CNN Parameters and Training

Giacomo Boracchi,
DEIB, Politecnico di Milano
October, 11th, 2022

giacomo.boracchi@polimi.it
https://boracchi.faculty.polimi.it/

Parameters in a CNN

Giacomo Boracchi

Convolutions as MLP

Convolution is a linear operation!

Therefore, if you unroll the input image to a vector, **you can consider convolution weights as the weights of a Multilayer Perceptron Network!**

$$a(x, y, k) = \sum_{u,v,z} w_k(u, v, z)\, I(x - u, y - v, z) + b_k$$

inputs

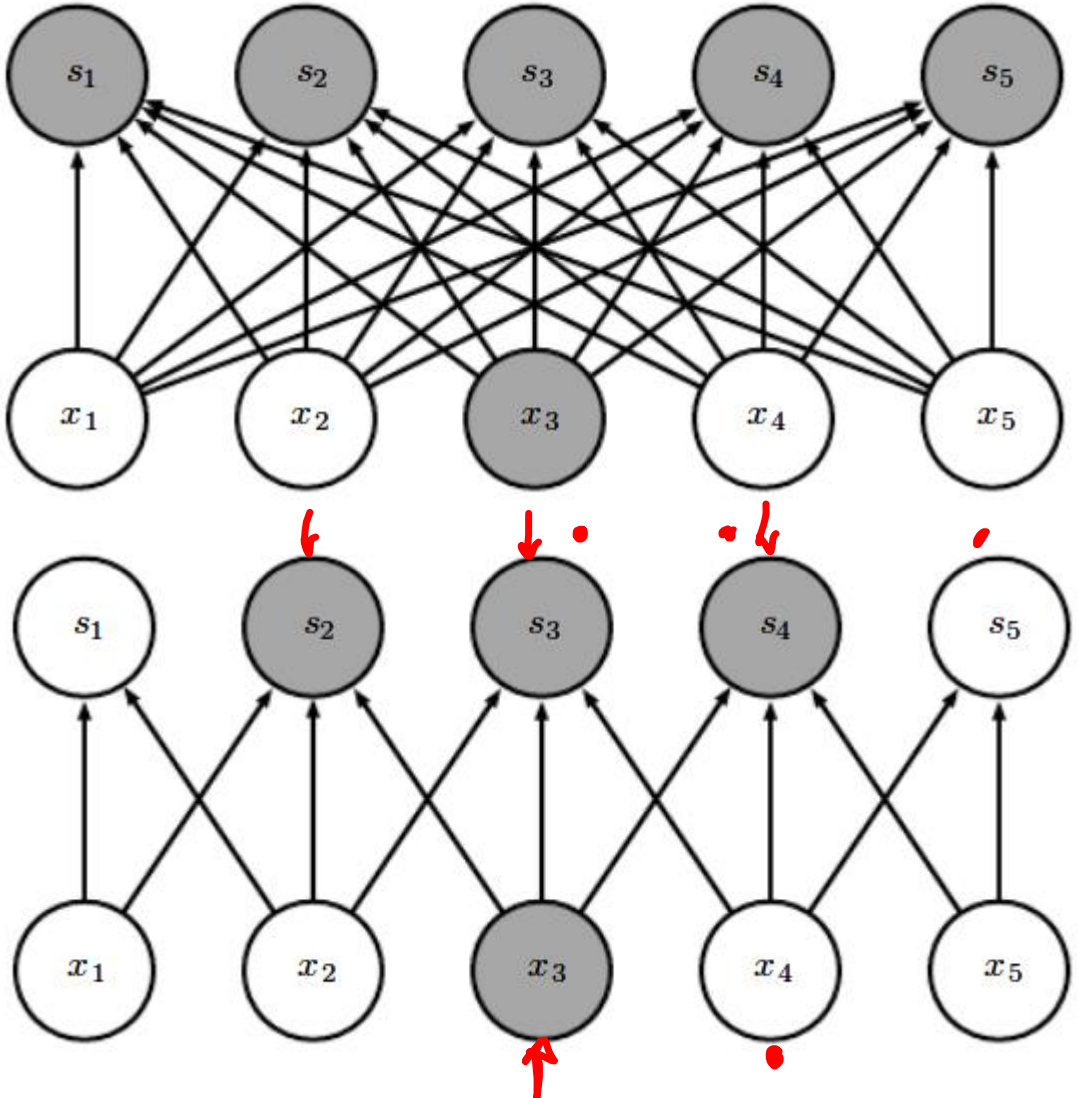
parameters

What are the differences between MLP and CNNs then?

CNNs has Sparse Connectivity

MLP, Fully connected

3x1 convolutional



Output neurons

Input neurons

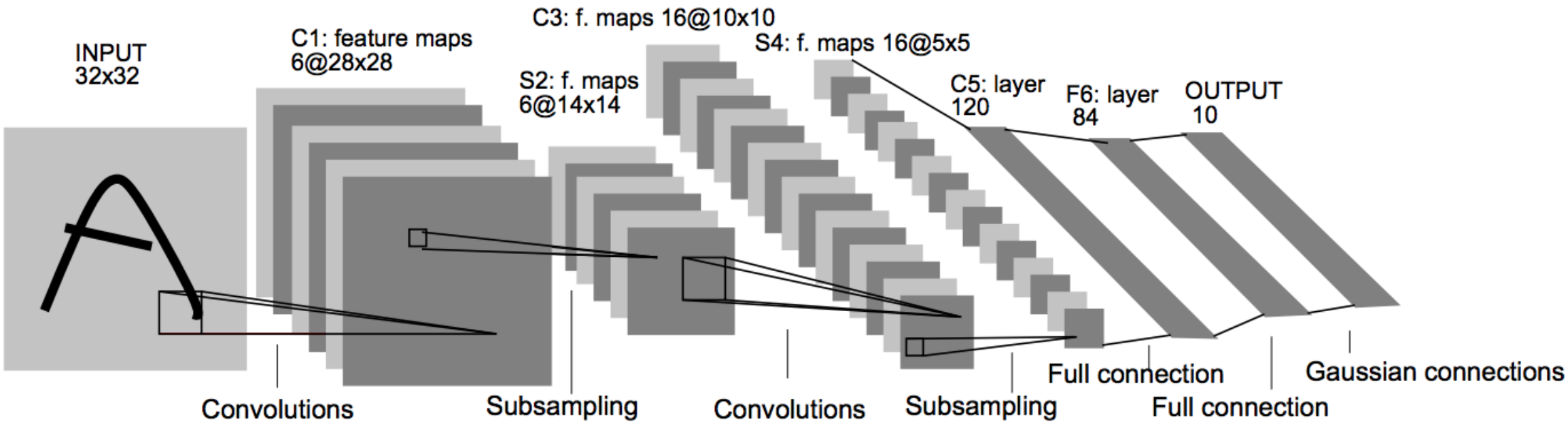
Output neurons (unrolled activations at the first layer)

Input neurons (unrolled image)

Weight Sharing / Spatial Invariance

In a CNN, all the neurons in the same slice of a feature map use the same weights and bias: this reduces the nr. of parameters in the CNN.

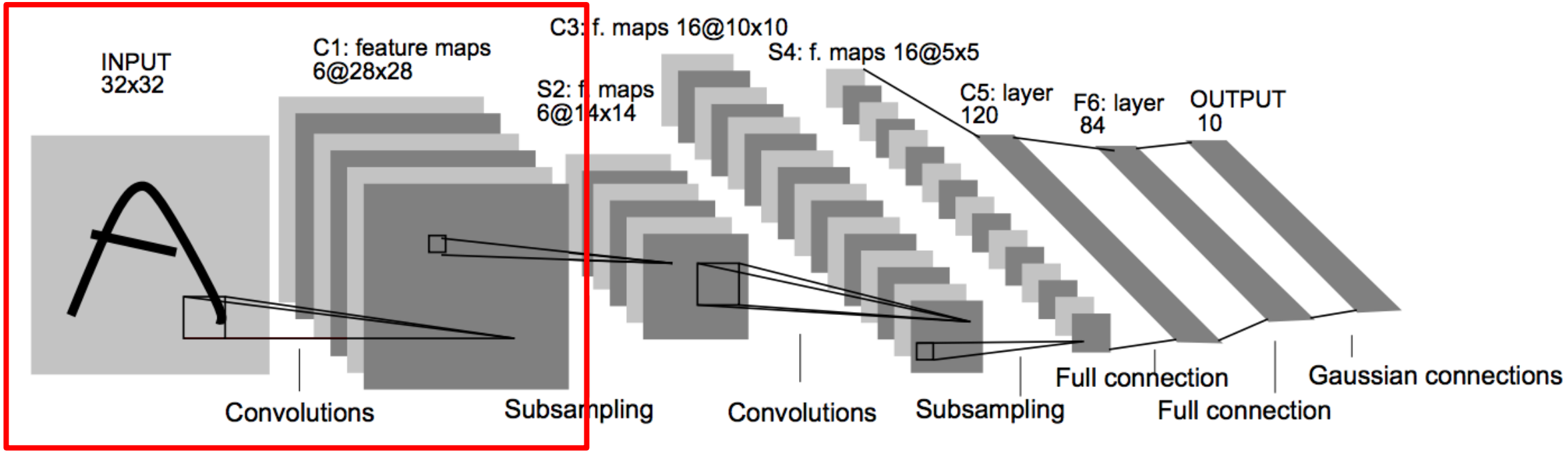
Underlying assumption: **if one feature is useful** to compute **at some spatial position** (x, y), **then it should also be useful** to compute at a **different position** (x_2, y_2)



Weight Sharing / Spatial Invariance

If the first layer were a MLP:

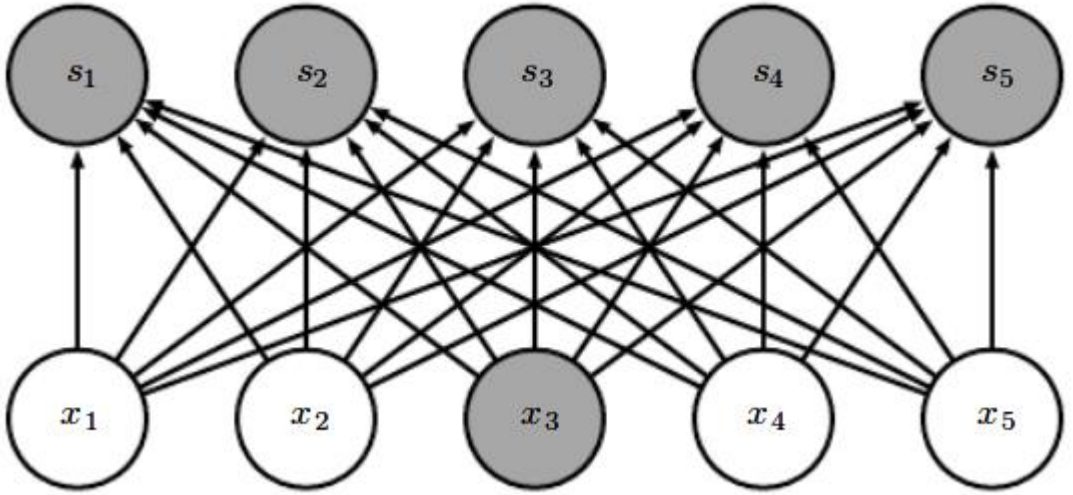
- MLP layer: this should have had 28 x 28 x 6 neurons in the output
- MLP layer with sparse connectivity: only 5x5 nonzero weights each neuron
- MLP layer: 28 x 28 x 6 x 25 weights + 28 x 28 x 6 biases (122 304)
- **Conv layer: 25 weights + 6 biases shared among neurons of the same layer**



Parameter sharing

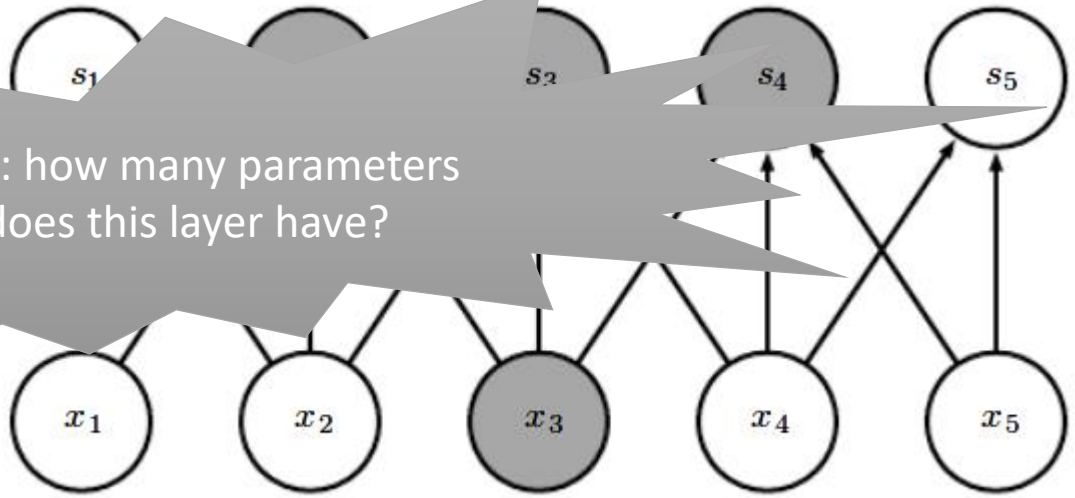
Fully connected

5x5 = 25 weights
(+ 5 bias)



3x1 convol

3 weights!
(+ 1 bias)

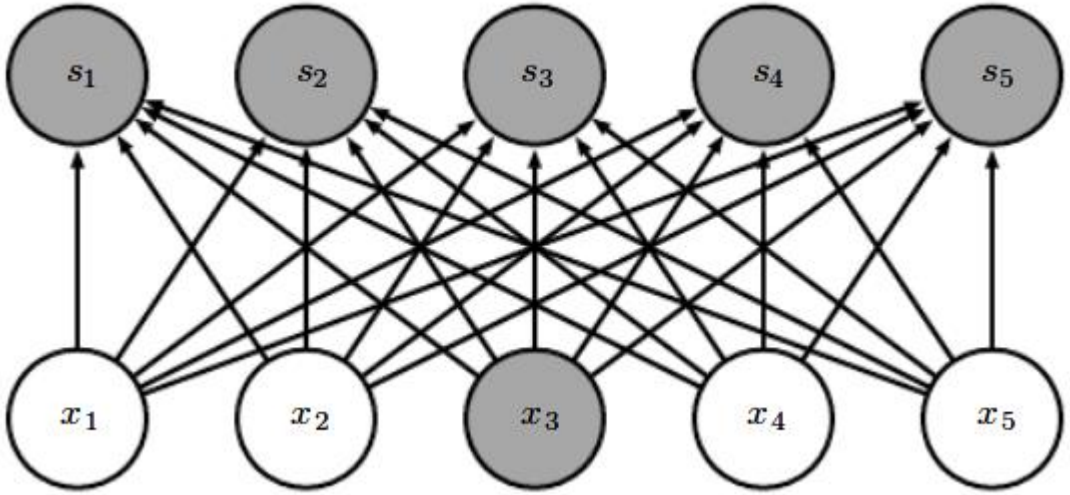


Quiz: how many parameters does this layer have?

Parameter sharing

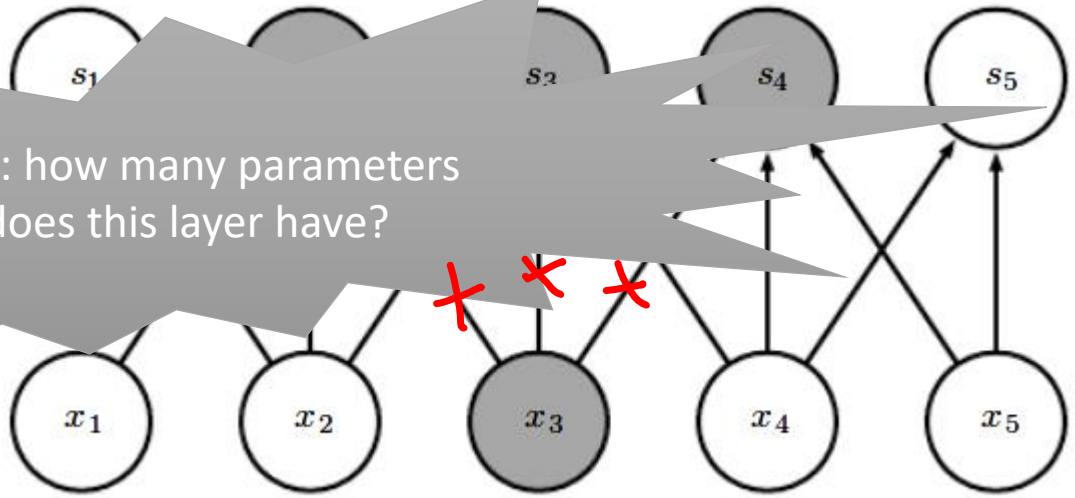
Fully connected

5x5 = 25 weights
(+ 5 bias)



3x1 convo

3 weights!
(+ 1 bias)



Quiz: how many parameters does this layer have?

+ 5 biases

5 outputs

3 inputs

To Summarize

Any CONV layer can be implemented by a FC layer performing exactly the same computations.

The weight matrix W of the FC layer would be

- a large matrix (#rows equal to the number of output neurons, #cols equal to the nr of input neurons).
- That is mostly zero except for at certain blocks where the local connectivity takes place.
- The weights in many of the blocks are equal due to parameter sharing.

… and we will see that the converse interpretation (FC as conv) is also viable and very useful!

The Receptive Field
A very important aspect in CNNs

Giacomo Boracchi

The Receptive Field

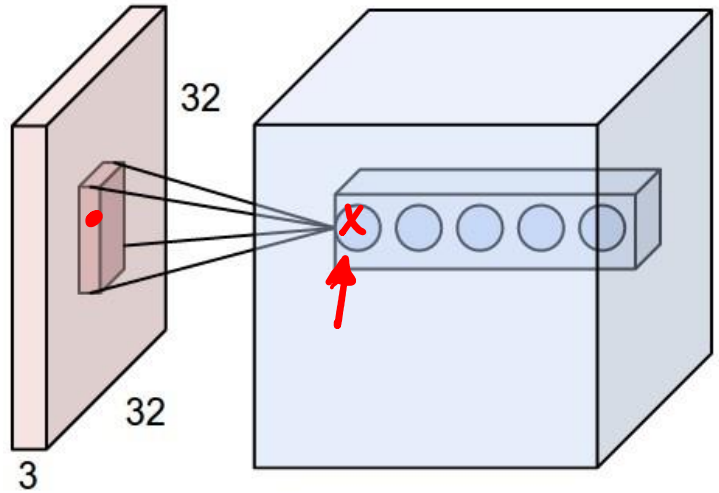
One of the basic concepts in deep CNNs.

Due to sparse connectivity, unlike in FC networks where the value of each output depends on the entire input, **in CNN each output only depends on a specific region in the input.**

This region in the input is the receptive field for that output

The deeper you go, the wider the receptive field is: maxpooling, convolutions and stride > 1 increase the receptive field

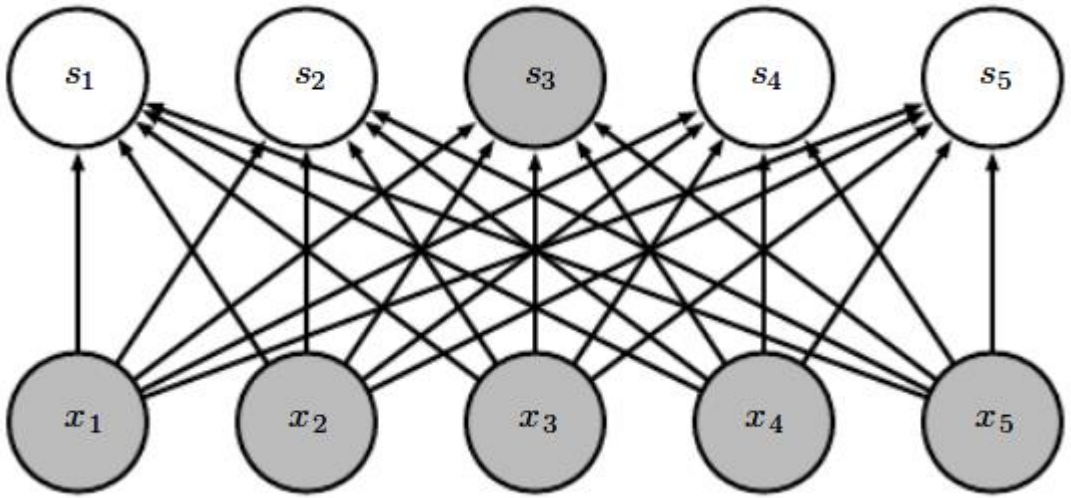
Usually, the receptive field refers to the final **output unit** of the network in relation to the network input, but the same definition holds for intermediate volumes



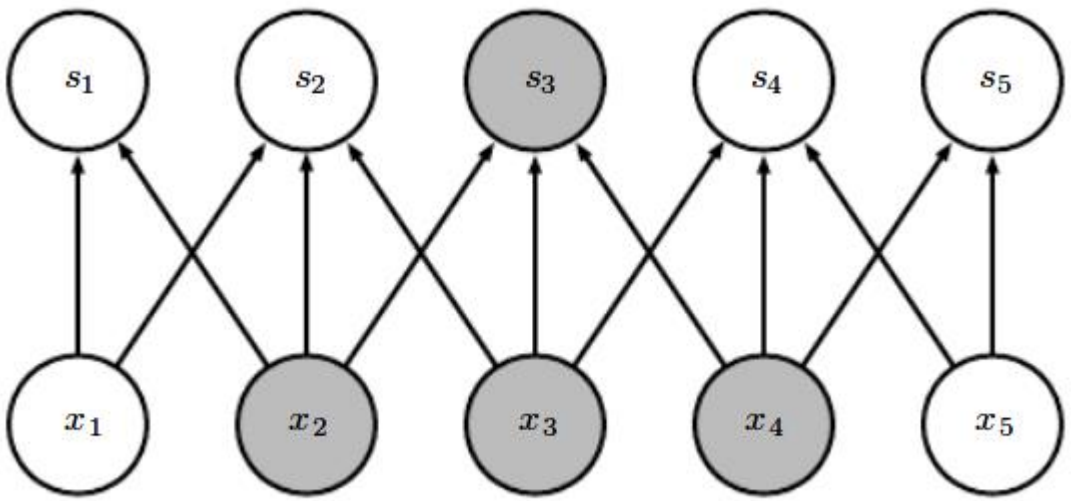
By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45659236

Receptive fields

MPL, Fully connected



3x1 convolutional

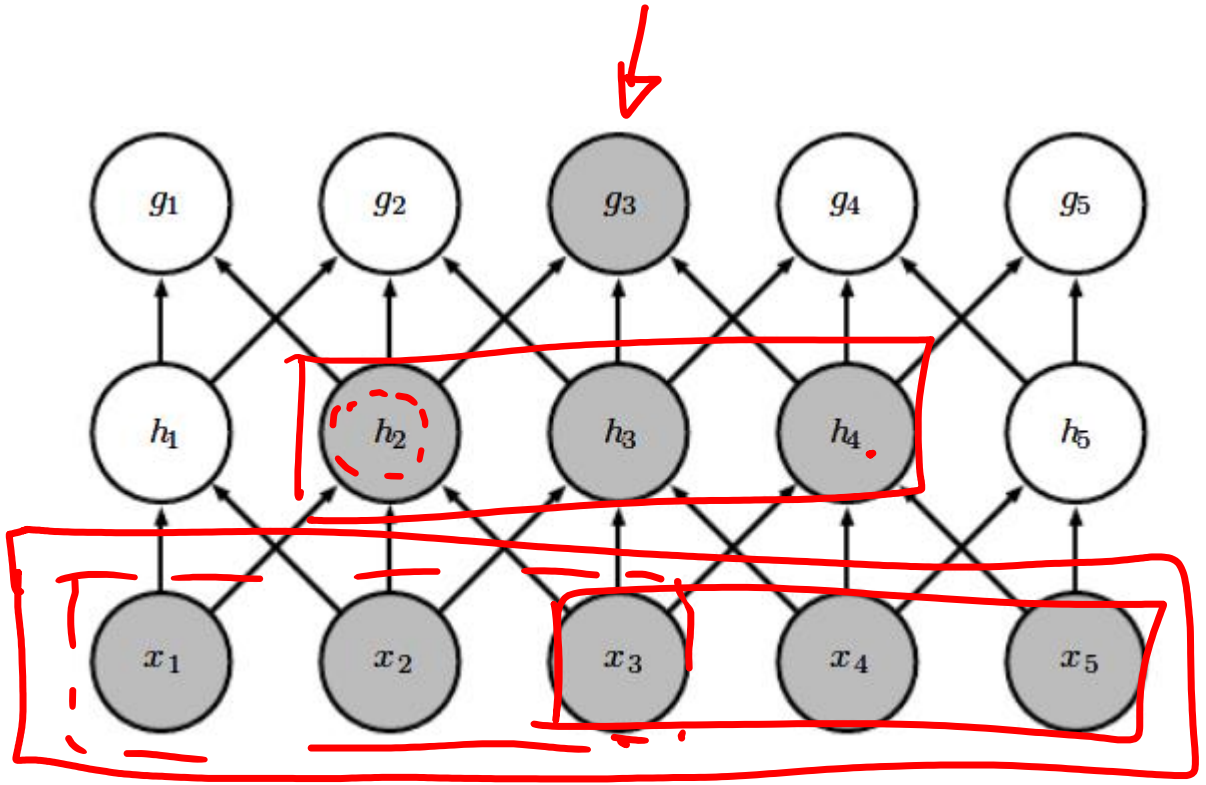


Receptive fields

Deeper neurons depend on wider patches of the input (convolution is enough to increase receptive field, no need of maxpooling)

