Convolutional Neural Networks and Image Processing

Jiří Materna

About me

- Ph.D. in Natural Language Processing and Artificial Intelligence at Masaryk University
- 10 years at Seznam.cz (last 8 years as Head Of Research)
- Founder and lecturer at ML College
- Founder and co-organizer of ML Prague
- ML Freelance and consultant

www.mlguru.com **www.mlprague.com** **www.mlcollege.com**

Outline

Day 1 and 2

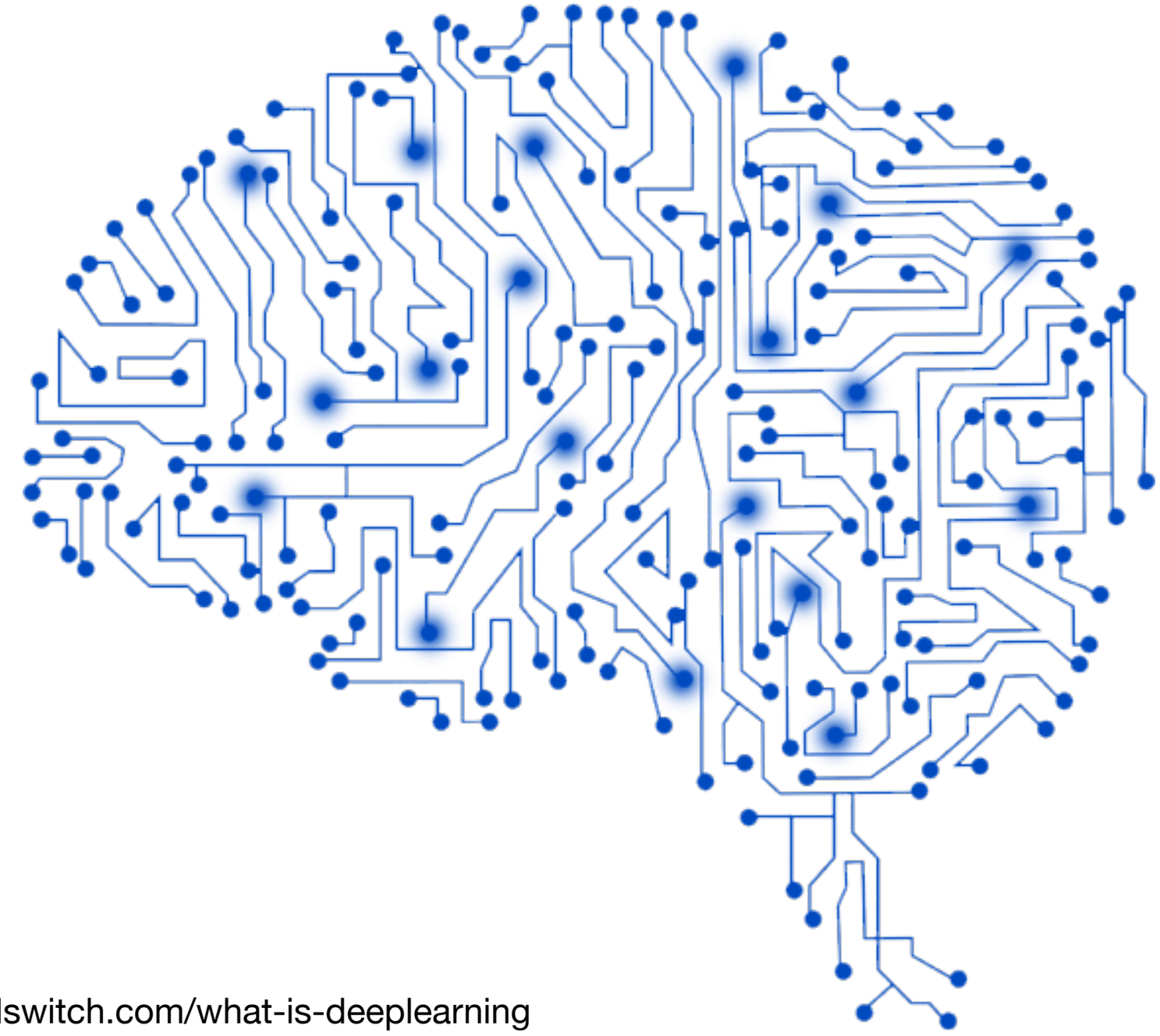
- Introduction to neural networks
- Activation functions for neural networks
- Multilayered neural networks
- Methods for training neural networks
- Convolutional neural networks
- Keras tutorial
- Practical classification and regression tasks solved using neural networks

Outline

Day 3 and 4

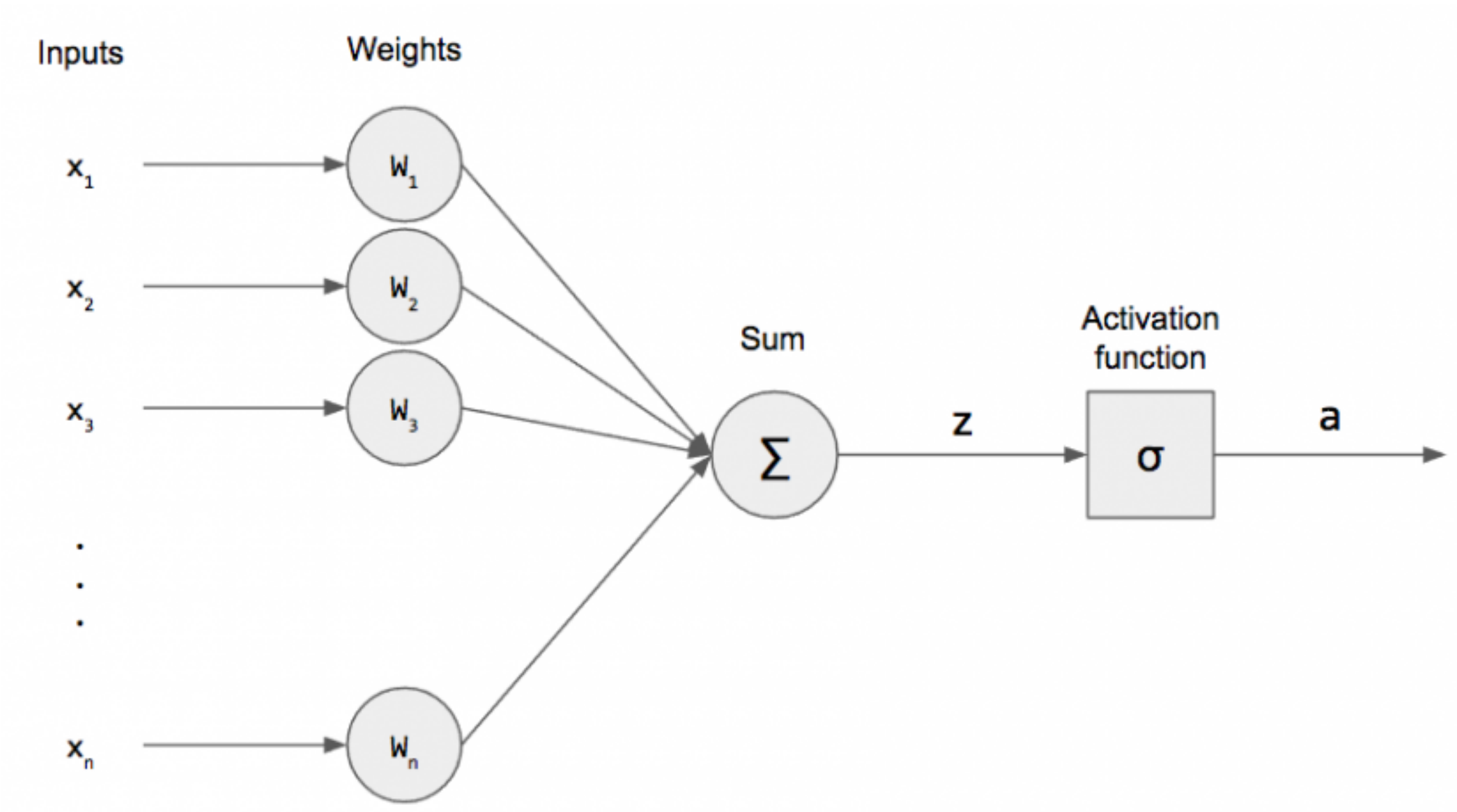
- VGG 16 and ResNet
- Transfer learning and fine-tuning
- Image classification
- Batch normalization and data augmentation
- U-net and Image segmentation
- GANs and superresolution
- Neural network explainability
- Adversarial patch

Neural networks and deep learning



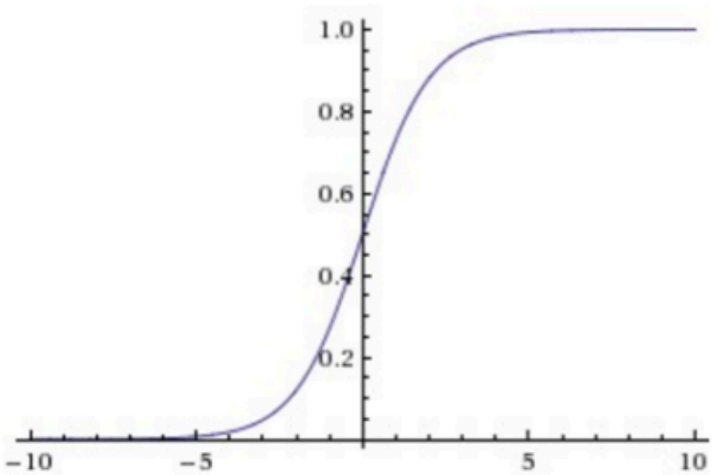
Source: http://www.rapidswitch.com/what-is-deeplearning

Perceptron

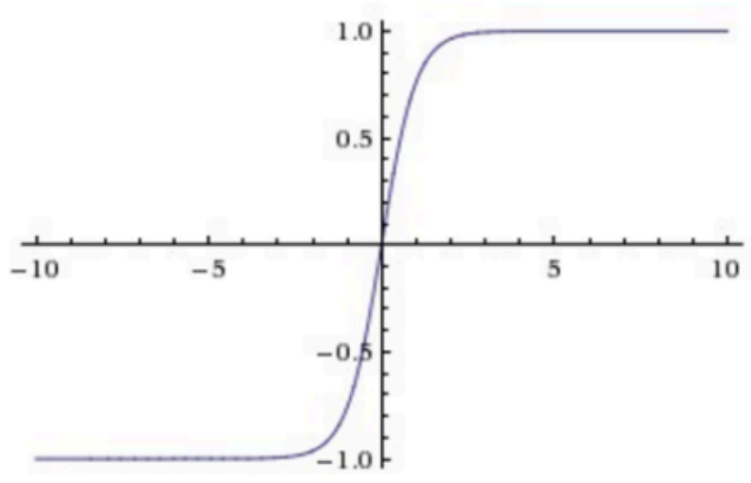


Source: https://pythonmachinelearning.pro

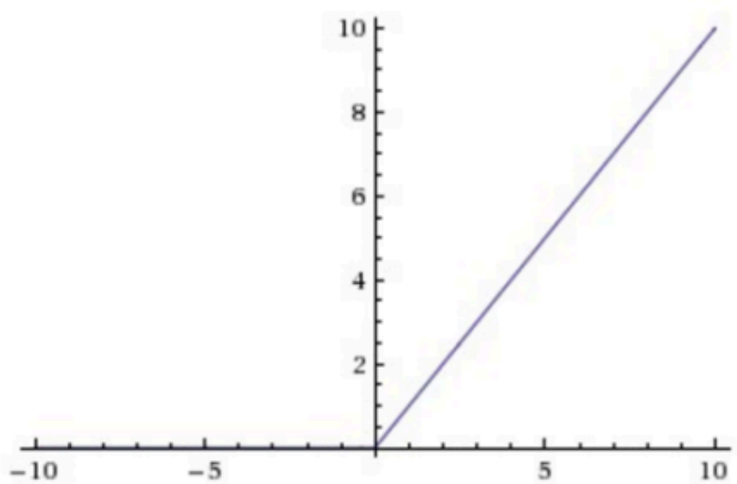
Activation functions



Sigmoid



tanh

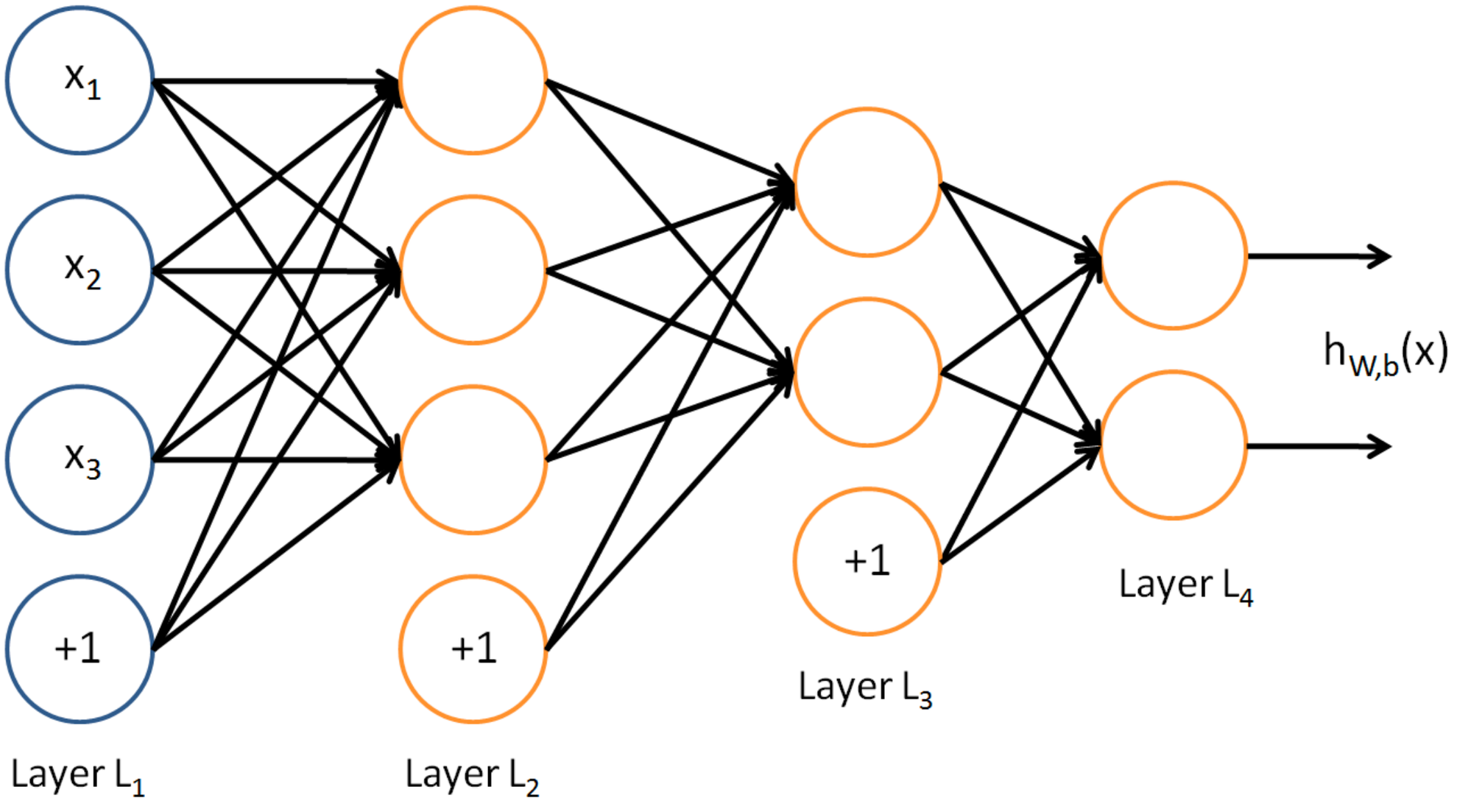


ReLU

Softmax:

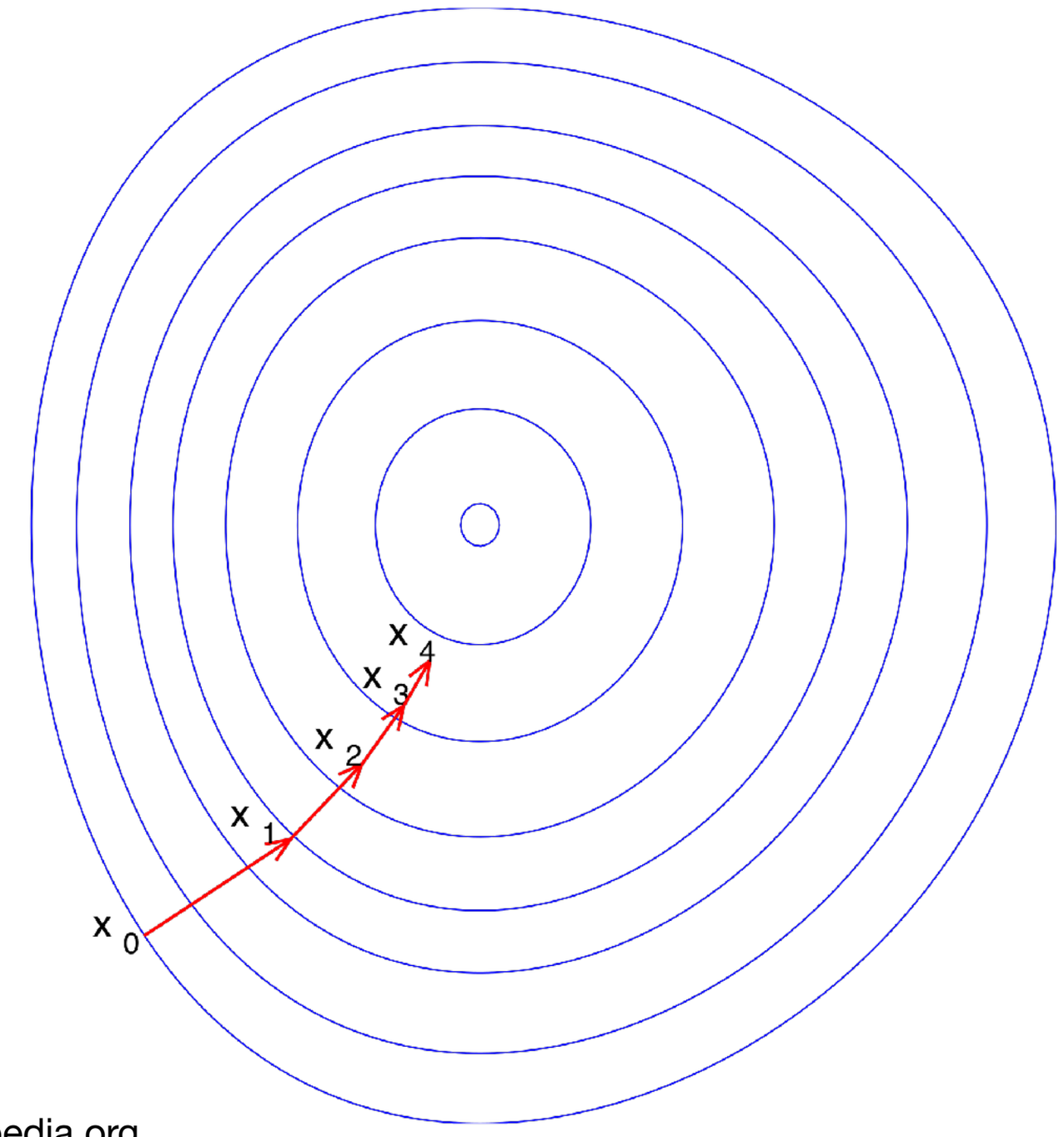
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Multilayer Neural Networks



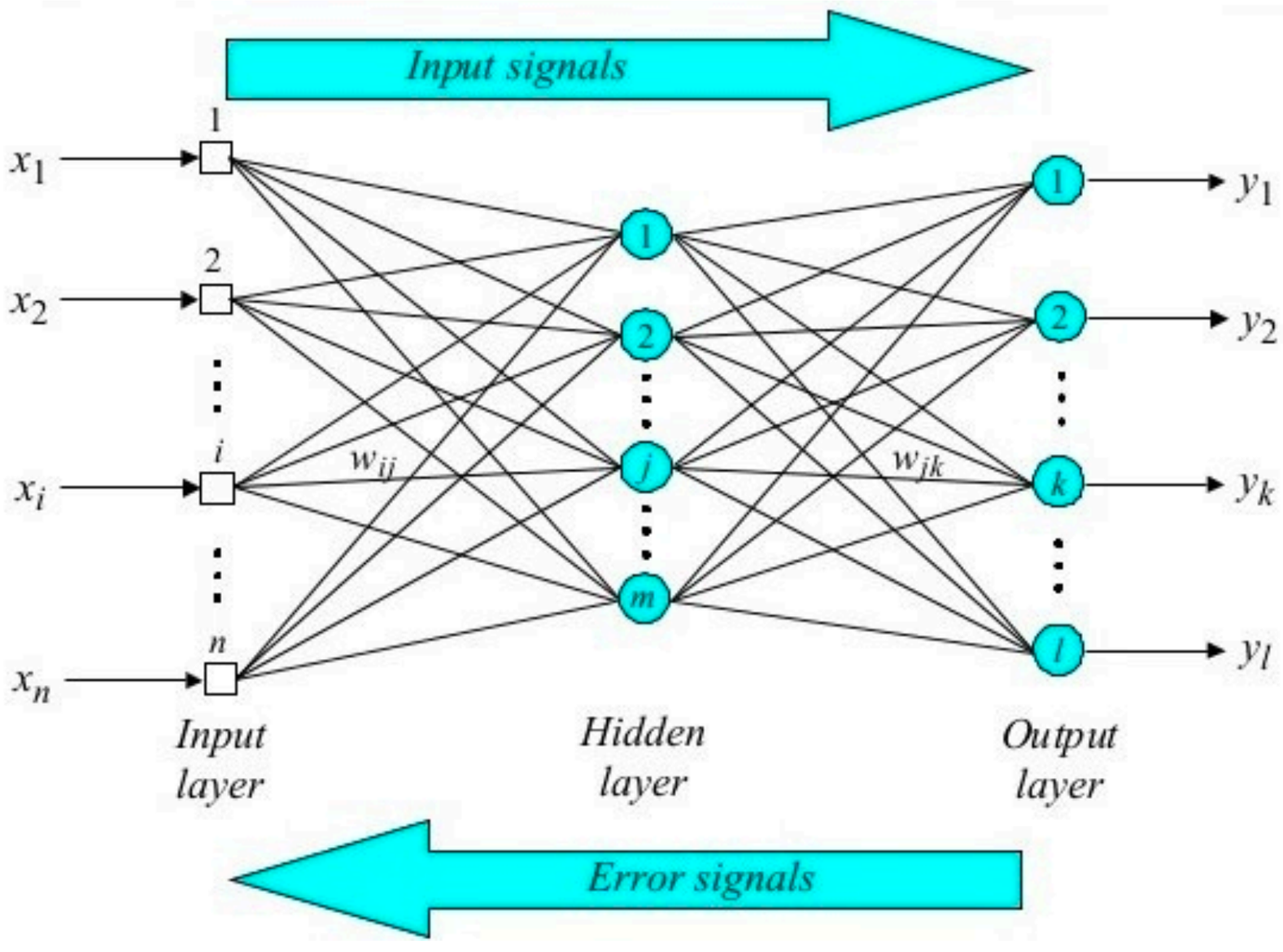
Source: http://ufldl.stanford.edu/

Steepest gradient descent



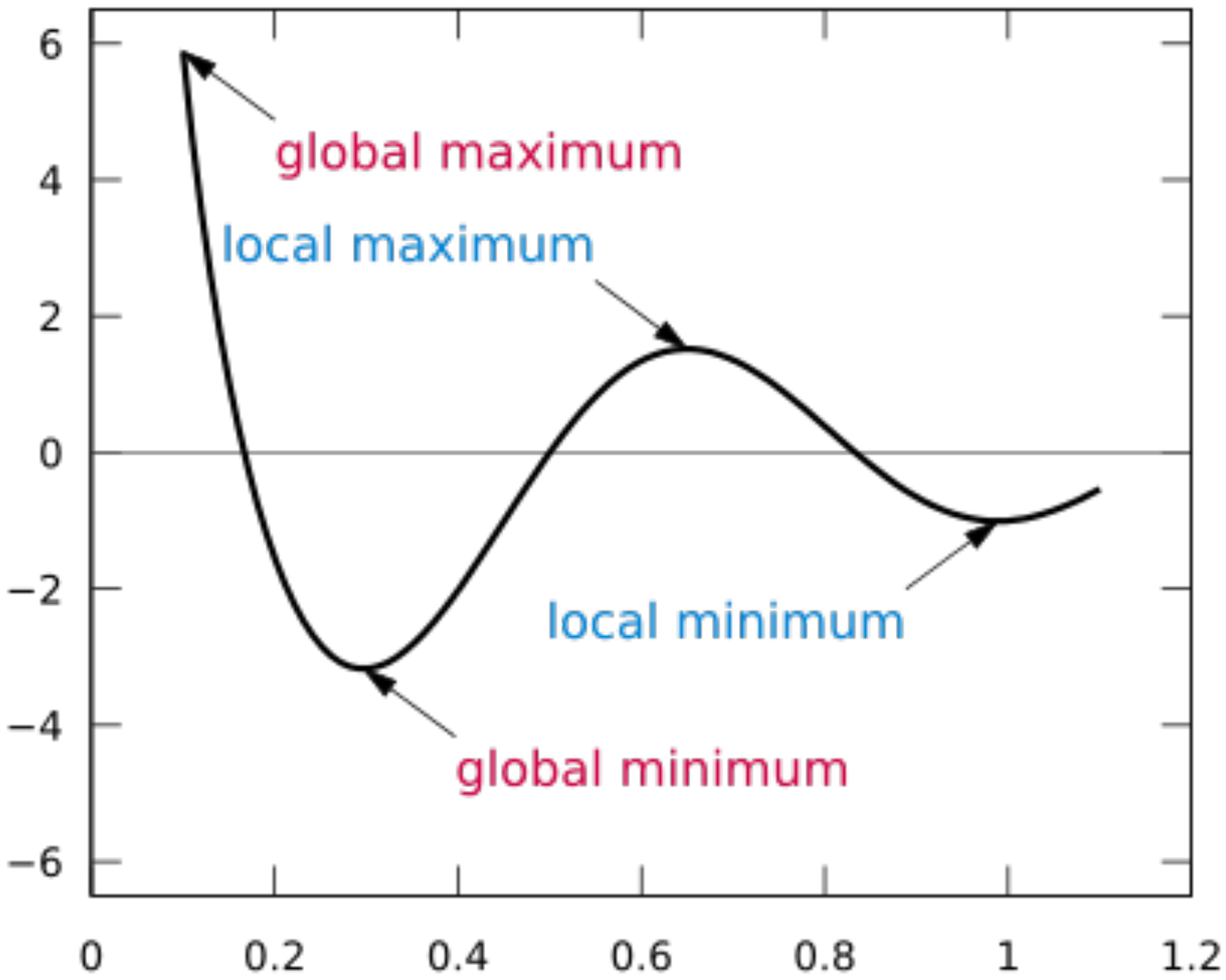
Source: https://www.wikipedia.org

Back propagation



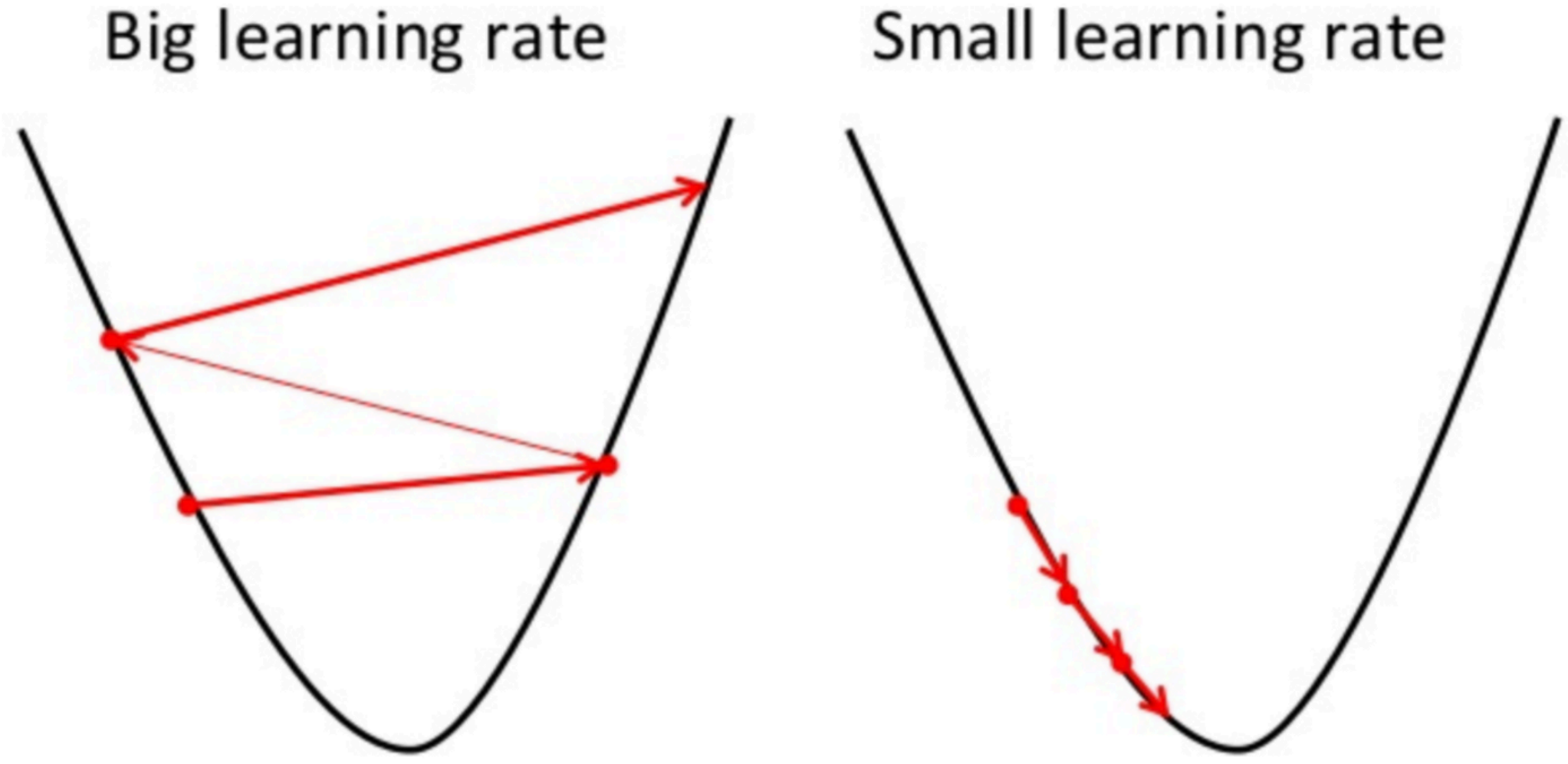
Source: https://www.quora.com/How-do-artificial-neural-networks-work

