CONVOLUTIONAL AND RECURRENT NEURAL NETWORKS

Neural networks

- Fully connected networks
  - Neuron
  - Non-linearity
  - Softmax layer
- DNN training
  - Loss function and regularization
  - SGD and backprop
  - Learning rate
  - Overfitting – dropout, batchnorm
- CNN, RNN, LSTM, GRU <- This class

Convolution Neural Networks

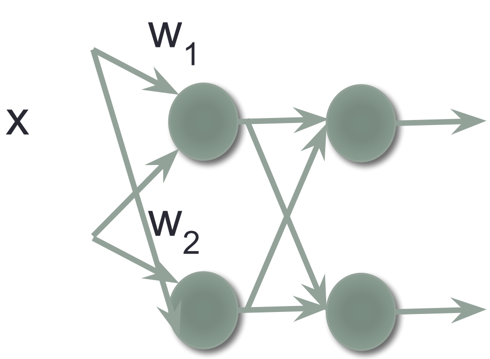
Convolution

Convolution Layers and Pooling Layers

CNN progress

PCA as a filter

- $W^T x$



- What happens if I have a person that is off-frame?
- Need another filter that is shifted

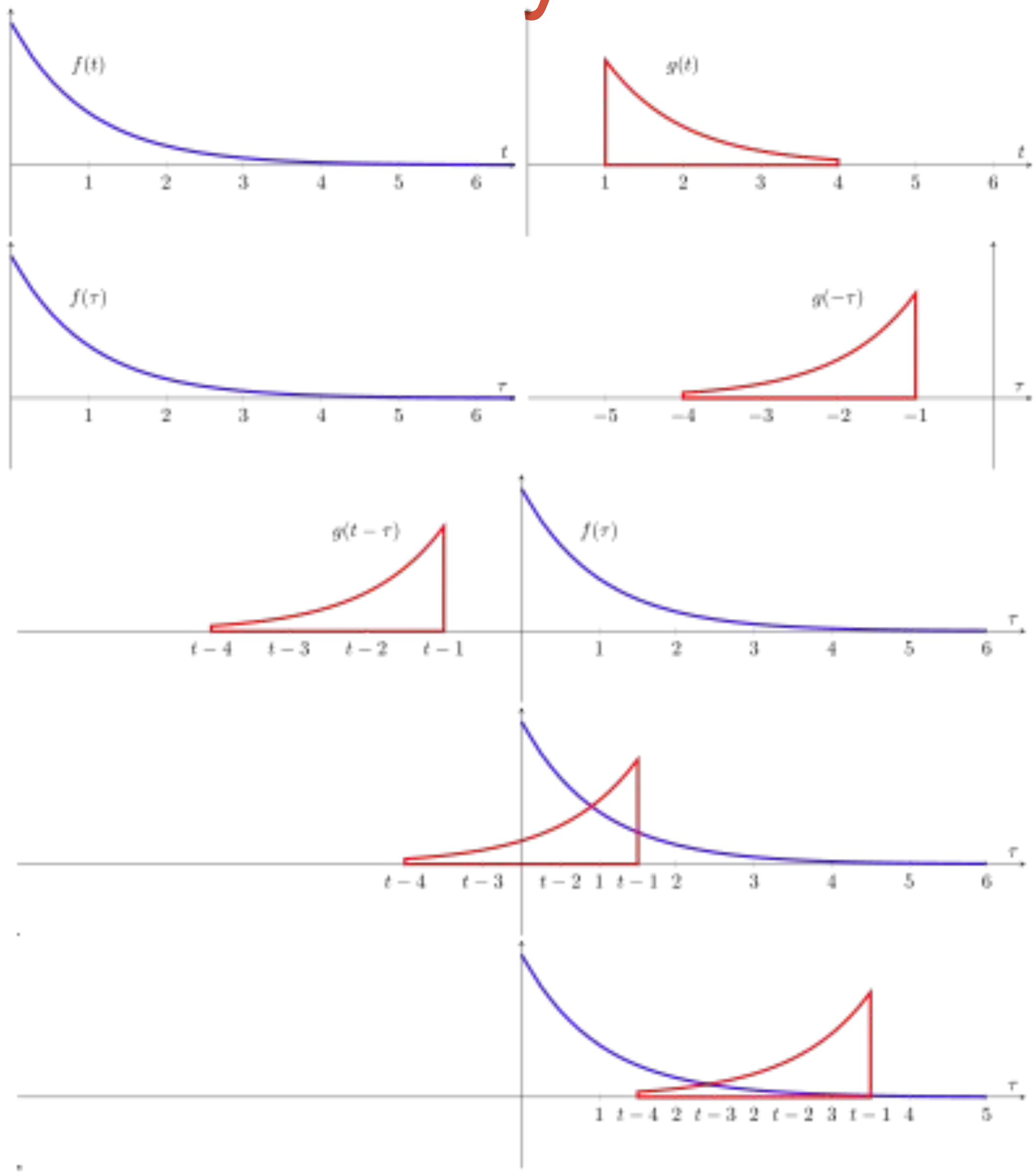
Convolution

- Continuous convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau.$$

- Meaning of (t-T) : Flip then shift

Convolution visually



Demo

https://en.wikipedia.org/wiki/Convolution

Convolution discrete

- Discrete convolution

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$
$$= \sum_{m=-\infty}^{\infty} f[n-m]g[m]$$

- Continuous convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau.$$

- Same concept as continuous version

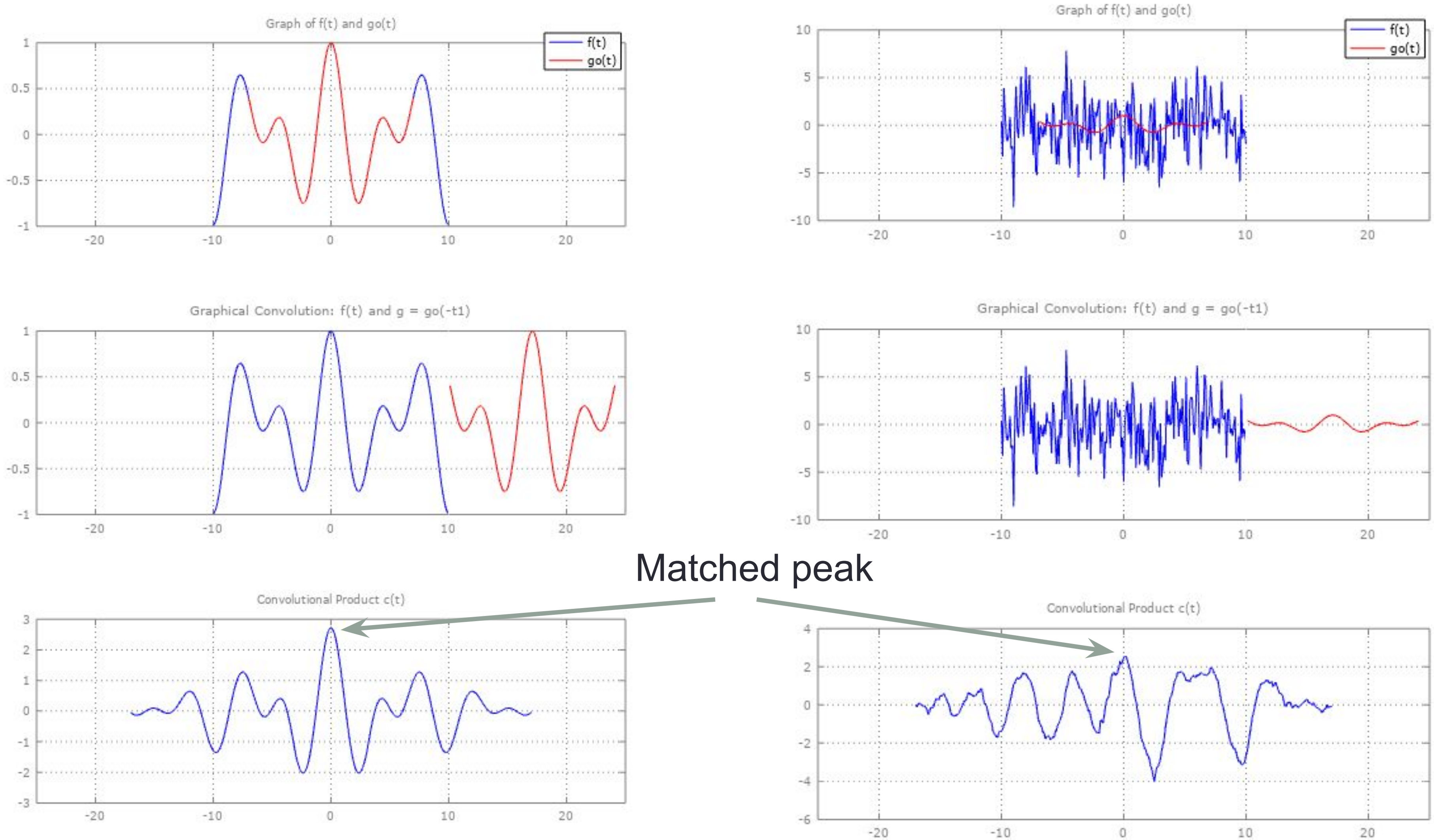
Matched filters

- We can use convolution to detect things that match our pattern
- Convolution can be considered as a filter (Why? Take ASR next semester ☺)
- If the filter detects our pattern, it will show up as a nice peak even if there are noise.

- Demo

Matched filters

Red: matched filter
Blue: signal



Convolution and Cross-Correlation

- Convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau) g(\tau)\, d\tau.$$

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n-m]$$
$$= \sum_{m=-\infty}^{\infty} f[n-m] g[m]$$

- (Cross)-Correlation

$$(f \star g)(\tau) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f^*(t)\, g(t+\tau)\, dt,$$

$$(f \star g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f^*[m]\, g[m+n].$$

Convolution and cross-correlation are the same if g(t) is symmetric (even function).
Using convolution or cross-correlation yield the same results, if the filters are learned.
However, cross-correlation is simpler to code.
In image processing, people used convolution then switched to cross-correlation, but kept the same name.

2D convolution

- Flip and shifts in 2D

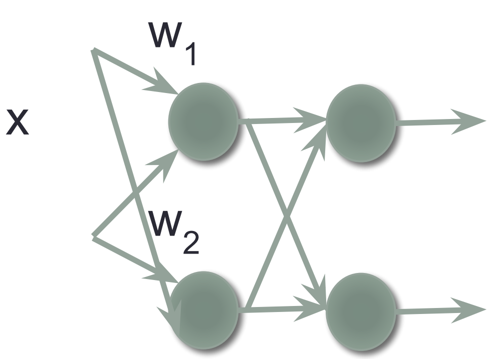
- But, we no longer flips

Our match filter

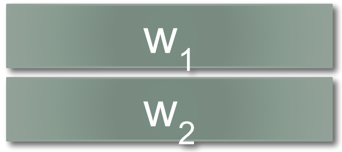
Will get some peak here

PCA as a filter

- $W^T x$



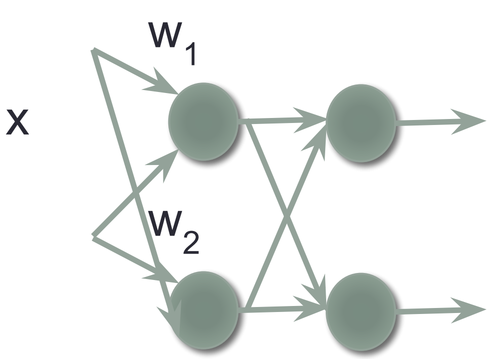
- What happens if I have a person that is off-frame?
- Need another filter that is shifted



$x = y$

Shift in feature space

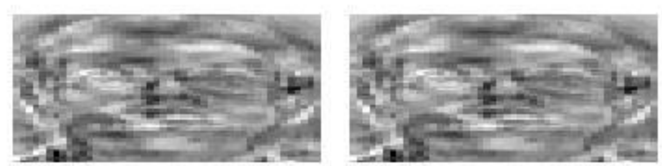
- $W^T x$



- What happens if I have a person that is off-frame?
- Ans: Convolution with W as filter

fischer projection = $V^T W^T x$
$= (WV)^T x$

Wv_1 Wv_2



LDA projections

$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} x = y$

$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} y$

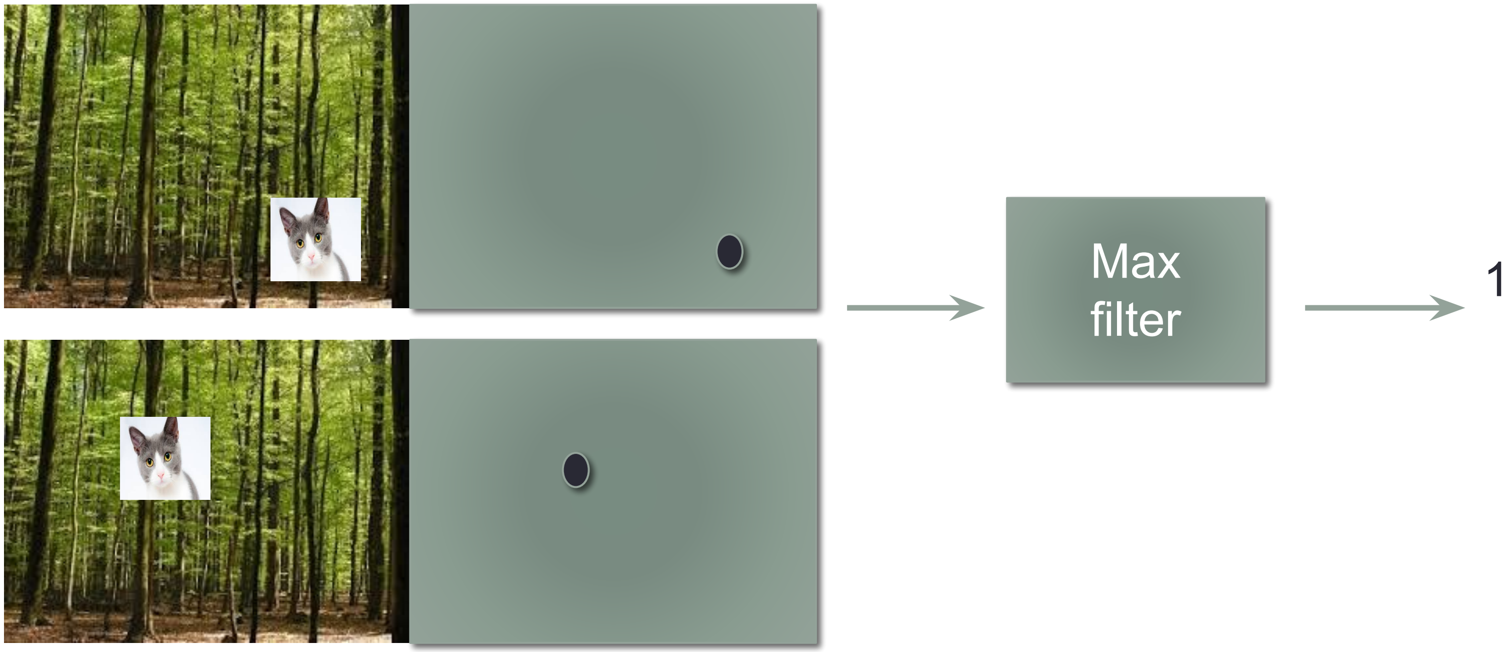
Convolutional Neural networks

- A neural network with convolutions! (cross-correlation to be precise)
- But we have peaks at different location

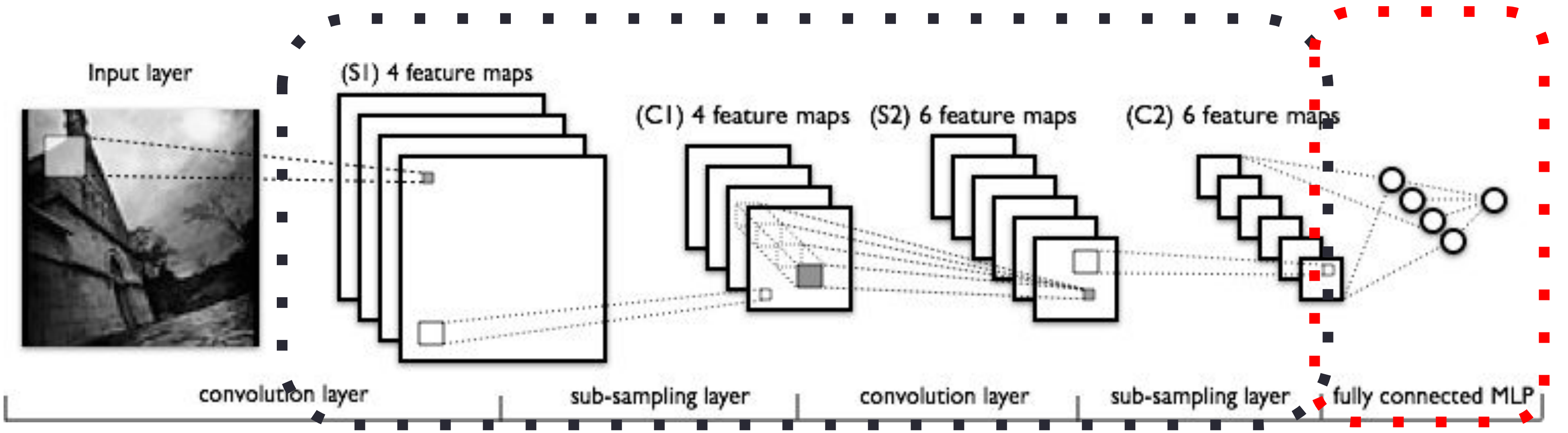
From the point of view of a network, these are two different things.

Pooling layers/Subsampling layers

- Combine different locations into one
- One possible method is to use a max
  - Interpretation: Yes, I found a cat somewhere



Convolution Neural Network (CNN)

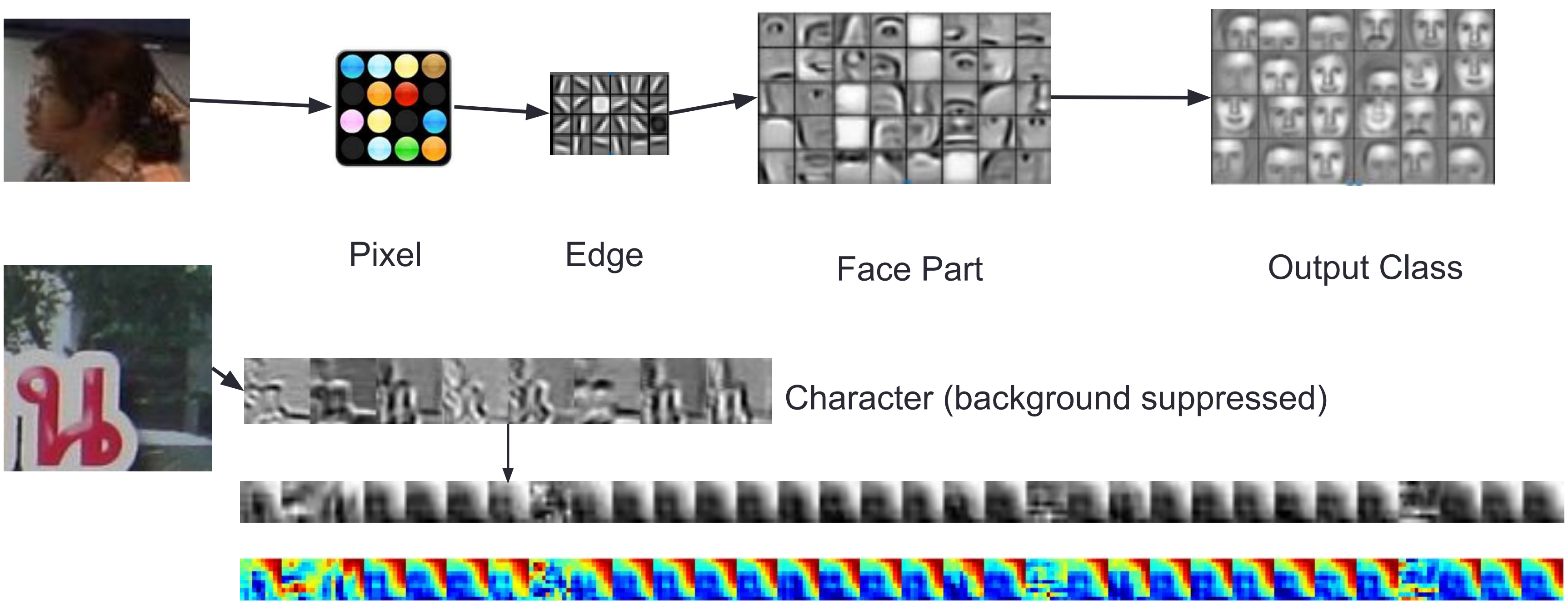


Trainable Feature Extractor Layer **Classification Layer**

Convolution layers followed by sub-sampling (pooling) layers
Output of each layer is called a feature map.

Convolution Neural Network (CNN)

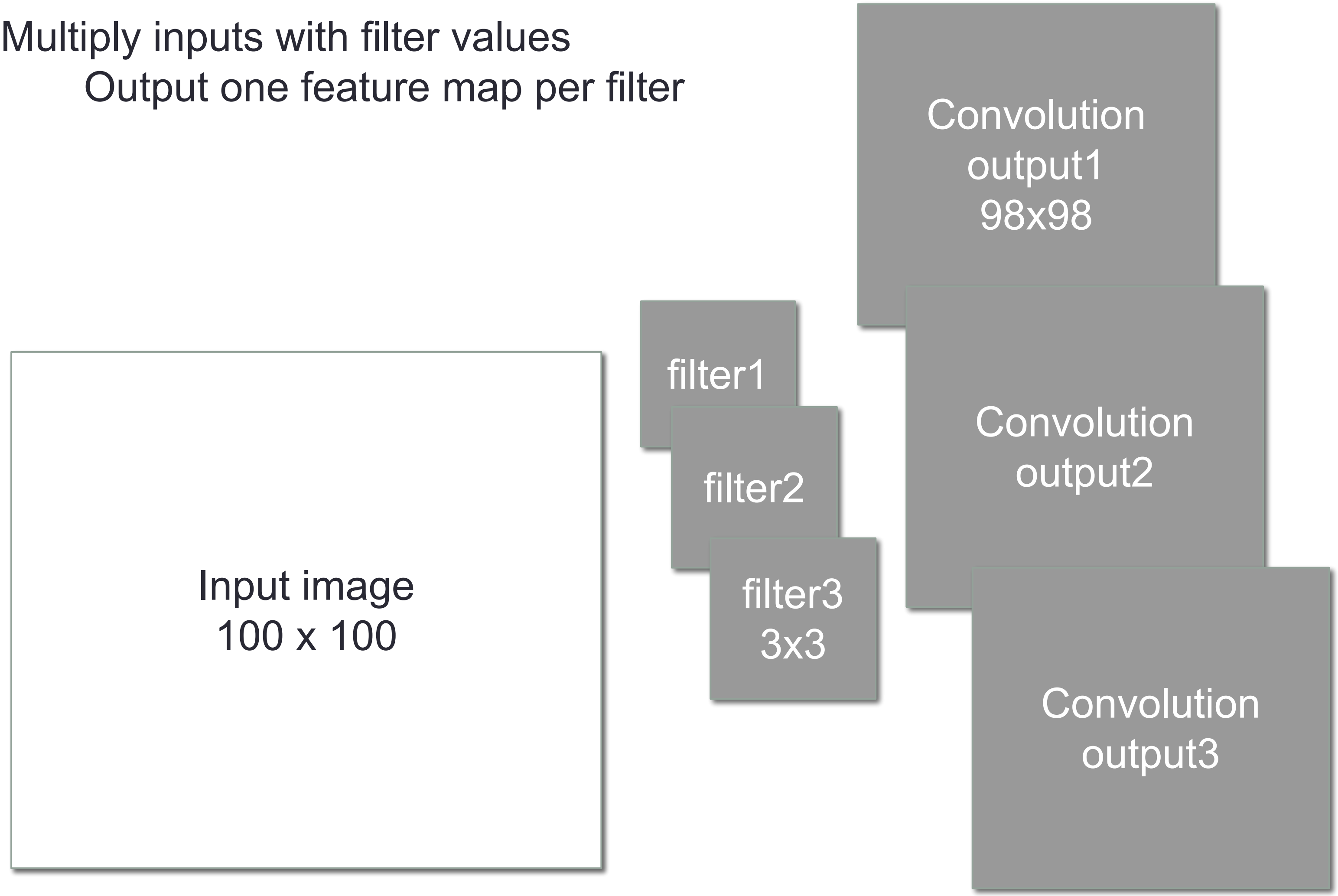
- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition: Pixel →edge → texture → part → object



