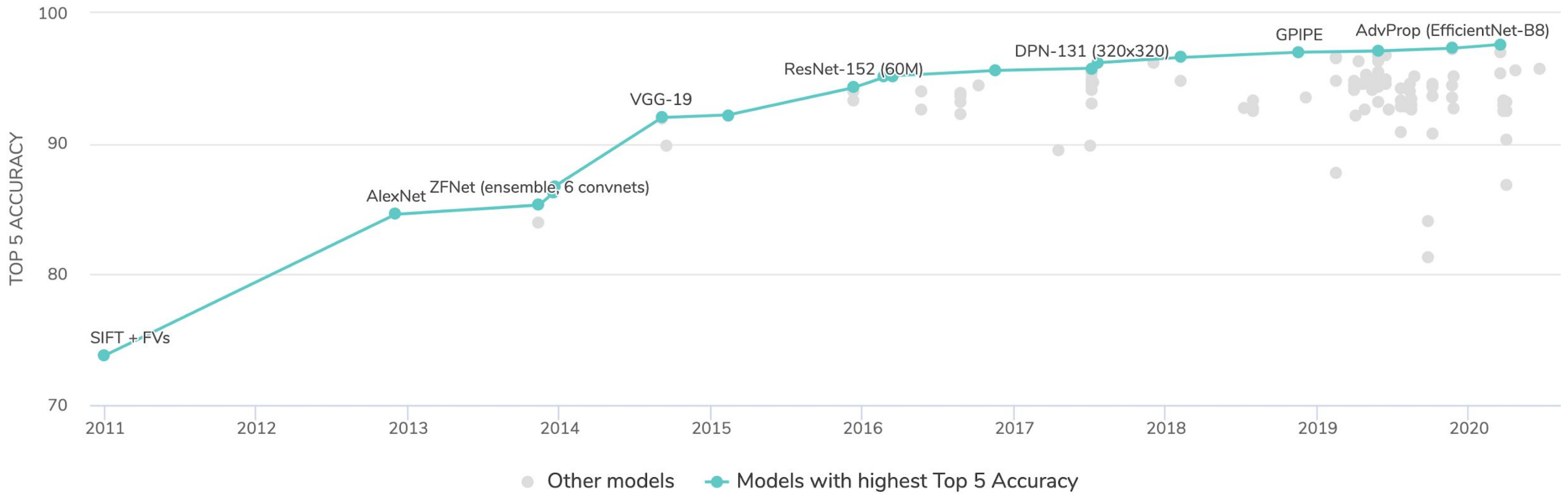
- DL research slowed down due to inefficient CPU training ...
- **Early 2000's** = GPU development thanks to gaming industry

2012 GPU GeForce GTX 580

The ImageNet Challenge

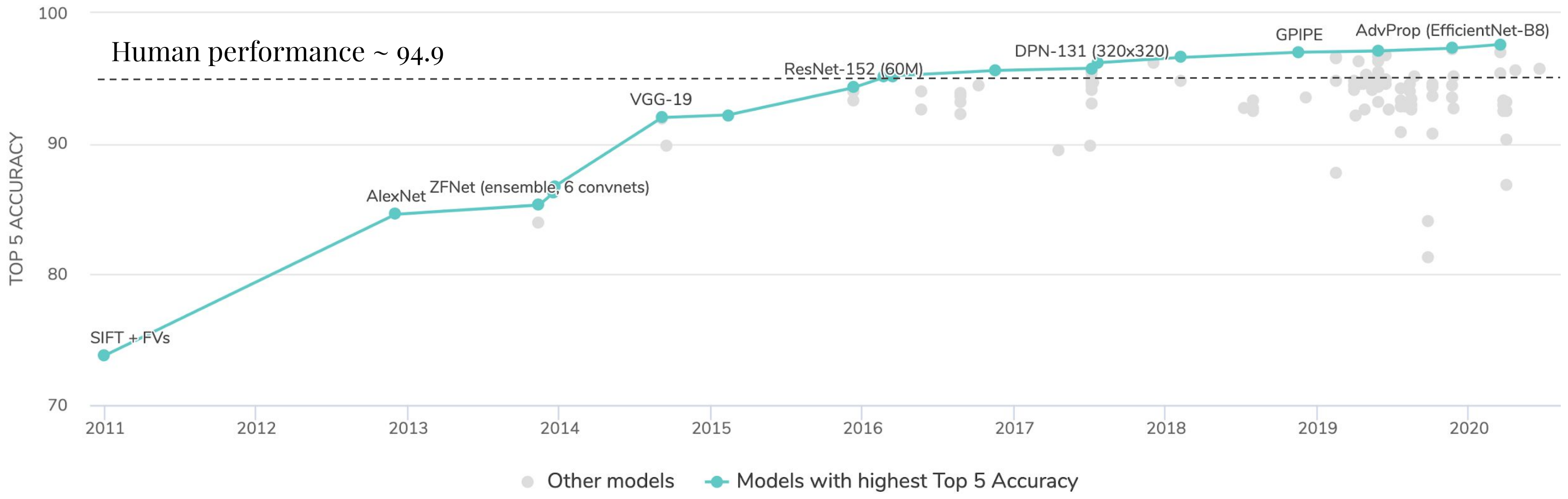


ImageNet Classification Challenge



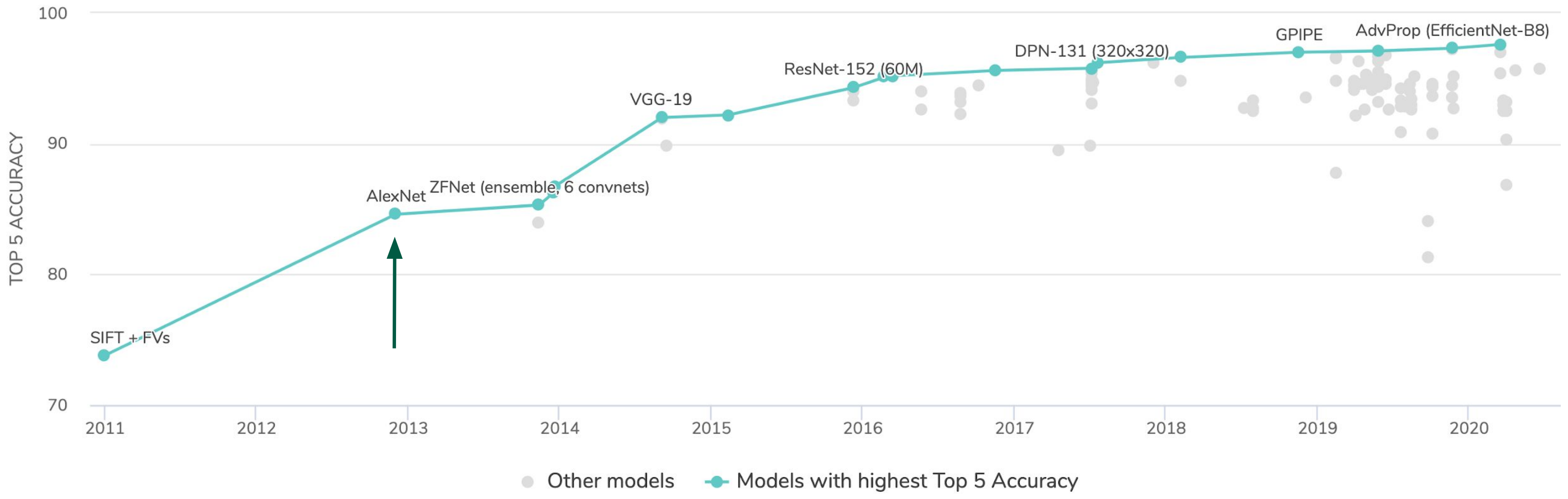
https://paperswithcode.com/sota/image-classification-on-imagenet