Parameter optimization strategies



https://heartbeat.fritz.ai/an-empirical-comparison-of-optimizers-for-machine-learning-models-b86f29957050

Learning rate tuning



Gradient Descent Variants

(Batch) Gradient Descent

$$w_{t+1} = w_t - \lambda \frac{\partial e(X, y)}{\partial w_t}$$

Stochastic Gradient Descent

$$w_{t+1} = w_t - \lambda \frac{\partial e(X^i, y^i)}{\partial w_t}$$

Mini-Batch Gradient Descent

$$w_{t+1} = w_t - \lambda \frac{\partial e(X^{(i,i+n)}, y^{(i,i+n)})}{\partial w_t}$$

Momentum and Nesterov Accelerated Gradient

Naive momentum

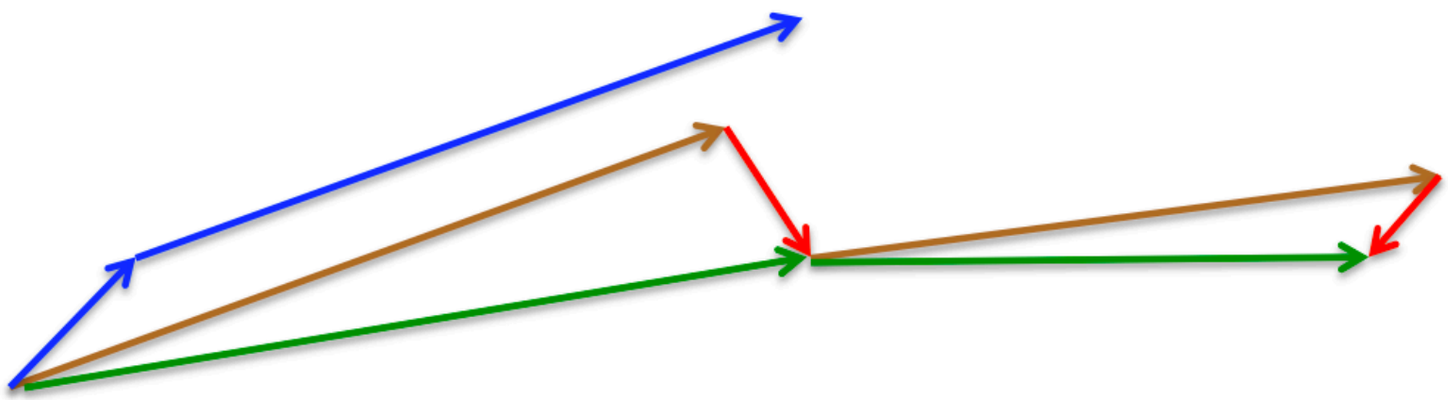
$$v_t = \gamma v_{t-1} + \lambda \frac{\partial e(w_t)}{\partial w_t}$$

$$w_{t+1} = w_t - v_t$$

Nesterov Accelerated Gradient

$$v_t = \gamma v_{t-1} + \lambda \frac{\partial e(w_t - \gamma v_{t-1})}{\partial w_t}$$

$$w_{t+1} = w_t - v_t$$



https://ruder.io/optimizing-gradient-descent/index.html#nesterovacceleratedgradient

Adaptive Gradient Algorithms

Adagrad

$$w_{t+1} = w_t - \frac{\lambda}{\sqrt{\sum_{i=1}^{t} g_i^2 + \epsilon}} g_t$$

$$g_i = \frac{\partial e(w_i)}{\partial w_i}$$

RMSProp

$$w_{t+1} = w_t - \frac{\lambda}{\sqrt{\mathbb{E}[g^2]_t - \epsilon}} g_t^2$$

$$\mathbb{E}[g^2]_t = \gamma \mathbb{E}[g^2]_{t-1} + (1-\gamma) g_t^2$$

Adam and Nadam

Adam Combination of RMSProp with momentum

Nadam Combination of RMSProp with Nesterov momentum

Loss functions for deep learning

Mean Squared Error

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

Mean Absolute Error

$$MSE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

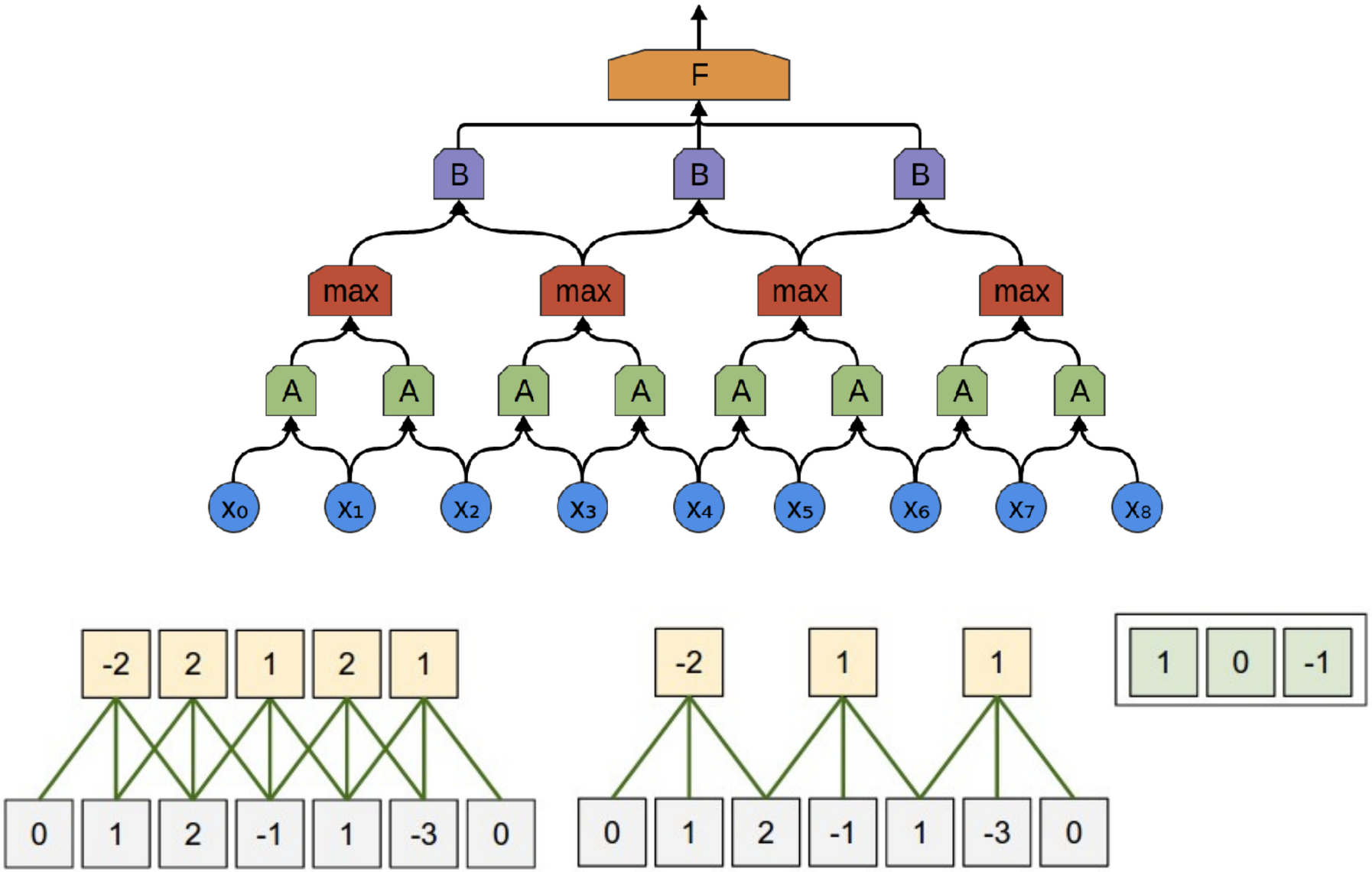
Cross Entropy (Negative Log Likelihood)

Categorical Cross Entropy $$CCE = -\frac{\sum_{i=1}^{n}\sum_{j=1}^{c} y_{i,j} log(\hat{y}_{i,j})}{n}$$

Binary Cross Entropy

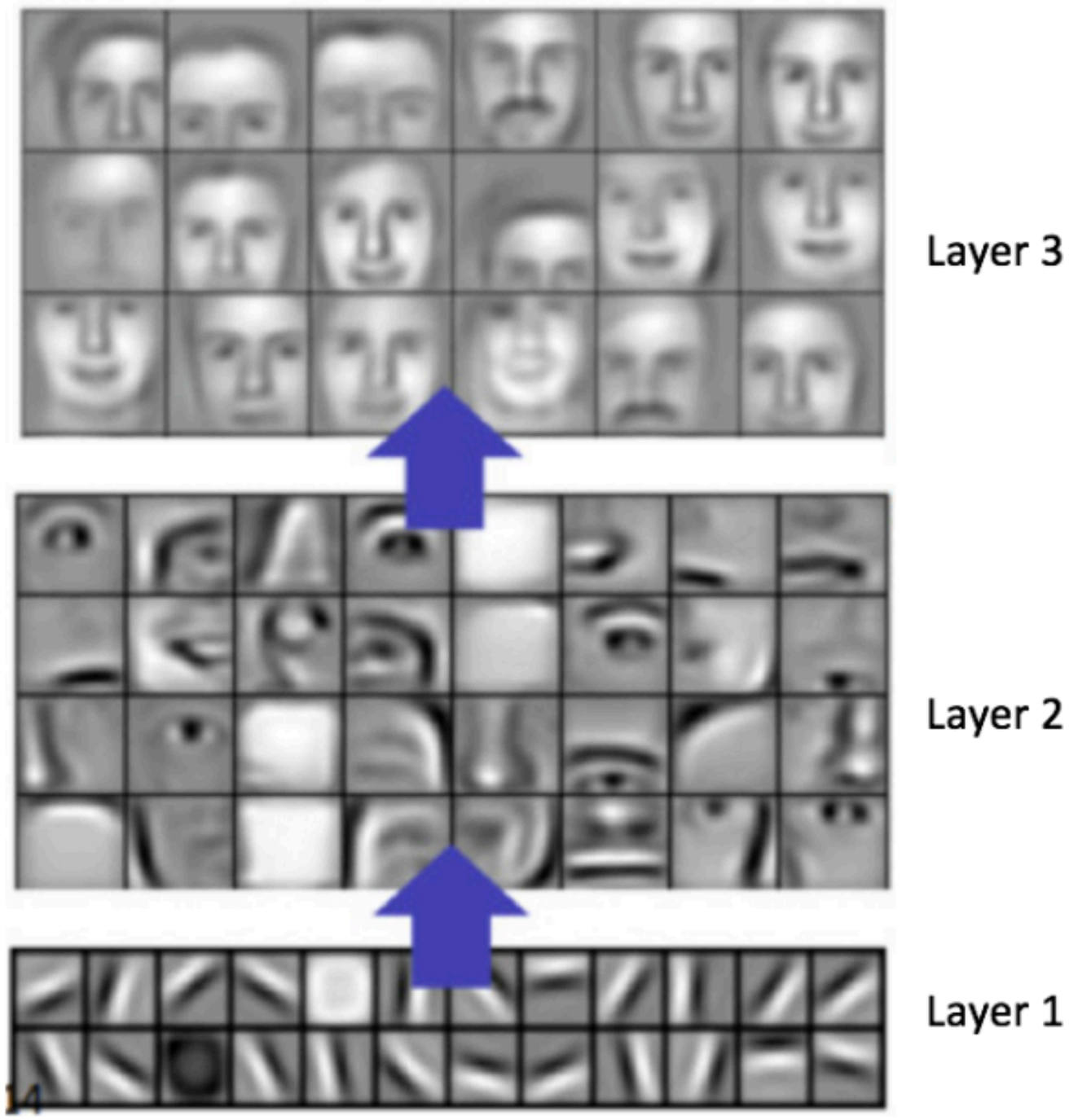
$$BCE = -\frac{\sum_{i=1}^{n}[y_{i,j} log(\hat{y}_{i,j}) + (1 - y_{i,j}) log(\hat{y}_{i,j})]}{n}$$

