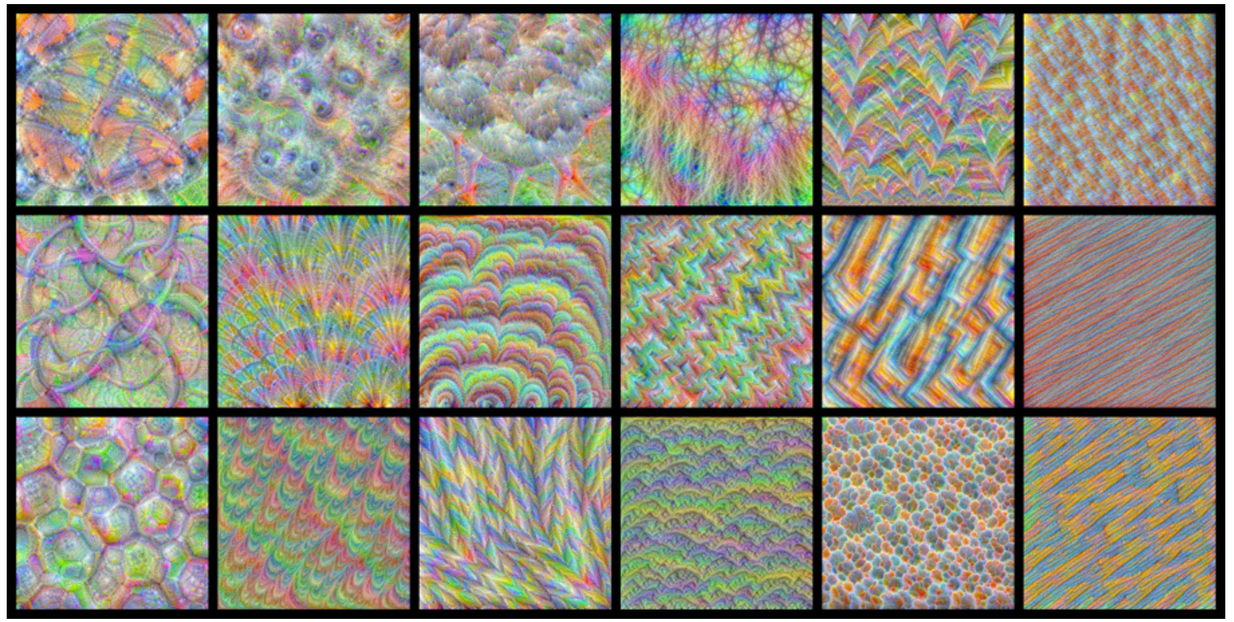
ICME Summer Workshops 2020

Introduction to Deep Learning

Session 3: 2:00—3:15 PM
Instructor: Sherrie Wang

icme-workshops.github.io/deep-learning



https://towardsdatascience.com/how-to-visualize-convolutional-features-in-40-lines-of-code-70b7d87b0030

Workshop Schedule

Session 1 (9:00—10:30 AM)
- Introduction
- Current state-of-the-art in deep learning
- Math review
- Fully connected neural networks

Session 2 (10:45—12:00 PM)
- Loss functions
- Gradient descent
- Backpropagation
- Overfitting and underfitting

Lunch (12:00—2:00 PM)

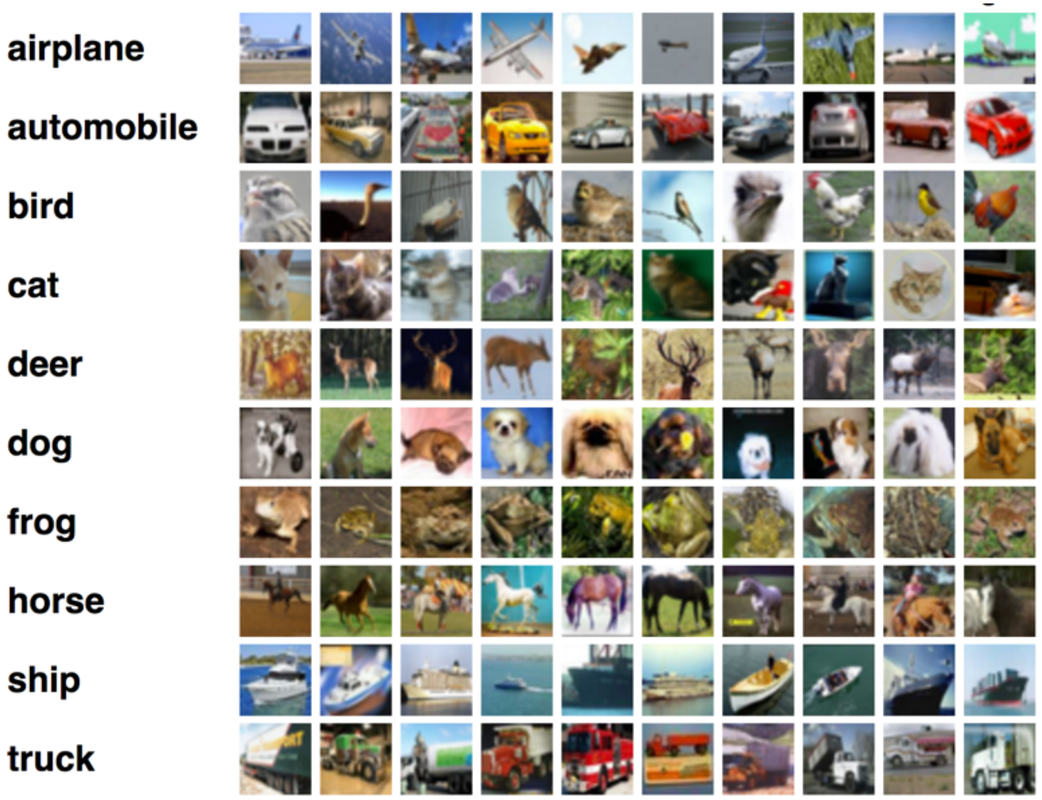
Session 3 (2:00—3:15 PM)
- Convolutional neural networks
- Recurrent neural networks
- Other architectures
- Deep learning libraries
- Hands-on coding session in Tensorflow

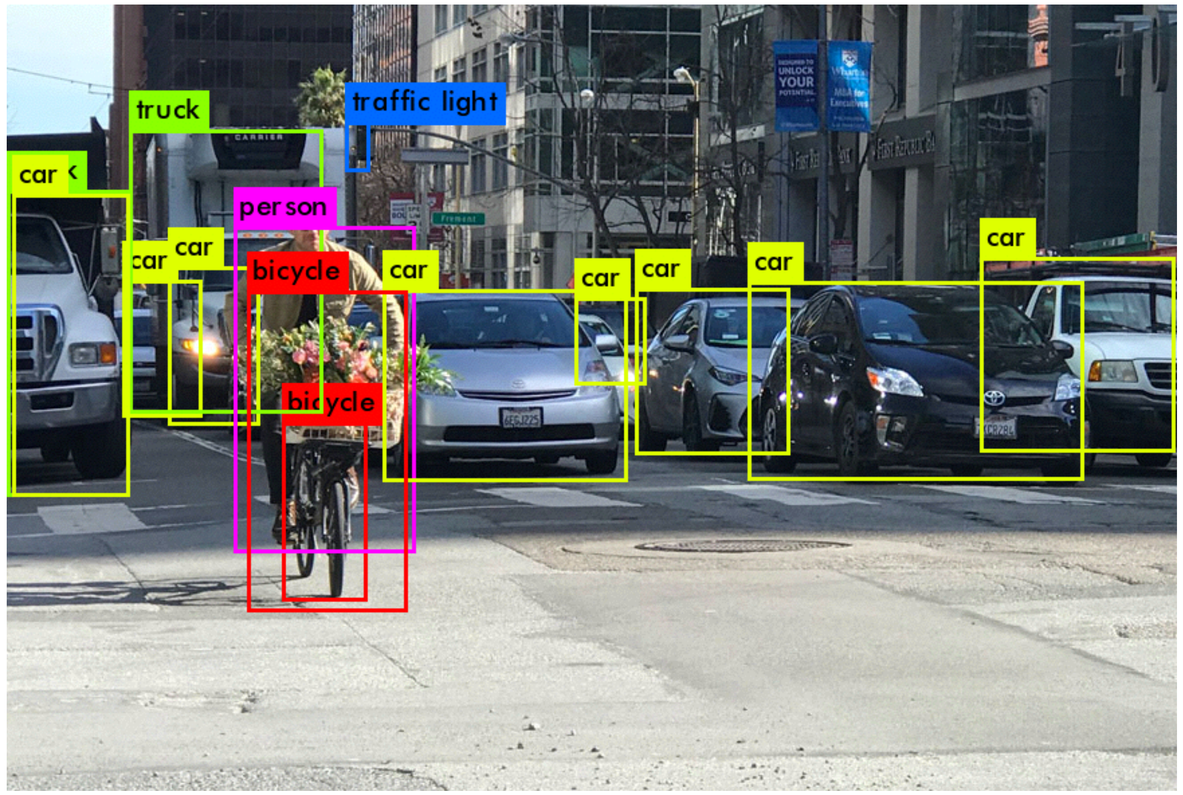
Session 4 (3:30—4:45 PM)
- Hands-on coding session in Keras
- Hands-on coding session on transfer learning
- Failures of deep learning

Convolutional Neural Networks

Convolutional neural networks

- Neural networks have been very successful at image classification and object detection





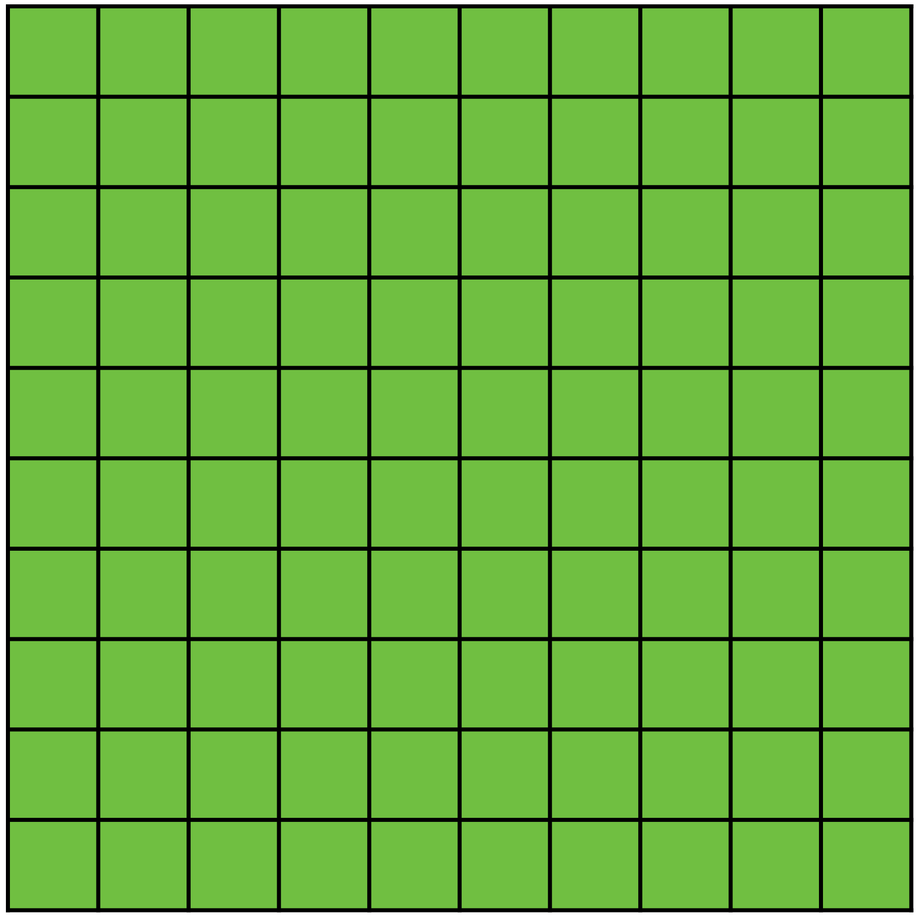
Convolutional layers

- Convolutional layers are a special type of layer for image inputs
- Convolutions replace matrix multiplication with the convolution operation

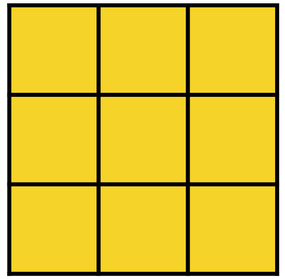
$$g(\mathbf{W}\mathbf{x} + \mathbf{b}) \longrightarrow g(\mathbf{f} * \mathbf{x} + \mathbf{b})$$

Convolutions

Image

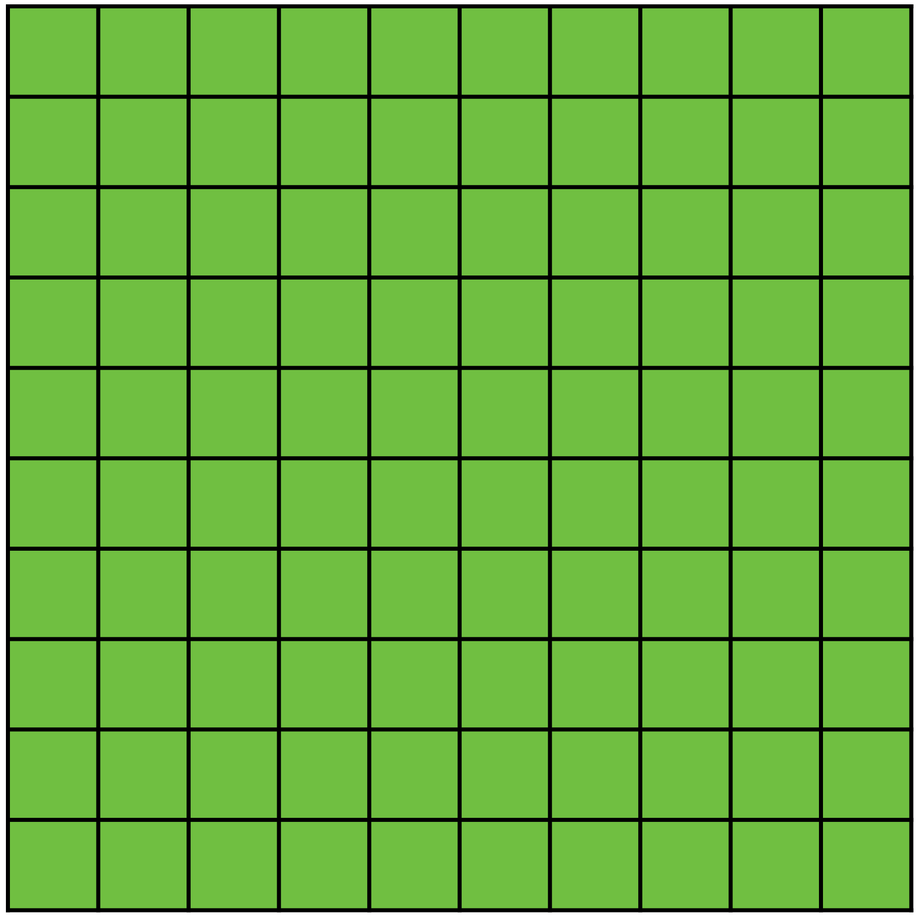


Filter



Convolutions

Image



Filter

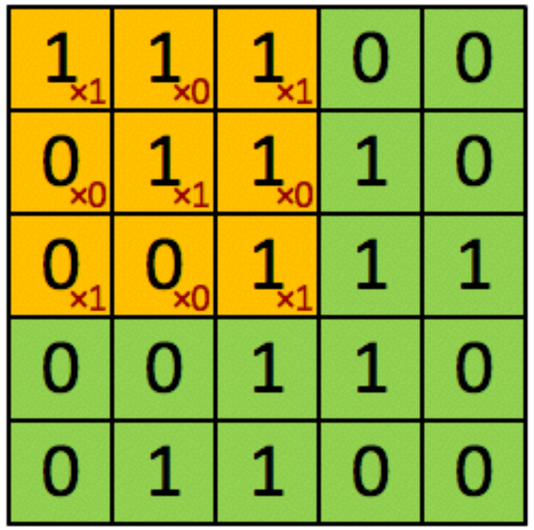
1	4	0
0	2	1
3	0	1

Learnable weights

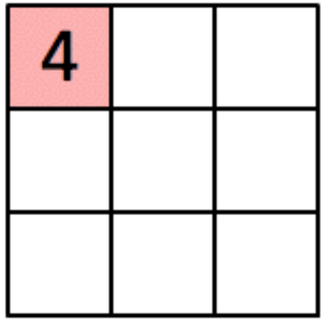
Convolutions

An example on an image with 1 channel

Notice that the dimensions of the output are smaller than the dimensions of the image by 2 pixels



