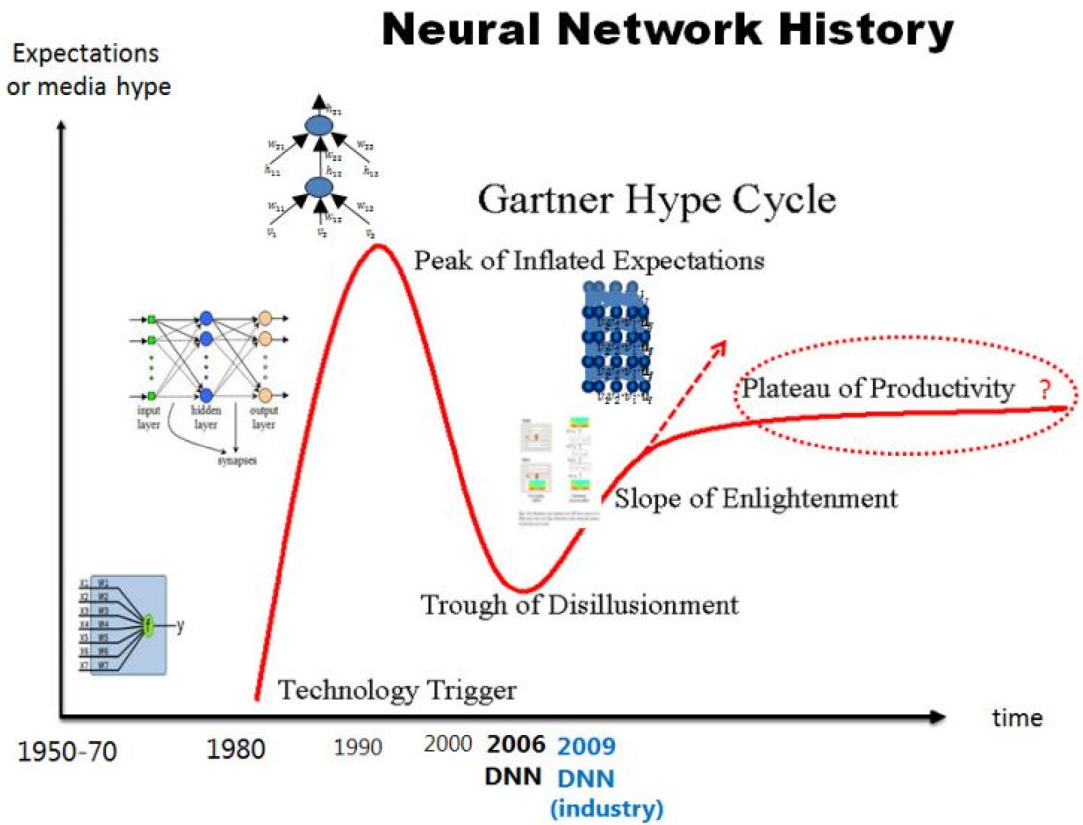
Deep Neural Networks

UEF SUMMER SCHOOL 2017

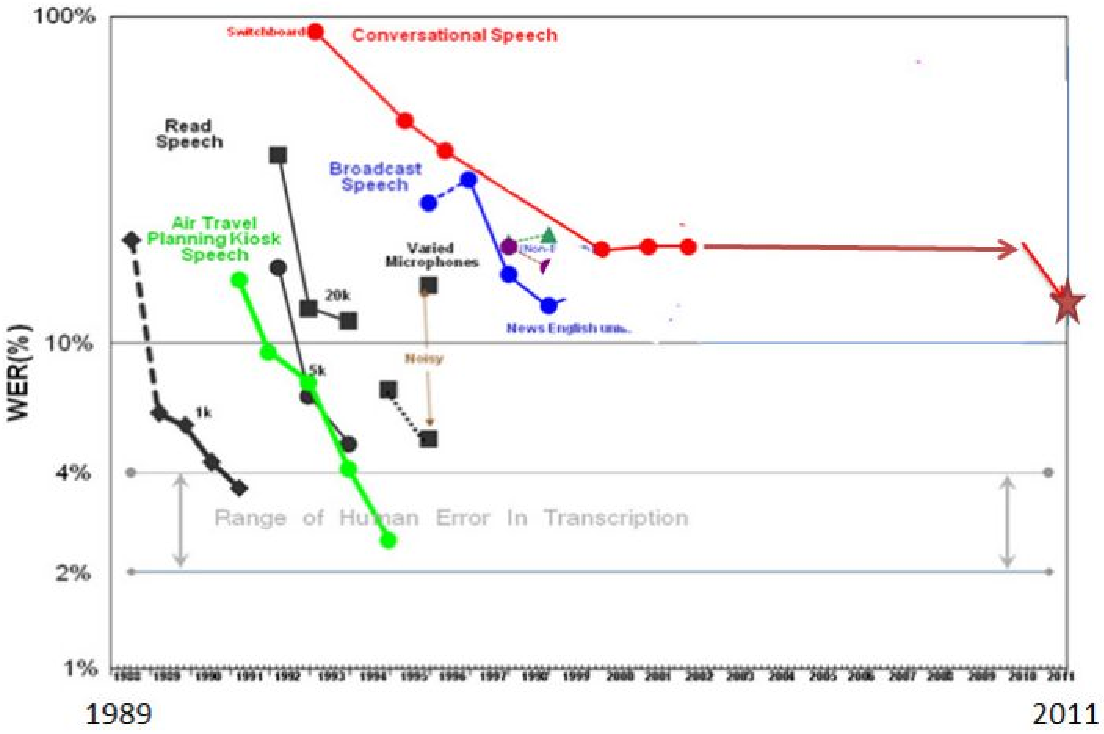
Ville Hautamäki

What is deep learning? (1 / 2)

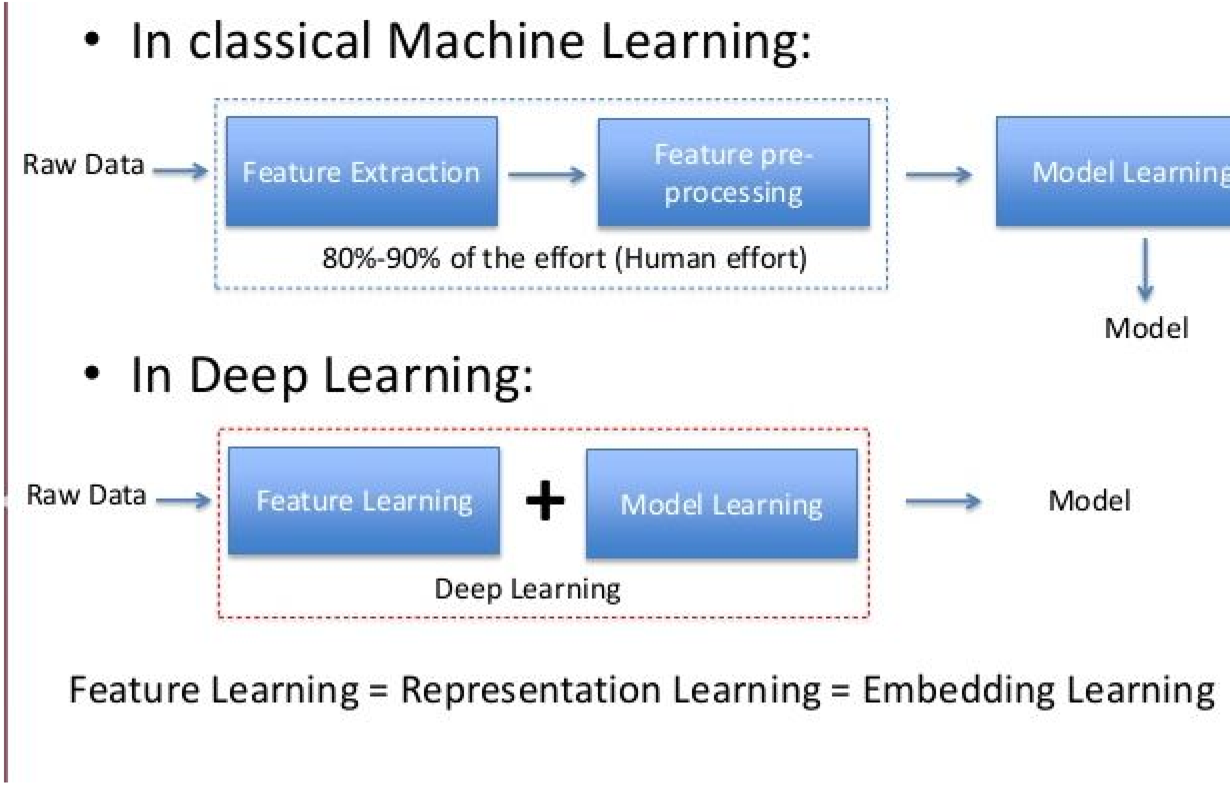
What is deep learning? (2 / 2)



