Projects

Bálint Ármin Pataki

Technical info

- **Kaggle challange**, Happiness detection:
  - 21 participants
  - Deadline will be extended

Agenda

- Reminder
  - Fully connected neural networks
  - Gradient descent
  - Training process
  - Convolutional neural networks
  - Residual neural networks
- Project ideas

Fully connected neural network

$x \in \mathbb{R}^N, y \in \mathbb{R}^K$, neural network: $\mathbb{R}^N \rightarrow \mathbb{R}^K$

$$z^{[1]} = W^{[1]}x + b^{[1]}, \quad W: n^{[1]} \times N, \quad b: n^{[1]} \times 1$$
$$a^{[1]} = g\left(z^{[1]}\right)$$

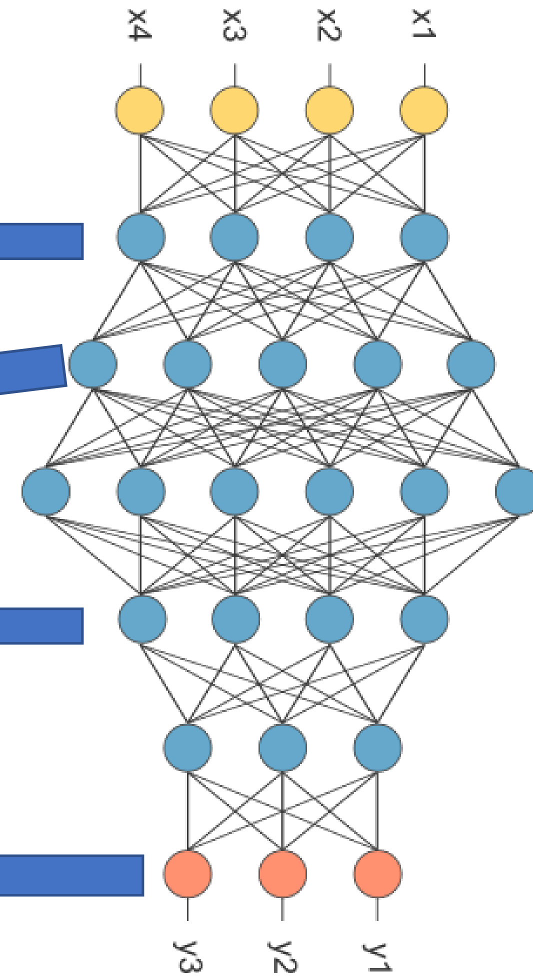
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}, \quad W: n^{[2]} \times n^{[1]}, \quad b: n^{[2]} \times 1$$
$$a^{[2]} = g\left(z^{[2]}\right)$$

$\vdots$

$$z^{[i]} = W^{[i]}a^{[i-1]} + b^{[i]}, \quad W: n^{[i]} \times n^{[i-1]}, \quad b: n^{[i]} \times 1$$
$$a^{[i]} = g\left(z^{[i]}\right)$$

$\vdots$

$$z^{[L]} = W^{[L]}a^{[L-1]} + b^{[L]}, \quad W: n^{[L]} \times n^{[L-1]}, \quad b: n^{[L]} \times 1$$
$$y = a^{[L]} = softmax\left(z^{[L]}\right)$$

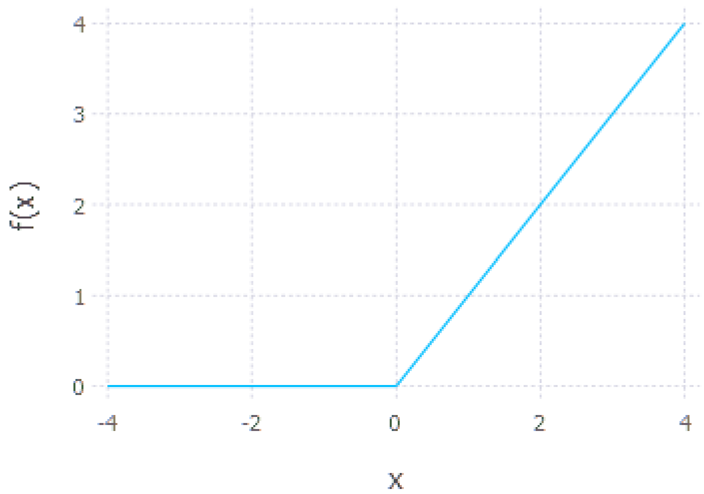


Credit: OpenNN

Activation function

- non-linear
  - else: whole networks is just a matrix product

- ReLu Rectified Linear Unit
  - $g(z) = \max(0, z)$



- softmax
  - $g(z) = \frac{e^{z_0}}{\sum_{j=0}^{K} e^{z_j}}$

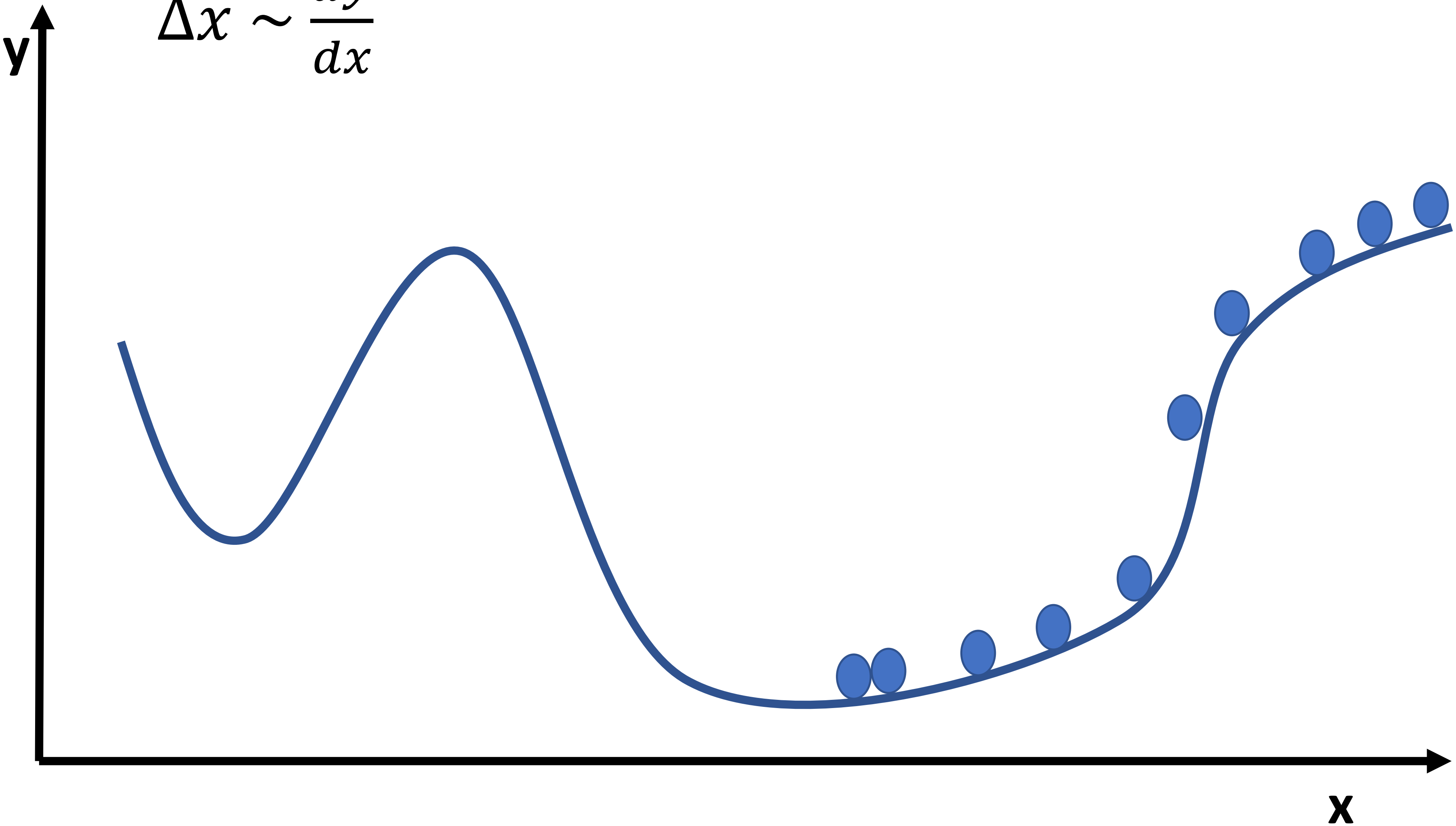
Loss function

- Cross-entropy loss
  - $L = -\frac{1}{M}\sum_i y_i \cdot \log(y_{pred_i})$

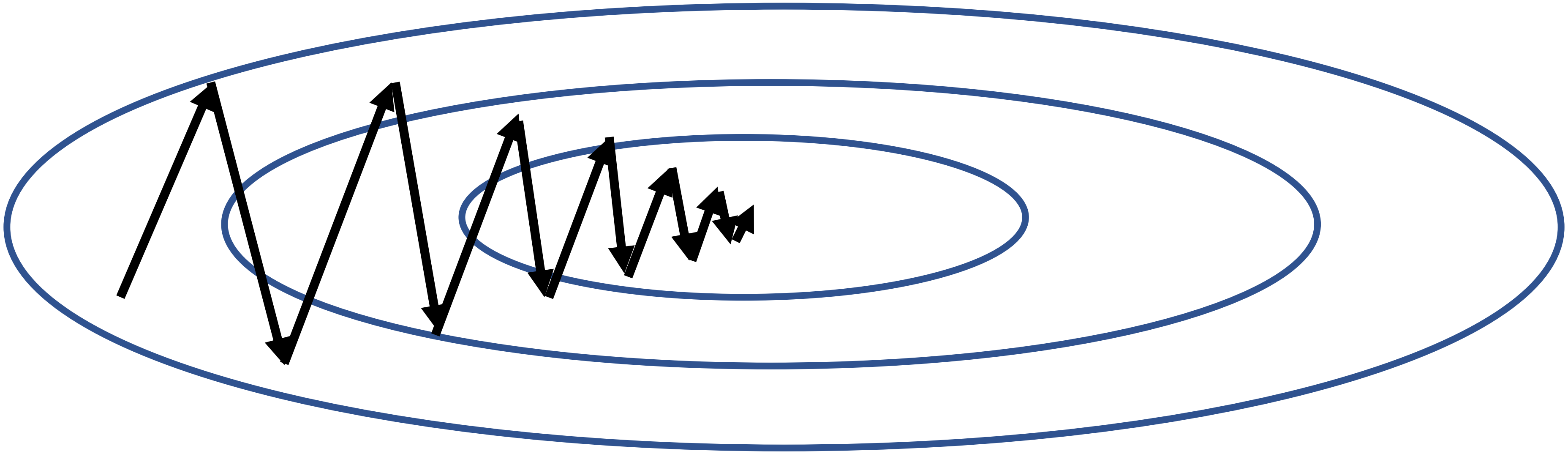
- Mean squared error:
  - $L = \frac{1}{M}\sum_i \left(y_i - y_{pred_i}\right)^2$

Step size in x is proportional to the derivate.

