

Deep Learning for Computer Vision

(Adjusted) slides from Hao Chen, PhD.
<https://chenhao2345.github.io/>

About me

Tomasz Stańczyk

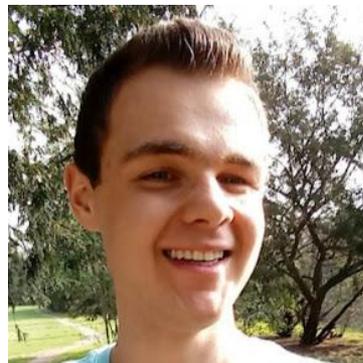
tomasz.stanczyk@inria.fr

I'm a PhD student at Inria Stars Team.

My research topic is “Deep Learning methods for human behavior monitoring”, which can be used in healthcare applications and video surveillance.

Week 4: Tracking!

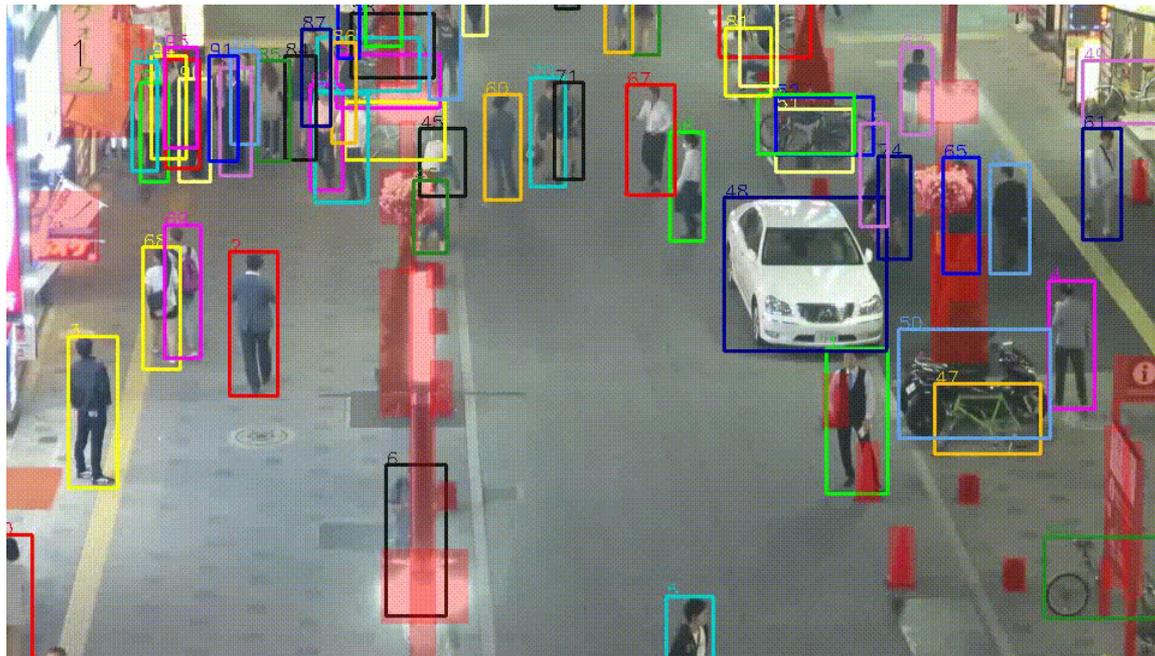
(slides still to be uploaded)



Long-term multi-object tracking

MOT17 dataset (MOT17-04-SDP)