Convolution



Source: https://www.tensorflow.org

Weights visualization

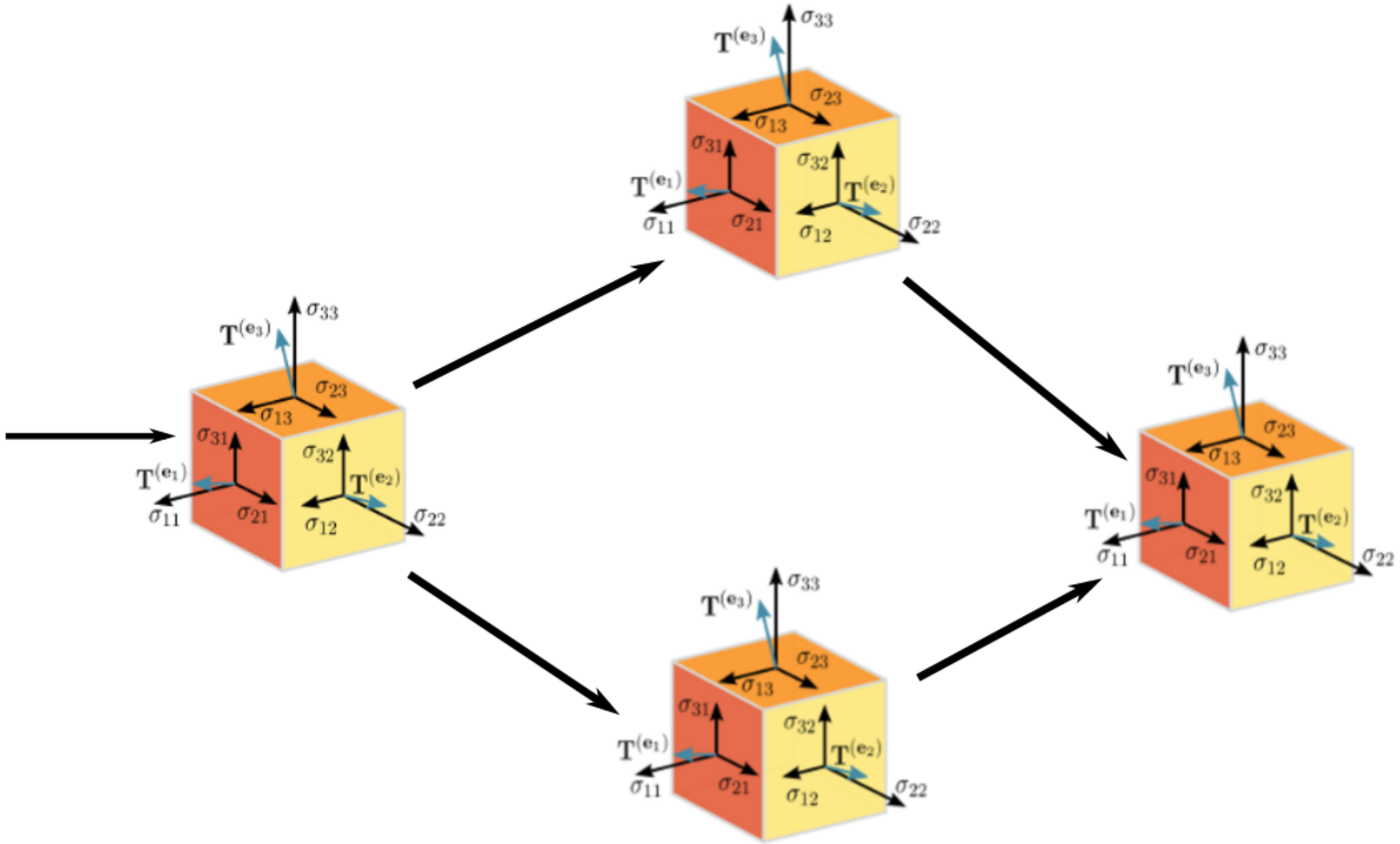


Important terms

- deep learning
- stochastic gradient descent
- batch and mini-batch learning
- epoch
- dropout

What is not TensorFlow

What is TensorFlow?



Keras tutorial

Keras-introduction.ipynb

Neural Network architectures design

Neural Network design best practices

- ⭐ **Start from simple architectures**
- ⭐ **Get inspiration from architectures for similar problems**
- ⭐ **Change one parameter only and then validate**

Common architectures

Feed forward network

Convolutional network

Recurrent network

Autoencoder network and Restricted Blotzmann Machines

Transformer network

U-Net

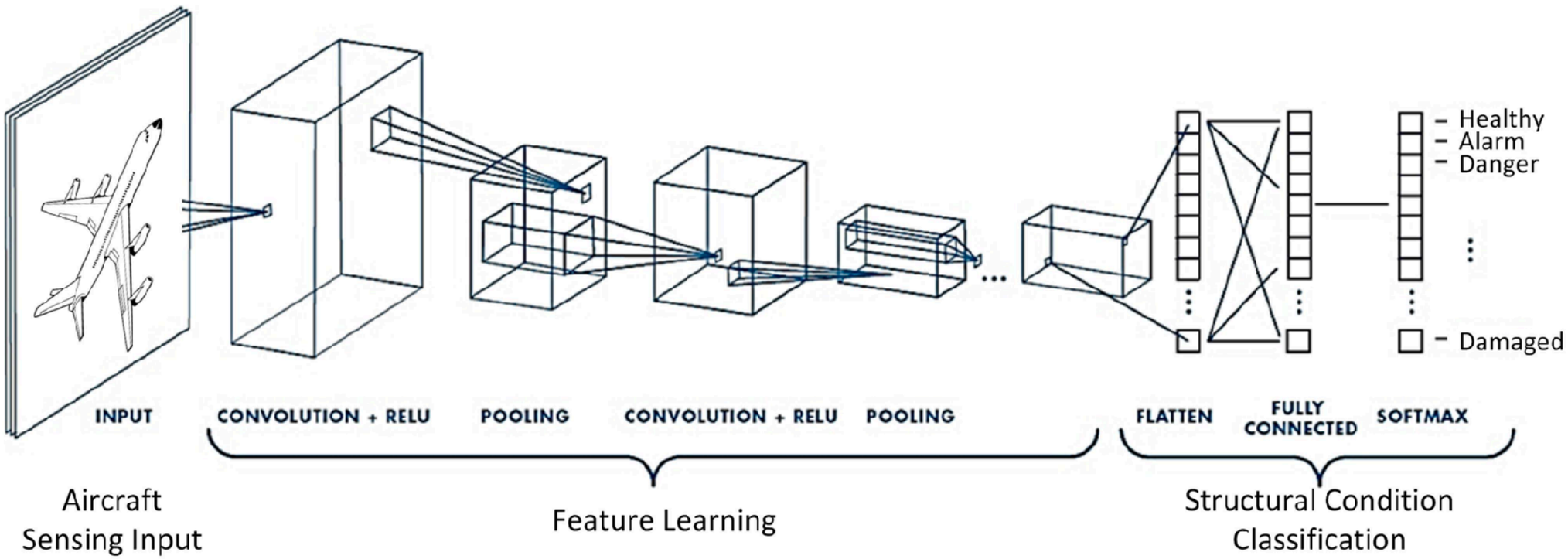
Generative adversarial network

Implemenation of a classification and regression task using NN

Classification-nn-assignment.ipynb

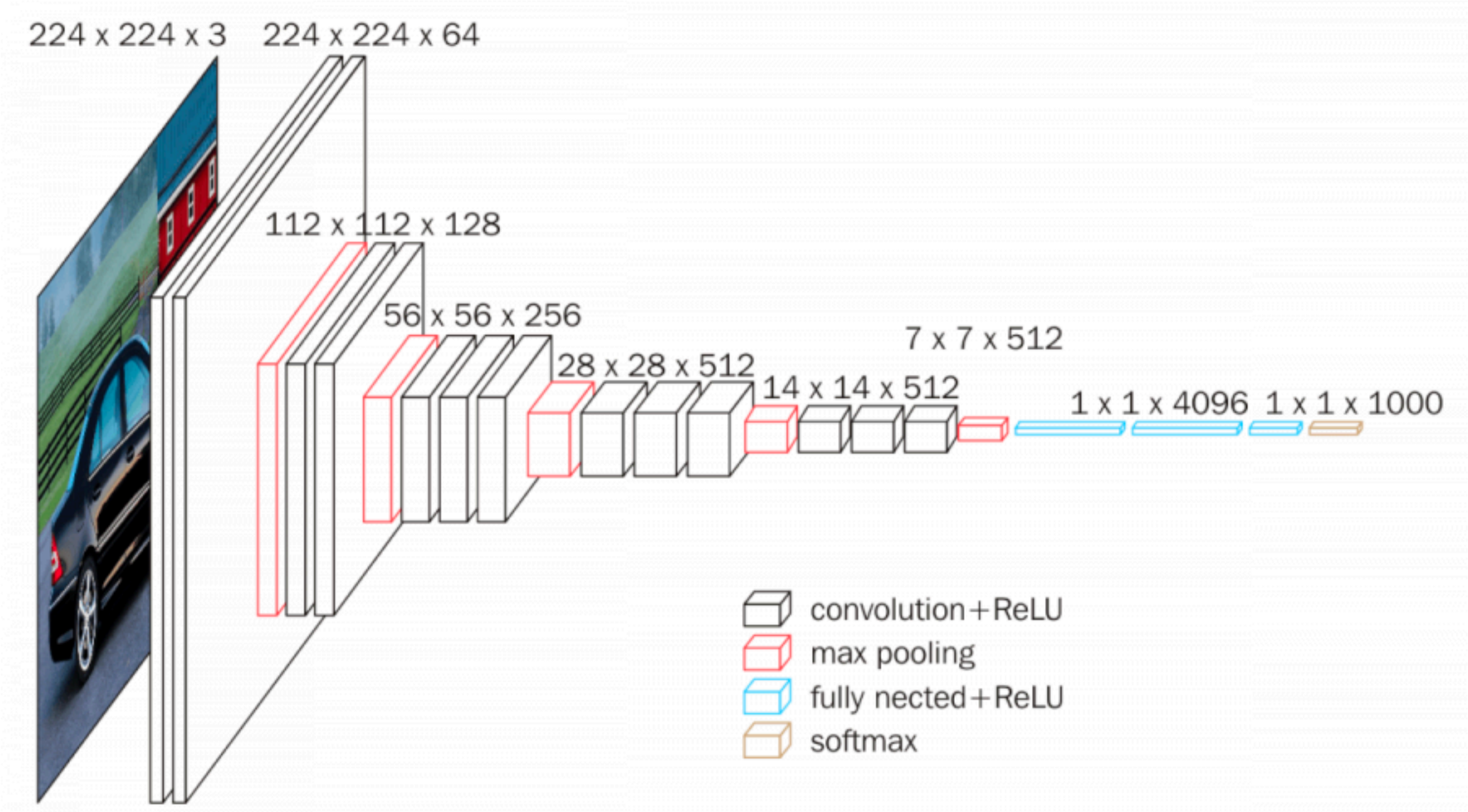
Regression-nn-assignment.ipynb

Convolutional Neural Network

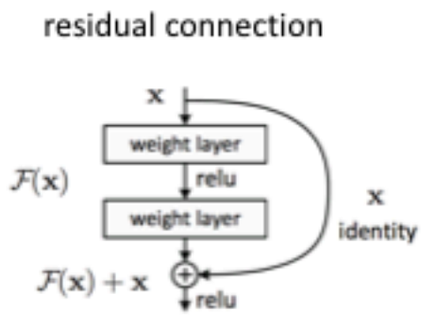


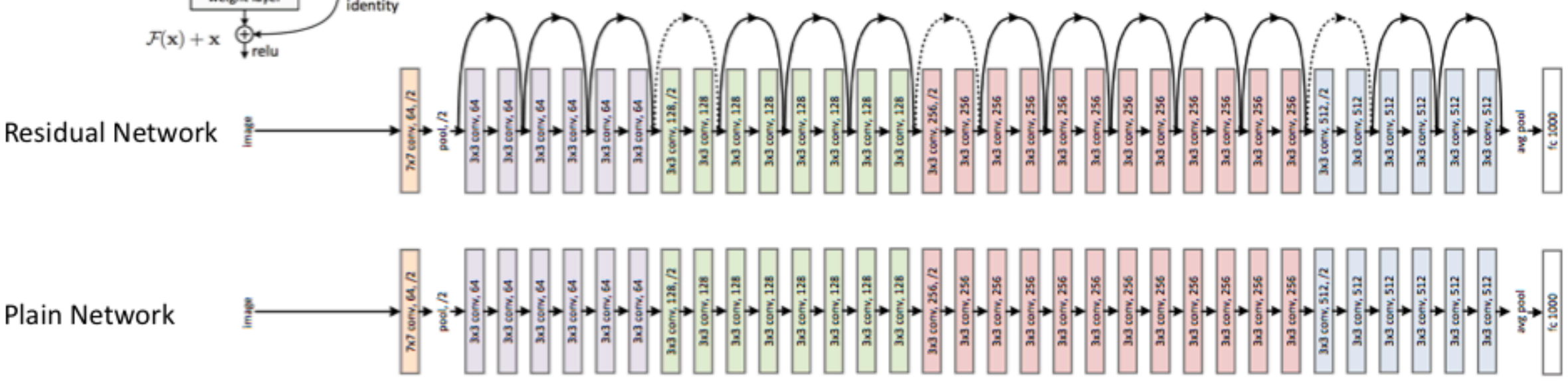
https://www.mdpi.com/sensors/sensors-19-04933/article_deploy/html/images/sensors-19-04933-g001.png

VGG 16

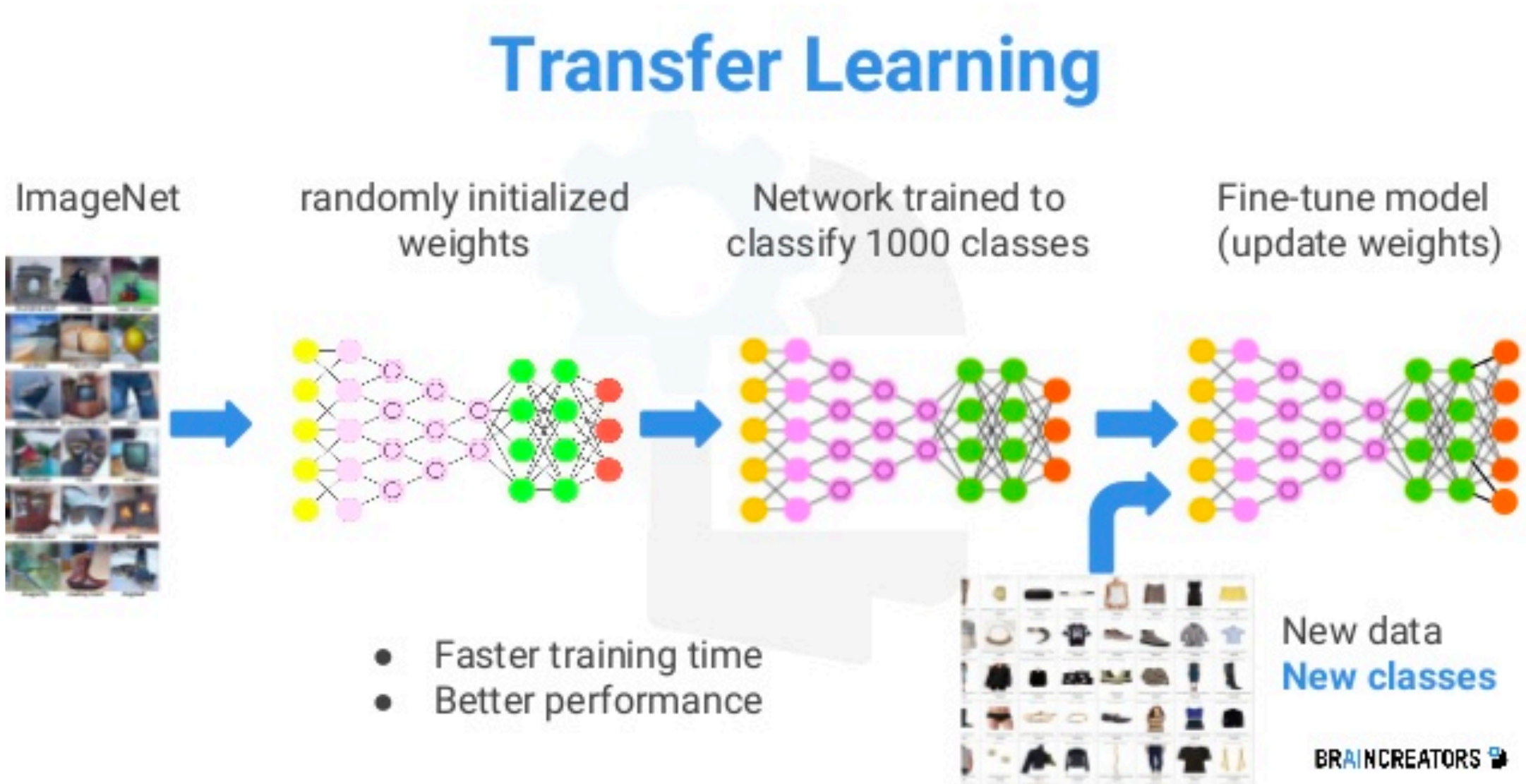

224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

ResNet


residual connection
x
weight layer
relu
weight layer
F(x)
x
identity
F(x) + x
relu


