DEPARTMENT OF COMPUTER SCIENCE

Advanced Machine Learning in Finance

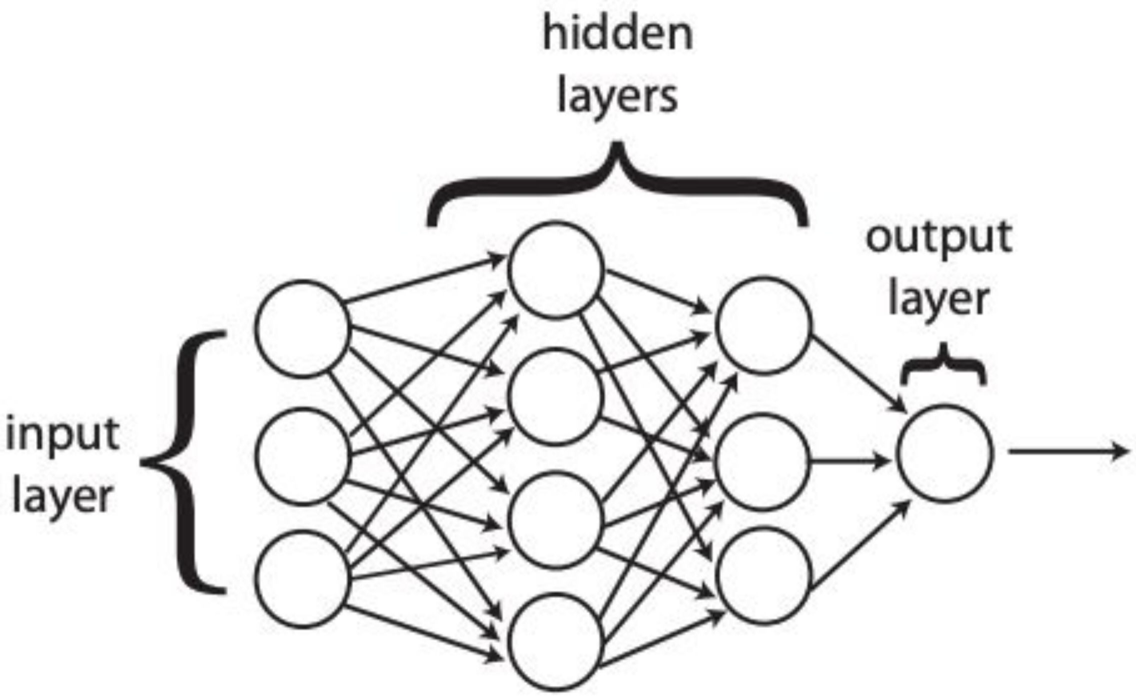
Paolo Barucca

Tomaso Aste

LECTURE 5

- Feedforward Neural Networks
- Loss Functions
- Activation Functions
- Convolutional Neural Networks
- Case study: Time Series Classification

Multilayer Perceptron



$$\hat{y} = f_L\Big(\sum_{j=1}^{N_{L-1}} w_{1j}^{(L)} a_j^{(L-1)}(x) + b_1^{(L)}\Big)$$

$$z_i^{(l)} = (w^{(l)T} a^{(l-1)} + b^{(l)})_i \qquad a_i^{(l)}(x) = f_l\Big(\sum_{j=1}^{N_{l-1}} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}\Big) = f_l(z_i^{(l)})$$

Figures from A high-bias, low-variance introduction to Machine Learning for physicists. arXiv:1803.08823. Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre Day, Clint Richardson, Charles Fisher, David Schwab.

Multilayer Perceptron

Kolmogorov-Arnold’s representation theorem and universal approximation theorems provide the mathematical background to prove that multi-layer perceptrons can be used as universal approximators of arbitrary continuous many variables functions

$$\sup_{x \in K} \| f(x) - g(x) \| < \varepsilon$$

- Arbitrary-width case [Cybenko, Hornik]
- Arbitrary-depth case [Gripenberg, Lu et al.]
- Bounded width and depth case [Maiorov and Pinkus]

