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 6: Introduction to Deep Learning (2)

Convolutional Neural Network (CNN)

Widely used in image processing and computer vision

The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

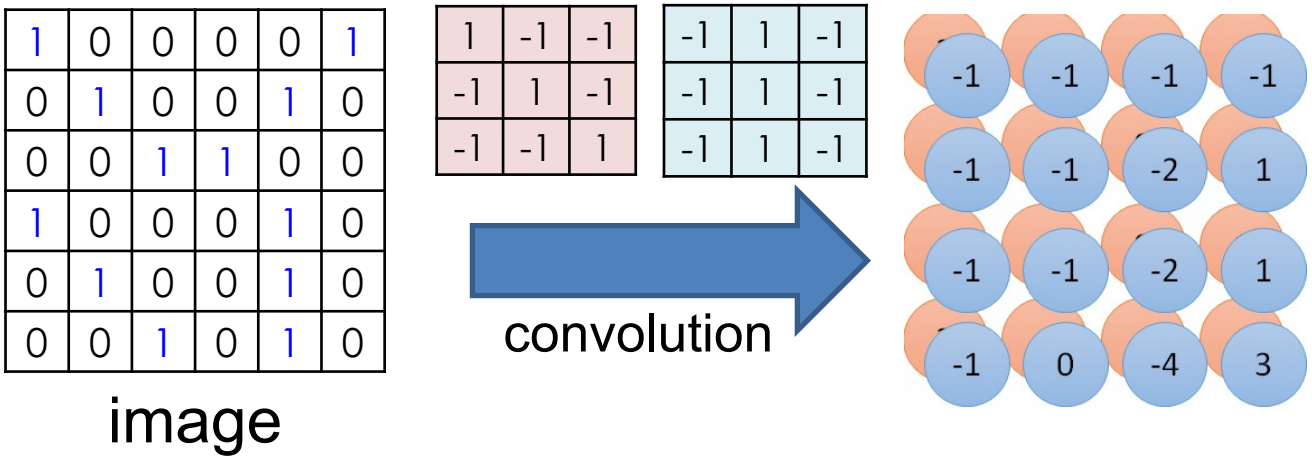
Convolution

Max Pooling

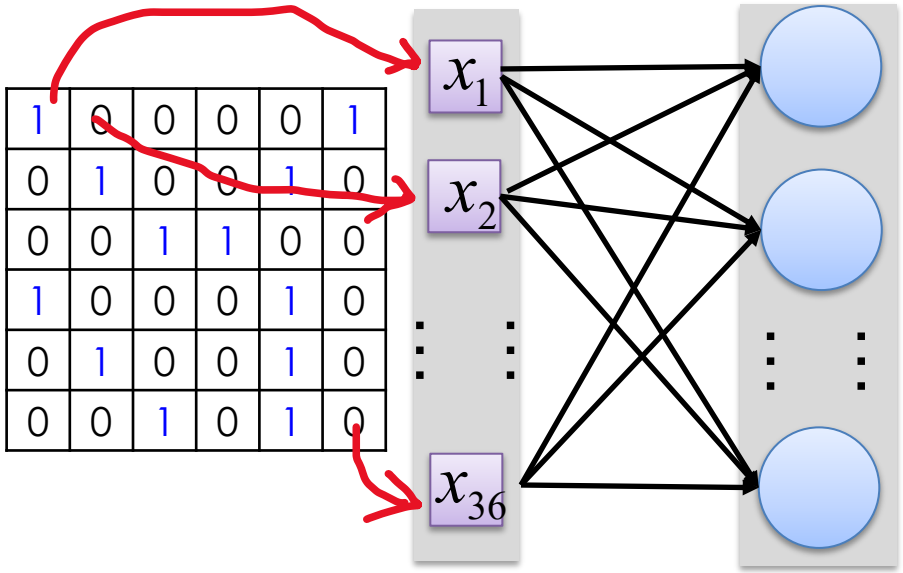
Can repeat many times

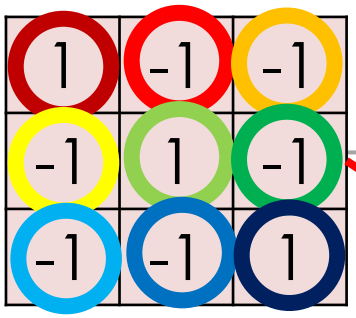
Flatten

Convolution v.s. Fully Connected

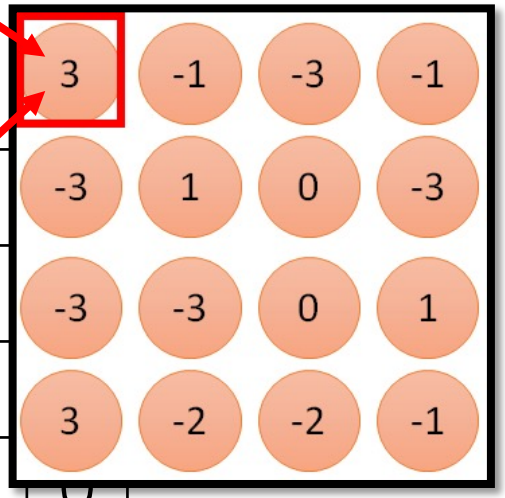


Fully-connected





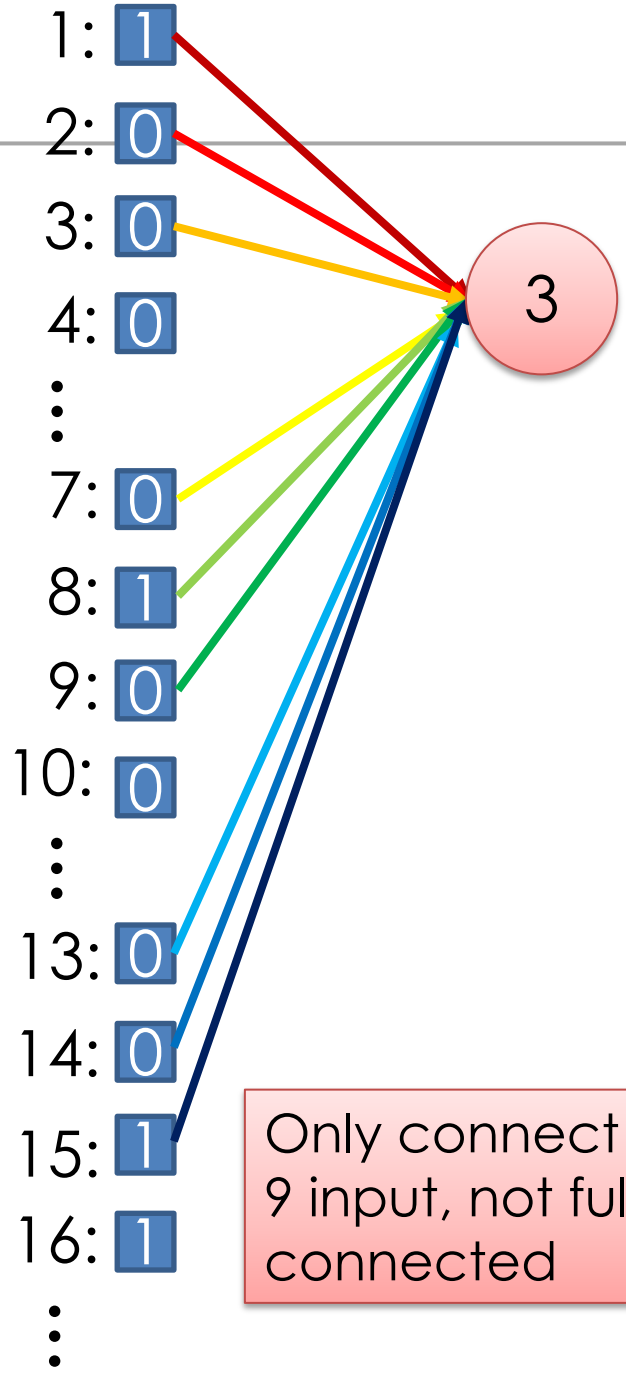
Filter 1



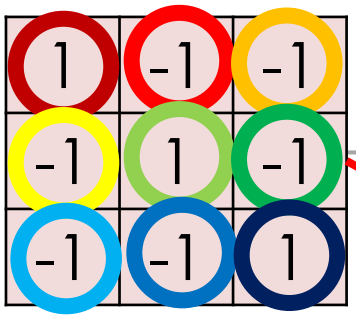
1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

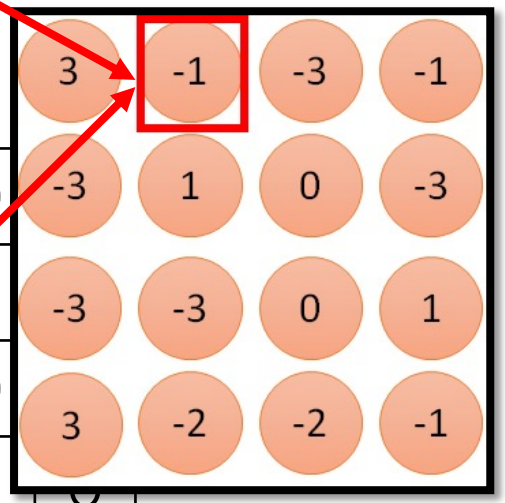
Less parameters!



Only connect to 9 input, not fully connected



Filter 1

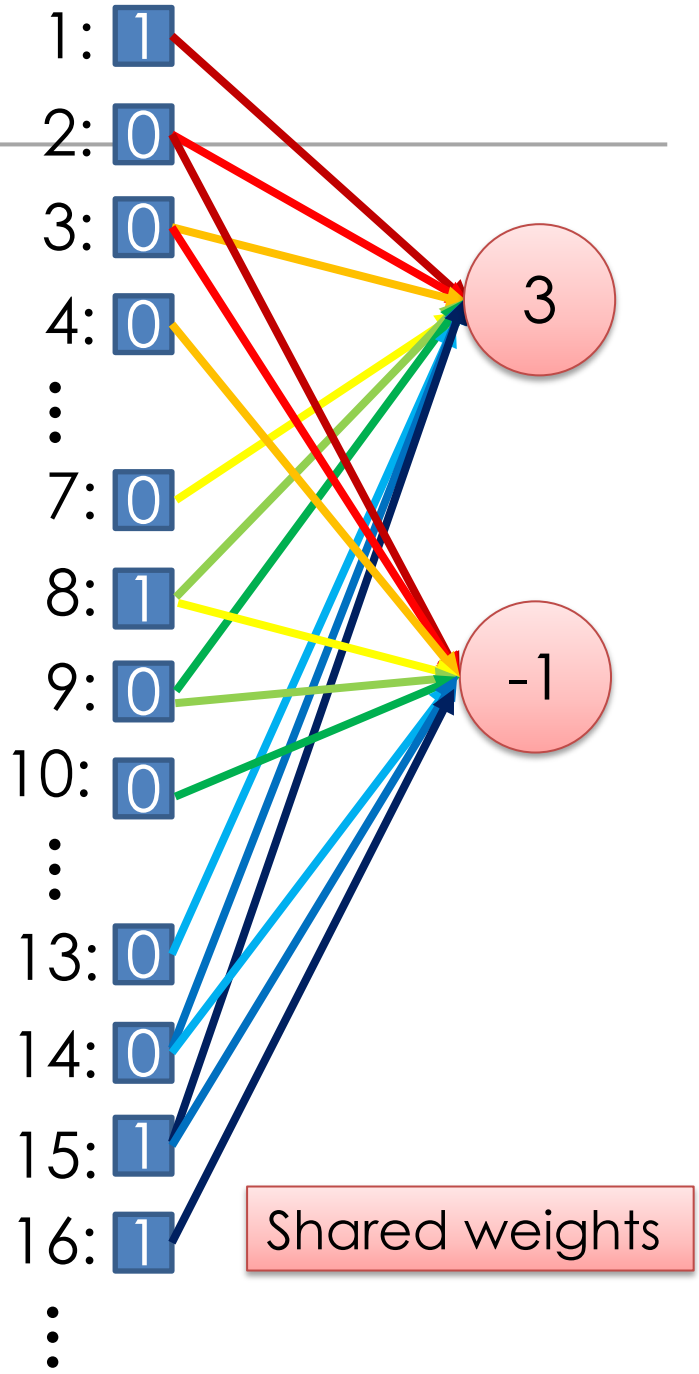


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Less parameters!

Even less parameters!



Shared weights

Question

- What kernel size to use? 3x3, 5x5, …
- How many kernels to use in each layer?

Variants of Convolution Operation

- Dilated convolution [1]
- Depth-wise separable convolution [2]
- Grouped convolution [3]

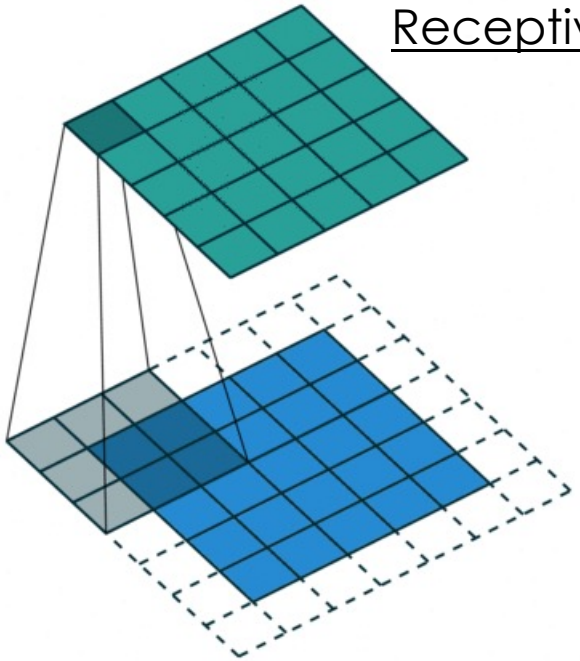
[1] Yu, Fisher, and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions." arXiv preprint arXiv:1511.07122 (2015).
[2] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
[3] Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Variants of Convolution Operation