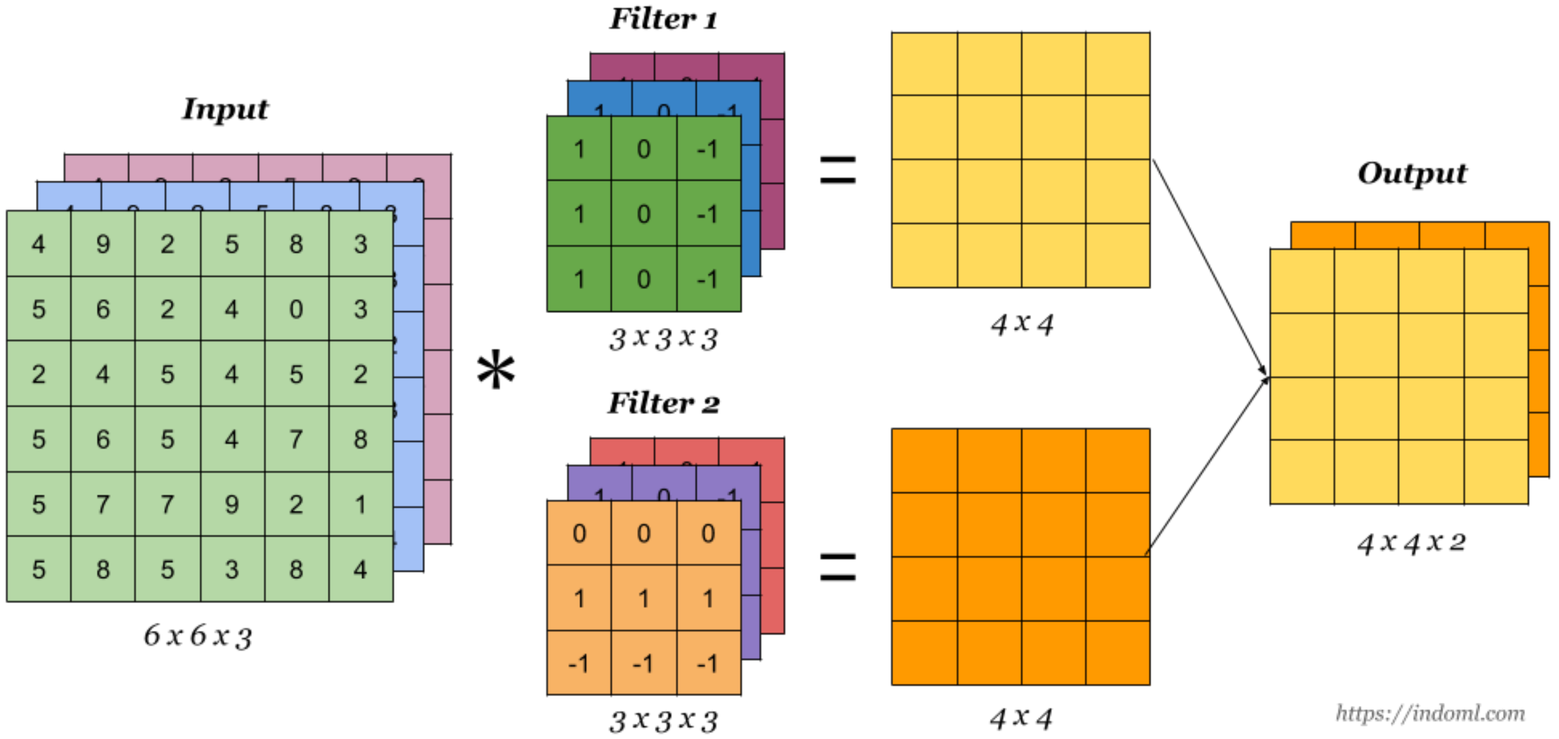
Image



Convolved Feature

Convolutions: 3 channels

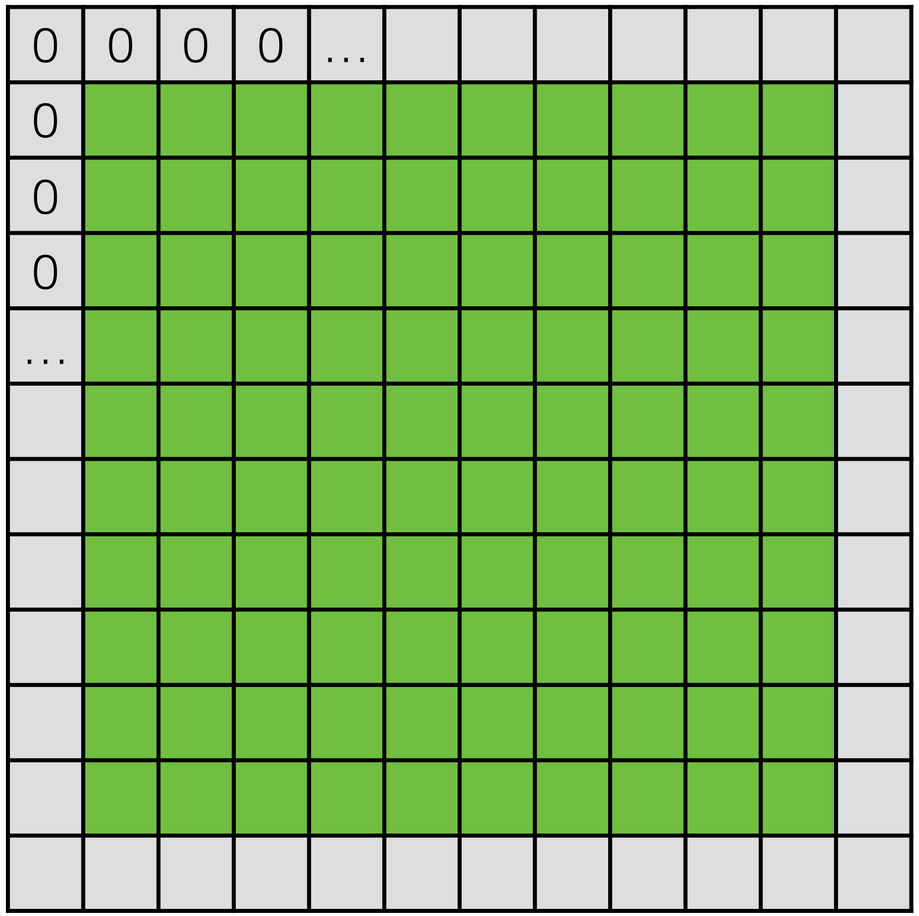
Most images are RGB



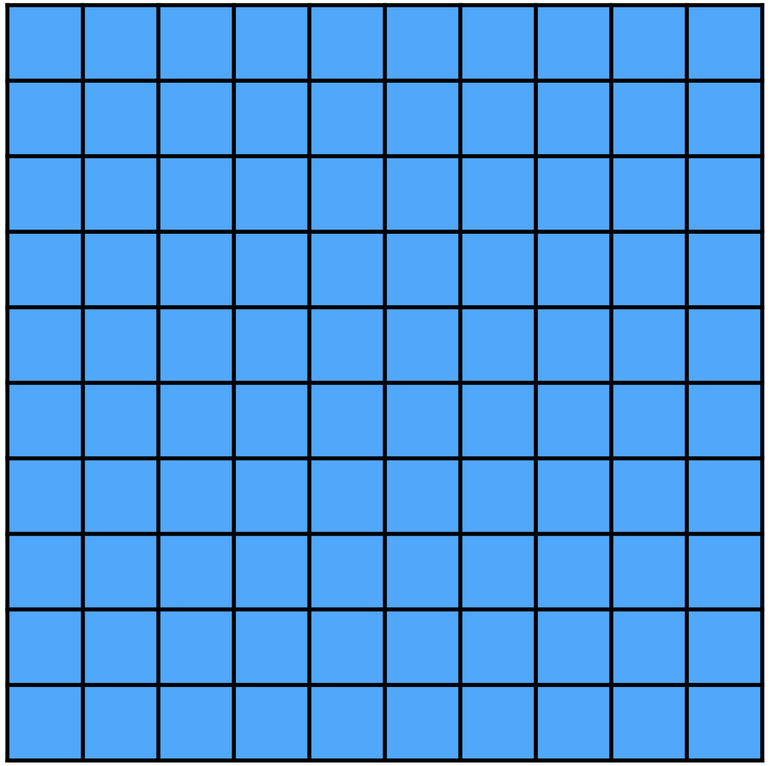
Padding

Zero-padding allows us to control the spatial size of the output — can be a hyperparameter

Image

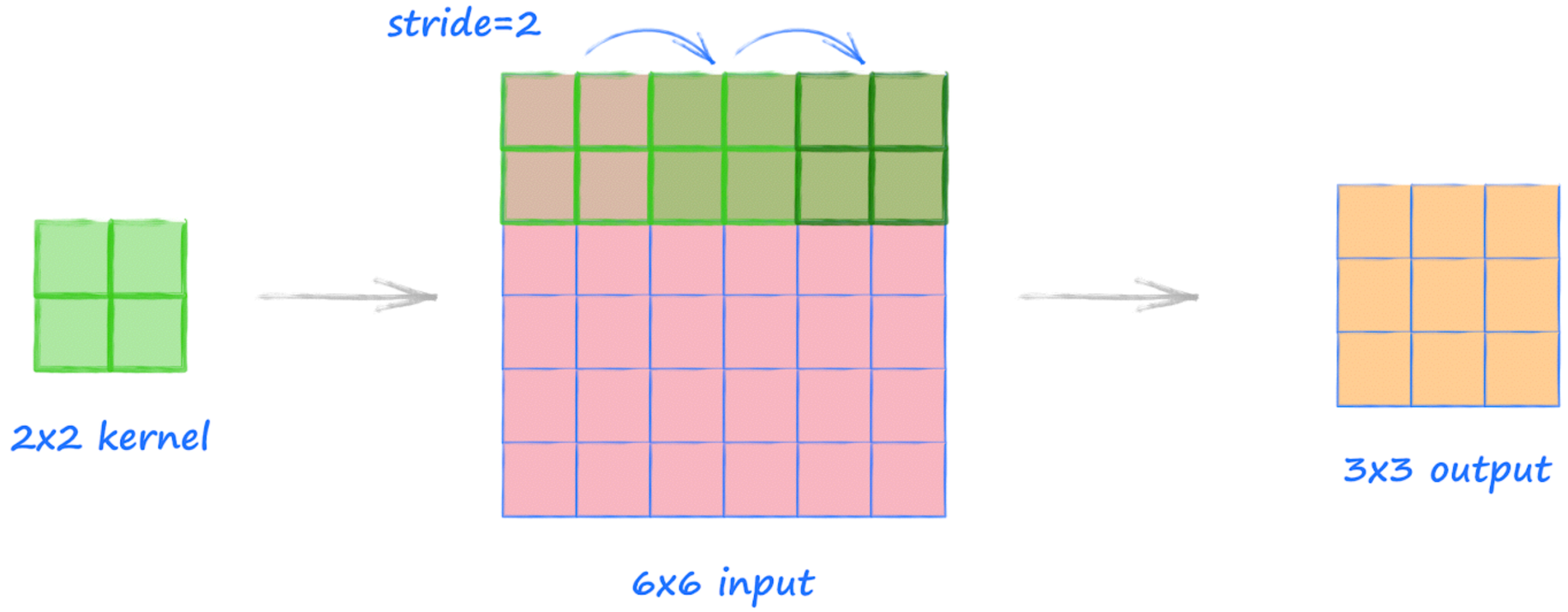


Output



Stride

Stride refers to how many pixels we shift the filter when convolving



http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html

Filter size

Filters can be of any spatial dimension smaller than or equal to the image size, but must match the number of channels in the image

The filter size is also known as the **receptive field** of the neuron

3x3

0.91	0.32	0.07
0.73	0.26	0.81
0.53	0.68	0.14

5x5

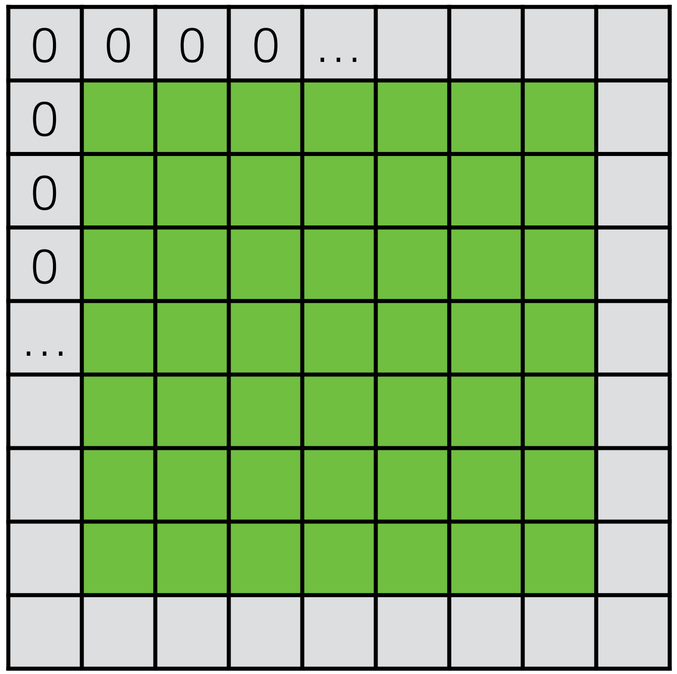
0.27	0.64	0.44	0.84	0.29
0.28	0.06	0.89	0.99	0.33
0.64	0.67	0.08	0.38	0.03
0.04	0.31	0.16	0.57	0.08
0.87	0.85	0.97	0.71	0.96

Bigger filters have wider receptive fields — but this can also be accomplished with a deeper CNN

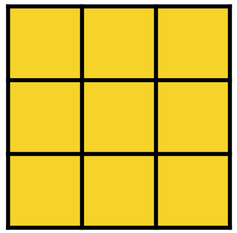
Poll: Output dimensions

Given an input image of dimension 7x7x1, one filter of size 3x3x1, 1-pixel thick zero-padding, and a stride of 3, what is the output size?

