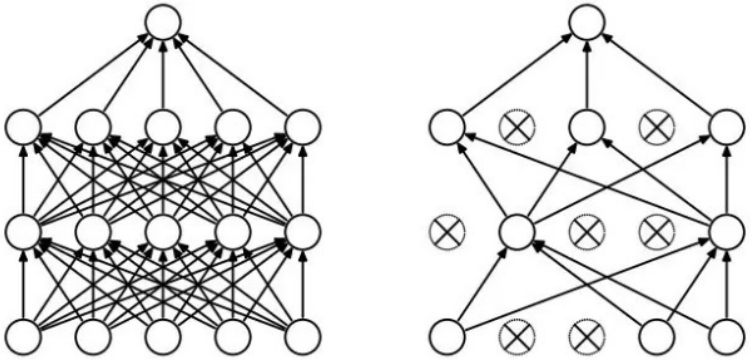
CAP 5516
Medical Image Computing
(Spring 2022)

Dr. Chen Chen
Center for Research in Computer Vision (CRCV)
University of Central Florida
Office: HEC 221
Address: 4328 Scorpius St., Orlando, FL 32816-2365
Email: chen.chen@crcv.ucf.edu
Web: https://www.crcv.ucf.edu/chenchen/

Lecture 7: Introduction to Deep Learning (3)

Regularization



Dropout

- Randomly drop units (along with their connections) during training
- Each unit retained with fixed probability p, independent of other units
- Hyper-parameter p to be chosen (tuned)

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research (2014)

L2 = weight decay

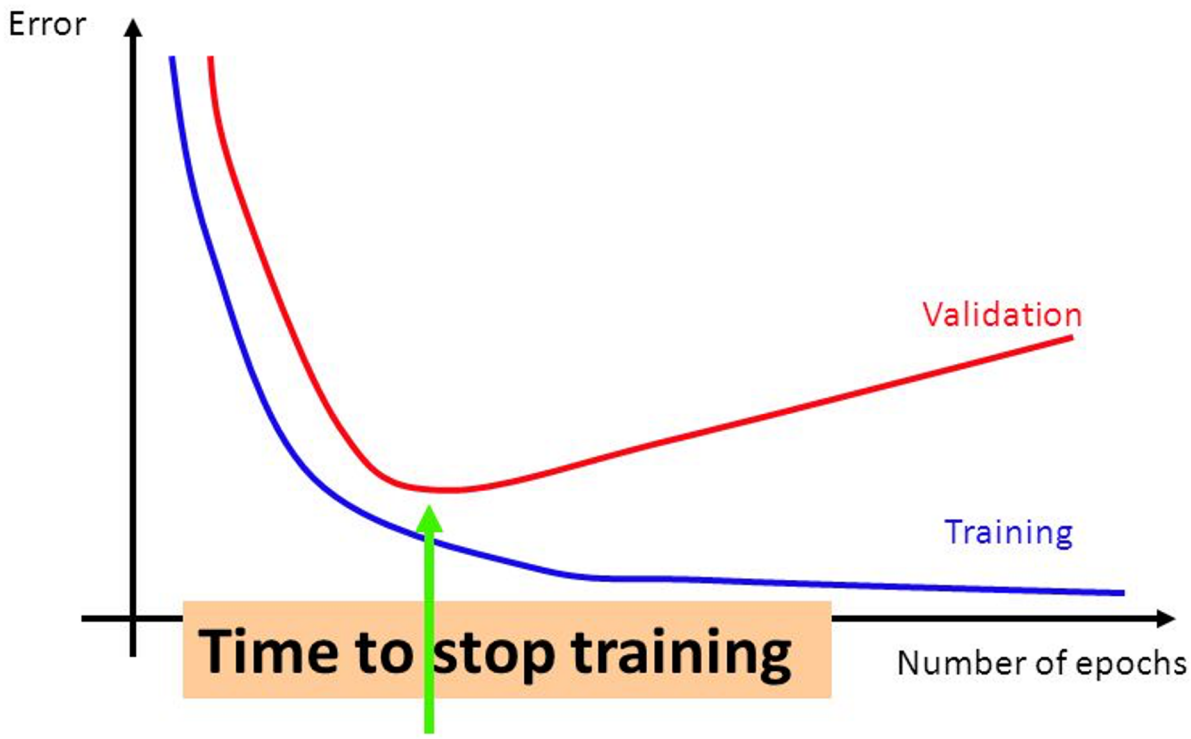
- Regularization term that penalizes big weights, added to the objective – reduce overfitting
- Weight decay value determines how dominant regularization is during gradient computation
- Big weight decay coefficient → big penalty for big weights

$$J_{reg}(\theta) = J(\theta) + \lambda \sum_k \theta_k^2$$

Early-stopping

- Use validation error to decide when to stop training
- Stop when monitored quantity has not improved after n subsequent epochs

Early Stopping



Credit: Stephen Marsland

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

Batch Normalization

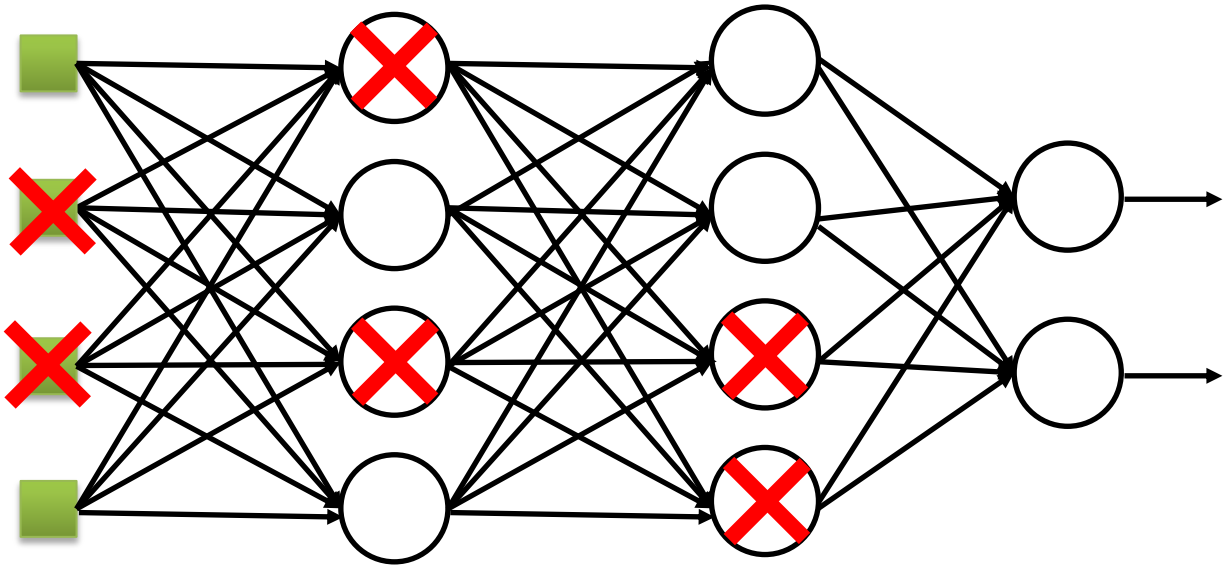
```
1  # example of batch normalization for an cnn
2  from keras.layers import Dense
3  from keras.layers import Conv2D
4  from keras.layers import MaxPooling2D
5  from keras.layers import BatchNormalization
6  ...
7  model.add(Conv2D(32, (3,3), activation='relu'))
8  model.add(Conv2D(32, (3,3), activation='relu'))
9  model.add(BatchNormalization())
10 model.add(MaxPooling2D())
11 model.add(Dense(1))
12 ...
```

Resnet architecture (conv/bn/relu)

https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/

Dropout

Training:

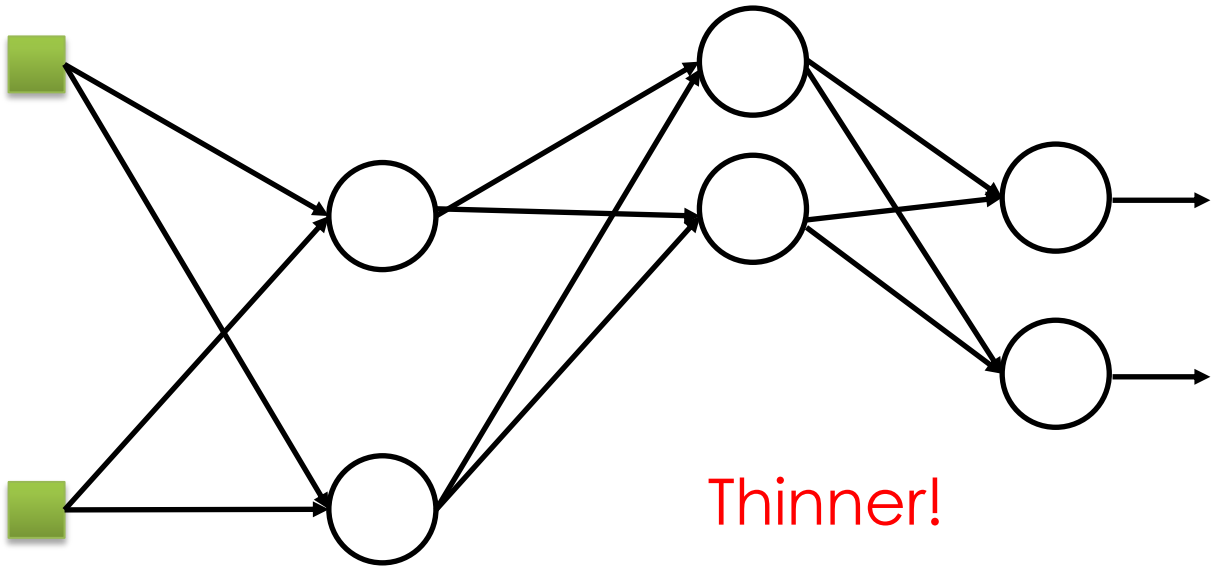


- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Dropout

Training:

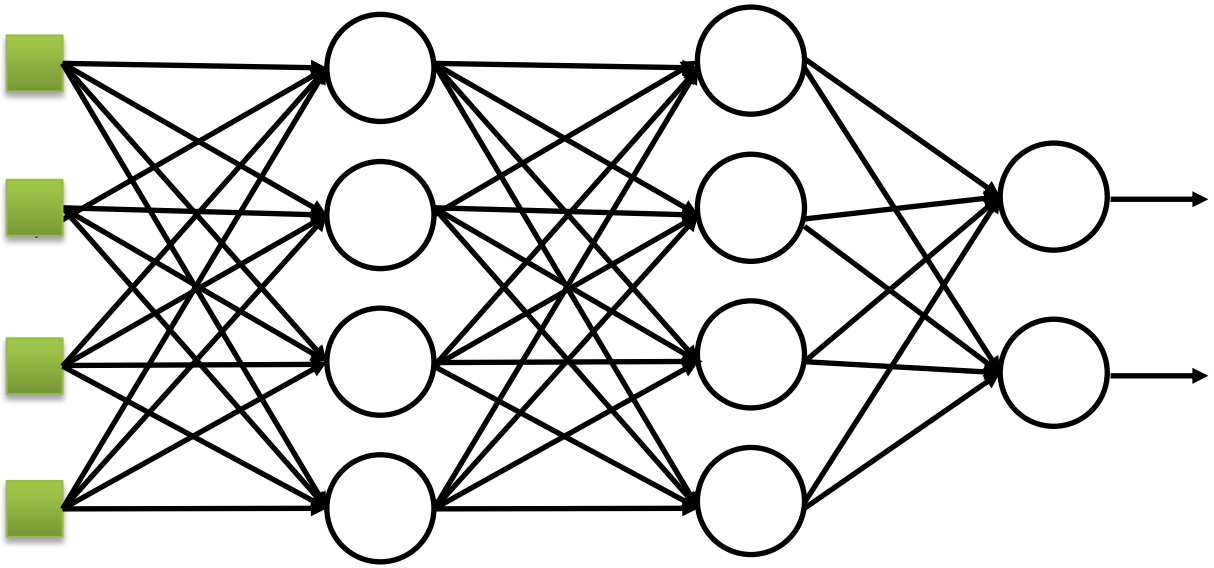


- **Each time before updating the parameters**
  - Each neuron has p% to dropout
  - Using the new network for training

The structure of the network is changed.