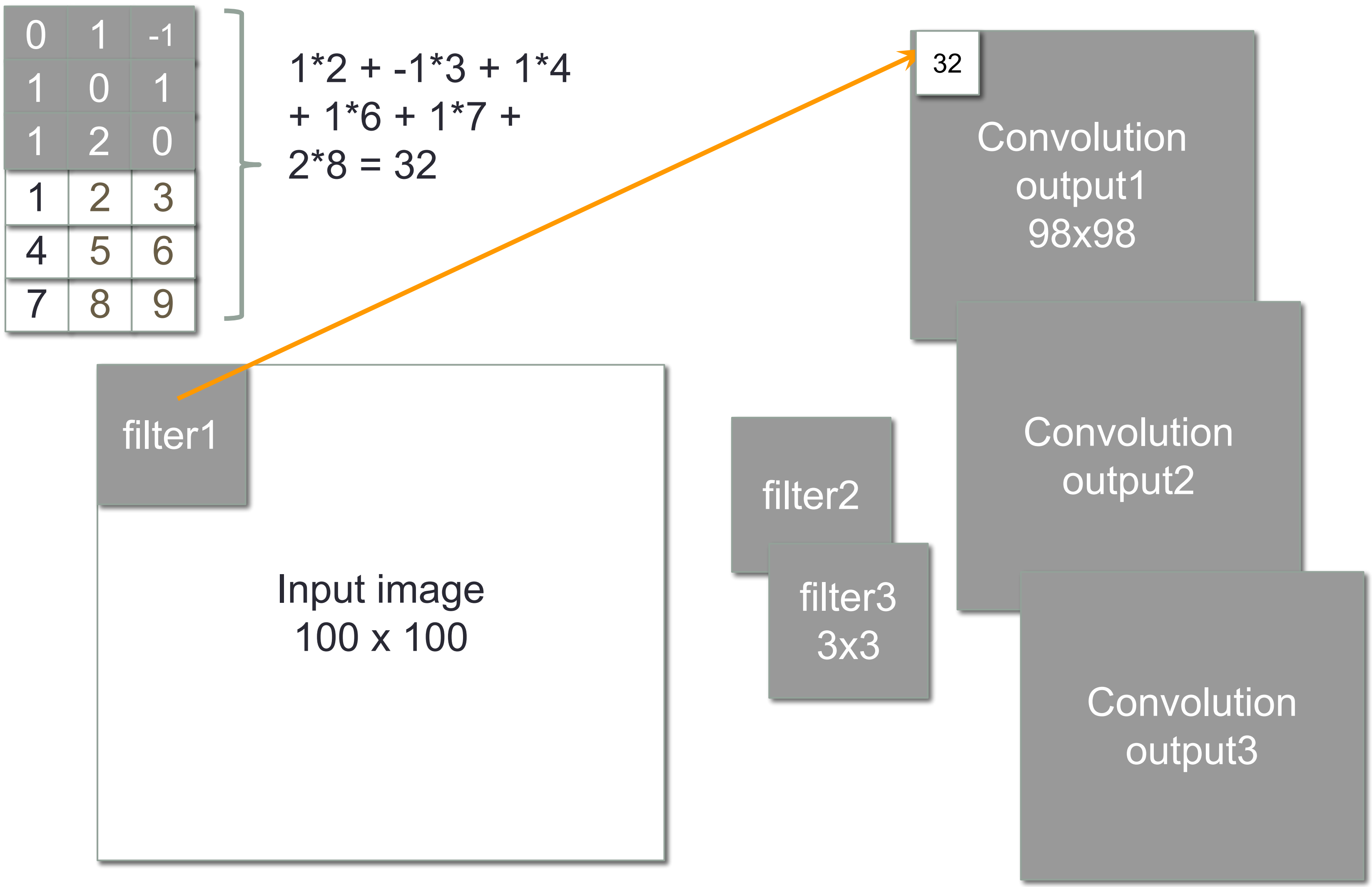
Convolution filters

Multiply inputs with filter values

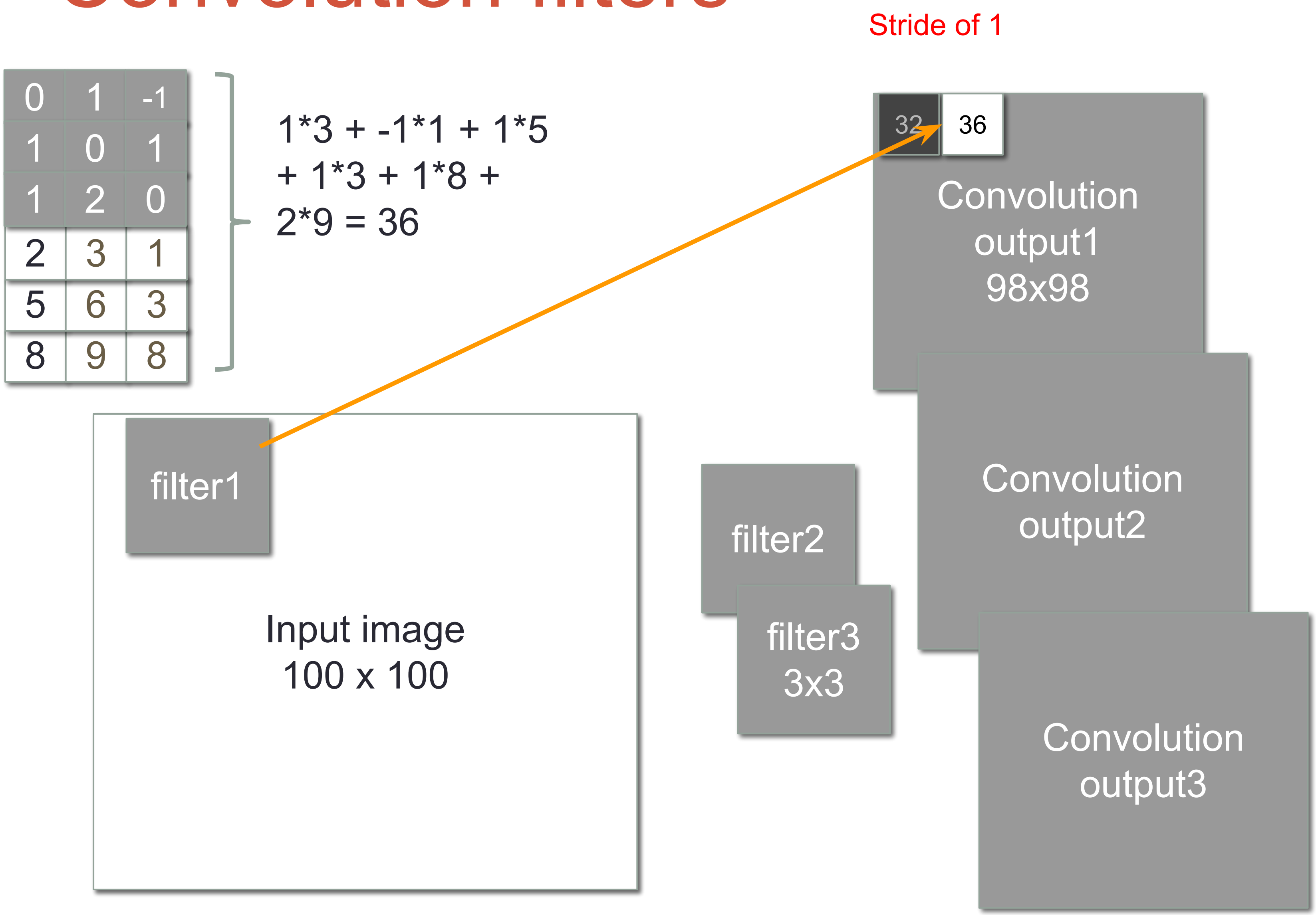
Output one feature map per filter



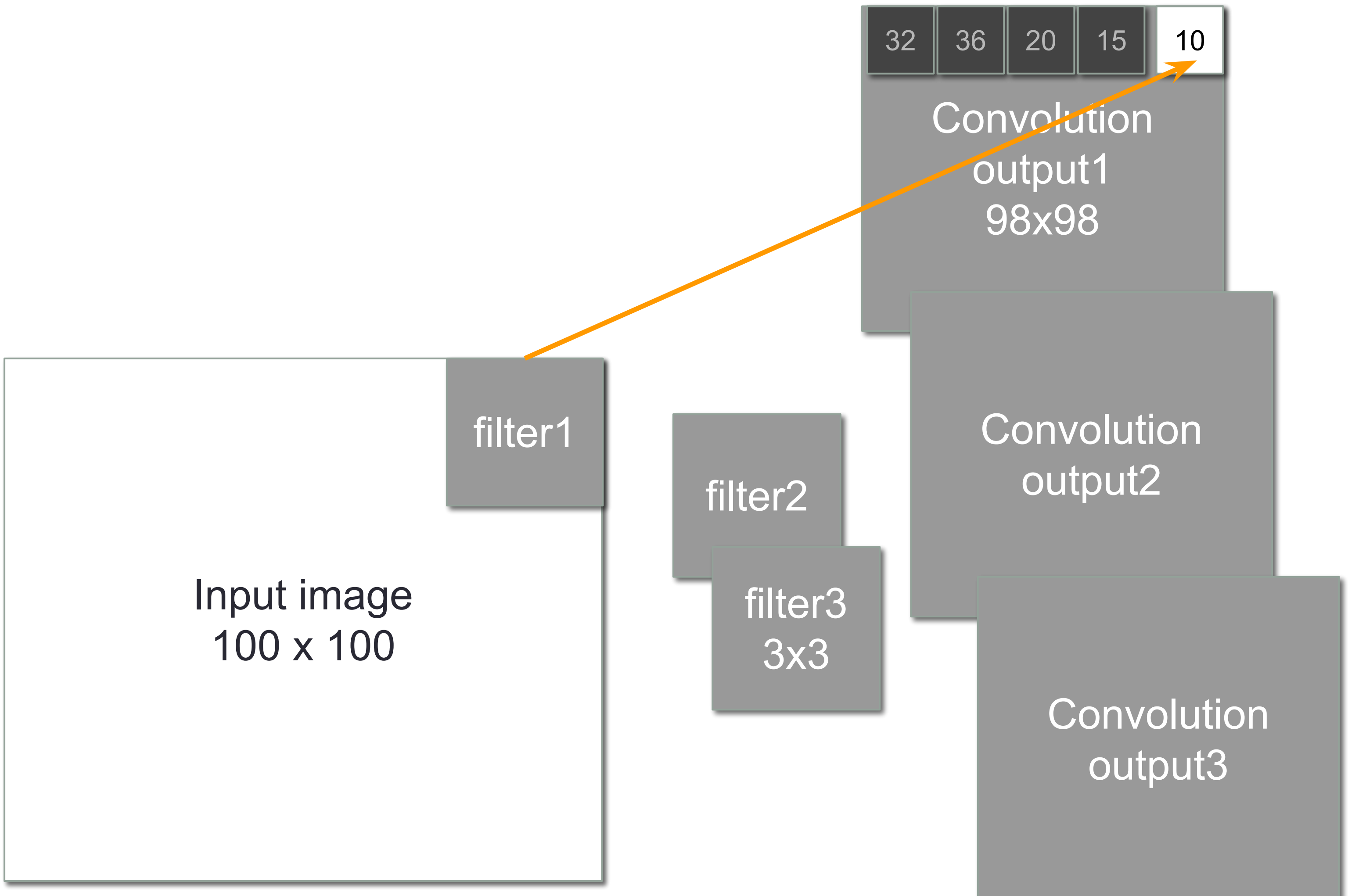
Convolution filters



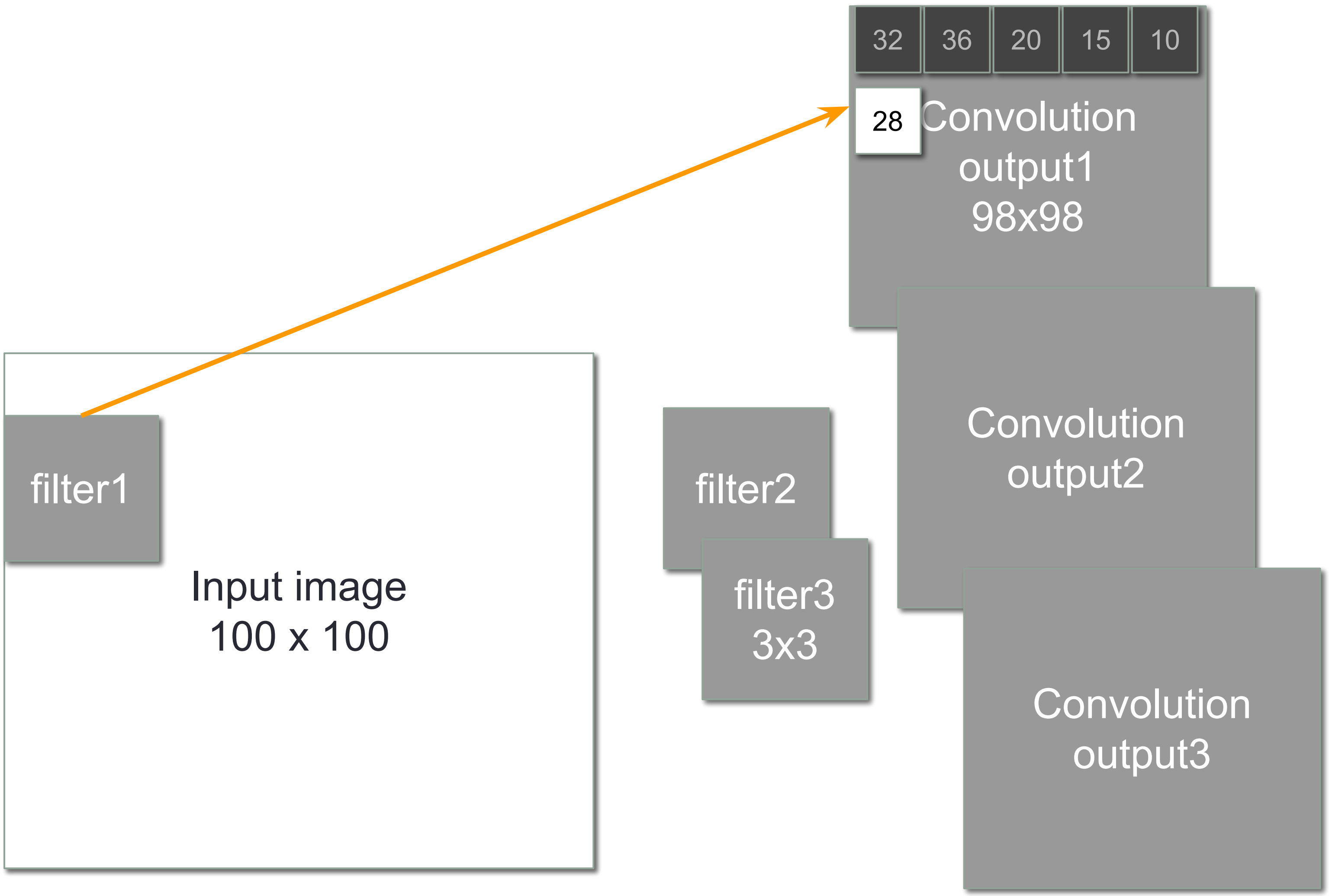
Convolution filters



Convolution filters

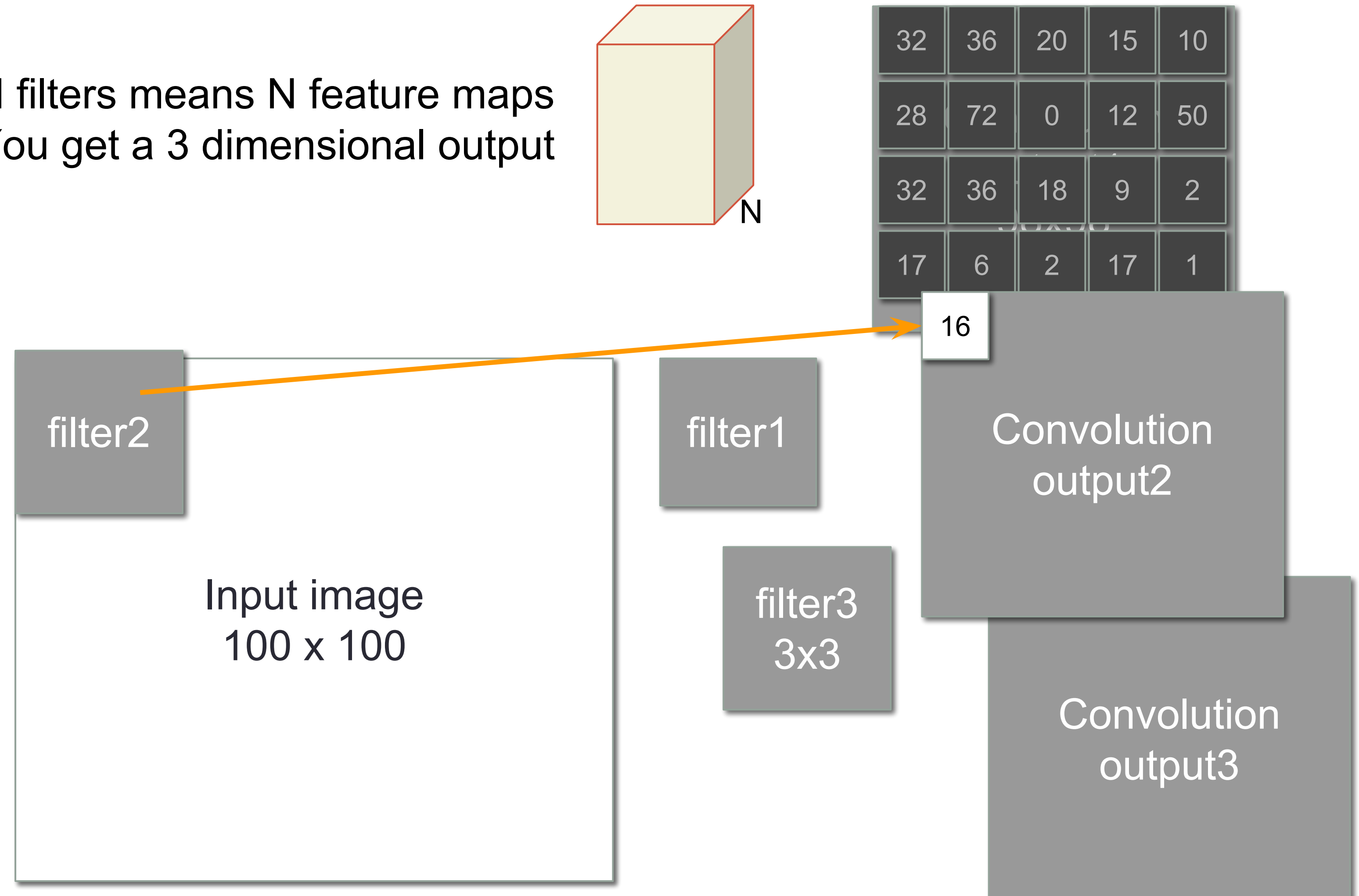


Convolution filters



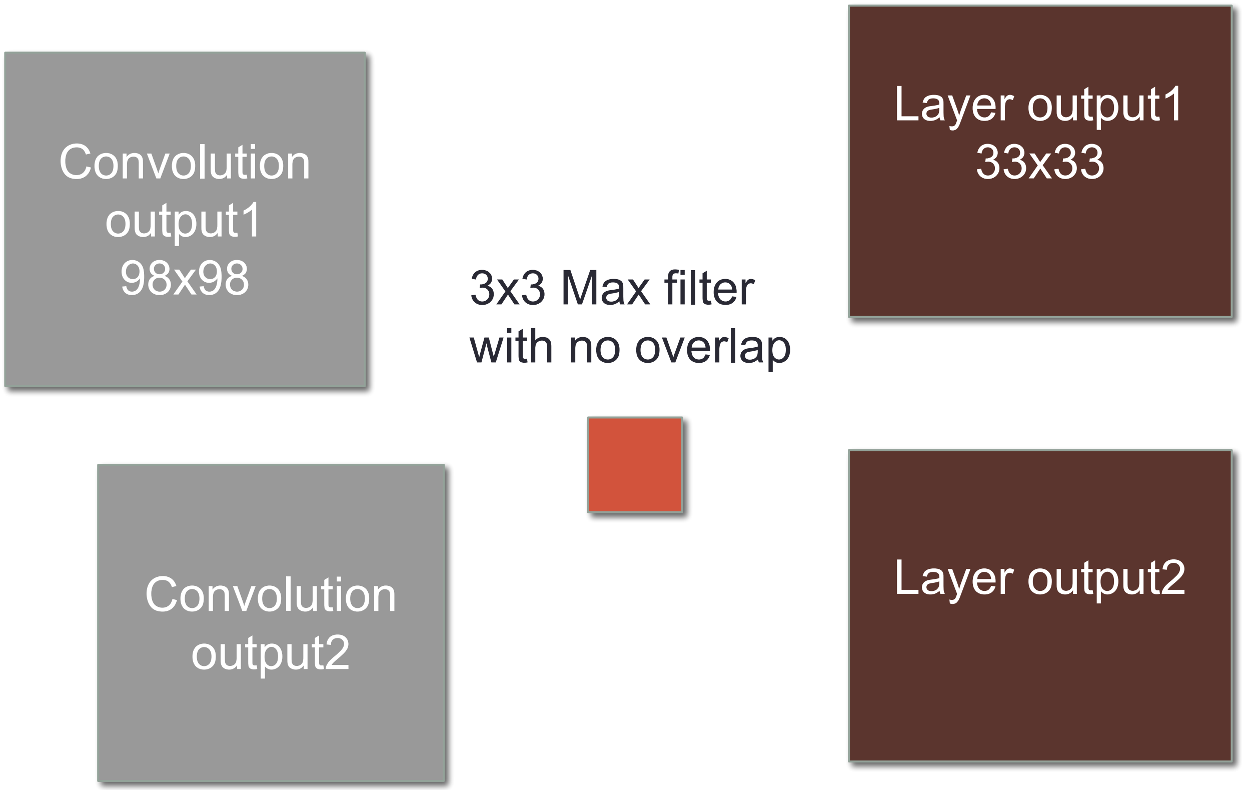
Convolution filters

N filters means N feature maps
You get a 3 dimensional output



Pooling/subsampling

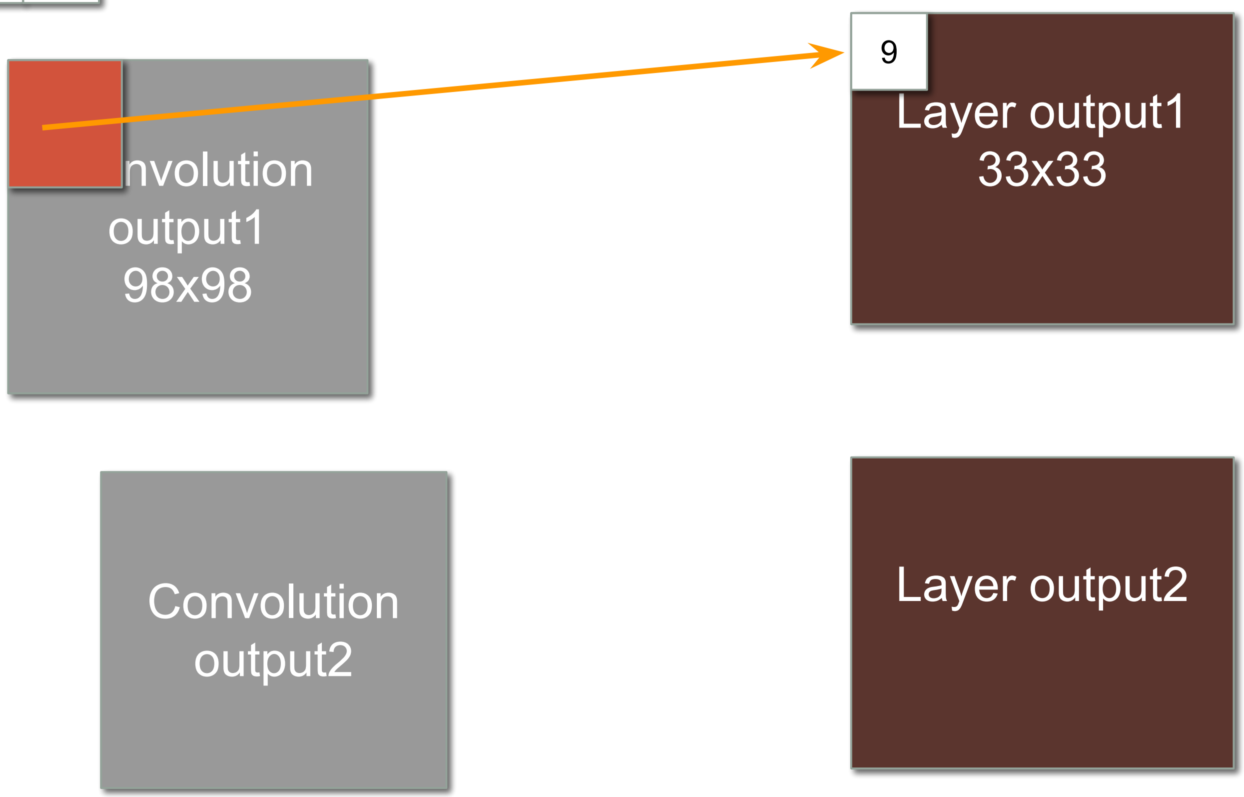
Reduce dimension of the feature maps



Pooling/subsampling

1	2	3
4	5	6
7	8	9

Max = 9

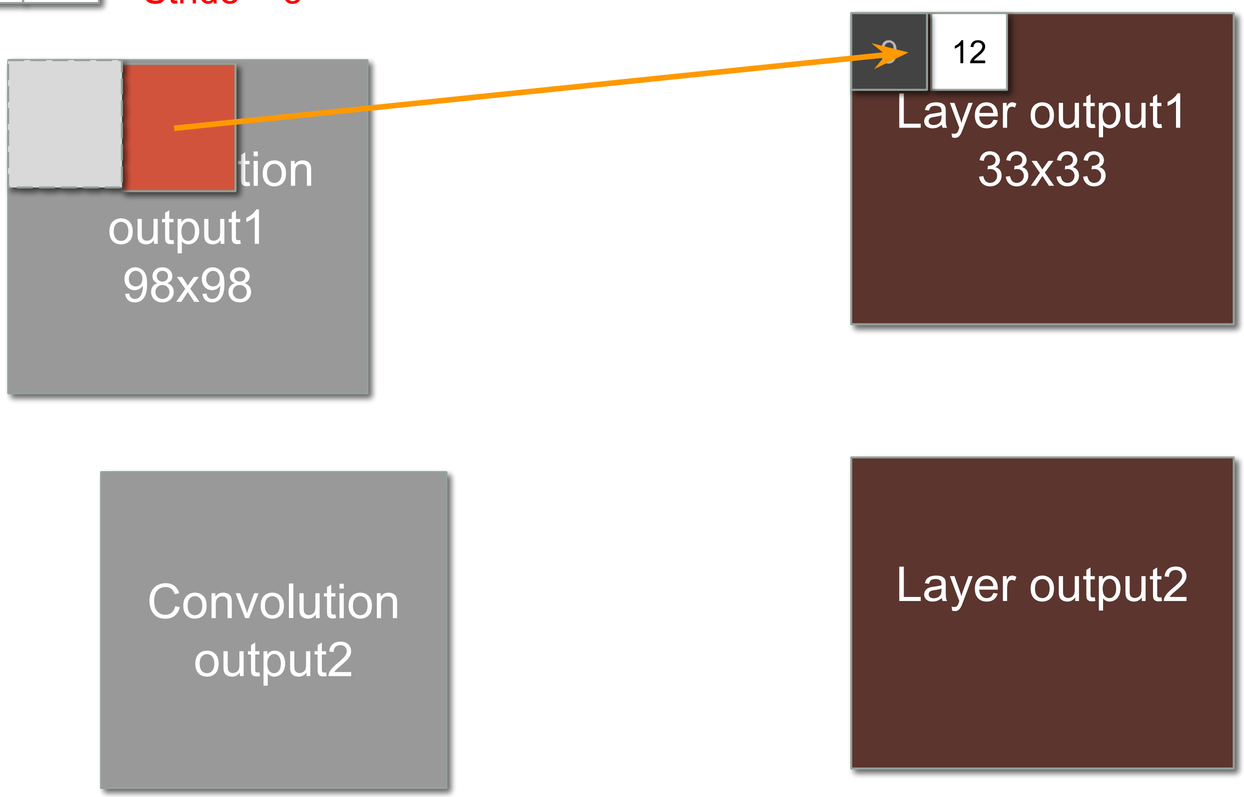


Pooling/subsampling

5	2	1
5	7	1
9	5	12

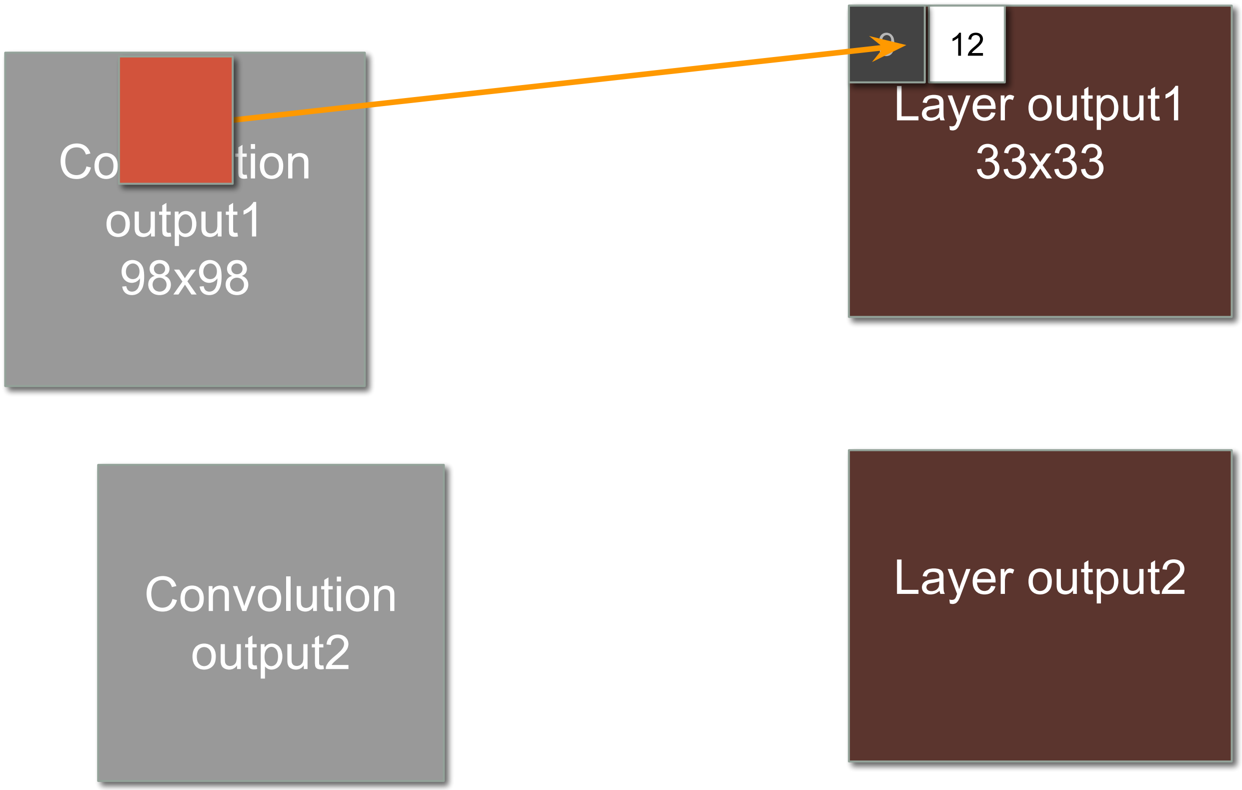
Max = 12

Stride = 3



Pooling/subsampling

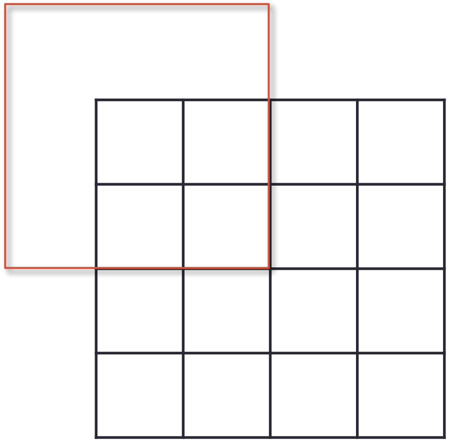
Can use other functions besides max
Example, average



Convolution puzzle

5 filters 3x3 filter pad, stride 1, pad 1

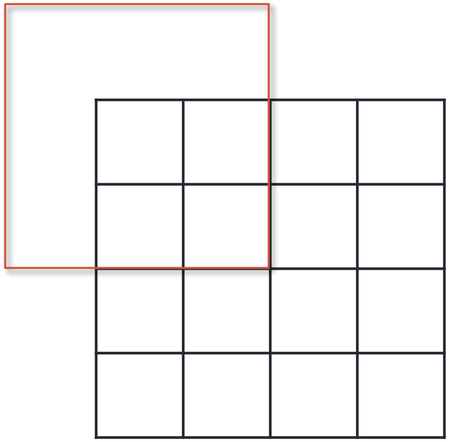
What is the output size?

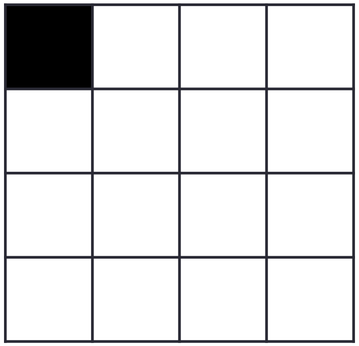


Convolution puzzle

5 filters 3x3 filter pad, stride 1, pad 1

What is the output size?

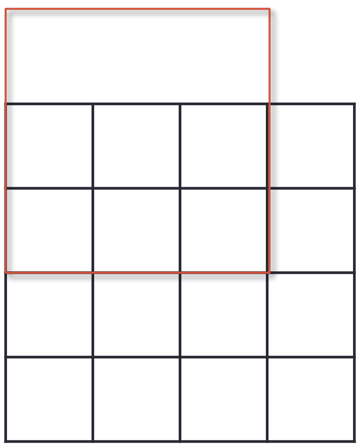


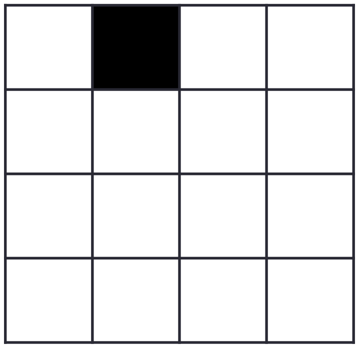


Convolution puzzle

5 filters 3x3 filter pad, stride 1, pad 1

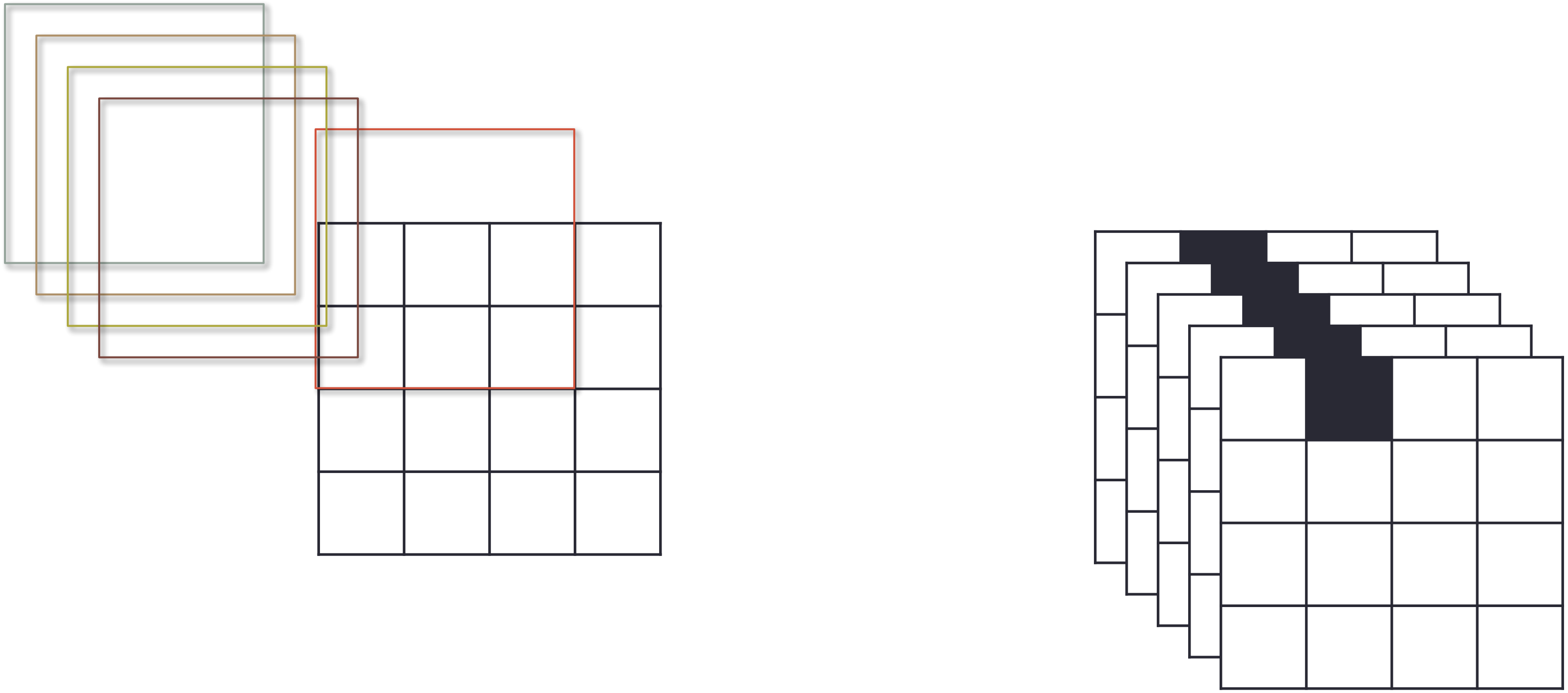
What is the output size?