ImageNet Classification Challenge



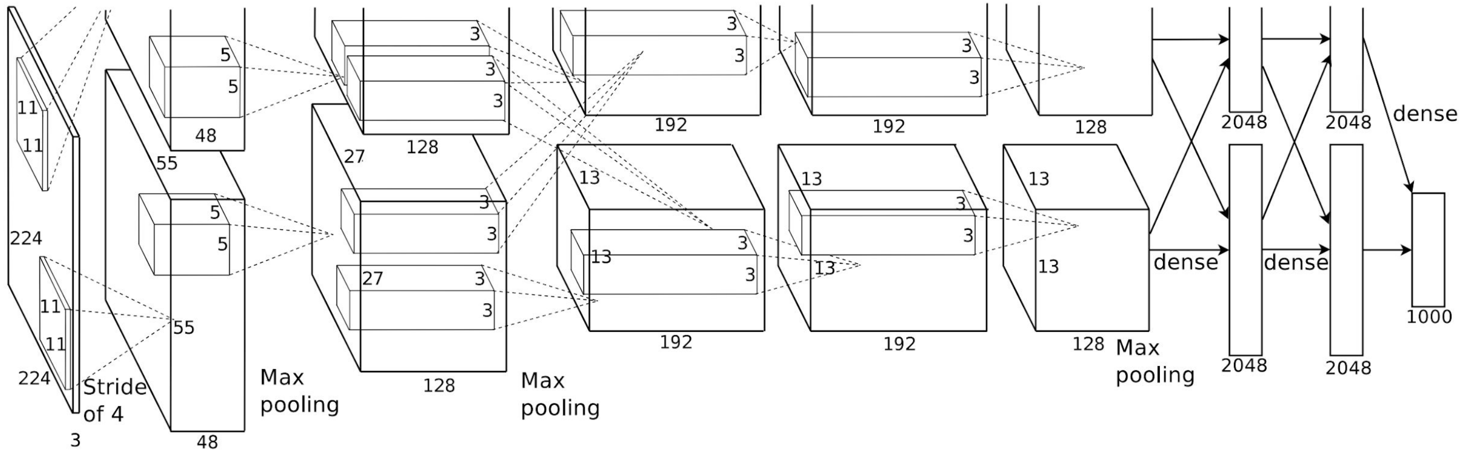
https://paperswithcode.com/sota/image-classification-on-imagenet

ImageNet Classification Challenge



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

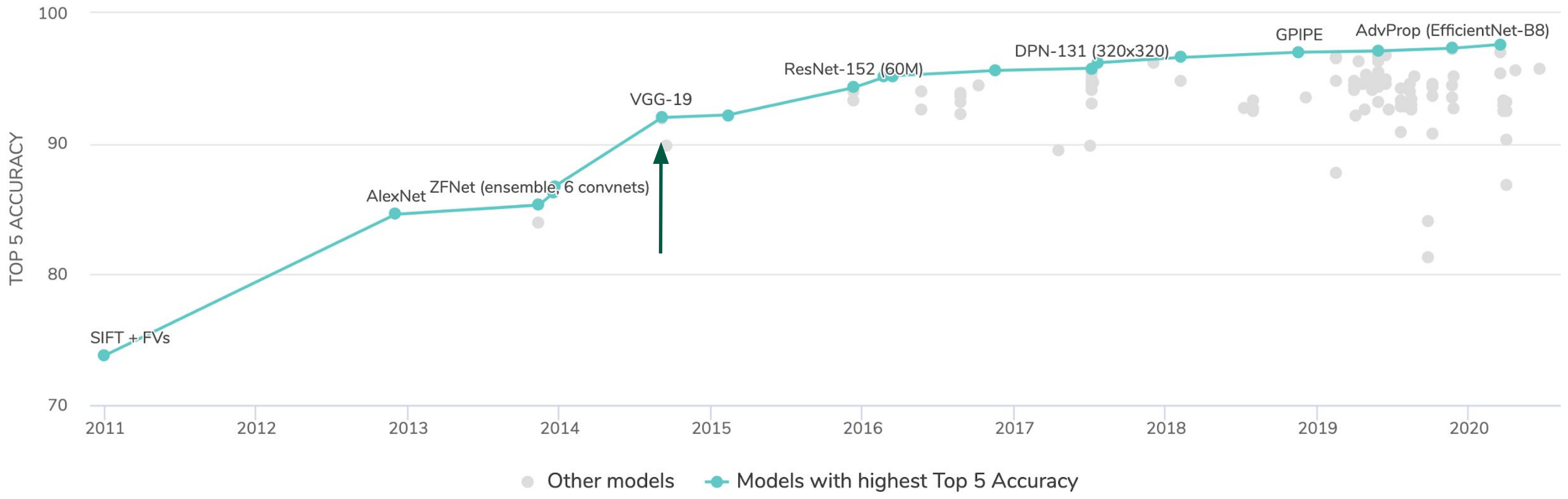
AlexNet



[Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012]

- Local Response Normalization (Similar to Batch Norm in principle, Batch Norm was introduced in 2015)
- Cited 71,641 times!
- ReLU activation

ImageNet Classification Challenge



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

VGG

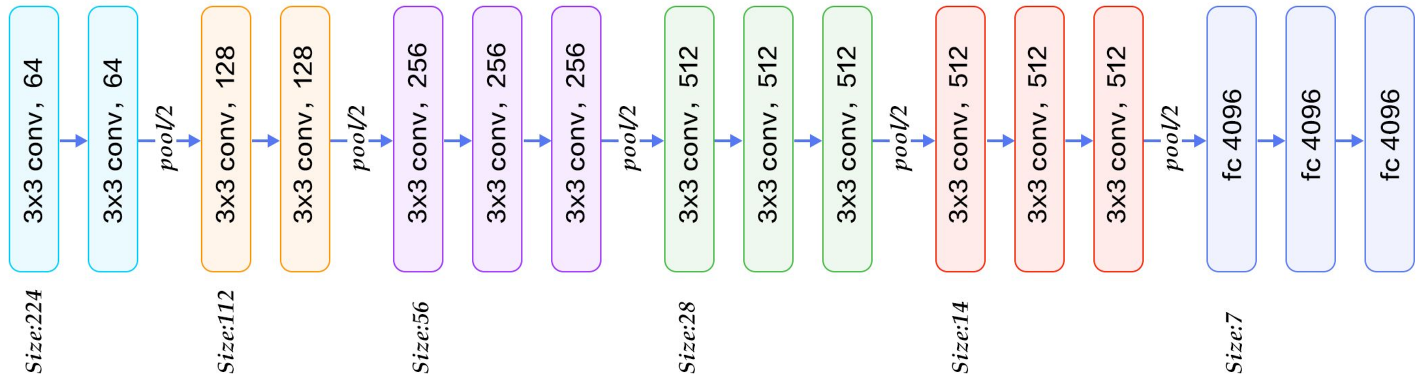


Figure Credits: Yugandhar Nanda's answer on Quora: What is the VGG neural network?
VGG paper: Simonyan, K. and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." CoRR abs/1409.1556 (2015)

