CAP 6412
Advanced Computer Vision

Lecture 2

Yogesh Rawat

What will we cover?

Part -I

- Convolutional Neural Network
  - There widespread use in deep learning
- Case study – AlexNet

Part-II

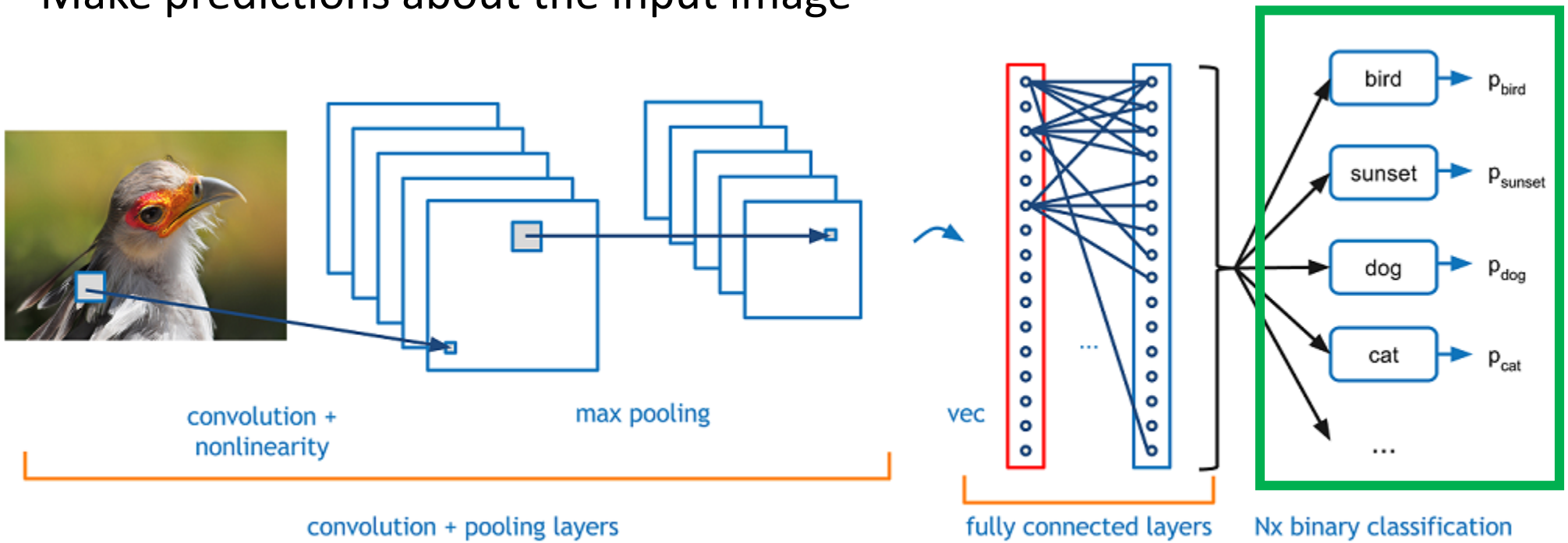
- Network training
- Issues
- Other networks
  - CNN variants
  - Recurrent neural network
- Generative adversarial network

Part-I

- Convolutional Neural Network
  - There widespread use in deep learning
- Case study – AlexNet

Convolutional Neural Network (CNN)

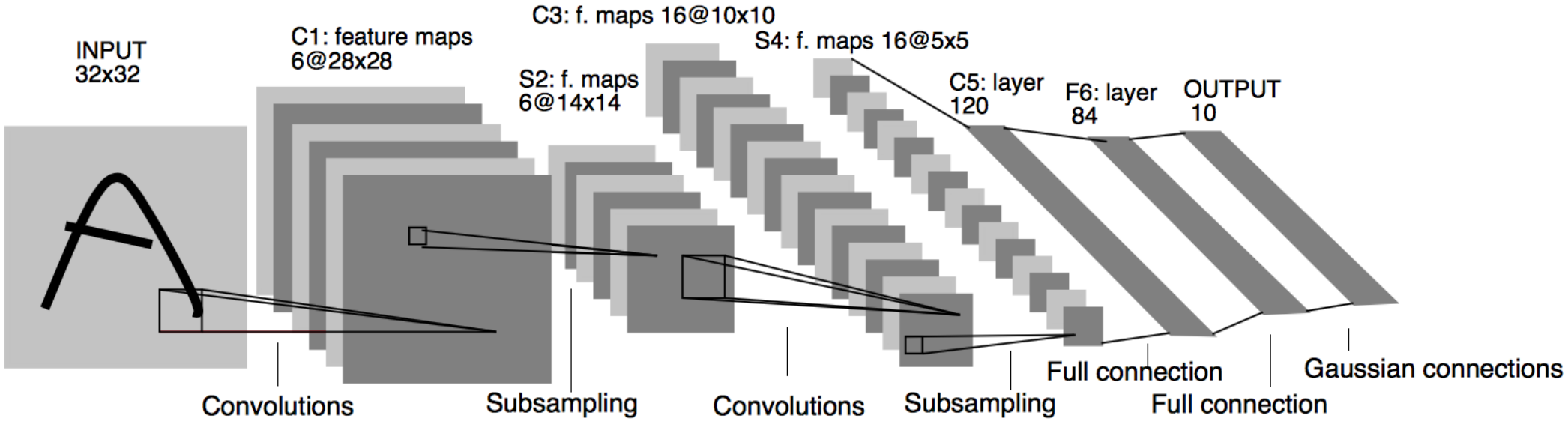
- A class of Neural Networks
  - Takes image as input
  - Make predictions about the input image



Source : https://adeshpande3.github.io

History

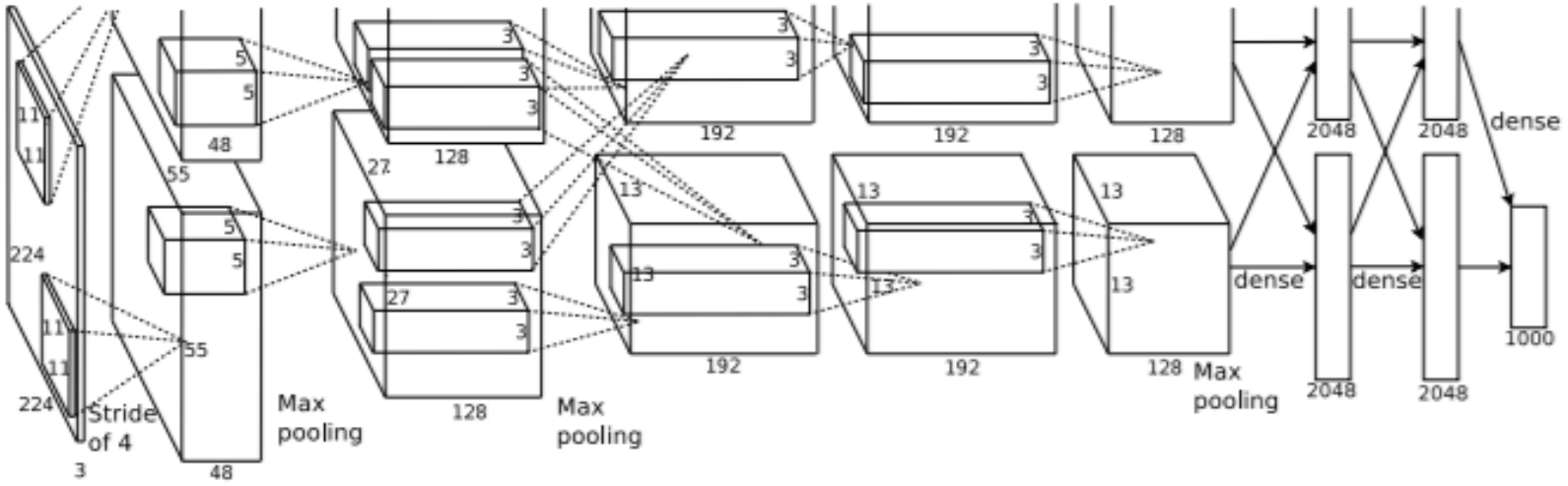
- The LeNet architecture (1990s)



Gradient-based learning applied to document recognition
LeCun Y, Bottou L, Bengio Y, Haffner P. Proceedings of the IEEE. 1998

First Strong Results

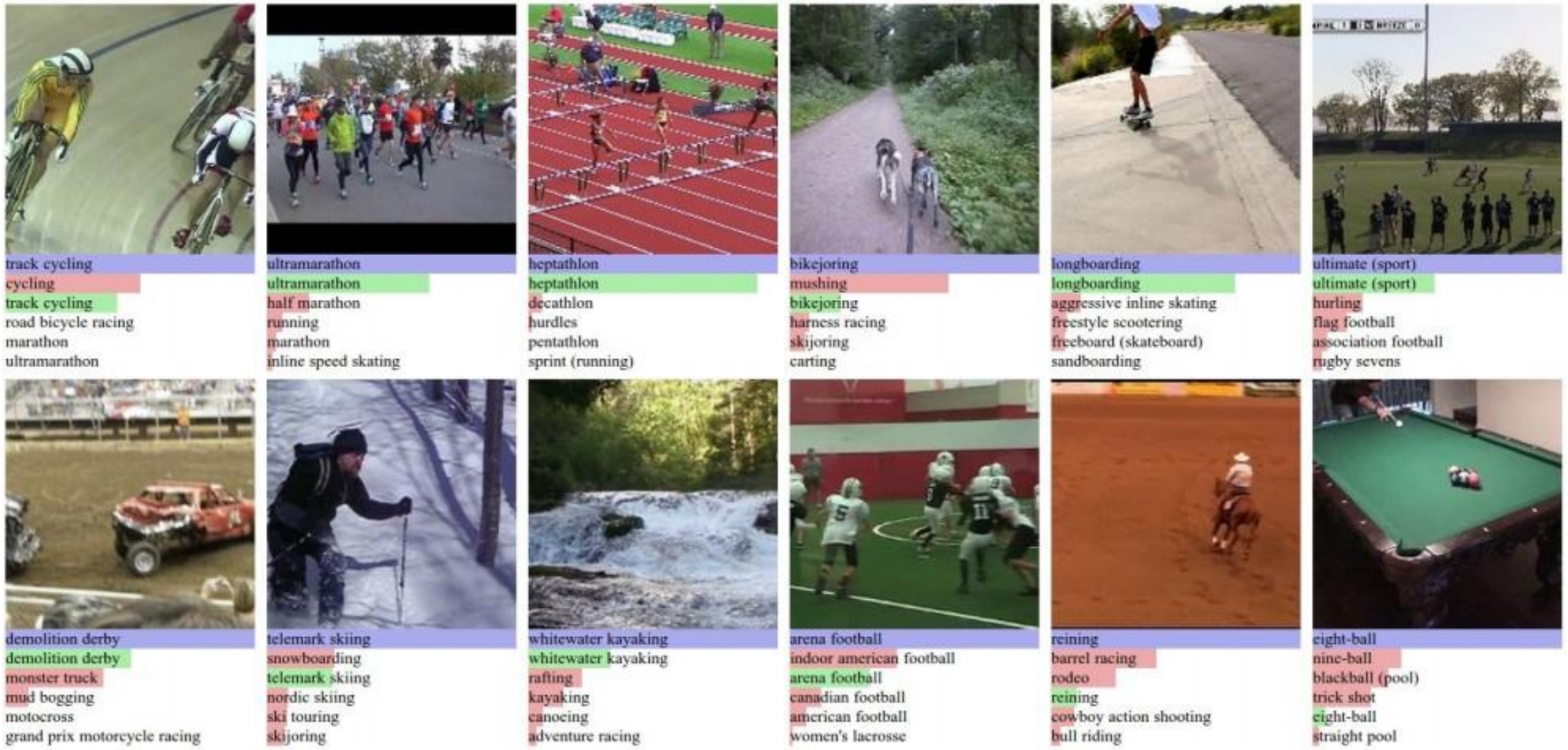
- AlexNet 2012
  - Winner of ImageNet Large-Scale Visual Recognition Challenge (ILSVRC 2012)
  - Error rate – 15.4% (the next best entry was at 26.2%)



Imagenet classification with deep convolutionalneural networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012

Today: CNNs are everywhere

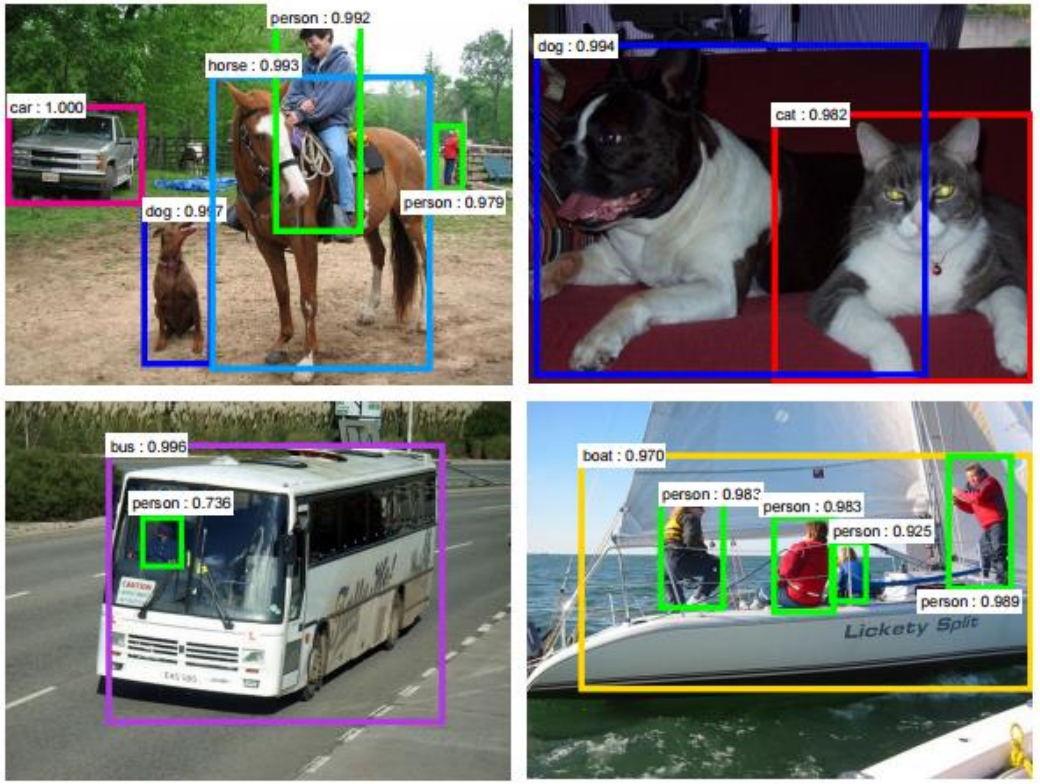
Classification



Source : http://karpathy.github.io

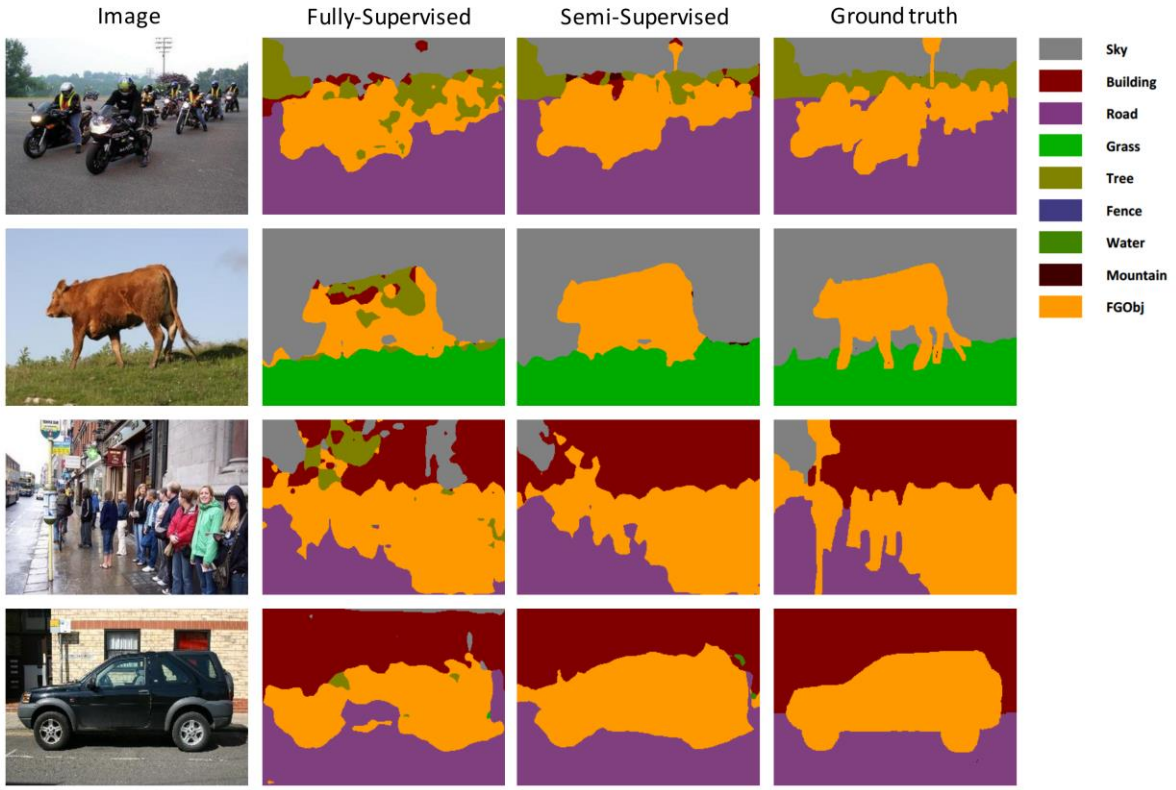
Today: CNNs are everywhere

Object detection



Faster R-CNN: Ren, He, Girshick, Sun 2015

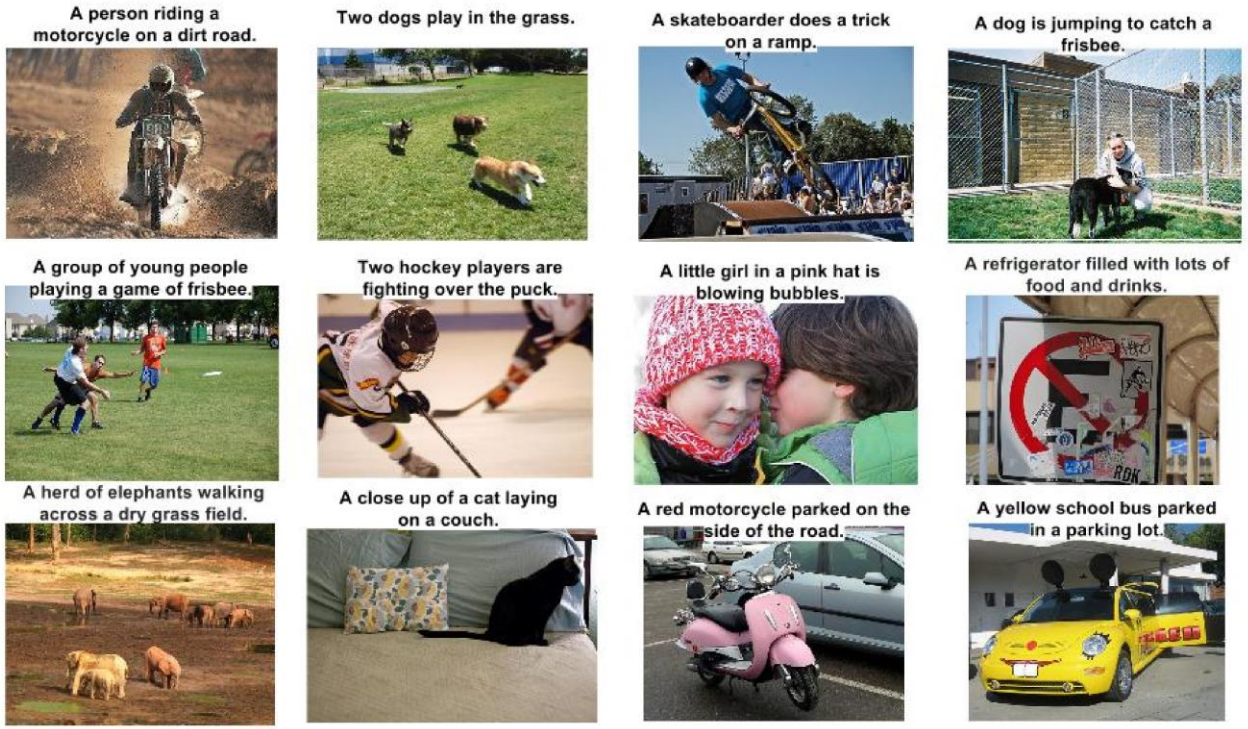
Semantic Segmentation



Semantic Segmentation Using GAN, Nasim, Concetto, and Mubarak, 2017.

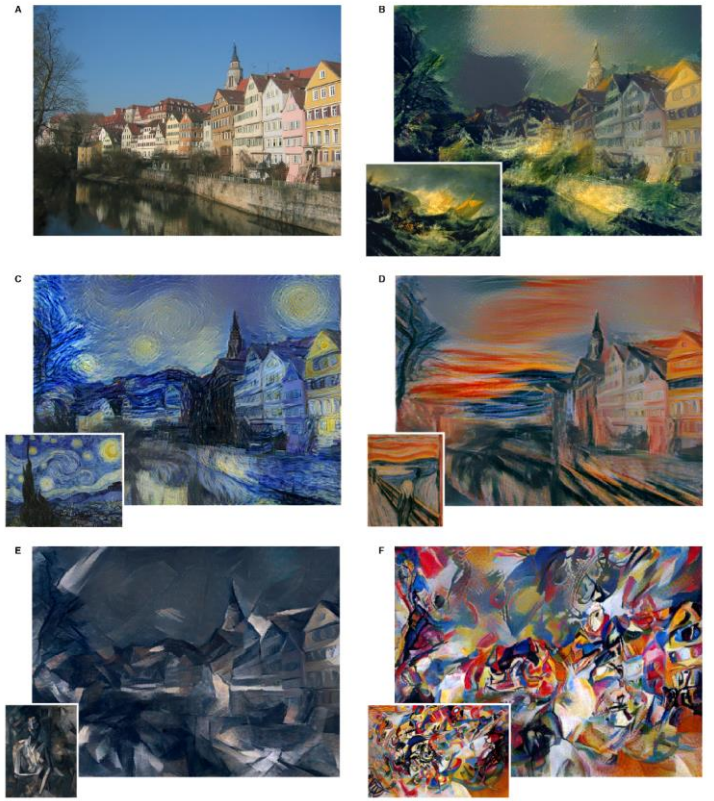
Today: CNNs are everywhere

Image captioning



"Show and tell: A neural image caption generator."
Vinyals, Oriol, et al. CVPR 2015.

Style transfer

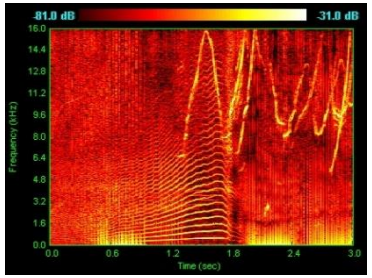


A Neural Algorithm of Artistic Style
L. Gatys et al. 2015).

CNN – Not just images

- Natural Language Processing (NLP)
  - Text classification
  - Word to vector
- Audio Research
  - Speech recognition
  - Can be represented as spectrograms

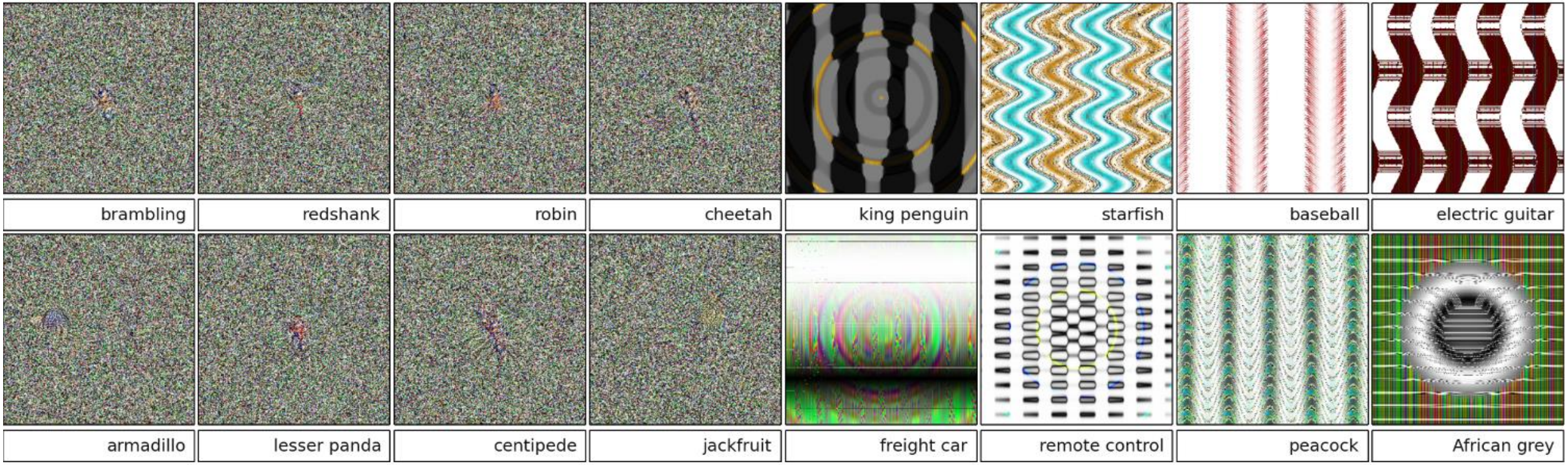
```
I = [0 1 0 1 1 0]
Love = [1 0 1 0 0 0]
NLP = [0 1 0 1 0 0]
And = [1 0 1 0 0 0]
Like = [1 0 0 0 0 1]
Dogs = [0 0 0 0 1 0]
```



- Converting data to a matrix (2-D) format
  - 1D convolution – Audio, EEG, etc.
  - 3D convolution - Videos

https://adeshpande3.github.io

CNN – Is it perfect?