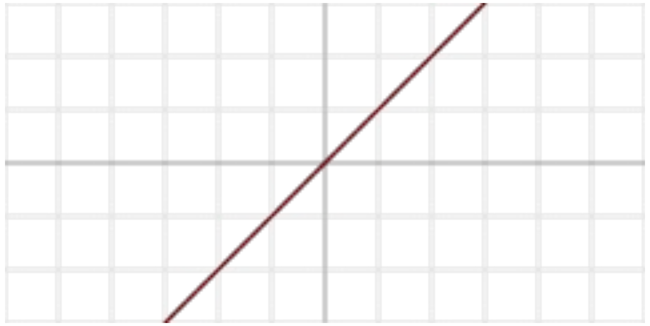
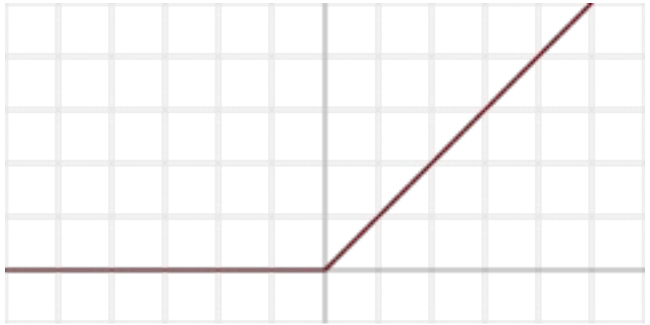
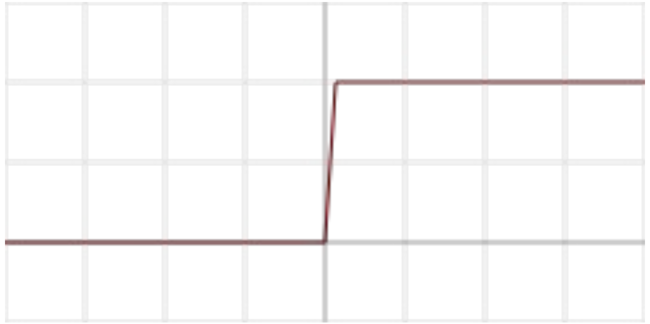
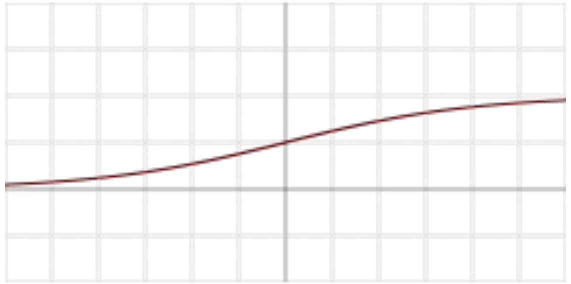
Activation functions

Equally important is to choose the activation functions for each layer

- Linear activation
- ReLu activation
- Heaviside activation
- Logistic activation

In particular, the activation function for the last layer should be able to output values in the same range of the real output data

Activation functions

Loss functions

In general, we need to define a cost function that we hope to minimise

- L-1 norm Loss
- Quadratic Loss
- Binary Cross-Entropy
- Multinary Cross-Entropy

depending on the kind of output data and task we have.

Loss functions

- L-1 and L-2 norm loss functions

$$E_1(w,b) = \frac{1}{P}\sum_{k=1}^{P} |\hat{y}_k(w,b) - y_k|$$

$$E_2(w,b) = \frac{1}{P}\sum_{k=1}^{P} (\hat{y}_k(w,b) - y_k)^2$$

Loss functions

- Cross-entropy loss functions

$$E_{CE}(w,b) = -\sum_{k=1}^{P} y_k \log \hat{y}_k(w,b) + (1 - y_k)\log\left[1 - \hat{y}_k(w,b)\right]$$

$$E_{CE}(w,b) = -\sum_{k=1}^{P}\sum_{m=1}^{M} y_{km} \log \hat{y}_{km}(w,b) + (1 - y_{km})\log\left[1 - \hat{y}_{km}(w,b)\right]$$

Minimising losses

- Whatever cost function we choose then we would hope to be able to minimise it, i.e.

$$(w^*, b^*) = \underset{w,b}{argmin} E(w, b)$$

- In order to minimise the cost function, we need to compute its relative change with respect to network weights and biases so that we can update them in a direction that will eventually converge to a minimum value.