- Local Response Normalization not used
- Fix kernel size to 3x3 - increase receptive field by stacking more 3x3 conv
- ReLU activation

(Not including biases for simplicity)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0

CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728

CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864

POOL2: [112x112x64] memory: 112*112*64=800K params: 0

CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728

CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456

POOL2: [56x56x128] memory: 56*56*128=400K params: 0

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824

POOL2: [28x28x256] memory: 28*28*256=200K params: 0

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296

POOL2: [14x14x512] memory: 14*14*512=100K params: 0

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

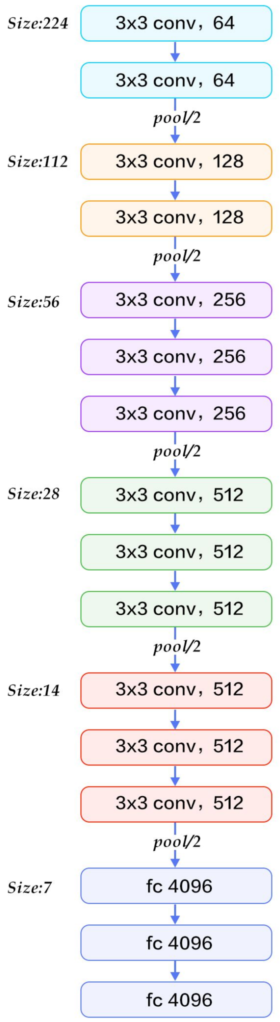
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

POOL2: [7x7x512] memory: 7*7*512=25K params: 0

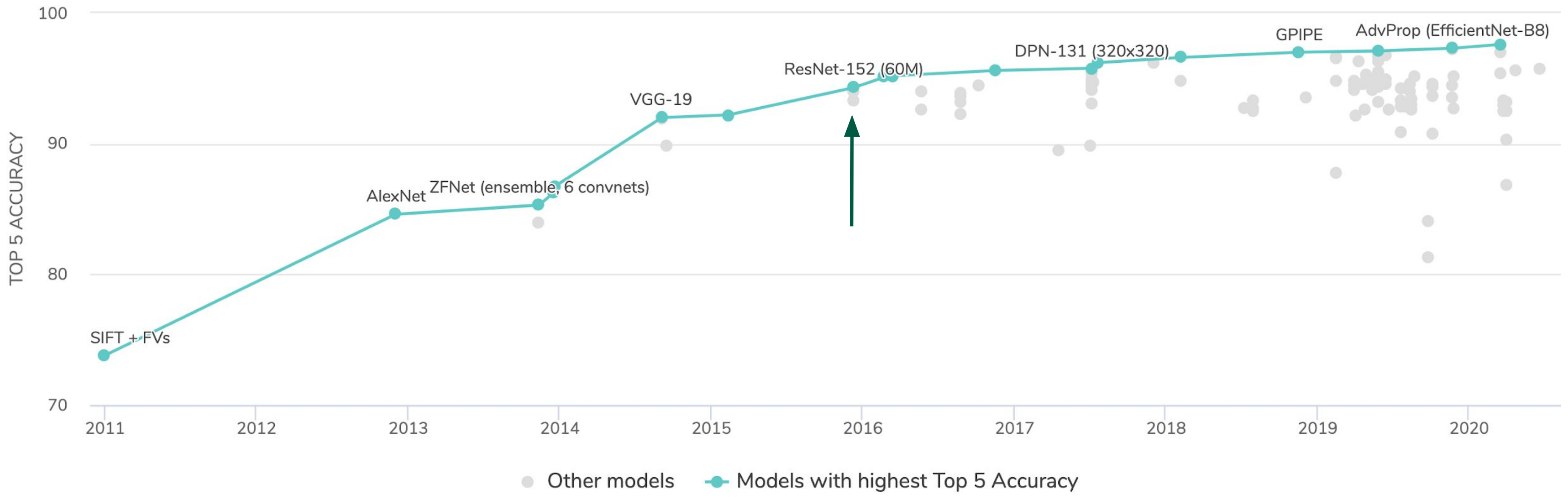
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448

FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216

FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



ImageNet Classification Challenge



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

ResNet

He, Kaiming, et al. "Deep residual learning for image recognition." CVPR 2016.

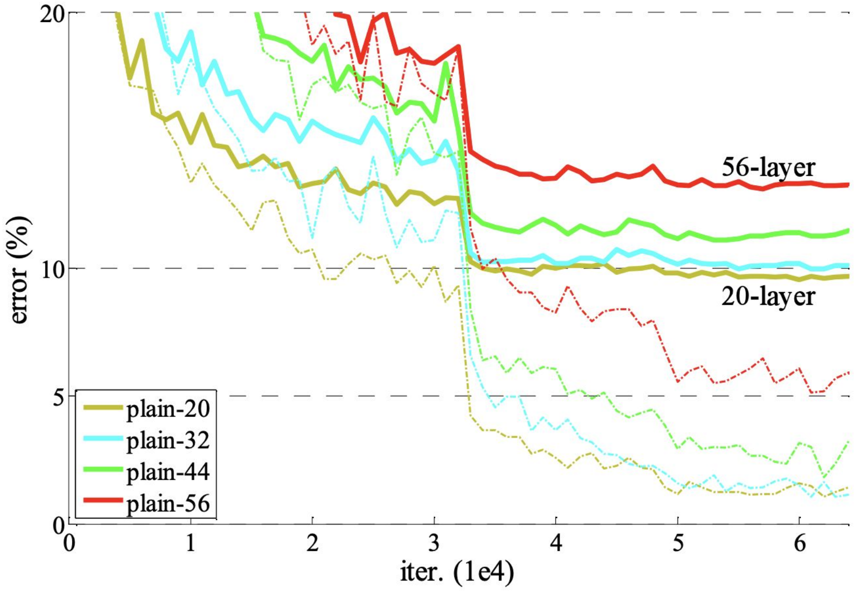
Abstract

Deeper *neural networks are* ***more difficult to train****. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can* ***gain accuracy from considerably increased depth****. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.*

Credits: The following slides use figures from the ResNet paper

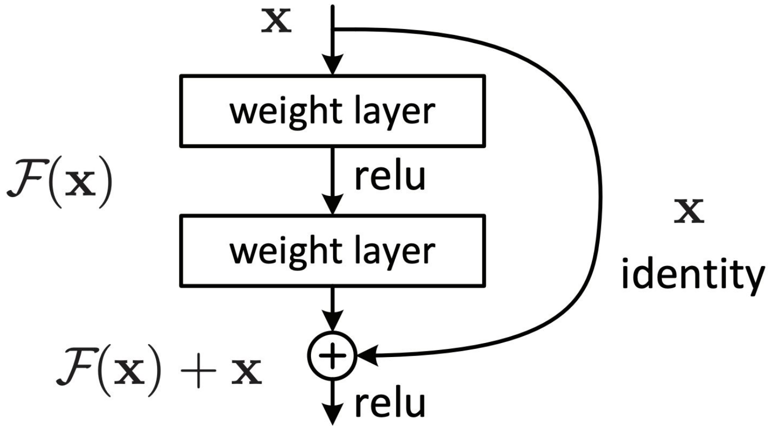
ResNet

- **Deeper** CNNs give **worse** performance?
- Hard to train deeper CNNs
- Not an overfitting problem
- In theory
  A deeper CNN should be able to represent a shallower CNN?
- Not easy in practice...