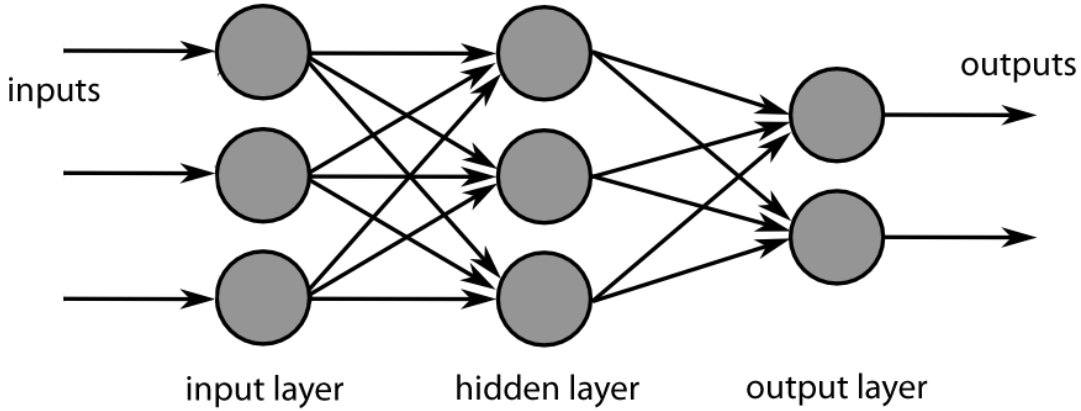


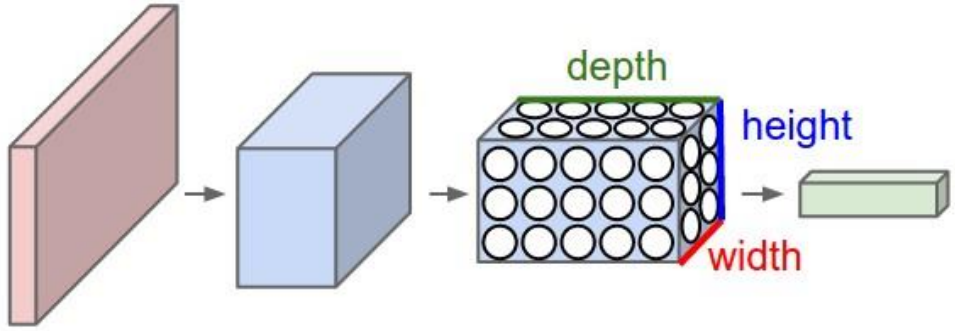
Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: (CVPR '15), IEEE, 2015

Convolutional Neural Network

Neural Network vs CNN

- Neural Network
  - Fully connected layers
- Image as input in neural network
  - Size of feature vector = HxWxC
  - For 256x256 RGB image
    - 196608 dimensions
- CNN - Special type of neural network
  - Operate with volume of data
  - Weight sharing in form of kernels

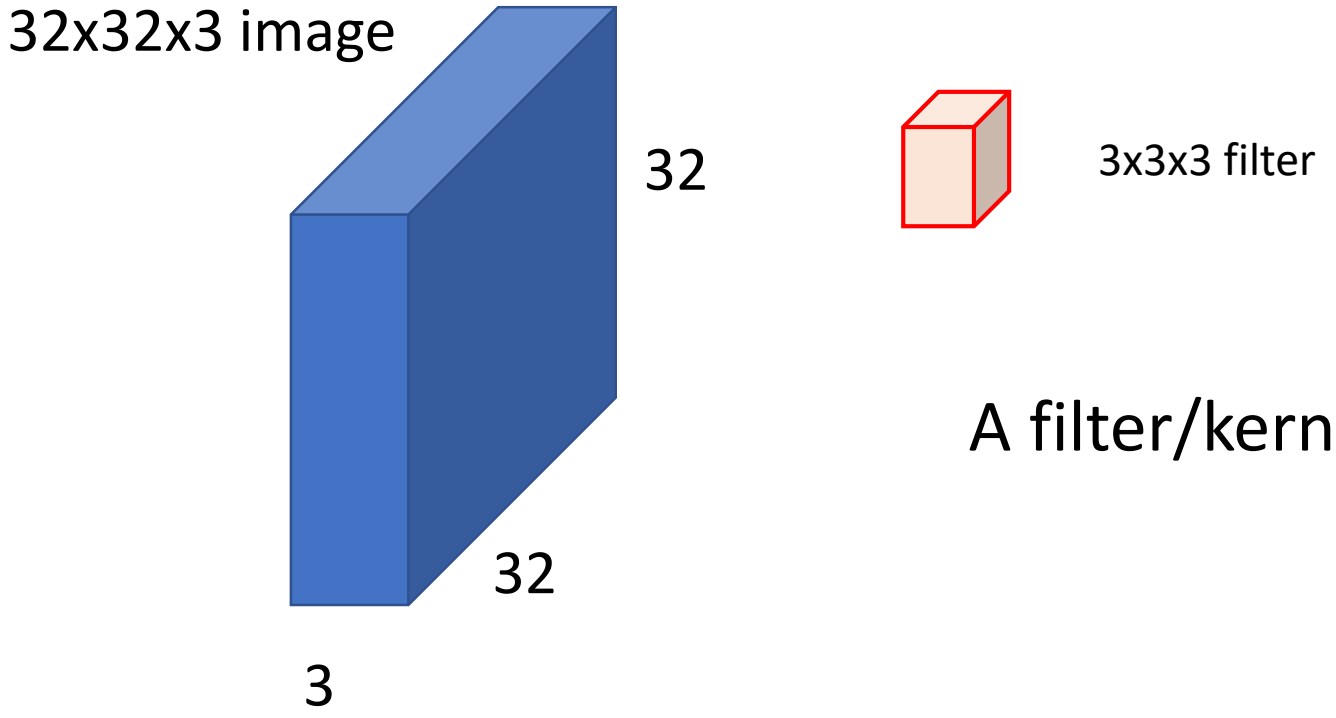




Source: http://cs231n.github.io

Convolution

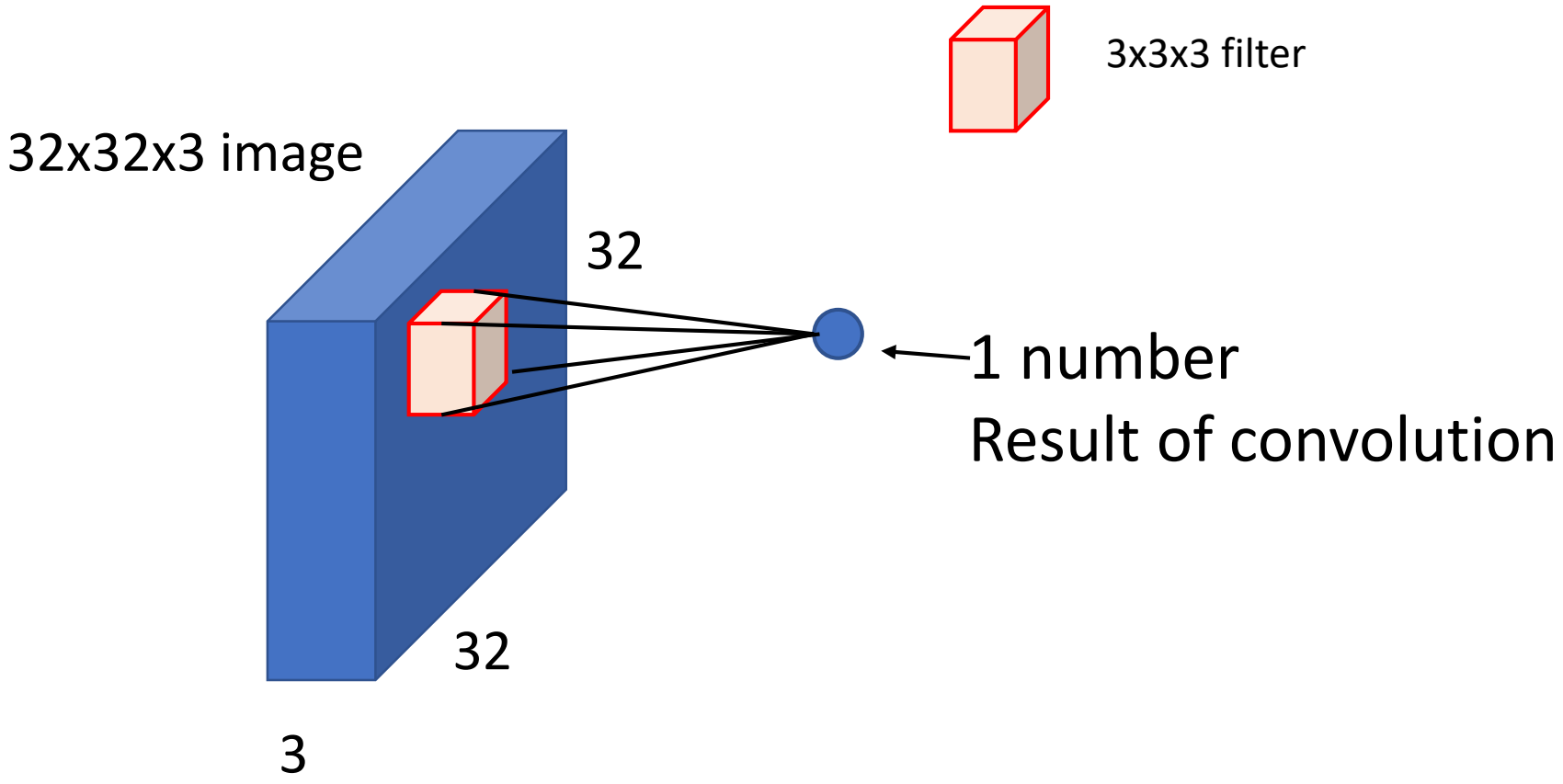
- Core building block of a CNN
  - Spatial structure of image is preserved



A filter/kernel is **convolved** with the image

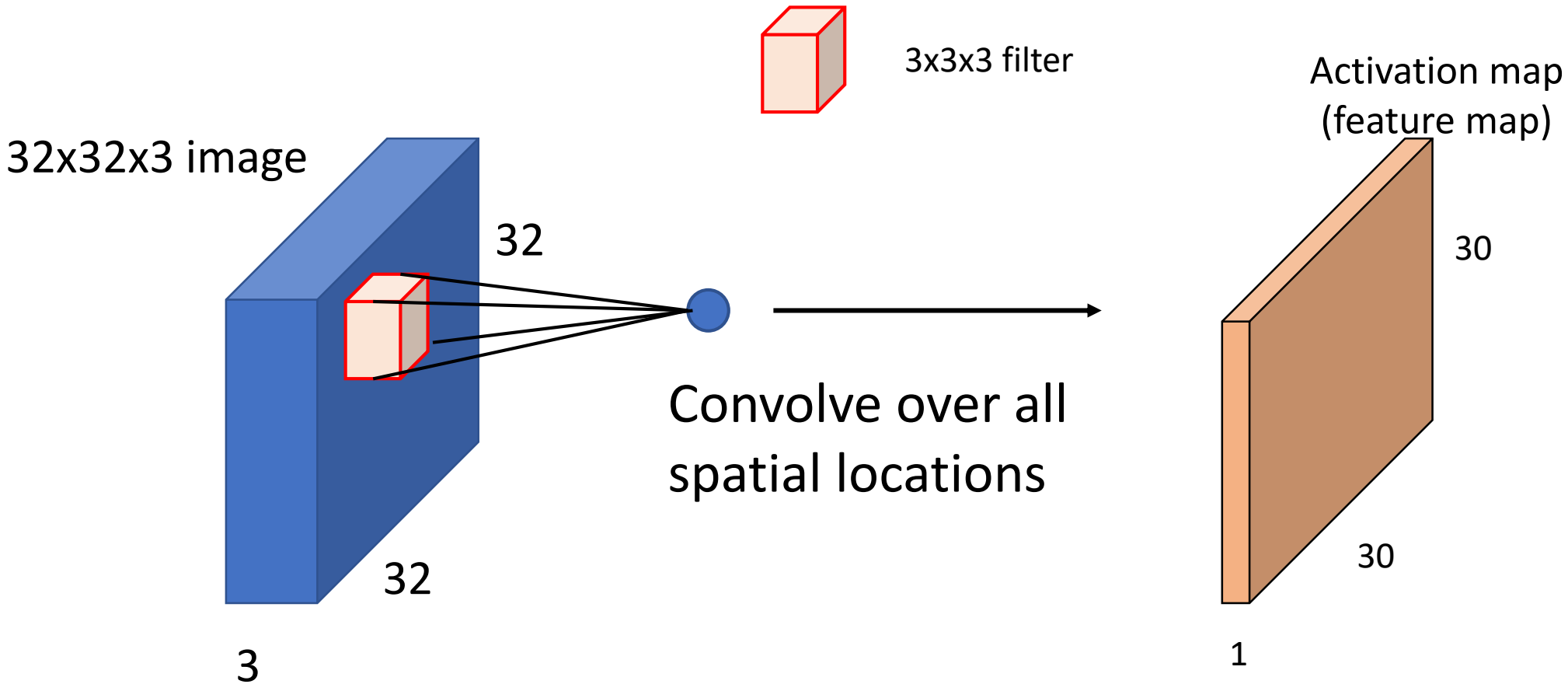
Convolution

- Convolution at one spatial location



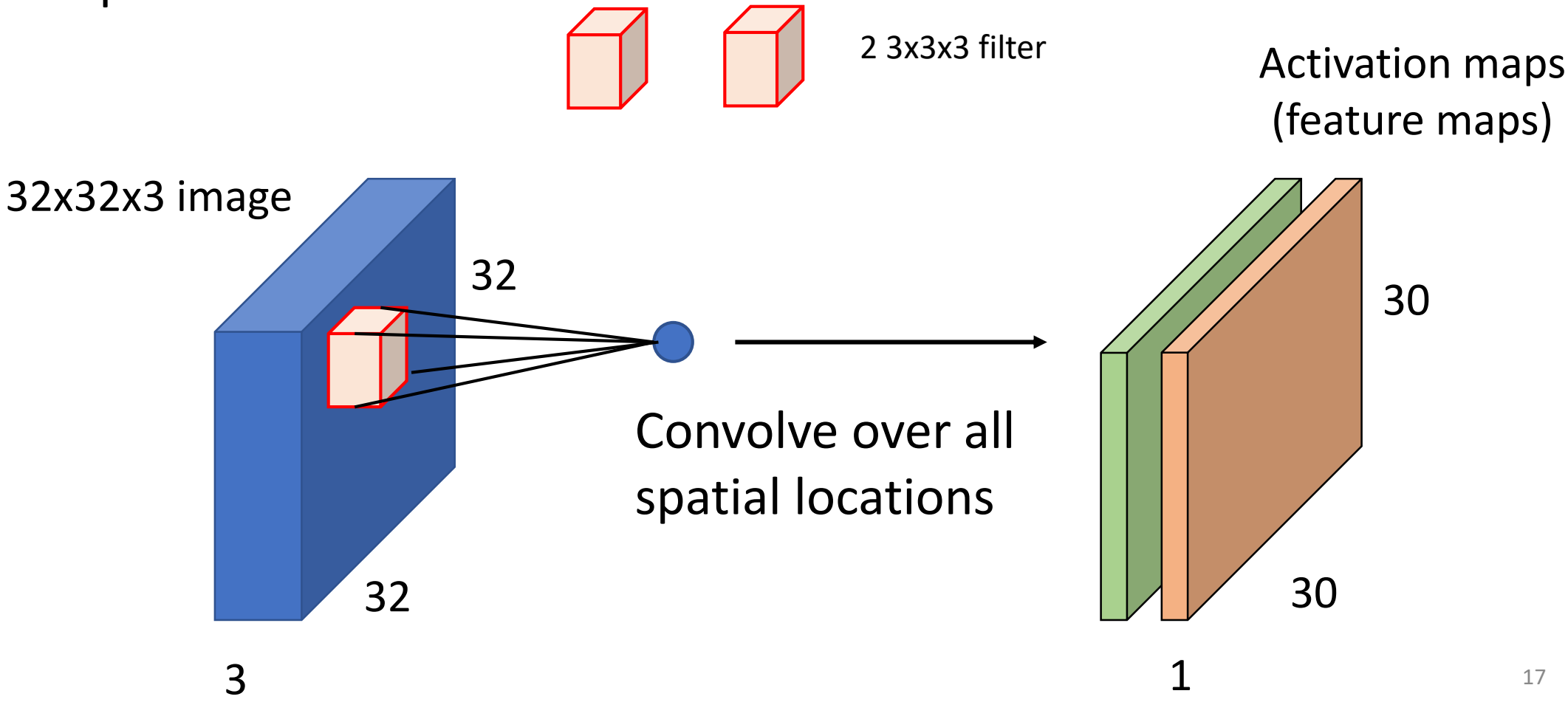
Convolution

- Convolution over whole image



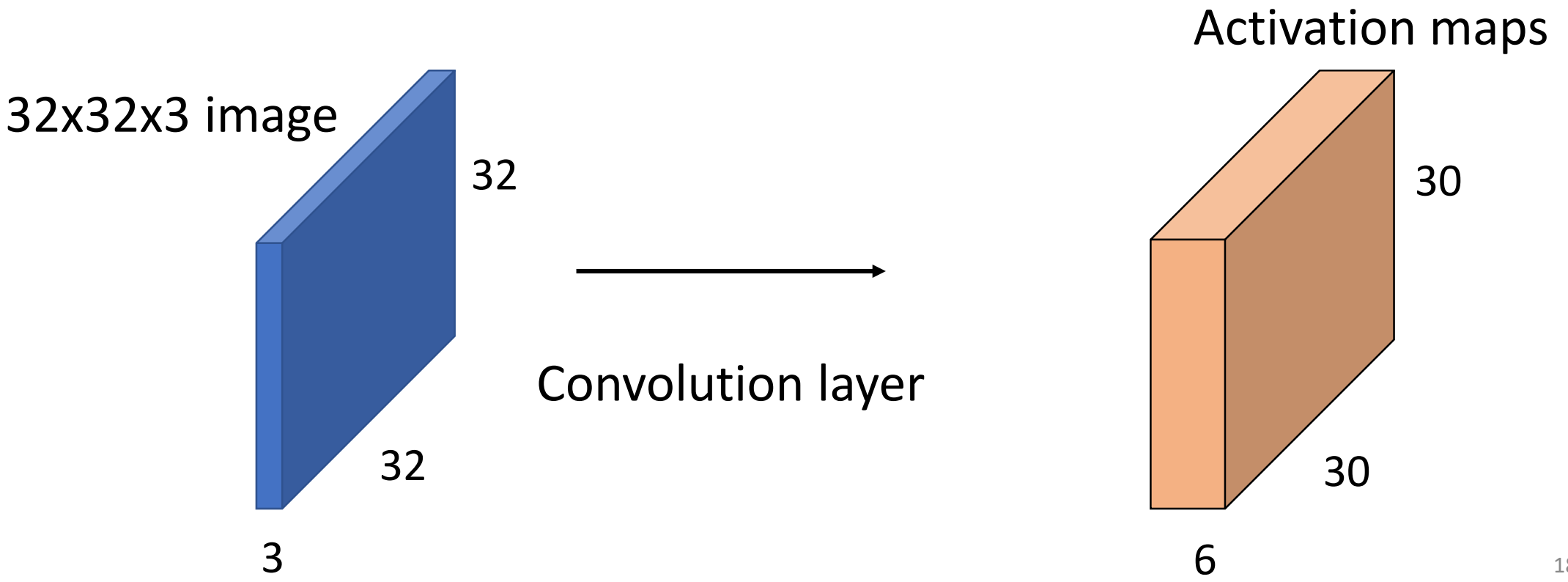
Convolution

- Multiple filters



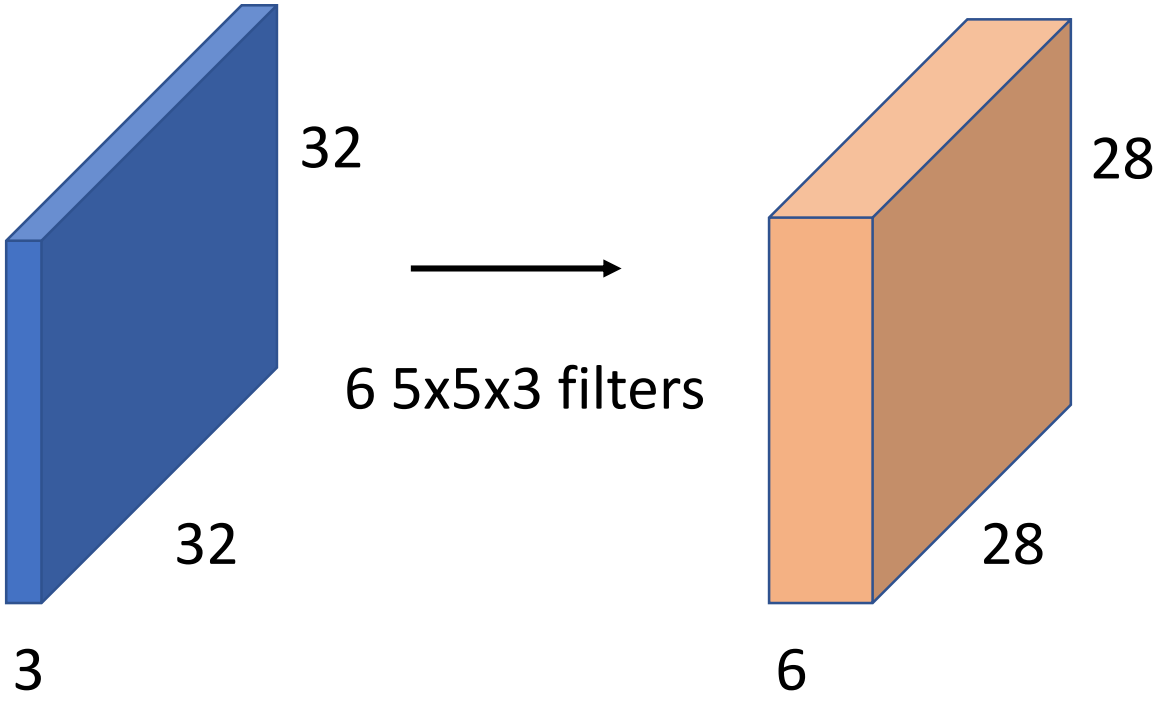
Convolution

- One convolution layer
  - 6 3x3x3 kernels



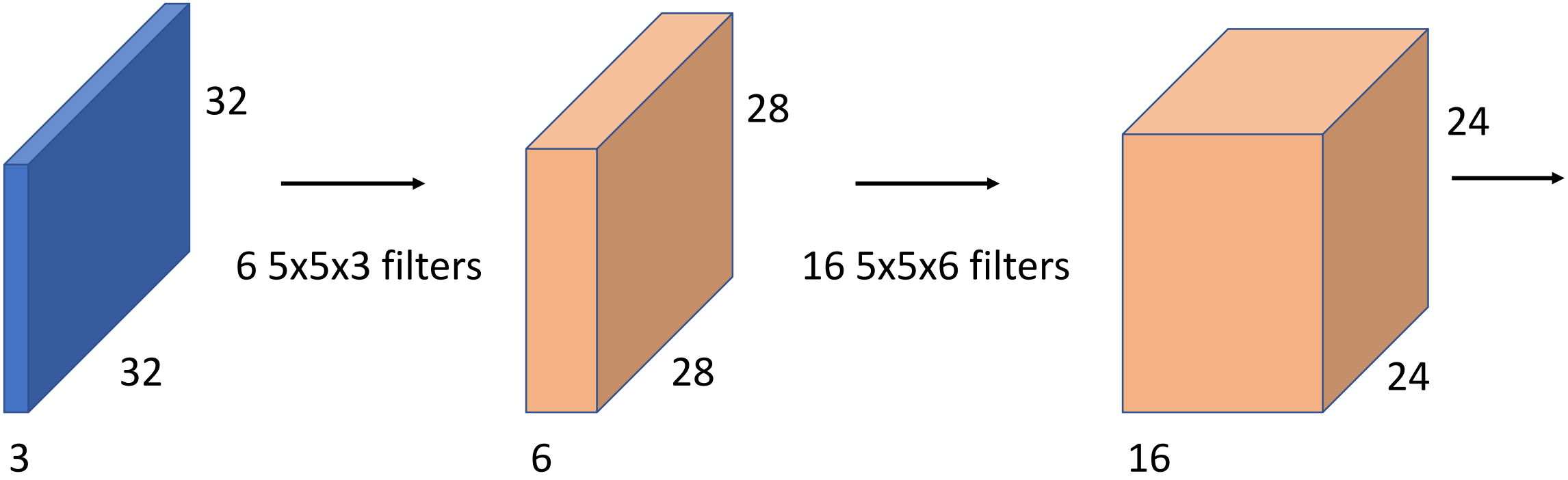
Convolutional Network

- Convolution network is a sequence of these layers



Convolutional Network

- Convolution network is a sequence of these layers



Convolution Operation

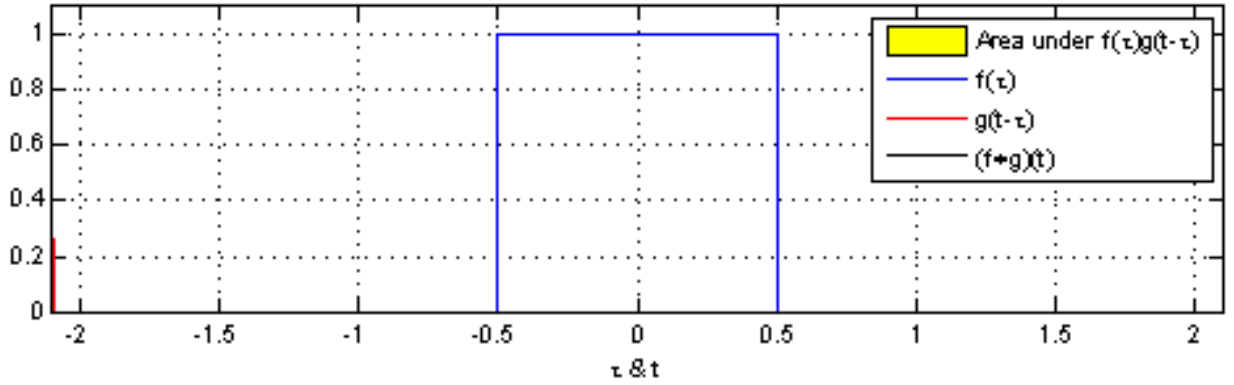
- Convolution of two functions f and g

function f(t)



kernel g(t)





$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

For images we use 2D convolutions

Demo

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

Input image

filter

1	0	1
0	1	0
1	0	1

4		

output

Demo

filter

1	0	1
0	1	0
1	0	1

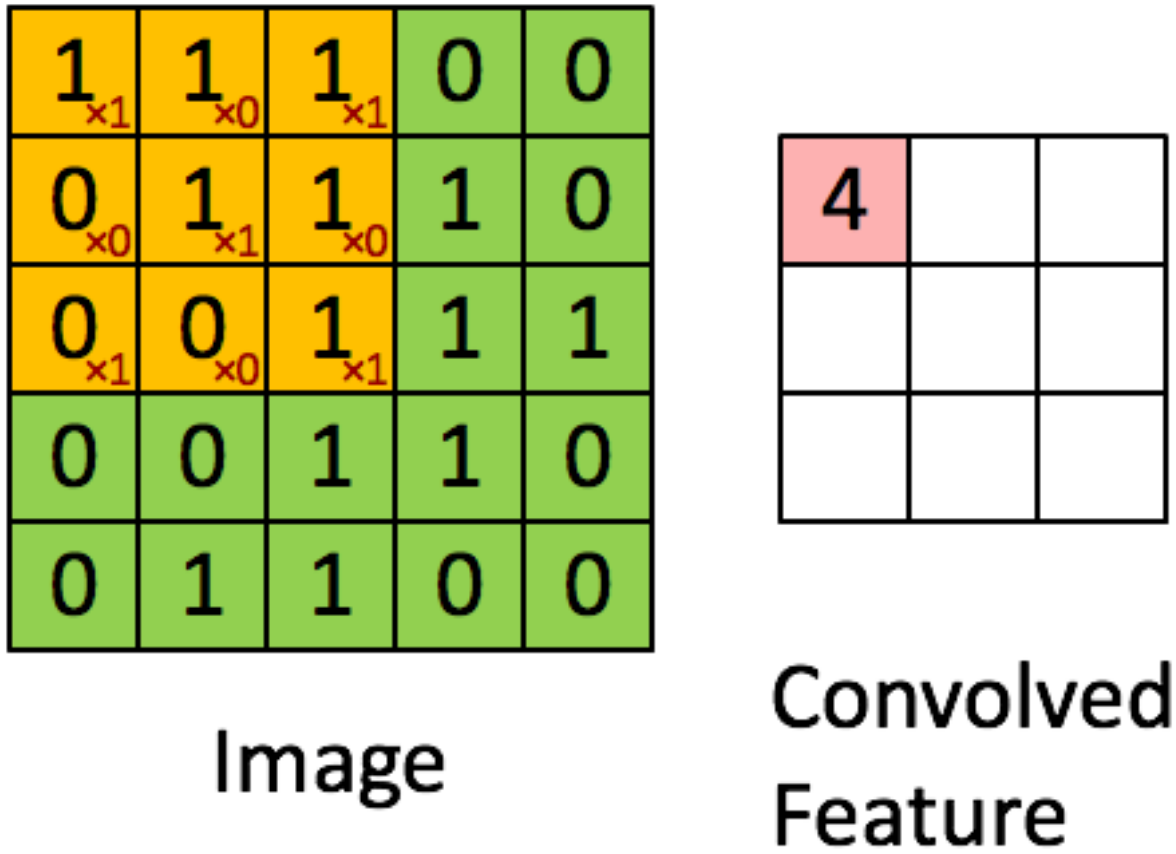
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	0	0	0