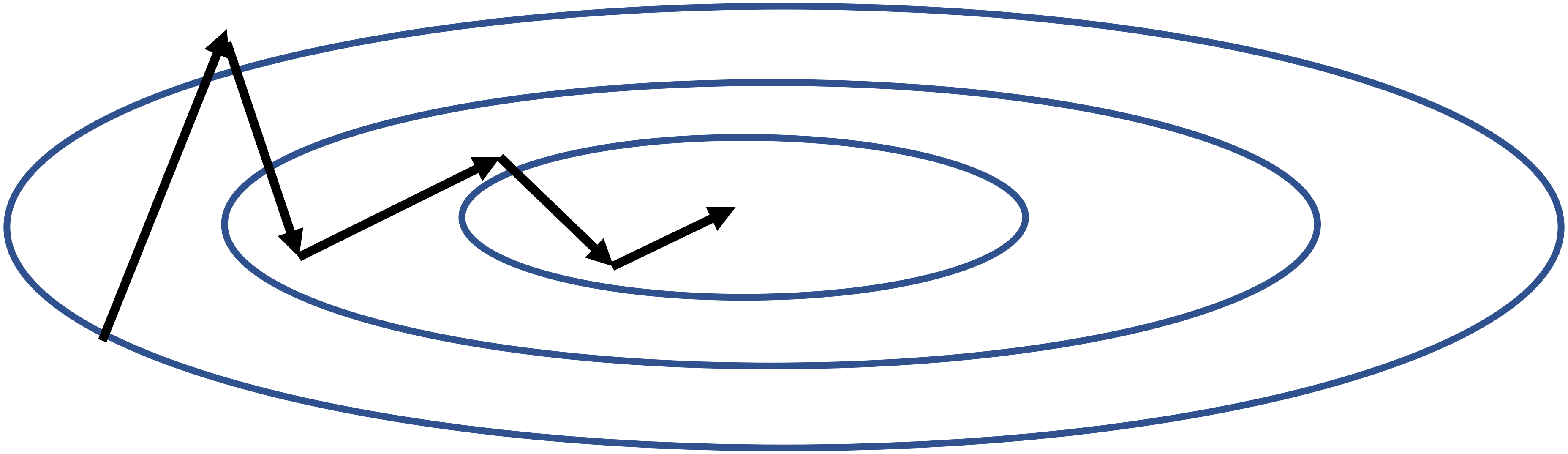
$$\Delta x \sim \frac{dy}{dx}$$



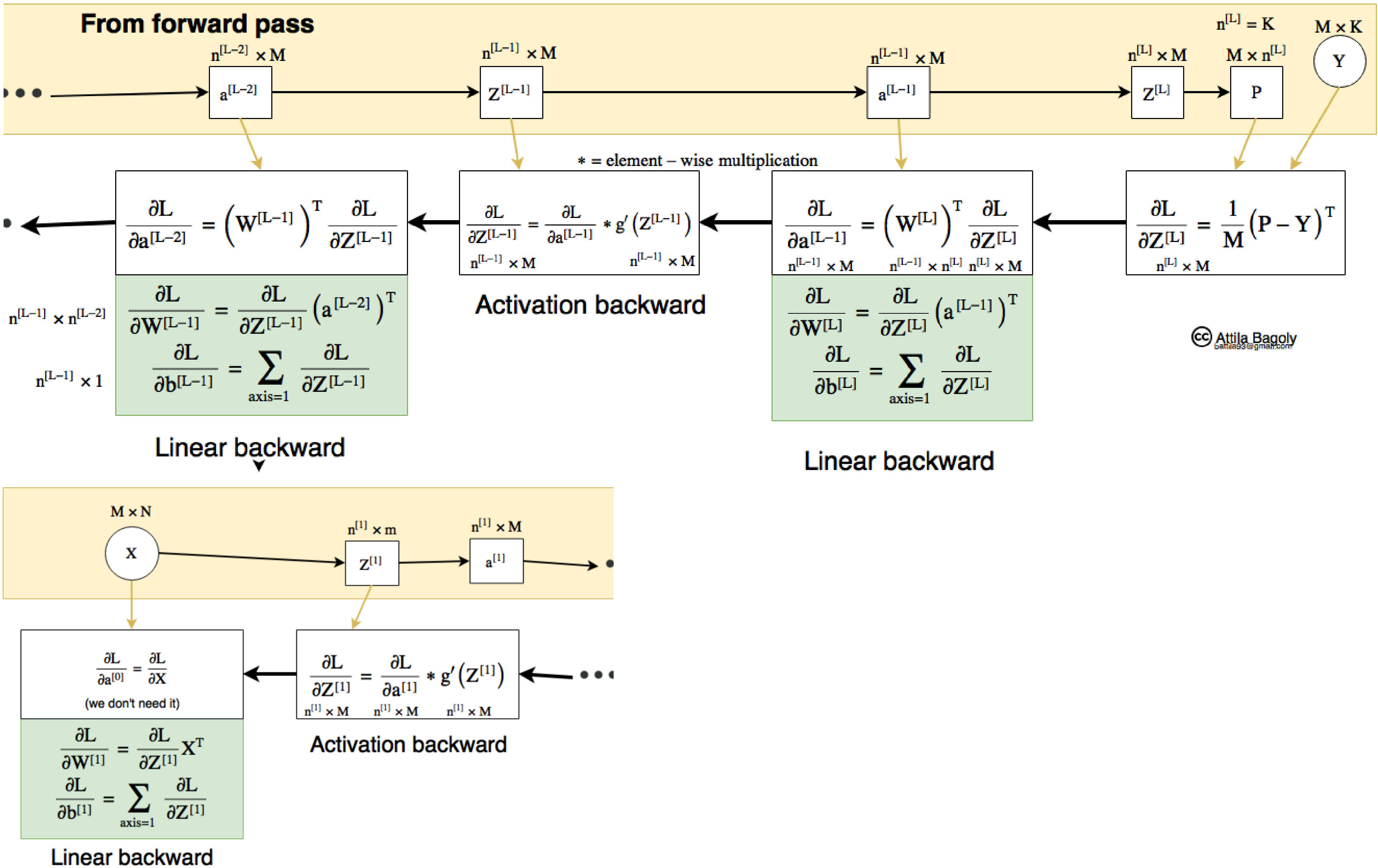
Gradient descent



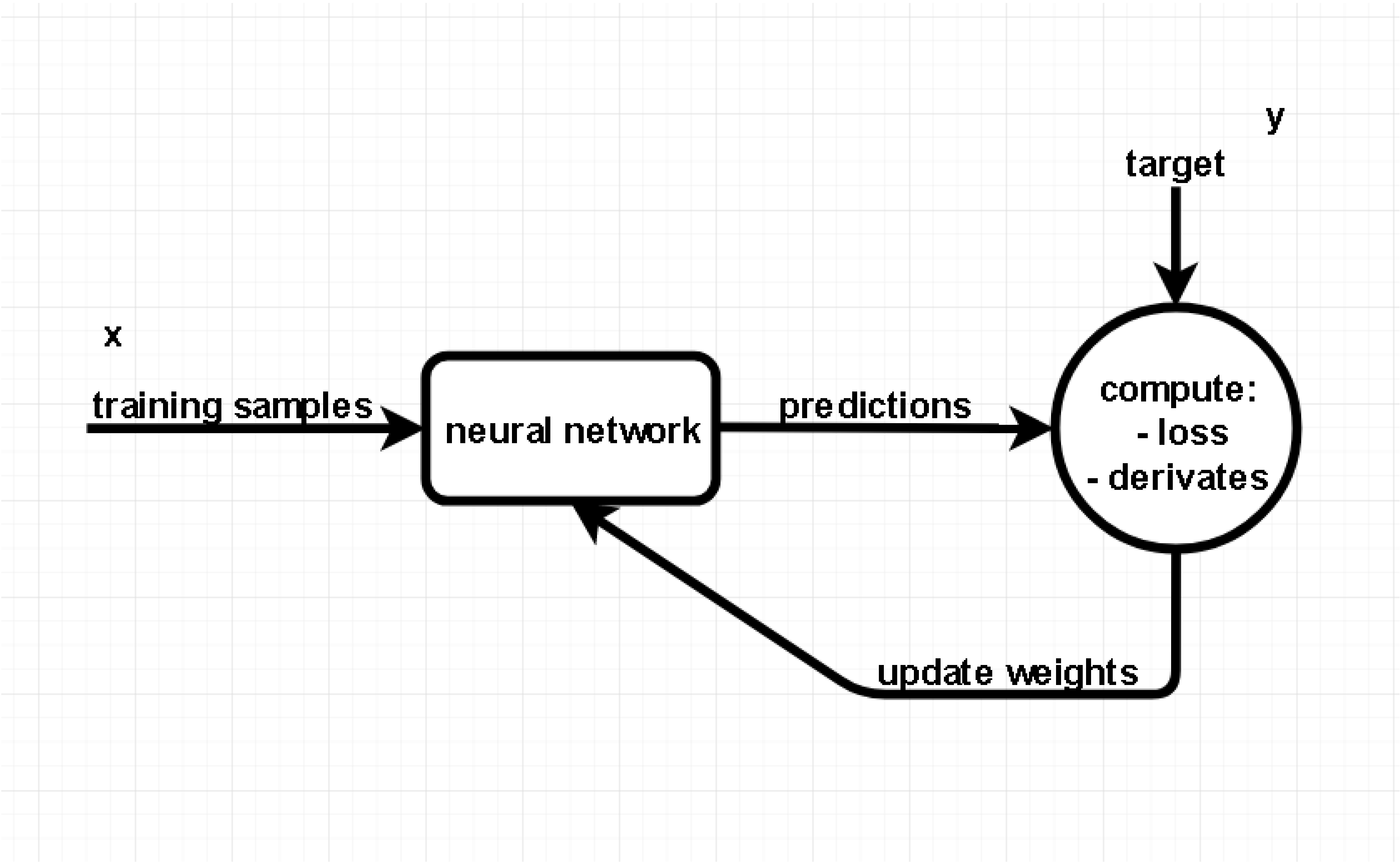
Gradient descent – with momentum



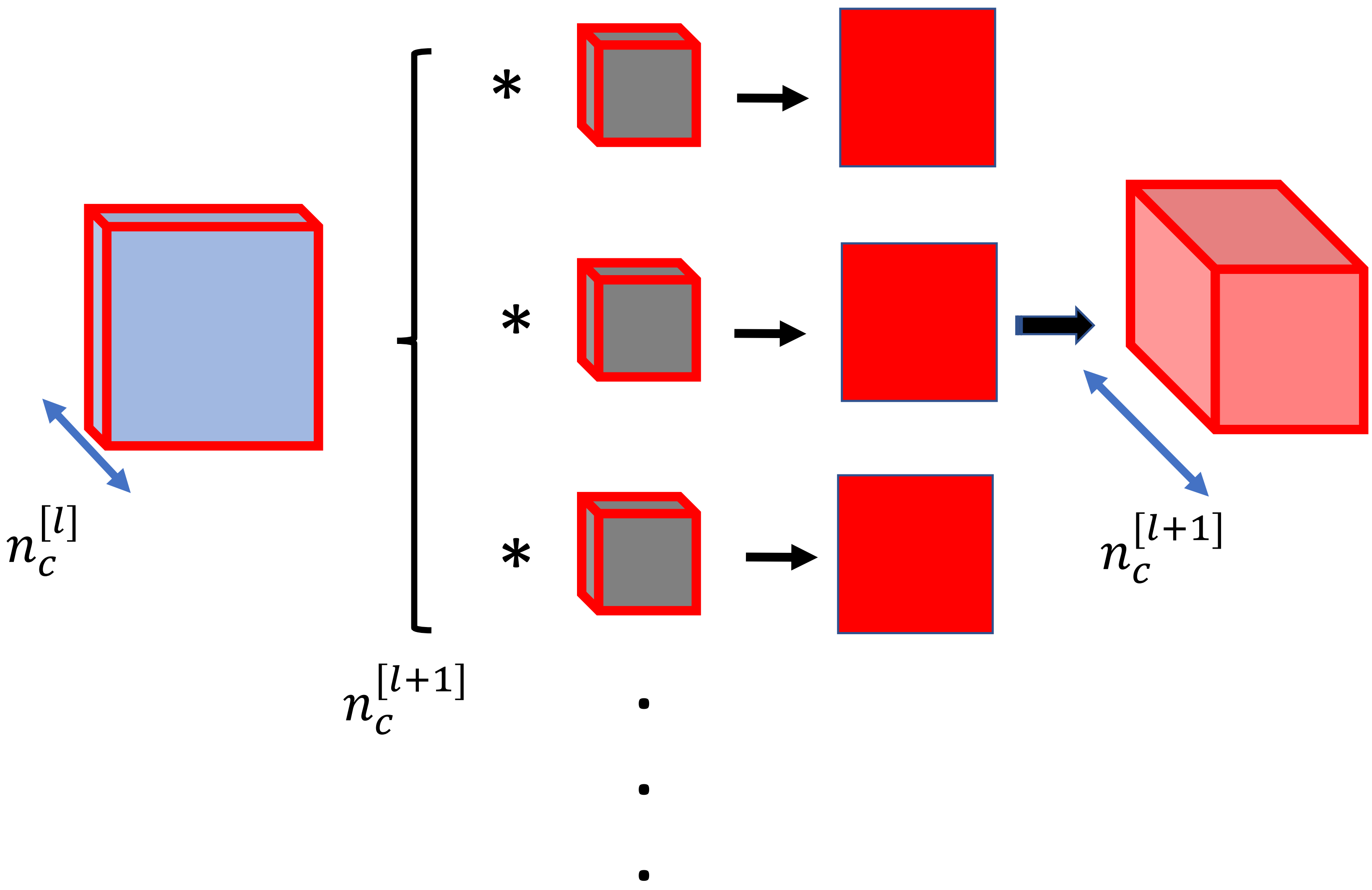
Backpropagation



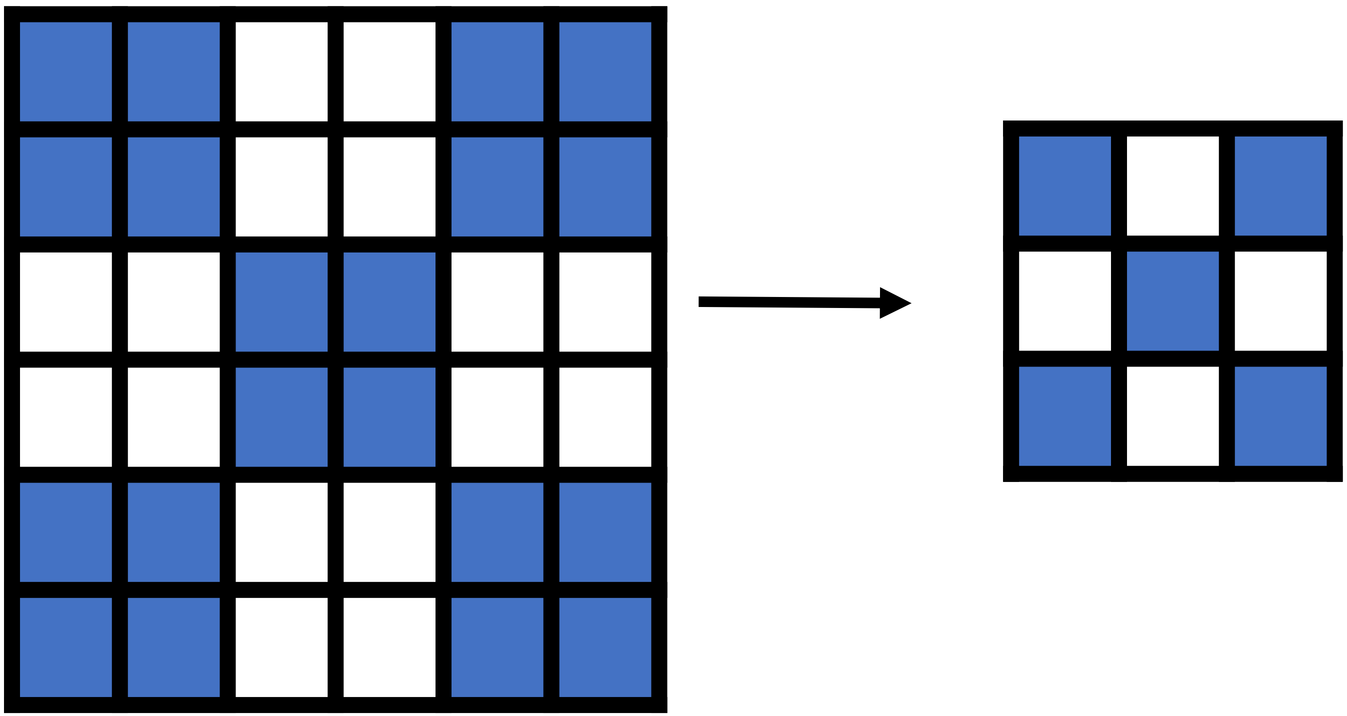
Training process



Convolution

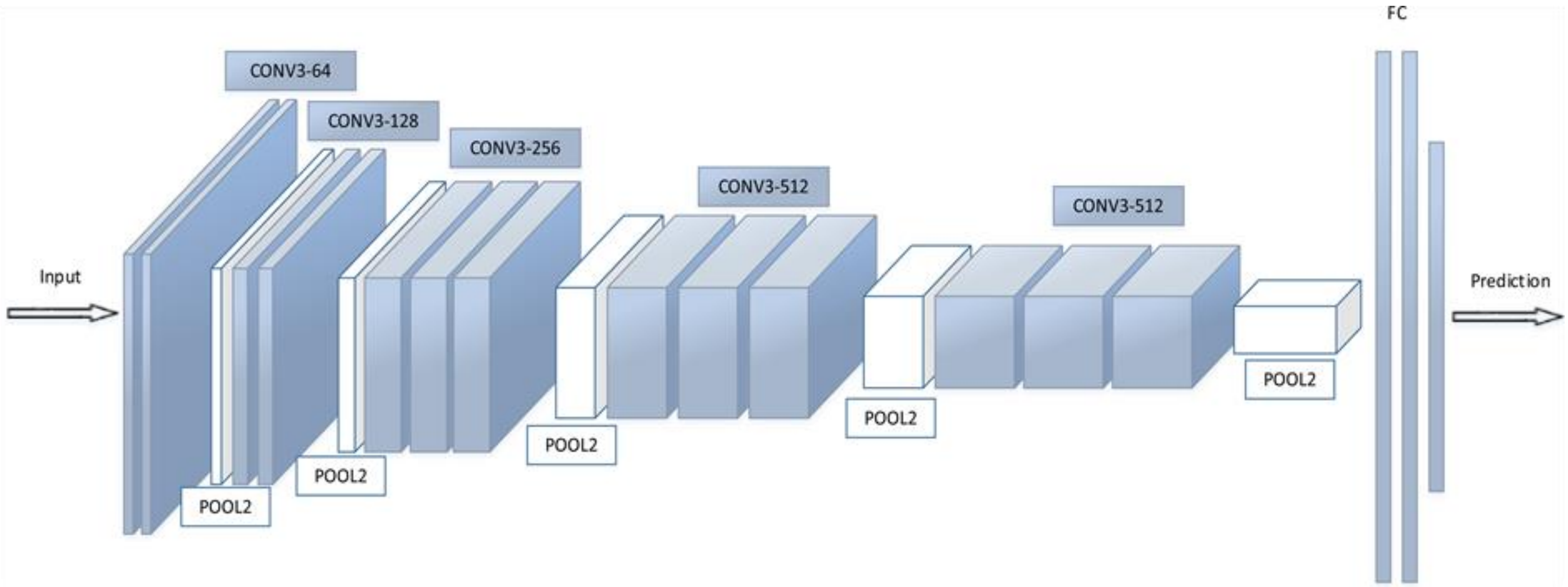


Done for each channel separately!



Take the max for each 2x2 window → keep only that value

Convolutional neural network



[http://file.scirp.org/Html/4-7800353_65406.htm]

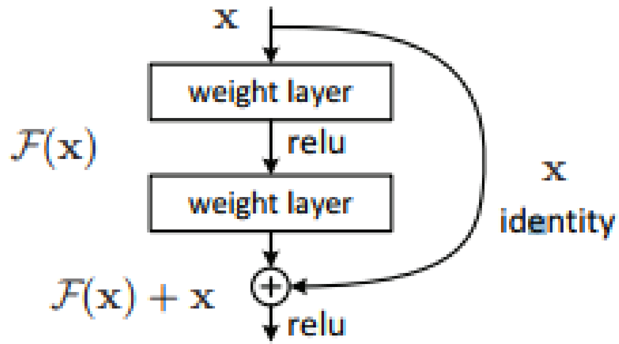
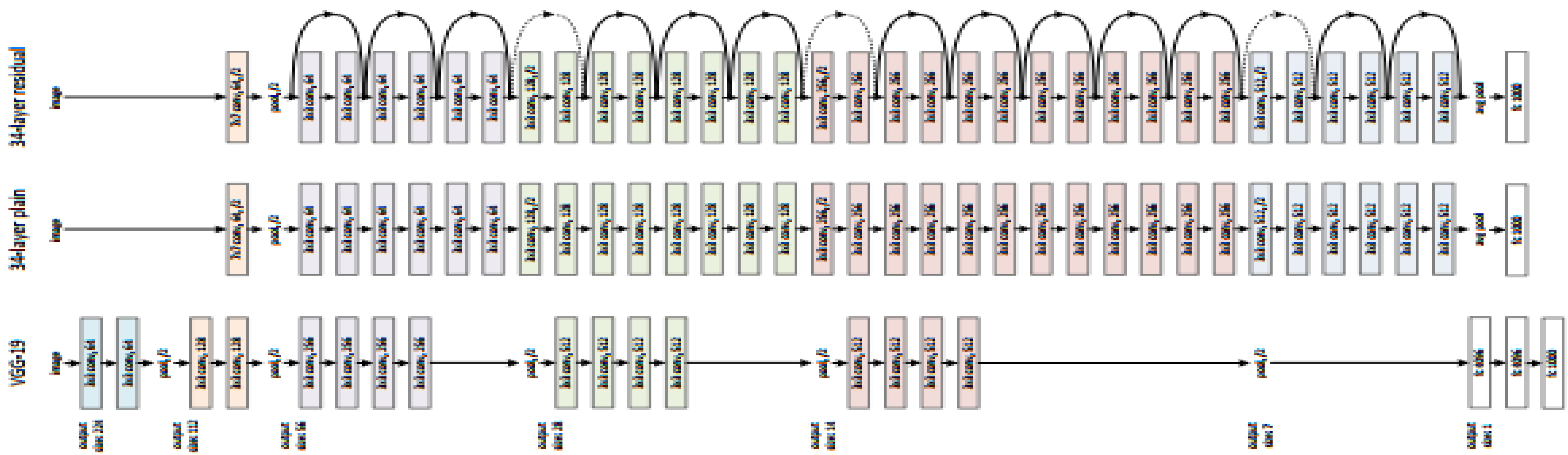


Figure 2. Residual learning: a building block.



[Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: Deep Residual Learning for Image Recognition]

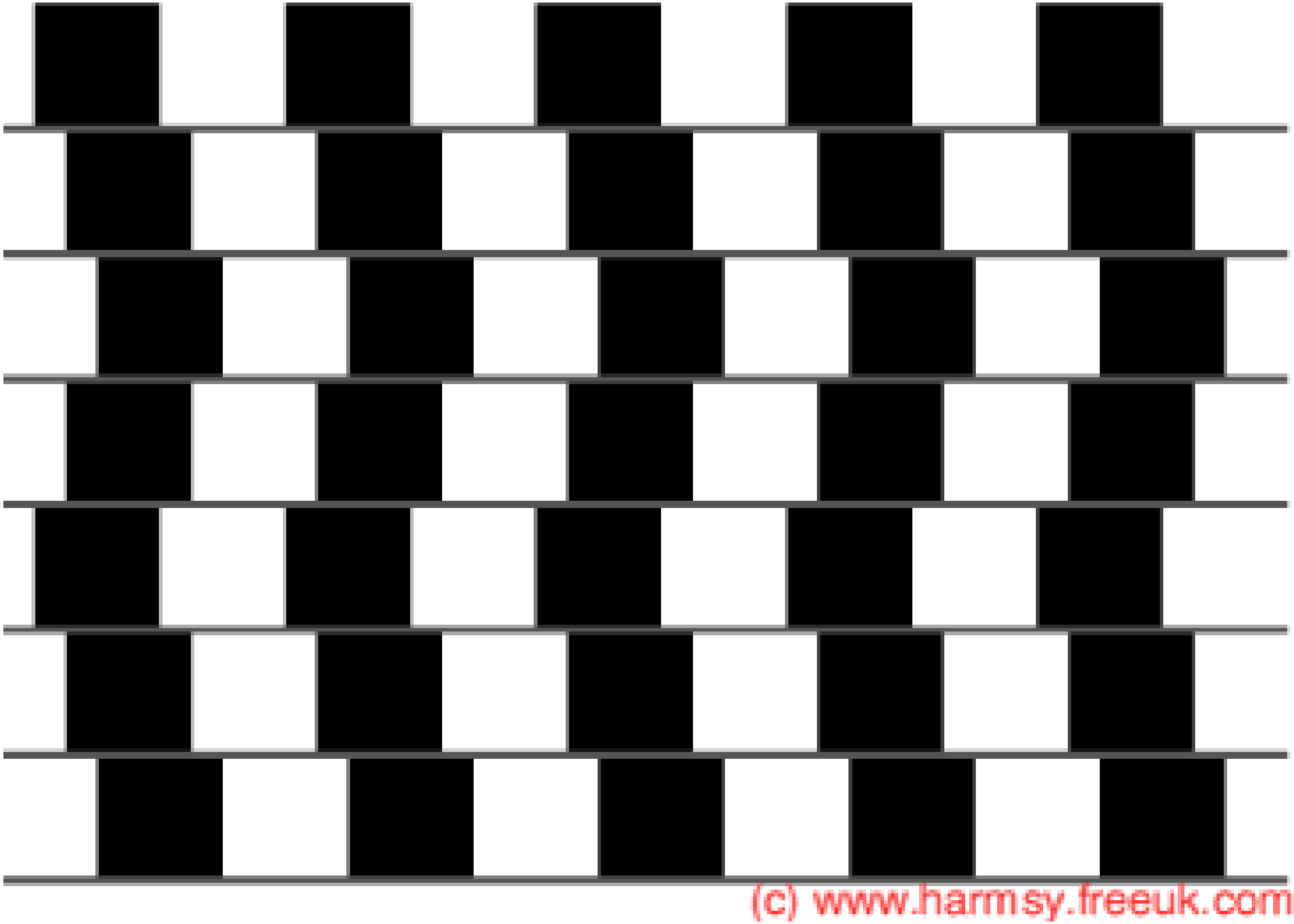
Project ideas

- **Adverserial pictures**
  - how to fool a neural network
- **Face recognition/verification**
  - who is on the picture?
  - 2 picture contains the same person or not
- **Style transfer**
  - paintings
  - voice
- **Weather forecast:**
  - weather radar images → rain/wind forecast
- **Object localisation**
  - Crop galaxies from an image
- **Instance segmentation**
  - cells for science / cars to remove background
- **Visualisation of inner layers**
  - to understand what's going on inside
- **Natural language processing**
  - add emojis to sentences
  - speech to text

