Classification

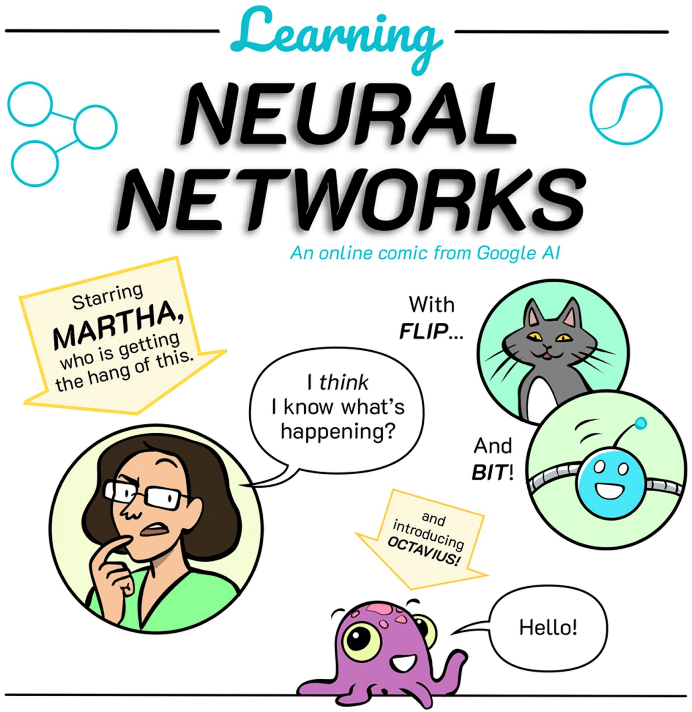
Prototyping with Deep Learning

Learning outcomes

After this lesson you will be able to:

- Understand binary and multi-class classification
- Identify appropriate evaluation metrics for classification tasks
- Recognize key building blocks in DL models
- Know popular network architectures for classification

Recap: DL preliminaries



https://cloud.google.com/products/ai/ml-comic-2

What is classification?

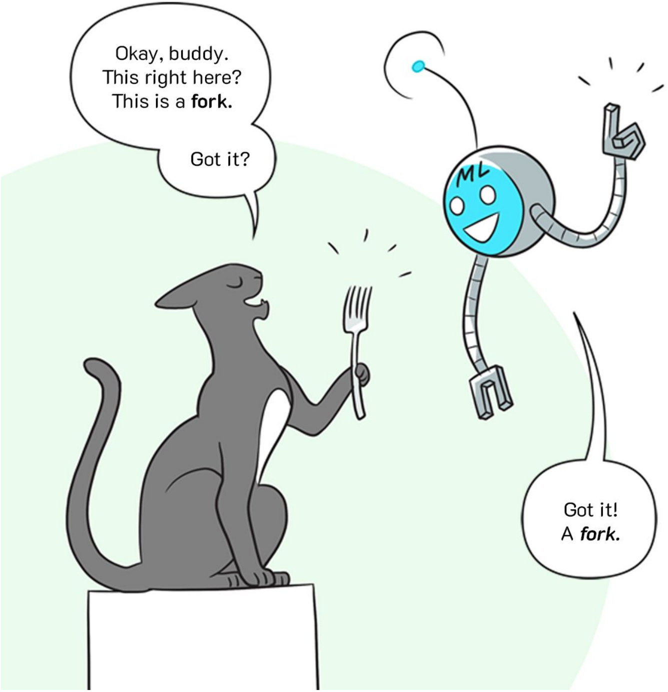
Predict a **discrete** value associated with a feature vector

Examples:

f(image) = cat

f(email) = spam

…



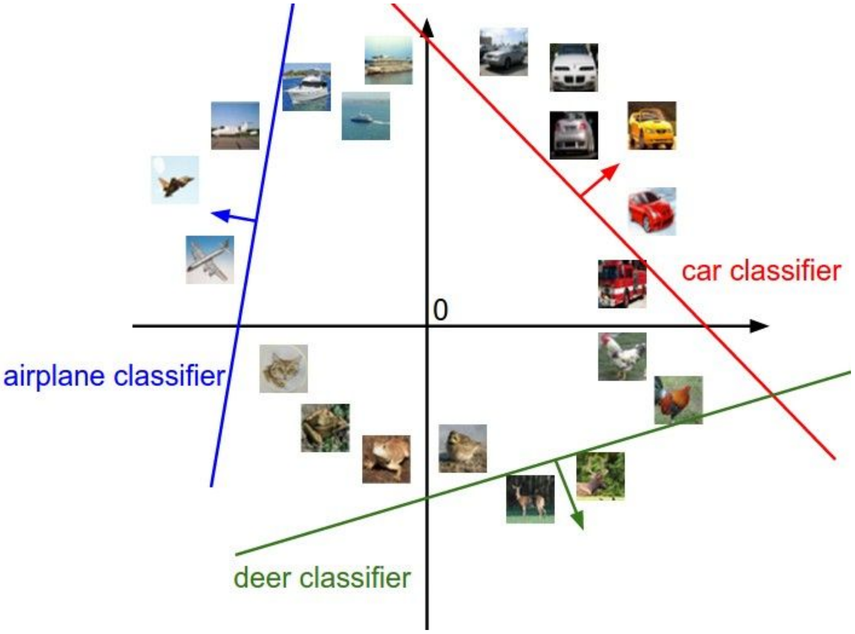
https://cloud.google.com/products/ai/ml-comic-1/

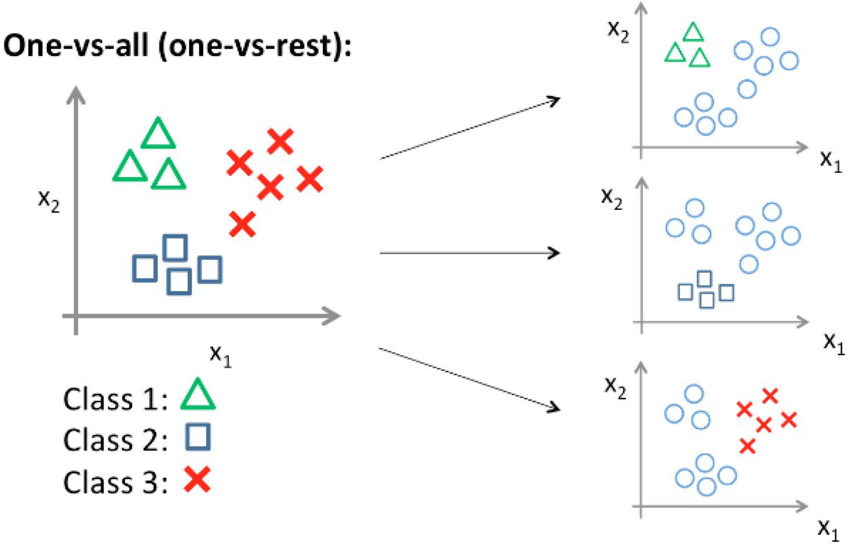
Use case: Not hotdog app

https://www.youtube.com/watch?v=vIci3C4JkL0

Linear classification

$$y = f(z) = \mathbf{w}^T \mathbf{x}$$



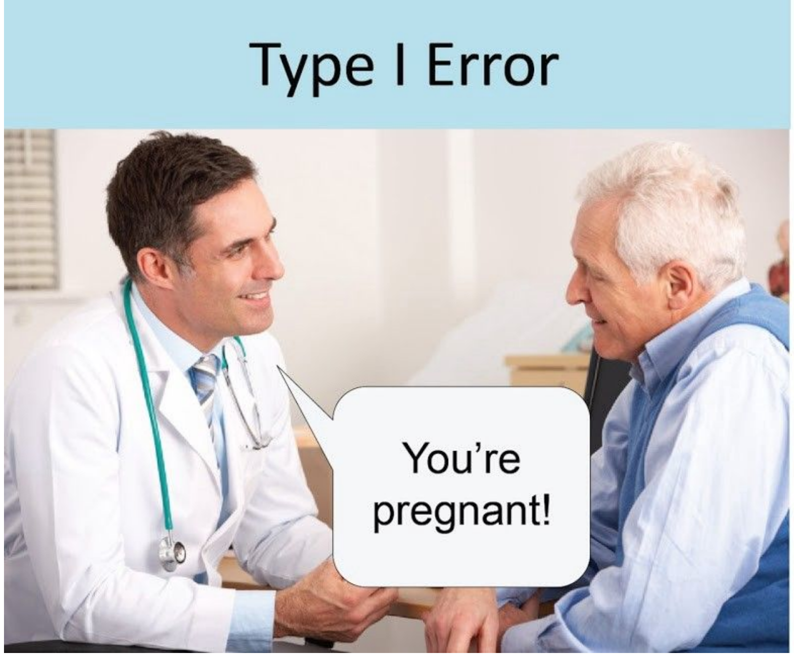


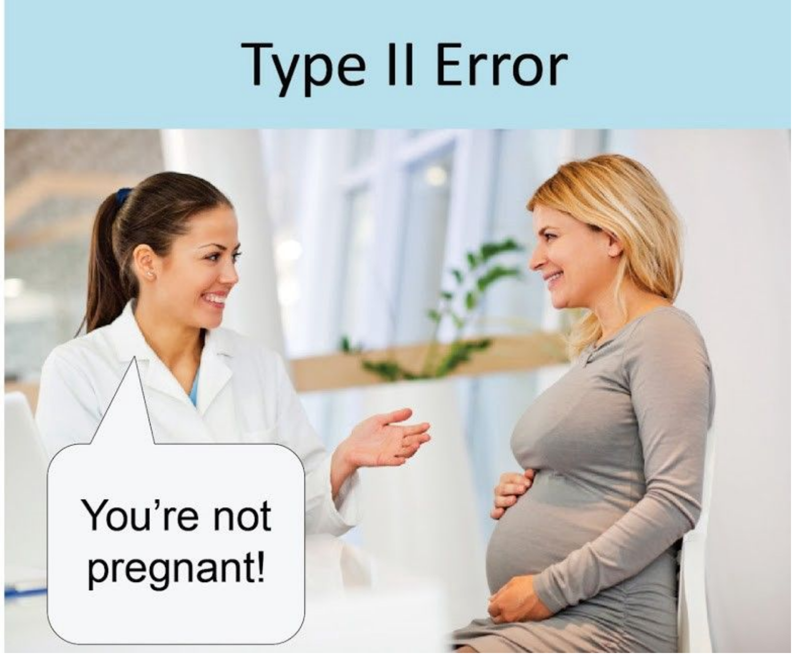
Confusion matrix

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	**Sensitivity** $\frac{TP}{(TP+FN)}$
Actual Class	Negative	False Positive (FP) Type I Error	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html

