UNIVERSITY OF TECHNOLOGY
IN THE EUROPEAN CAPITAL OF CULTURE
CHEMNITZ

Neurocomputing

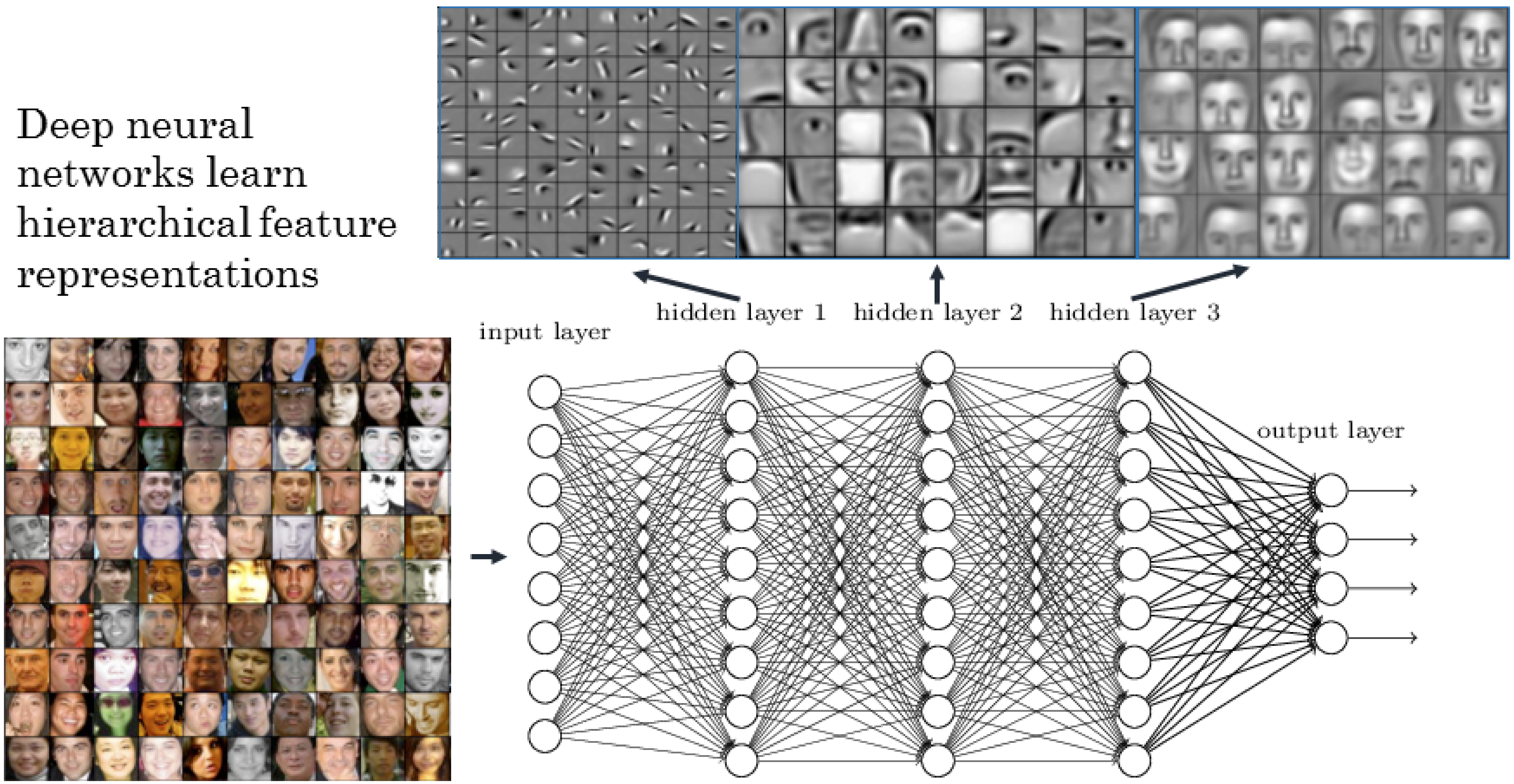
Convolutional neural networks

Julien Vitay
Professur für Künstliche Intelligenz - Fakultät für Informatik

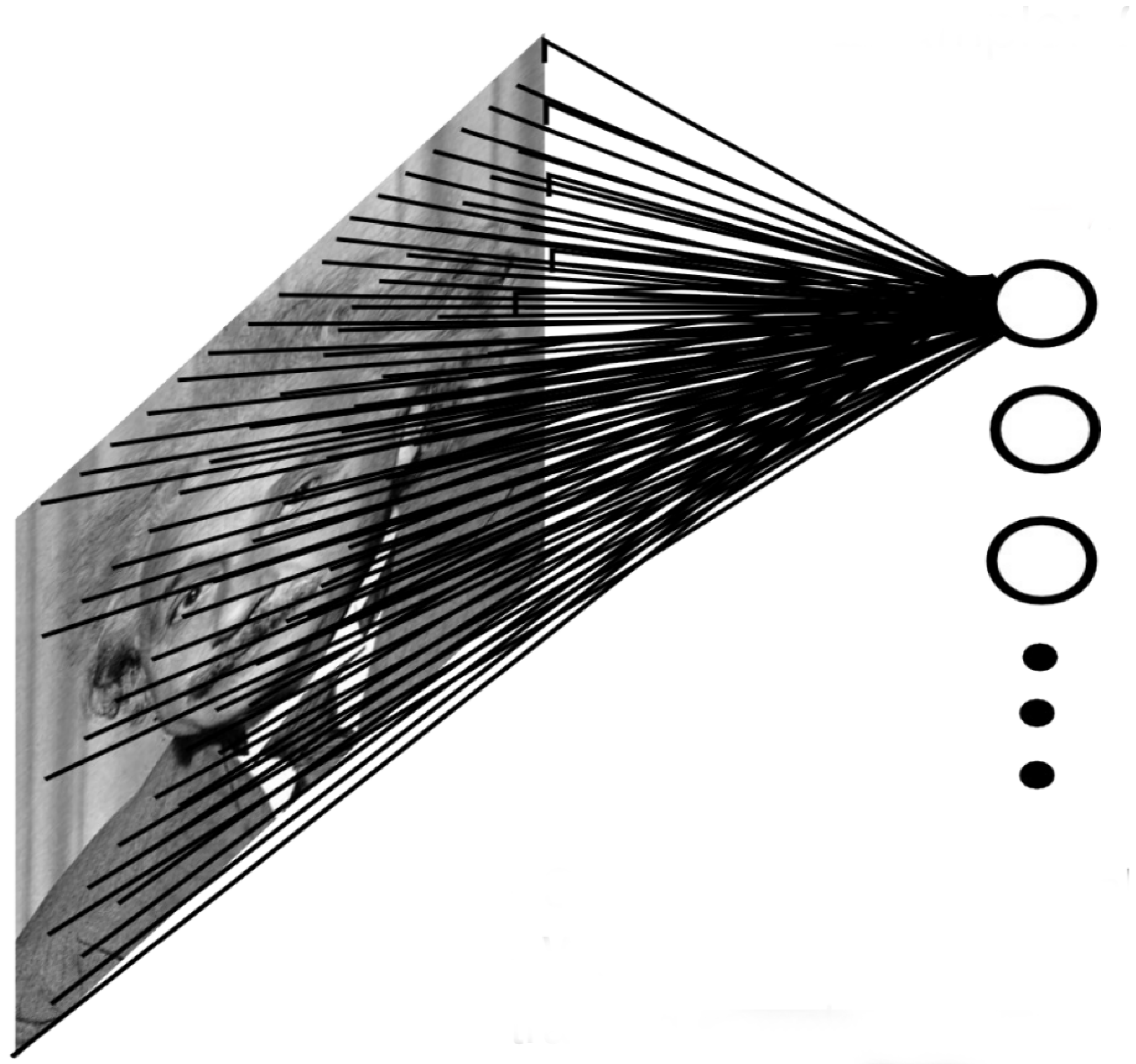
1 - Convolutional neural networks

Deep Neural Network

- The different layers of a deep network extract increasingly complex features.
  - edges $\longrightarrow$ contours $\longrightarrow$ shapes $\longrightarrow$ objects



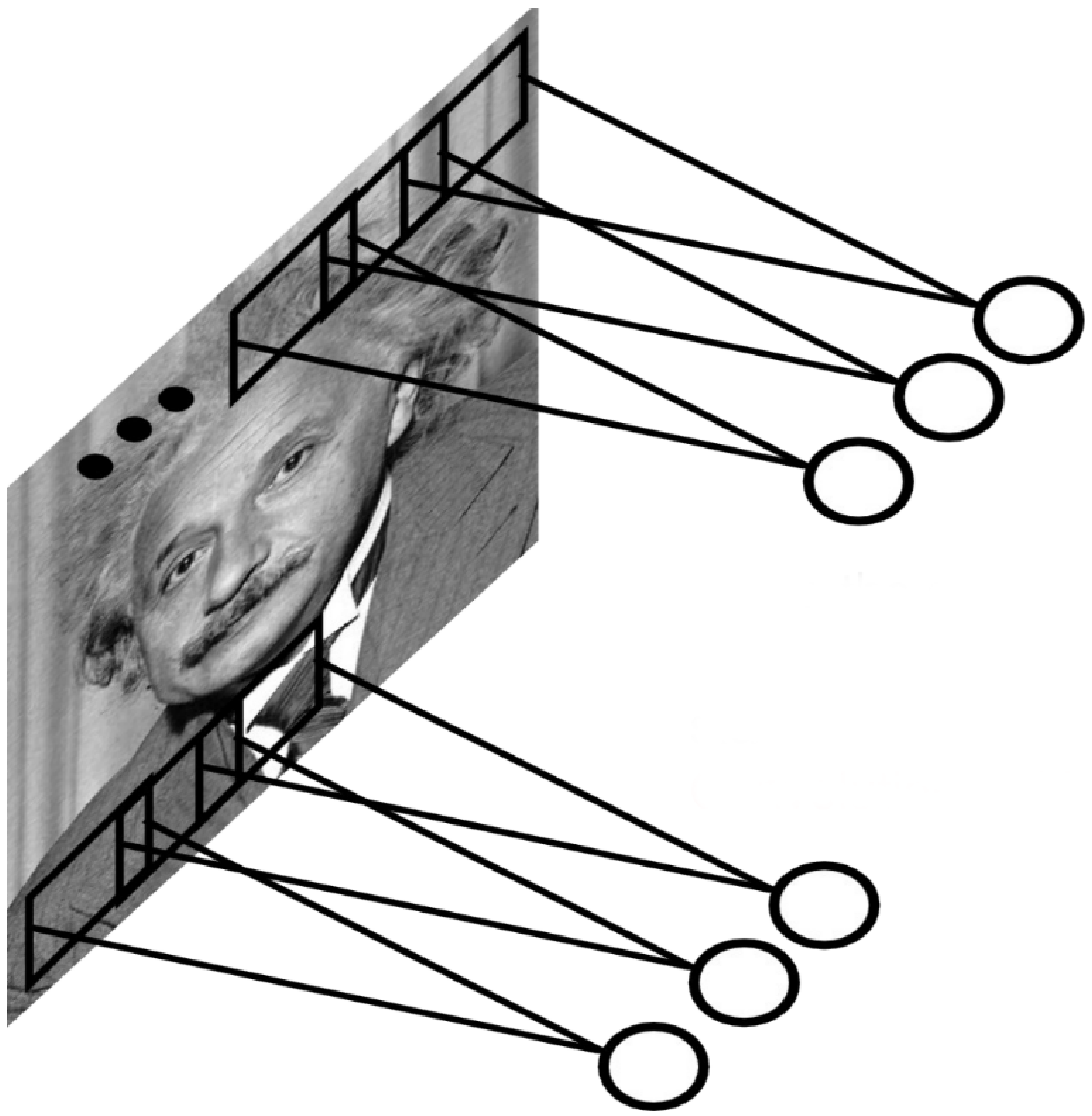
Problem with fully connected networks



- Using full images as inputs leads to an explosion of the number of weights to be learned:
  - A moderately big 800 * 600 image has 480,000 pixels with RGB values.
  - The number of dimensions of the input space is 800 * 600 * 3 = 1.44 million.
  - Even if you take only 1000 neurons in the first hidden layer, you get 1.44 **billion** weights to learn, just for the first layer.
- To obtain a generalization error in the range of 10%, you would need at least 14 billion training examples…

