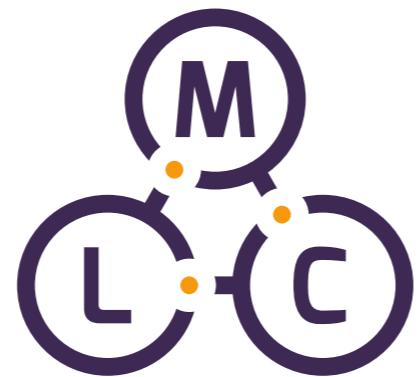


Convolutional Neural Networks and Image Processing

Jiří Materna



Machine
Learning
College

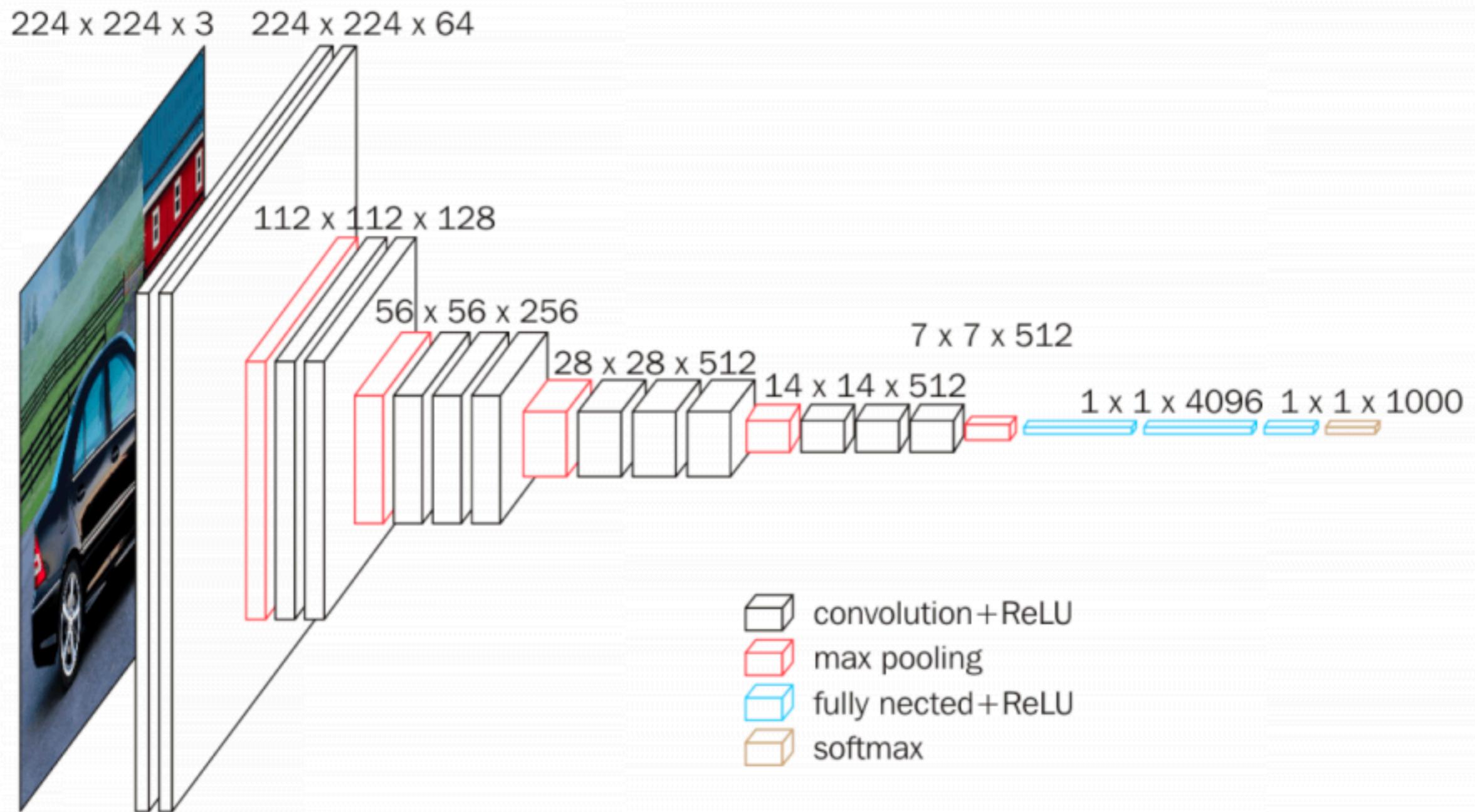
About me

- 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-organiser of ML Prague
- ML Freelace and consultant