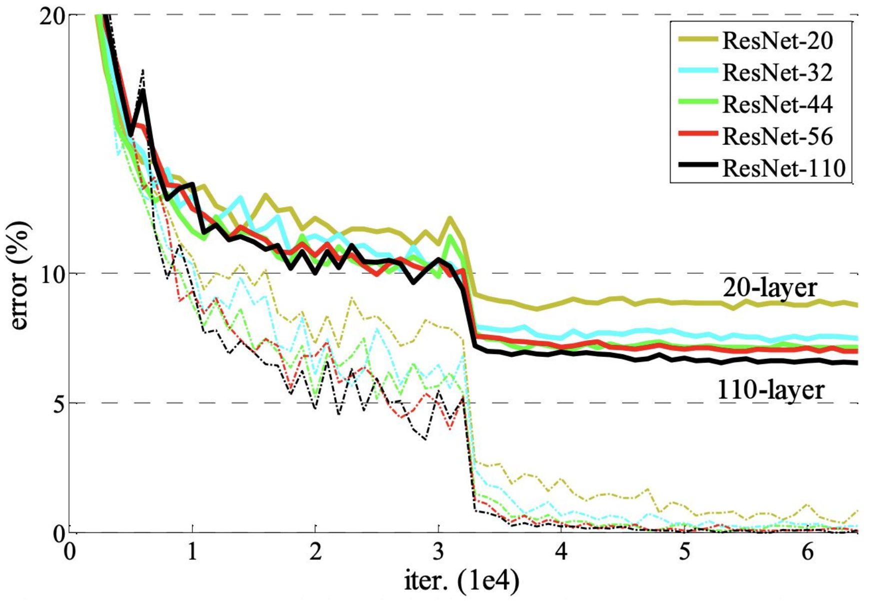
Residual blocks

- Learn the residual
- Suppose you want to learn: $\mathcal{H}(x)$
- Residual blocks learn: $\mathcal{F}(x) = \mathcal{H}(x) - x$
- Output is equivalent
- The identity connection is called **shortcut** or **skip** connection
- **Easier to learn a zero mapping than an identity mapping**



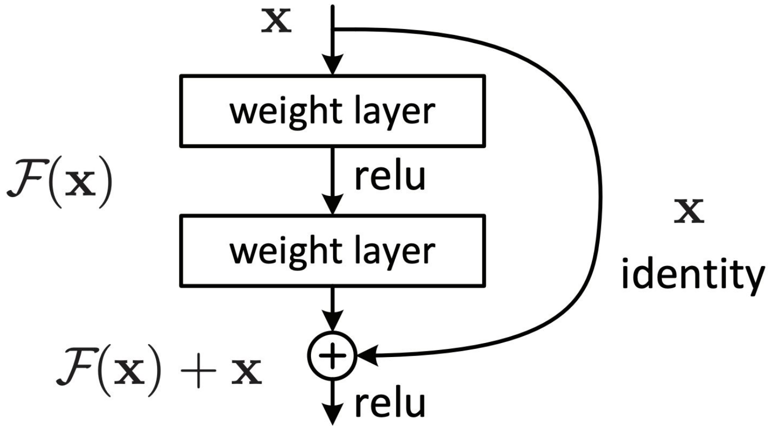
ResNet

- Easier to train residual CNNs
- **Deeper** = **Better**
- Residual blocks = the new norms in conv blocks



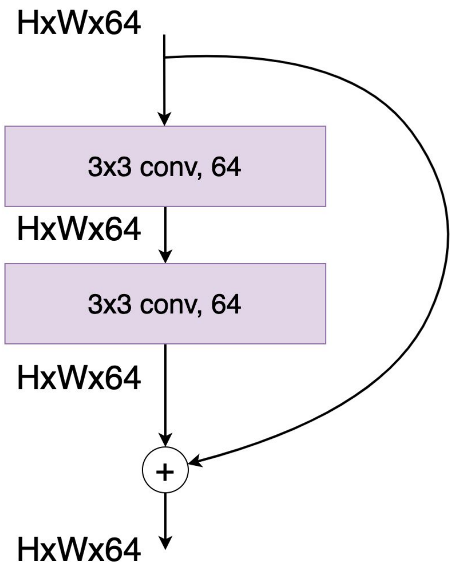
Residual blocks

- Learn the residual
- Suppose you want to learn: $\mathcal{H}(x)$
- Residual blocks learn: $\mathcal{F}(x) = \mathcal{H}(x) - x$
- Output is equivalent
- The identity connection is called **shortcut** or **skip** connection
- **Easier to learn a zero mapping than an identity mapping**
- **Can we always add the input back to the output of conv operations?**



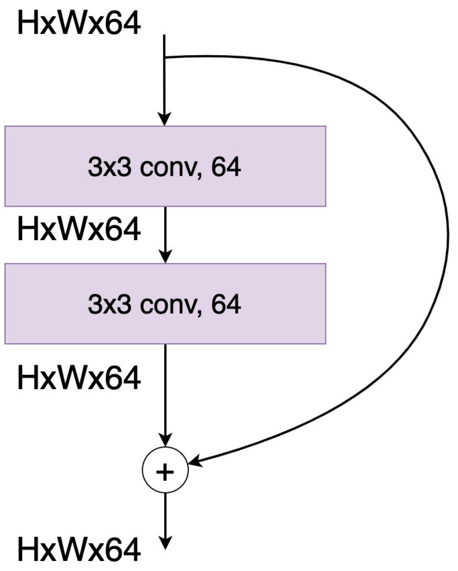
Residual blocks

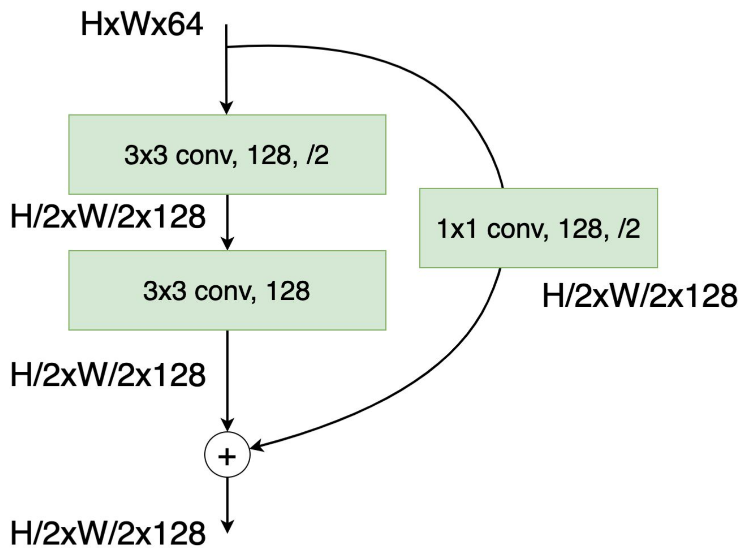
- If the output of the main branch has smaller resolution (due to stride or pooling)
- The shortcut connection has to bring x to that resolution
- Using 1x1 conv-stride