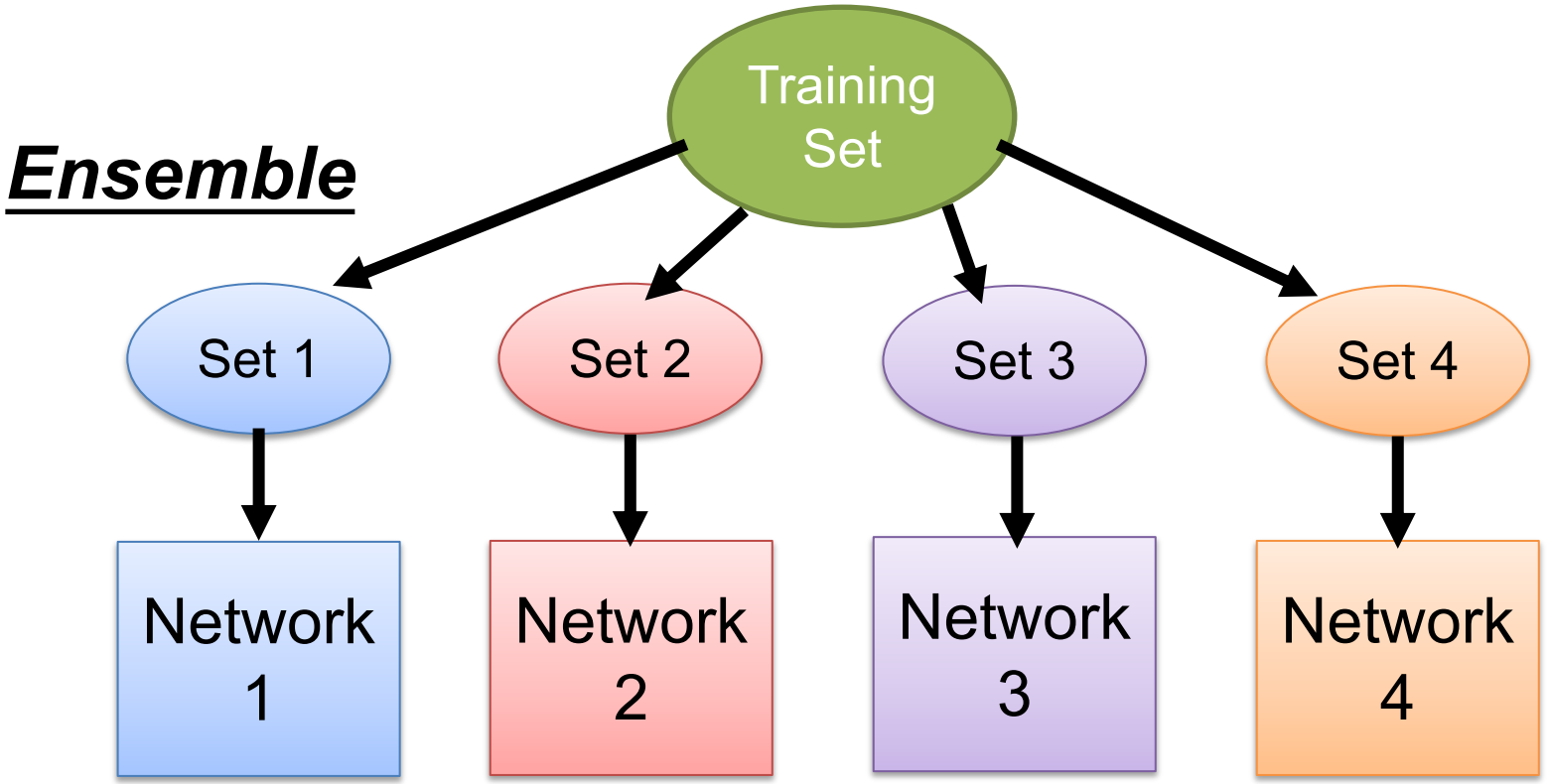
Dropout

Testing:



- No dropout

Dropout is a kind of ensemble



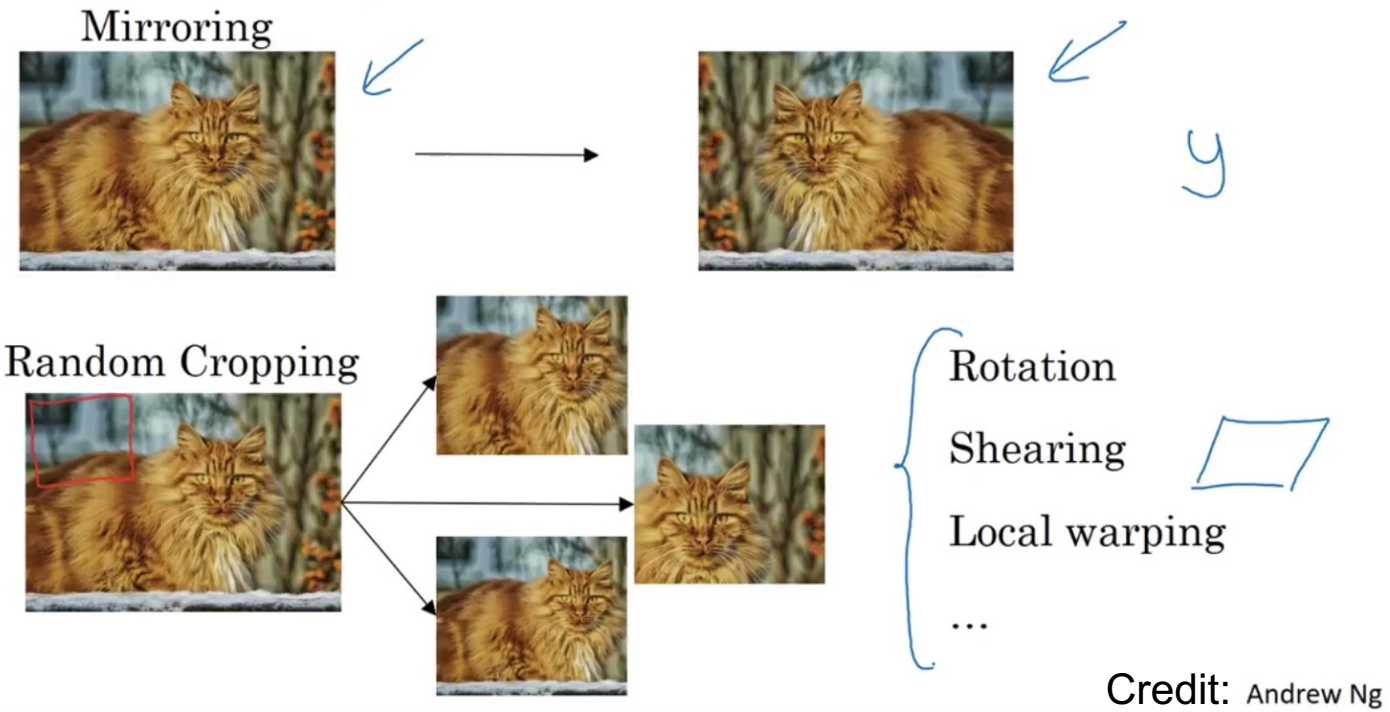
Train a bunch of networks with different structures

More about dropout

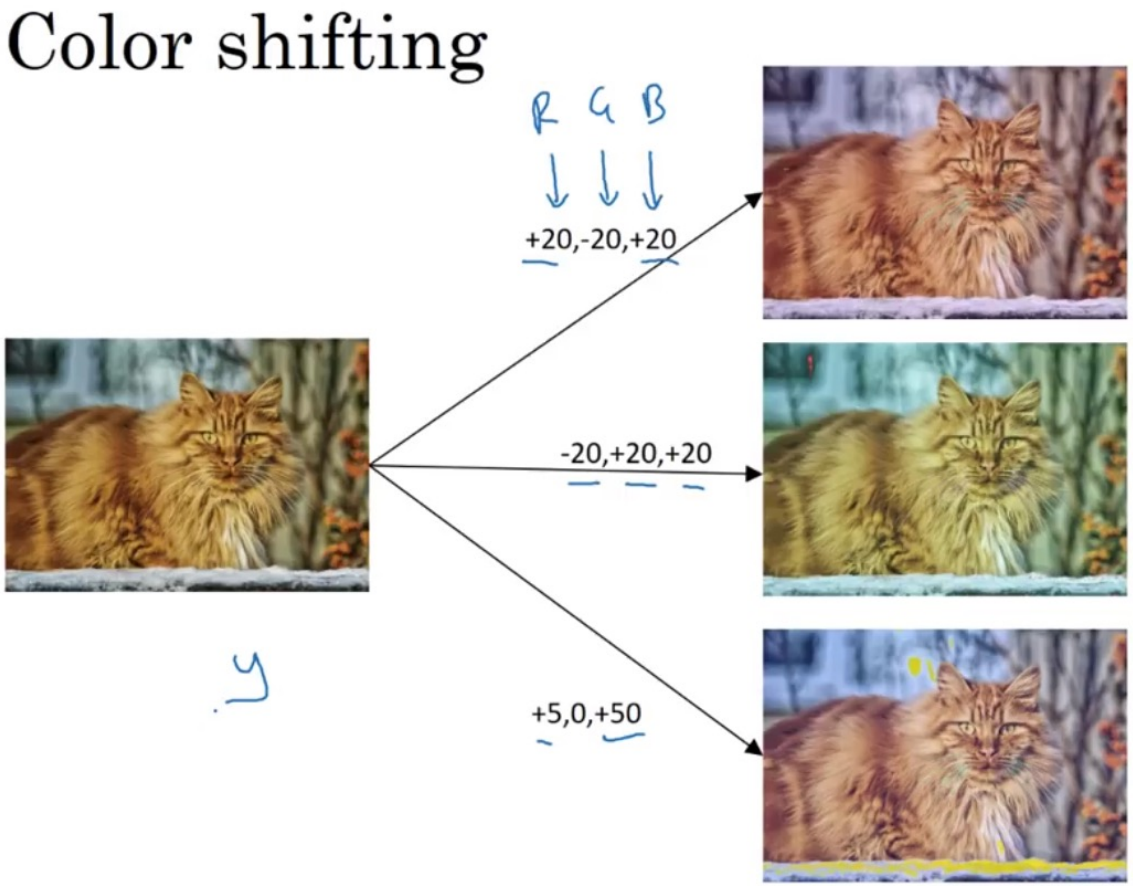
- More reference for dropout [Nitish Srivastava, JMLR'14] [Pierre Baldi, NIPS'13][Geoffrey E. Hinton, arXiv'12]
- Dropout works better with Maxout [Ian J. Goodfellow, ICML'13]
- Dropconnect [Li Wan, *ICML'13*]
  - Dropout delete neurons
  - Dropconnect deletes the connection between neurons
- Annealed dropout [S.J. Rennie, SLT'14]
  - Dropout rate decreases by epochs
- Standout [J. Ba, NISP'13]
  - Each neural has different dropout rate

Data Augmentation

- Why data augmentation -> increase training data



Data Augmentation



Credit: Andrew Ng

Data Augmentation

- Color space conversion

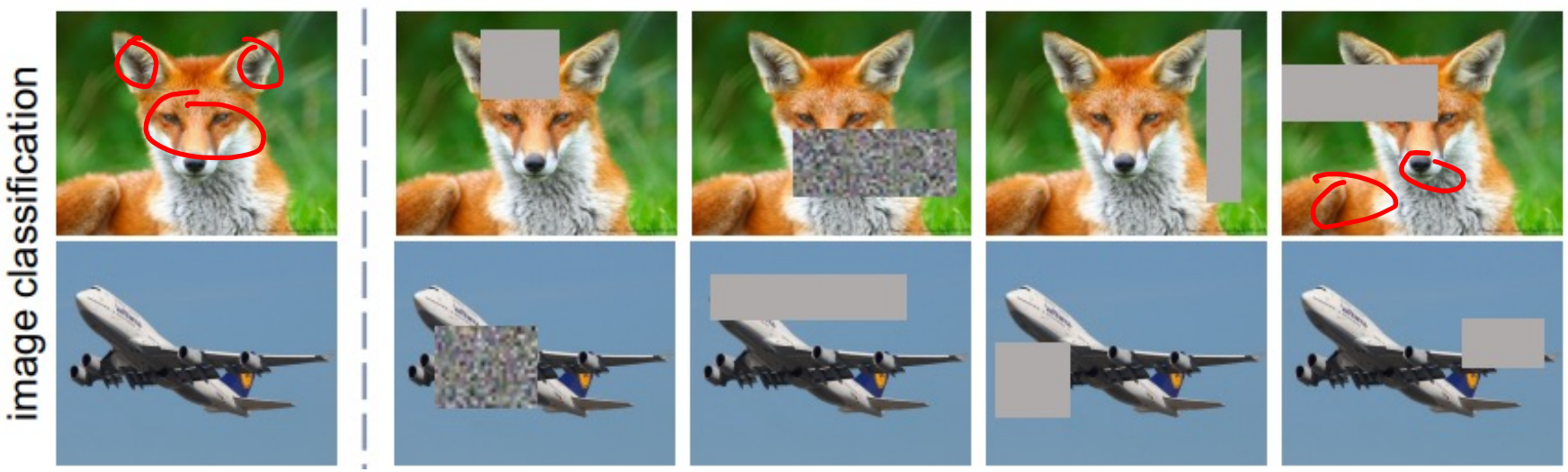
Comparison of results for different color spaces on CIFAR-10 with simple CNN

Color Space	Accuracy	Time
RGB	78.89	26 secs
HSV	78.57	26 secs
YUV	78.89	26 secs
LAB	**80.43**	26 secs
YIQ	78.79	26 secs
XYZ	78.72	26 secs
YPbPr	78.78	26 secs
YCbCr	78.81	26 secs
HED	78.98	26 secs
LCH	78.82	26 secs

Gowda, Shreyank N., and Chun Yuan. "ColorNet: Investigating the importance of color spaces for image classification." *arXiv preprint arXiv:1902.00267* (2019).