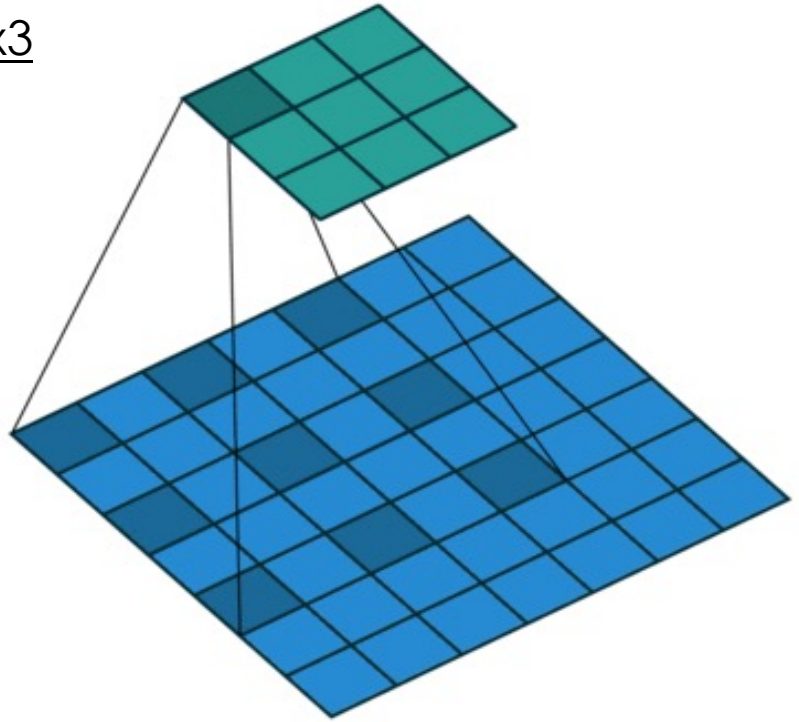
- Dilated/Atrous Convolution

Dilation rate = 2
Receptive filed = 5x5
With only 9 parameters

Receptive filed = 3x3



Normal convolution

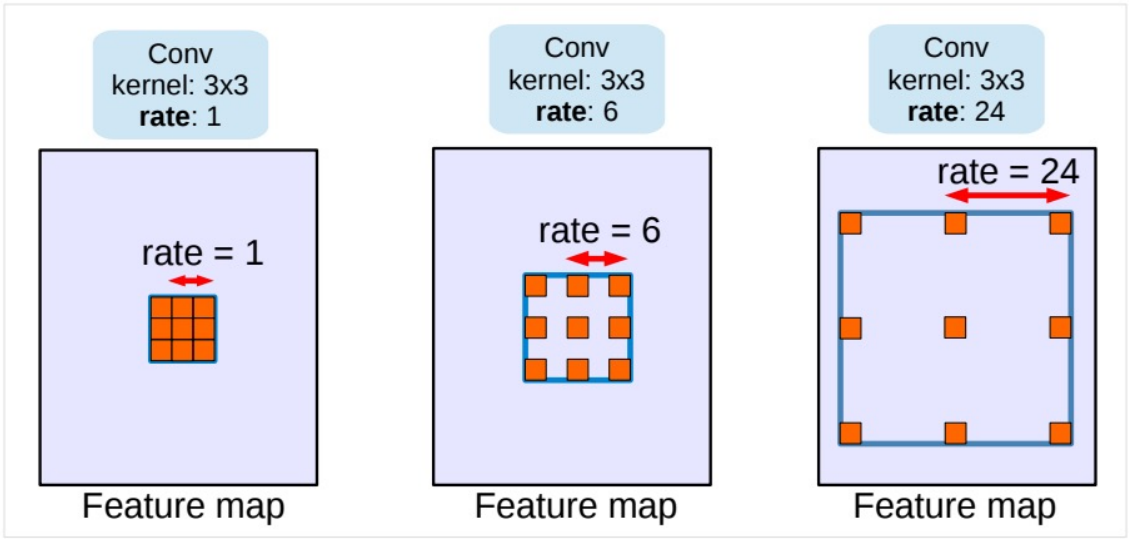


Effective in segmentation task

https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Variants of Convolution Operation

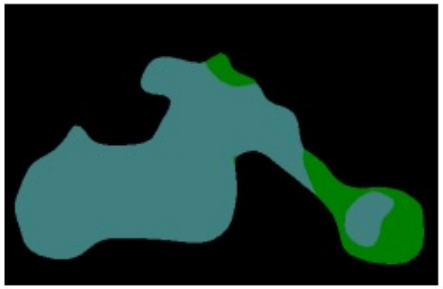
- Dilated Convolution

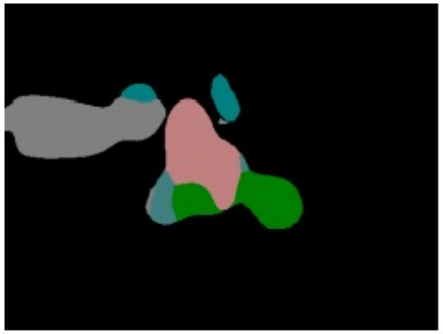


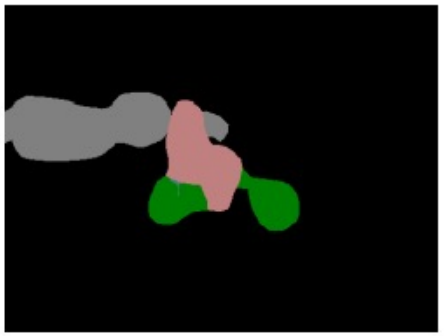
- Large value of atrous rate enlarges the model's field-of-view
- Enabling object encoding at multiple scales

Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." *arXiv preprint arXiv:1706.05587* (2017).

Variants of Convolution Operation







Input

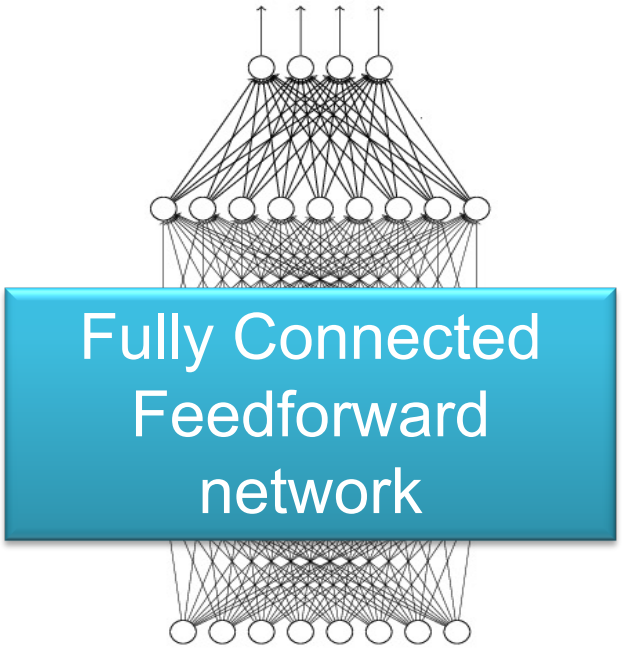
LargeFOV

Atrous Spatial Pyramid Pooling (ASPP)

Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018): 834-848.

The whole CNN

cat dog ……



Convolution

Max Pooling

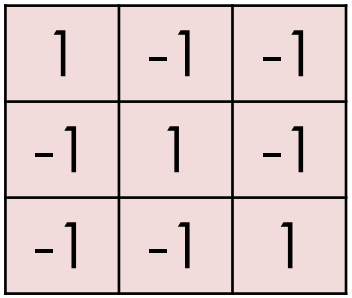
Convolution

Max Pooling

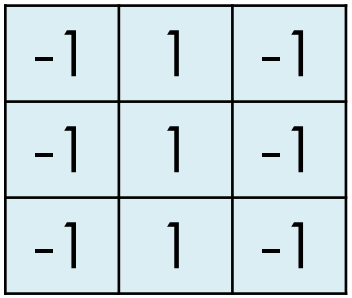
Can repeat many times

Flatten

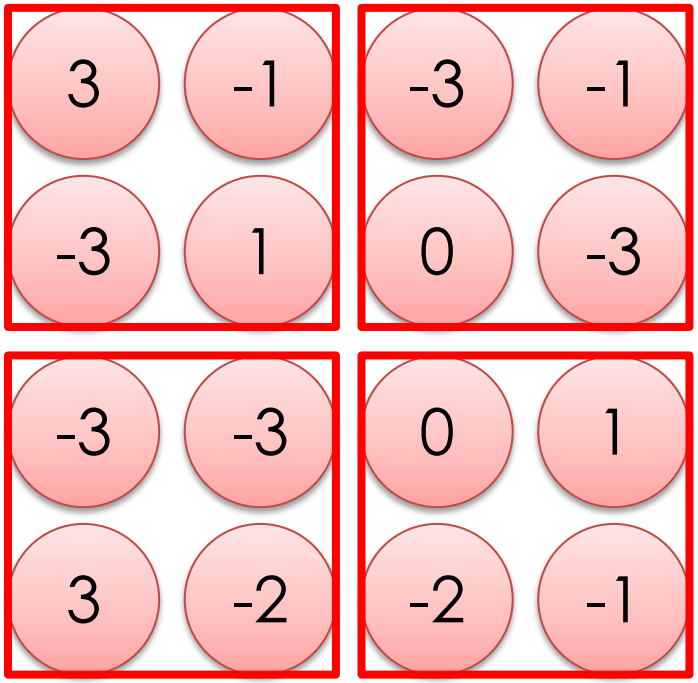
CNN – Max Pooling

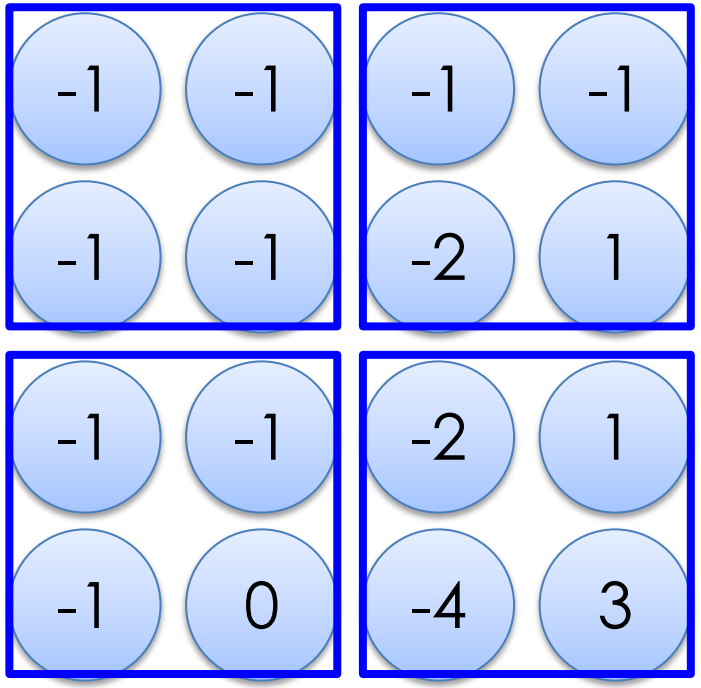


Filter 1

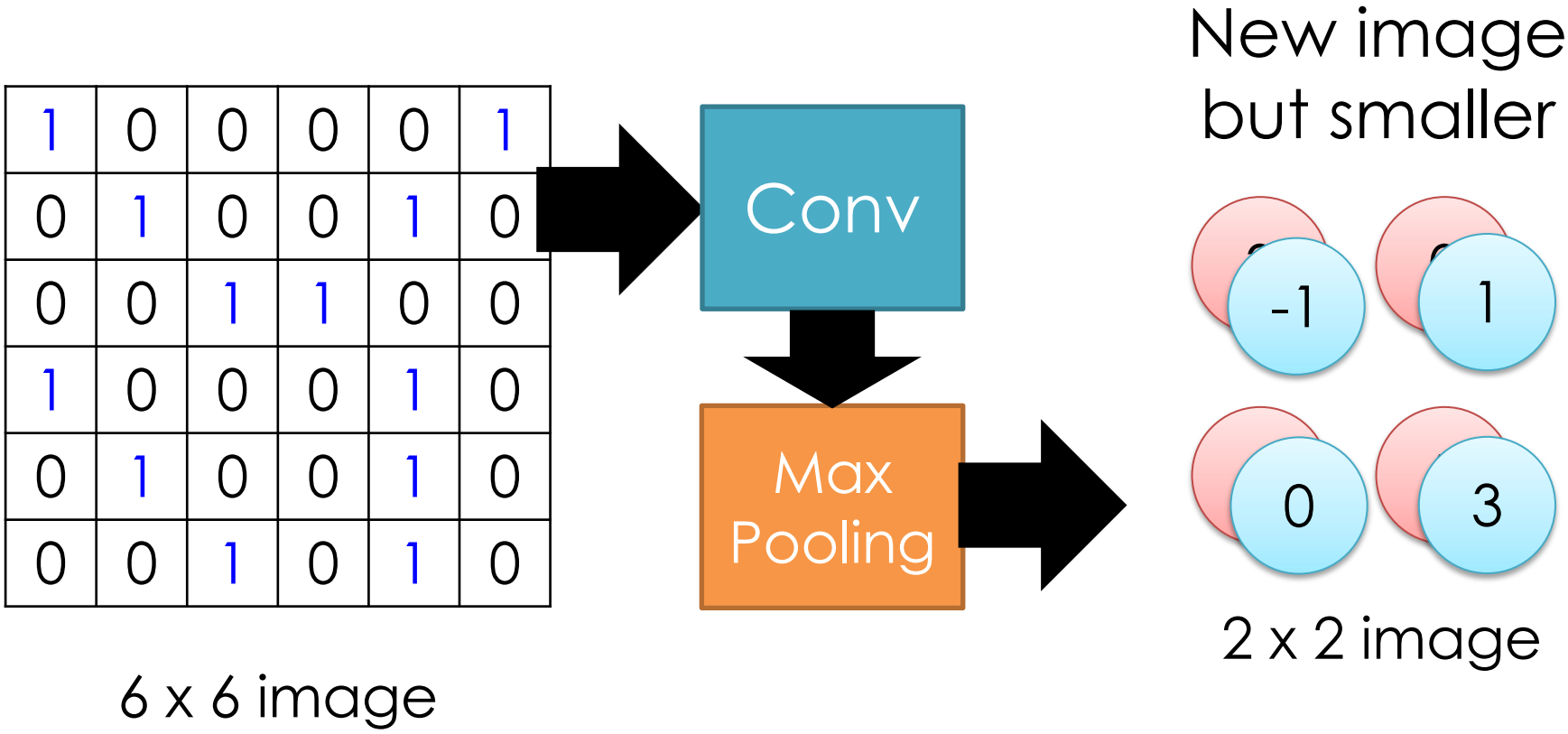


Filter 2

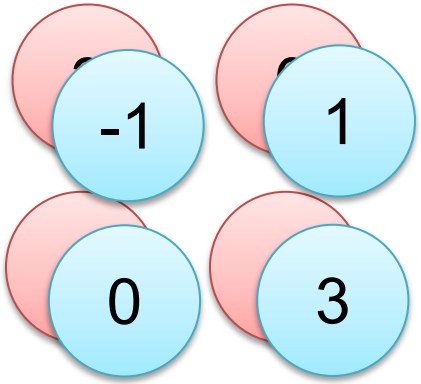




CNN – Max Pooling



The Whole CNN



A new image

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Smaller than the original image

The number of the channel is the number of filters

Why Pooling

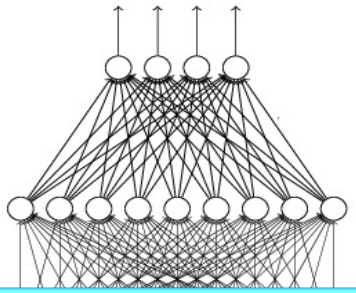
- Pooling is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network

- Another way to reduce feature map size? (convolution with stride, e.g., 2)

- If instead of taking the maximum, using the average will be the average pooling.

Other benefit?

