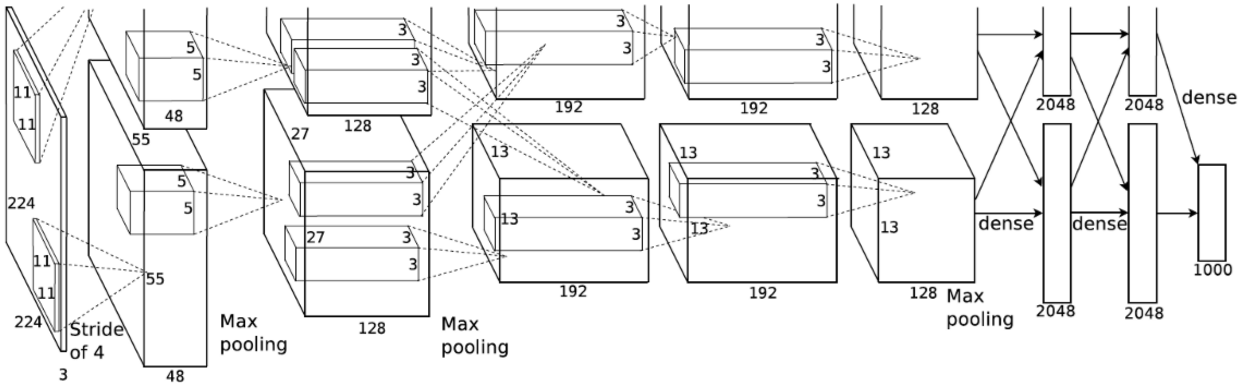
CNN Architectures

Blank Slide

AlexNet



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

PyTorch Class for AlexNet:

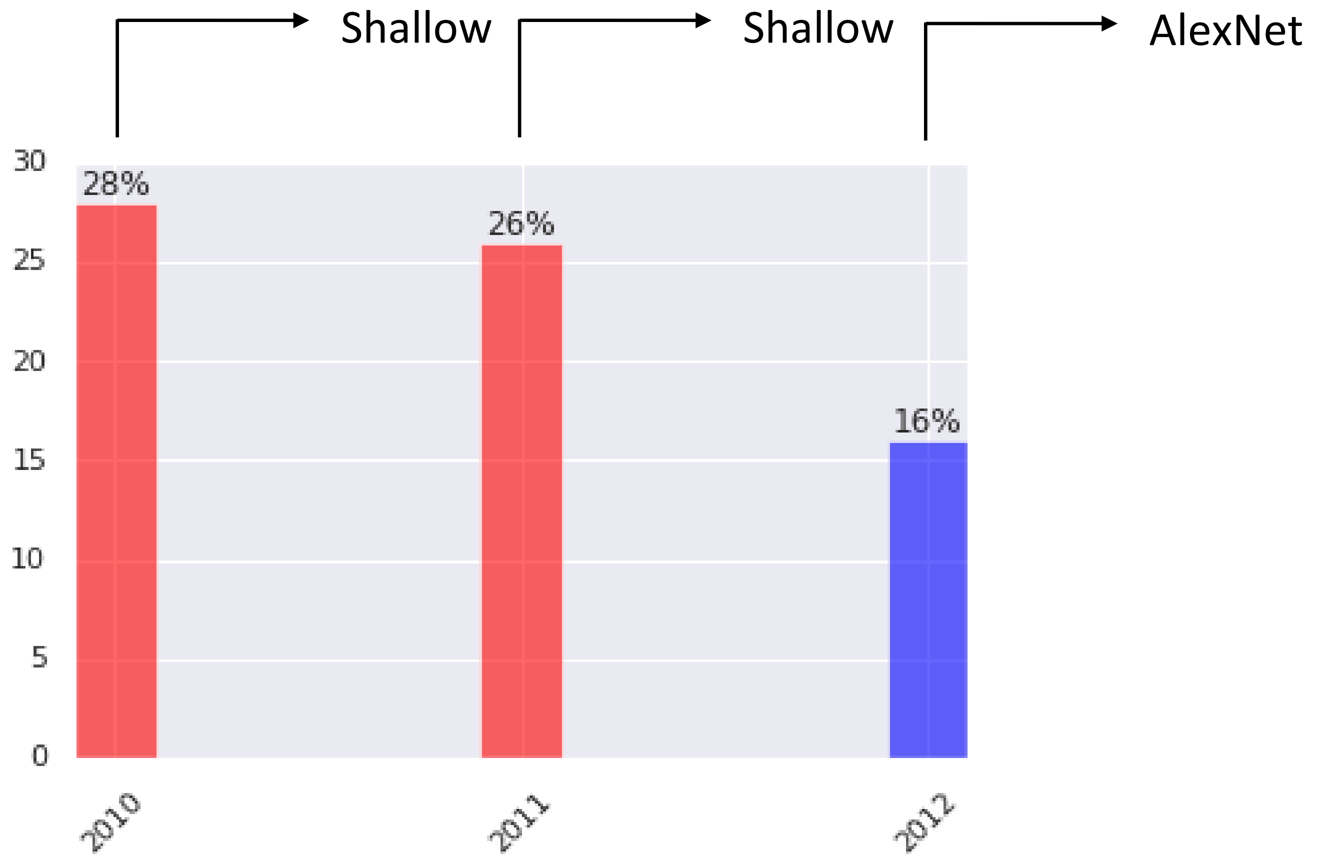
```
import torchvision

alexnet_model = torchvision.models.alexnet(pretrained = True)
```

Image Credit: http://cs231n.stanford.edu/

[Krizhevsky *et. al.*, NeurIPS, 2012]

AlexNet: Performance on ImageNet



VGGNet

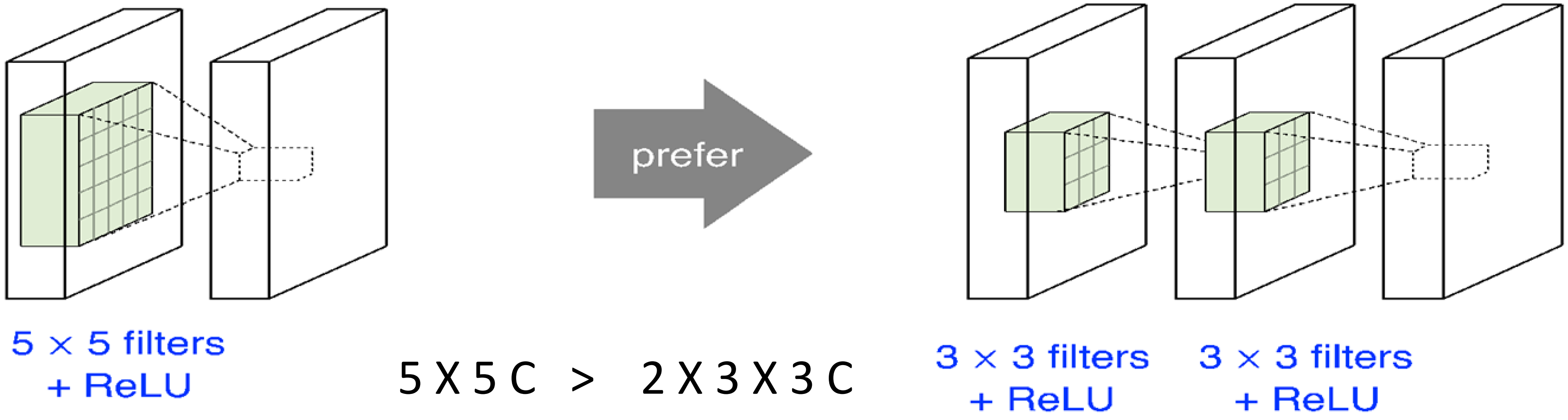
Improvements over AlexNet - I

- Smaller receptive field throughout the network
  - AlexNet used $k = 11 \times 11$ and $s = 4$
  - VGGNet used $k = 3 \times 3$ and $s = 1$