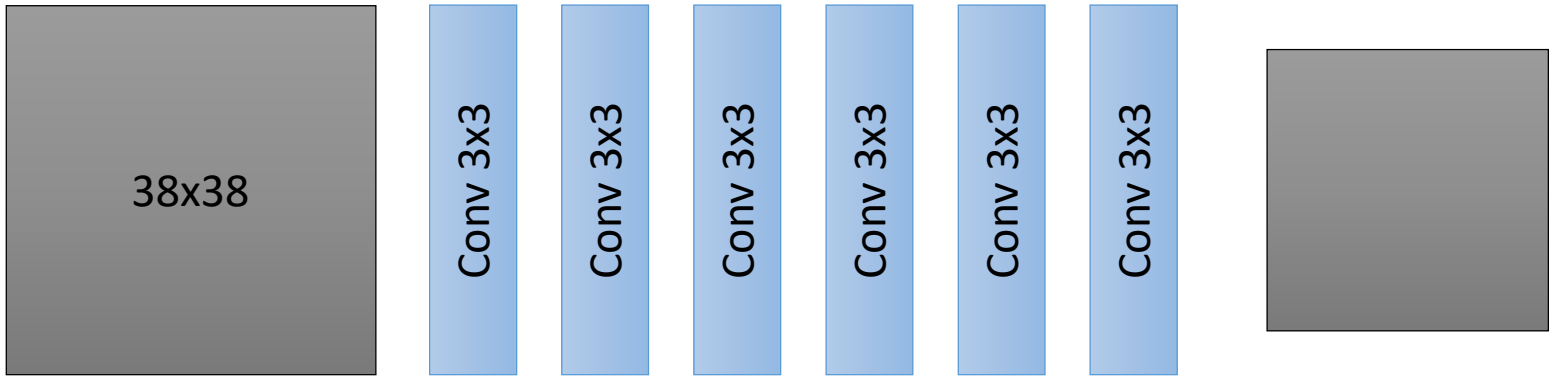
3x1 convolutional

3x1 convolutional



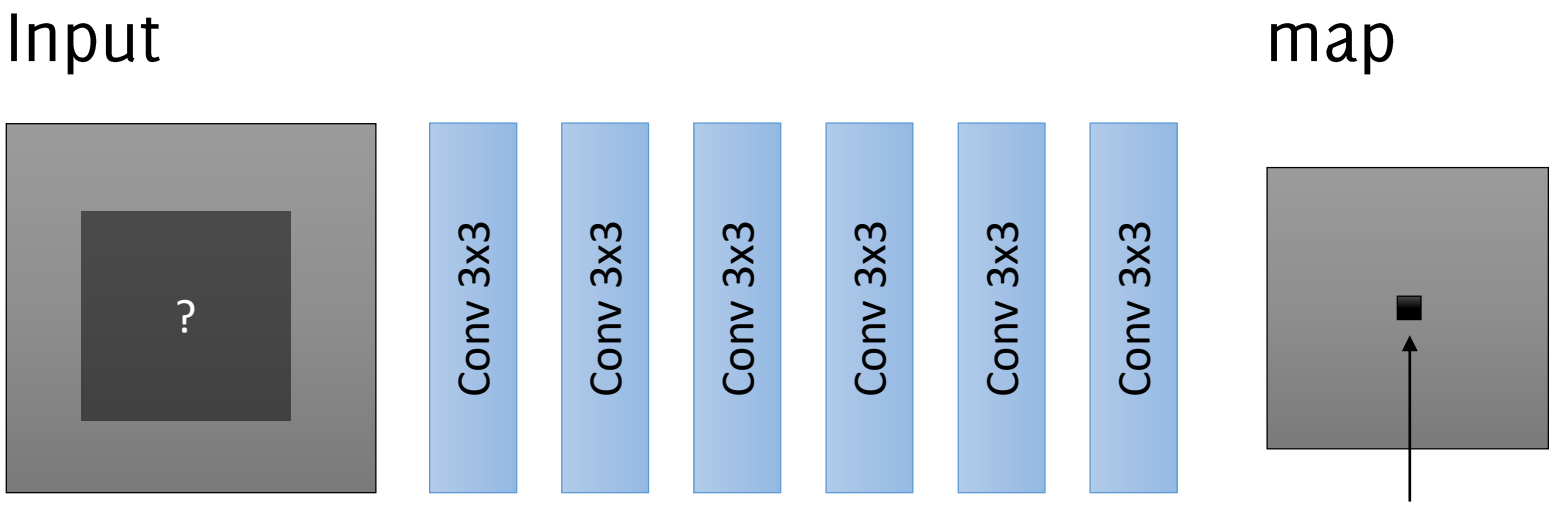
Exercise

Input:



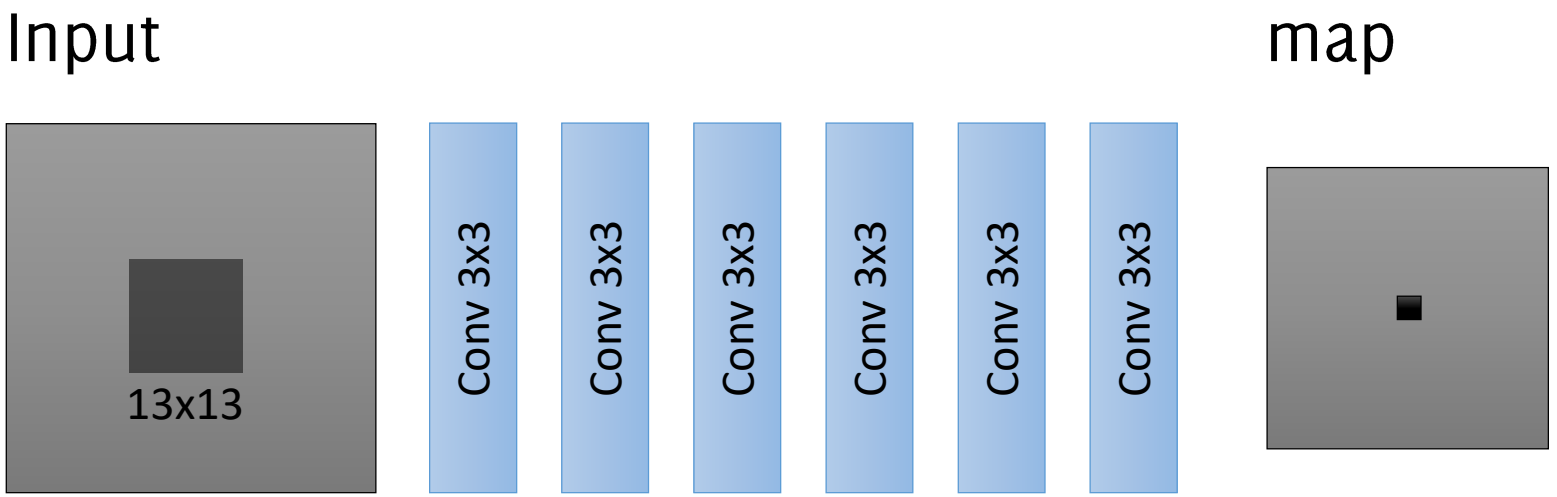
map

Receptive fields



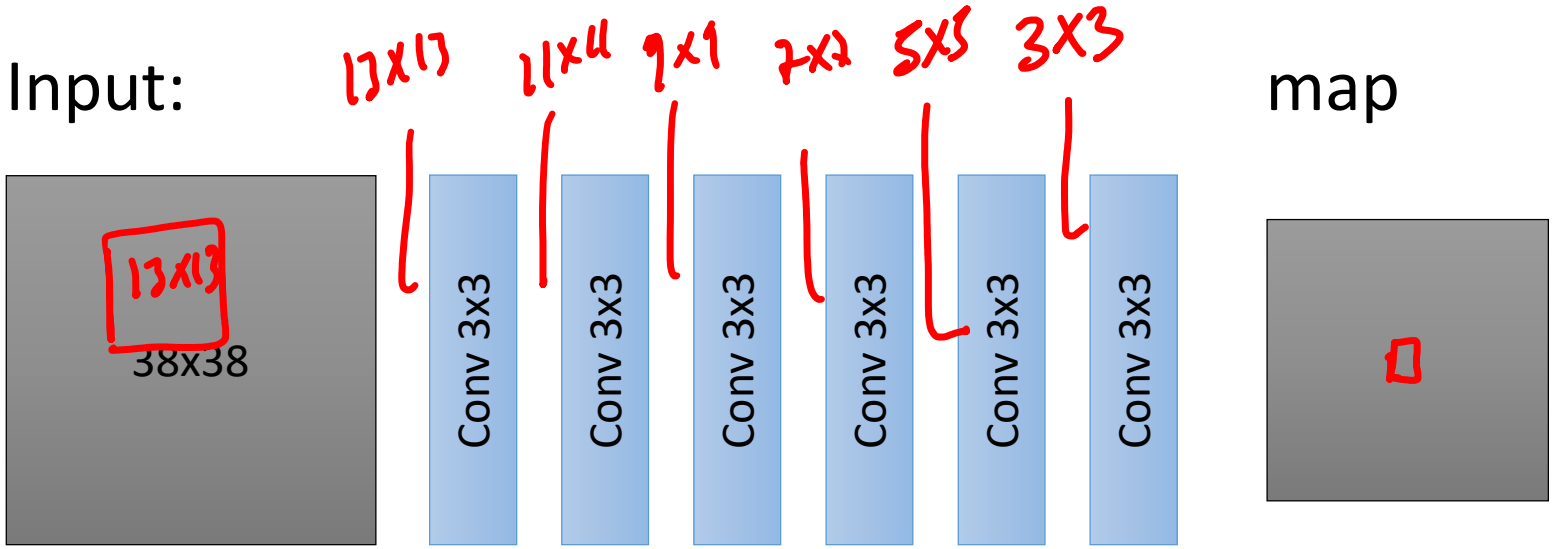
How large is the receptive field of the black neuron?

Receptive fields

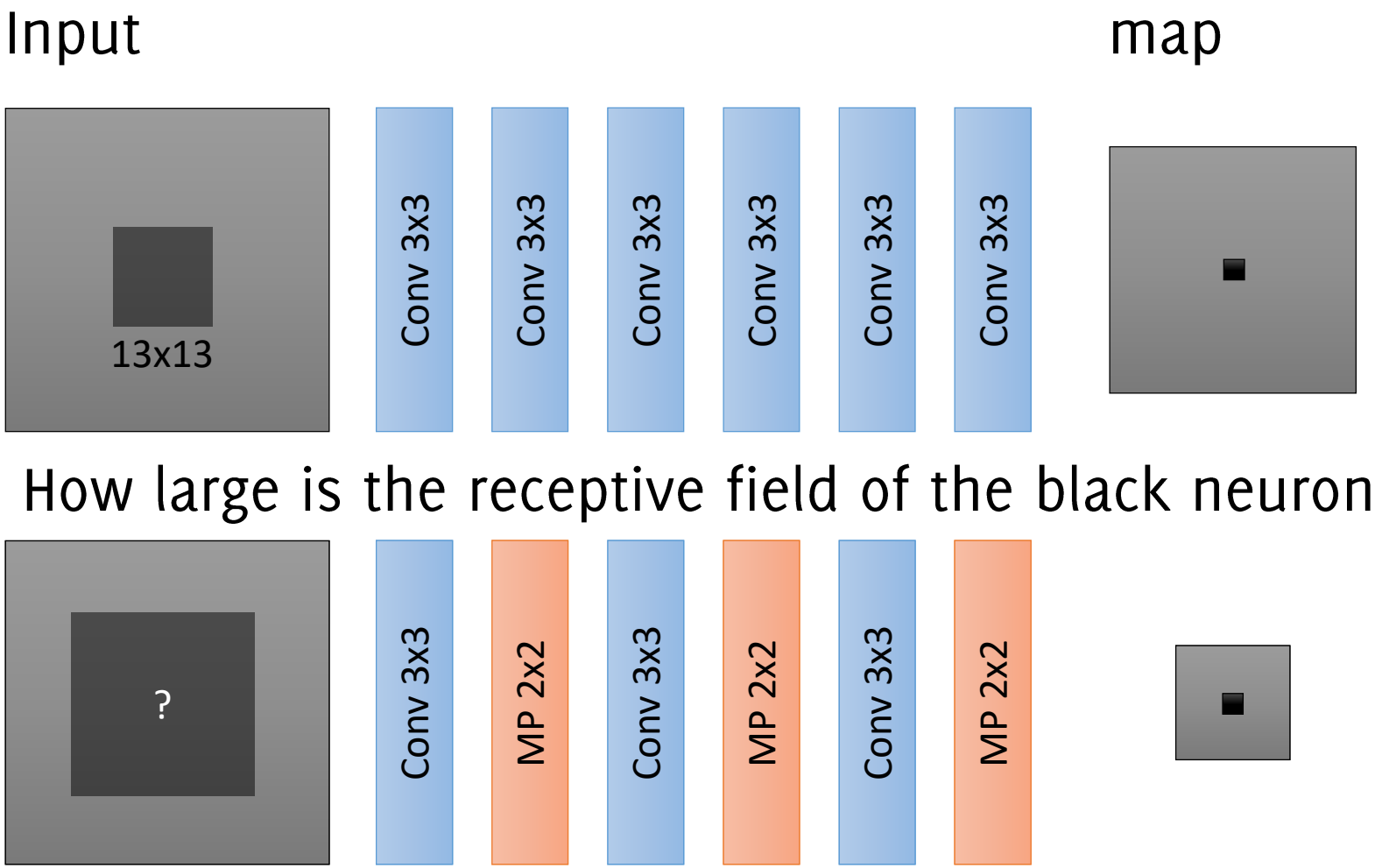


How large is the receptive field of the black neuron?

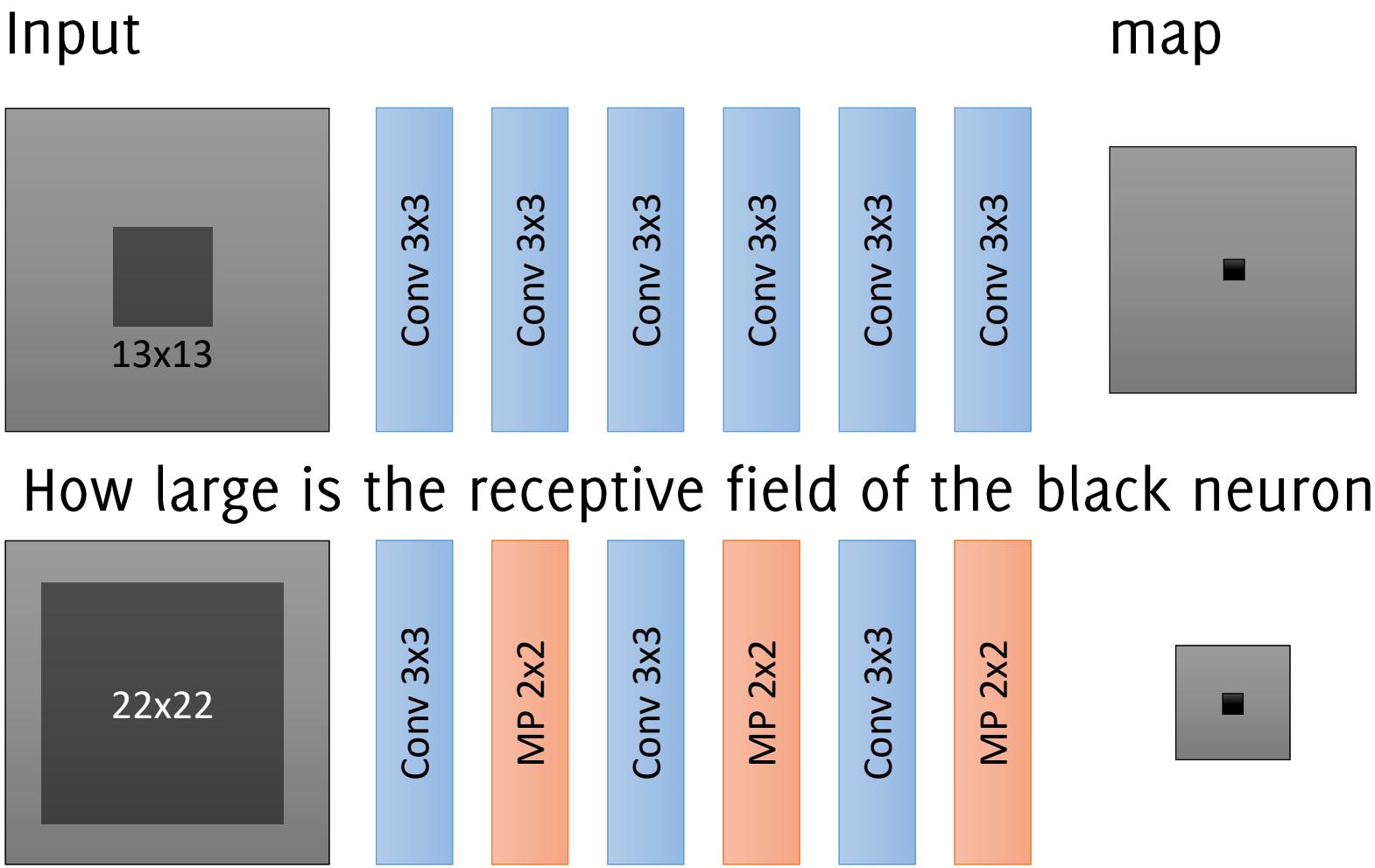
Exercise


Input:
13x13
38x38
Conv 3x3
Conv 3x3
Conv 3x3
Conv 3x3
Conv 3x3
Conv 3x3
map

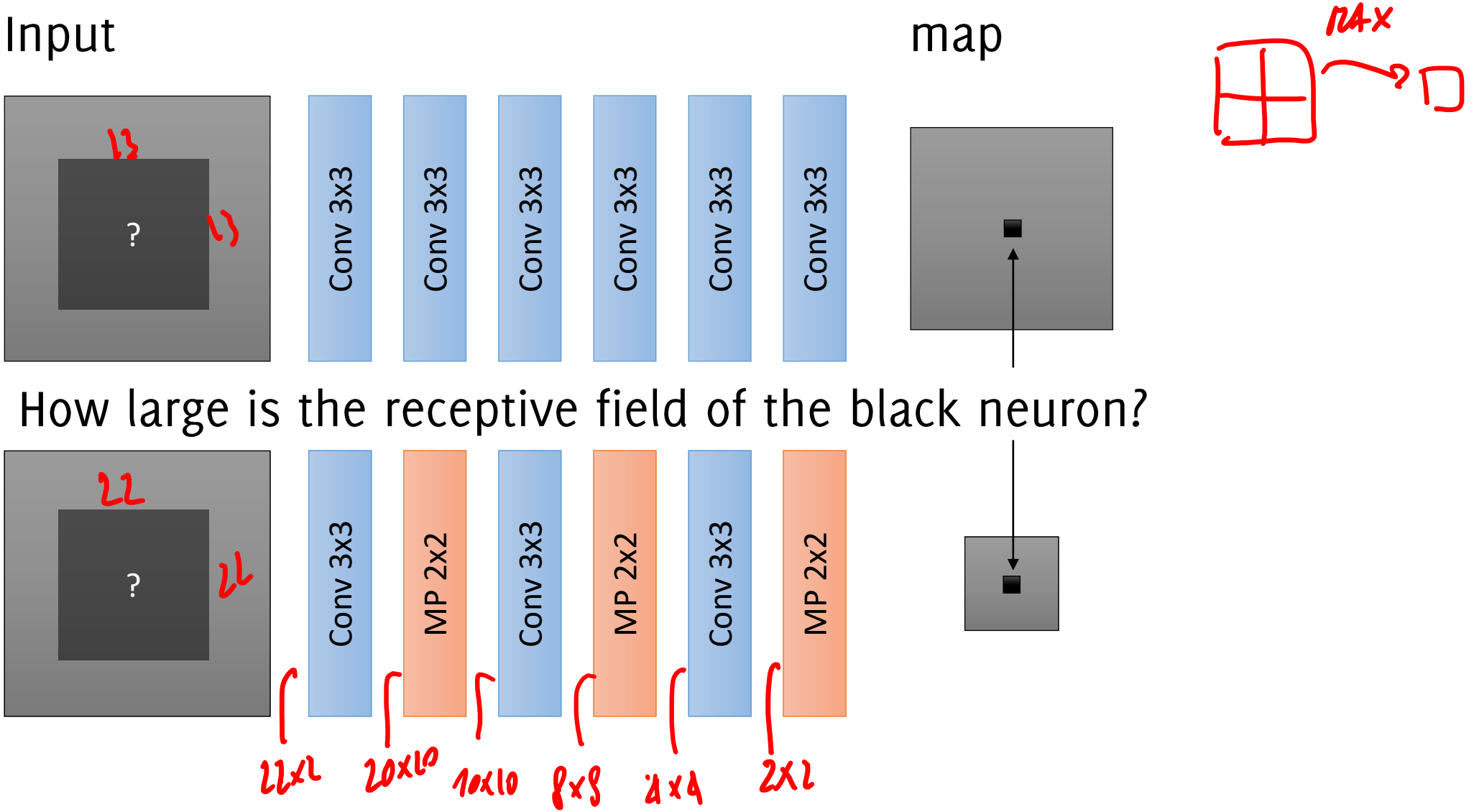
Receptive fields



Receptive fields



Receptive fields



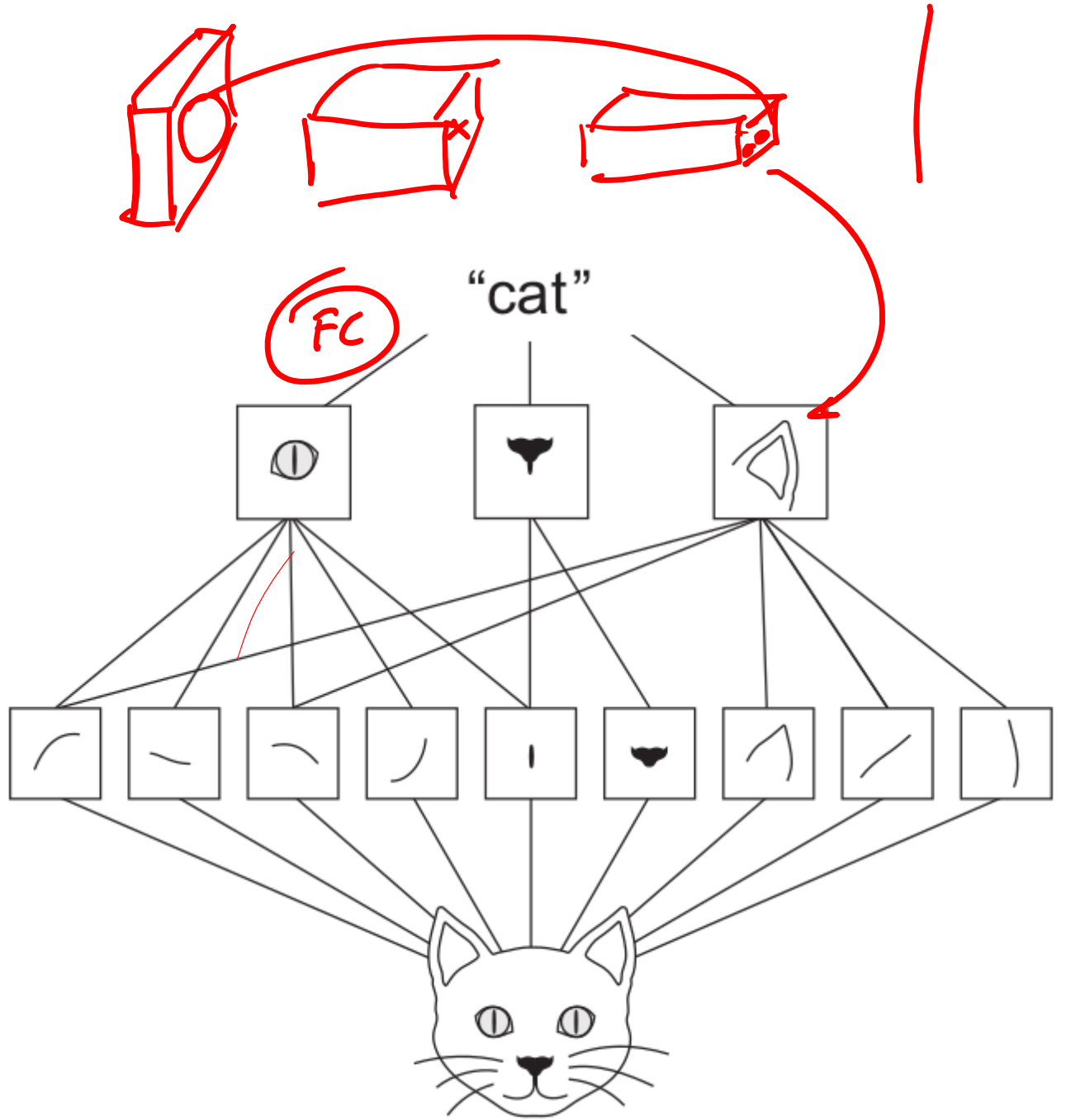
As we move deeper…

As we move to deeper layers:

- spatial resolution is reduced
- the number of maps increases

We search for **higher-level patterns**, and **don't care too much about their exact location.**

There are more high-level patterns than low-level details!