The Whole CNN



cat dog ……

Fully Connected Feedforward network

Flatten

Convolution

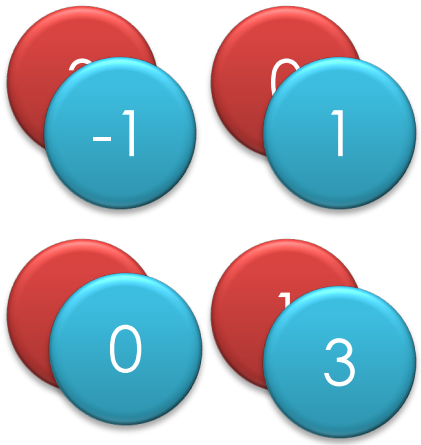
Max Pooling

Convolution

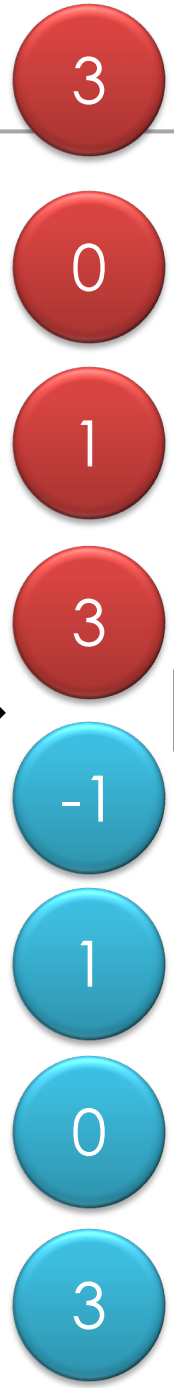
Max Pooling

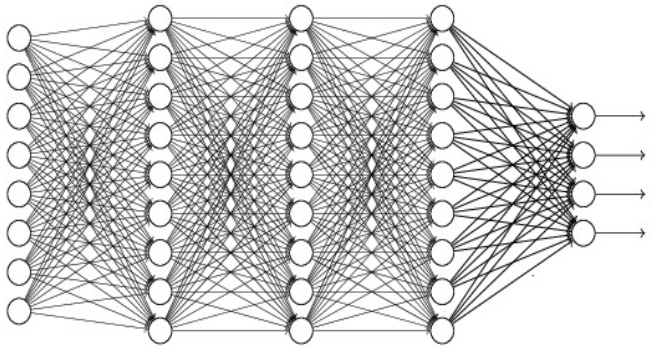
Can repeat many times

Flatten



Flatten

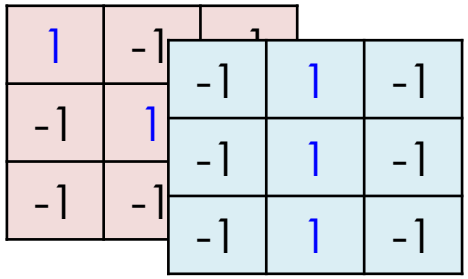




Fully Connected Feedforward network

CNN in Keras

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(1,28,28) ) )
```



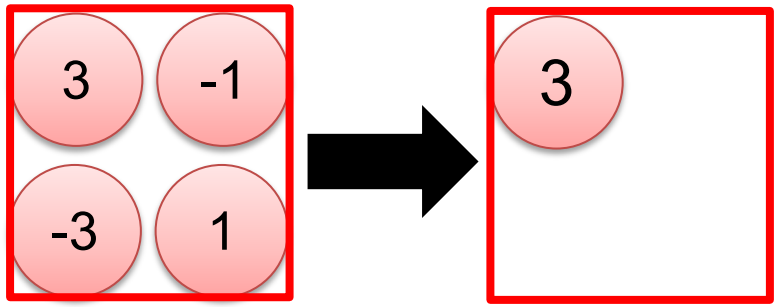
There are 25 3x3 filters.

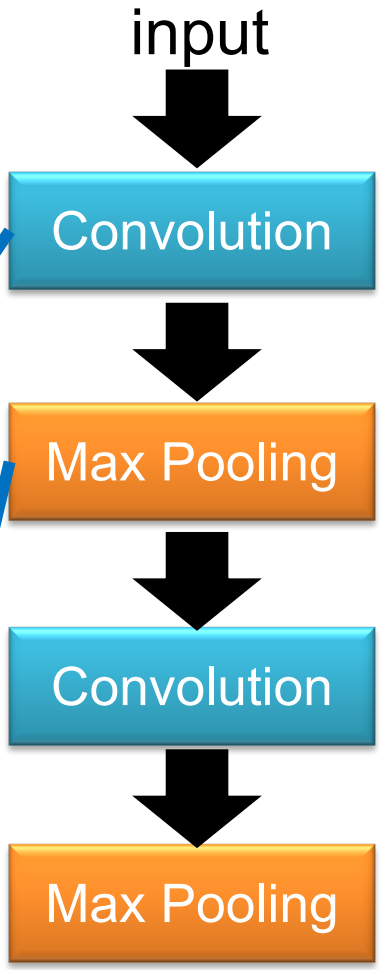
Input_shape = (1 , 28 , 28)

1: grayscale, 3: RGB

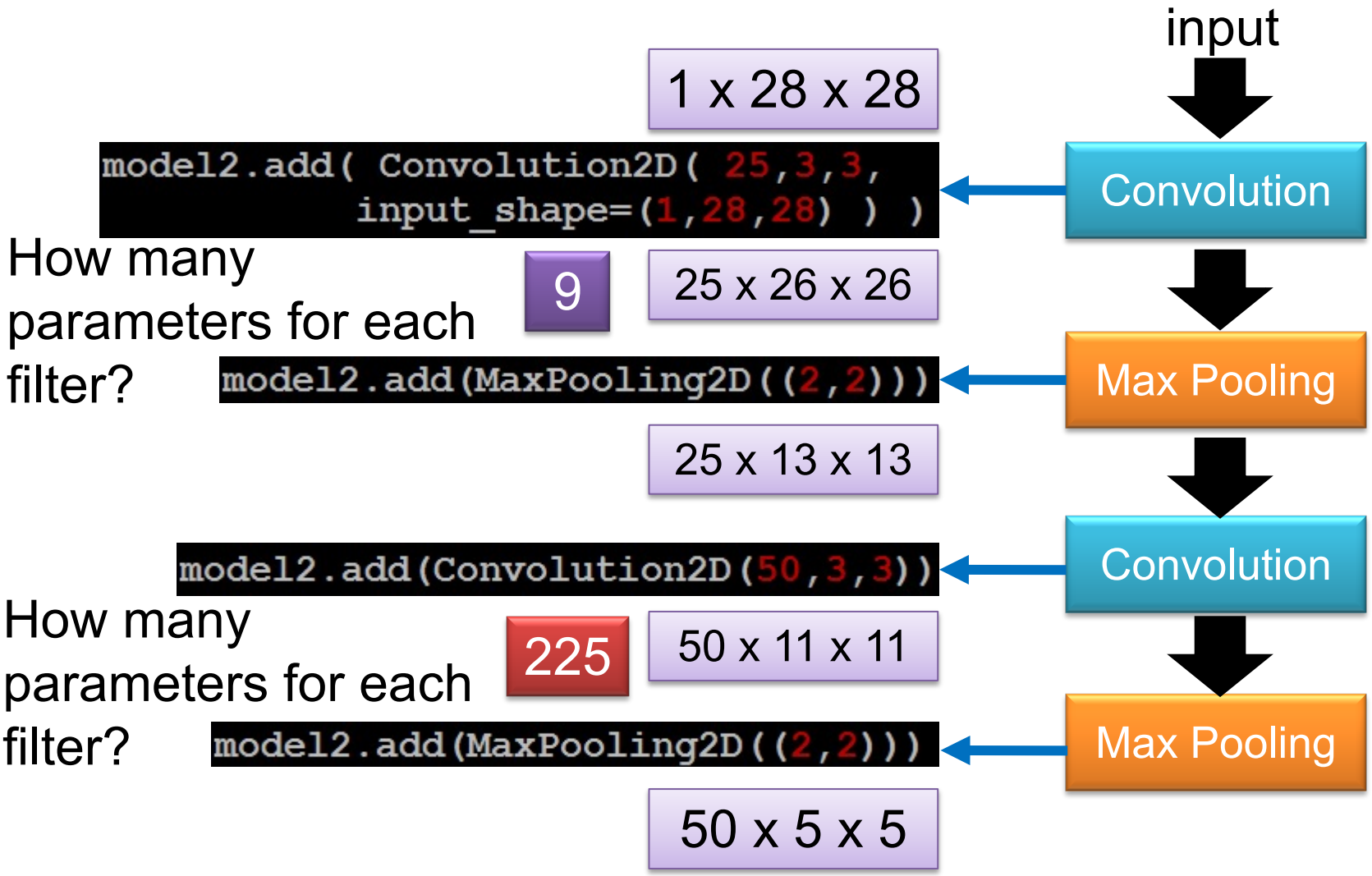
28 x 28 pixels