Image



Filter

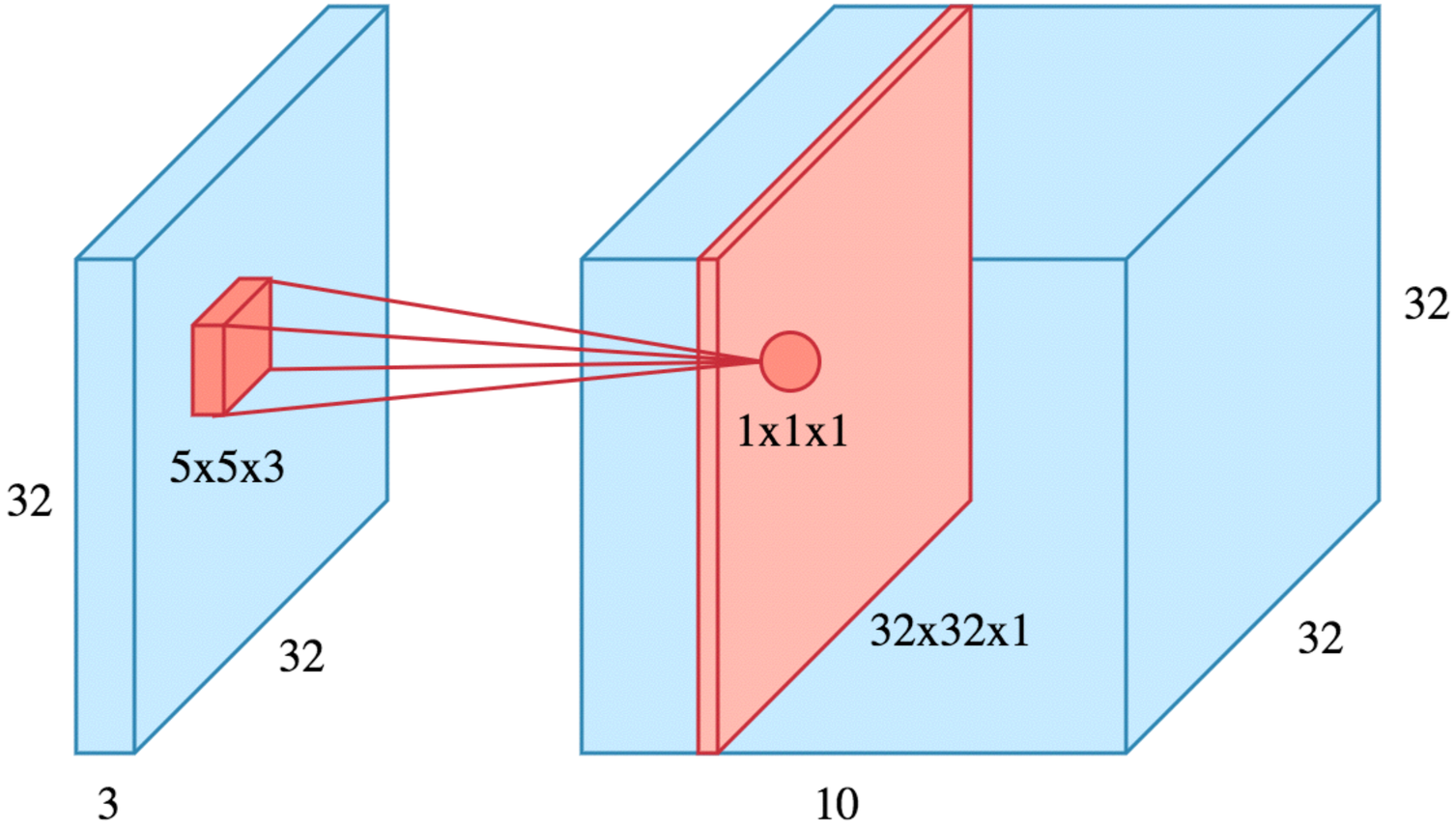


A. 3x3x1
B. 4x4x1
C. 5x5x1
D. 7x7x1

Go to PollEv.com/dlworkshop2020

Depth

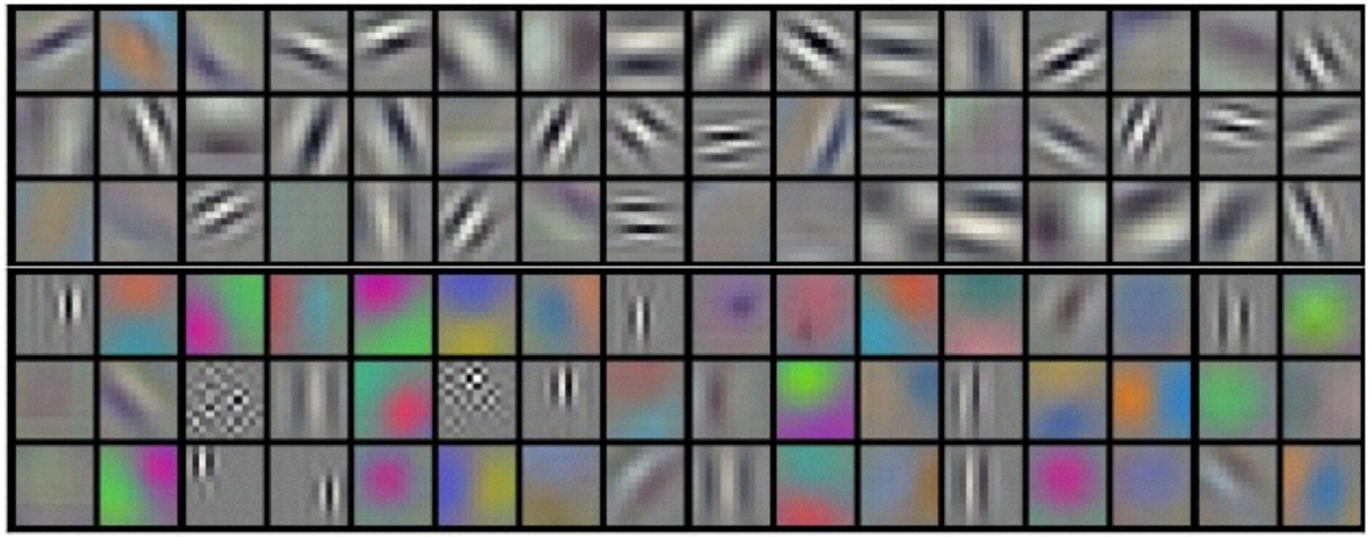
The number of filters used — more filters mean more learnable features



https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

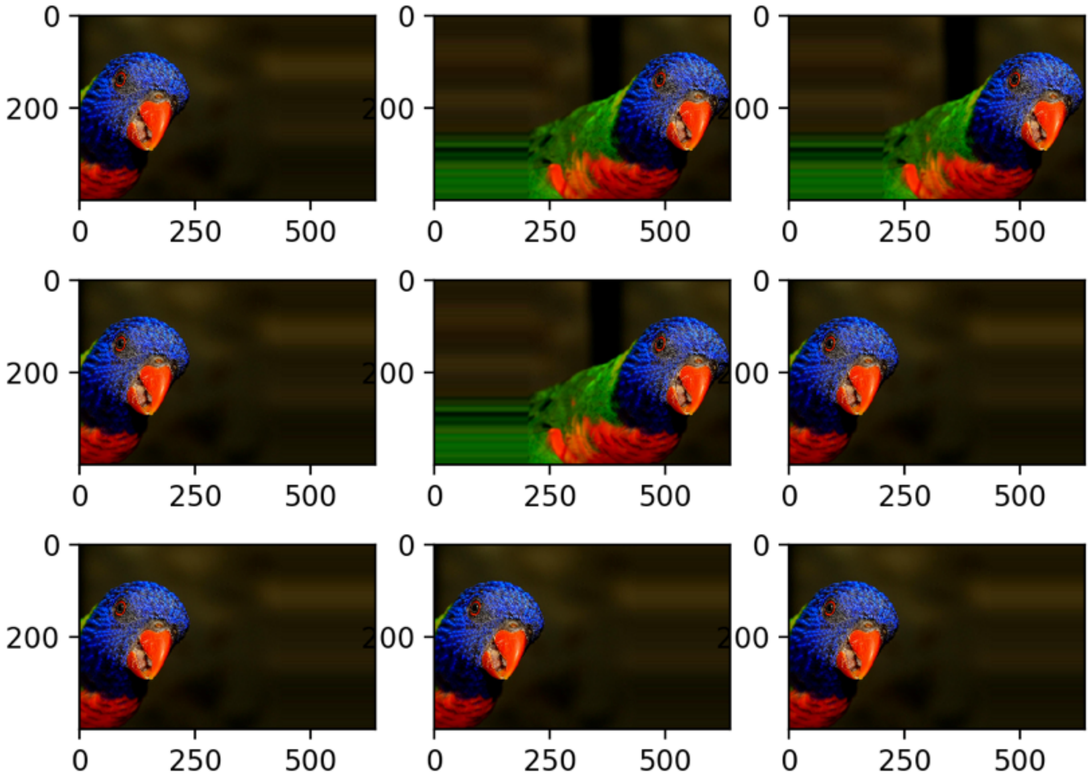
Example learned filters

- 96 filters learned by the AlexNet architecture trained on ImageNet
- Each filter is of dimension 11x11



Why use convolutions?

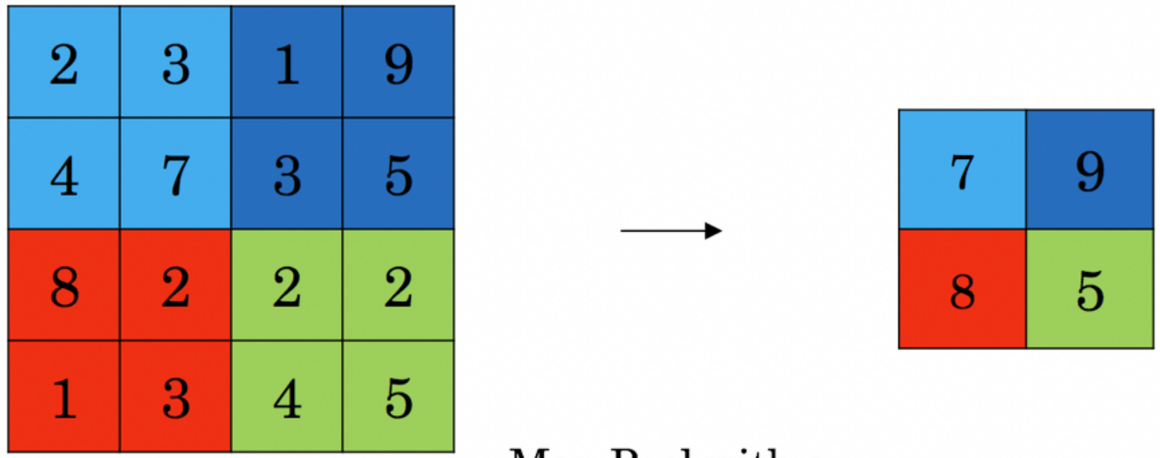
- Features in images should be invariant to translations
- Weight sharing — reduces the number of parameters to learn



No matter where the parrot is in the image, the image still contains a parrot

Pooling layers

It is common to insert pooling layers between successive Conv layers.
Pooling reduces the spatial size of the representation.
The most common pooling layer:

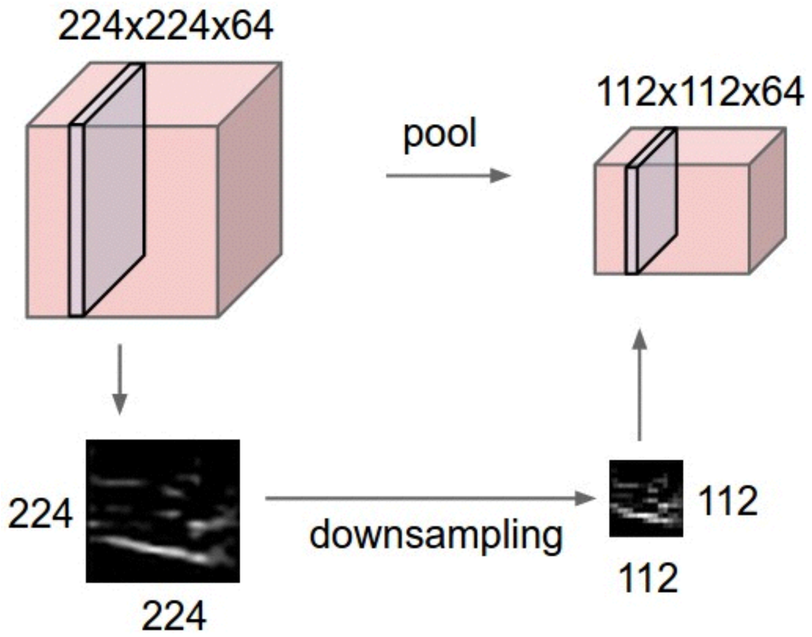


Max-Pool with a 2 by 2 filter and stride 2.

Andrew Ng

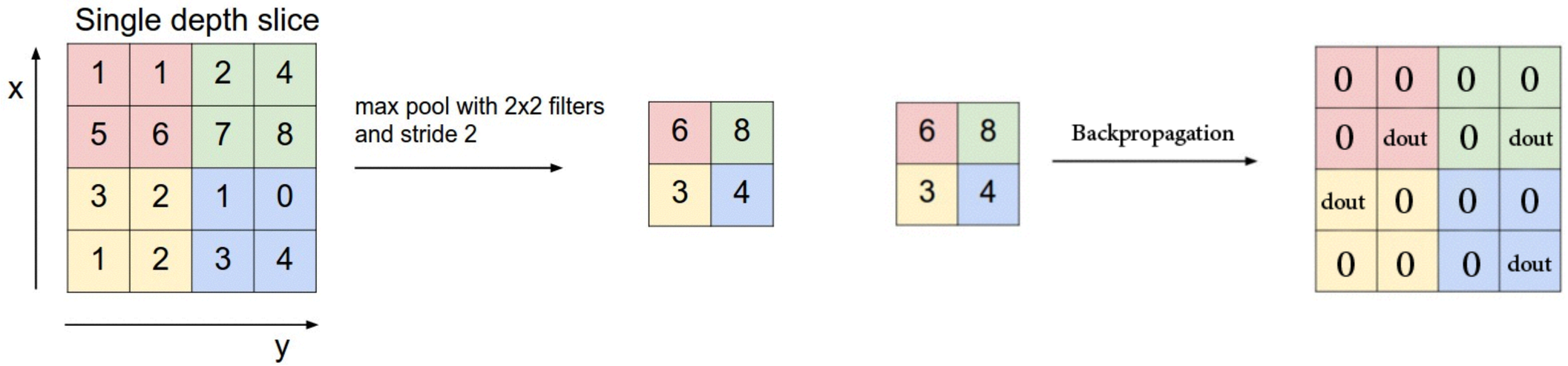
Pooling layers

Pooling layers contain no trainable parameters.
It is uncommon to use padding in pooling layers.