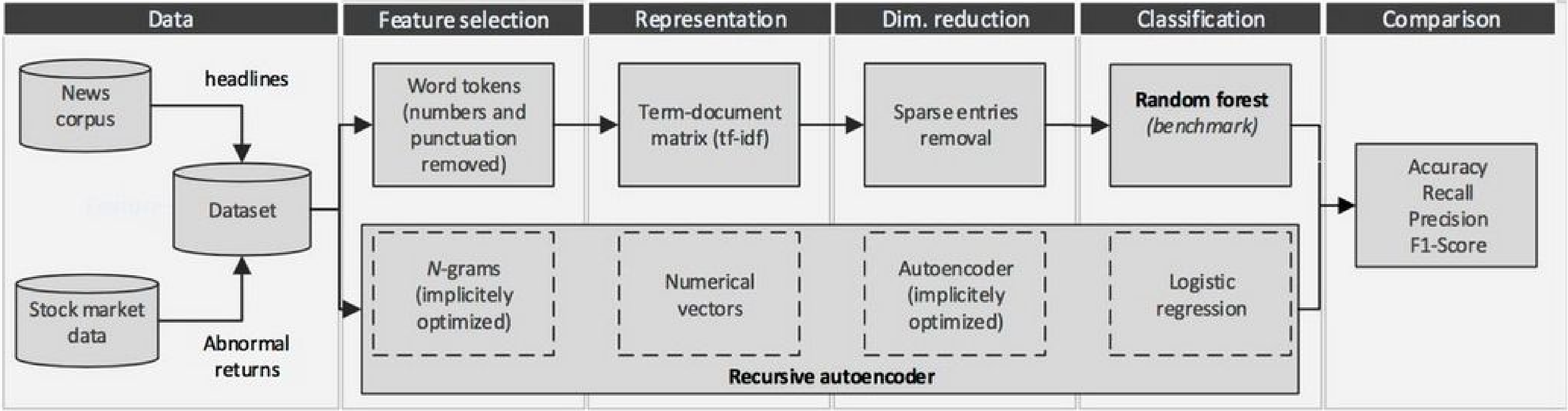
ASR performance a historical perspective



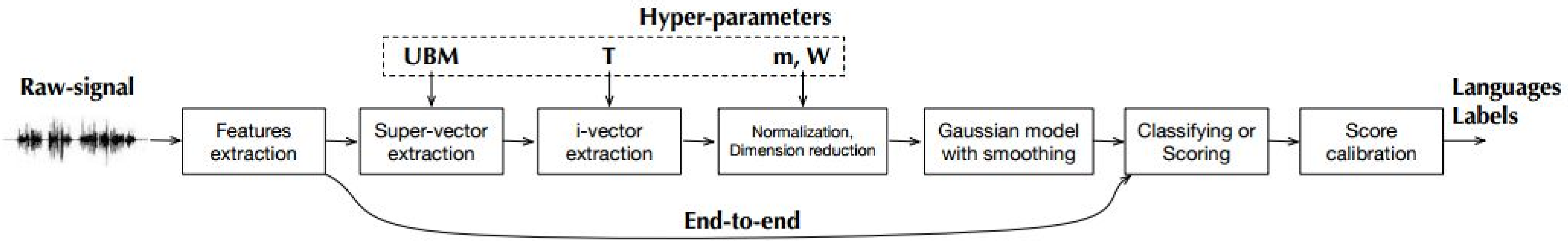
Comparison on classical ML vs. deep learning



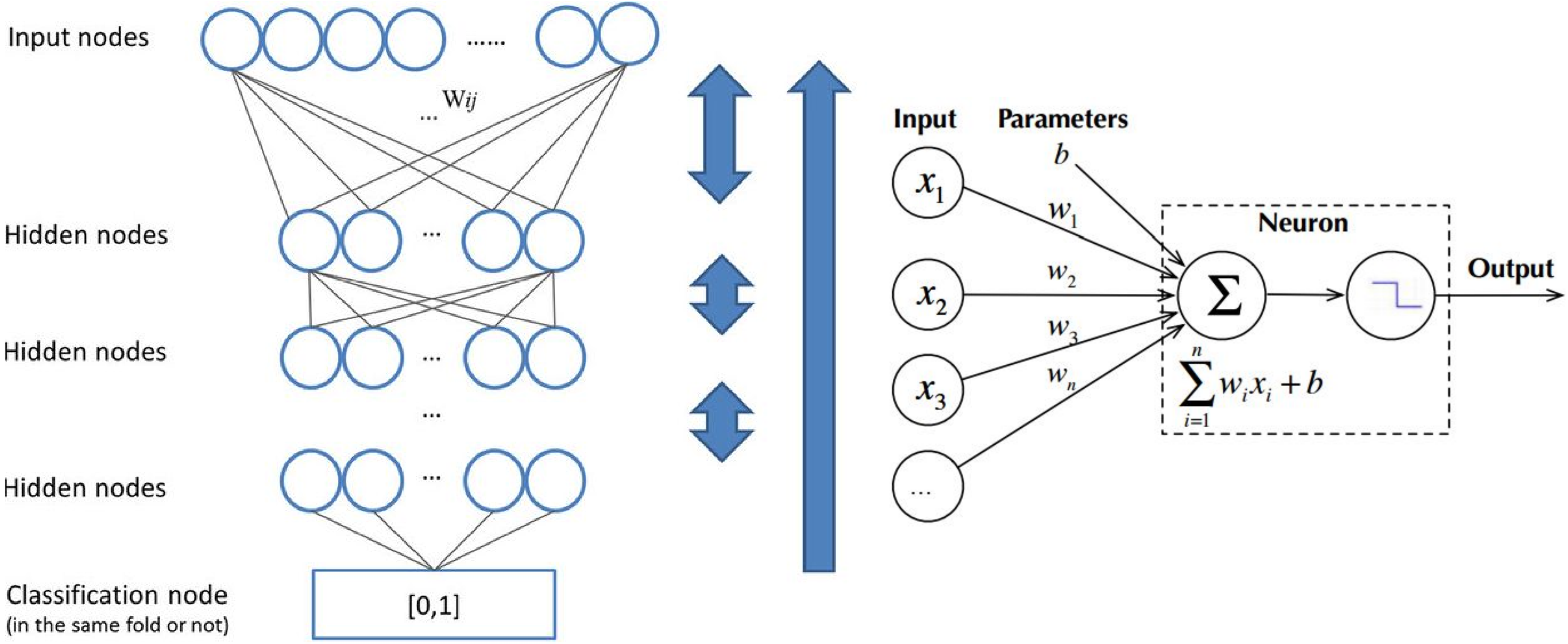
Comparison on classical ML vs. deep learning (NLP)



Comparison on classical ML vs. deep learning (LID)