Data Augmentation

- Random erasing

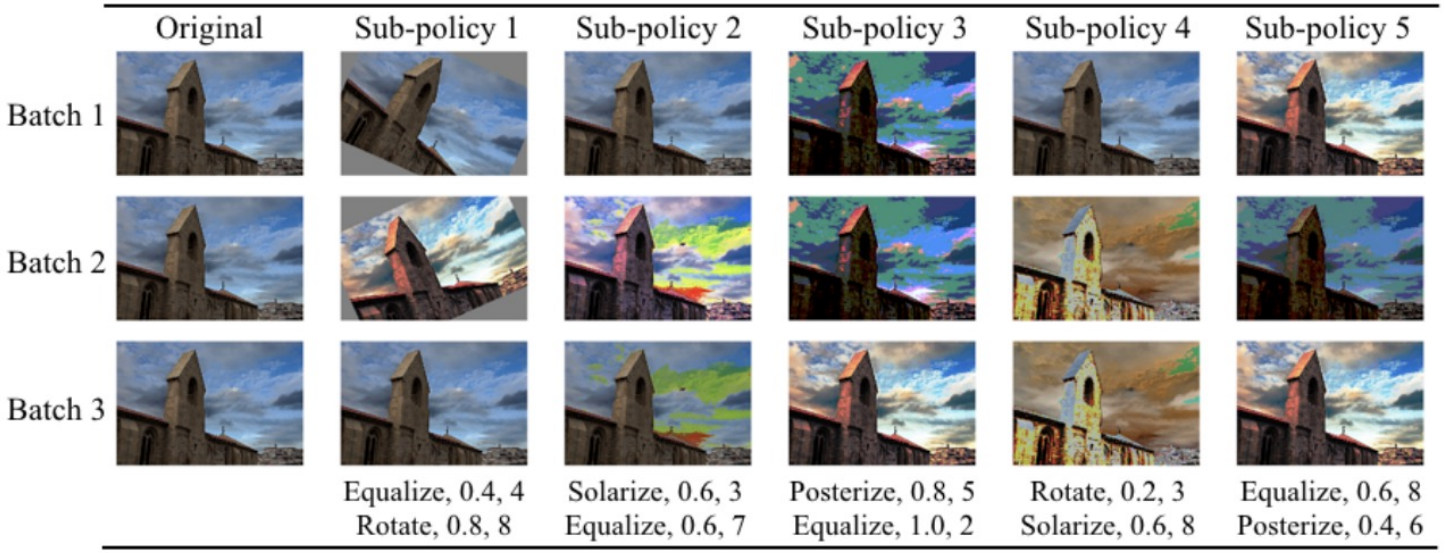


Reduces the risk of over-fitting and makes the model robust to occlusion
Improve the generalization ability of CNNs

Data Augmentation

- Learning Augmentation Policies from Data

Search space of operations: ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness,



Cubuk, Ekin D., et al. "Autoaugment: Learning augmentation policies from data." *arXiv preprint arXiv:1805.09501* (2018).

Data Augmentation

- Cutmix

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6** **(+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** **(+1.1)**

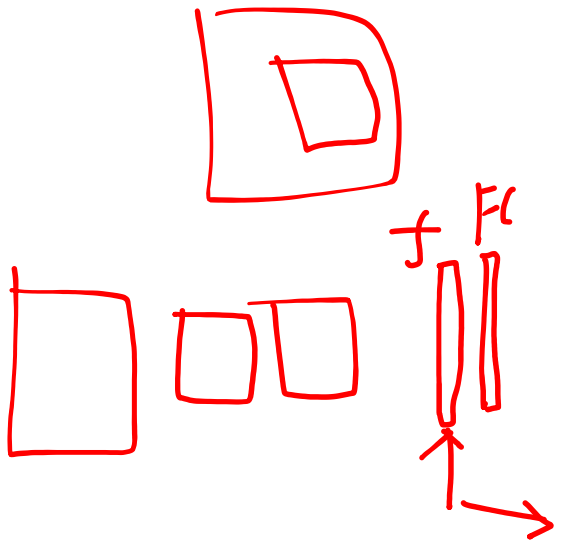
Yun, Sangdoo, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

- More on data augmentation
  - (Pytorch) torchvision.transforms: https://pytorch.org/vision/master/transforms.html
  - https://github.com/AgaMiko/data-augmentation-review

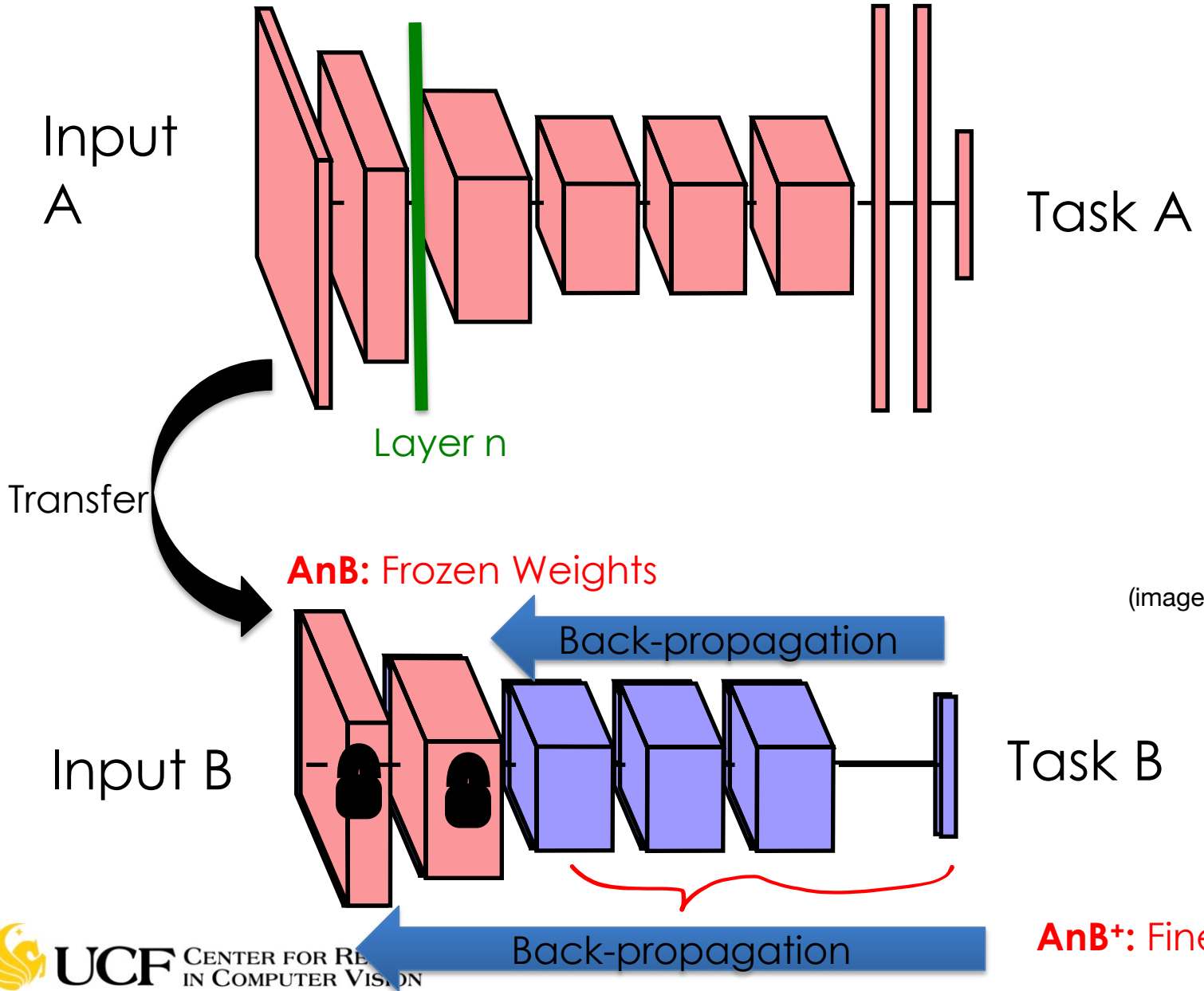
Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
- Weight initialization for CNN

- Two major strategies
  - ConvNet as fixed feature extractor
  - Fine-tuning the ConvNet



Transfer Learning Overview



(image: Aghamirzaie & Salomon)

When and how to fine-tune?

- Suppose we have model A, trained on dataset A
- Q: How do we apply transfer learning to dataset B to create model B?

When and how to fine-tune?

- New dataset is small and similar to original dataset.
  - train a linear classifier on the CNN codes
- New dataset is large and similar to the original dataset
  - fine-tune through the full network
- New dataset is small but very different from the original dataset
  - SVM classifier from activations somewhere earlier in the network
- New dataset is large and very different from the original dataset
  - fine-tune through the entire network

Dataset size	Dataset similarity	Recommendation
Large	Very different	Train model B from scratch, initialize weights from model A
Large	Similar	OK to fine-tune (less likely to overfit)
Small	Very different	Train classifier using the earlier layers (later layers won't help much)
Small	Similar	Don't fine-tune (overfitting). Train a linear classifier

https://cs231n.github.io/transfer-learning/

Examples

- https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Transfer Learning for Medical Imaging

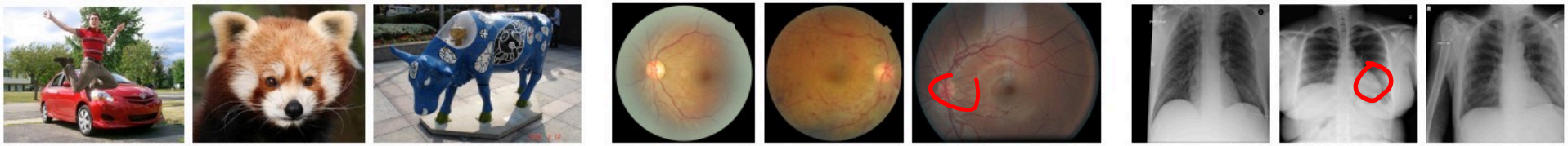


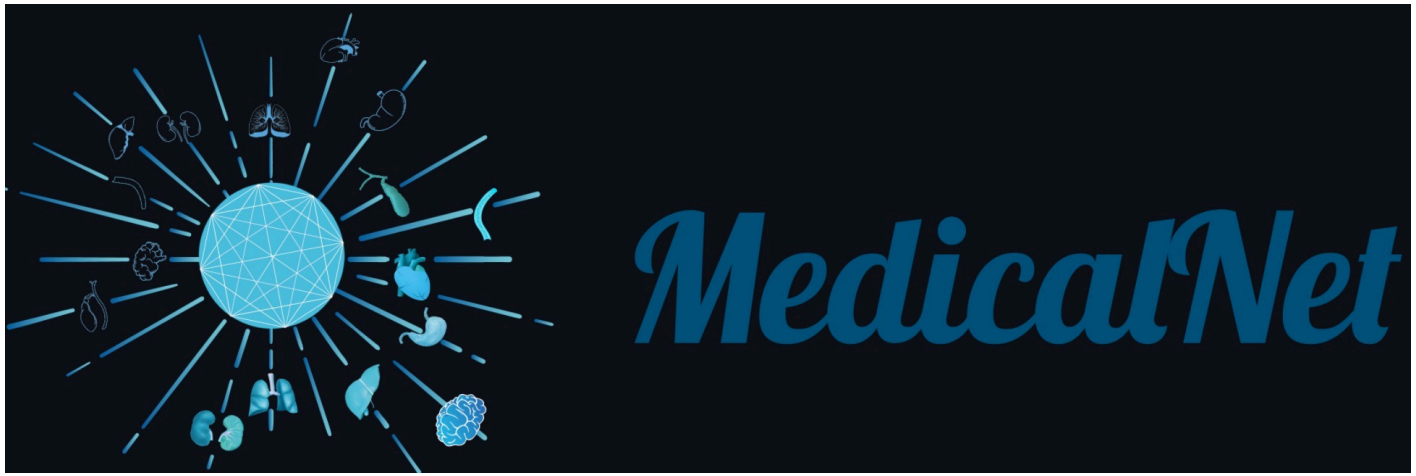
Figure 1: Example images from the IMAGENET, the *retinal fundus photographs*, and the CHEXPERT datasets, respectively. The fundus photographs and chest x-rays have much higher resolution than the IMAGENET images, and are classified by looking for small local variations in tissue.