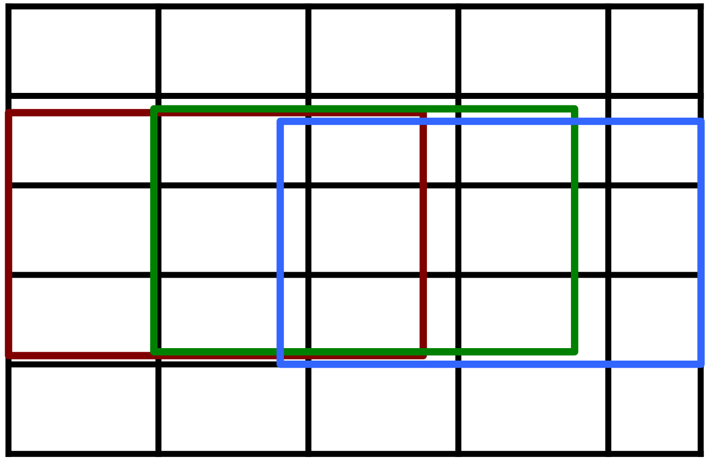
Intuition: A stack of **two** 3 x 3 convolutional layer is equivalent to a 5 x 5 convolution layer
- More non-linearity
- Less number of parameters

[Simonyan *et. al.*, ICLR, 2015]

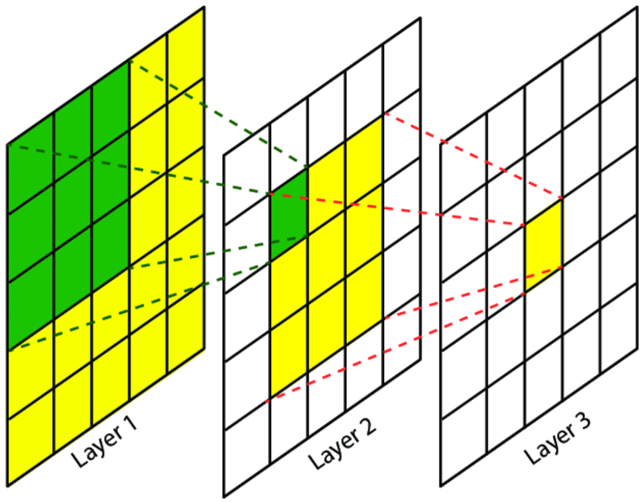
Design Guidelines



5 X 5 C > 2 X 3 X 3 C



1. Less Parameters; Faster
2. Same Receptive Field
3. More nonlinearities (2 ReLU)



Blank Slide

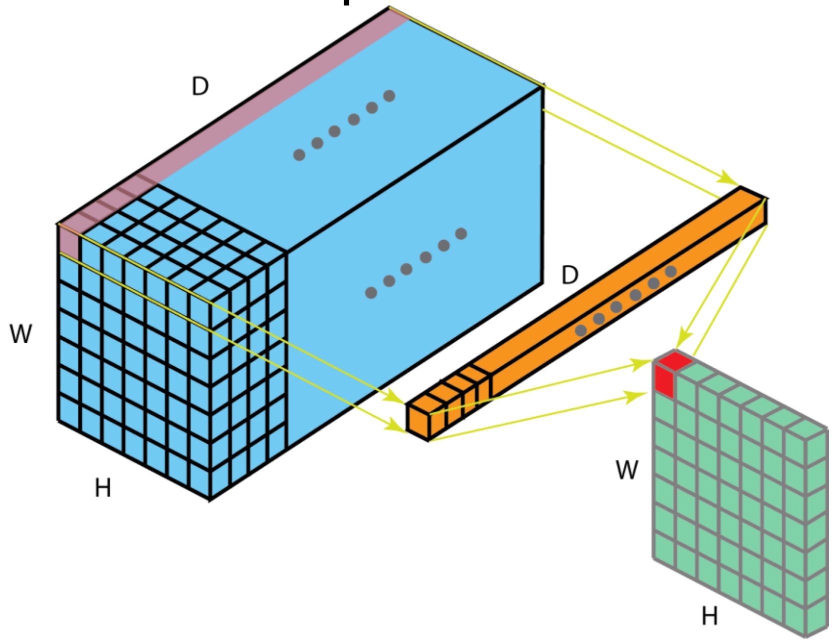
VGGNet

Improvements over AlexNet - II

- 1×1 convolution
  Increases the non-linearity without affecting the receptive field

Usage:

- Dimensionality reduction
- Building deeper network w/ large increase in parameters
- Increased non-linearity



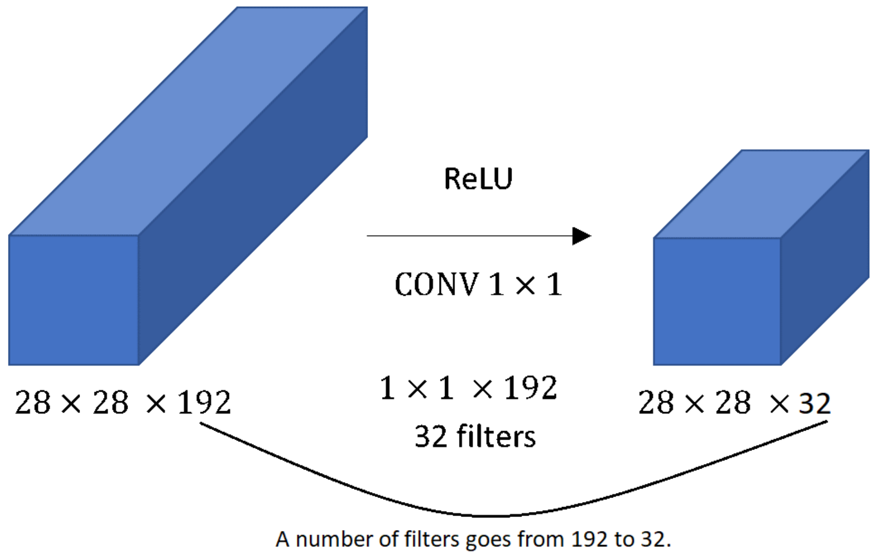


Image Credit: https://cutt.ly/2bK0Gvk

[Lin *et. al.*, ICLR, 2014]

VGGNet

Improvements over AlexNet - III

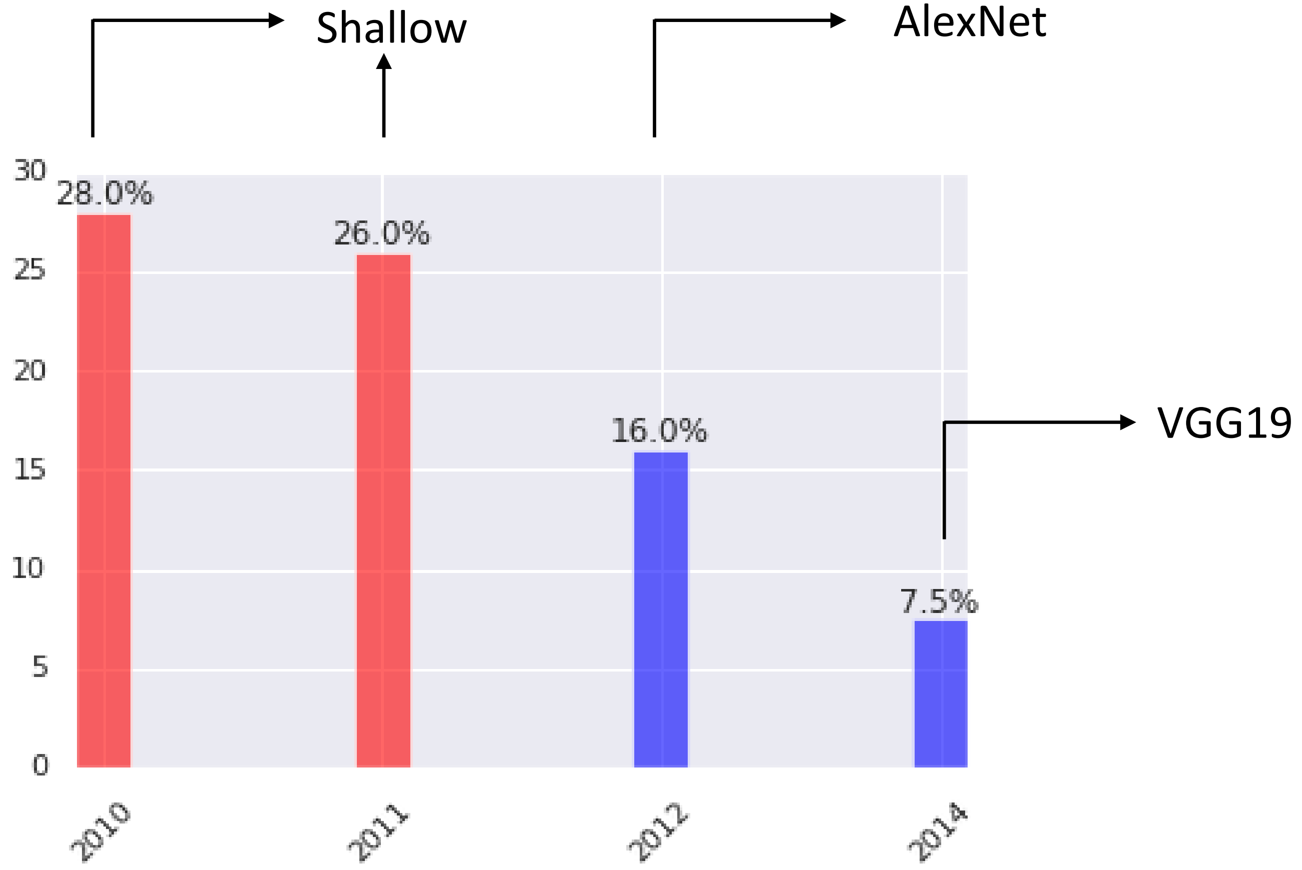
- Deeper Networks
  - VGG-A, VGG-A-LRN (11 layers)
    - 133M parameters
  - VGG-B (13 layers)
    - 133M parameters
  - VGG-C, VGG-D (16 layers)
    - 134M and 138M parameters
  - VGG-E (19 layers)
    - 144M parameters

```python
import torchvision

vggnet_11_model = torchvision.models.vgg11(pretrained = True)
vggnet_13_model = torchvision.models.vgg13(pretrained = True)
vggnet_16_model = torchvision.models.vgg16(pretrained = True)
vggnet_19_model = torchvision.models.vgg19(pretrained = True)
```