Image processing

- 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

VGG 16

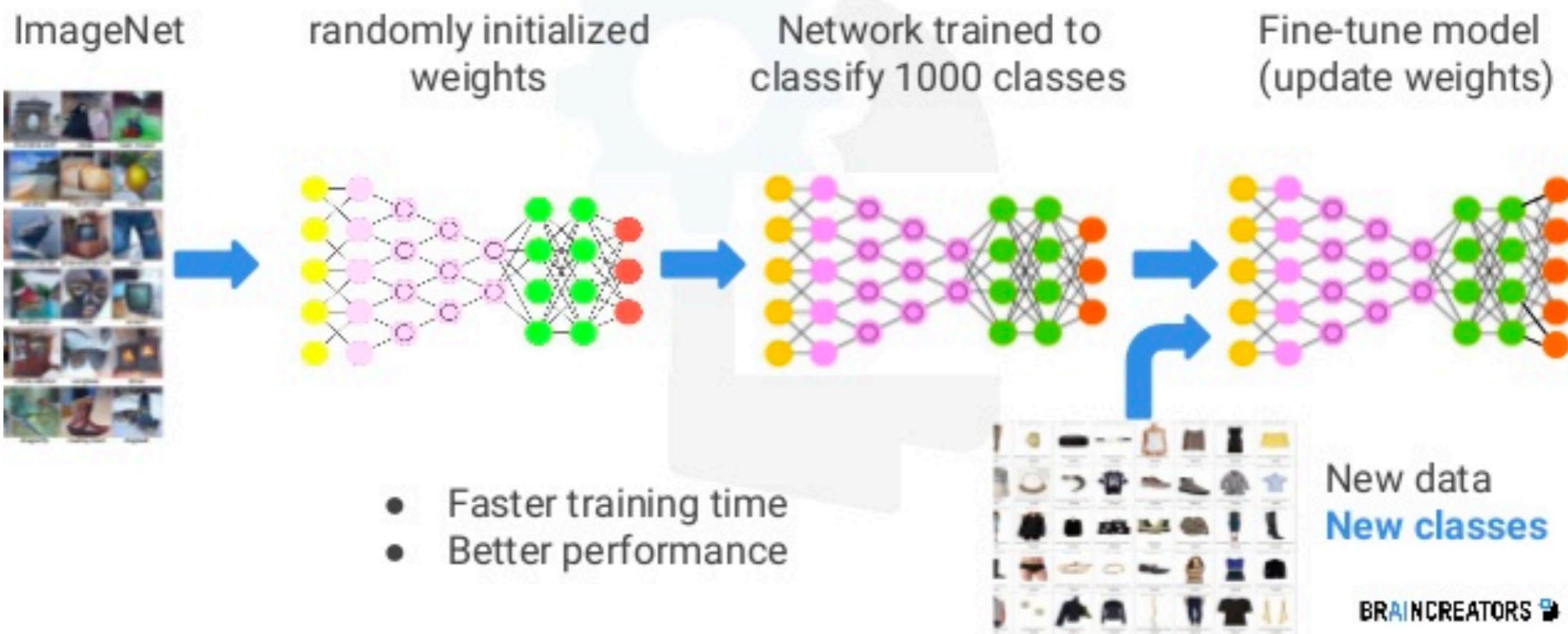


ResNet



Finetuning

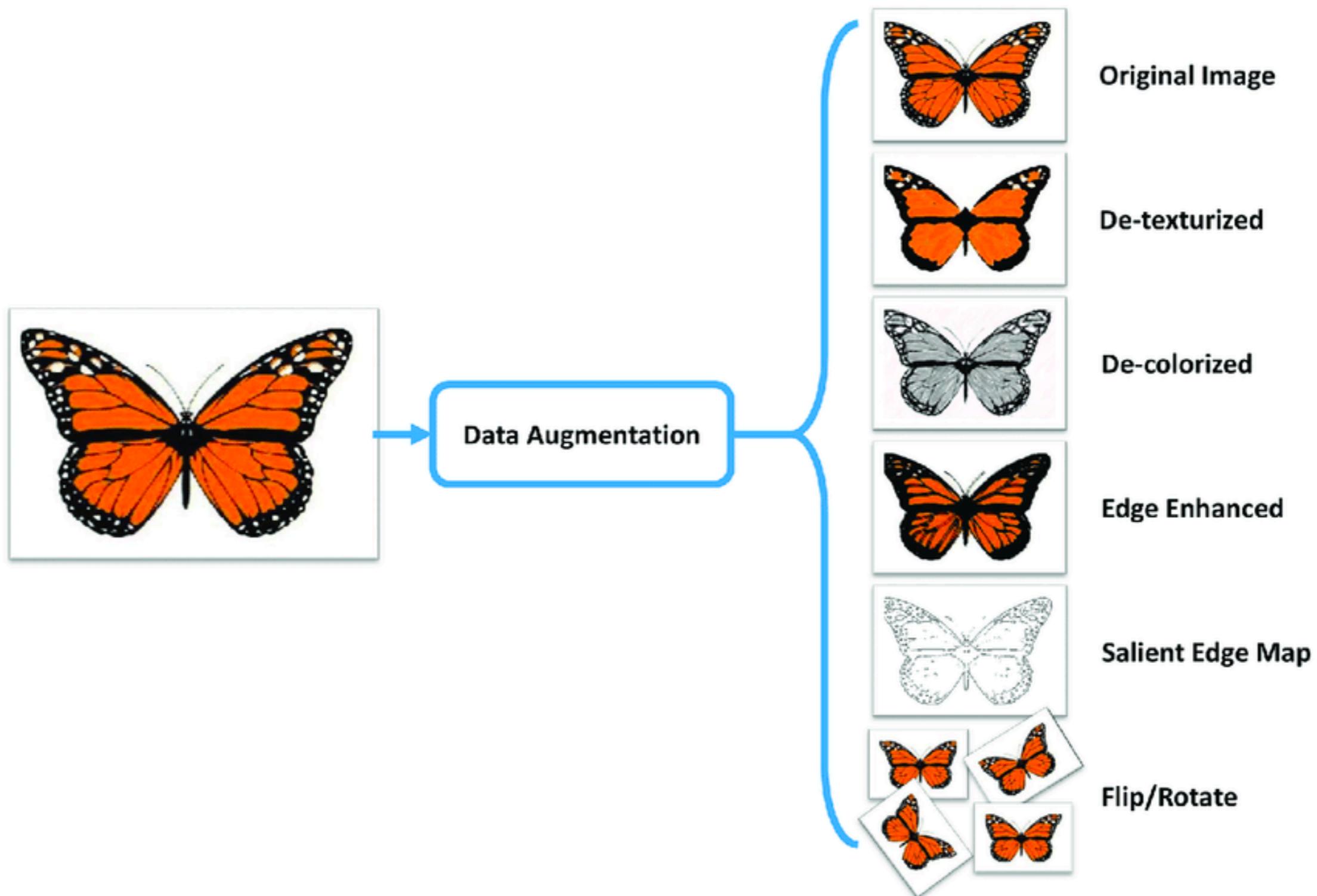
Transfer Learning



Transfer learning example

01-Transfer_learning.ipynb

Data augmentation



Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;

Parameters to be learned: γ, β

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

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

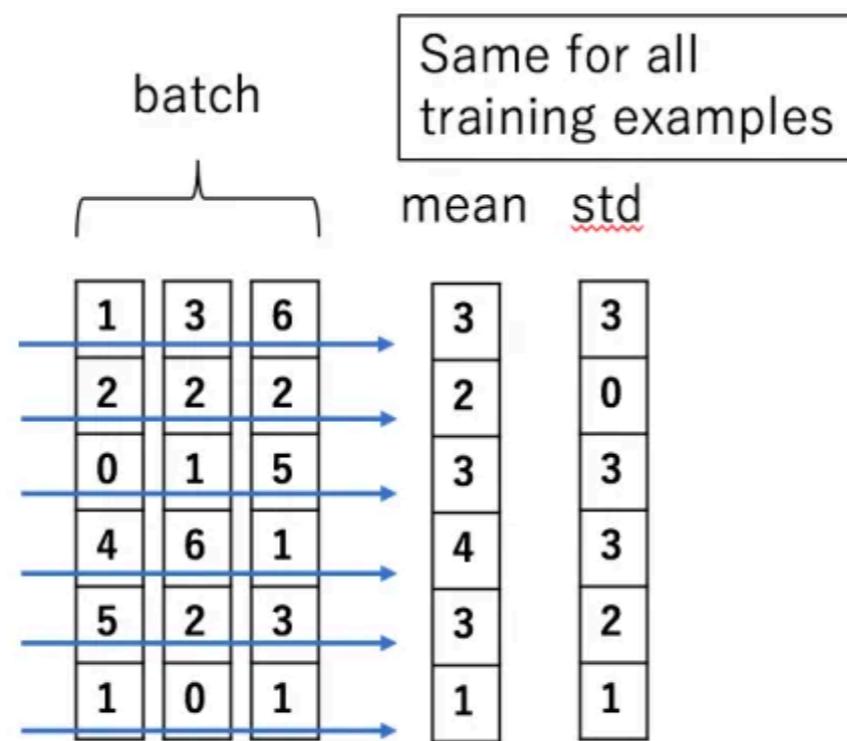
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