```
model2.add(MaxPooling2D((2,2)))
```

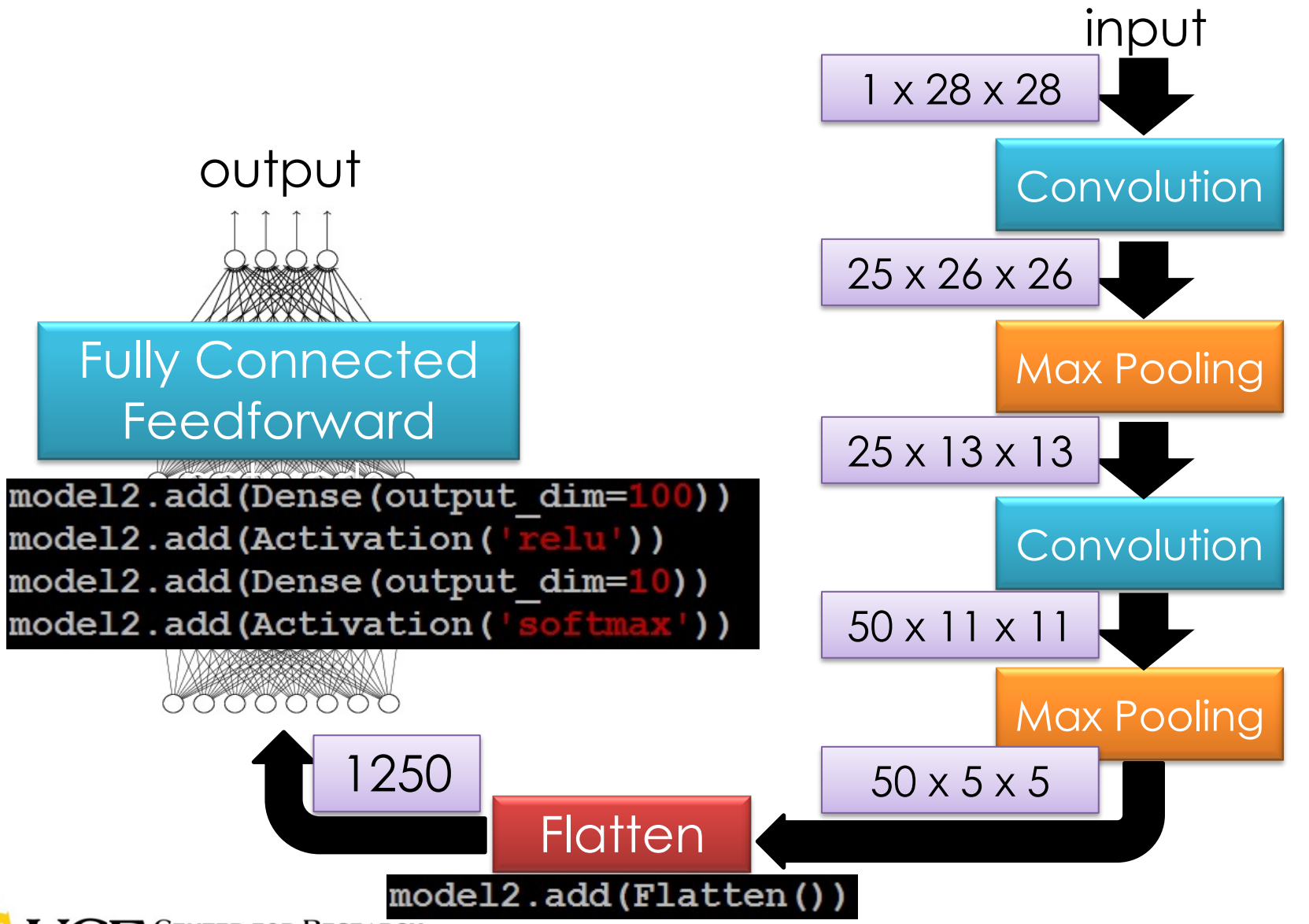




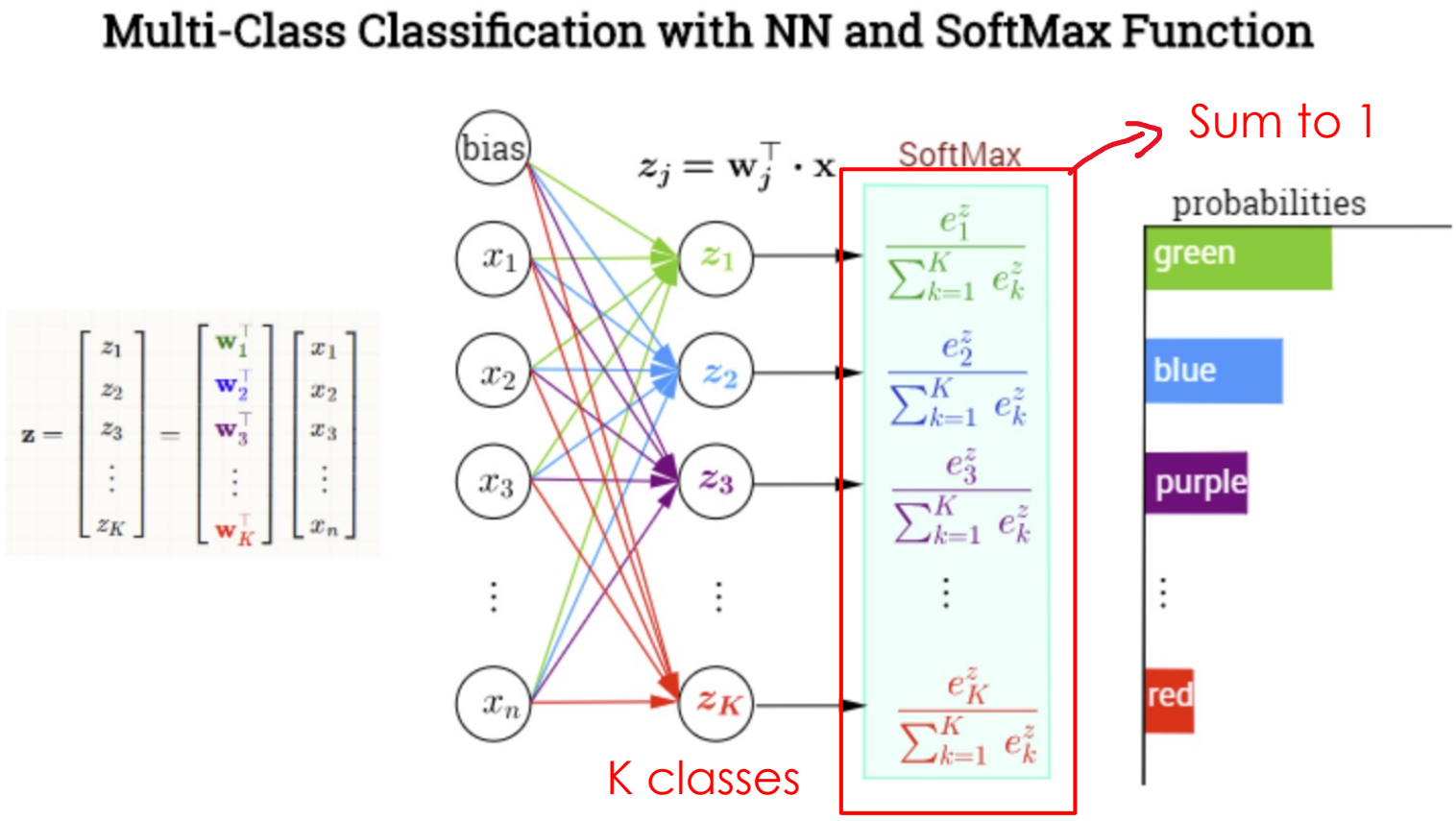
CNN in Keras


input
1 x 28 x 28
model2.add(Convolution2D(25,3,3,
input_shape=(1,28,28)))
Convolution
How many parameters for each filter?
9
25 x 26 x 26
model2.add(MaxPooling2D((2,2)))
Max Pooling
25 x 13 x 13
model2.add(Convolution2D(50,3,3))
Convolution
How many parameters for each filter?
225
50 x 11 x 11
model2.add(MaxPooling2D((2,2)))
Max Pooling
50 x 5 x 5

CNN in Keras


input
1 x 28 x 28
Convolution
25 x 26 x 26
Max Pooling
25 x 13 x 13
Convolution
50 x 11 x 11
Max Pooling
50 x 5 x 5
Flatten
model2.add(Flatten())
1250
Fully Connected Feedforward
output
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))

Softmax



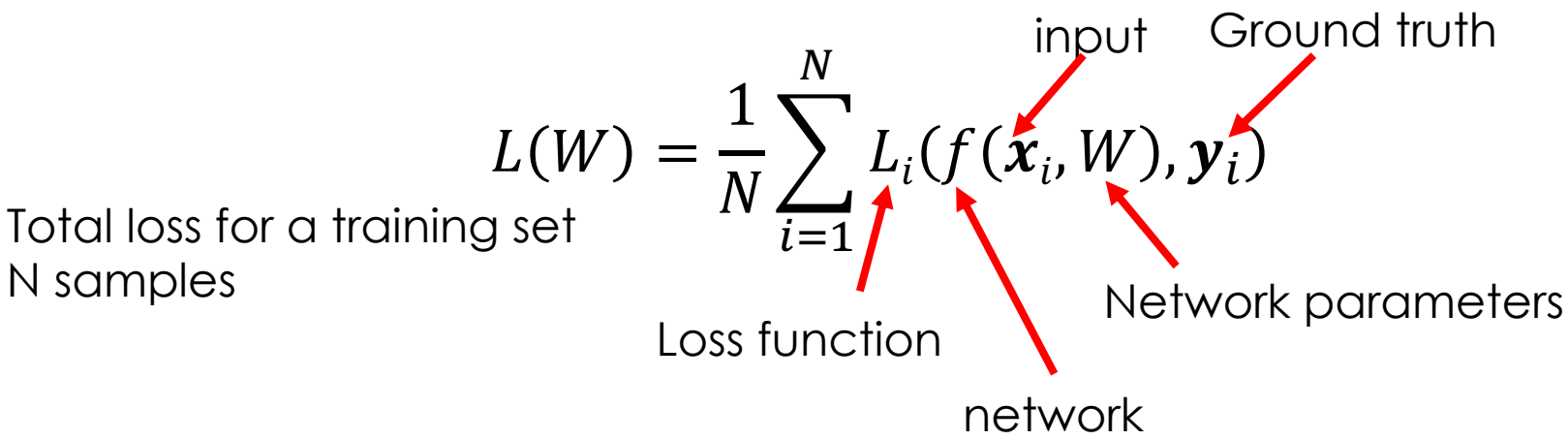
The softmax as

$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}$$

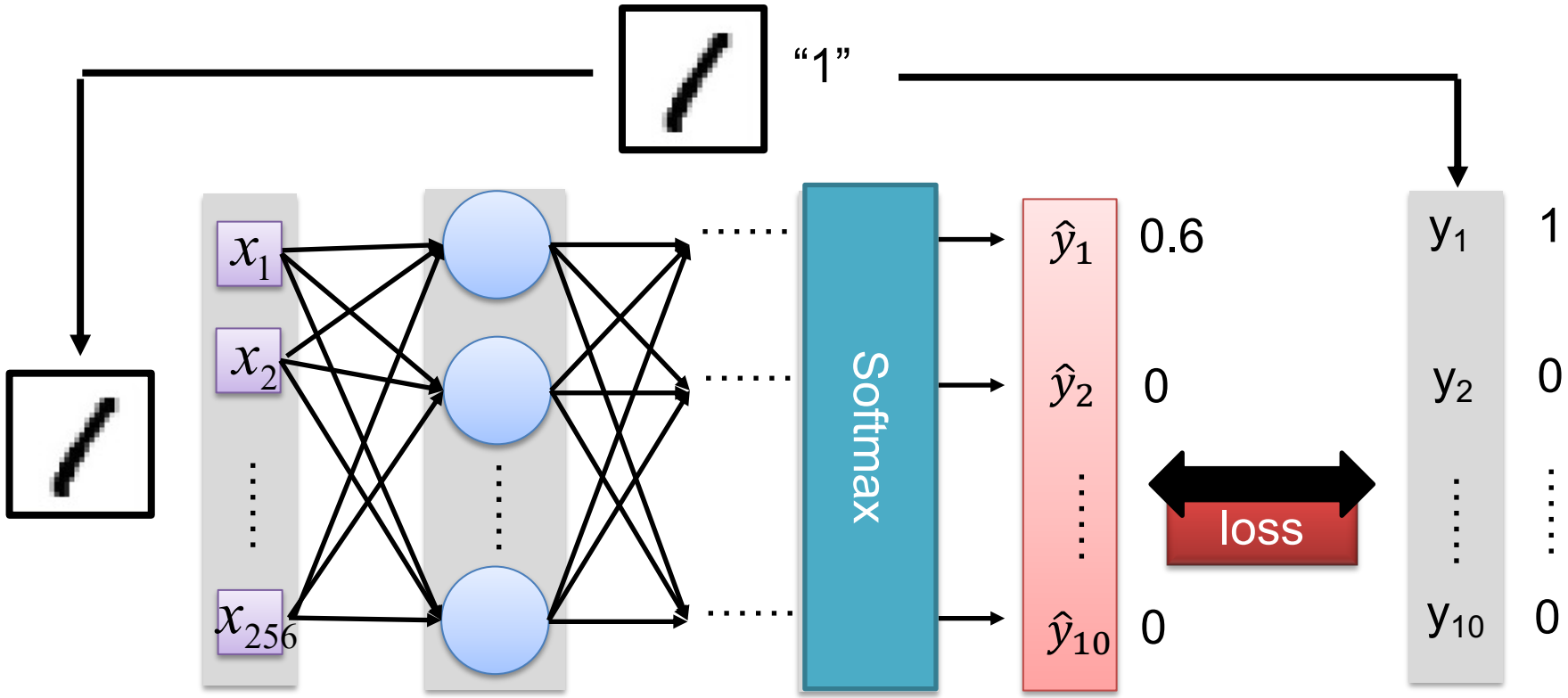
https://victorzhou.com/blog/softmax/

Loss Function

- Way to define how good the network is performing
  - In terms of prediction
- Network training (Optimization)
  - Find the best network parameters to minimize the loss



Choosing Proper Loss



Square Error

$$\sum_{i=1}^{10} (y_i - \hat{y}_i)^2$$

Target (one hot encoding)

Cross-entropy loss

- Binary case (we know this from logistic regression)

$$-(ylog(\hat{y}) + (1-y)lo\,g(1-\hat{y}))$$

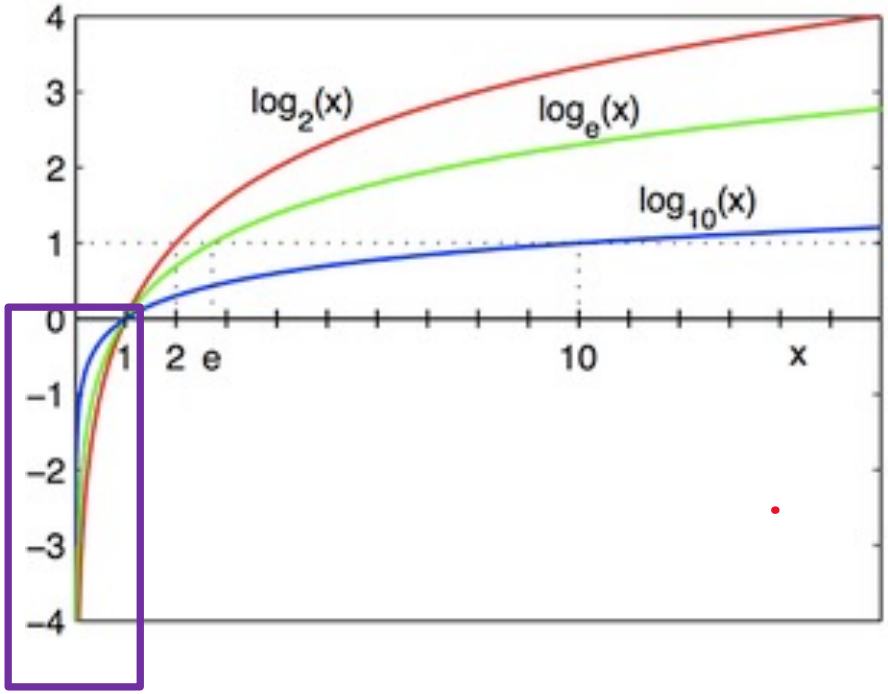
- General form

K classes

$$-\sum_{i=1}^{K} y_i log(\hat{y}_i)$$

Label $\boldsymbol{y}$ is a one-hot vector

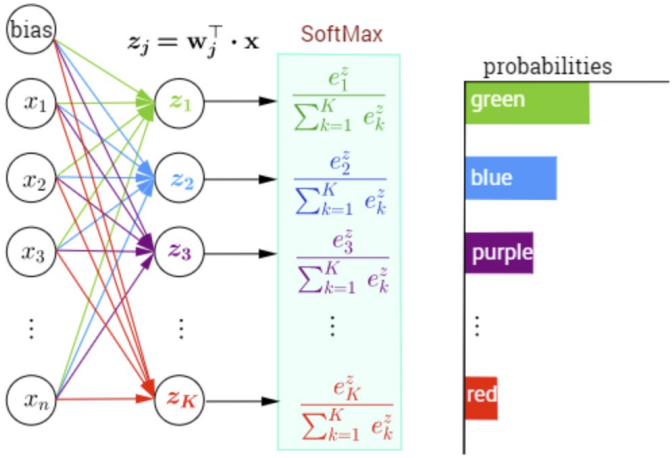
$y_i \in [0,1]$



Loss Function

- Multi-class classification

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



Cross-Entropy loss for one sample

$$-\sum_{i=1}^{K} y_i log(\hat{y}_i)$$

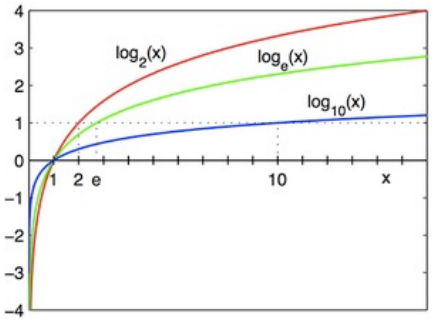
Label $\boldsymbol{y}$ is a one-hot vector

$(y_i) \in [0,1]$

Softmax

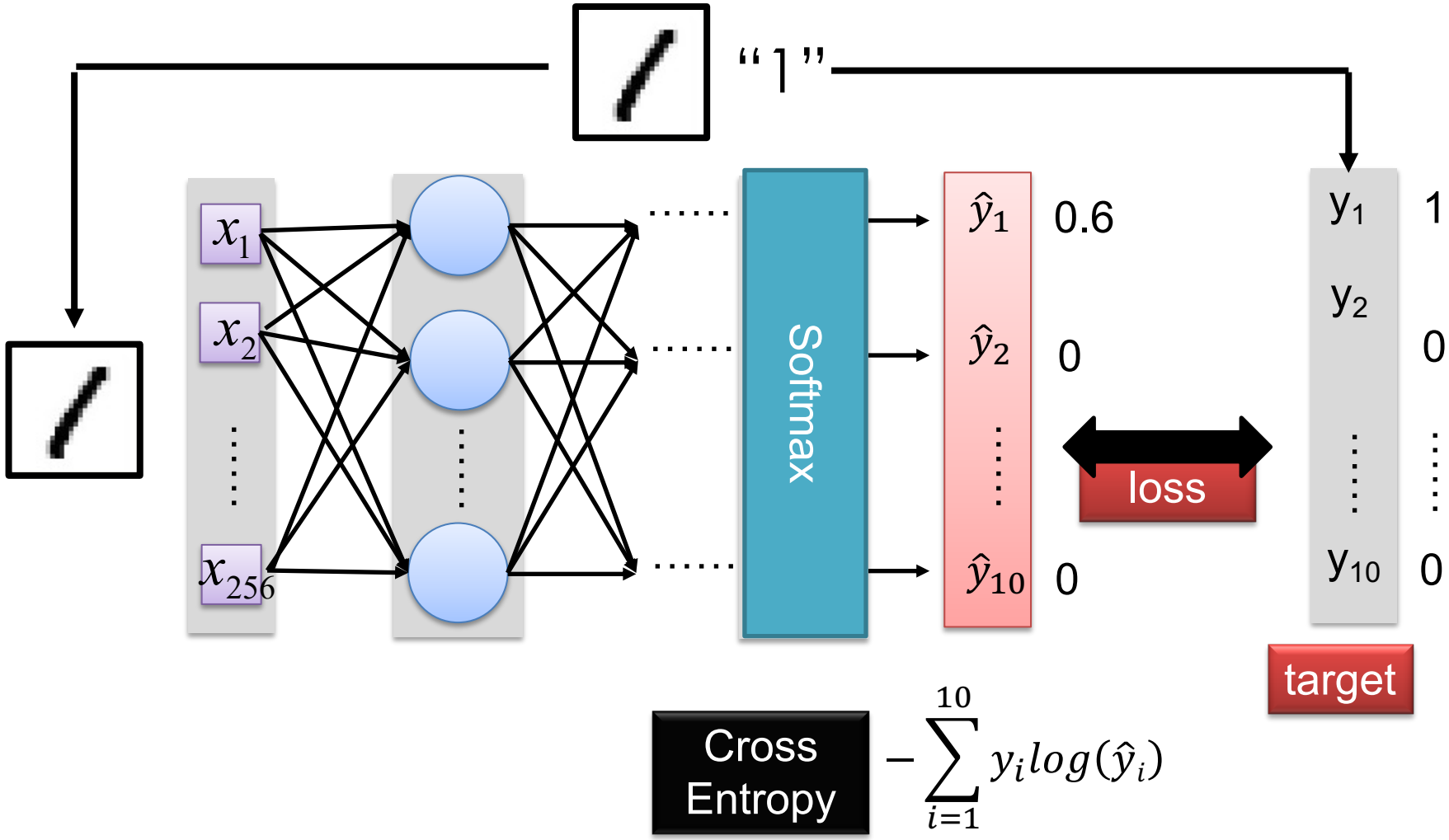
$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}$$

computed	targets	correct?
0.3 0.3 0.4	0 0 1 (democrat)	yes
0.3 0.4 0.3	0 1 0 (republican)	yes
0.1 0.2 0.7	1 0 0 (other)	no



$$-((\log(0.3) \times 0) + (\log(0.3) \times 0) + (\log(0.4) \times 1)) = -\log(0.4)$$

Choosing Proper Loss



Square Error

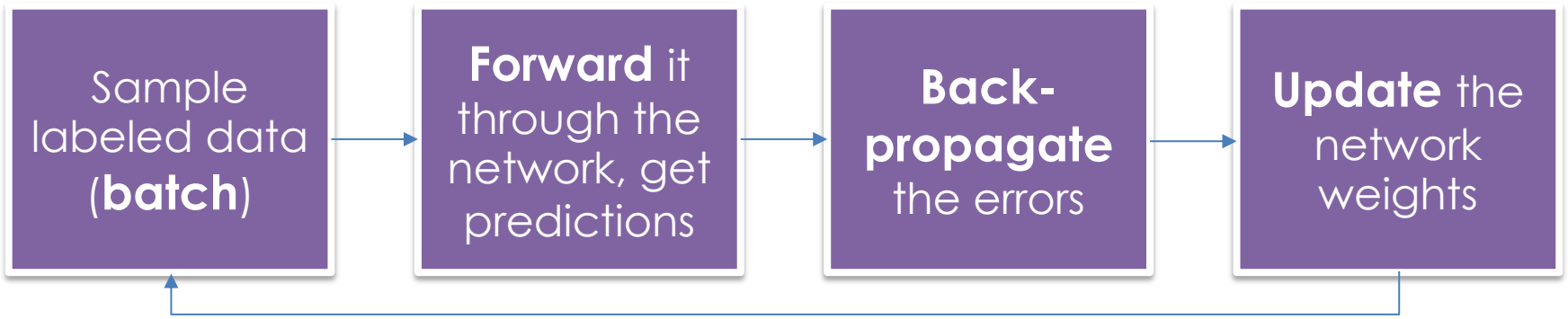
```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Cross Entropy