<https://motchallenge.net/vis/MOT17-04-SDP/gt/>



Lecture 1: One Artificial Neuron

Objective

In the first lecture, you will see

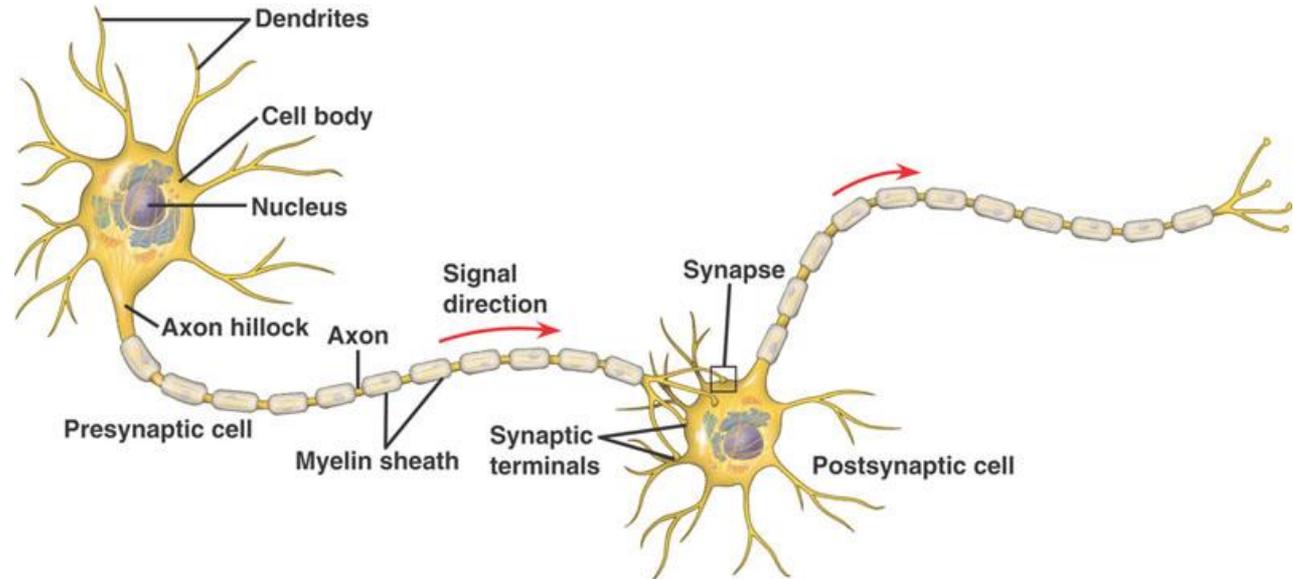
- The difference between a biological neuron and an artificial neuron
- Components of an artificial neuron
- How an artificial neural network works

Outline of Lecture 1

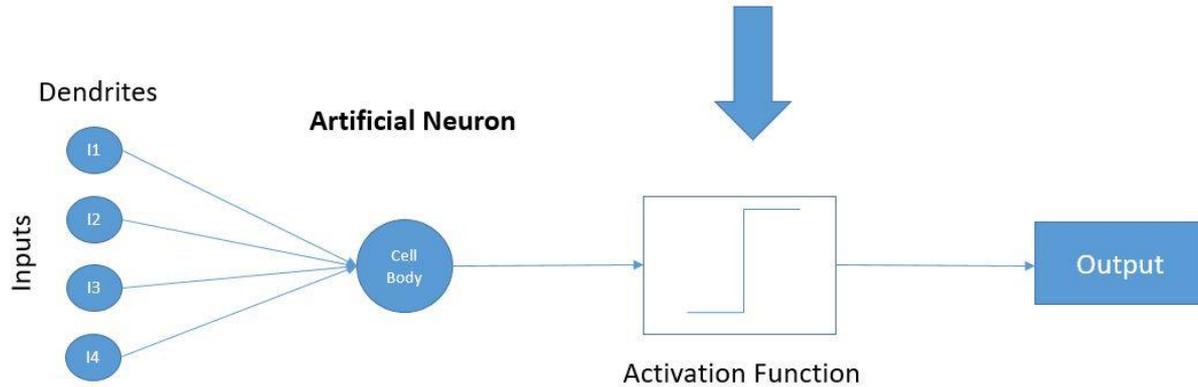
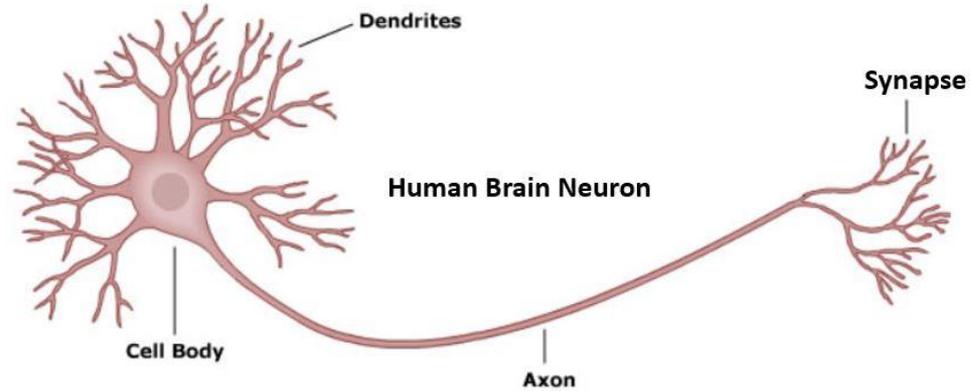
- Artificial neuron
- Logistic regression
- Cost function
- Gradient descent