Input image

4	3	

output

Demo

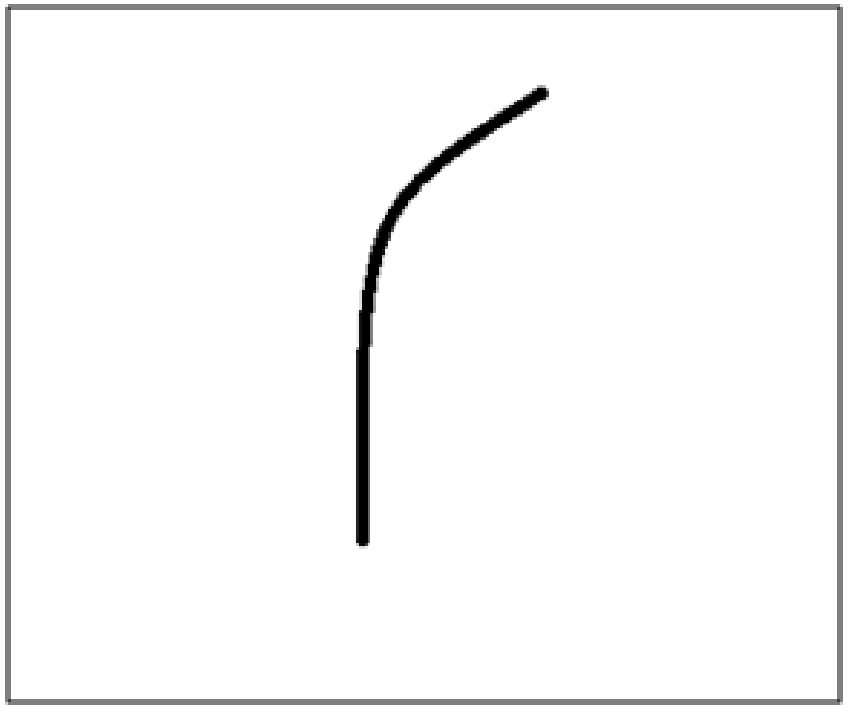


https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Convolution - Intuition

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

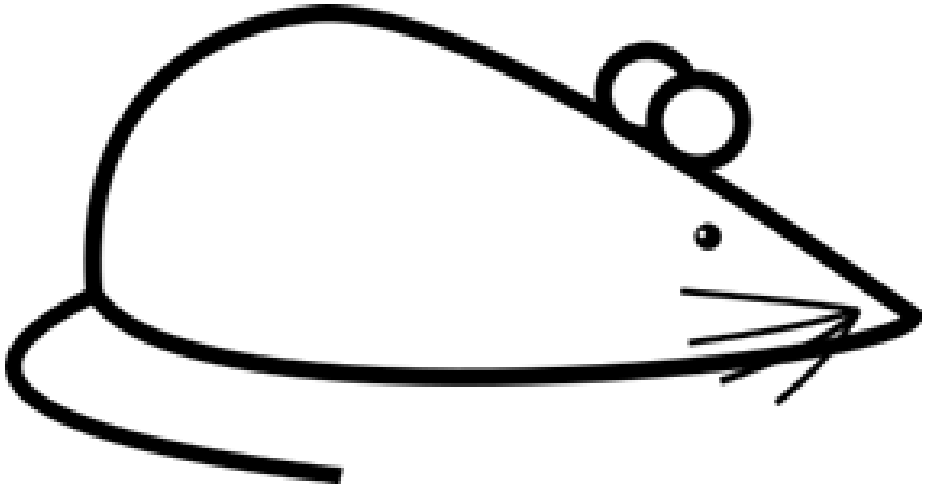
Pixel representation of filter



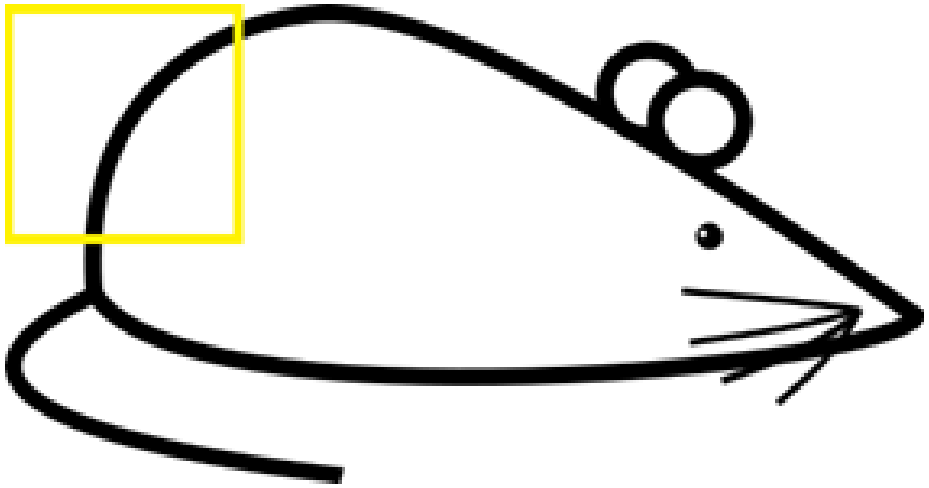
Visualization of a curve detector filter

Source : https://adeshpande3.github.io

Convolution - Intuition



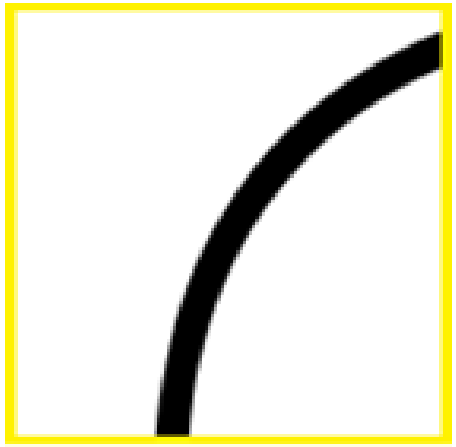
Original image



Visualization of the filter on the image

Source : https://adeshpande3.github.io

Convolution - Intuition



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

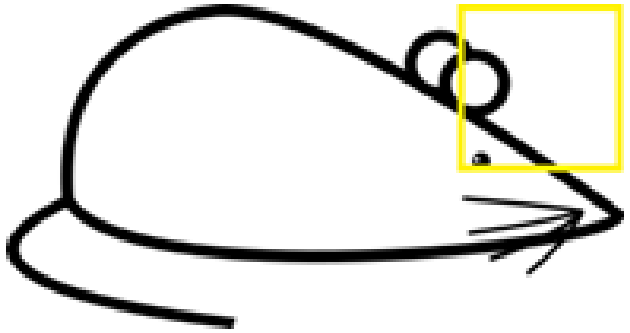
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Source : https://adeshpande3.github.io

Convolution - Intuition



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

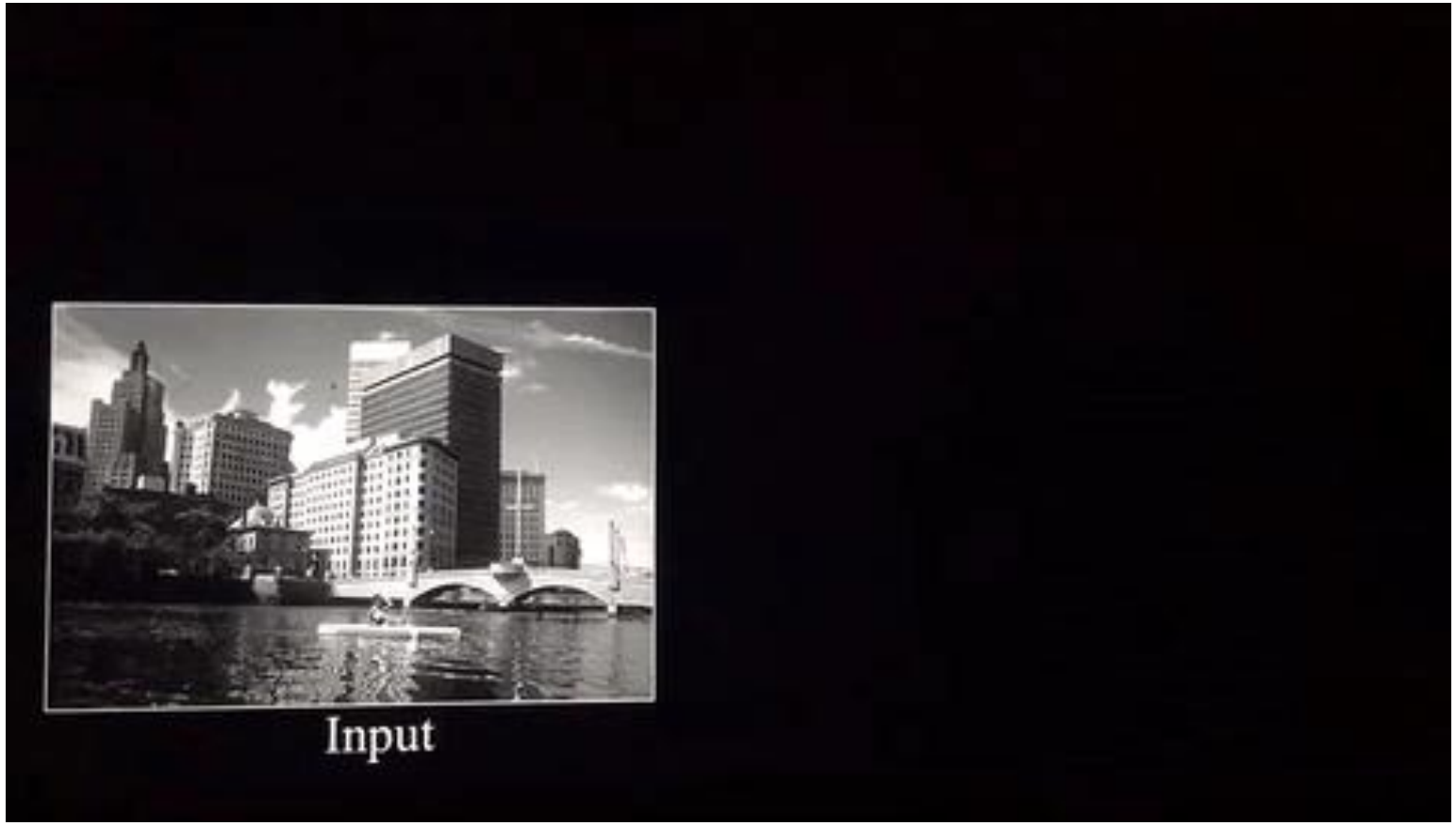
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