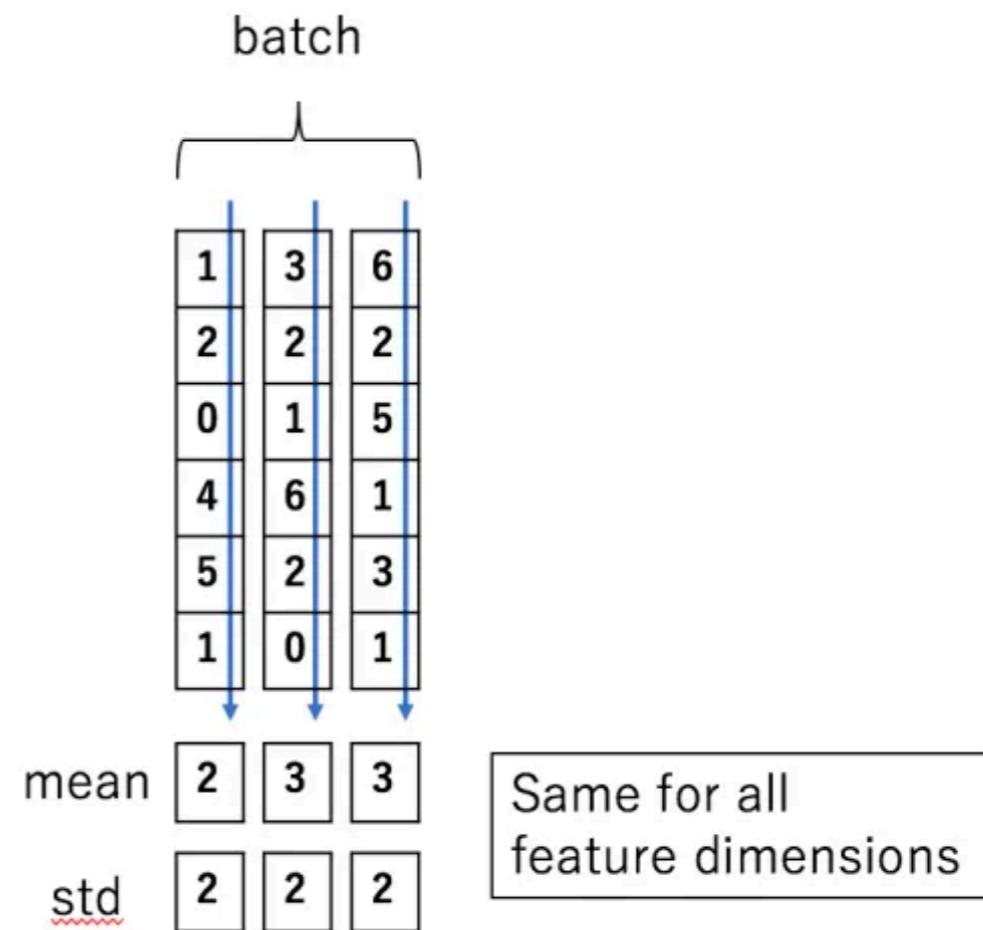
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Layer normalization

Batch Normalization



Layer Normalization



Functional API in Keras

```
1 # Sequential model
2 from keras.models import Sequential
3 from keras.layers import Dense
4
5 model = Sequential()
6 model.add(10, input_shape=(10,), activation='relu')
7 model.add(Dense(20, activation='relu'))
8 model.add(Dense(10, activation='relu'))
9 model.add(Dense(1, activation='sigmoid'))
```

```
1 # Functional model
2 from keras.models import Model
3 from keras.layers import Input, Dense
4
5 visible = Input(shape=(10,))
6 hidden1 = Dense(10, activation='relu')(visible)
7 hidden2 = Dense(20, activation='relu')(hidden1)
8 hidden3 = Dense(10, activation='relu')(hidden2)
9 output = Dense(1, activation='sigmoid')(hidden3)
10 model = Model(inputs=visible, outputs=output)
```

Shared Input

```
1 # Shared Input Layer
2 from keras.utils import plot_model
3 from keras.models import Model
4 from keras.layers import Input, Dense, Flatten
5 from keras.layers.convolutional import Conv2D
6 from keras.layers.pooling import MaxPooling2D
7 from keras.layers.merge import concatenate
8 # input layer
9 visible = Input(shape=(64,64,1))
10 # first feature extractor
11 conv1 = Conv2D(32, kernel_size=4, activation='relu')(visible)
12 pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
13 flat1 = Flatten()(pool1)
14 # second feature extractor
15 conv2 = Conv2D(16, kernel_size=8, activation='relu')(visible)
16 pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
17 flat2 = Flatten()(pool2)
18 # merge feature extractors
19 merge = concatenate([flat1, flat2])
20 # interpretation layer
21 hidden1 = Dense(10, activation='relu')(merge)
22 # prediction output
23 output = Dense(1, activation='sigmoid')(hidden1)
24 model = Model(inputs=visible, outputs=output)
```