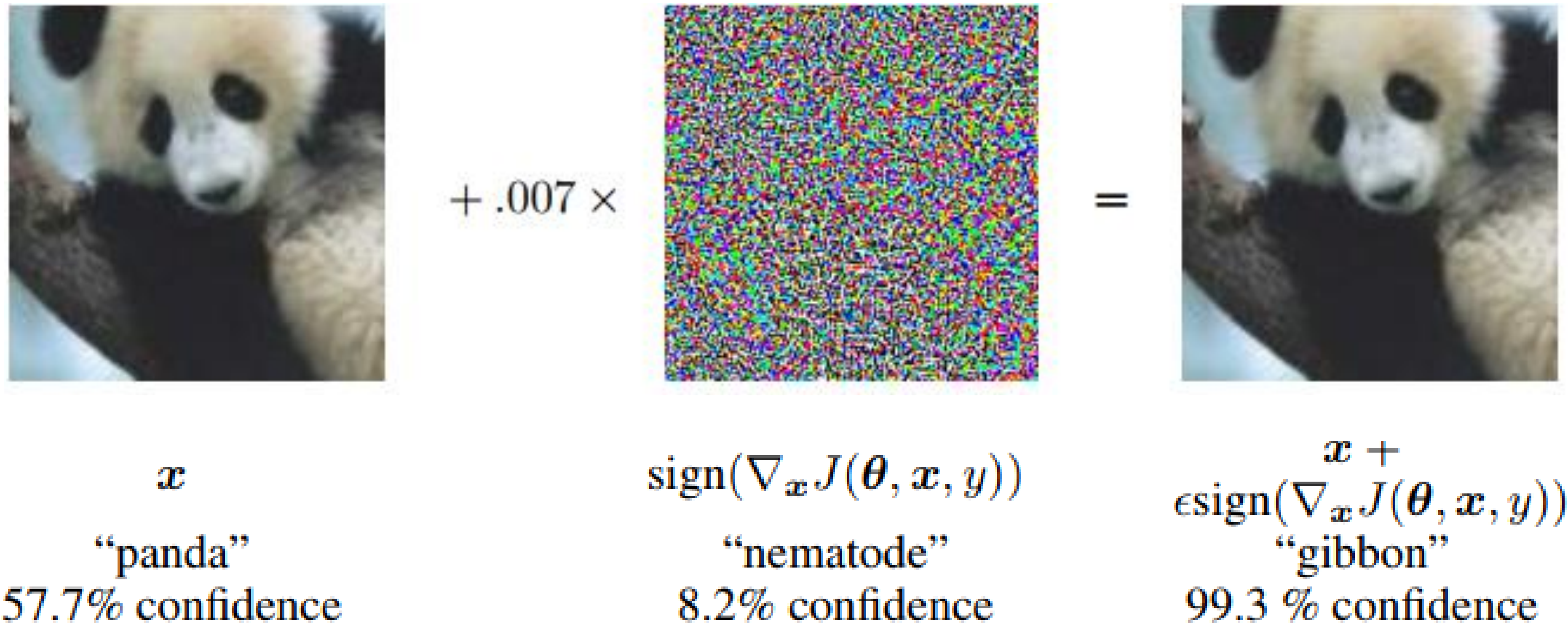
Project steps:

- idea
- check if there is data
- modeling
- documentation

Adverserial images for humans



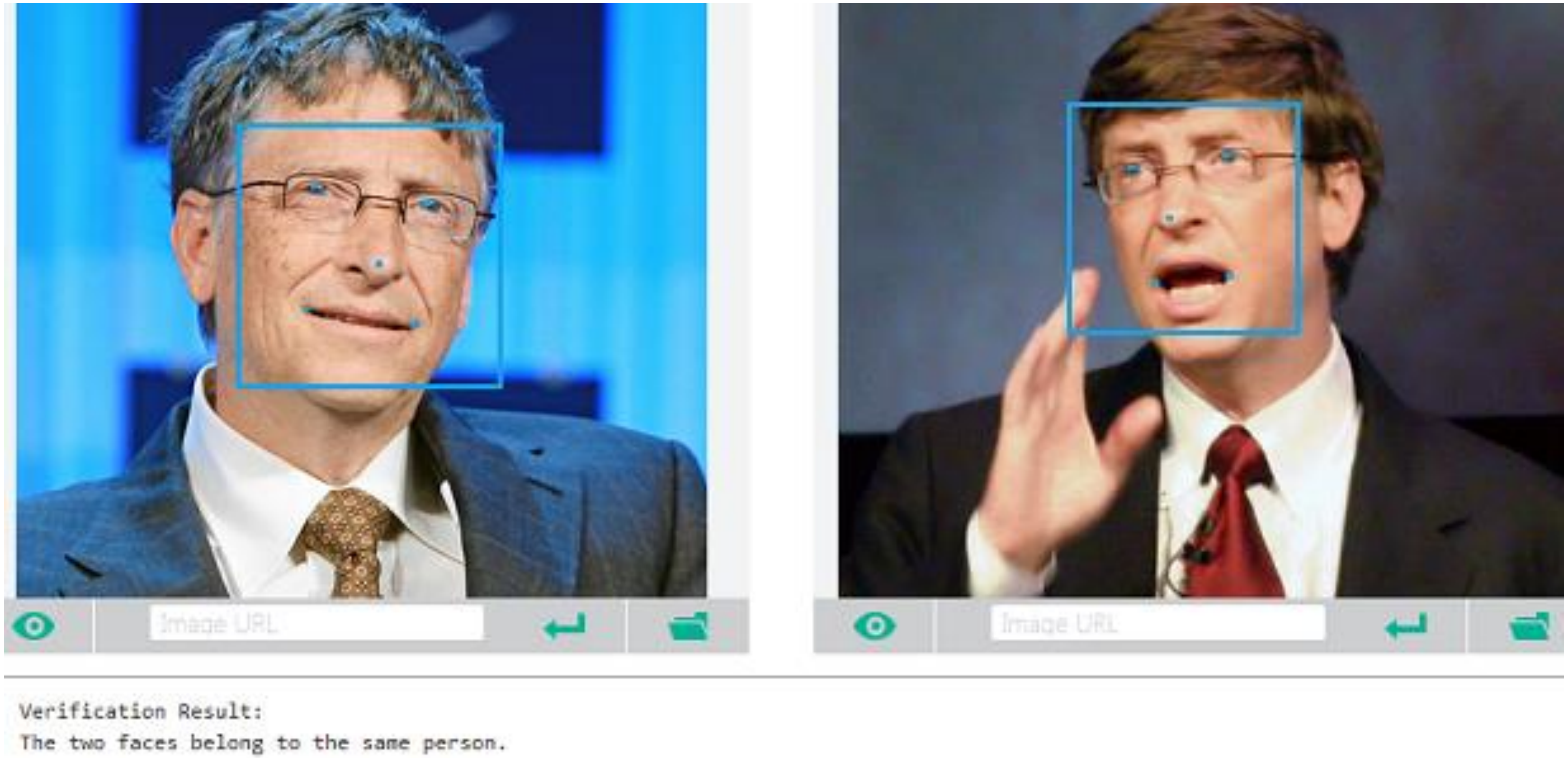
Adverserial images for neural networks



[Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy: Explaining and Harnessing Adversarial Examples, 2015]

Why is it colorful (the perturbation) if that is a sign() function?

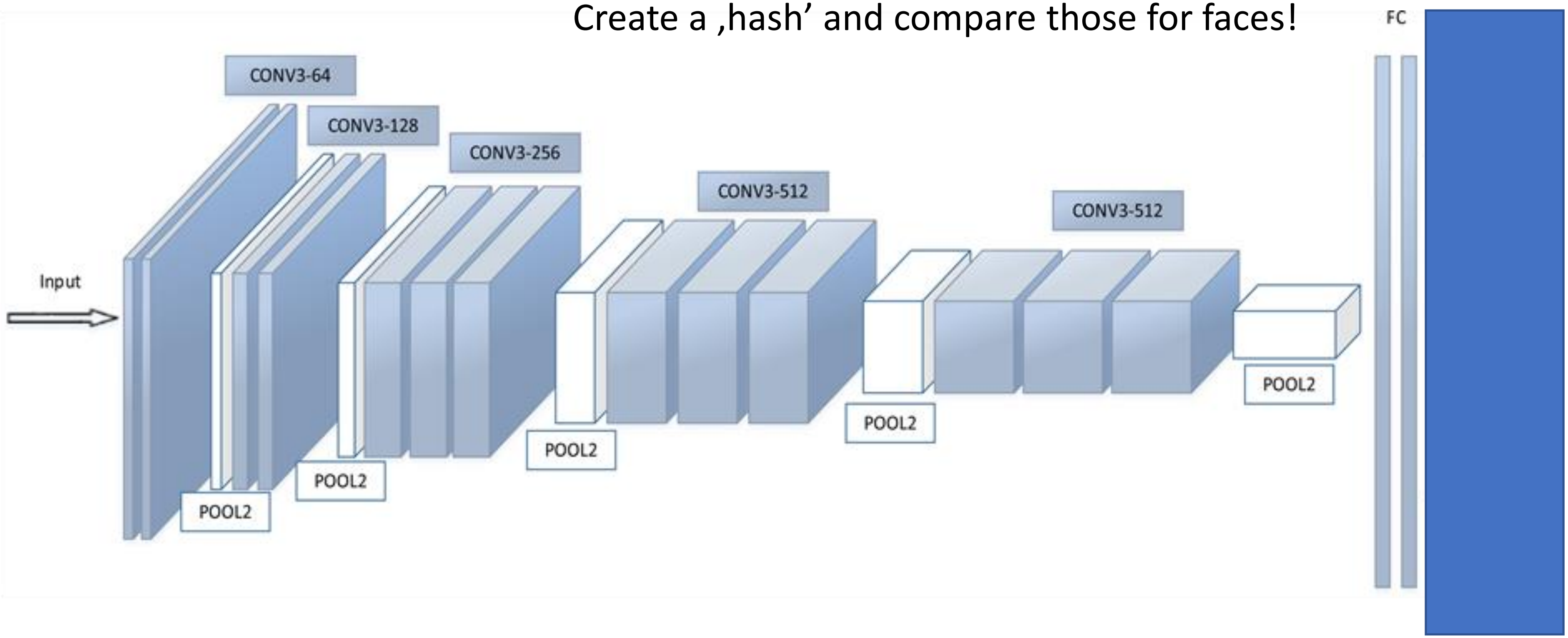
Face verification



https://www.sitepoint.com/use-react-native-to-a-create-a-face-recognition-app/

Face verification

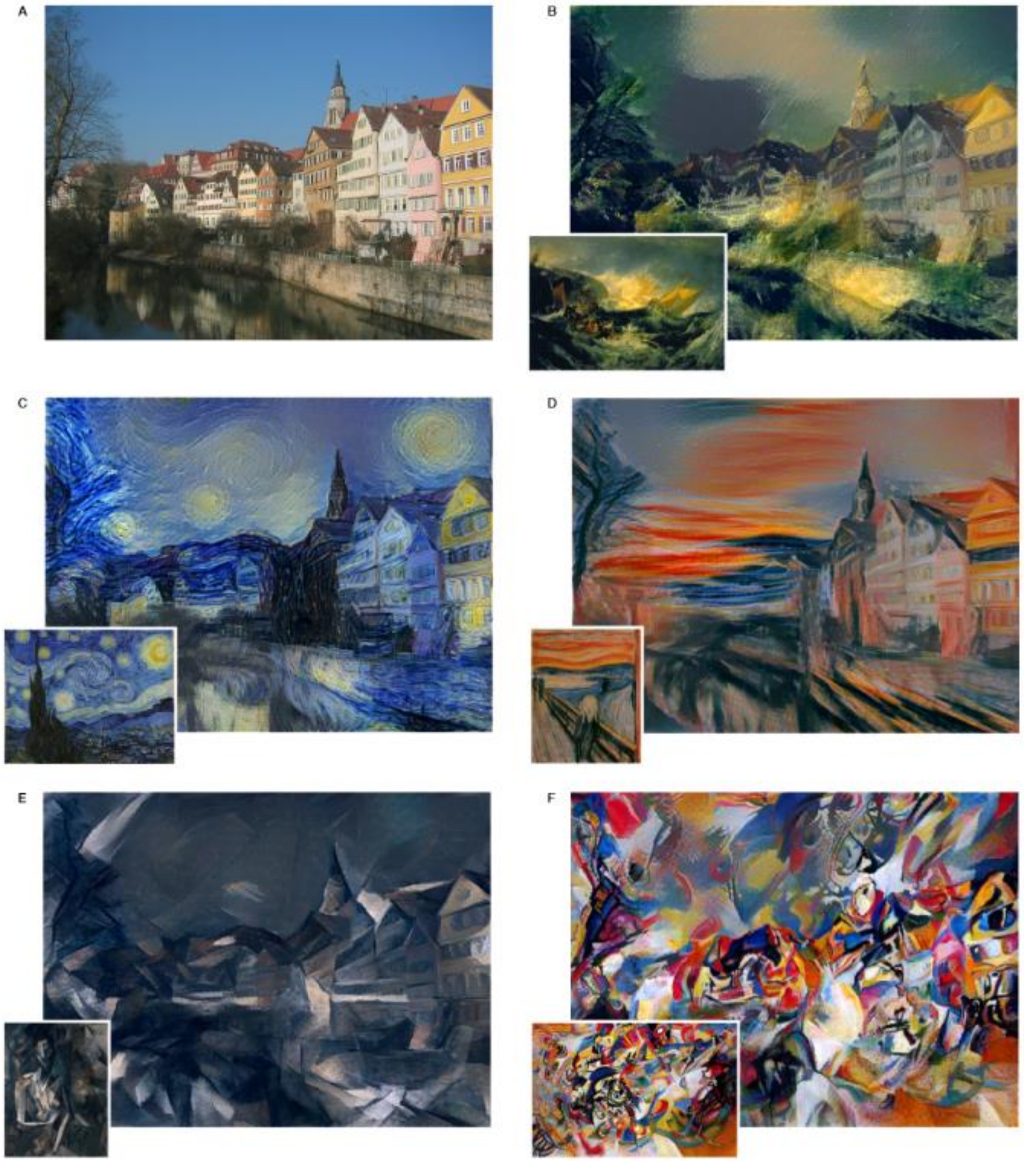
Train a network on faces. Class: different people.
IMDB database
Create a ‚hash’ and compare those for faces!