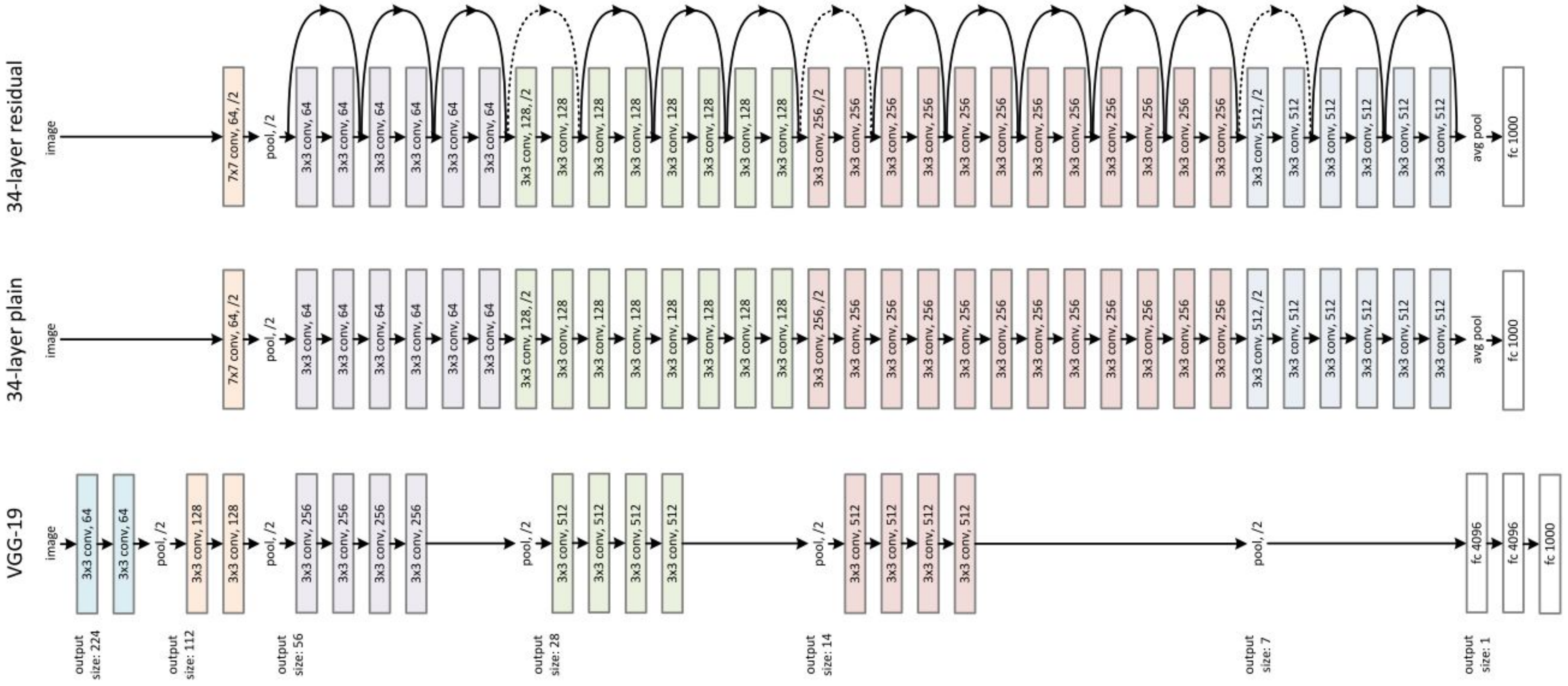
Residual blocks

- If the output of the main branch has smaller resolution (due to stride or pooling)
- The shortcut connection has to bring x to that resolution
- Using 1x1 conv-stride