CNN Training

Giacomo Boracchi

Training a CNN

- **Each CNN can be seen as a MLP**, with sparse and shared connectivities)
- CNN can be in principle **trained by gradient descent** to minimize a loss function over a batch (e.g. binary cross-entropy, RMSE, Hinge loss..)
- Gradient can be computed by **backpropagation** (chain rule) as long as we can derive each layer of the CNN
- **Weight sharing needs to be taken into account** (fewer parameters to be used in the derivatives) while computing derivatives
- There are just a few details missing…

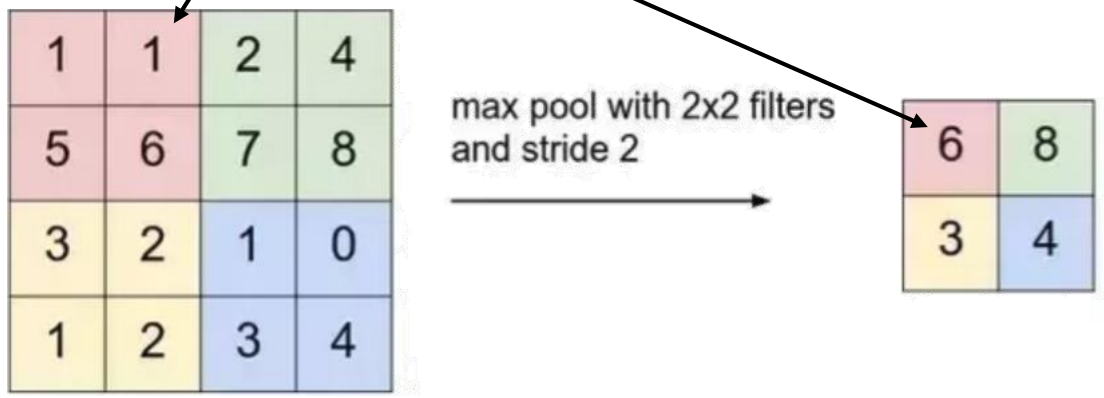
Detail: backprop with max pooling

The gradient is only routed through the input pixel that contributes to the output value; e.g.:

Gradient of • with respect to • = 0

The derivative is:

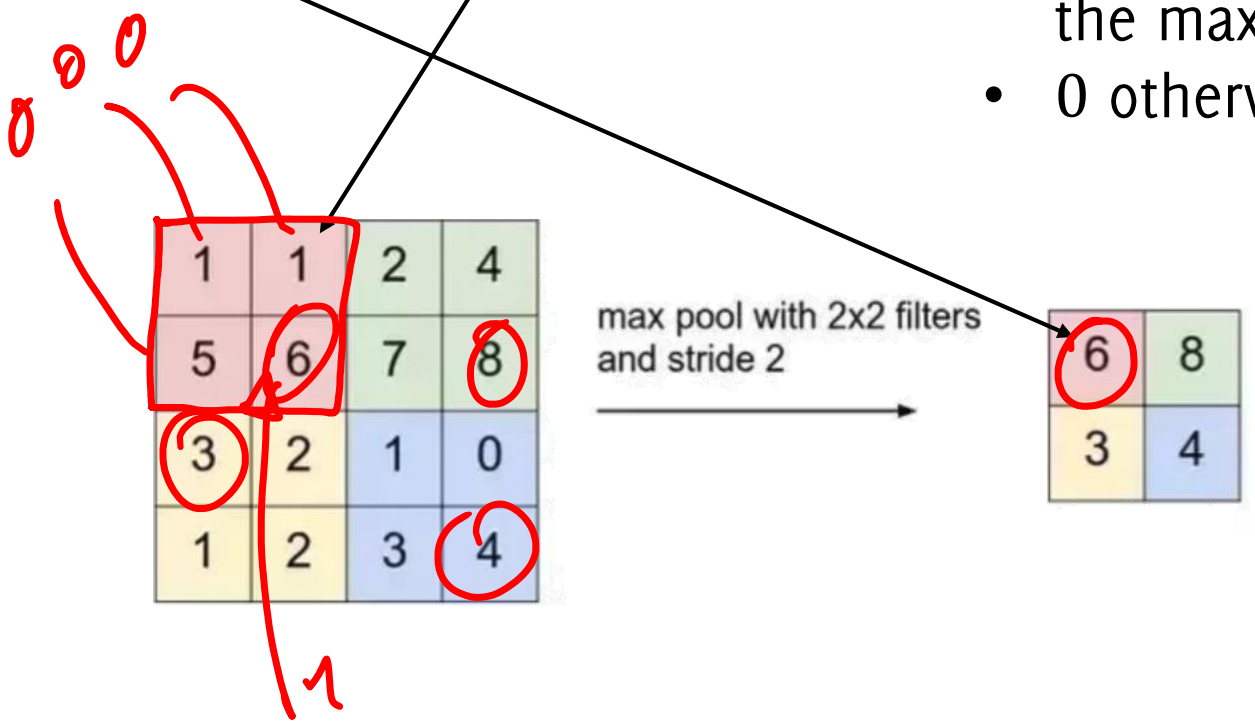
- 1 at the location corresponding the maximum
- 0 otherwise



Detail: backprop with max pooling

The gradient is only routed through the input pixel that contributes to the output value; e.g.:

Gradient of • with respect to • = 0

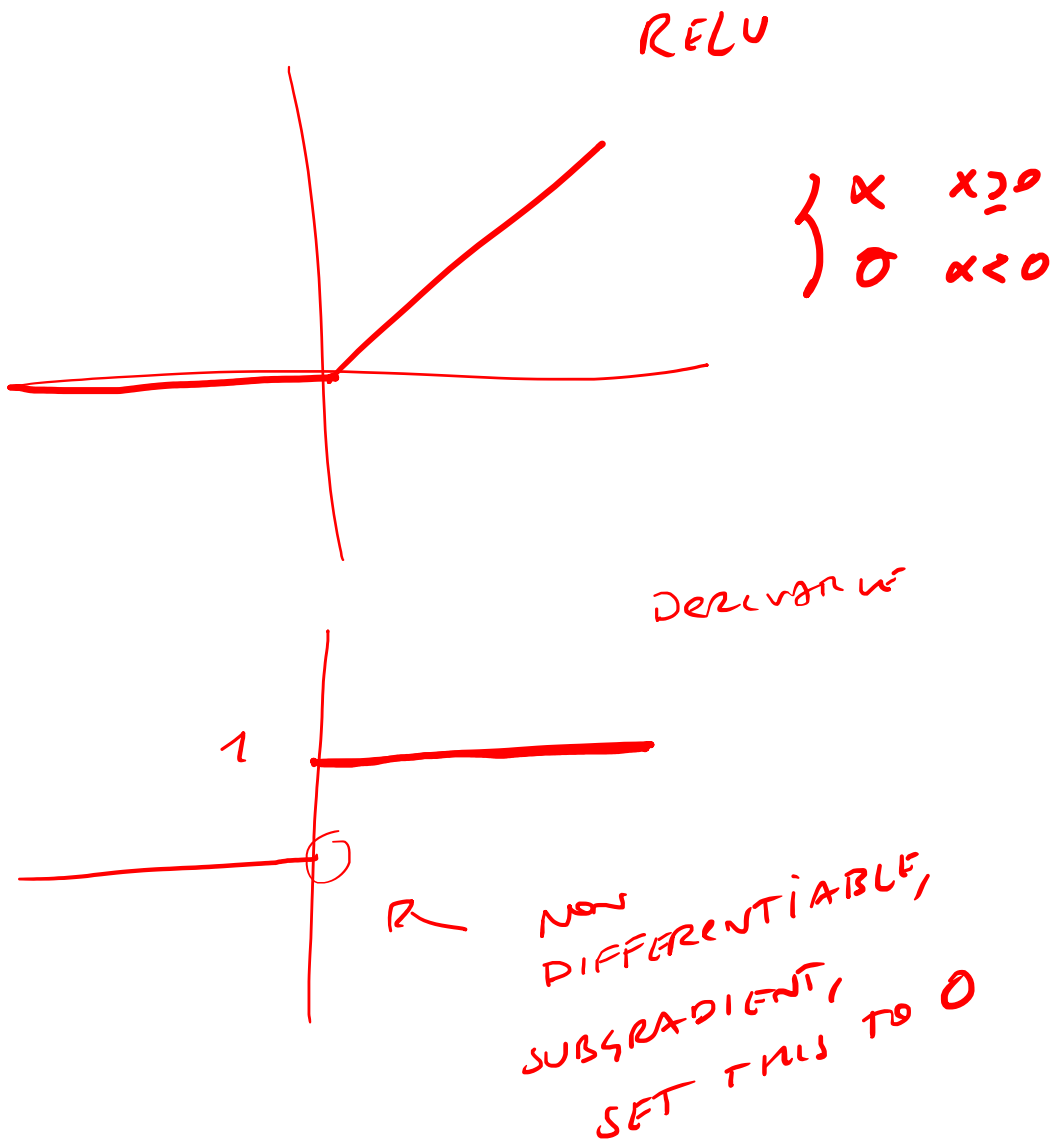


The derivative is:

- 1 at the location corresponding the maximum
- 0 otherwise

Detail: derivative of ReLU

The ReLU derivative is straightforward

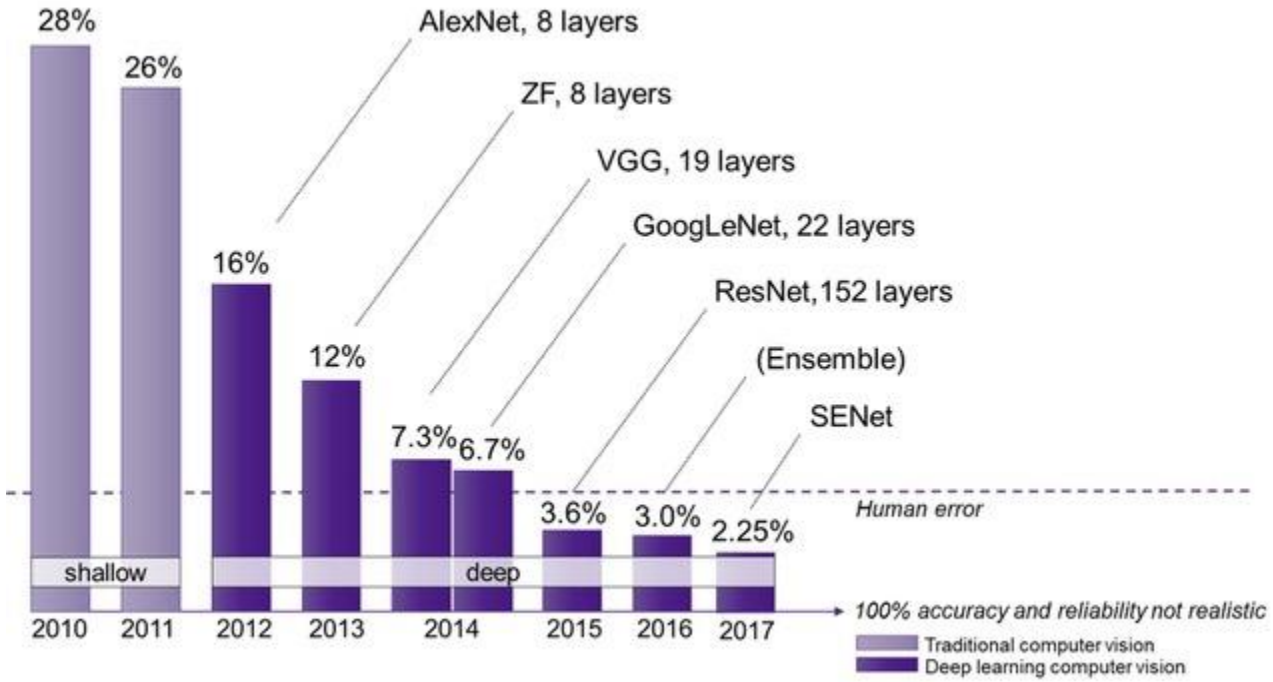


A Breaktrough in Image Classification

Giacomo Boracchi

The impact of Deep Learning in Visual Recognition

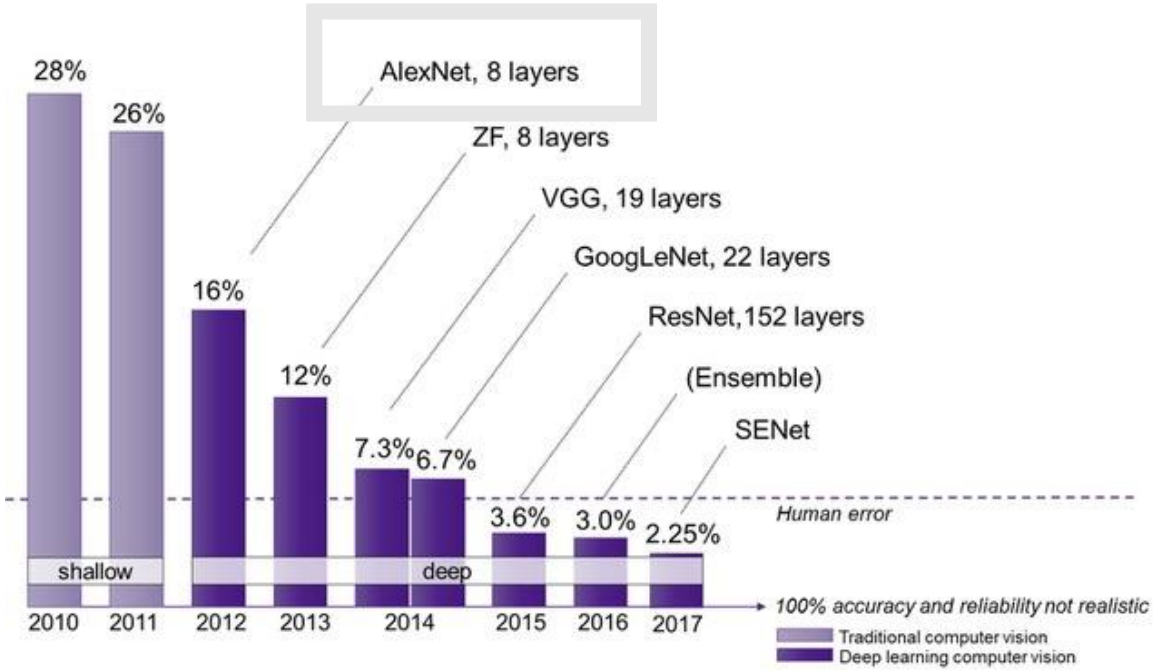
Classification accuracy on **ILSVRC**

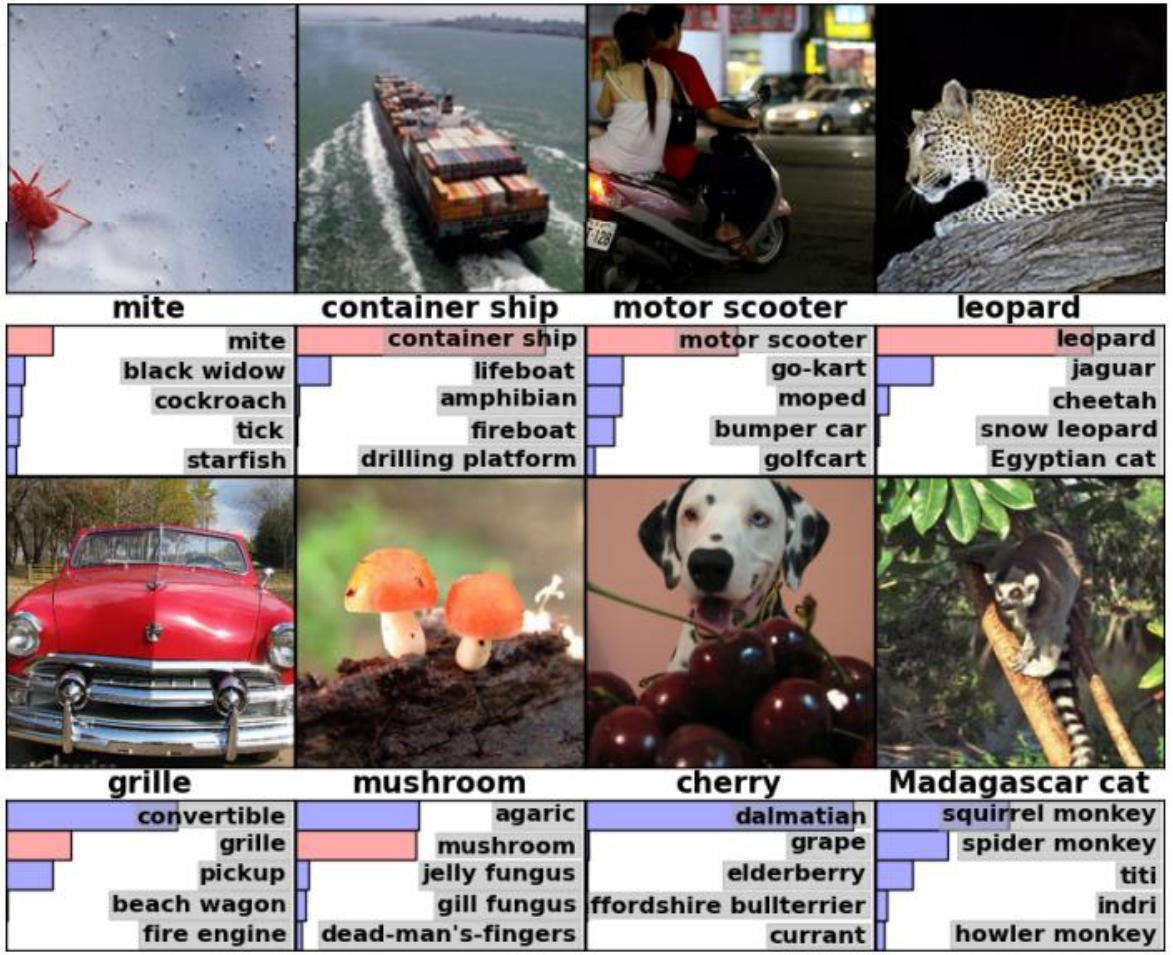


Many layers!

ILSVRC: ImageNet Large Scale Visual Recognition Challenge

AlexNet / Imagenet Images

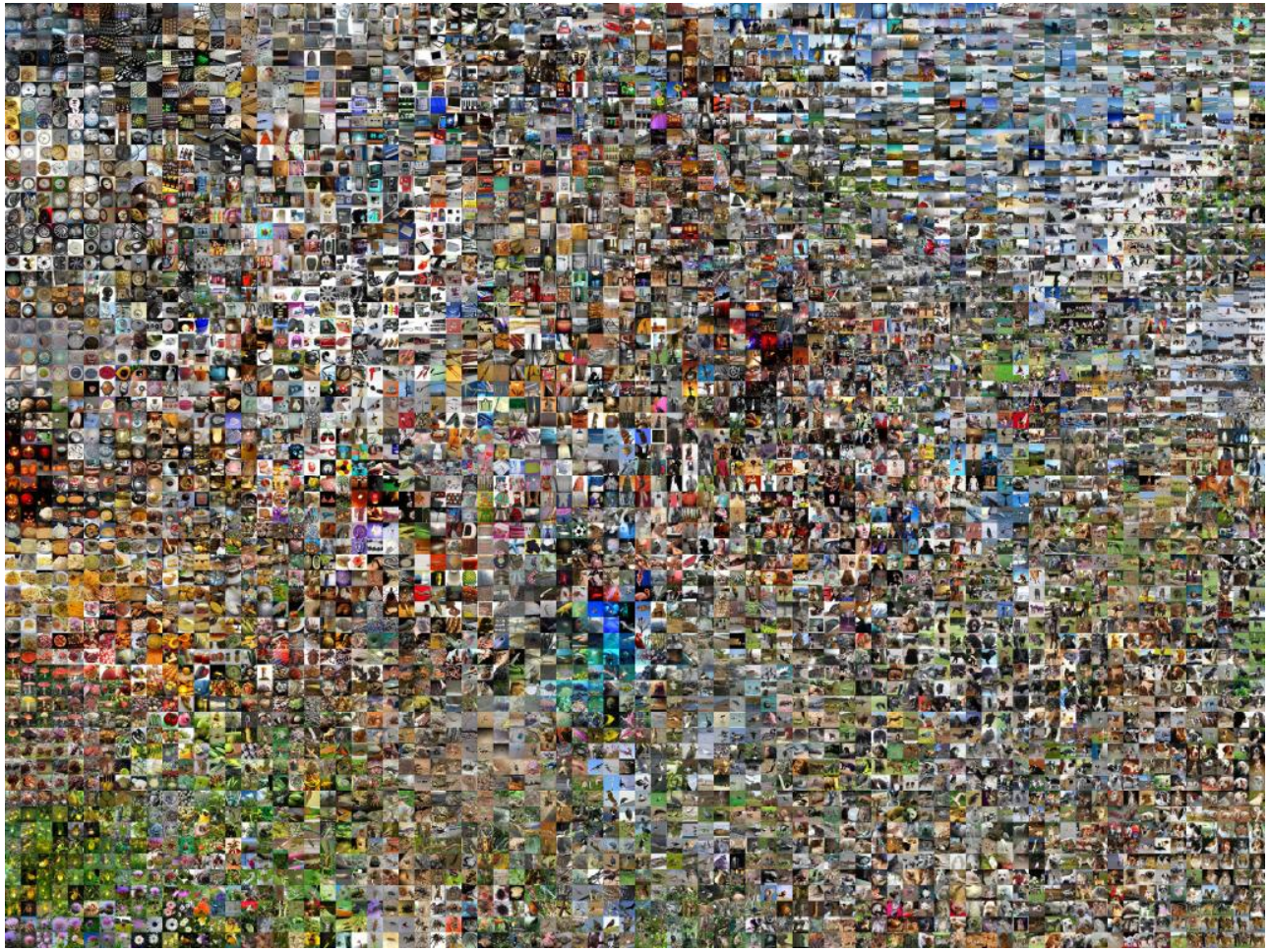




Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

How was this possible?

Large Collections of Annotated Data



The ImageNet project is a large visual database designed for use in visual object recognition software research. ***More than 14 million images*** *have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided.[3]* ***ImageNet contains more than 20,000 categories***

From Wikipedia October 2021

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. *CVPR, 2009*.

Parallel Computing Architectures

And more recently…. Software libraries

TensorFlow

PyTorch

Google LLC, Public domain, via Wikimedia Commons

PyTorch, BSD <http://opensource.org/licenses/bsd-license.php>, via Wikimedia Commons

Data Scarcity

Training a CNN with Limited Aumont of Data

Giacomo Boracchi

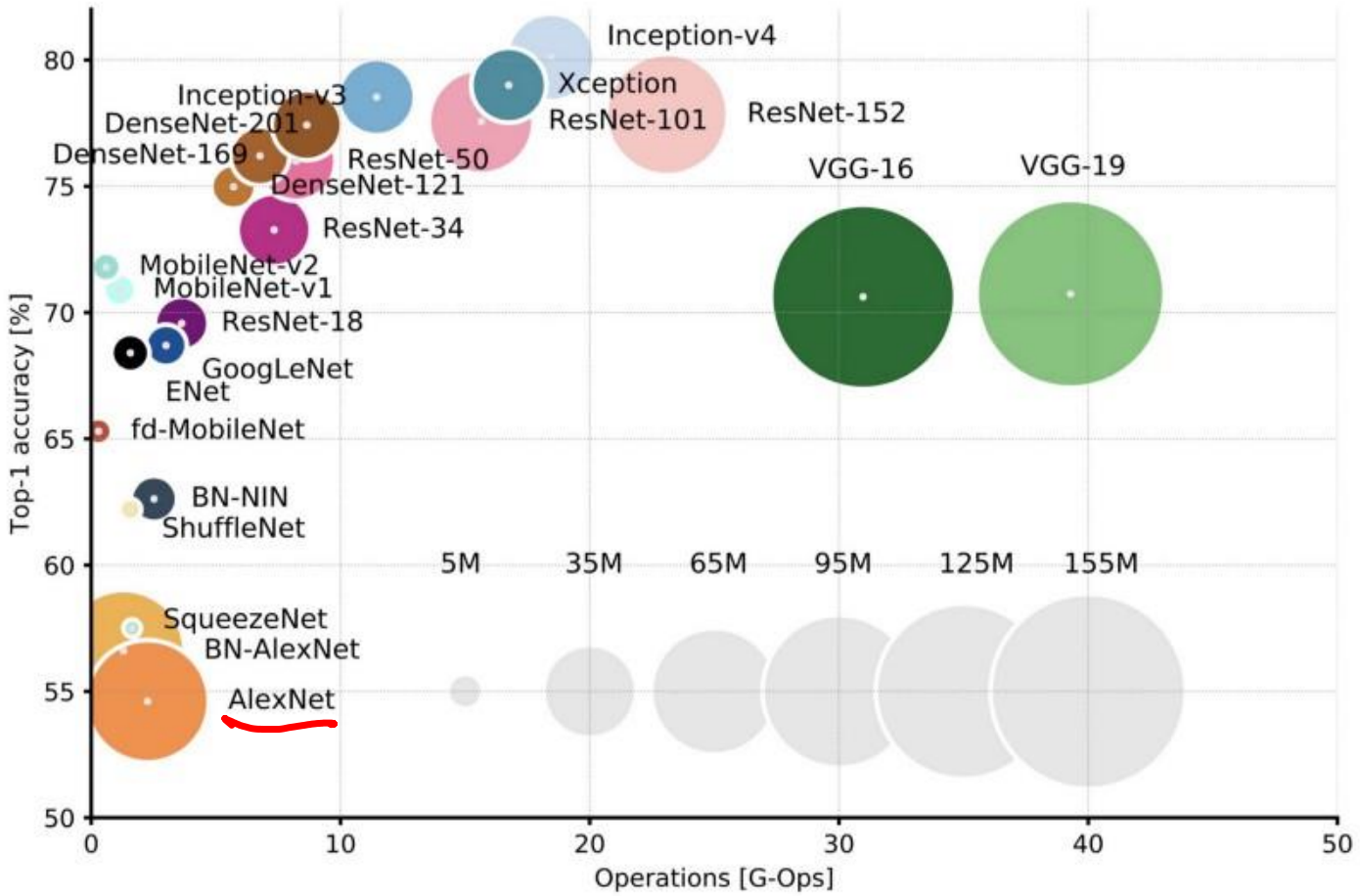
The need of data

Deep learning models are very data hungry.

Networks such as AlexNet have been trained on ImageNet datasets containing tens of thousands of images over hundreds of classes

The need of data

This is necessary to define millions of parameters characterizing these networks



Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

The need of data

Deep learning models are very data hungry.

… watch out: each image in the training set have to be annotated!

How to train a deep learning model with a few training images?