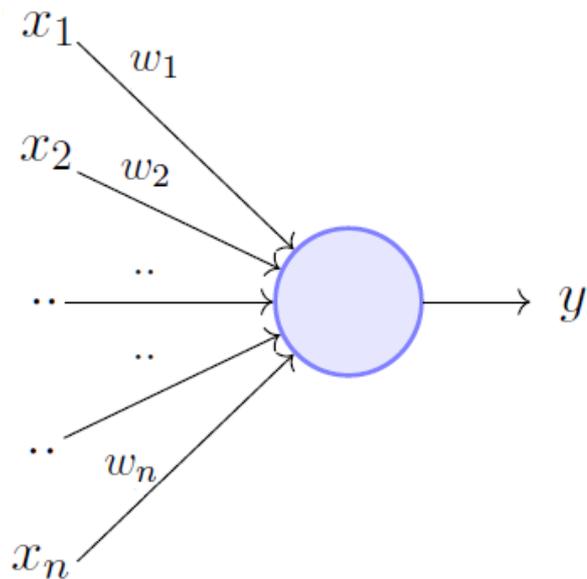
Neuron

Human brain is made of approximately 100 billion nerve cells, called **neurons**. Each neuron is connected to other neurons. Neurons gather and transmit electrochemical signals to send messages to each other.



Artificial Neuron





$$y = 1 \quad \text{if } \sum_{i=1}^n w_i * x_i \geq \theta$$
$$= 0 \quad \text{if } \sum_{i=1}^n w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad \text{if } \sum_{i=1}^n w_i * x_i - \theta \geq 0$$
$$= 0 \quad \text{if } \sum_{i=1}^n w_i * x_i - \theta < 0$$

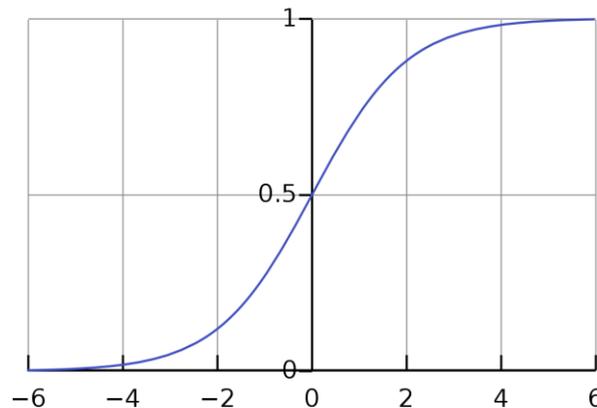
θ is the activation threshold.