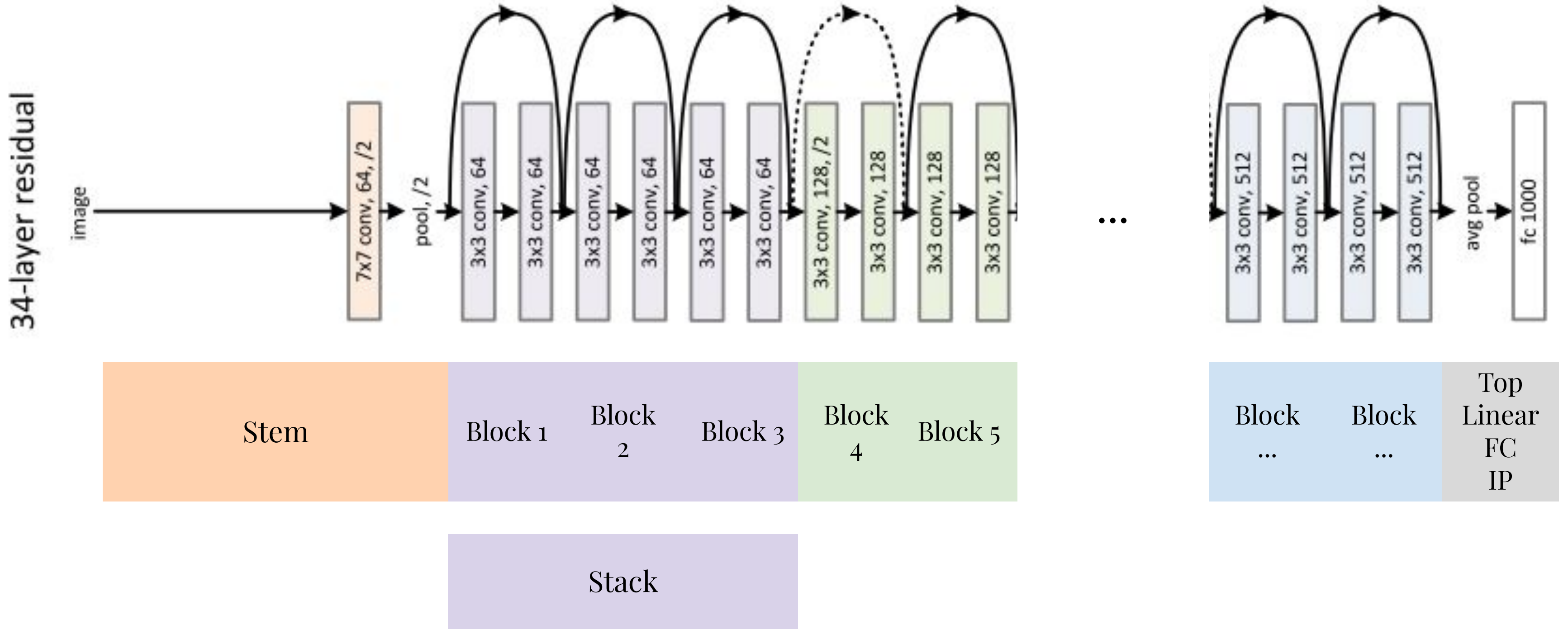


ResNet-34

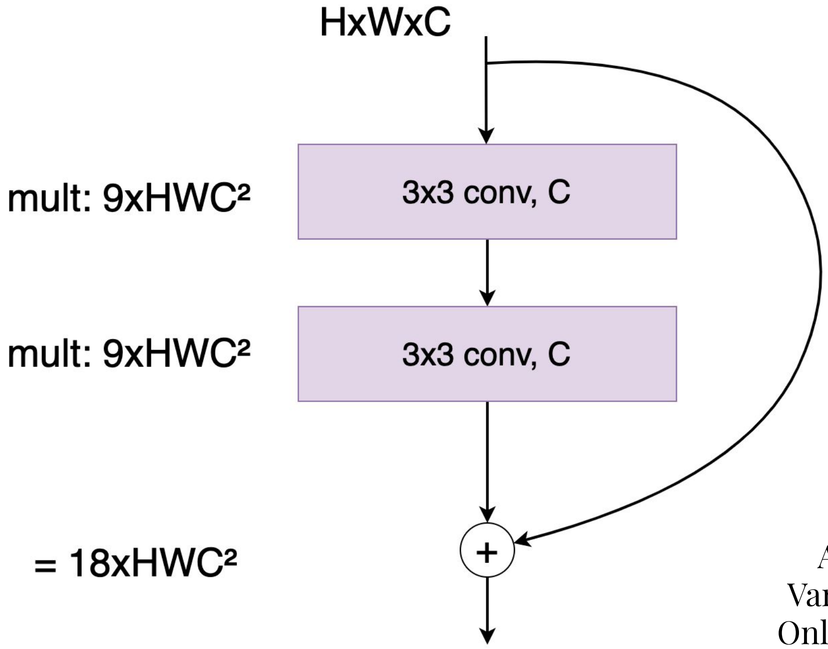


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1

Terminology

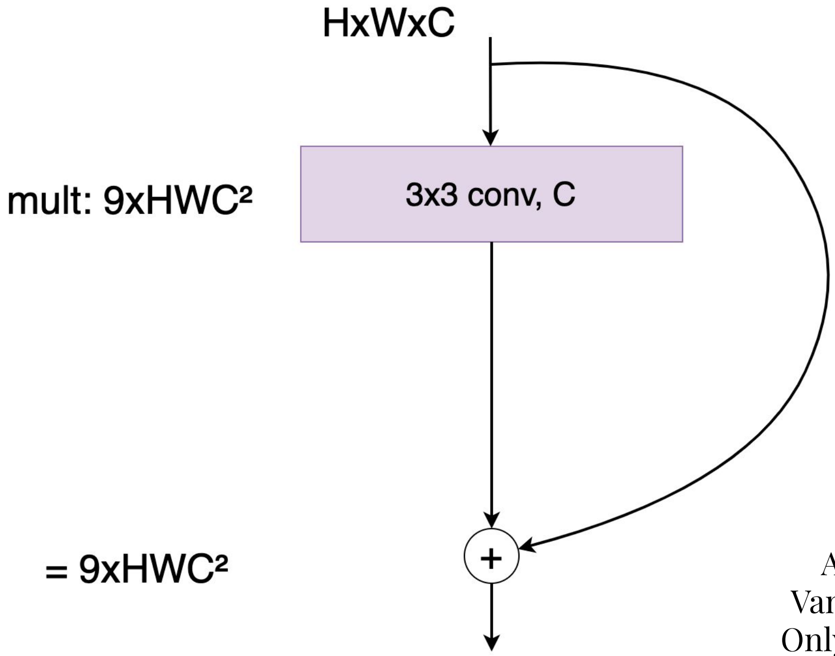


Going deeper with the bottleneck block



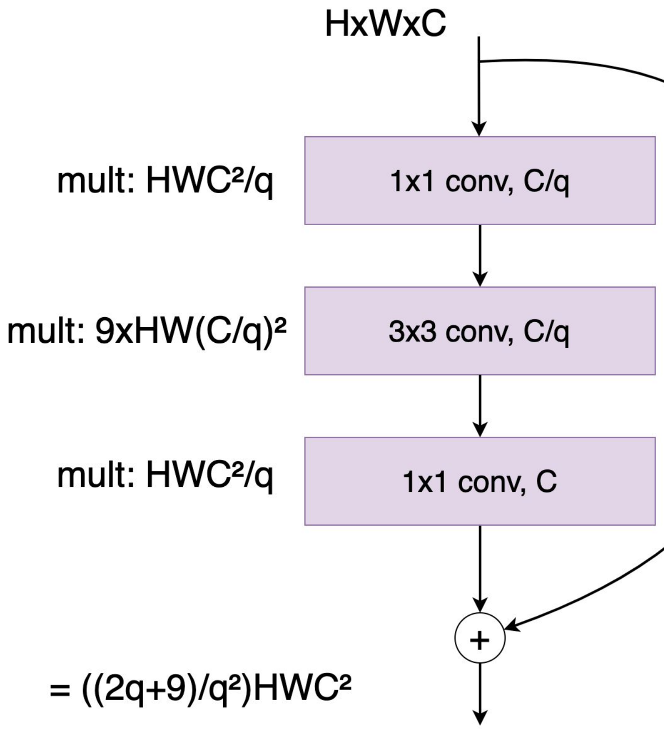
Assuming same padding
Vanilla conv implementation
Only counting conv multiplies

Going deeper with the bottleneck block



Assuming same padding
Vanilla conv implementation
Only counting conv multiplies

Going deeper with the bottleneck block



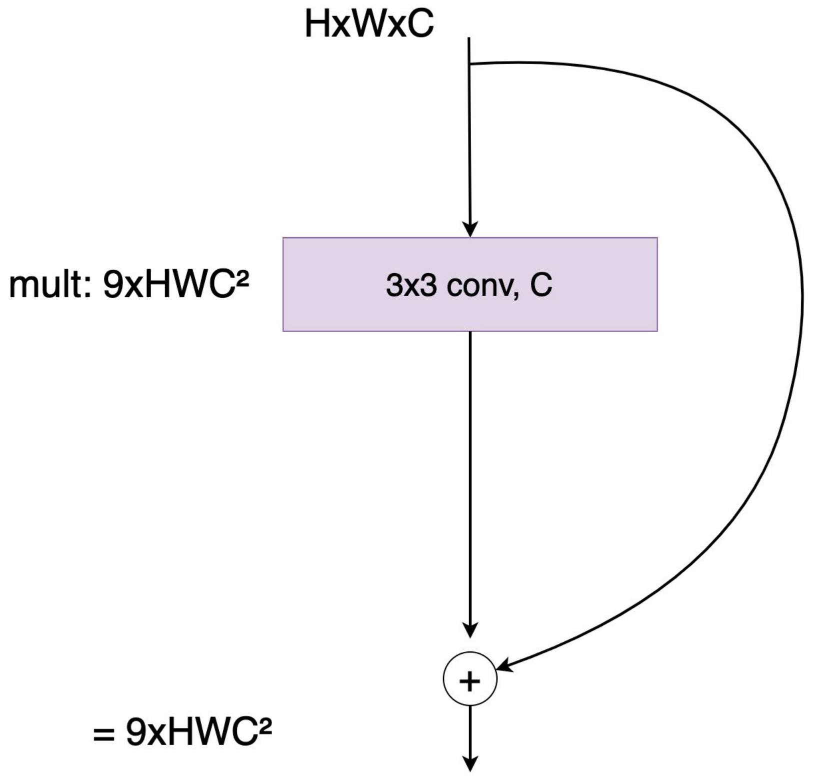
q=2 or 4 usually

Assuming same padding
Vanilla conv implementation
Only counting conv multiplies

Going deeper with the bottleneck block

Architecture	GFLOPS	ImageNet top 5 error
ResNet-34 (basic res block)	3.6	8.58
ResNet-50 (bottleneck block)	3.8	7.13
ResNet-152 (bottleneck block)	11.3	5.84