Types of error





Classification accuracy

ACC = All trues / All cases = (TP + TN) / (TP + FP + FN + TN)

Very sensible to *imbalanced* data:

- Consider e.g. dataset with 95 negative + 5 positive cases

Accuracy is not enough

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

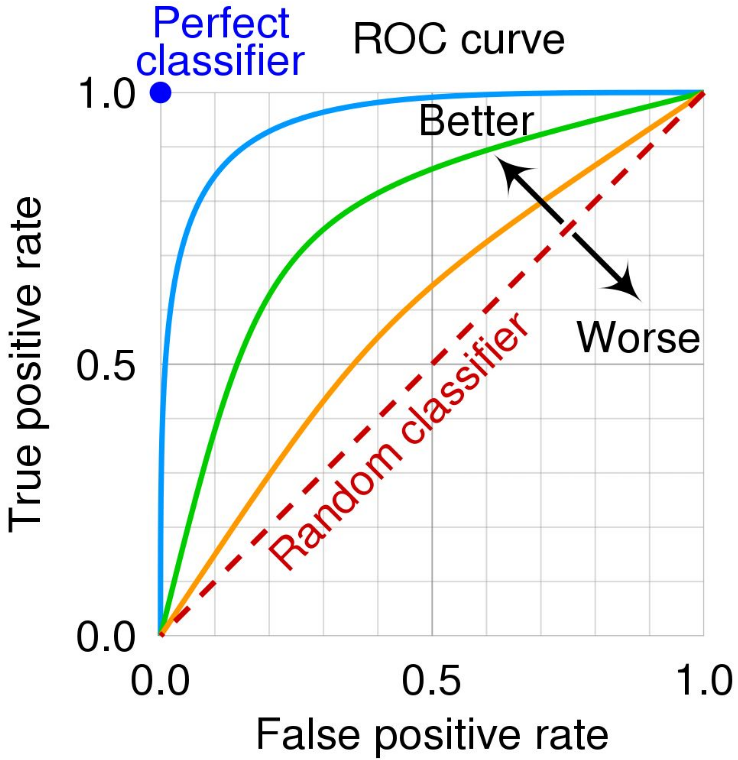
F-measure = 2 · Precision · Recall / (Precision + Recall)

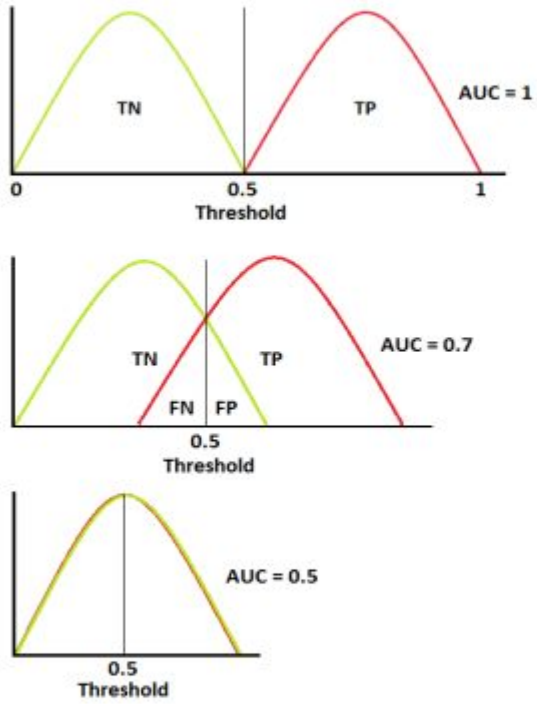
AUC: TPR vs FPR

- Sensitivity (TPR) = Recall
- FPR = 1 - Specificity
- Specificity (TNR) = TN / (TN + FP)

… and many more

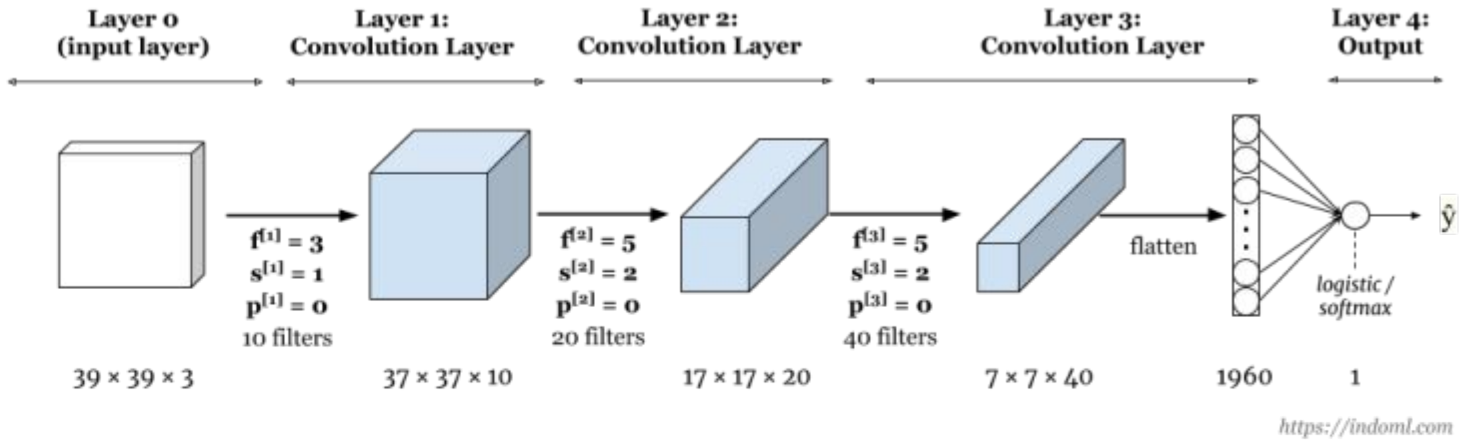
ROC and AUC





https://towardsdatascience.com/68b2303cc9c5

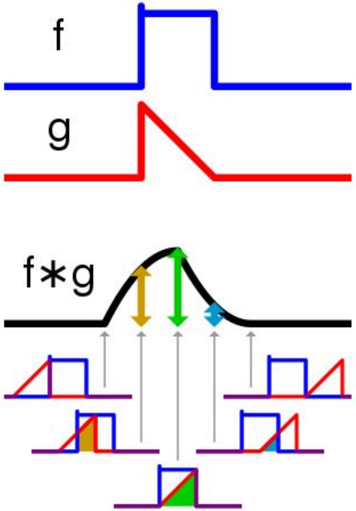
Convolutional Neural Net (CNN)



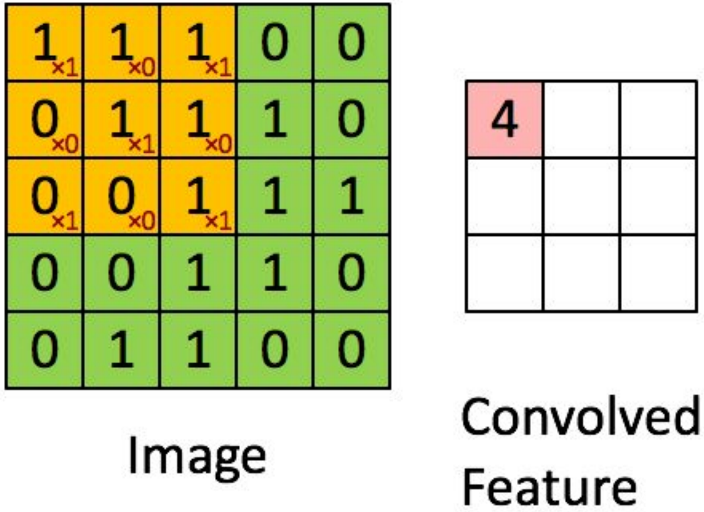
https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Convolution operation