- Data augmentation
- Transfer Learning

Limited Amount of Data: Data Augmentation

Training a CNN with Limited Aumont of Data

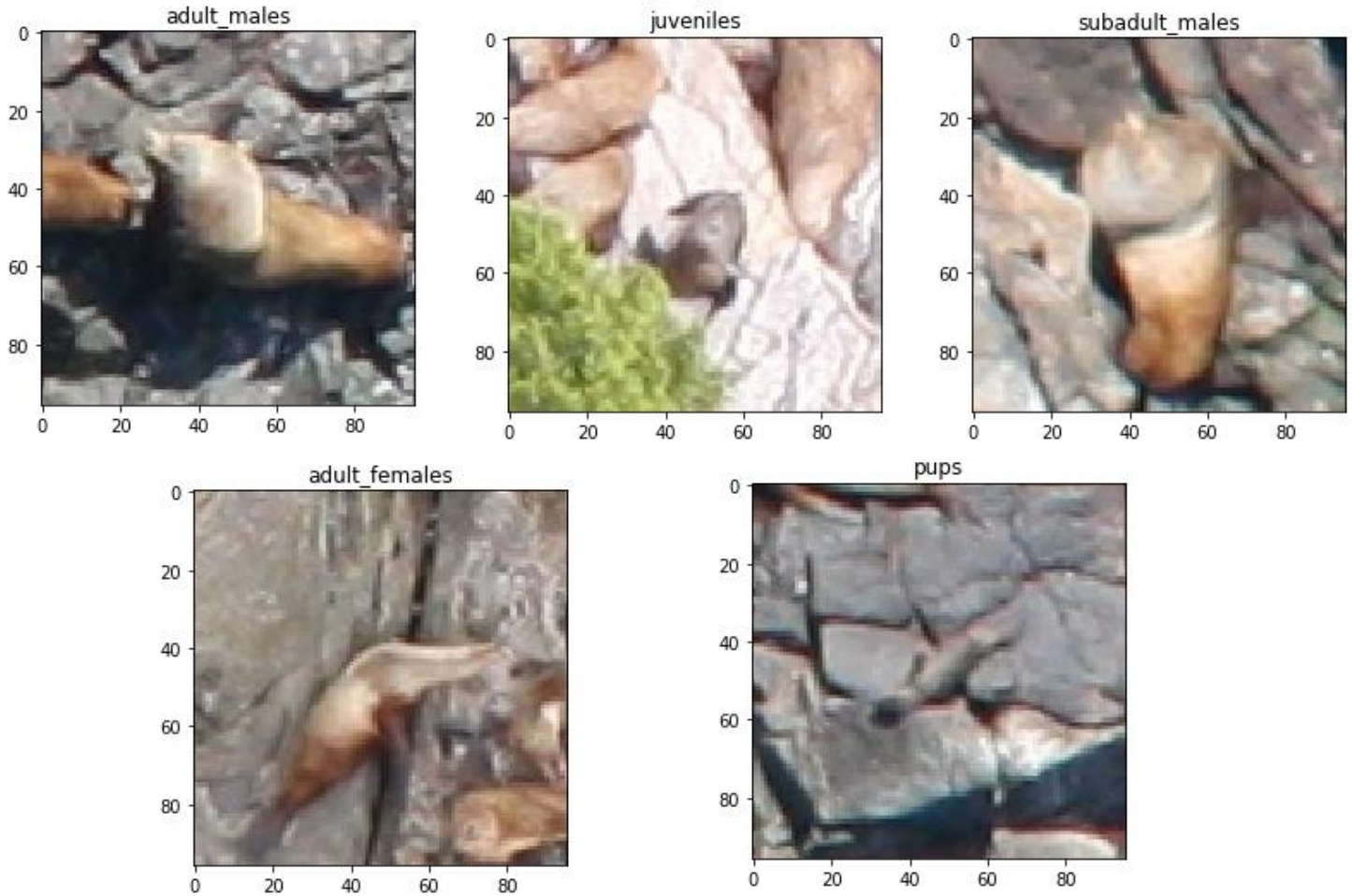
Giacomo Boracchi

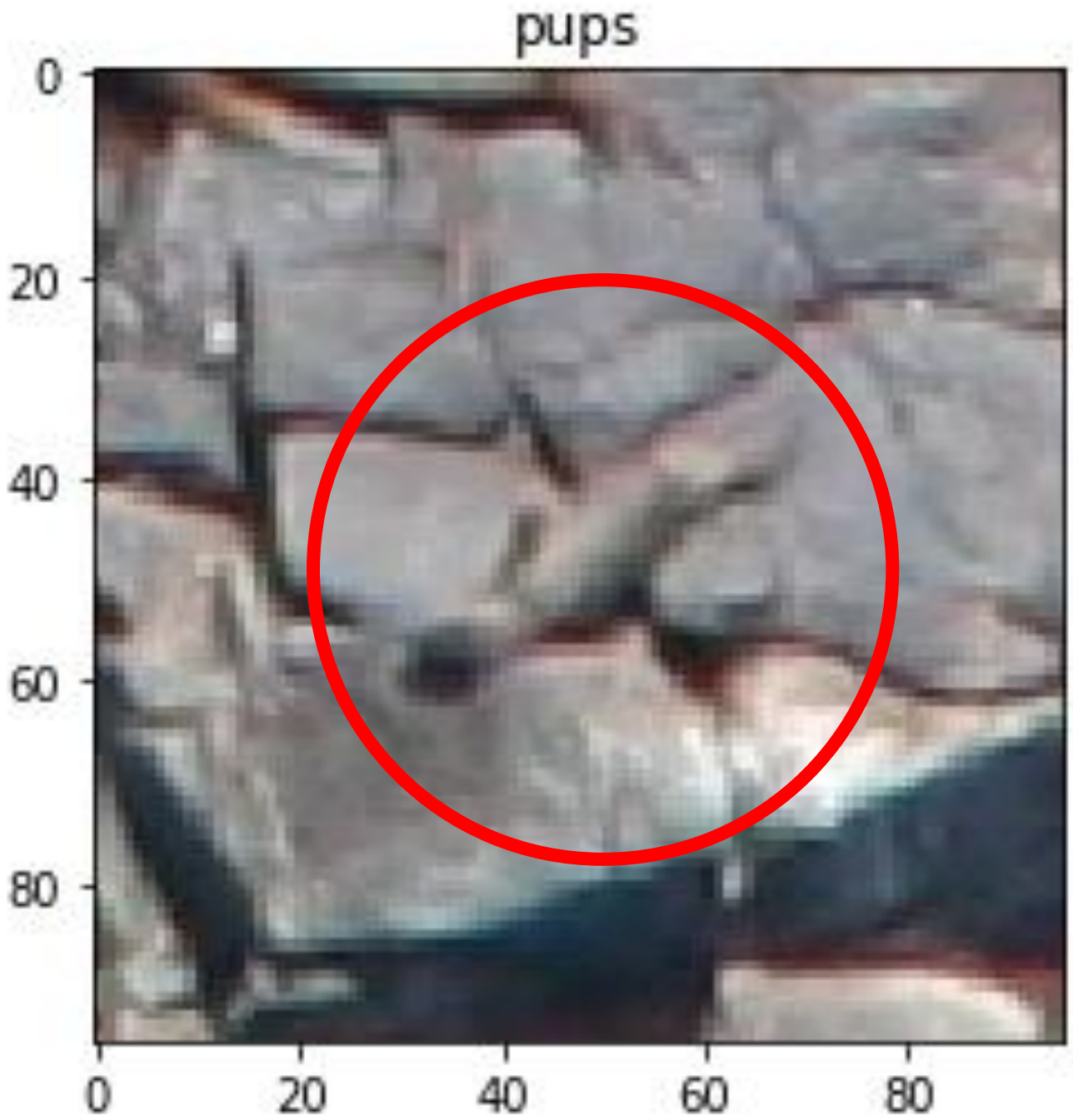

Canada
Stati Uniti
Messico
Aleutian Islands
Steller sea lions in the western Aleutian Islands have declined 94% in the last 30 years.
3D


Kaggle in 2017 have opened a competition to develop algorithms which accurately count the number of sea lions in aerial photographs
Credits Yinan Zhou
https://github.com/marioZYN/FC-CNN-Demo

The Challenge

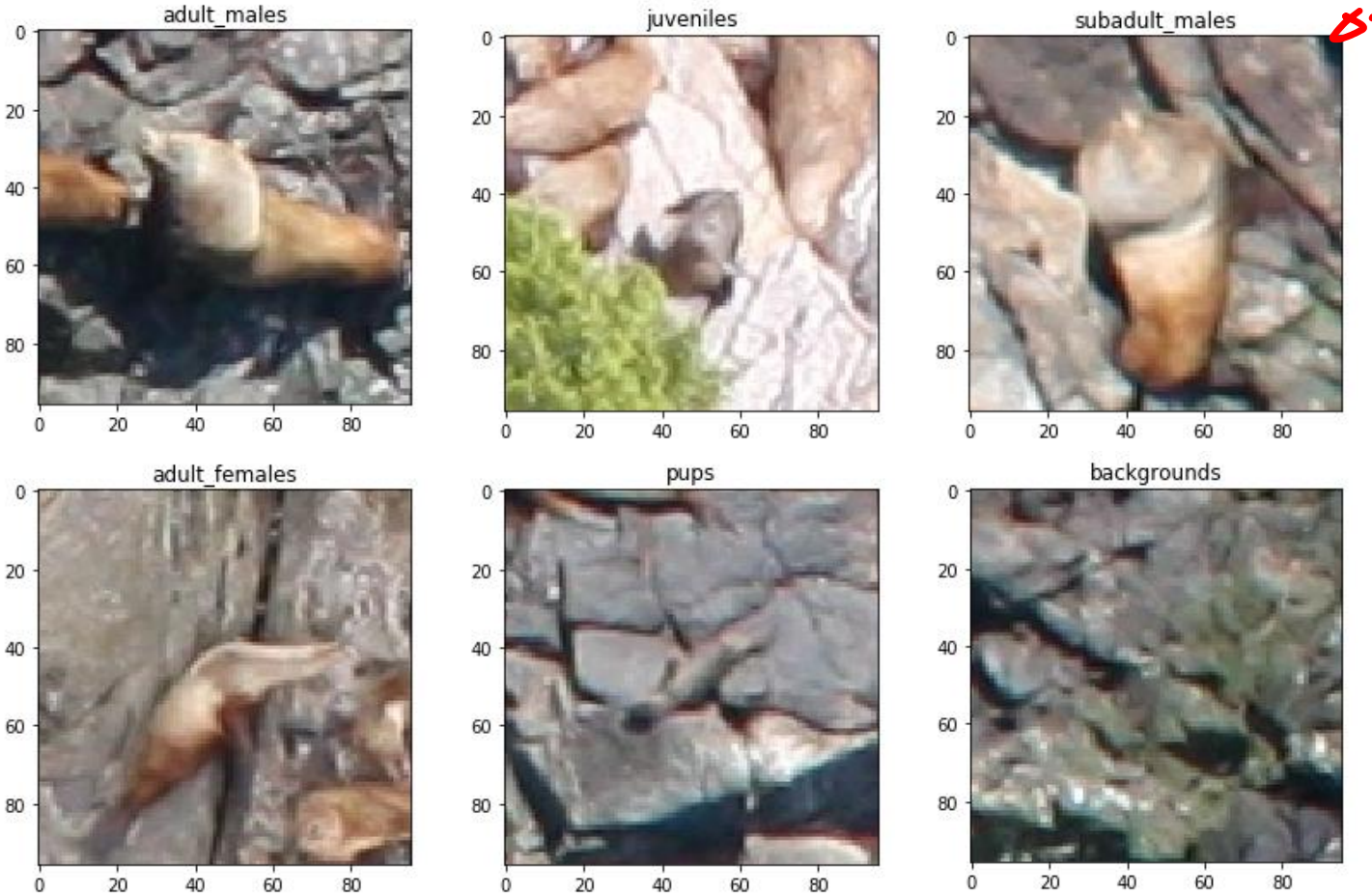
In very large aerial images (≈ 5K x 4K) shot by drones, automatically count the number of sealions per each category



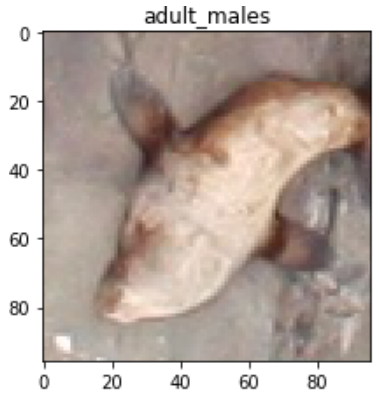

pups
0
20
40
60
80
0
20
40
60
80

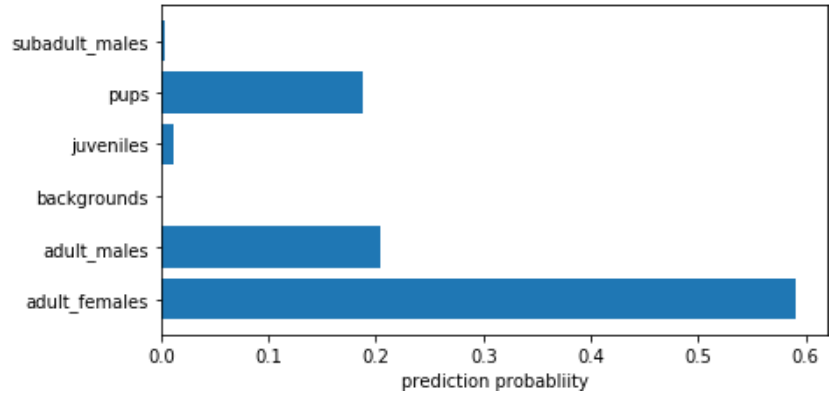
The Challenge

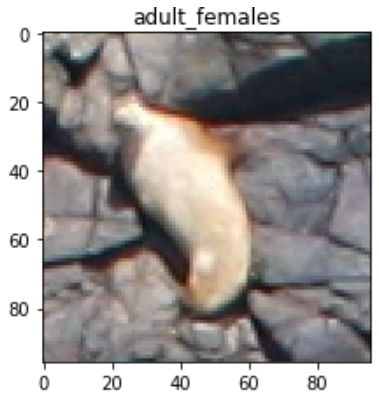
This problem can be naively casted in a patch-by-patch 6-class classification problem, where we include also background

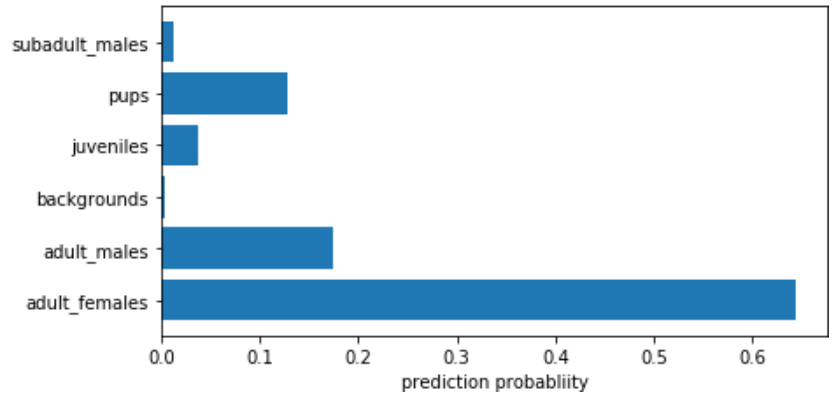


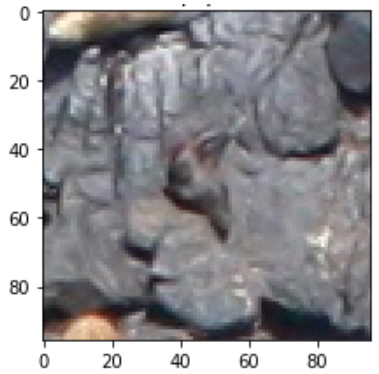
An Example of CNN predictions

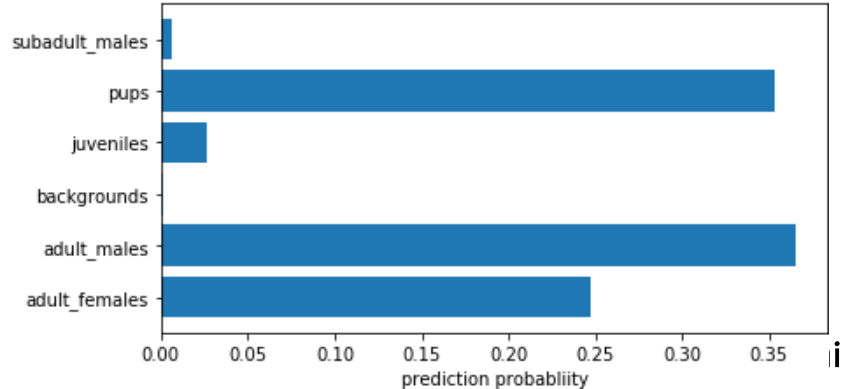












Credits Yinan Zhou
https://github.com/marioZYN/FC-CNN-Demo

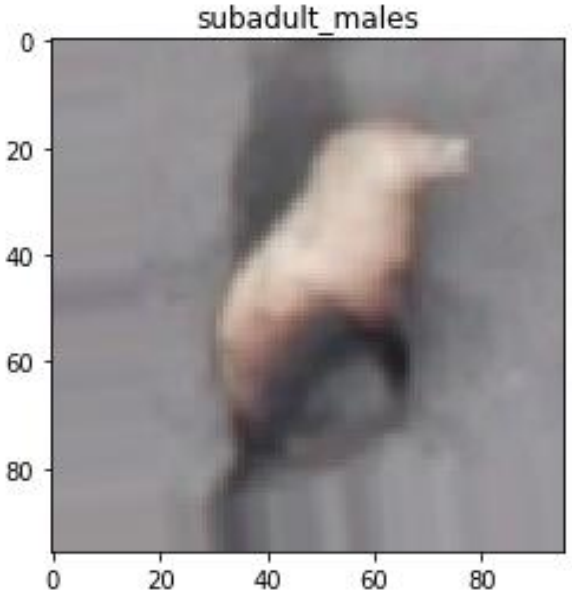
Data Augmentation

Often, each annotated image represents a class of images that are all likely to belong to the same class

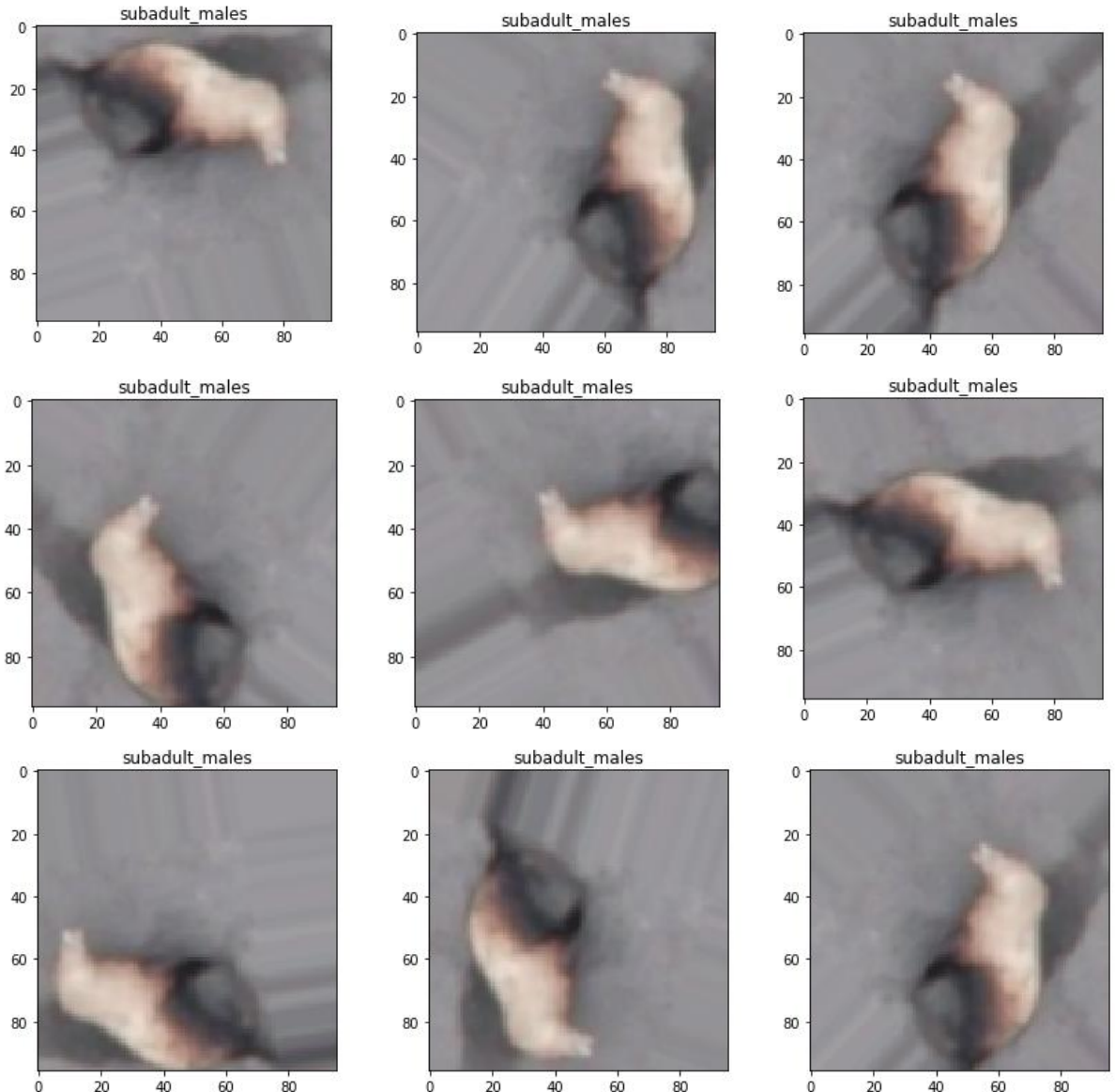
In aereal photograps, for instance, it is normal to have rotated, shifted or scaled images without changing the label

Data Augmentation

Original image



Augmented Images



Data Augmentation

Data augmentation is typically performed by means of

Geometric Transformations:

- Shifts /Rotation/Affine/perspective distortions
- Shear
- Scaling
- Flip

Photometric Transformations:

- Adding noise
- Modifying average intensity
- Superimposing other images
- Modifying image contrast

Data Augmentation: Criteria

Augmented versions **should preserve the input label**

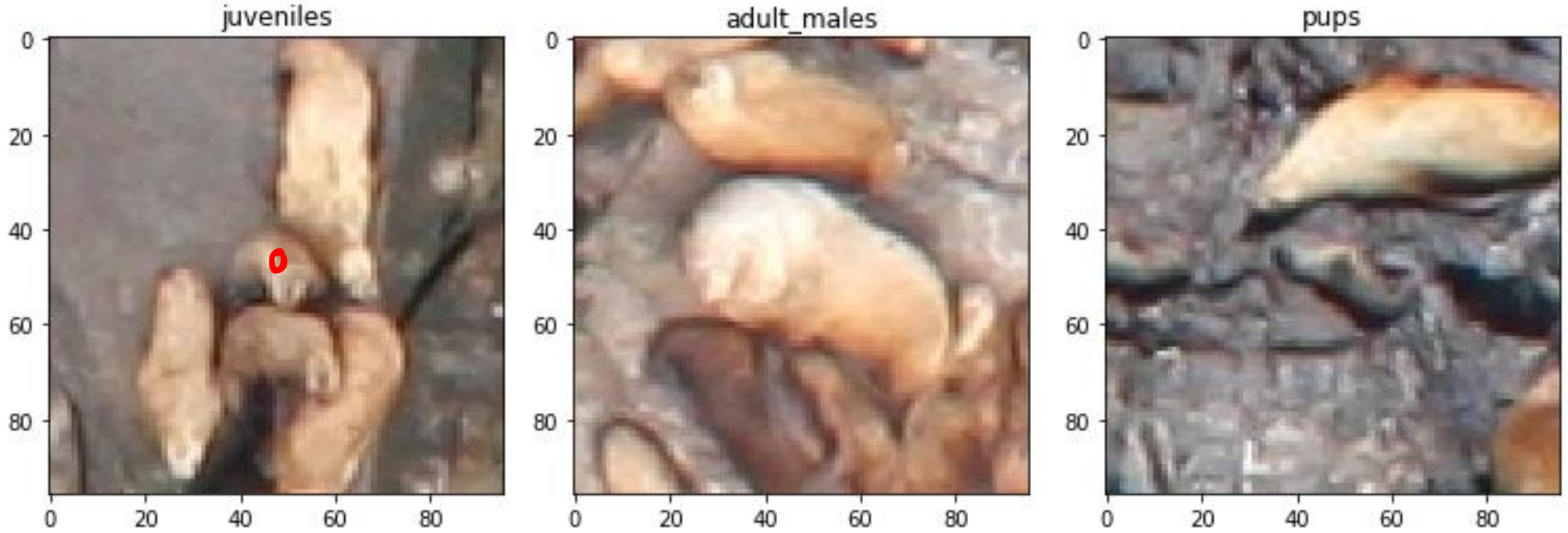
- e.g. if size/orientation is a key information to determine the output target (either the class or the value in case of regression), wisely consider scaling/rotation as transformation

Augmentation is meant to **promote network invariance** w.r.t. transformation used for augmentation

However...