- Proposed Solution
  - **Transfer the scale (range) of the weights instead of the weights themselves**. This offers feature-independent benefits that facilitate convergence. In particular, they initialized the weights from a normal distribution $N(\mu;\sigma)$. The mean and the variance of the weight matrix is calculated from the pretrained weights. This calculation was performed for each layer separately.
  - **Use the pretrained weights only from the lowest two layers**. The rest of the network is randomly initialized and fine-tuned for the medical imaging task. This hybrid method has the biggest impact on convergence. To summarize, most of the most meaningful feature representations are learned in the lowest two layers.

Raghu, Maithra, et al. "Transfusion: Understanding transfer learning for medical imaging." arXiv preprint arXiv:1902.07208 (2019).

Transfer Learning for Medical Imaging

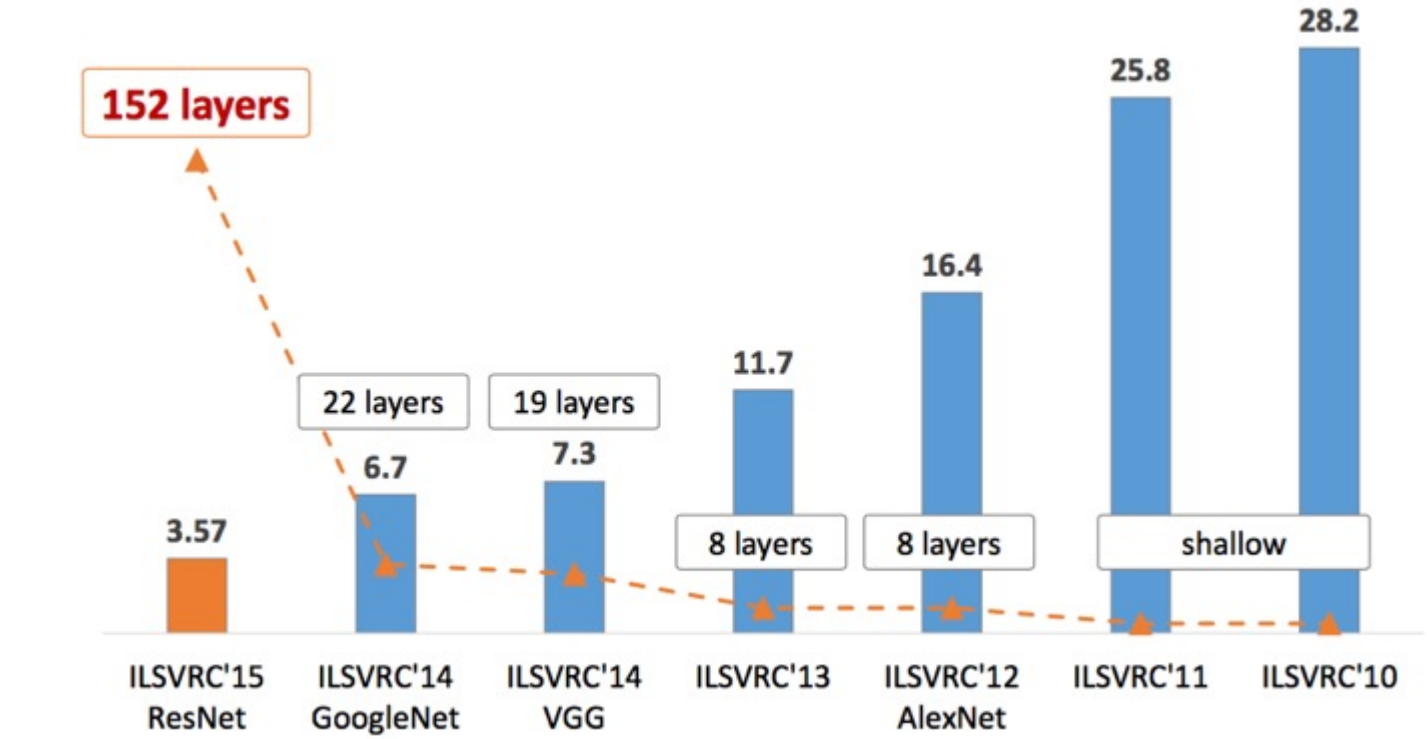
- Med3D: Transfer Learning for 3D Medical Image Analysis



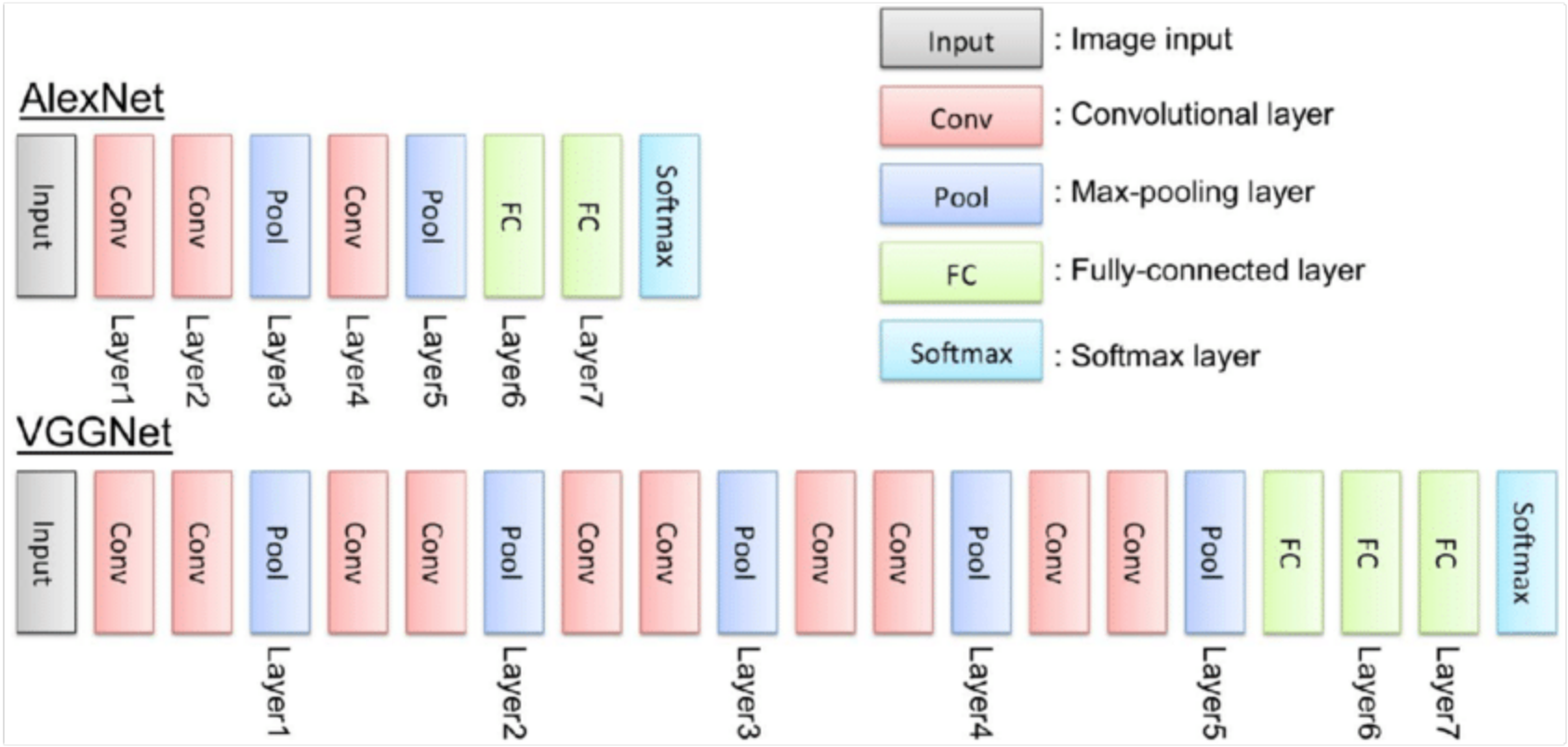
https://github.com/Tencent/MedicalNet

Common CNNs

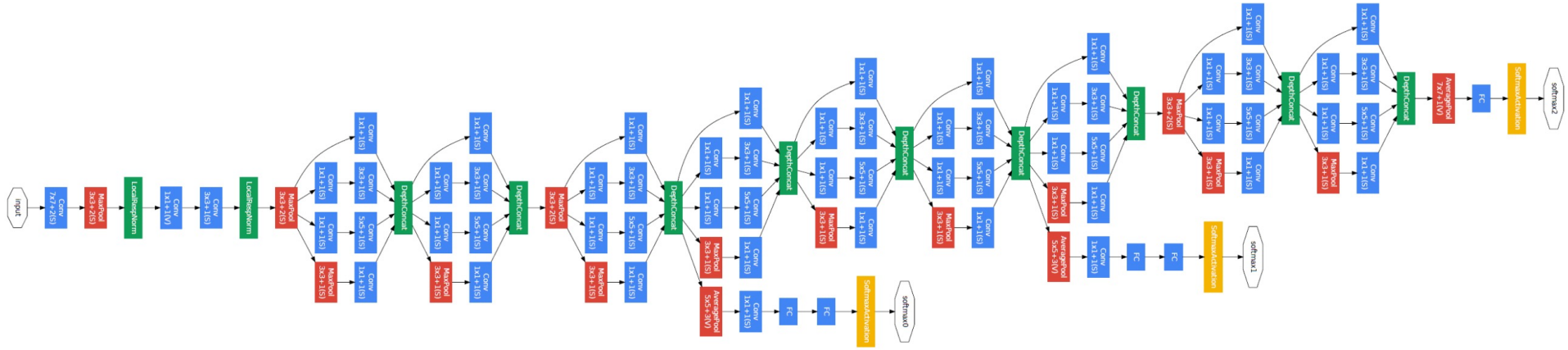
- ImageNet challenge



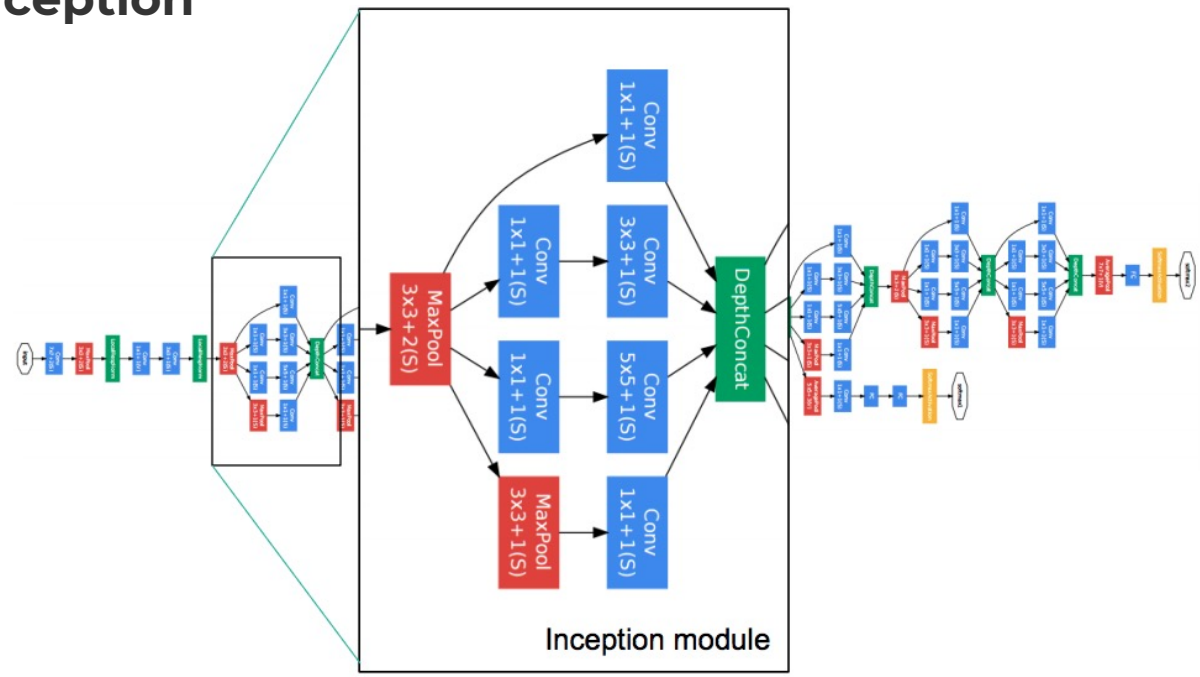
Common CNNs


Input : Image input
Conv : Convolutional layer
Pool : Max-pooling layer
FC : Fully-connected layer
Softmax : Softmax layer
AlexNet
Input
Conv
Layer1
Conv
Layer2
Pool
Layer3
Conv
Layer4
Pool
Layer5
FC
Layer6
FC
Layer7
Softmax
VGGNet
Input
Conv
Conv
Pool
Layer1
Conv
Conv
Pool
Layer2
Conv
Conv
Pool
Layer3
Conv
Conv
Pool
Layer4
Conv
Conv
Pool
Layer5
FC
FC
FC
Layer6
Layer7
Softmax