$$(f * g)(t) = \int_0^t f(\tau)g(t-\tau)\,d\tau$$

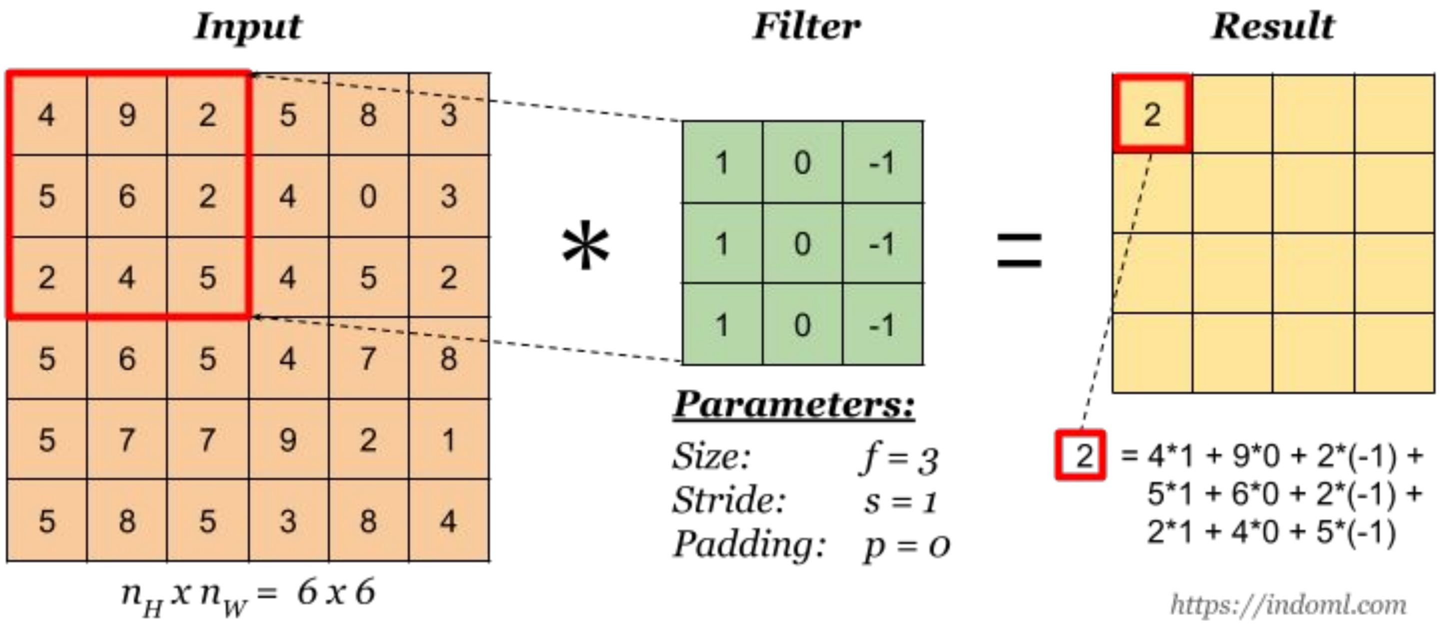


$$G[m,n] = (f * g)[m,n] = \sum_j \sum_k g[j,k] f[m-j, n-k]$$

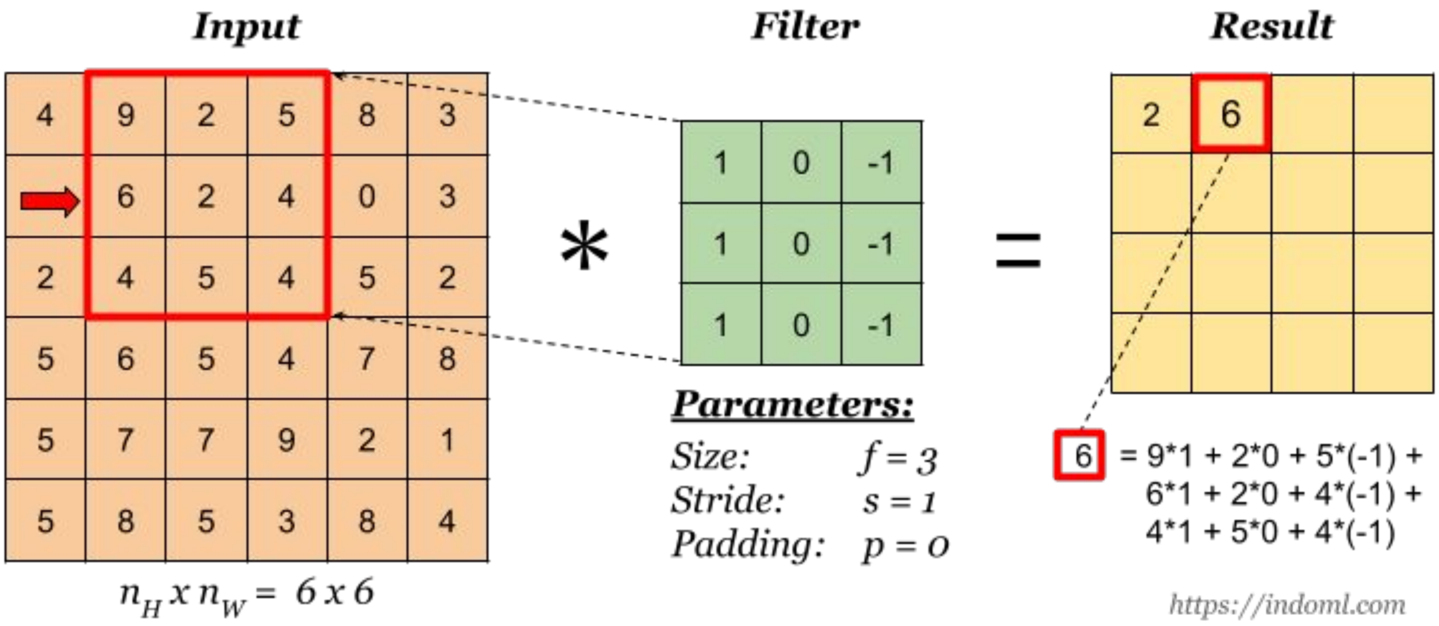


Demo at https://setosa.io/ev/image-kernels/

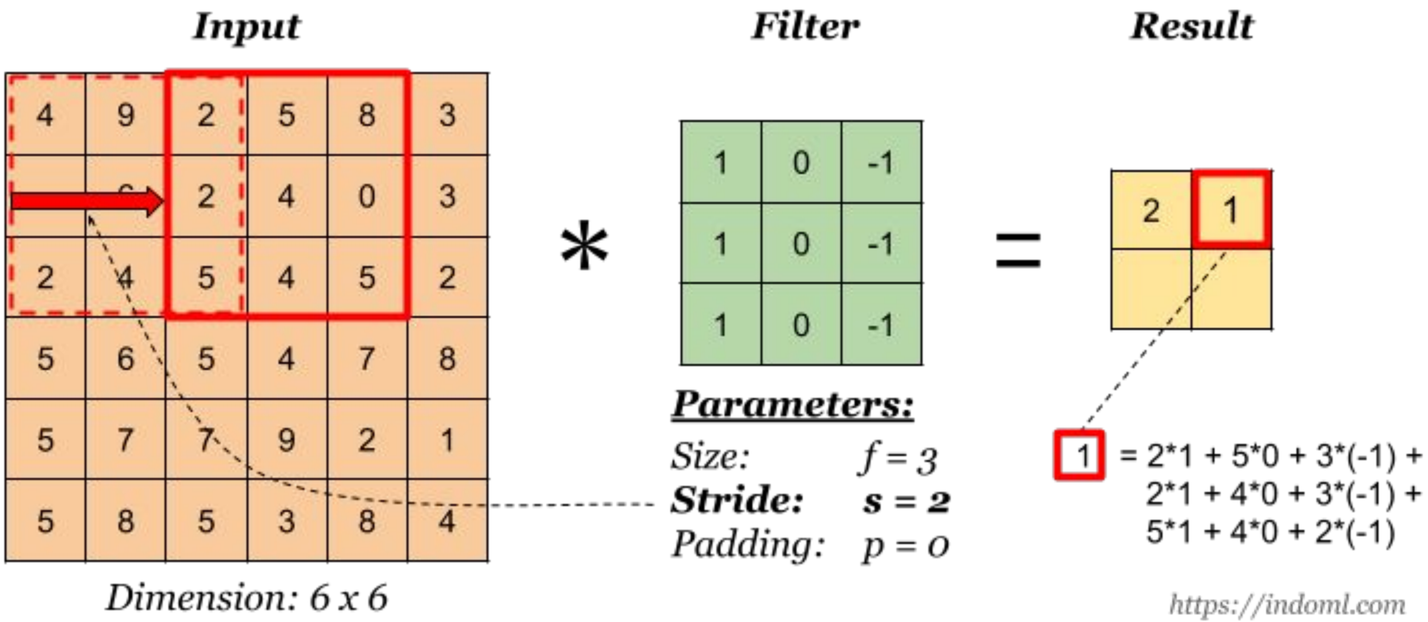
Convolution operation: filters



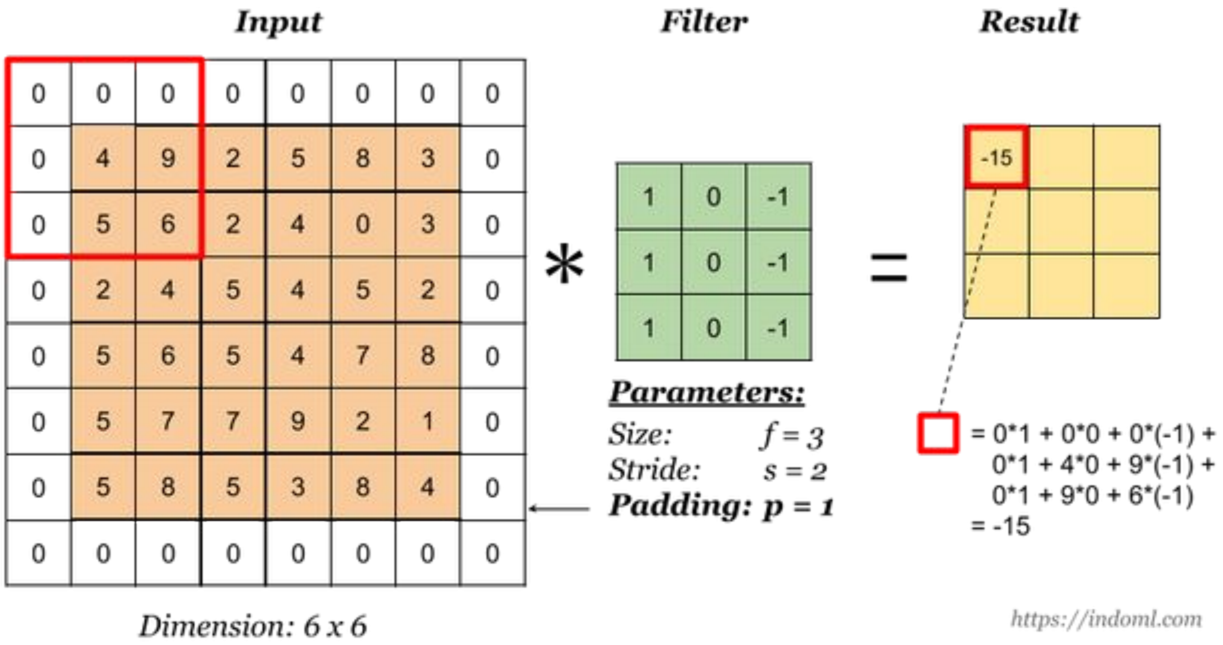
Convolution operation: filters