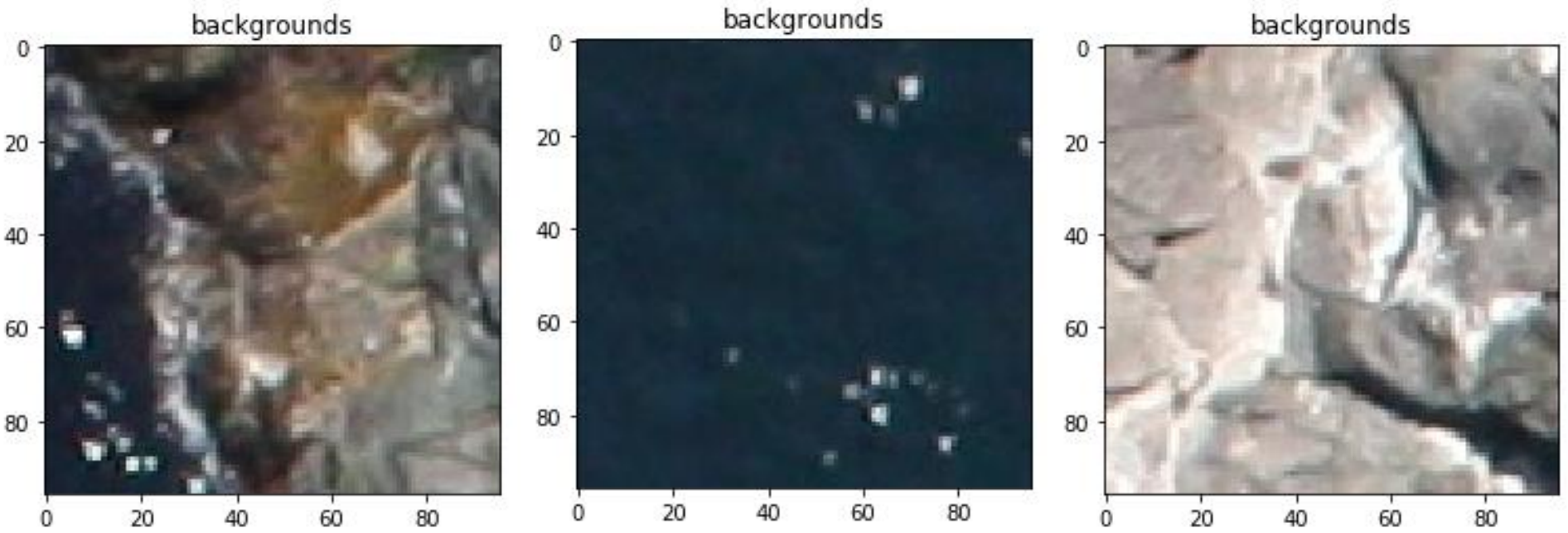
This sort of data augmentation might not be enough to capture the inter-class variability of images...

Superimposition of targets



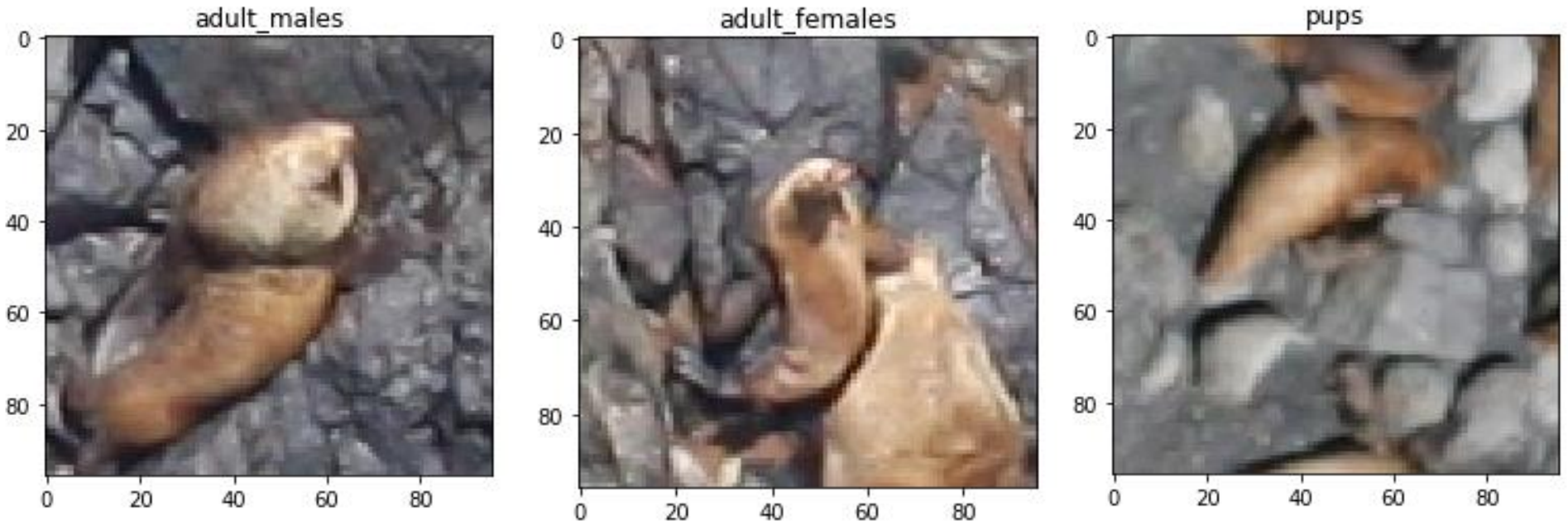
However…

Background variations



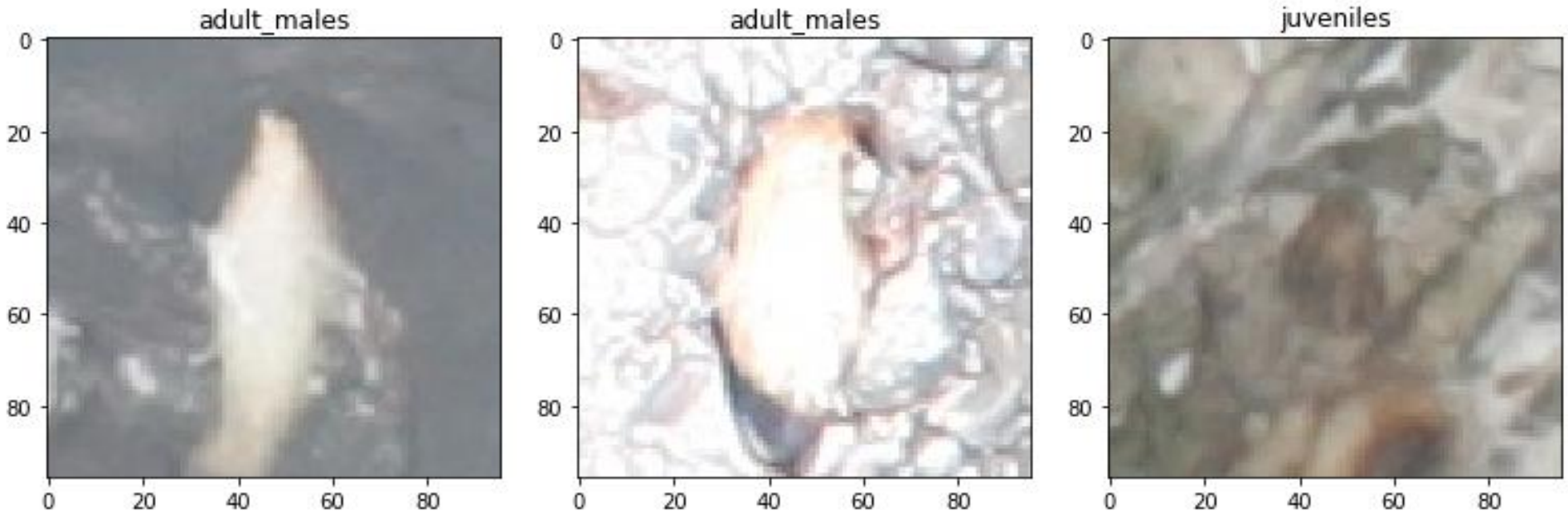
However…

Background variations



However…

Out of focus, bad exposure



Test Time Augmentation (TTA) or Self-ensembling

Even if the CNN is trained using augmentation, **it won't achieve perfect invariance** w.r.t. considered transformations

Test time augmentation (TTA): augmentation can be also performed at test time to improve prediction accuracy.

- Perform random augmentation of each test image I
- Average the predictions of each augmented image
- Take the average vector of posterior for defining the final guess

Test Time Augmentation is particularly useful for test images where the model is pretty unsure.

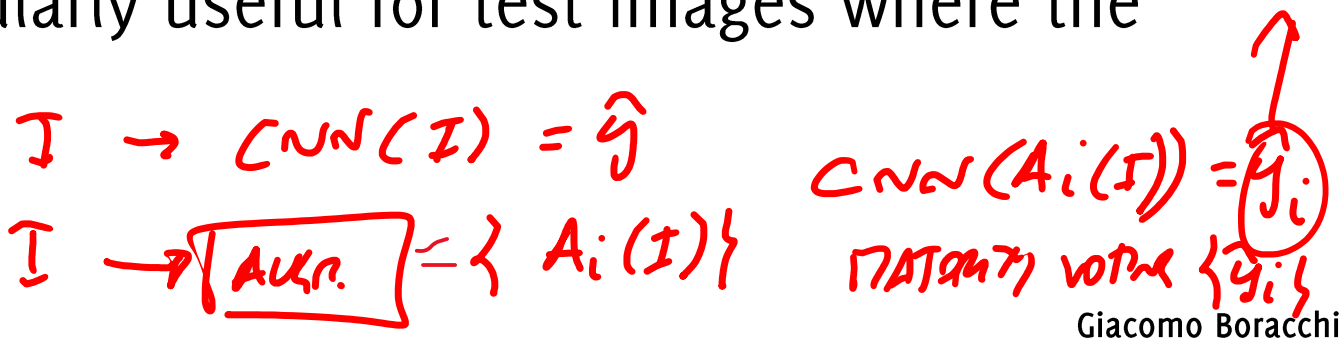
Test Time Augmentation (TTA)

Even if the CNN is trained using augmentation, **it won't achieve perfect invariance** w.r.t. considered transformations

Test time augmentation (TTA): augmentation can be also performed at test time to improve prediction accuracy.

- Perform random augmentation of each test image I
- Average the predictions of each augmented image
- Take the average vector of posterior for defining the final guess

Test Time Augmentation is particularly useful for test images where the model is pretty unsure.



Augmentation in Keras

```
from keras.preprocessing.image import
ImageDataGenerator

ImageDataGenerator(
rotation_range=0,
width_shift_range=0.0, height_shift_range=0.0,
brightness_range=None, shear_range=0.0,
zoom_range=0.0, channel_shift_range=0.0,
fill_mode='nearest',
horizontal_flip=False, vertical_flip=False,
rescale=None,
preprocessing_function=None)
```

Augmentation in Keras: flow from images

The Image generator has a method **`flow_from_directory`** that allows to load images in folder where different classes are arranged in subfolders.

```
ImageDataGenerator.flow_from_directory(
    directory=PATCH_PATH + 'train/',
    target_size=(img_width, img_width),
    batch_size=batch_size,
    shuffle=True)
```

Augmentation in Keras

```
from keras.preprocessing.image import
ImageDataGenerator

ImageDataGenerator(
rotation_range=0,
width_shift_range=0.0, height_shift_range=0.0,
brightness_range=None, shear_range=0.0,
zoom_range=0.0, channel_shift_range=0.0,
fill_mode='nearest',
horizontal_flip=False, vertical_flip=False,
rescale=None,
preprocessing_function=None)
```

… in case you need some extra flexibility

Benefits of Data Augmentation

Giacomo Boracchi

Image Augmentation and Overfitting

Given an annotated image (I, y) and a set of augmentation transformations $\{A_l\}_l$, we train the network using these pairs

$$\{(A_l(I), y)_l\}_l$$

Training including augmentation **reduces the risk of overfitting**, as it significantly increase the training set size

Moreover, data augmentation can be used to compensate for class imbalance in the training set, by **creating more realistic examples from the minority class**

Transformations used in data-augmentation $\{A_l\}$ can be also class-specific, in order to preserve the image label

Image Augmentation and CNN invariance

Given an annotated image (I, y) and a set of augmentation transformations $\{A_l\}_l$, we train the network using these pairs

$$\{(A_l(I), y)_l\}_l$$

Through data augmentation we train the network to «become invariant» to selected transformations. Since the same label is associated to I and $A_l(I)\ \forall l$

Unfortunately, invariance might not be always achived in practice

Test Time Augmentation (TTA) or Self-ensembling

Test time augmentation (TTA): augmentation can be also performed at test time to improve prediction accuracy.

- Perform a few random augmentation of each test image I

$$\{A_l(I)\}_l$$

- Classify all the augmented images and save the posterior vectors

$$\boldsymbol{p_l} = CNN(A_l(I))$$

- Define the CNN prediction by aggregating the posterior vectors $\{\boldsymbol{p}_l\}$

$$\text{e.g. } \boldsymbol{p} = Avg(\{\boldsymbol{p}_l\}_l)$$

Test Time Augmentation is particularly useful for test images where the model is pretty unsure.

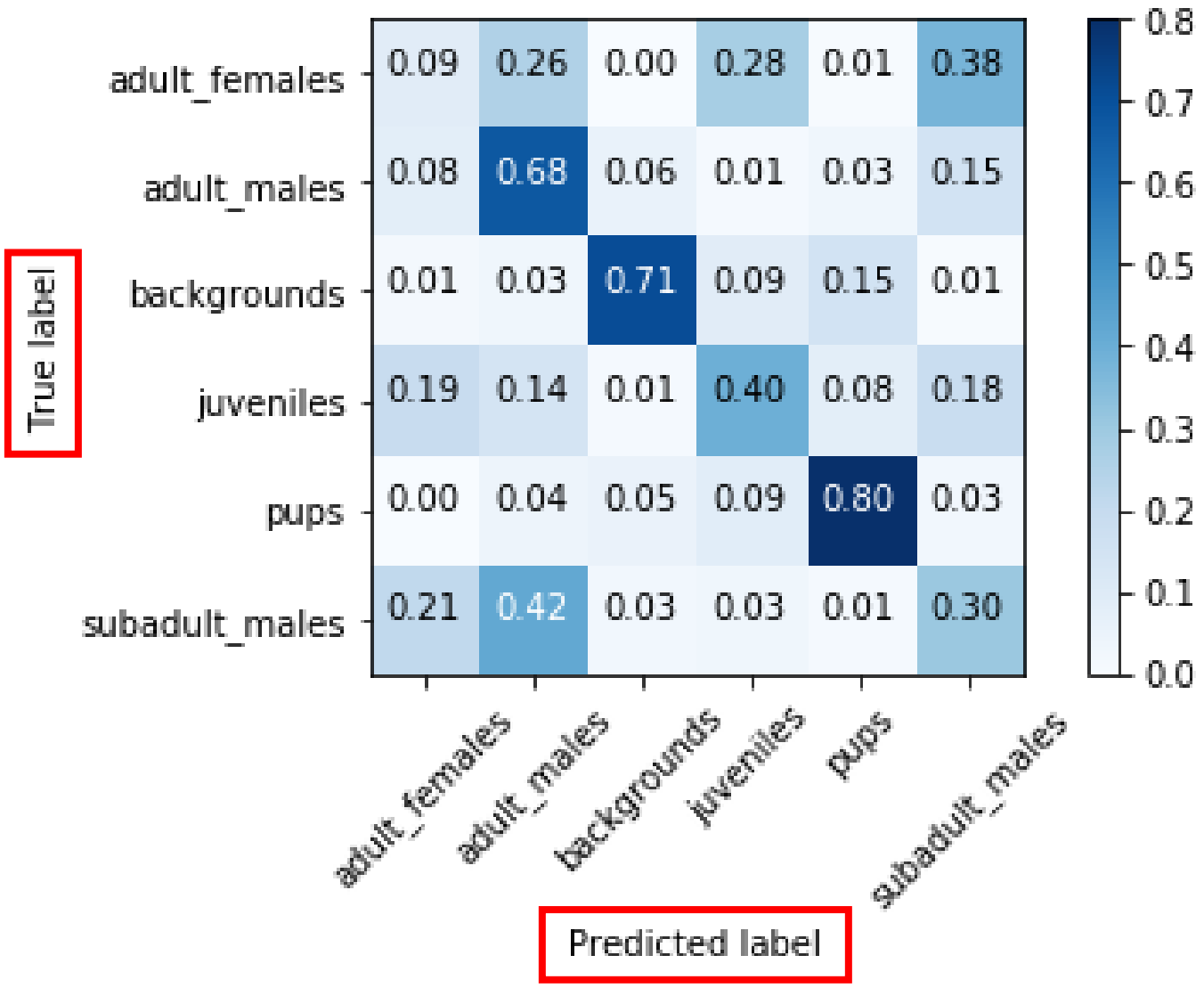
A bit more of background

Performance measures

and an overview of successful architectures

Giacomo Boracchi

Confusion Matrix

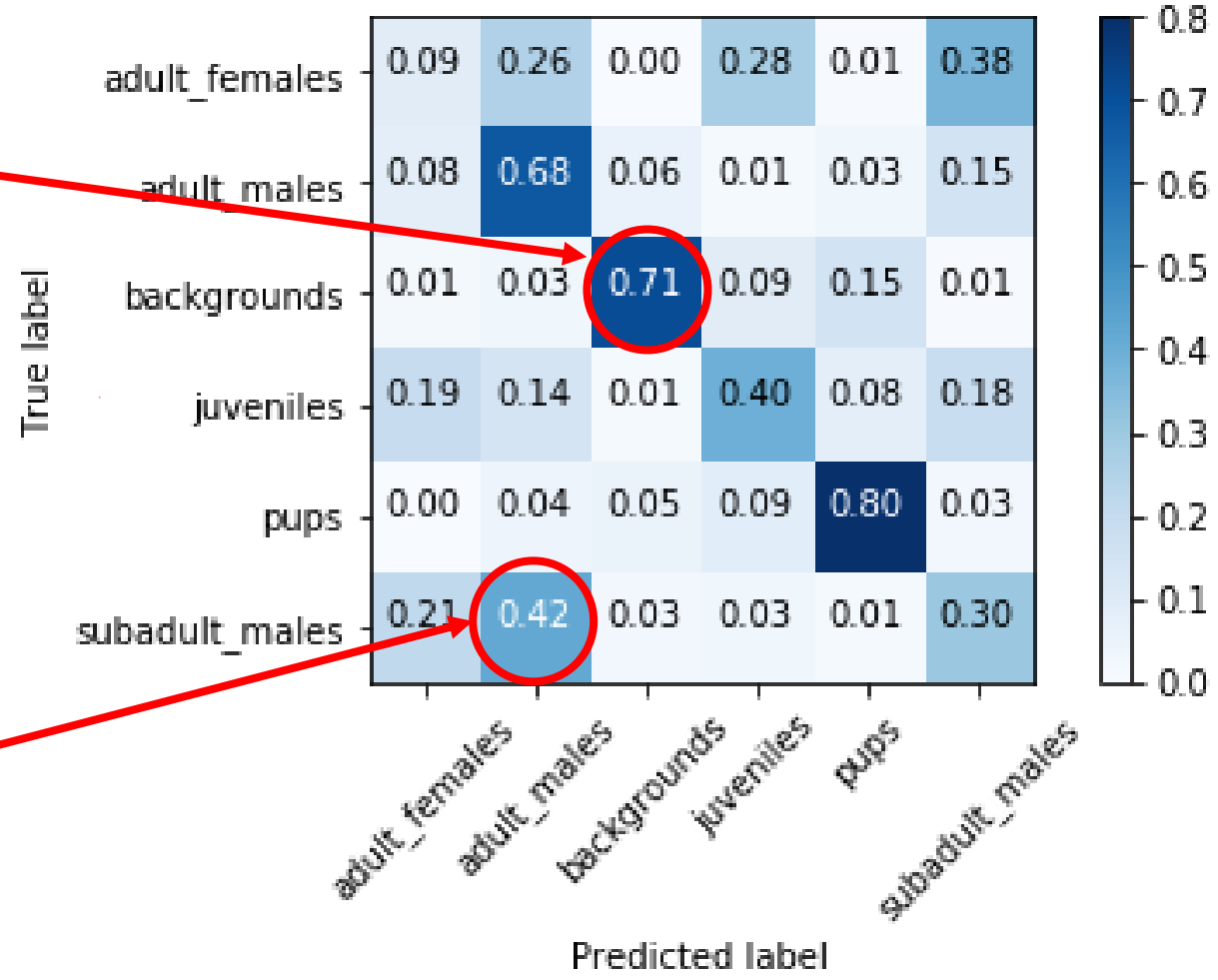


Confusion Matrix

The element $C(i,j)$ i.e. at the i-th row and j-th column corresponds to the percentage of elements belonging to class i classified as elements of class j

71% of background patches have been correctly classified as background

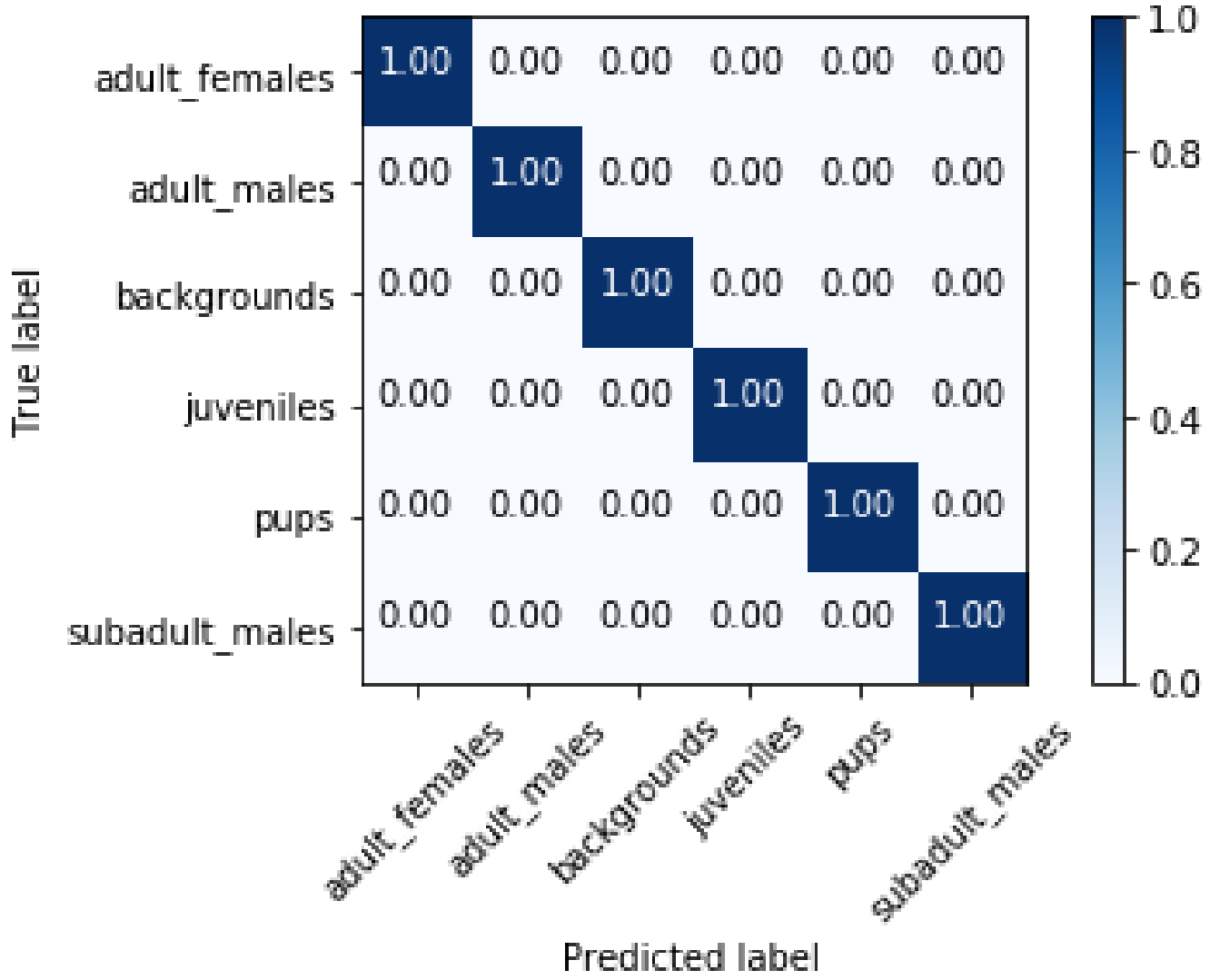
42% of sub-adult males patches have been wrongly classified as adult-males



hi

… so, the ideal confusion matrix

Which rarely happens



Two-Class Classification

Background:

In a two-class classification problem (binary classification), the **CNN output is equivalent to a scalar,** since

$$CNN(I) = [p, 1-p]$$

being p the probability of I to belong to the first class.

Thus we can write

$$CNN(I) = p$$

Then, we can decide that I belongs to the first class when

$$CNN(I) > \Gamma$$

and use Γ different from 0.5, which is the standard.