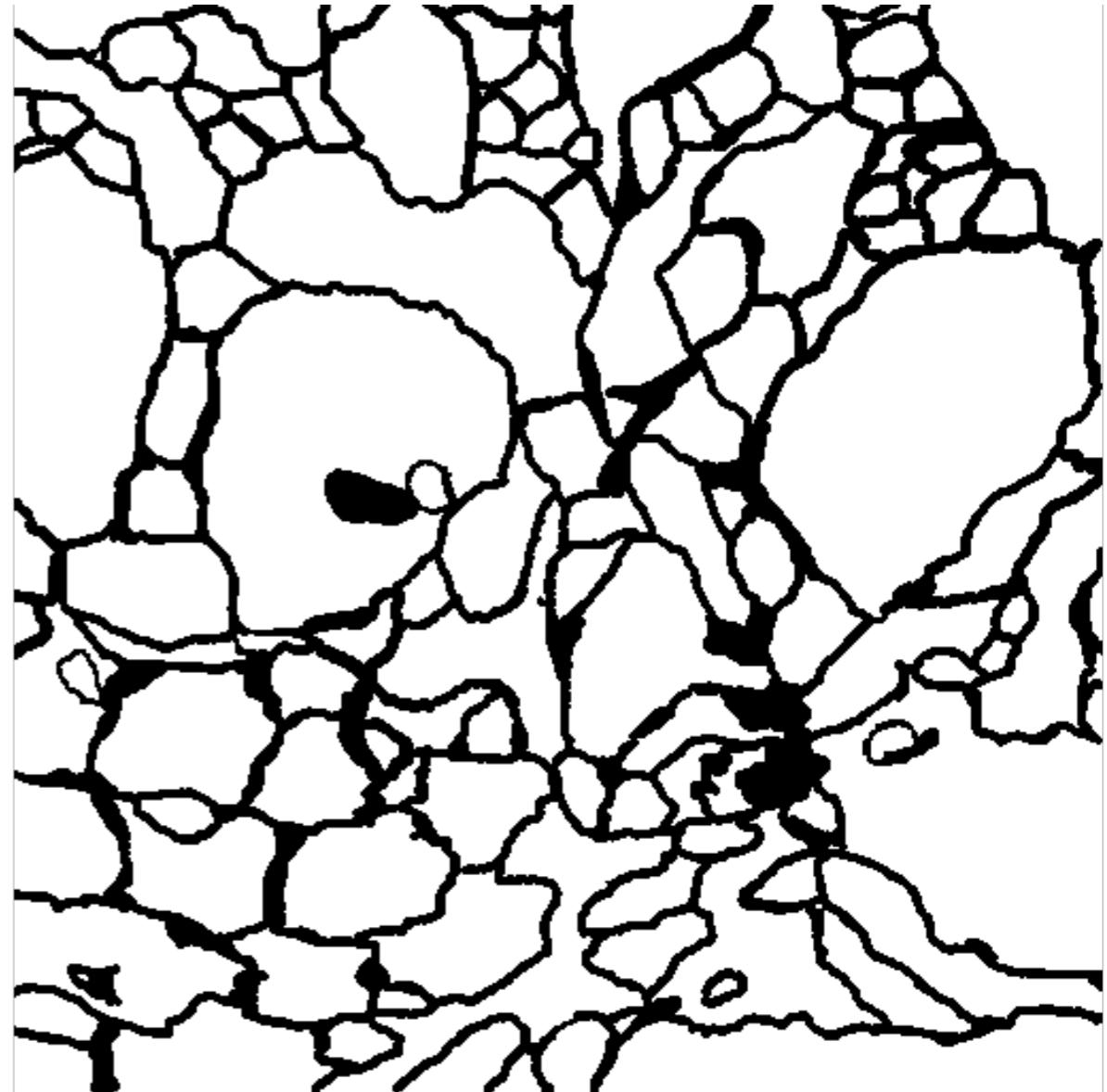
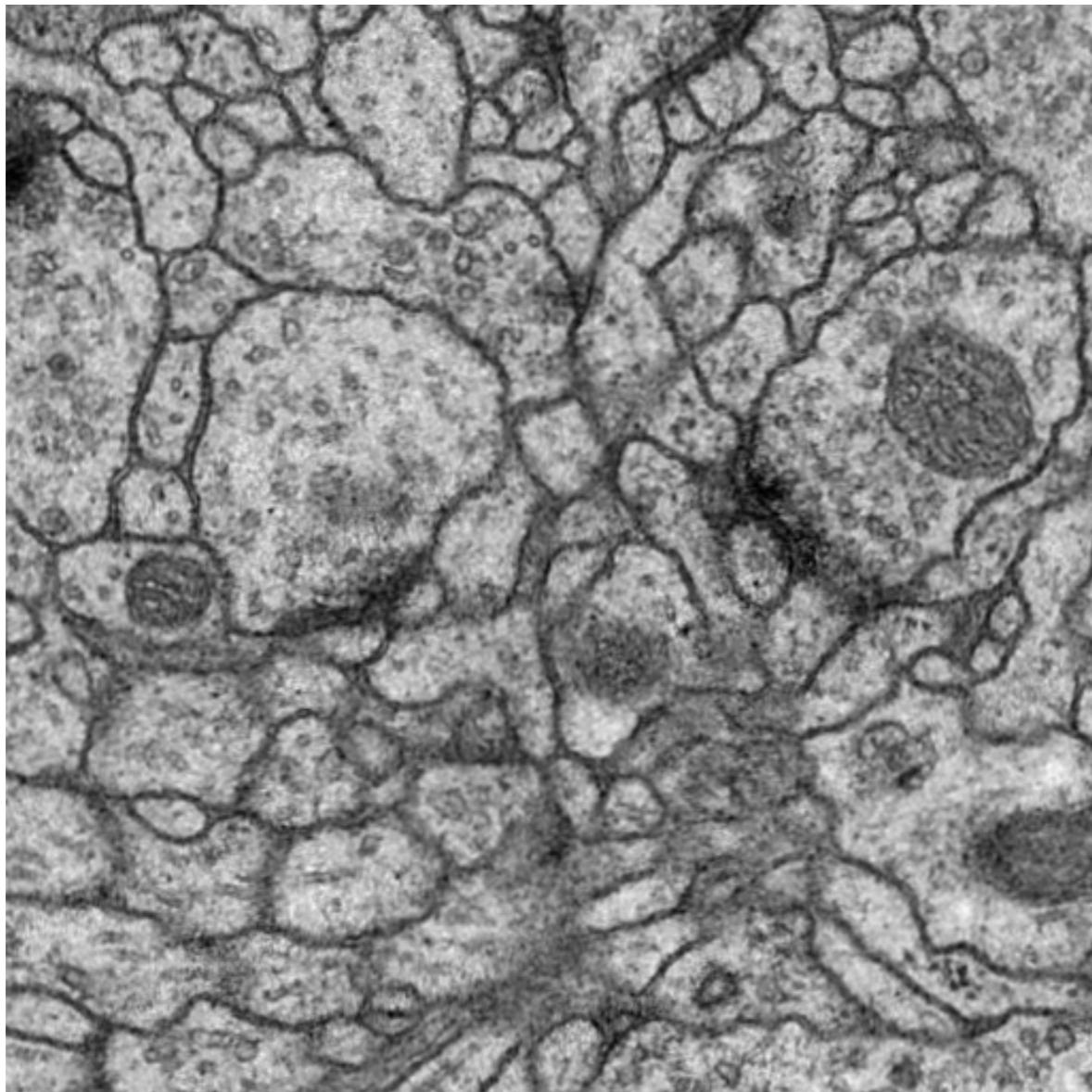
Multiple inputs (outputs)

```
1 # Multiple Inputs
2 from keras.utils import plot_model
3 from keras.models import Model
4 from keras.layers import Input
5 from keras.layers import Dense
6 from keras.layers import Flatten
7 from keras.layers.convolutional import Conv2D
8 from keras.layers.pooling import MaxPooling2D
9 from keras.layers.merge import concatenate
10 # first input model
11 visible1 = Input(shape=(64,64,1))
12 conv11 = Conv2D(32, kernel_size=4, activation='relu')(visible1)
13 pool11 = MaxPooling2D(pool_size=(2, 2))(conv11)
14 conv12 = Conv2D(16, kernel_size=4, activation='relu')(pool11)
15 pool12 = MaxPooling2D(pool_size=(2, 2))(conv12)
16 flat1 = Flatten()(pool12)
17 # second input model
18 visible2 = Input(shape=(32,32,3))
19 conv21 = Conv2D(32, kernel_size=4, activation='relu')(visible2)
20 pool21 = MaxPooling2D(pool_size=(2, 2))(conv21)
21 conv22 = Conv2D(16, kernel_size=4, activation='relu')(pool21)
22 pool22 = MaxPooling2D(pool_size=(2, 2))(conv22)
23 flat2 = Flatten()(pool22)
24 # merge input models
25 merge = concatenate([flat1, flat2])
26 # interpretation model
27 hidden1 = Dense(10, activation='relu')(merge)
28 hidden2 = Dense(10, activation='relu')(hidden1)
29 output = Dense(1, activation='sigmoid')(hidden2)
30 model = Model(inputs=[visible1, visible2], outputs=output)
```