[http://file.scirp.org/Html/4-7800353_65406.htm]

Triple loss $d(A,P) + \alpha \leq d(A,N)$

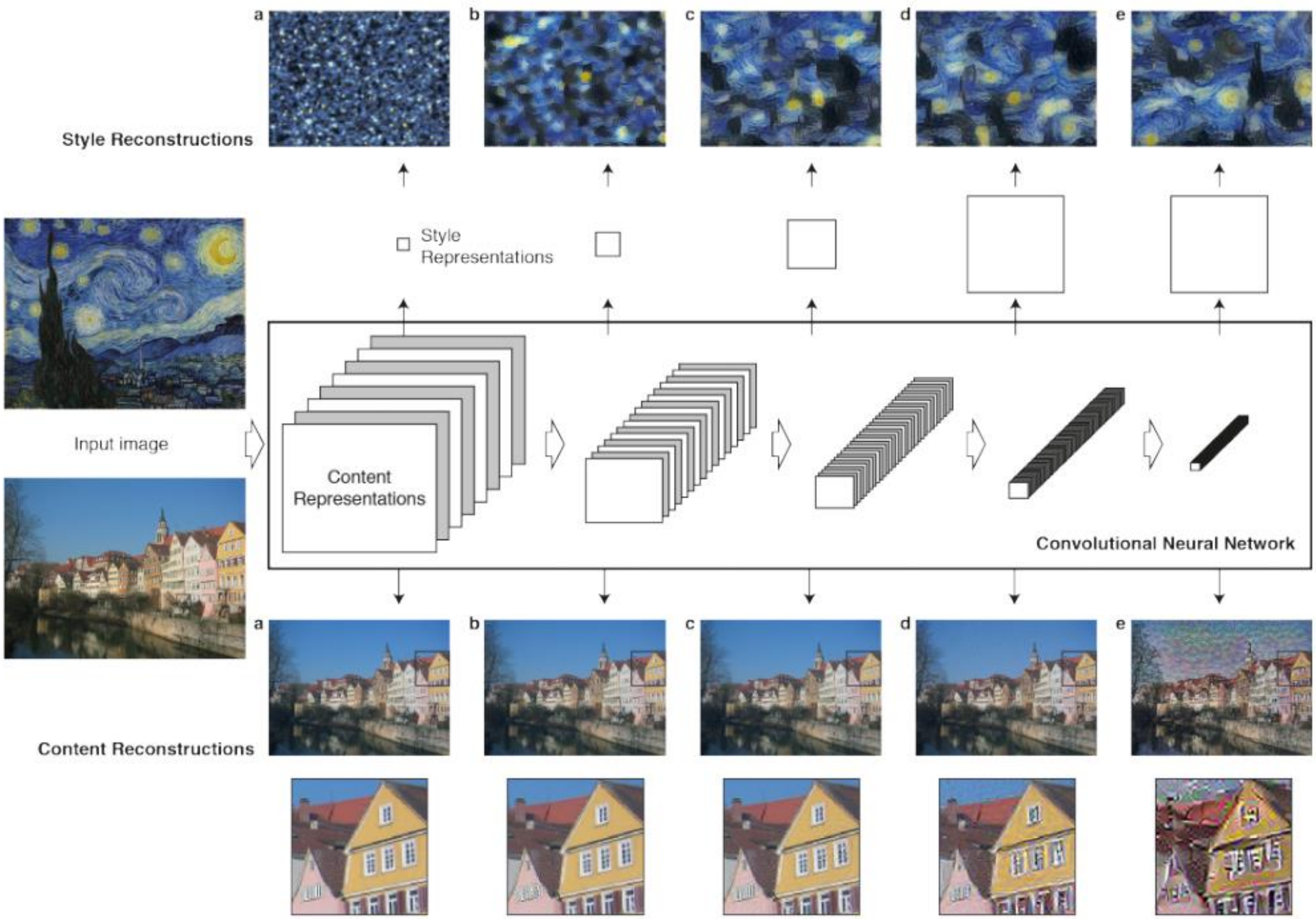
Style transfer



[Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style, 2015]

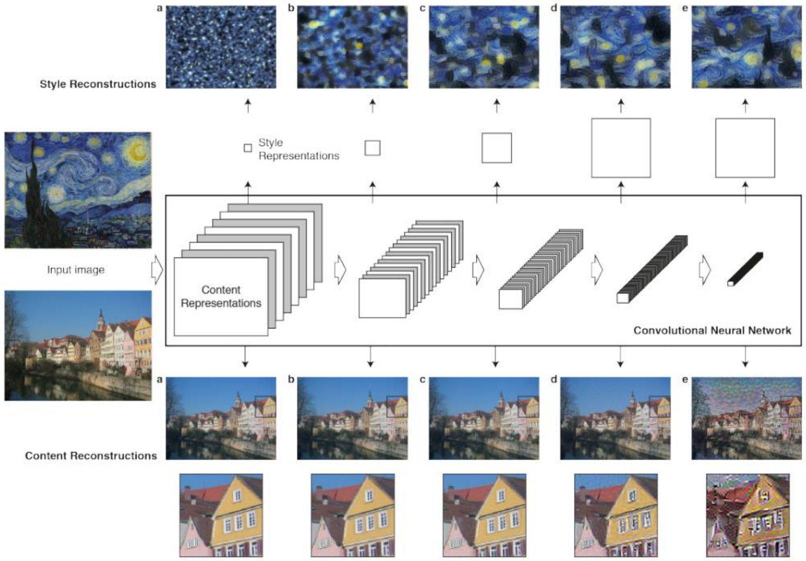
Style transfer

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha\mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta\mathcal{L}_{style}(\vec{a}, \vec{x})$$



[Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style, 2015]

Style transfer



F_{ij}^{l} : Is the activation of ith filter at position j in the layer l.

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}.$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l}\right)^2$$

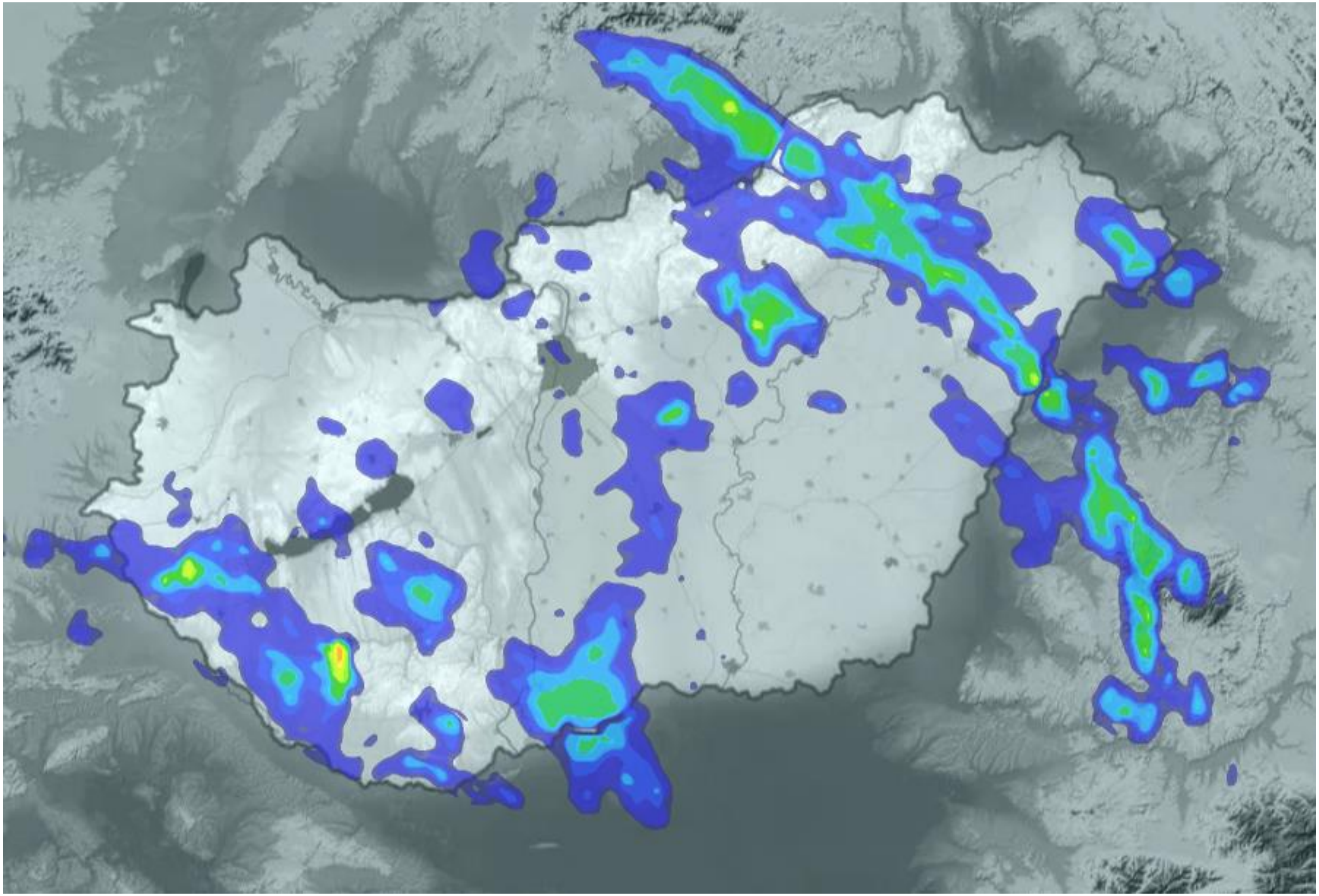
$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2 .$$

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

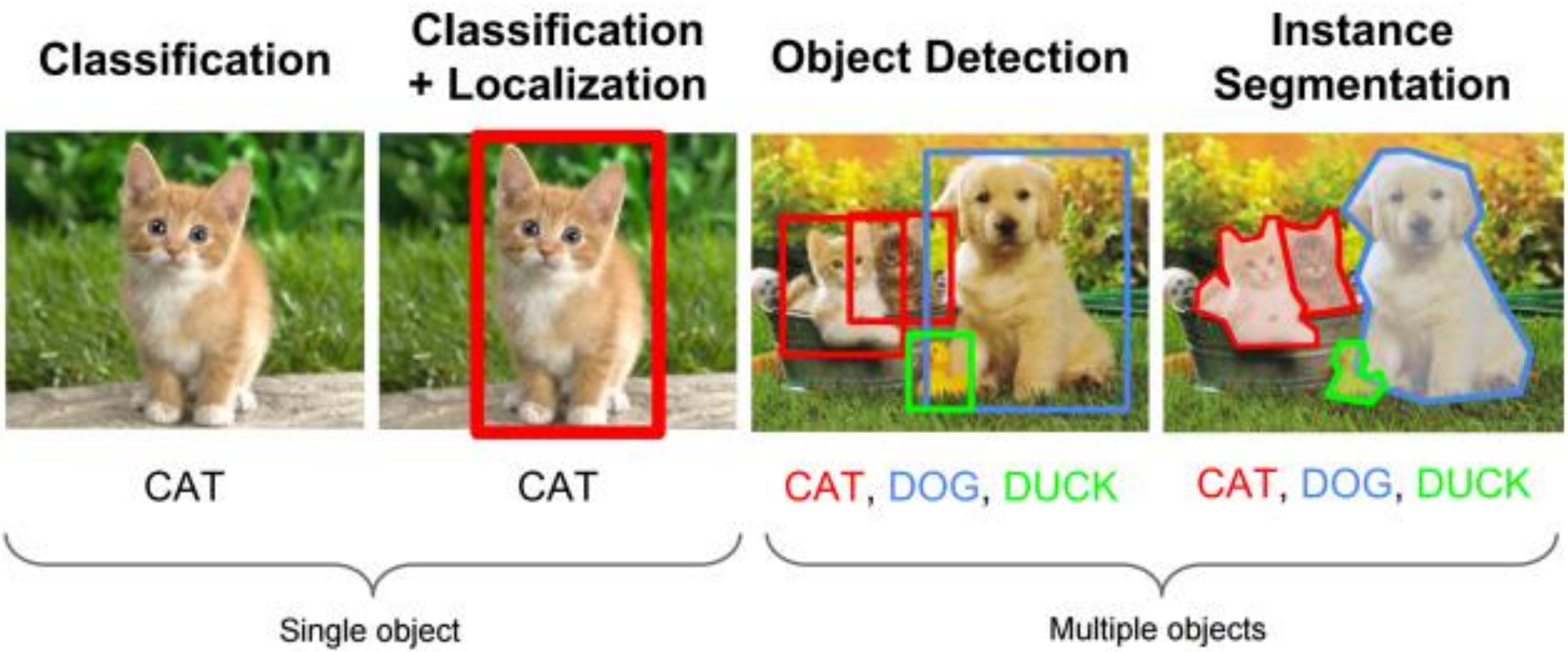
[Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style, 2015]