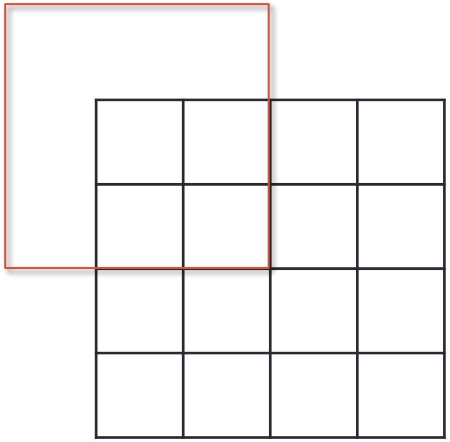
Convolution puzzle

5 filters 3x3 filter pad, stride 1, pad 1



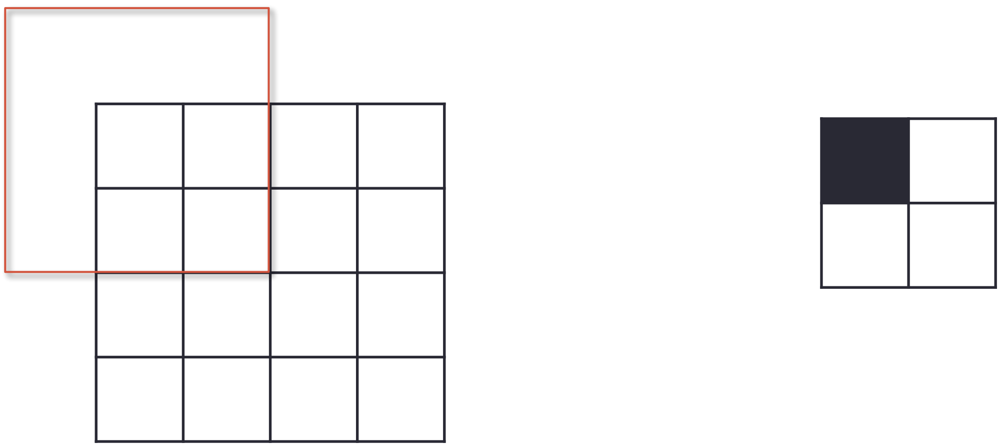
Convolution puzzle

3x3 filter pad, stride 2, pad 1



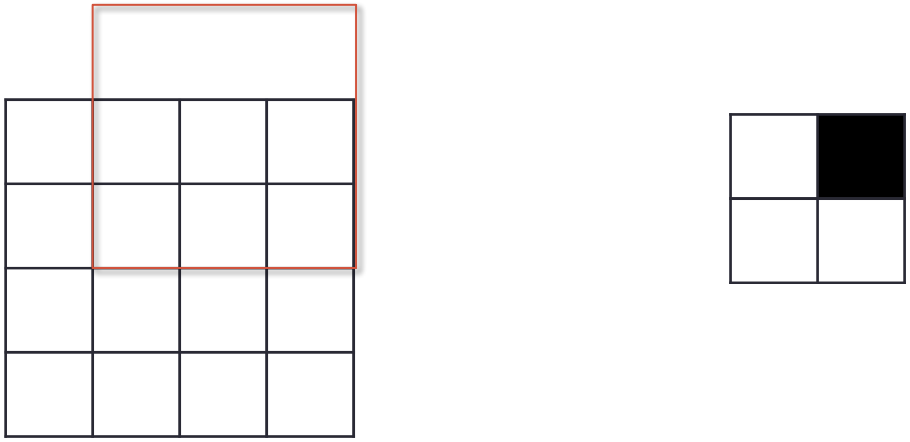
Convolution puzzle

3x3 filter pad, stride 2, pad 1



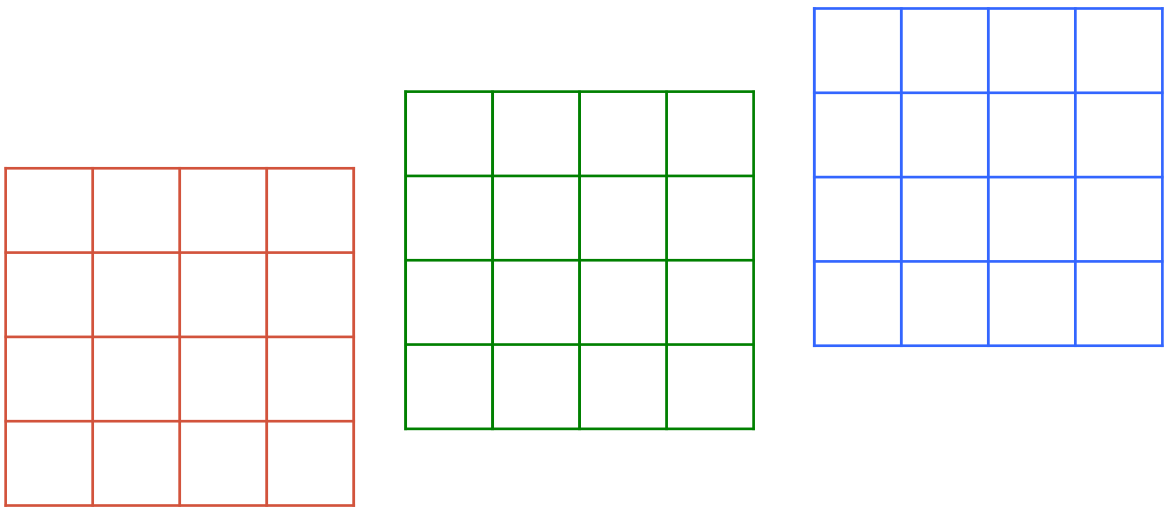
Convolution puzzle

3x3 filter pad, stride 2, pad 1



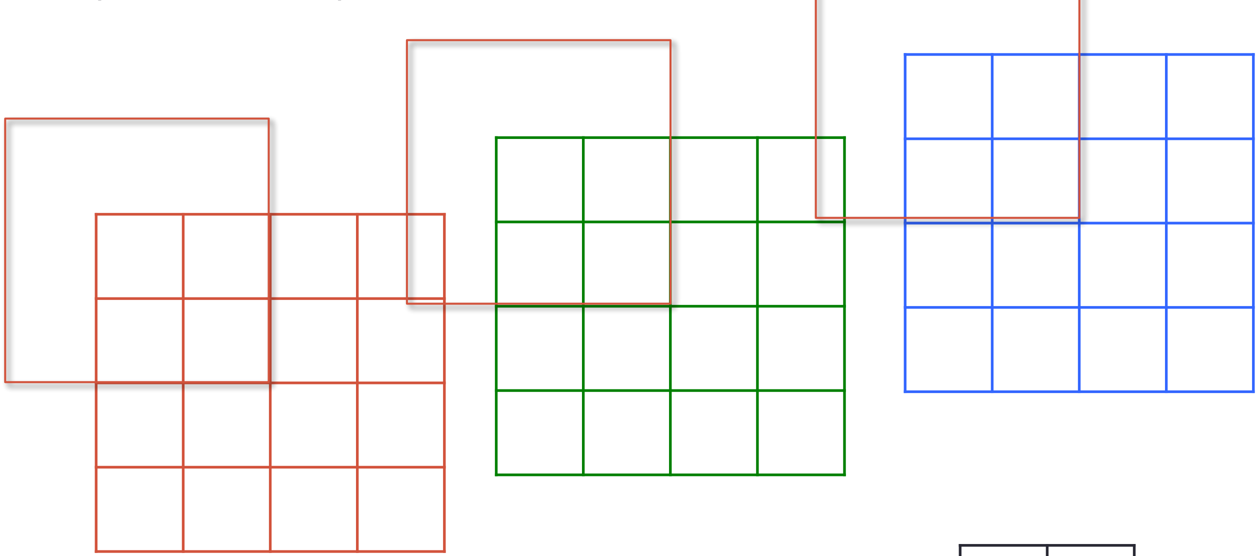
Convolution puzzle

RGB input (3 channels) 5 filters 3x3 filter pad, stride 2, pad 1

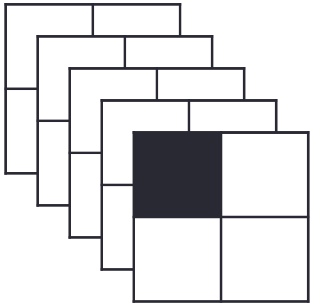


Convolution puzzle

RGB input (3 channels) 5 filters 3x3 filter pad, stride 2, pad 1



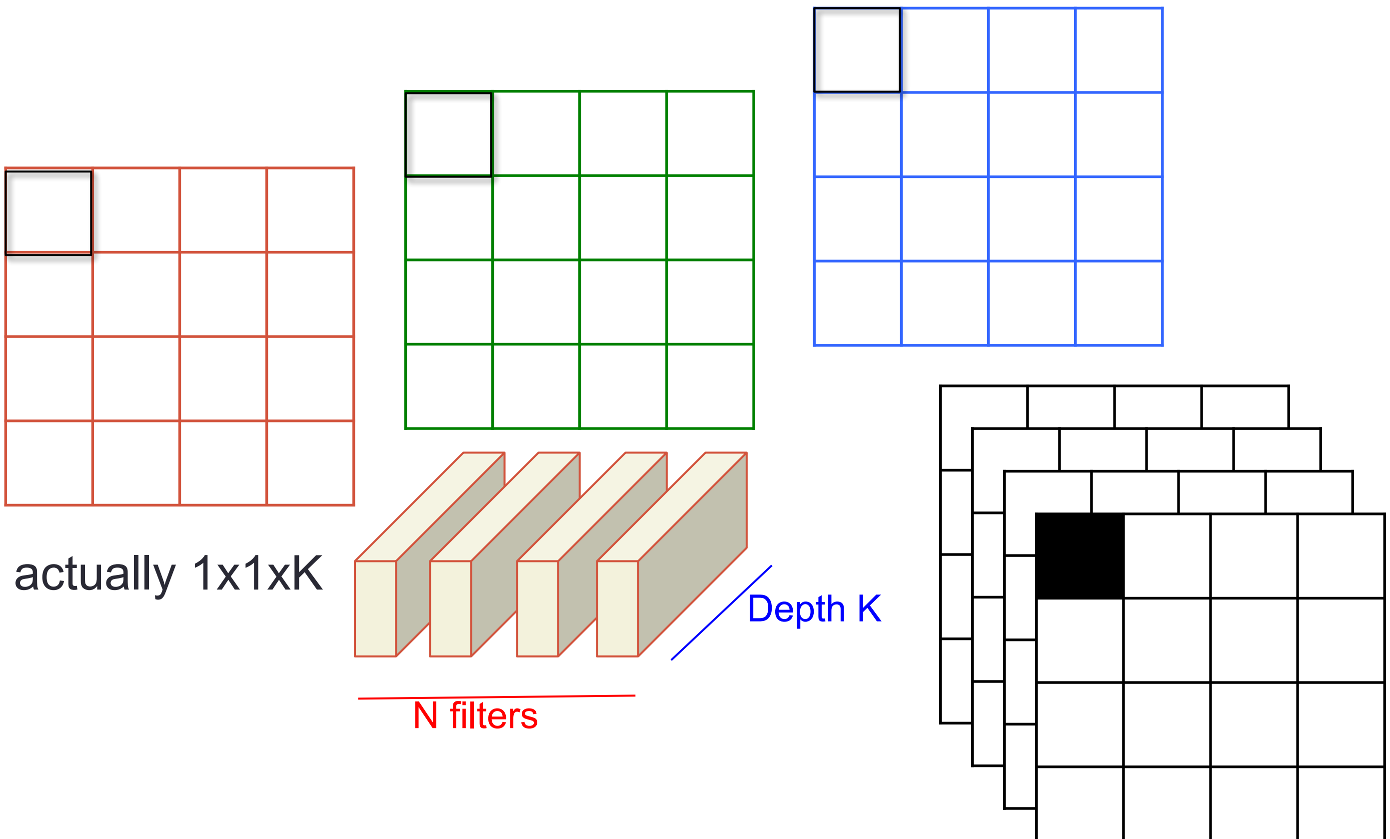
The filter is actually 3x3x3



1x1 convolution

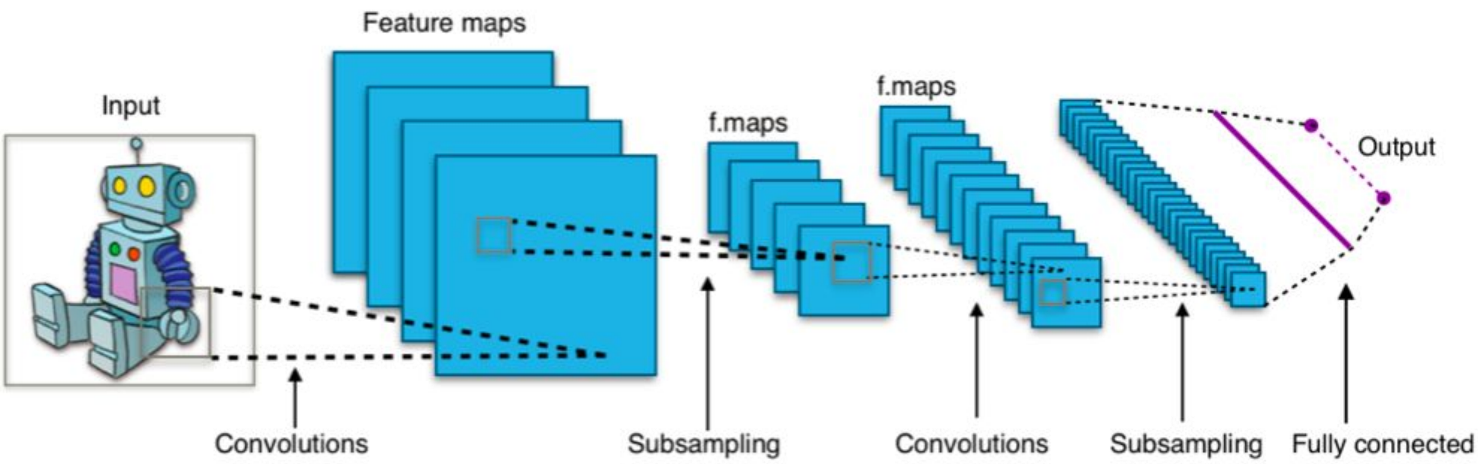
Reduces the dimension of feature maps

The filter is actually 1x1xK



CNN overview

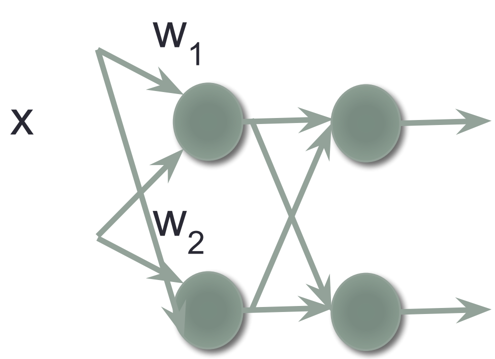
- Filter size, number of filters, filter shifts, and pooling rate are all parameters
- Usually followed by a fully connected network at the end
  - CNN is good at learning low level features
  - DNN combines the features into high level features and classify



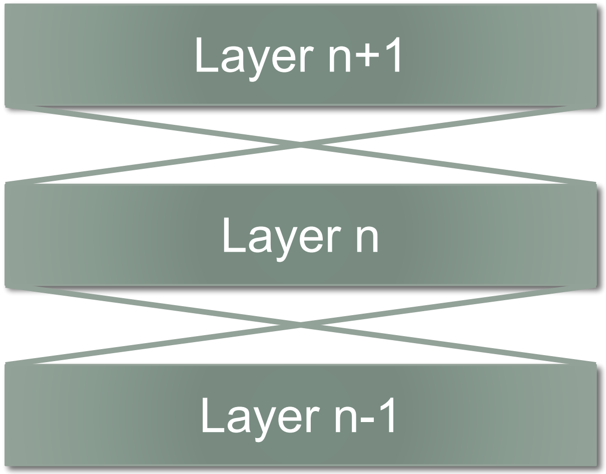
https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

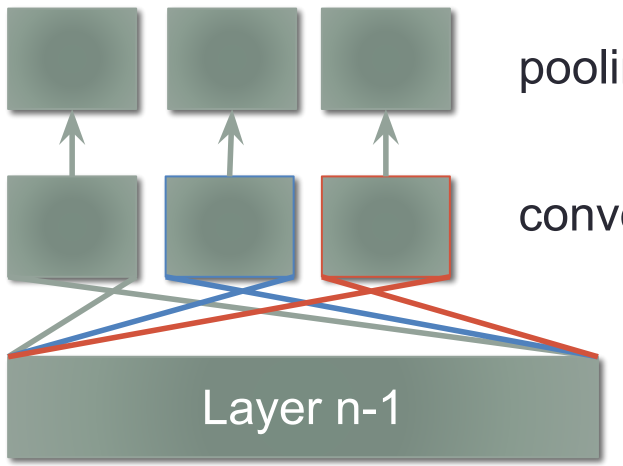
Parameter sharing in convolution neural networks

- $W^T x$



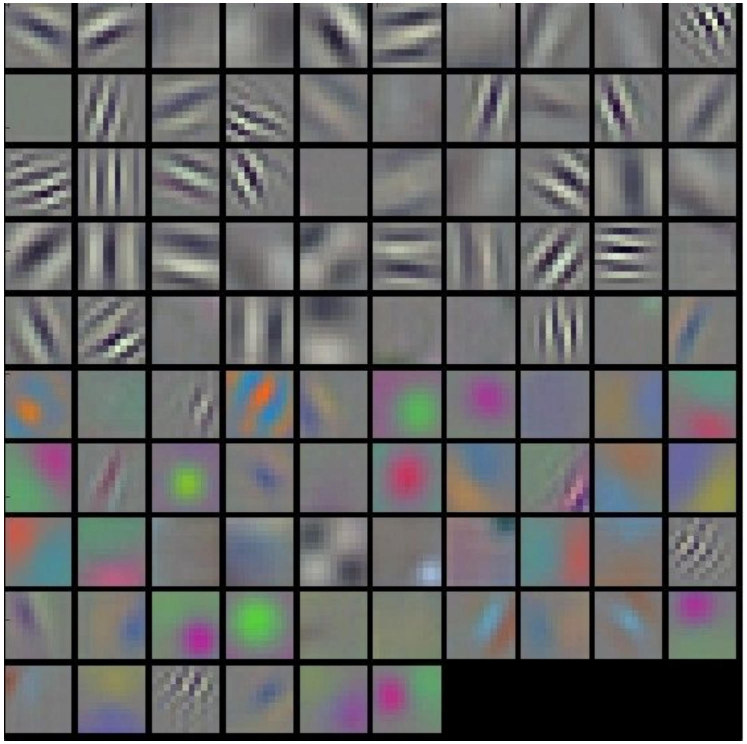
- Cats at different location might need two neurons for different locations in fully connect NNs.
- CNN shares the parameters in 1 filter
- The network is no longer fully connected





Visualizing convolutional layers

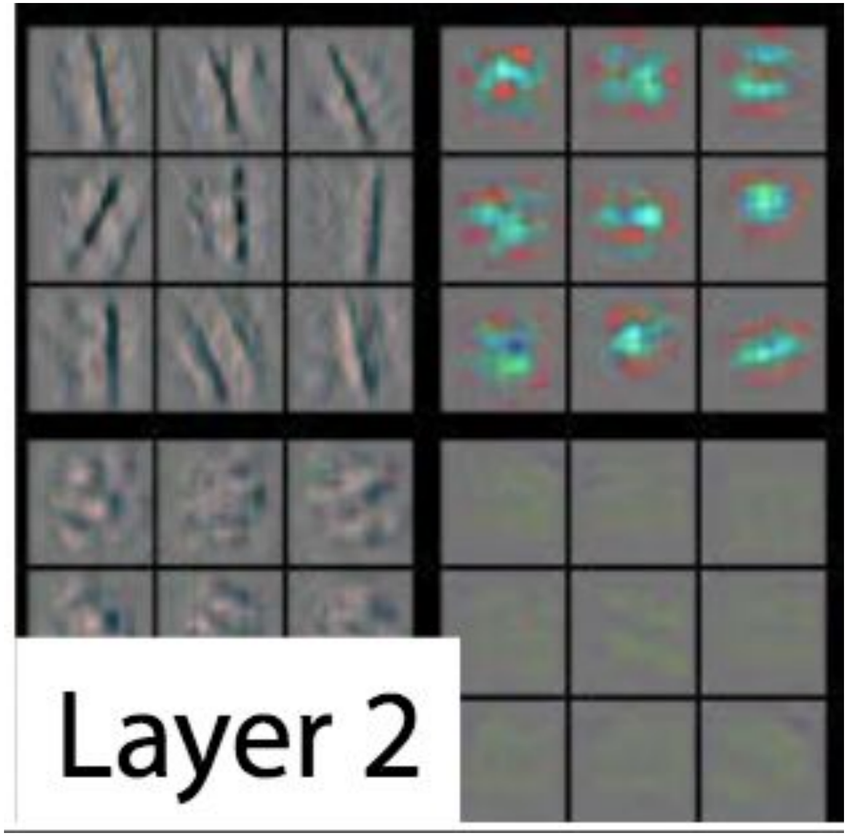
- Just like PCA, we can visualize the weights of a transform
- “Matched filters”



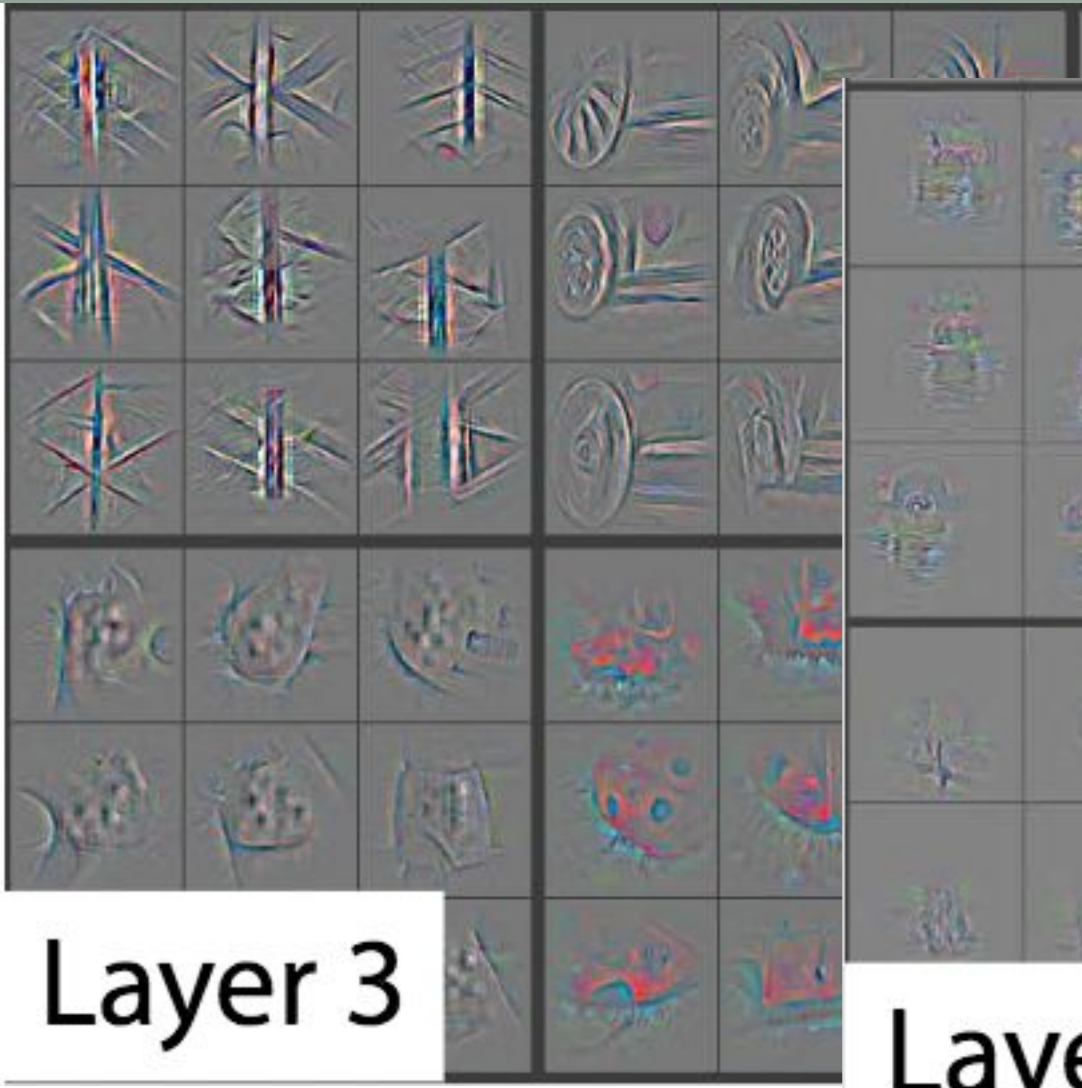
http://cs231n.github.io/understanding-cnn/

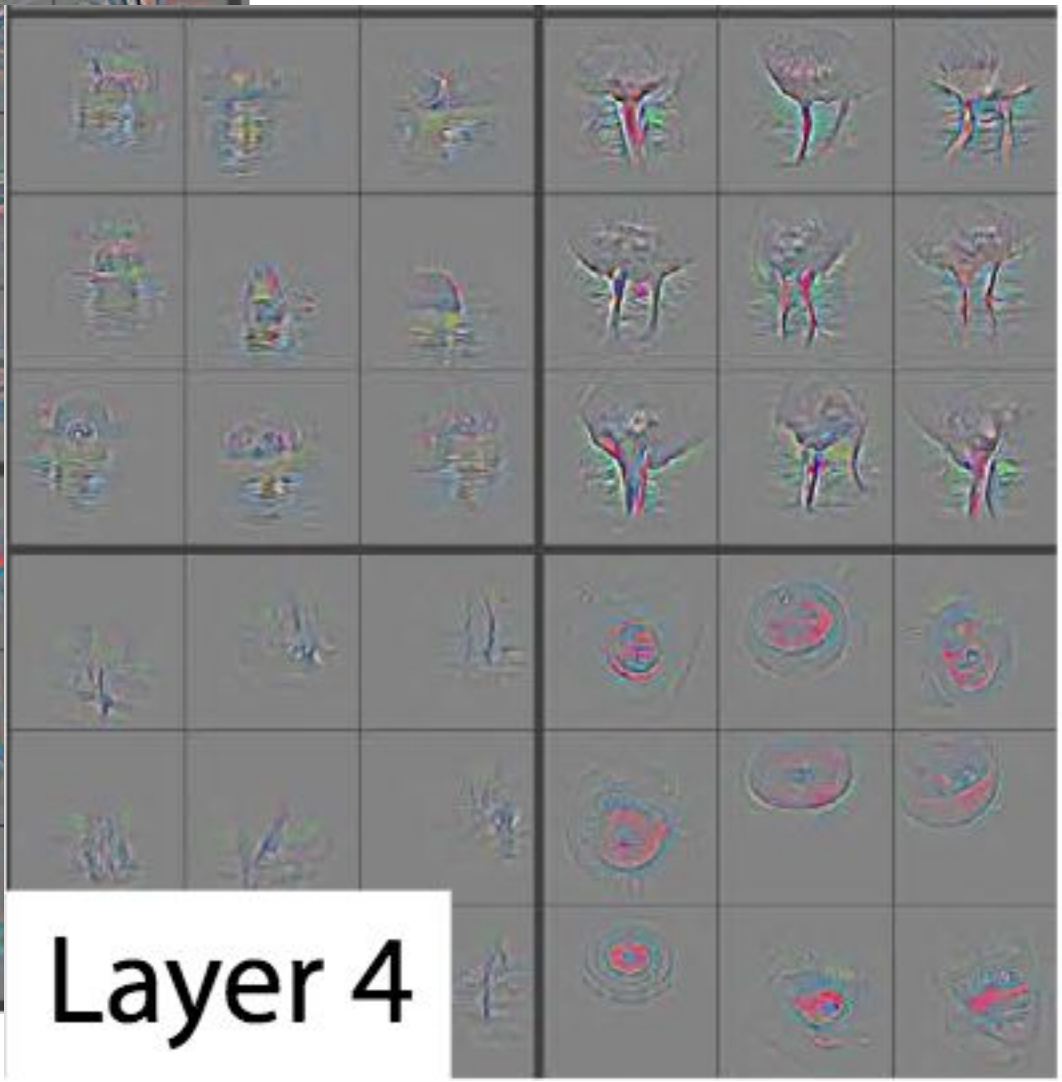
Higher layer captures higher-level concepts

Layer 1



https://arxiv.org/pdf/1311.2901.pdf


Layer 3


Layer 4

Pooling/subsampling

- Max filter -> Maxout
  - Backward pass?
    - Gradient pass through the maximum location, 0 otherwise

Common schemes

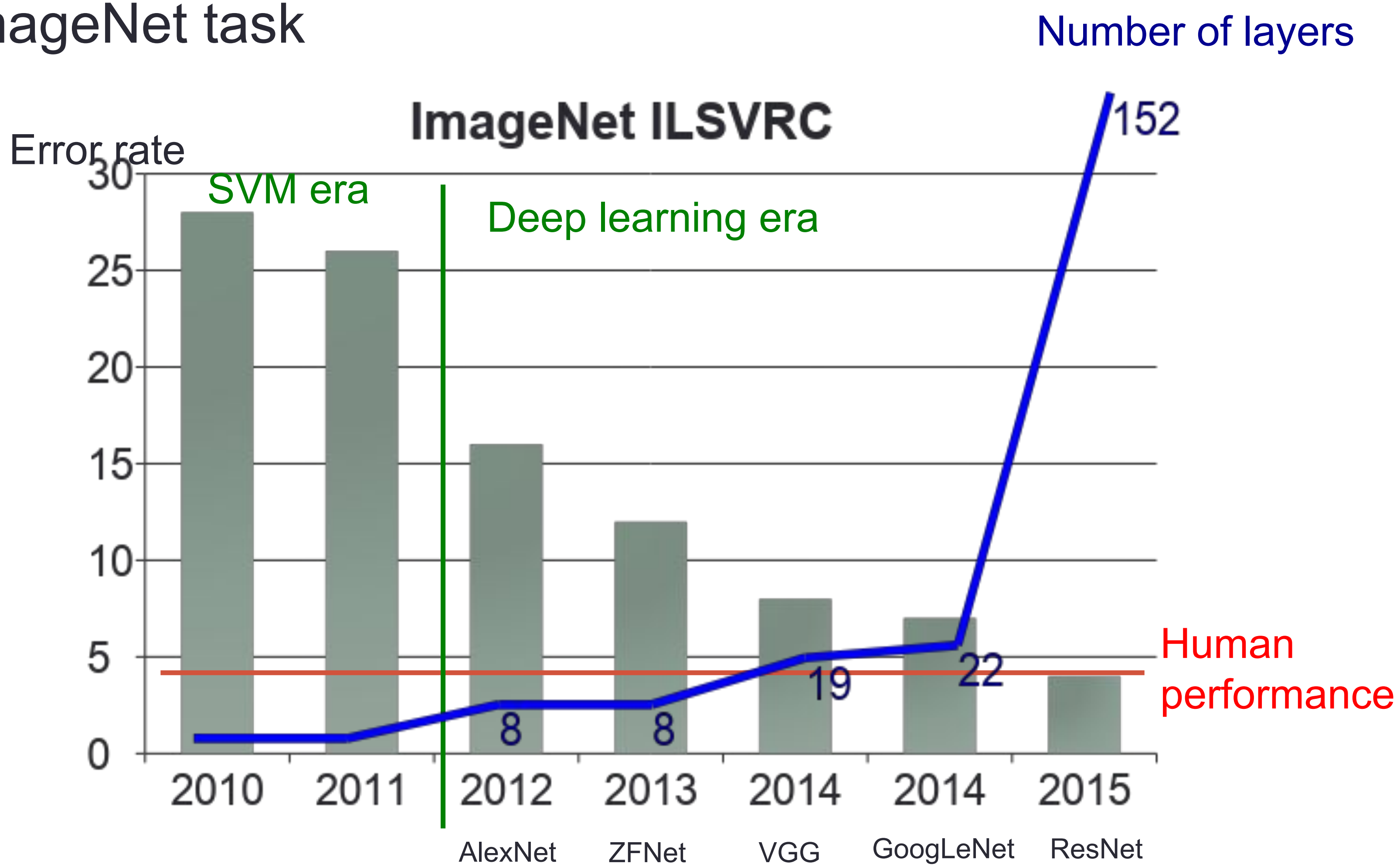
- INPUT -> [CONV -> RELU -> POOL]*N -> [FC -> RELU]*M -> FC

- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*N -> [FC -> RELU]*M -> FC

- If you working with images, just use a winning architecture.

A brief history of imagenet architectures

- ImageNet task



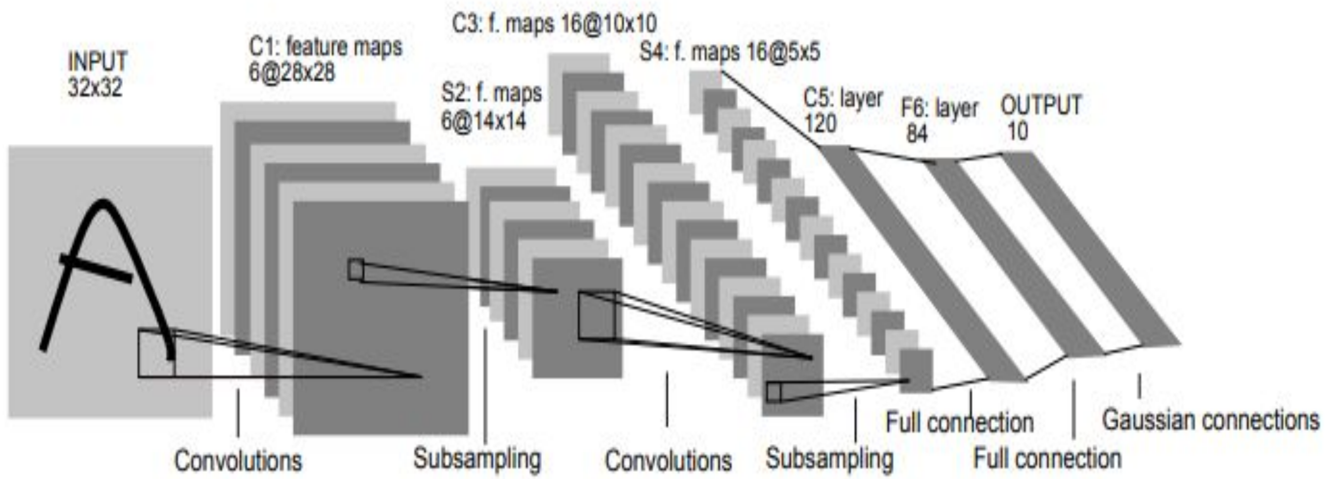
Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

LeNet

Convolutions and poolings followed by fully connected layers

Tanh activations

Ability to handle larger images limited by compute

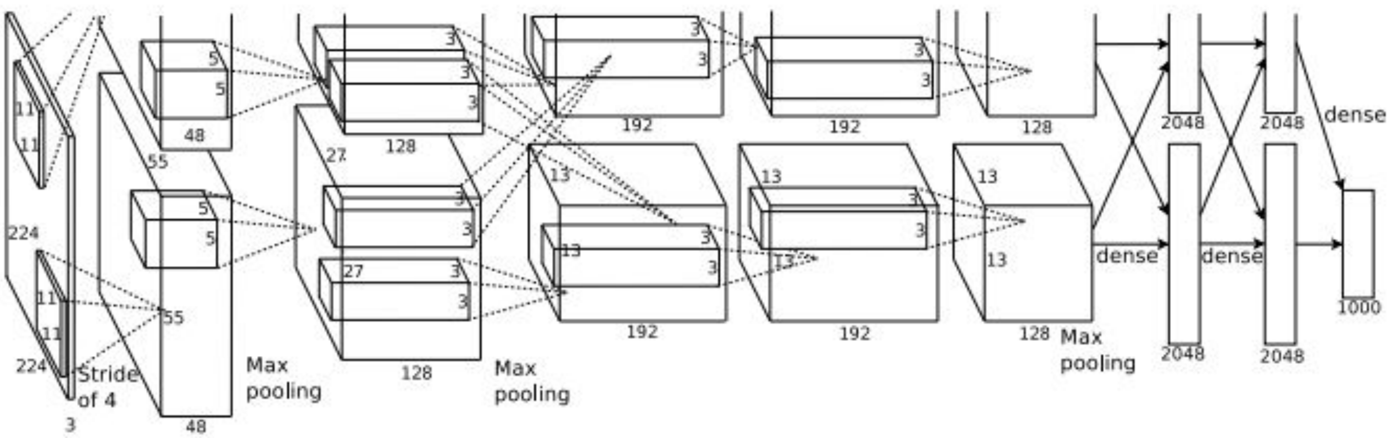


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition,” 1998.