In the previous example, the activation function is binary step function (also called Heaviside)

$$\sigma(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Because it's not continuous at 0, in practice, we usually use sigmoid function (also called logistic)

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Reminder: image matrix

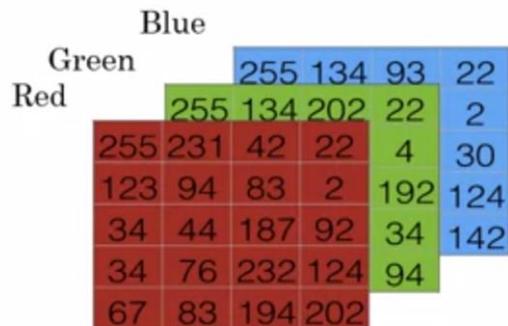


For image # i

$$y^{(i)} \in \{0, 1\}$$

For m training samples

$$\mathbf{y} = [y^{(1)} \quad \dots \quad y^{(m)}]$$



$$x^{(i)} = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 142 \end{bmatrix}$$

$$\mathbf{x} = [x^{(1)} \quad \dots \quad x^{(m)}] = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & \dots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,m} \end{bmatrix}$$

Recap

1. Each neuron can be regarded as a linear function.

$$\mathbf{z} = \boldsymbol{\omega}^T \mathbf{x} + b$$

1. Activation function make it possible to learn non-linear complex functional mappings. They introduce non-linear properties to our Network, which is important for solving complex visual tasks.

$$\mathbf{y} = \sigma(\mathbf{z}) = \sigma(\boldsymbol{\omega}^T \mathbf{x} + b)$$

Logistic Regression with Cost Function

From previous slide, we have a logistic regression model.

$$\hat{\mathbf{y}} = \sigma(\boldsymbol{\omega}^T \mathbf{x} + b), \text{ where } \sigma(x) = \frac{1}{1+e^{-x}}$$

Given m samples,

$$\mathbf{x} = [\mathbf{x}^{(1)} \quad \dots \quad \mathbf{x}^{(m)}]$$

$$\mathbf{y} = [\mathbf{y}^{(1)} \quad \dots \quad \mathbf{y}^{(m)}]$$

We want to minimize the distance between the prediction and the ground truth,

$$\hat{\mathbf{y}}^{(i)} \approx \mathbf{y}^{(i)}$$

Logistic Regression with Cost Function

Define the distance with a loss function, for example, one half a square error

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

However, people don't usually use this to learn parameters. In practice, we usually use the cross entropy loss to maximize the likelihood of classifying the input data correctly.

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y}))$$

The cost function on the m samples will be

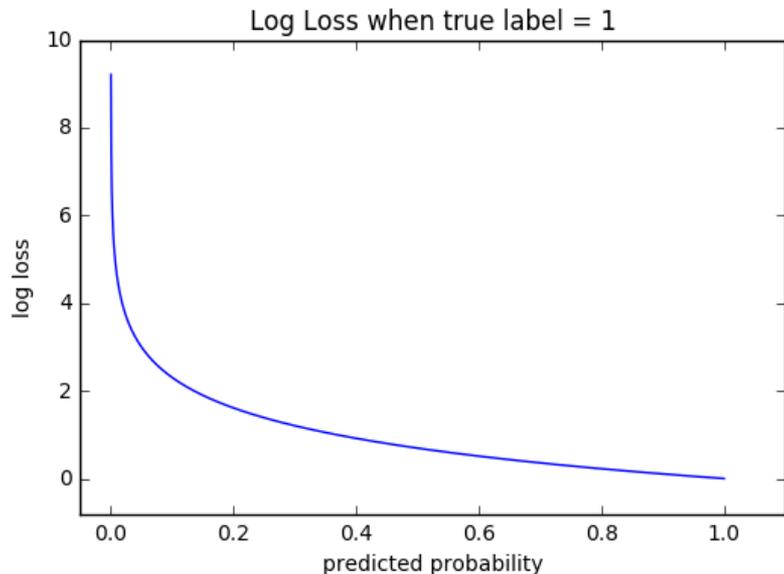
$$J(\omega, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}_i, y_i)$$

Cross Entropy

Let's look deeper into the cross entropy loss.

$$L(\hat{y}, y) = \begin{cases} -\log(1 - \hat{y}), & \text{if } y = 0 \\ -\log\hat{y}, & \text{if } y = 1 \end{cases}$$