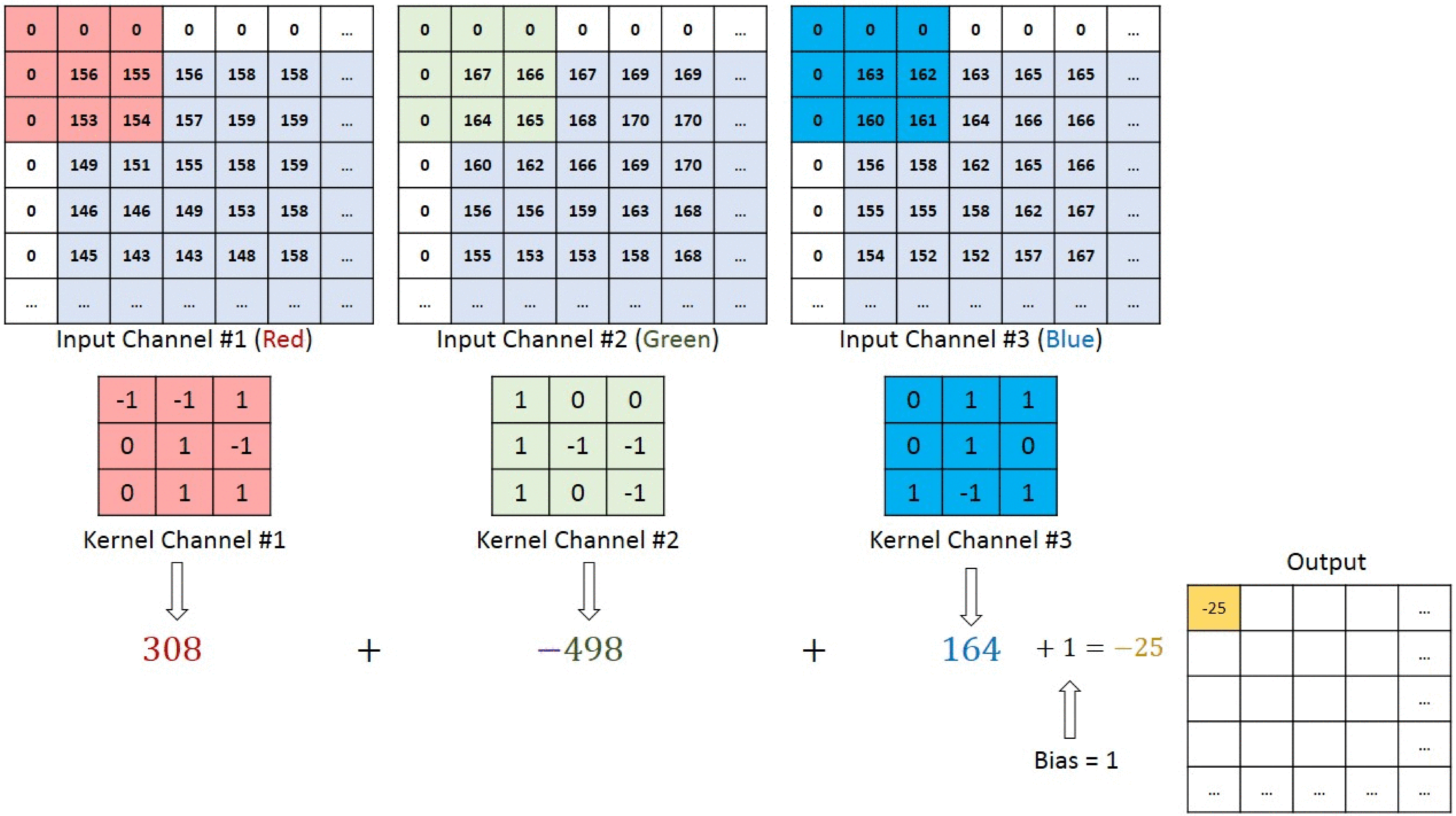
$$\epsilon \approx \frac{\mathrm{VC}_{\mathrm{dim}}}{N}$$

Problem with fully connected networks



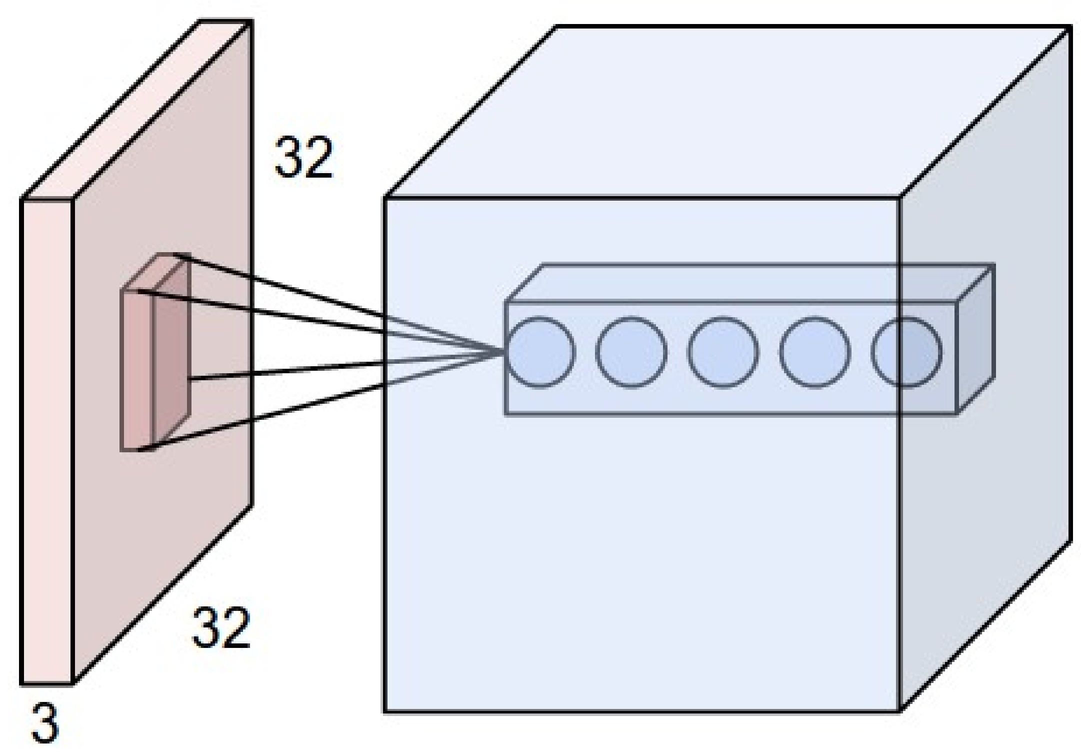
- Early features (edges) are usually local, there is no need to learn weights from the whole image.
- Natural images are stationary: the statistics of the pixel in a small patch are the same, regardless the position on the image.
- **Idea:** One only needs to extract features locally and **share the weights** between the different locations.
- This is a **convolution operation**: a filter/kernel is applied on small patches and slided over the whole image.
- Note: implemented as a cross-correlation, but it does not matter…

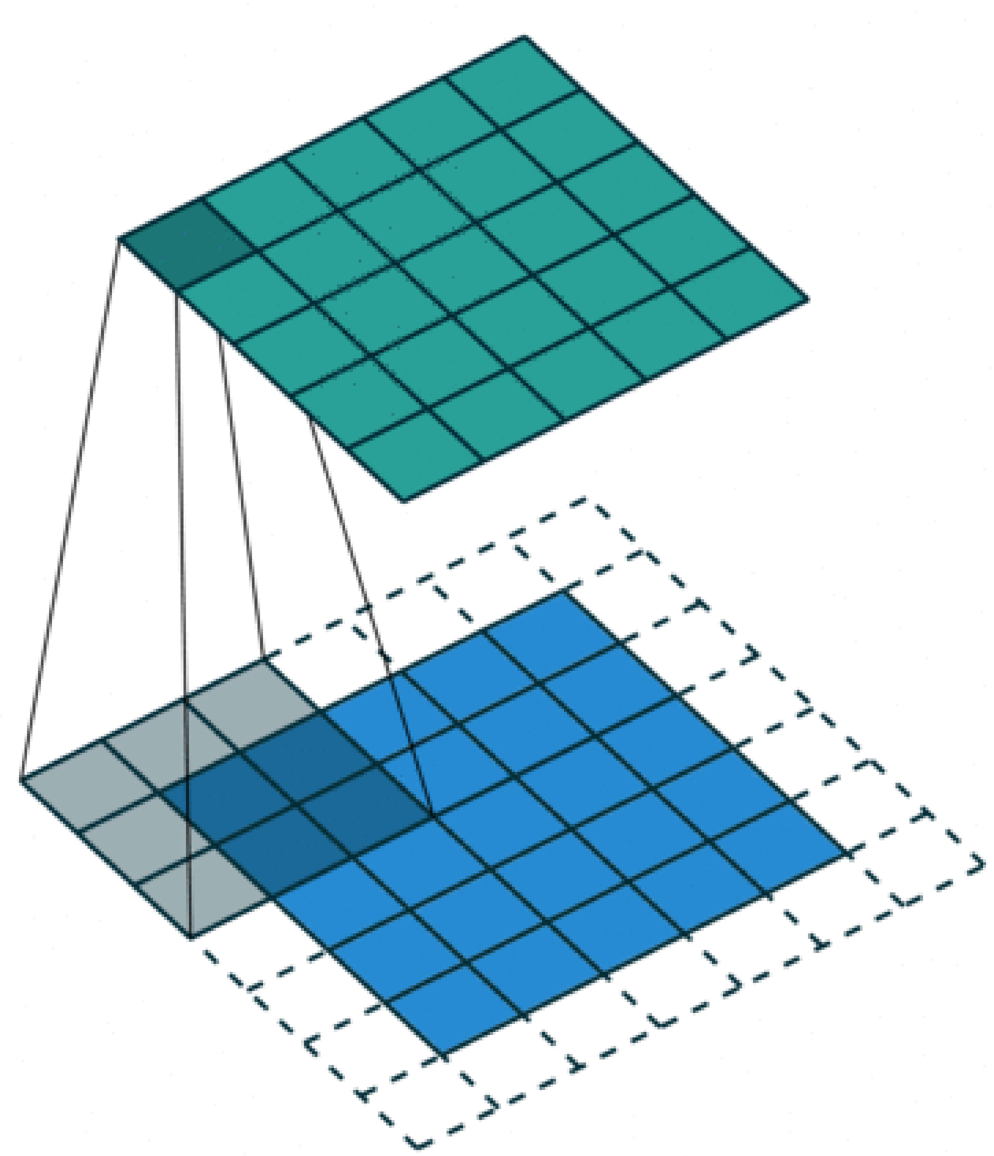
The convolutional layer



Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

The convolutional layer





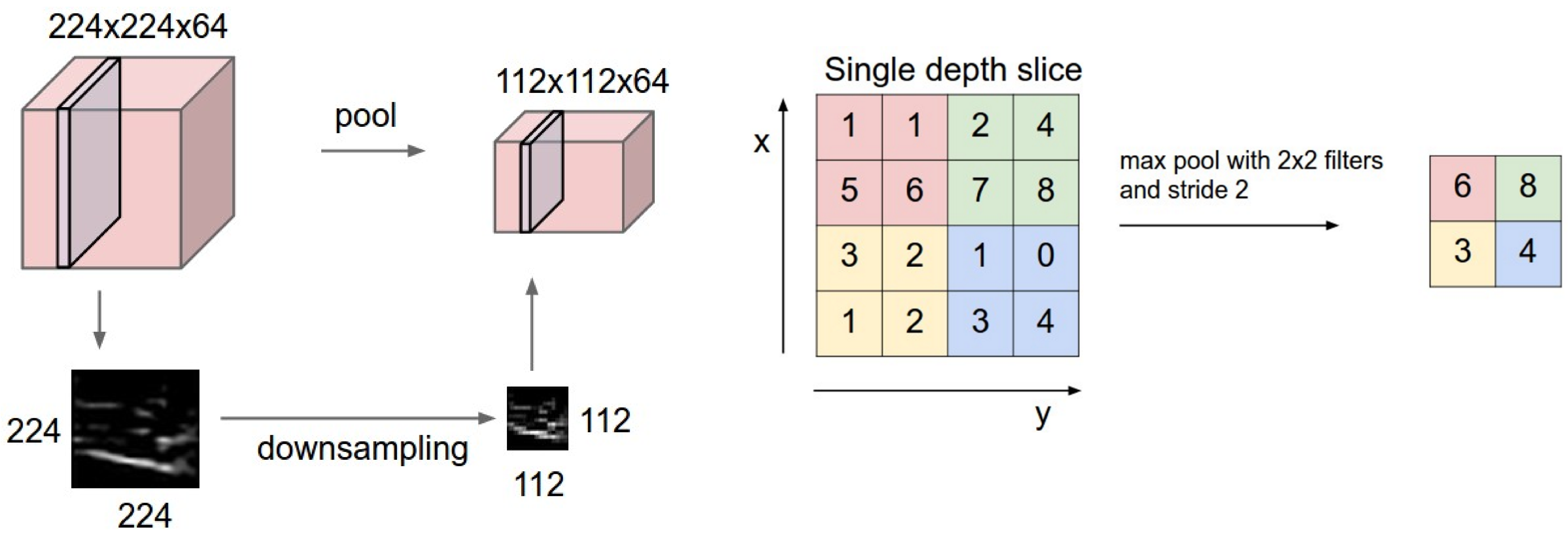
- In a **convolutional layer**, d filters are defined with very small sizes (3x3, 5x5…).
- Each filter is convoluted over the input image (or the previous layer) to create a **feature map**.
- The set of d feature maps becomes a new 3D structure: a **tensor**.

$$\mathbf{h}_k = W_k * \mathbf{h}_{k-1} + \mathbf{b}_k$$

- If the input image is 32x32x3, the resulting tensor will be 32x32xd.
- The convolutional layer has only very few parameters: each feature map has 3x3x3 values in the filter plus a bias, i.e. 28 parameters.
- As in image processing, a padding method must be chosen (what to do when a pixel is outside the image).