AlexNet

Convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum

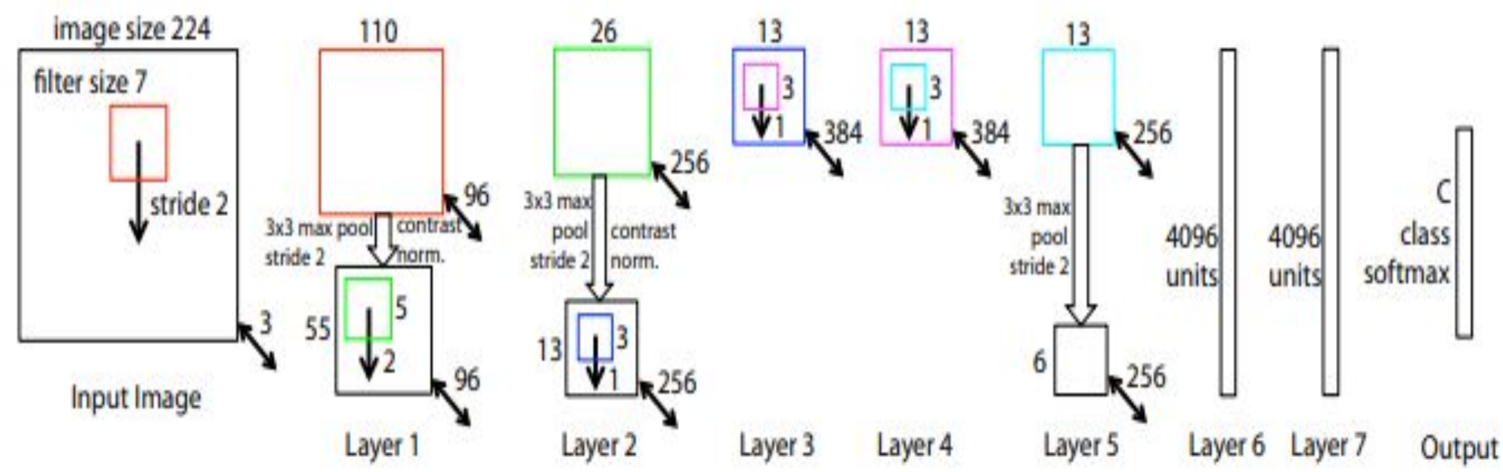
Two pipelines to fit into two GPUs



Alex Krizhevsky, et al. “ImageNet Classification with Deep Convolutional Neural Networks”, 2012.

ZFNet

Tweaking hyperparameters



Matthew D. Zeiler, Rob Fergus, “Visualizing and Understanding Convolutional Networks,” 2013

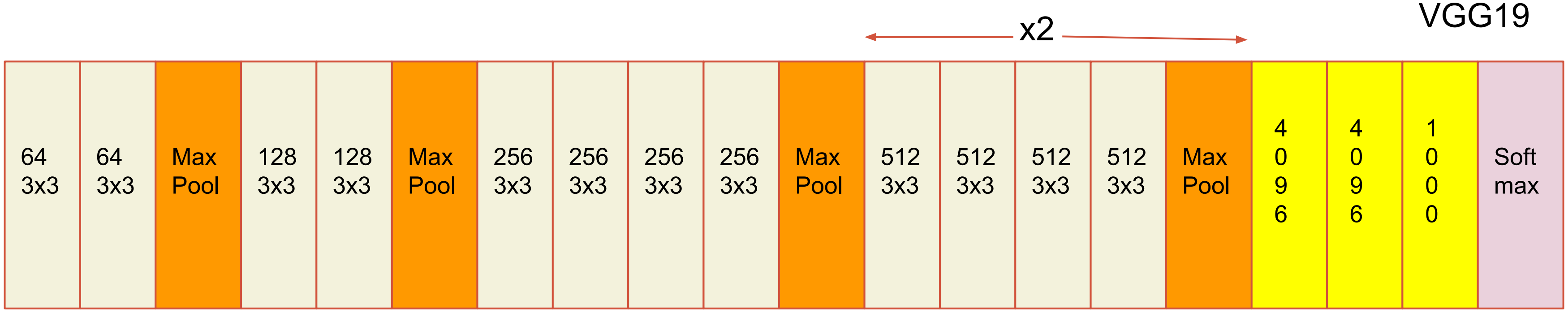
VGG

Uniform 3x3 convolutional filters

19 layers! Pushing the limits of conventional wisdom at that time.

Used by many since pre-train weights are publically available

VGG19



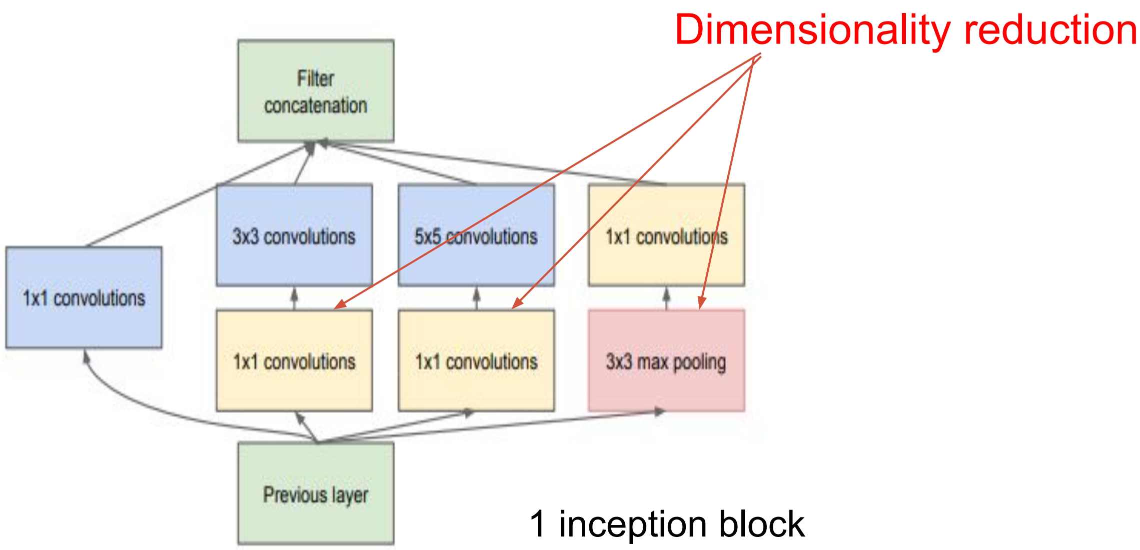
Karen Simonyan, Andrew Zisserman"Very Deep Convolutional Networks for Large-Scale Image Recognition" 2014

GoogLeNet (Inception v1)

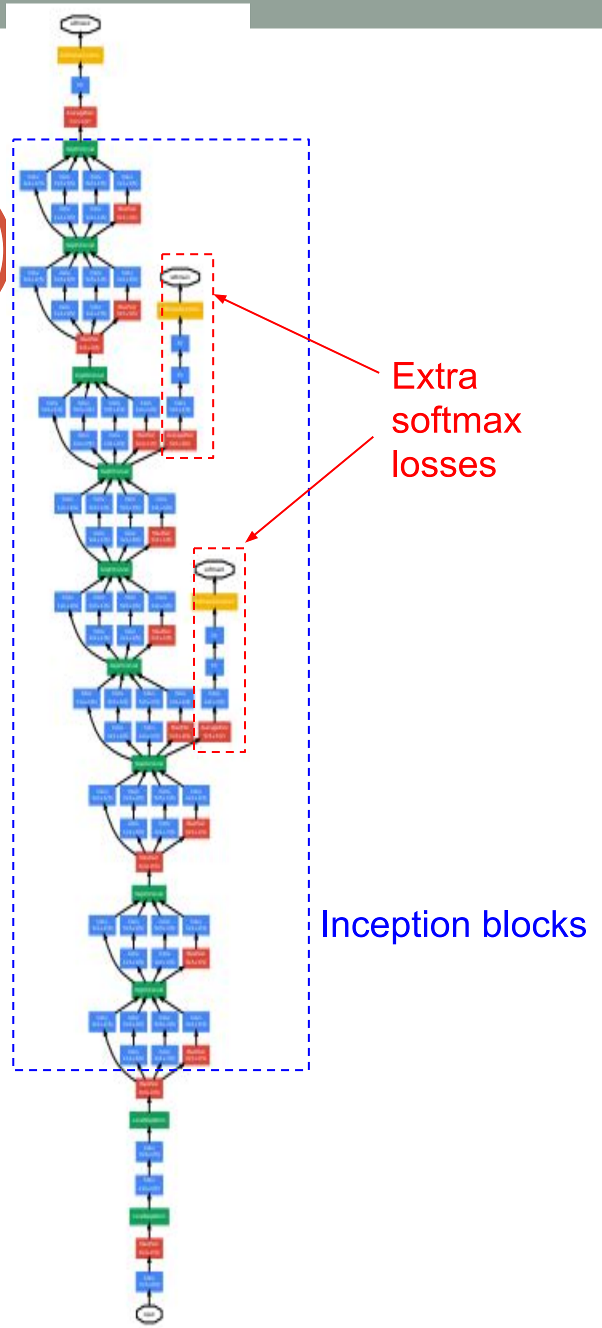
Multiple filter sizes per layer (objects come in different scales)

Dimensionality reduction via 1x1 convolution

Multiple softmax losses to help the gradient problem



1 inception block

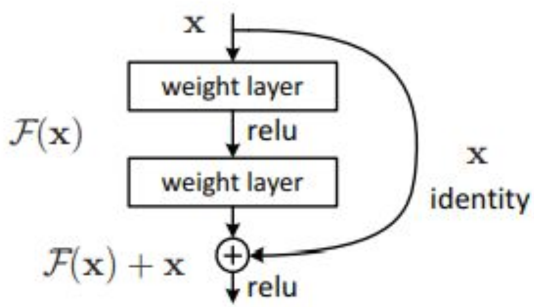


Christian Szegedy, et al. “Going Deeper with Convolutions,” 2014

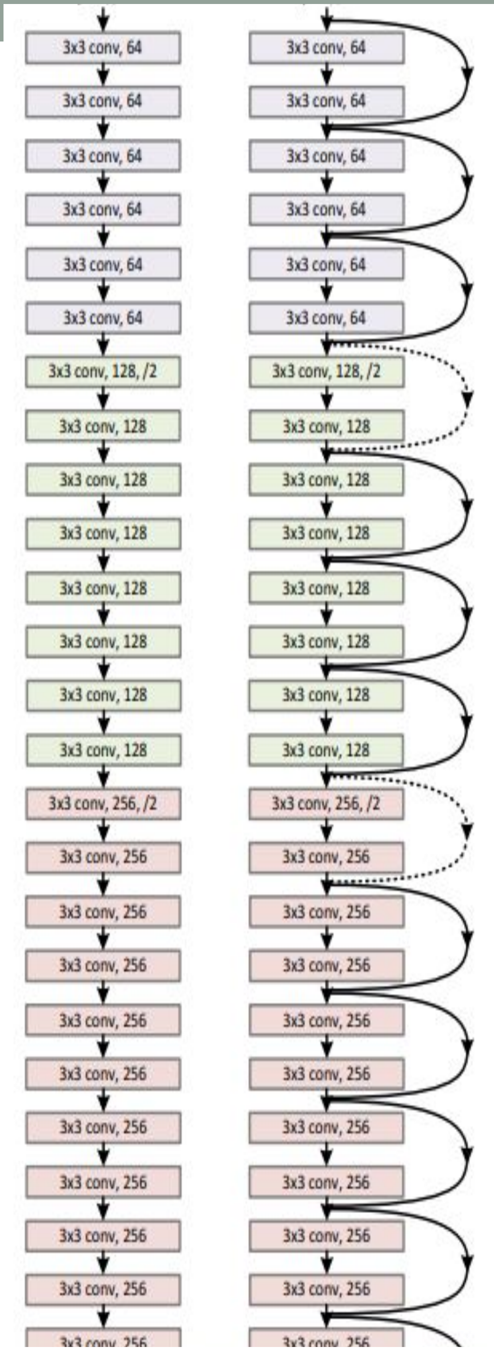
ResNet (Residual Network)

Batch norm

Extra “skip connections” to reduce the vanishing gradient problem

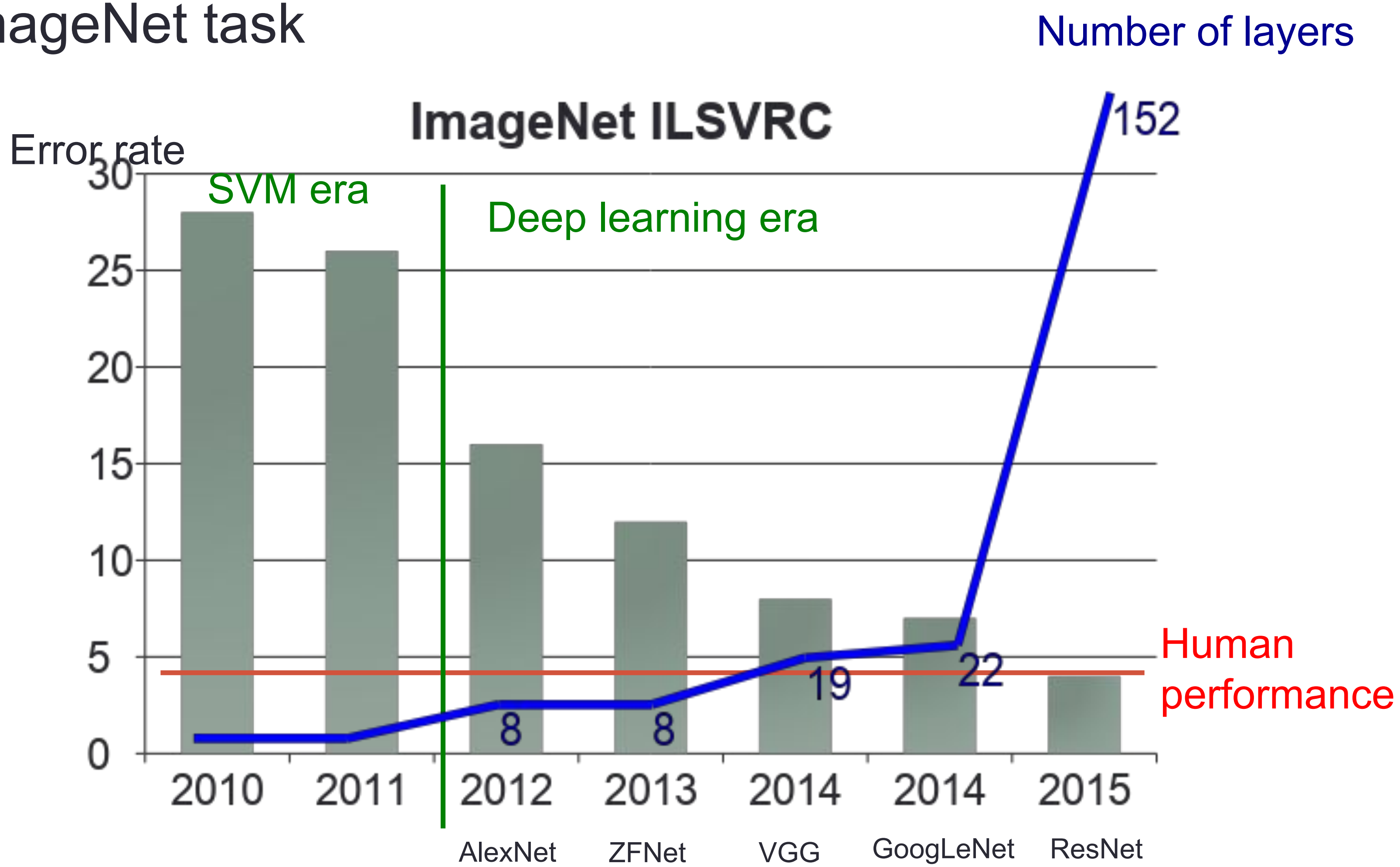


Kaiming He, et al. “Deep Residual Learning for Image Recognition” 2015



A brief history of imagenet architectures

- ImageNet task



Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

Inception v2+v3

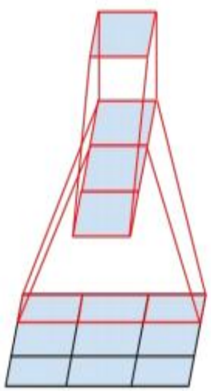
Implement 5x5 with two 3x3s

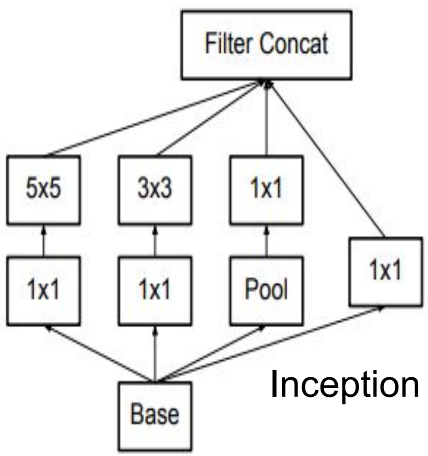
Factorized convolution

3x3 -> 3x1 and 1x3

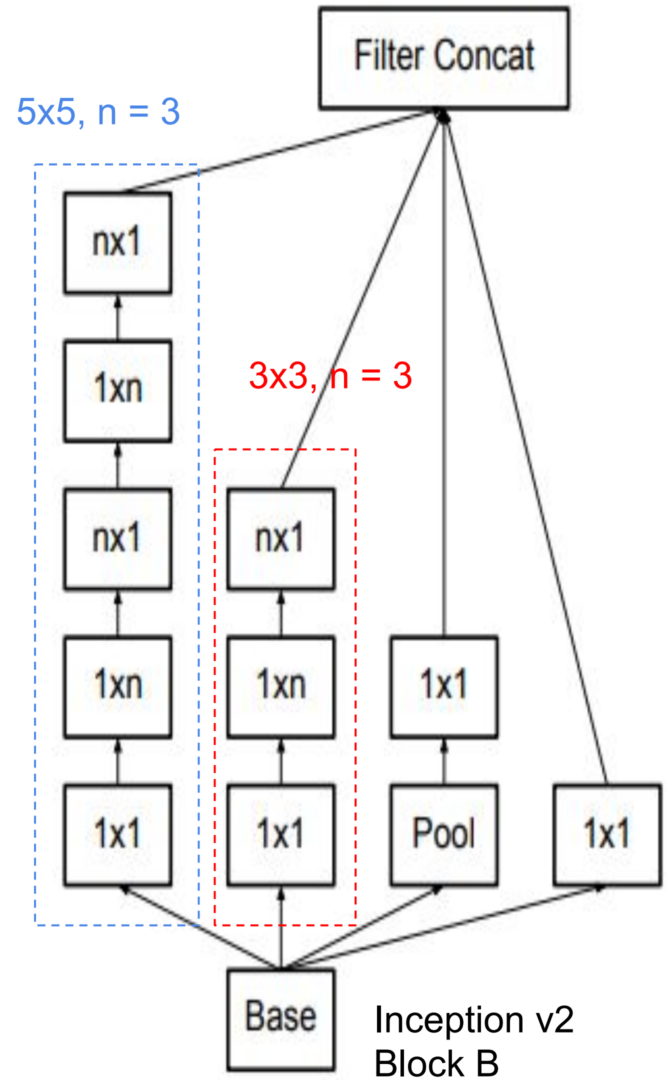
3 types of inception blocks

RMSprop, Batch norm, label smoothing





Inception v1



Inception v2
Block B

Christian Szegedy, “Rethinking the Inception Architecture for Computer Vision,” 2015

Xception

Depthwise separable convolution: two-step convolution

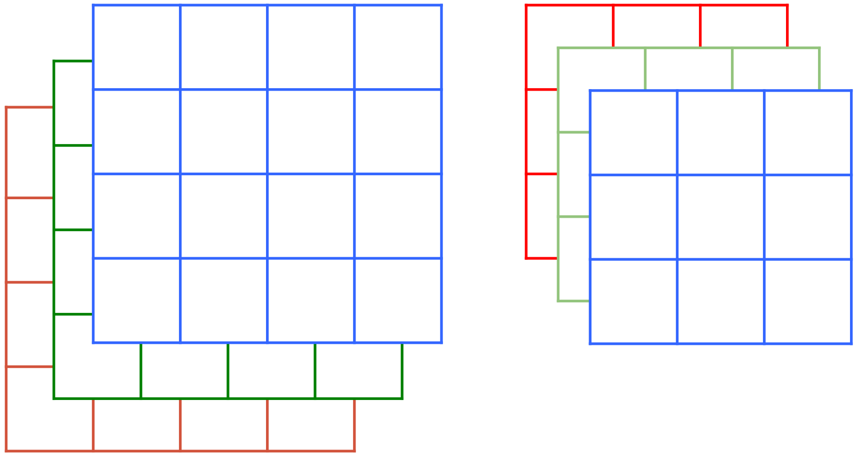
1. Depthwise convolution
2. 1x1 convolution

Typical convolution 3x3 filter is 3x3xinput channel

Depthwise convolution 3x3 filter is 3x3x1
1 filter per input channel

input

One regular convolution filter



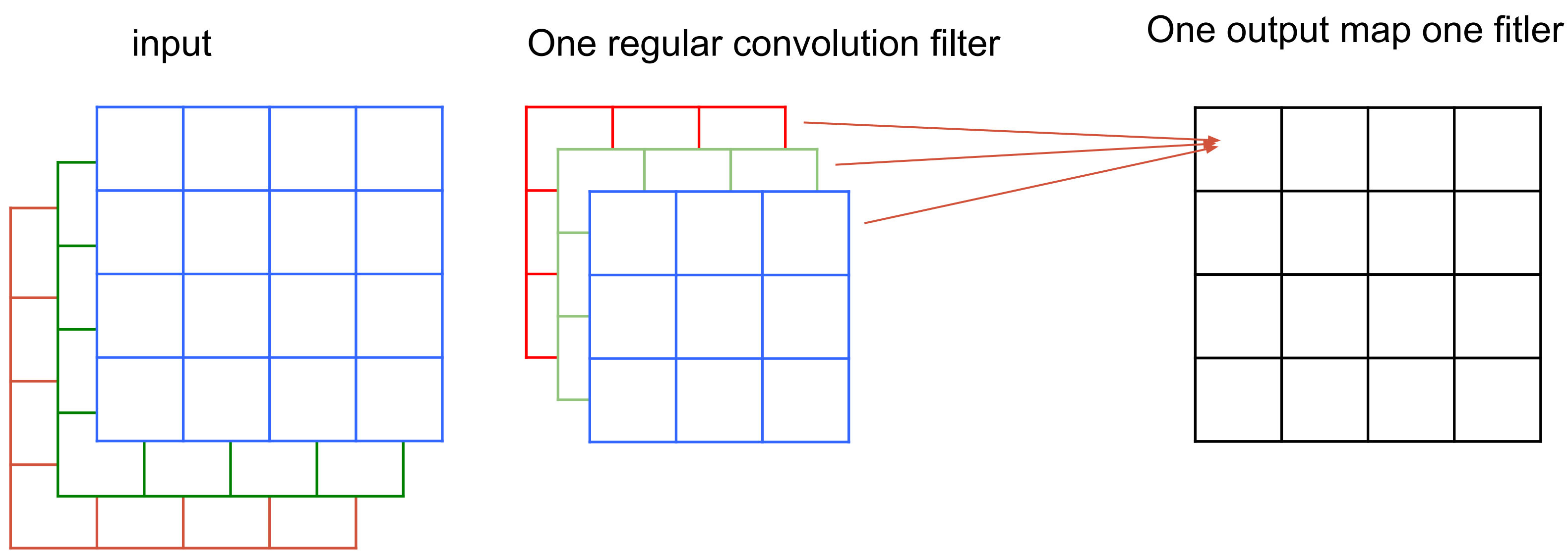
Xception

Depthwise separable convolution: two-step convolution

1. Depthwise convolution
2. 1x1 convolution

Typical convolution 3x3 filter is 3x3xinput channel

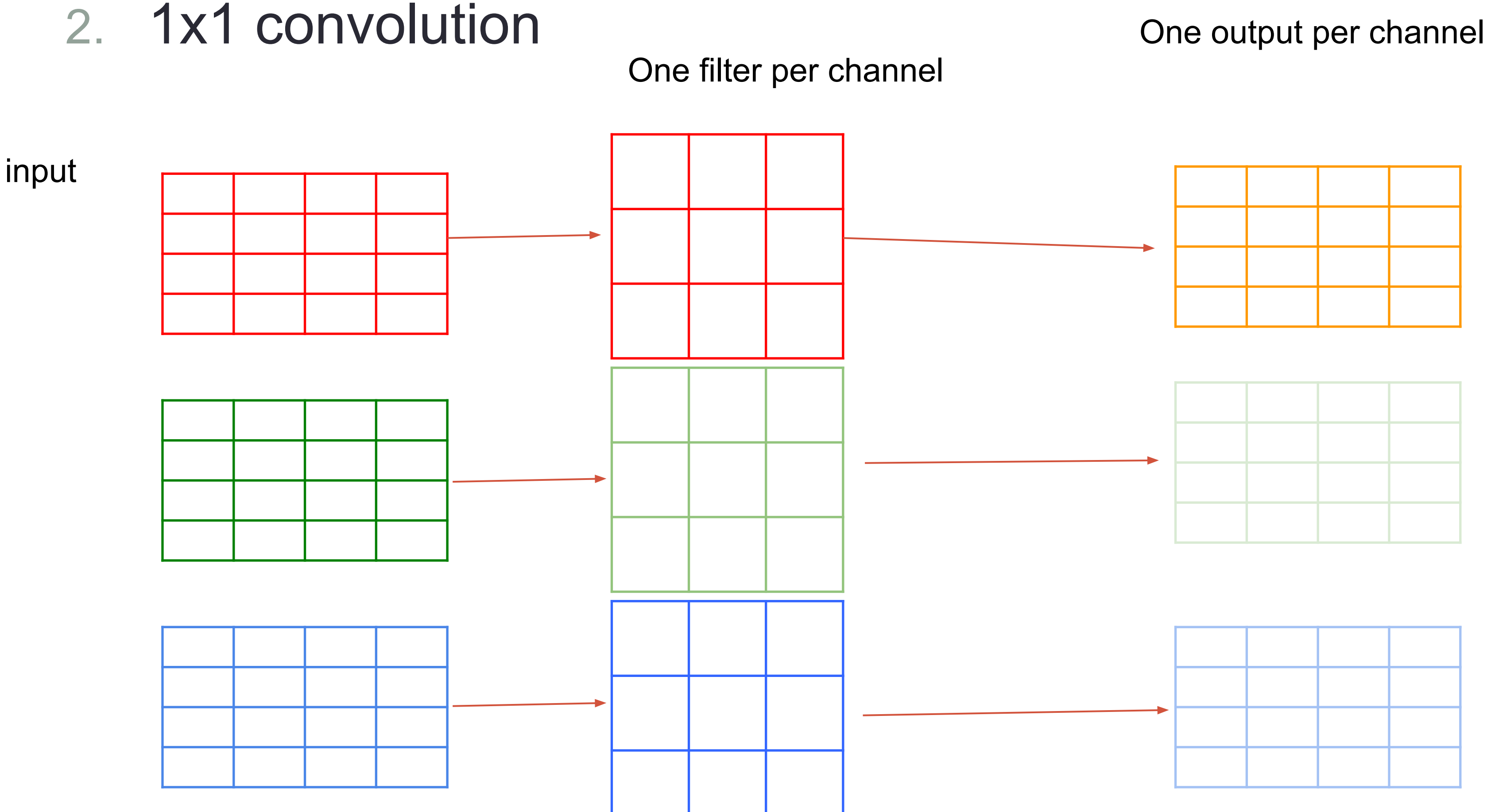
Depthwise convolution 3x3 filter is 3x3x1
1 filter per input channel



Xception

Depthwise separable convolution: two-step convolution

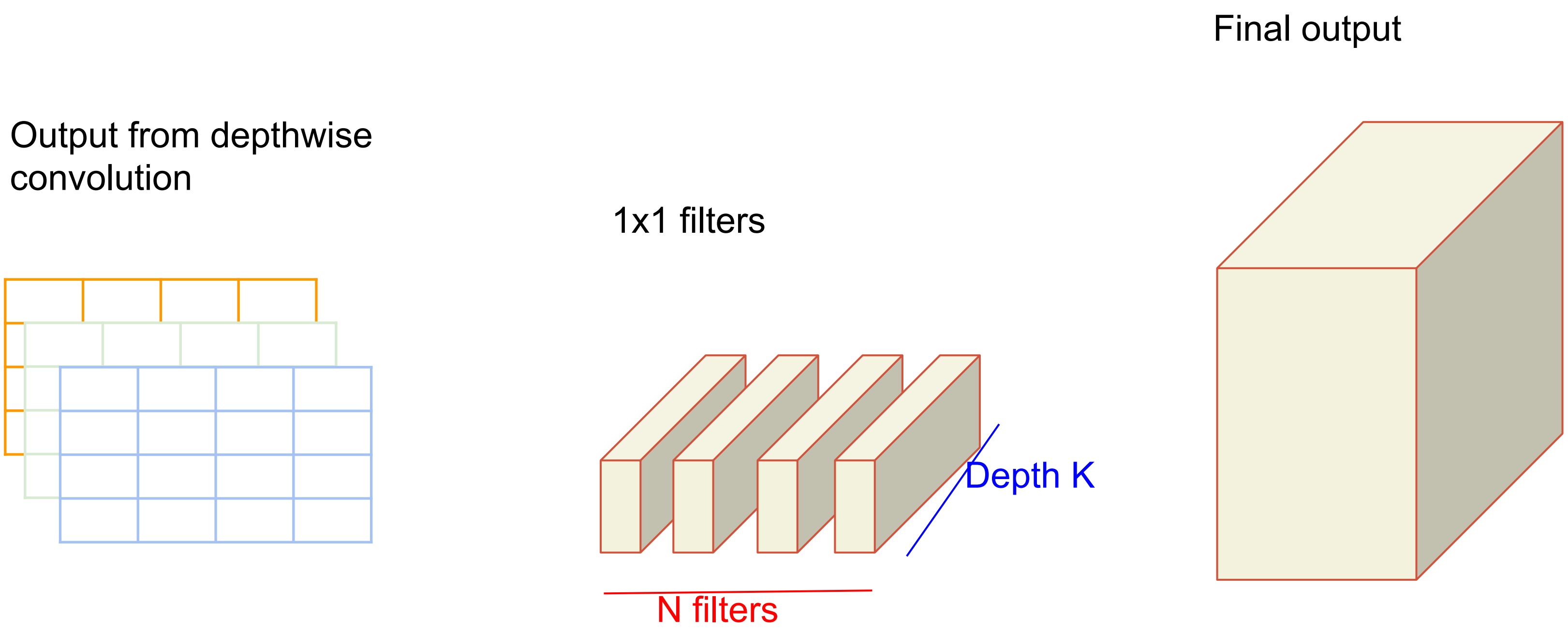
1. Depthwise convolution
2. 1x1 convolution



Xception

Depthwise separable convolution: two-step convolution

1. Depthwise convolution
2. 1x1 convolution



Xception

Replace convolutions in inception with depthwise separable convolutions

Smaller model

Faster compute

Comparable with Inception v3 while much faster

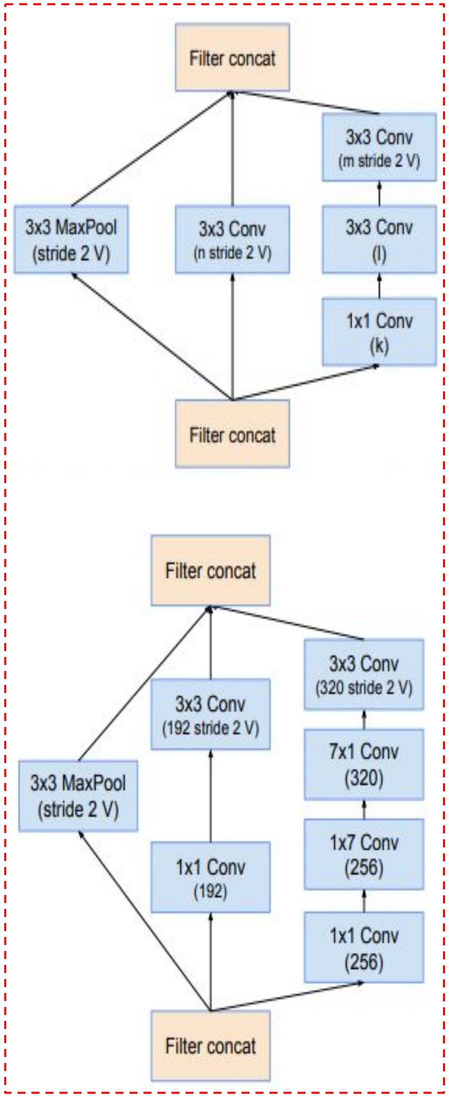
Used in MobileNet and other models

François Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions” 2016.
Andrew G. Howard, et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications” 2017.