Residual Network
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000
Plain Network
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

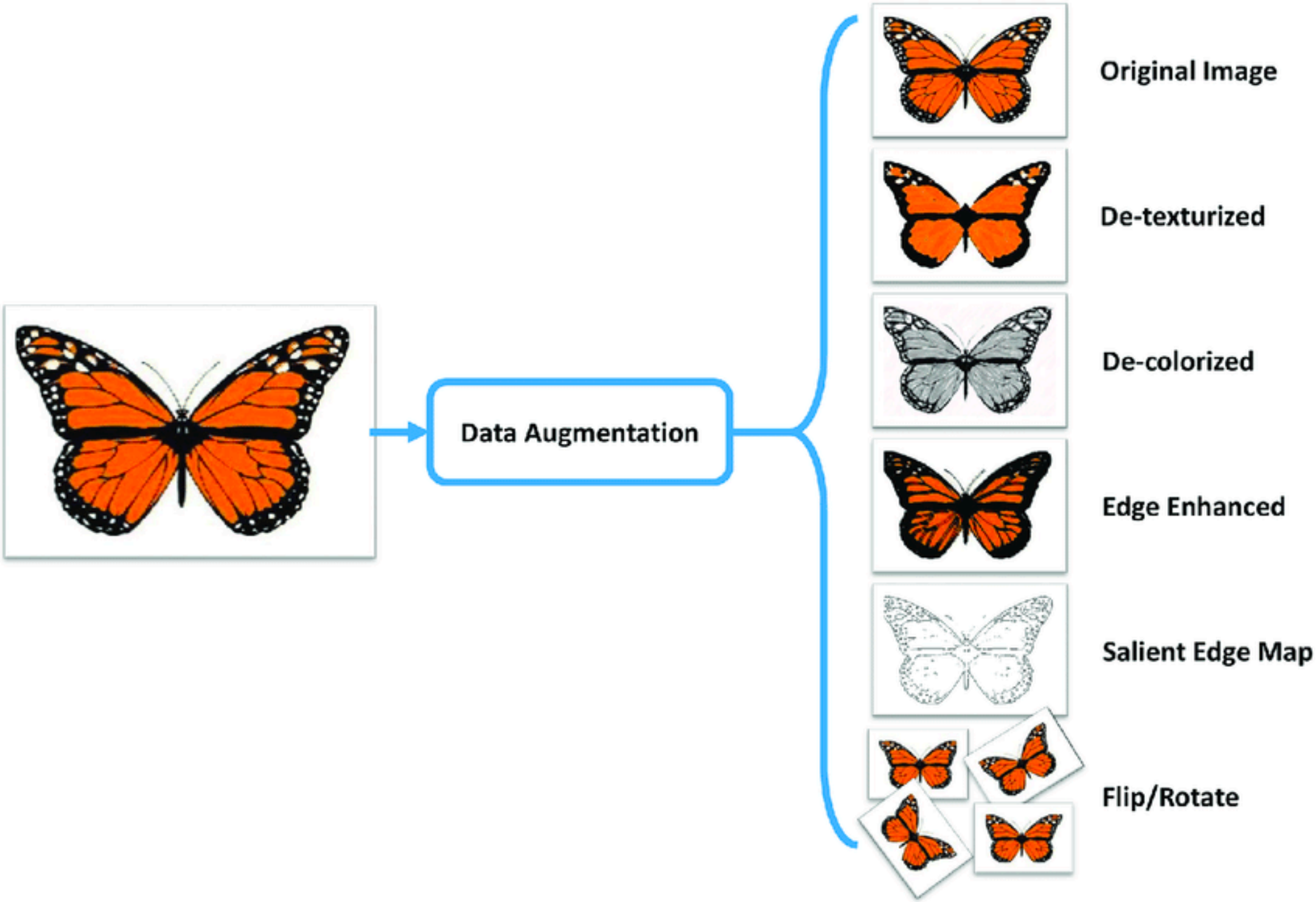
Finetuning


Transfer Learning
ImageNet
randomly initialized weights
Network trained to classify 1000 classes
Fine-tune model (update weights)
Faster training time
Better performance
New data
New classes
BRAINCREATORS

Transfer learning example

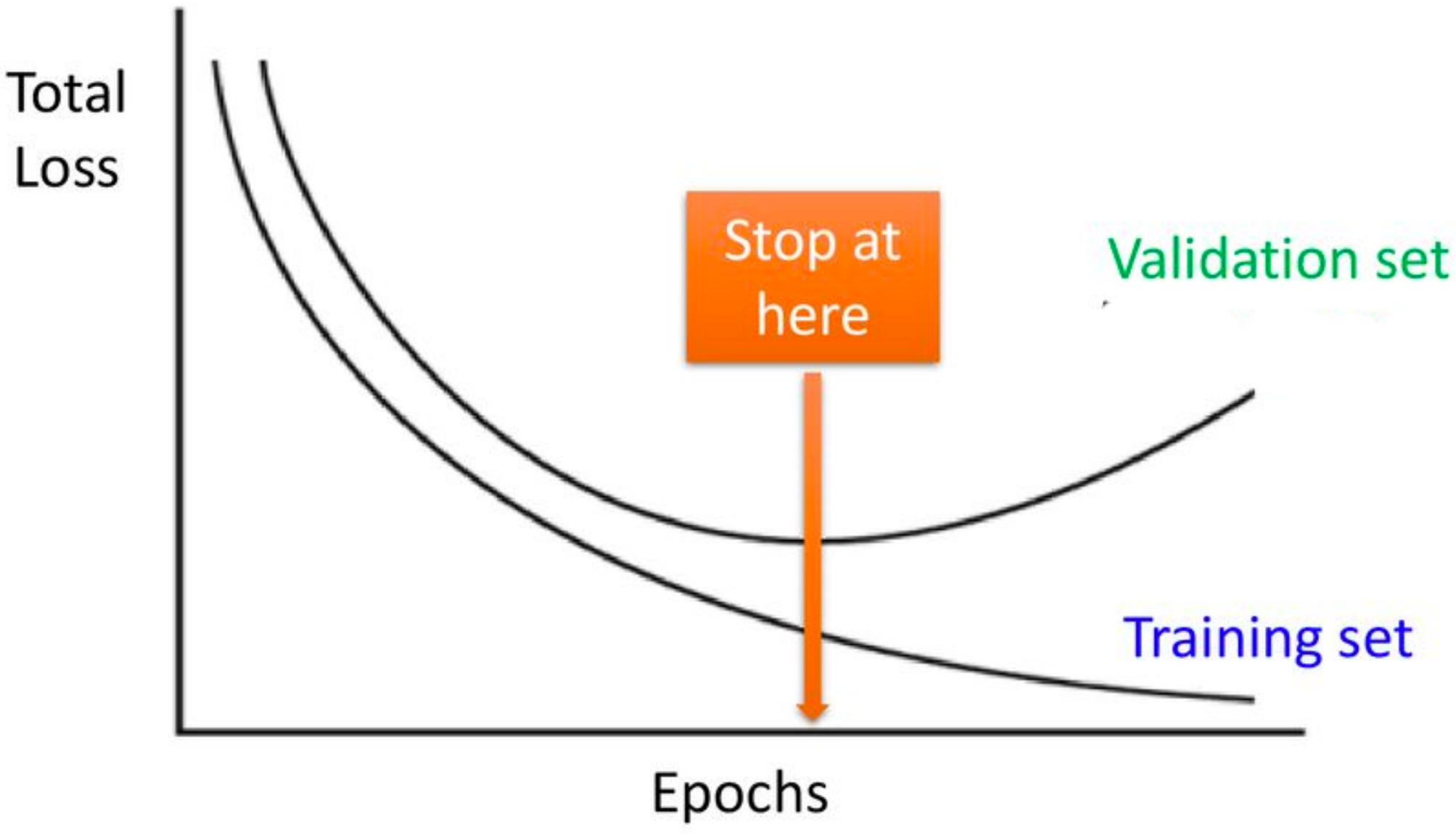
Transfer_learning.ipynb

Data augmentation



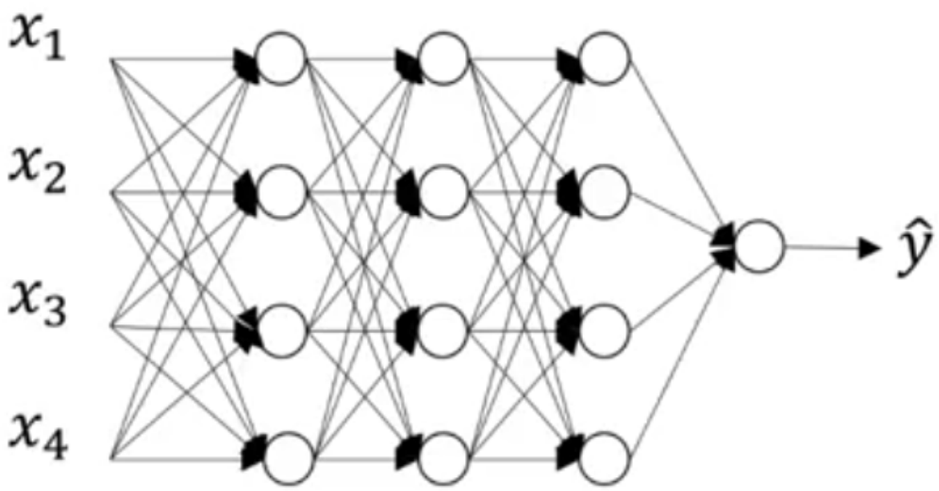
https://medium.com/secure-and-private-ai-writing-challenge/data-augmentation-increases-accuracy-of-your-model-but-how-aa1913468722

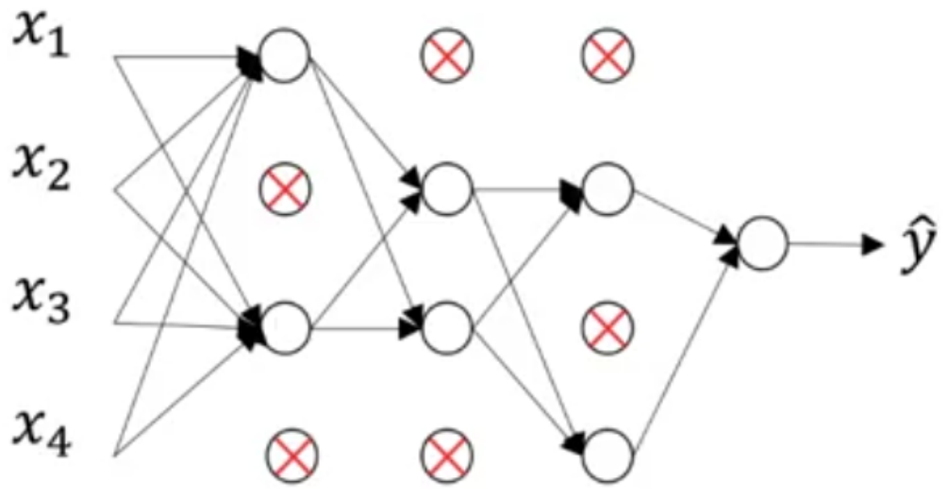
Early stopping



https://keras.io/getting_started/intro_to_keras_for_engineers/

Dropout





https://towardsdatascience.com/how-to-improve-a-neural-network-with-regularization-8a18ecda9fe3

L2 Regularization in deep learning

$$cost(w^1, b^1, \ldots, w^L, b^L) = \frac{1}{n}\sum_{i=1}^{n} Loss(y_i, \hat{y}_i) + \frac{\lambda}{2n}\sum_{l=1}^{L} \|w^l\|_F^2$$

Frobenius norm $$||w||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |w_{i,j}|^2}$$

Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

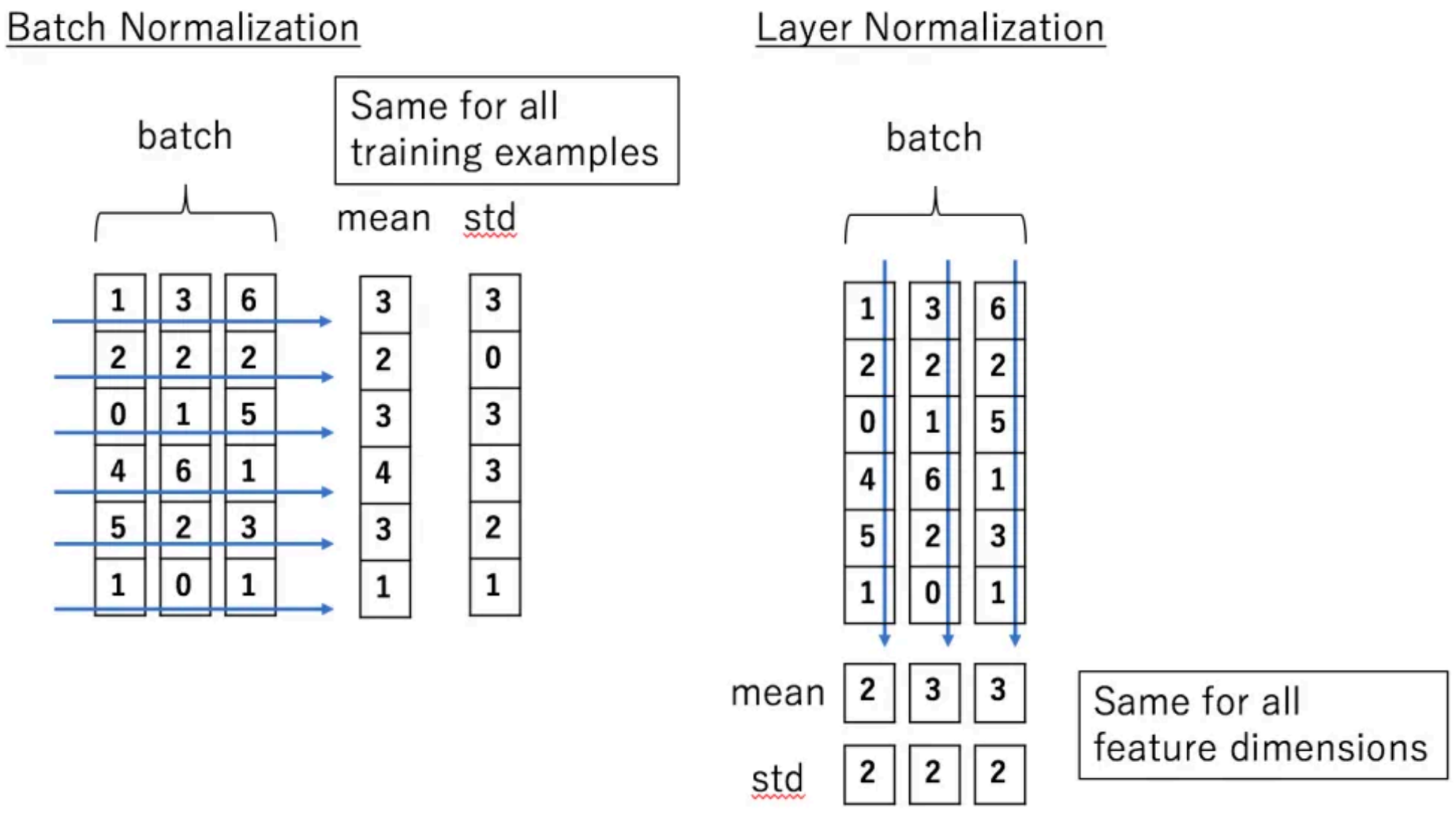
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

https://arxiv.org/abs/1502.03167

Layer normalization



https://mlexplained.com/2018/01/13/weight-normalization-and-layer-normalization-explained-normalization-in-deep-learning-part-2/

Functional API in Keras

```
# Sequential model
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(10, input_shape=(10,),activation='relu')
model.add(Dense(20, activation='relu')
model.add(Dense(10, activation='relu')
model.add(Dense(1, activation='sigmoid')
```

```
# Functional model
from keras.models import Model
from keras.layers import Input, Dense

visible = Input(shape=(10,))
hidden1 = Dense(10, activation='relu')(visible)
hidden2 = Dense(20, activation='relu')(hidden1)
hidden3 = Dense(10, activation='relu')(hidden2)
output = Dense(1, activation='sigmoid')(hidden3)
model = Model(inputs=visible, outputs=output)
```

Shared Input

```
# Shared Input Layer
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input, Dense, Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
# input layer
visible = Input(shape=(64,64,1))
# first feature extractor
conv1 = Conv2D(32, kernel_size=4, activation='relu')(visible)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
flat1 = Flatten()(pool1)
# second feature extractor
conv2 = Conv2D(16, kernel_size=8, activation='relu')(visible)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flat2 = Flatten()(pool2)
# merge feature extractors
merge = concatenate([flat1, flat2])
# interpretation layer
hidden1 = Dense(10, activation='relu')(merge)
# prediction output
output = Dense(1, activation='sigmoid')(hidden1)
model = Model(inputs=visible, outputs=output)
```

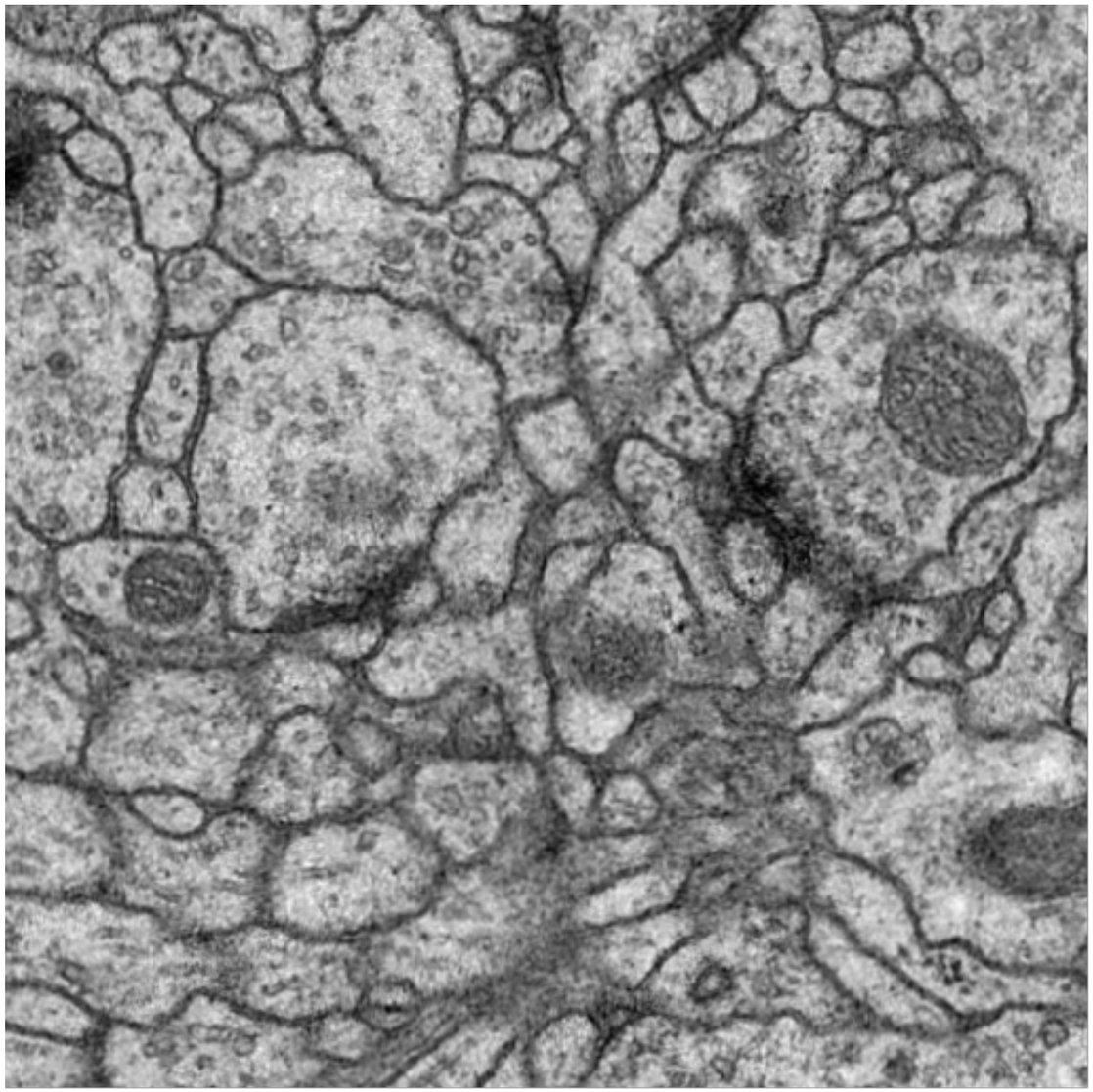
Multiple inputs (outputs)

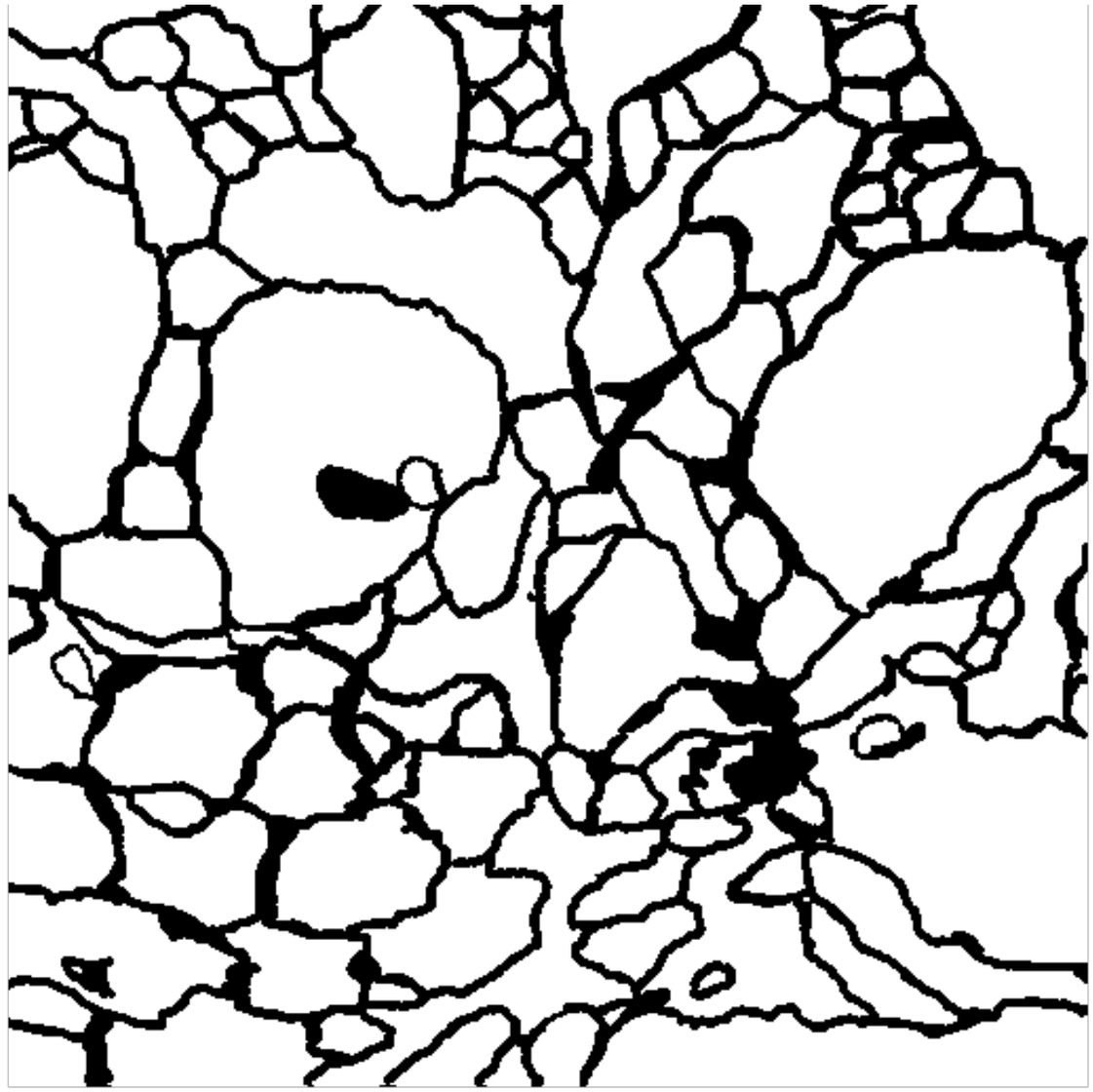
```python
# Multiple Inputs
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
# first input model
visible1 = Input(shape=(64,64,1))
conv11 = Conv2D(32, kernel_size=4, activation='relu')(visible1)
pool11 = MaxPooling2D(pool_size=(2, 2))(conv11)
conv12 = Conv2D(16, kernel_size=4, activation='relu')(pool11)
pool12 = MaxPooling2D(pool_size=(2, 2))(conv12)
flat1 = Flatten()(pool12)
# second input model
visible2 = Input(shape=(32,32,3))
conv21 = Conv2D(32, kernel_size=4, activation='relu')(visible2)
pool21 = MaxPooling2D(pool_size=(2, 2))(conv21)
conv22 = Conv2D(16, kernel_size=4, activation='relu')(pool21)
pool22 = MaxPooling2D(pool_size=(2, 2))(conv22)
flat2 = Flatten()(pool22)
# merge input models
merge = concatenate([flat1, flat2])
# interpretation model
hidden1 = Dense(10, activation='relu')(merge)
hidden2 = Dense(10, activation='relu')(hidden1)
output = Dense(1, activation='sigmoid')(hidden2)
model = Model(inputs=[visible1, visible2], outputs=output)
```