Common CNNs

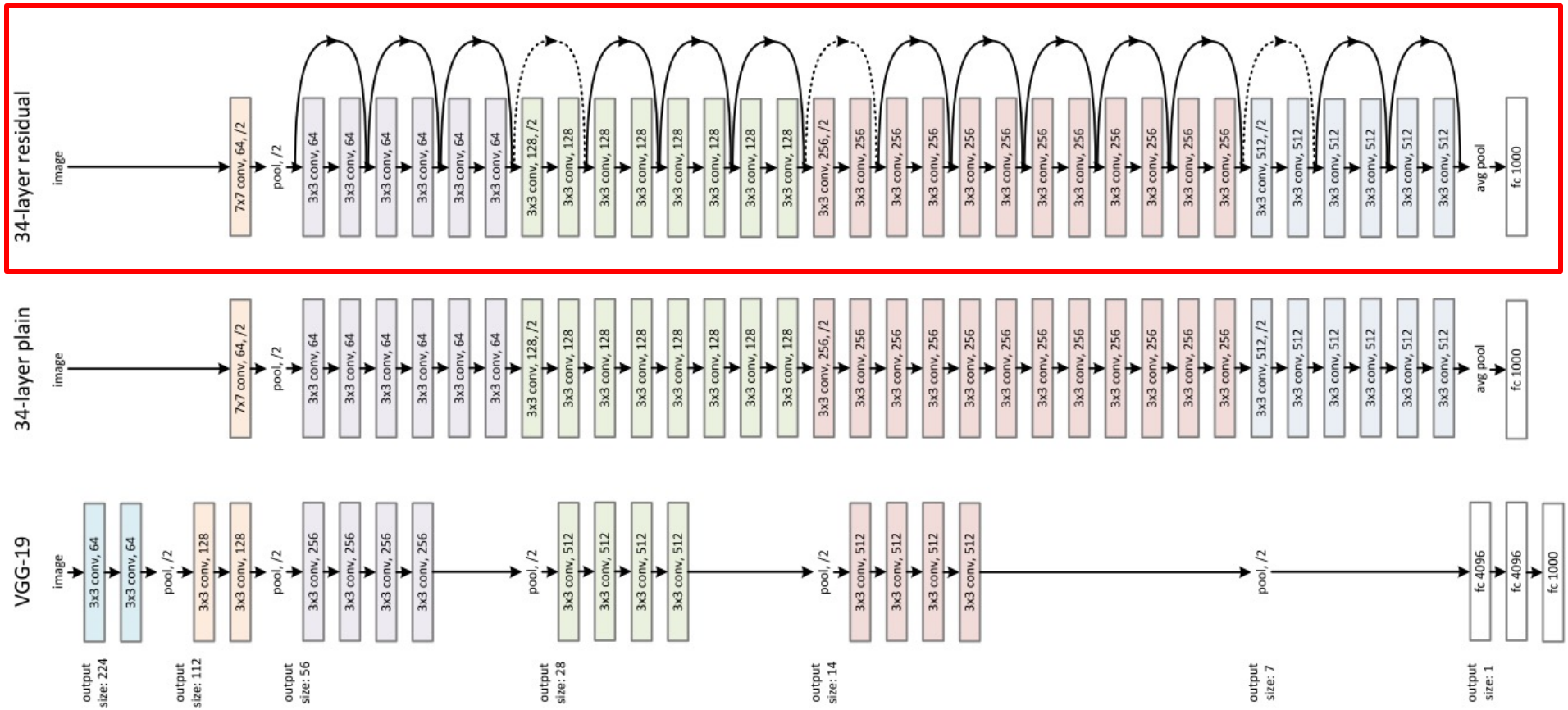


GoogLeNet/Inception



Common CNNs

Residual Networks



Residual Networks

- Deep networks performs worse
  - As we add more layers
- Problem
  - Vanishing gradients
- It models
  - H(x) = F(x) + x
- Skip connections
  - Help in backpropagation

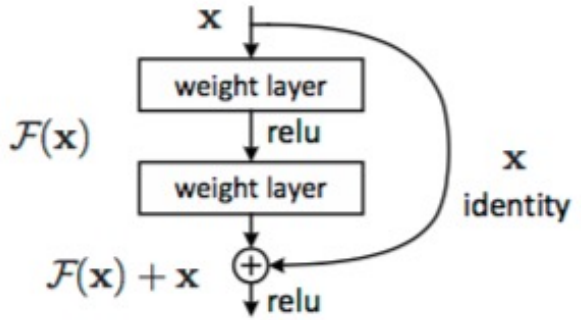


Figure 2. Residual learning: a building block.

He et. al. Deep Residual Learning for Image Recognition, 2015

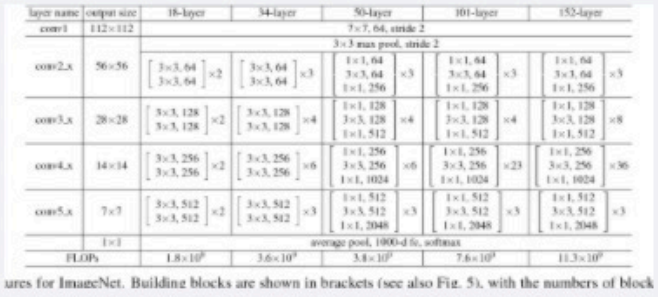
Residual Networks

RESNET

By Pytorch Team

Deep residual networks pre-trained on ImageNet

View on Github › Open on Google Colab ›



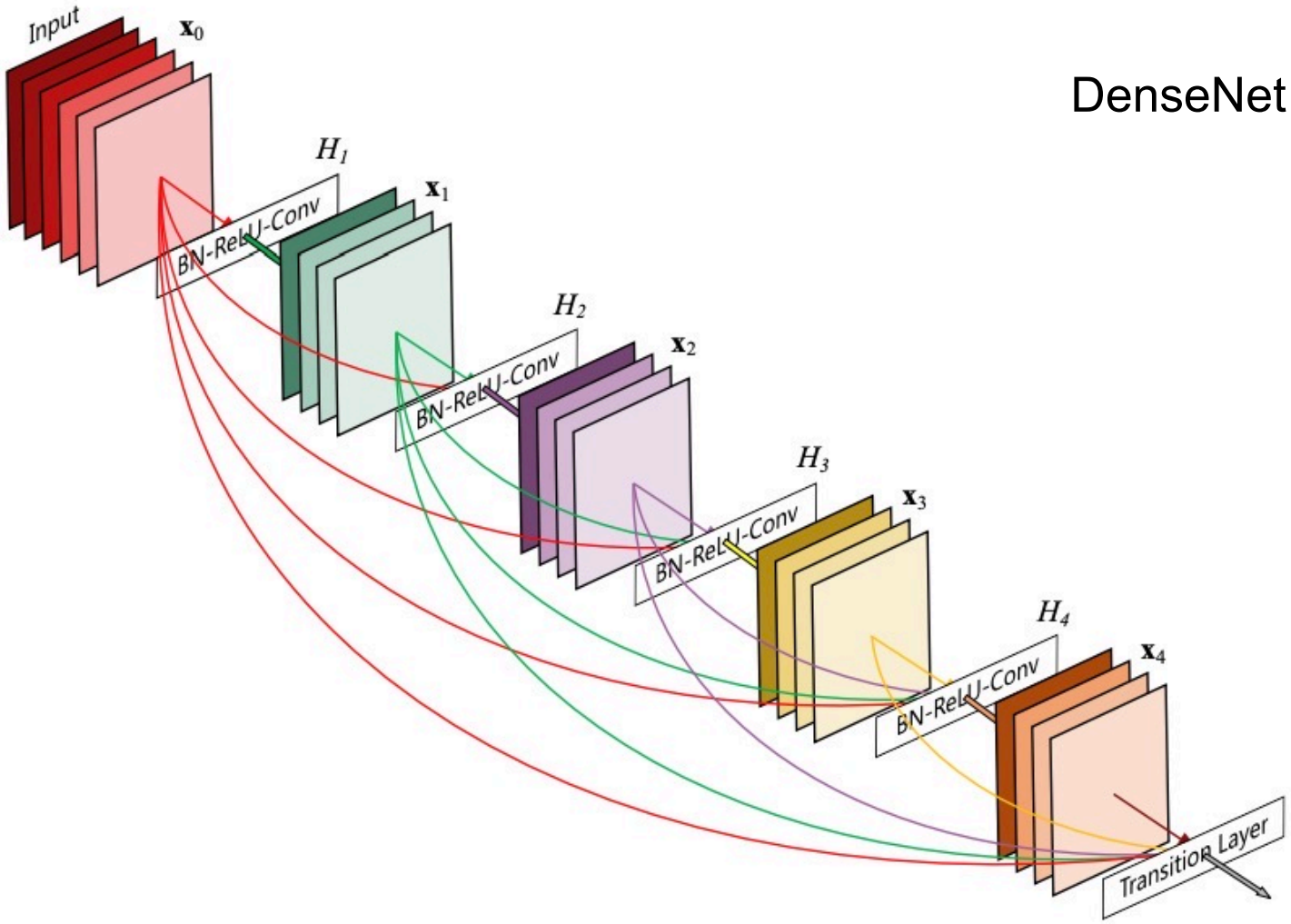
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

```python
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18', pretrained=True)
# or any of these variants
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet34', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet50', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet101', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet152', pretrained=True)
model.eval()
```