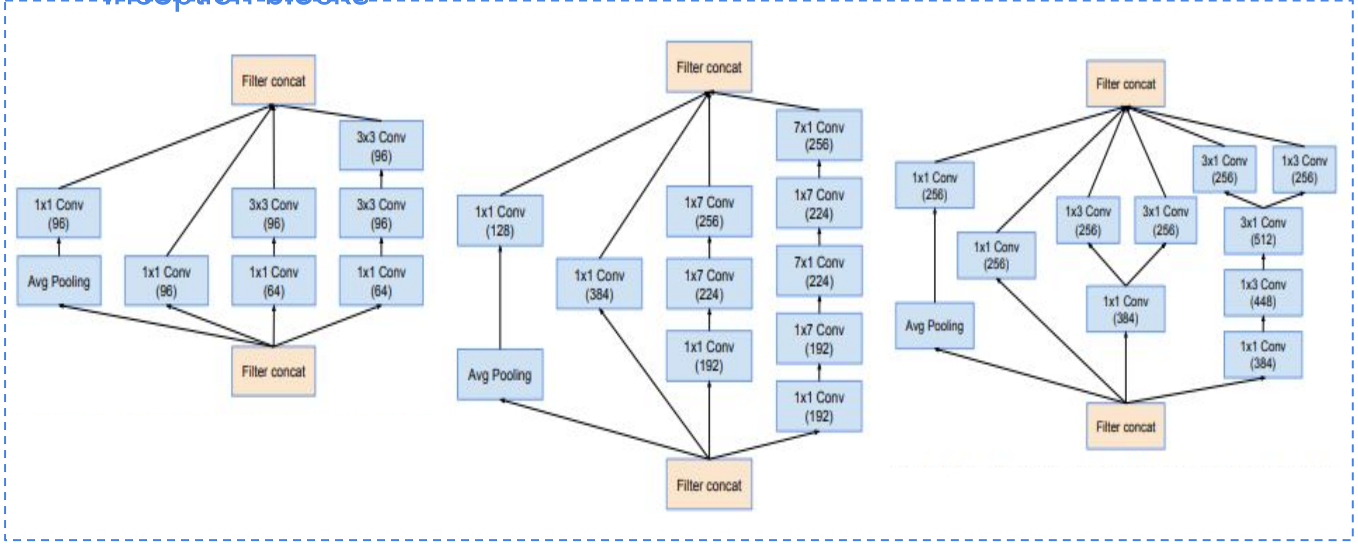
Inception v4

Same three types of inception blocks from v3

Add two types of reduction blocks for reducing the size of the grid (super pooling blocks)



Inception blocks



Reduction blocks

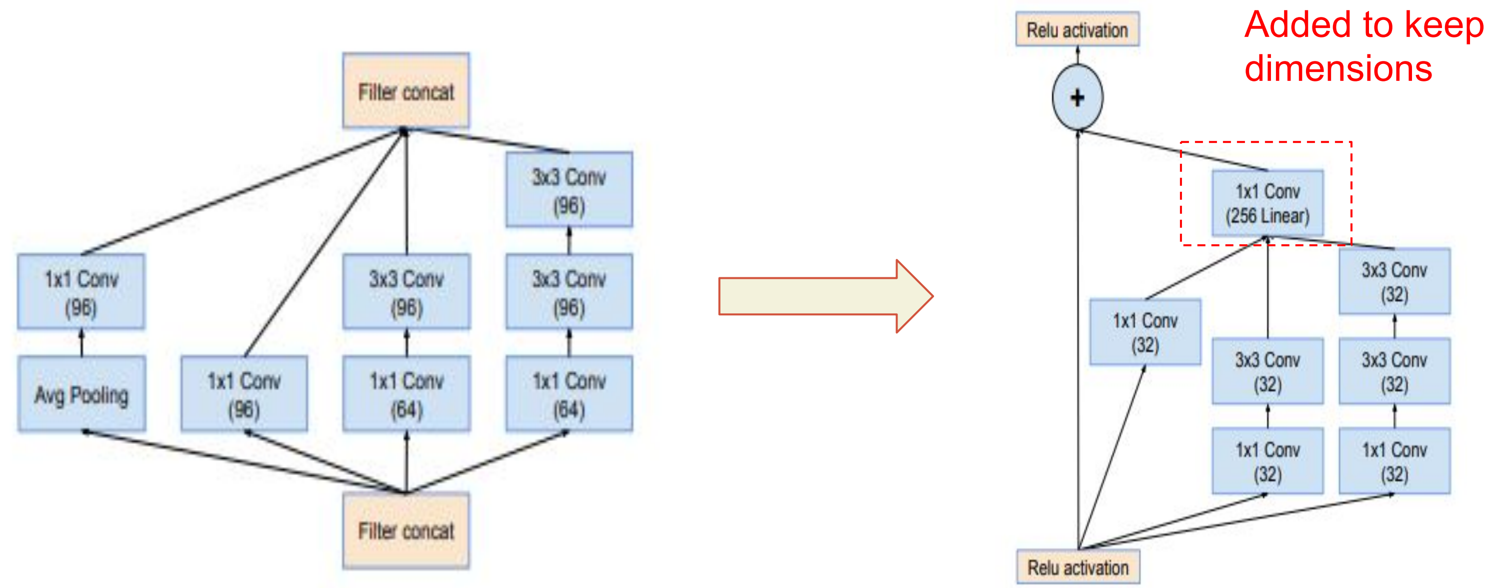
Christian Szegedy, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” 2016.

Inception-ResNet v1-v2

Introduce residual connections into inception blocks

Poolings changed to additions, 1x1 added to keep dimensions for the residual

Similar idea to ResNeXt



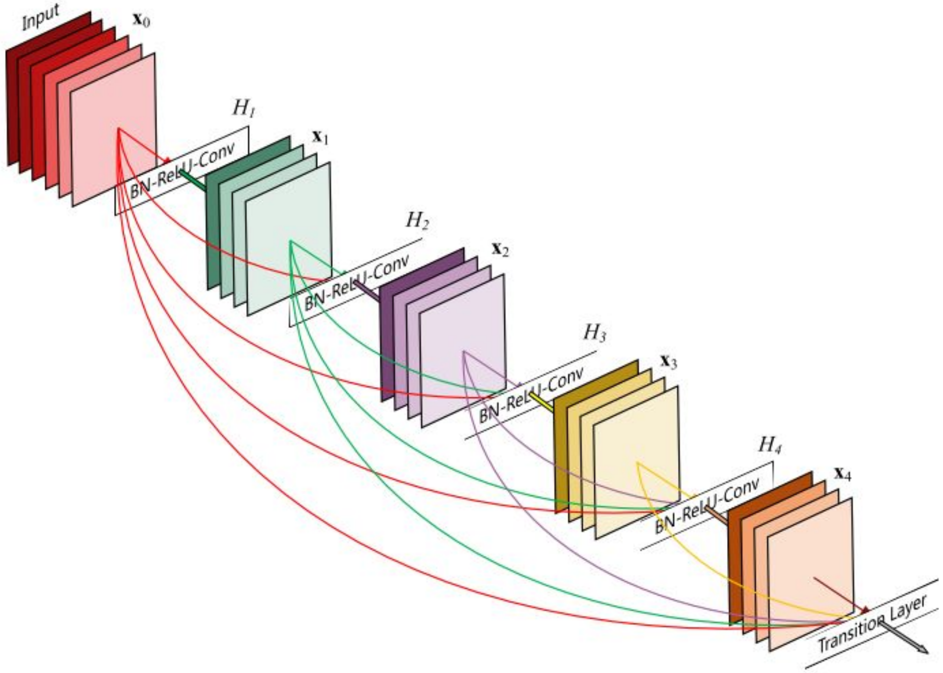
Christian Szegedy, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning" 2016.

DenseNet

Instead of adding the residuals, concatenates the feature maps from previous layers

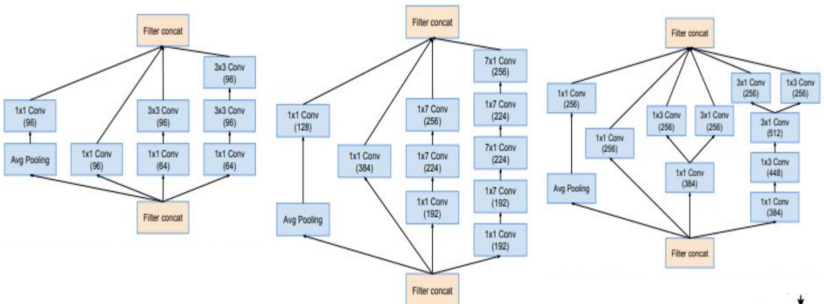
Densely connect multiple layers

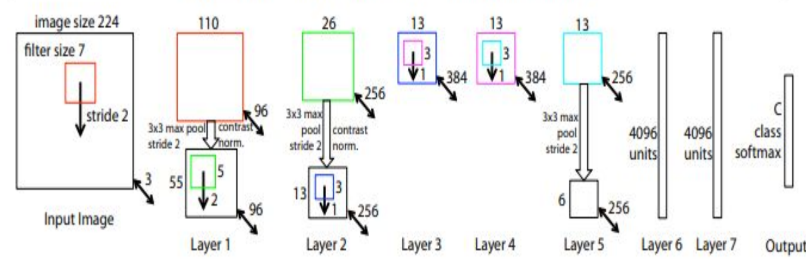
Multi-scale feature maps

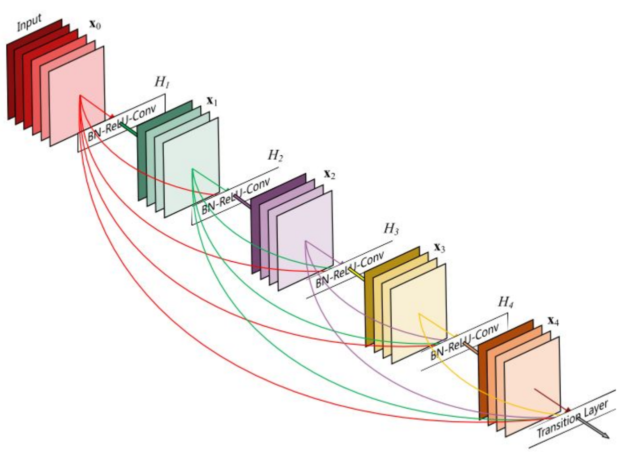


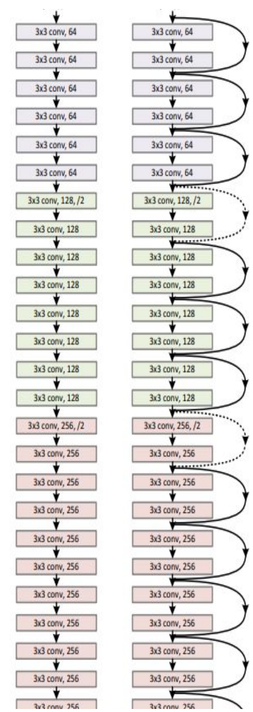
Densely Connected Convolutional Networks 2016

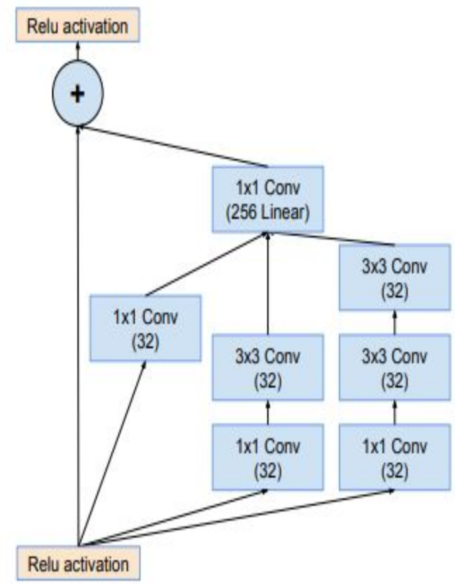
Do people keep tweaking network architectures until the end of time?









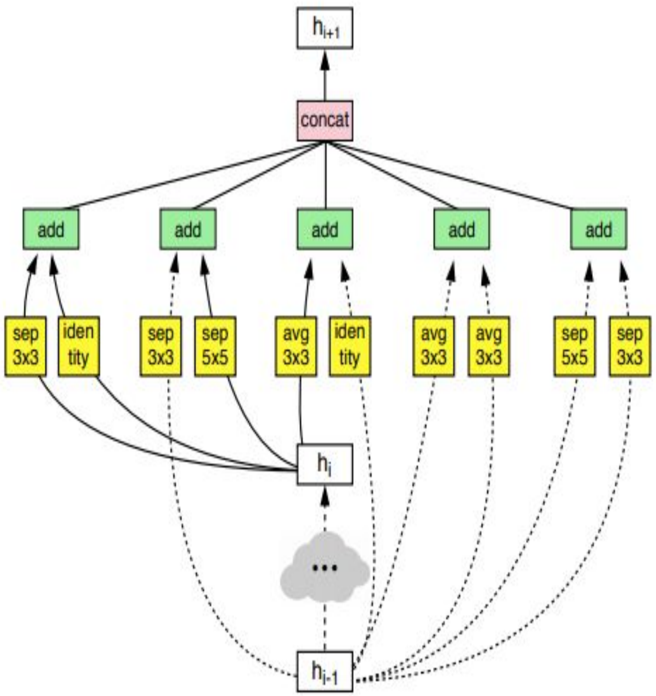


NasNet

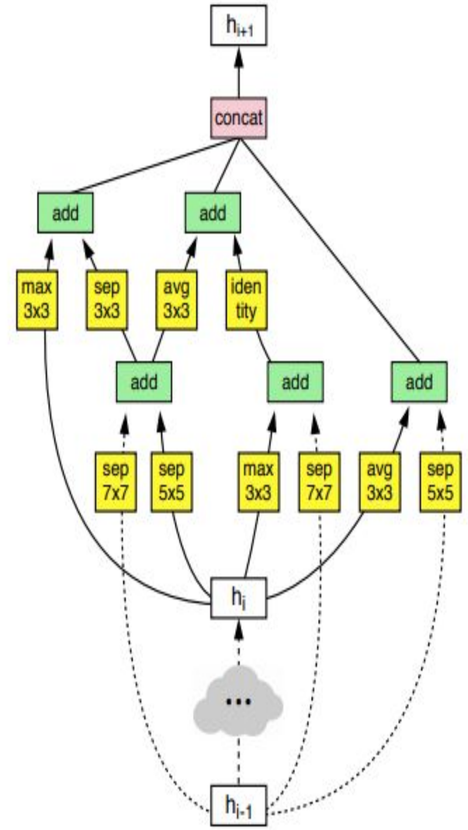
Neural Architecture Search Network

Use reinforcement learning to search for the best network configuration

Lowers compute while maintaining high accuracy



Inception block



Reduction block

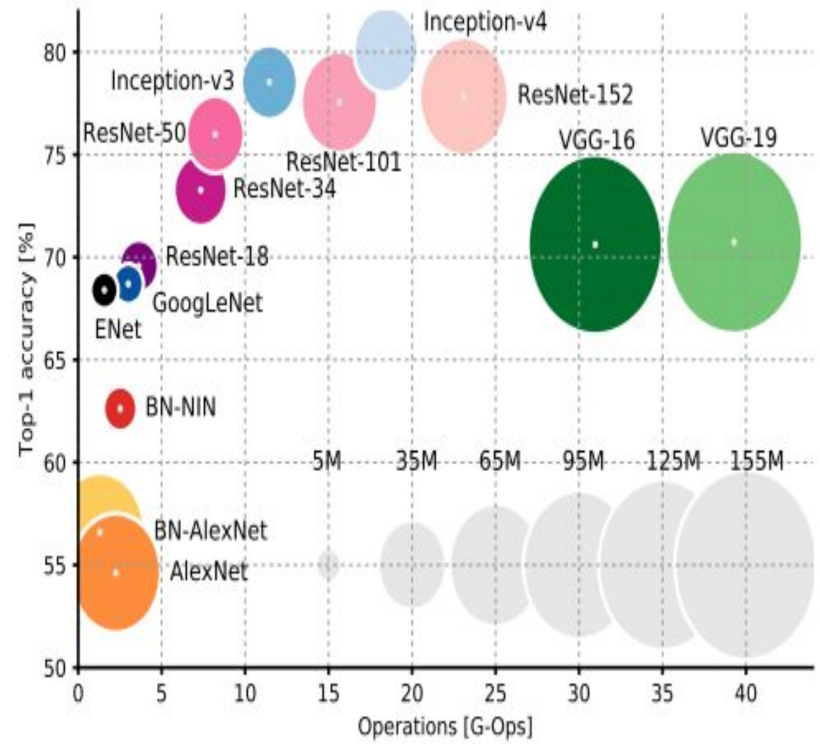
Barret Zoph, et al. “Learning Transferable Architectures for Scalable Image Recognition,” 2017.

Object classifiers summary

Most successful tricks:
Dropout, residual, batch norms, multi-path, data augmentation

These object classifiers are often called backbones and used by other models

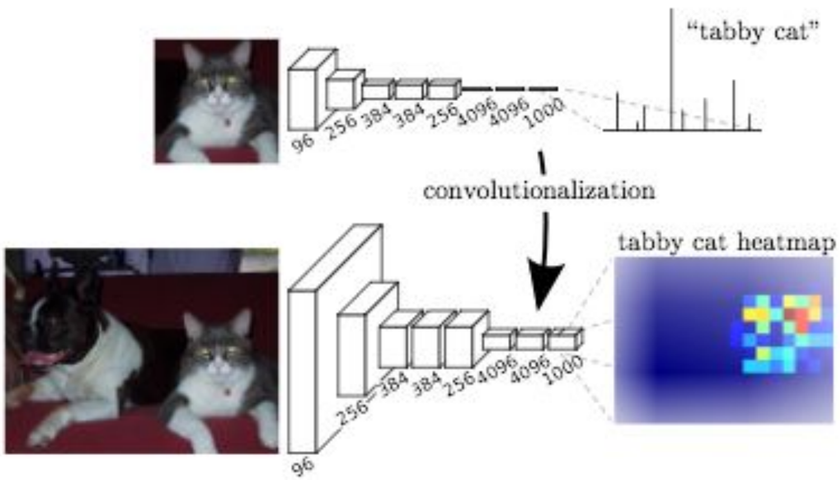
Comparing accuracy, model size, transferability and compute.

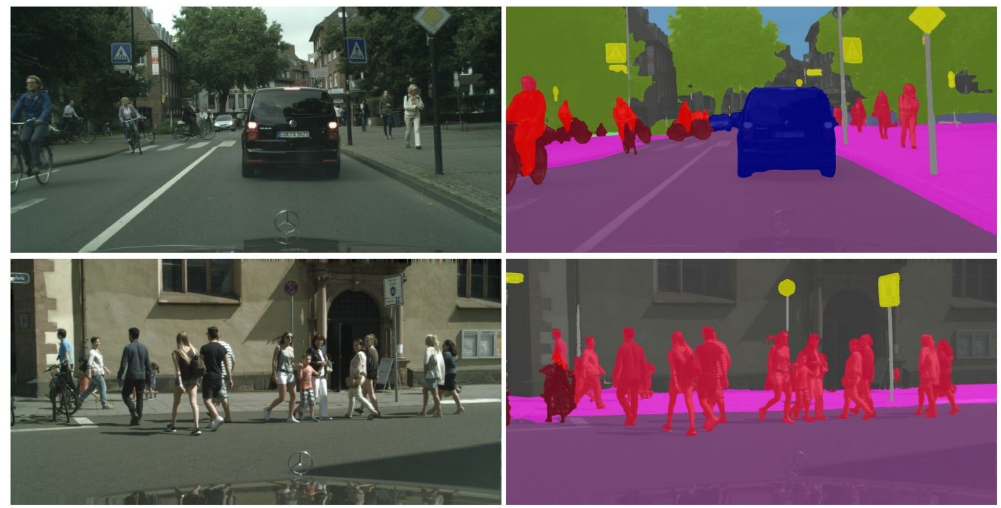


Alfredo Canziani, et al. "An analysis of deep neural network models for practical applications," 2017

Sometimes you want to increase the size of your feature map

Image segmentation



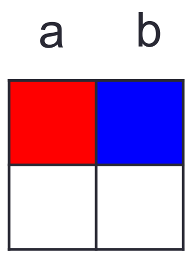


2 main approaches to upsample
De-convolution
resize (unpooling) +convolution

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf
http://vladlen.info/publications/feature-space-optimization-for-semantic-video-segmentation/

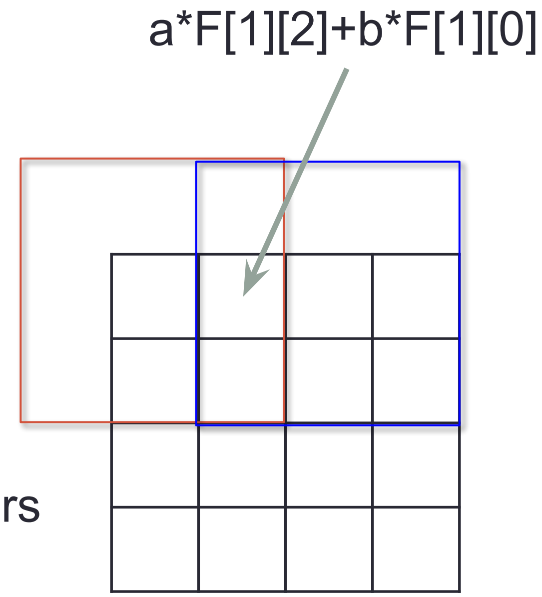
De-convolution (Upsampling)

- 3x3 de-convolution filter, stride 2, pad 1



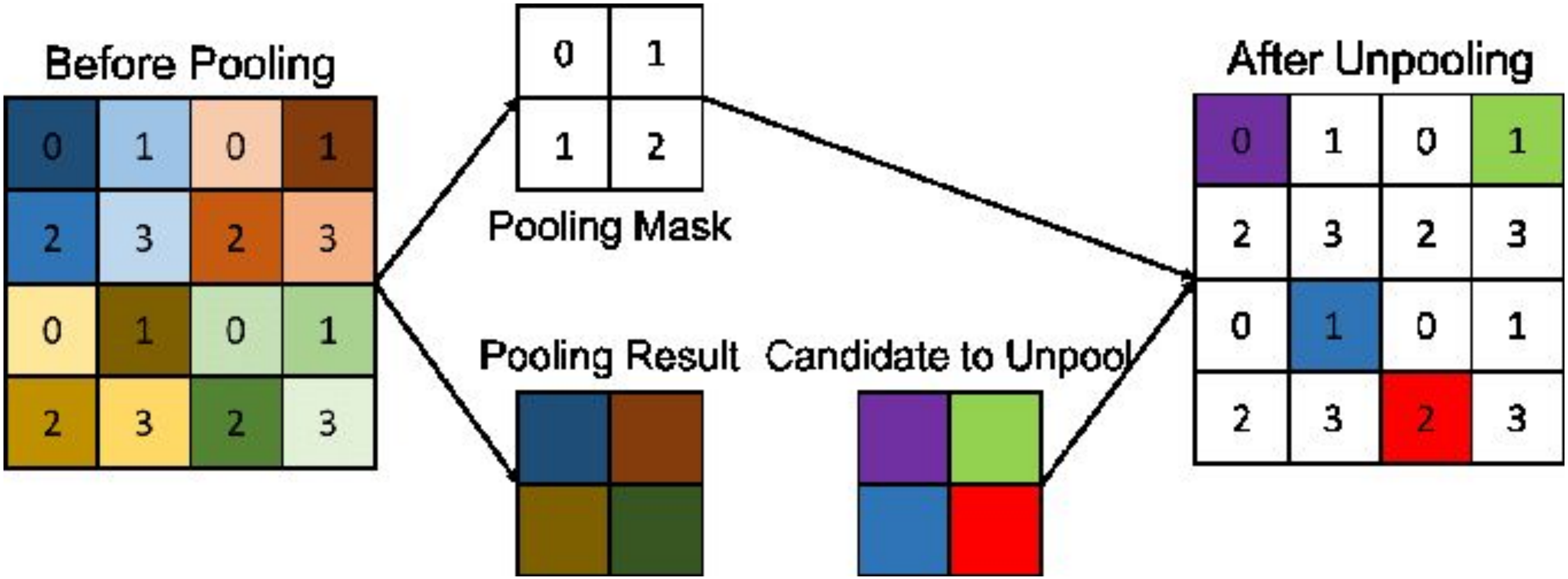
Input gives
Weight for filter

Rubber stamp of filters



Other names because this name sucks (for me)
- Convolution transpose, upconvolution, backward strided convolution

Unpooling + conv



Remember the location of the max value

Put the value to unpool at that location

Fill the rest with zeroes

Follow by regular convolution to smooth the image

Fully Automatic Acute Ischemic Lesion Segmentation in DWI Using Convolutional Neural Networks, 2017.

Upsampling notes

Deconvolution filter size should be a multiple of the stride to avoid checkerboard artifacts

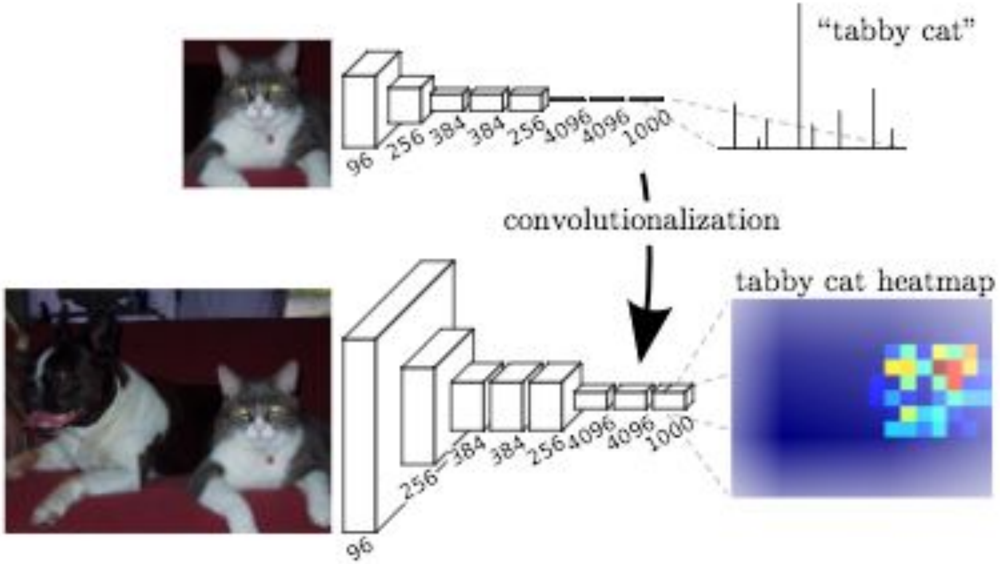
Unpooling can be replaced with regular image resizing techniques (interpolation)

Image using deconv

Image using resize upsampling

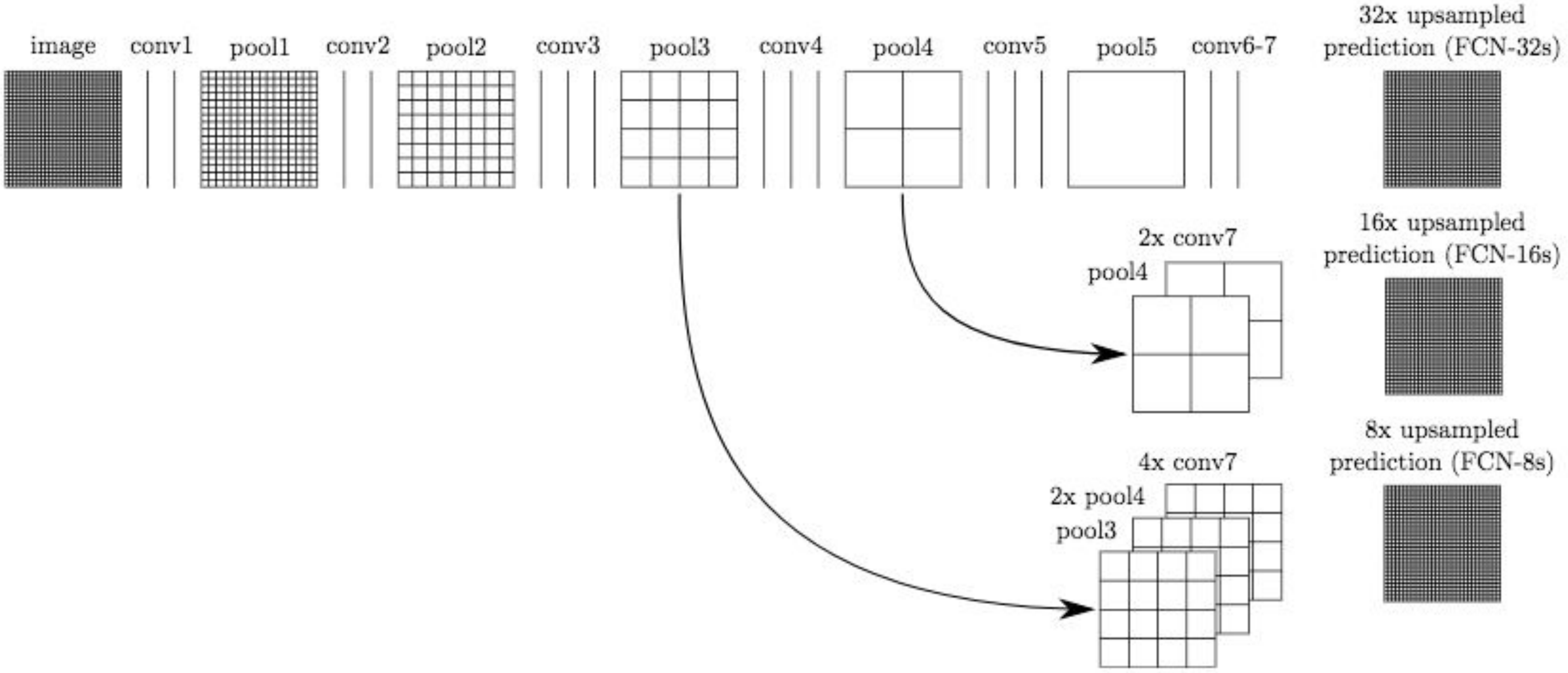
https://distill.pub/2016/deconv-checkerboard/

De-convolution for segmentation



https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

De-convolution for segmentation



https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

De-convolution for segmentation

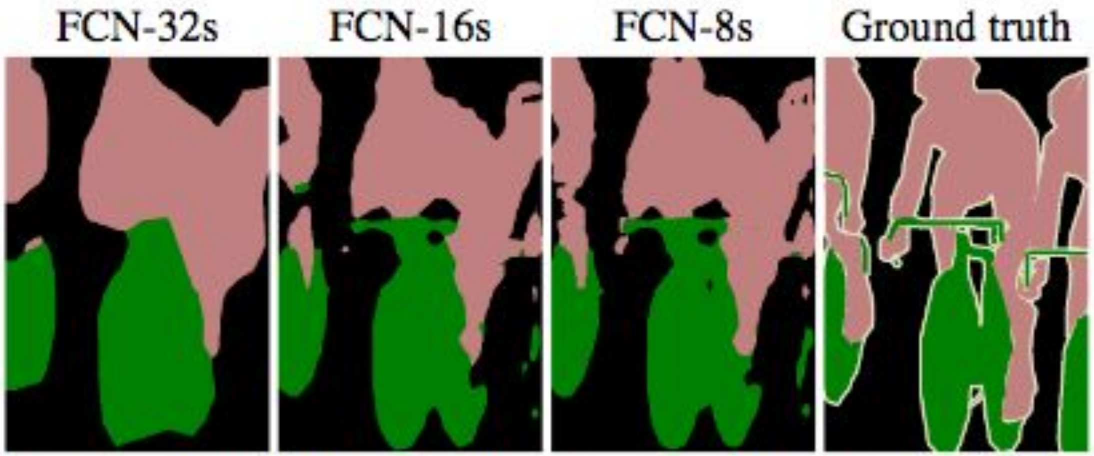


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Other networks to look for

Backbones

Feature Pyramid Networks (multi-scale backbone) https://arxiv.org/abs/1612.03144

SqueezeNet (small network) https://arxiv.org/abs/1602.07360

DARTS (improved NASNet) https://arxiv.org/abs/1806.09055

MNASNet (mobile version of NASNET) https://arxiv.org/abs/1807.11626

CNN Summary

Building blocks

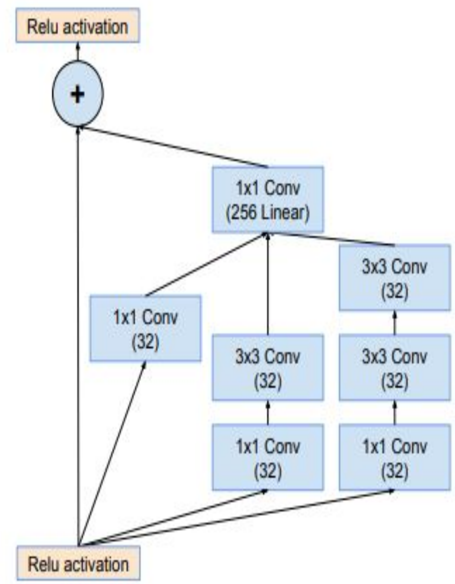
Matched filters and pooling

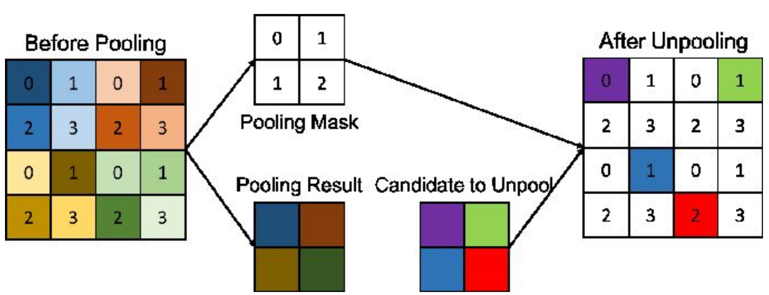
Filter factorization

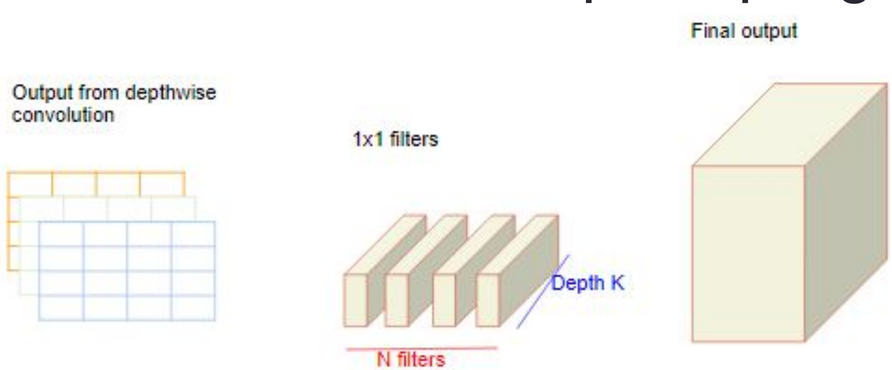
- 5x5 -> two 3x3
- depthwise vs matrix factorization