Practical example on regularization and normalization

Normalization-and-regularization-assignment.ipynb

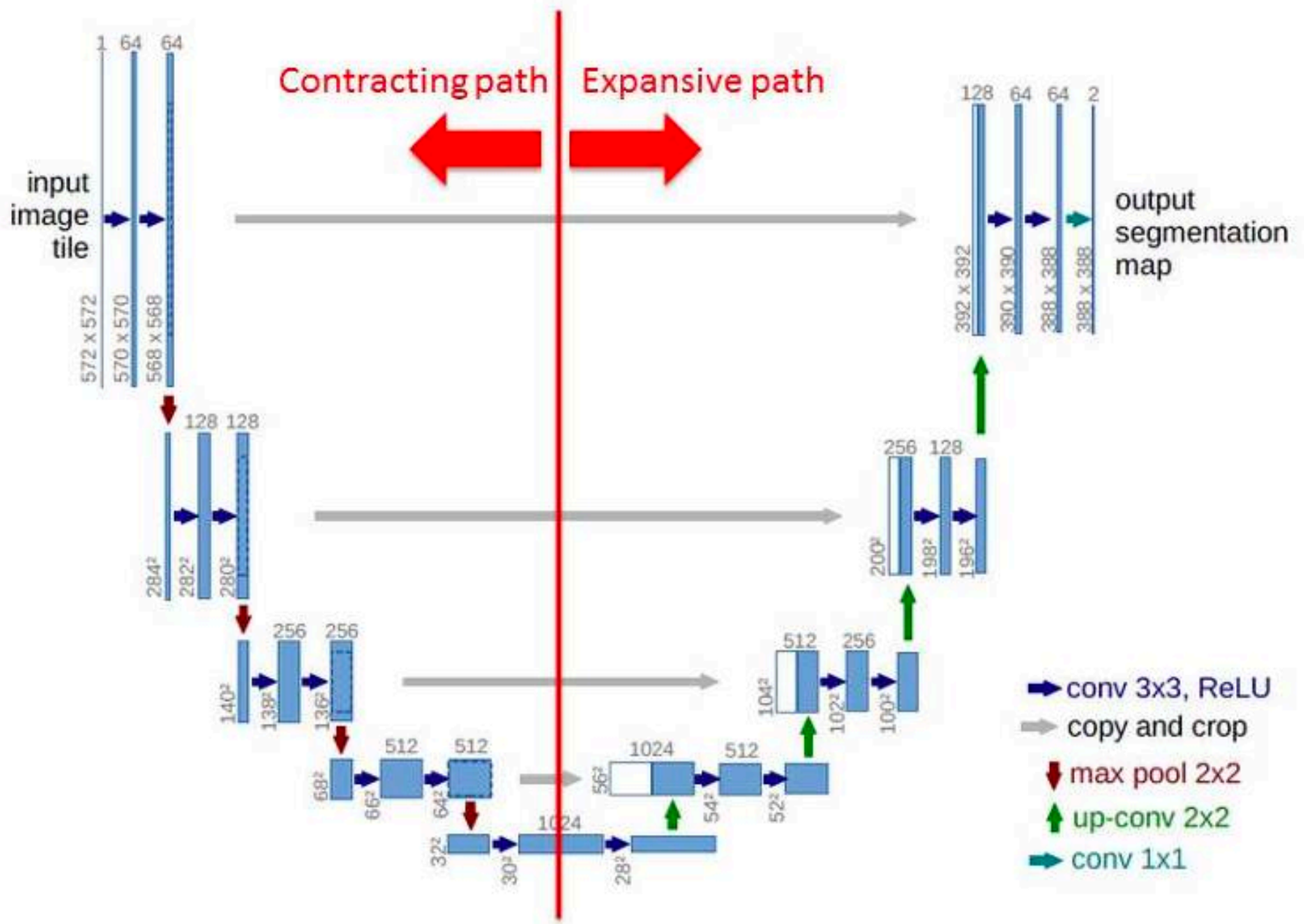
Image segmentation





U-Net

Network Architecture



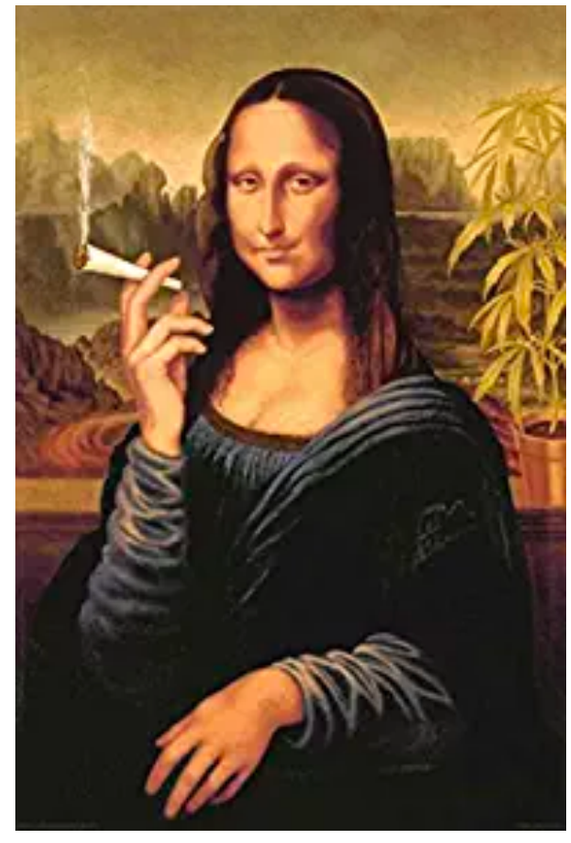
U-Net segmentation example

Segmentation.ipynb

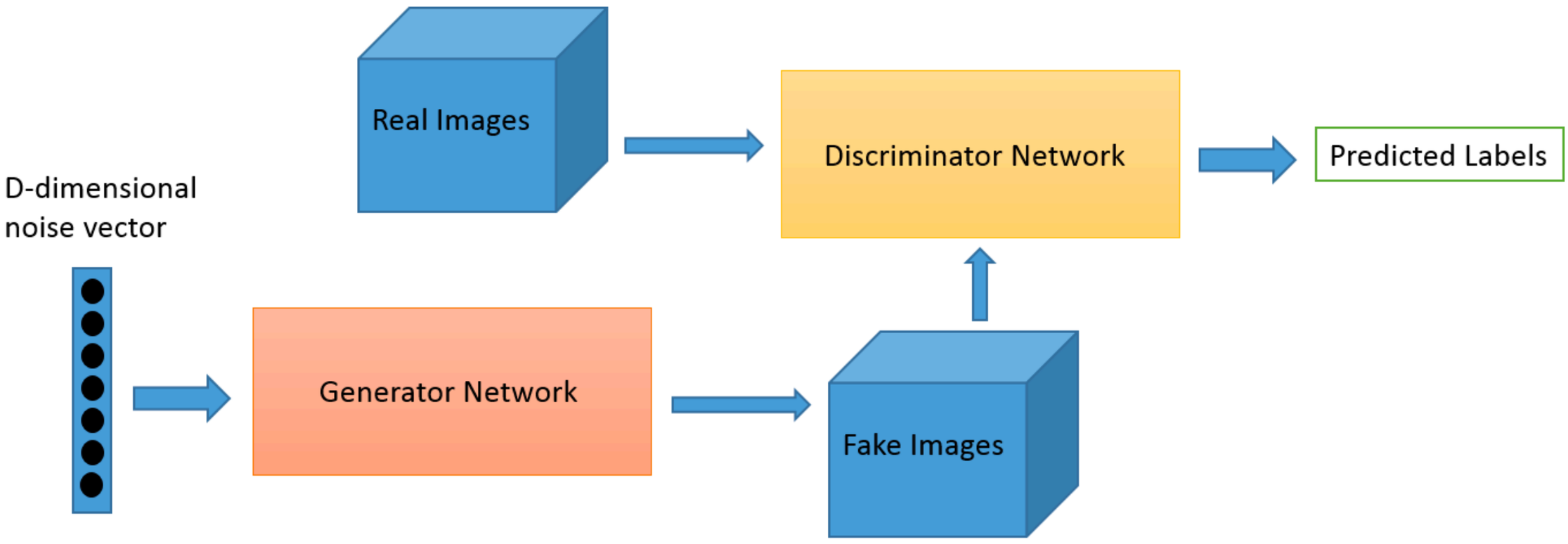
Generative models with neural networks

Random noice

Artificial Neural network



Generative Adversarial Networks



Superresolution

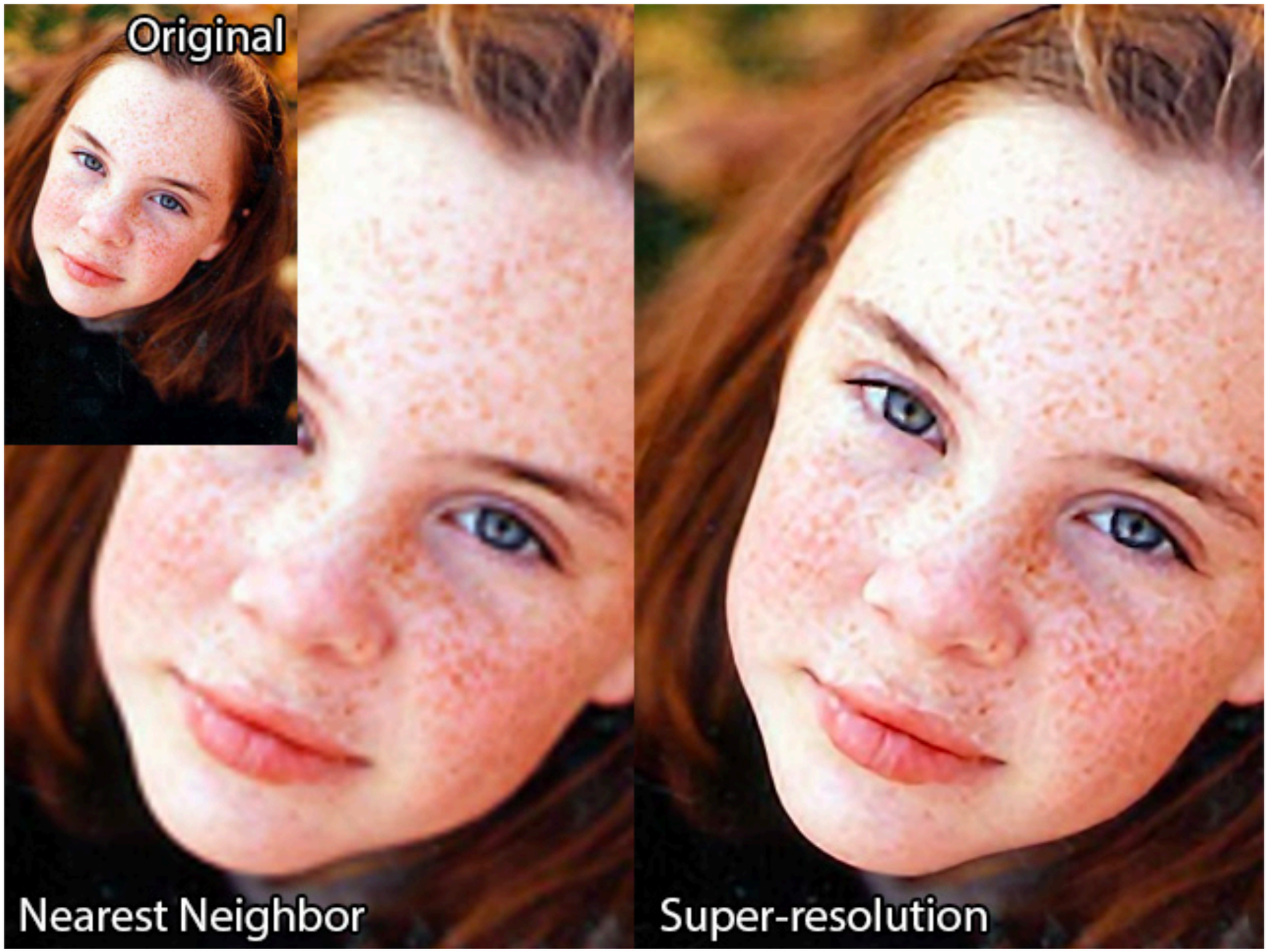
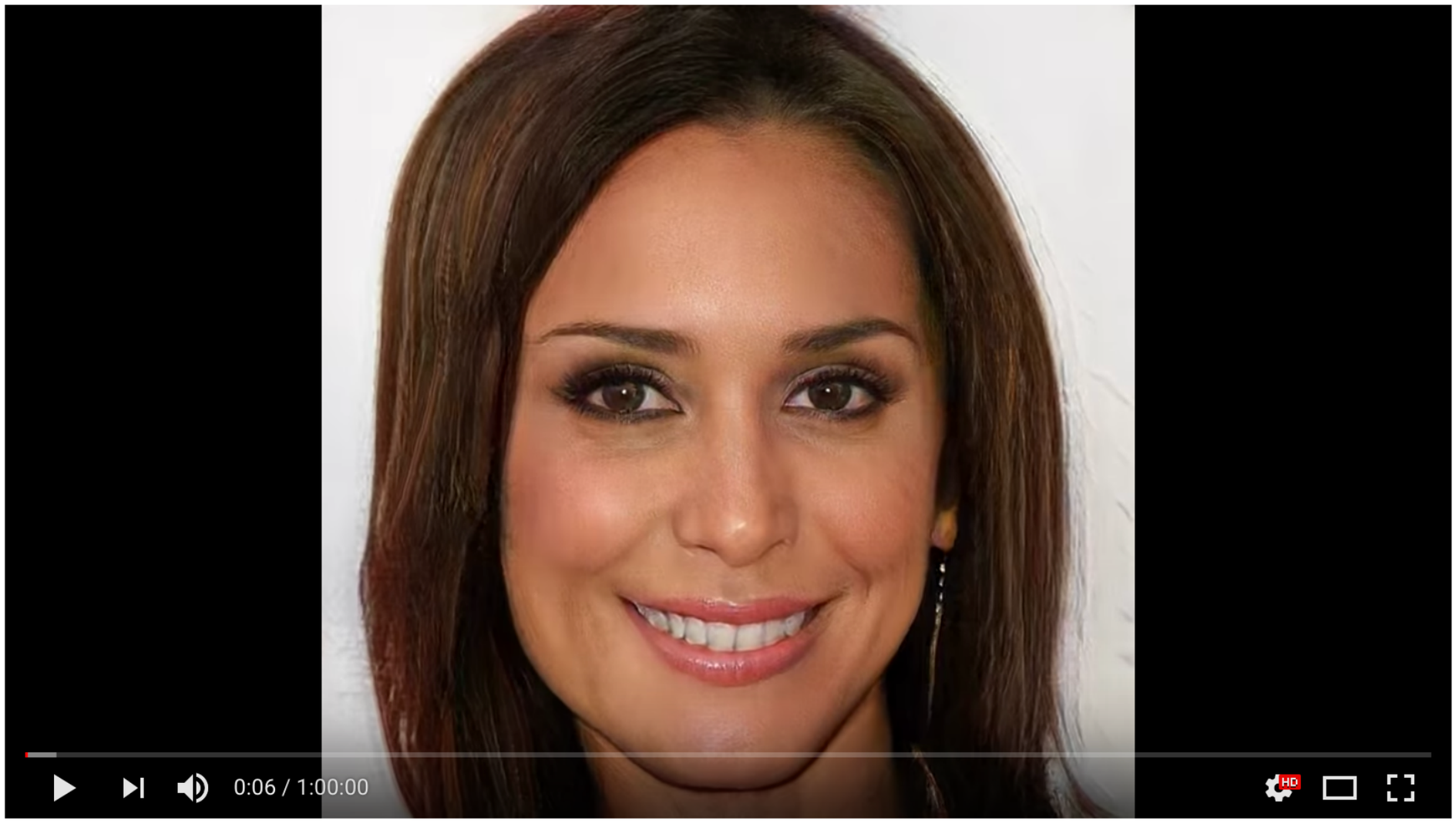