Neural networks as universal function approximators



Estimating parameters: need of a loss function

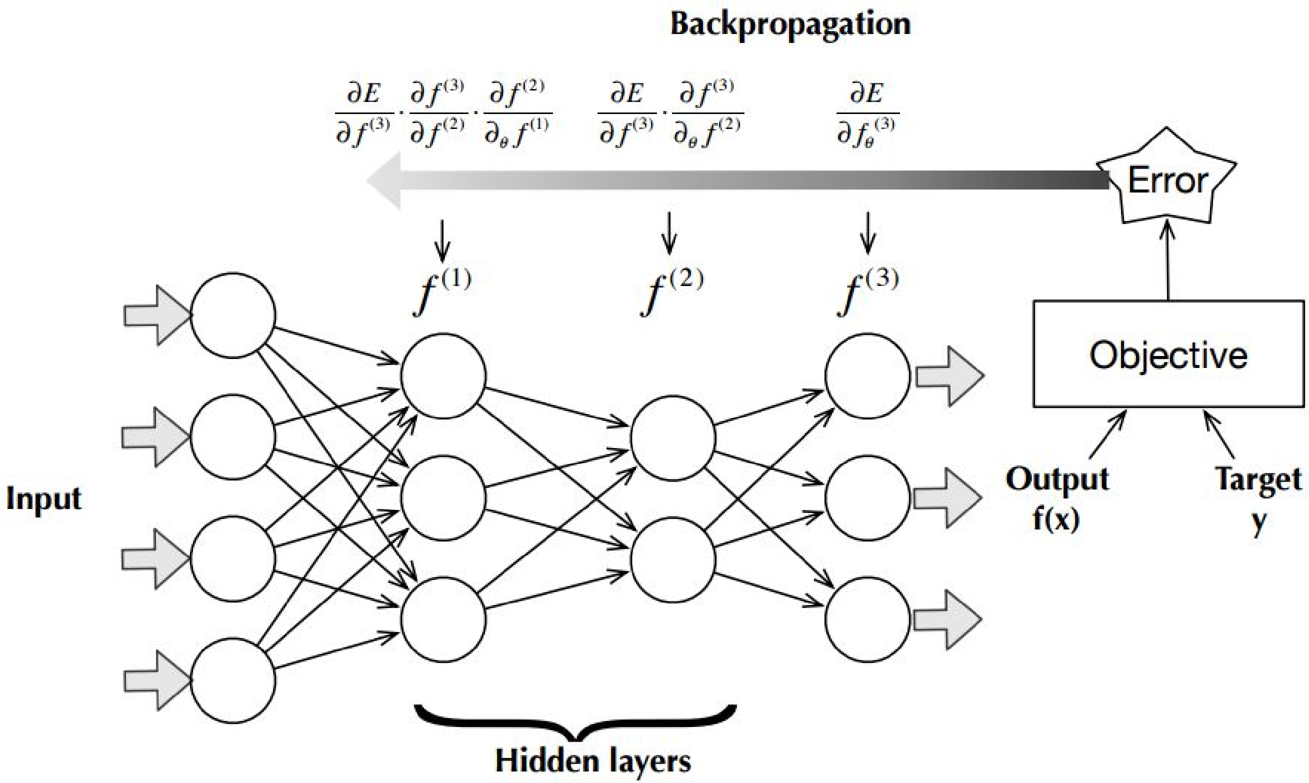
$$E_i = f_o(y_i, f(x_i)),$$

Loss function, such as MSE or Cross-entropy (CE)

$$E_{train} = \frac{1}{n}\sum_{i=0}^{n} E_i,$$

$$\mathbf{W}^l(t) = \mathbf{W}^l(t-1) - \eta \cdot \frac{1}{n_{batch}} \sum_{i=0}^{n_{batch}} \frac{\partial E_i}{\partial \mathbf{W}^l(t-1)},$$

Backprop: gradient chain-rule is your friend



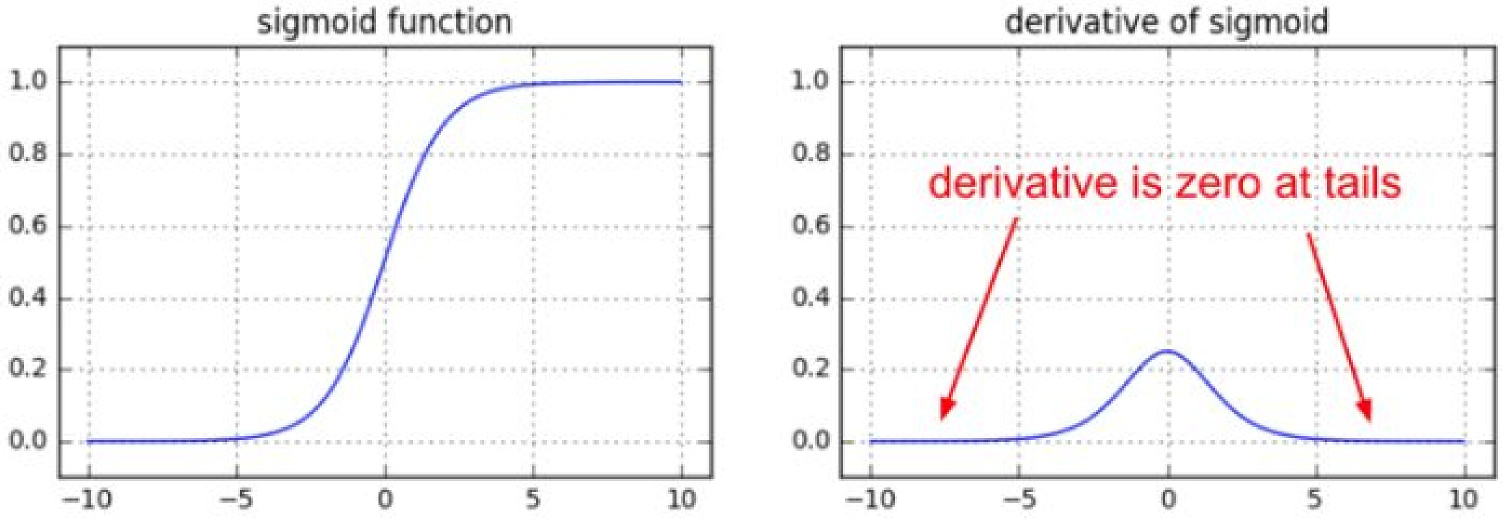
When things go wrong with backprop: gradient vanishing/ explosion

Let's see an example using numpy.

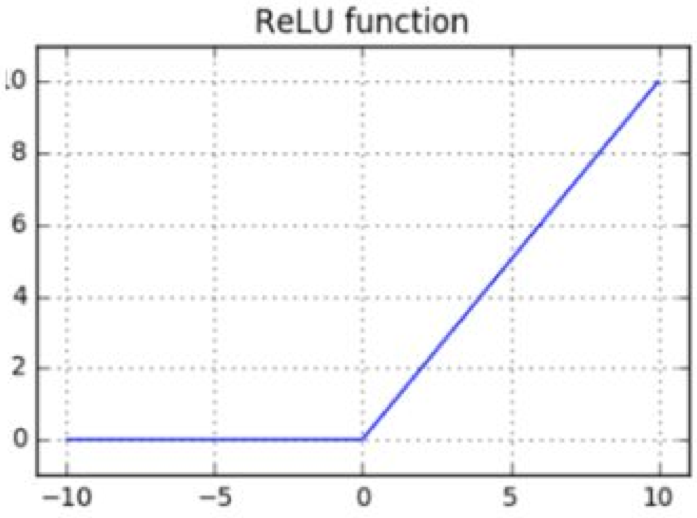
```
z = 1/(1 + np.exp(-np.dot(W, x))) # forward pass
dx = np.dot(W.T, z*(1-z)) # backward pass: local gradient for x
dW = np.outer(z*(1-z), x) # backward pass: local gradient for W
```

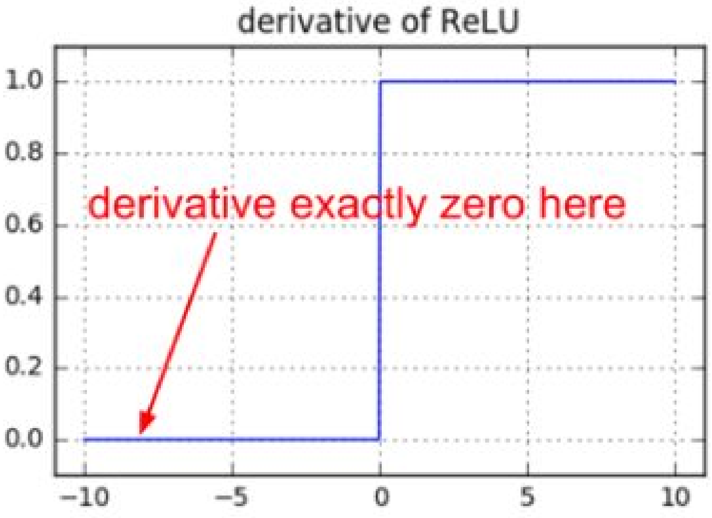
If your weight matrix W is initialized too large, the output of the matrix multiply could have a very large range (e.g. numbers between -400 and 400), which will make all outputs in the vector z almost binary: either 1 or 0. But if that is the case, z*(1-z), which is local gradient of the sigmoid non-linearity, will in both cases become **zero** ("vanish"), making the gradient for both x and W be **zero**. **The rest of the backward pass will come out all zero from this point on due to multiplication in the chain rule.**