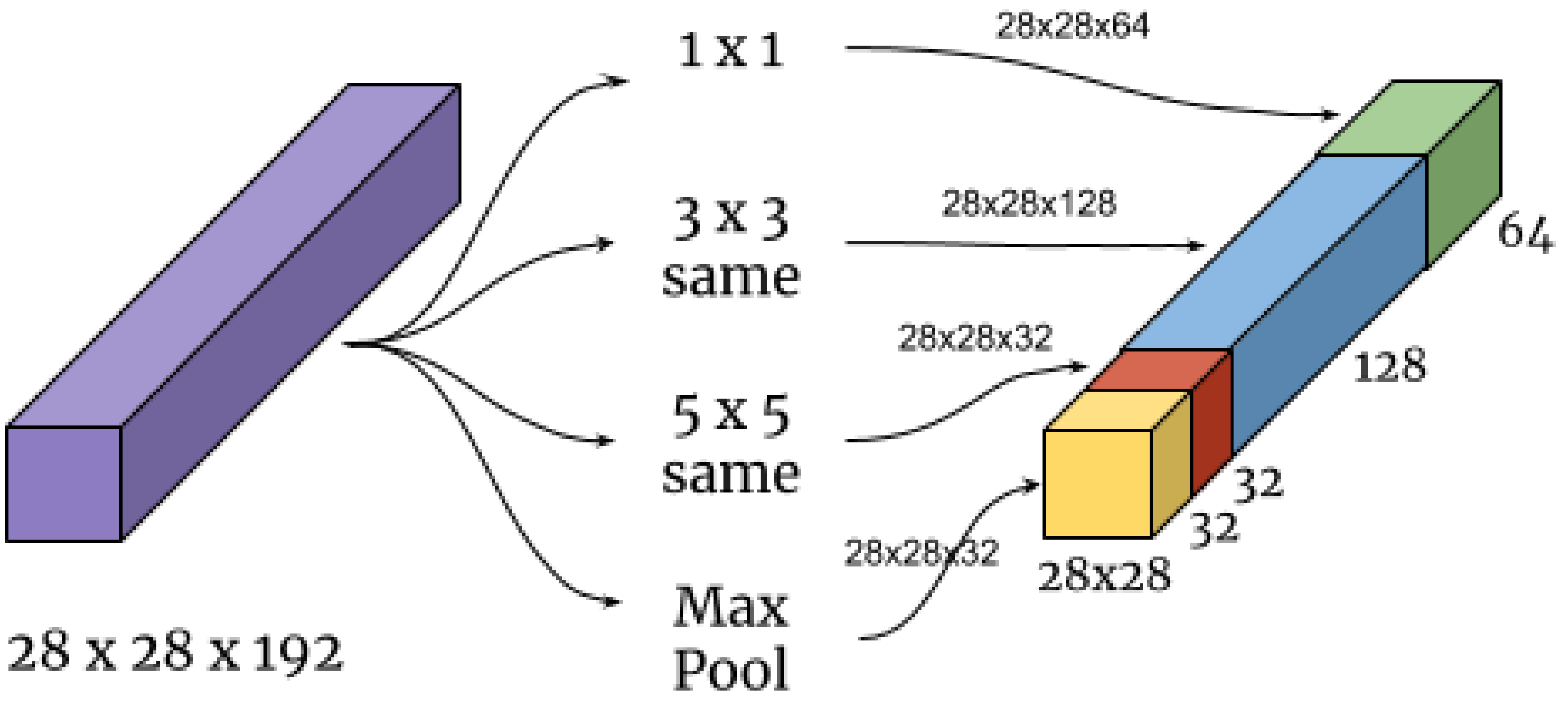
[Simonyan *et. al.*, ICLR, 2015]

VGGNet: Performance on ImageNet



[Simonyan *et. al.*, ICLR, 2015]

GoogLeNet



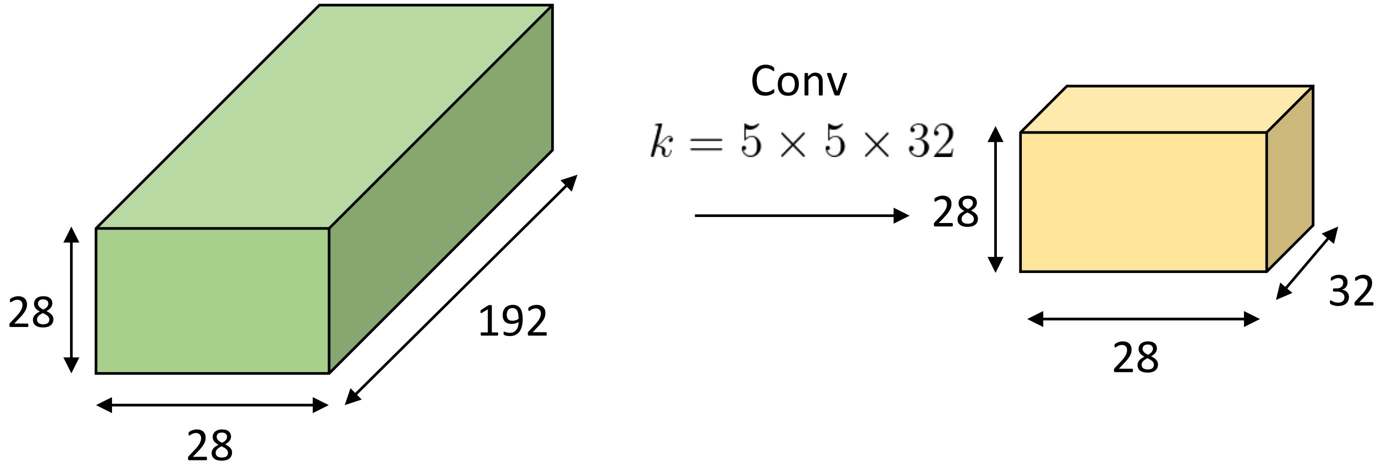
This would take a large number of computations! Can we reduce it?