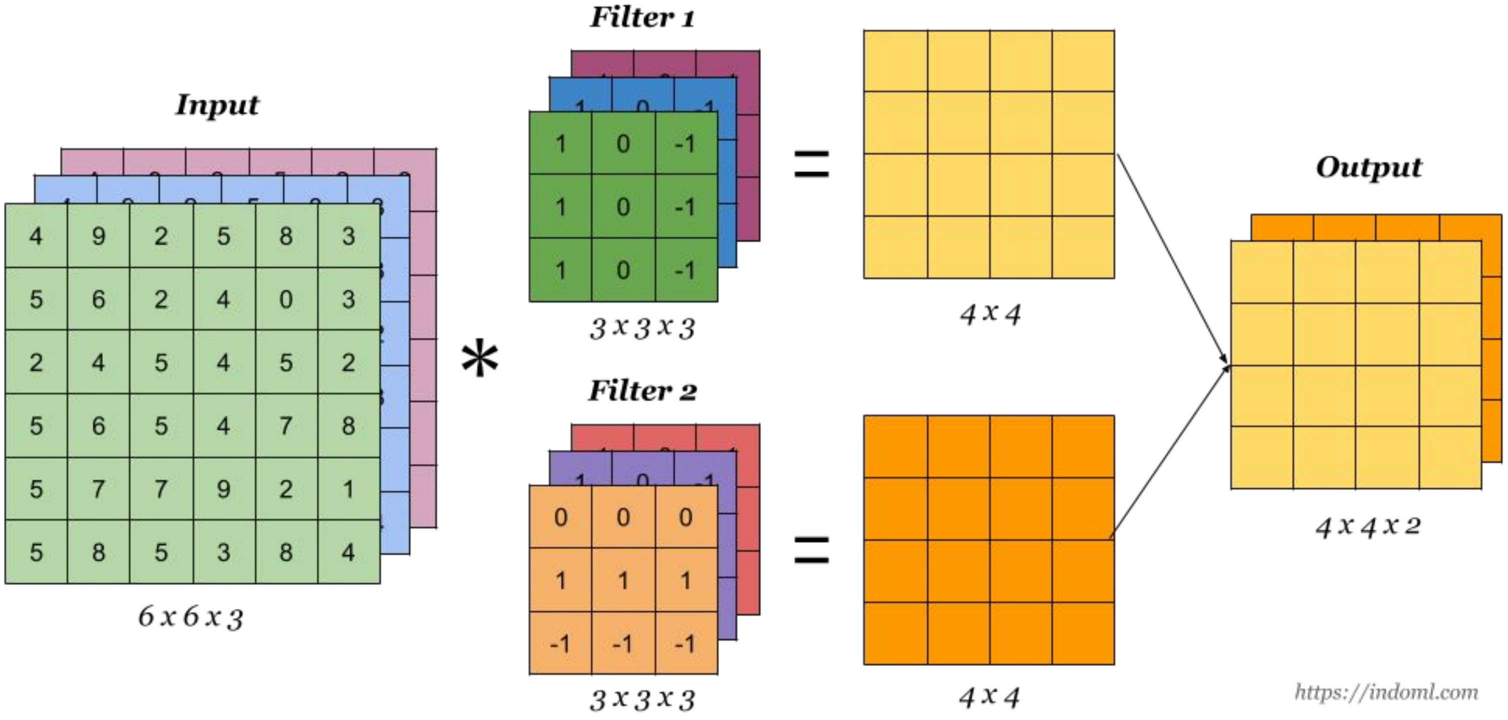
Convolution operation: stride



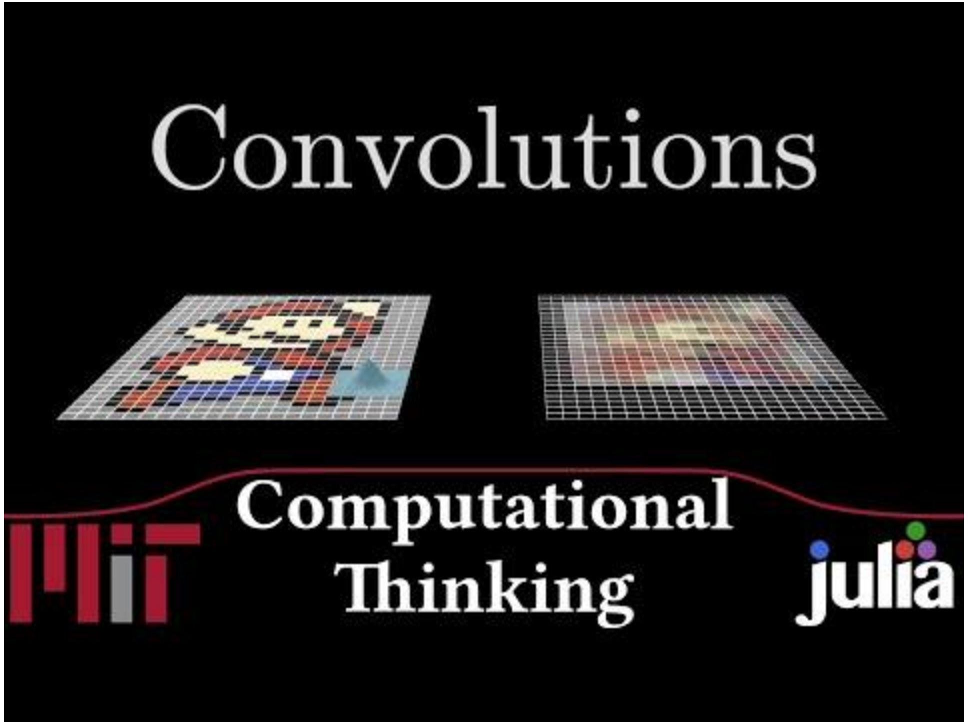
Convolution operation: padding



Convolution operation: image channels

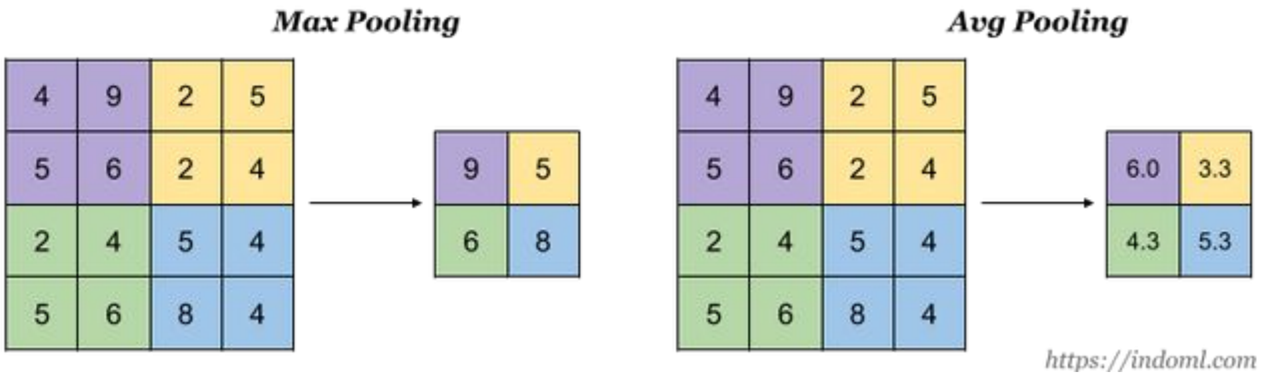


Convolution operation lecture



https://www.youtube.com/watch?v=8rrHTtUzyZA

Pooling operation



Classic CNN architectures

LeNet (1998)