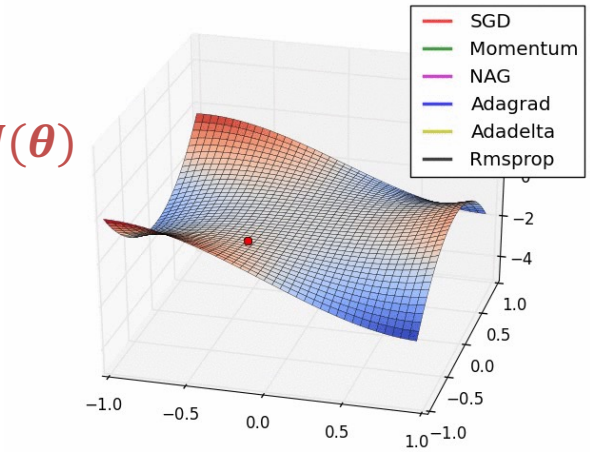
```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

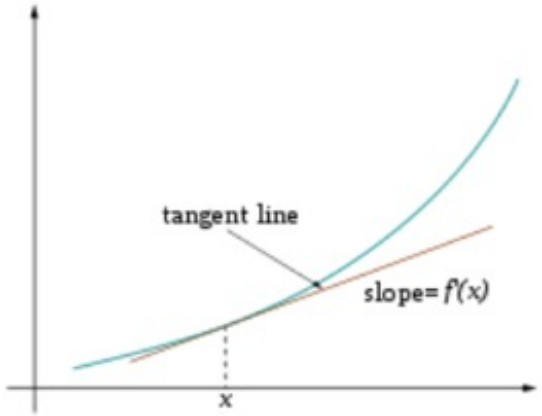
Several alternatives: https://keras.io/objectives/

Training



Optimize (min. or max.) **objective/cost function $J(\theta)$**
Generate **error signal** that measures difference between predictions and target values





Use error signal to change the **weights** and get more accurate predictions
Subtracting a fraction of the **gradient** moves you towards the **(local) minimum of the cost function**

https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39

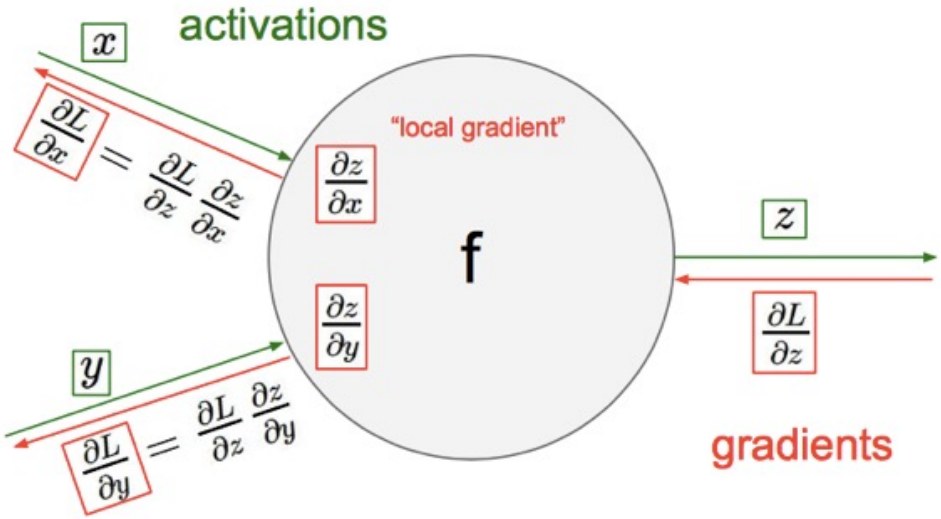
Gradient Descent

objective/cost function $J(\theta)$

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{d}{d\theta_j^{old}} J(\theta)$$ Update each element of θ

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$ Matrix notation for all parameters

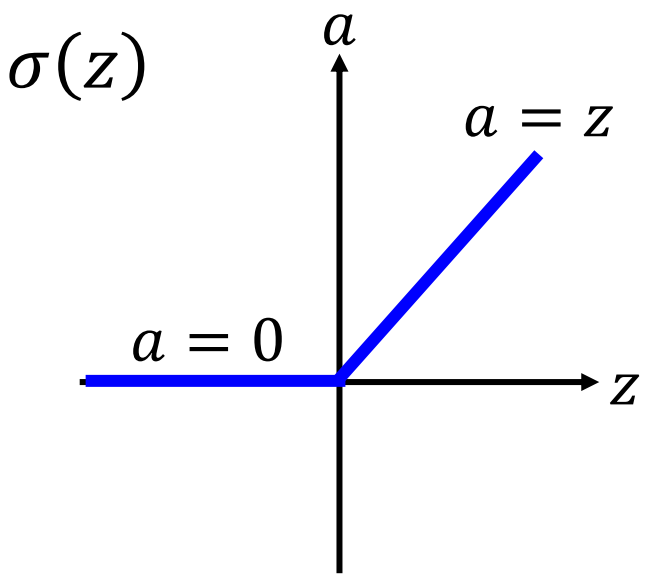
learning rate



Recursively apply **chain rule** though each node

ReLU

- Rectified Linear Unit (ReLU)

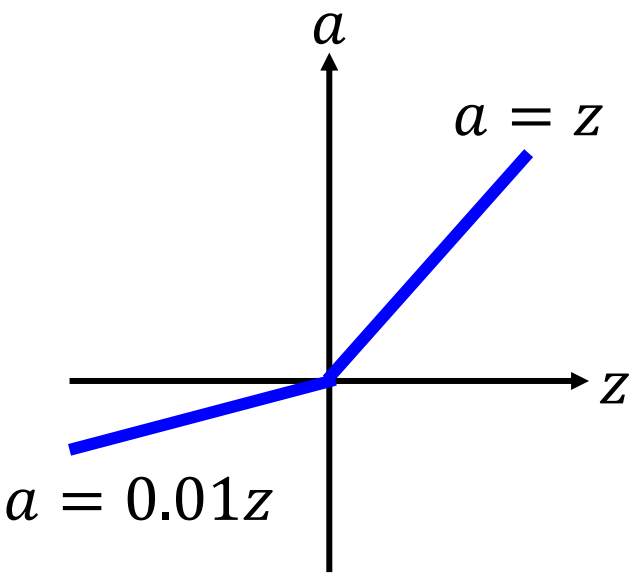


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

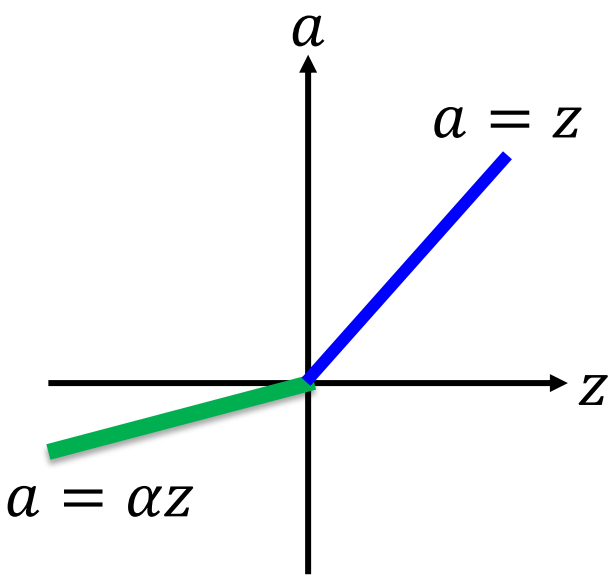
Vanishing gradient problem

ReLU - variant

Leaky ReLU



Parametric ReLU



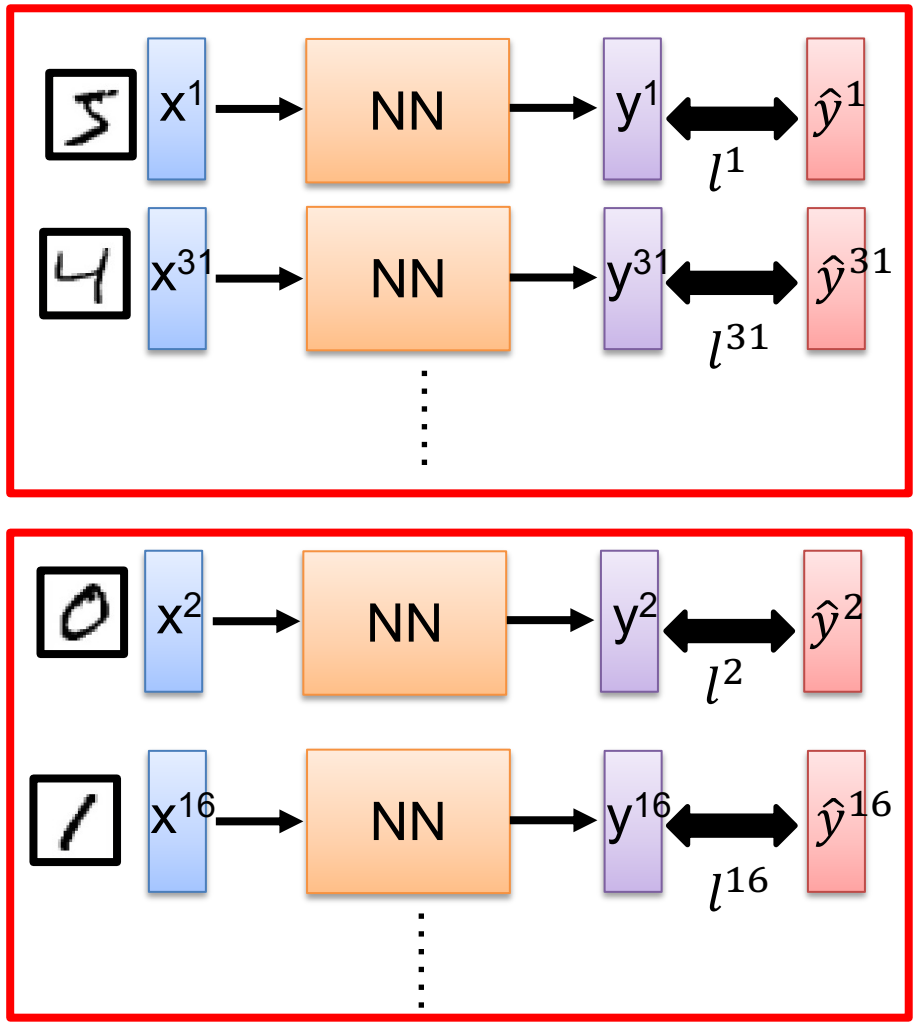
α also learned by gradient descent

Stochastic Gradient Descent

- Gradient over entire dataset is impractical
- Better to take quick, noisy steps
- Estimate gradient over a mini-batch of examples

Mini-batch

We do not really minimize total loss!



- Randomly initialize network parameters

- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  Update parameters once

- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  Update parameters once

  ⋮

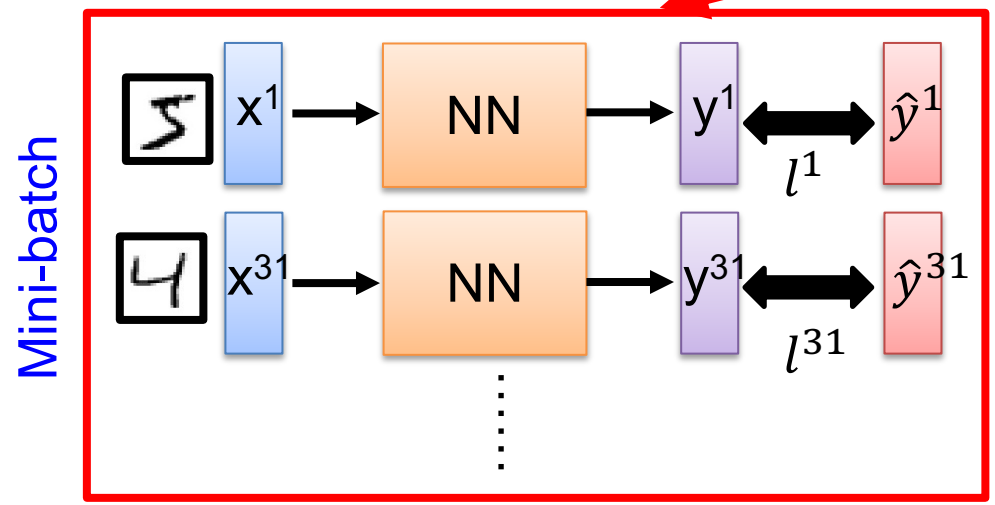
- Until all mini-batches have been picked

one epoch

Repeat the above process

Mini-batch

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```



100 examples in a mini-batch

- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  Update parameters once

- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  Update parameters once

  ⋮

- Until all mini-batches have been picked

Repeat 20 times

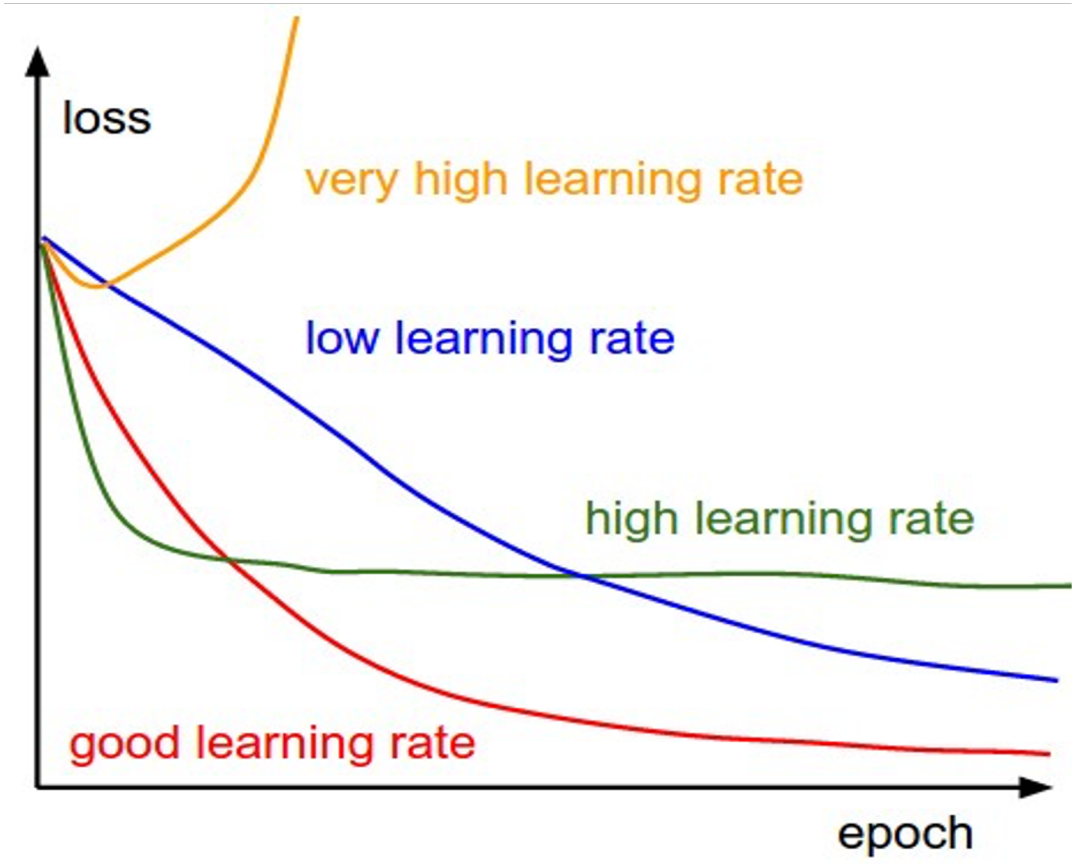
one epoch

Other optimizers

- Adagrad [John Duchi, JMLR'11]
- RMSprop
  - https://www.youtube.com/watch?v=O3sxAc4hxZU
- Adadelta [Matthew D. Zeiler, arXiv'12]
- "No more pesky learning rates" [Tom Schaul, arXiv'12]
- AdaSecant [Caglar Gulcehre, arXiv'14]
- Adam [Diederik P. Kingma, ICLR'15]
- Nadam
  - http://cs229.stanford.edu/proj2015/054_report.pdf

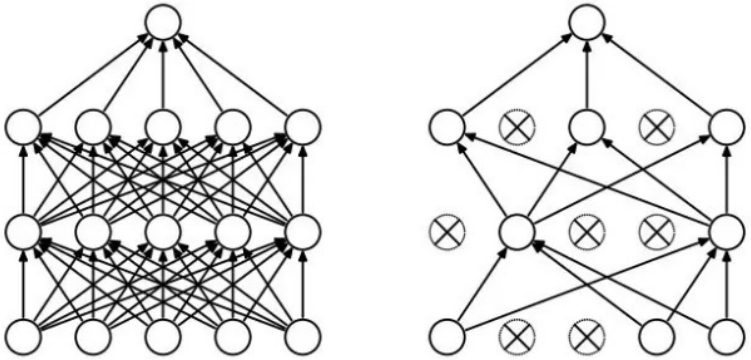
How to pick the learning rate?

- Too big = diverge, too small = slow convergence
- No “one learning rate to rule them all”
- Start from a high value and keep cutting by half if model diverges
- Learning rate schedule: decay learning rate over time



http://cs231n.github.io/assets/nn3/learningrates.jpeg

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

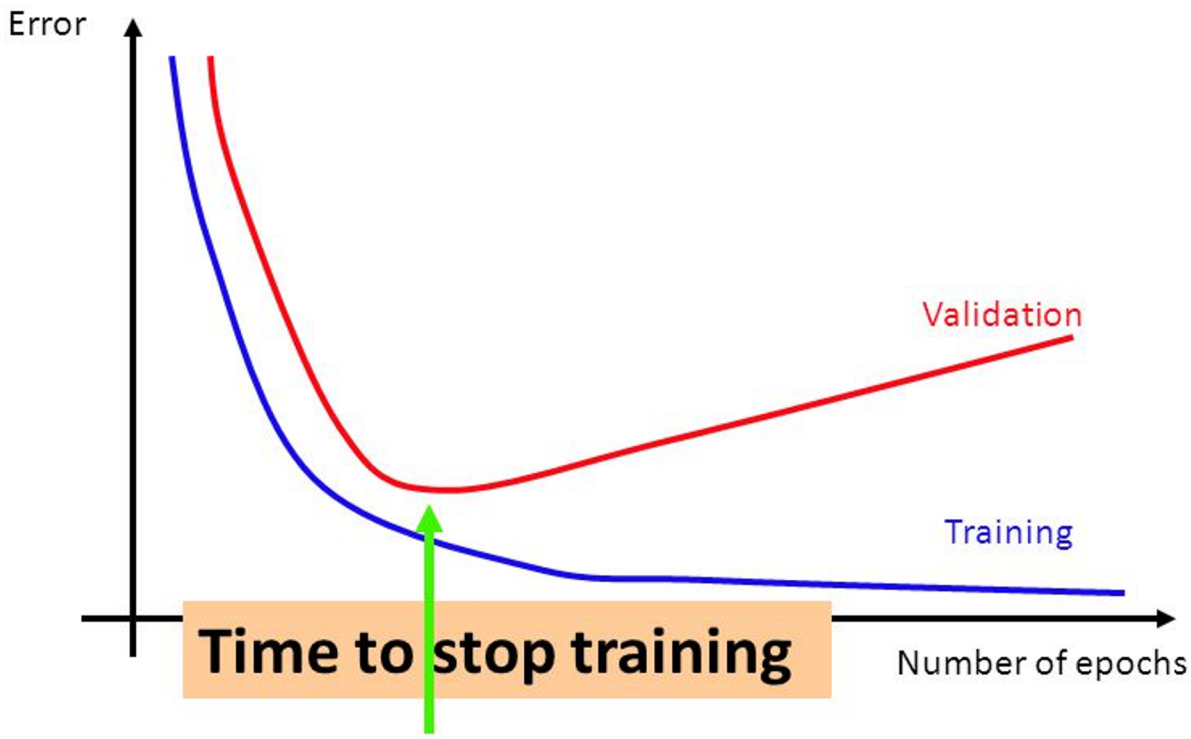
- Regularization term that penalizes big weights, added to the objective – reduce overfitting
- Weight decay value determines how dominant regularization is during gradient computation
- Big weight decay coefficient → big penalty for big weights

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

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

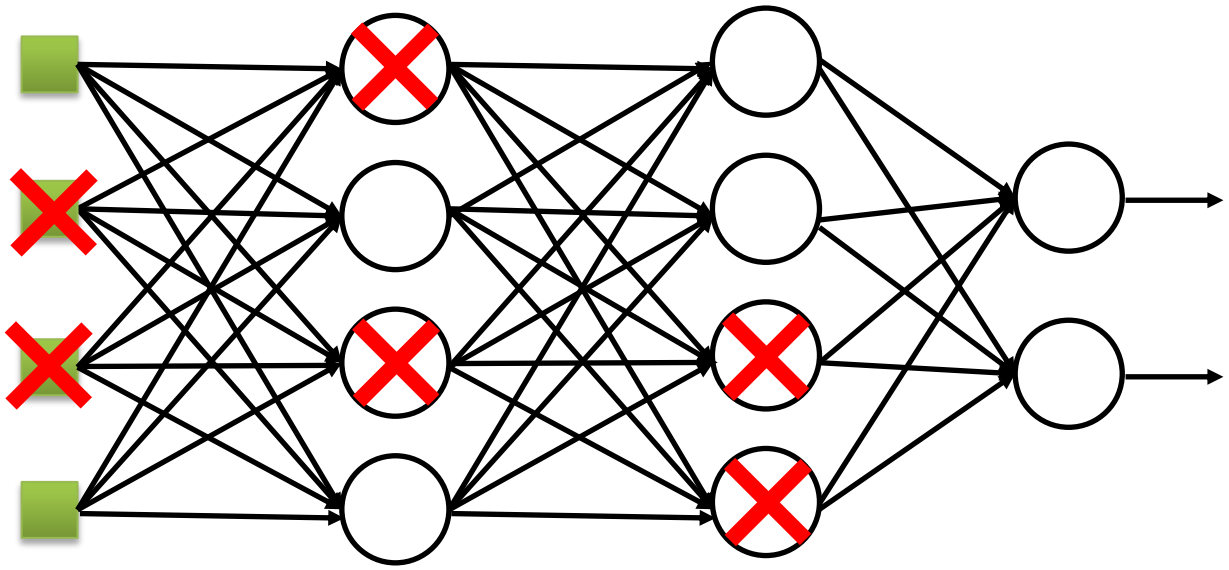
```
1  # example of batch normalization for an cnn
2  from keras.layers import Dense
3  from keras.layers import Conv2D
4  from keras.layers import MaxPooling2D
5  from keras.layers import BatchNormalization
6  ...
7  model.add(Conv2D(32, (3,3), activation='relu'))
8  model.add(Conv2D(32, (3,3), activation='relu'))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