Image Credit: indoml.com

[Szegedy *et. al.*, CVPR, 2015]

GoogLeNet

Motivation of using a 1x1 Convolutional Layer:



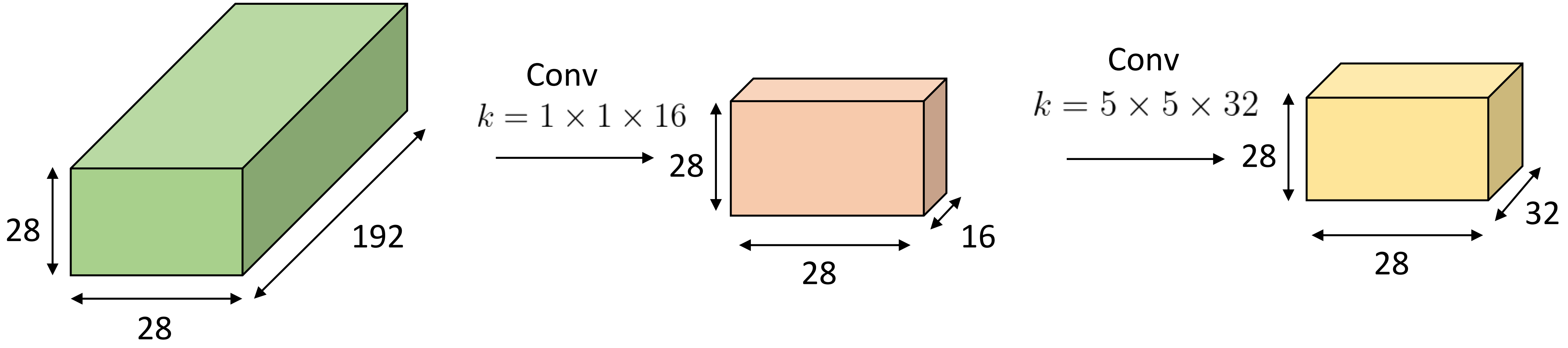
Total number of operations:

(28 x 28 x 32) x (5 x 5 x 192) = 120M

[Szegedy *et. al.*, CVPR, 2015]

GoogLeNet

Motivation:



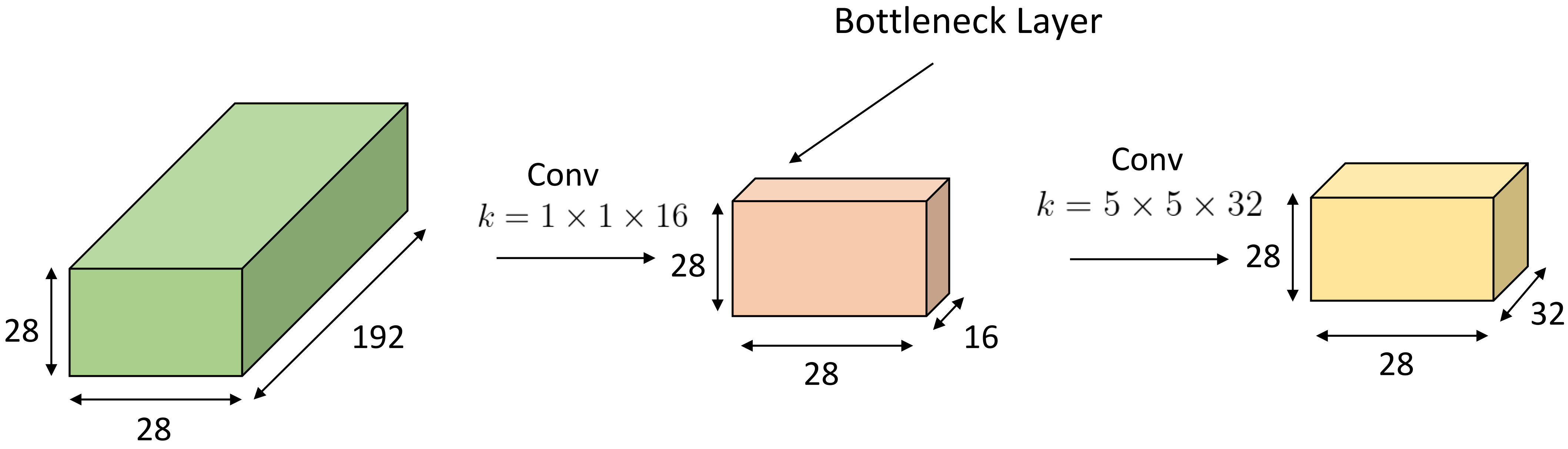
Total number of operations:

(28 x 28 x 16) x (1 x 1 x 192) + (28 x 28 x 32) x (5 x 5 x 16) = 12.4M

[Szegedy *et. al.*, CVPR, 2015]

GoogLeNet

Motivation:



Total number of operations:

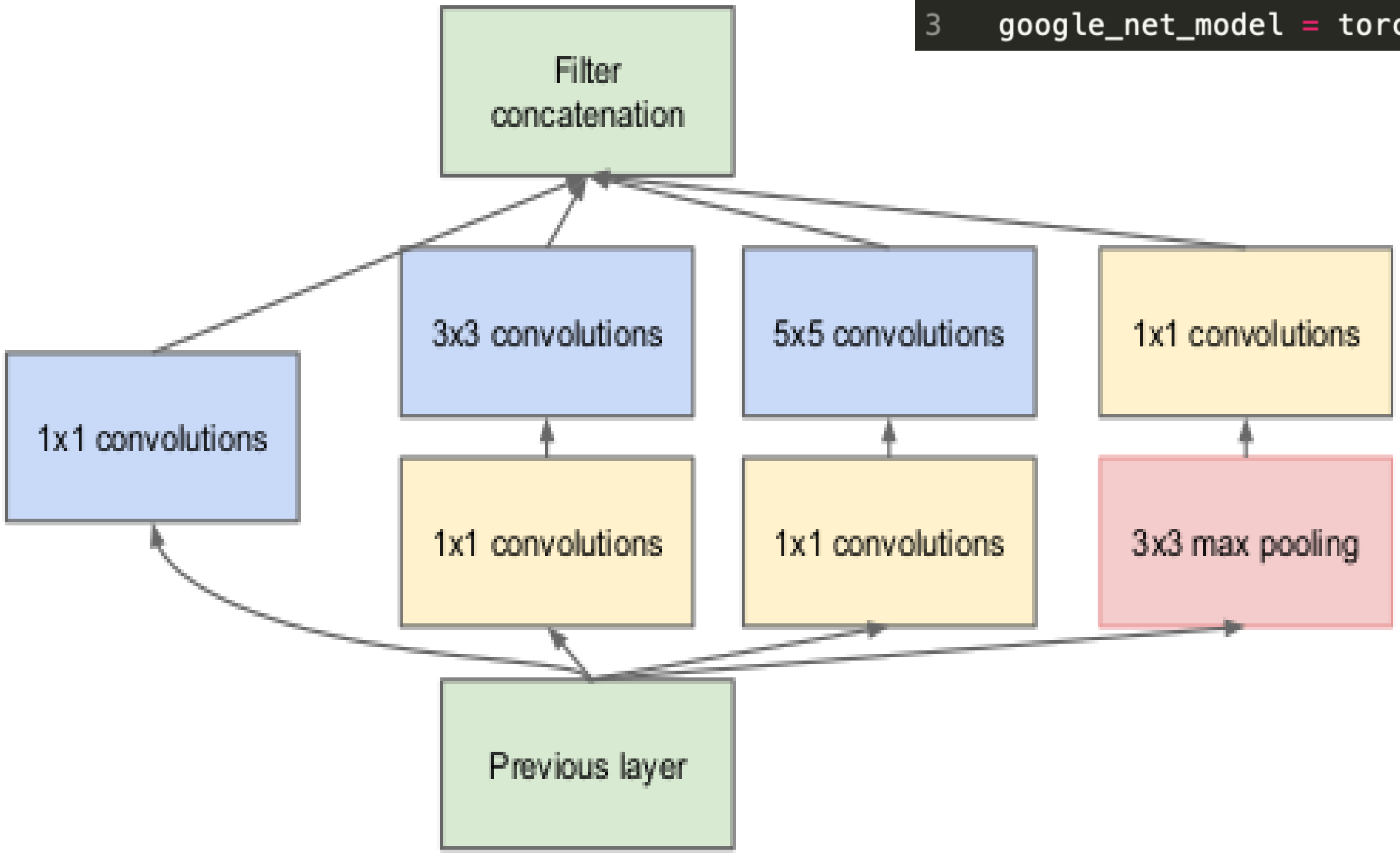
(28 x 28 x 16) x (1 x 1 x 192) + (28 x 28 x 32) x (5 x 5 x 16) = 12.4M

[Szegedy *et. al.*, CVPR, 2015]