https://pytorch.org/hub/pytorch_vision_resnet/

Common CNNs

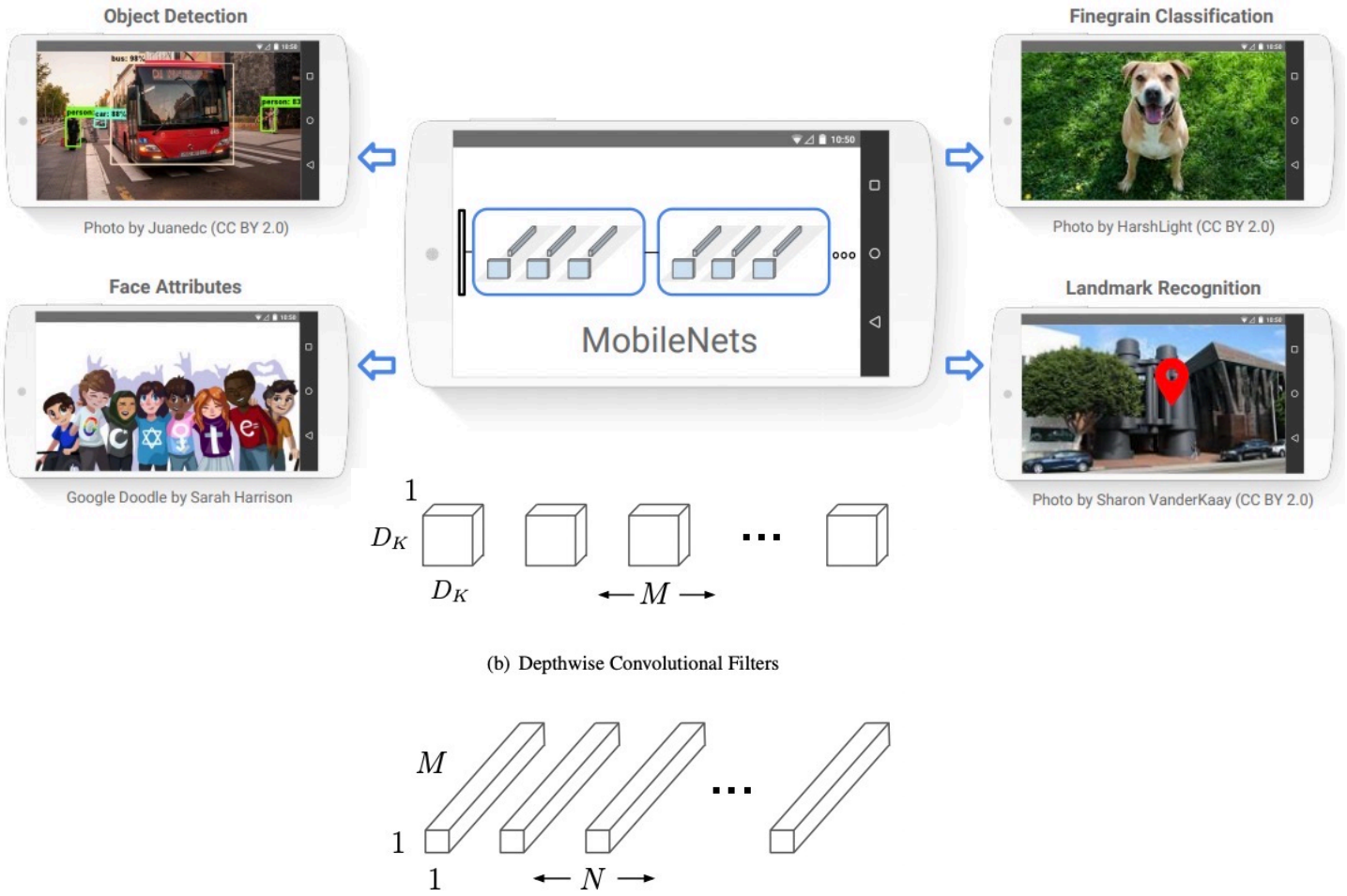


DenseNet

Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.v

Common CNNs

MobileNet

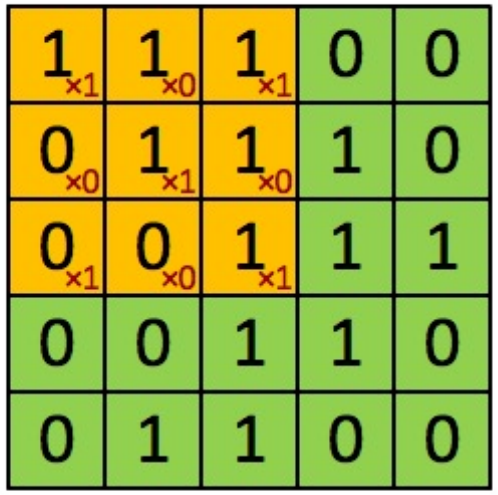


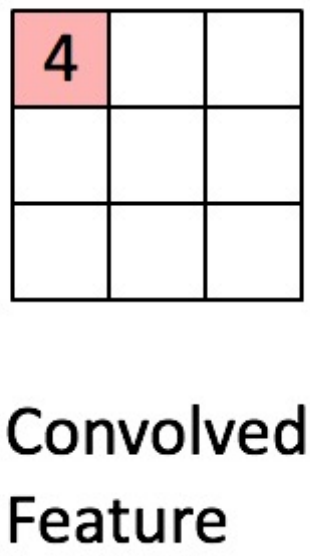
Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

More on MobileNet

Variants of Convolution Operation

- What is the computation of a convolution

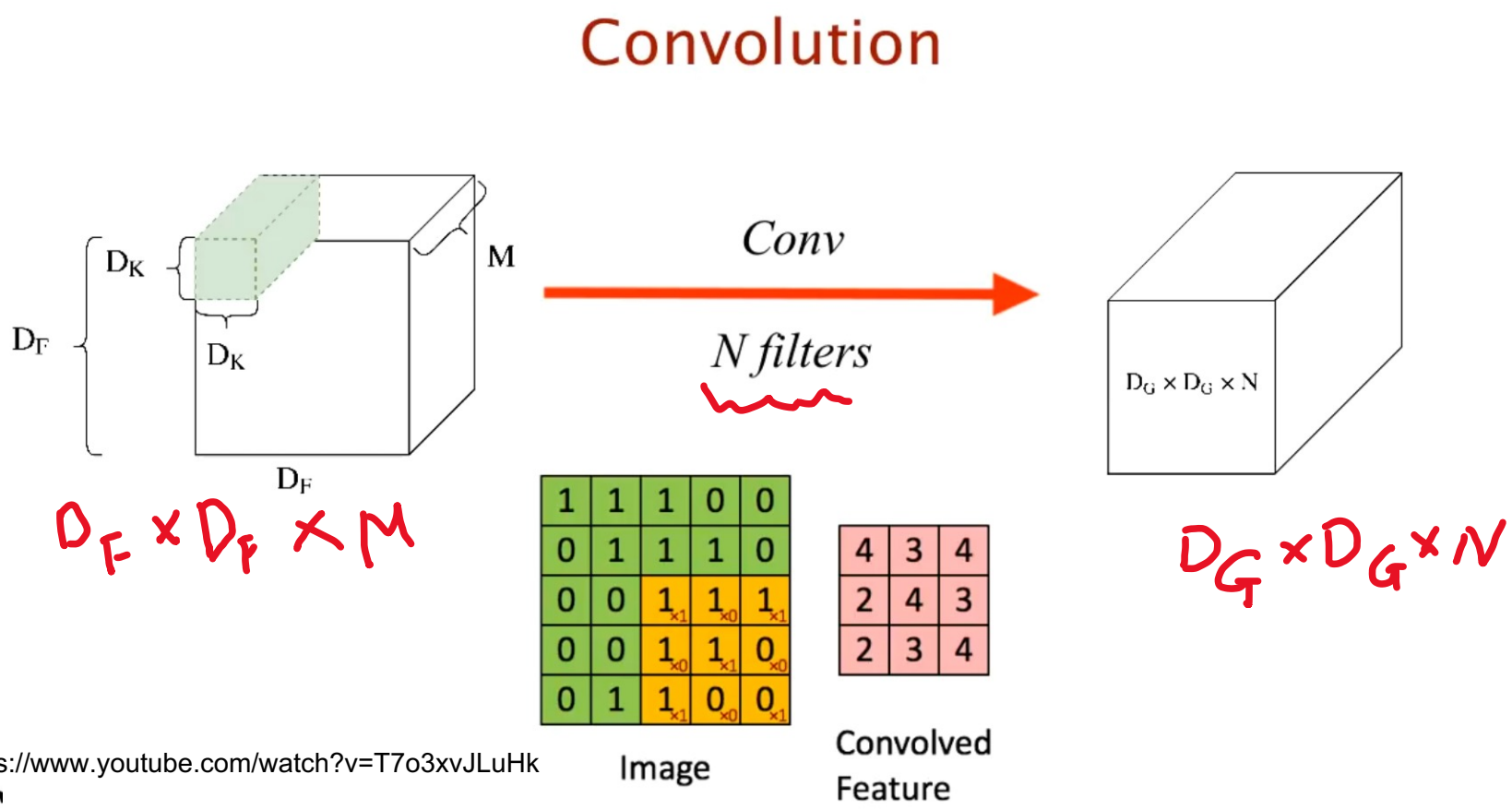




Multiplication is an expensive operation relative to addition

Variants of Convolution Operation

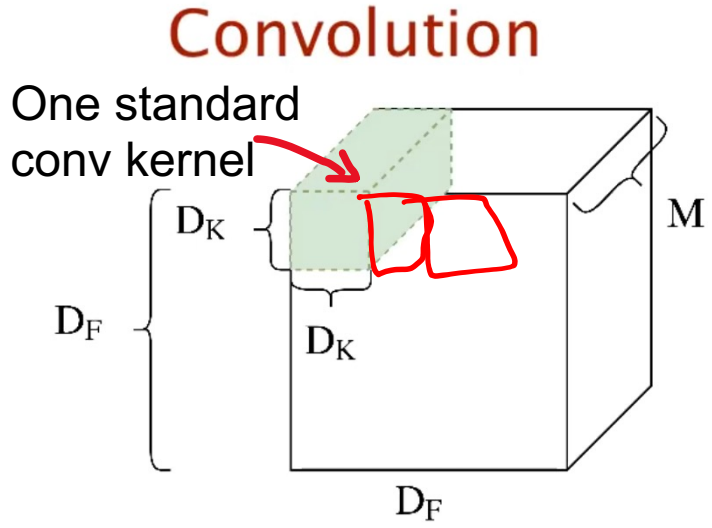
- Determine the number of multiplications



Source: https://www.youtube.com/watch?v=T7o3xvJLuHk

Variants of Convolution Operation

- Determine the number of multiplications



Multiplications one position ?

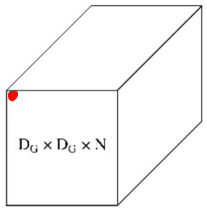
$$D_K^2 \times M$$

Multiplications one kernel over the entire input?

$$D_G^2 \times D_K^2 \times M$$

Multiplications N kernel ?

$$N \times D_G^2 \times D_K^2 \times M$$



Output feature map

$D_G \times D_G \times N$

Variants of Convolution Operation

Depthwise Separable Convolution

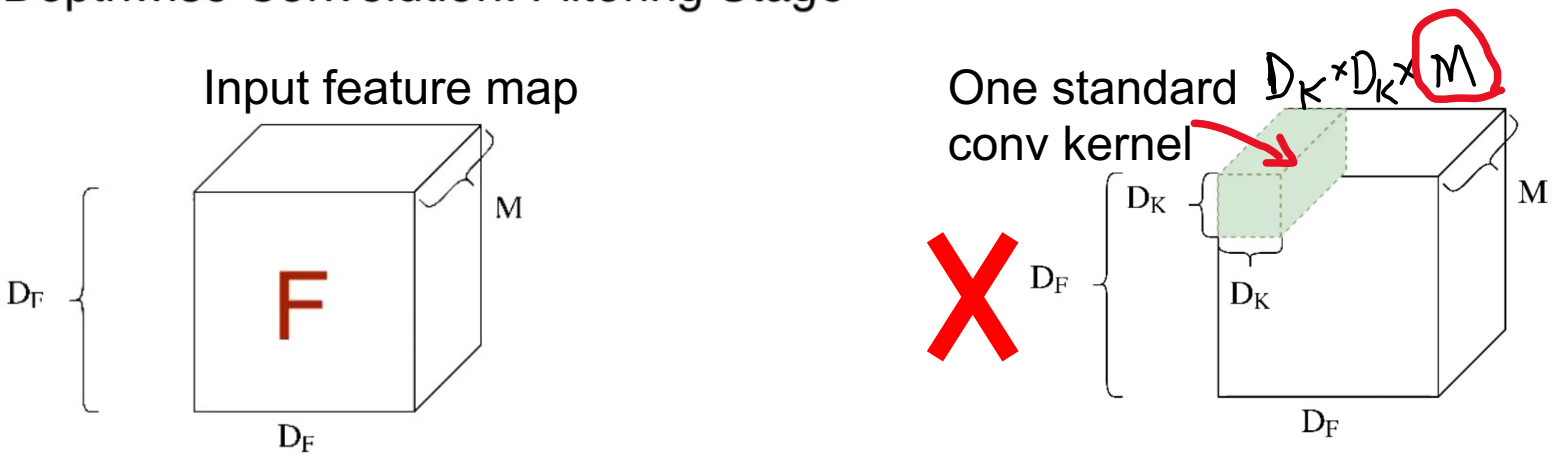
1. Depthwise Convolution: Filtering Stage
2. Pointwise Convolution: Combination Stage

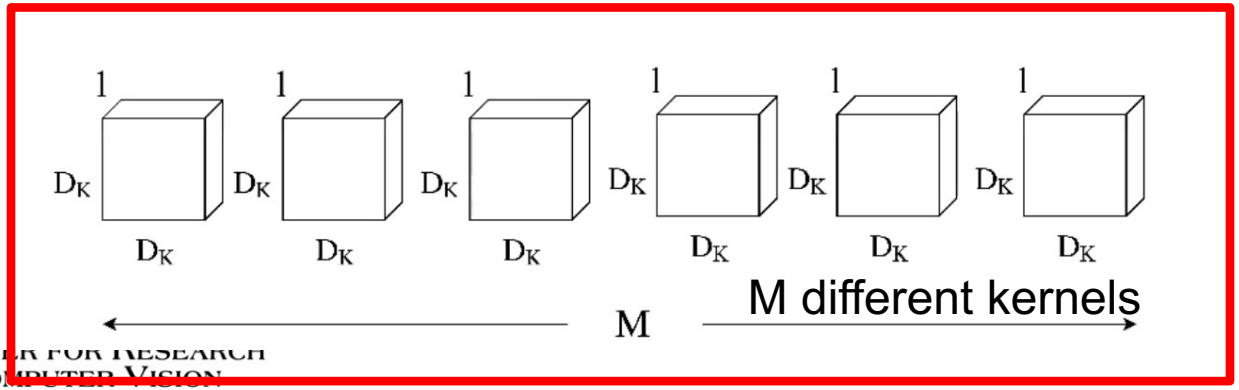
Reduce computation (# of multiplications)