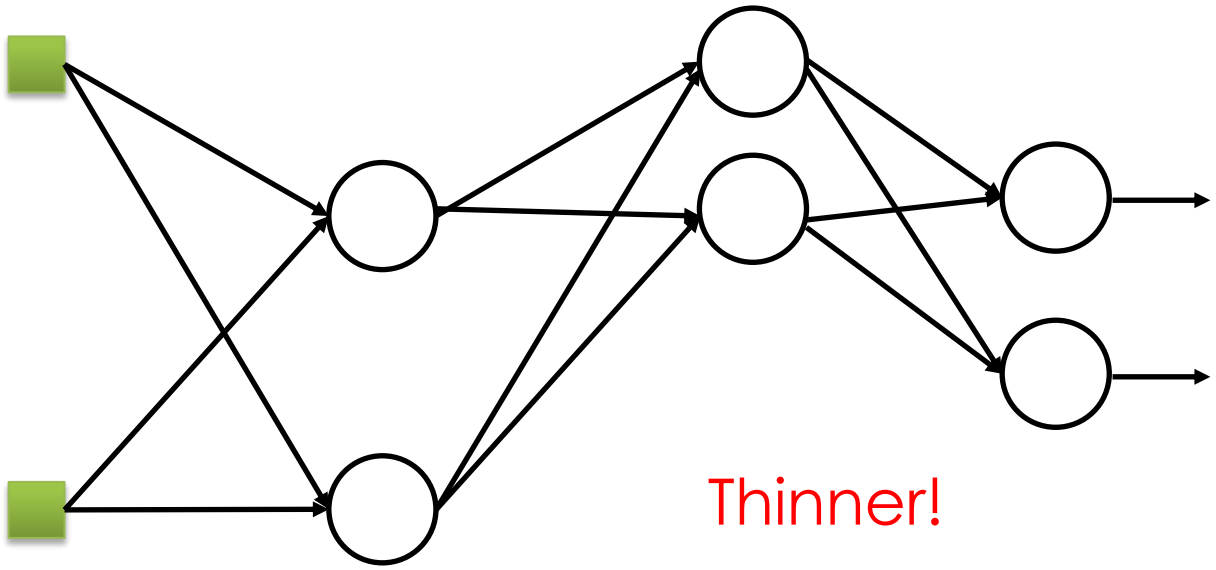
Training:



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

Dropout

Training:

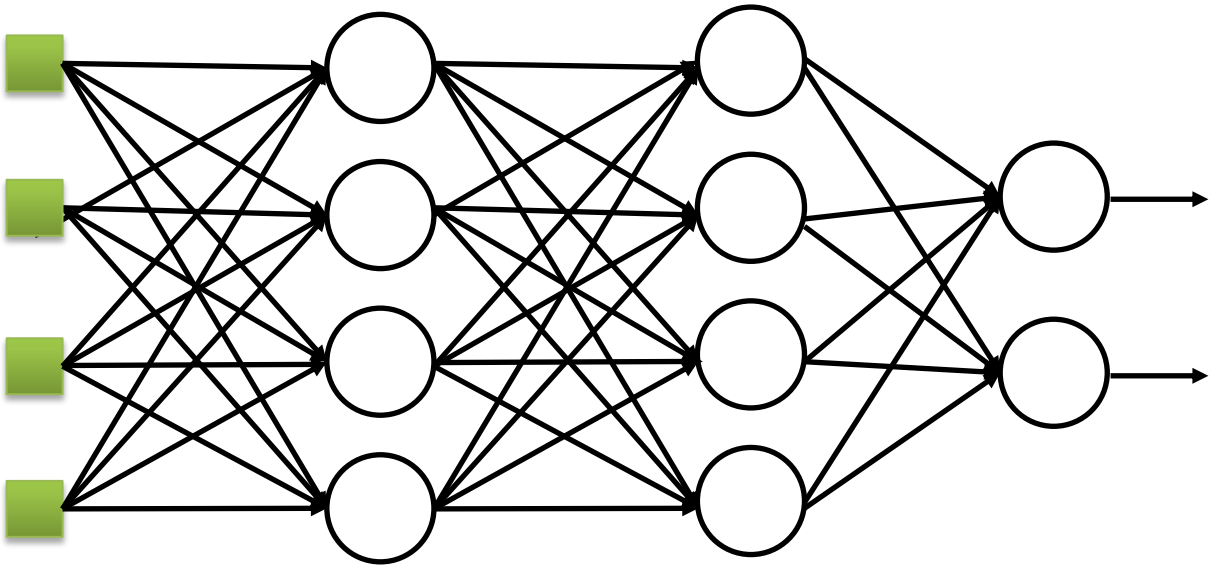


- **Each time before updating the parameters**
  - Each neuron has p% to dropout
  - Using the new network for training

The structure of the network is changed.