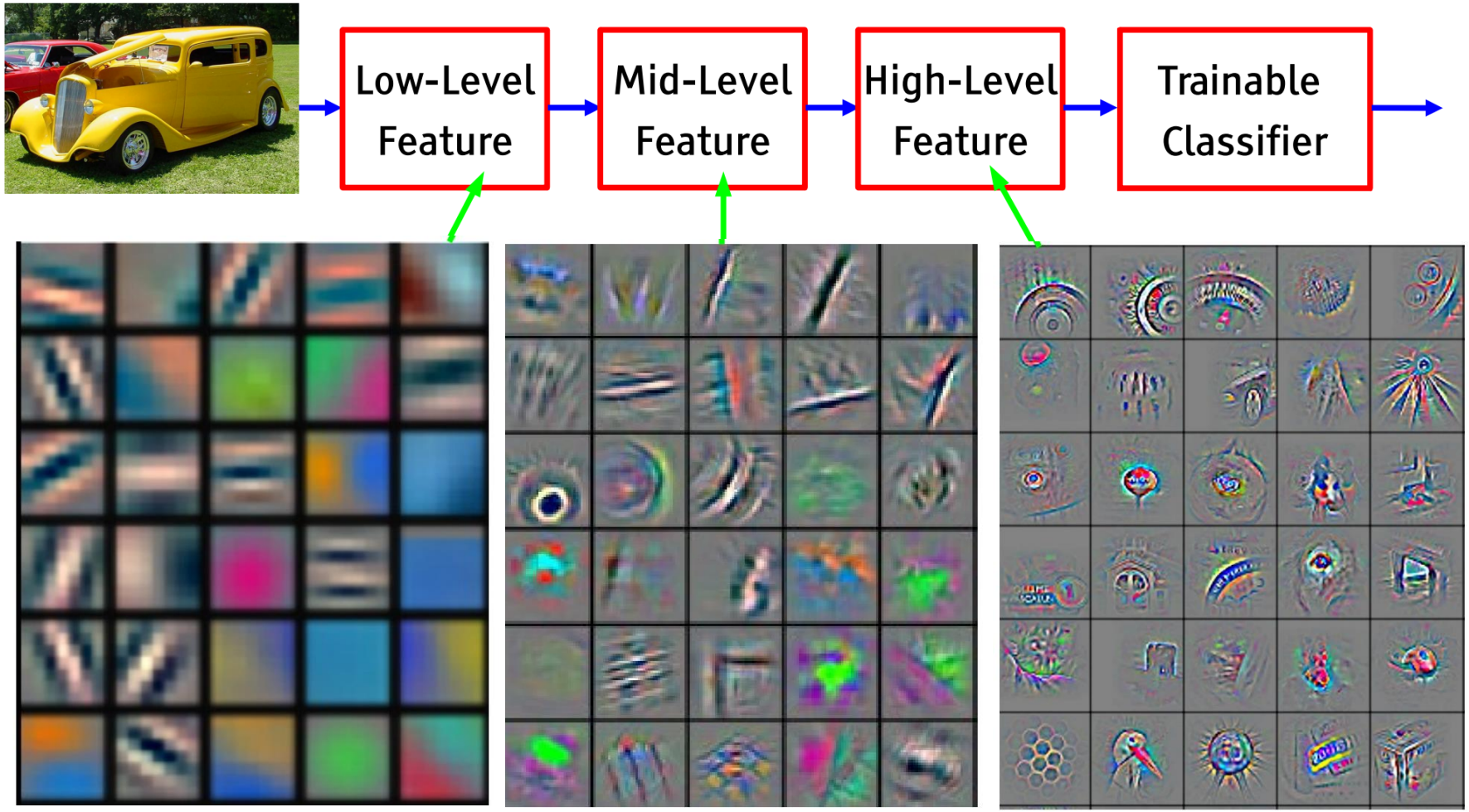
Source : https://adeshpande3.github.io

Convolution - Intuition



Source : https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

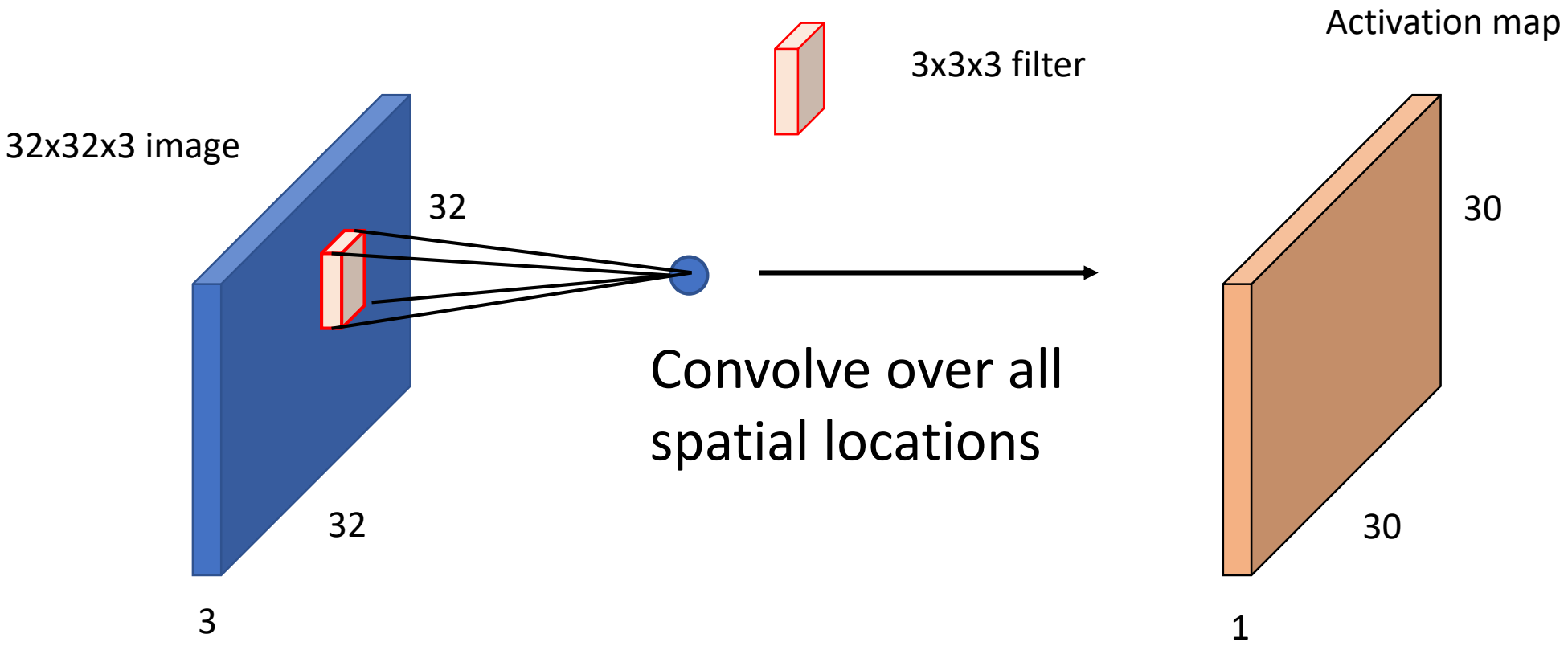
Visualizing Convolution



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

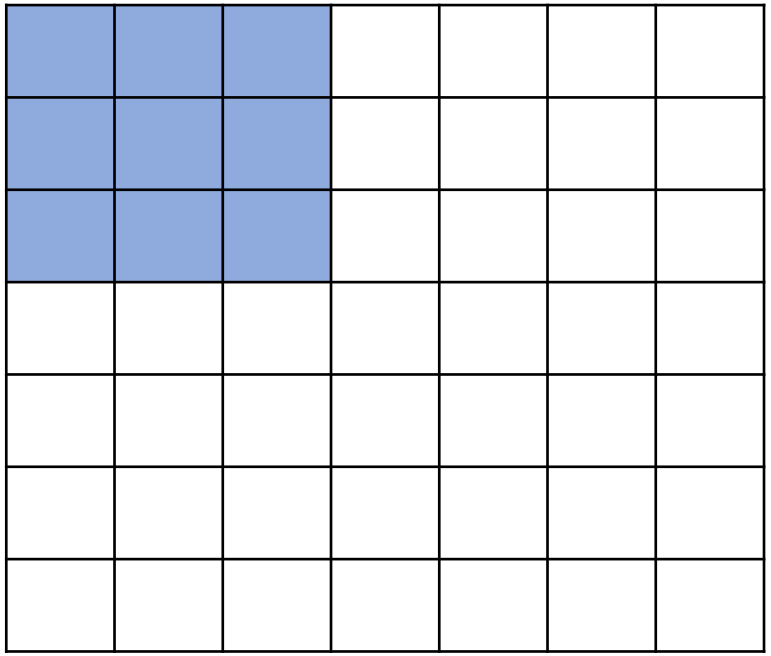
2D Convolution

- A closer look at spatial dimensions



2D Convolution

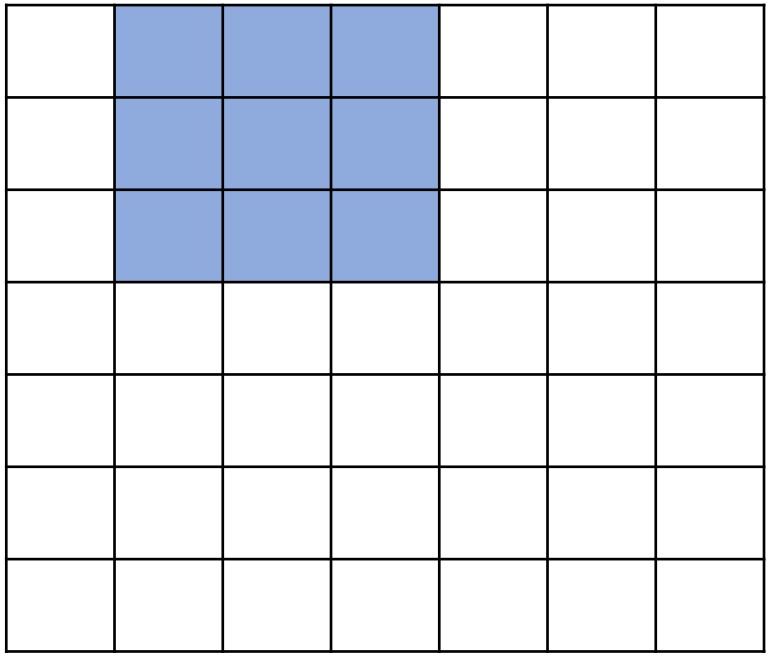
7x7 map



3x3 filter

2D Convolution

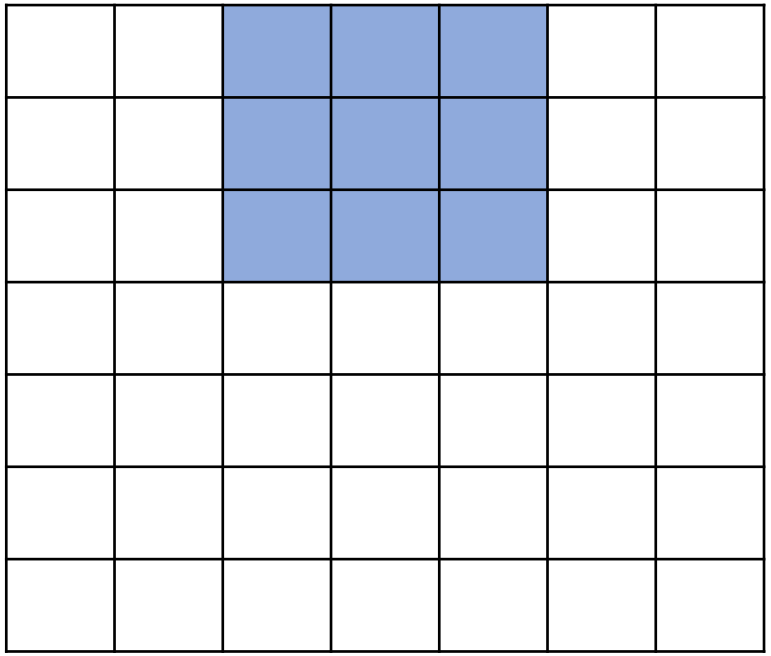
7x7 map



3x3 filter

2D Convolution

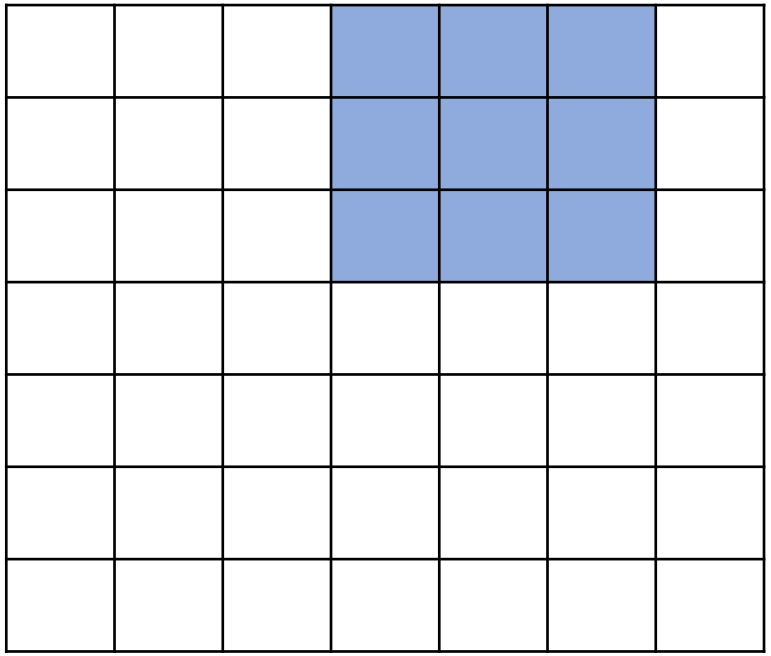
7x7 map



3x3 filter

2D Convolution

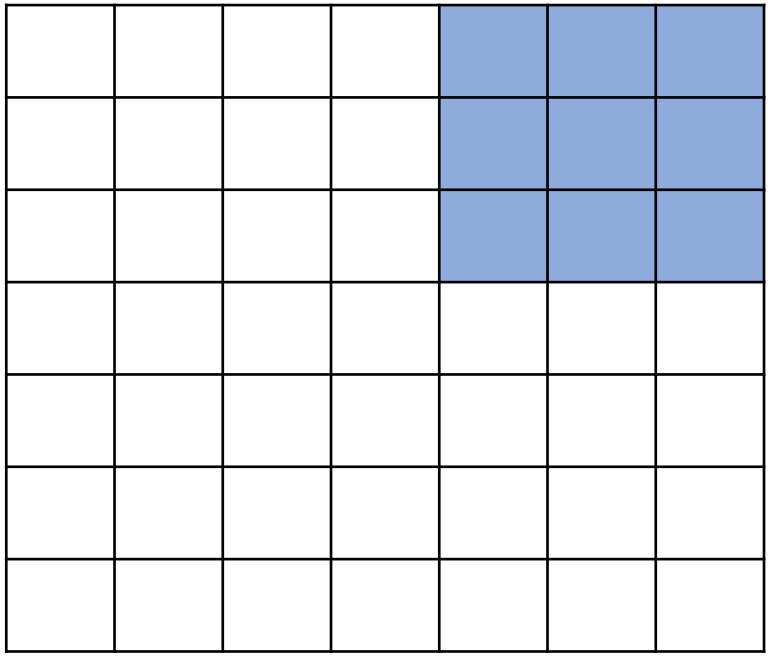
7x7 map



3x3 filter

2D Convolution

7x7 map



3x3 filter

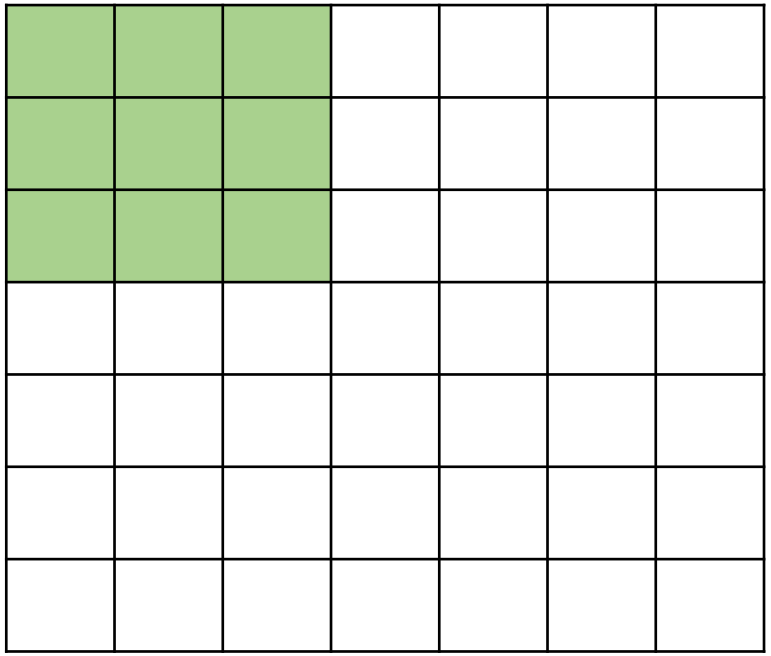
Output activation map 5x5

Output size
N-F+1
(7 – 3 + 1) = 5

N – input size
F – filter size

Stride

7x7 map

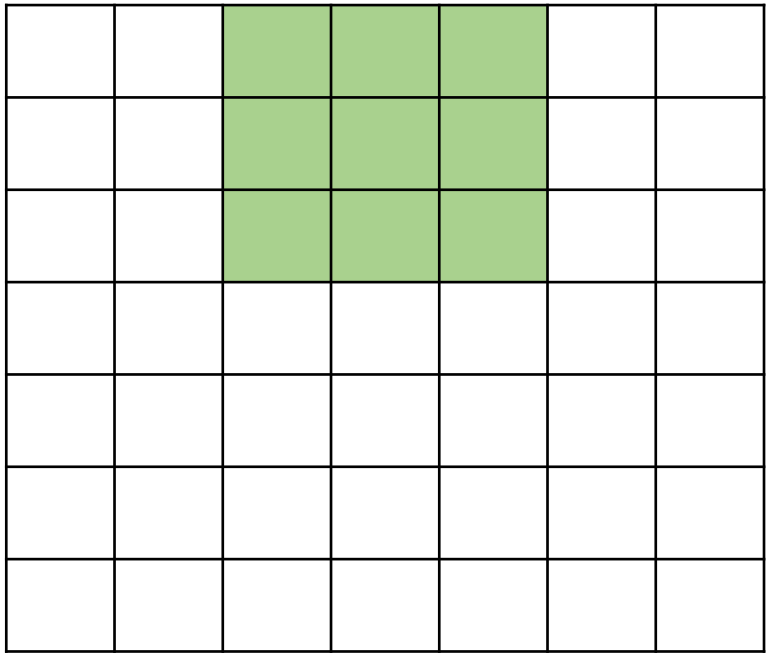


3x3 filter

Filter applied with stride 2

Stride

7x7 map

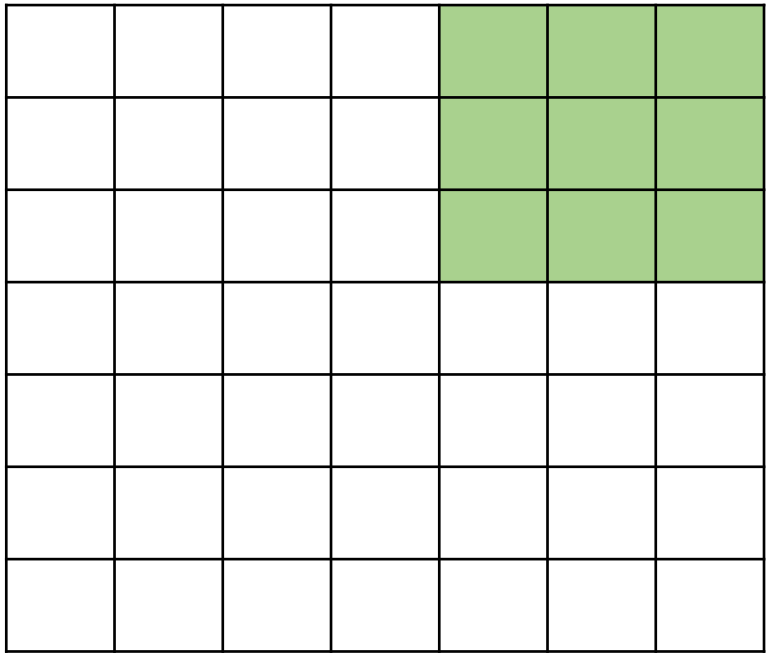


3x3 filter

Filter applied with stride 2

Stride

7x7 map



3x3 filter

Filter applied with stride 2

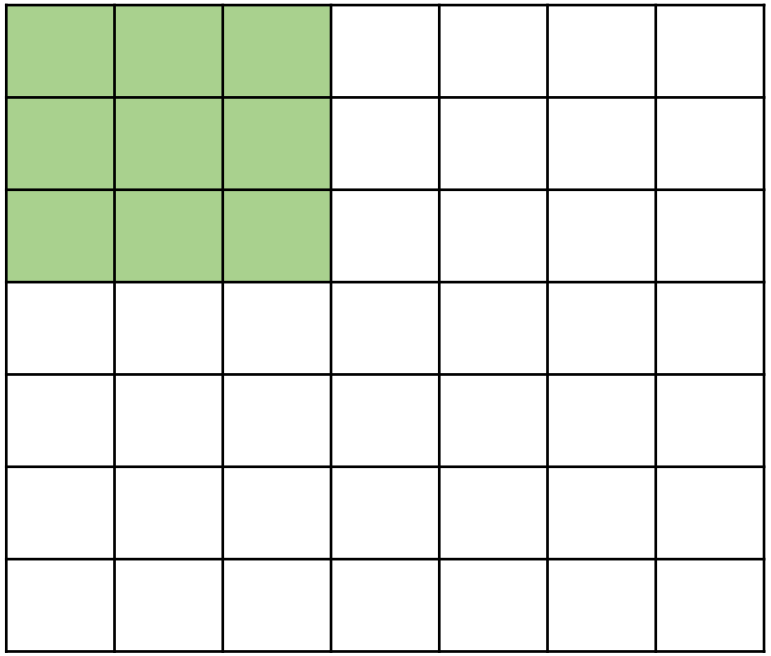
Activation map size 3x3

Output size
(7-3)/2 + 1 = 3

(N-F)/S + 1

Stride

7x7 map

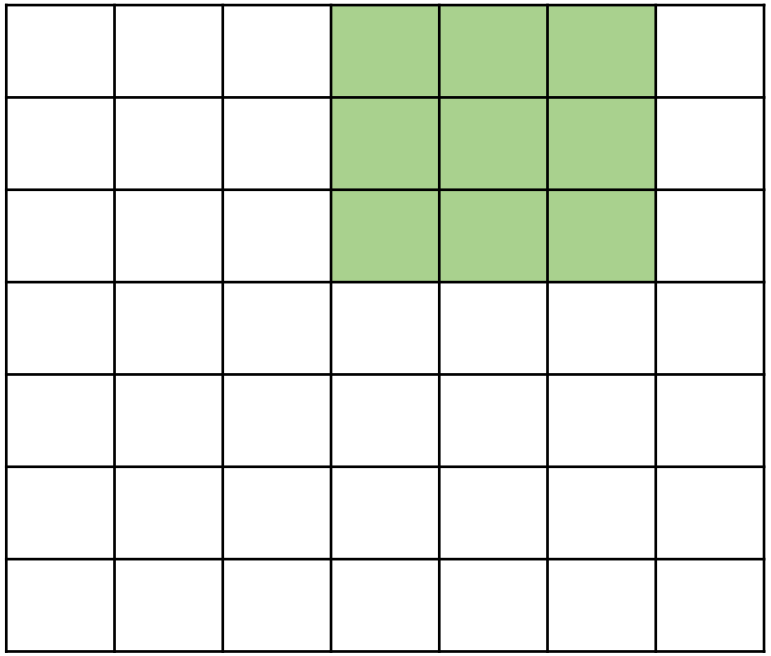


3x3 filter

Filter applied with stride 3

Stride

7x7 map



3x3 filter

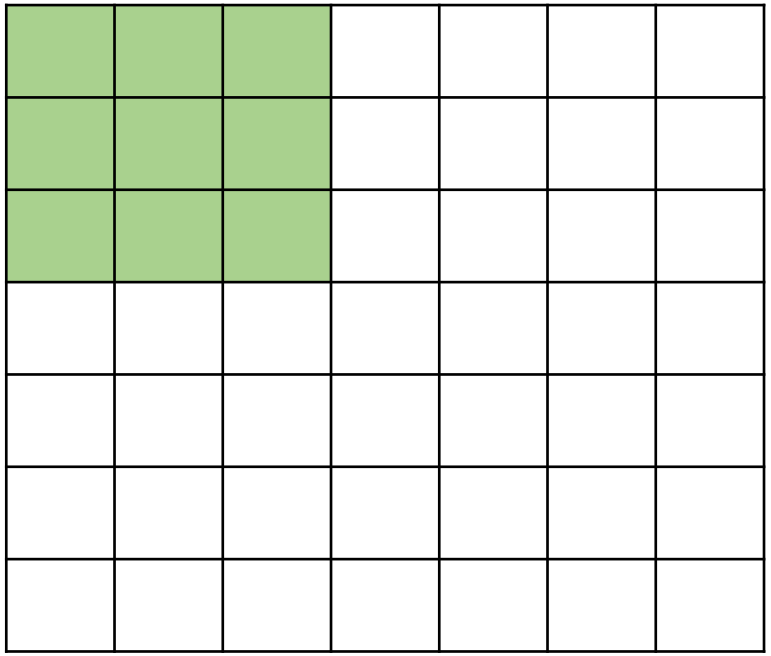
Filter applied with stride 3

Cannot cover perfectly

Not all parameters will fit

Stride

7x7 map



3x3 filter

Output size (N-F)/S + 1

N = 7, F = 3

Stride 1

(7-3)/1 + 1 => 5

Stride 2

(7-3)/2 + 1 => 3

Stride 3

(7-3)/3 + 1 => 2.33

Padding

- Zero padding in the input

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

For 7x7 input and 3x3 filter

If we have padding of one pixel

Output
7x7

Size (recall (N-F)/S+1)
(N-F+2P)/S + 1

Padding

- Zero padding in the input

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Common to see,
(F-1)/2 padding with stride 1 to preserve the map size

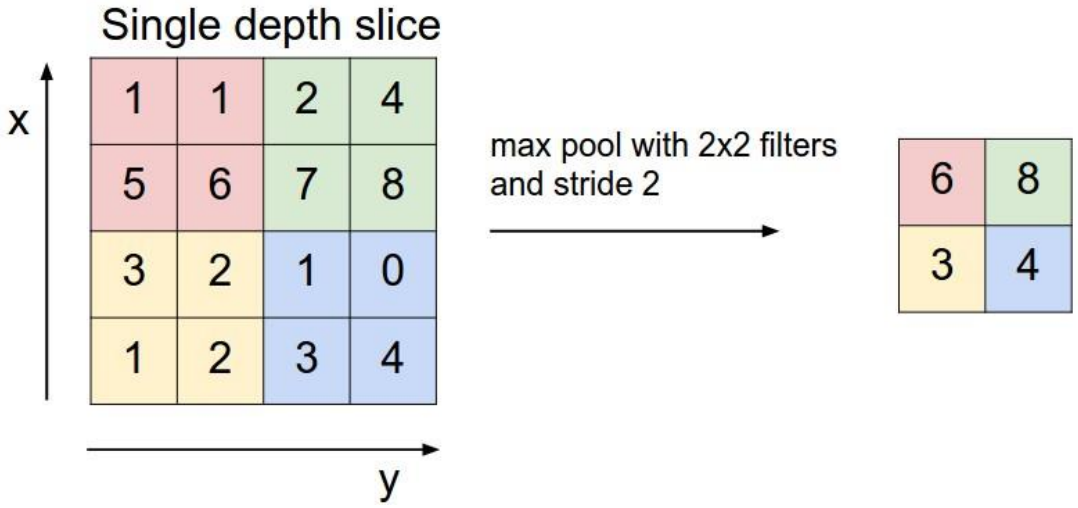
$N = (N-F+2P)/S + 1$

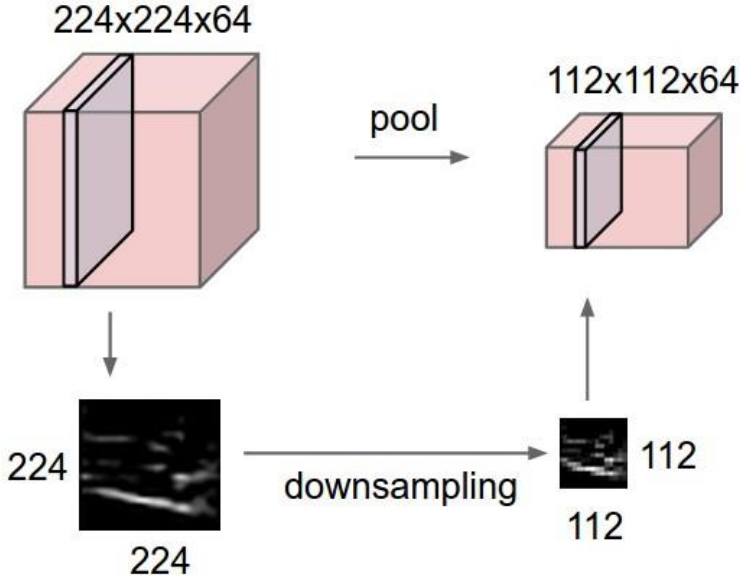
$\Rightarrow (N-1)S = N-F+2P$

$\Rightarrow P = (F-1)/2$

Pooling

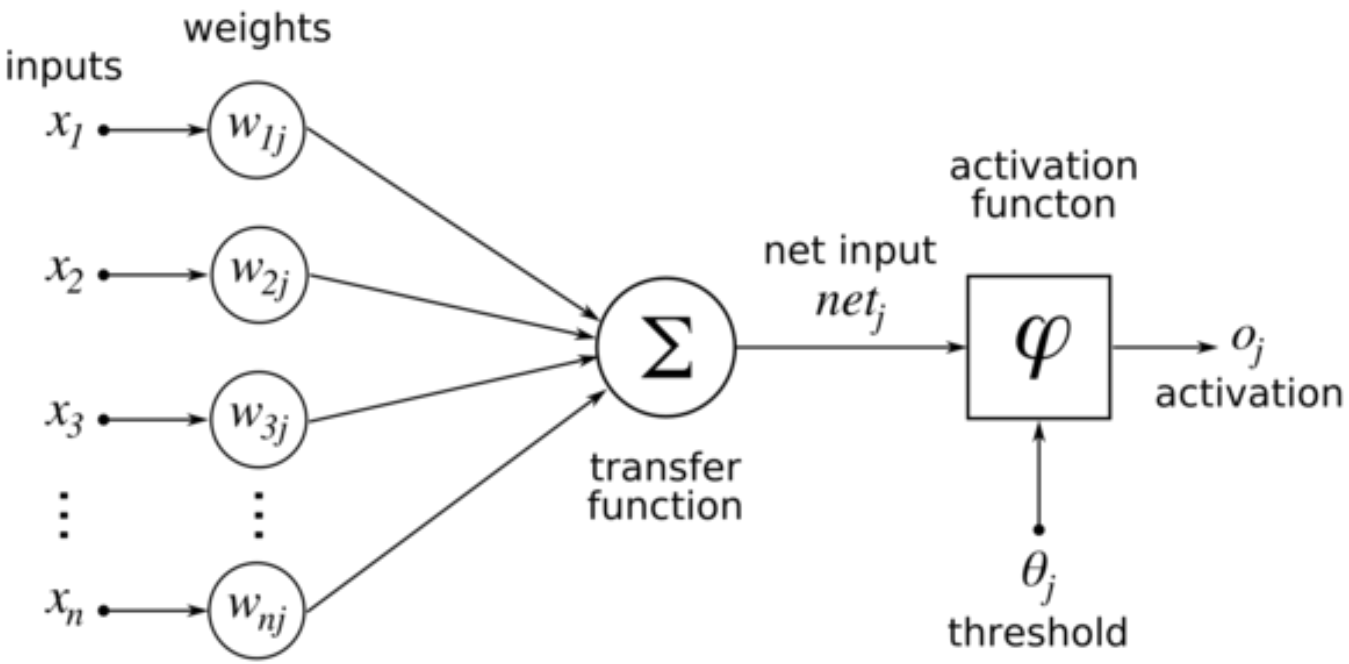
- Makes the representations smaller
- Operates over each activation map independently





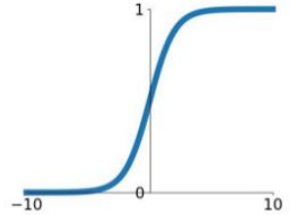
Source : http://cs231n.github.io

Activation Functions



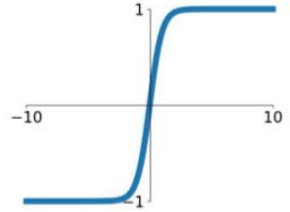
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



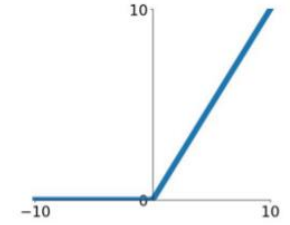
tanh

$\tanh(x)$

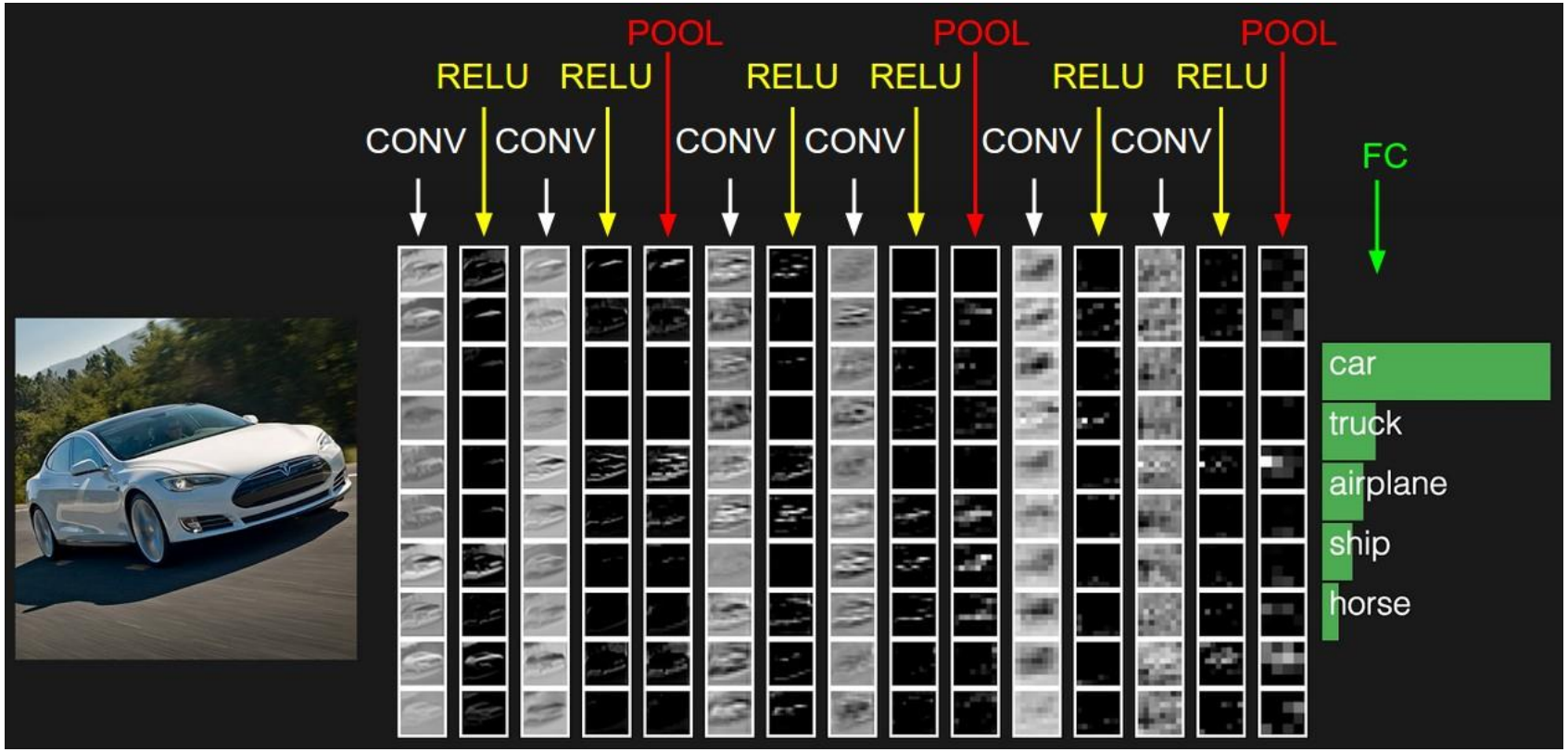


ReLU

$\max(0, x)$

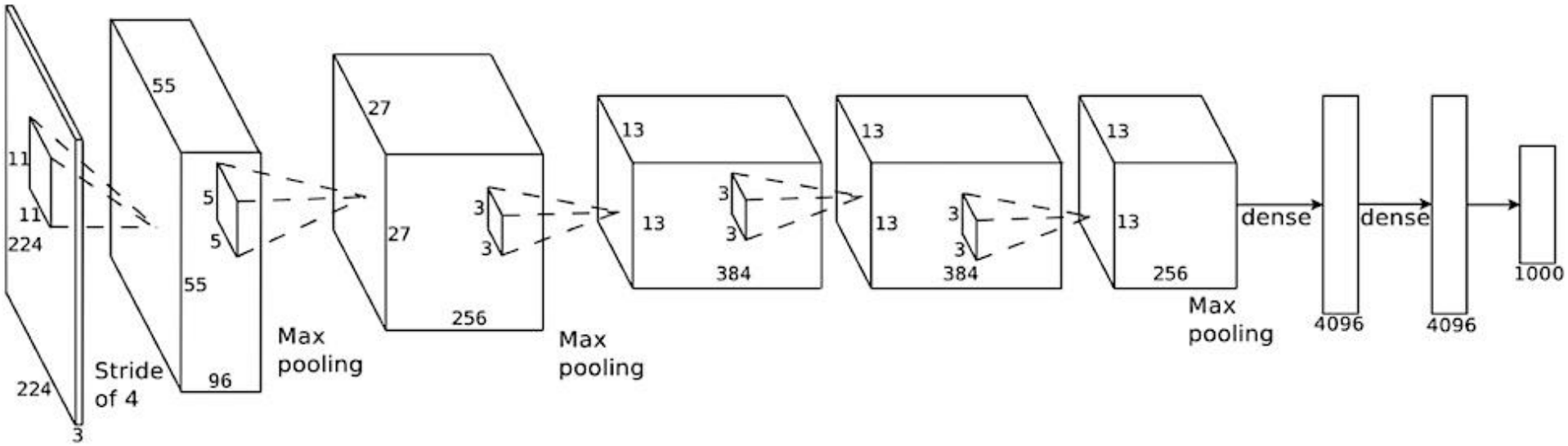


Visualizing CNN



Source : http://cs231n.github.io

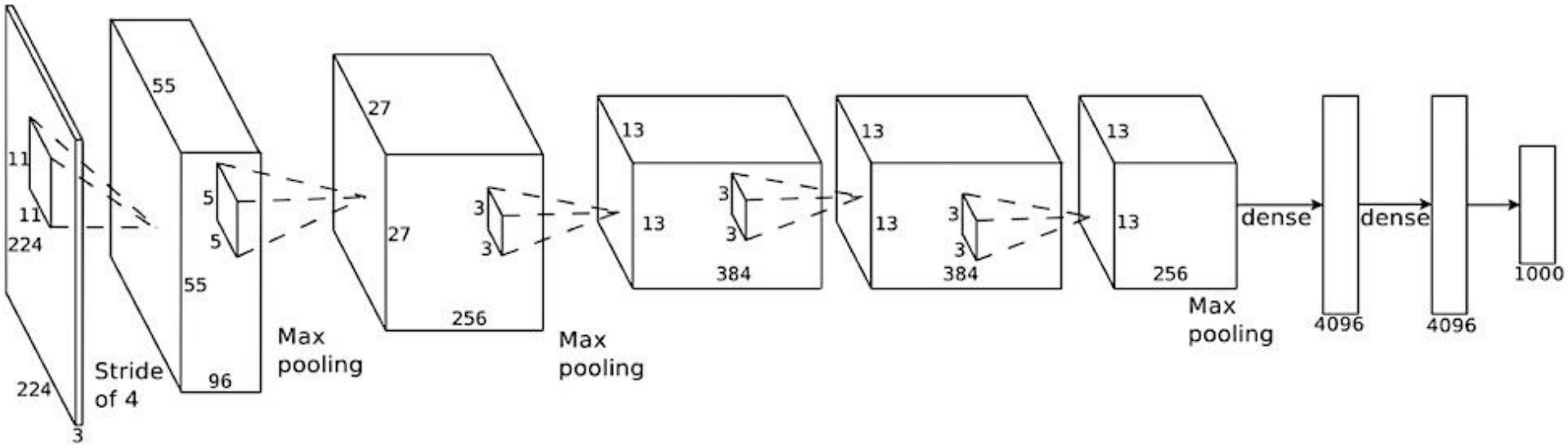
AlexNet : Network Size



- Input 227x227x3
- 5 convolution layers
- 3 dense layers
- Output 1000-D vector

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
MAX POOL3
FC6
FC7
FC8

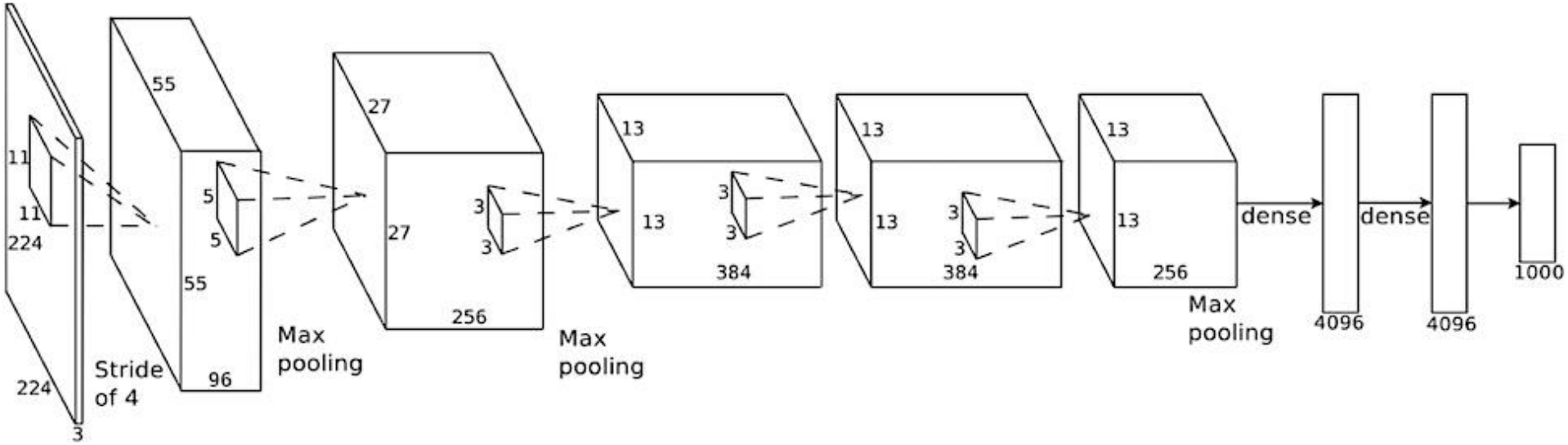
AlexNet : Network Size



CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
MAX POOL3
FC6
FC7
FC8

- Input: 227x227x3 images
- First layer (CONV1): 96 11x11 filters applied at stride 4
- What is the output volume size? (227-11)/4+1 = 55
- What is the number of parameters? 11x11x3x96 = 35K