1x1 convolution

Deconvolution and upsampling

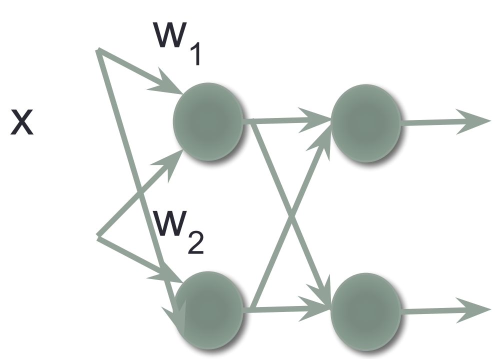




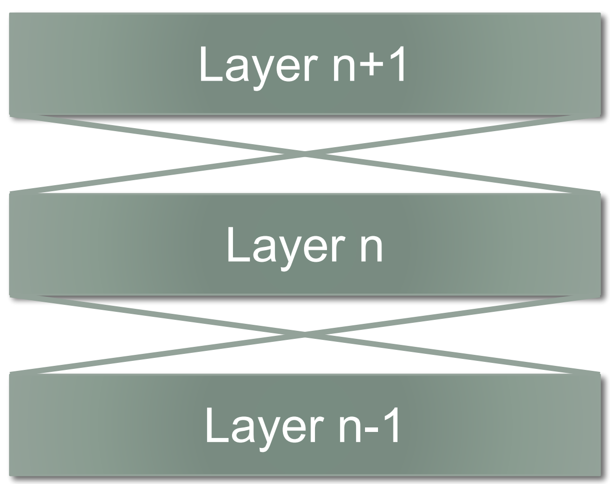


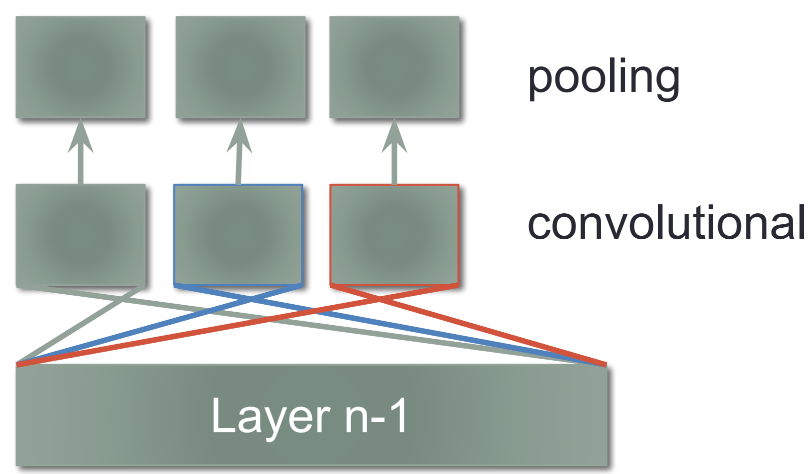
Parameter sharing in convolution neural networks

- $W^T x$



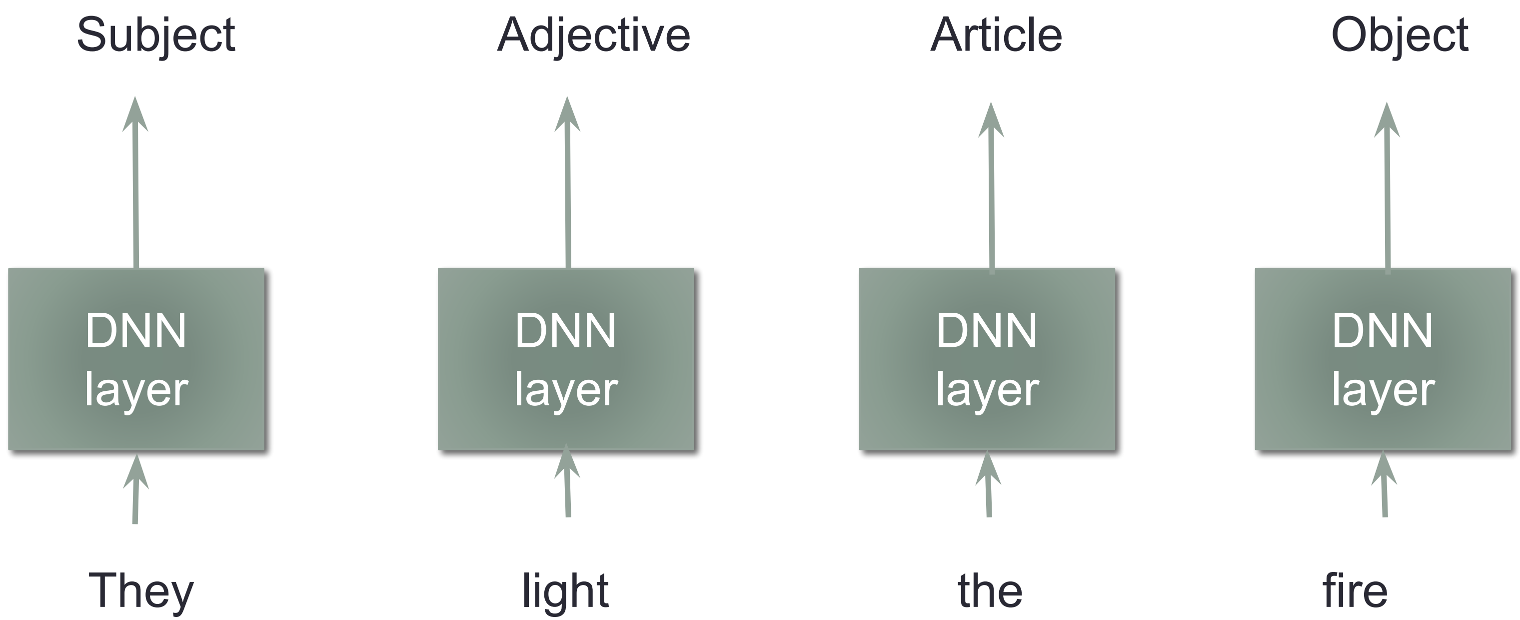
- CNN shares parameters in space
- What about sharing parameters in time?





Recurrent neural network (RNN)

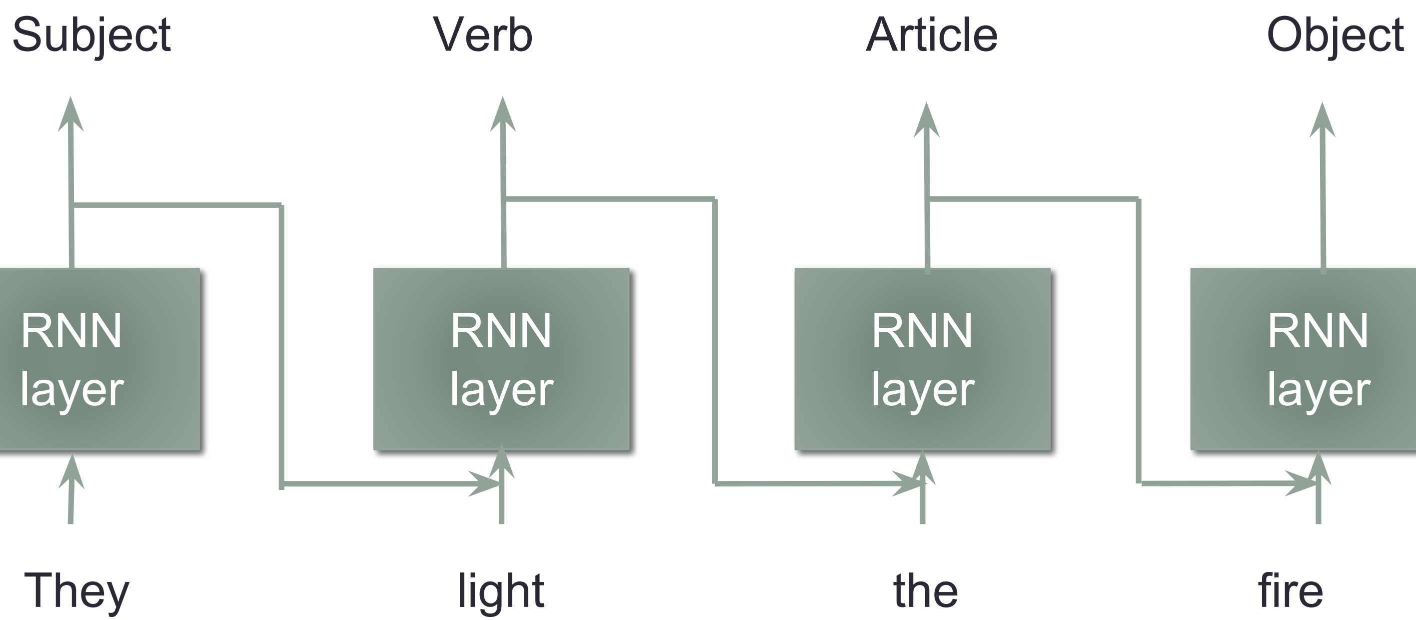
- DNN framework



Problem 1: need a way to remember the past

Recurrent neural network (RNN)

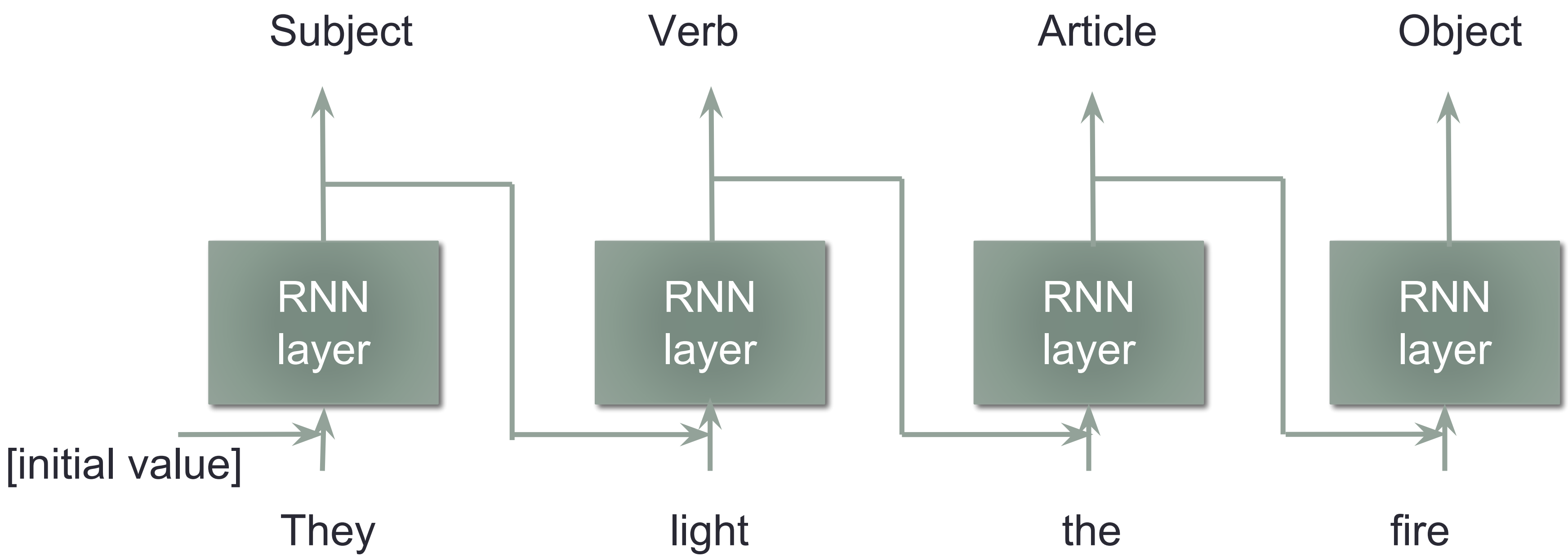
- RNN framework



Output of the layer encodes something meaningful about the past

Recurrent neural network (RNN)

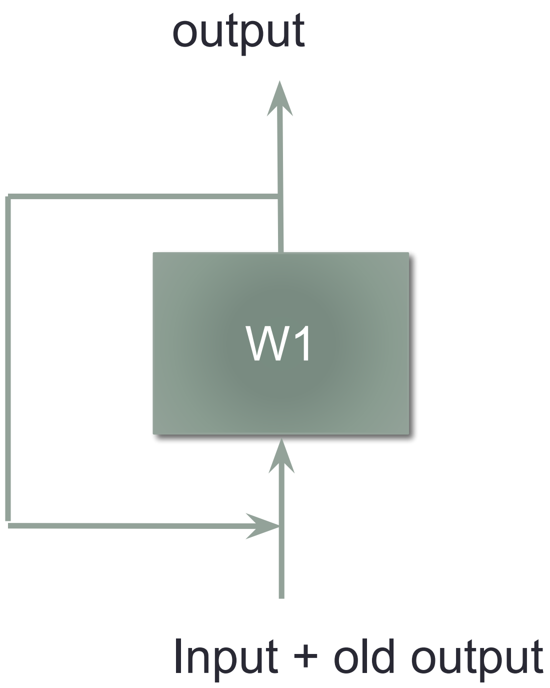
- RNN framework



New input feature = [original input feature, output of the layer at previous time step]

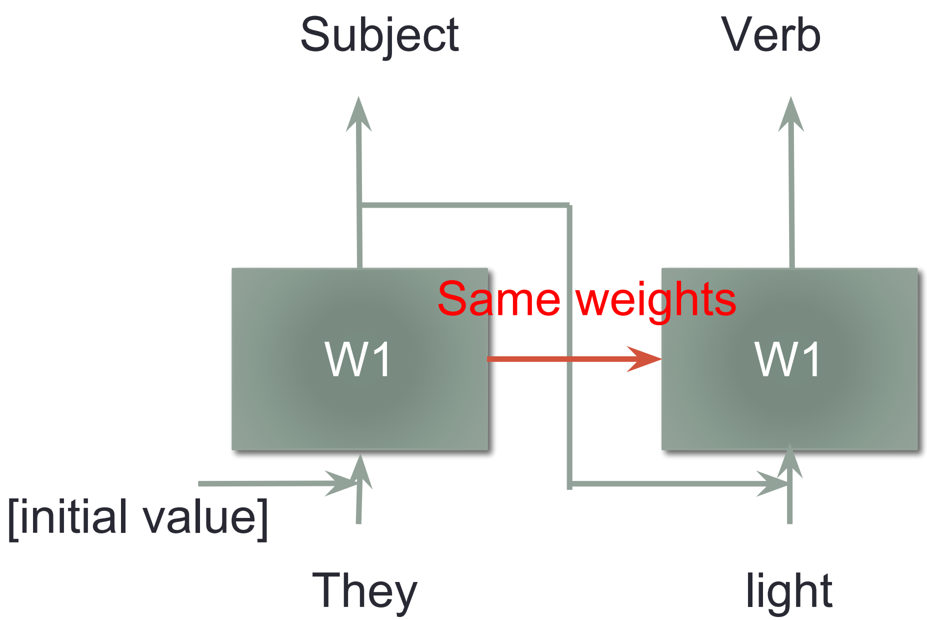
Recurrent neural network (RNN)

- Unrolling of a recurrent layer.



Recurrent neural network (RNN)

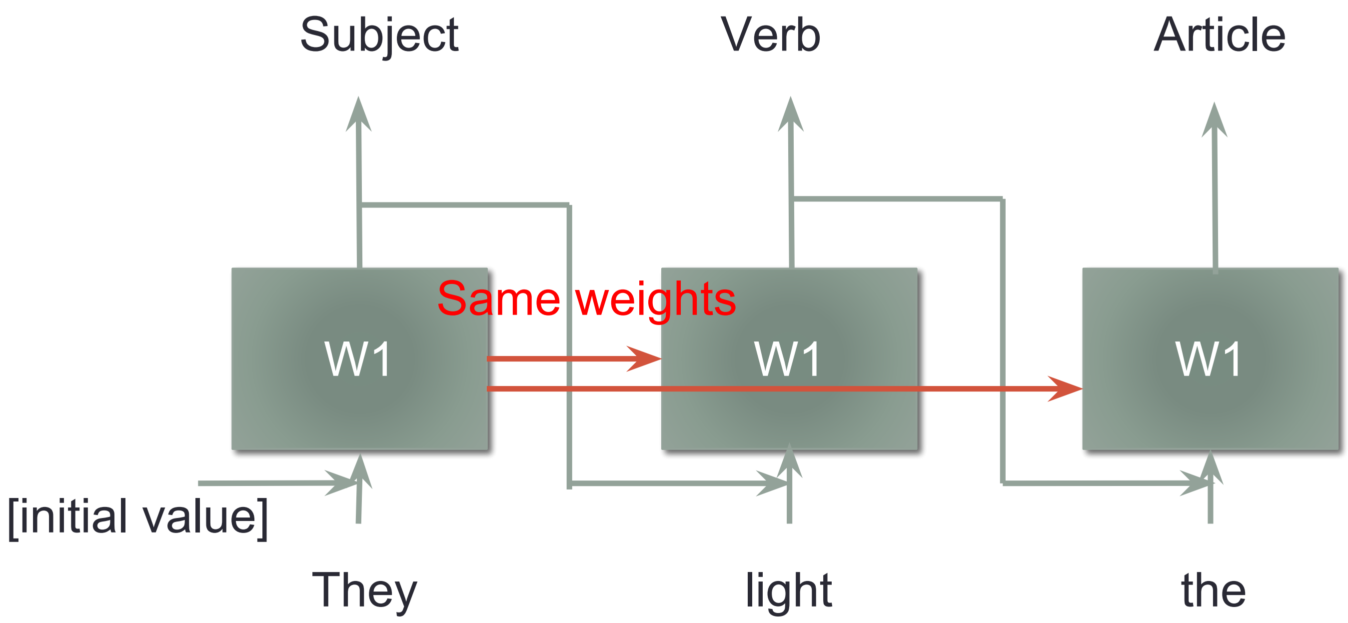
- Unrolling of a recurrent layer.



Parameter sharing across time

Recurrent neural network (RNN)

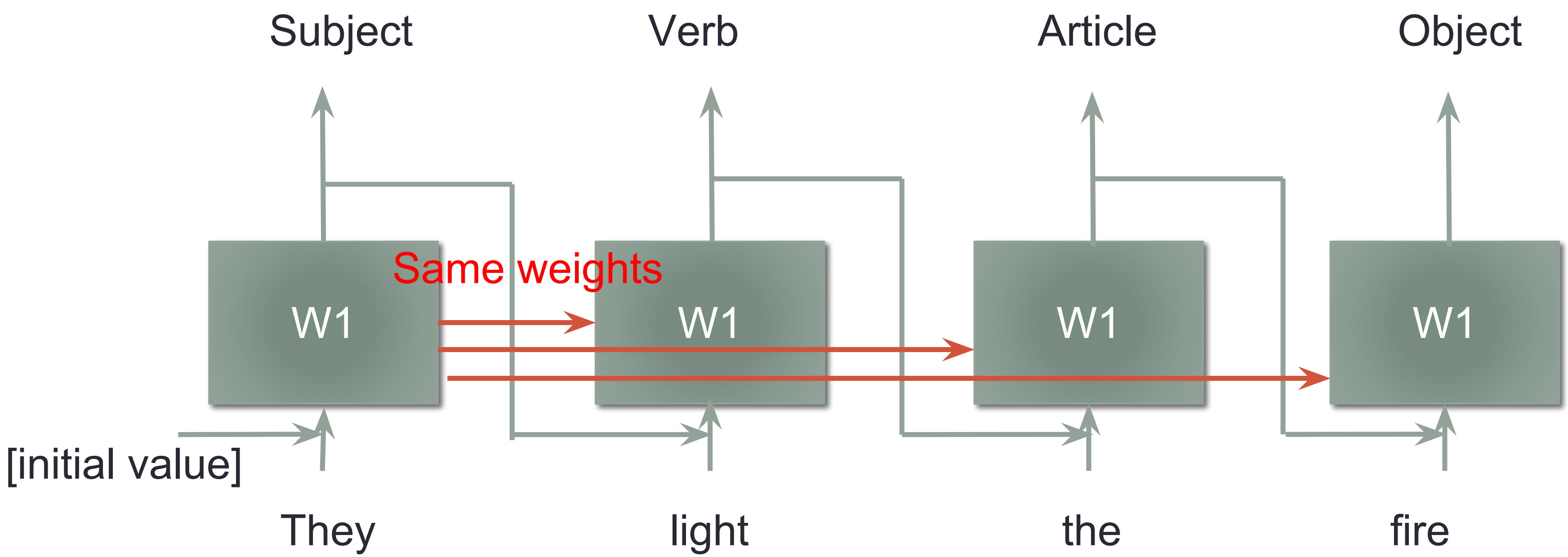
- Unrolling of a recurrent layer.



Parameter sharing across time

Recurrent neural network (RNN)

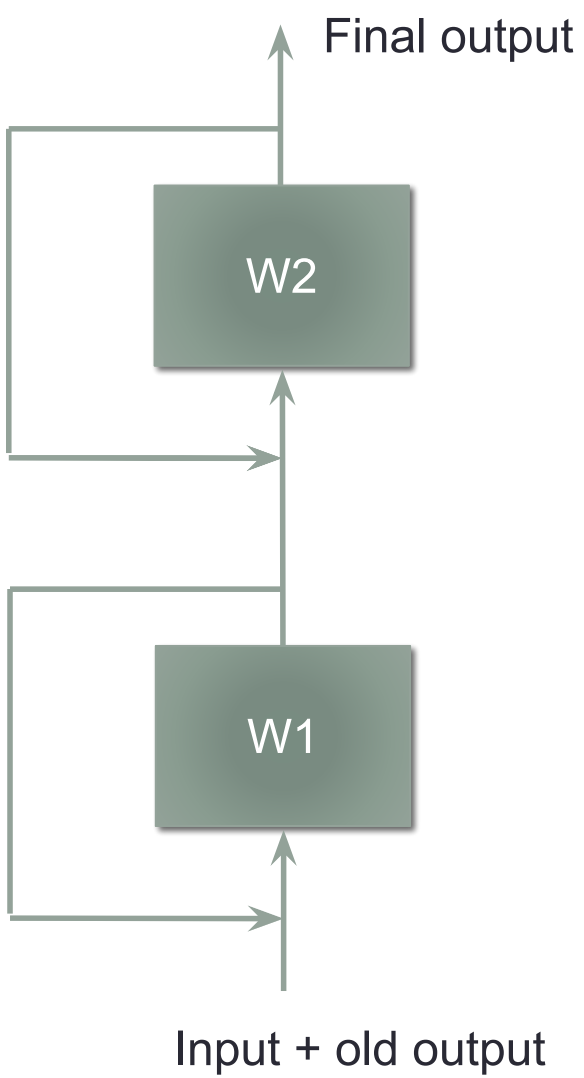
- Unrolling of a recurrent layer.



Parameter sharing across time

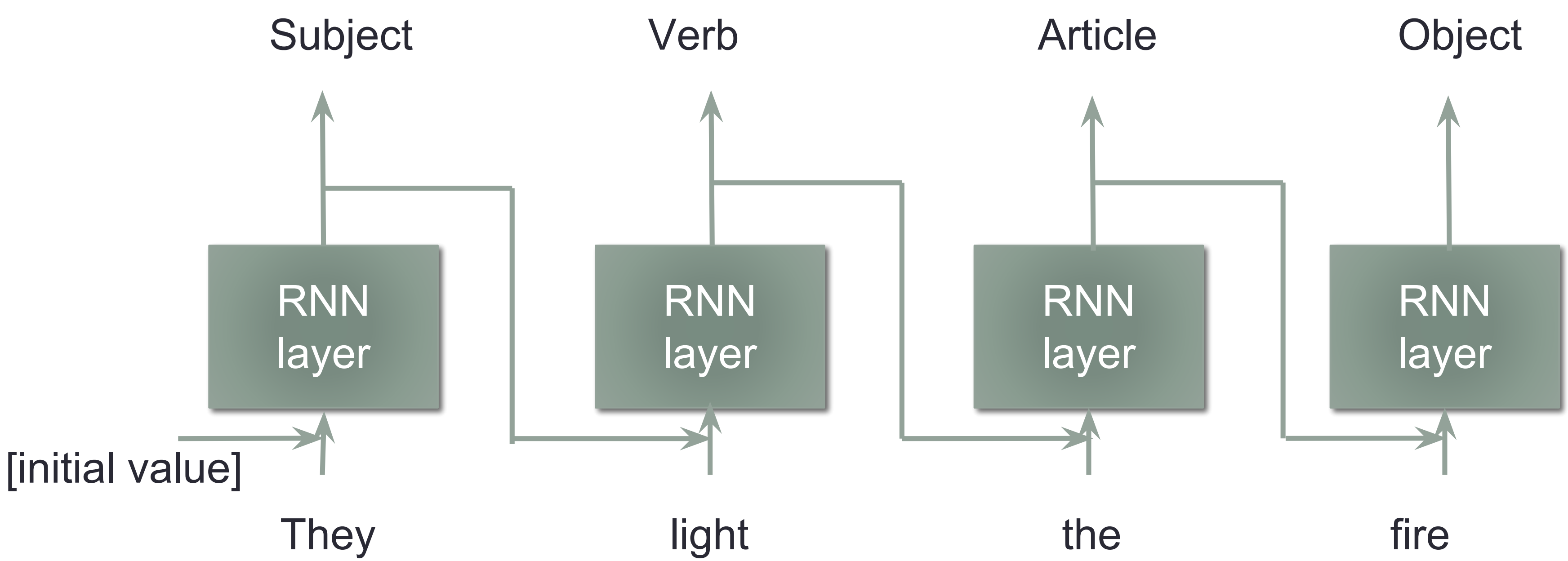
Recurrent neural network (RNN)

- Stacks of recurrent layer



Training a recurrent neural network

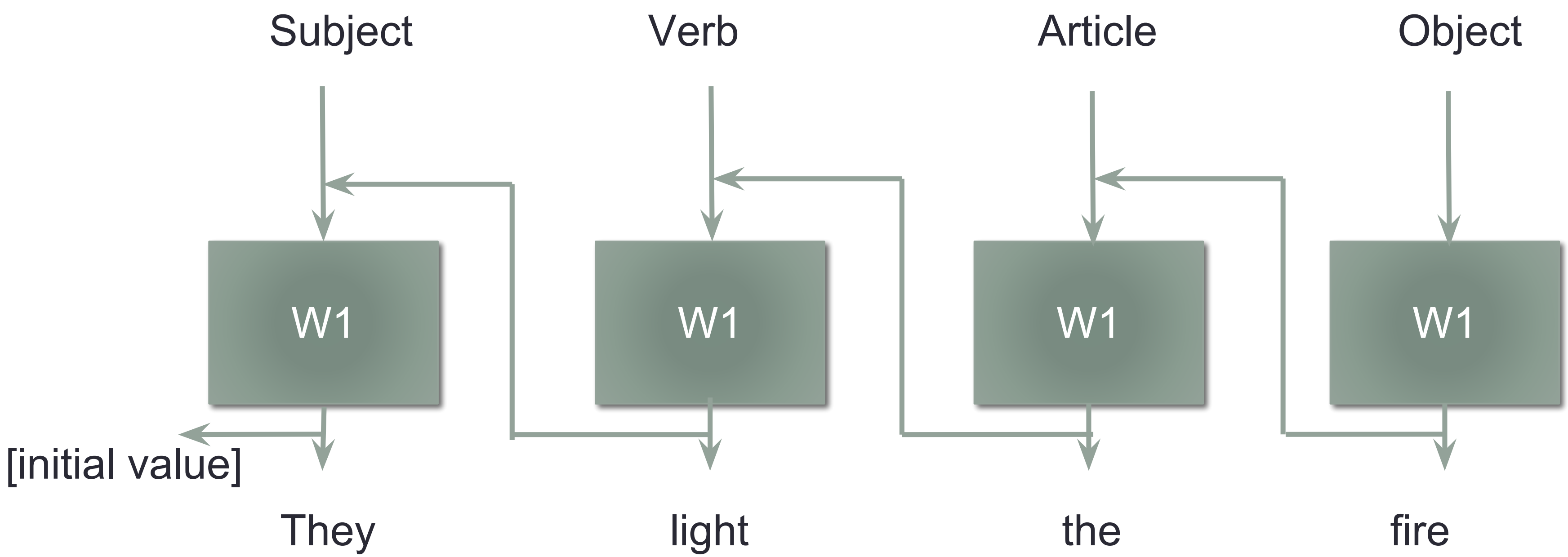
- RNN framework



New input feature = [original input feature, output of the layer at previous time step]

Training a recurrent neural network

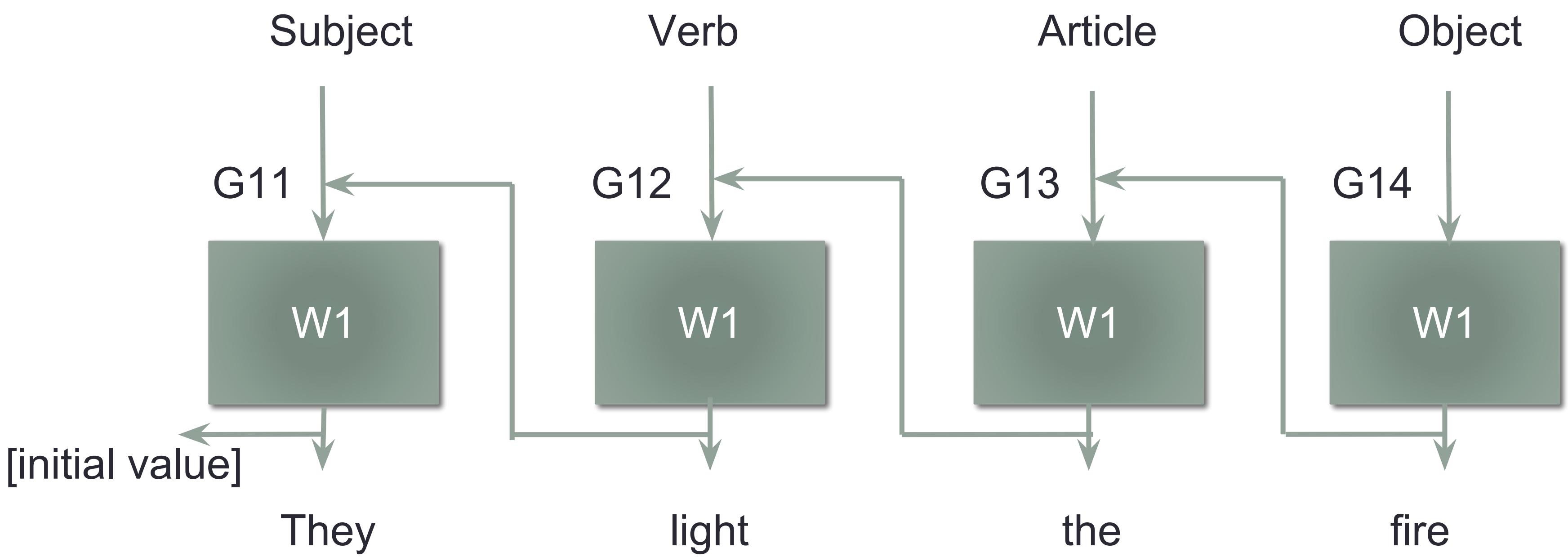
- Backward Computation graph



Backpropagation through time (BPTT)