Source: https://github.com/vdumoulin/conv_arithmetic

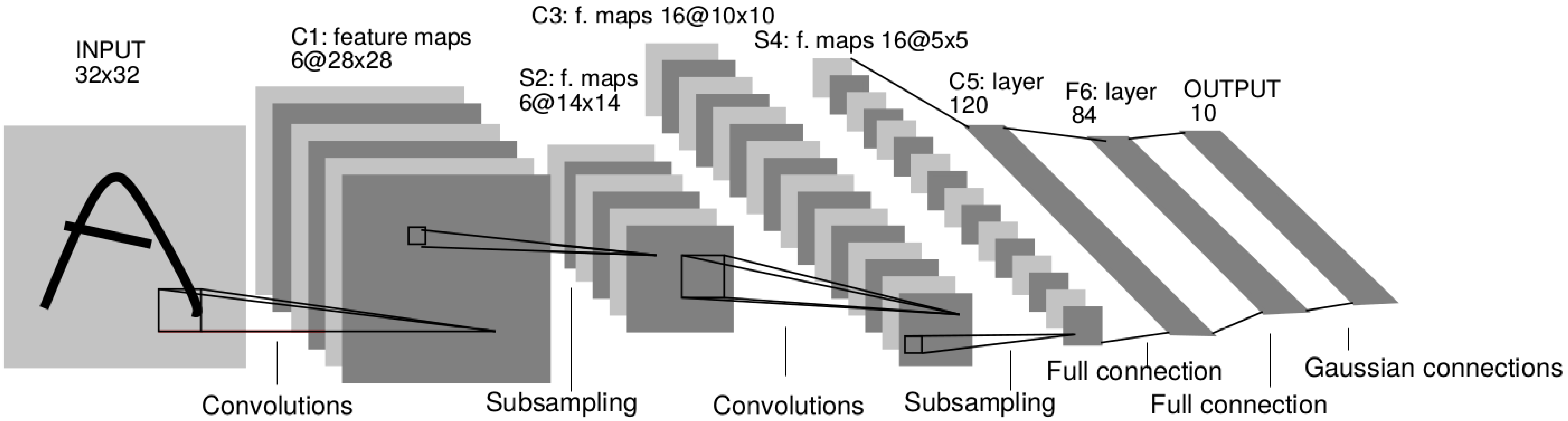
Max-pooling



Source: http://cs231n.github.io/convolutional-networks/

- The number of elements in a convolutional layer is still too high. We need to reduce the spatial dimension of a convolutional layer by **downsampling** it.
- For each feature, a **max-pooling** layer takes the maximum value of a feature for each subregion of the image (generally 2x2).
- Mean-pooling layers are also possible, but they are not used anymore.
- Pooling allows translation invariance: the same input pattern will be detected whatever its position in the input image.

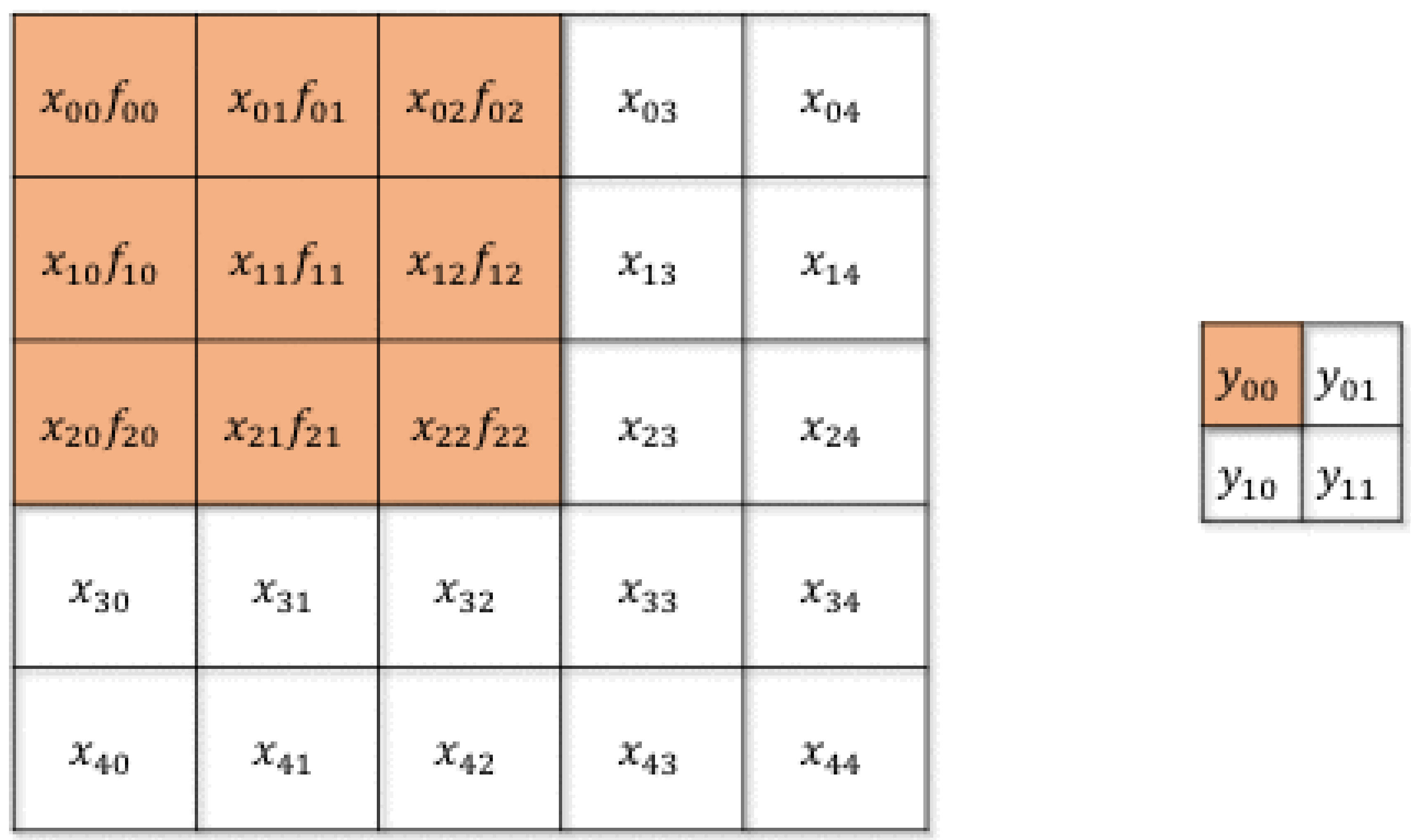
Convolutional Neural Networks



- A **convolutional neural network** (CNN) is a cascade of convolution and pooling operations, extracting layer by layer increasingly complex features.
- The spatial dimensions decrease after each pooling operation, but the number of extracted features increases after each convolution.
- One usually stops when the spatial dimensions are around 7x7.
- The last layers are fully connected (classical MLP).
- Training a CNN uses backpropagation all along: the convolution and pooling operations are differentiable.

Backpropagation through a convolutional layer

- How can we do backpropagation through a convolutional layer?



$$y_{00} = x_{00}f_{00} + x_{01}f_{01} + x_{02}f_{02} + x_{10}f_{10} + x_{11}f_{11} + x_{12}f_{12} + x_{20}f_{20} + x_{21}f_{21} + x_{22}f_{22}$$

Source: https://medium.com/@mayank.utexas/backpropagation-for-convolution-with-strides-8137e4fc2710

- In the example above, the four neurons of the feature map will receive a gradient from the upper layers.
- How can we use it to learn the filter values and pass the gradient to the lower layers?

Backpropagation through a convolutional layer

- Answer: simply by convolving the output gradients with the flipped filter!

$$\frac{\partial L}{\partial x_{00}} = \frac{\partial L}{\partial y_{00}} f_{00}$$

$\frac{\partial L}{\partial x_{00}}$	$\frac{\partial L}{\partial x_{01}}$	$\frac{\partial L}{\partial x_{02}}$	$\frac{\partial L}{\partial x_{03}}$	$\frac{\partial L}{\partial x_{04}}$
$\frac{\partial L}{\partial x_{10}}$	$\frac{\partial L}{\partial x_{11}}$	$\frac{\partial L}{\partial x_{12}}$	$\frac{\partial L}{\partial x_{13}}$	$\frac{\partial L}{\partial x_{14}}$
$\frac{\partial L}{\partial x_{20}}$	$\frac{\partial L}{\partial x_{21}}$	$\frac{\partial L}{\partial x_{22}}$	$\frac{\partial L}{\partial x_{23}}$	$\frac{\partial L}{\partial x_{24}}$
$\frac{\partial L}{\partial x_{30}}$	$\frac{\partial L}{\partial x_{31}}$	$\frac{\partial L}{\partial x_{32}}$	$\frac{\partial L}{\partial x_{33}}$	$\frac{\partial L}{\partial x_{34}}$
$\frac{\partial L}{\partial x_{40}}$	$\frac{\partial L}{\partial x_{41}}$	$\frac{\partial L}{\partial x_{42}}$	$\frac{\partial L}{\partial x_{43}}$	$\frac{\partial L}{\partial x_{44}}$

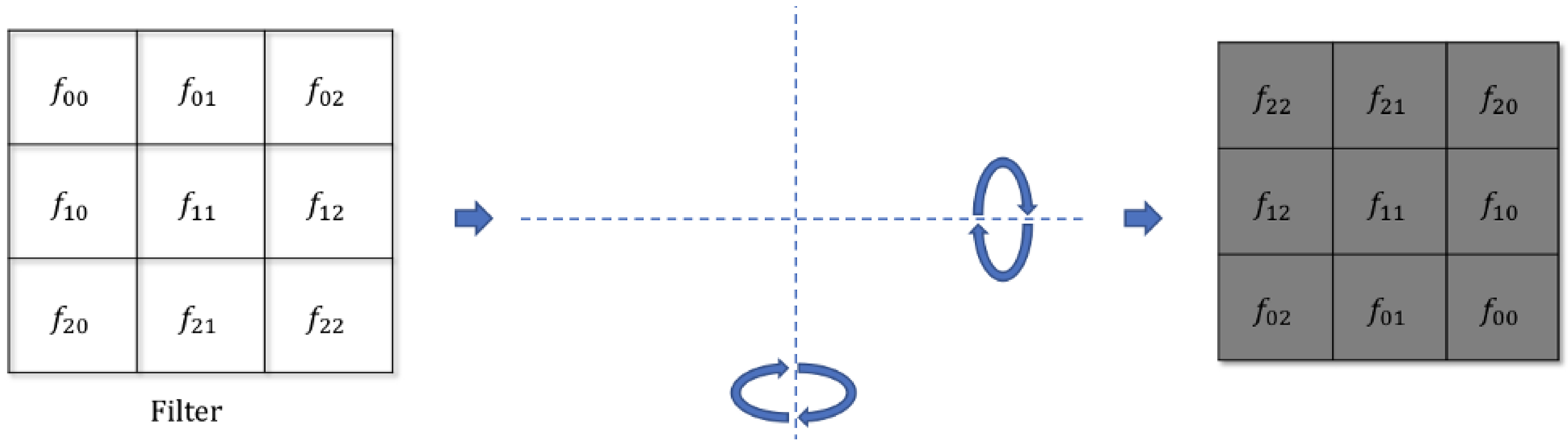
=

$0 * f_{22}$	$0 * f_{21}$	$0 * f_{20}$	0	0	0	0
$0 * f_{12}$	$0 * f_{11}$	$0 * f_{10}$	0	0	0	0
$0 * f_{02}$	$0 * f_{01}$	$\frac{\partial L}{\partial y_{00}} f_{00}$	0	$\frac{\partial L}{\partial y_{01}}$	0	0
0	0	0	0	0	0	0
0	0	$\frac{\partial L}{\partial y_{10}}$	0	$\frac{\partial L}{\partial y_{11}}$	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Source: https://medium.com/@mayank.utexas/backpropagation-for-convolution-with-strides-8137e4fc2710

Backpropagation through a convolutional layer

- The filter just has to be flipped (180^o symmetry) before the convolution.