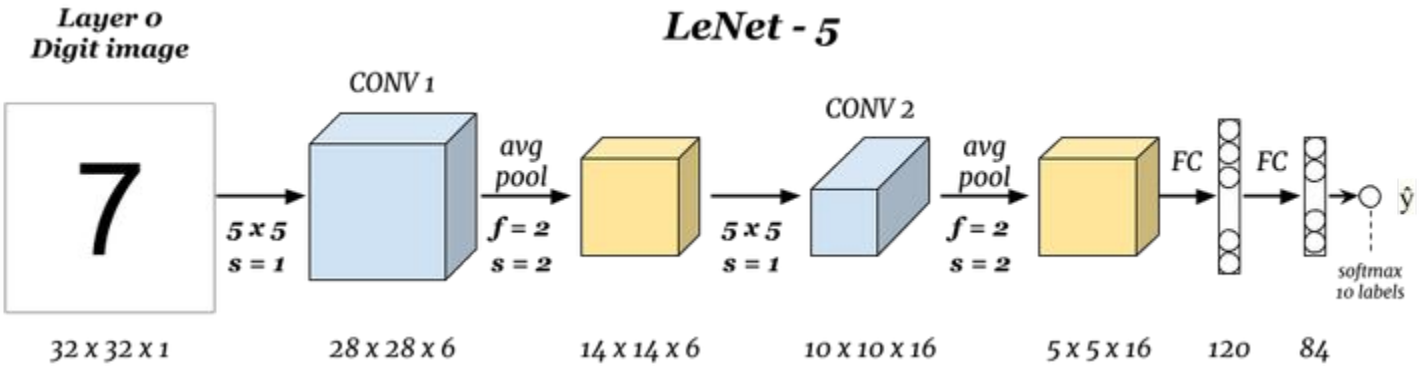
AlexNet (2012)

GoogLeNet (2014)

VGGNet (2015)

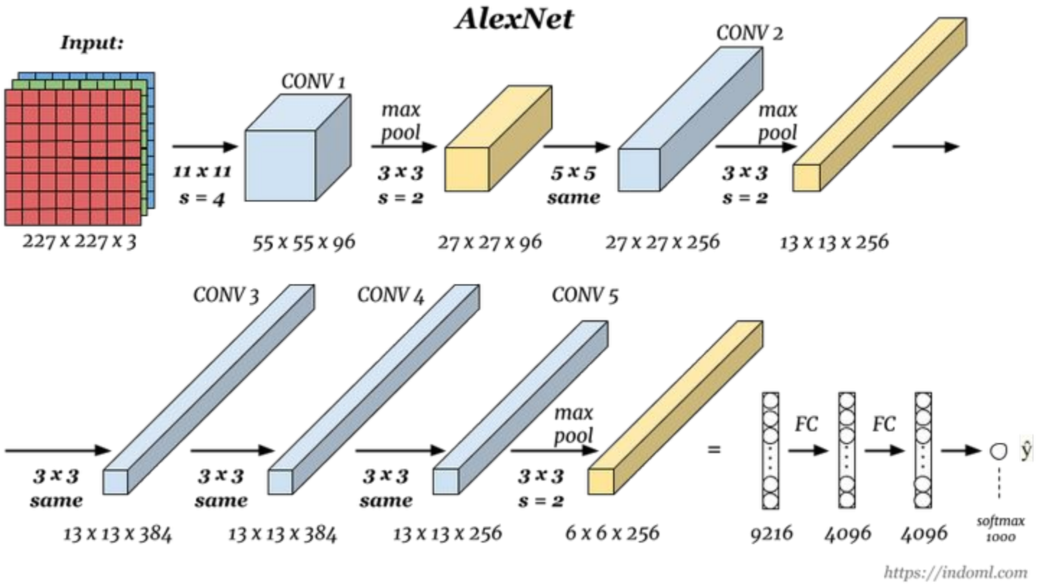
ResNet (2015)

Classic CNN: **LeNet**



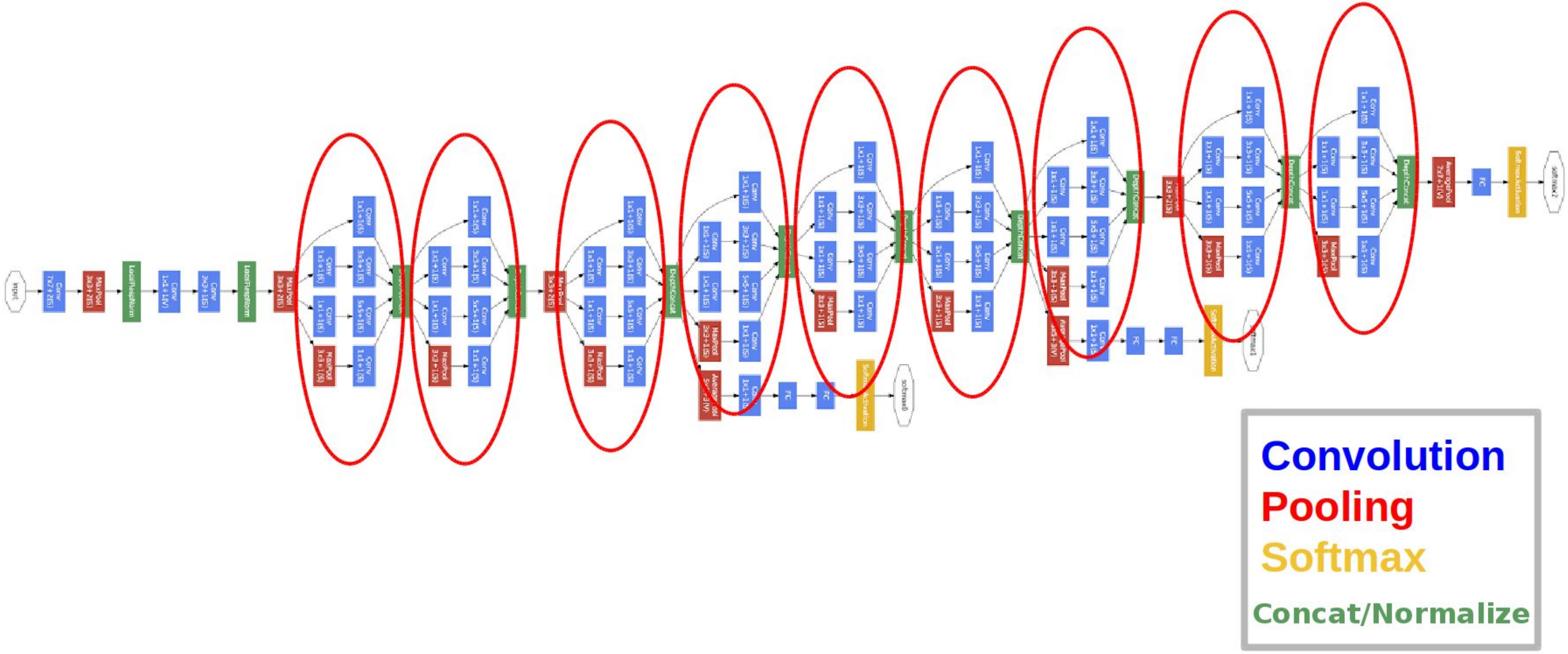
https://towardsdatascience.com/a2d531ebc342

Classic CNN: **AlexNet**



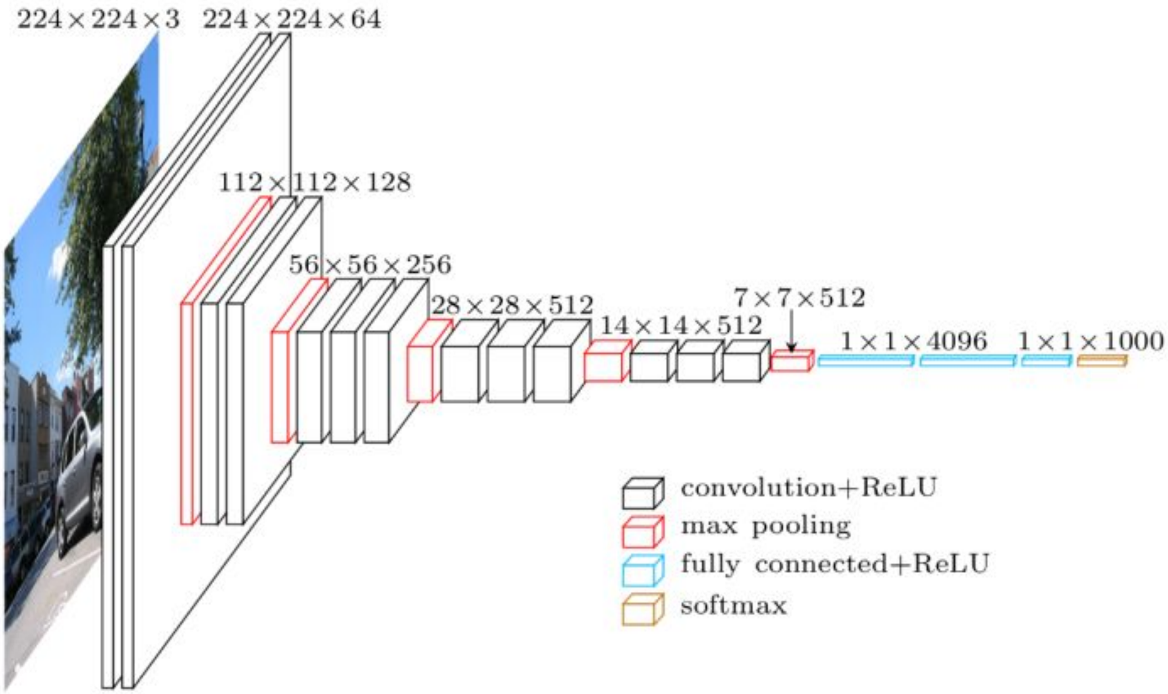
https://www.learnopencv.com/understanding-alexnet/