The Inception Layer

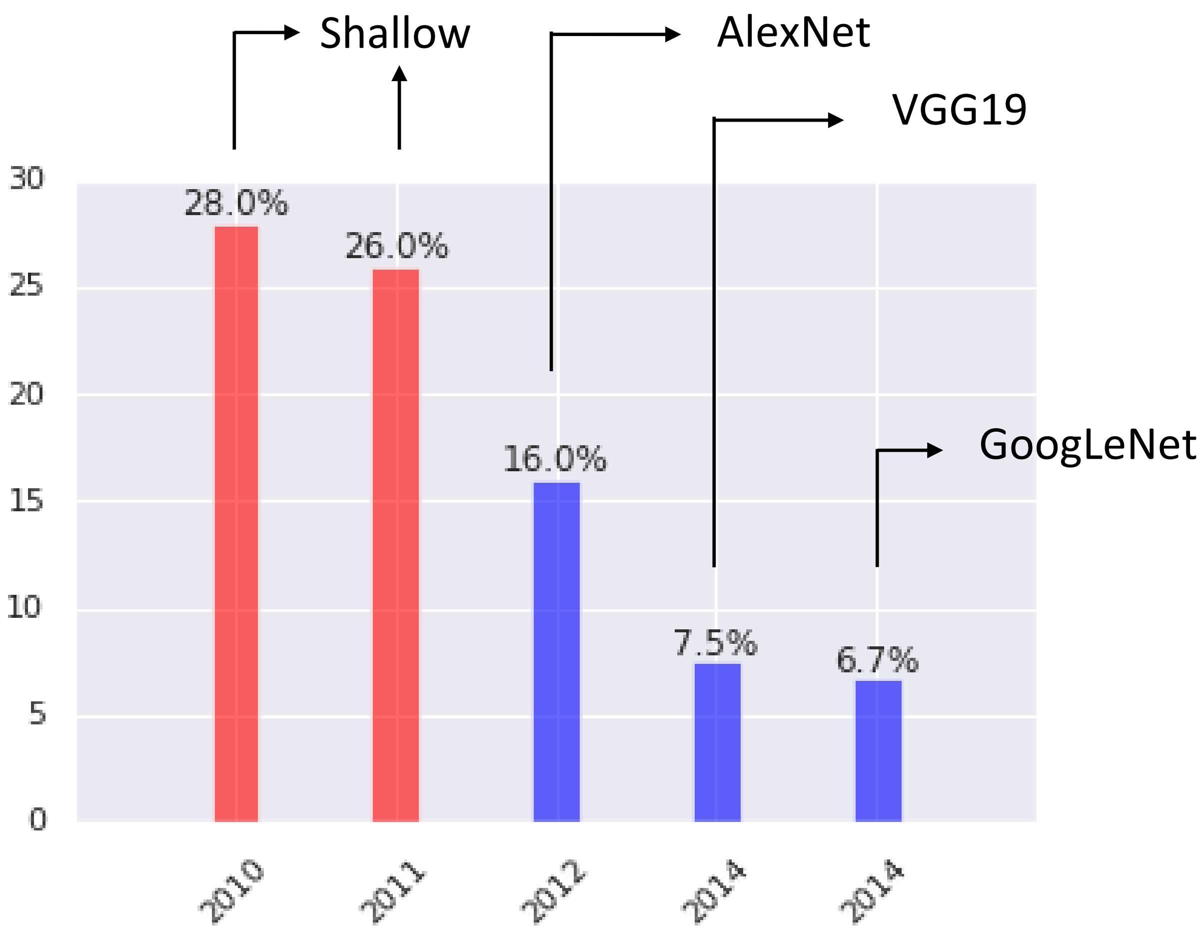
```python
import torchvision

google_net_model = torchvision.models.googlenet(pretrained = True)
```



[Szegedy *et. al.*, CVPR, 2015]

GoogLeNet: Performance on ImageNet



[Szegedy *et. al.*, CVPR, 2015]

ResNets

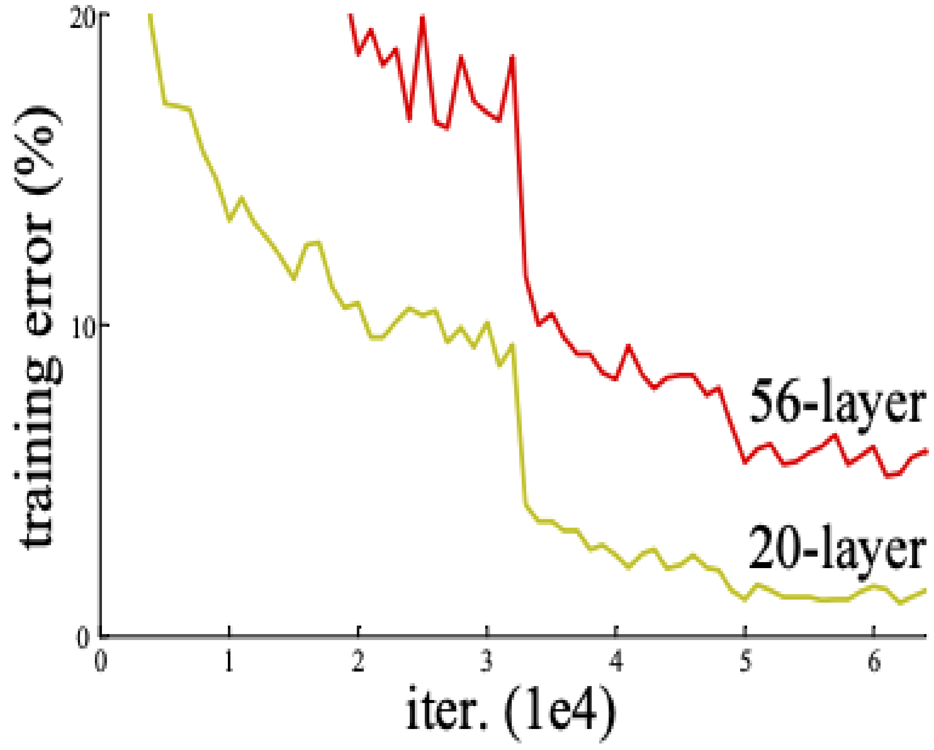
Does increasing the depth of a network always leads to better performance?

[He *et. al.*, CVPR, 2015]

ResNets

Does increasing the depth of a network always leads to better performance?

No!!



[He *et. al.*, CVPR, 2015]

ResNets

The Vanishing Gradient Problem:

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

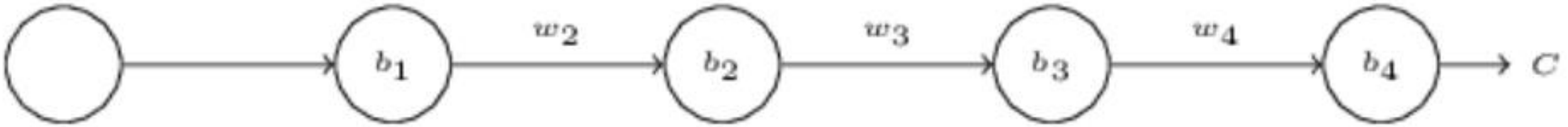
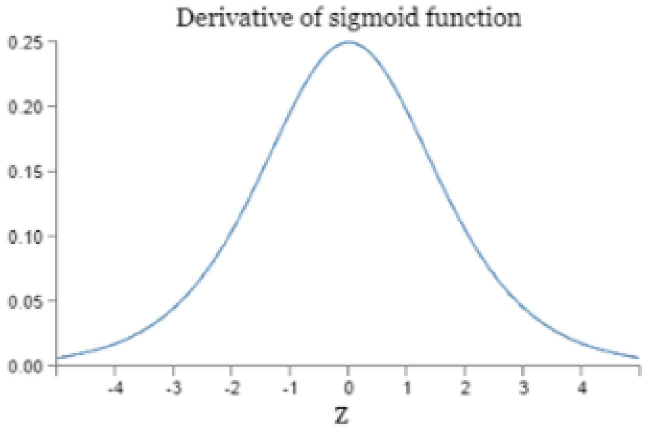


Image Credit: becominghuman.ai

[He *et. al.*, CVPR, 2015]

ResNets

The Vanishing Gradient Problem:



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

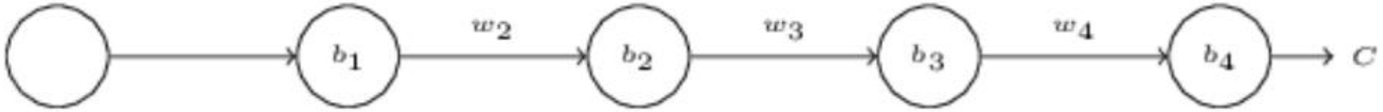
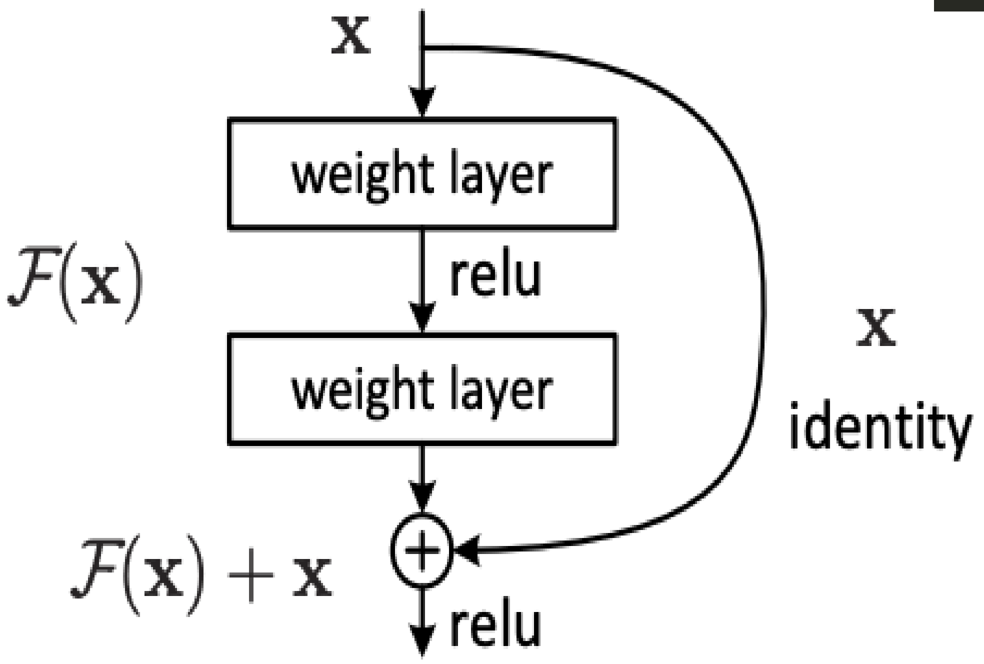


Image Credit: becominghuman.ai

[He *et. al.*, CVPR, 2015]

ResNets

Solution: Skip Connection



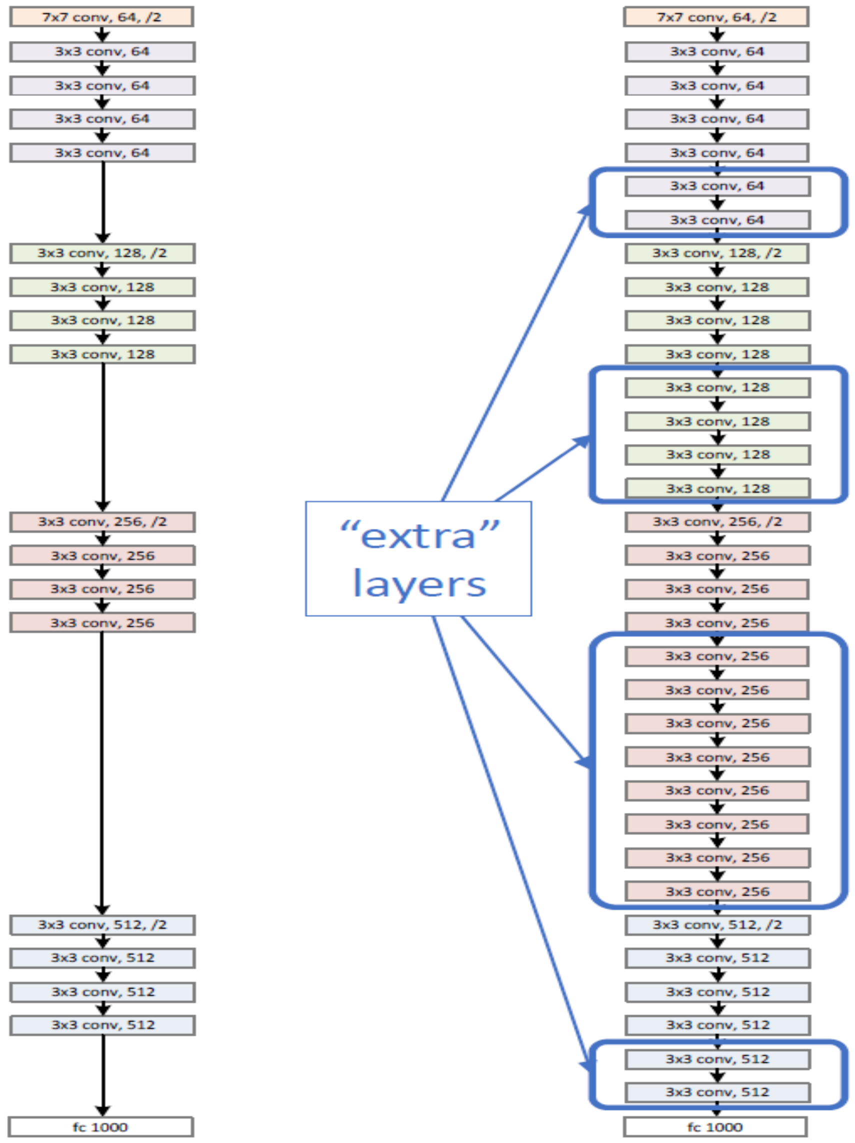

```python
import torchvision

resnet_18_model = torchvision.models.resnet18(pretrained = True)
```

[He *et. al.*, CVPR, 2015]

Simple Argument

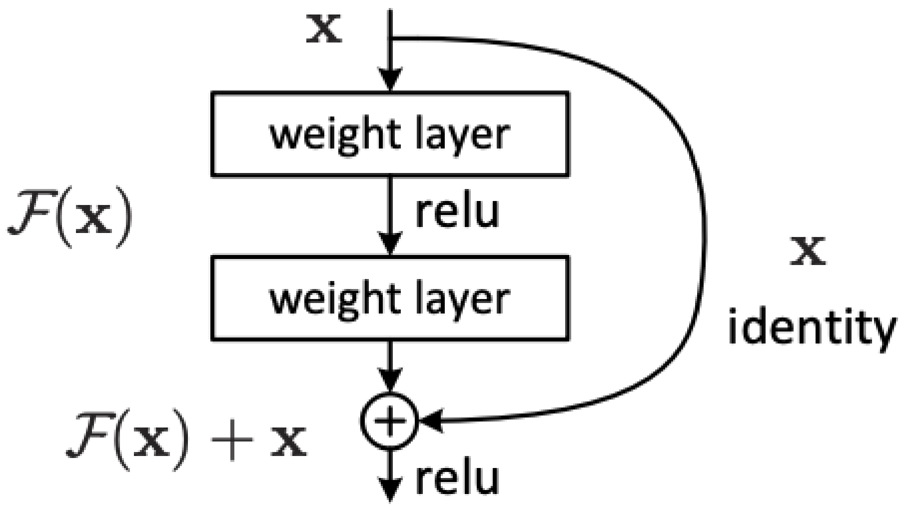
- Naïve solution
  - If extra layers are an identity mapping, then training errors do not increase