Problems with classical sigmoidal activation



Rectified linear as an alternative activation





Stochastic gradient descent / decomposable loss

Algorithm 1 General learning procedure of neural network

Require: initialize all weights $\mathbf{W}(0)$ (sufficient small values is important [70])

for 1 **to** n_{epoch} **do**

shuffle-training-set *# suggested in [70]*

for mini-batch **to** training-batches **do**

 # Forward pass

mini-batch = normalize-data(mini-batch) *# suggested in [70]*

prediction = network-output(mini-batch | $\mathbf{W}(\mathbf{t-1})$)

error = objective-function(target, prediction)

 # Backward pass

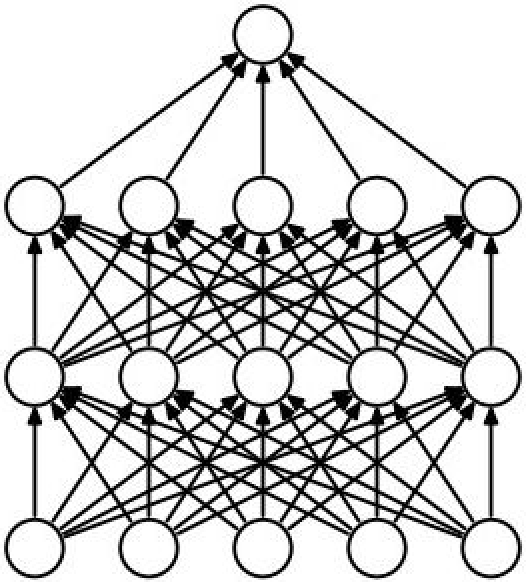
gradients = $\partial error / \partial \mathbf{W}(t-1)$

gradients = apply-constraint(gradients) *# prevent grad. vanishing, exploding [101]*

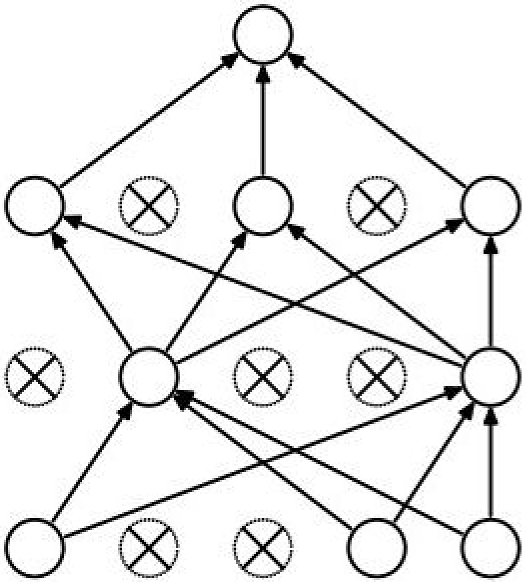
$\mathbf{W}(t)$ = update-algorithm($\mathbf{W}(t-1)$, η, gradients)