BPTT

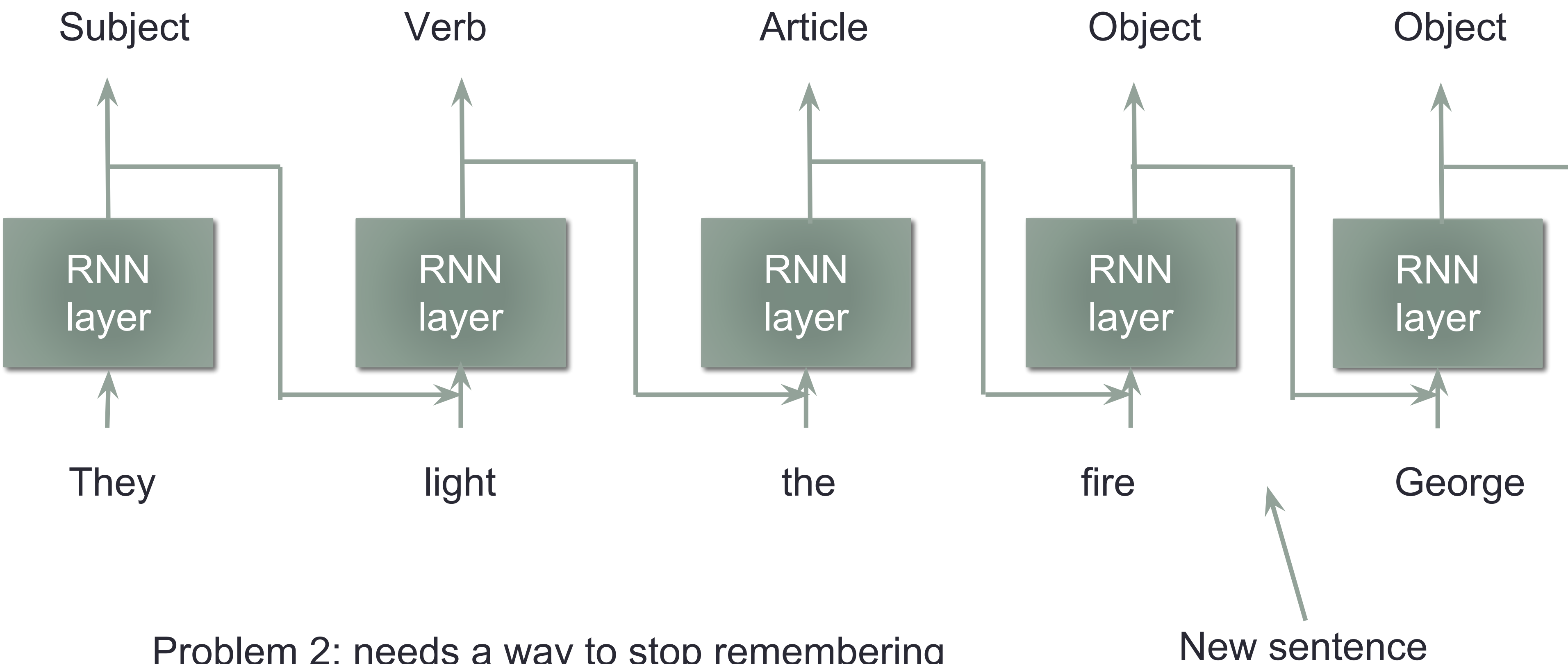
- Backward Computation graph



W1 <- W1 + G11 + G12 + G13 + G14

Problem 1: cannot deal with infinitely long recurrent
Gradient explosion, vanishing gradient

Recurrent neural network (RNN)

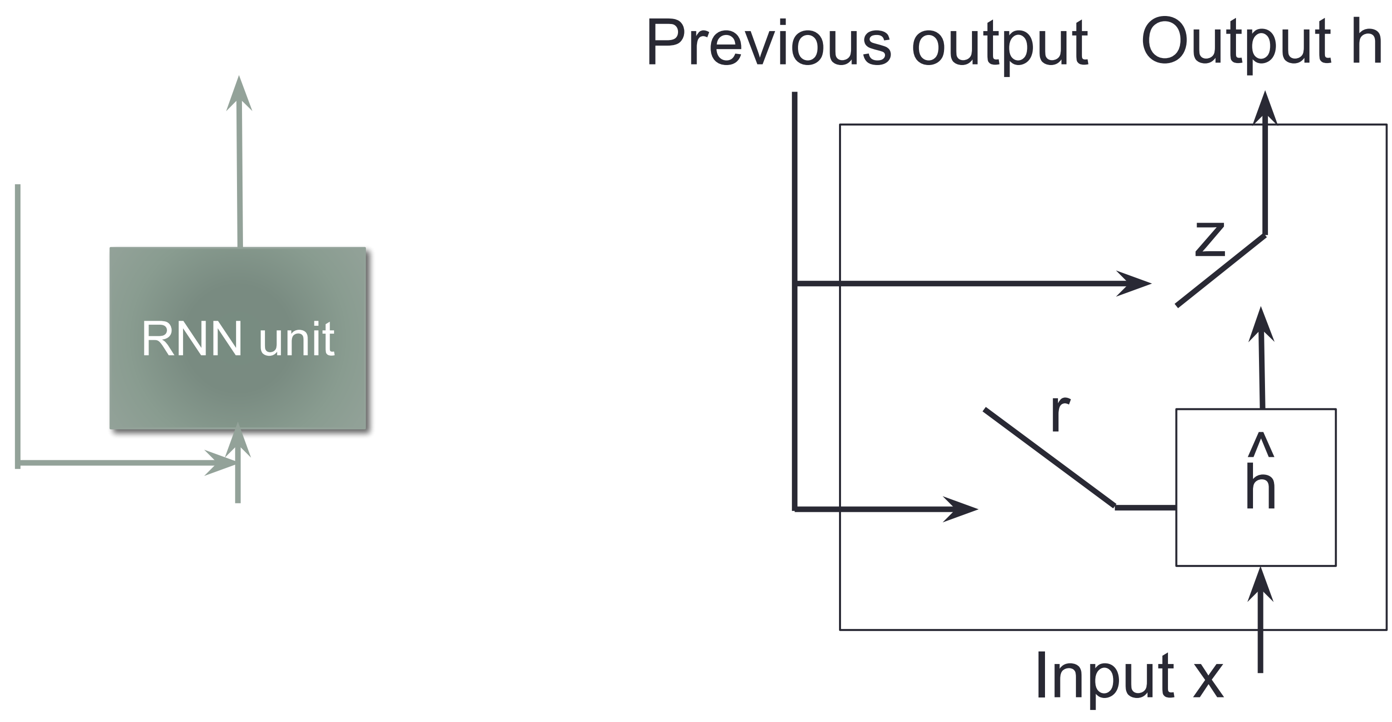


Problem 2: needs a way to stop remembering

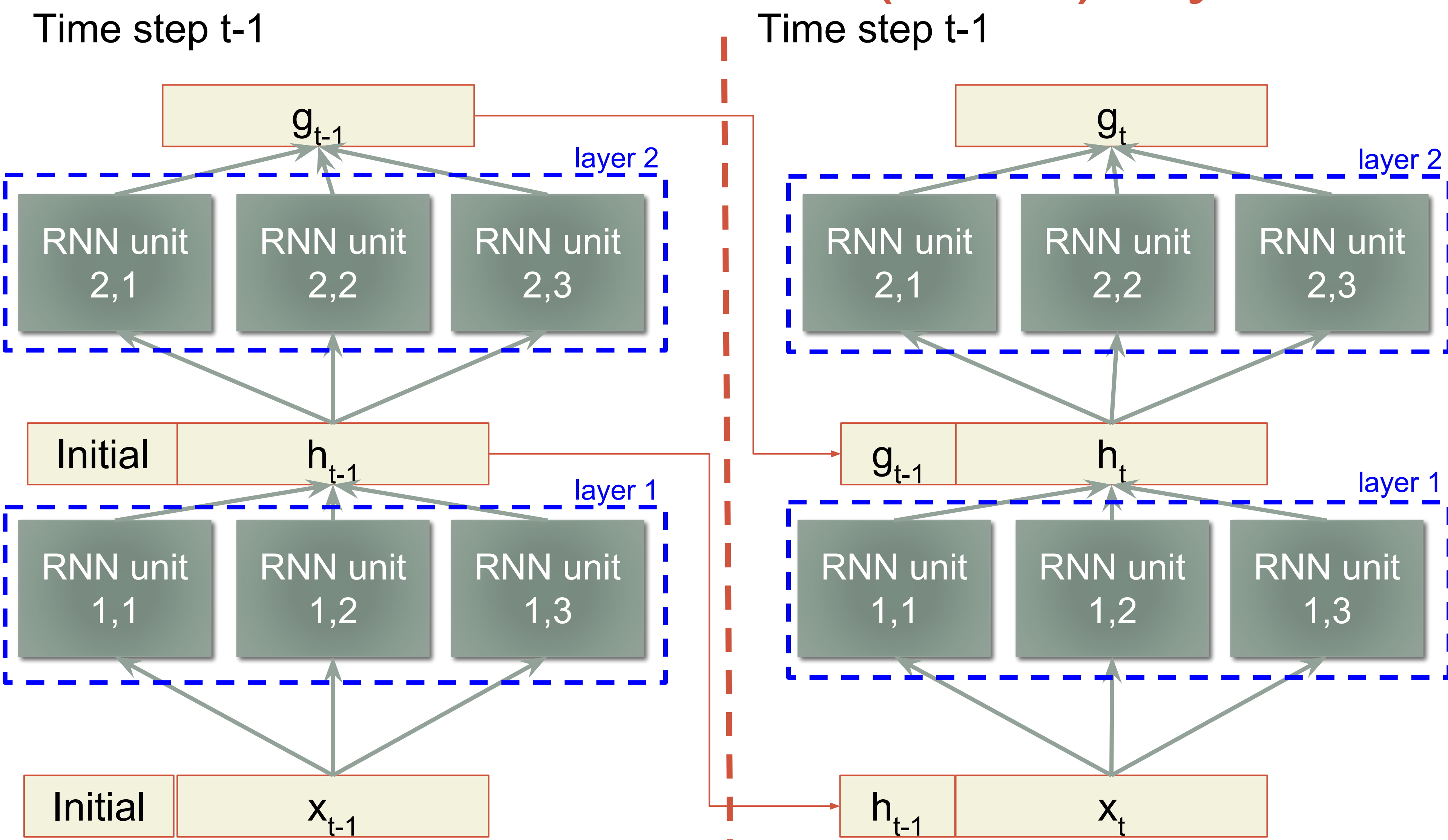
Can the network learn when to start and stop remembering things?

Gated Recurrent Unit (GRU)

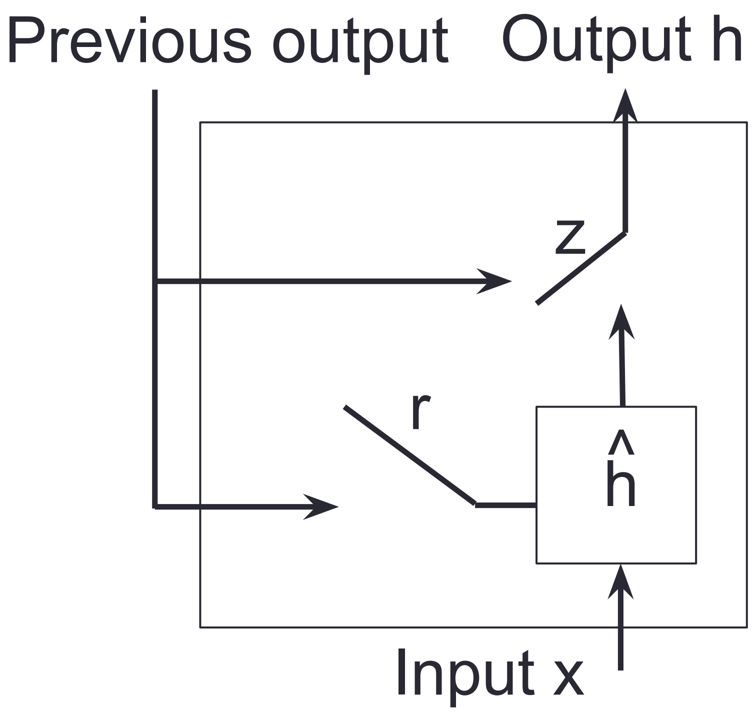
- Forms a Gated Recurrent Neural Networks (GRNN)
- Add gates that can choose to reset (r) or update (z)



Gated Recurrent Unit (GRU) layer



Gated Recurrent Unit (GRU)

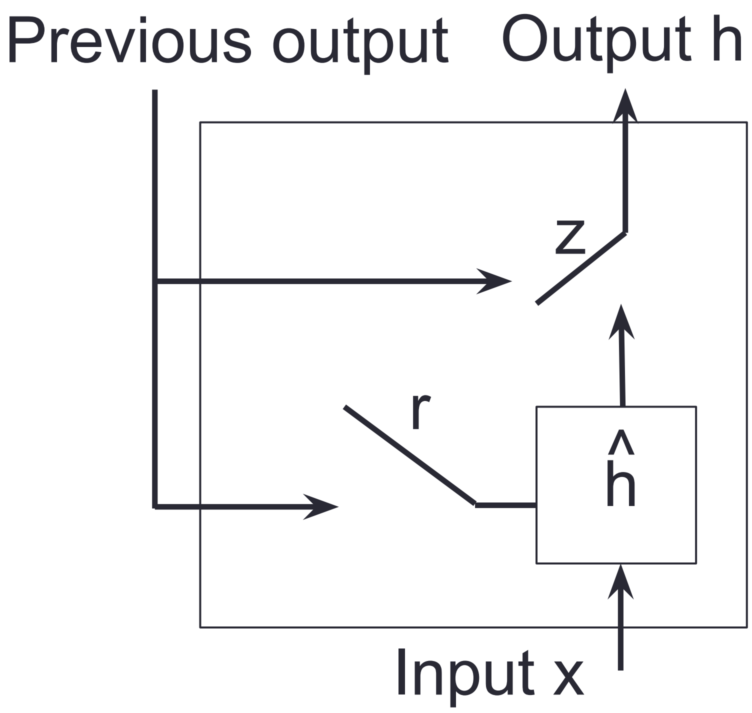


Neuron index

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

time index

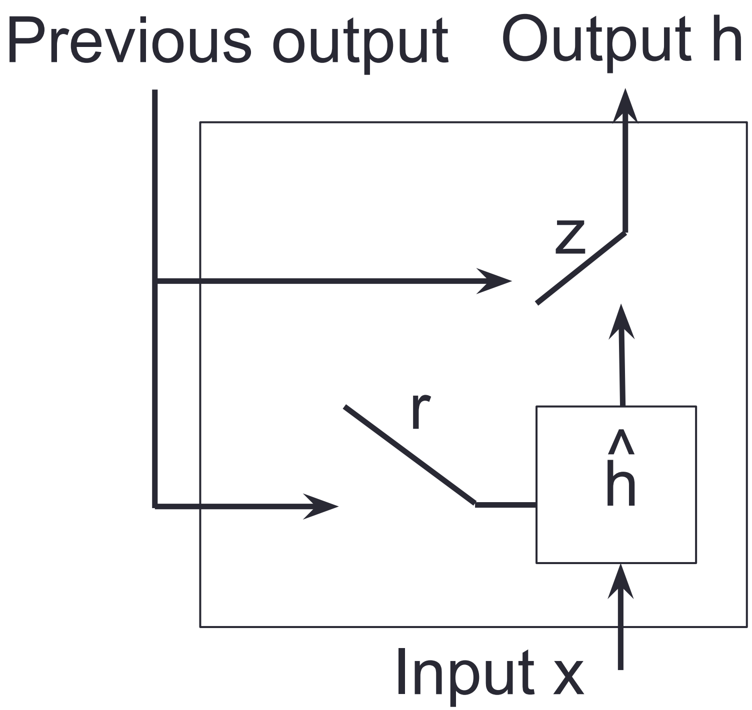
Gated Recurrent Unit (GRU)



$$\boxed{h_t^j} = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$

One GRU neuron output (scalar)

Gated Recurrent Unit (GRU)



$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(\boxed{W}\boxed{\mathbf{x}_t} + \boxed{U}(\underline{\mathbf{r}_t \odot \mathbf{h}_{t-1}}))$$

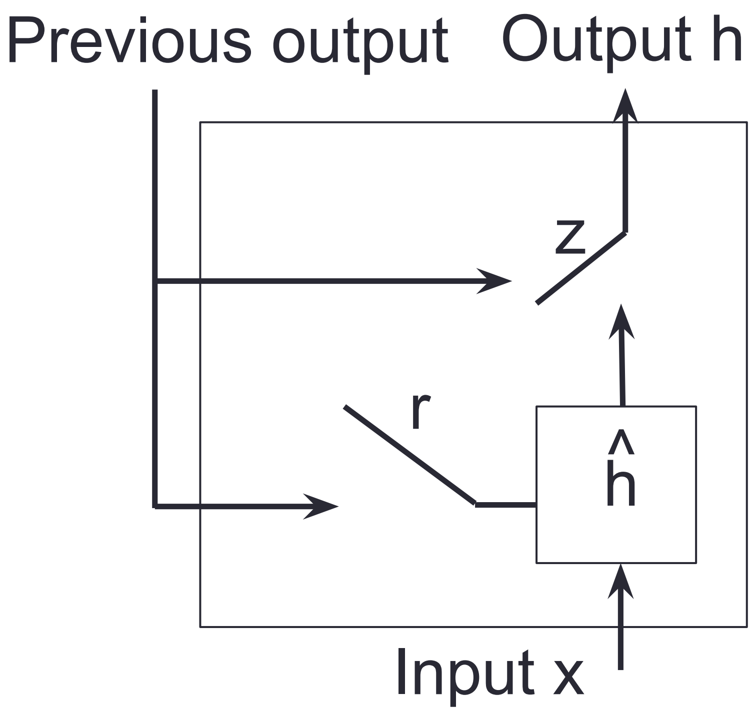
Element-wise product

Linear transform with matrix multiply

Vector (each value from each GRU unit in the previous layer)

$x_t^j = h_t^j$

Gated Recurrent Unit (GRU)



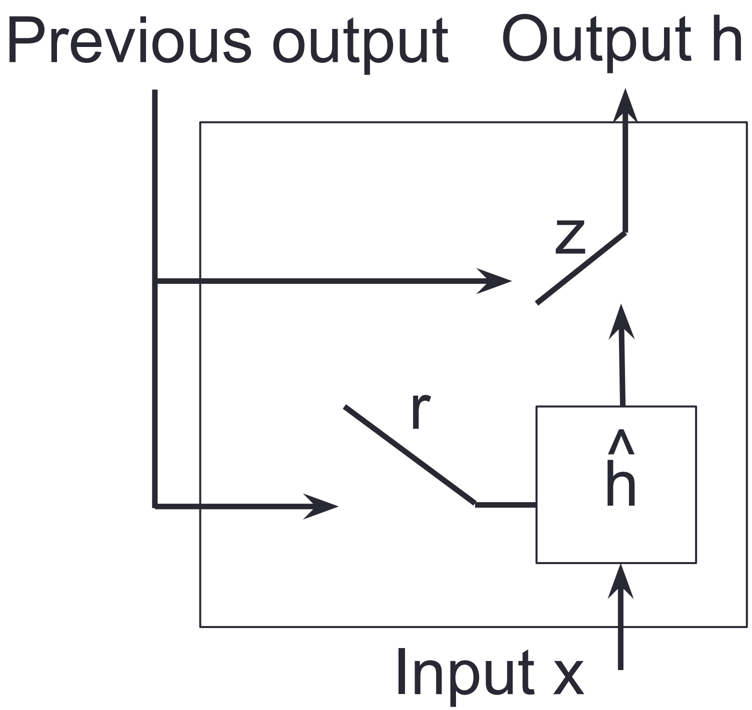
$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = \underline{tanh}\boxed{^j}(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

Takes the j-th element

Bounds the output

Gated Recurrent Unit (GRU)



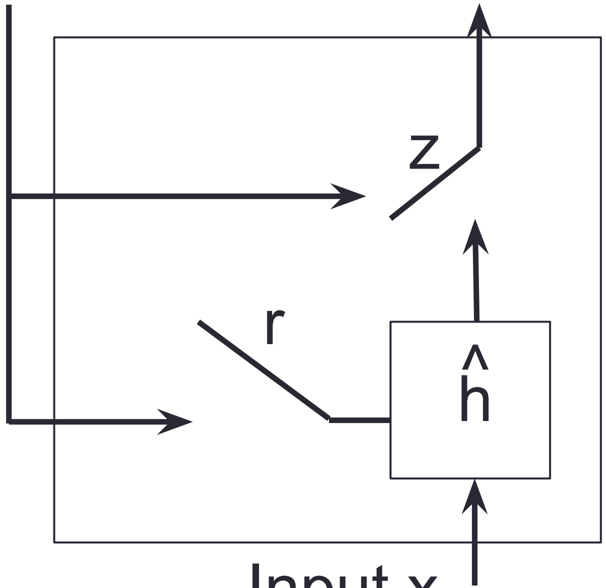
$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$z_t^j = sigmoid^j(W_{\boxed{z}}\mathbf{x}_t + U_{\boxed{z}}\mathbf{h}_{t-1})$$

Indicates a different set of weights

Gated Recurrent Unit (GRU)



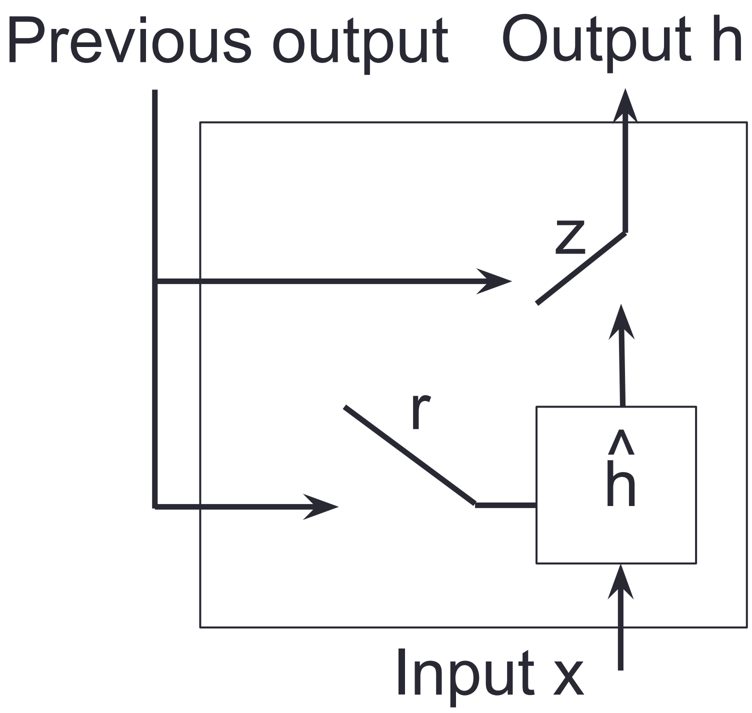
$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$z_t^j = \underline{sigmoid}^j(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})$$

Bounds the output to 0 to 1 for interpolation

Gated Recurrent Unit (GRU)



$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

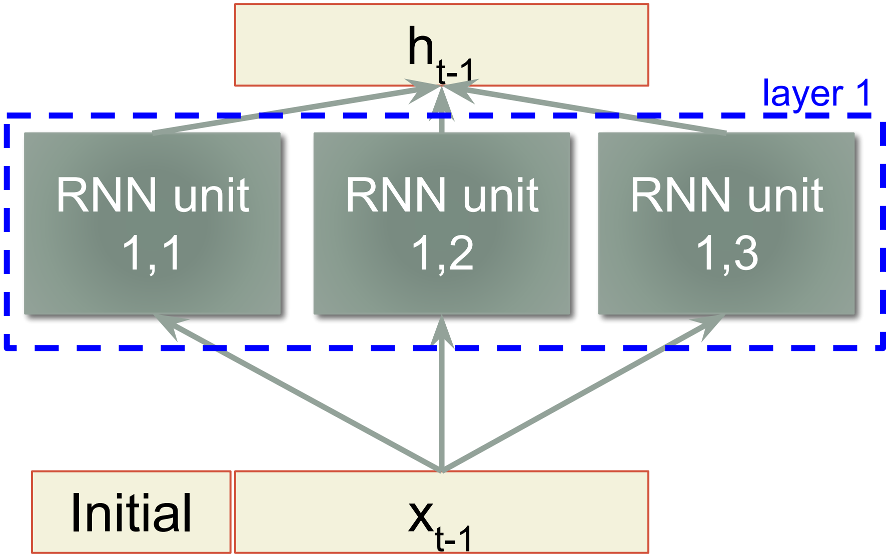
$$z_t^j = sigmoid^j(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})$$