Weather forecast



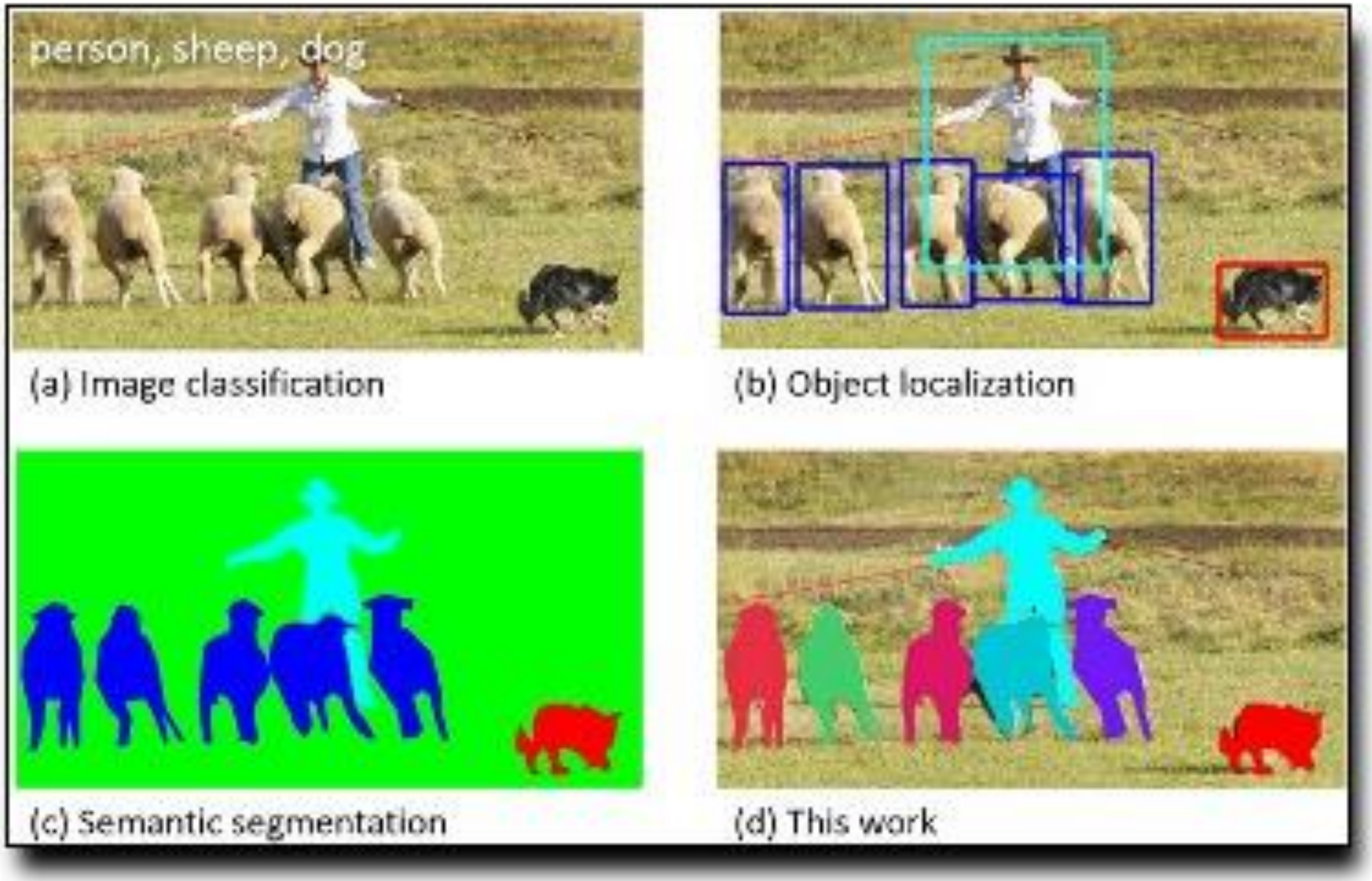
[https://idokep.hu/radar]

Object localisation/detection



https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html

Object localisation/detection

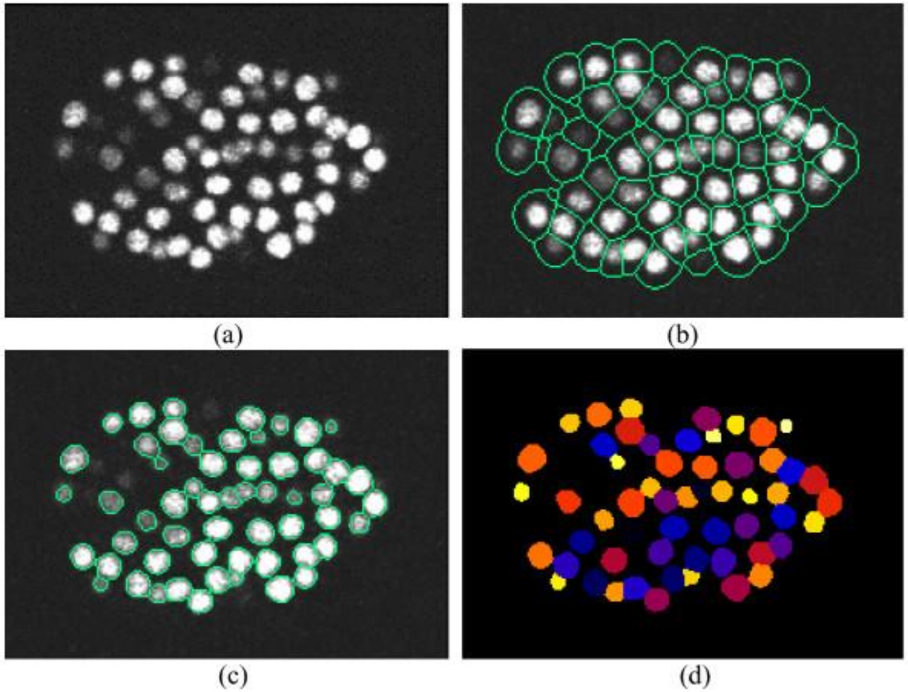


(a) Image classification
(b) Object localization
(c) Semantic segmentation
(d) This work

Microsoft COCO: Common Objects in Context

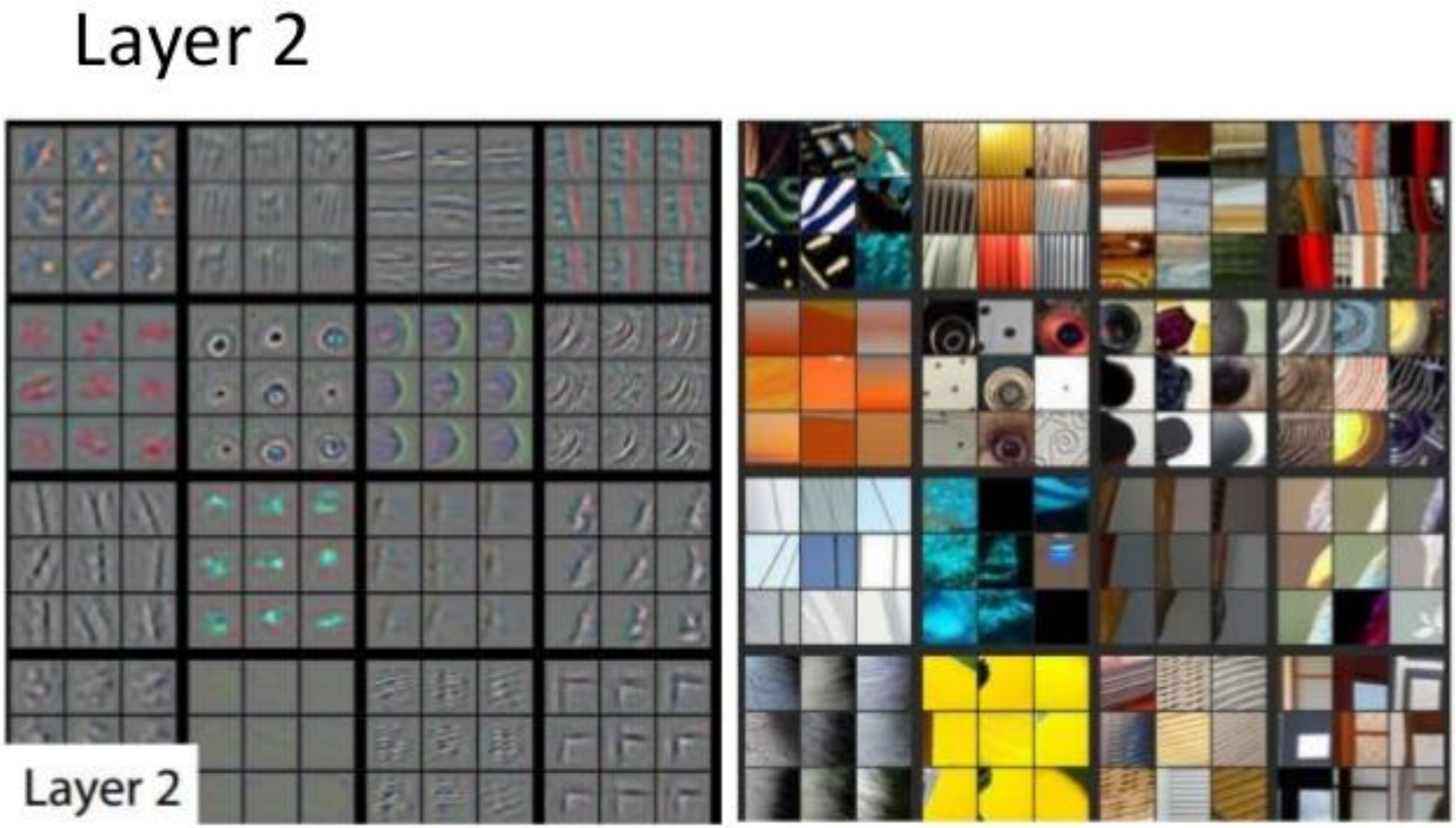
Instance segmentation

[https://www.kaggle.com/c/carvana-image-masking-challenge]



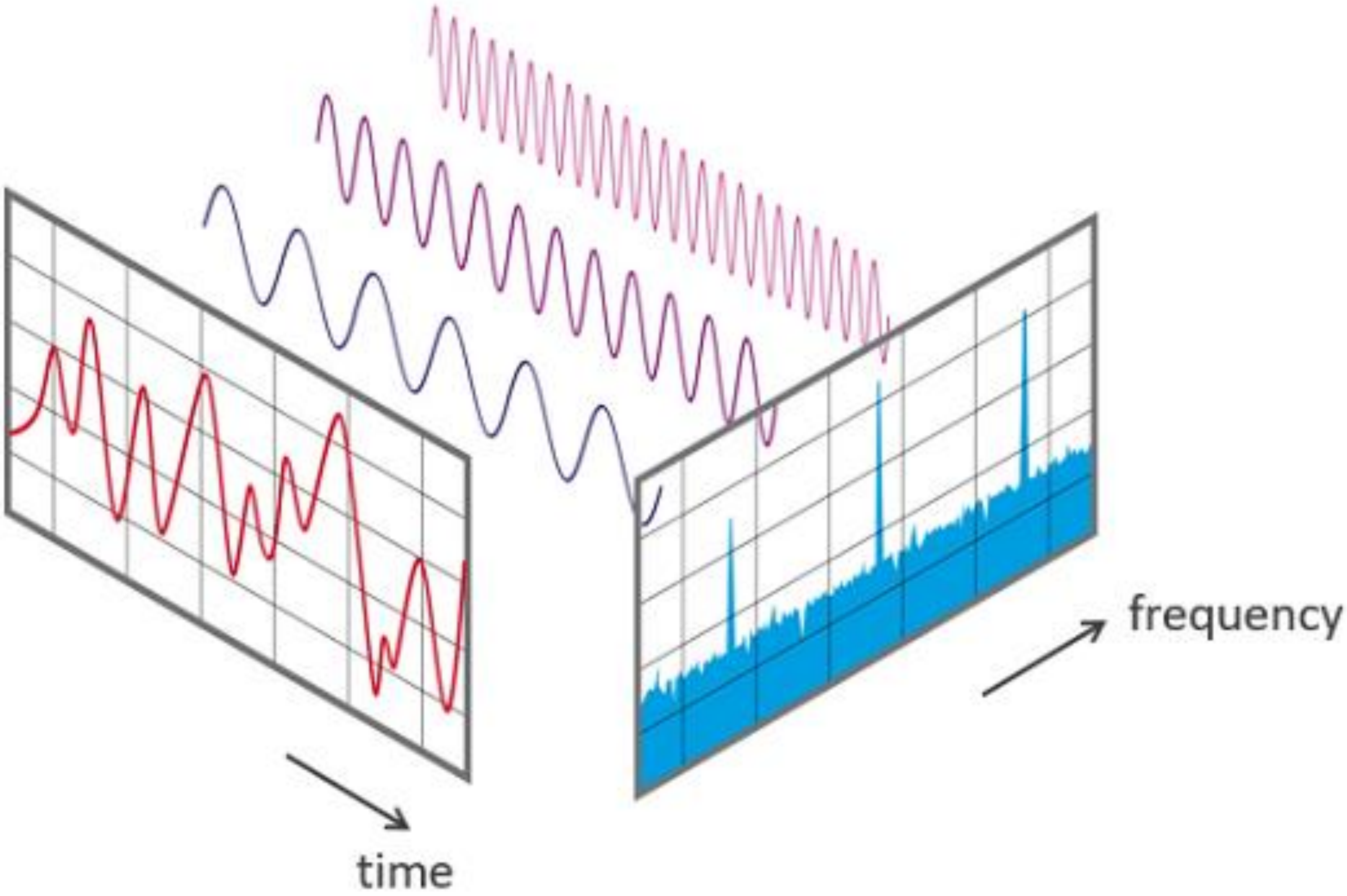
Li, Gang & Liu, Tianming & Tarokh, Ashley & Nie, Jingxin & Li, Kaiming & Mara, Andrew & Holley, Scott & Wong, Stephen. (2007). 3D cell nuclei segmentation based on gradient flow tracking. BMC cell biology. 8. 40. 10.1186/1471-2121-8-40.