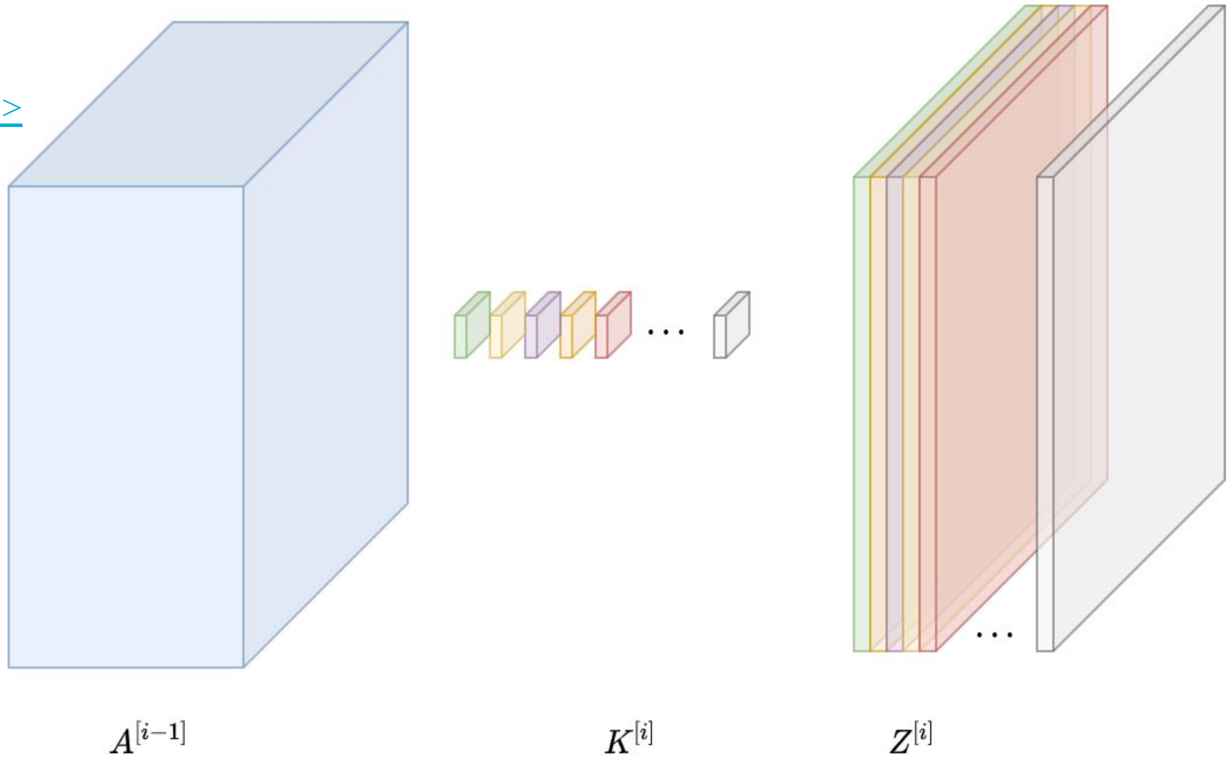
Going deeper with the MobileNet block



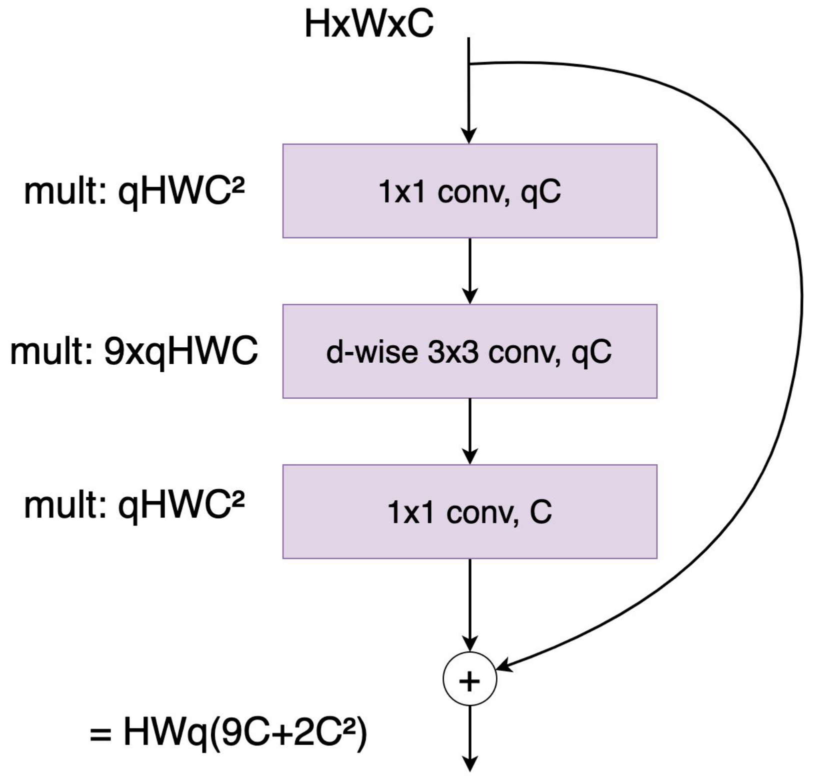
Assuming same padding
Vanilla conv implementation
Only counting conv multiplies