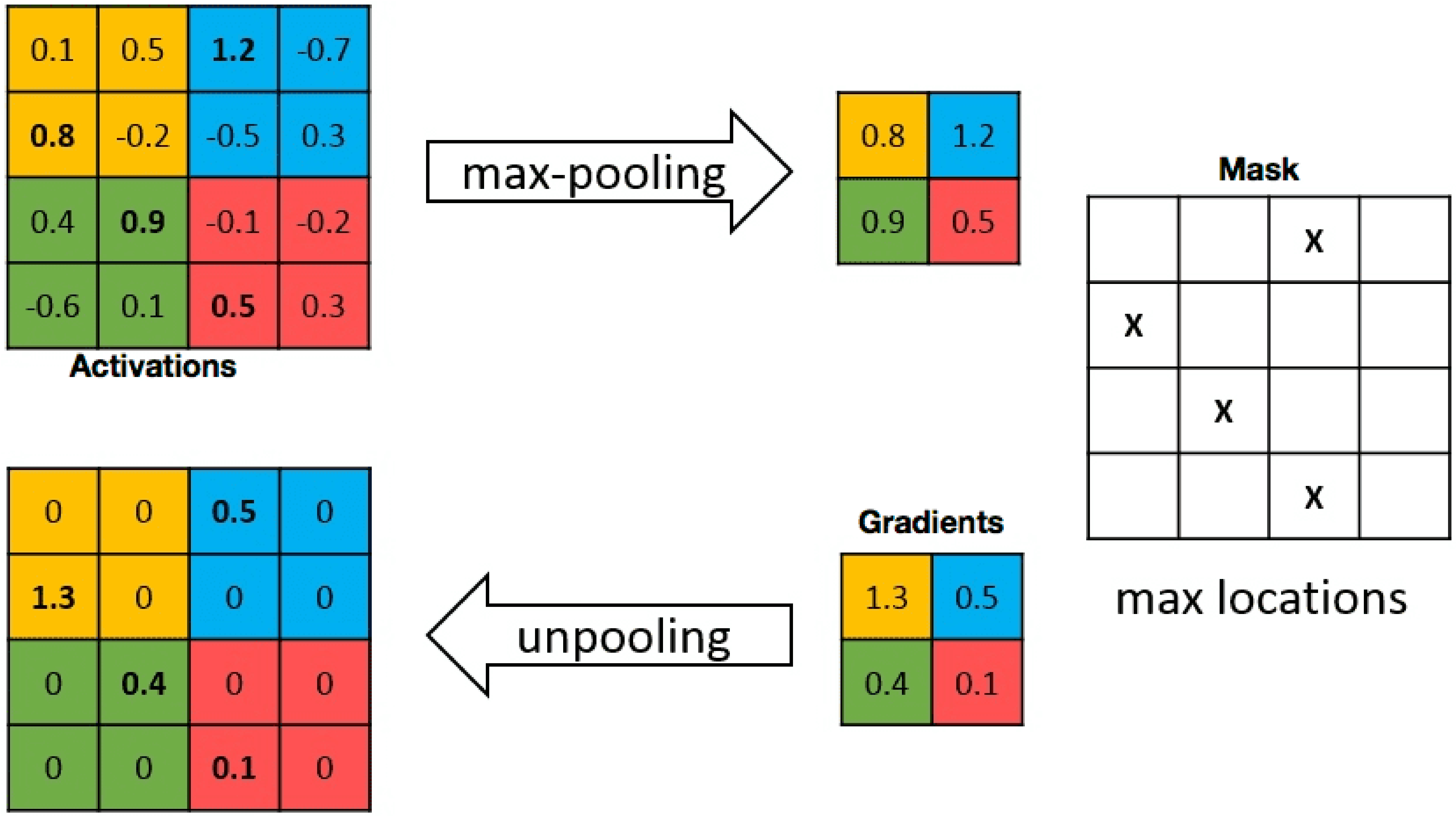
Source: https://medium.com/@mayank.utexas/backpropagation-for-convolution-with-strides-8137e4fc2710

- The convolution operation is differentiable, so we can apply backpropagation and learn the filters.

$$\mathbf{h}_k = W_k * \mathbf{h}_{k-1} + \mathbf{b}_k$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h}_{k-1}} = W_k^F * \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{h}_k}$$

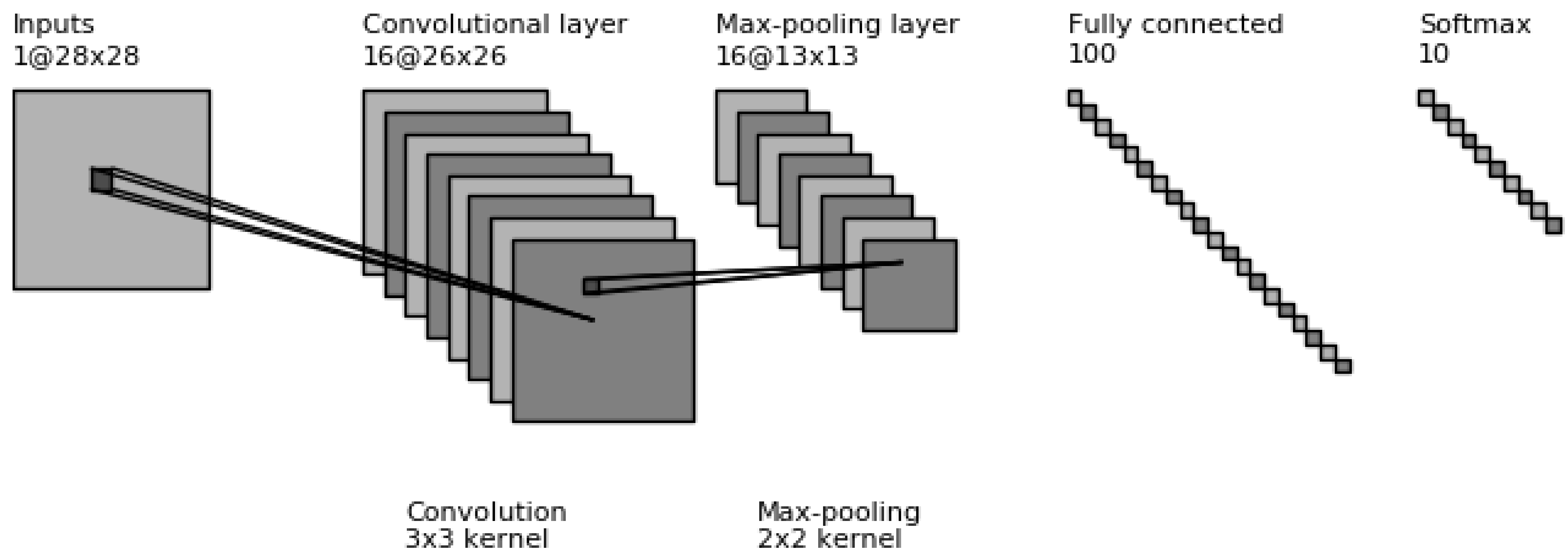
Backpropagation through a max-pooling layer



Source: https://mukulrathi.com/demystifying-deep-learning/conv-net-backpropagation-maths-intuition-derivation/

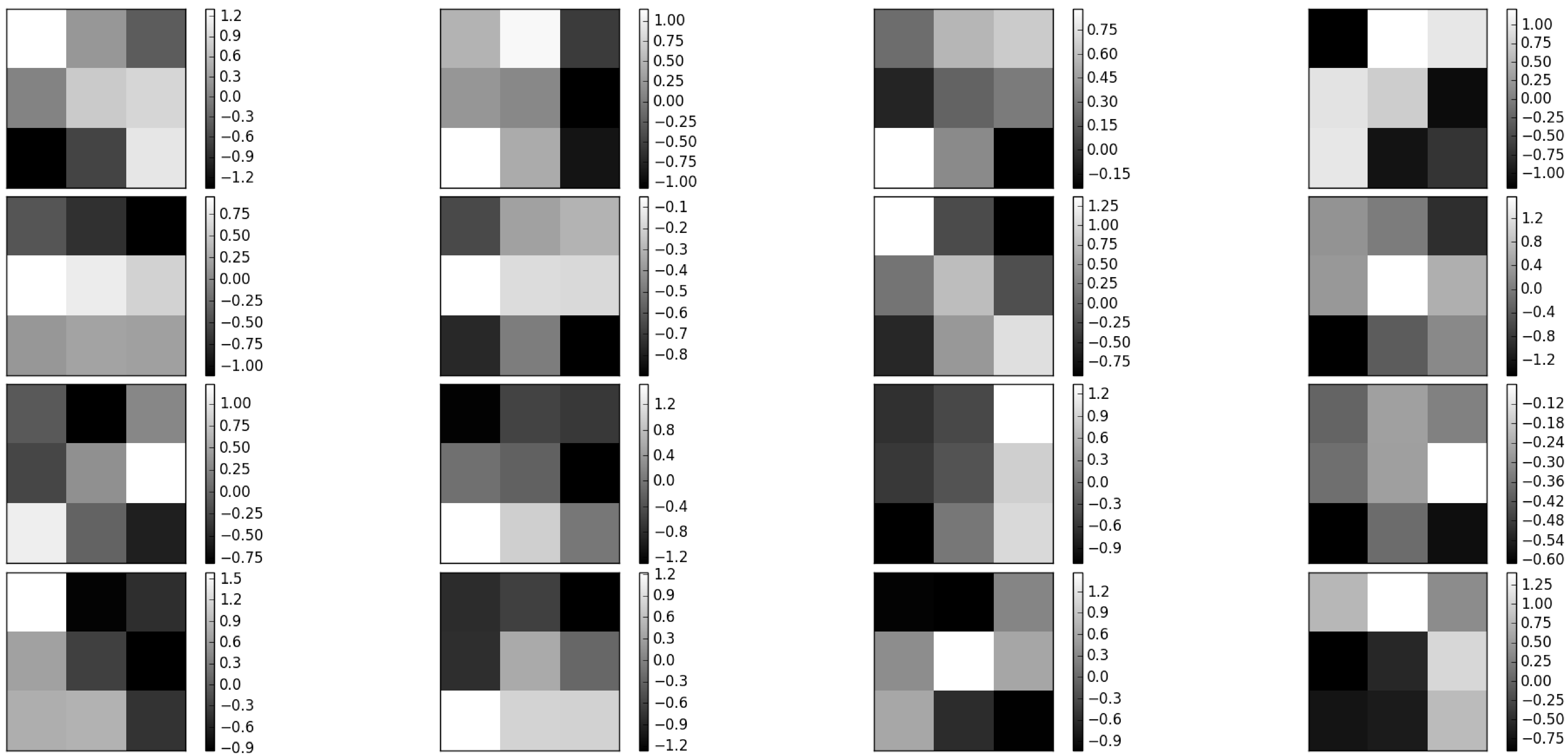
- We can also use backpropagation through a max-pooling layer.
- We need to remember which location was the winning location in order to backpropagate the gradient.
- A max-pooling layer has no parameter, we do not need to learn anything, just to pass the gradient backwards.