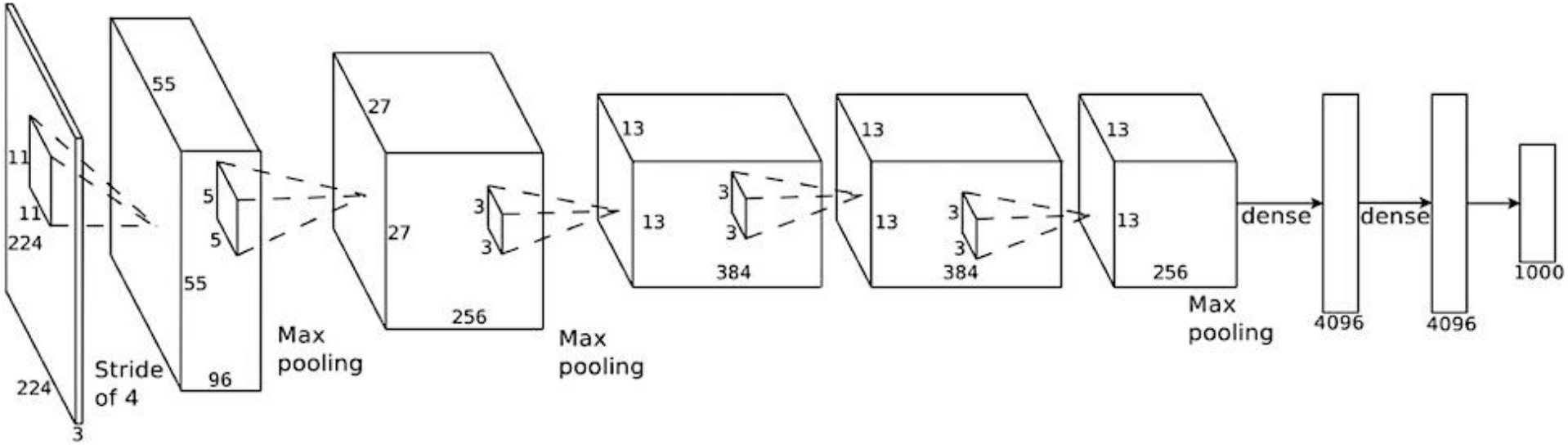
AlexNet : Network Size



CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
MAX POOL3
FC6
FC7
FC8

- After CONV1: 55x55x96
- Second layer (POOL1): 3x3 filters applied at stride 2
- What is the output volume size? (55-3)/2+1 = 27
- What is the number of parameters in this layer? 0

AlexNet : Network Size



- After POOL1: 27x27x96
- Third layer (NORM1): Normalization
- What is the output volume size? 27x27x96

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
MAX POOL3
FC6
FC7
FC8

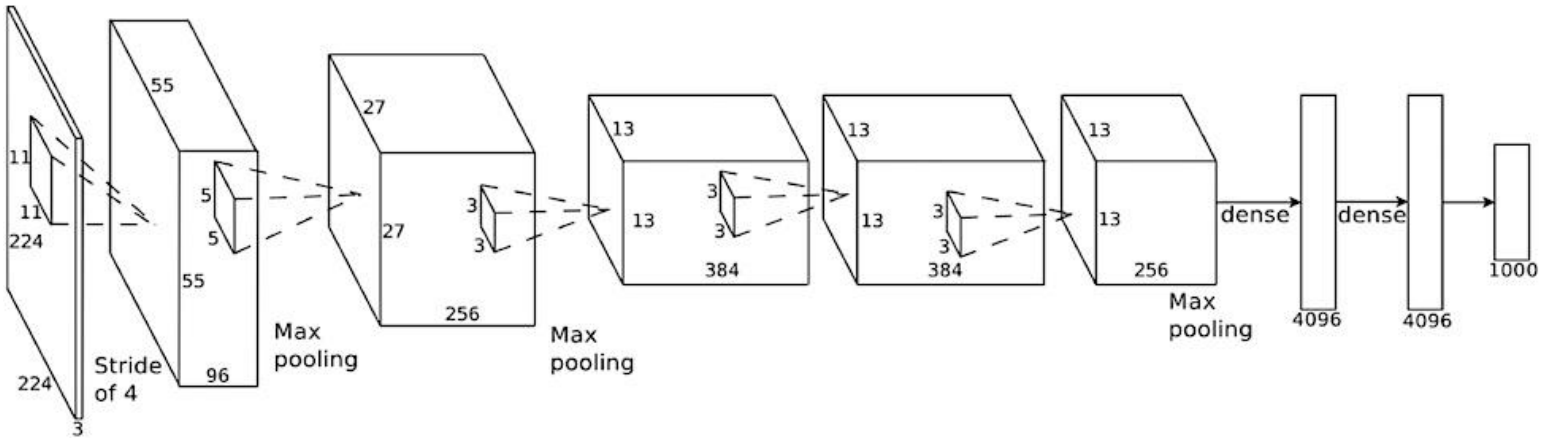
AlexNet : Network Size

1. [227x227x3] INPUT
2. [55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
3. [27x27x96] MAX POOL1: 3x3 filters at stride 2
4. [27x27x96] NORM1: Normalization layer
5. [27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
6. [13x13x256] MAX POOL2: 3x3 filters at stride 2
7. [13x13x256] NORM2: Normalization layer
8. [13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
9. [13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
10. [13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
11. [6x6x256] MAX POOL3: 3x3 filters at stride 2
12. [4096] FC6: 4096 neurons
13. [4096] FC7: 4096 neurons
14. [1000] FC8: 1000 neurons (class scores)

CONV1	35K
MAX POOL1	
NORM1	
CONV2	307K
MAX POOL2	
NORM2	
CONV3	884K
CONV4	1.3M
CONV5	442K
MAX POOL3	
FC6	37M
FC7	16M
FC8	4M

AlexNet - Training

- Parameters:
  - First use of ReLU
  - Dropout 0.5
  - Batch size 128
  - Optimizer SGD
  - Momentum 0.9
  - Learning rate 1e-2
  - Decay – lr reduced by 10 manually when val accuracy plateaus
  - L2 weight decay 5e-4



Part -II

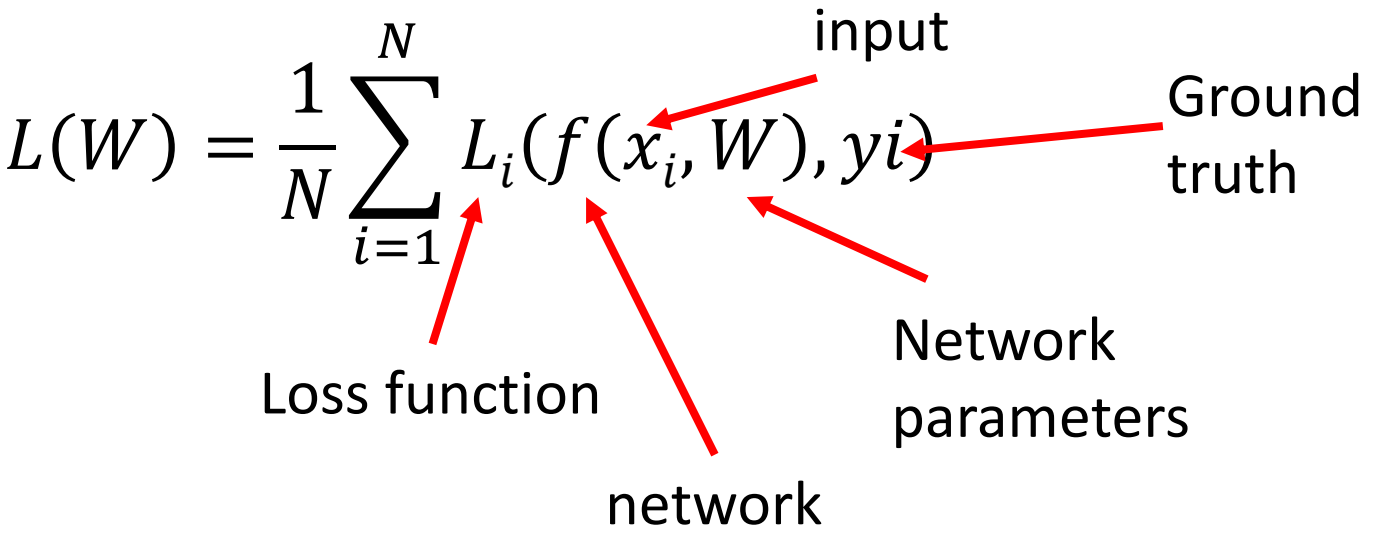
- Network training
  - Loss function
  - Optimization
- Issues
  - Overfitting
- Other network
  - CNN variants
  - Recurrent networks
- Generative adversarial network

Part -II

- Network training
  - Loss function
  - Optimization
- Issues
  - Overfitting
- Other network
  - CNN variants
  - Recurrent networks
- Generative adversarial network

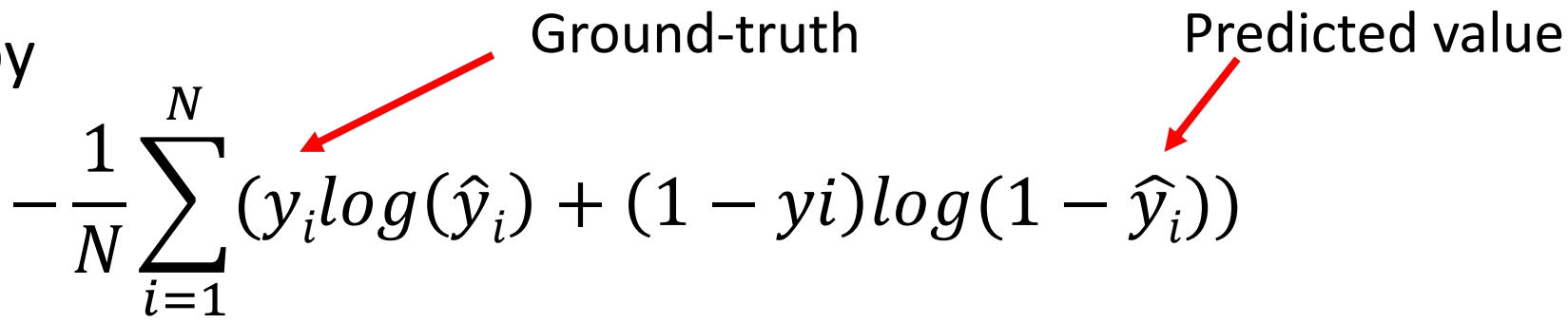
Loss Function

- 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



Loss Functions

- Cross entropy



- Kullback–Leibler divergence (KL divergence)

$$KL(P||Q) = \sum_i P(i) log(\frac{P(i)}{Q(i)})$$

Part -II

- Network training
  - Loss function
  - Optimization
- Issues
  - Overfitting
- Other network
  - CNN variants
  - Recurrent networks
- Generative adversarial network

Network Training – Minimize Cost

- Gradient Descent
  - Way to minimize a cost function

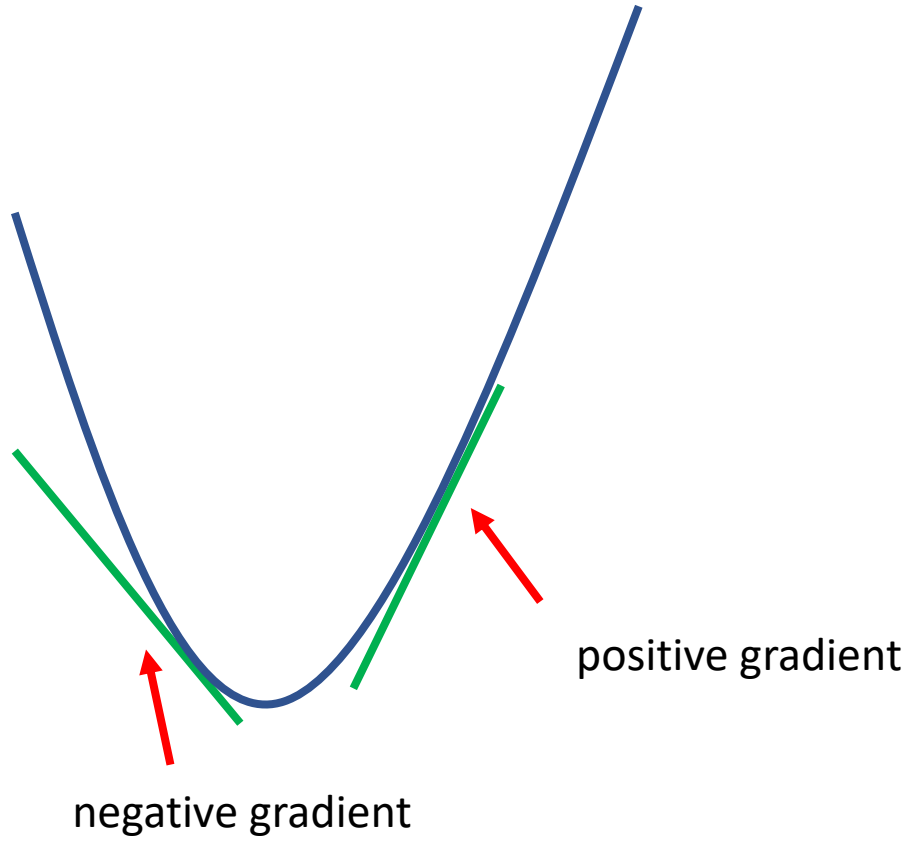
$$w := w - \alpha \nabla Qi(w)$$

cost function

weight update

learning rate

gradient

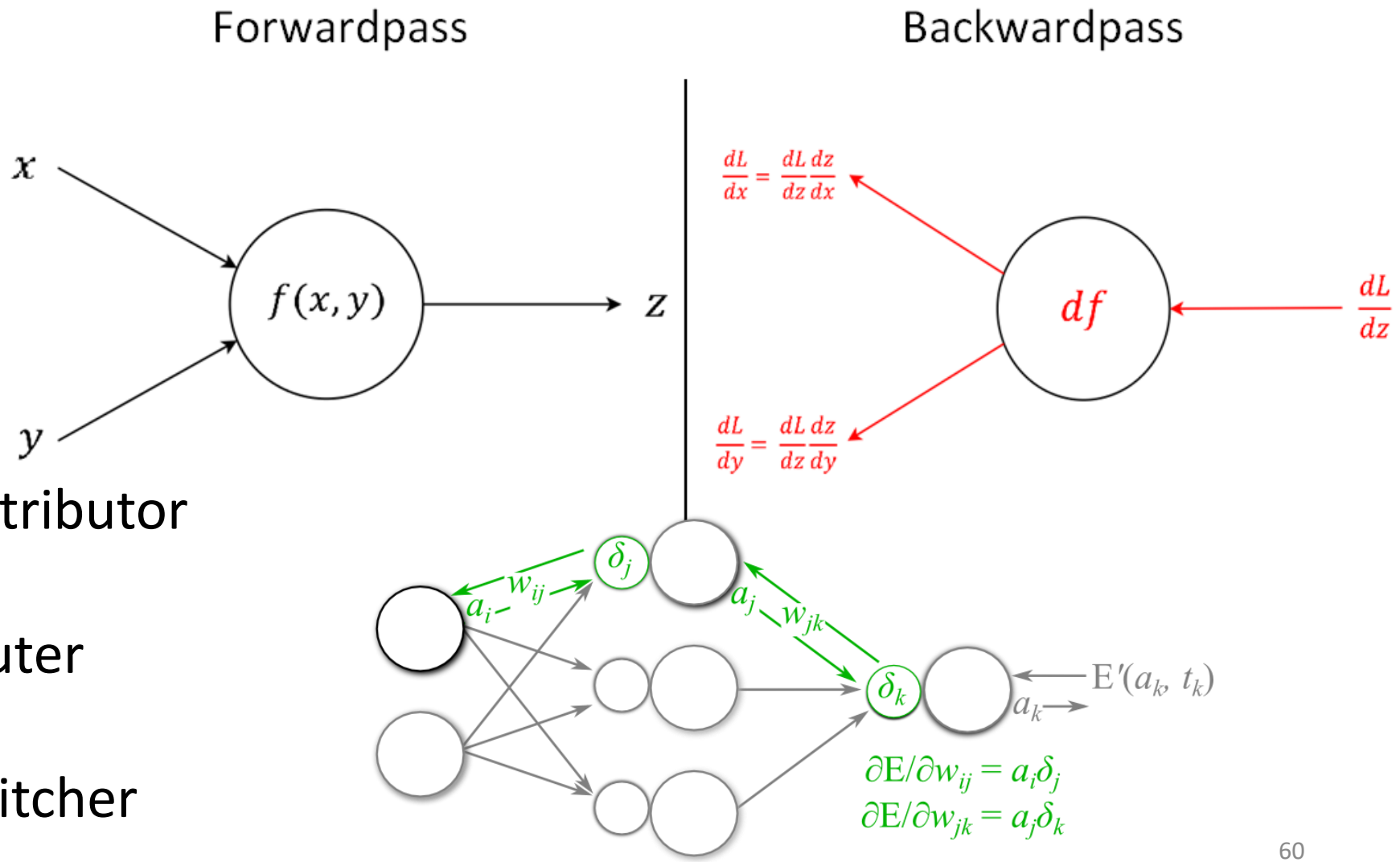


Backpropagation – Chain Rule

- Chain rule

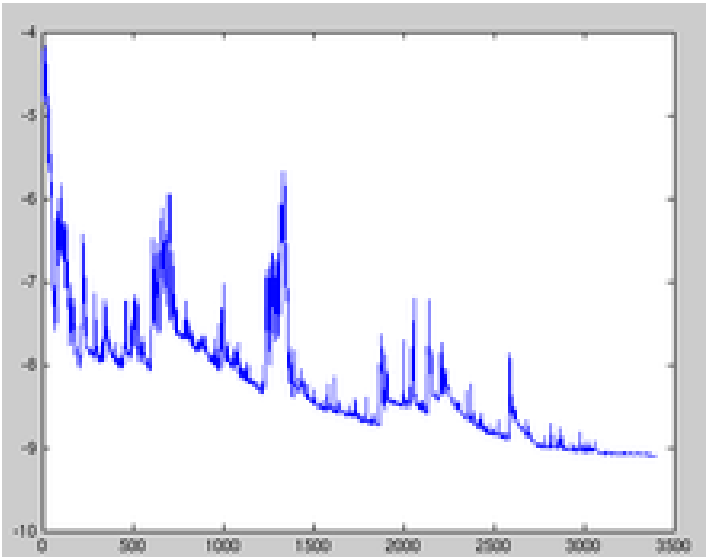
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

- Add gate:
  - gradient distributor
- Max gate:
  - gradient router
- Mul gate:
  - gradient switcher



Stochastic Gradient Descent (SGD)

- For large scale datasets,
  - Not possible to compute cost for all samples
- Select mini-batch of samples
  - Batch size
  - Update weights for each batch
  - Also called mini-batch gradient descent
- Momentum
  - Use previous weight update for current update
  - Accelerates learning

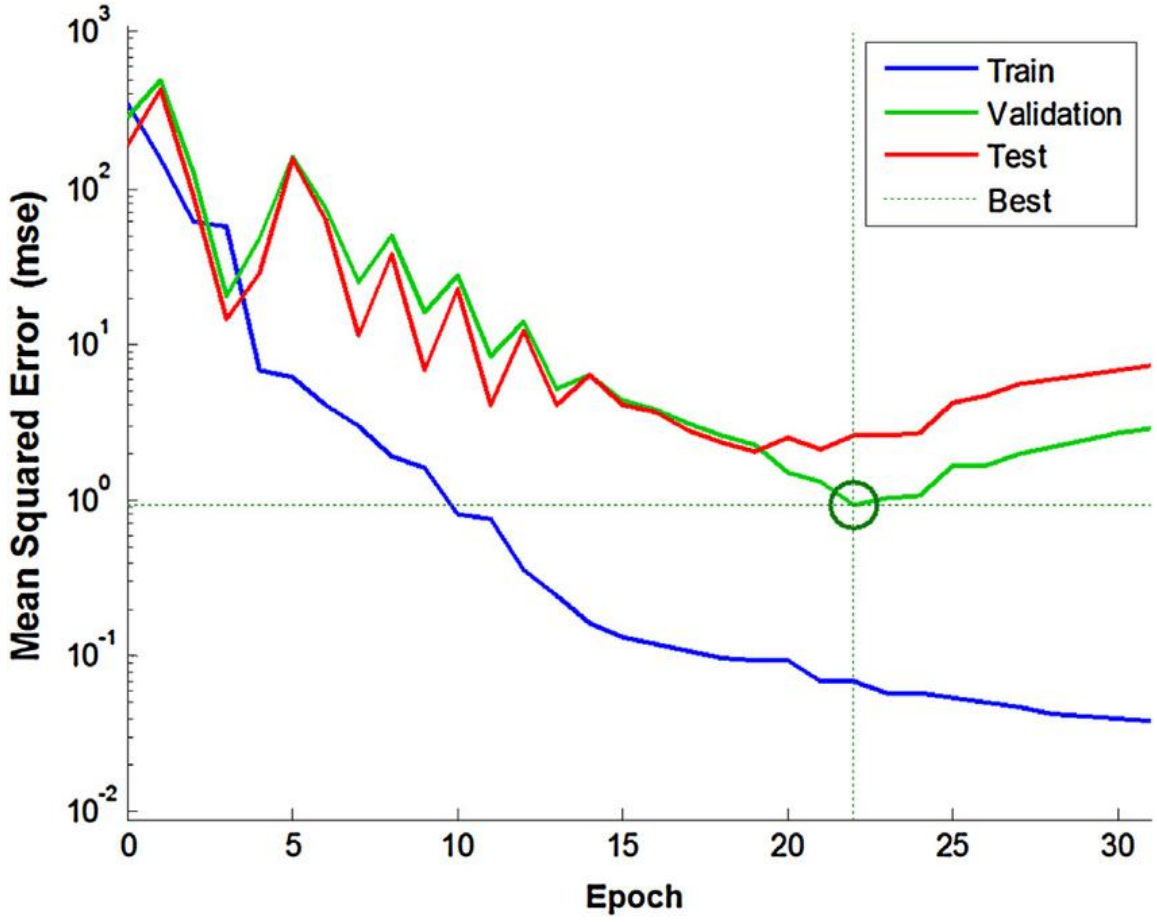


Part -II

- Network training
  - Loss function
  - Optimization
- Issues
  - Overfitting
- Other network
  - CNN variants
  - Recurrent networks
- Generative adversarial network

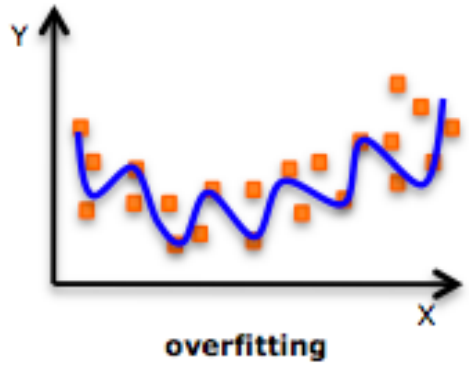
Validation, training and testing error

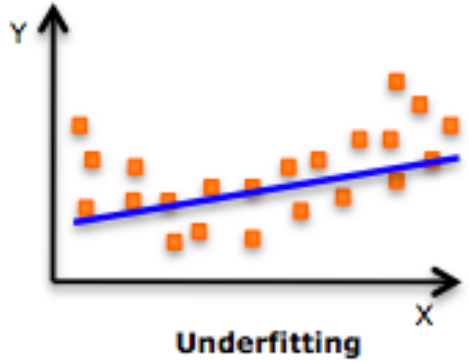
- The dataset is randomly divided
  - Train, validation, and test
- Training
  - train the model
- Validation
  - optimize the hyperparameters
- Test
  - test the performance

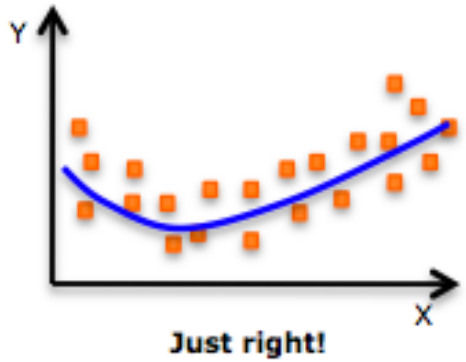


Overfitting and underfitting

- Overfitting
  - network models the training set too perfectly
  - performs poorly in the testing set
- Underfitting
  - network models the training set poorly
  - doesn't learn the function that describes the dataset







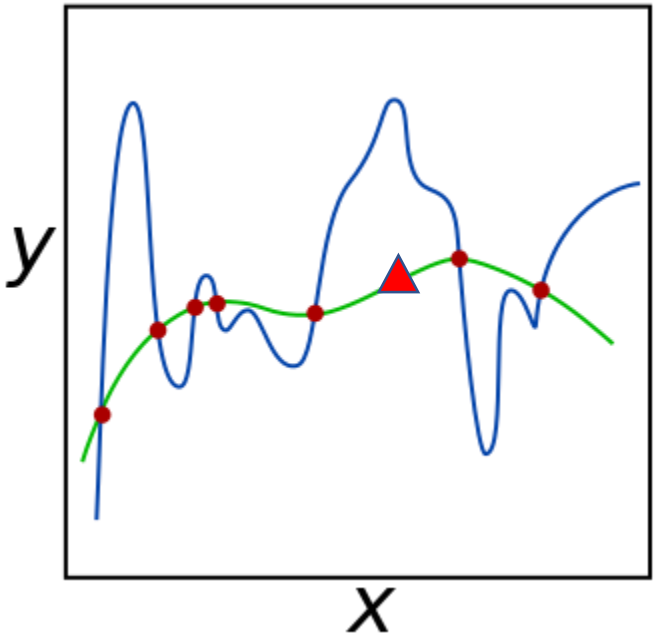
Regularization

- Avoid over-fitting

Loss Weight decay Regularization

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), yi) + \lambda R(W)$$

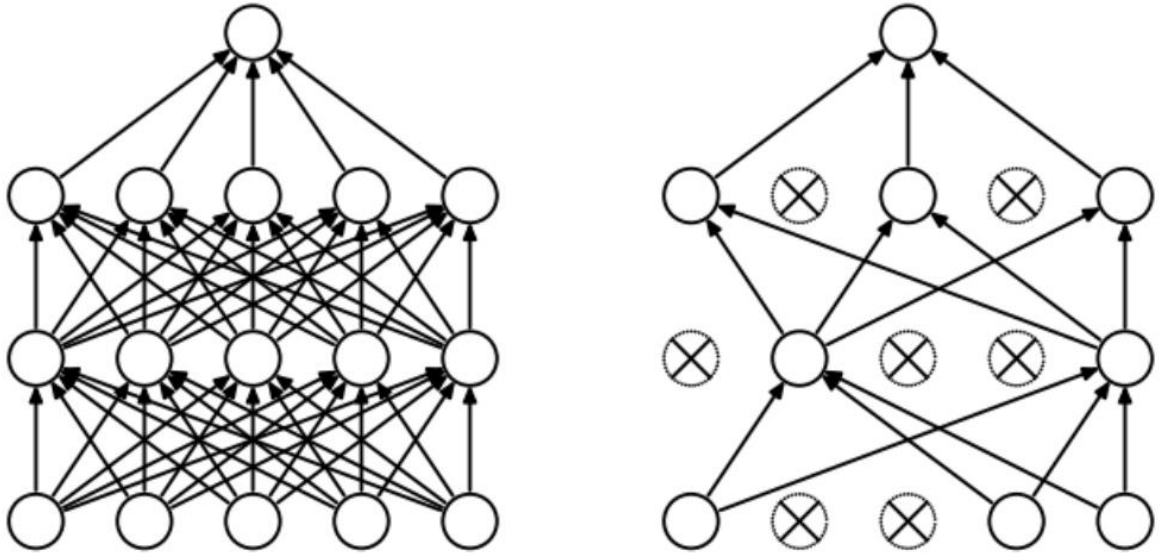
- In common use,
  - L1 Norm – $R(W) = \sum_i \sum_j |Wij|$
  - L2 Norm – $R(W) = \sum_i \sum_j W^2{}_{ij}$
  - Elastic net – $R(W) = \sum_i \sum_j \beta W^2{}_{ij} + |Wij|$



- Dropout, batch normalization

Dropout

- Stochastically switch neurons off
  - Each neuron is set to 0 with probability p
  - Hidden units cannot co-adapt to each other
  - Units are useful independently
- Hyperparameter
  - P is usually set to 0.5



(a) Standard Neural Net

(b) After applying dropout.