Gradients

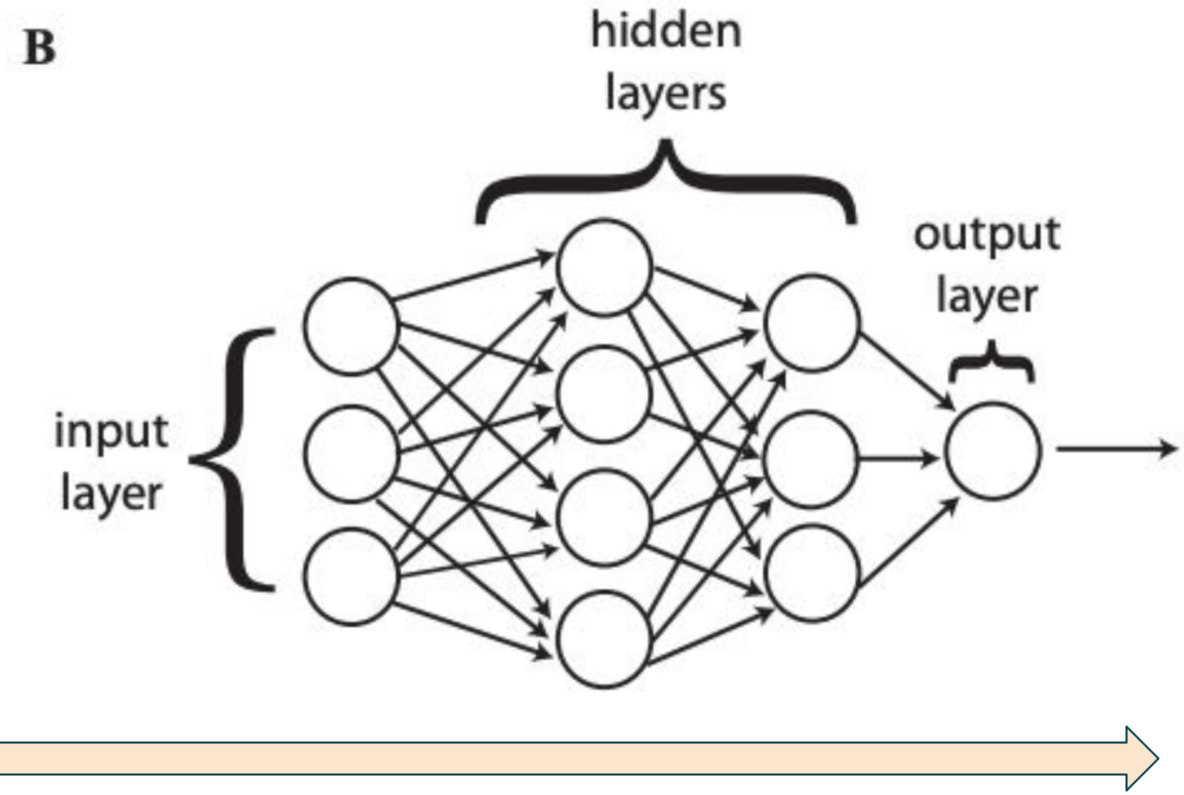
- We can use relative changes and derivatives in multiple algorithms (GD, SGD, etc.), but in all cases we need to compute:

$$\frac{\partial E(w,b)}{\partial w}$$

- And update weights accordingly,

$$w^{(new)} = g\left(w^{(old)}, \frac{\partial E(w)}{\partial w}\right)$$

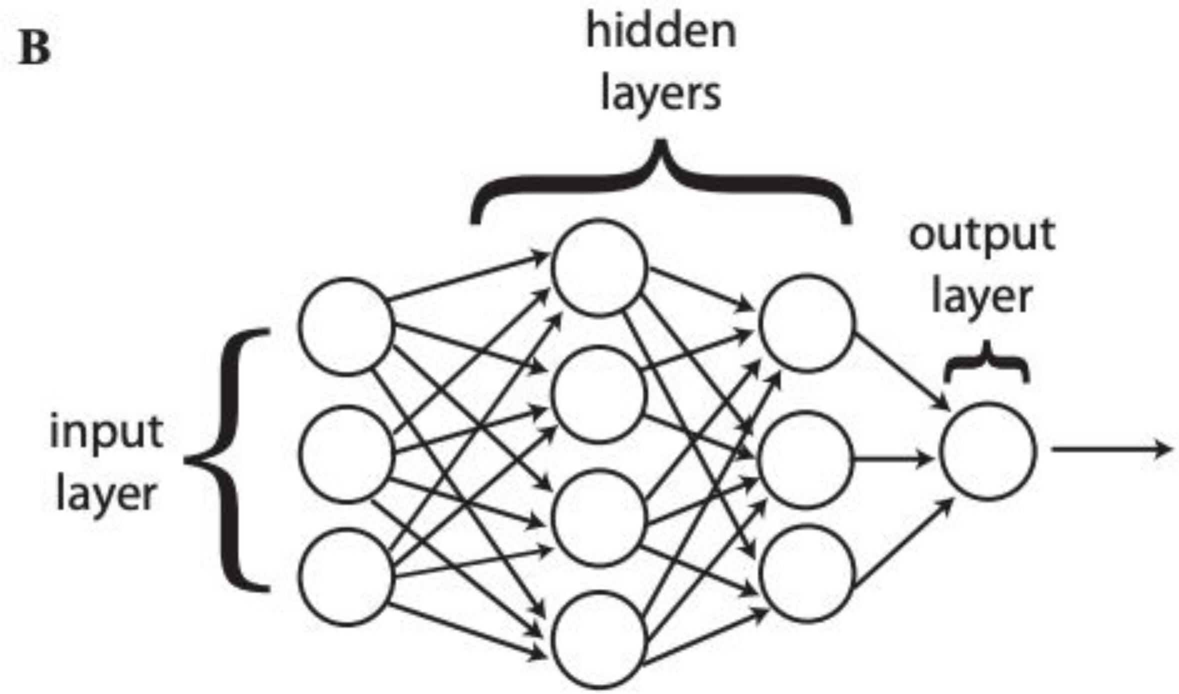
Backpropagation



$$\Delta_j^L = \frac{\partial E}{\partial z_j^L}$$

$$z_i^{(l)} = (w^{(l)T} a^{(l-1)} + b^{(l)})_i \qquad a_i^{(l)}(x) = f_l(\sum_{j=1}^{N_{l-1}} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}) = f_l(z_i^{(l)})$$