Depthwise (separable) convolution

Implemented as >>>



Going deeper with the MobileNet block

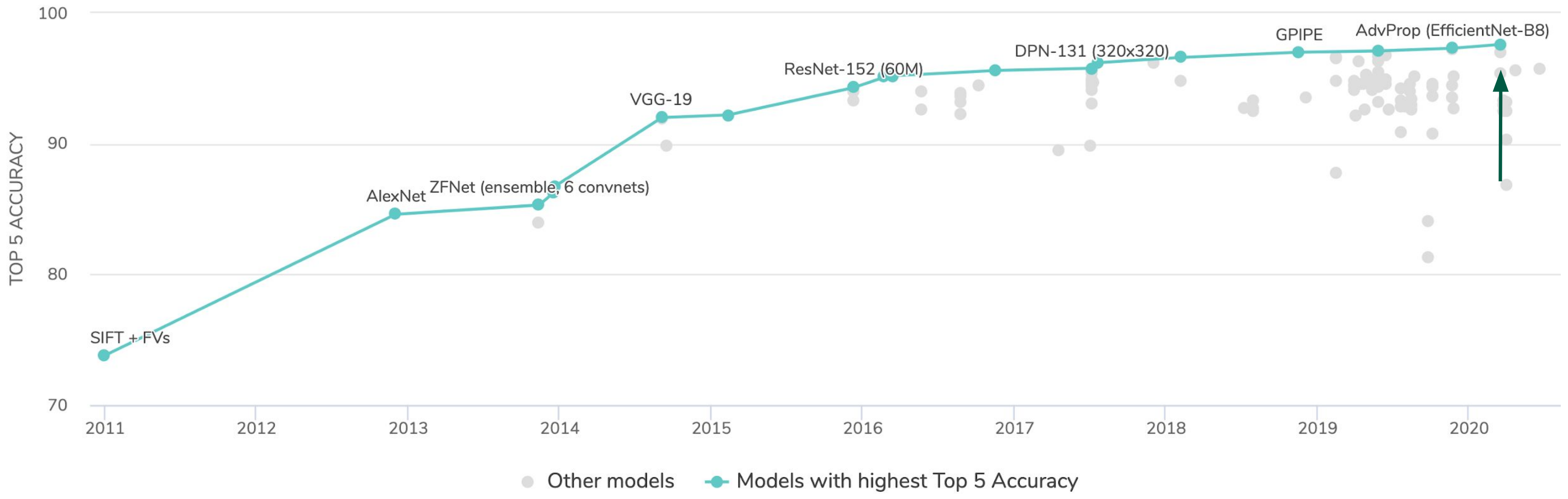


Assuming same padding
Vanilla conv implementation
Only counting conv multiplies

Going deeper with the MobileNet block

- Solving for same FLOPS
- $q \times (9C+2C^2) = 9C^2$
- $q = 9C / (2C+9)$
- $C = 64 \twoheadrightarrow q \sim 4$
- Usually $q = 6$
- … q is an **expansion** parameter.
- “Inverted residual” or MobileNet V2
- Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

ImageNet Classification Challenge

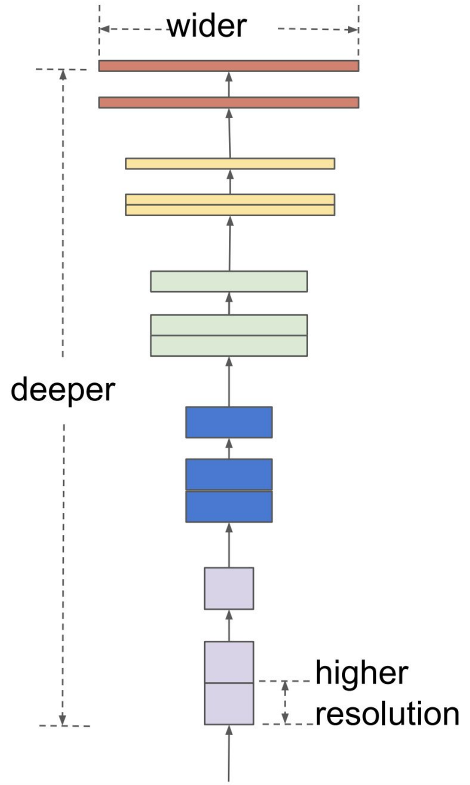


https://paperswithcode.com/sota/image-classification-on-imagenet

Scaling CNNs - EfficientNet

So many hyper-parameters...

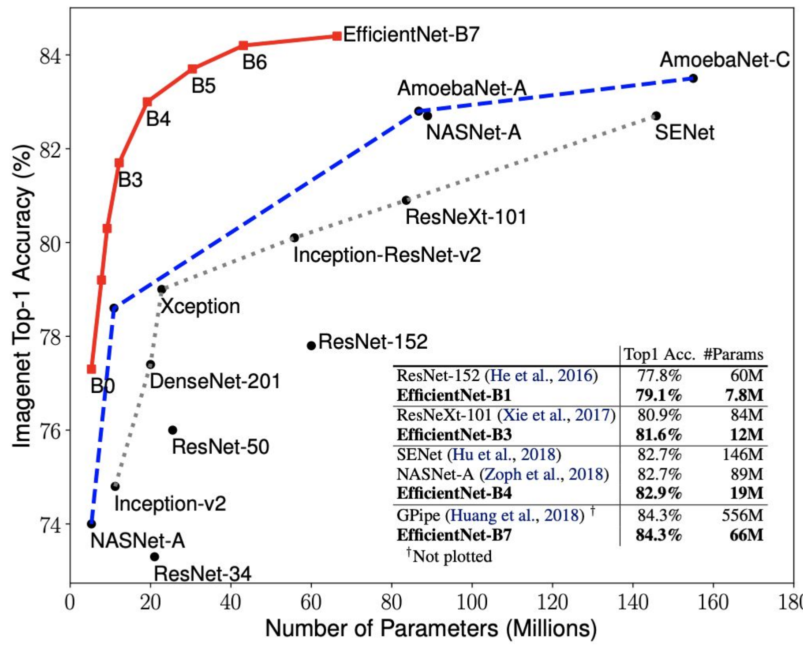
- Number of layers = depth
- Number of channels per layer = width
- Expansion parameter of each layer = q
- Resolution of each layer
- **Need to be balanced w.r.t. Compute constraint**
- **EfficientNet** = figured out a balancing using a heuristic + grid search



Tan, Mingxing, and Quoc V. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." arXiv preprint arXiv:1905.11946 (2019).

EfficientNet

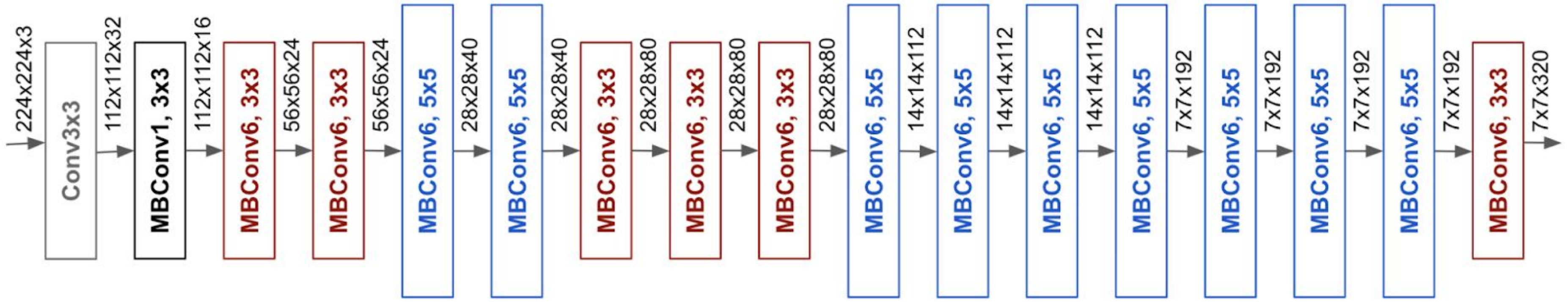
- Easy to use
- State-of-the-art (as of Oct 2020)
- Wins many kaggle competitions



	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-B0



End