Part -II

- Network training
  - Loss function
  - Optimization
- Issues
  - Overfitting
- Other network
  - CNN variants
  - Recurrent networks
- Generative adversarial network

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

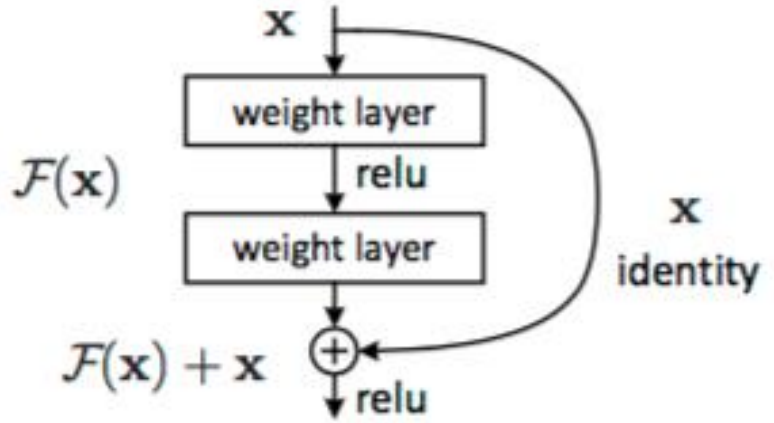
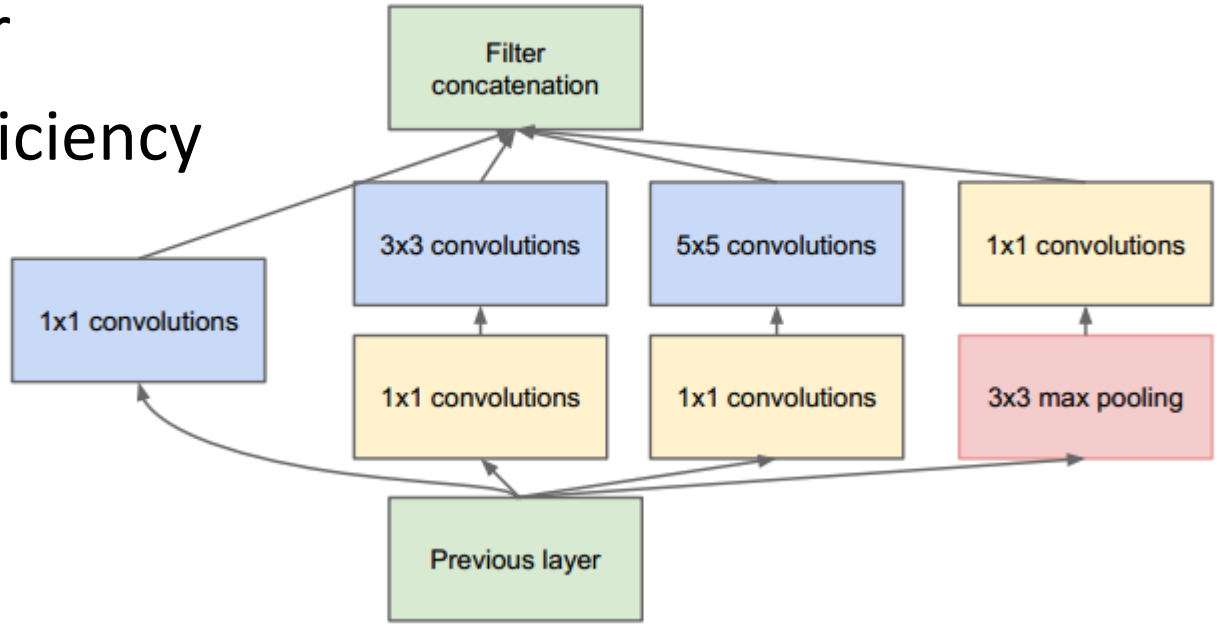


Figure 2. Residual learning: a building block.

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

GoogleNet - Inception

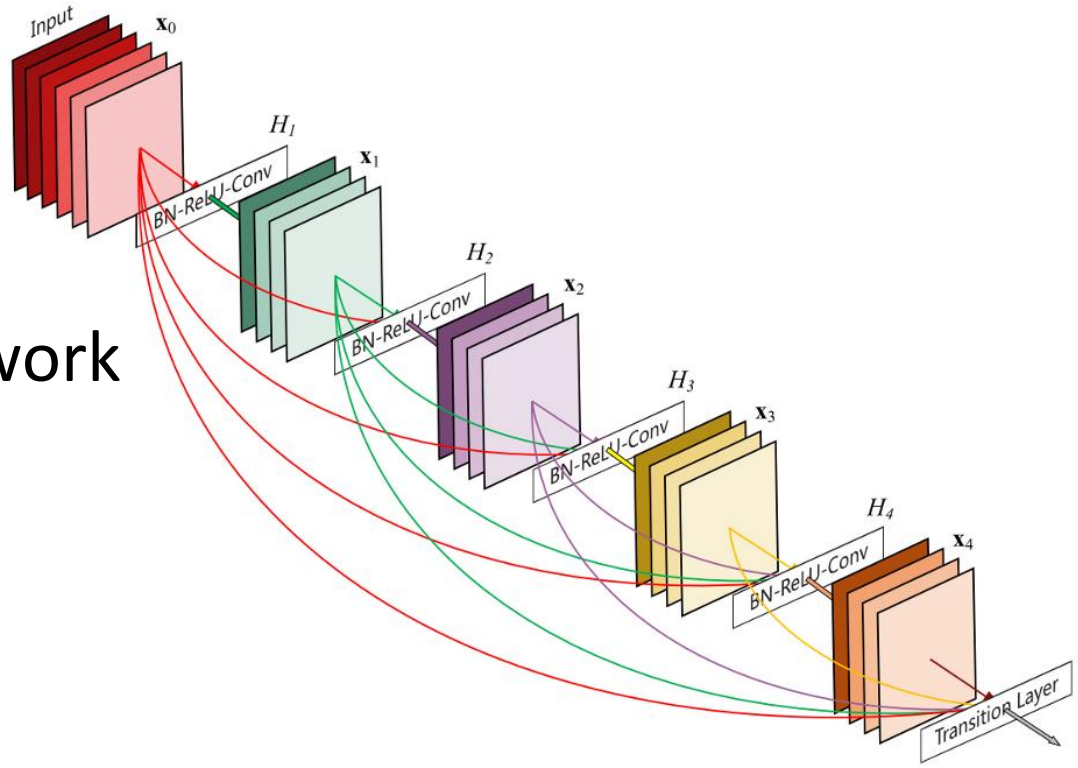
- ResNet is about going deeper
- Inception is about going wider
- Focused on computational efficiency
- The network learns
  - Which features are useful

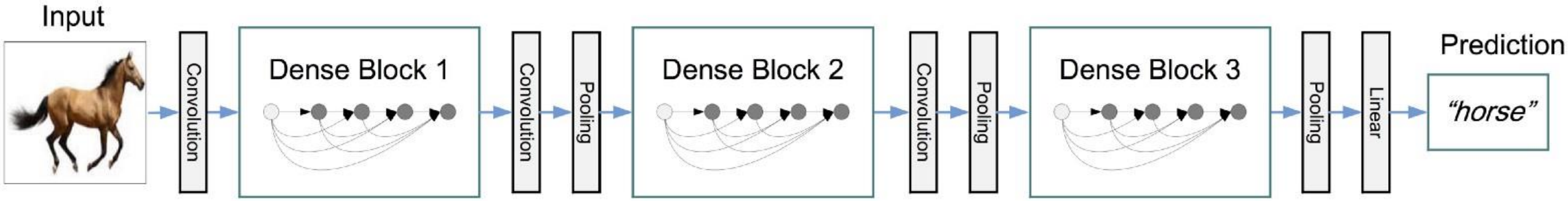


Going deeper with convolutions, Szegedy et al. (2014)
Inception-v4, Szegedy et al. (2016)

DenseNet

- Densely Connected Convolutional Network
  - Similar to ResNet
  - Concat features instead of summation
  - A layer is passed all previous maps
  - Sequence of dense blocks





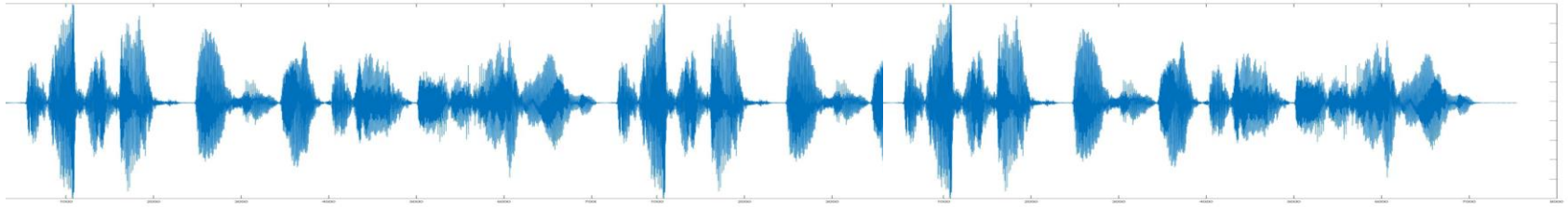
Huang, Gao, et al. "Densely connected convolutional networks." (2016).

Part -II

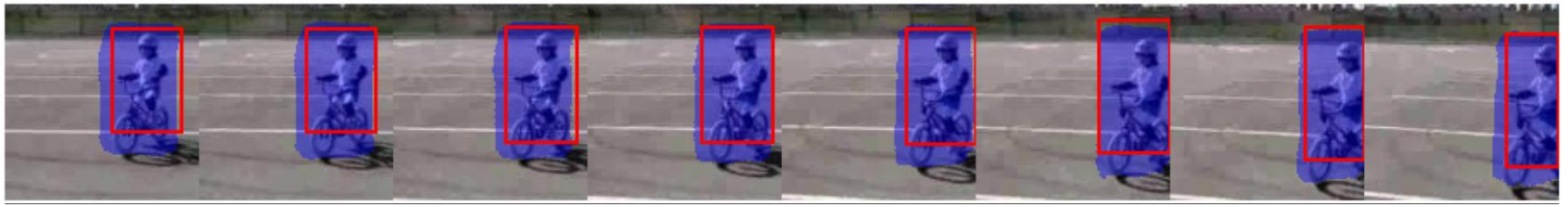
- Network training
  - Loss function
  - Optimization
- Issues
  - Overfitting
- Other network
  - CNN variants
  - Recurrent networks
- Generative adversarial network

Recurrent neural network

- Processing sequential data
  - Text - “It is fun to learn deep learning.”
  - Audio

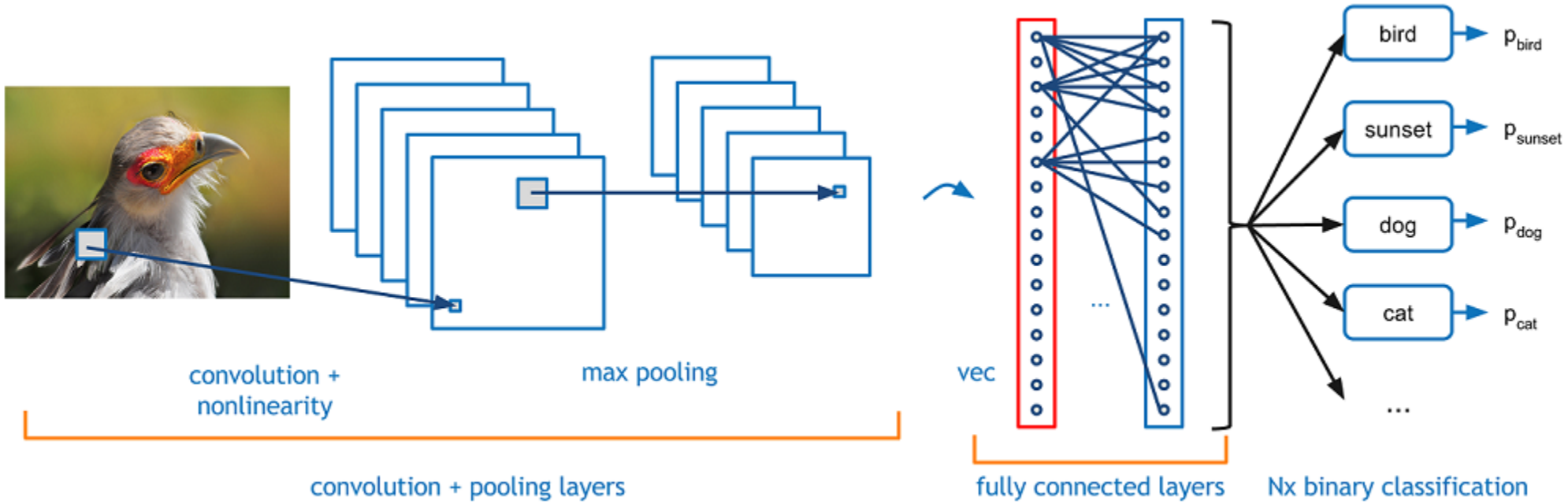


  - Video



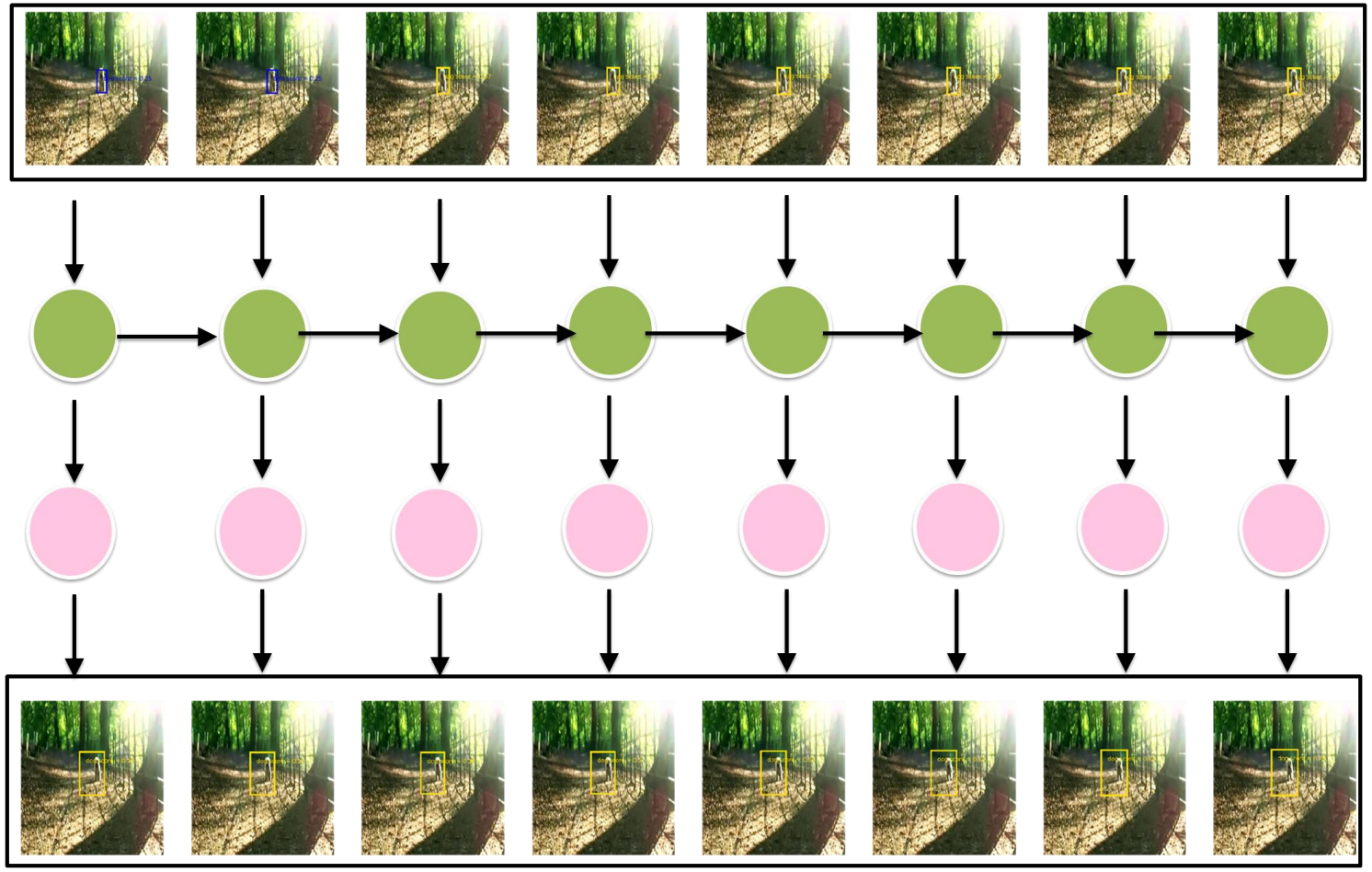
Recurrent neural network

- Comparison with convolution neural networks



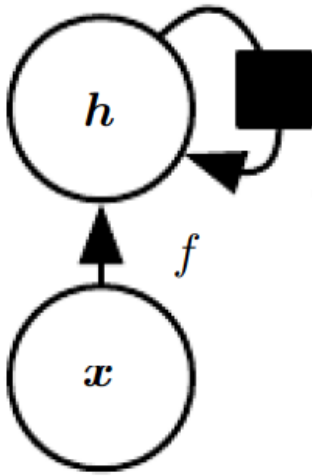
Source : https://adeshpande3.github.io

Recurrent neural network



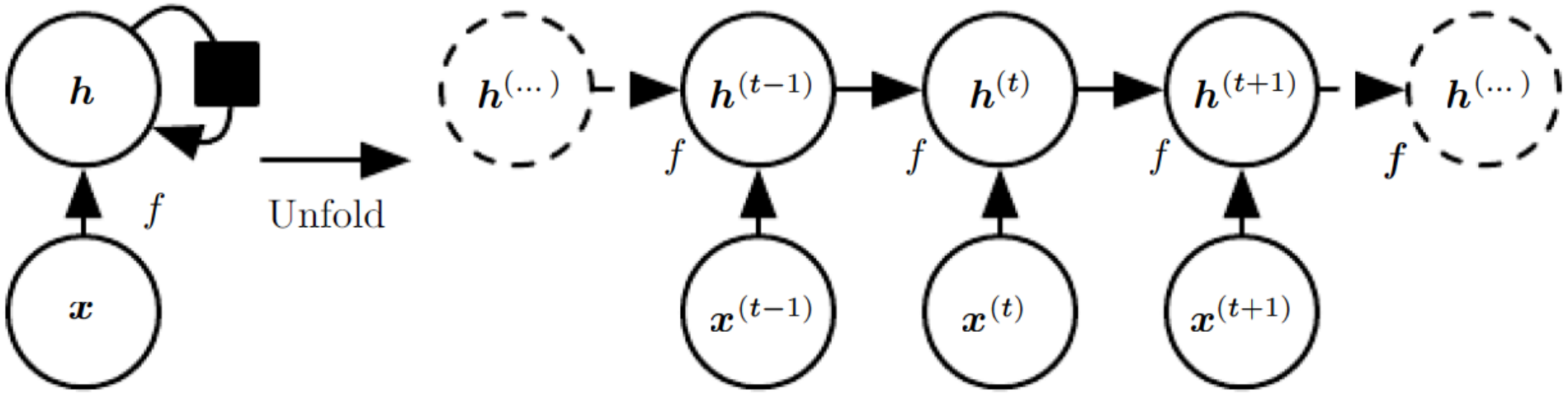
http://acsweb.ucsd.edu/~stripath/research/VOP.html

Recurrent neural network



Source: Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

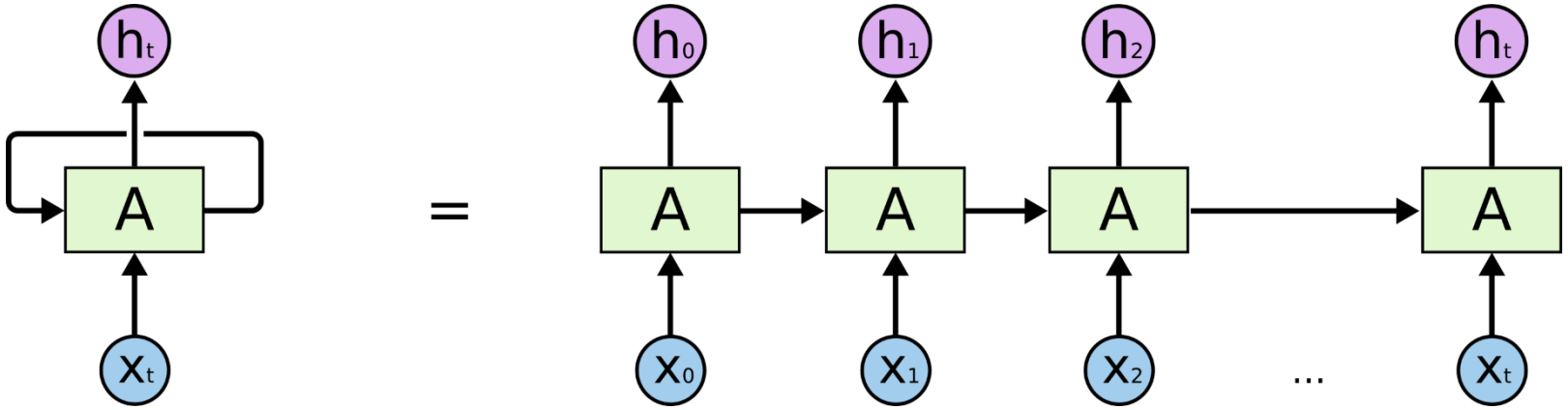
Recurrent neural network



Source: Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Recurrent neural network

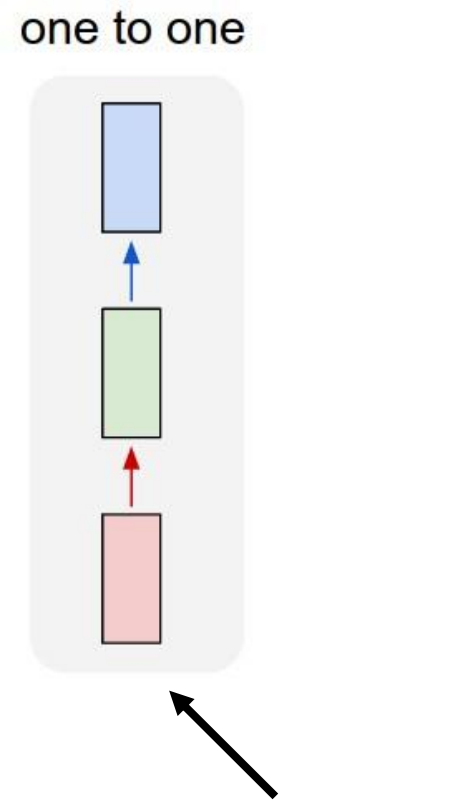
- Process sequences
- Feedback from previous time step



An unrolled recurrent neural network.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent neural network