Blank Slide

ResNets

Gradient through Skip Connection



$$y = x + \mathcal{F}(x)$$

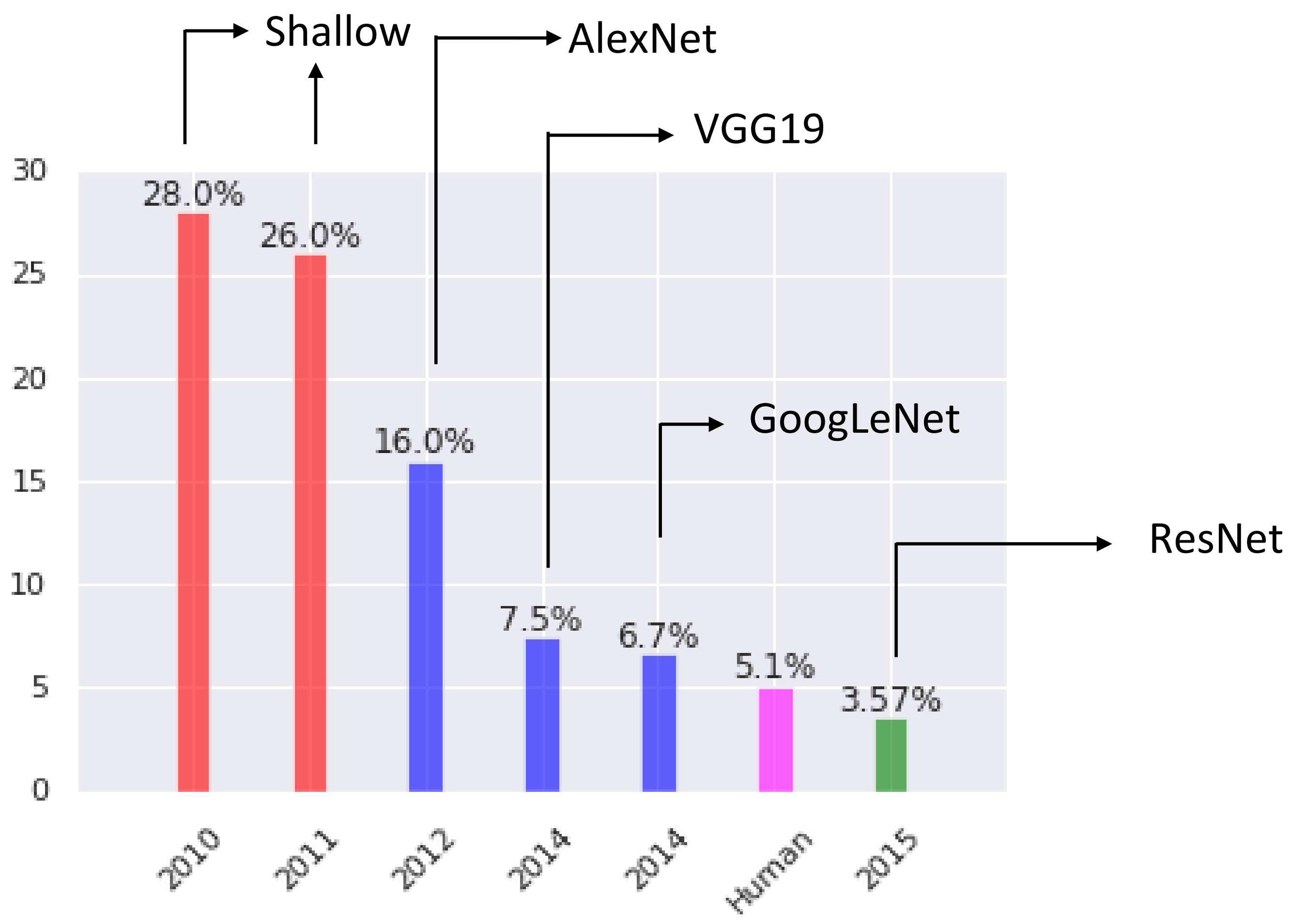
$$\frac{\delta L}{\delta x} = \frac{\delta L}{\delta y}\frac{\delta y}{\delta x}$$
$$= \frac{\delta L}{\delta y}\left(1 + \frac{\delta \mathcal{F}(x)}{\delta x}\right)$$

Gradient from later layer directly passed to the earlier layers!

[He *et. al.*, CVPR, 2015]

Blank Slide

ResNets: Performance on ImageNet



[He *et. al.*, CVPR, 2015]

Which network to choose when?

Which Architecture is the Best?

- ImageNet is to benchmark deep neural networks for image classification task
- All are designed to better accuracy
- Should we choose a network solely based on their performance?
- What are your production constraints?
- How to quantify them into the reasonable metrics to evaluate a CNN?

[Wurm *et. al.*, JPRS, 2019]

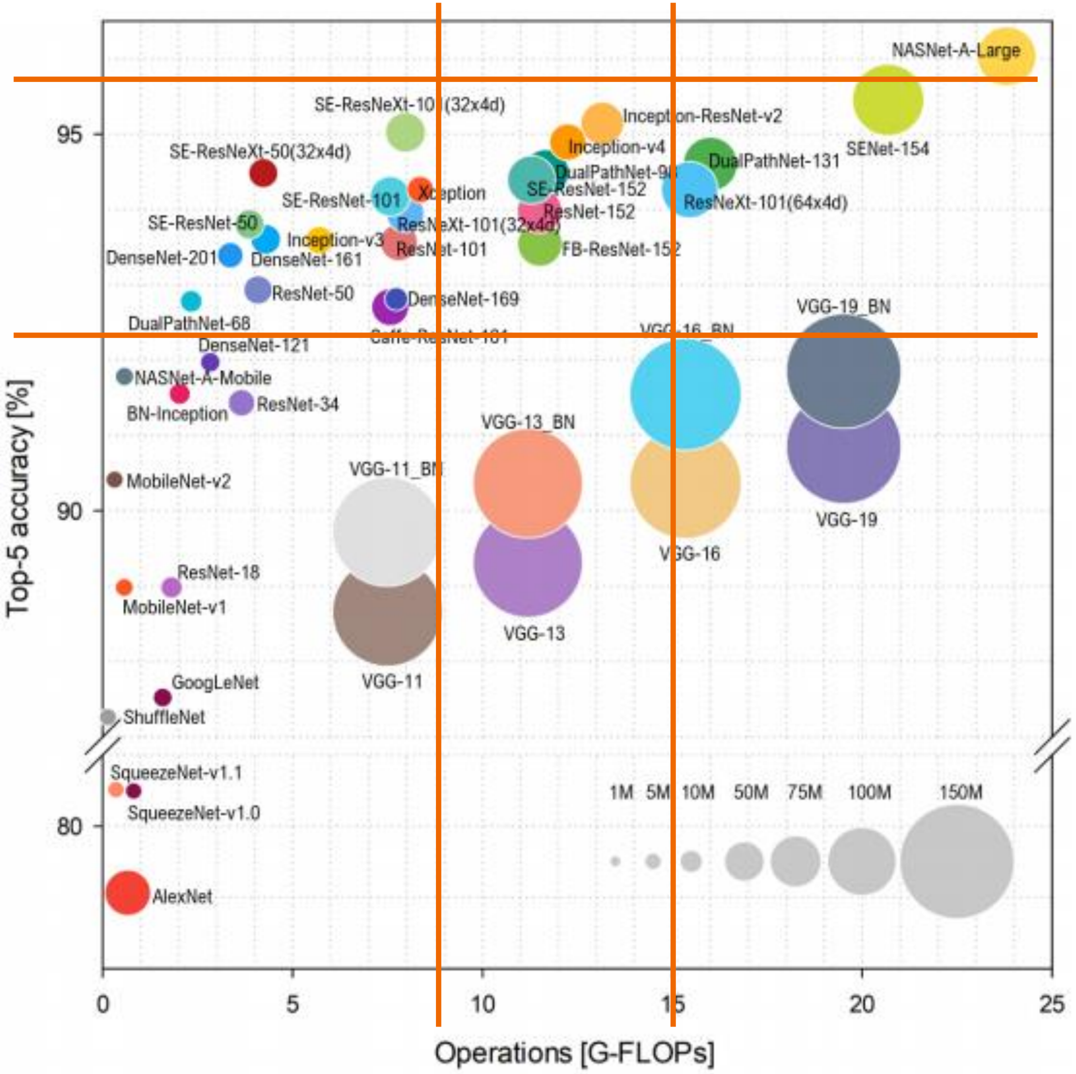
Performance Indices

- Accuracy
- Model complexity
- Memory usage
- Computational complexity
- Inference time

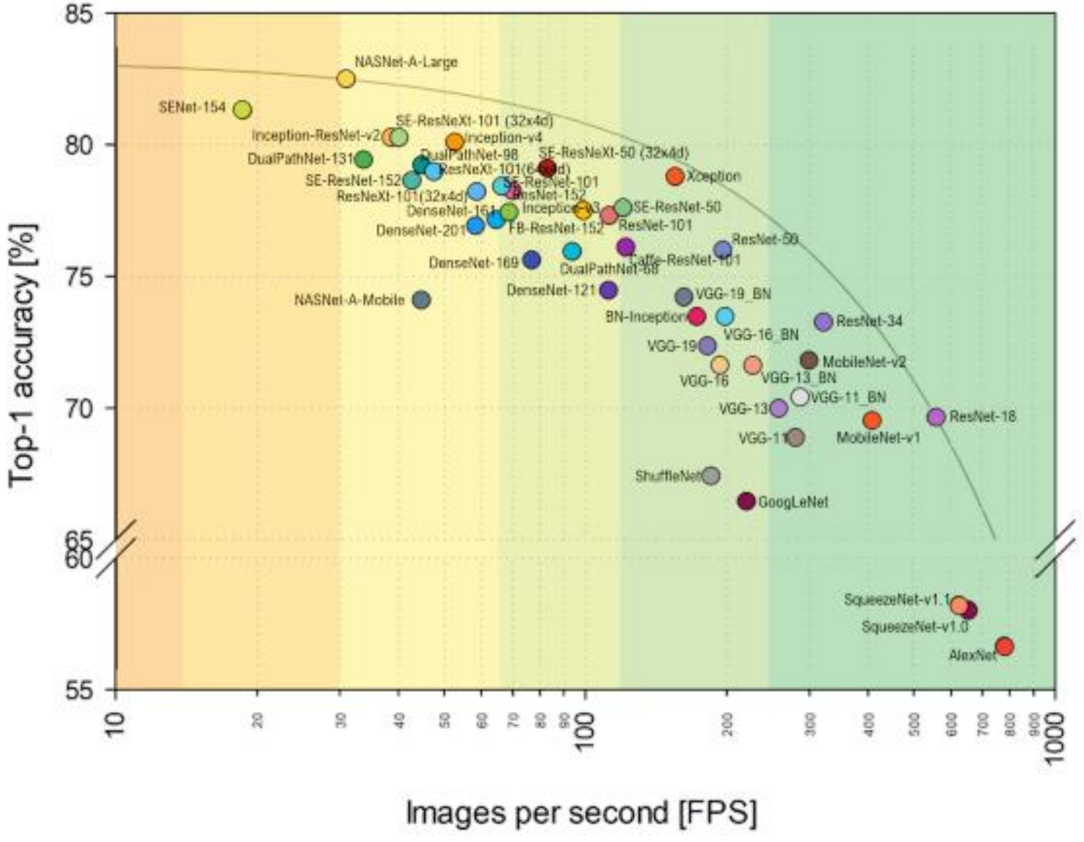
[Wurm *et. al.*, JPRS, 2019]