Recall backpropagation

Backprop through max-pooling layers routes the gradient through the index of the max activation



https://cs231n.github.io/convolutional-networks/

Stacking layers

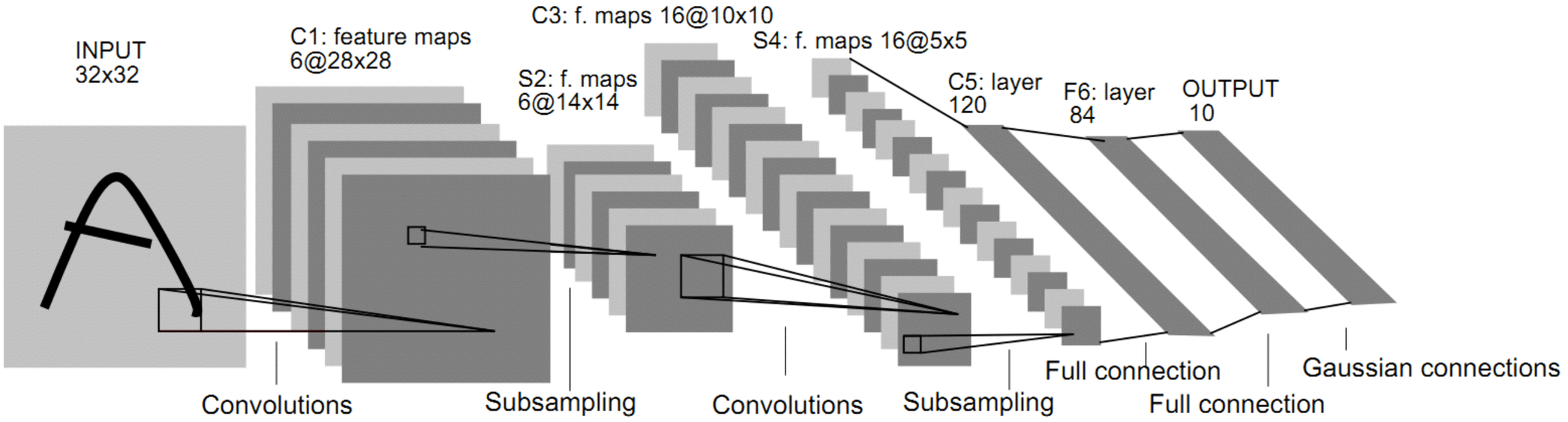
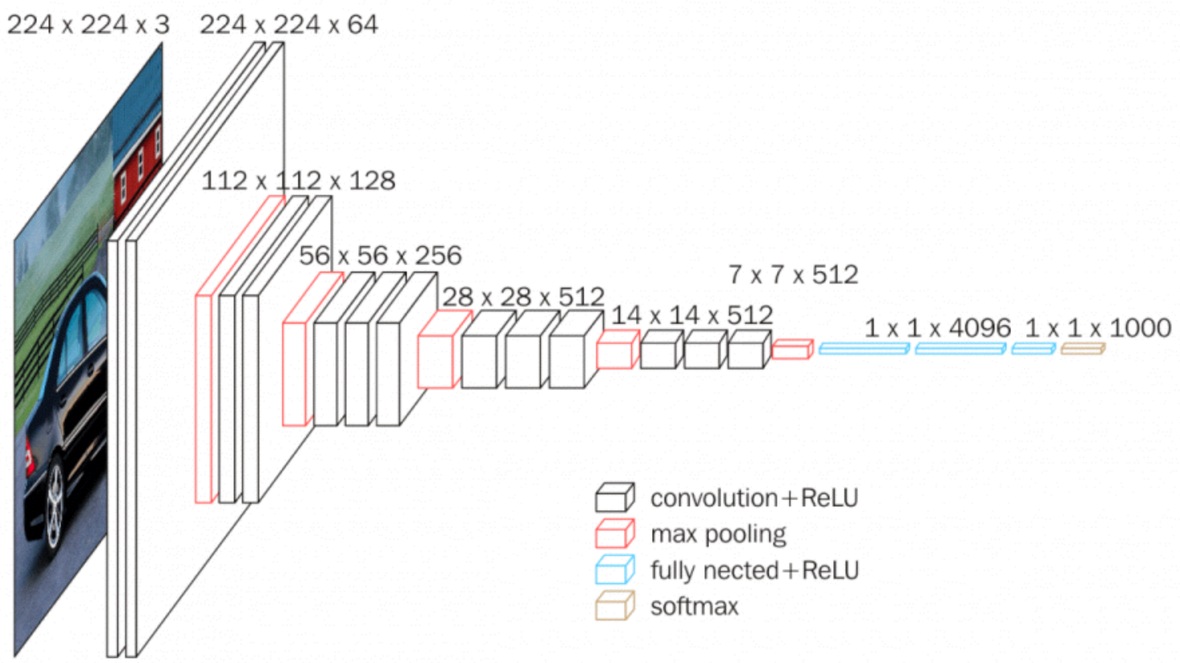


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Stacking layers

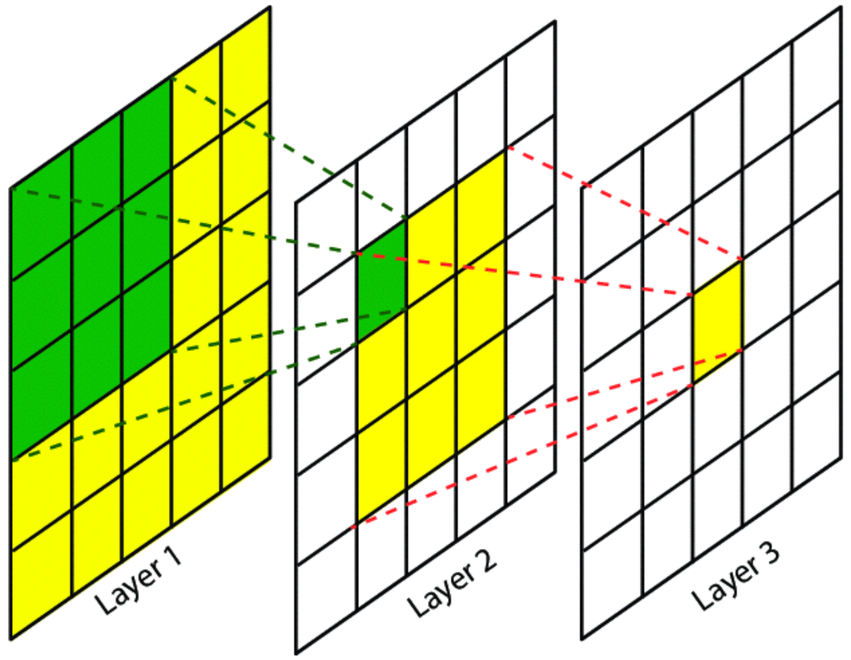
VGG-16



https://neurohive.io/en/popular-networks/vgg16/

Growing the receptive field

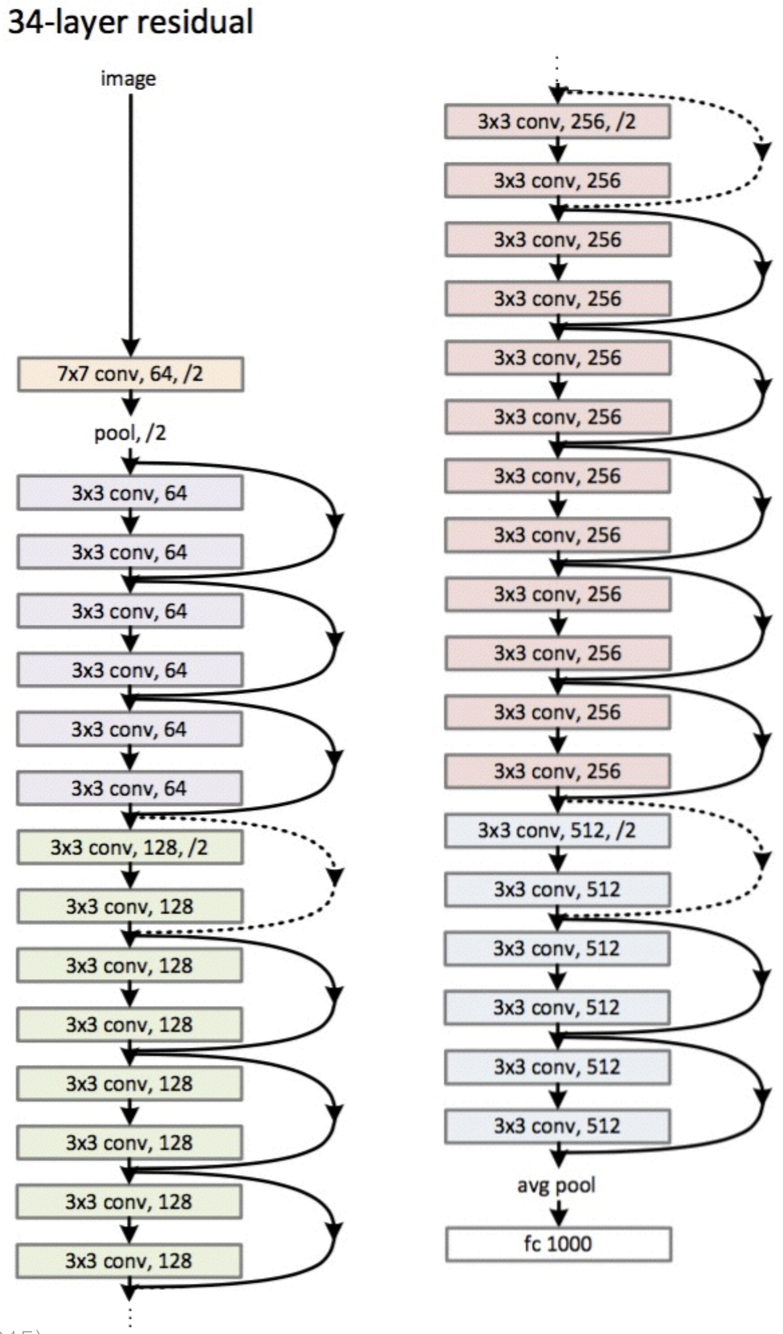
Deeper layers “see” more and more of the input all at once



https://dev.to/jai00271/cnn-general-terms-and-their-meaning-42nc

ResNet

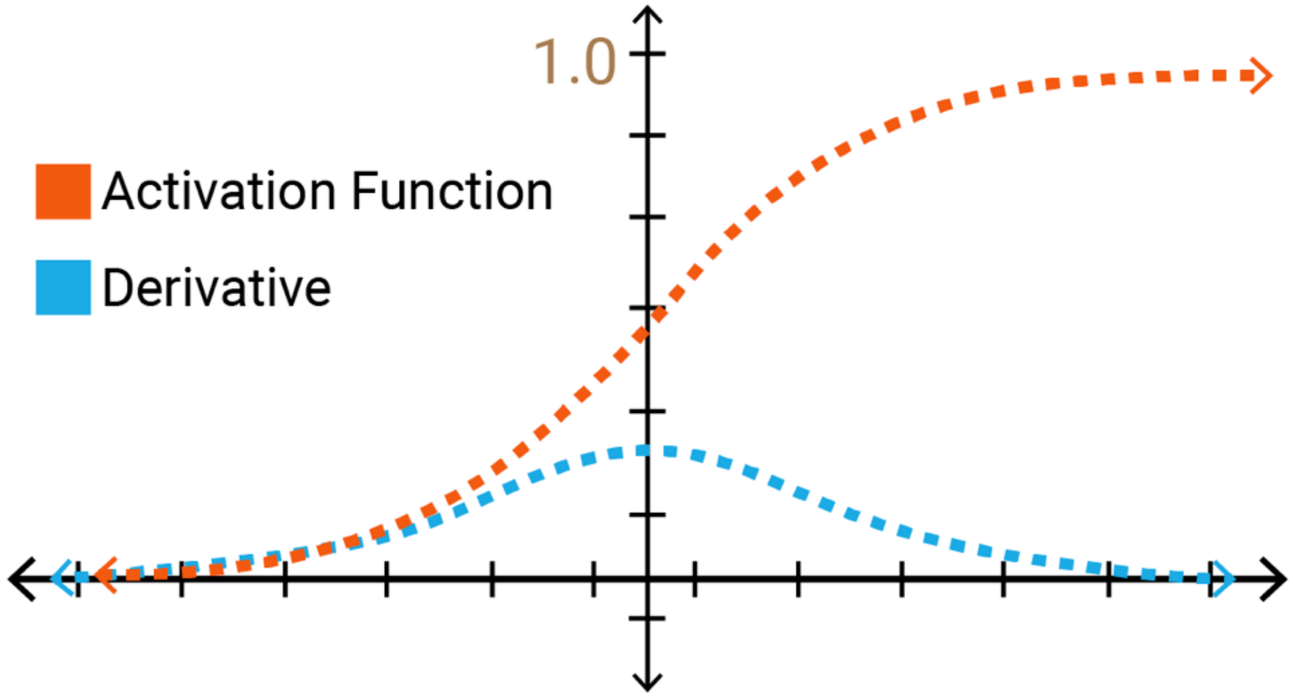
Residual neural network (ResNet) is one of the most commonly used architectures.
It contains **skip connections** that jump over some layers.
Skip connections improve training by avoiding vanishing gradients.



He *et al.* “Deep Residual Learning for Image Recognition” (2015)

Aside: Vanishing gradients

Gradients can become very small after stacking many layers of a neural network. This makes training difficult.



Recap: CNNs

CNNs take advantage of the structure of images.

They use:

- Convolutional layers
- Pooling layers

Parameters are shared for more efficient learning and translational invariance.