We require stronger evidence before claiming I **belongs to class 1.**

Changing Γ **establishes a trade off between FPR and TPR.**

Two-Class Classification

Classification performance in case of **binary classifiers** can be also measured in terms of the **ROC** (receiver operating characteristic) **curve**, which does not depend on the threshold you set for each class

This is useful in case you plan to modify this and not use 0.5

The ideal detector would achieve:

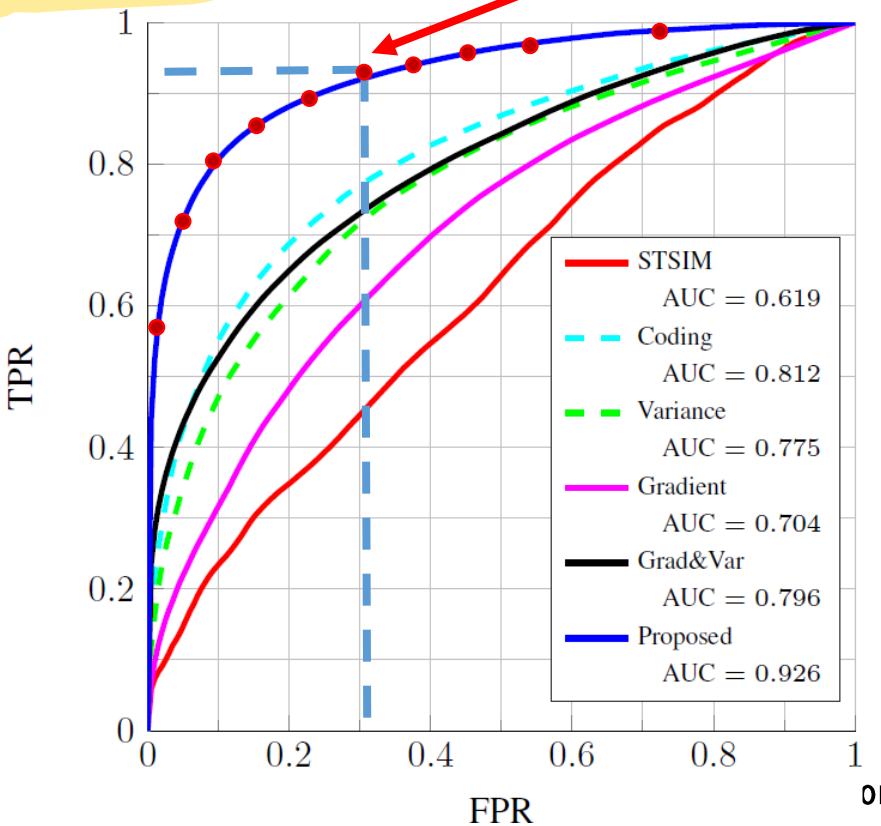
- $FPR = 0\%$,
- $TPR = 100\%$

Thus, the closer to (0,1) the better

The largest the **Area Under the Curve** (AUC), the better

The optimal parameter is the one yielding the point closest to (0,1)

(FPR, TPR) for a specific parameter



omo Boracchi

CNN for Quality Inspection

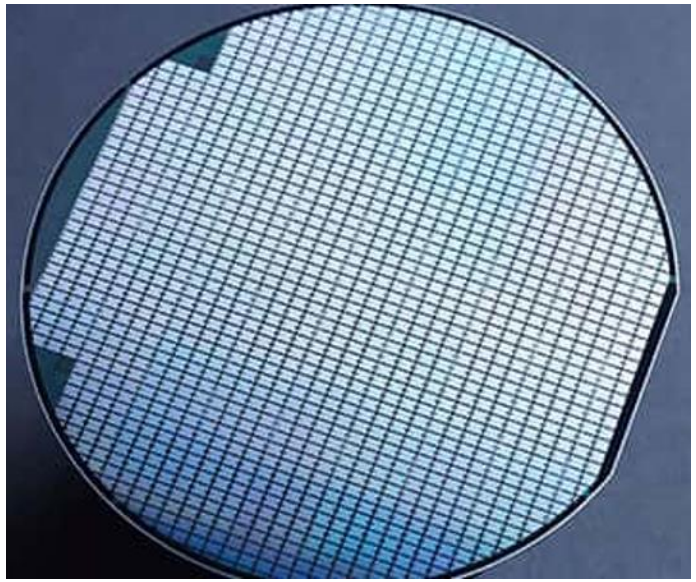
In collaboration with

Giacomo Boracchi

Scenario

Chip Manufacturer

Silicon Wafer



Chips / Memories / Sensors are everywere

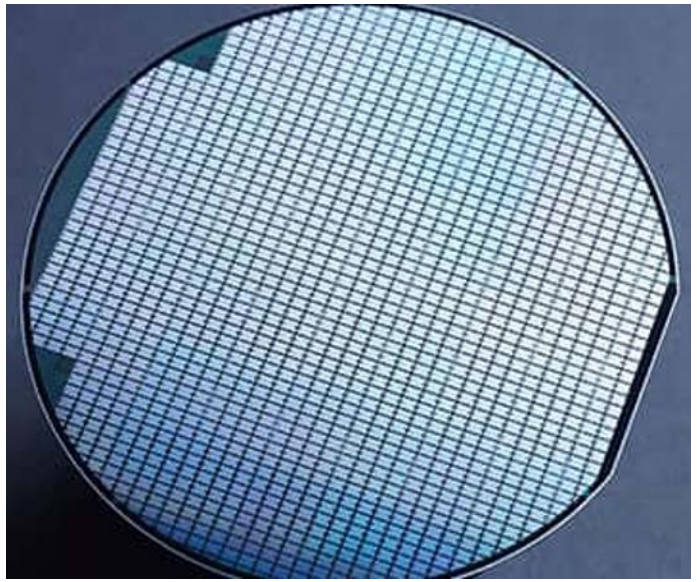
In collaboration with

life.augmented

Frittoli, L., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2022). Deep open-set recognition for silicon wafer production monitoring. *Pattern Recognition*, *124*, 108488.

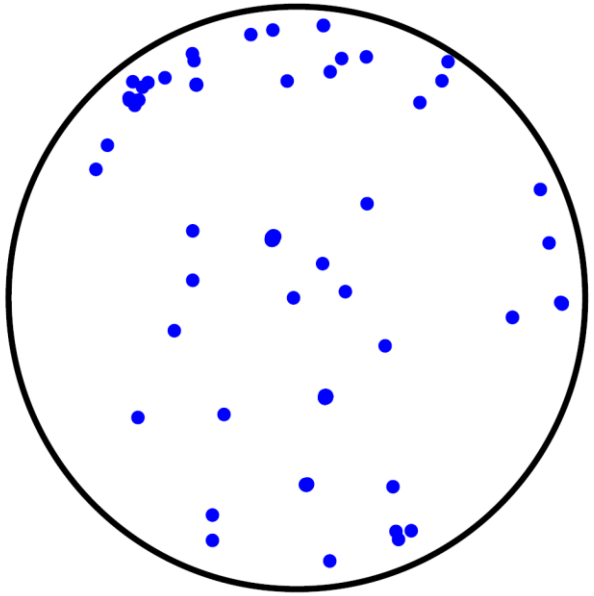
Monitoring Silicon Wafer Manufacturing Process

Wafer



Inspection Tool

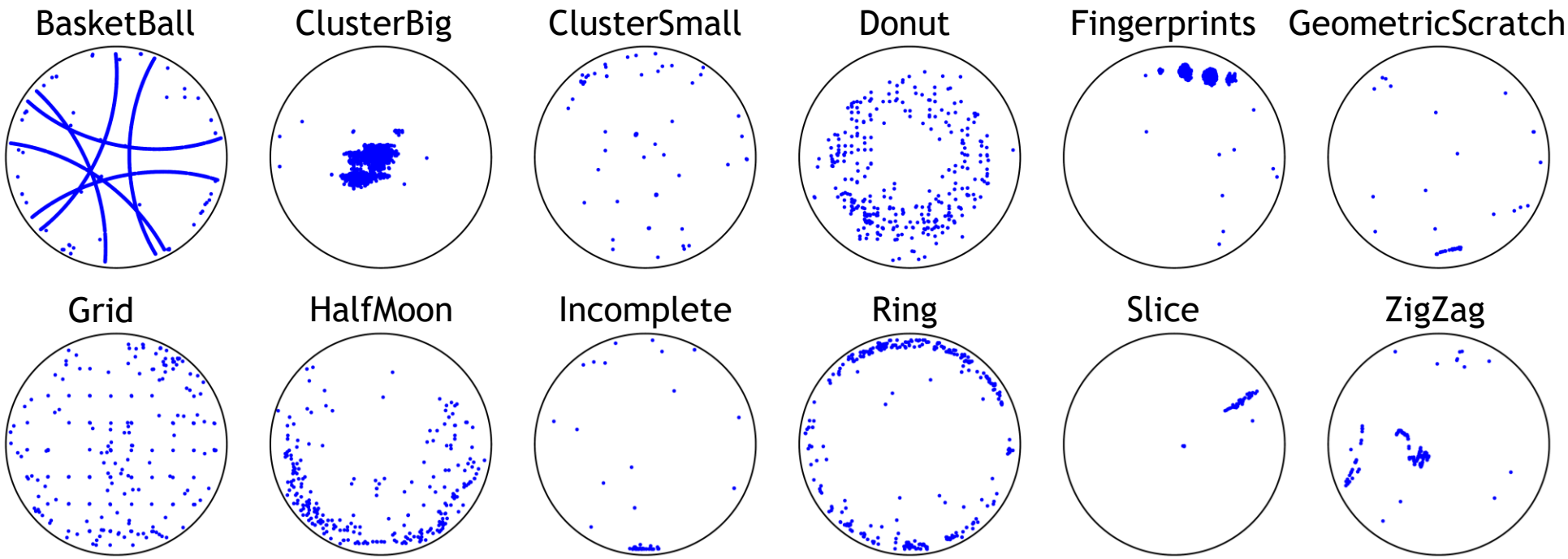
Wafer Defect Map (WDM)



Frittoli, L., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2022). Deep open-set recognition for silicon wafer production monitoring. *Pattern Recognition, 124,* 108488.

Classess of WDM Patterns

Specific **patterns** in WDMs might indicate **problems** in the production

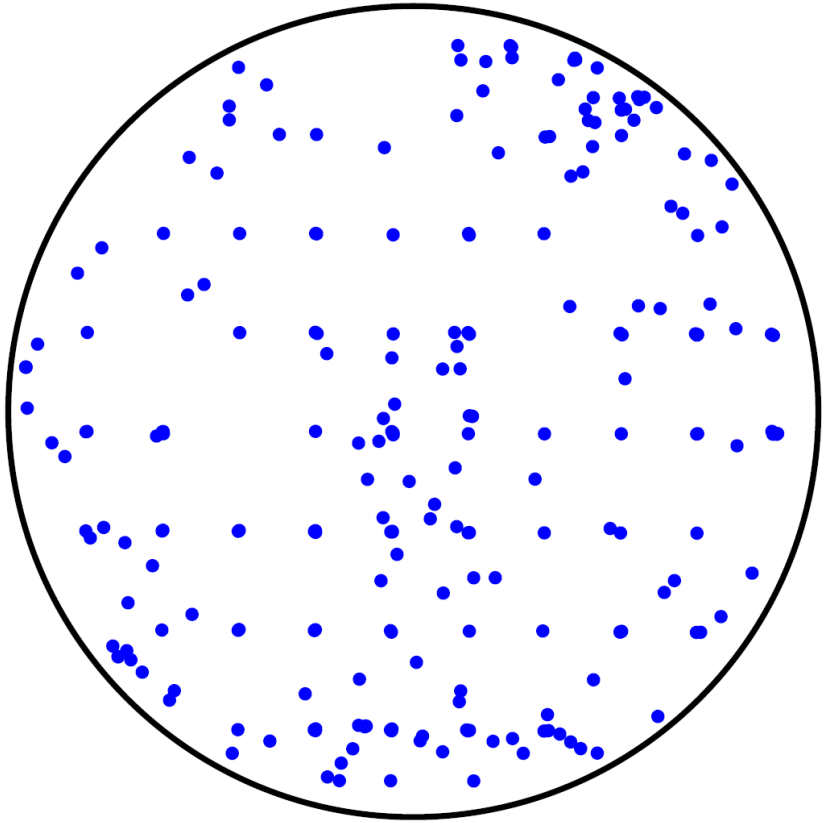


Classify WDM to raise prompt alerts

Frittoli, L., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2022). Deep open-set recognition for silicon wafer production monitoring. *Pattern Recognition, 124,* 108488.

Challenges

- Huge resolution: a WDM as a grayscale would require ~ 3 GB to store w in memory
- Very Limited Supervision
- Some defects occur very rarely

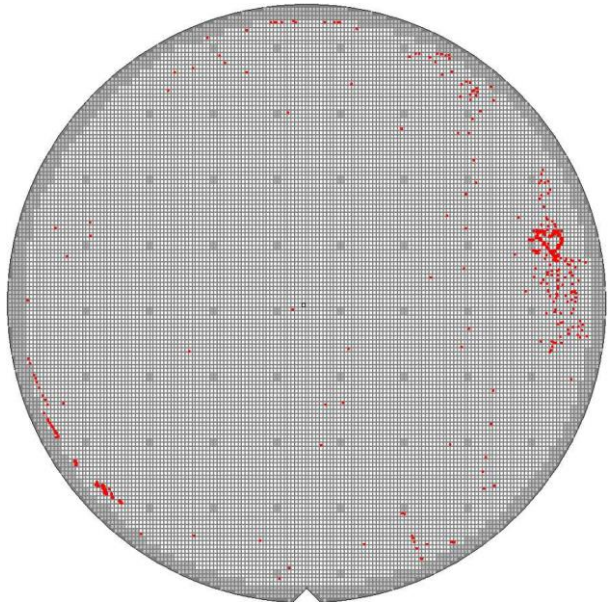


Frittoli, L., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2022). Deep open-set recognition for silicon wafer production monitoring. *Pattern Recognition*, *124*, 108488.

Our CNN

Collaboration with

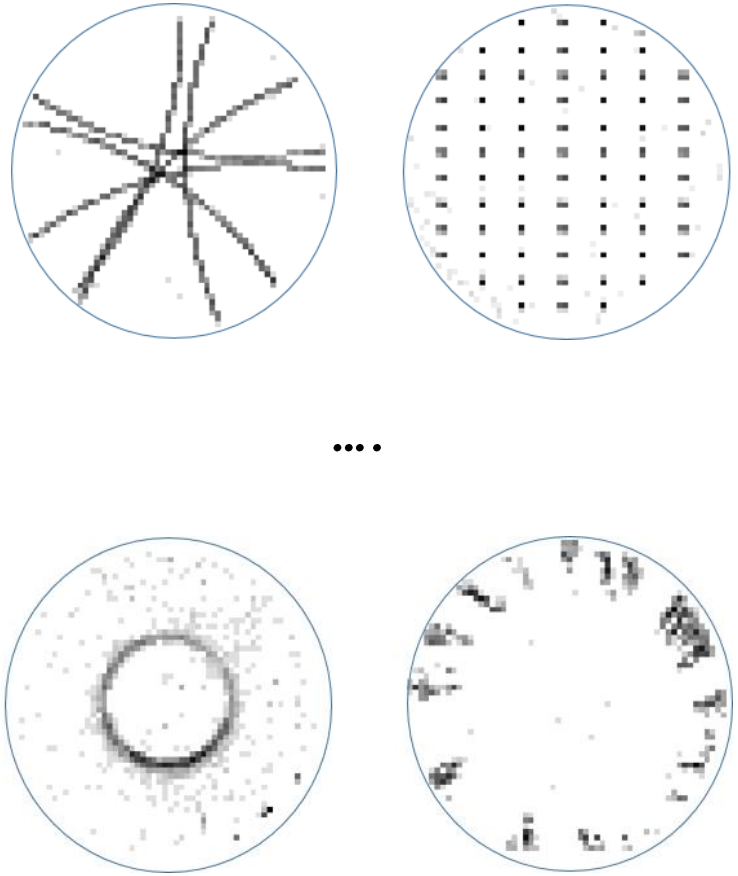
Train a deep learning model to identify defective patterns



Wafer Defect Map

Deep Learning

Defect Patterns



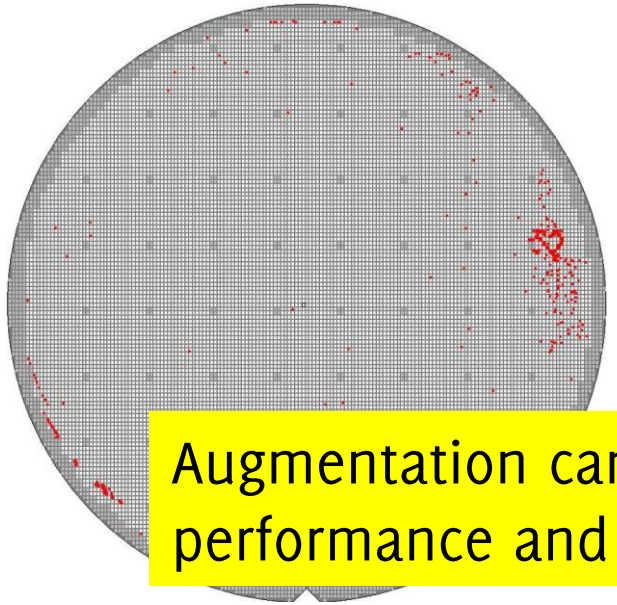
R. di Bella, D. Carrera, B. Rossi, P. Fragneto, G. Boracchi «Wafer Defect Map Classification Using Sparse Convolutional Networks" ICIAP 2019

Data Augmentation is often key..

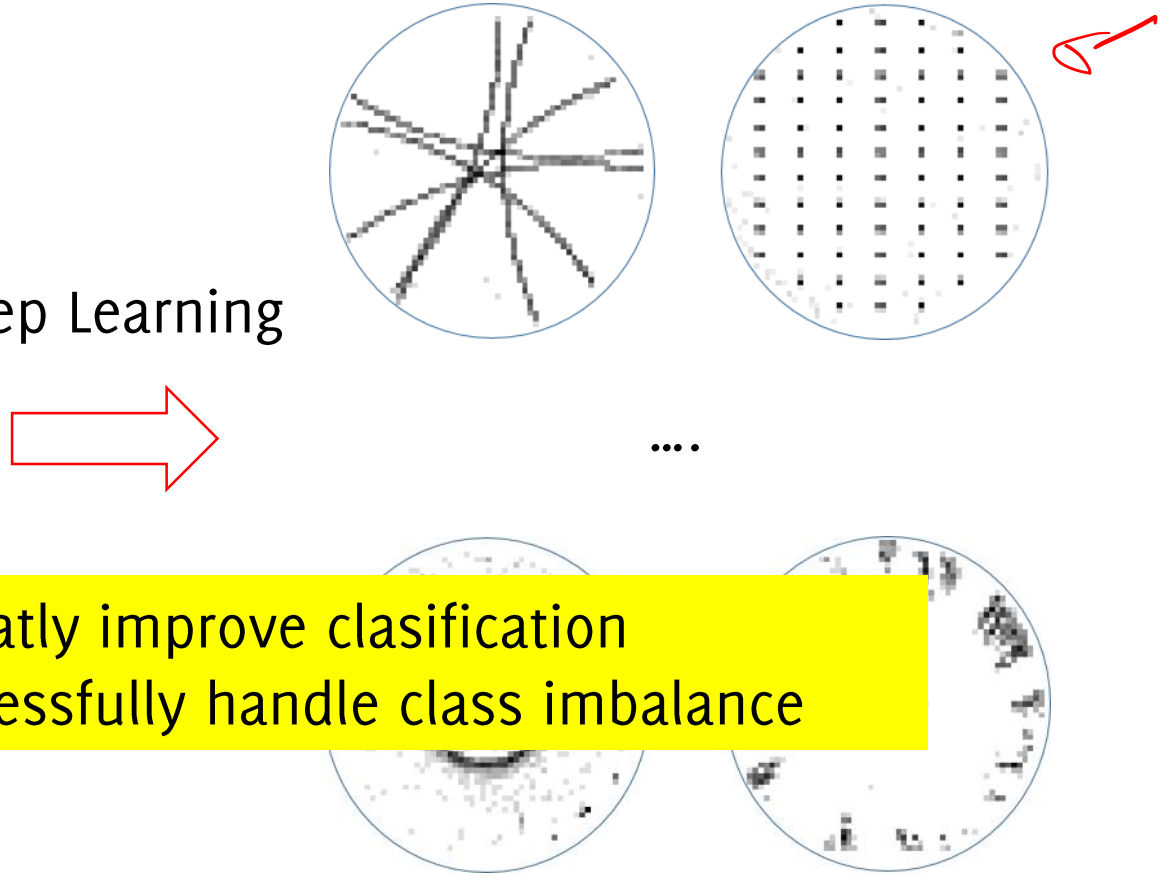
Collaboration with life.augmented

Train a deep learning model to identify defective patterns

Defect Patterns



Deep Learning

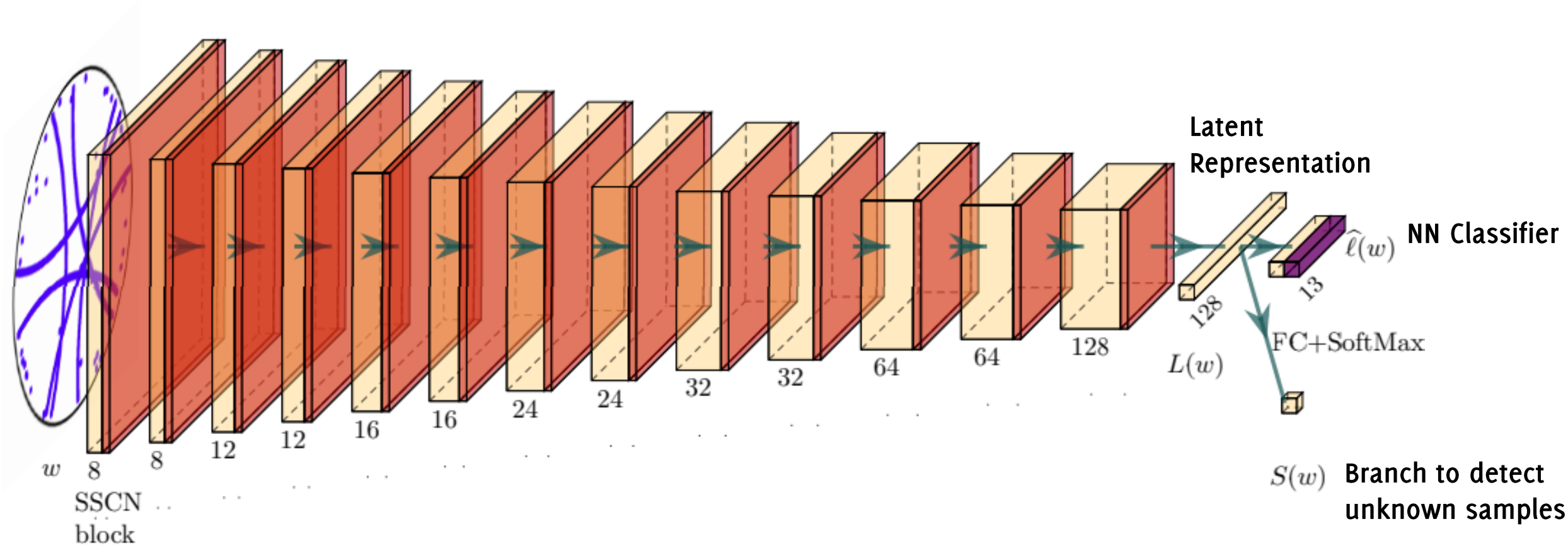


Augmentation can greatly improve clasification performance and successfully handle class imbalance

Wafer Defect Map

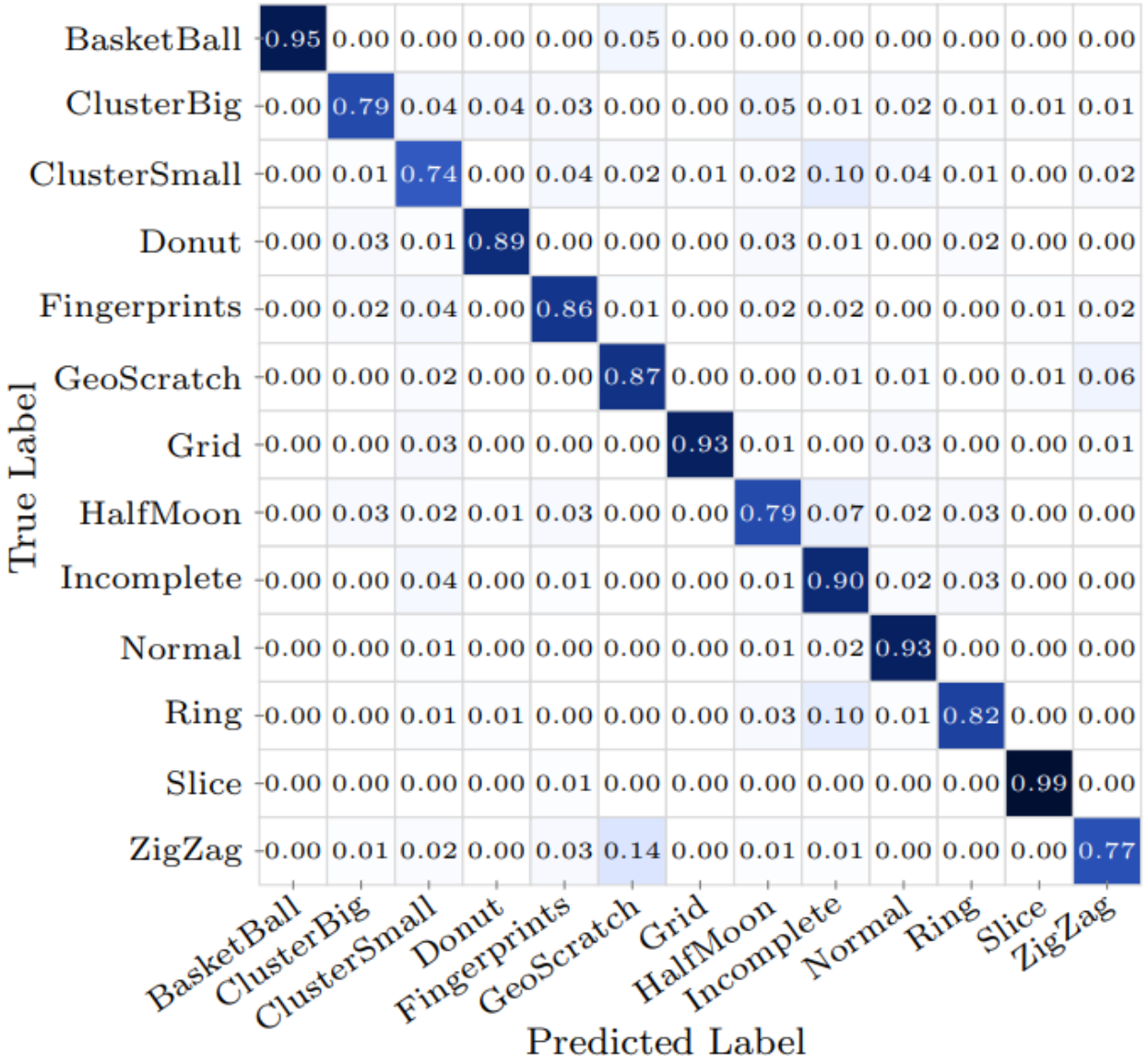
R. di Bella, D. Carrera, B. Rossi, P. Fragneto, G. Boracchi «Wafer Defect Map Classification Using Sparse Convolutional Networks" ICIAP 2019