 # validating can be in the middle or in the end of an epoch

Avoiding overfit by dropout regularization

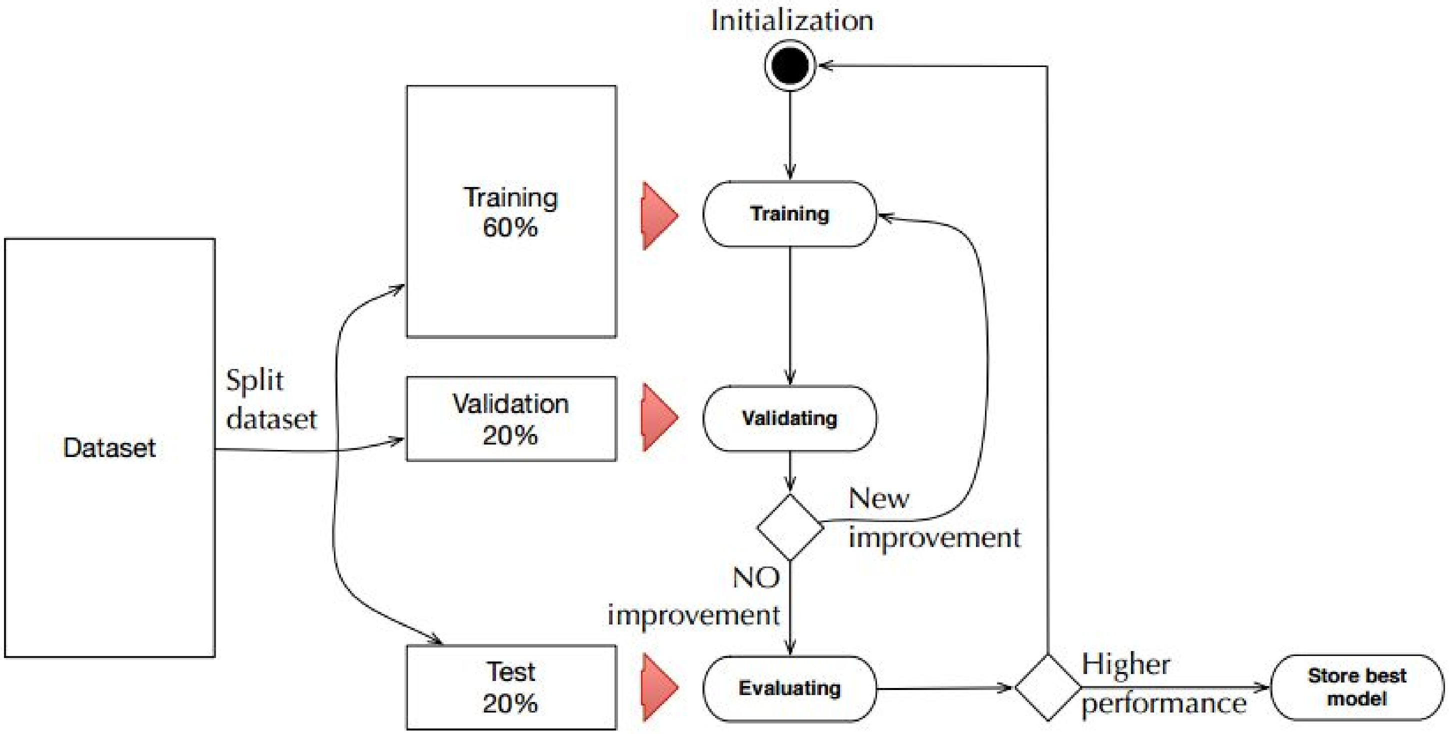


(a) Standard Neural Net

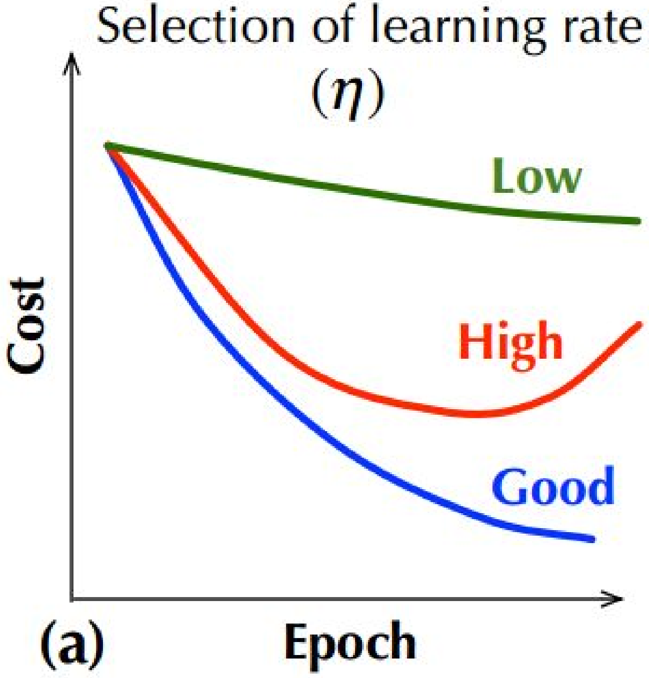


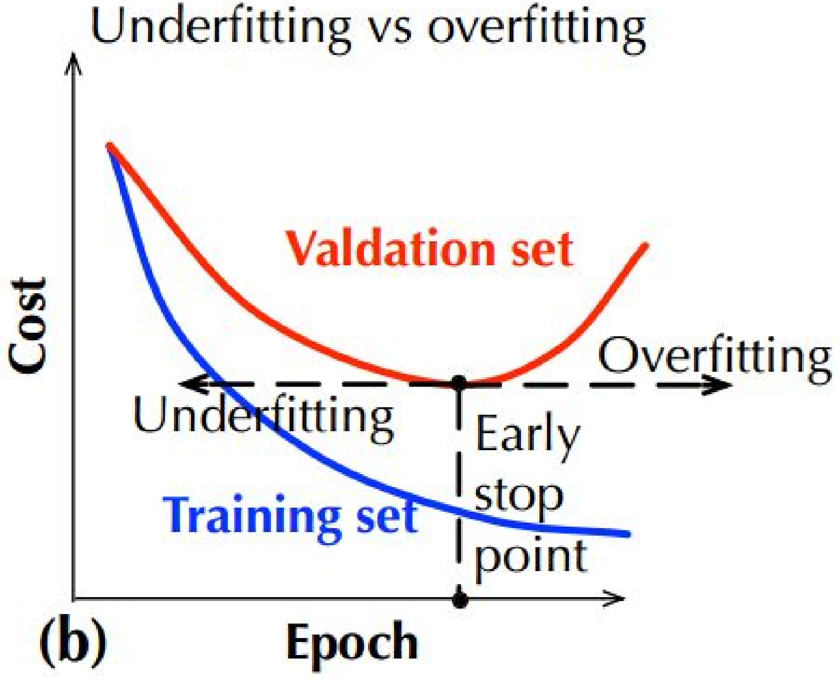
(b) After applying dropout.

Validation set: avoiding overfit



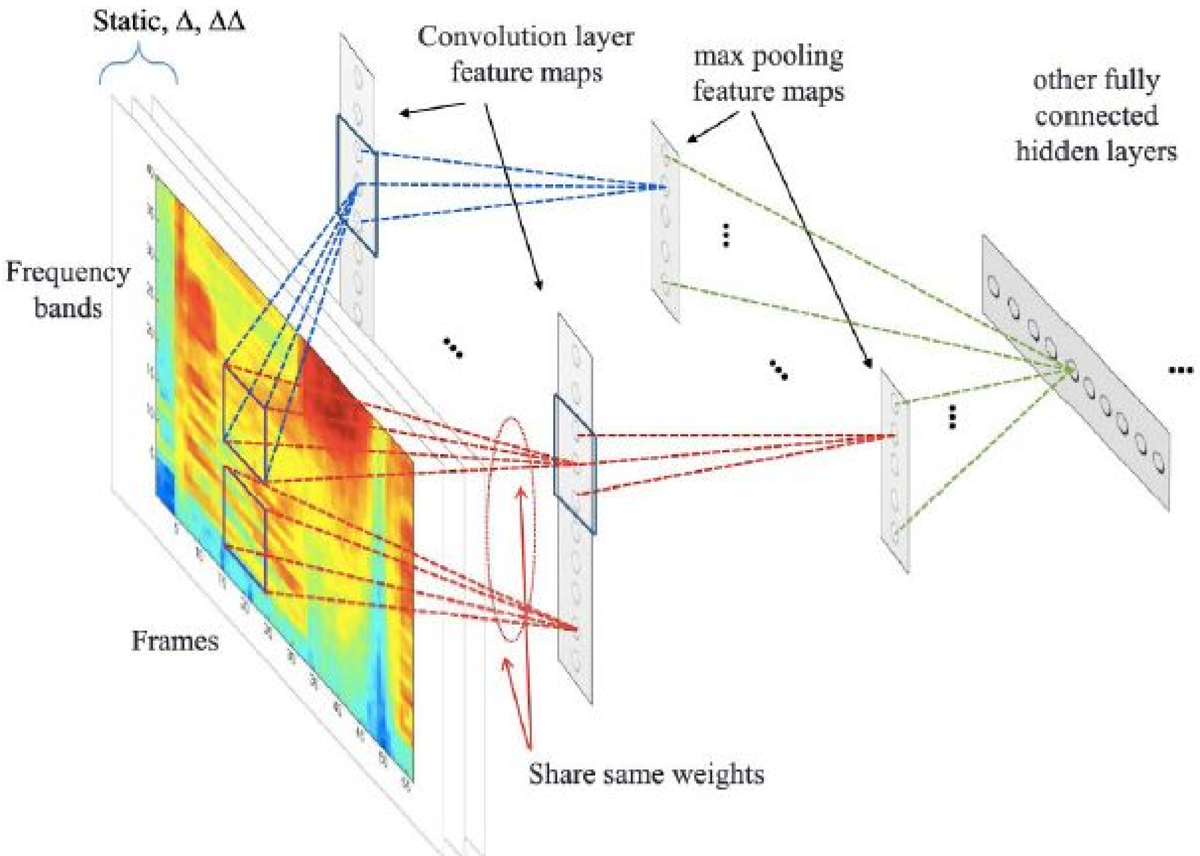
Validation set: avoiding overfit





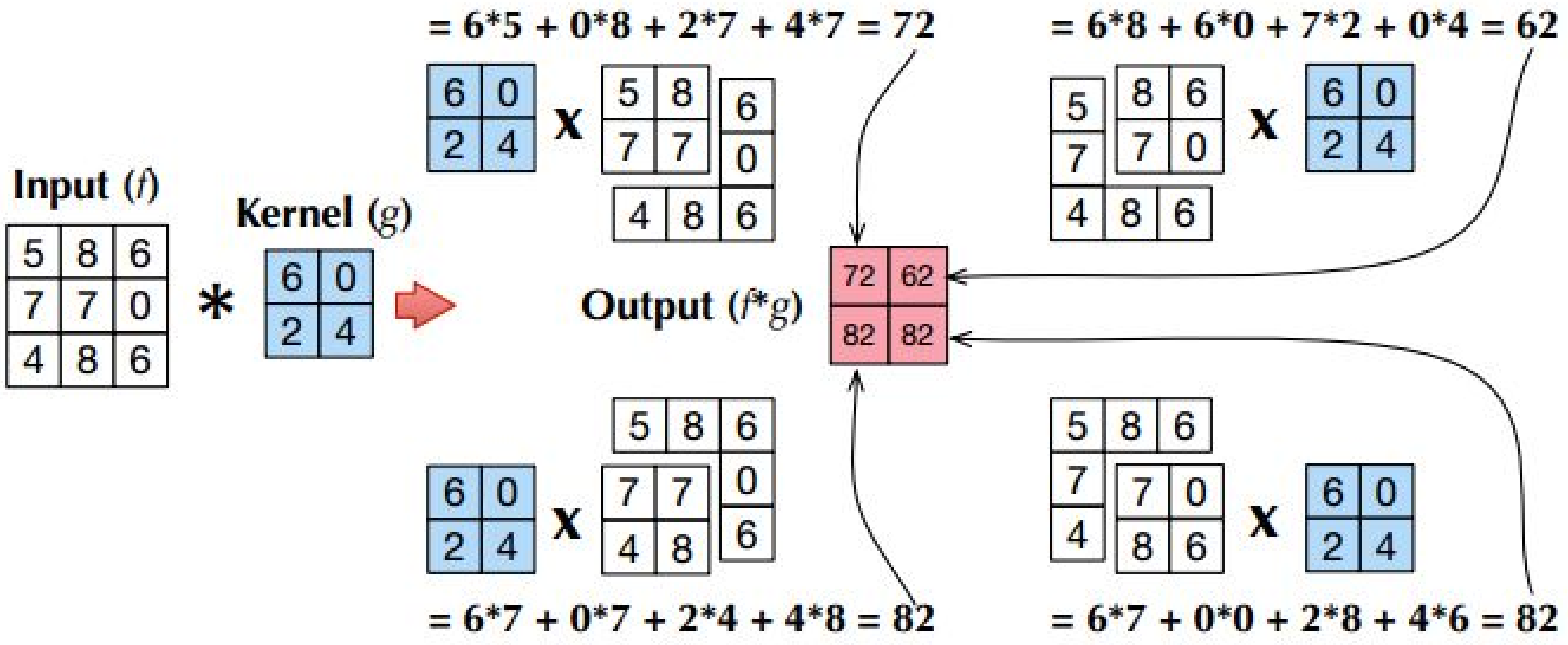
Convolutional neural networks

Local connectivity is beneficial (vs. fully connected)

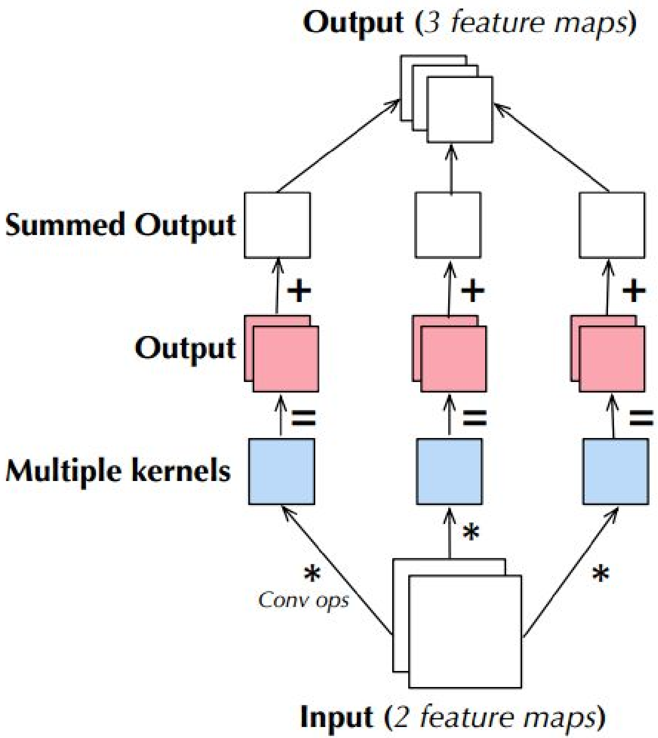


What is convolution?