Convolutional layer on MNIST



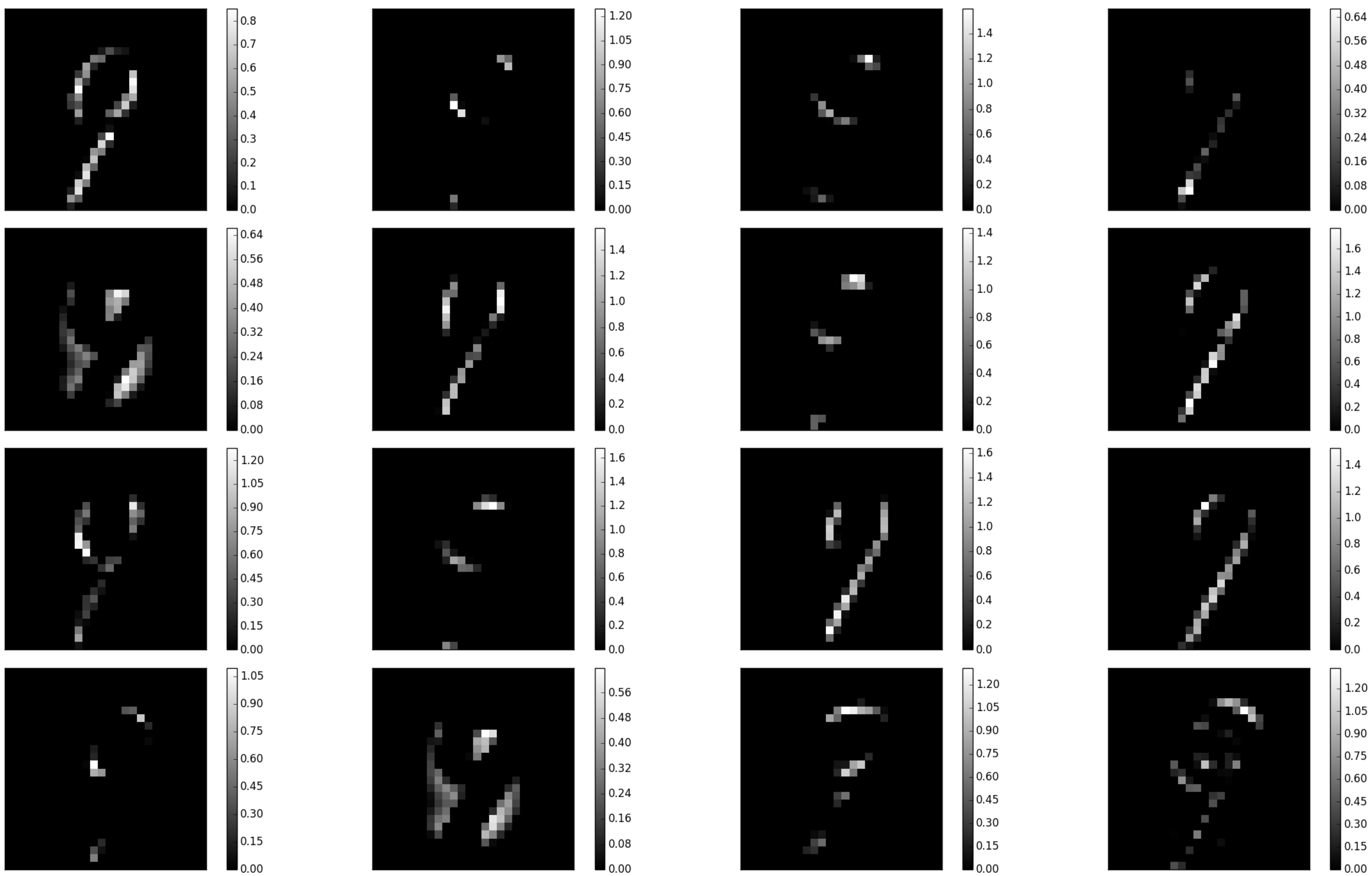
Convolutional layer on MNIST

- Each feature map extracts **edges** of different orientations.
- Here are the weights learned in the convolutional layer:

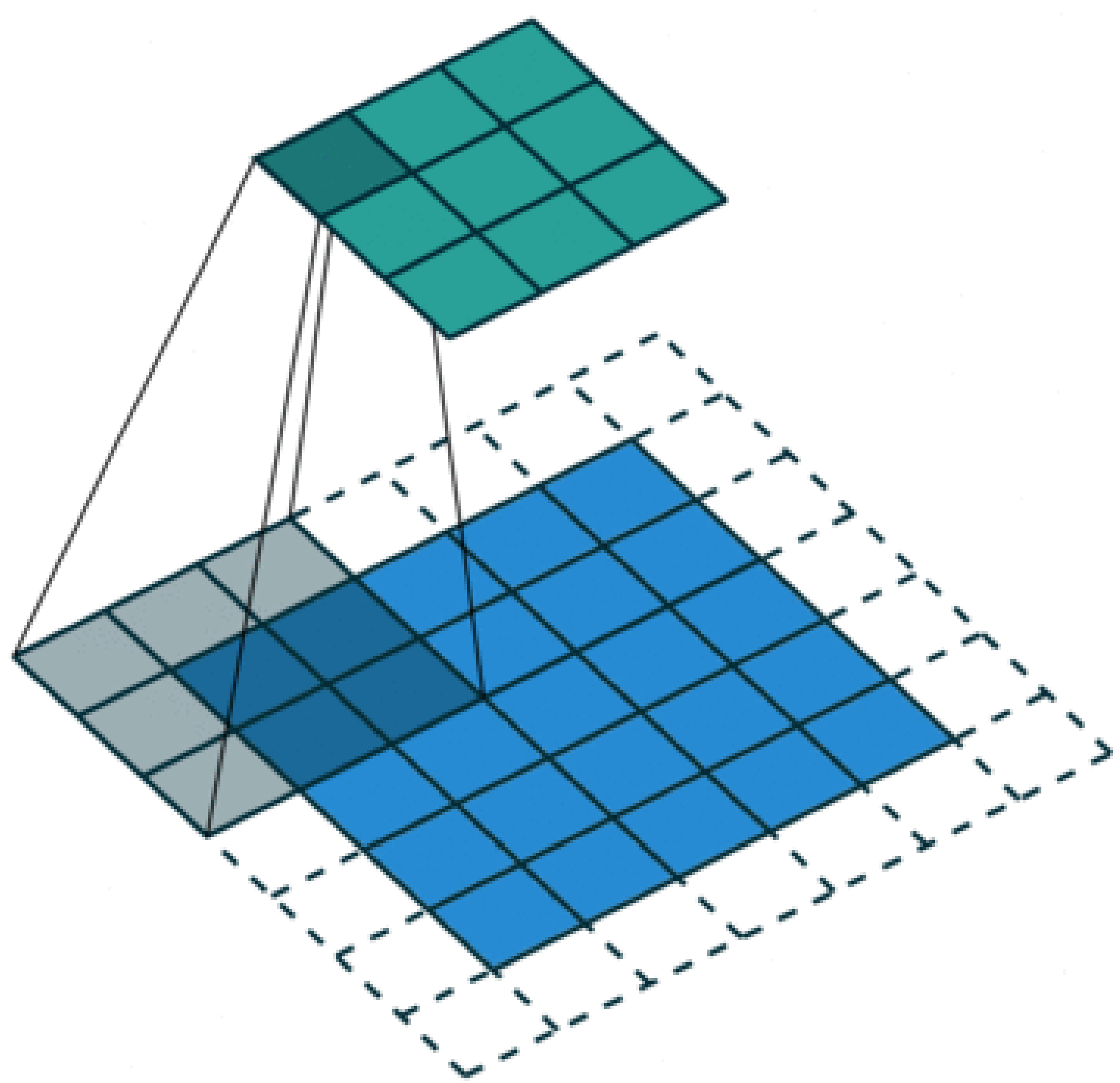


Convolutional layer on MNIST

- A convolutional layer is like a bank of (adaptive) filters applied on the image.
- **Feature maps** are the results of the convolution of these weights with the input image:



Convolution with strides

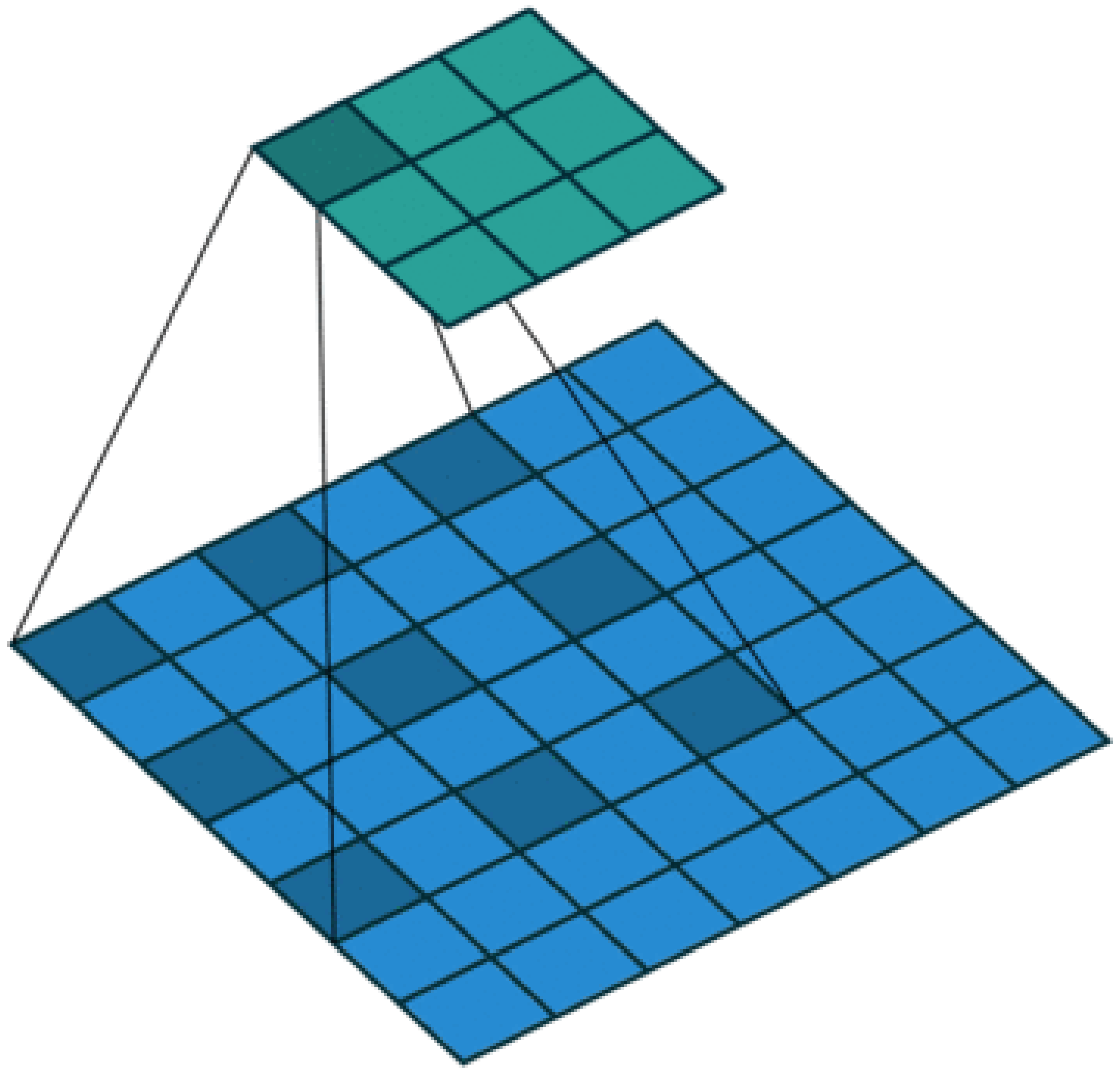


Source: https://github.com/vdumoulin/conv_arithmetic

- Convolution with strides is an alternative to max-pooling layers.
- The convolution simply “jumps” one pixel when sliding over the image (stride 2).
- This results in a smaller feature map.
- Much less operations to do than convolution with stride 1 followed by max-pooling, for the same performance.
- Particularly useful for generative models (VAE, GAN, etc).

Dilated convolutions

- A **dilated convolution** is a convolution with holes (à trous).
- The filter has a bigger spatial extent than its number of values.