Our CNN



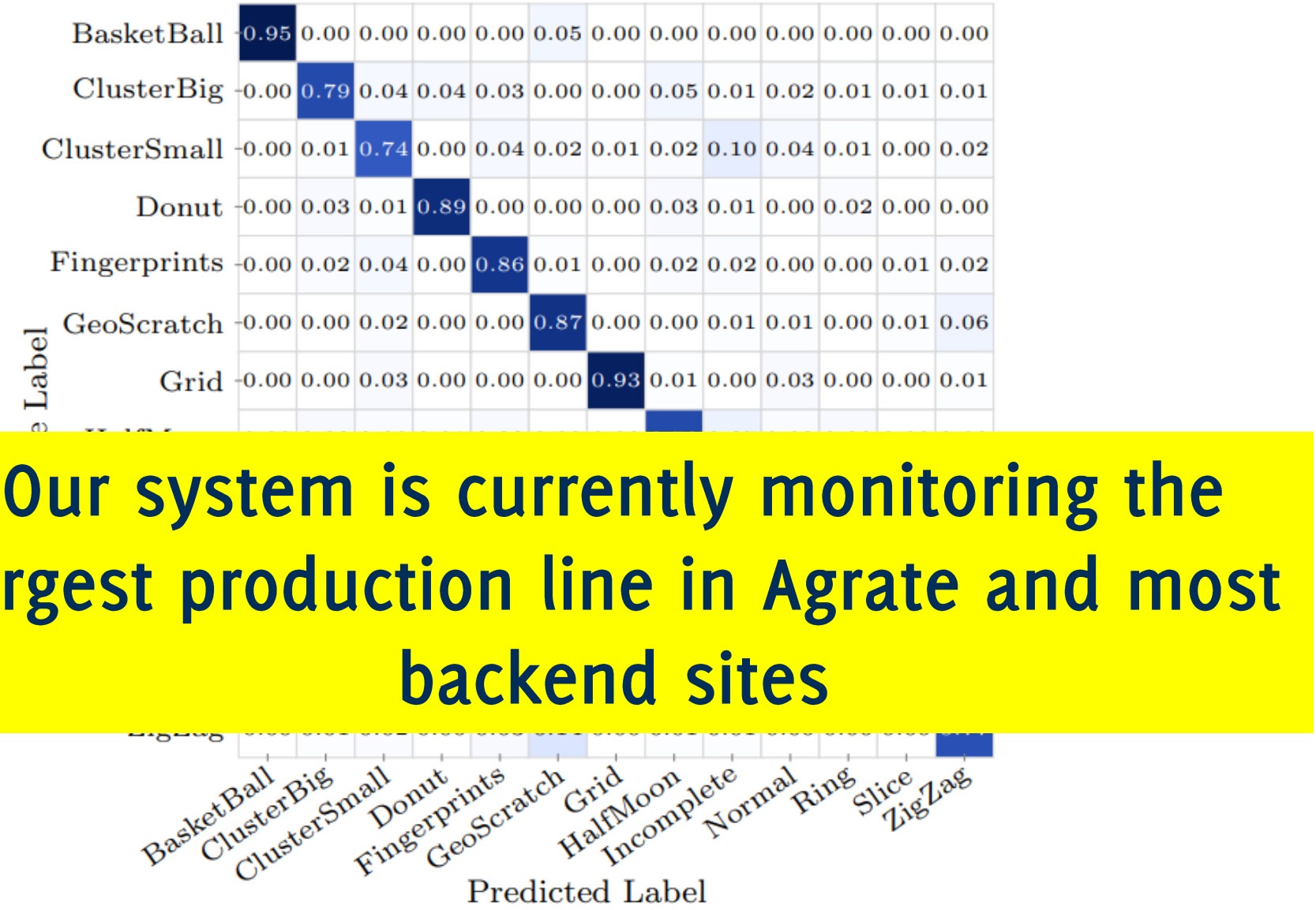
Frittoli, L., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2022). Deep open-set recognition for silicon wafer production monitoring. *Pattern Recognition, 124,* 108488.

Results



Frittoli, L., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2022). Deep open-set recognition for silicon wafer production monitoring. *Pattern Recognition, 124,* 108488.

Results



Our system is currently monitoring the largest production line in Agrate and most backend sites

Frittoli, L., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2022). Deep open-set recognition for silicon wafer production monitoring. *Pattern Recognition, 124*, 108488.

Limited Amount of Data: Transfer Learning

Training a CNN with Limited Aumont of Data

Giacomo Boracchi

The Rationale Behind Transfer Learning

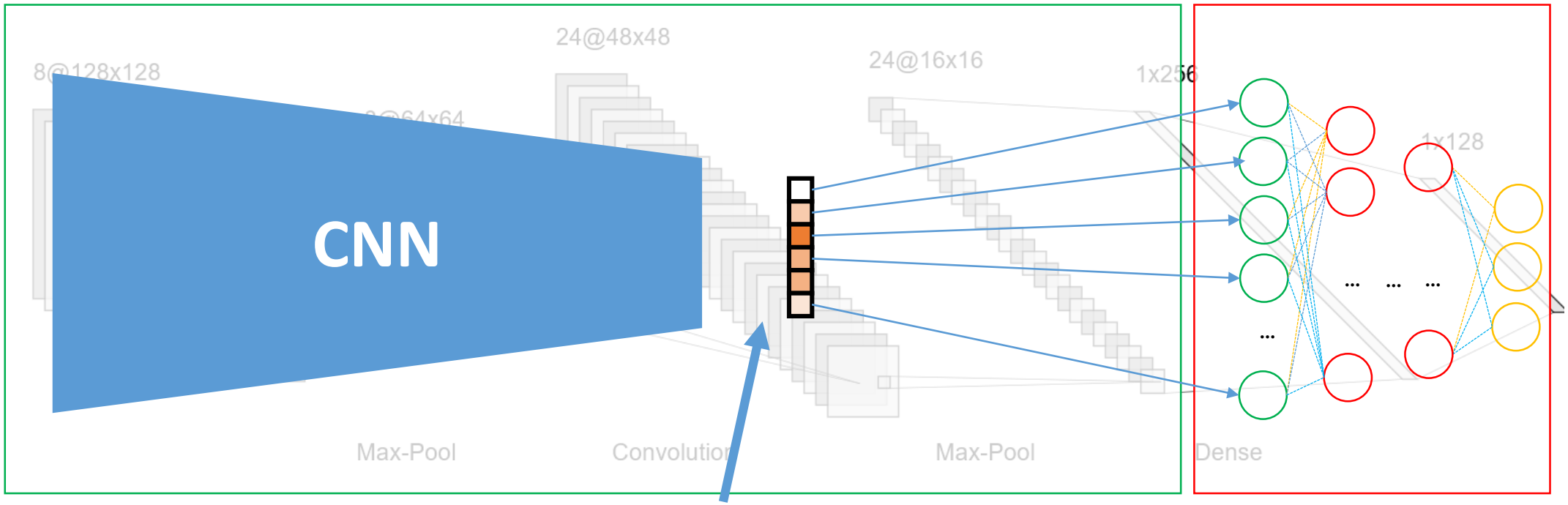
Giacomo Boracchi

The typical architecture of a CNN

Convolutional Layers

Extract high-level features from pixels

Classify



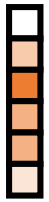
Latent Representation:
Data-Driven Feature Vector

MLP for feature classification

Very Good Data-driven Features

Feed a test image and compute its latent representation

Test image



Latent Representation:
Data-Driven Feature Vector

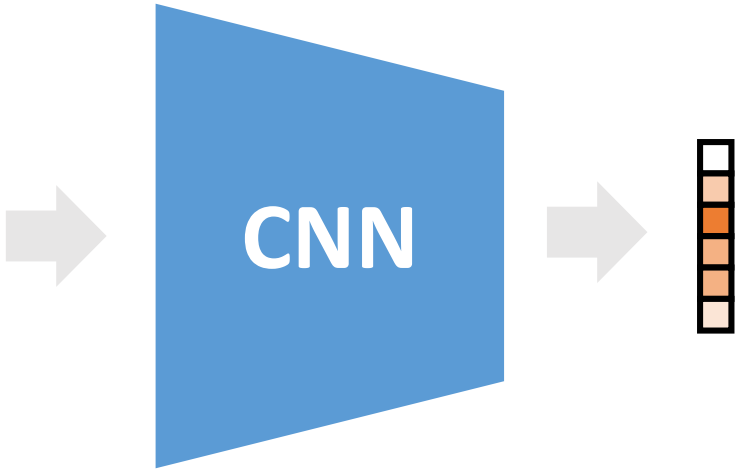
Retrieve the training images having the closest latent representation

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." NIPS 2012

Very Good Features!

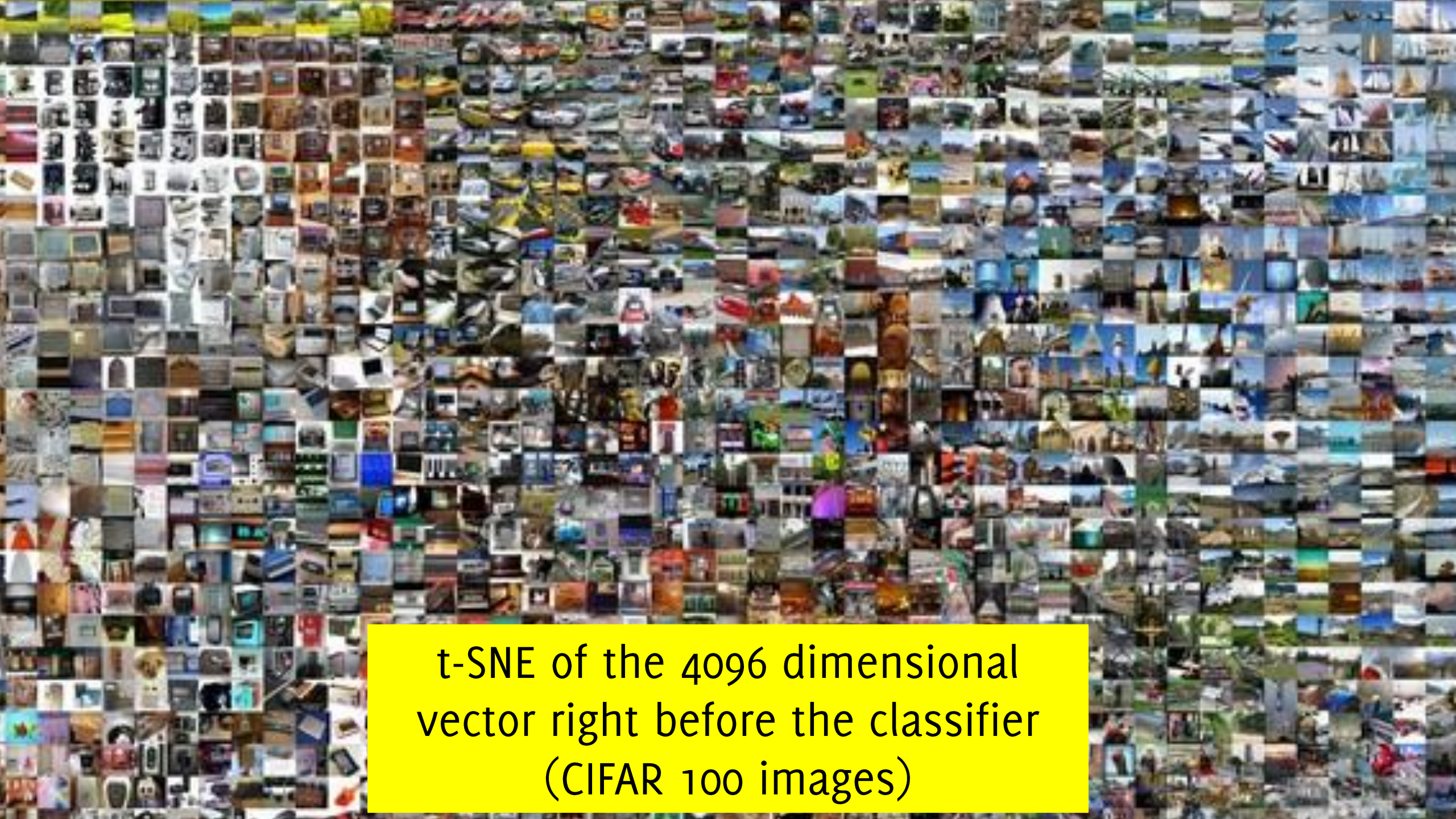
Feed a test image and compute its latent representation

Test image



Training Images corresponding to the closest latent representations!

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).


t-SNE of the 4096 dimensional
vector right before the classifier
(CIFAR 100 images)

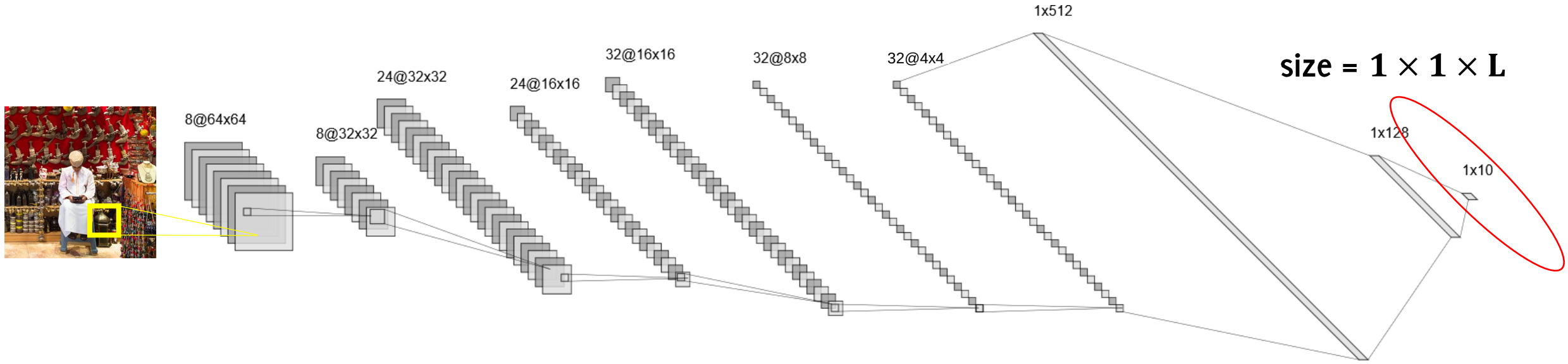
Distances in the latent representation of a CNN are meaningful for image semantics, in contrast with the Euclidean Distance (remember previous visualization)

Transfer Learning:
How to use these features for different classification tasks

Giacomo Boracchi

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



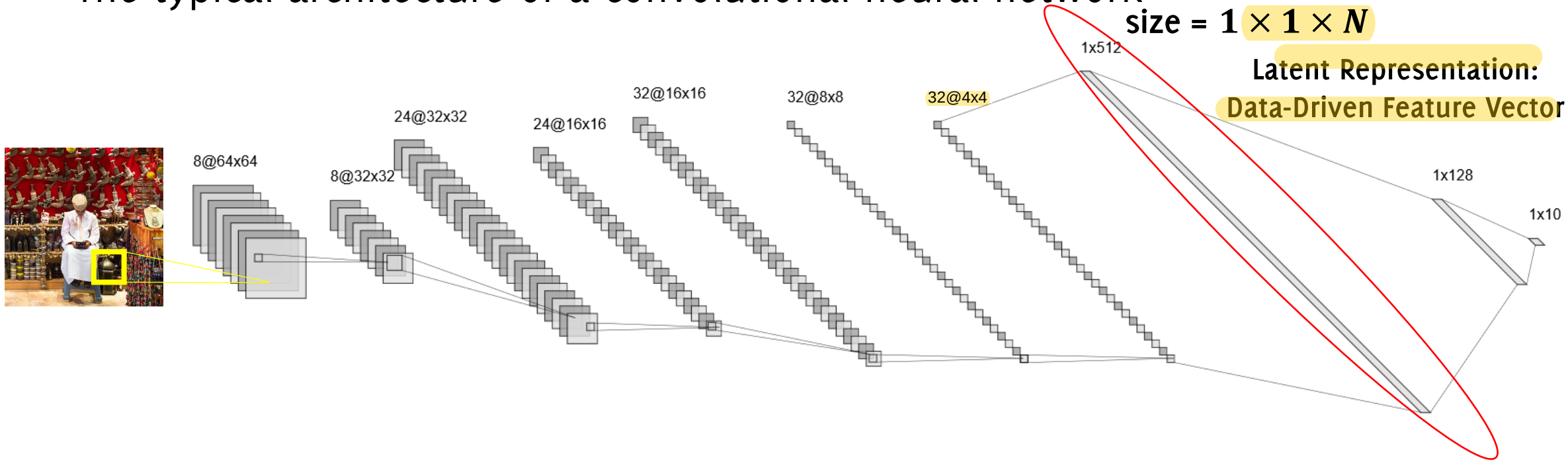
The **output of the fully connected layer** has the same size as the number of classes L, and each component provide a score for the input image to belong to a specific class.

This is **very task-specific**

By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45679374

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



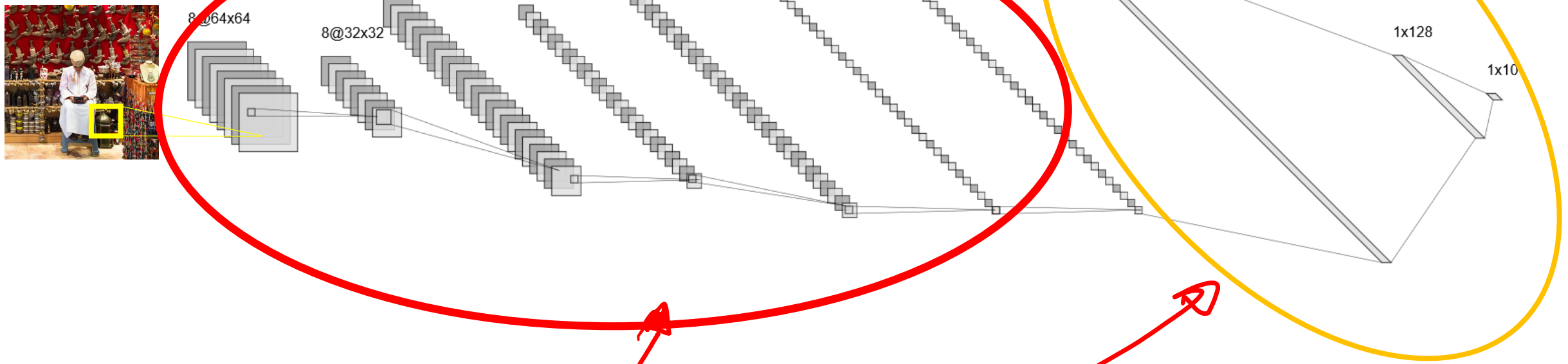
The **latent** representation, the input of the fully connected layer can be seen as a data-driven descriptor of the input image, i.e. a feature vector. These features are:

- defined to maximize classification performance
- Trained via backpropagation as the NN they are fed

By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45679374

Convolutional Neural Networks (CNN)

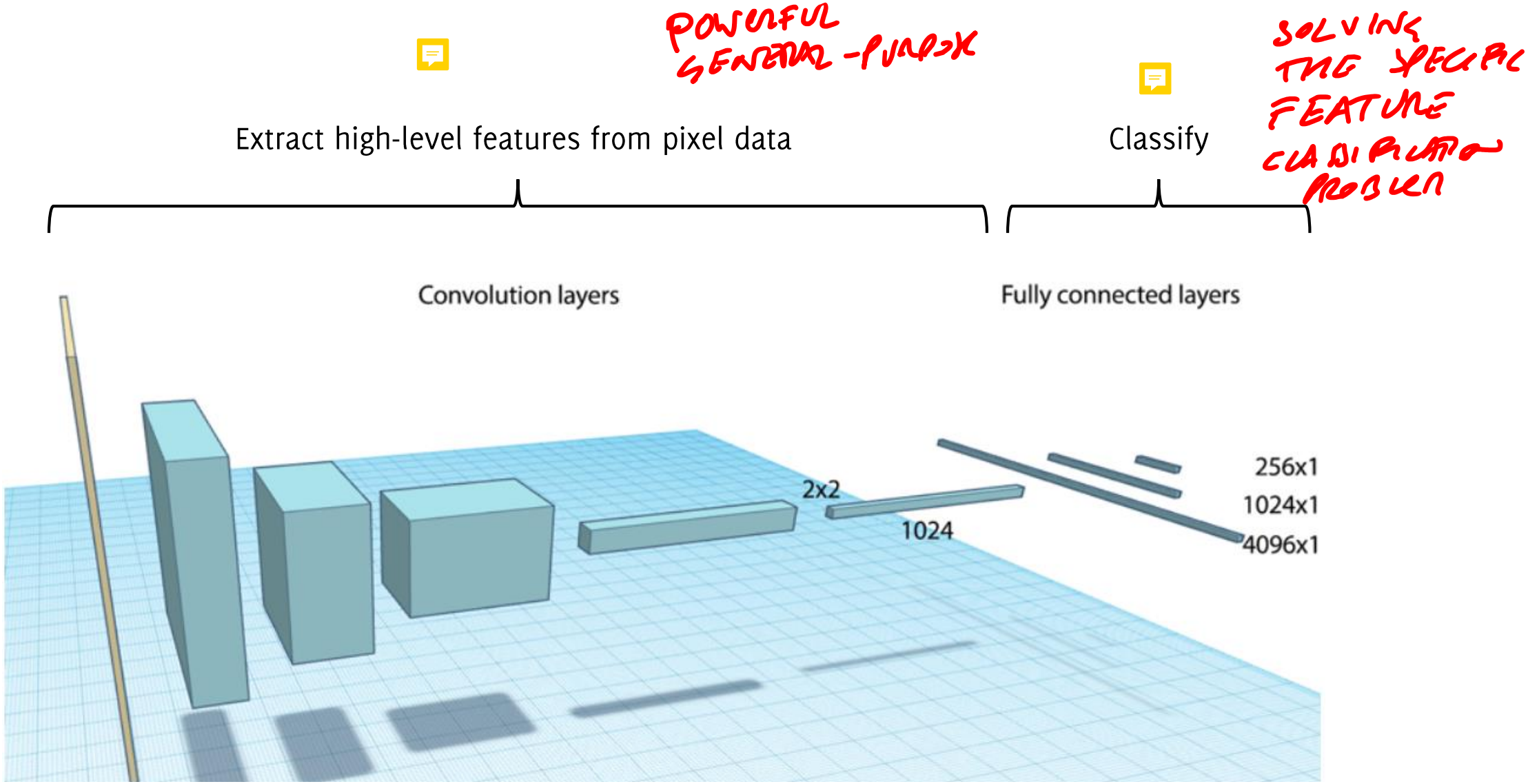
The typical architecture of a convolutional neural network



Data-driven feature extraction

Feature Classifier

Let's take a CNN trained on ImageNet

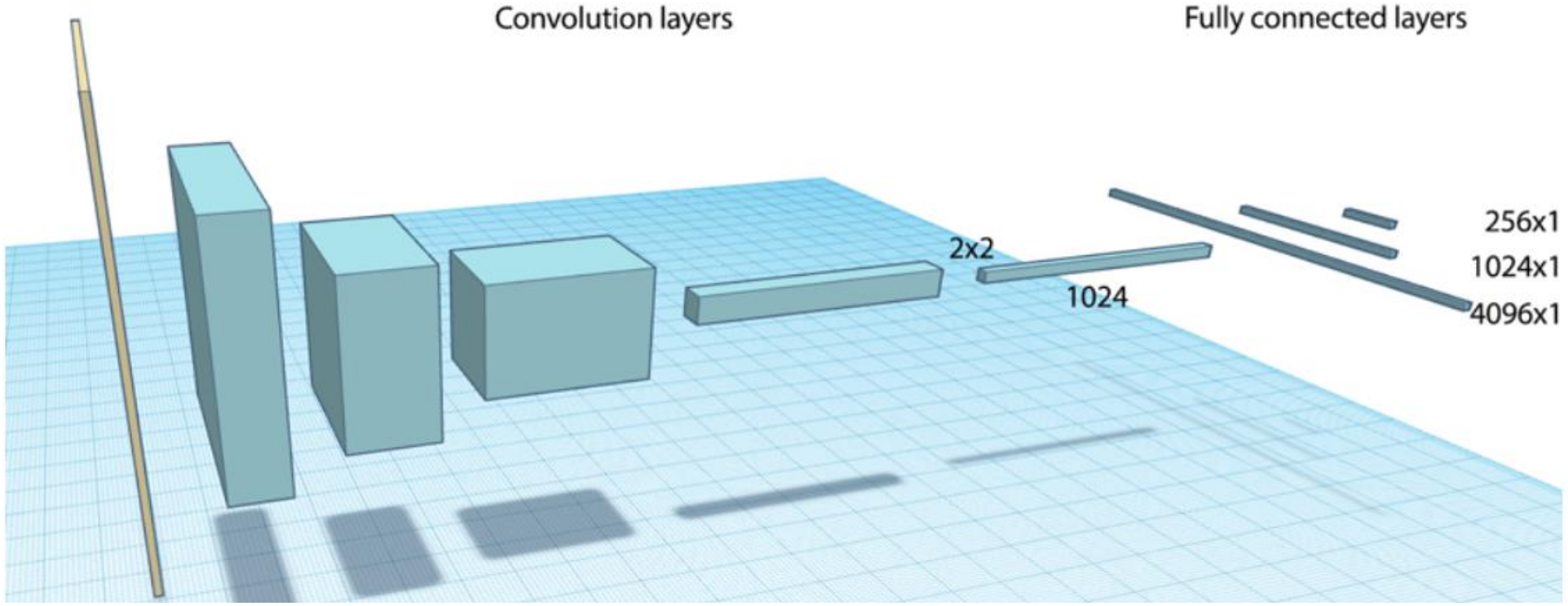


https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

A typical architecture

As we move to deeper layers:

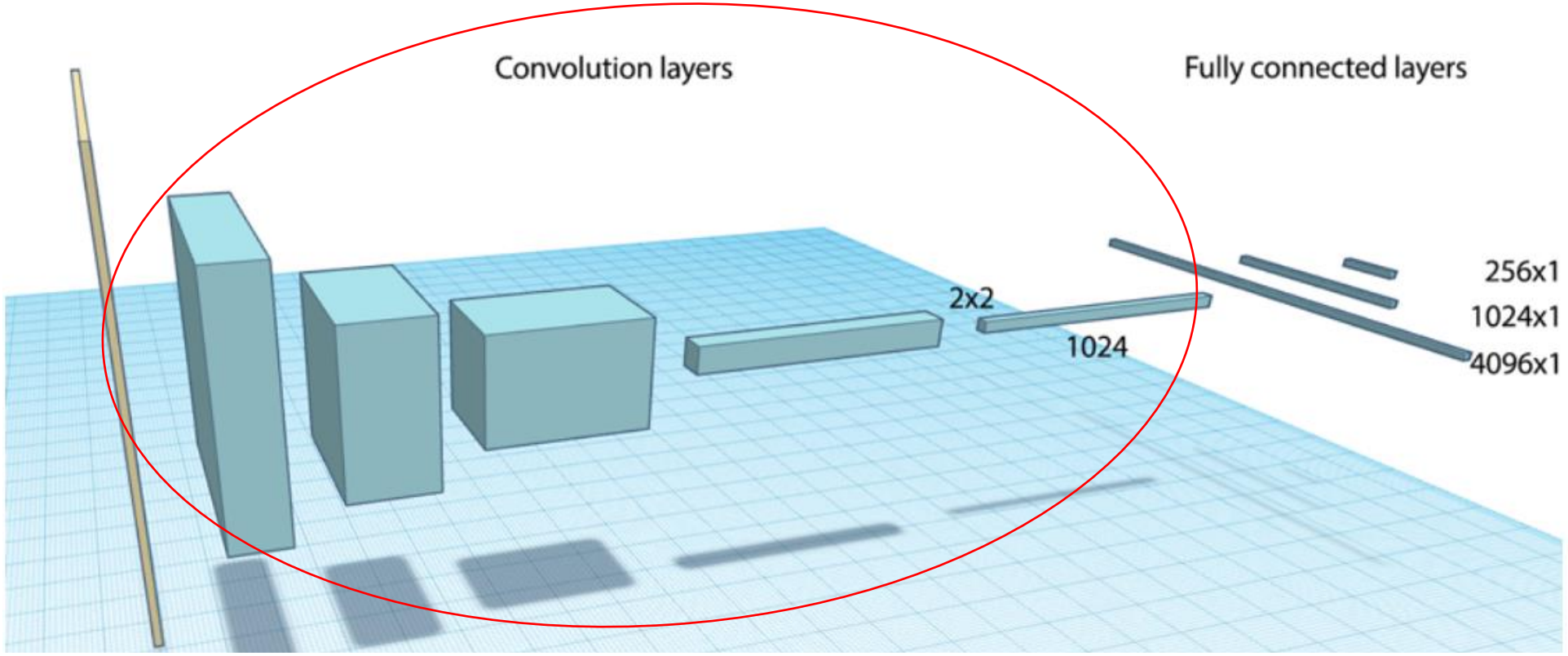
- The **spatial resolution decreases**: these layers search for higher-level patterns, and **don't care too much about their exact location.**
- the **number of maps increases**: there are **more high-level patterns** than low-level details!