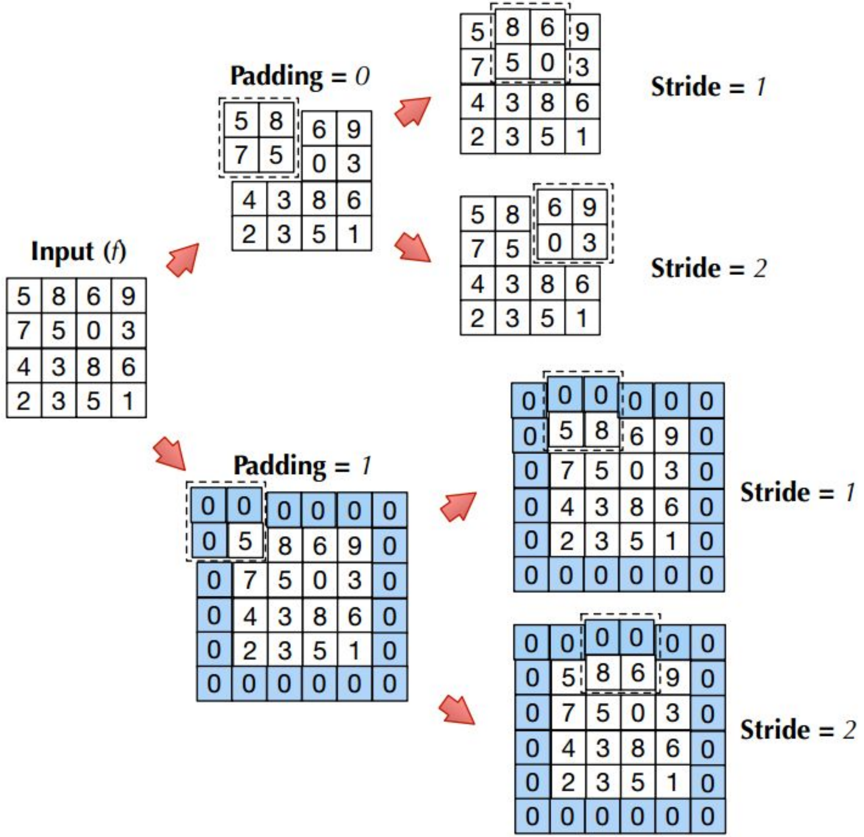
$$(\mathrm{f} \star \mathrm{g})(t) = \int_{-\infty}^{+\infty} \mathrm{f}(\tau)\mathrm{g}(t-\tau)d\tau = \int_{-\infty}^{+\infty} \mathrm{f}(t-\tau)\mathrm{g}(\tau)d\tau.$$



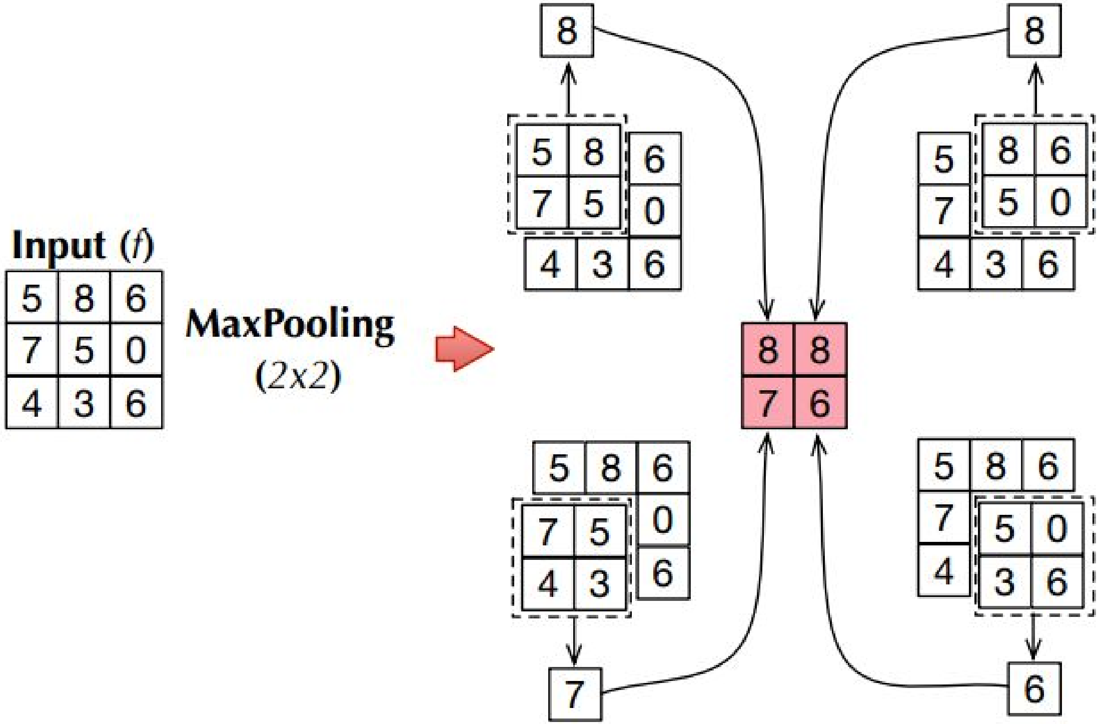
Structure of the convolutional neural networks (CNN)