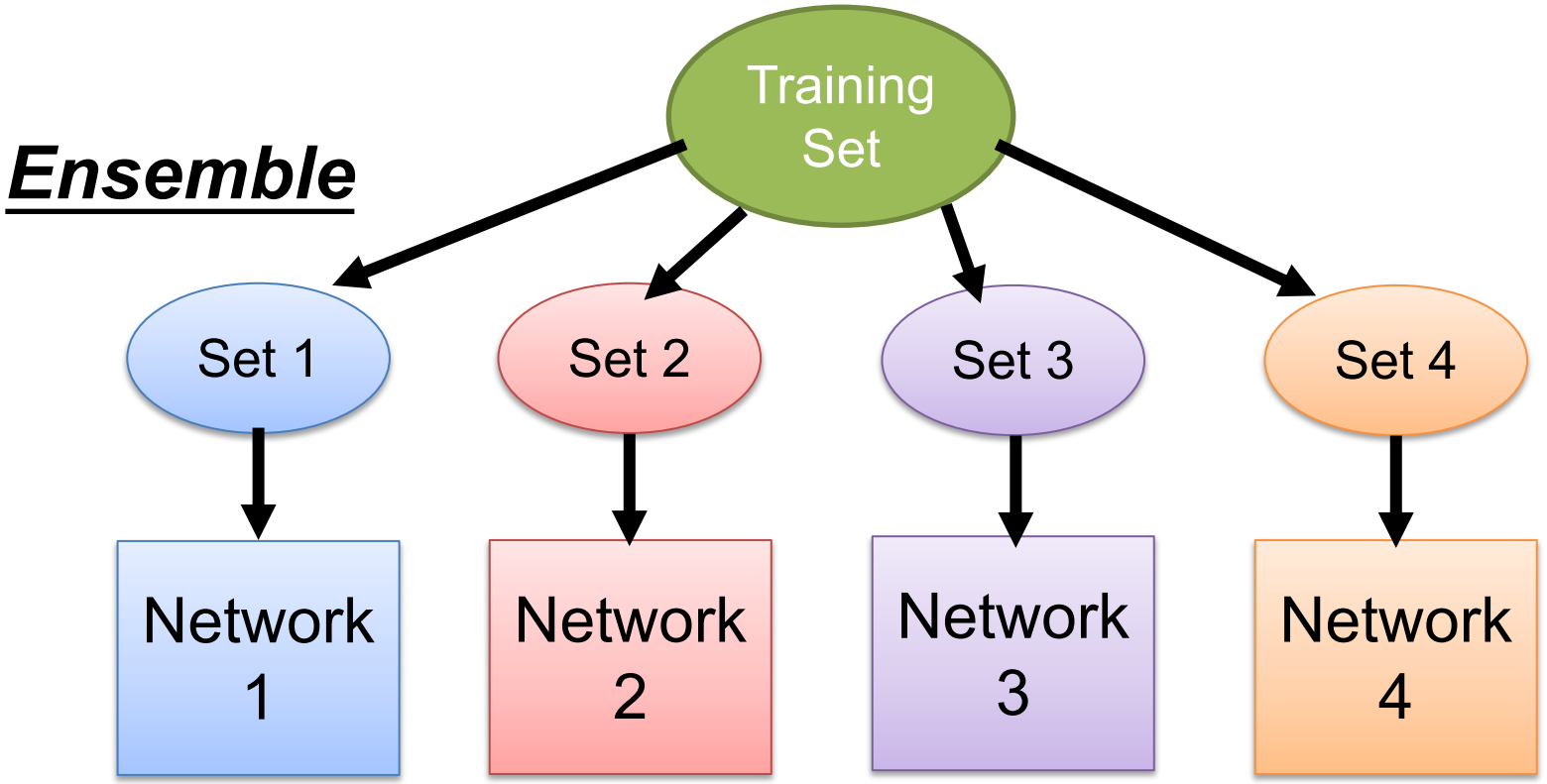
Dropout

Testing:



- No dropout

Dropout is a kind of ensemble



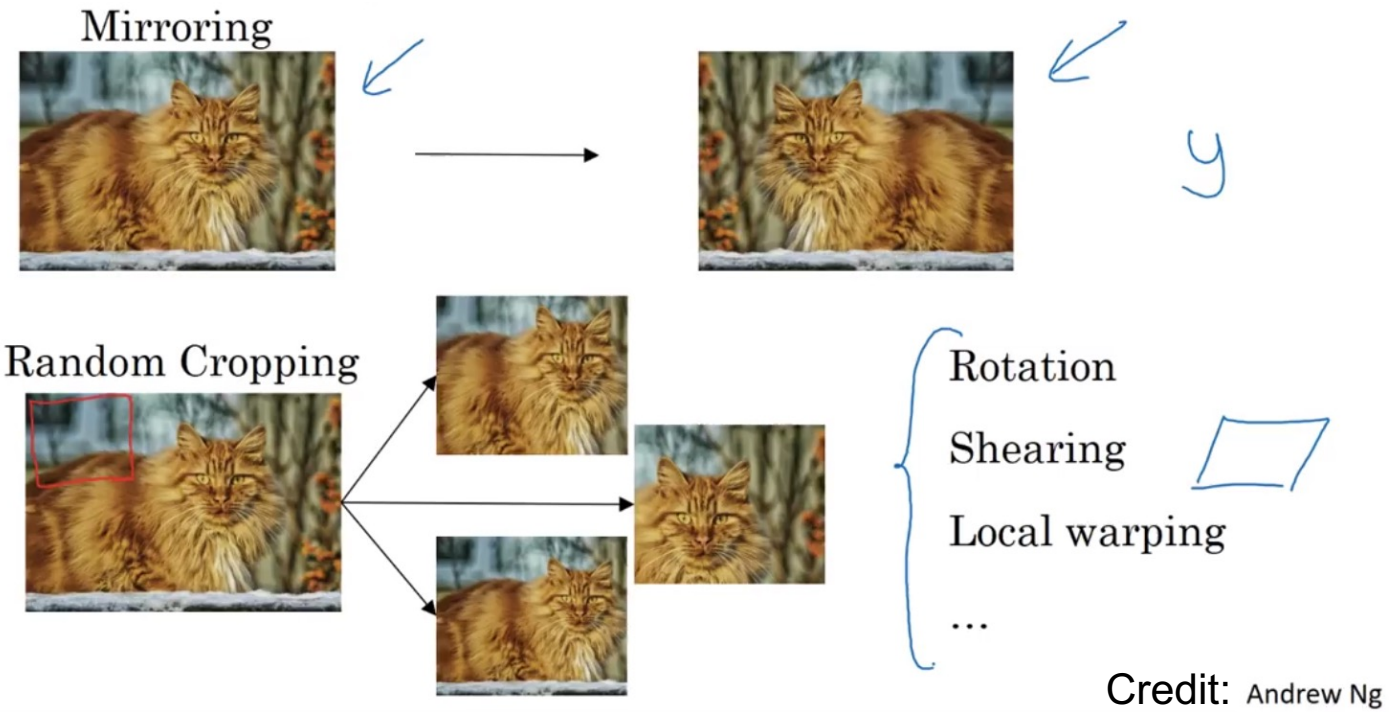
Train a bunch of networks with different structures

Reading Material

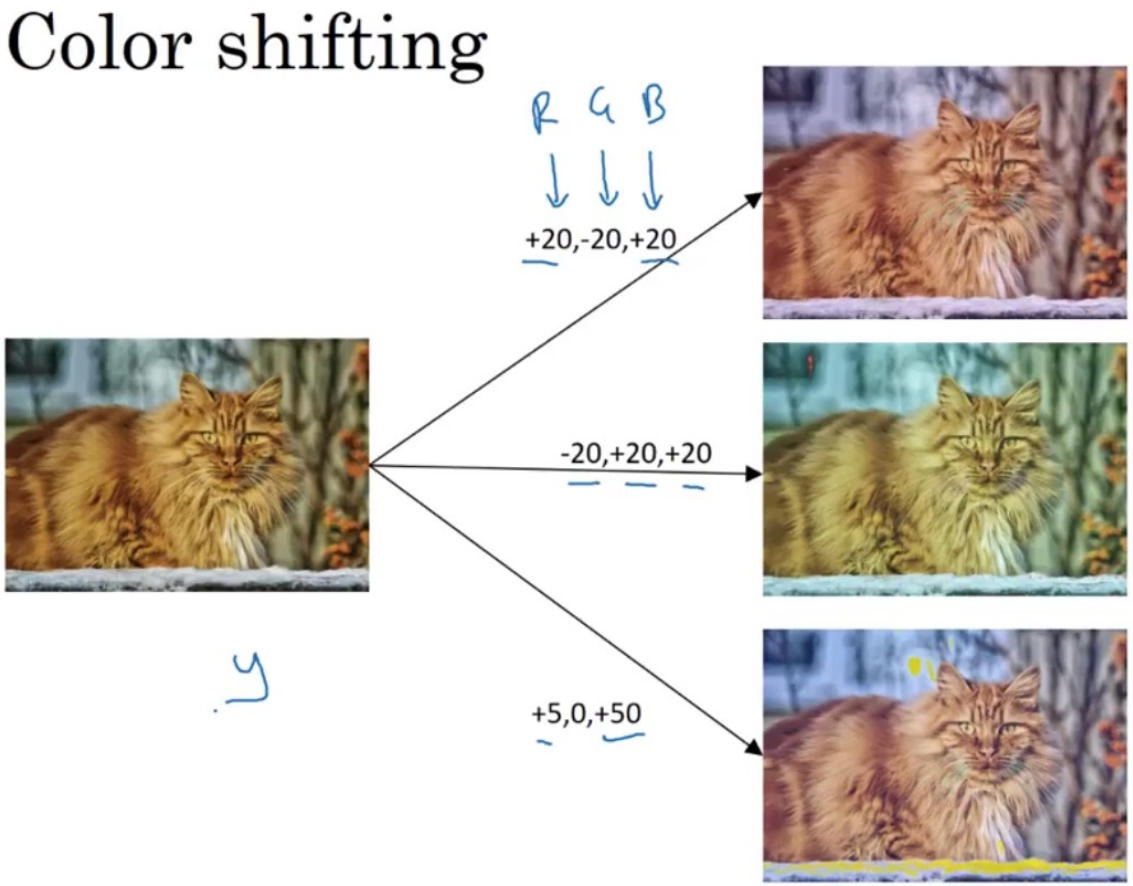
- Training a Classifier — PyTorch Tutorials: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- CNNs, Part 1: An Introduction to Convolutional Neural Networks (Keras implementation)
  - https://victorzhou.com/blog/intro-to-cnns-part-1/
- CNNs, Part 2: Training a Convolutional Neural 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

Data Augmentation

- Why data augmentation -> increase training data



Data Augmentation



Credit: Andrew Ng

Data Augmentation

- Color space conversion

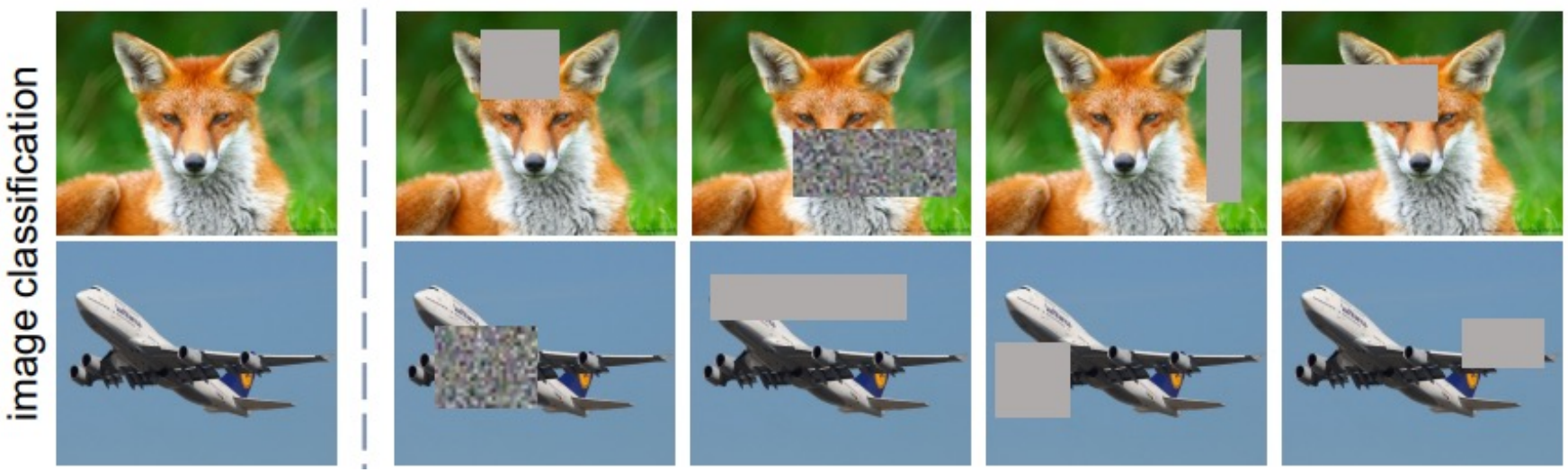
Comparison of results for different color spaces on CIFAR-10 with simple CNN

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

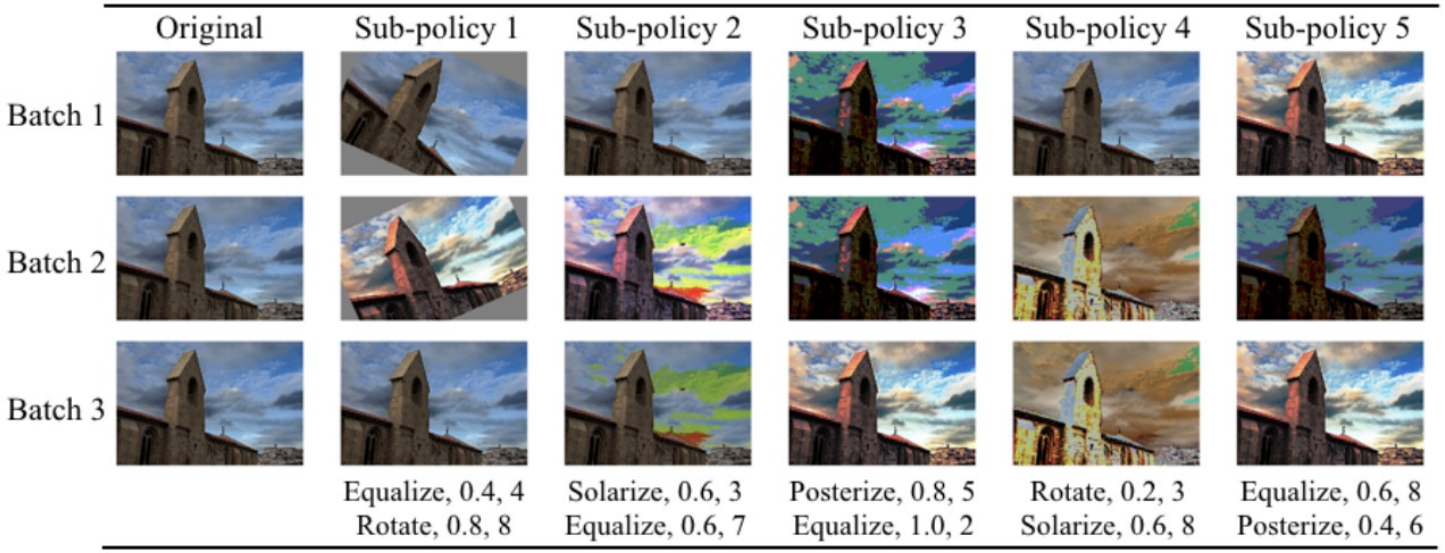


Reduces the risk of over-fitting and makes the model robust to occlusion
Improve the generalization ability of CNNs

Data Augmentation

- Learning Augmentation Policies from Data

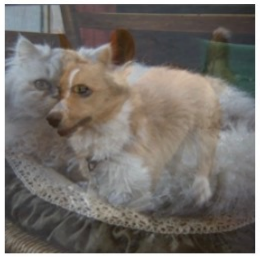
Search space of operations: ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness,