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

The Feature Extraction Network

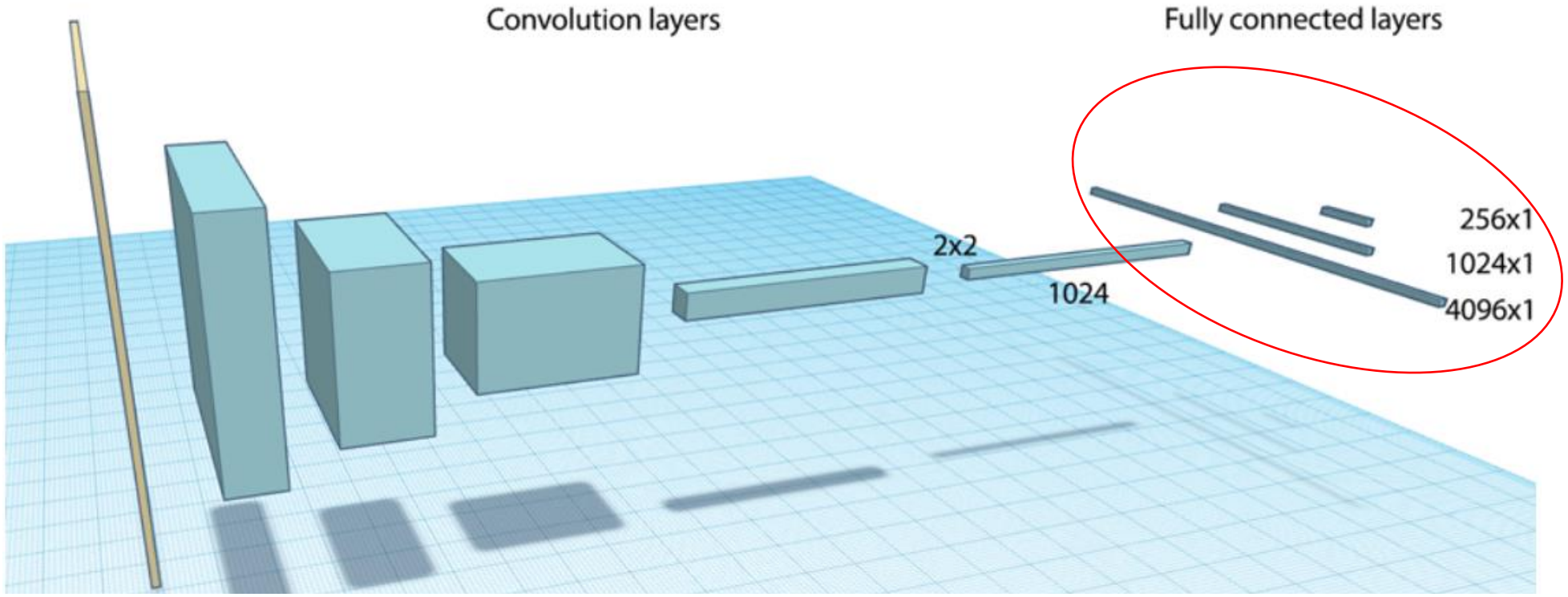
Feature Extraction Network: take the **convolutional layers** of a network trained to solve a large classification problem (e.g., 1000 classes from Imagenet)
This **should be a very effective and general feature extractor!**



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

A typical architecture: FC layers

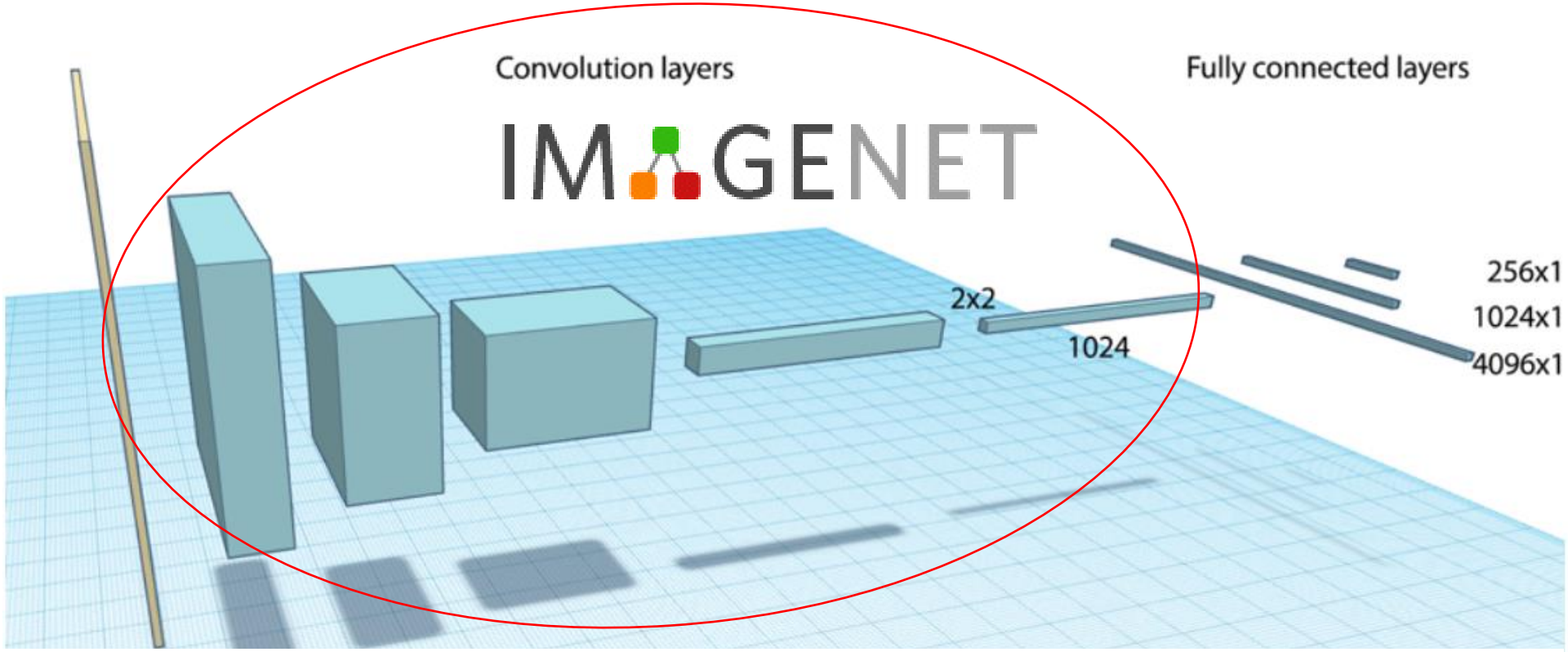
The **FC layers are instead very custom and task-specific**, as they are meant to solve the specific classification task at hand



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

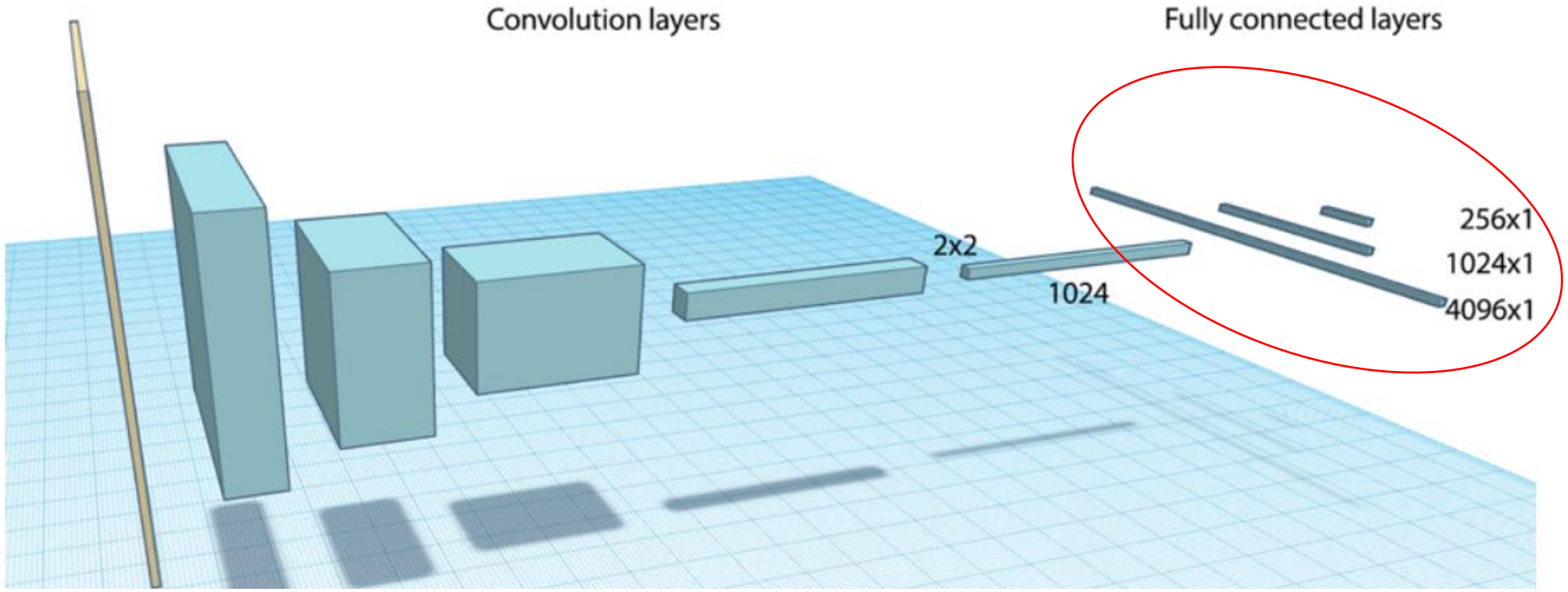
- Take a successful pre-trained model such as VGG



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

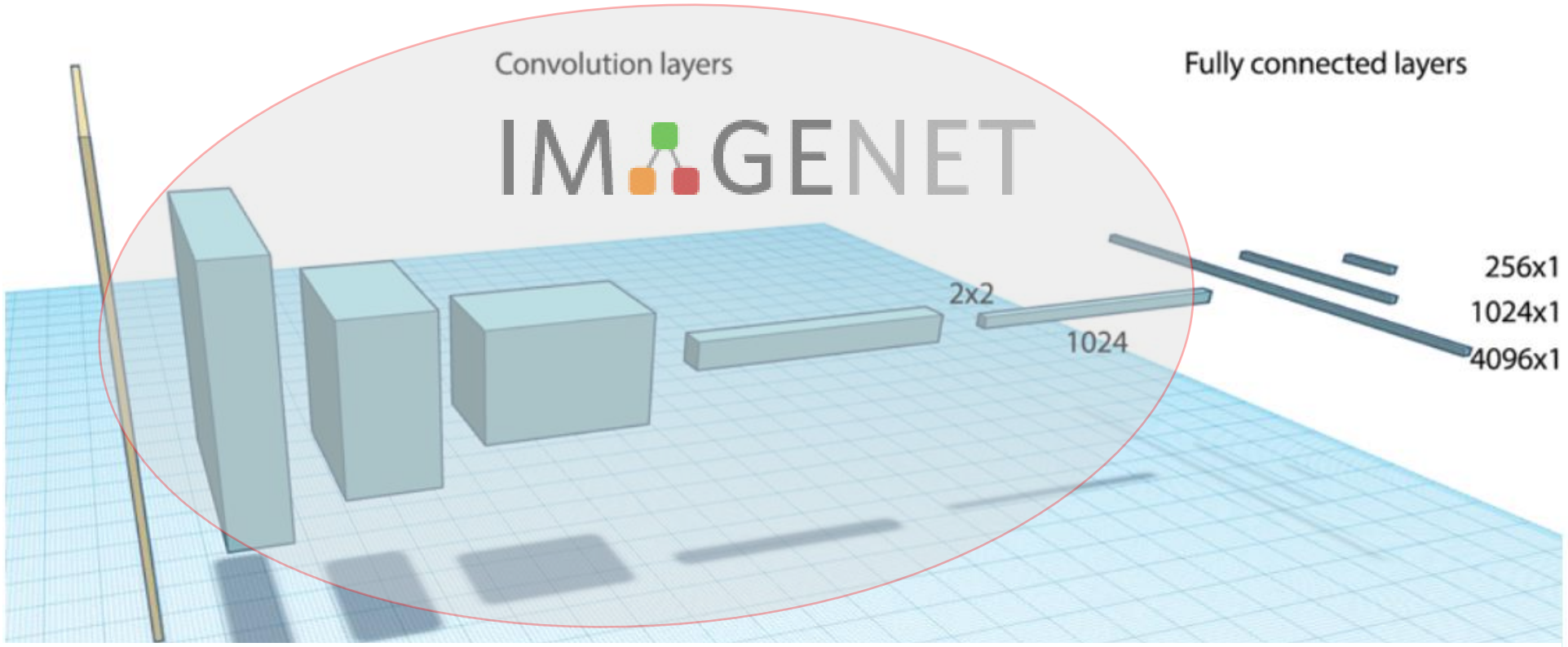
- Take a successful pre-trained model such as VGG
- Remove and the FC layers. **Design new FC layers to match the new problem.**



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

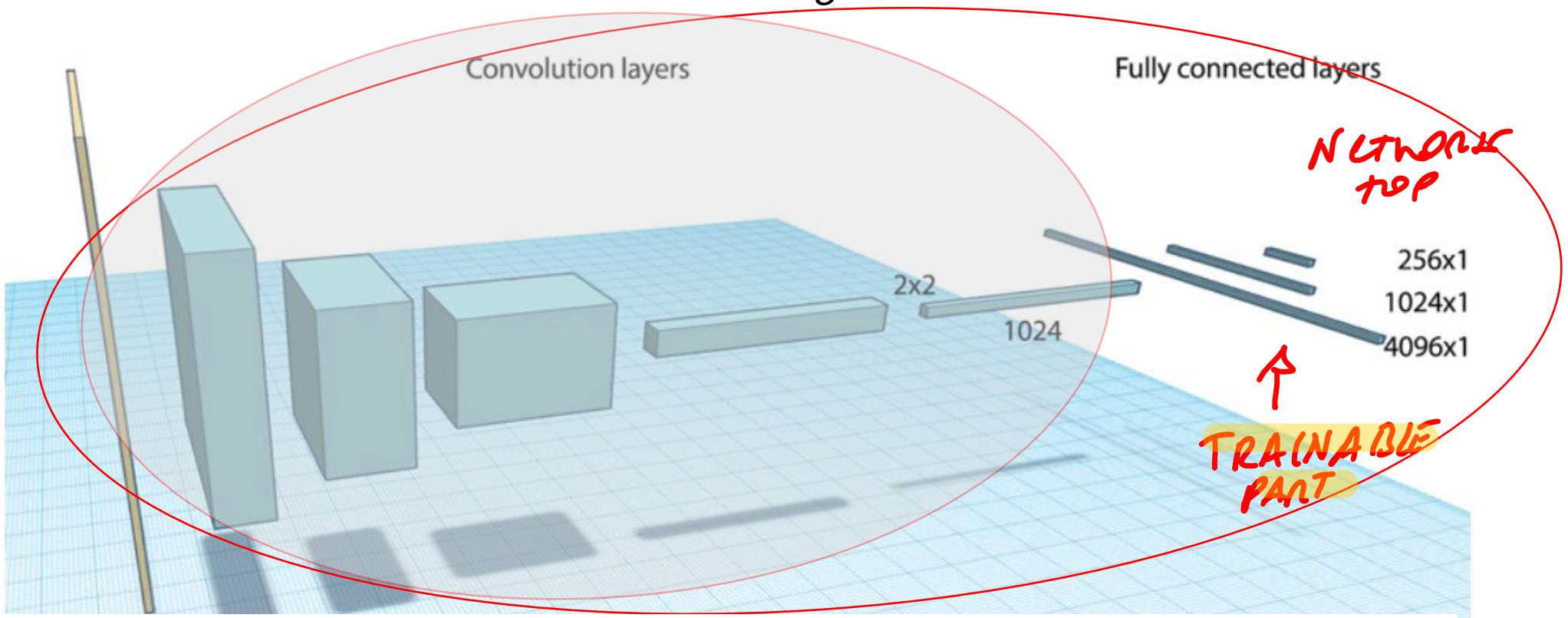
- Take a successful pre-trained model such as VGG
- Remove and the FC layers. **Design new FC layers to match the new problem.**
- «Freeze» the weights in the convolutional layers.



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

- Take a successful pre-trained model such as VGG
- Remove and the FC layers. **Design new FC layers to match the new problem.**
- «Freeze» the weights in the convolutional layers.
- Train the whole network on the new training data



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

In keras...

Pre-trained models are available, typically in two ways:

- **`include_top = True`:** provides the entire network, including the fully convolutional layers. This network can be used to solve the classification problem it was trained for
- **`include_top = False`:** contains only the convolutional layers of the network, and it is specifically meant for transfer learning.

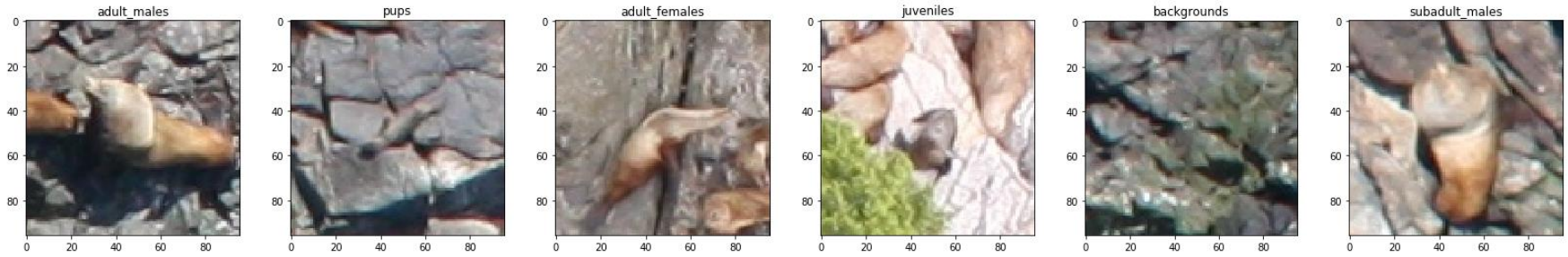
Have a look at the size of these models in the two options!

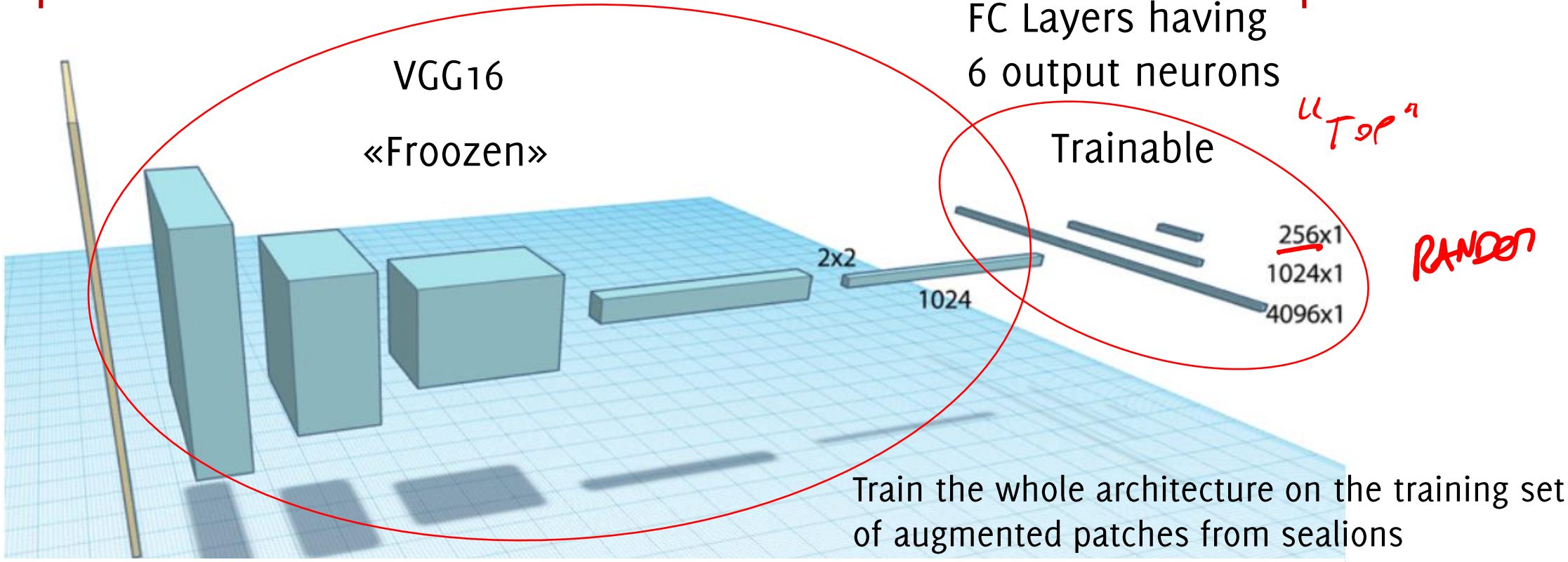
VGG16 in keras...

```
from keras import applications

base_model = applications.VGG16(weights =
"imagenet", include_top=False, input_shape =
(img_width, img_width, 3))
```

Transfer Learning in the Sealion Case





Train the whole architecture on the training set of augmented patches from sealions

https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning in Keras…

Requires a bit of TensorFlow Backend to add the modified Fully connected layer at the top of a pretrained model

Then, before training it is necessary to loop through the network layers (they are in **`model.layers`**) and then modify the trainable property

```
for layer in model.layers[: lastFrozen]:
    layer.trainable=False
```

Transfer Learning vs Fine Tuning

Different Options:

- **Transfer Learning**: **only the FC** layers are being trained. A good option when **little training data** are provided and the **pre-trained model** is expected to **match** the problem at hand
- **Fine tuning:** the whole **CNN is retrained**, but the **convolutional layers** are **initialized to the pre-trained model**. A good option when **enough training data** are provided or when the **pre-trained model is not expected to match** the problem at hand.

Typically, for the same optimizer, **lower learning rates** are used when performing fine tuning than when training from scratches