Practical example on regularization and normalization

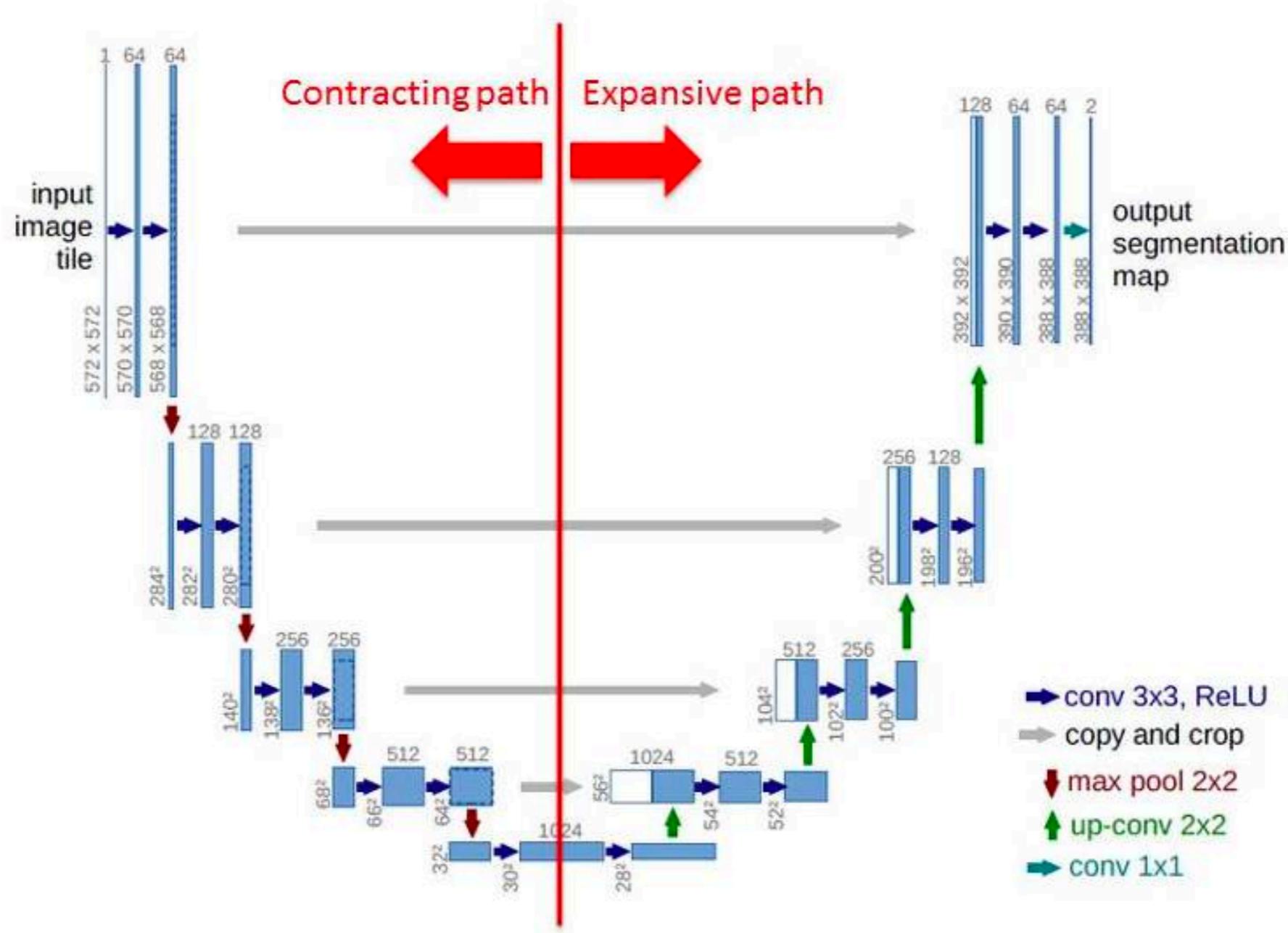
[**02-Normalization-and-regularization-assignment.ipynb**](#)