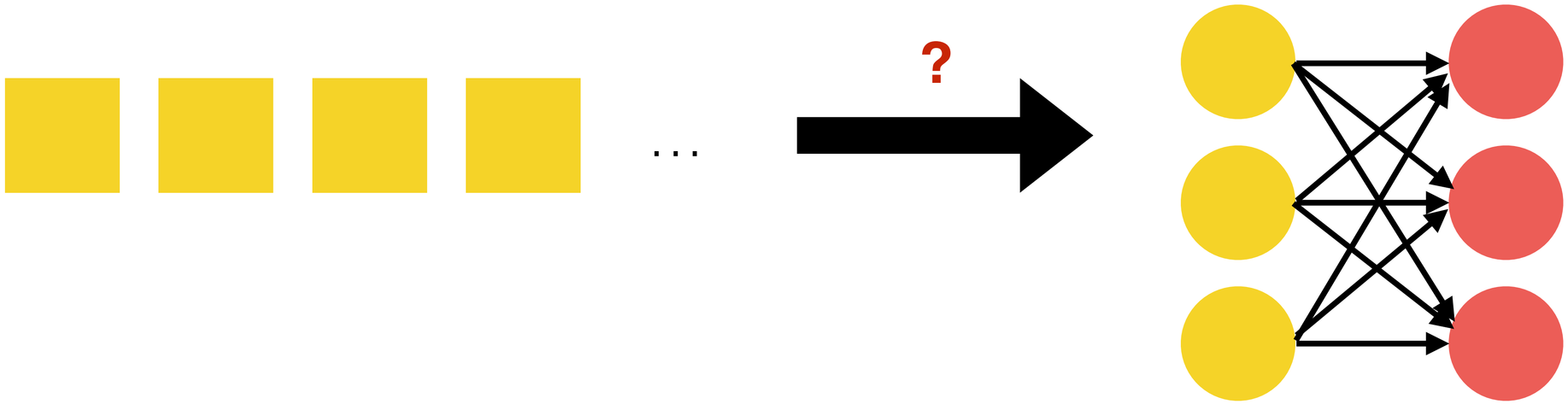
Recurrent Neural Networks

Recurrent neural networks

How to use neural networks for time series input?

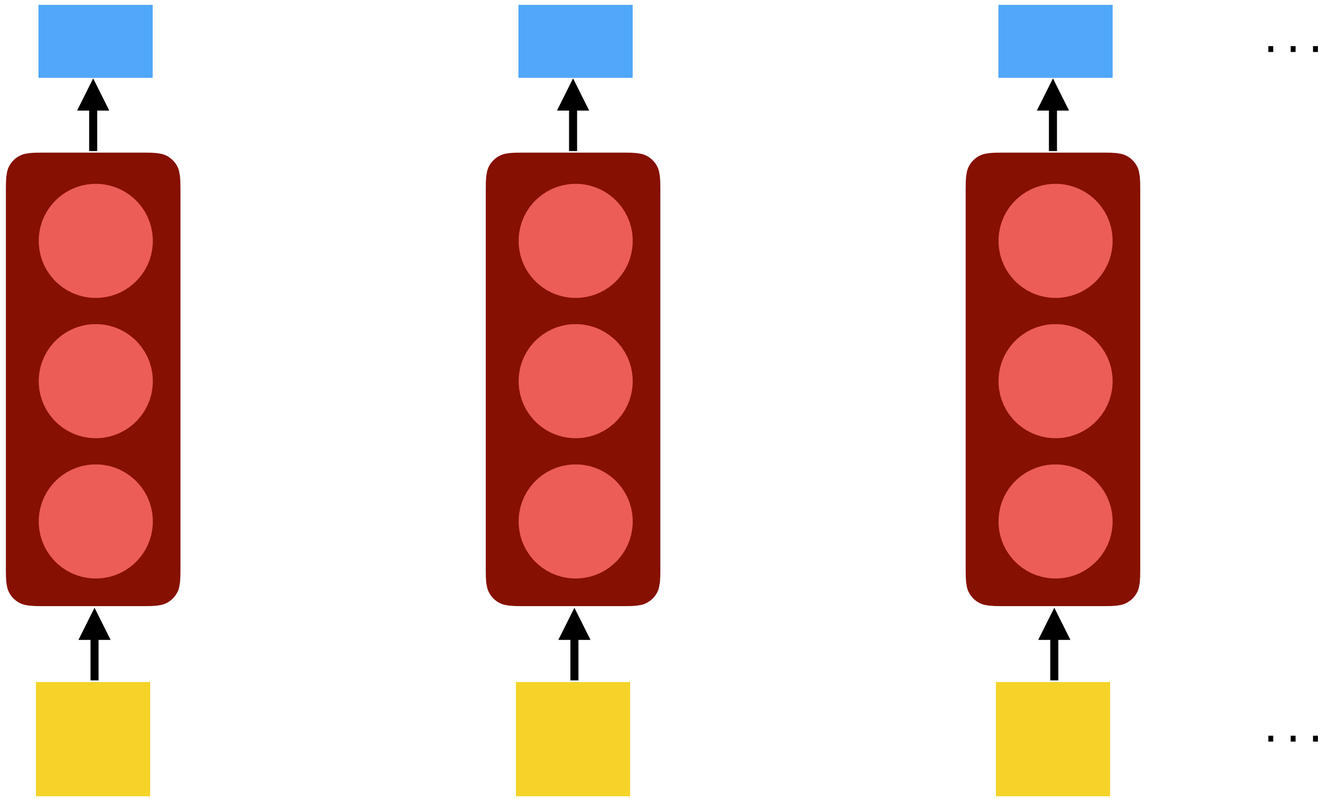
The length of input is not fixed or of indefinite length

Examples: audio, text, financial time series



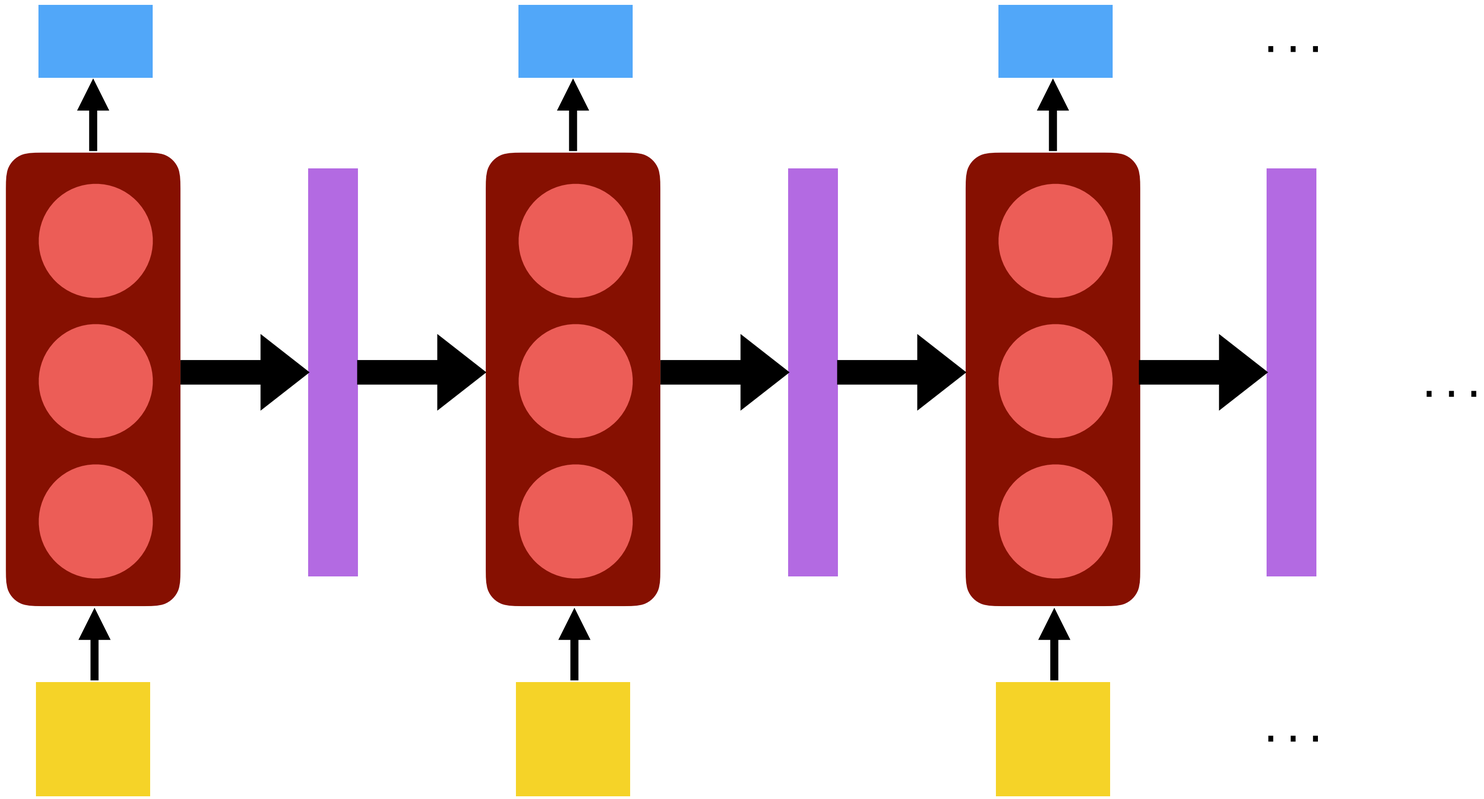
Recurrent neural networks

Share weights for each time step



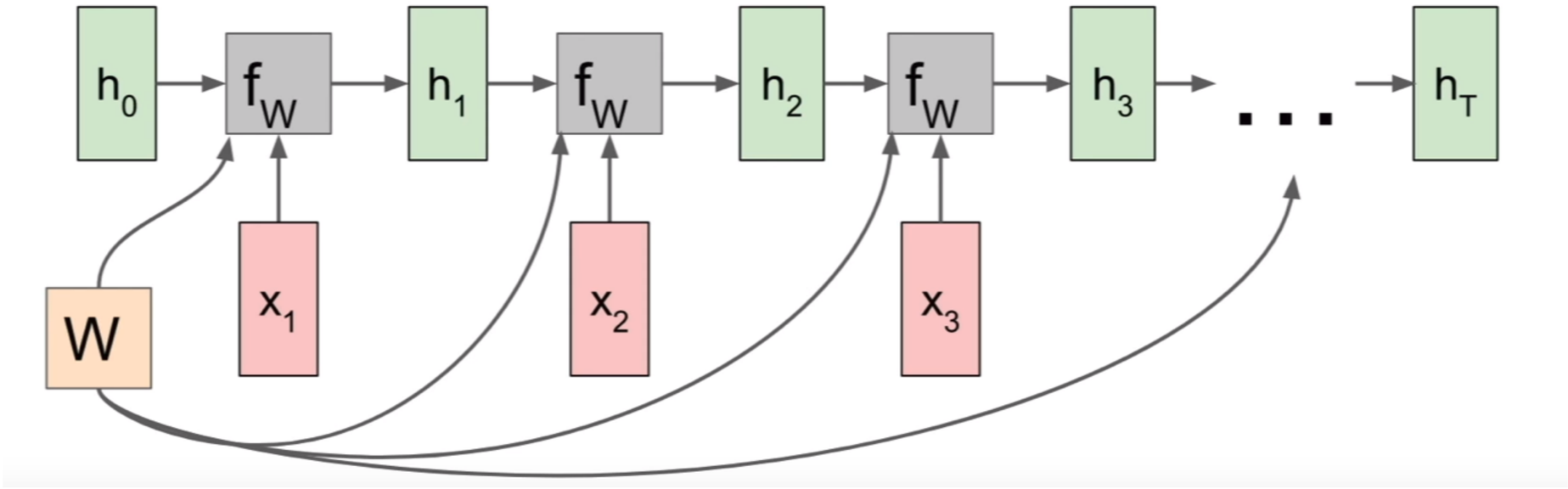
Recurrent neural networks

Pass information across time steps using **hidden states**



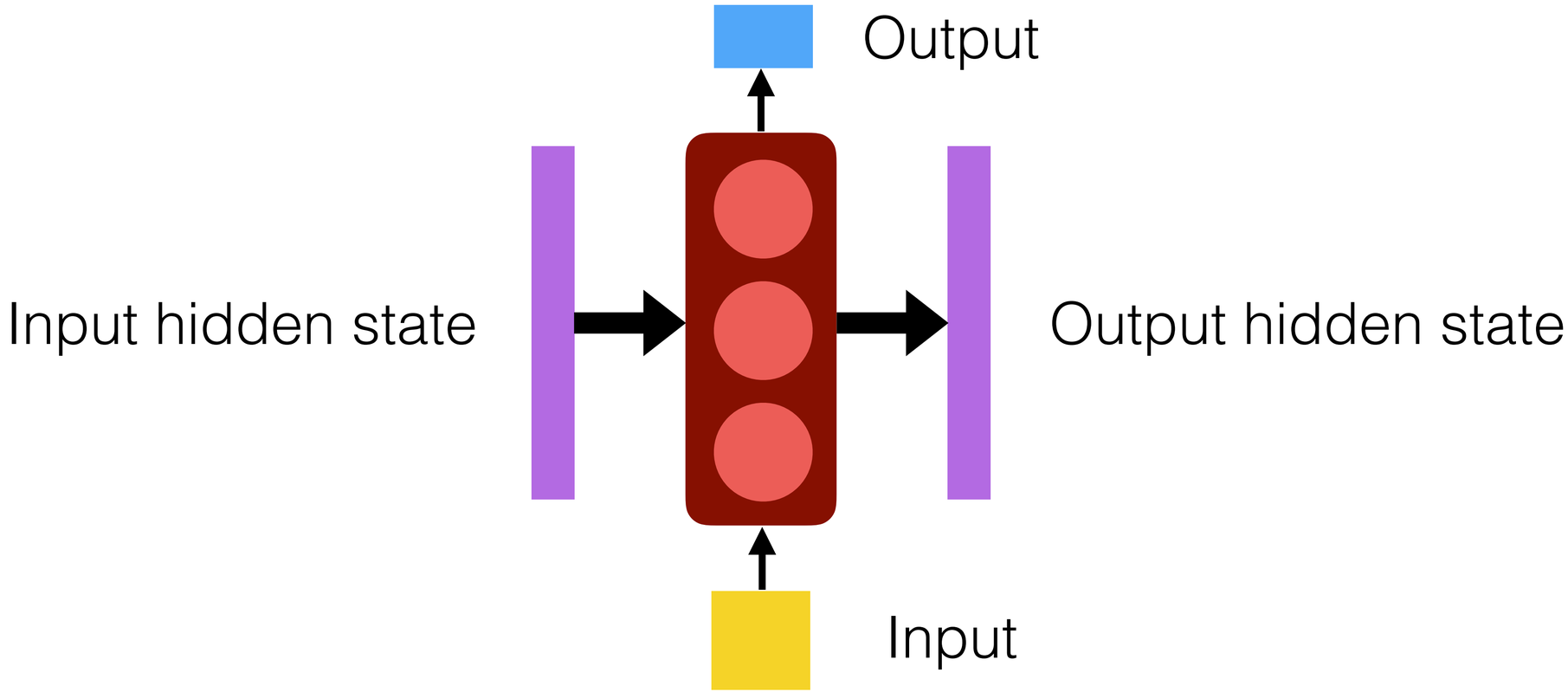
Recurrent neural networks

Computational graph view of things:

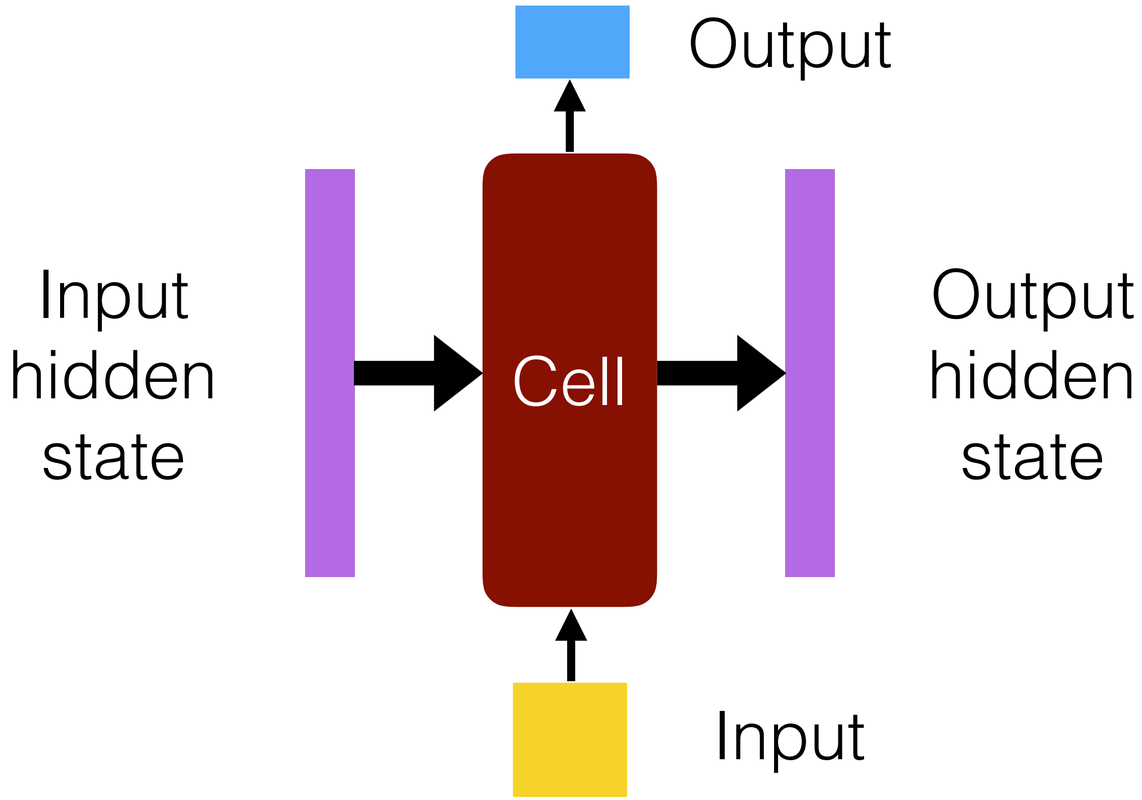


https://calvinfeng.gitbook.io/machine-learning-notebook/supervised-learning/recurrent-neural-network/recurrent_neural_networks

Hidden states



Types of cells

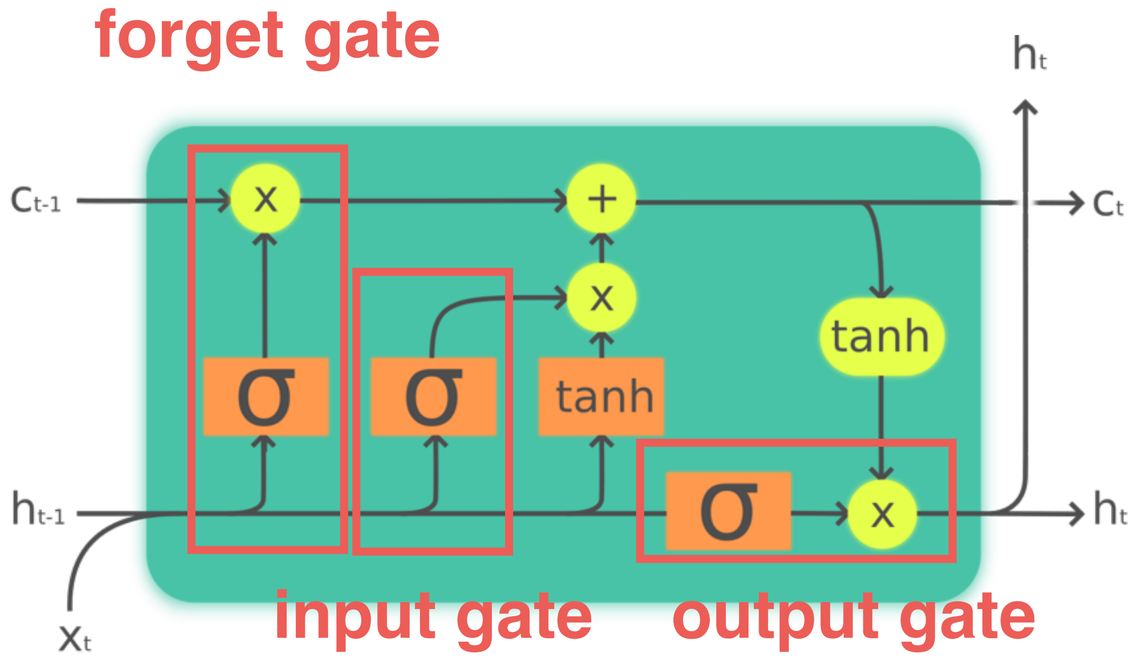


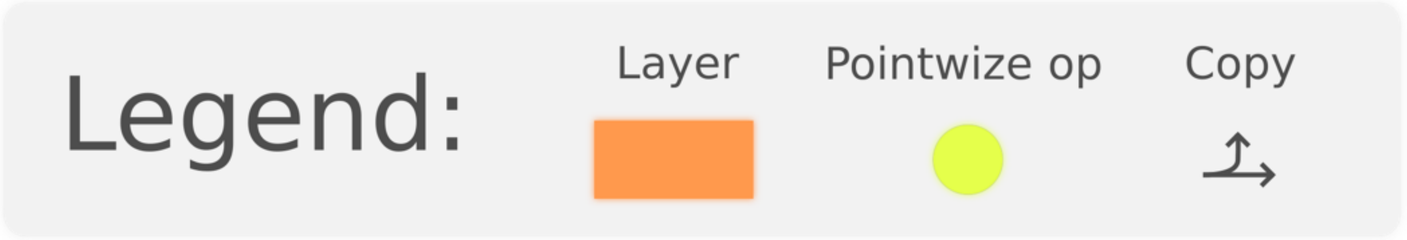
Many types of cells:

1. Vanilla RNN
2. **LSTM** - Long short term memory
3. **GRU** - Gated recurrent unit

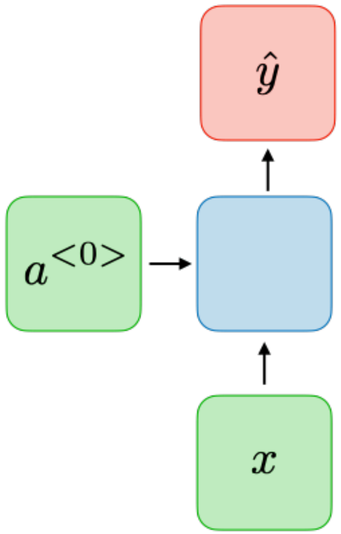
LSTM

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

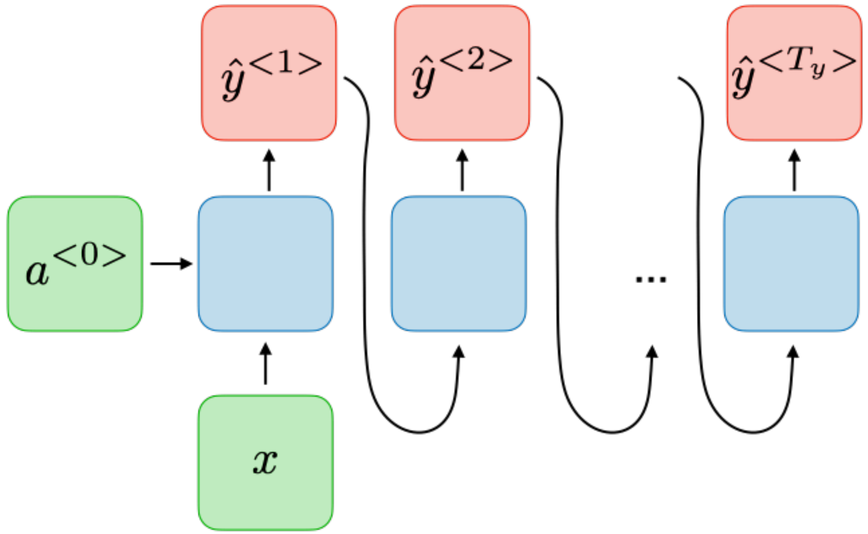




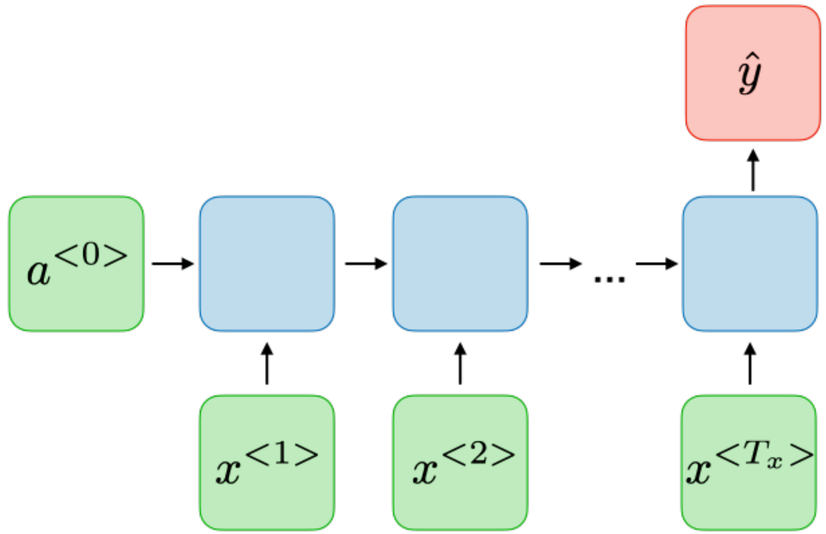
How many outputs?



One-to-one



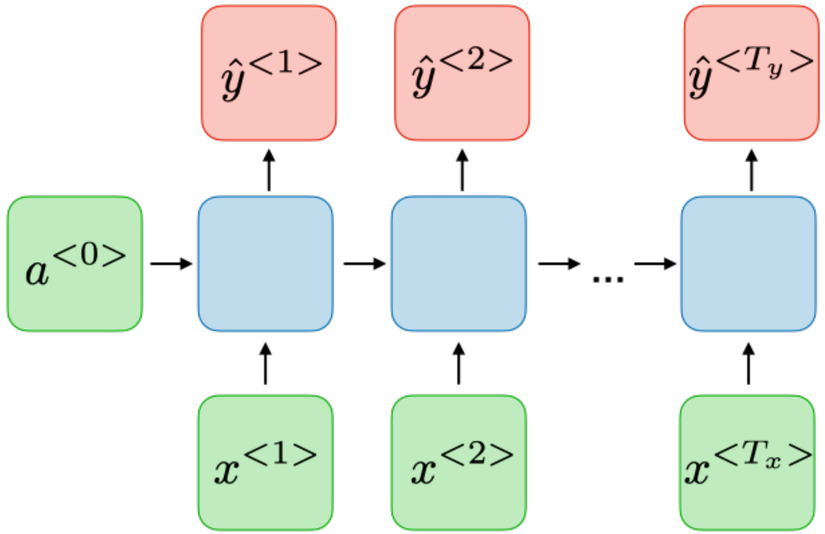
One-to-many



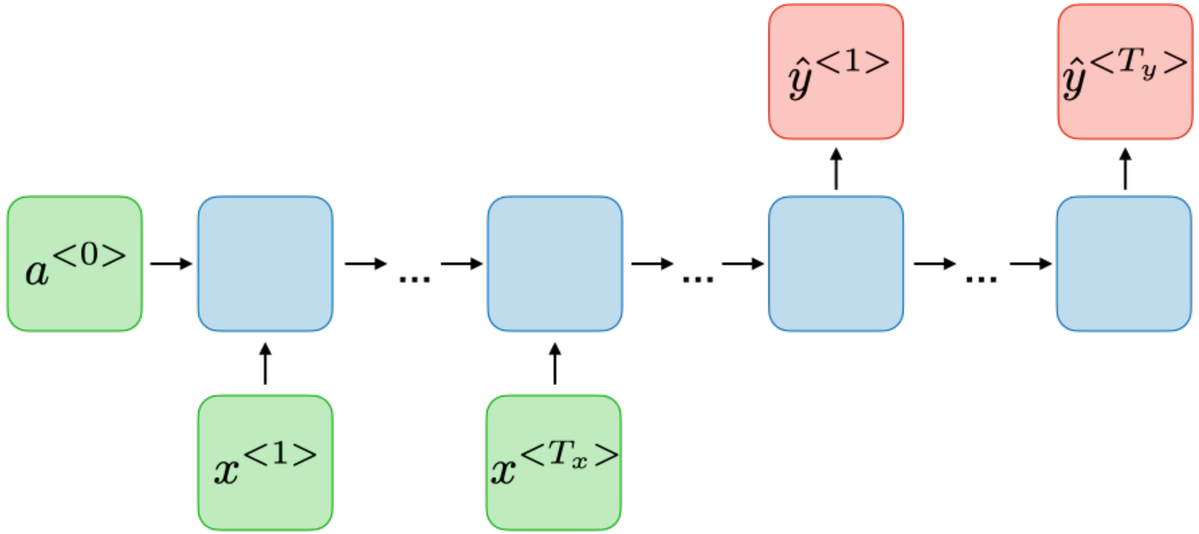
Many-to-one

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

How many outputs?