Variants of Convolution Operation

Depthwise Separable Convolution

1. Depthwise Convolution: Filtering Stage



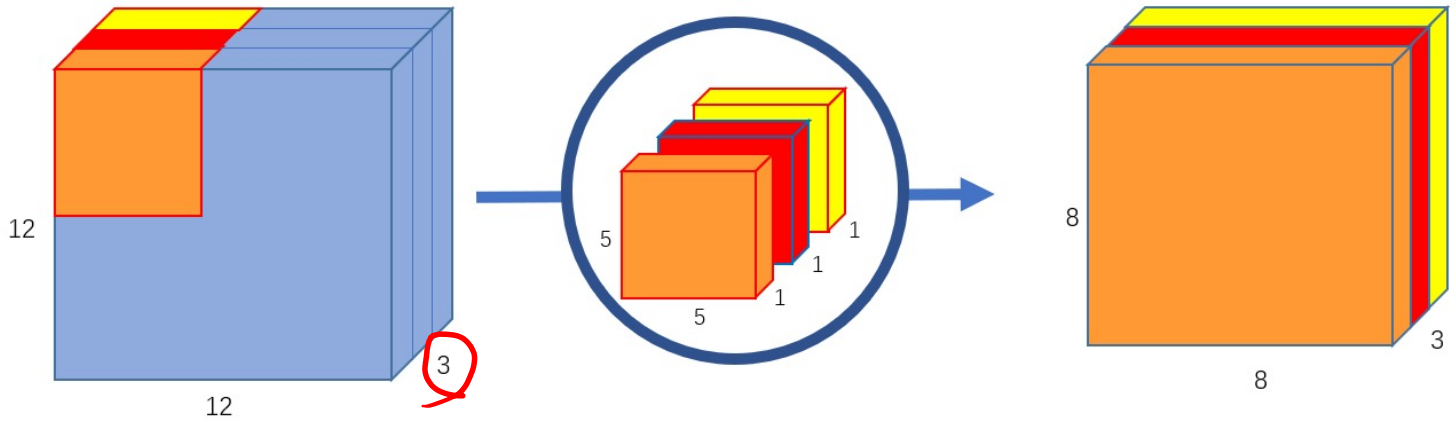


Variants of Convolution Operation

Depthwise Separable Convolution

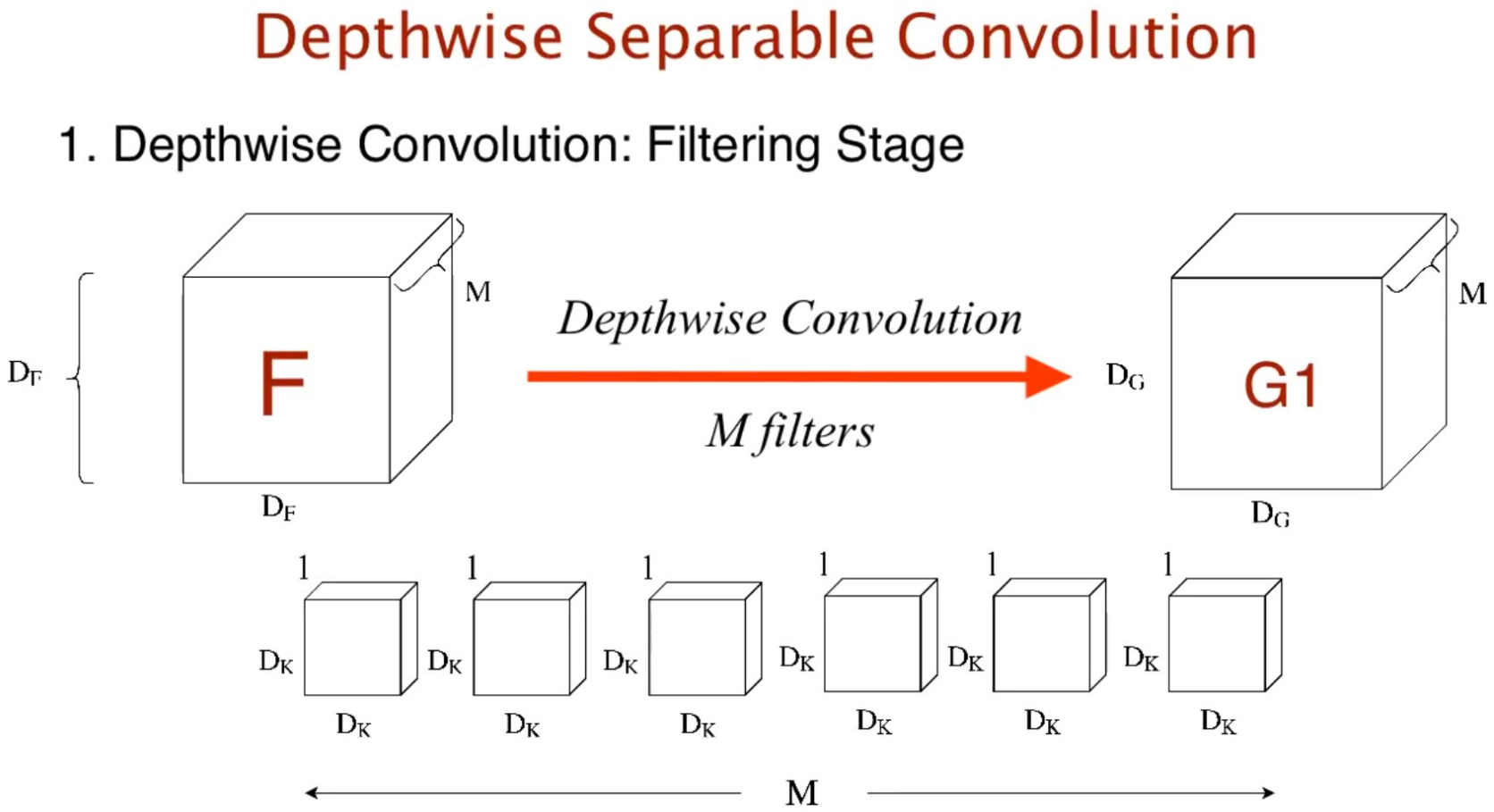
1. Depthwise Convolution: Filtering Stage

Example:



https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Variants of Convolution Operation

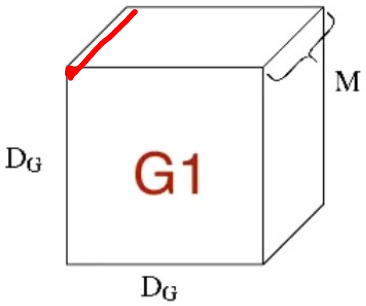


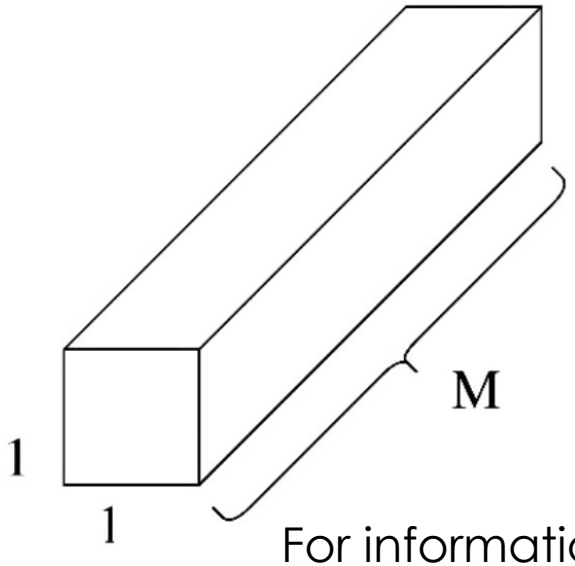
Source: https://www.youtube.com/watch?v=T7o3xvJLuHk