Image synthesis

YouTube CZ

Hledat



One hour of imaginary celebrities

95 832 zhlédnutí

TO SE MI LÍBÍ NELÍBÍ SE SDÍLET

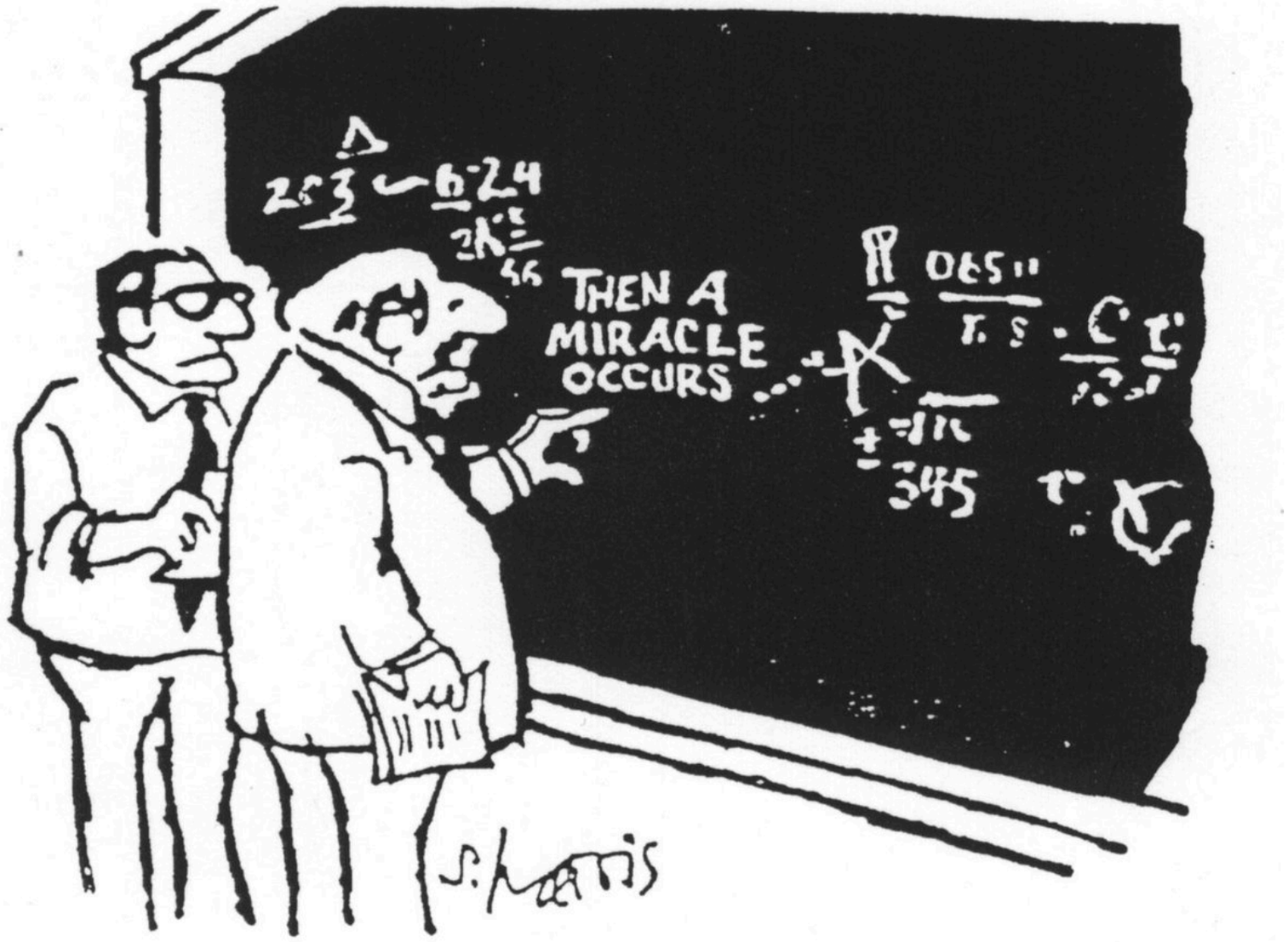
Which one is fake?

Generative Adversarial Networks

GANs.ipynb

Image manipulation

Neural network explainability

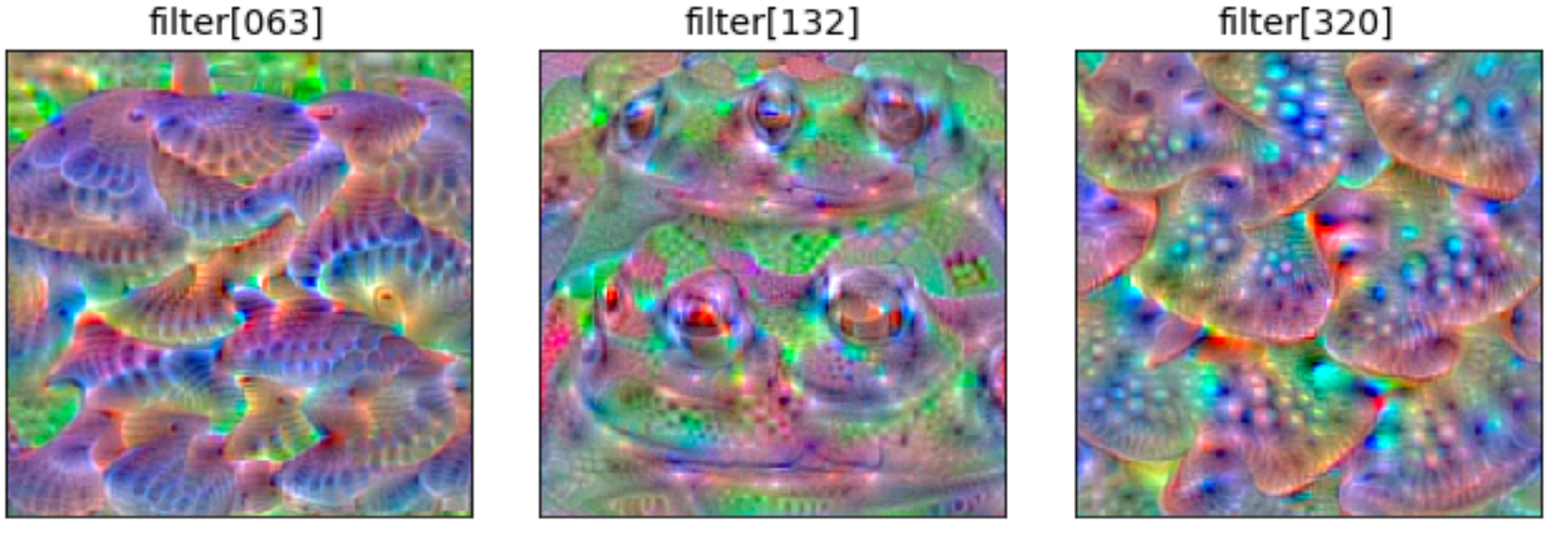


https://analyticsindiamag.com/deep-learning-is-not-so-much-a-black-box-anymore-and-thats-a-great-development/

Activation Maximization

Visualized output classification Layer

Visualized hidden convolutional layers

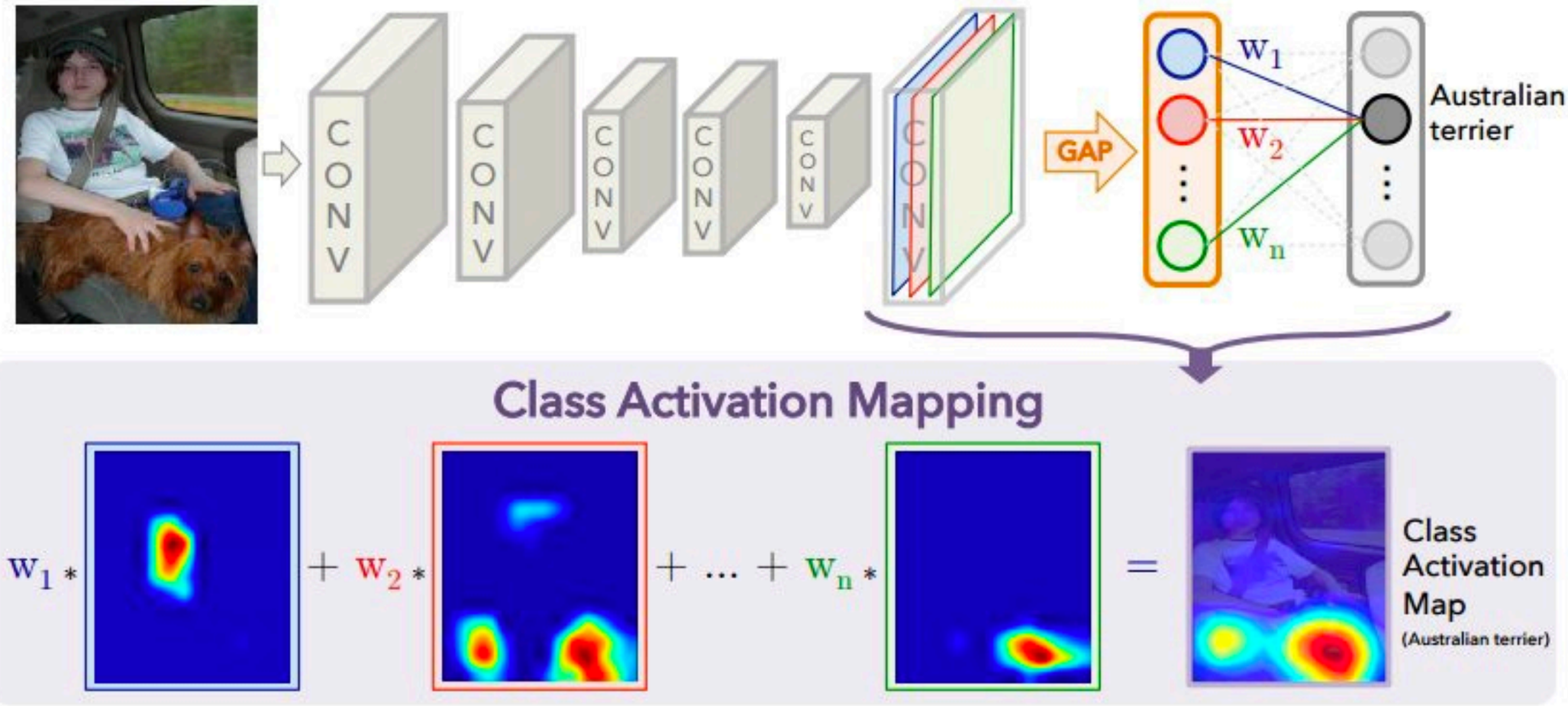


https://pypi.org/project/tf-keras-vis/

Grad-CAM heat maps

https://keras.io/examples/vision/grad_cam/

CAM heat maps

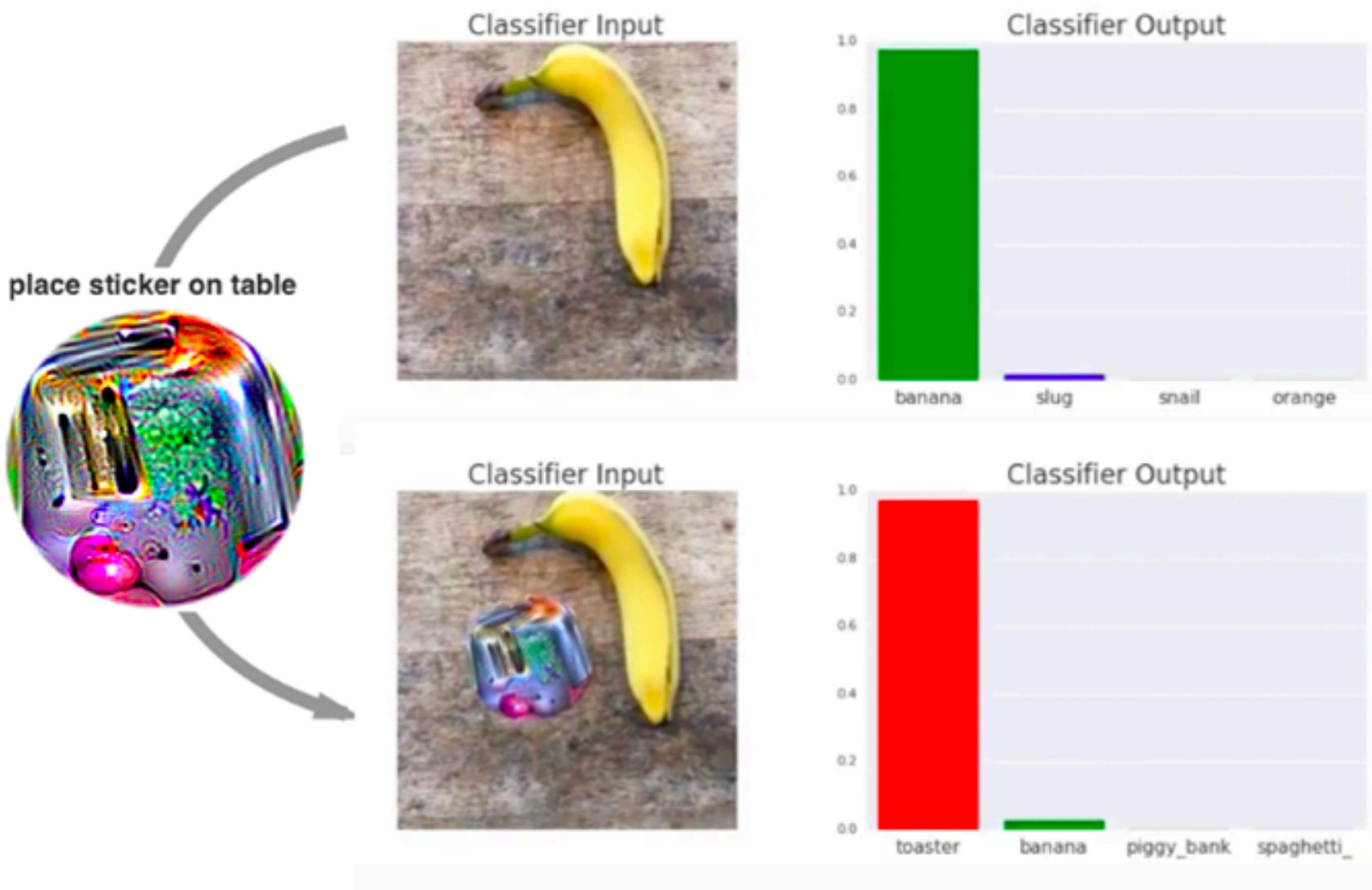


https://towardsdatascience.com/demystifying-convolutional-neural-networks-using-class-activation-maps-fe94eda4cef1

Convolutional Neural Networks Explainability in Keras

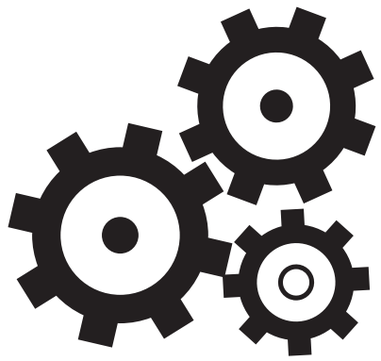
Explainability.ipynb

Adversarial Patch



What next?

https://www.mlcollege.com/en/#courses



Machine Learning Prague

MACHINE LEARNING
meetups

Thank you for your attention

e-mail: jiri@mlcollege.com

Web: www.mlcollege.com

Twitter: @JiriMaterna

Facebook: https://www.facebook.com/maternajiri

LinkedIn: https://www.linkedin.com/in/jirimaterna/