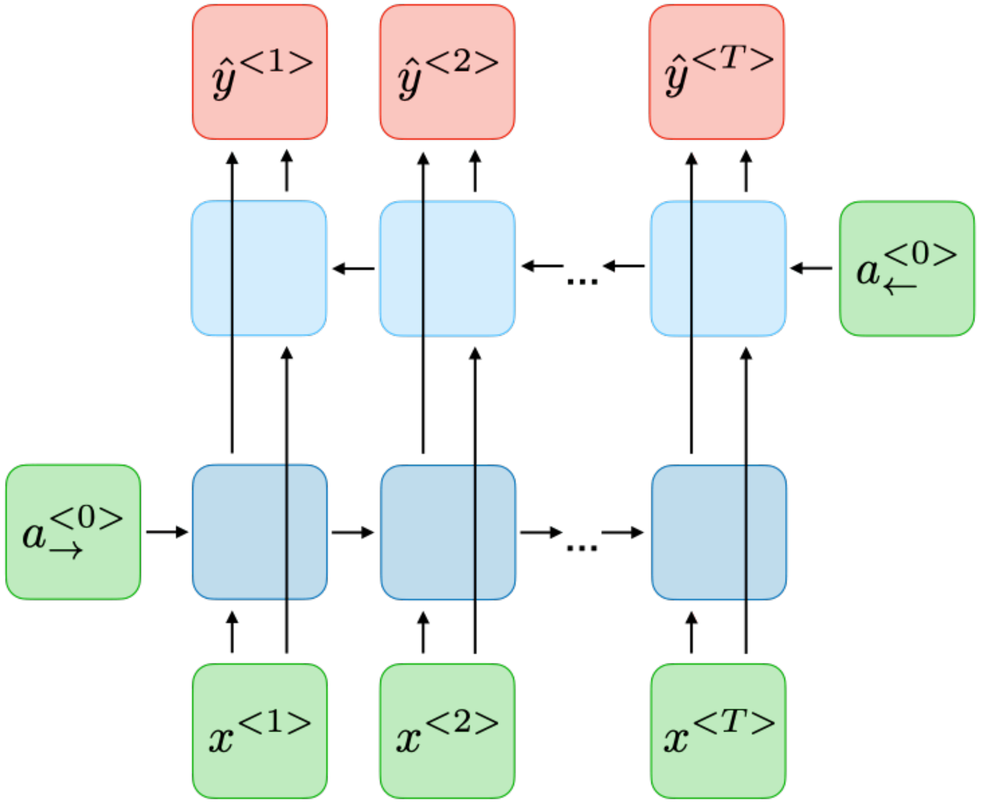
Many-to-many



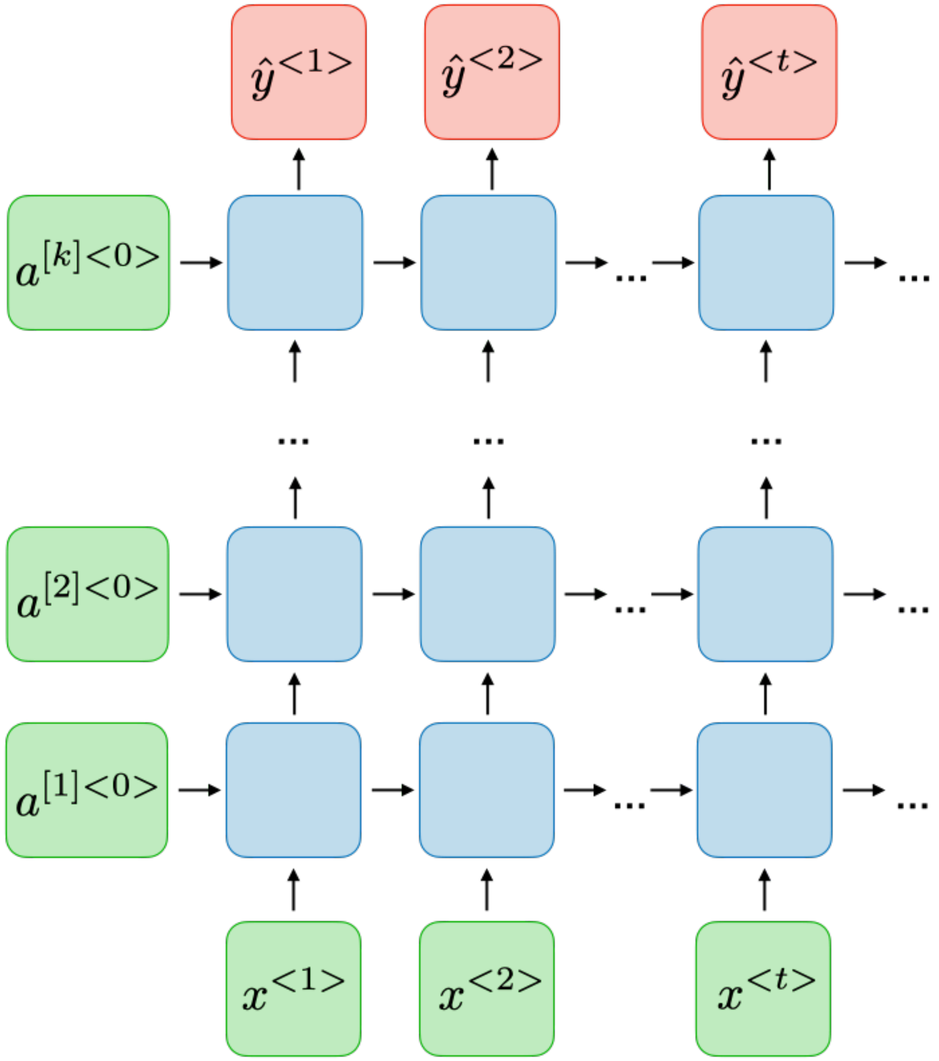
Many-to-many

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

RNN variants



Bidirectional



Deep

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recap: RNNs

RNNs allow for processing of variable-length inputs

Weights are shared across time steps; model size does not increase with size of input

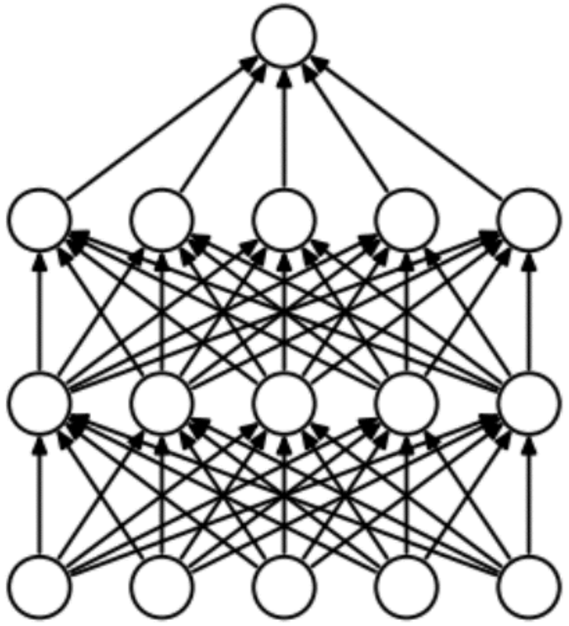
Computation is often slow, and the network “forgets” information from a long time ago

Other Architectures

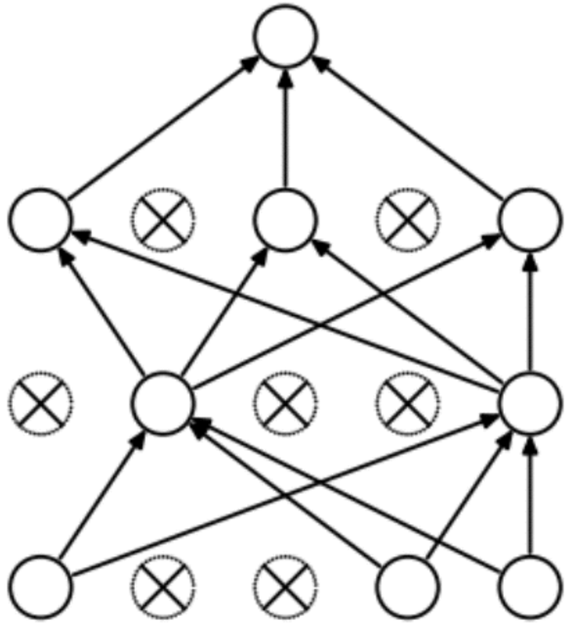
Dropout layers

Another way to regularize neural networks.

Randomly drop some fraction of neurons at a layer during training.



(a) Standard Neural Net

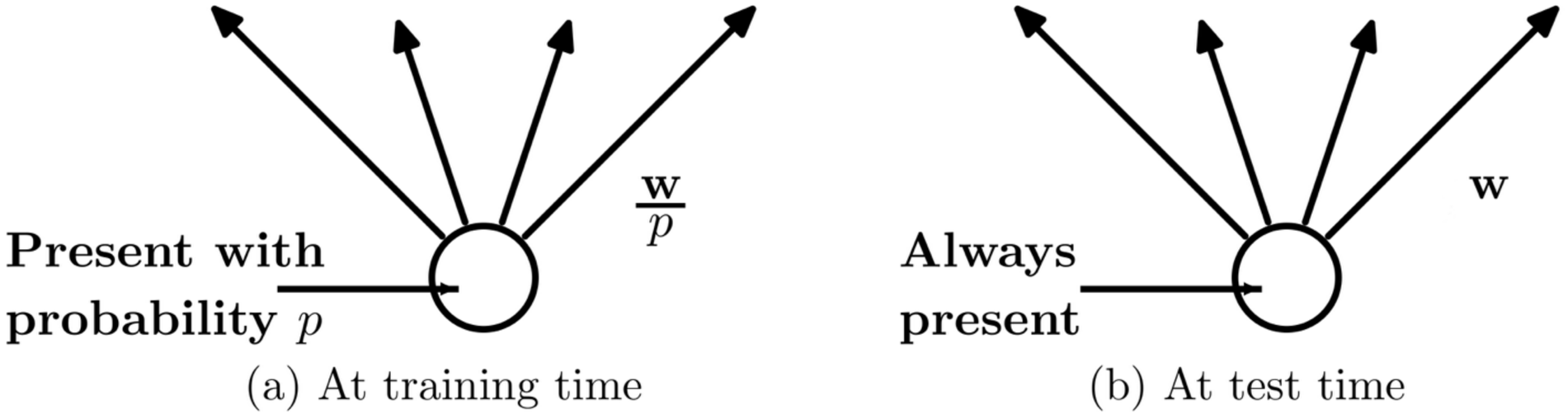


(b) After applying dropout.

Dropout layers

The output of non-dropped neurons is scaled by the inverse probability of keeping the neuron alive.

At test time, you do not use dropout.



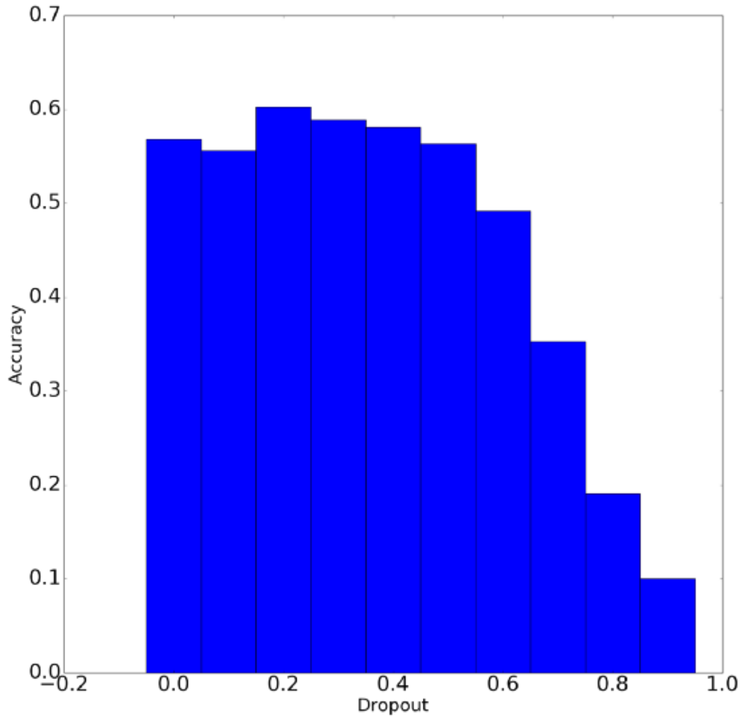
(a) At training time

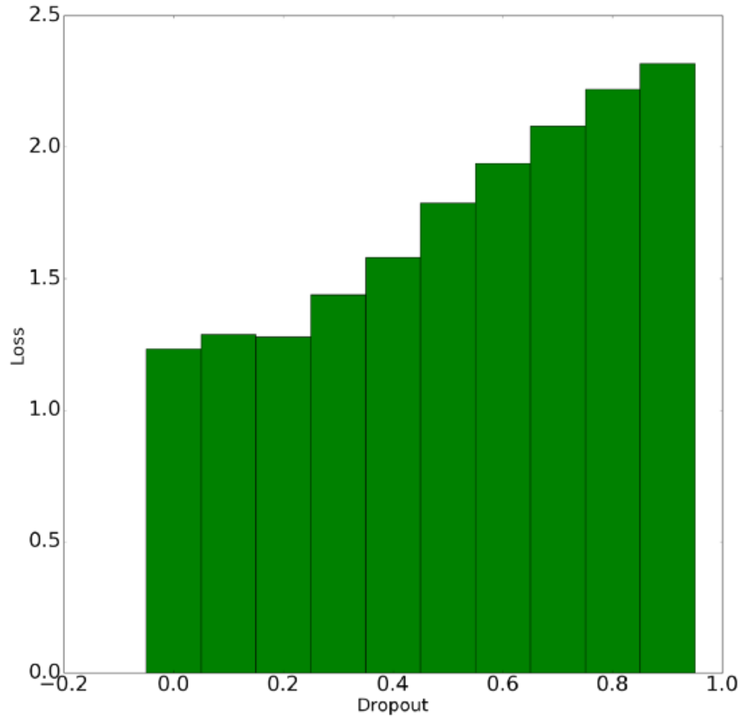
(b) At test time

Dropout layers

The point is to reduce dependence across neurons

Empirical results:



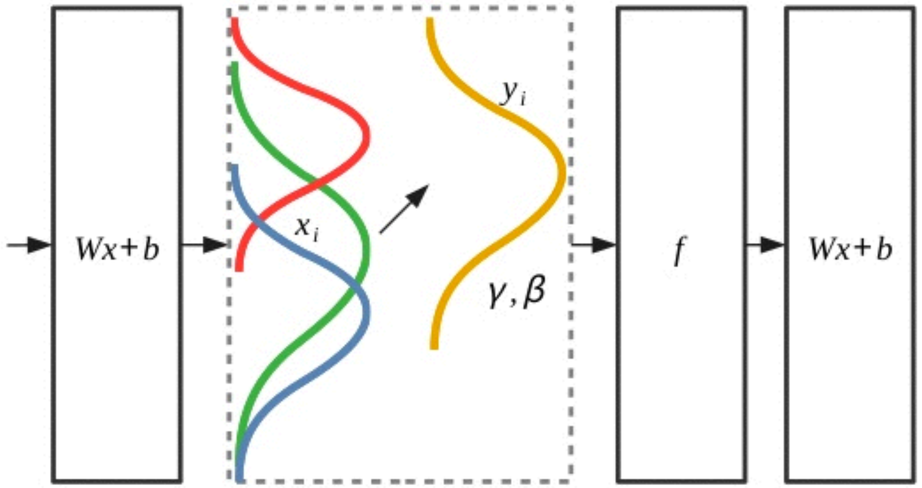


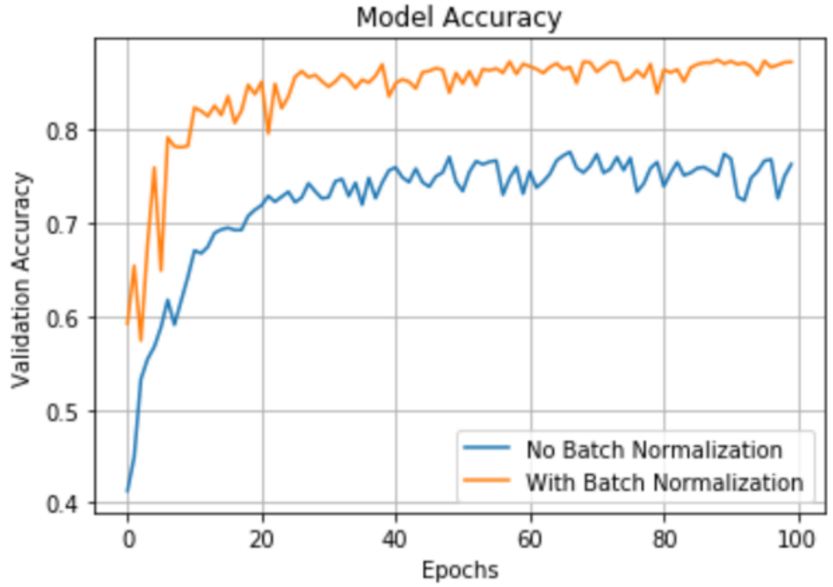
https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

Batch normalization

Goal is to control the mean and variance of each layer's output.