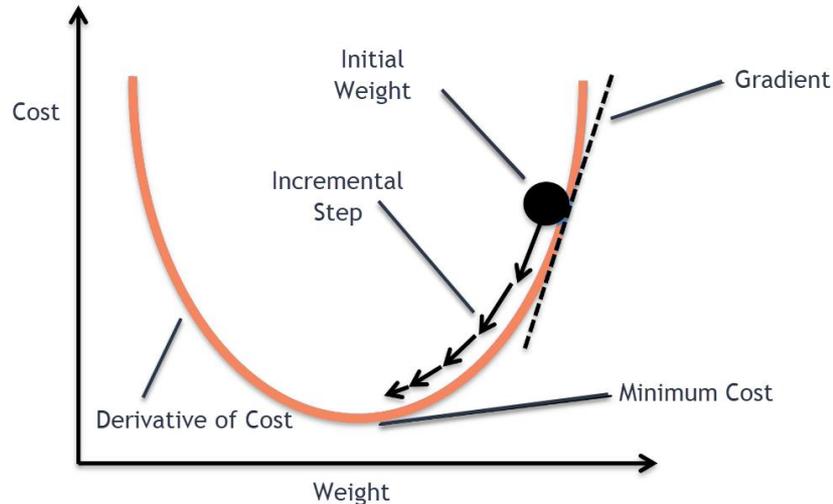
When your prediction get further to the true label, your loss will grow exponentially.



Gradient Descent

Our objective is to find ω , b that minimize

$$J(\omega, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}, y)$$



$$\text{Repeat } \begin{cases} \omega = \omega - \alpha \frac{\partial J(\omega, b)}{\partial \omega} \\ b = b - \alpha \frac{\partial J(\omega, b)}{\partial b} \end{cases}$$

Reminder: $\frac{\partial J(\omega, b)}{\partial \omega}$ is the partial derivative of the cost with respect to ω , which gives the **slope** of the **tangent line** to the **graph of the function** at that point.

Learning rate decay

In the previous slide, α is called learning rate. To find the minima, we need to gradually decrease the learning rate.

For example,

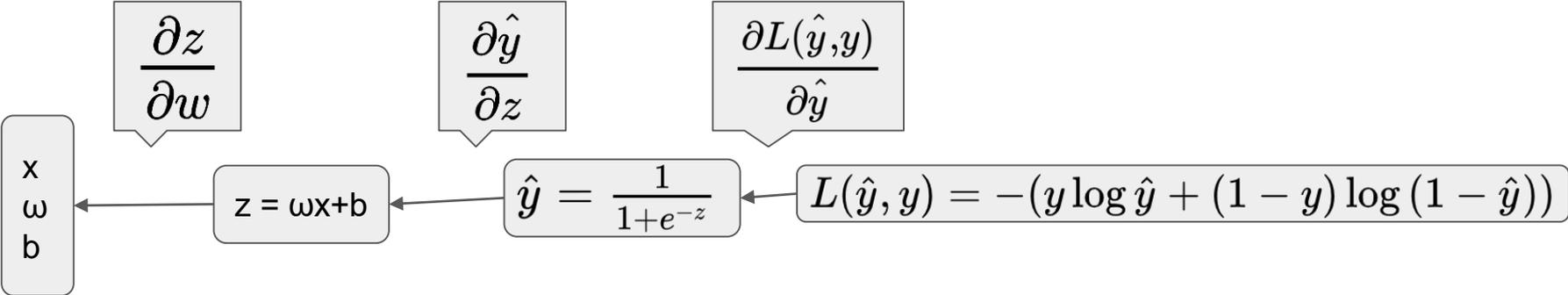
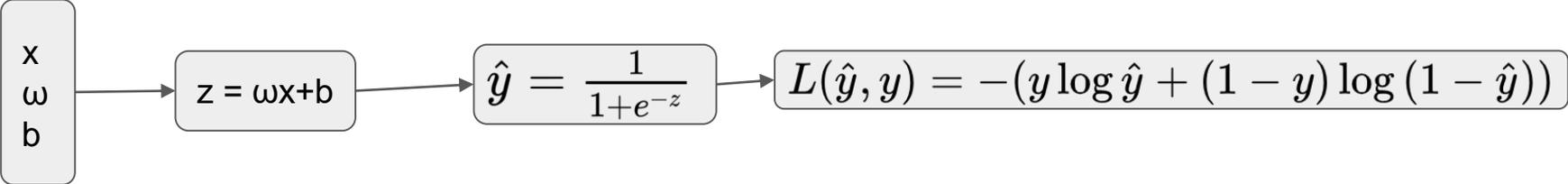
1. Multi-step learning rate