Backpropagation



$$\Delta_j^L = \frac{\partial E}{\partial z_j^L}$$

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial a_j^l} f_l'(z_j^l) = \frac{\partial E}{\partial b_j^l}$$

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \sum_k \frac{\partial E}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \left(\sum_k \Delta_k^{l+1} w_{kj}^{l-1} \right) f_l'(z_j^l)$$

$$\frac{\partial E}{\partial w_{jk}^l} = \frac{\partial E}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \Delta_j^l f_{l-1}(z_j^{l-1)}) = \Delta_j^l a_k^{l-1}$$

Backpropagation

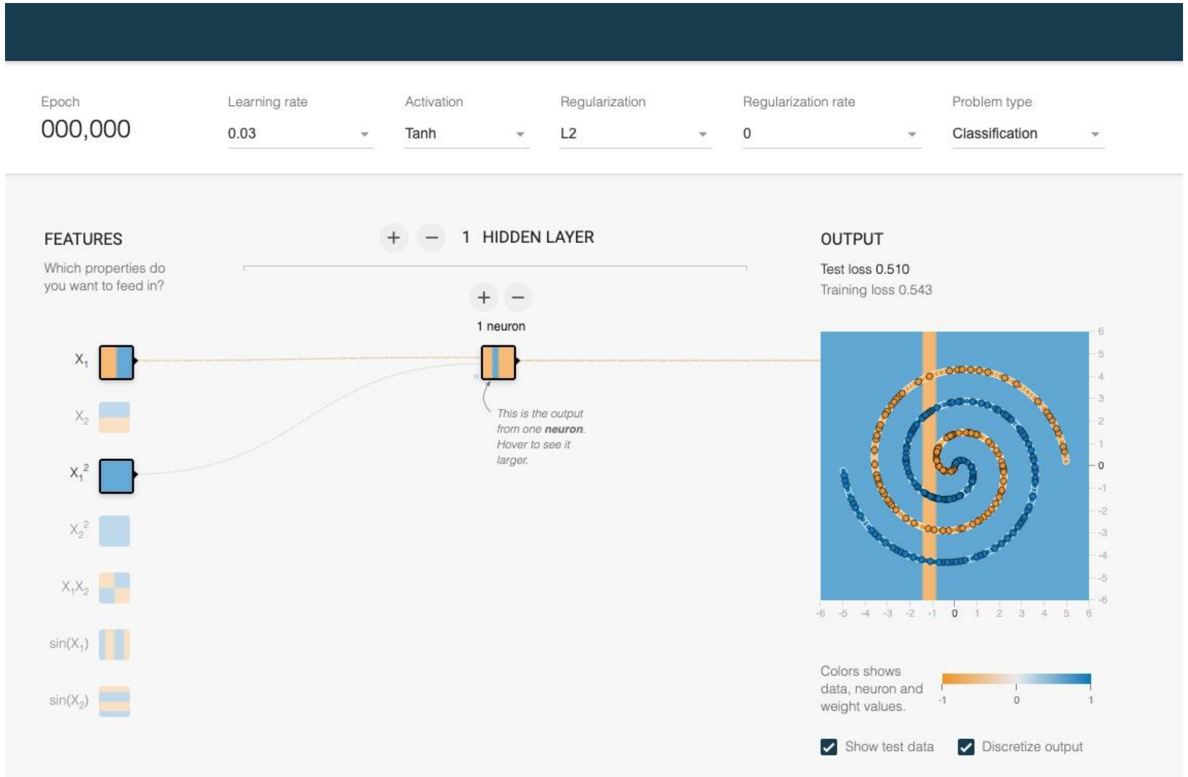
1. Activation at input layer
2. Feedforward calculation of zetas and activations
3. Error at top layer, only real derivative of the cost function
4. Backpropagate the error, i.e. gradients over zetas
5. Calculate gradient for weights and biases (using *batch-size samples)
6. Update weights and biases according to gradients
7. Repeat *number-of-epochs times

Pseudo-algorithm from A high-bias, low-variance introduction to Machine Learning for physicists. arXiv:1803.08823. Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre Day, Clint Richardson, Charles Fisher, David Schwab.