Source: https://github.com/vdumoulin/conv_arithmetic

Implementing a CNN in keras

- Convolutional and max-pooling layers are regular objects in keras/tensorflow/pytorch/etc.
- You do not need to care about their implementation, they are designed to run fast on GPUs.
- You have to apply to the CNN all the usual tricks: optimizers, dropout, batch normalization, etc.

```
model = Sequential()
model.add(Input(X_train.shape[1:]))

model.add(Conv2D(32, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))L
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

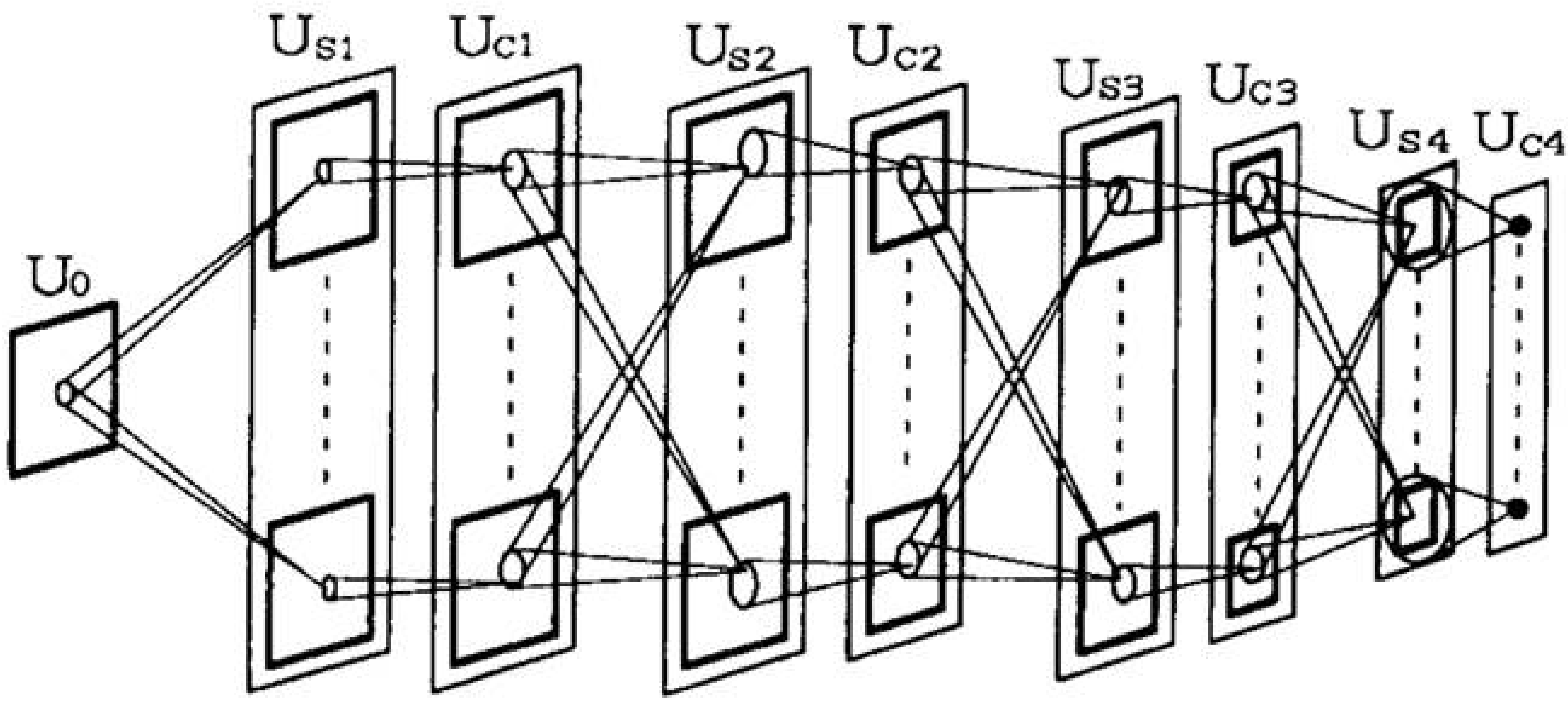
```
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))

opt = RMSprop(
    lr=0.0001,
    decay=1e-6
)