Accuracy vs. Model Complexity vs. Computational Complexity

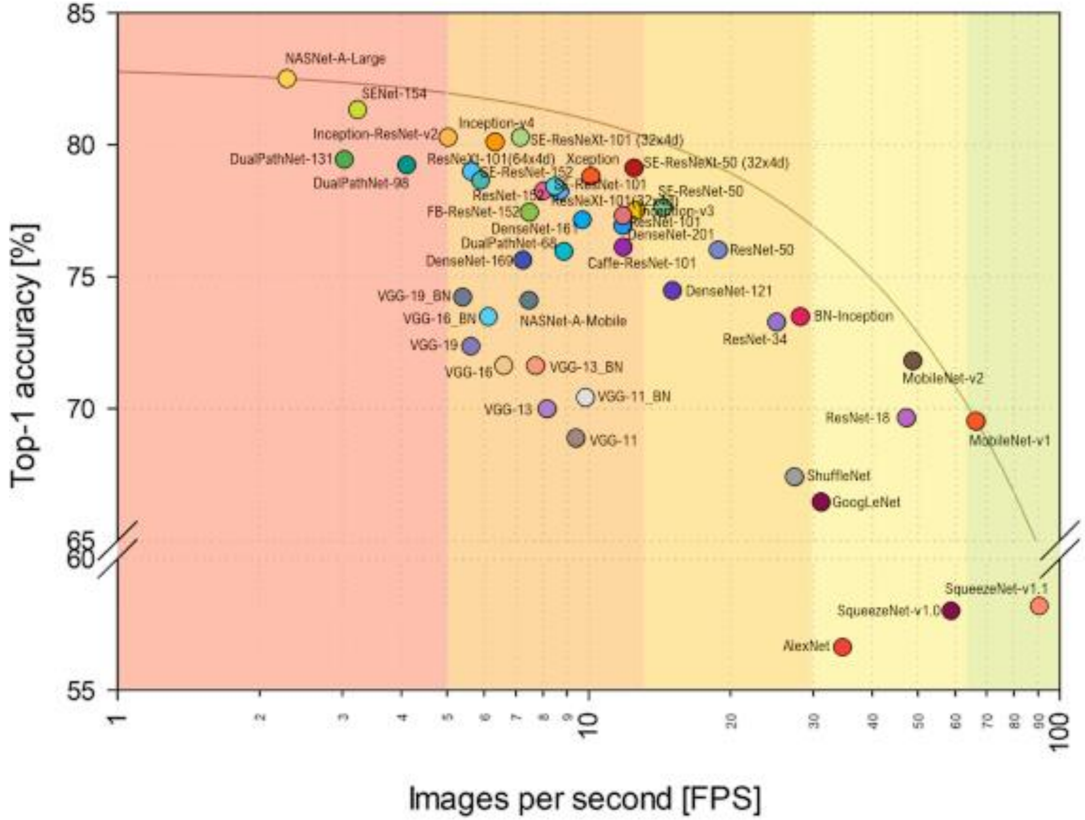
- Size of point denotes the Model complexity
- The band around 95% accuracy has varying complexity of 4-25 G-FLOPs
- The band between 10-15 G-FLOPs have high variance in both Model Complexity (size of the point) and accuracy
- Recognition accuracy is not only dependent on the model or computational complexity



Accuracy vs. Model Complexity vs. Computational Complexity



On TitanX GPU

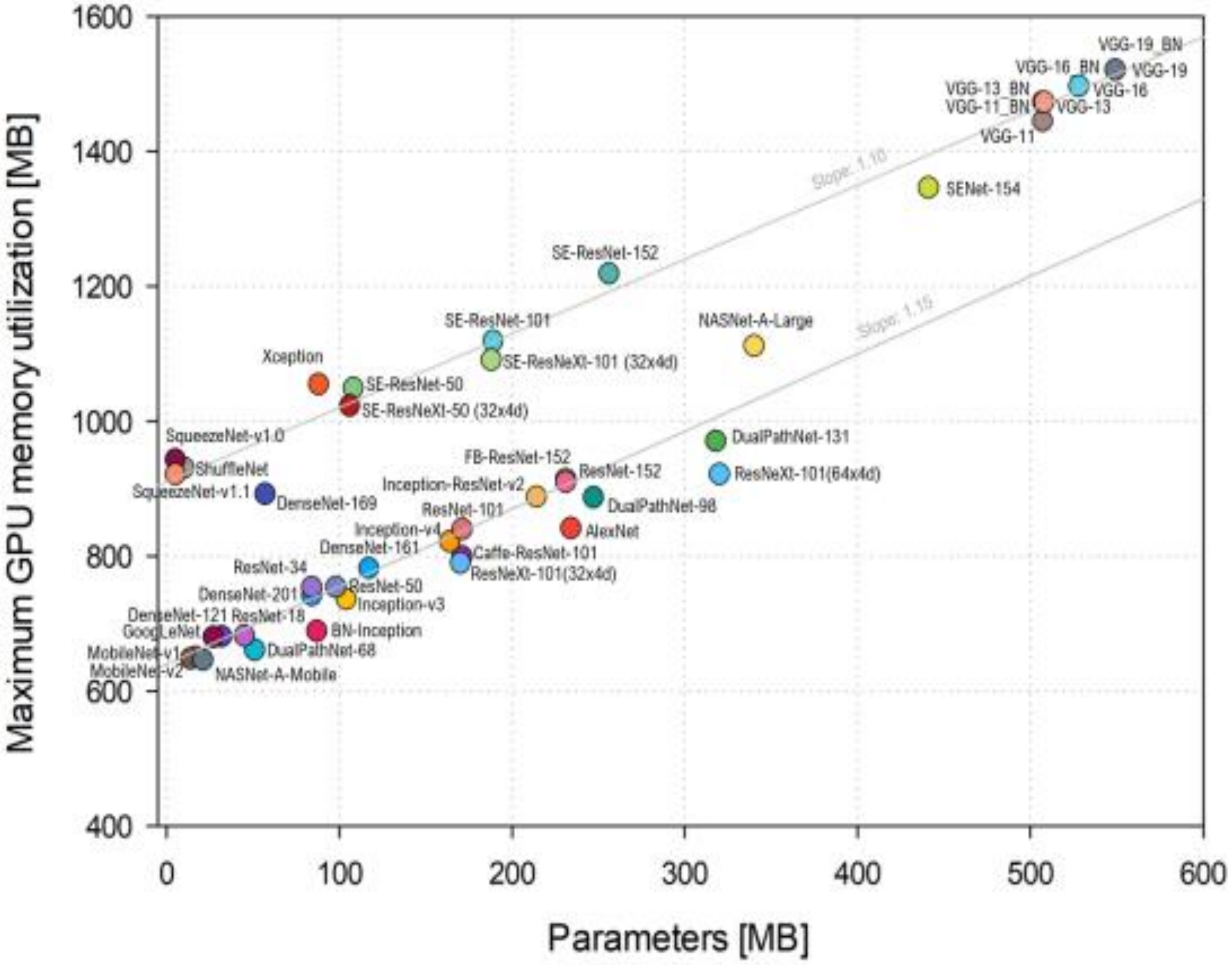


On Nvidia Jetson Embedded Board

Model Complexity vs. Memory Usage

- Follows nearly a linear relationship

- Higher the complexity, higher memory it takes



Thanks!!

Questions?