$$\alpha = \begin{cases} \alpha_0 & \text{if } \#epoch < 100 \\ 0.1 * \alpha_0 & \text{if } 100 < \#epoch < 200 \\ 0.01 * \alpha_0 & \text{if } 200 < \#epoch < 300 \end{cases}$$

1. Exponential learning rate

$$\alpha = \alpha_0 * decay_rate^{\#epoch}$$

Let's put them together



ω get updated $\longrightarrow \omega = \omega - \alpha \frac{\partial L(\omega, b)}{\partial \omega}$

Derivative results

$$\frac{\partial L}{\partial \hat{y}} = \frac{\hat{y} - y}{\hat{y}(1 - \hat{y})}$$

$$\frac{\partial \hat{y}}{\partial z} = \hat{y}(1 - \hat{y})$$

$$\frac{\partial z}{\partial w} = x$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w} = x(\hat{y} - y)$$

Conclusion

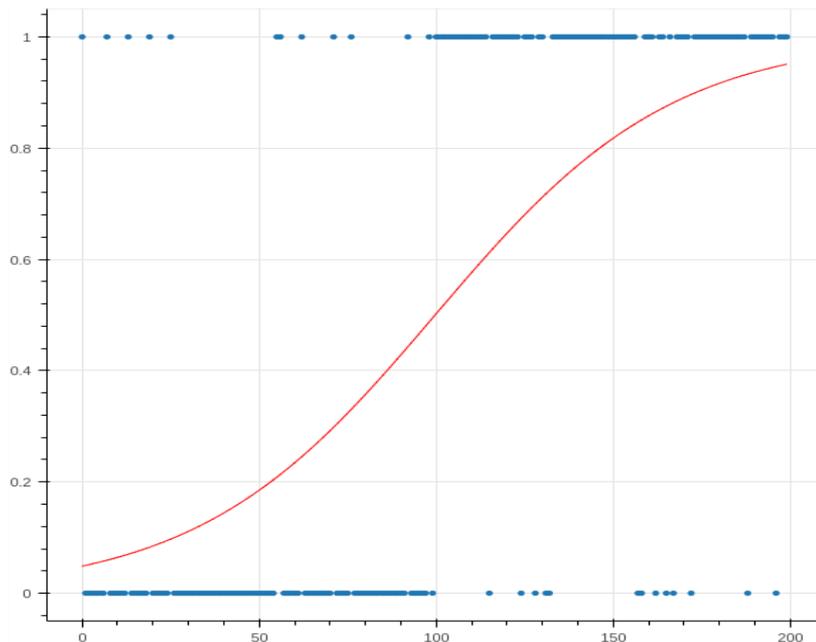
1. An artificial neuron is composed by a linear transformation and an activation function.
2. To adjust parameters in an artificial neuron, we need to define a cost function. By decreasing the cost function with gradient descent, the parameters get updated step by step.

Practice-1

Estimate coefficients for a step function with logistic regression. You will need to implement:

1. Sigmoid function
2. Cross-entropy loss
3. Gradient for backpropagation

Go to: <https://tinyurl.com/yckvzfnh>
and make a copy of the notebook.



Extra (strongly recommended)

1. How to read a (machine learning) scientific paper by Andrew Ng.

<https://www.youtube.com/watch?v=733m6qBH-jl>

(<https://tinyurl.com/2p84mj48>)

From 0:59 to 28:51, yet in fact, the whole video might be of interest to you.

Handy summary after watching: <https://forums.fast.ai/t/how-to-read-research-papers-andrew-ng/66892>

(<https://tinyurl.com/2s4mc3eu>)

1. Deep Learning with PyTorch: A 60 Minute Blitz:

https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

(<https://tinyurl.com/bddzwcka>)