Vanilla Neural Networks

http://karpathy.github.io

Recurrent neural network

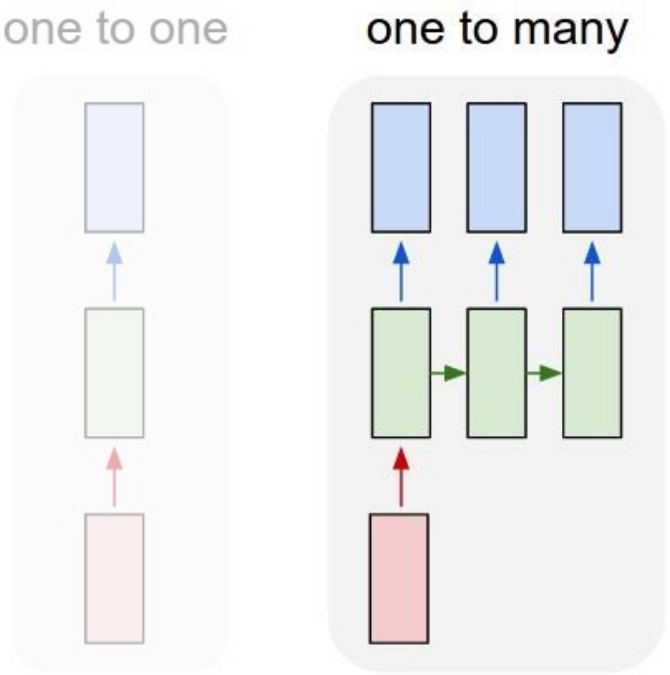
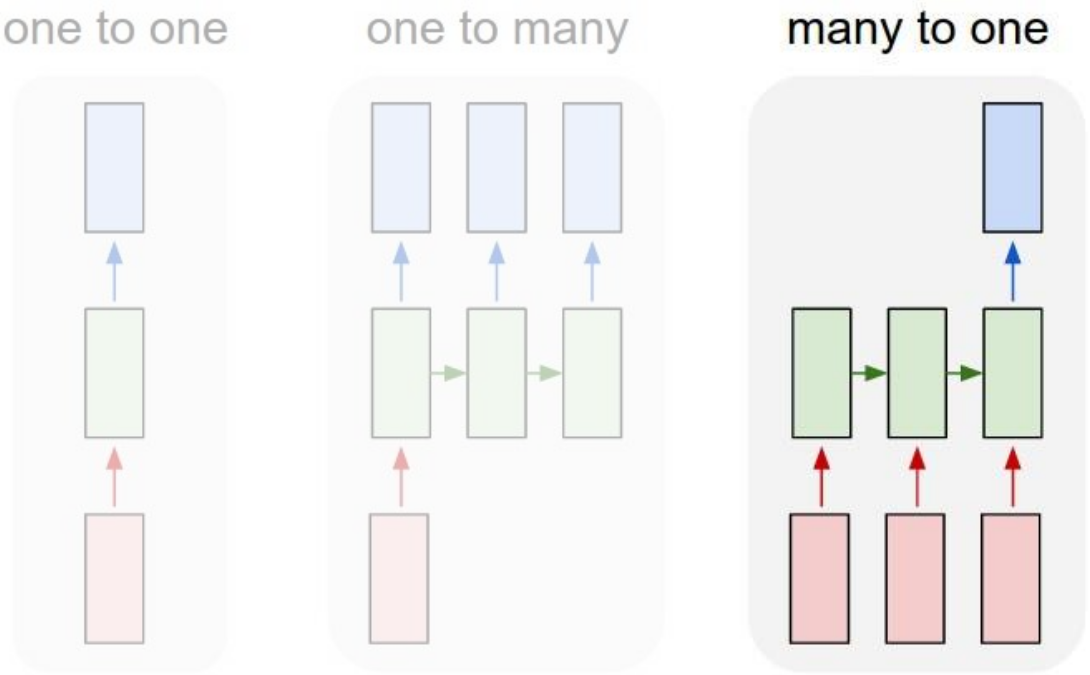


Image captioning, image to words

http://karpathy.github.io

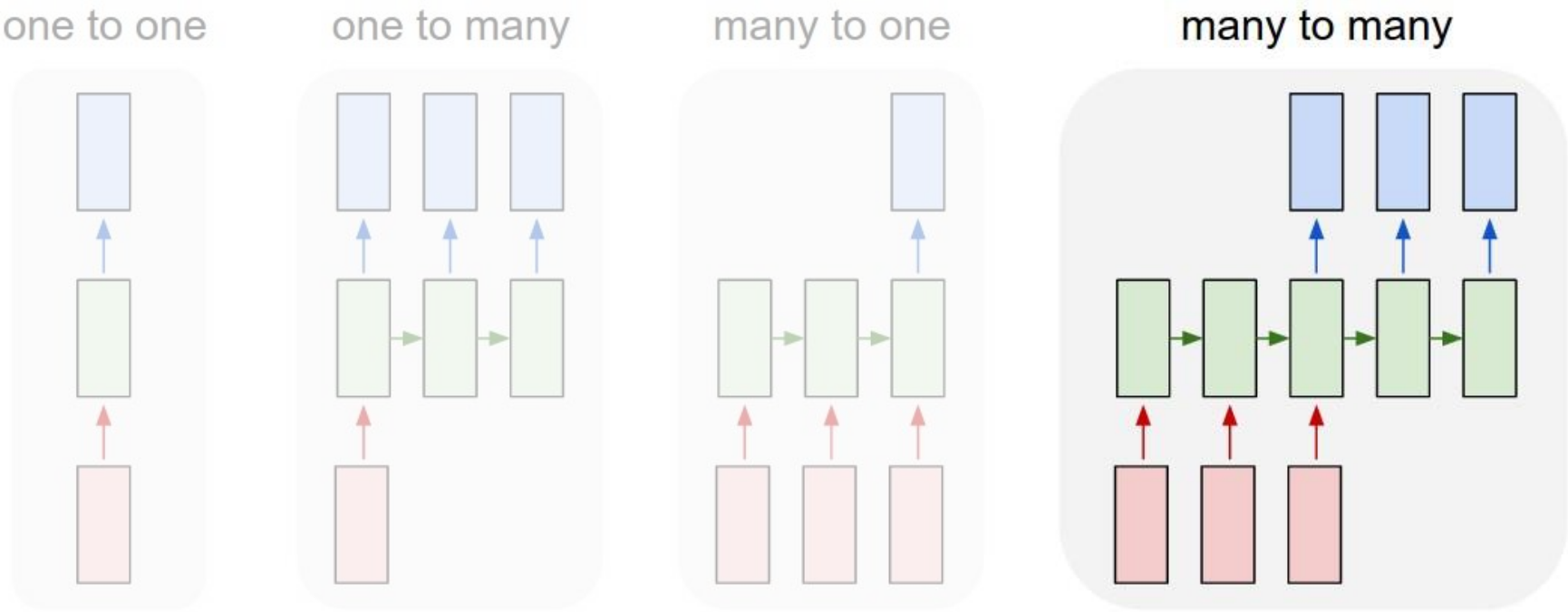
Recurrent neural network



Sentiment Classification
sequence of words -> sentiment

http://karpathy.github.io

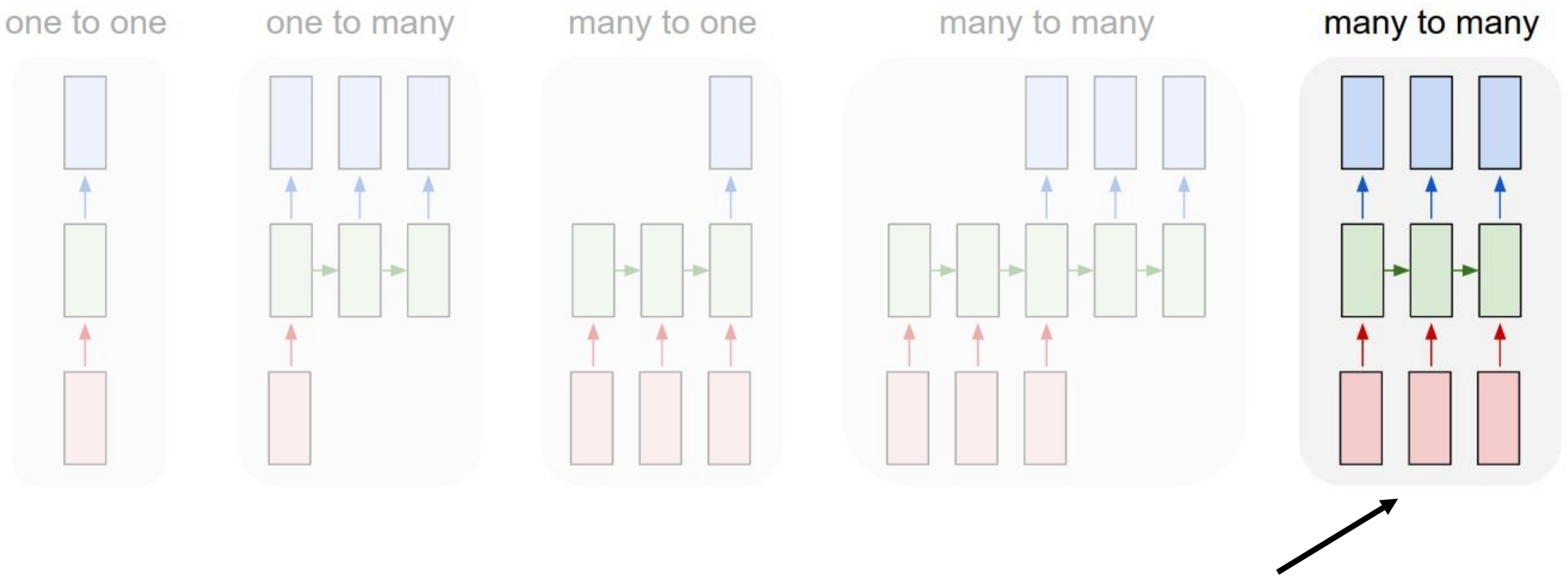
Recurrent neural network



Machine Translation
seq of words -> seq of words

http://karpathy.github.io

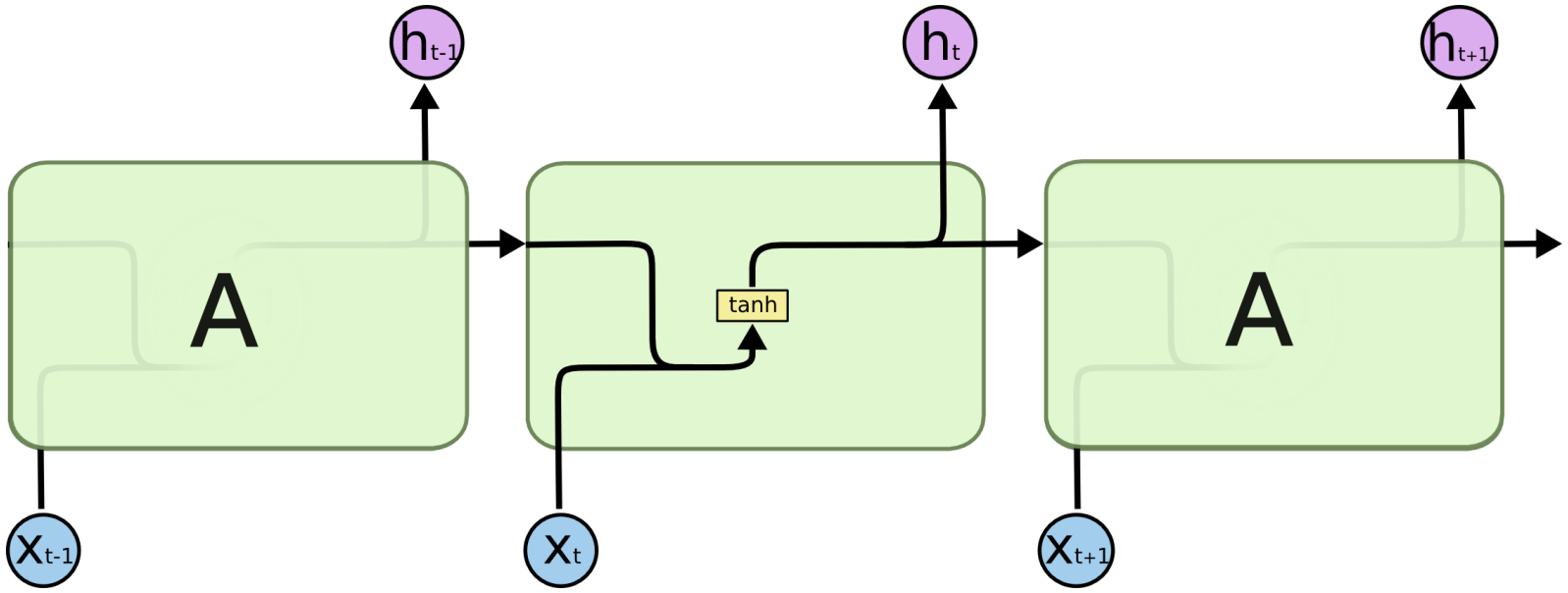
Recurrent neural network



Video classification on frame level

http://karpathy.github.io

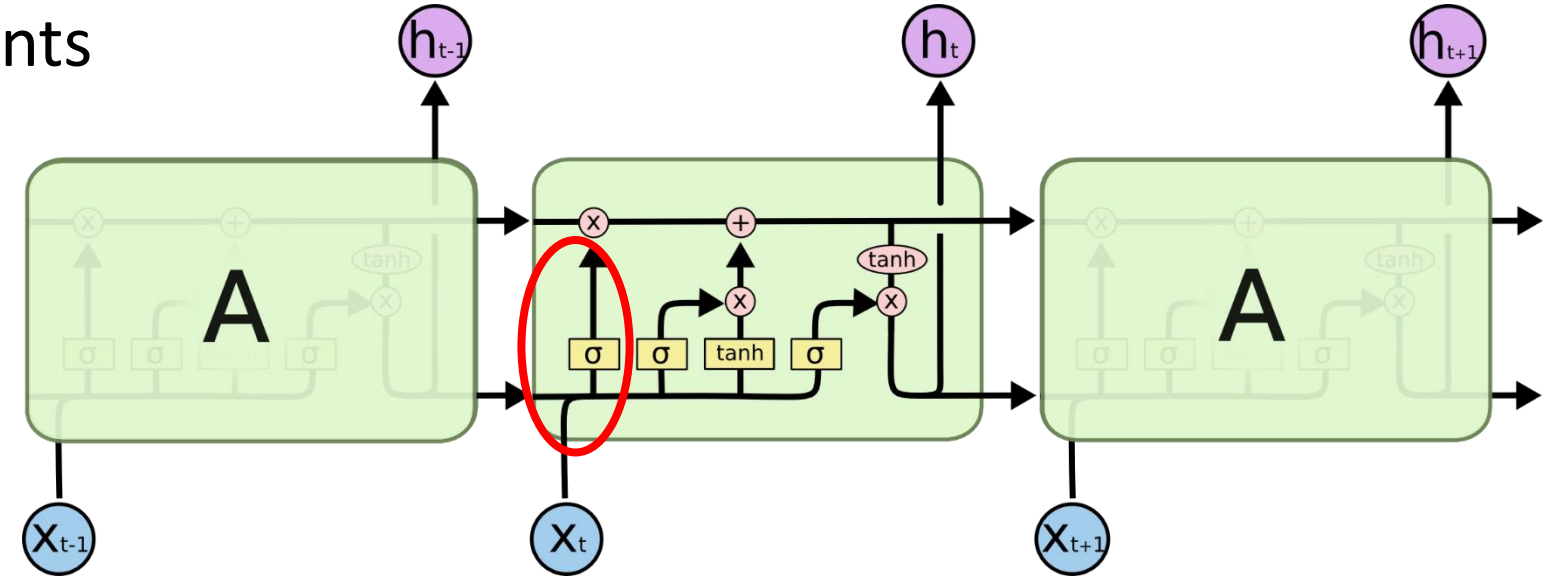
Vanilla RNN



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

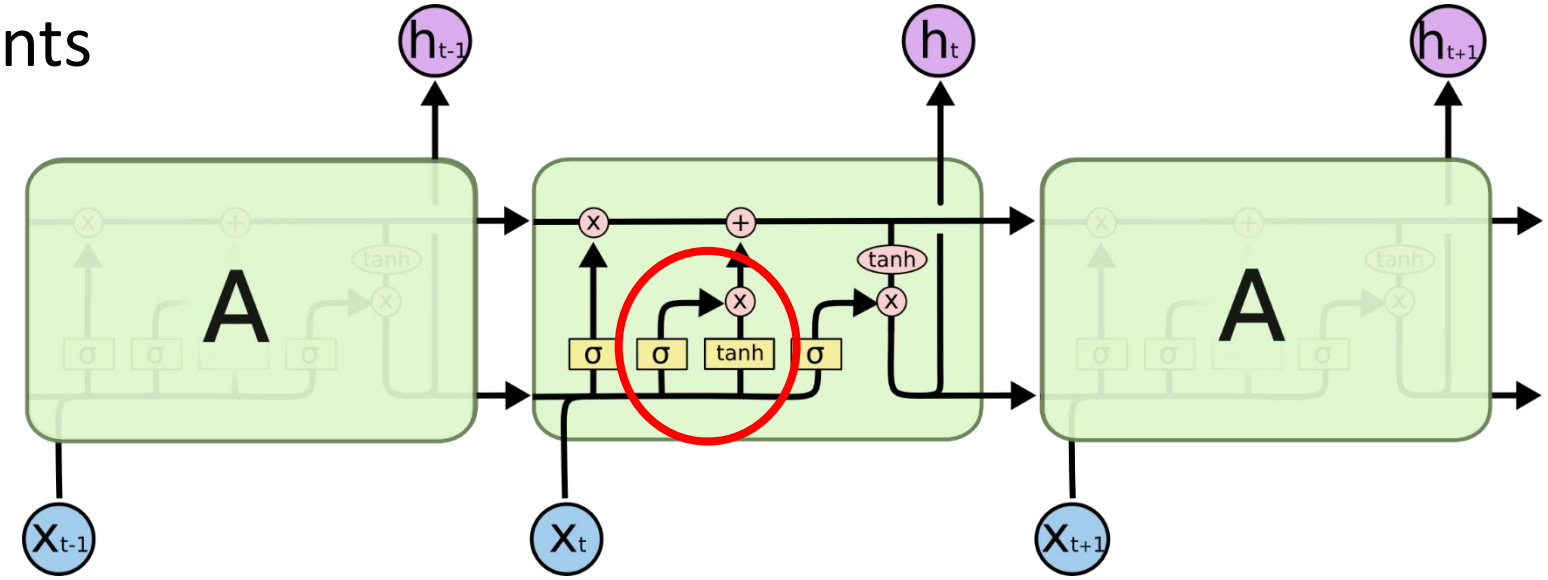
- Three main components
  - Forget gate layer



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

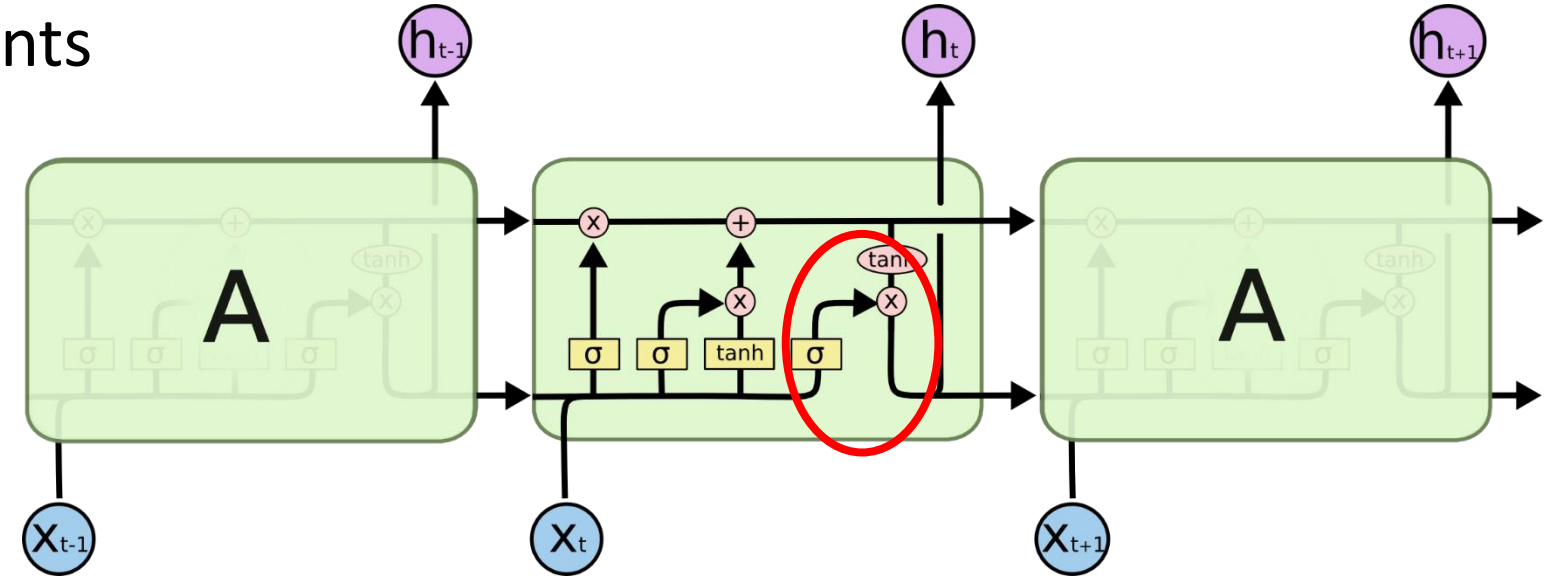
- Three main components
  - Forget gate layer
  - Input gate layer



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

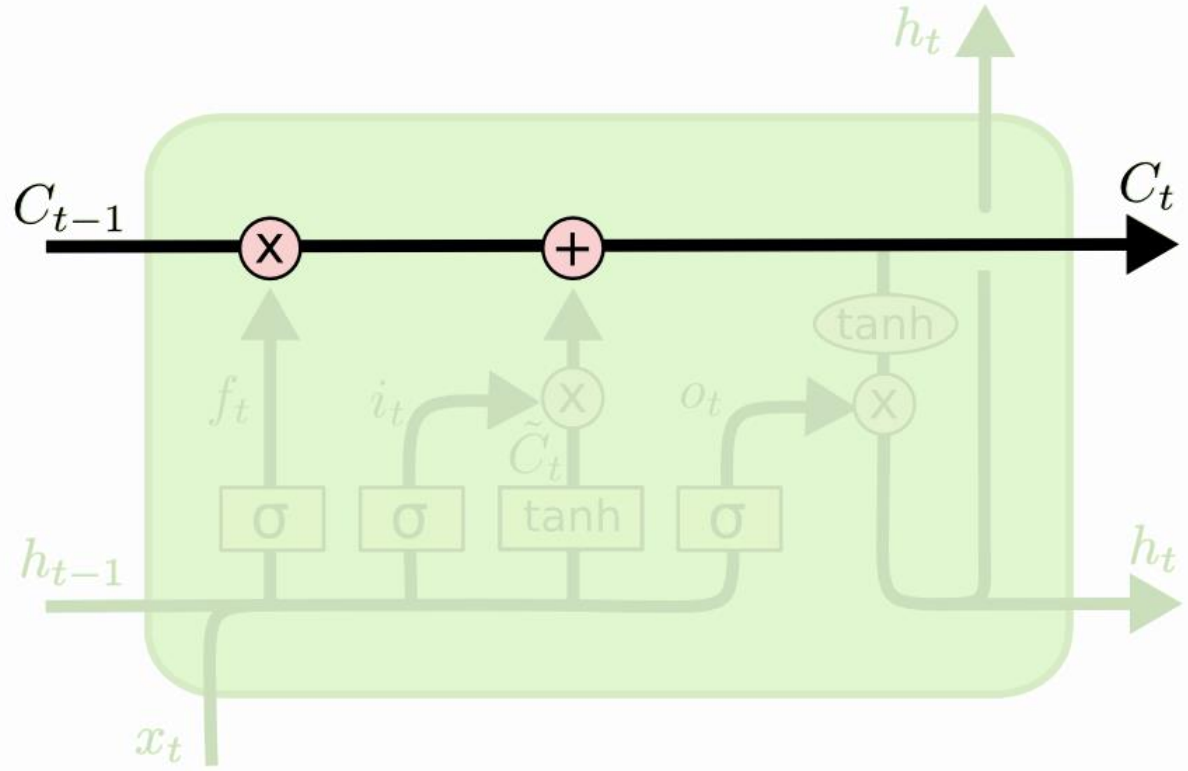
- Three main components
  - Forget gate layer
  - Input gate layer
  - Output gate layer