1. Normalize output of each neuron to be mean zero and unit variance across the batch at every training step.
2. Rescale the output of the neuron to be some specific mean and variance, both of which are trainable parameters.



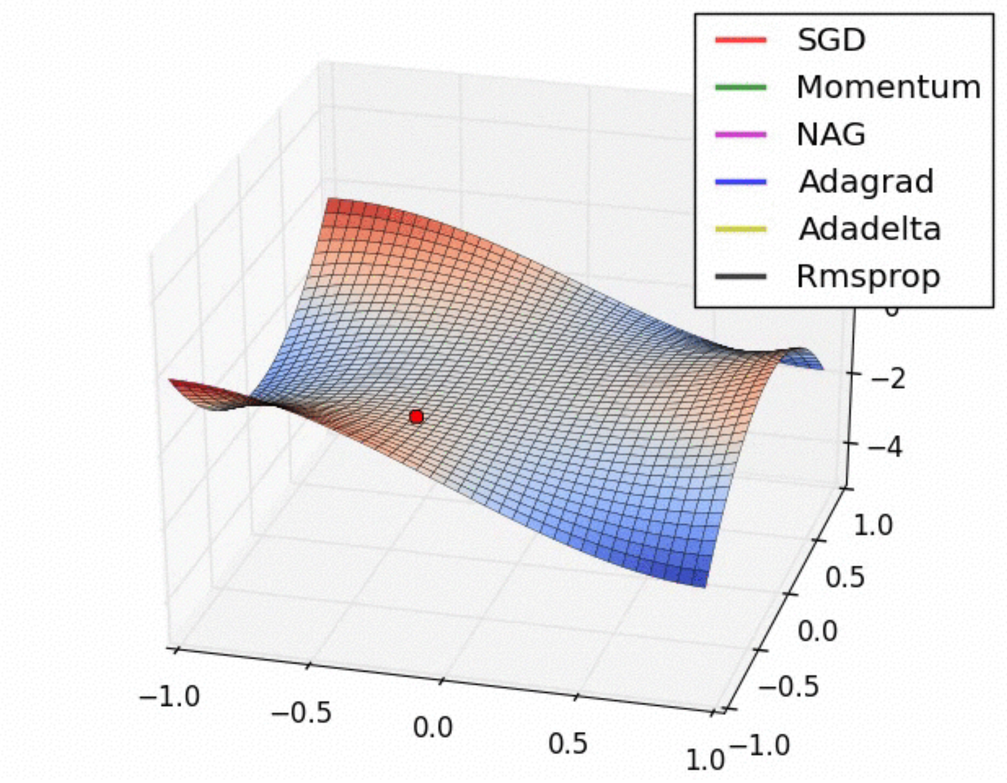


Other optimizers

Adaptive learning rate: adjust learning rate based on mean and variance of gradients

Momentum: instead of current gradient, take weighted average of past gradients to update the weights

Techniques: RMSProp, Adagrad, Adam, etc.

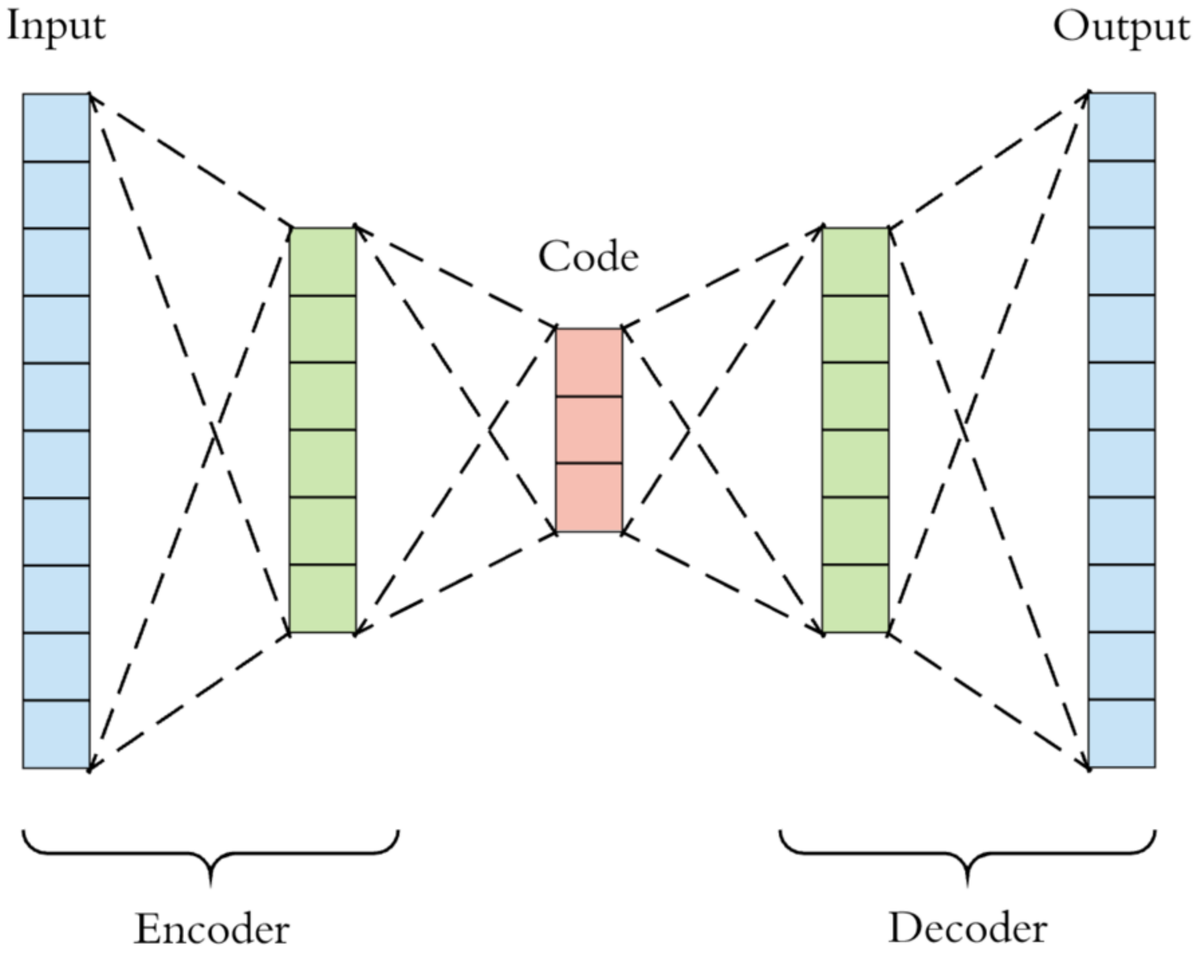


Autoencoders

A way to do unsupervised deep learning

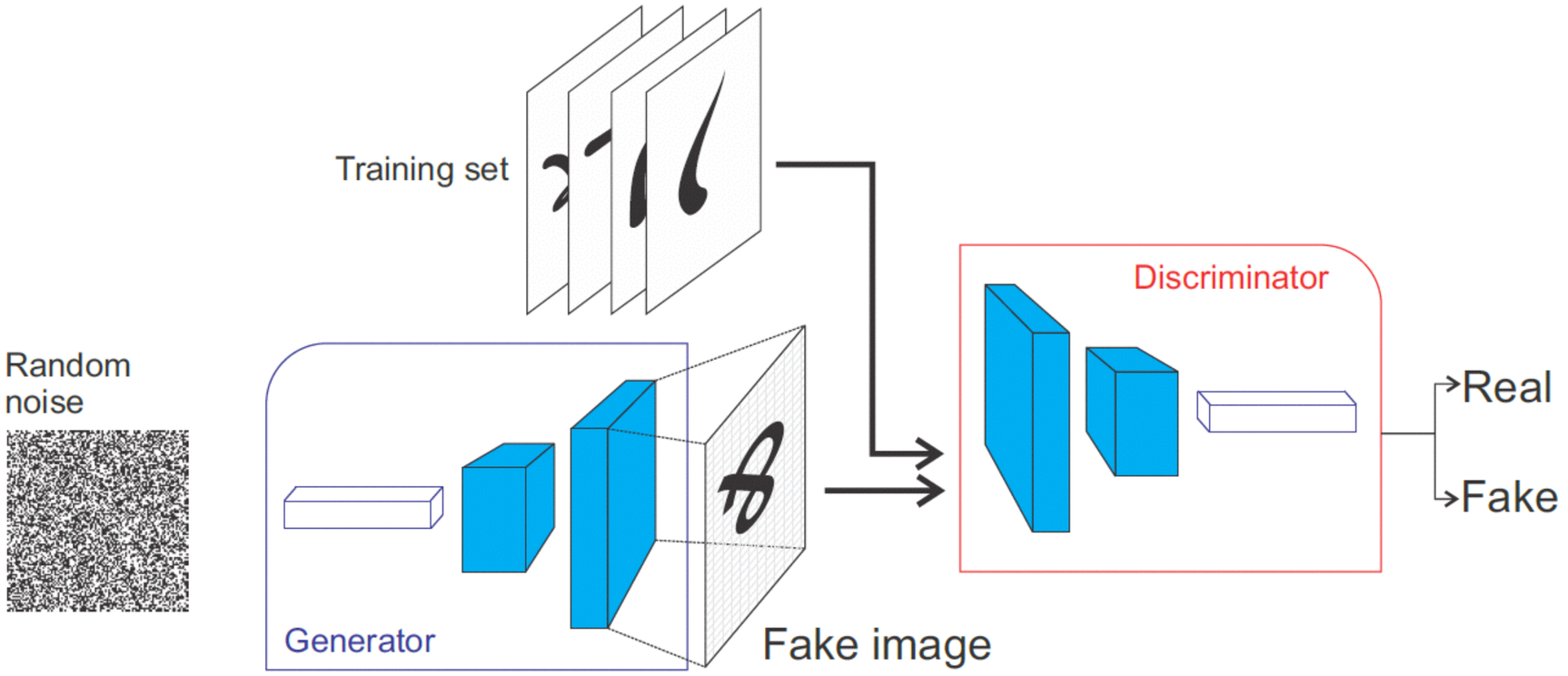
Input and output are the same: the network is tasked with recreating the input but going through a lower-dimensional bottleneck

Bottleneck layer is interpreted as a latent representation for the input



https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

Generative adversarial networks



https://www.oreilly.com/library/view/java-deep-learning/9781788997454/60579068-af4b-4bbf-83f1-e988fbe3b226.xhtml

Generative adversarial networks

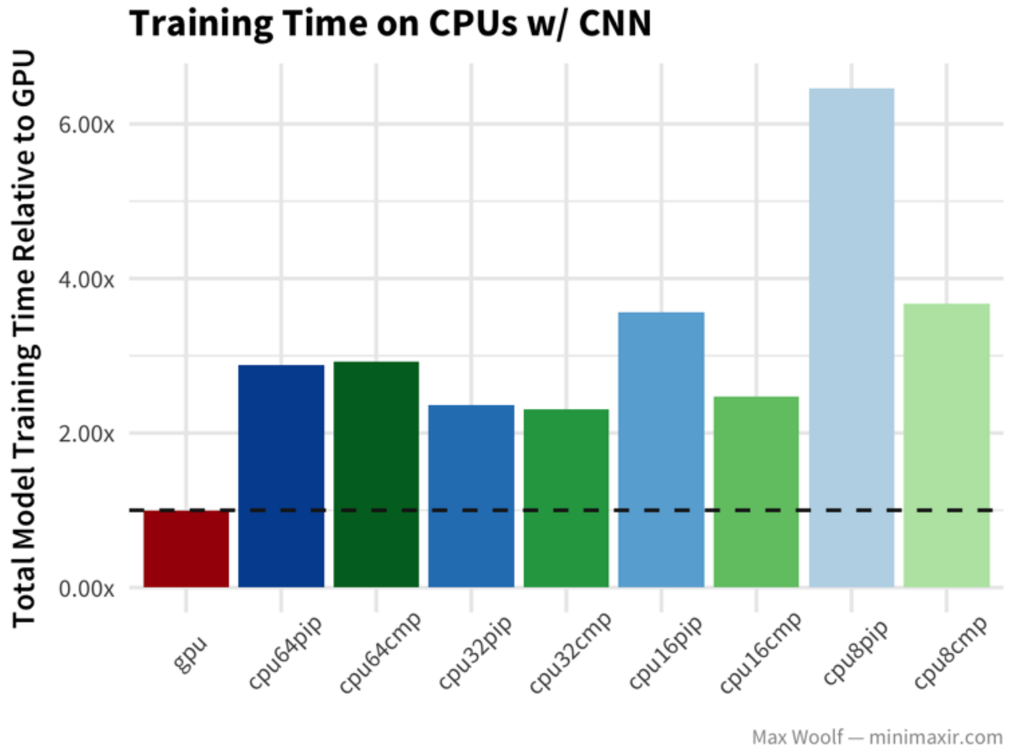
Deep Learning Libraries

What do deep learning packages provide?

1. Ability to run on GPUs
2. Build computation graphs and provide automatic differentiation
3. Useful medium- to high-level components for deep learning already implemented (convolutions, batch norm, VGG, ResNet, etc.)

CPUs vs GPUs

GPUs are optimized to perform matrix multiplications, and speed up neural network training many-fold



Deep learning libraries

1. TensorFlow — mainly used in industry
2. Keras — easy-to-use API for TensorFlow
3. PyTorch — mainly used in research
4. Others include Caffe, Theano, MXNET…

Coding Session