Classic CNN: **GoogLeNet**



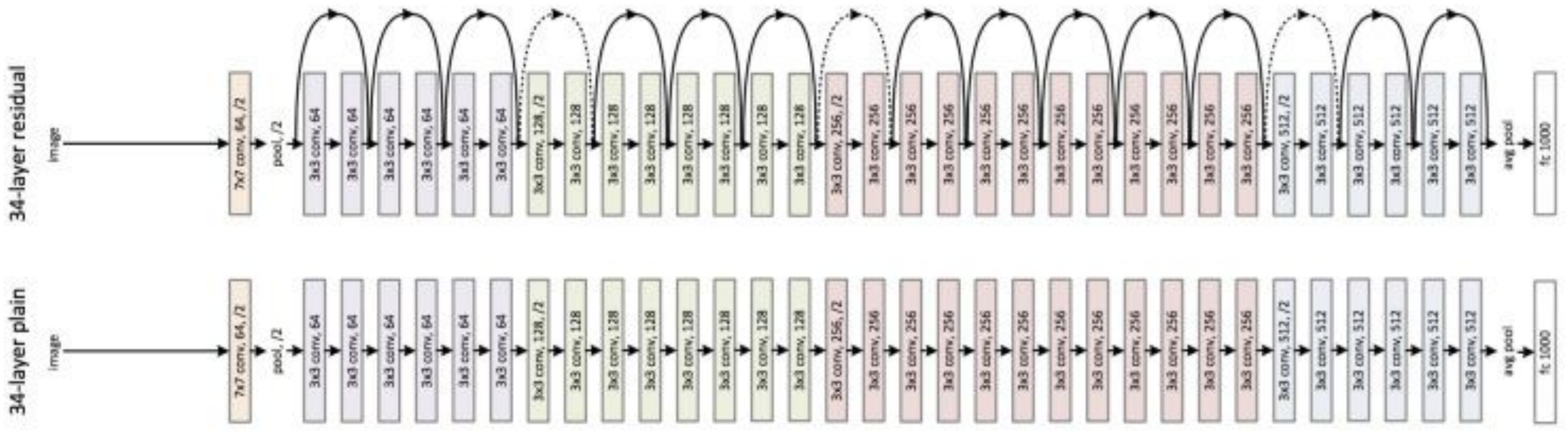
https://medium.com/coinmonks/c2b3565a64e7

Classic CNN: **VGGNet**



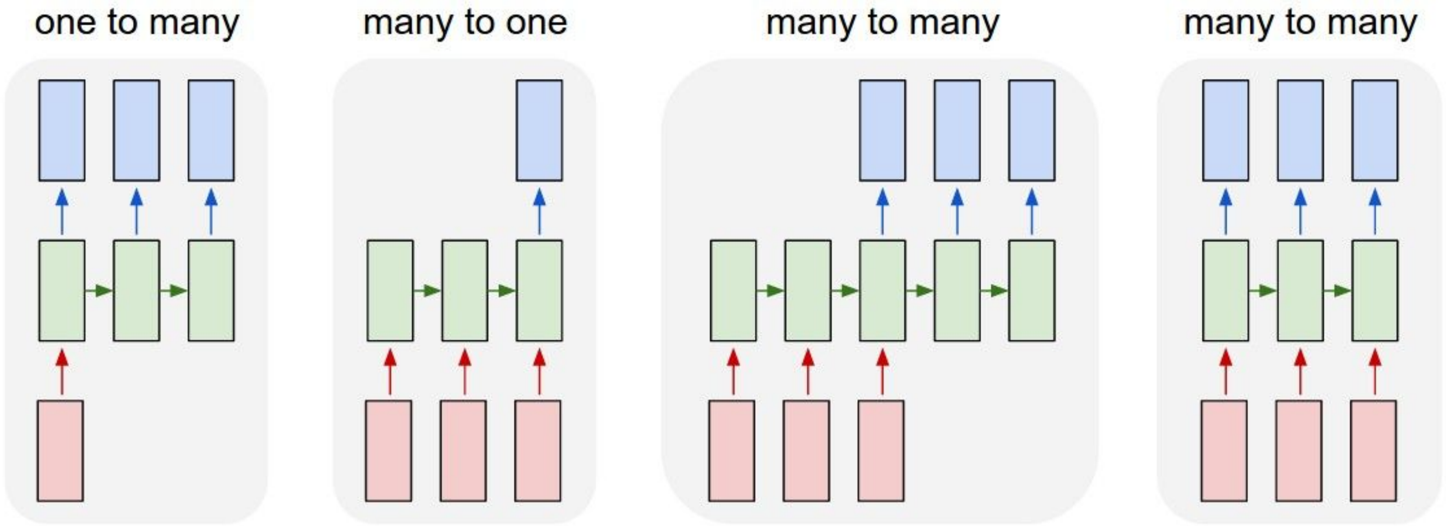
https://medium.com/coinmonks/d02355543a11

Classic CNN: **ResNet**



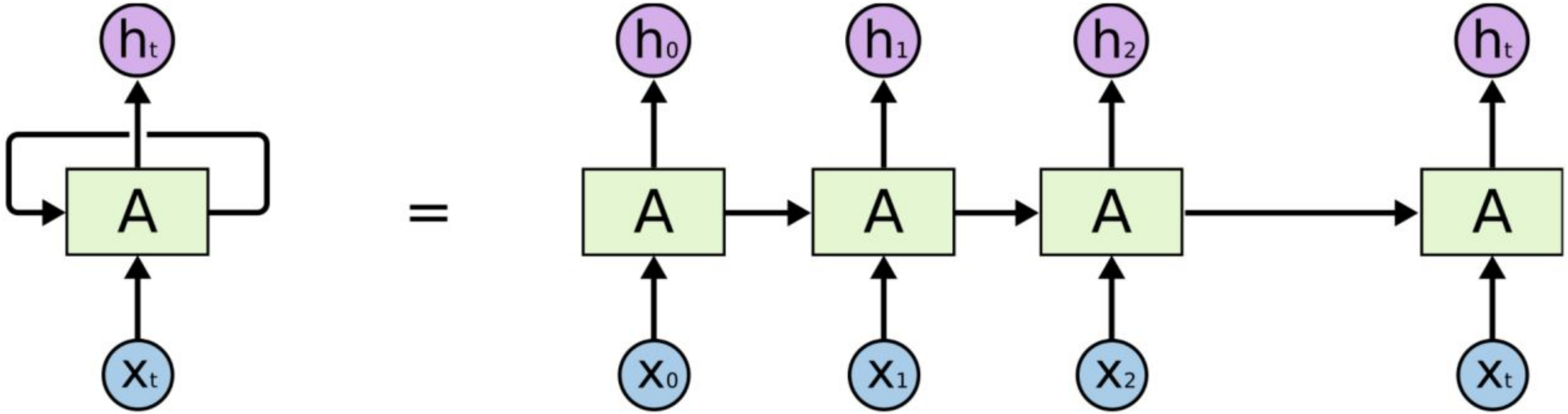
https://towardsdatascience.com/c0a830a288a4

Recurrent Neural Net (RNN)



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Net (RNN)



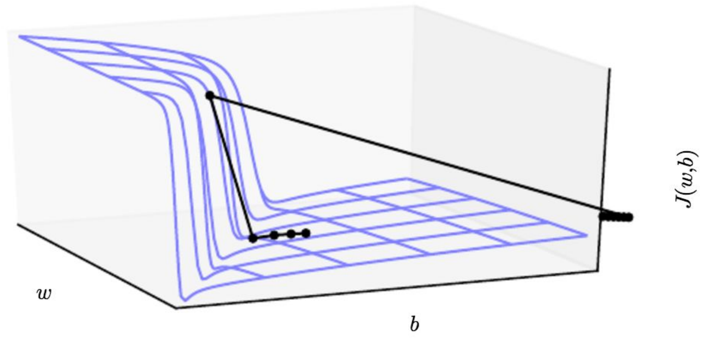
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

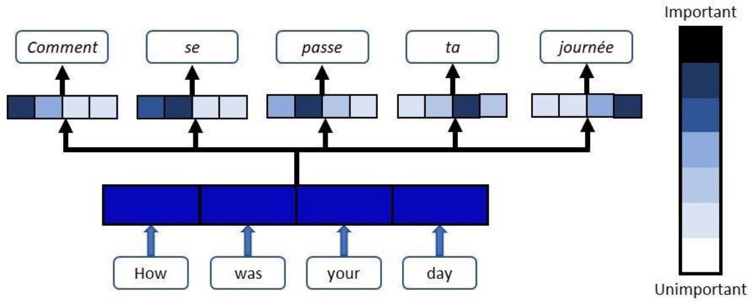
Challenges in RNNs

Exploding gradients

Vanishing gradients

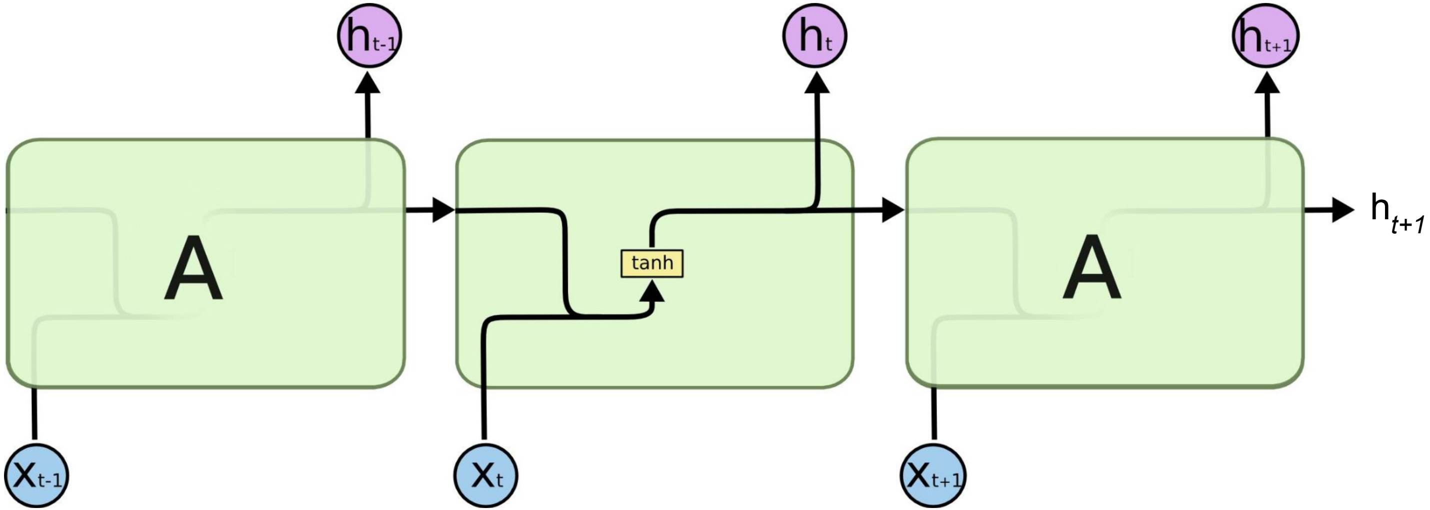
Coping with context (solved with *attention*)