$$r_t^j = sigmoid^j(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1})$$

Gated Recurrent Unit (GRU) layer

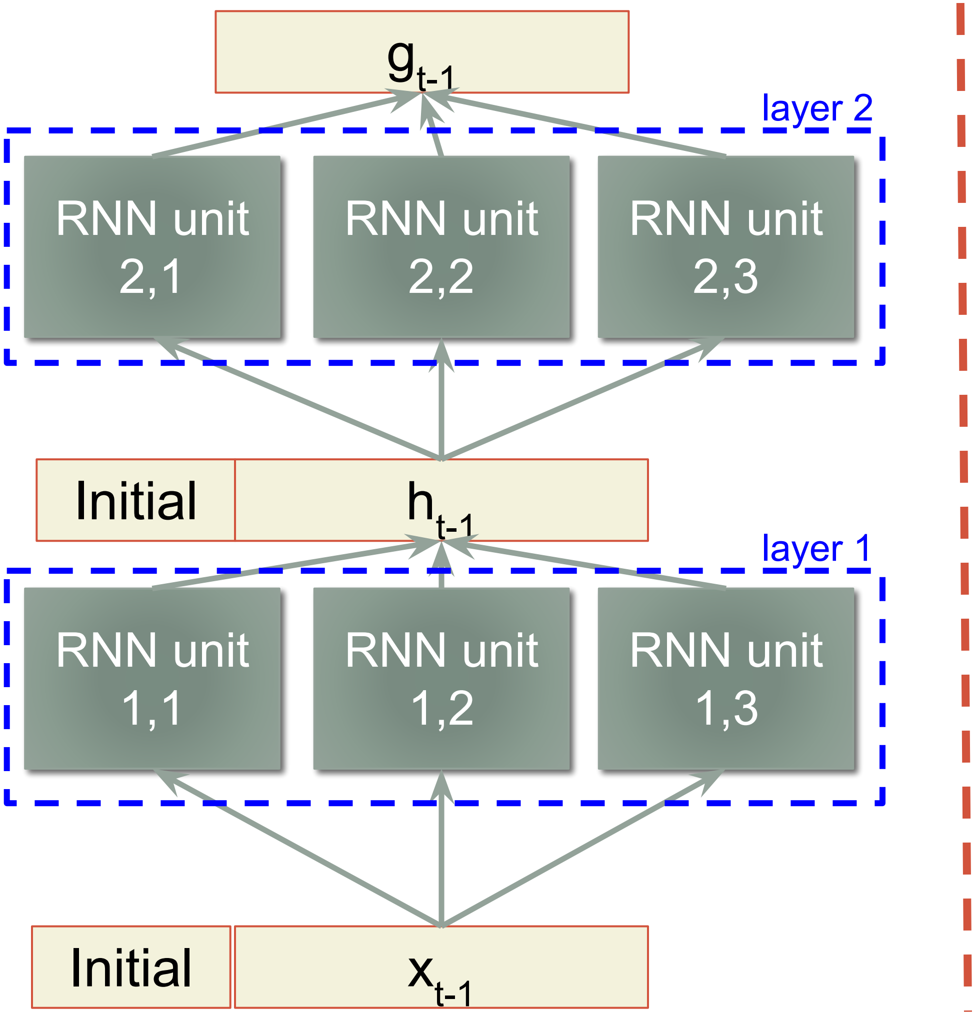
Time step 1

Time step 2



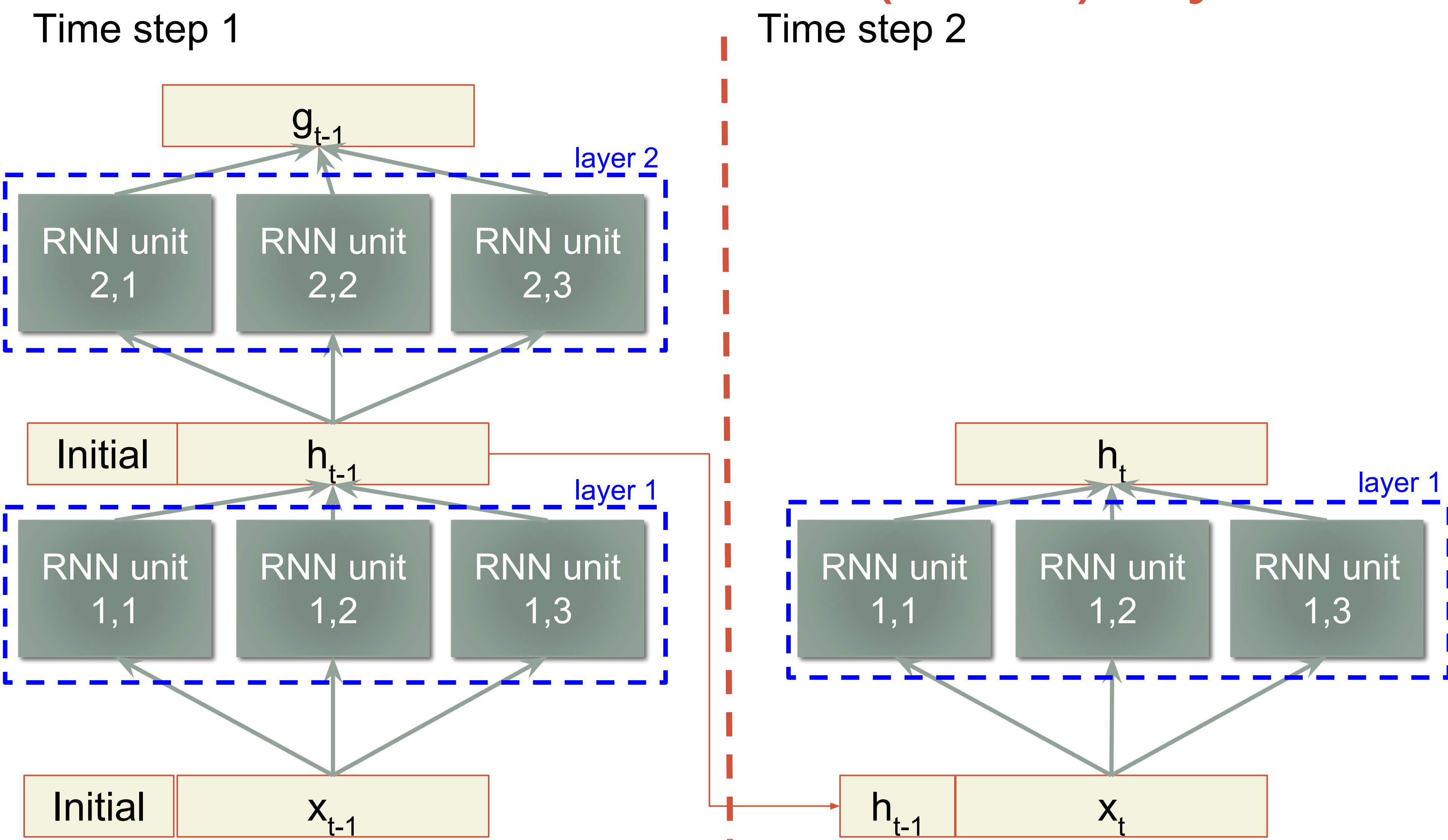
Gated Recurrent Unit (GRU) layer

Time step 1

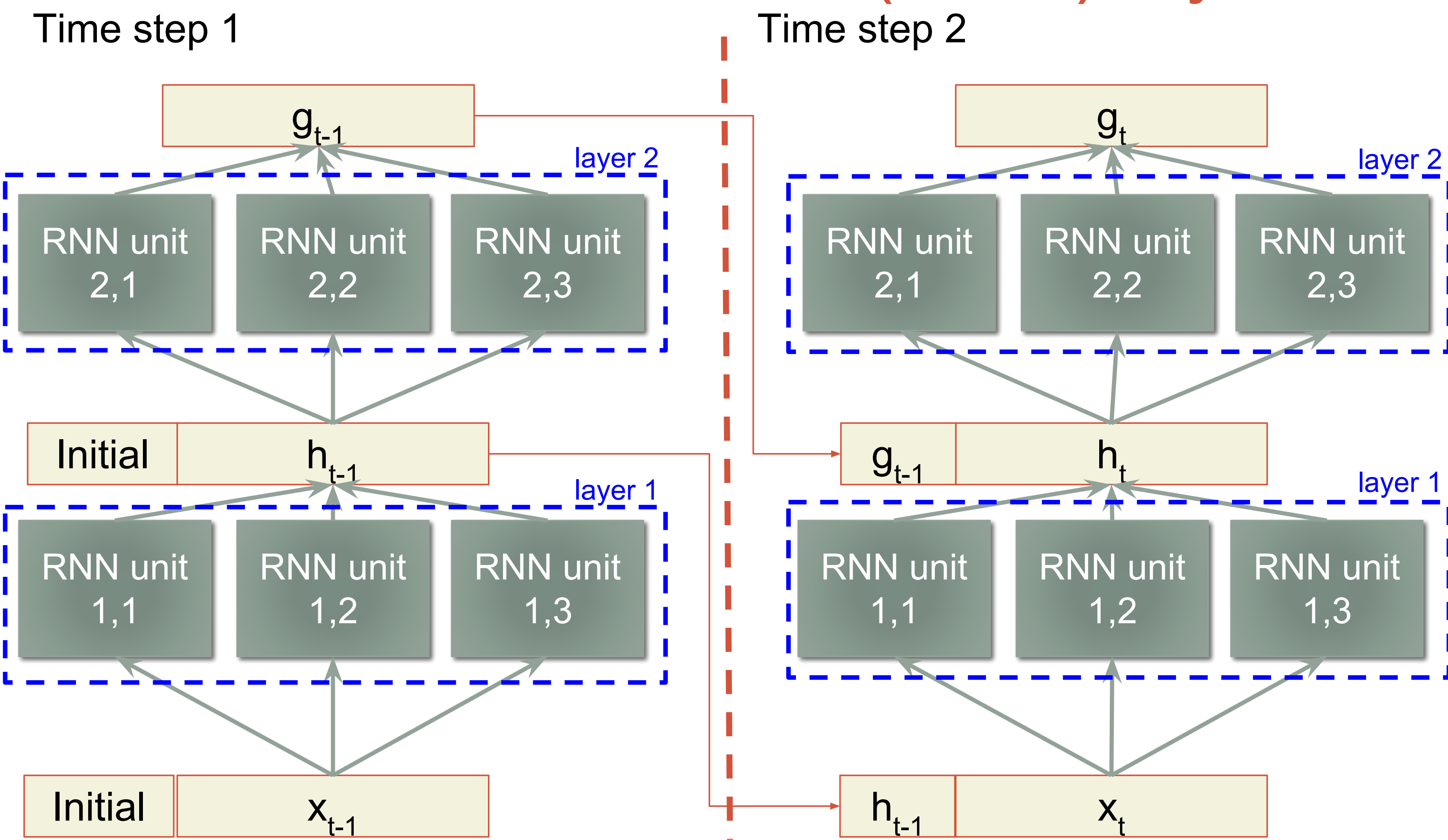


Time step 2

Gated Recurrent Unit (GRU) layer

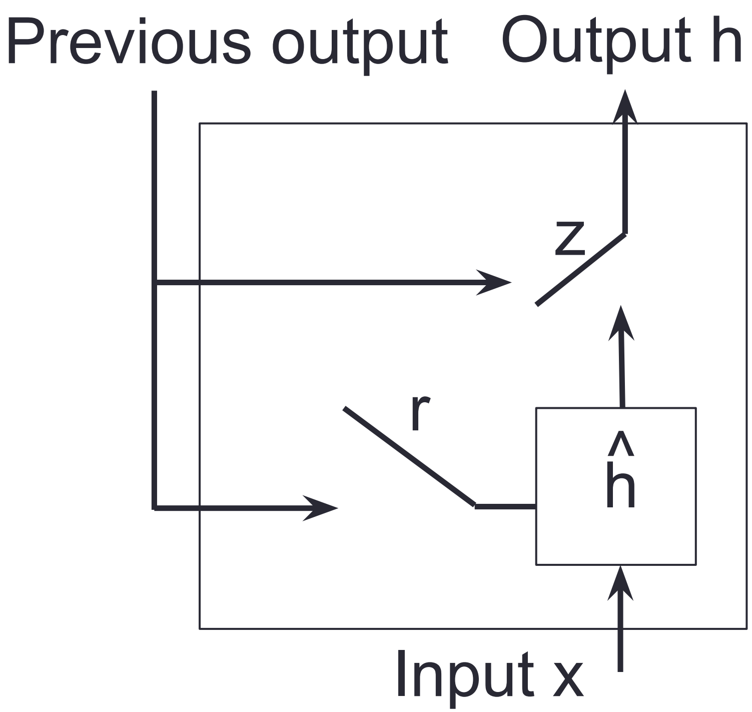


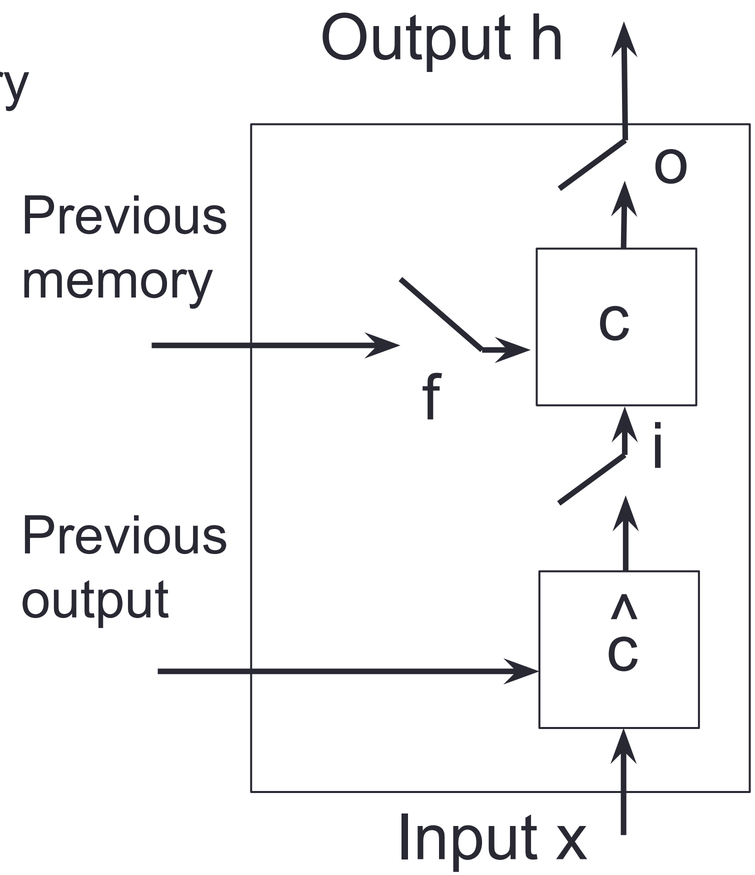
Gated Recurrent Unit (GRU) layer



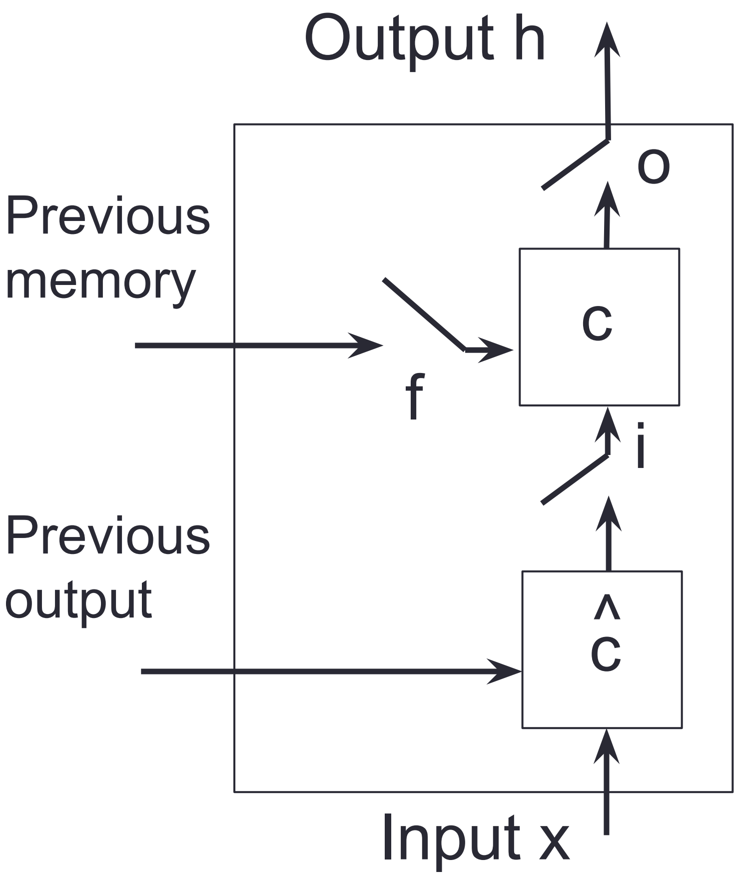
Long Short-Term Memory (LSTM)

- Have 3 gates, forget (f), input (i), output (o)
- Has an explicit memory cell (c)
  - Does not have to output the memory





Long Short-Term Memory (LSTM)



$$i_t^j = F^j(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + V_i\mathbf{c_{t-1}})$$

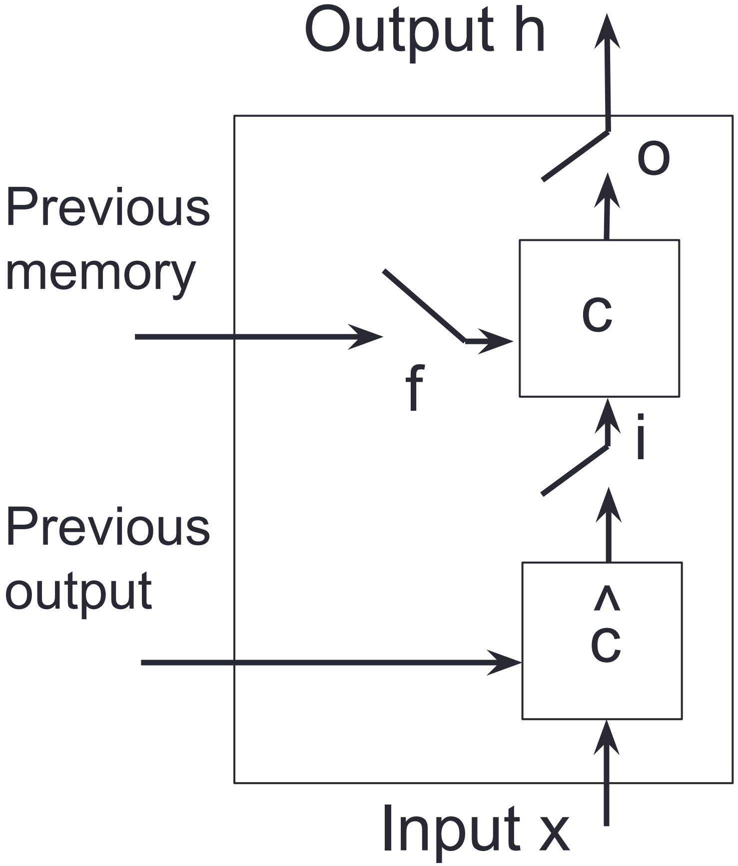
$$o_t^j = F^j(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + V_o\mathbf{c_t})$$

$$f_t^j = F^j(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + V_j\mathbf{c_{t-1}})$$

Contribution from memory "Peephole connection"

Vs are diagonal matrices(Each cell can only see its own memory)

Long Short-Term Memory (LSTM)



$$i_t^j = F^j(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + V_i\mathbf{c_{t-1}})$$

$$o_t^j = F^j(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + V_o\mathbf{c_t})$$

$$f_t^j = F^j(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + V_j\mathbf{c_{t-1}})$$

$$h_t^j = o_t^j tanh(c_t^j)$$

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \hat{c}_t^j$$

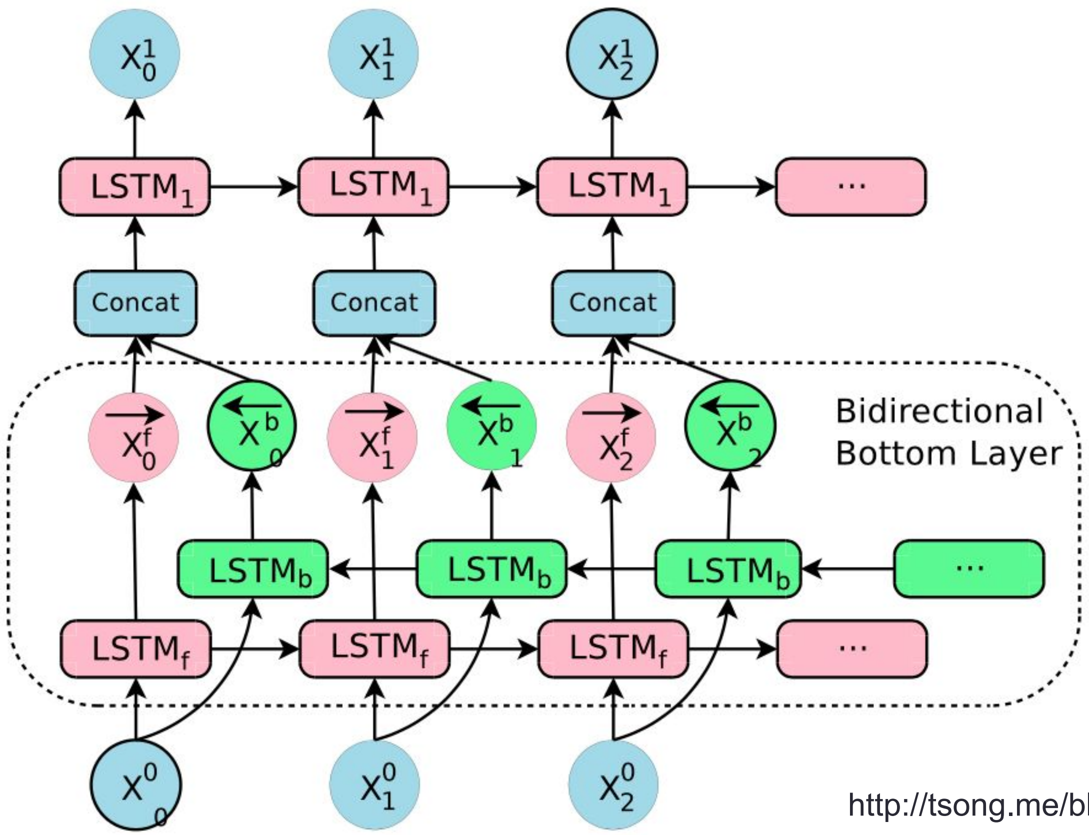
$$\hat{c}_t^j = tanh^j(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1})$$

GRU vs LSTM

- GRU and LSTM offers the same performance with large dataset
  - GRU better for smaller dataset (less parameters)
  - GRU faster to train and faster runtime (smaller model)
- Use GRUs!

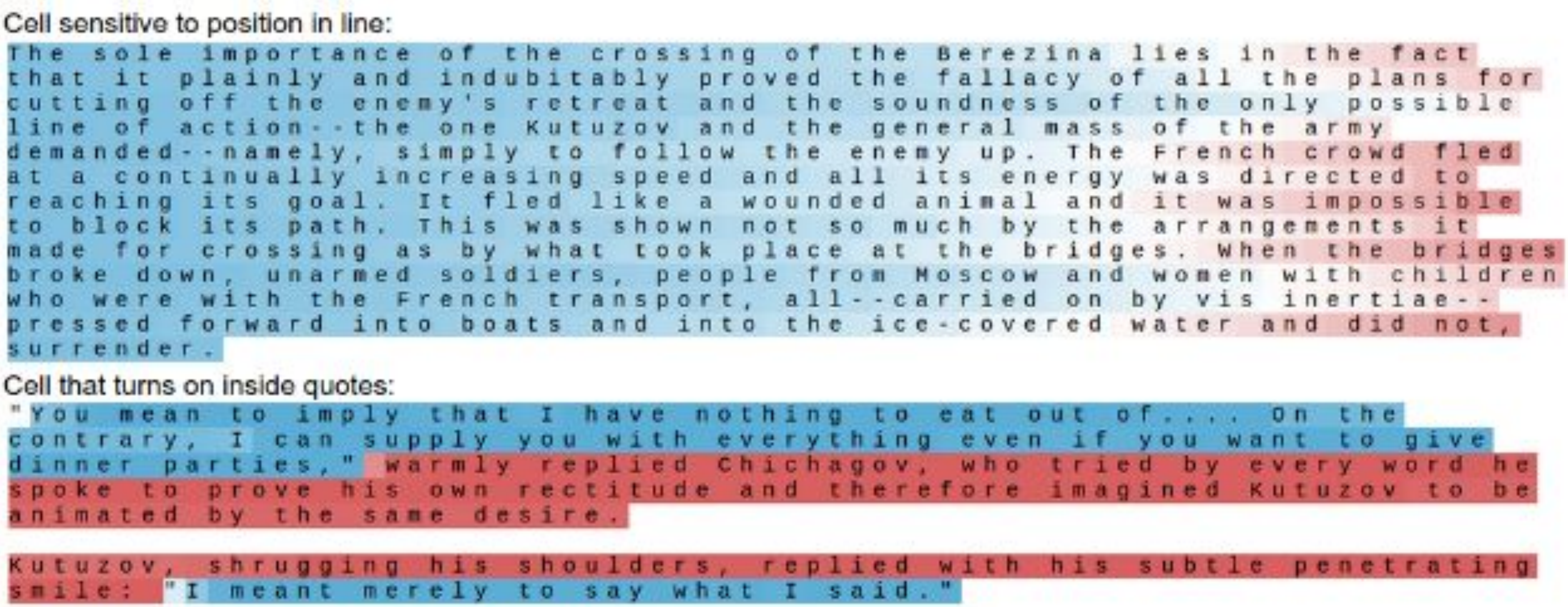
Bi-directional LSTM

- The previous GRU/LSTM only goes backward in time (uni-directional)
- Most of the time information from the future is useful for predicting the current output



http://tsong.me/blog/google-nmt/

LSTM remembers meaningful things



Andrej Karpathy, Visualizing and Understanding Recurrent Networks, 2015,

https://arxiv.org/abs/1506.02078

Embeddings

- A way to encode information to a lower dimensional space
  - PCA
  - We learn about this lower dimensional space through data

One hot encoding

- Categorical representation is usually represented by one hot encoding

- Categorical representations examples:
  - Words in a vocabulary, characters in Thai language

Apple -> 1 -> [1, 0, 0, 0, …]

Bird -> 2 -> [0, 1, 0, 0, …]

Cat -> 3 -> [0, 0, 1, 0, …]

- Sparse representation
  - Spare means most dimension are zero

One hot encoding

- Sparse – but lots of dimension
  - Curse of dimensionality
- Does not represent meaning.

Apple -> 1 -> [1, 0, 0, 0, …]

Bird -> 2 -> [0, 1, 0, 0, …]

Cat -> 3 -> [0, 0, 1, 0, …]

|Apple – Bird| = |Bird – Cat|

Getting meaning into the feature vectors

- You can add back meanings by hand-crafted rules
- Old-school NLP is all about feature engineering
- Word segmentation example:
  - Cluster Numbers
  - Cluster letters
- Concatenate them
- 1 = [0 0 0 0 1 0 0 0, 1, 0]
- ก = [0 0 0 1 0 0 0 0, 0, 1]
- า = [1 0 0 0 0 0 0 0, 0, 2]
- Which rules to use?
  - Try as many as you can think of, and do feature selection or use models that can do feature selection

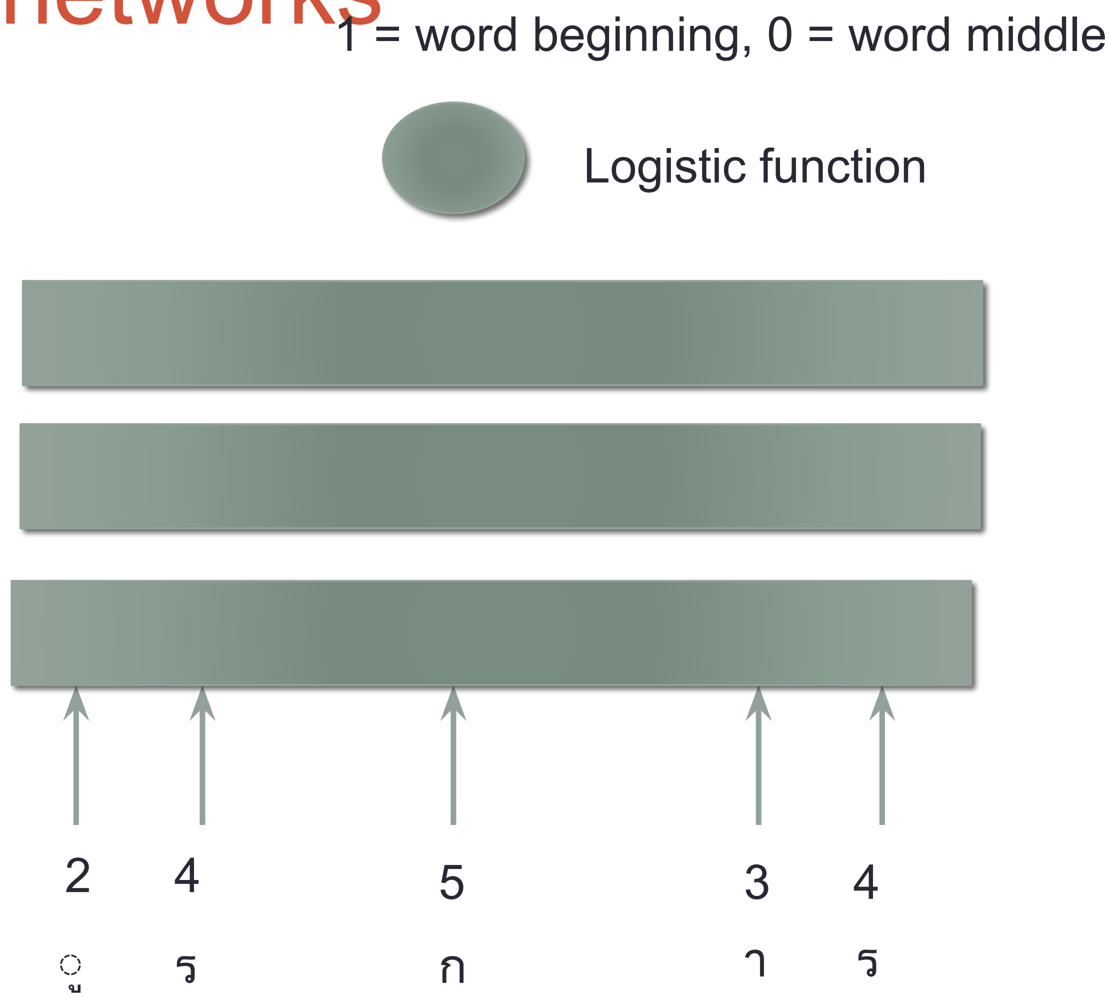
Dense representation

- We can encode sparse representation into a lower dimensional space
  - F: R^N -> R^M, where N>M

Apple -> 1 -> [1, 0, 0, 0, …] -> [2.3, 1.2]
Bird -> 2 -> [0, 1, 0, 0, …] -> [-1.0, 2.4]
Cat -> 3 -> [0, 0, 1, 0, …] -> [-3.0, 4.0]

- We can do this by using an embedding layer

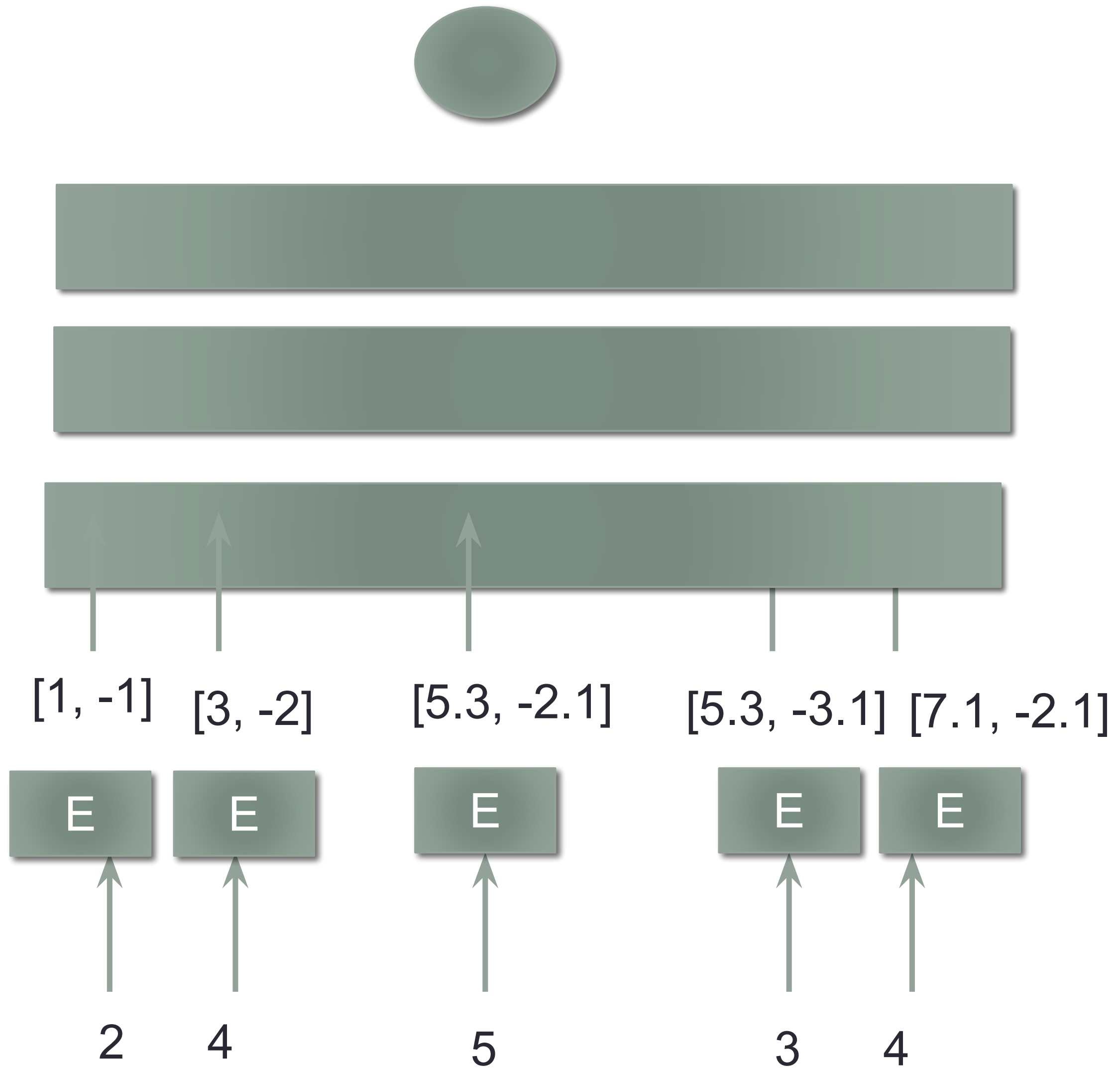
Word segmentation with fully connected networks



Adding embedding layer

Embedding layer shares the same weights

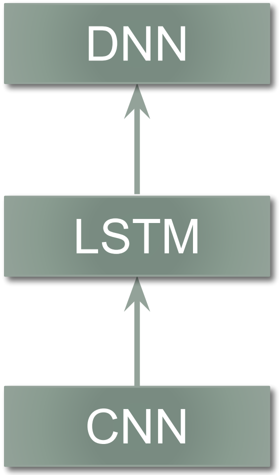
Parameter sharing!

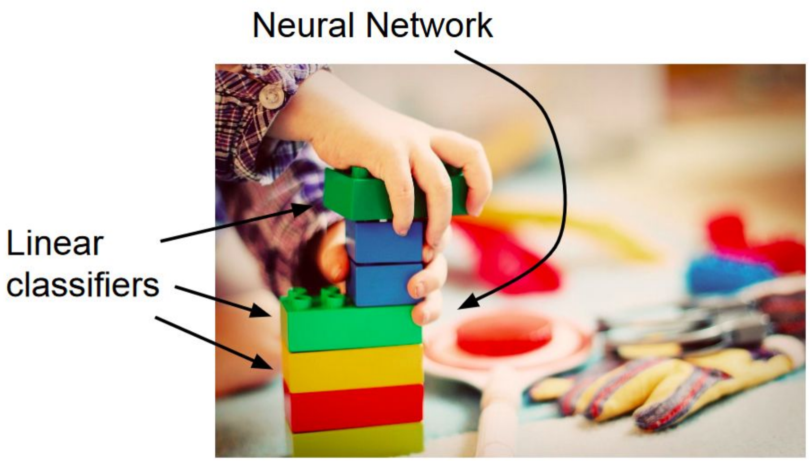


More on embeddings in the next two lectures!

DNN Legos

- Typical models now consists of all 3 types
  - CNN: local structure in the feature. Used for feature learning.
  - LSTM: remembering longer term structure or across time
  - DNN: Good for mapping features for classification. Usually used in final layers

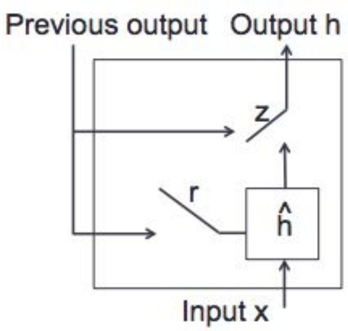


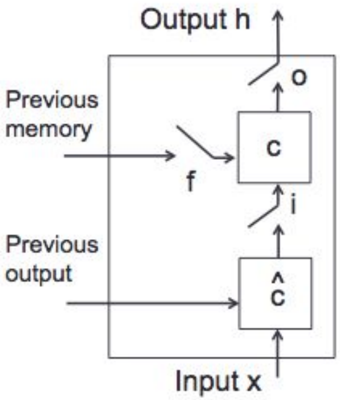


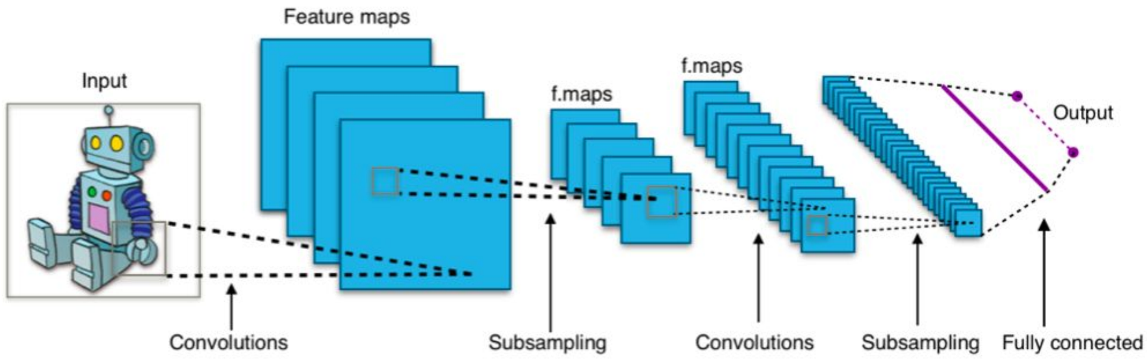
https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/

Neural networks

- Fully connected networks
  - SGD, backprop
- CNN
- RNN, LSTM, GRU







Attention modeling
Object detection ← Next lecture

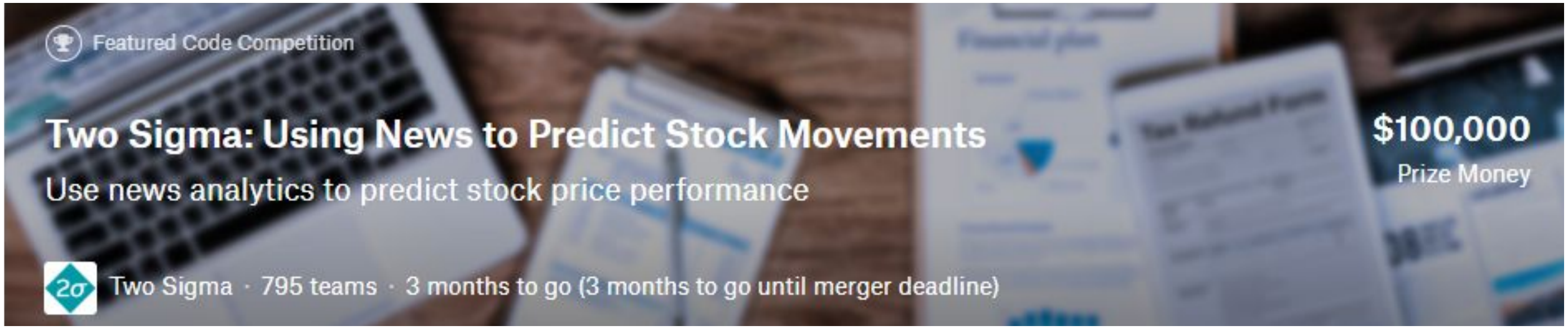
Tensorboard demo

Gcloud demo

Course project

- 4 people
- Topic of your choice
  - Can be implementing a paper
  - Extension of a homework
  - Project for other courses with an additional machine learning component
  - Your current research (with additional scope)
  - Or work on a new application
  - Must already have existing data! No data collection!
- Topics need to be pre-approved

Projects idea



https://www.kaggle.com/c/two-sigma-financial-news

Project doodle

- Build your group on courseville (under project)
- Submit a proposal. Due next Tuesday.
  - Bullet points
  - What do you want to do?
  - What is the data?
  - How to evaluate the task?
- Sign up for time slots (next week's office hour)
  - Sign up sheet on courseville (as an assignment)