Tensorflow Playground

A neat example of how to build a feedforward neural network

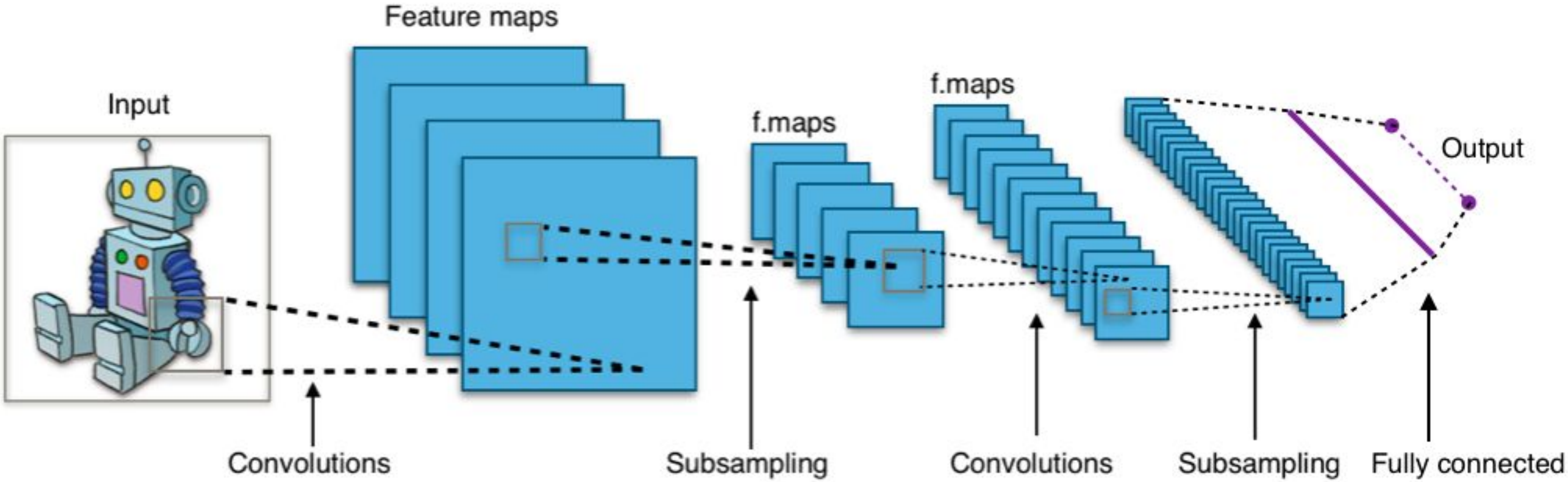
[Tensorflow Playground]