Vanilla RNN cell

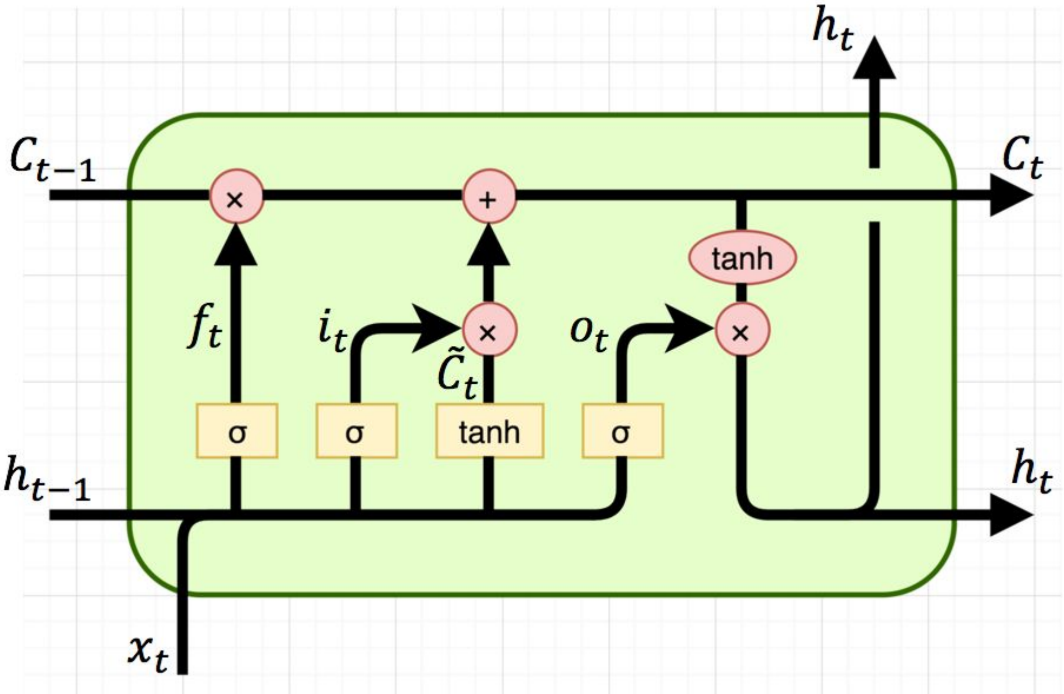
$$\mathbf{h}_t = \Phi(\mathrm{W}_h \mathbf{h}_{t-1} + \mathrm{W}_x \mathbf{h}_t + b)$$



https://towardsdatascience.com/44e9eb85bf21

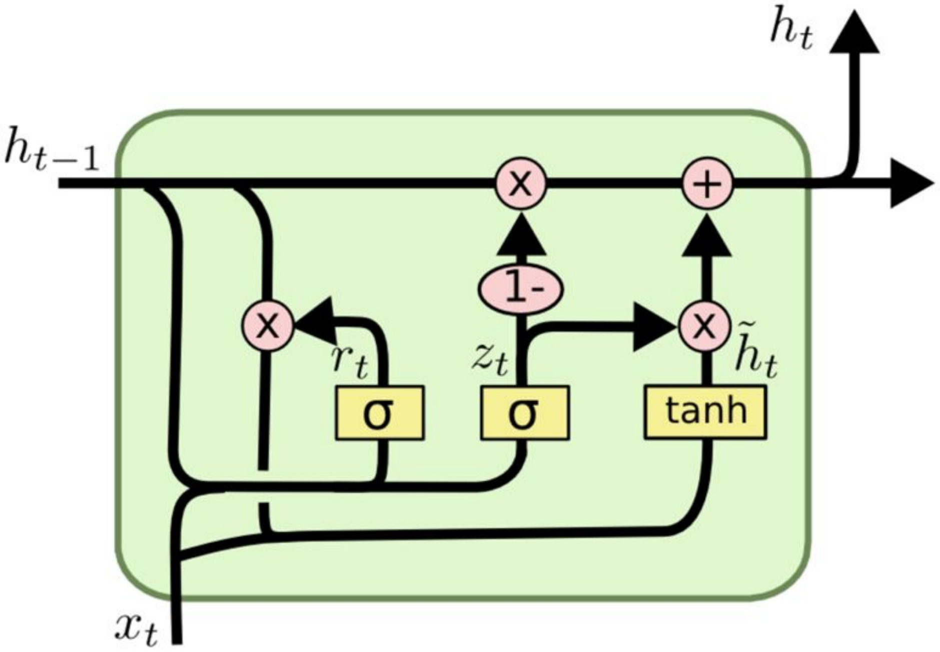
LSTM cell

$$\mathrm{G}_i = \sigma(\mathrm{W}_u[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i)$$
$$\mathrm{G}_f = \sigma(\mathrm{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f)$$
$$\mathrm{G}_o = \sigma(\mathrm{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_o)$$
$$\mathbf{c}_t = \mathrm{G}_i \odot \tilde{\mathbf{c}}_t + \mathrm{G}_f \odot \mathbf{c}_{t-1}$$
$$\tilde{\mathbf{c}}_t = \Psi(\mathrm{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_c)$$
$$\mathbf{h}_t = \mathrm{G}_o \odot \Psi(\mathbf{c}_t)$$



GRU cell

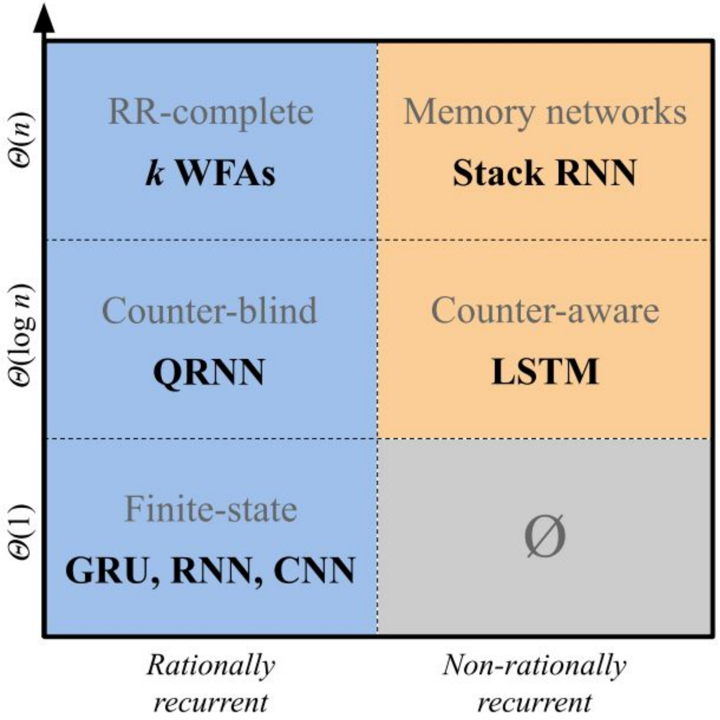
$$\mathrm{G}_u = \sigma(\mathrm{W}_u[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_u)$$
$$\mathrm{G}_r = \sigma(\mathrm{W}_r[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_r)$$
$$\tilde{\mathbf{c}}_t = \Psi(\mathrm{W}_c[\mathrm{G}_r \odot \mathbf{h}_{t-1}, \mathbf{x}_t] + b_c)$$
$$\mathbf{h}_t = \mathrm{G}_u \odot \tilde{\mathbf{c}}_t + (1 - \mathrm{G}_u) \odot \mathbf{h}_{t-1}$$