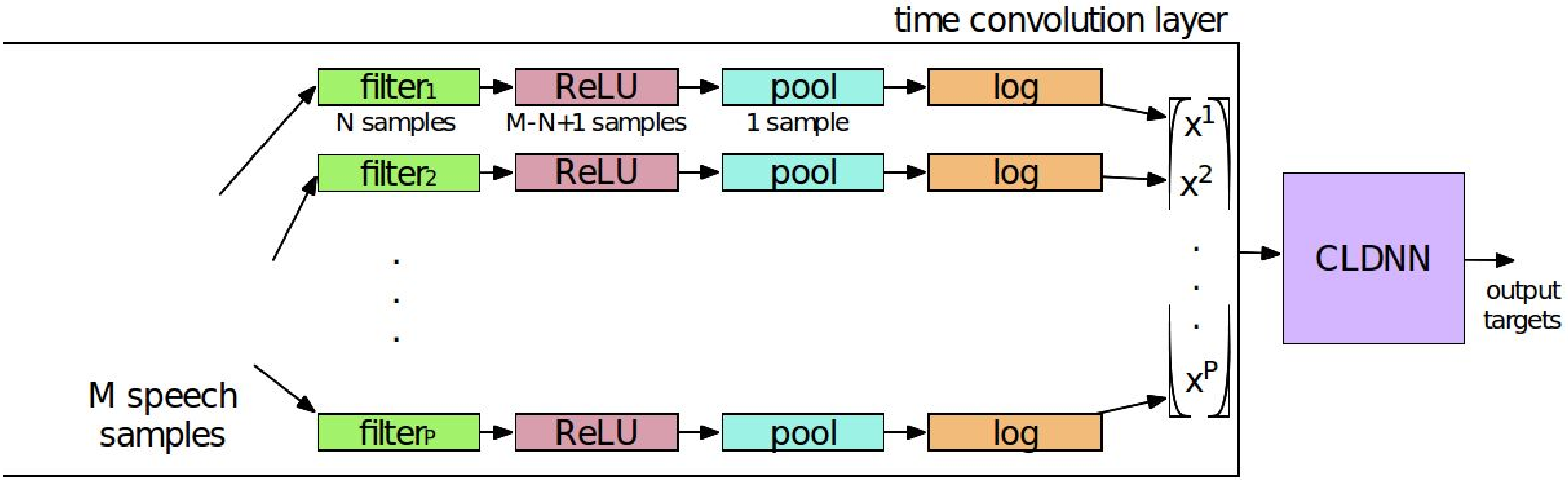
Stride and padding parameters



Max Pooling

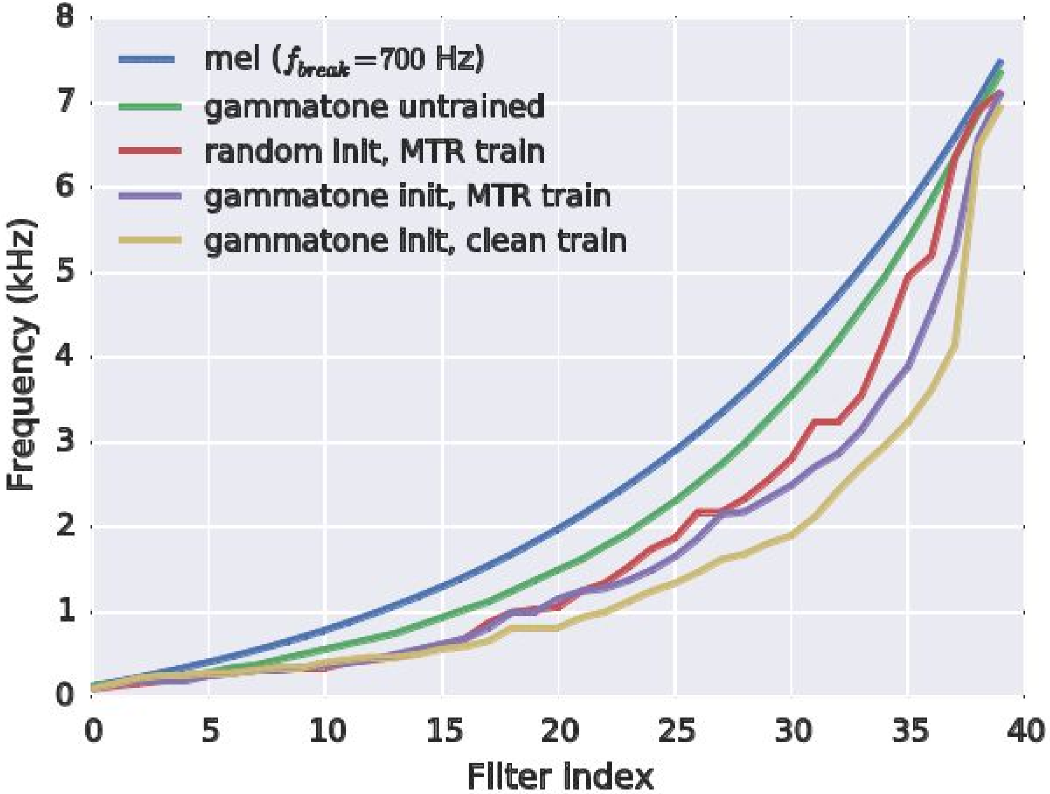

Input (f)
5 8 6
7 5 0
4 3 6
MaxPooling
(2x2)
8
5 8 6
7 5 0
4 3 6
8
5 8 6
7 5 0
4 3 6
8 8
7 6
5 8 6
7 5 0
4 3 6
7
5 8 6
7 5 0
4 3 6
6

CNN as a learned feature extractor

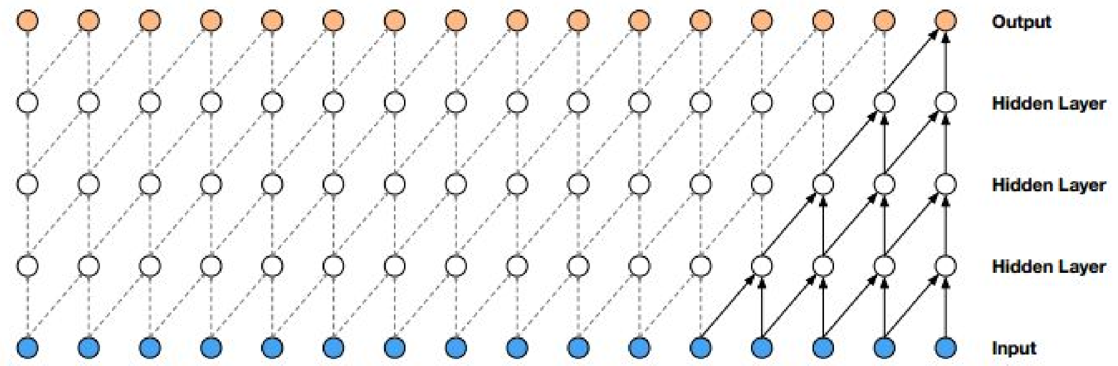


Frame-level features created by shifting window around M raw input samples by 10ms