Image segmentation



U-Net

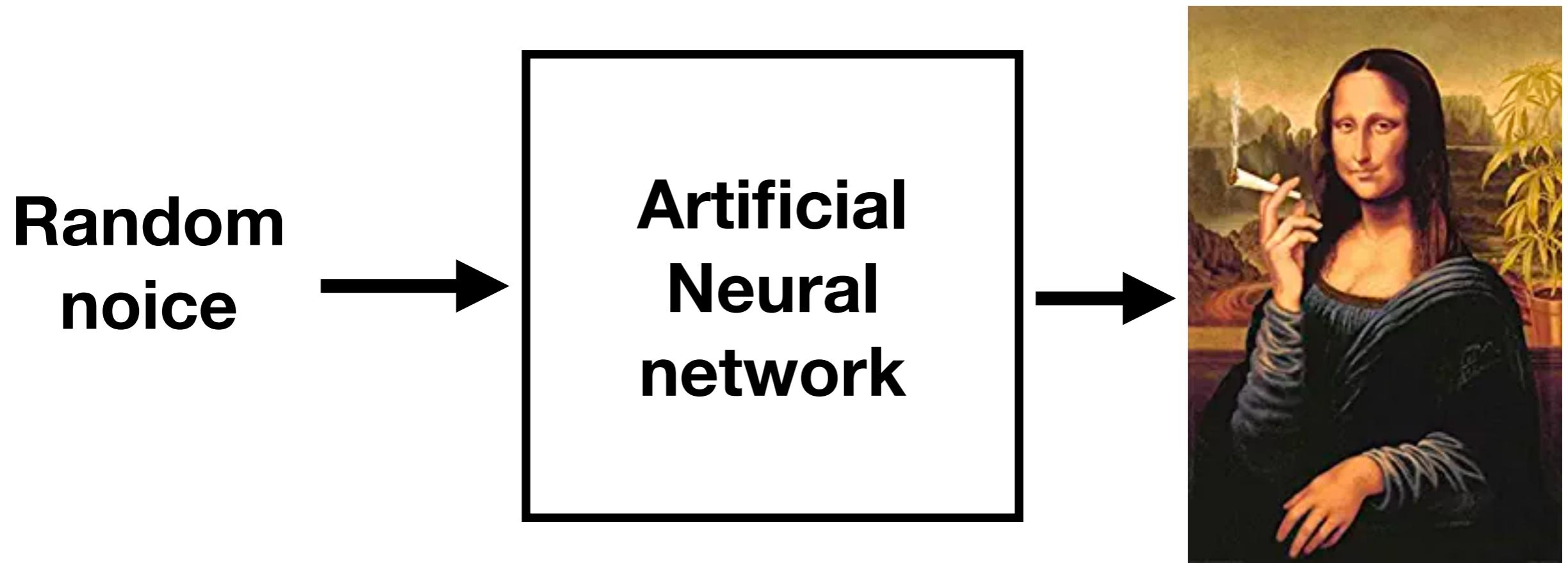
Network Architecture



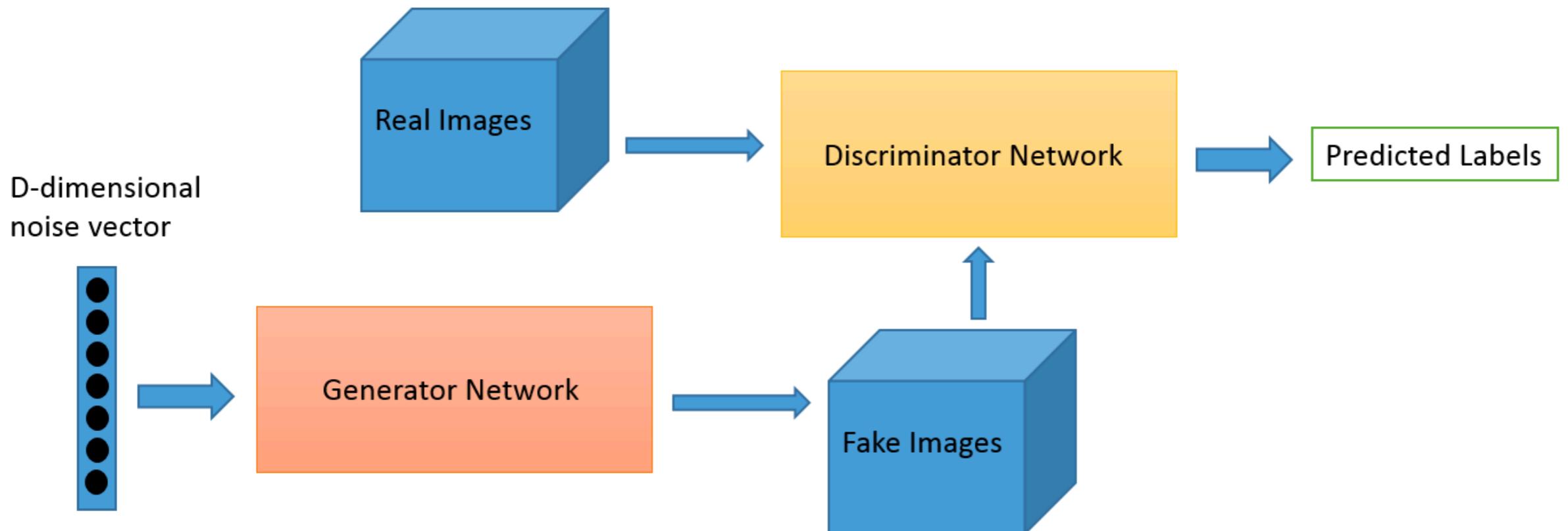
U-Net segmentation example

03-Segmentation.ipynb

Generative models with neural networks



Generative Adversarial Networks



Superresolution

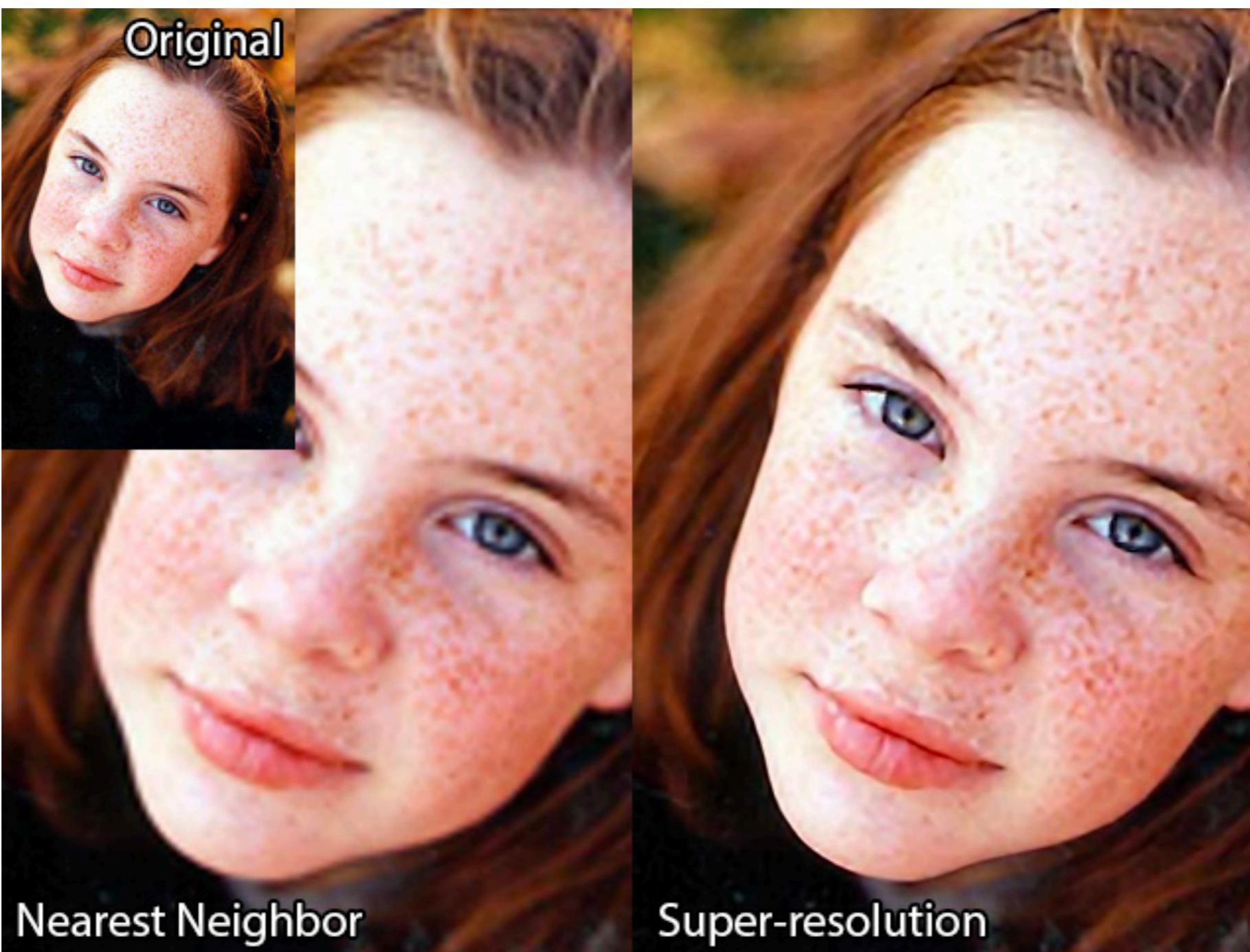
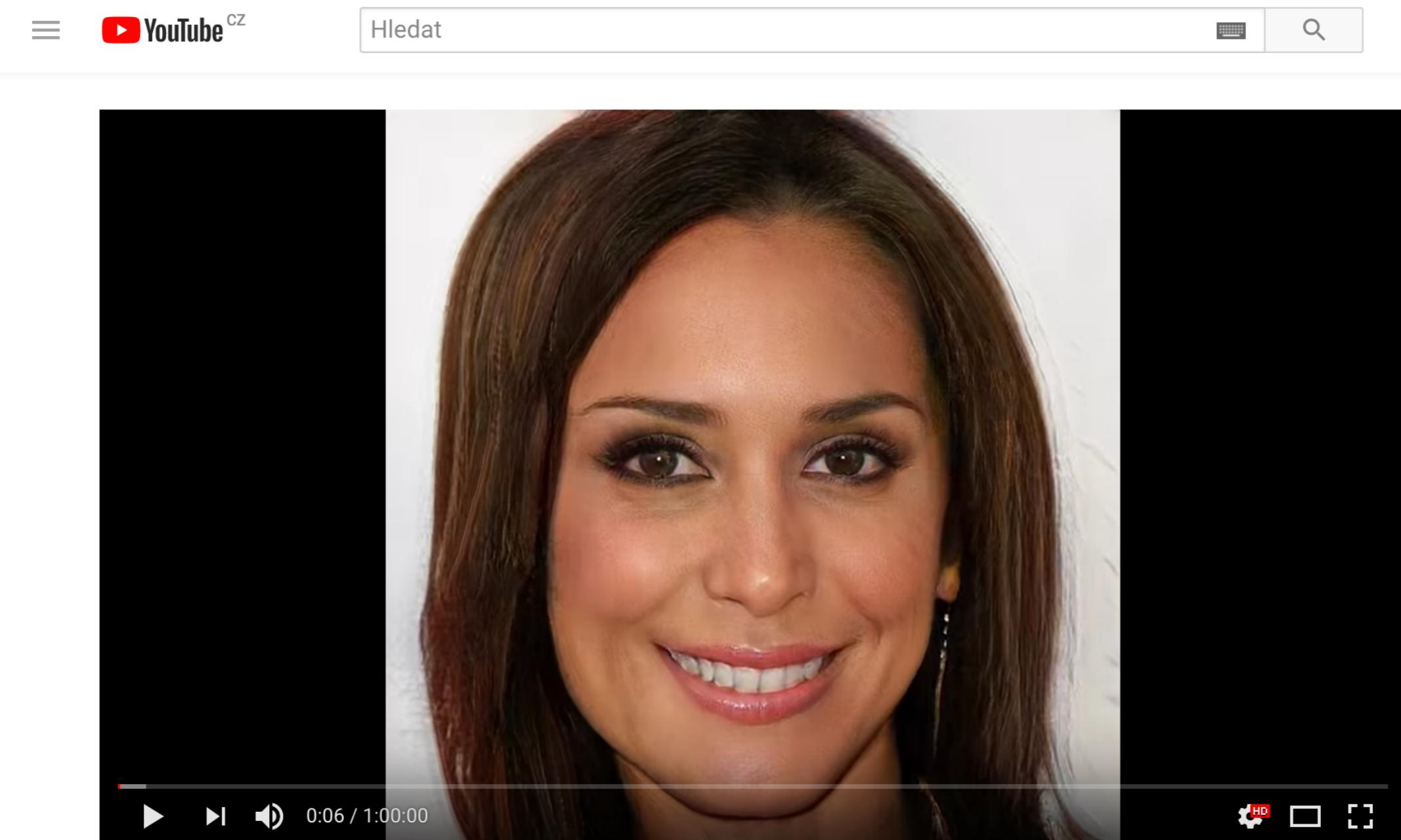


Image synthesis



One hour of imaginary celebrities

95 832 zhlédnutí

TO SE MI LÍBÍ NELÍBÍ SE SDÍLET ...