Neural network visualisation



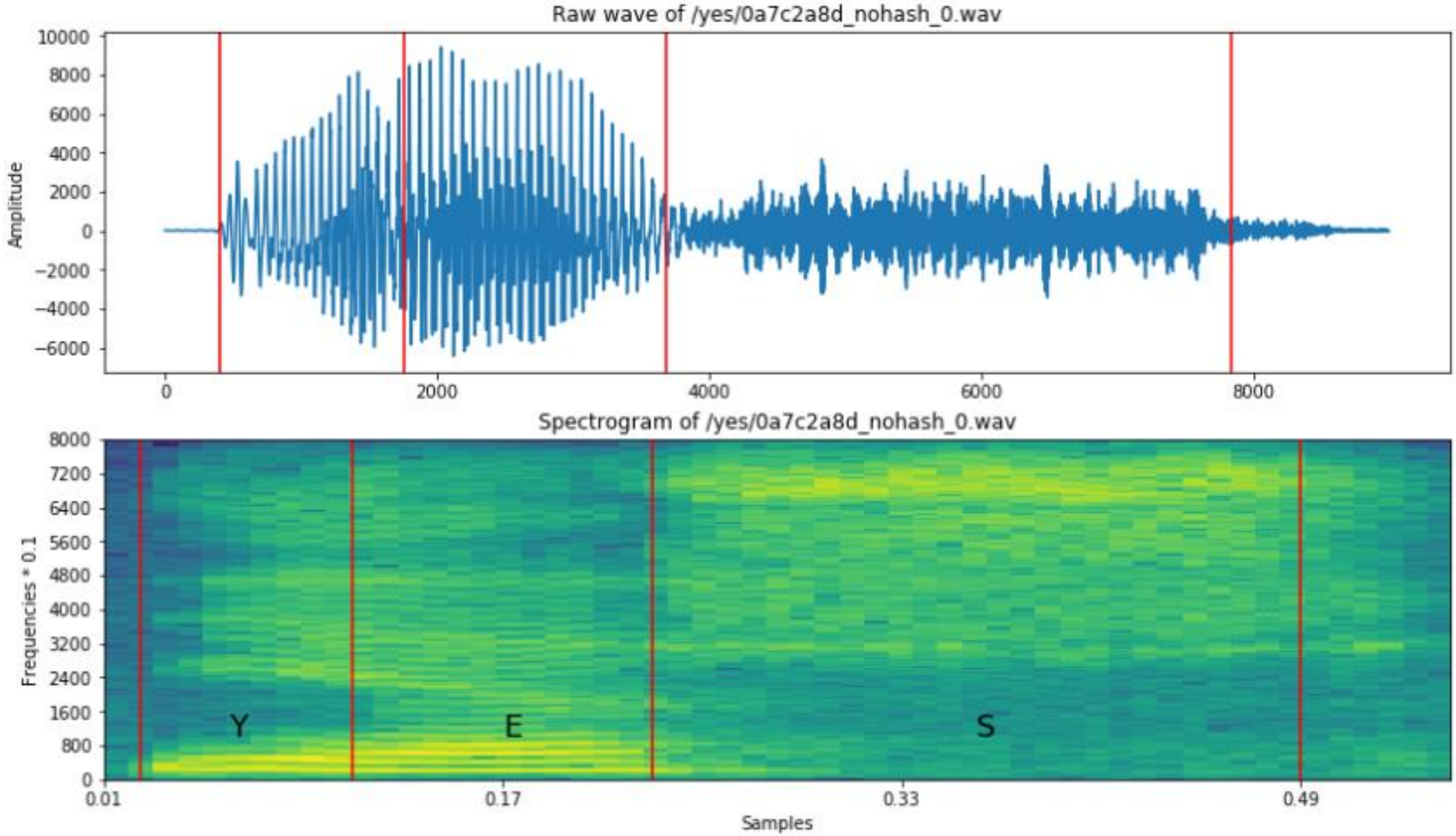
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Natural language processing – Fourier transformation



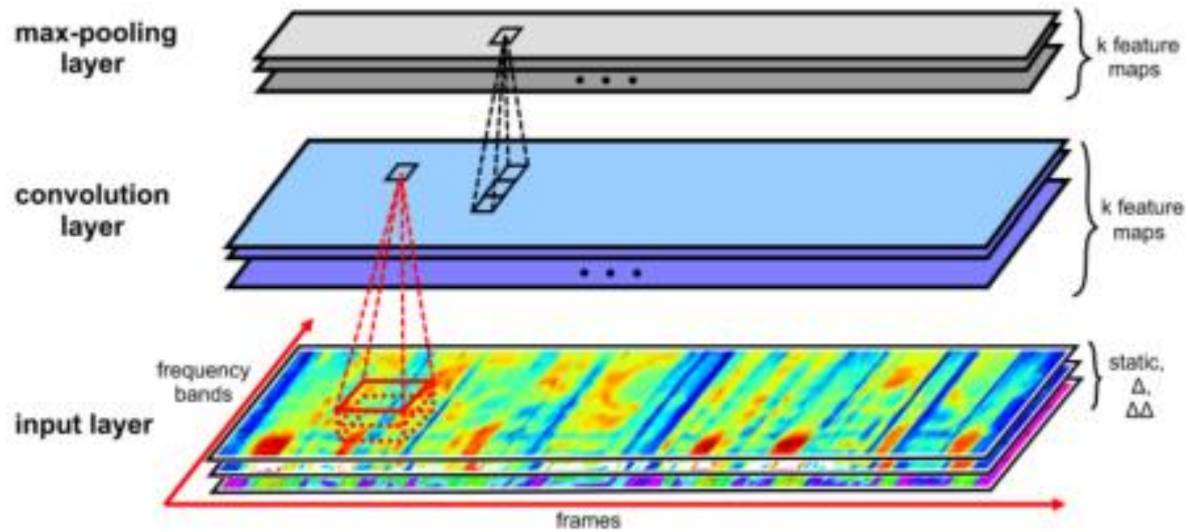
[By Phonical - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=64473578]

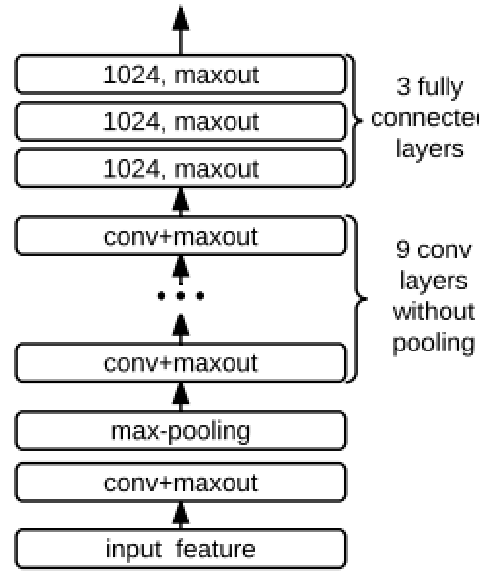
Natural language processing – Fourier transformation



[https://www.kaggle.com/davids1992/speech-representation-and-data-exploration]

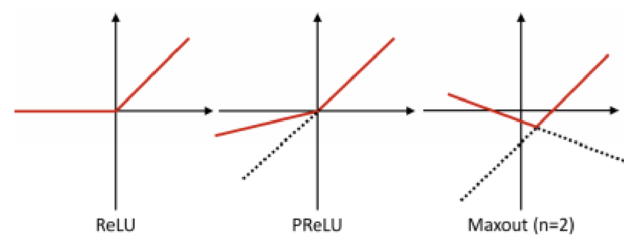
Natural language processing – speech to text





$$\left.\begin{array}{l}\sigma(a,b,c,-,-)\\ \sigma(a,b,-,c,c)\\ \sigma(a,a,b,b,c)\\ \sigma(-,a,-,b,c)\\ \vdots\\ \sigma(-,-,a,b,c)\end{array}\right\} = (a,b,c).$$

Model	NP	Dev PER	Test PER
BiLSTM-3L-250H [12]	3.8M	-	18.6%
BiLSTM-5L-250H [12]	6.8M	-	18.4%
TRANS-3L-250H [12]	4.3M	-	18.3%
CNN-(3,5)-10L-ReLU	4.3M	17.4%	19.3%
CNN-(3,5)-10L-PReLU	4.3M	17.2%	18.9%
CNN-(3,5)-6L-maxout	4.3M	18.7%	21.2%
CNN-(3,5)-8L-maxout	4.3M	17.7%	19.8%
CNN-(3,3)-10L-maxout	4.3M	18.4%	19.9%
CNN-(3,5)-10L-maxout	4.3M	**16.7%**	**18.2%**



[Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks,
Ying Zhang, Mohammad Pezeshki, Philemon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, Aaron Courville, 2017]

Listen audio!

What was it?

0 46797 She had your dark suit in greasy wash water all year.

Words:

3050 5723 she
5723 10337 had
9190 11517 your
11517 16334 dark
16334 21199 suit
21199 22560 in
22560 28064 greasy
28064 33360 wash
33754 37556 water
37556 40313 all
40313 44586 year

Phonemes:

0 3050 h#
3050 4559 sh
4559 5723 ix
5723 6642 hv
6642 8772 eh
8772 9190 dcl
9190 10337 jh
10337 11517 ih
11517 12500 dcl
12500 12640 d
...

NLP – tokenization, stemming

The quick brown fox jumps over the lazy dog.

[The] [quick] [brown] [fox] [jumps] [over] [the] [lazy] [dog]

[The] [quick] [brown] [fox] [jump] [over] [the] [lazy] [dog]

[the] [quick] [brown] [fox] [jump] [over] [the] [lazy] [dog]

→dictionary obtained from the corpus

Convert words to a one-hot encoded vector!