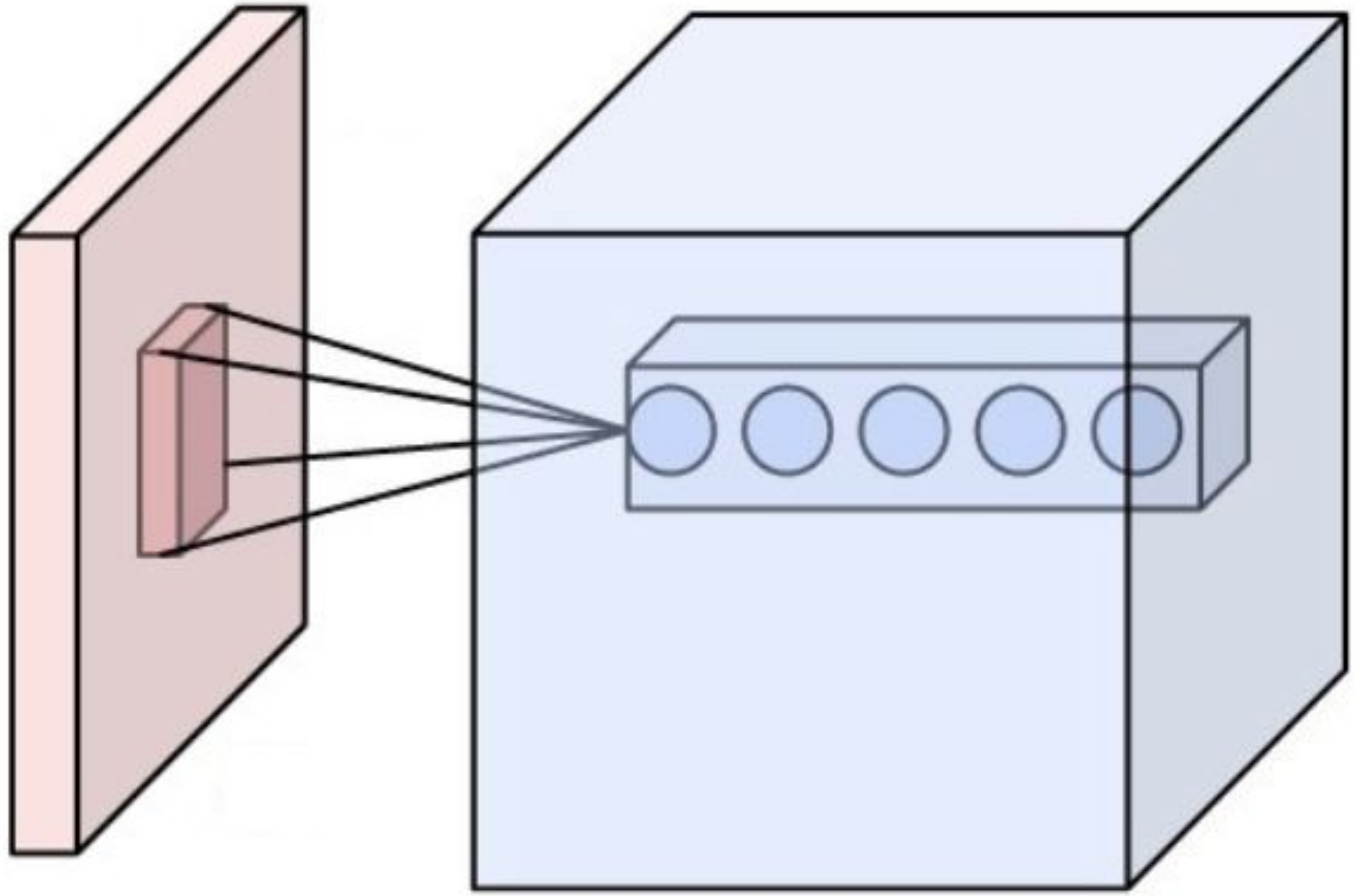
Types of Layers

- Dense layer
- Convolution layer
- Pool layer

Convolutional Neural Networks

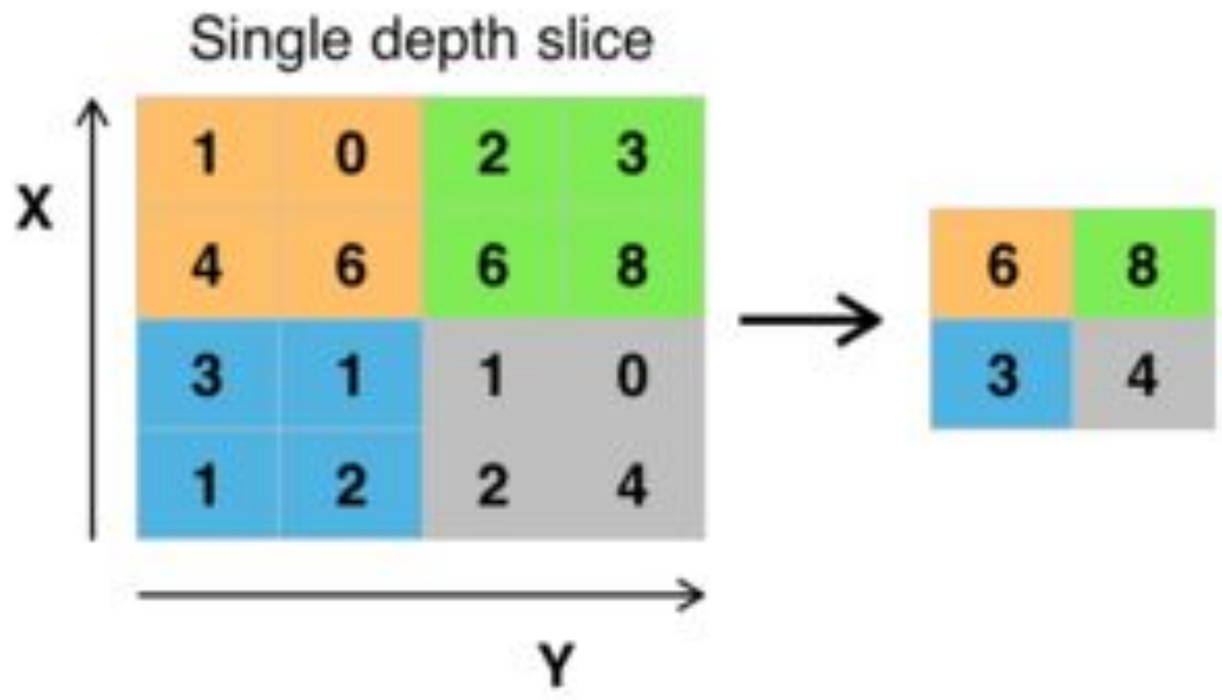


Convolution Layer



Each neuron has a small receptive field (a small local subset of the input)

Pool Layer



Each neuron has a small receptive field and performs a non-linear reduction of the input (e.g. max function)