Which one is fake?



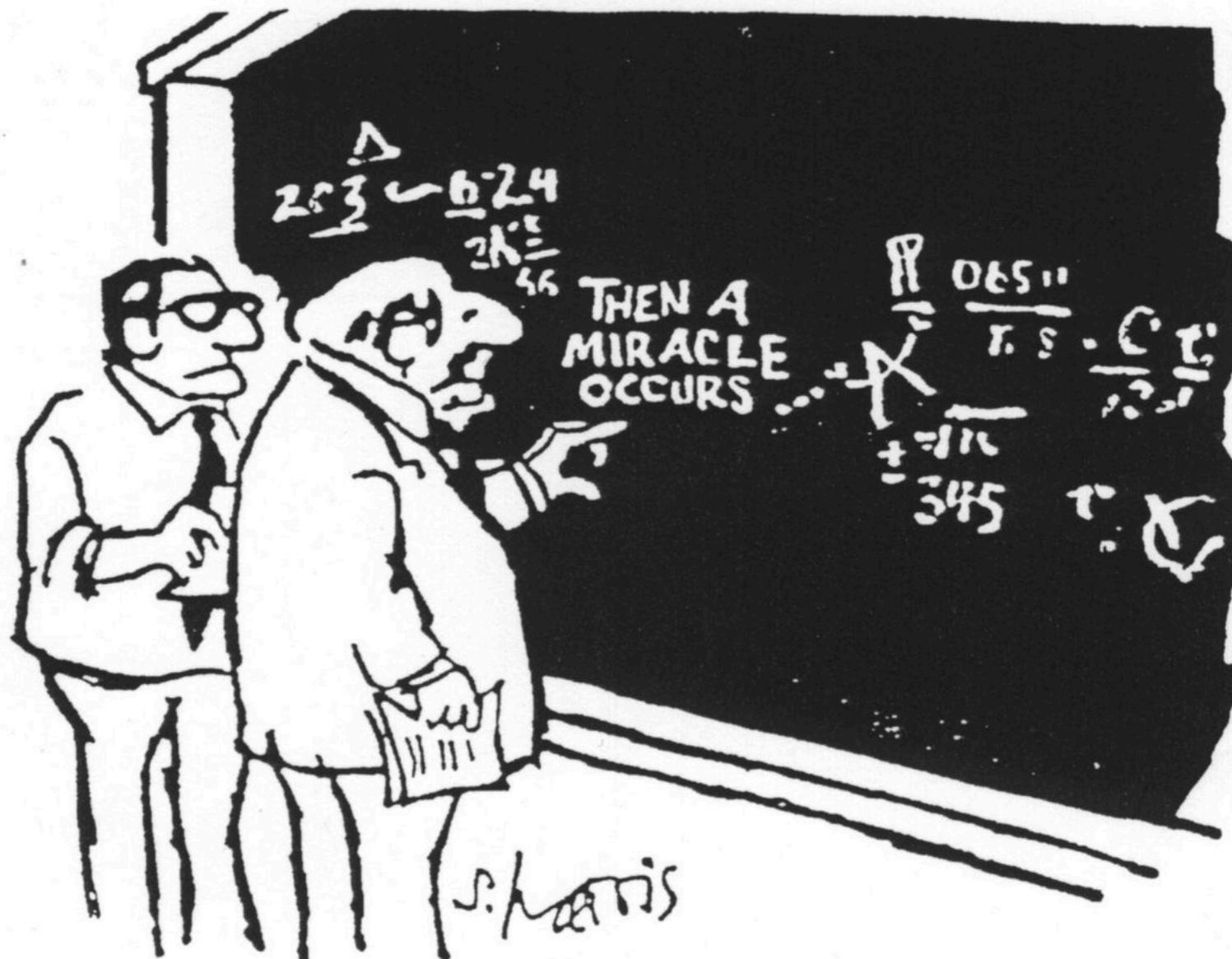
Generative Adversarial Networks

04_GANs.ipynb

Image manipulation



Neural network explainability



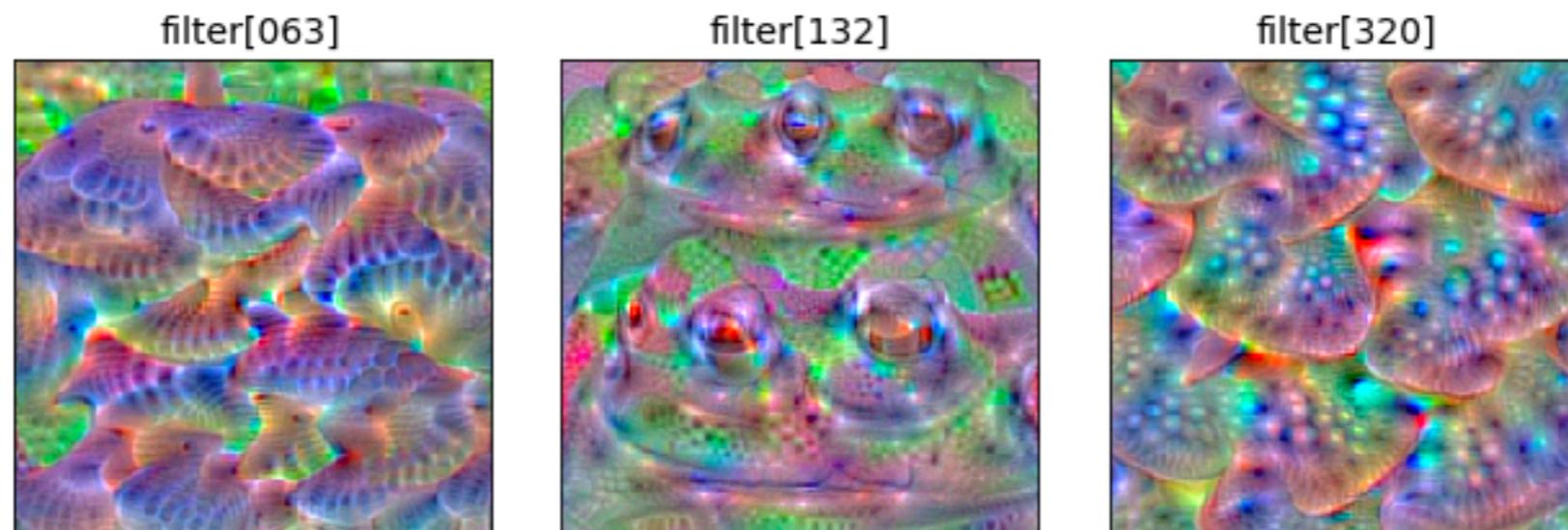
I think you should be a little
more specific, here in Step 2