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

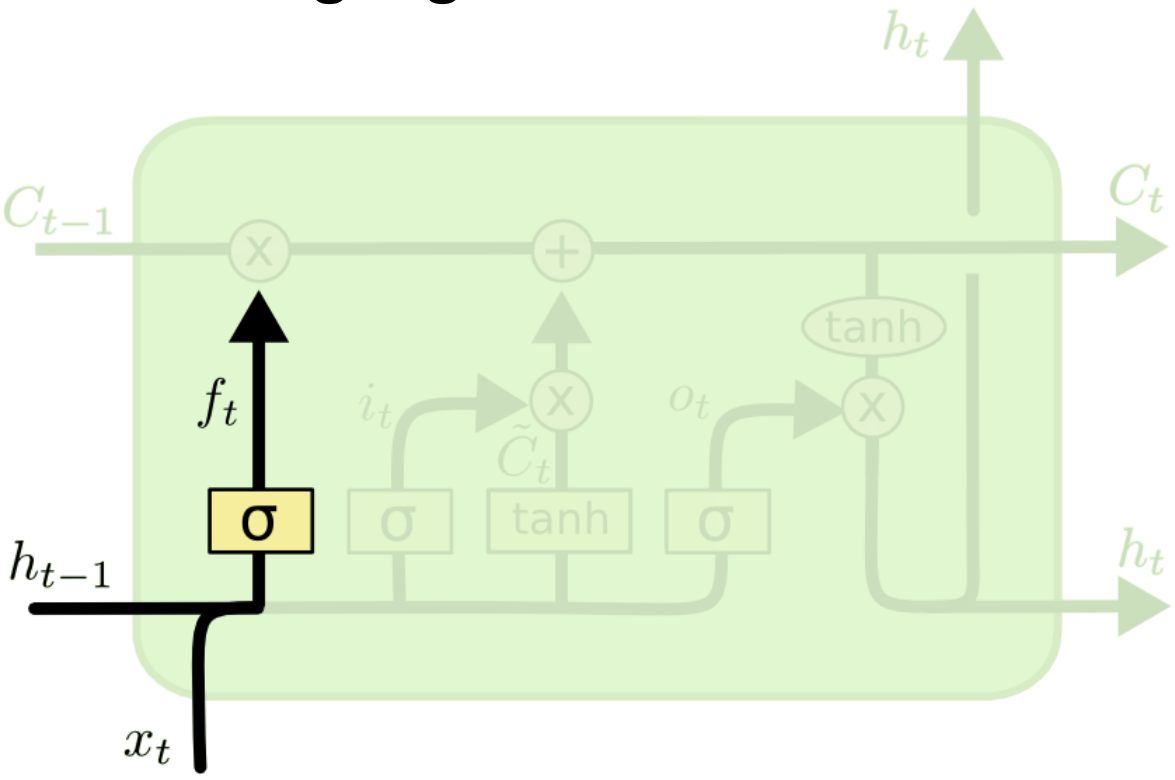
- Cell state



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

- Forget gate

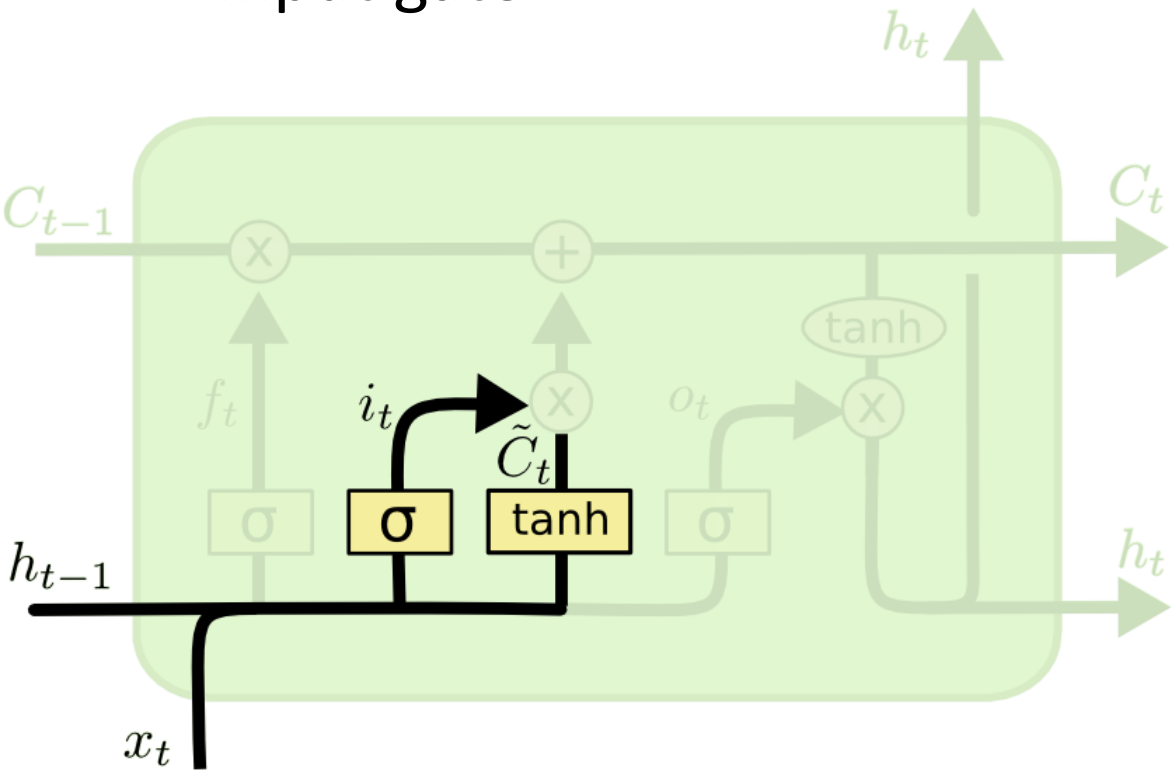


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

- Input gate



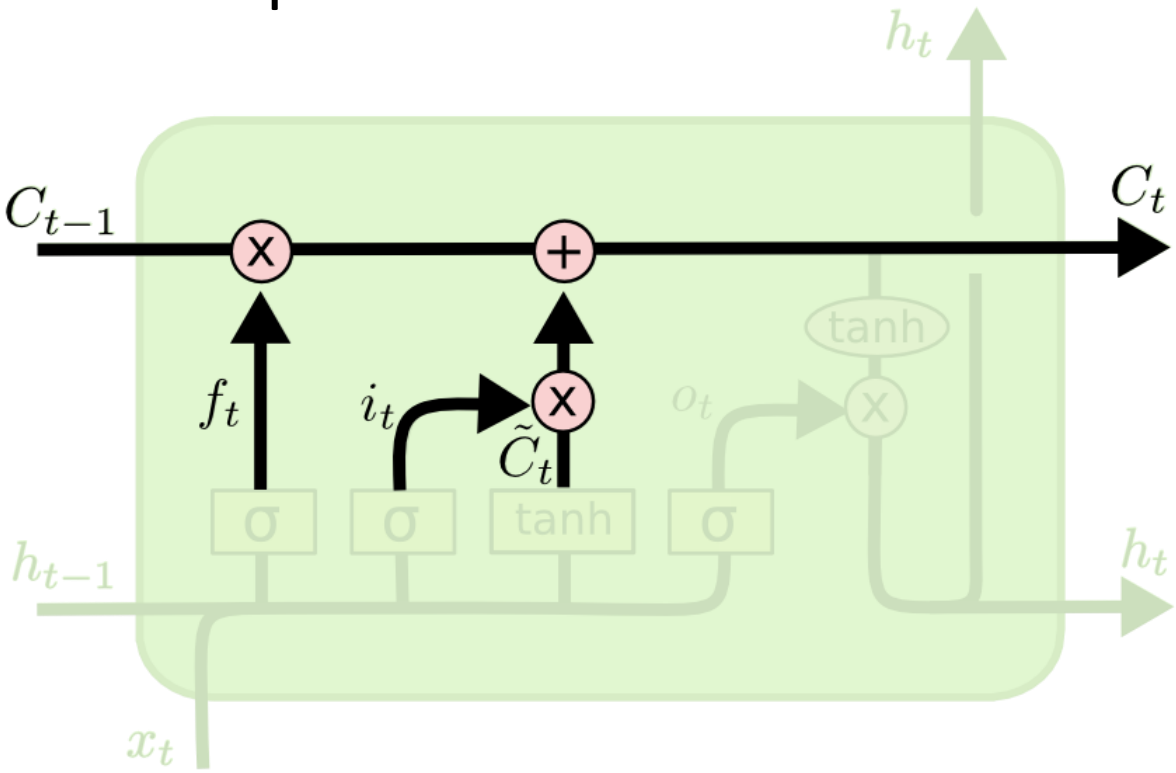
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

- Update cell state

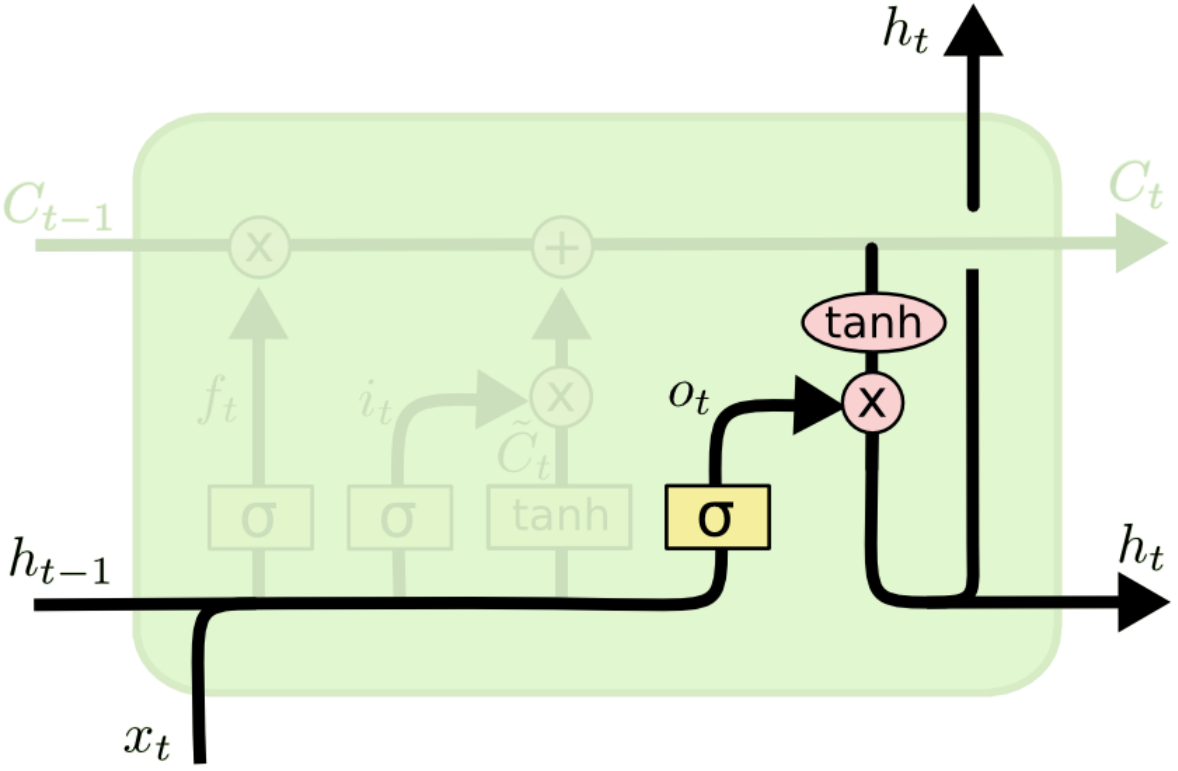


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-short term memory (LSTM)

- Output

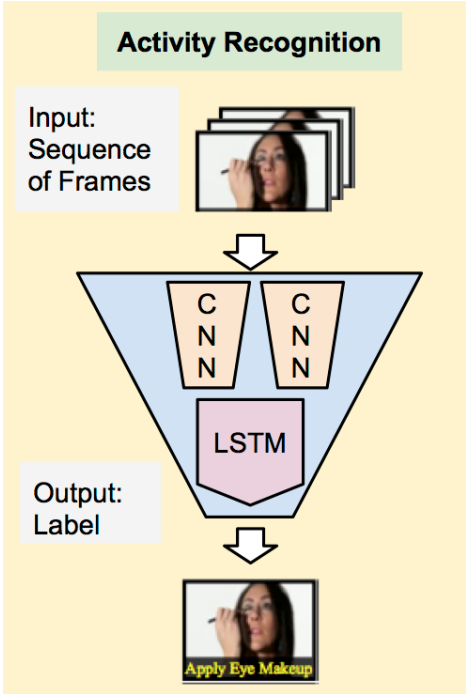


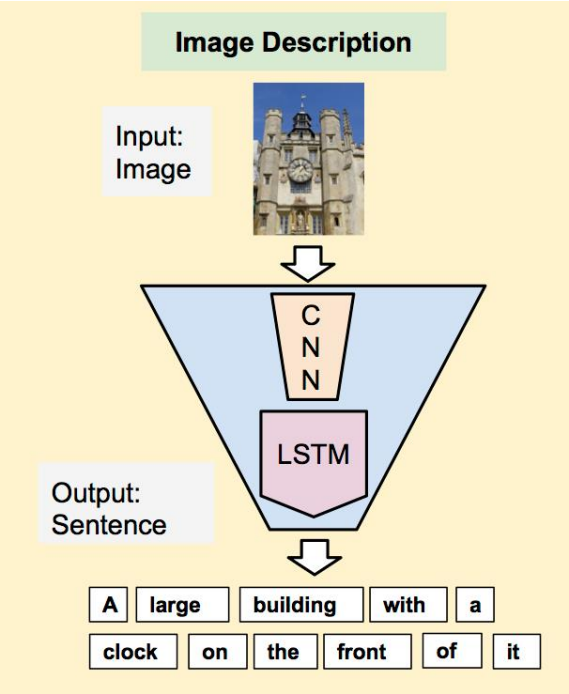
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

CNN-LSTM

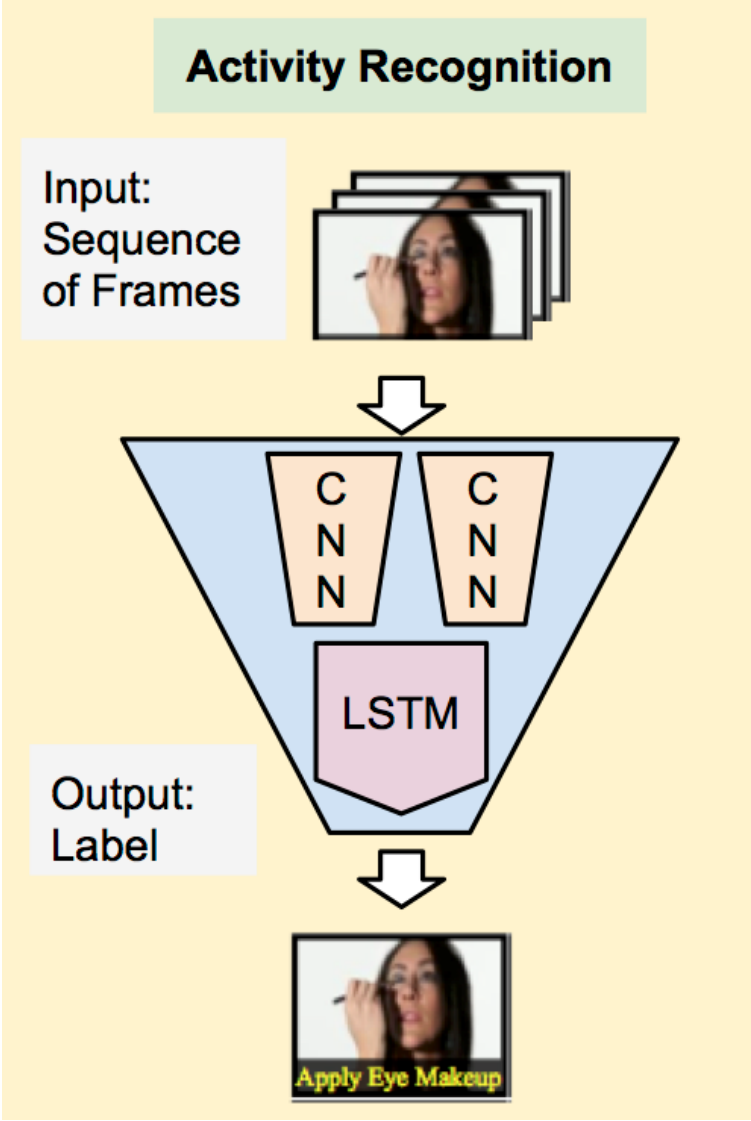


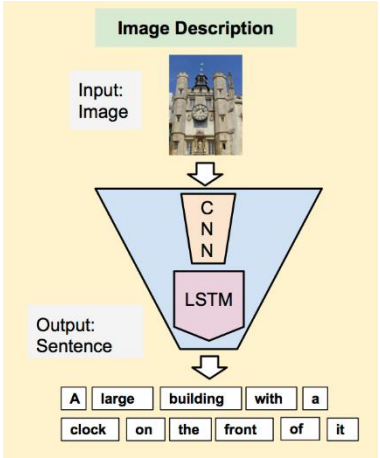


http://jeffdonahue.com/lrcn/

Vinyals et. al. Show and Tell, 2015
Jef et. al. Long-term Recurrent Convolutional Networks 2015

CNN-LSTM

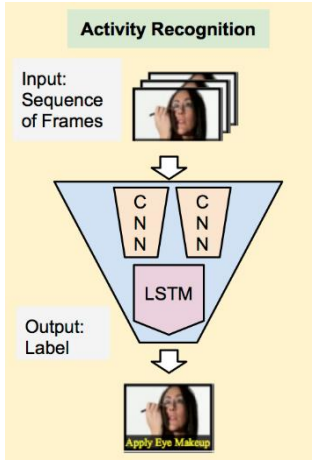


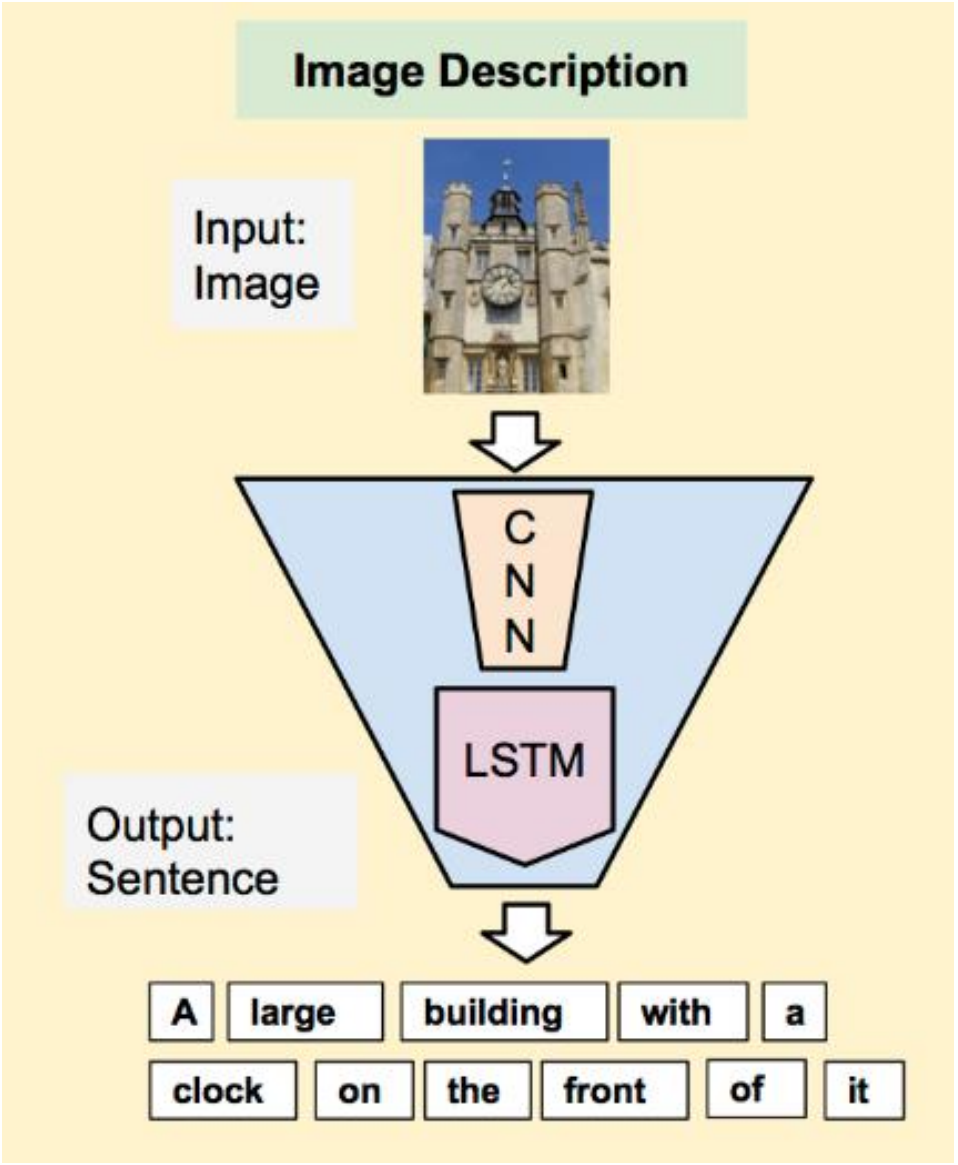


http://jeffdonahue.com/lrcn/

Vinyals et. al. Show and Tell, 2015
Jef et. al. Long-term Recurrent Convolutional Networks 2015

CNN-LSTM



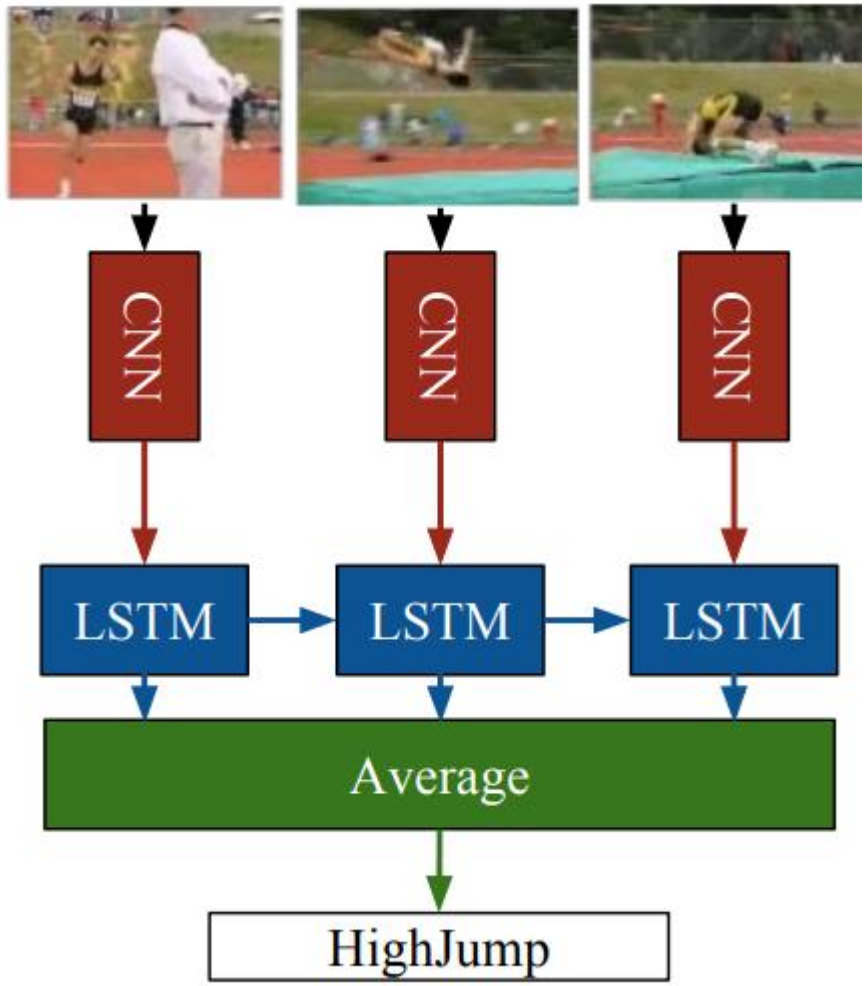


http://jeffdonahue.com/lrcn/

Vinyals et. al. Show and Tell, 2015
Jef et. al. Long-term Recurrent Convolutional Networks 2015

LSTM – Application 1

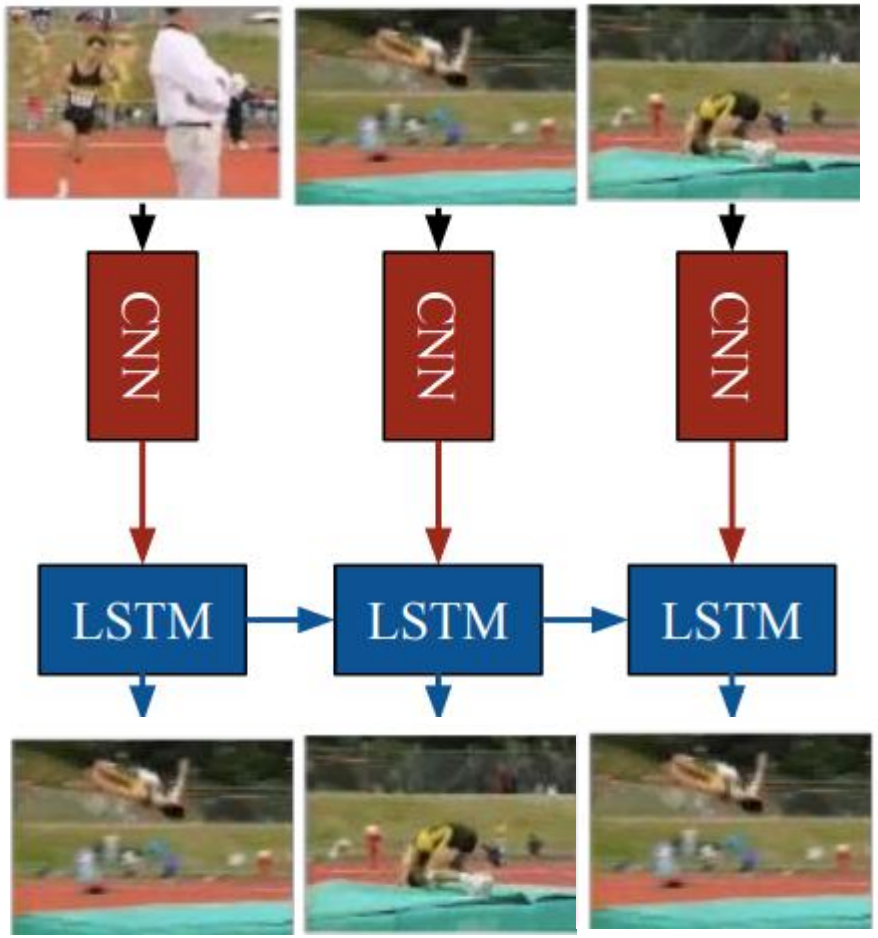
- Video activity recognition



http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review

LSTM – Application 2

- Future frame prediction

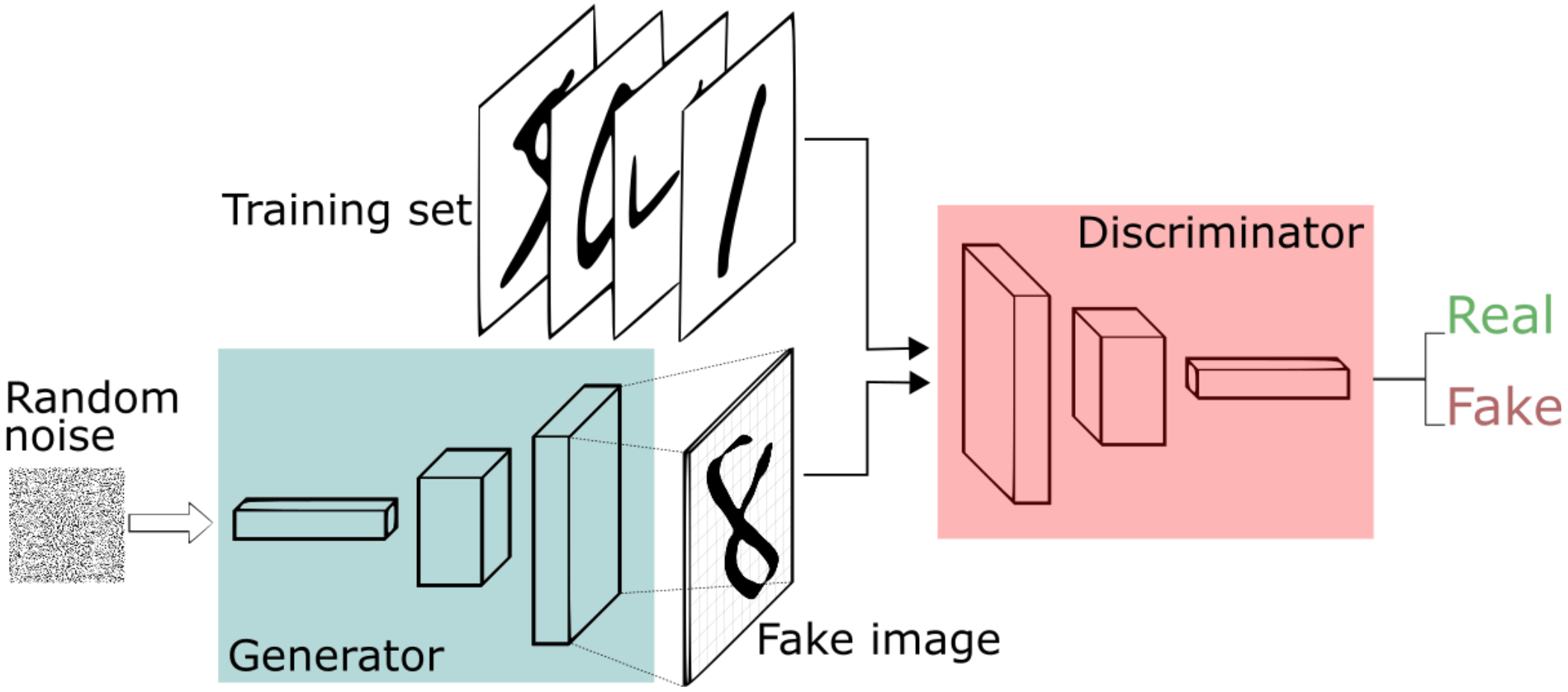


http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review

Part -II

- Network training
  - Loss function
  - Optimization
- Issues
  - Overfitting
- Other network
  - CNN variants
  - Recurrent networks
- Generative adversarial network

Generative adversarial network



Thank you!

Questions?