Convolutional Neural Networks

Imagenet

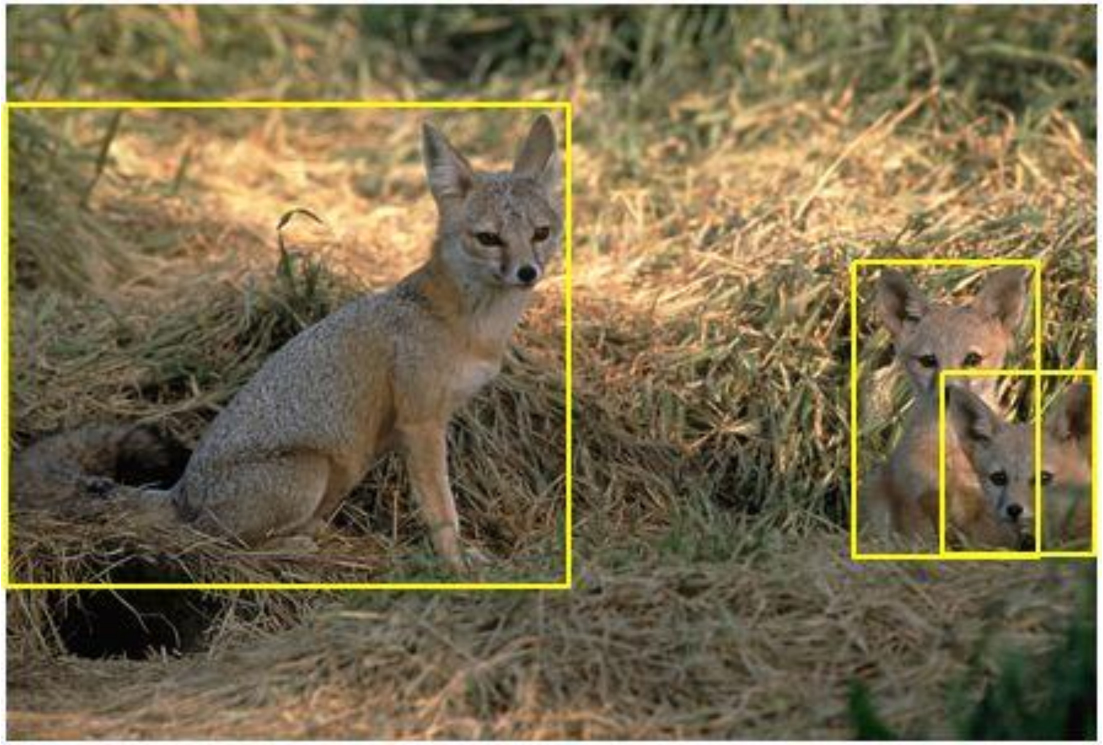
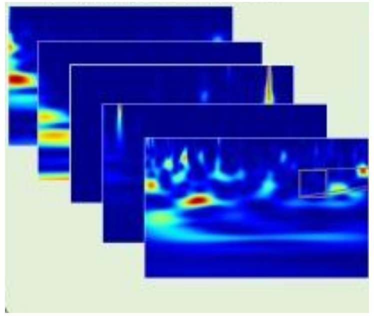


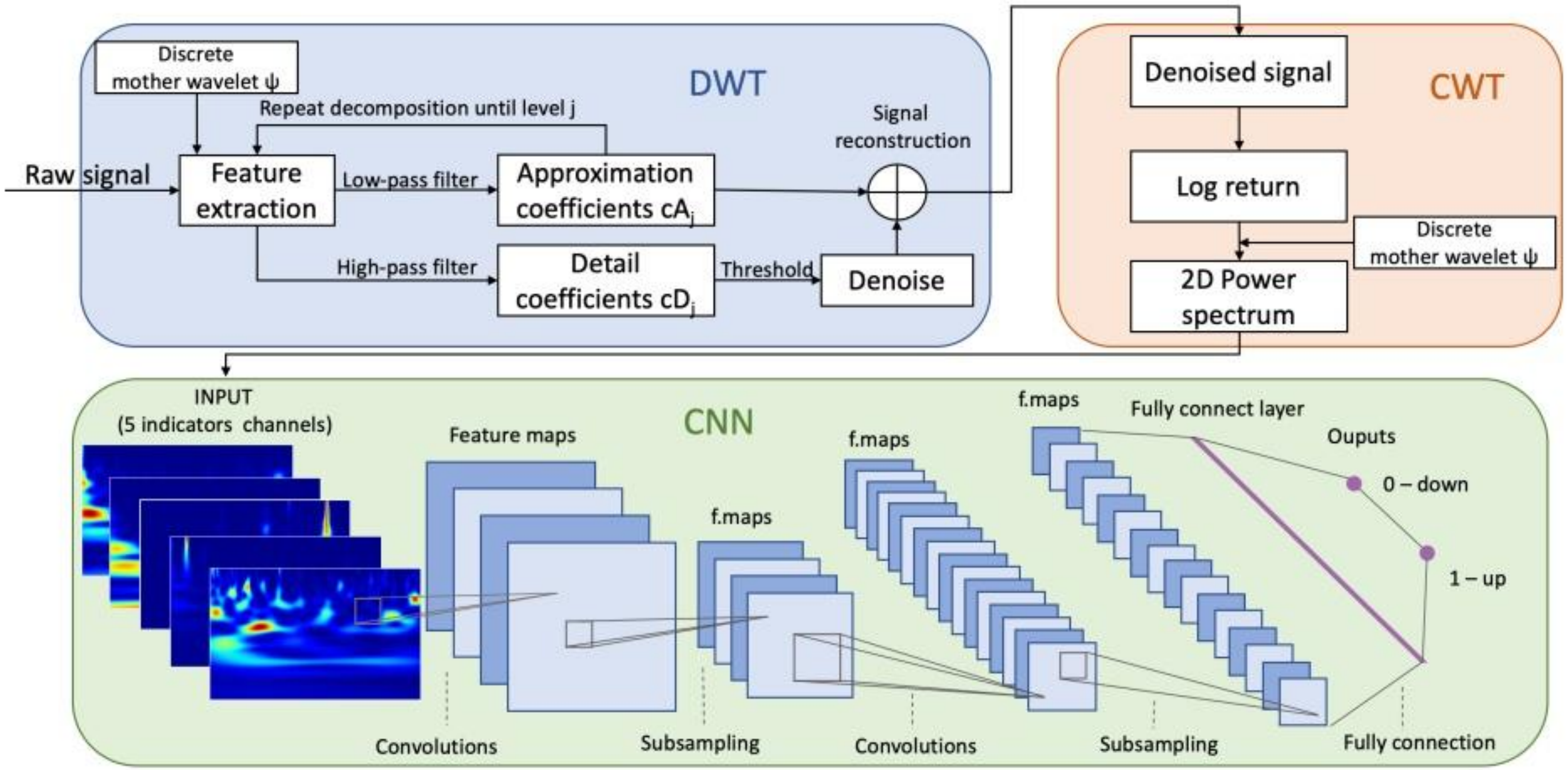
Figure from https://www.kaggle.com/c/imagenet-object-localization-challenge/overview/description