- We want:

$$\mathrm{oh}\colon \{0,1,\ldots,K\} \to [0,1]^K$$

$$\sum_{i=0}^{K} \mathrm{oh}(\mathrm{y}_i) = 1$$

- One-hot encoding:

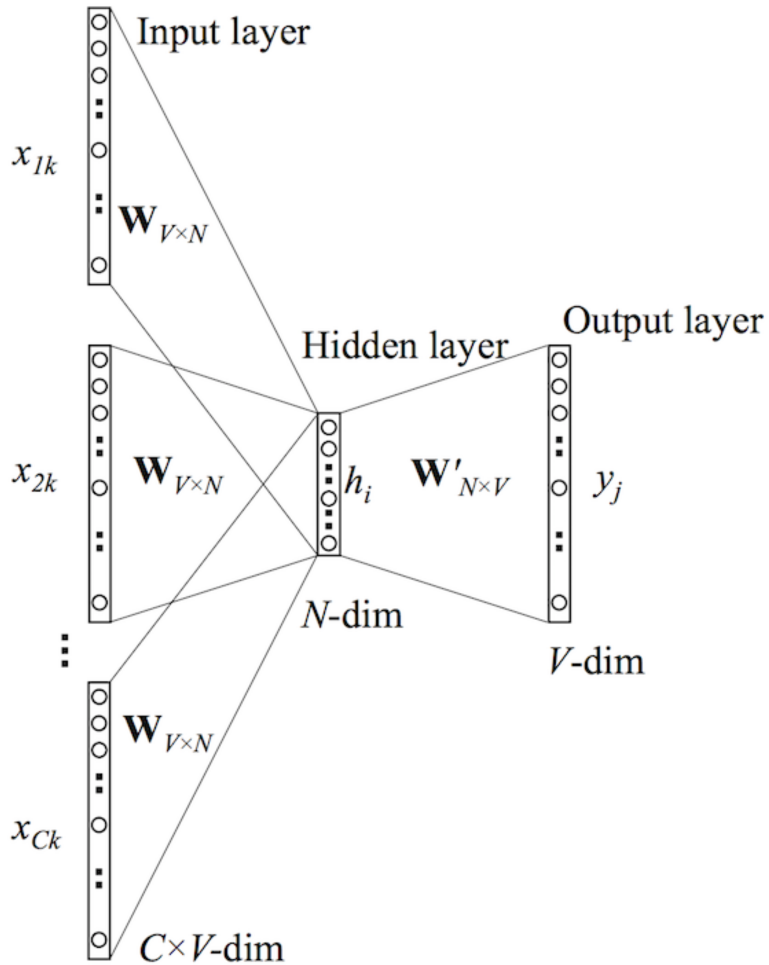
$$y = l \xRightarrow{one-hot} oh(y)_l = 1, oh(y)_i = 0, i = 0, \ldots, l-1, l+1, \ldots K$$

- Example: $K = 2$

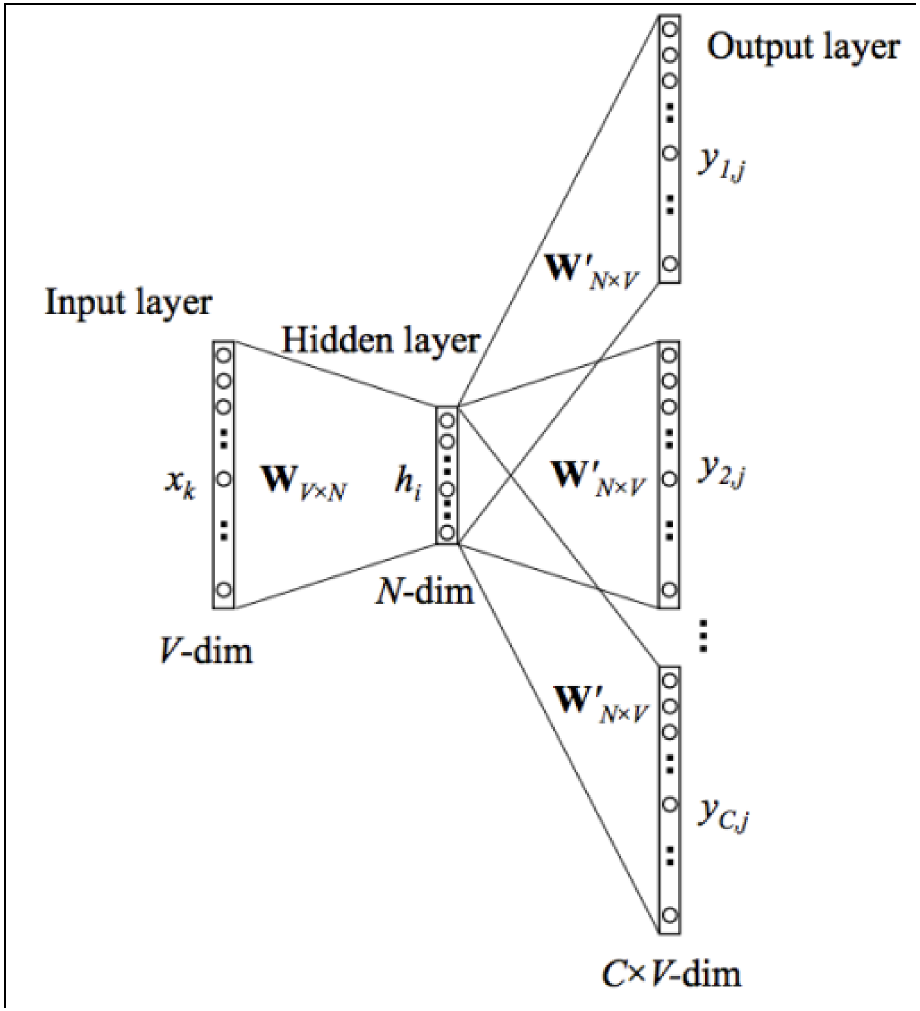
$$y = 0 \;\to \begin{bmatrix}1\\0\\0\end{bmatrix} \quad y = 1 \;\to \begin{bmatrix}0\\1\\0\end{bmatrix} \quad y = 2 \;\to \begin{bmatrix}0\\0\\1\end{bmatrix}$$

- Notation: $y_k = oh(y)_k$

[the] [quick] [brown] [fox] [jump] [over] [the] [lazy] [dog]



[the] [quick] [brown] [fox] [jump] [over] [the] [lazy] [dog]



NLP – word embedding

```
en_w2v.wv.get_vector('apple')
```

```
array([-2.25223231,  1.79967296,  0.52052546,  0.69880956, -0.96674138,
       -0.43120316, -0.51081914, -0.09760351, -1.87675786,  3.64533353,
        2.04445052,  0.33419853,  0.10876931, -0.0199236 , -1.3290658 ,
       -0.54760391,  0.33101451, -2.3777597 , -2.1069591 , -0.81782573,
        0.02968018, -1.16042852, -3.79935431, -0.02941807,  1.29824412,
       -0.19951613, -4.38423109, -1.76739872,  2.4510076 , -1.06378841,
        1.28968644, -1.76569963,  0.23196875,  2.89225411,  4.28000498,
        1.76823294,  1.62883067, -4.31515646,  1.15561104,  0.52216232,
        1.27078235,  0.79041451, -2.0780139 ,  0.41034013,  2.33784413,
        1.22297597,  3.73160815,  0.91349596, -0.06935301, -0.30641589,
       -0.69564182,  3.40794444,  0.32902223, -1.01418376,  1.77297831,
        1.24038219, -0.16458292,  0.12135817, -3.34925008, -2.00667858,
        0.89003199,  4.39943647,  0.18678869, -0.66747308, -4.27233362,
       -4.87201881,  0.98000288,  2.27560258,  0.03459861, -4.38171101,
        0.80729026, -0.92443126, -1.92179561,  2.02726626,  1.46704435,
       -0.31690702,  1.10866868,  2.41416979,  2.034863  , -0.07257579,
       -1.78879309, -1.61186671, -3.0232141 ,  1.03852248, -2.02575564,
        1.6589334 ,  2.78687406, -2.7956264 , -0.45835629,  0.32921287,
        1.69370782, -0.04152245,  4.29543209, -3.73792815, -2.16865706,
        0.56232905, -0.88750994,  4.84424067, -1.52330327,  1.5986172 ,
       -0.75493592, -4.36213779,  1.53122902, -2.96673155,  0.13642821,
       -2.68251276, -1.53297329,  1.35308564, -1.93756819,  1.08115268,
       -4.6438427 ,  3.71303248,  0.04859417, -0.73395061, -0.9872722 ,
        1.65776861, -0.30306721, -0.85497725, -1.82223523,  1.86270726,
        2.42779613,  2.28450656,  1.42392039,  1.11919343, -2.81615663,
        1.2226845 , -0.27100986,  1.69344366, -1.92687964,  3.53975511,
        2.05448508, -3.7142036 ,  0.02406235, -1.91634786,  1.24500644,
       -2.4066155 ,  0.94834107, -0.23953831, -1.43676019, -1.16314697,
        3.85159111, -0.59647632,  0.25417724,  1.76814449,  2.42557478,
        5.77475691,  2.25710011, -0.57142085, -3.07814813,  4.83230734,
       -0.98424572, -3.95217919,  0.99027419,  1.60168052, -0.91043991,
       -0.81072456,  1.01931286,  2.02447033,  4.61328077, -2.13164568,
       -1.34822476, -1.95118368, -0.75413716, -1.04838264,  0.85342103,
       -0.63646543, -4.96552658, -3.52666664,  0.87381017, -2.48047876,
        2.27663255, -0.74030322,  1.94776893, -3.14546323,  0.10569936,
        0.65624553, -2.36570859,  3.79818845,  3.58278966,  3.39272594,
       -1.54461873, -0.27346429,  0.23149812,  0.18188734, -2.39423633,
        4.9890008 ,  0.75473368, -0.19210243,  3.65836358,  3.15115833,
       -1.71657896,  0.83879387, -2.05918288,  0.39470637, -0.42049167,
       -3.64927292,  0.85835886,  1.17132759, -2.04276705, -1.03801847], dtype=float32)
```

Vector representation:

- cosine distance:

$$d(x,y) = \frac{xy}{\|x\| * \|y\|}$$

Visualising high-dimensional data: t-SNE

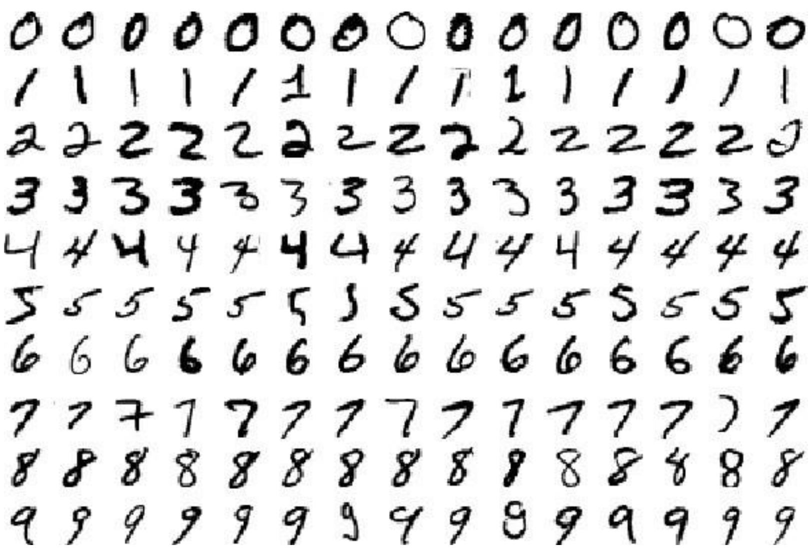
$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}, \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq i}(1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$

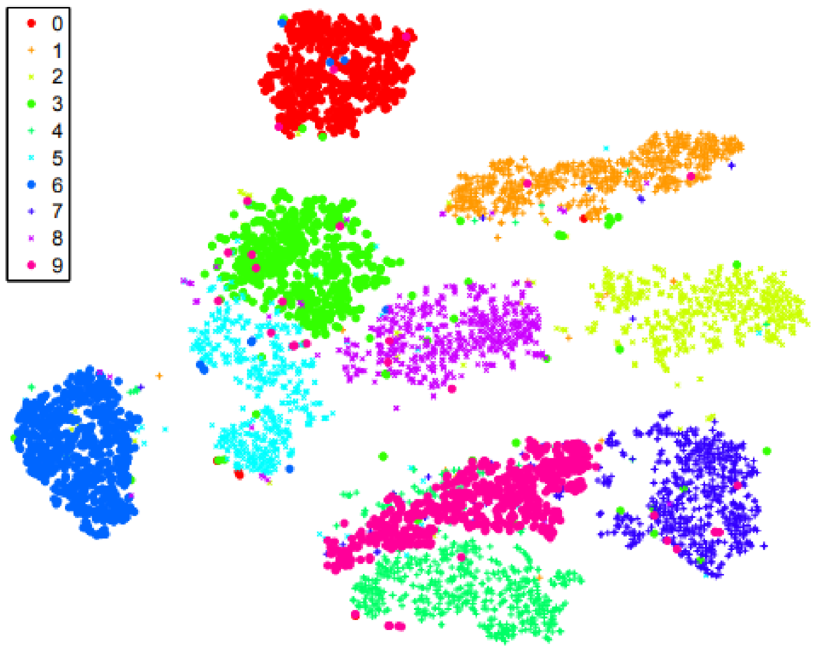
Minimize the Kullback-Leibler divergence:

$$KL(P||Q) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE example



VAN DER MAATEN AND HINTON



(a) Visualization by t-SNE.

[**Visualizing Data using t-SNE** ***Laurens van der Maaten, Geoffrey Hinton***; 9(Nov):2579--2605, 2008.]

DEMO notebook

Project ideas – but you can come up with others too…

- **Adverserial pictures**
  - how to fool a neural network
- **Face recognition/verification**
  - who is on the picture?
  - 2 picture contains the same person or not
- **Style transfer**
  - paintings
  - voice
- **Weather forecast:**
  - weather radar images → rain/wind forecast
- **Object localisation**
  - Crop galaxies from an image
- **Instance segmentation**
  - cells for science / cars to remove background
- **Visualisation of inner layers**
  - to understand what's going on inside
- **Natural language processing**
  - add emojis to sentences
  - speech to text

Project steps:

- idea
- check if there is data
- modeling
- documentation