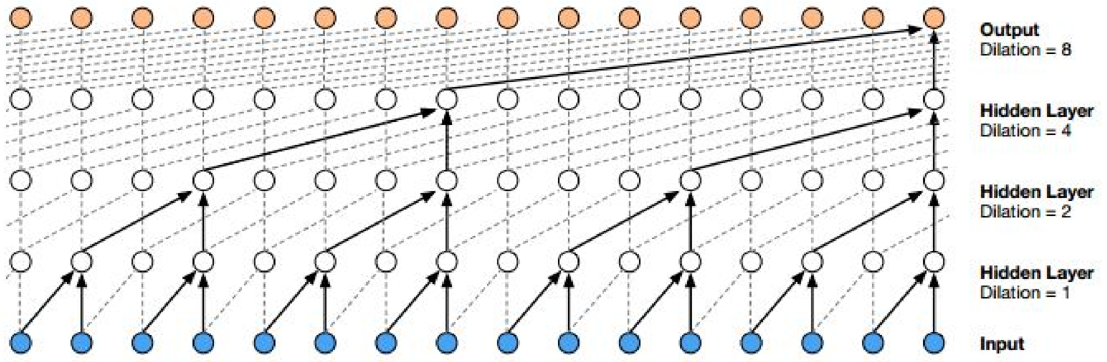
Interpretability of learned filters



WaveNet example



Time convolution



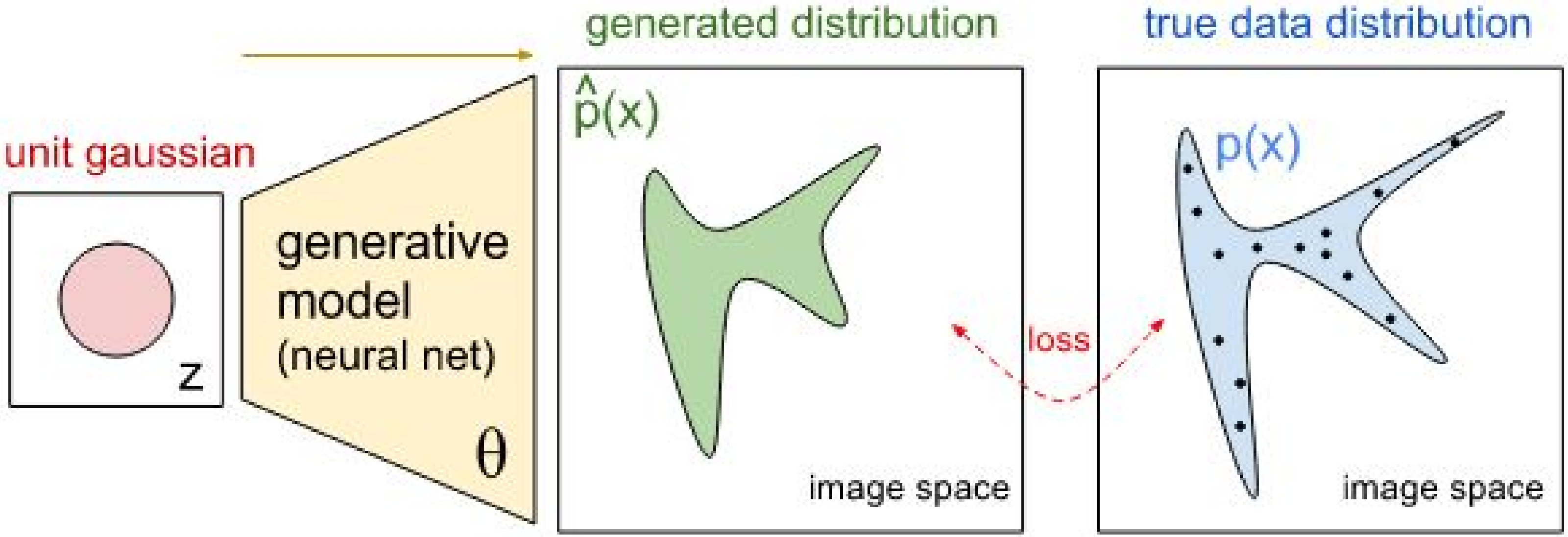
Dilated time convolution

Generative Modeling with Neural Networks

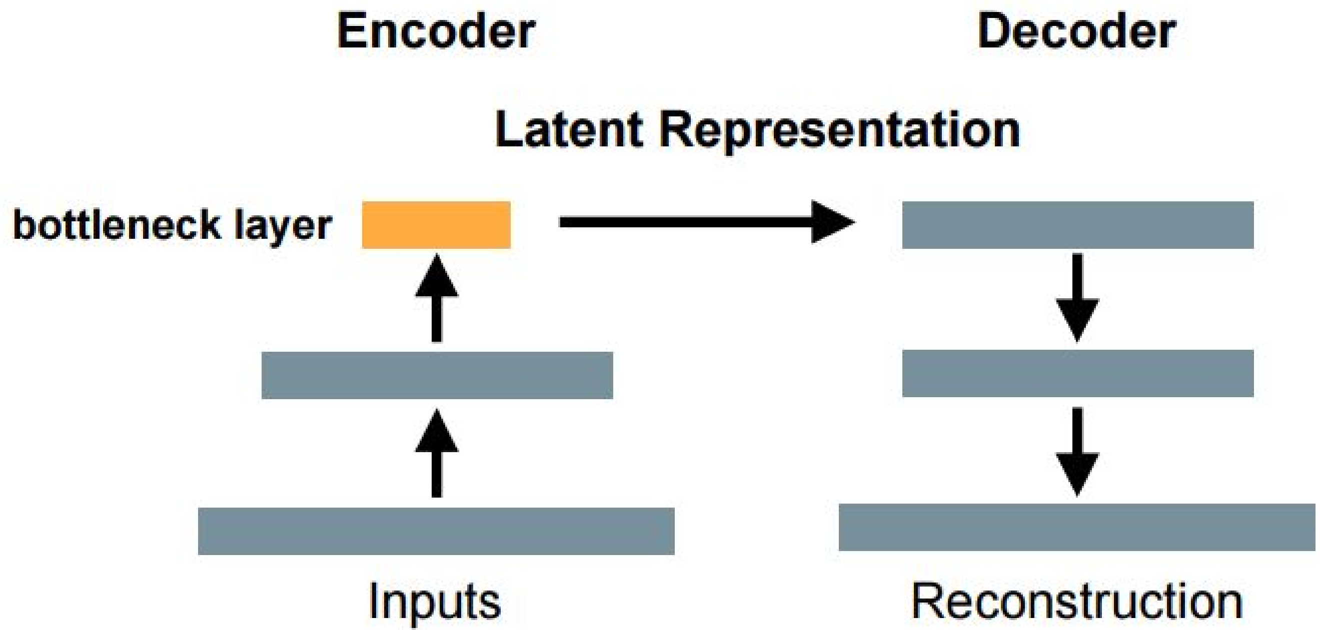
Why generative instead of discriminative?

- Probabilistic interpretation: we not only get a point estimates but the whole distribution.
- Possibility to sample fake data.
- Transfer learning. Such as in brain imaging, we can learn general image model from photographs (like ImageNet) and then use only a small set of application domain images to adapt the model to the new domain.
- It is unsupervised learning, the big open problem in machine learning.
- Idea is that, **you can only truly understand any phenomenon if you can generate it.**

We learn to generate from a latent code



First deep generative model: autoencoder

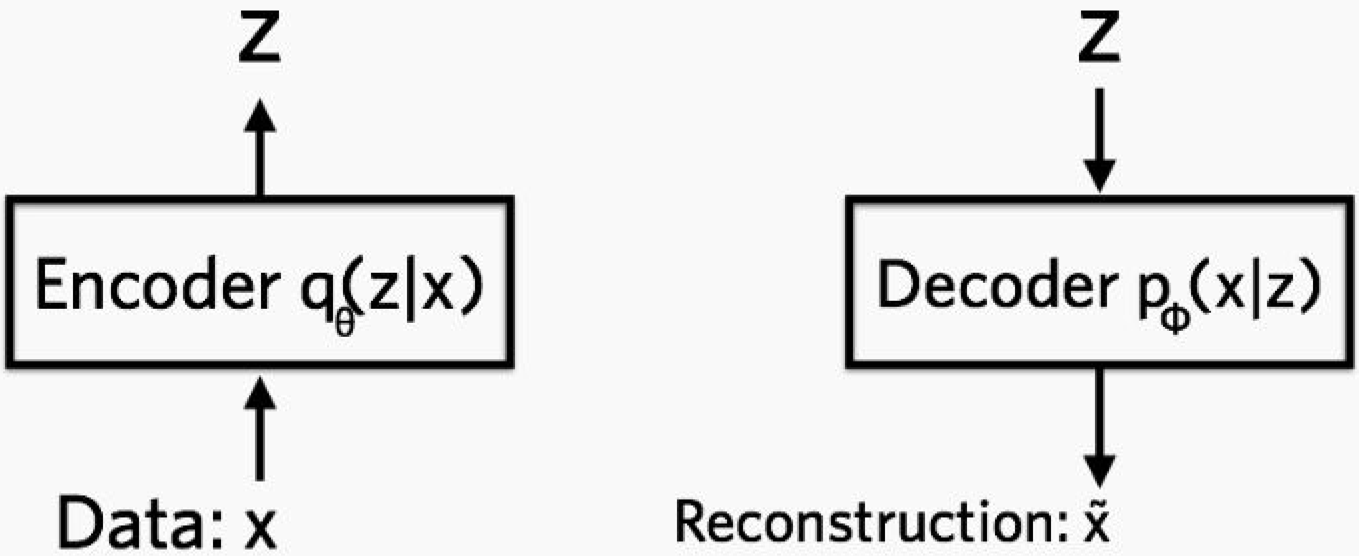


$$L = (x - \hat{x})^2$$

Unsupervised learning is *data compression*

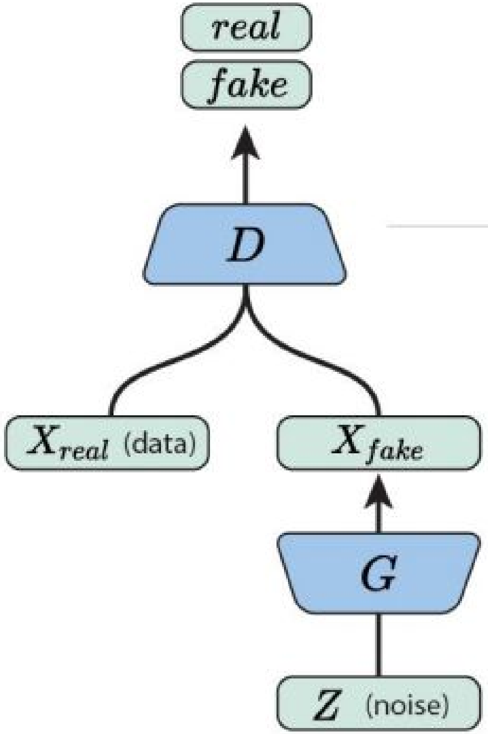
- If we are able to represent the data reliably with a short **code**, then we are able to compress it.
- Interestingly, already one of the founding fathers of data compression Prof. Jorma Rissanen alluded to the importance of this observation.

Variational autoencoder (VAE)



THE ENCODER COMPRESSES DATA INTO A LATENT SPACE (Z). THE DECODER RECONSTRUCTS THE DATA GIVEN THE HIDDEN REPRESENTATION.

Generative adversarial network (GAN)



The **discriminator** tries to distinguish genuine data from forgeries created by the generator.

The **generator** turns random noise into immitations of the data, in an attempt to fool the discriminator.