Wavelet Transform

$$[W_\psi f](a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \overline{\psi\left(\frac{x-b}{a}\right)} f(x)dx$$

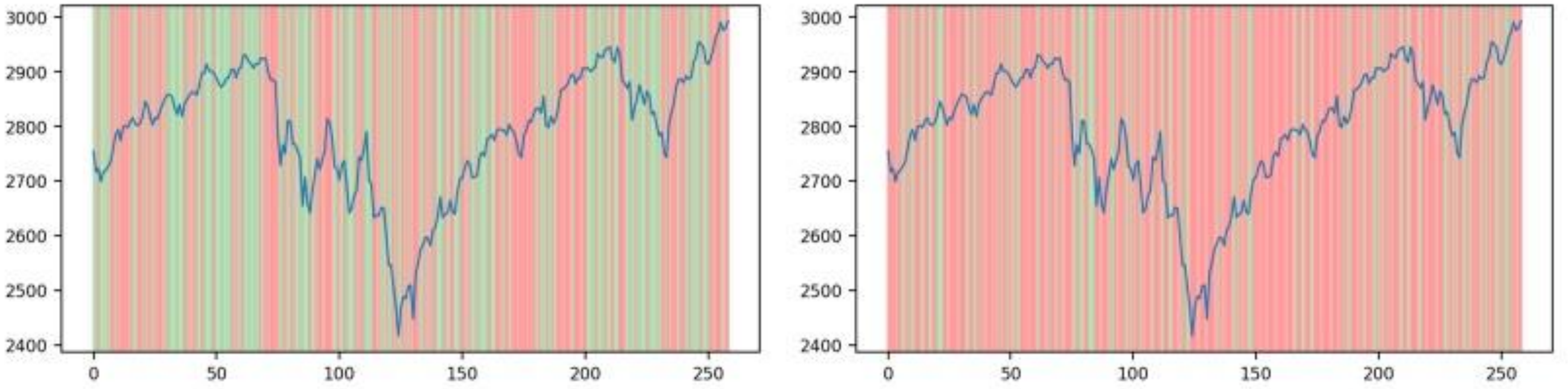


Wavelet Transforms and CNNs



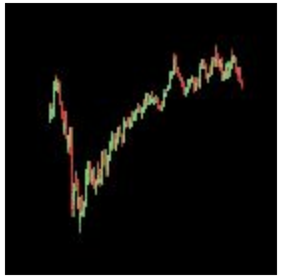
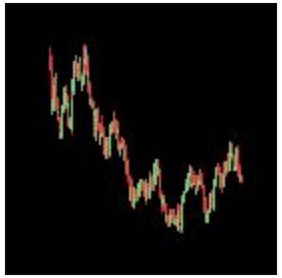
Figures from [Du, Bairui, Delmiro Fernandez-Reyes, and Paolo Barucca. "Image processing tools for financial time series classification." *arXiv preprint arXiv:2008.06042* (2020).]

Classifying market days



Figures from [Du, Bairui, Delmiro Fernandez-Reyes, and Paolo Barucca. "Image processing tools for financial time series classification." *arXiv preprint arXiv:2008.06042* (2020).]

Classifying candlesticks


Figures from Yicheng Wang COMP0162 2022/23 Project