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

Data Augmentation

- Cutmix

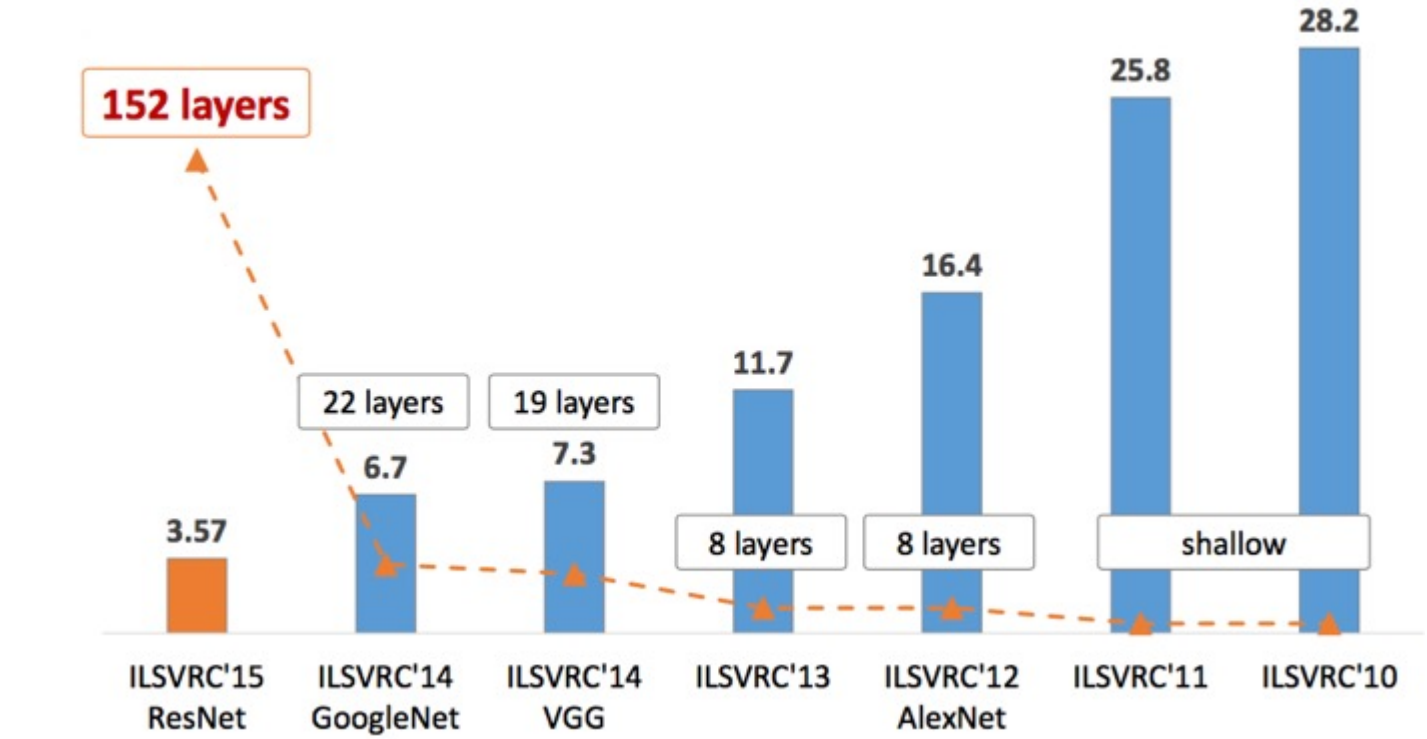
	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6** **(+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)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

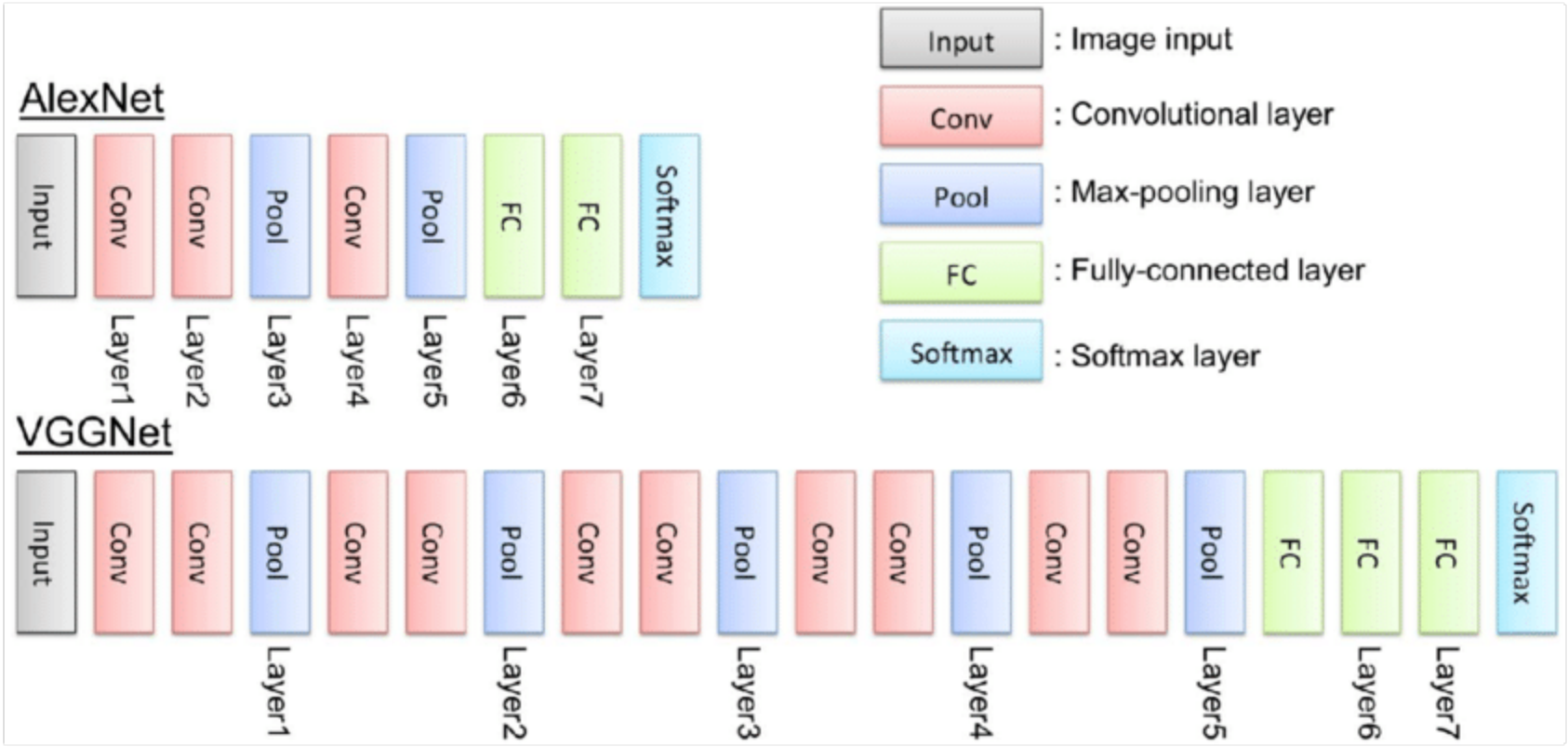
- More on data augmentation
  - (Pytorch) torchvision.transforms: https://pytorch.org/vision/master/transforms.html
  - https://github.com/AgaMiko/data-augmentation-review

Case study

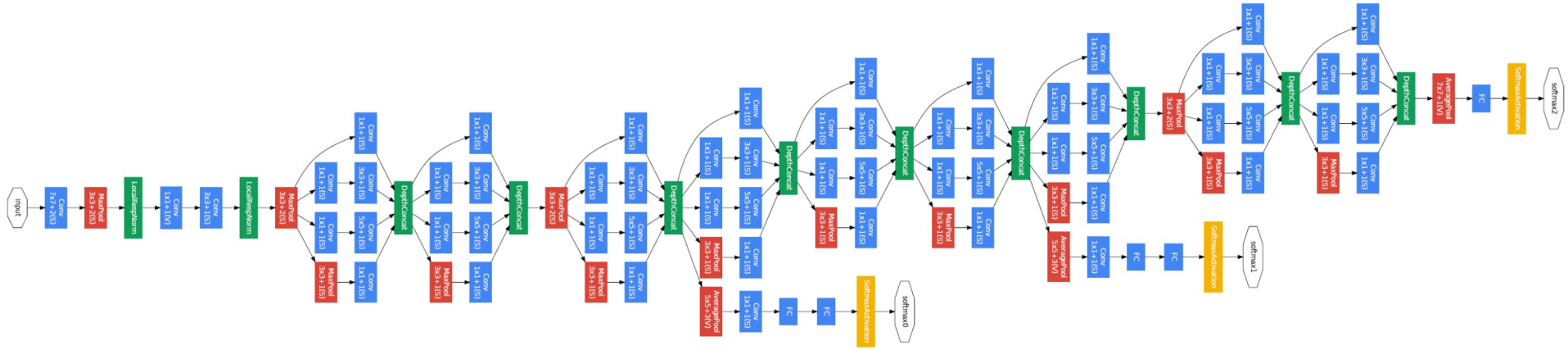
- ImageNet challenge



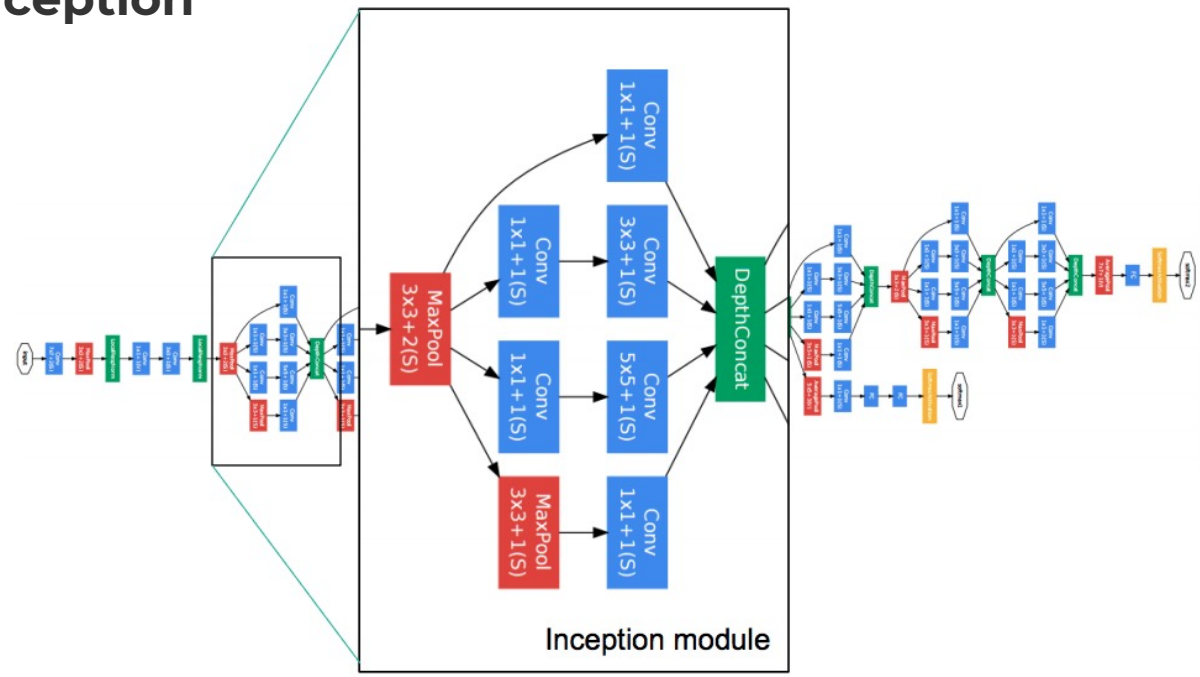
Case study


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

Case study

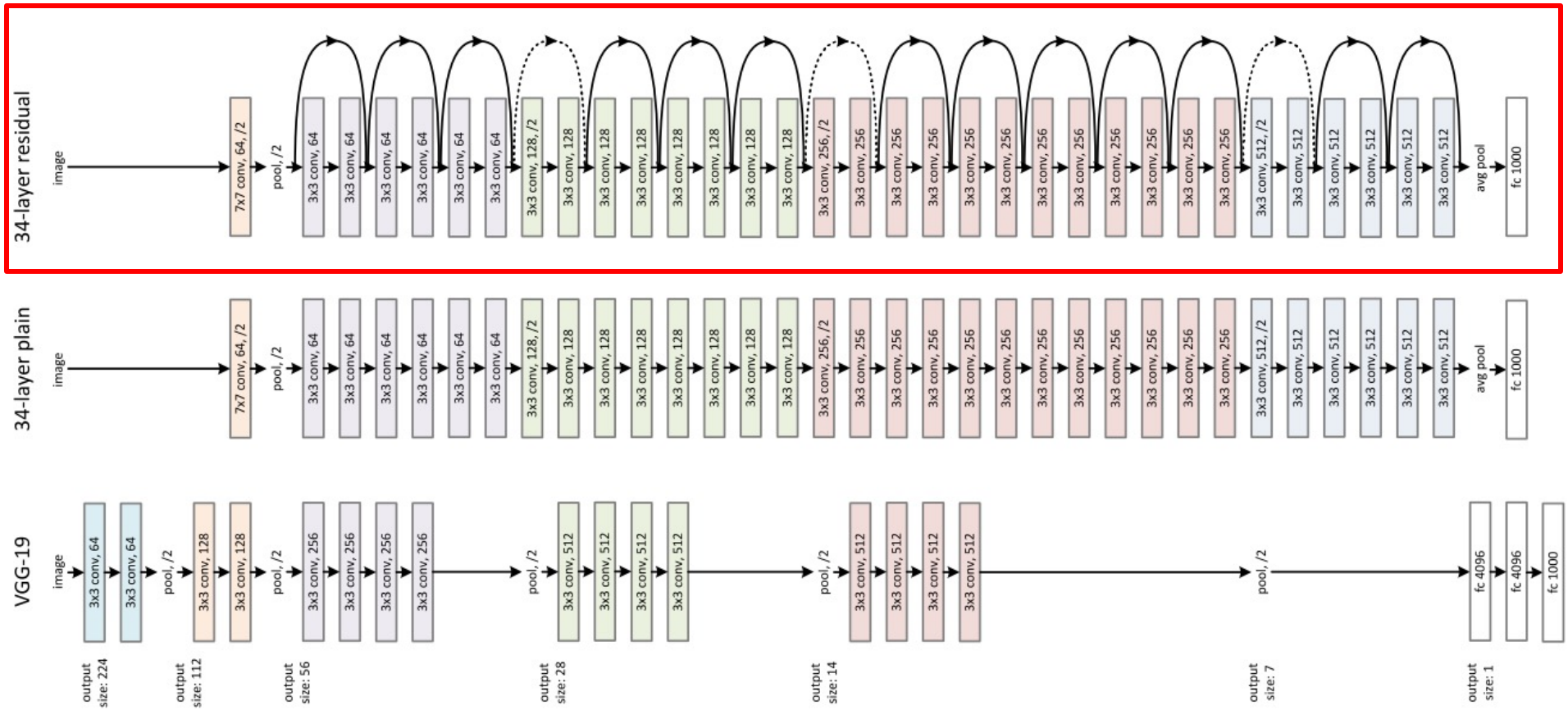


GoogLeNet/Inception



Case study

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

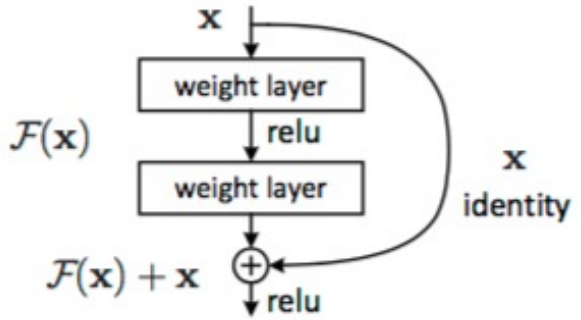


Figure 2. Residual learning: a building block.

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

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/

Applications

- Classification
- Detection
- Action recognition (spatial –temporal)
- Segmentation
- Image generation

.....

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