Variants of Convolution Operation

Depthwise Separable Convolution

2. Pointwise Convolution: Filtering Stage



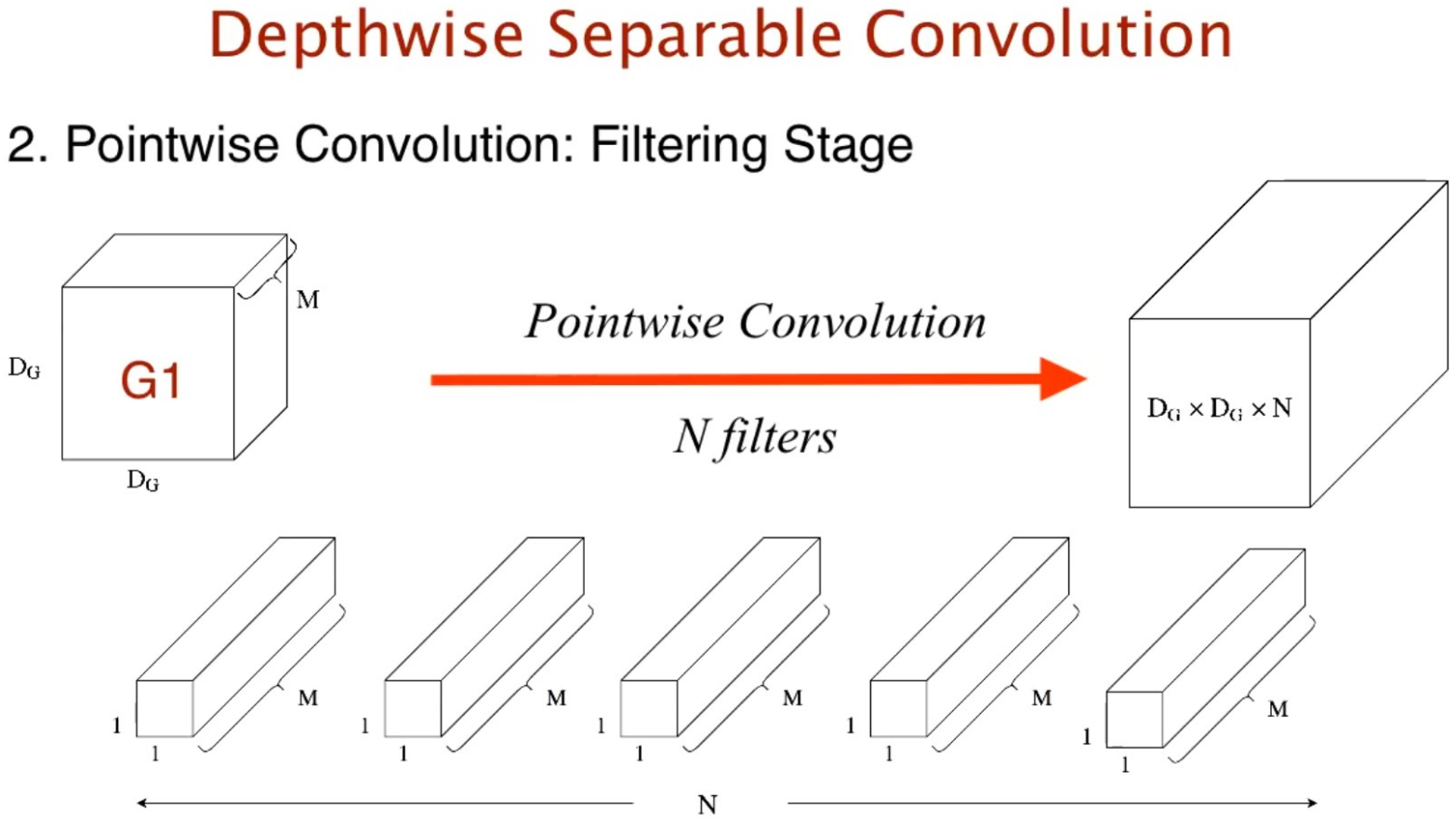


Filter shape

For information aggregation across channels

Source: https://www.youtube.com/watch?v=T7o3xvJLuHk

Variants of Convolution Operation



Source: https://www.youtube.com/watch?v=T7o3xvJLuHk

Variants of Convolution Operation

Mults once = D_K^2

Mults 1 Channel = $D_G^2 \times D_K^2$

DC Mults = $M \times D_G^2 \times D_K^2$

Mults once = M

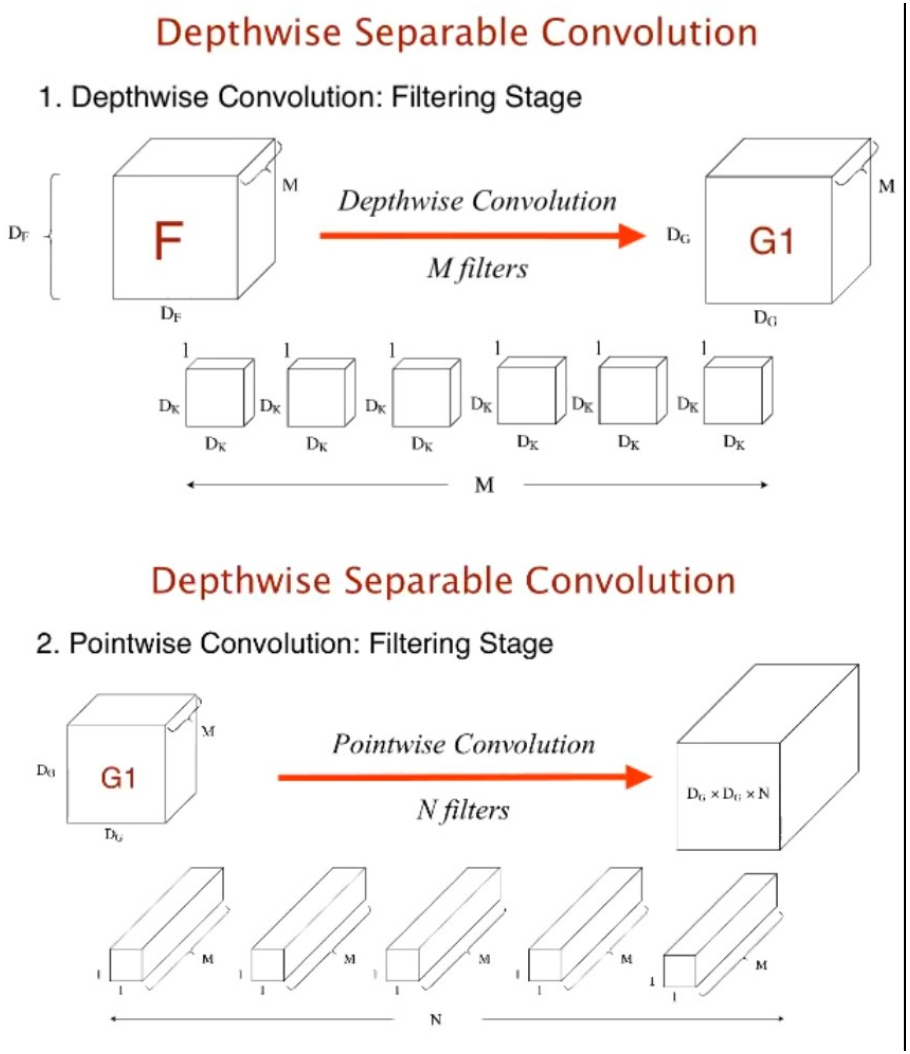
Mults 1 Kernel = $D_G \times D_G \times M$

PC Mults = $N \times D_G \times D_G \times M$

Total = DC Mults + PC Mults

$$M \times D_G^2 \times D_K^2 + N \times D_G^2 \times M$$

$$M \times D_G^2 (D_K^2 + N)$$



Variants of Convolution Operation

Comparison Standard Vs. Depthwise

$$\frac{No.Mults\ in\ Depthwise\ Separable\ Conv}{No.Mults\ in\ Standard\ Conv} = \frac{M \times D_G^2\ (D_K^2 + N)}{N \times D_G \times D_G \times D_K \times D_K \times M}$$

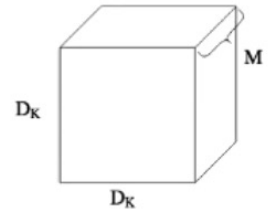
$$\frac{No.\ Mults\ in\ Depthwise\ Separable\ Conv}{No.\ Mults\ in\ Standard\ Conv} = \frac{D_K^2 + N}{(D_K^2 \times N)} = \frac{1}{N} + \frac{1}{D_K^2}$$

E.g. $N = 1{,}024 \qquad D_K = 3$

$$\frac{No.\ Mults\ in\ Depthwise\ Separable\ Conv}{No.\ Mults\ in\ Standard\ Conv} = \frac{1}{1024} + \frac{1}{3^2} = 0.112$$

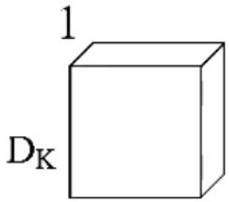
Variants of Convolution Operation

Comparison Standard Vs. Depthwise



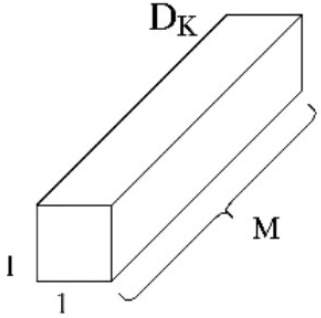
Param 1 Kernel = $D_K^2 \times M$

Param N Kernels = $N \times M \times D_K^2$



Param 1 Kernel = D_K^2

Param M Kernels = $M \times D_K^2$



Param 1 Kernel = M

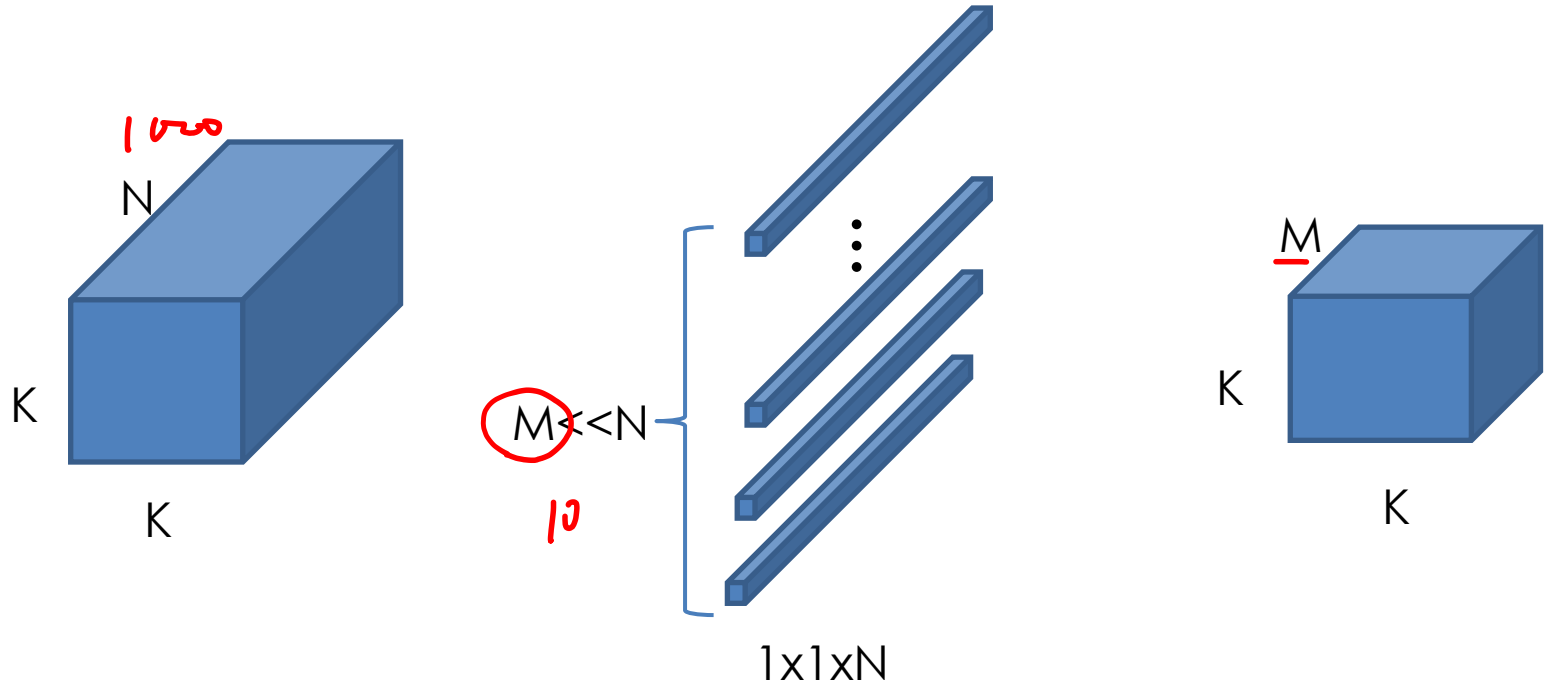
Param N Kernels = $N \times M$

$M(D_K^2 + N)$

$$\frac{No.params\ in\ Depthwise\ Separable\ Conv}{No.params\ in\ Standard\ Conv} = \frac{M \times (D_K^2 + N)}{N \times D_K^2 \times M} = \frac{1}{N} + \frac{1}{D_K^2}$$

One Note on 1x1 conv

- Often used for channel reduction



MobileNet

Depthwise Separable Convolution

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

MobileNet Family

- MobileNet V1
  Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

- MobileNet V2
  Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

- MobileNet V3
  Howard, Andrew, et al. "Searching for mobilenetv3." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Common network models

- Pytorch

```
import torchvision.models as models
resnet18 = models.resnet18()
alexnet = models.alexnet()
vgg16 = models.vgg16()
squeezenet = models.squeezenet1_0()
densenet = models.densenet161()
inception = models.inception_v3()
googlenet = models.googlenet()
shufflenet = models.shufflenet_v2_x1_0()
mobilenet = models.mobilenet_v2()
resnext50_32x4d = models.resnext50_32x4d()
wide_resnet50_2 = models.wide_resnet50_2()
mnasnet = models.mnasnet1_0()
```

TORCHVISION.MODELS

The models subpackage contains definitions of models for addressing differ
semantic segmentation, object detection, instance segmentation, person ke

Classification

The models subpackage contains definitions for the following model archite

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNet v2
- ResNeXt
- Wide ResNet
- MNASNet

https://pytorch.org/docs/stable/torchvision/models.html

Common network models

- Keras

Available models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	F
Xception	88 MB	0.790	0.945	
VGG16	528 MB	0.713	0.901	
VGG19	549 MB	0.713	0.900	
ResNet50	98 MB	0.749	0.921	
ResNet101	171 MB	0.764	0.928	
ResNet152	232 MB	0.766	0.931	
ResNet50V2	98 MB	0.760	0.930	
ResNet101V2	171 MB	0.772	0.938	
ResNet152V2	232 MB	0.780	0.942	
InceptionV3	92 MB	0.779	0.937	
InceptionResNetV2	215 MB	0.803	0.953	
MobileNet	16 MB	0.704	0.895	
MobileNetV2	14 MB	0.713	0.901	
DenseNet121	33 MB	0.750	0.923	
DenseNet169	57 MB	0.762	0.932	
DenseNet201	80 MB	0.773	0.936	
NASNetMobile	23 MB	0.744	0.919	

https://keras.io/api/applications/

Reading Material

- Training a Classifier — PyTorch Tutorials: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- CNNs, Part 1: An Introduction to Convolutional Neural Networks (Keras implementation)
  - https://victorzhou.com/blog/intro-to-cnns-part-1/
- CNNs, Part 2: Training a Convolutional Neural Network (Keras implementation)
  - https://victorzhou.com/blog/intro-to-cnns-part-2/
- **MNIST digit classification (Keras implementation)**
  - https://victorzhou.com/blog/keras-cnn-tutorial/#the-full-code
- Mnist classification (CNN Keras)
  - https://www.kaggle.com/elcaiseri/mnist-simple-cnn-keras-accuracy-0-99-top-1
- Bag of Tricks for Image Classification with Convolutional Neural Networks
  - https://openaccess.thecvf.com/content_CVPR_2019/papers/He_Bag_of_Tricks_for_Image_Classification_with_Convolutional_Neural_Networks_CVPR_2019_paper.pdf

Thank you!

Question?

References and Slide Credits

- Many slides are adapted from the existing teaching or tutorial slides by Hung-yi Lee, Andrew Ng, Alexander Amini, Lex Fridman, Stanford course - CS231n: Convolutional Neural Networks for Visual Recognition, and many others
- Special thanks to Dr. Hung-yi Lee for making his machine learning course slides and materials available
- Alexander Amini, MIT 6.S191 Introduction to Deep Learning: http://introtodeeplearning.com/
  - Youtube videos: https://www.youtube.com/watch?v=5tvmMX8r_OM&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=1
- Lex Fridman, MIT Deep Learning and Artificial Intelligence Lectures: https://deeplearning.mit.edu/ https://www.youtube.com/watch?v=O5xeyoRL95U