Classic RNN architectures

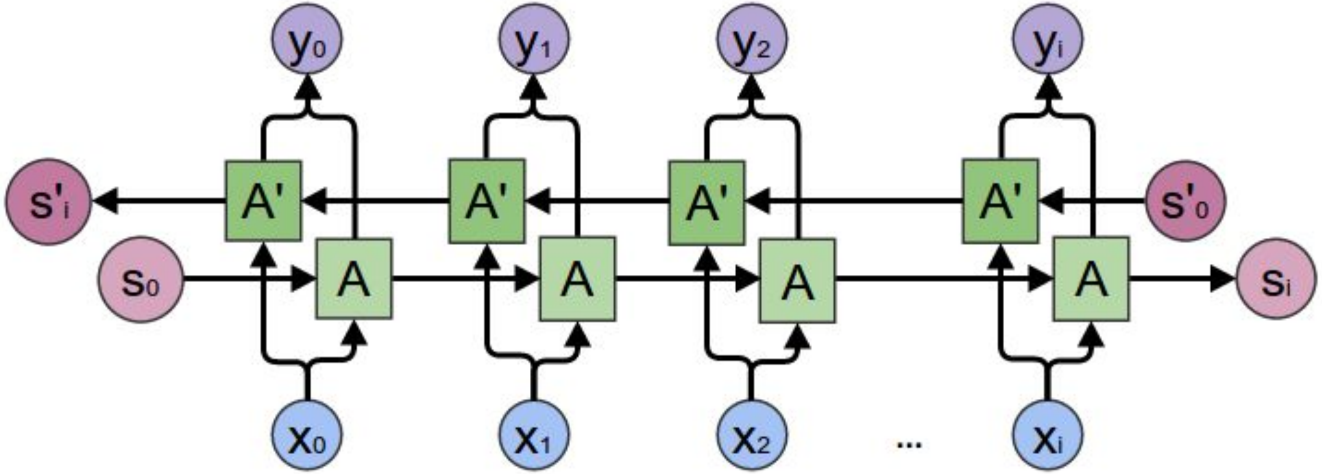
Bidirectional RNN (1997)

Sketch-RNN (2017)



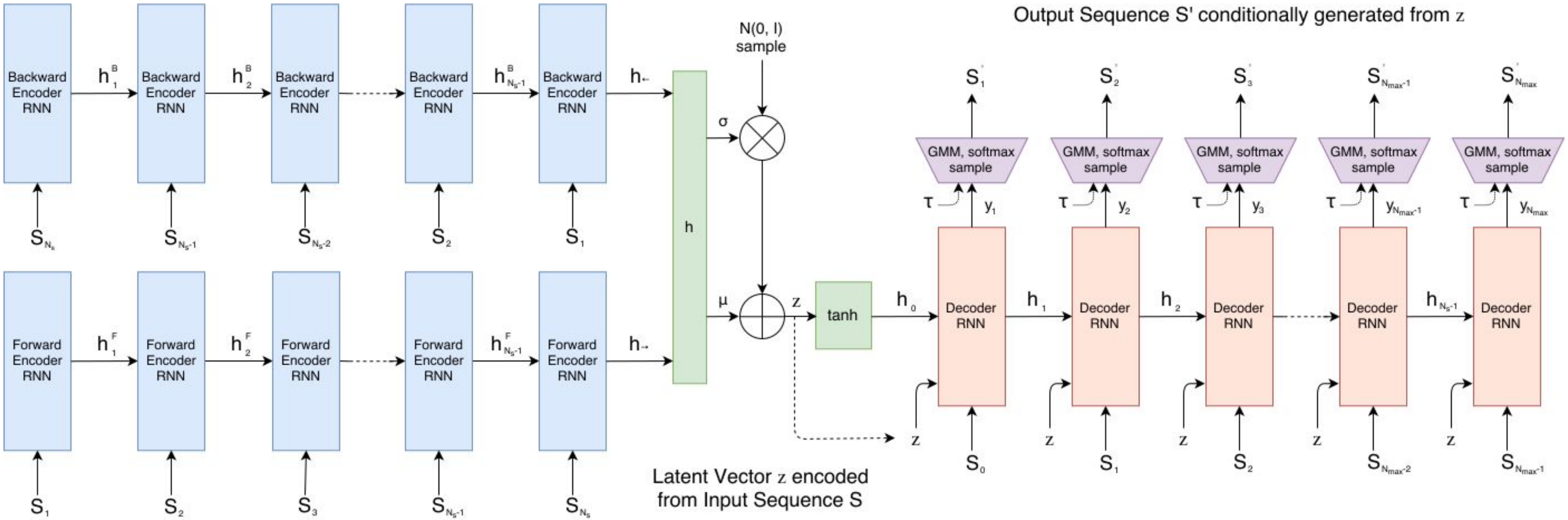
https://aclanthology.org/2020.acl-main.43.pdf

Classic RNN: **Bidirectional RNN**



http://colah.github.io/posts/2015-09-NN-Types-FP/

Classic RNN: **Sketch-RNN**

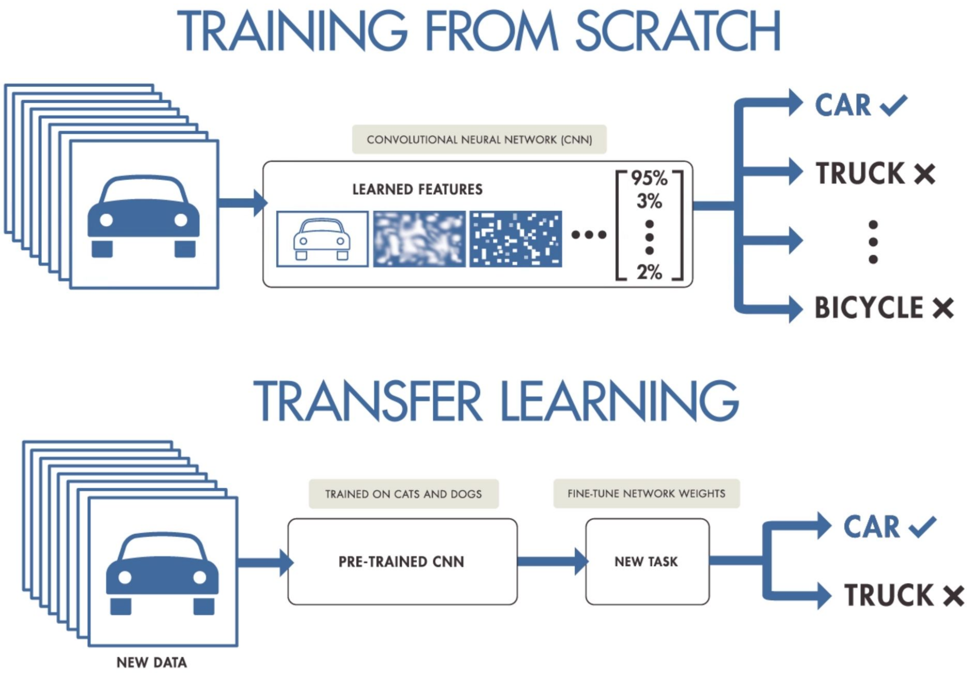


https://magenta.tensorflow.org/sketch-rnn-demo

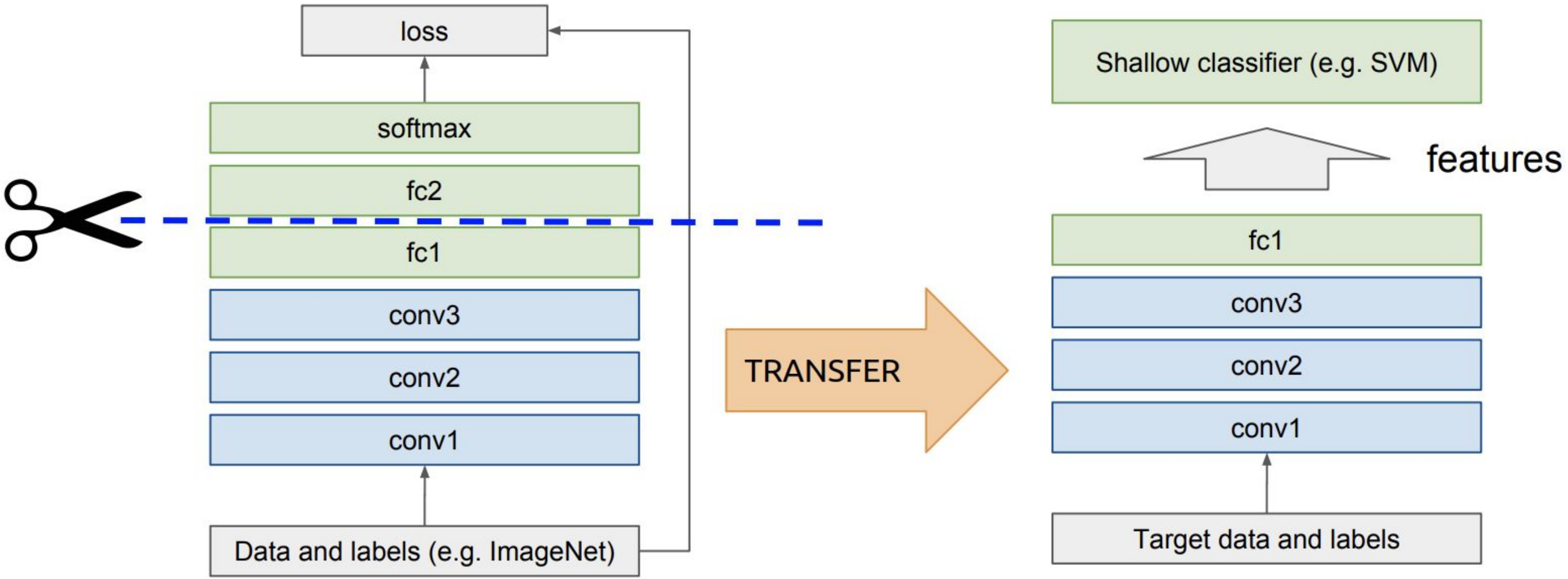
Transfer learning

Very popular with CNNs

Very scarce with RNNs



Transfer learning



http://imatge-upc.github.io/telecombcn-2016-dlcv/slides/D2L5-transfer.pdf