Activation Maximization

Visualized output classification Layer



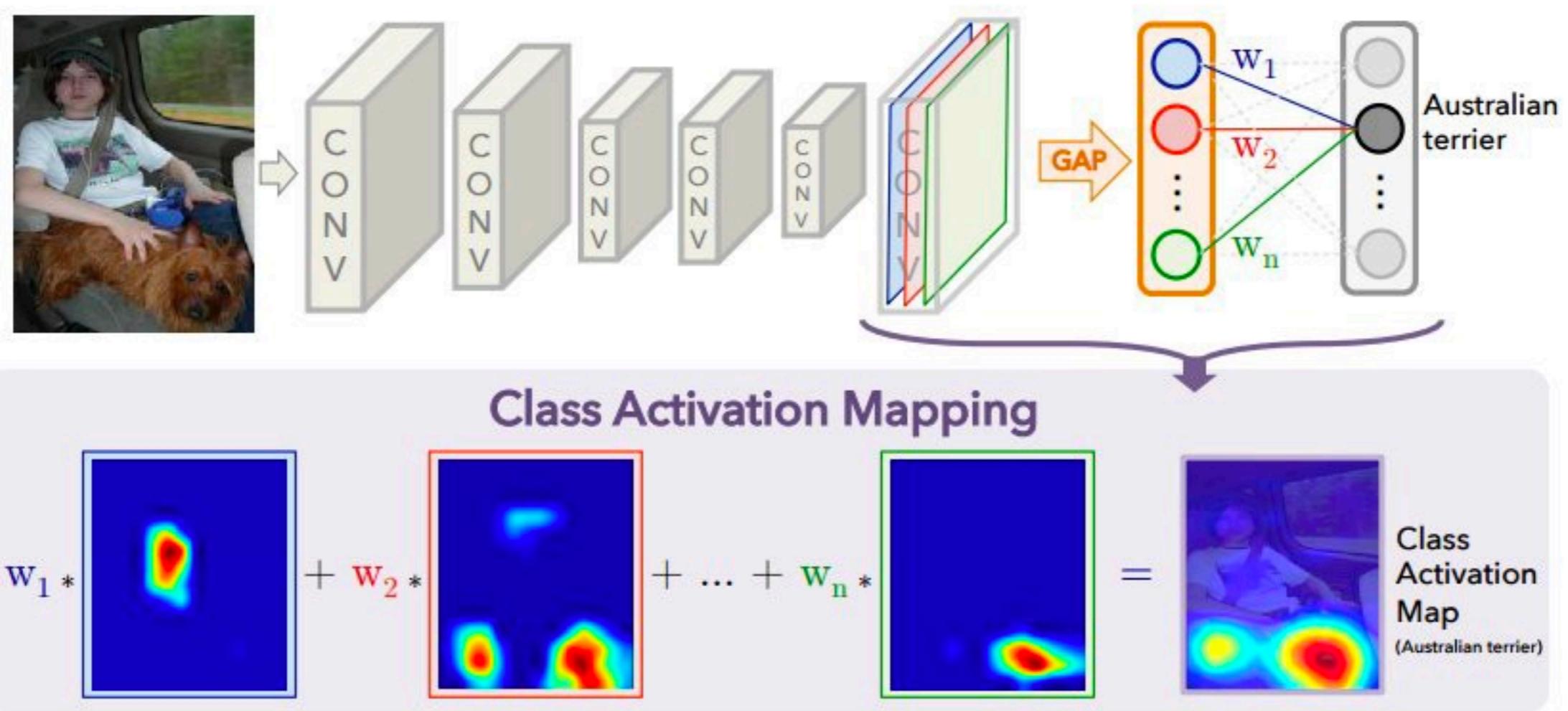
Visualized hidden convolutional layers



Grad-CAM heat maps



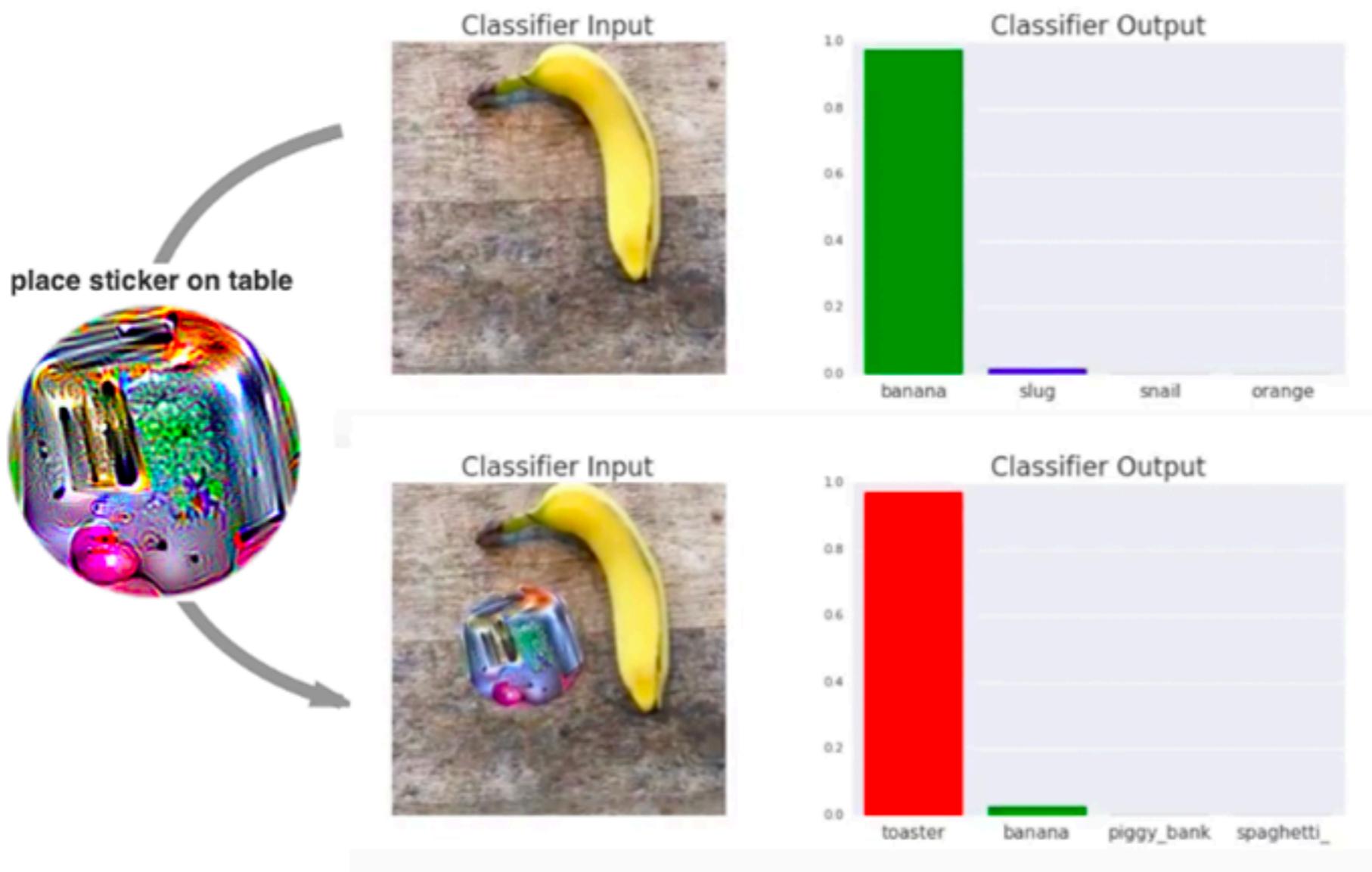
CAM heat maps



Convolutional Neural Networks Explainability in Keras

5-Explainability.ipynb

Adversarial Patch



Thank you for your attention

e-mail: jiri@mlguru.com

Web: www.mlguru.com

Twitter: @JiriMaterna

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

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