model.compile(
    loss='categorical_crossentropy',
    optimizer=opt,
    metrics=['accuracy']
)
```

2 - Some famous convolutional networks

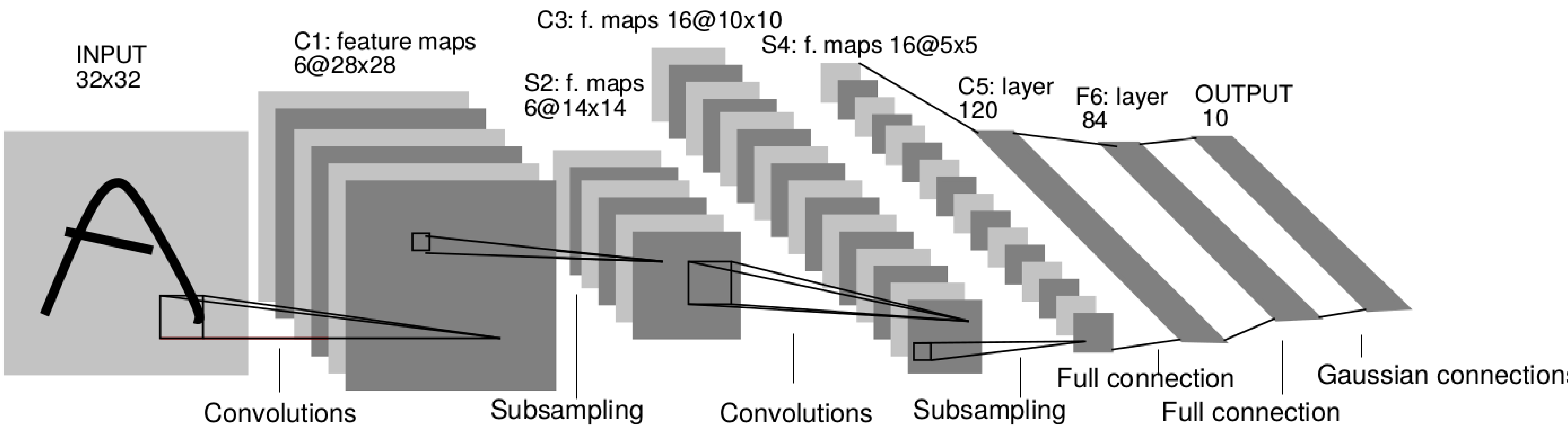
NeoCognitron



Source: https://uplLoad.wikimedia.org/wikipedia/uk/4/42/Neocognitron.jpg

- The **Neocognitron** (Fukushima, 1980) was actually the first CNN able to recognize handwritten digits.
- Training is not based on backpropagation, but a set of biologically realistic learning rules (Add-if-silent, margined WTA).
- Inspired by the human visual system.

LeNet

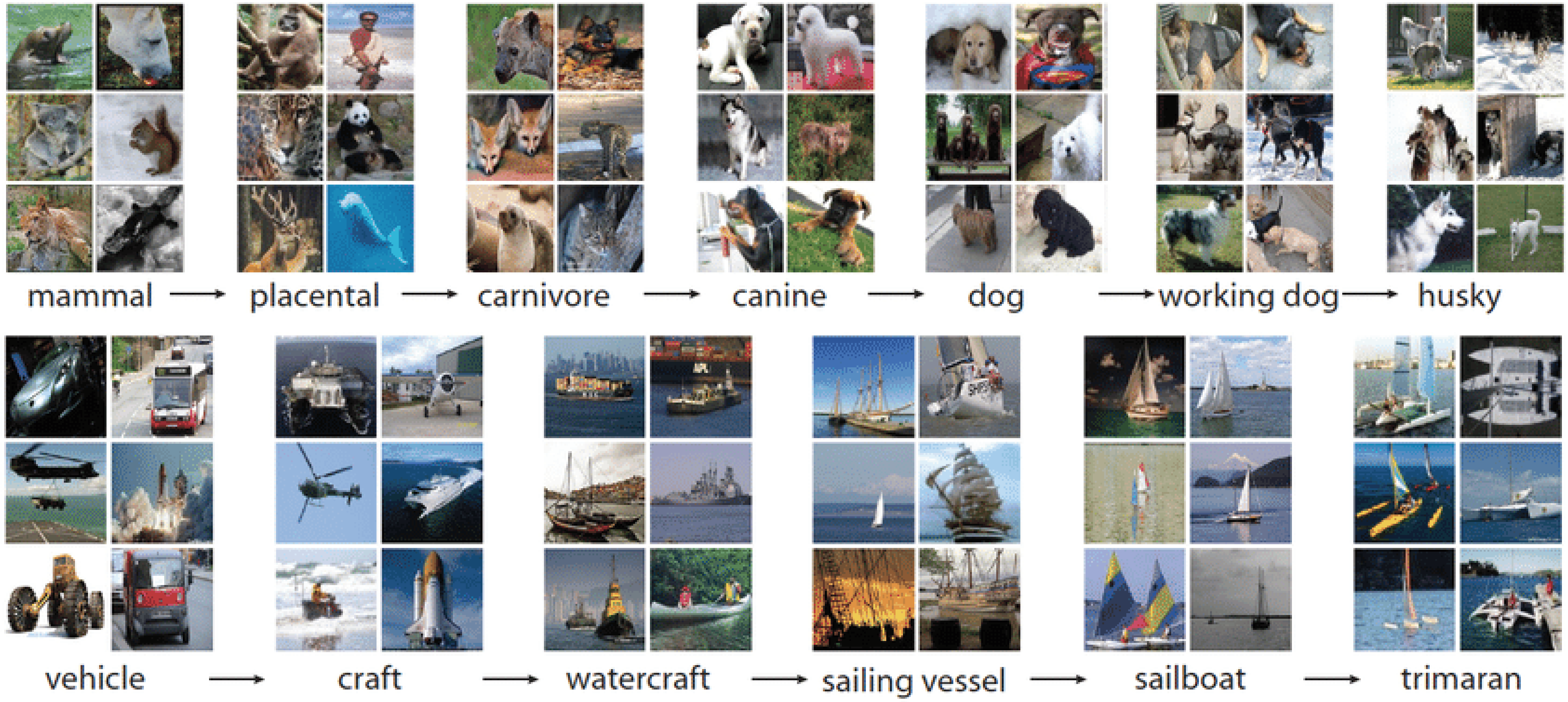


- **1998: LeNet** (AT&T labs) was one of the first CNN able to learn from raw data using backpropagation.
- It has two convolutional layers, two **mean**-pooling layers, two fully-connected layers and an output layer.
- It uses tanh as the activation function and works on CPU only.
- Used for handwriting recognition (for example ZIP codes).

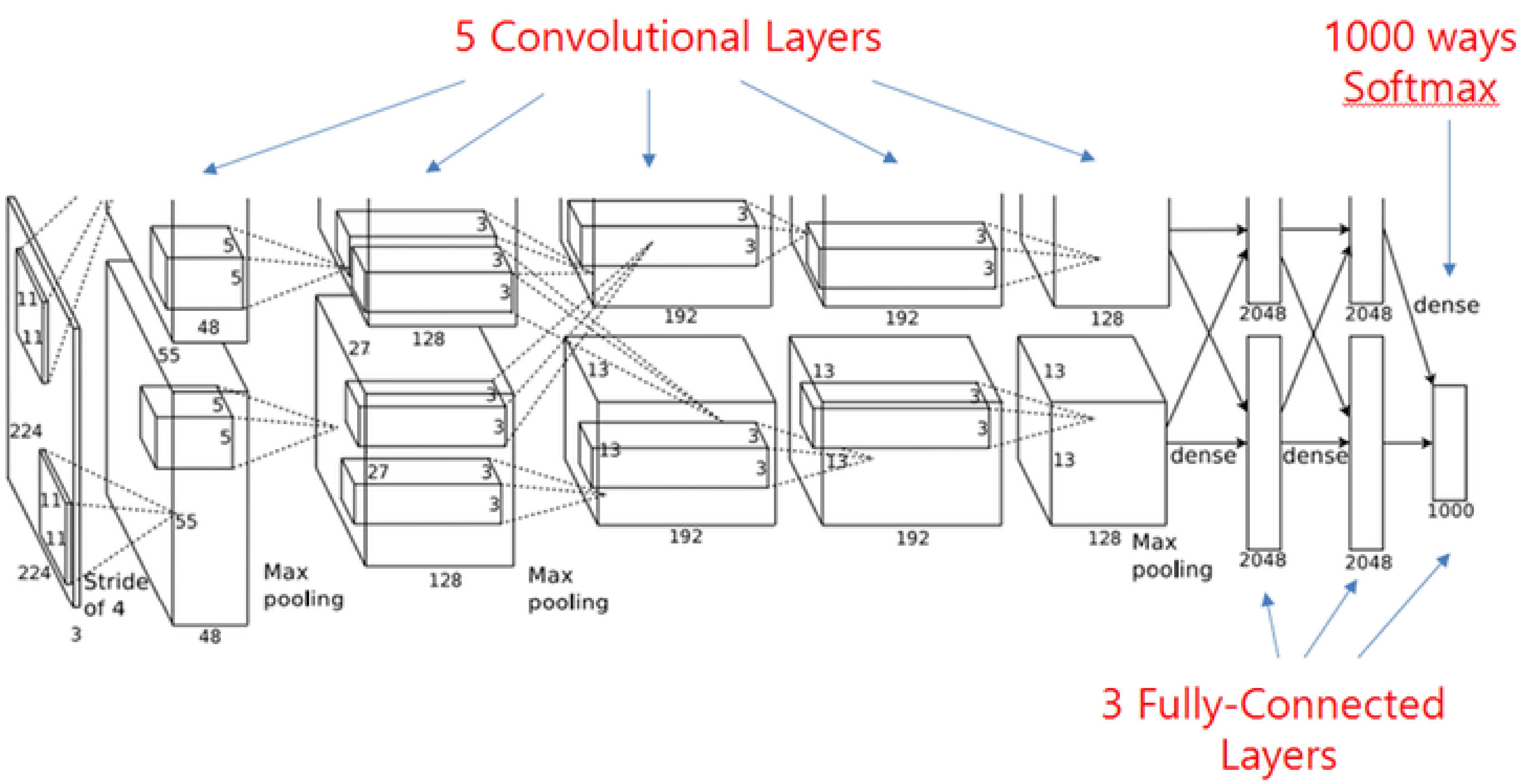
LeCun et al. (1998). Gradient Based Learning Applied to Document Recognition. Proceedings of the IEEE.

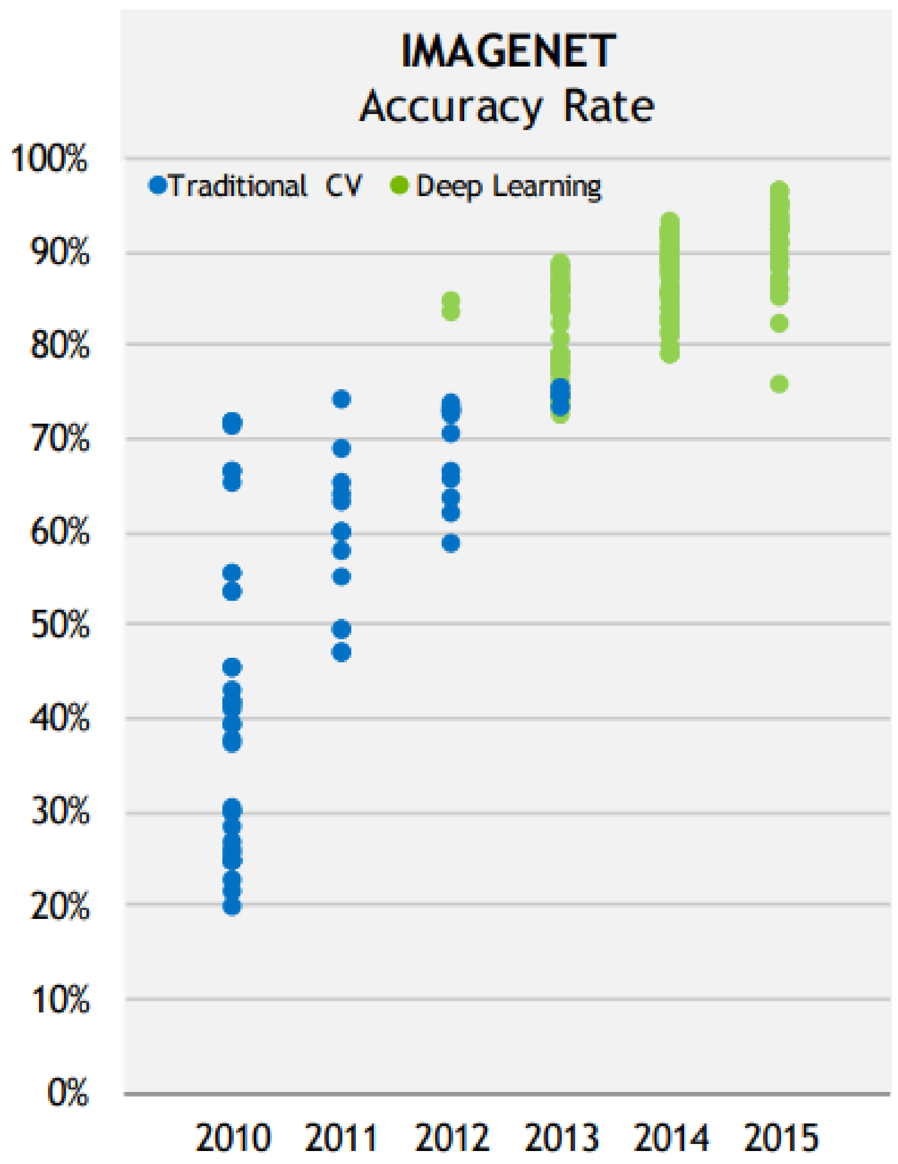
ImageNet object recognition challenge (image-net.org)

- The ImageNet challenge was a benchmark for computer vision algorithms, providing millions of annotated images for object recognition, detection and segmentation.
- 14 millions images (224x224), 1000 classes.



AlexNet

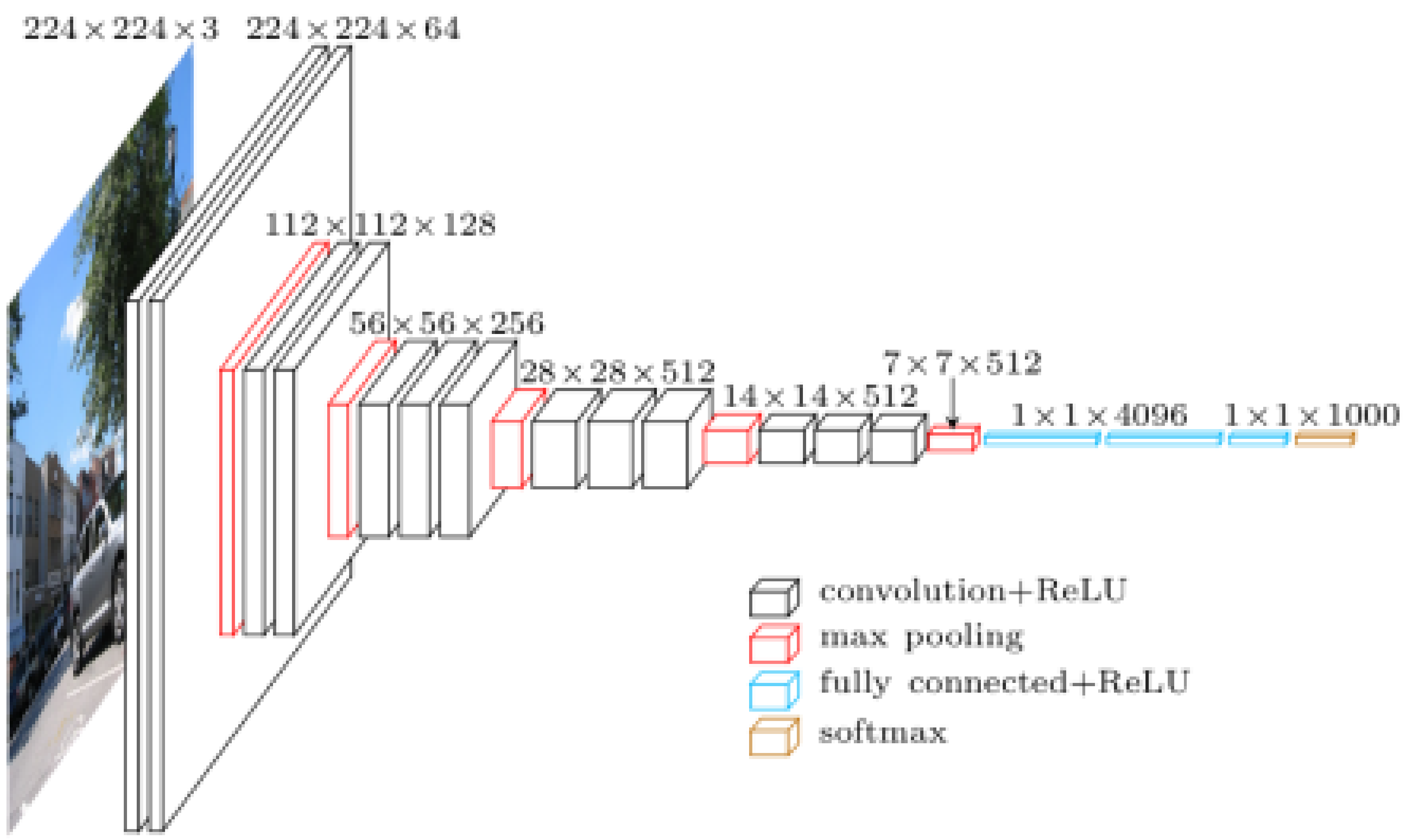




- **2012: AlexNet** (Toronto University) started the DL revolution by winning ImageNet 2012.
- Similar architecture to LeNet, but trained on two GPUs using augmented data.
- Uses ReLU, max-pooling, dropout, SGD with momentum, L2 regularization.

Krizhevsky, Sutskever and Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks. NIPS.

VGG-16

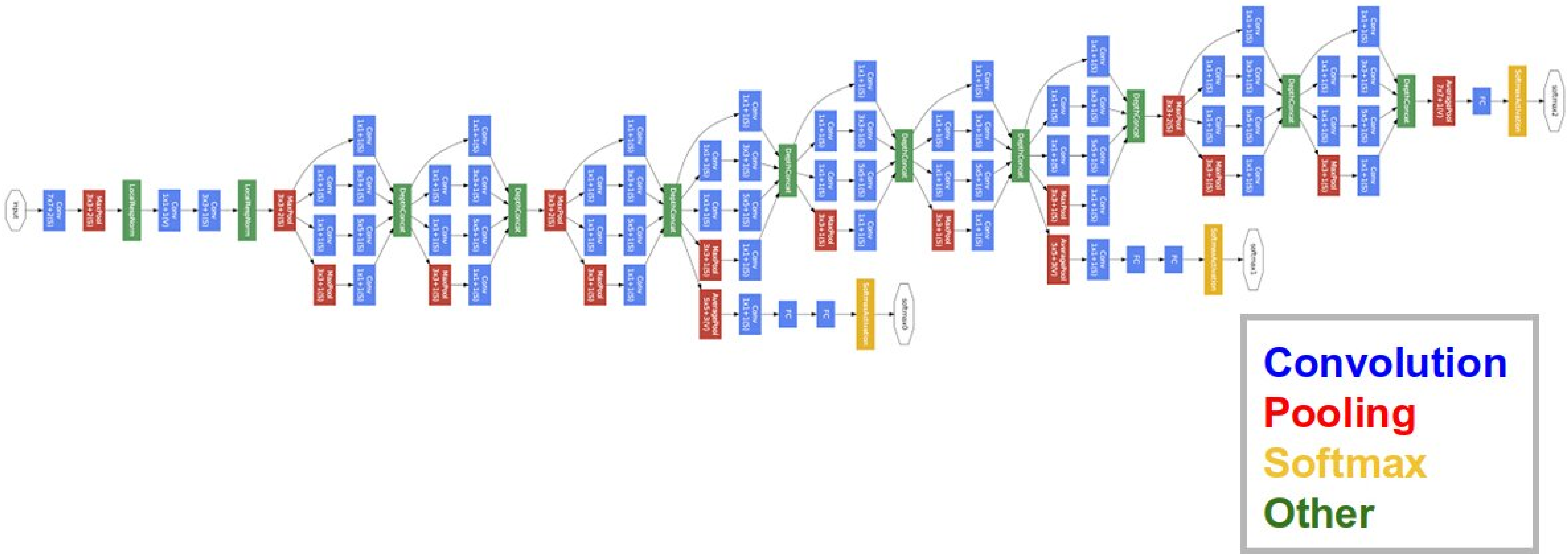


- **2014: VGG-16** (Visual Geometry Group, Oxford) placed second at ImageNet 2014.
- It went much deeper than AlexNet with 16 parameterized layers (a VGG-19 version is also available with 19 layers).
- Its main novelty is that two convolutions are made successively before the max-pooling, implicitly increasing the receptive field (2 consecutive 3x3 filters cover 5x5 pixels).
- Drawback: 140M parameters (mostly from the last convolutional layer to the first fully connected) quickly fill up the memory of the GPU.

Simonyan and Zisserman (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556.


THAT'S NOT ENOUGH
WE HAVE TO GO DEEPER

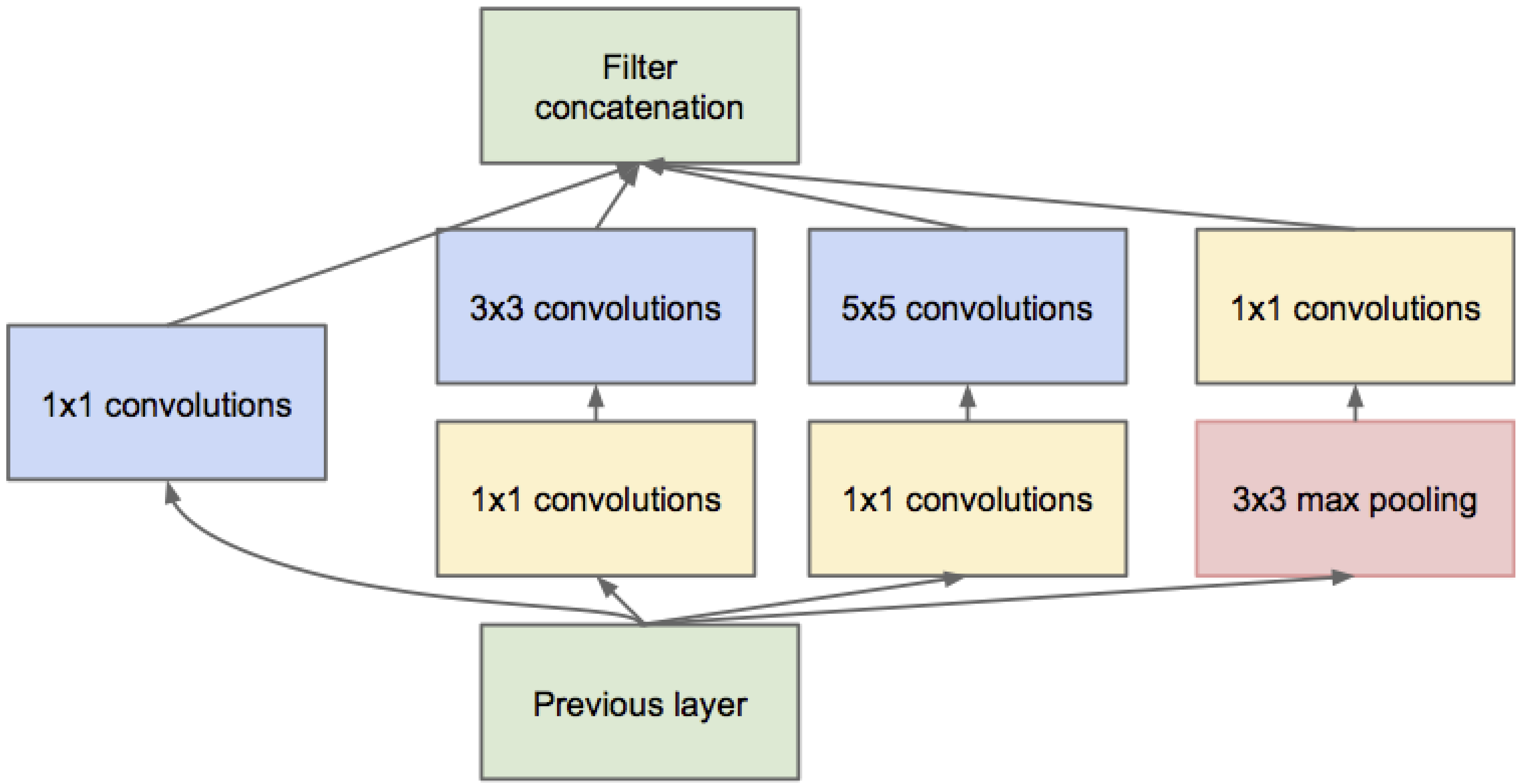
GoogLeNet - Inception v1



- **2014: GoogLeNet** (Google Brain) used Inception modules (Network-in-Network) to further complexify each stage.
- Won ImageNet 2014 with 22 layers. Dropout, SGD with Nesterov momentum.

Szegedy et al. (2014) Going Deeper with Convolutions, CVPR'14. arXiv:1409.4842.

Inception module



- Inside GoogleNet, each **Inception** module learns features at different resolutions using convolutions and max poolings of different sizes.
- 1x1 convolutions are **shared MLPS**: they transform a (w, h, d_1) tensor into (w, h, d_2) pixel per pixel.
- The resulting feature maps are concatenated along the feature dimension and passed to the next module.

Szegedy et al. (2014) Going Deeper with Convolutions, CVPR'14. arXiv:1409.4842.

GoogLeNet - Inception v1

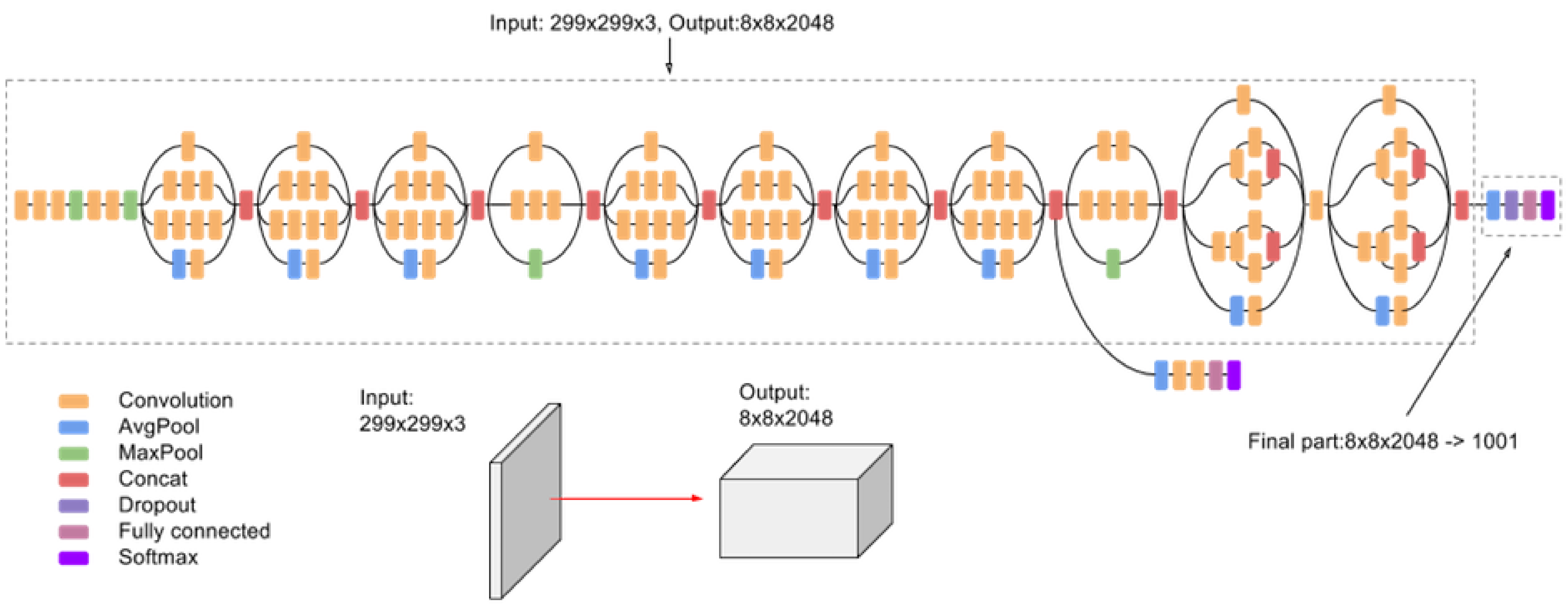
- Three softmax layers predict the classes at different levels of the network. Combined loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}}[- \mathbf{t} \, \log \mathbf{y}_1 - \mathbf{t} \, \log \mathbf{y}_2 - \mathbf{t} \, \log \mathbf{y}_3]$$

- Only the deeper softmax layer matters for the prediction.
- The additional losses improve convergence by fight vanishing gradients: the early layers get useful gradients from the lower softmax layers.

Szegedy et al. (2014). Going deeper with convolutions, CVPR'14. arXiv:1409.4842.

Inception networks



Source: https://cloud.google.com/tpu/docs/inception-v3-advanced

- Several variants of GoogleNet have been later proposed: Inception v2, v3, InceptionResNet, Xception…
- Xception has currently the best top-1 accuracy on ImageNet: 126 layers, 22M parameters (88 MB).
- Pretrained weights are available in `keras`:

```
tf.keras.applications.Xception(include_top=True, weights="imagenet")
```

References

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. (2015). Rethinking the Inception Architecture for Computer Vision. arXiv:151200567.

Chollet F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. arXiv:161002357.

Residual networks : ResNets

He and al. (2016). Deep Residual Learning for Image Recognition. ICML16

2015: ResNet (Microsoft). Won ImageNet 2015

Residual networks : ResNets

- Skip connections help overcome the **vanishing gradients** problem, as the contribution of bypassed layers to the backpropagated gradient is 1.

$$\mathbf{h}_n = f_W(\mathbf{h}_{n-1}) + \mathbf{h}_{n-1}$$

$$\frac{\partial \mathbf{h}_n}{\partial \mathbf{h}_{n-1}} = \frac{\partial f_W(\mathbf{h}_{n-1})}{\partial \mathbf{h}_{n-1}} + 1$$

- The norm of the gradient stays roughly around one, limiting vanishing.
- Skip connections can bypass whole blocks of layers.
- ResNet can have many layers without vanishing gradients. The most popular variants are:
  - ResNet-50.
  - ResNet-101.
  - ResNet-152.
- It was the first network to make an heavy use of **batch normalization**.

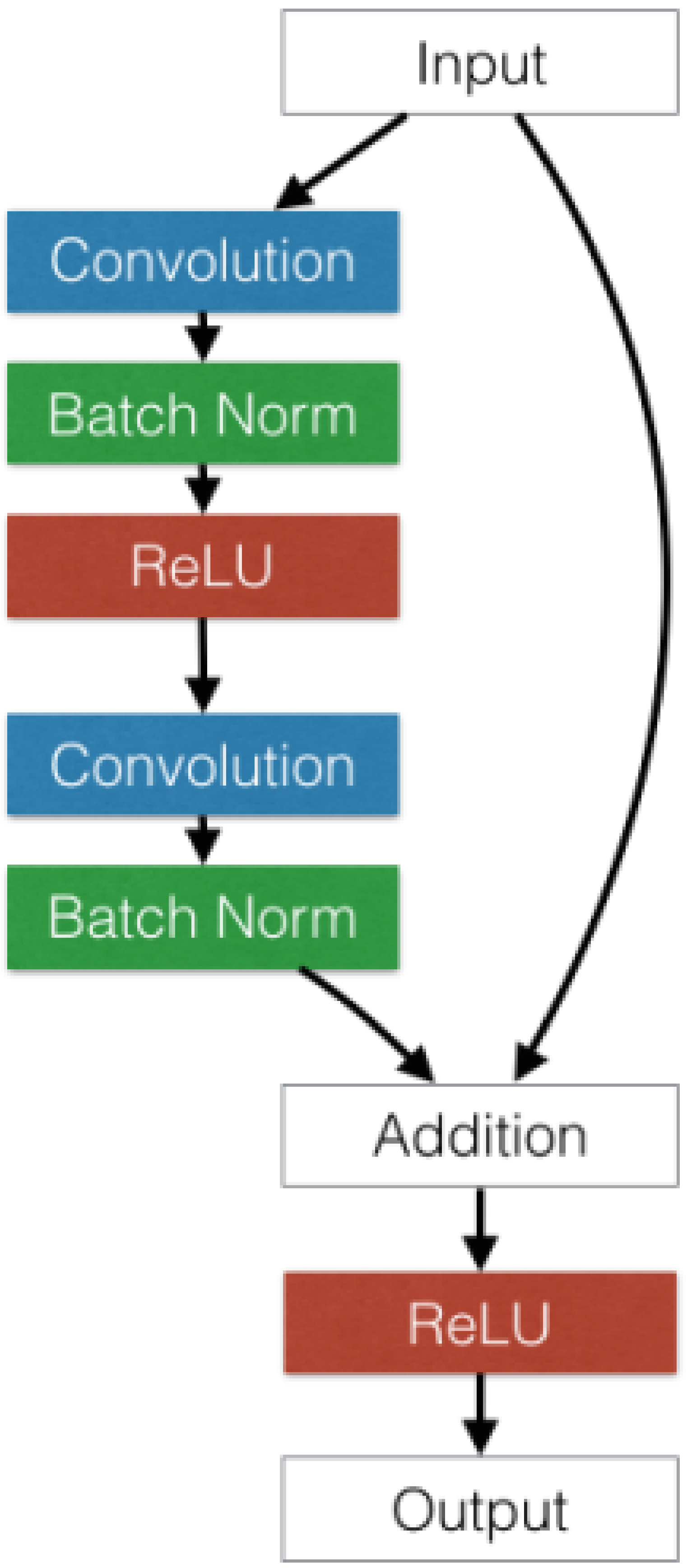
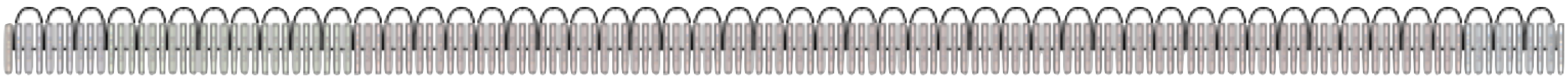


Figure 1. A RestNet basic block

He and al. (2016). Deep Residual Learning for Image Recognition. ICML16

HighNets: Highway networks



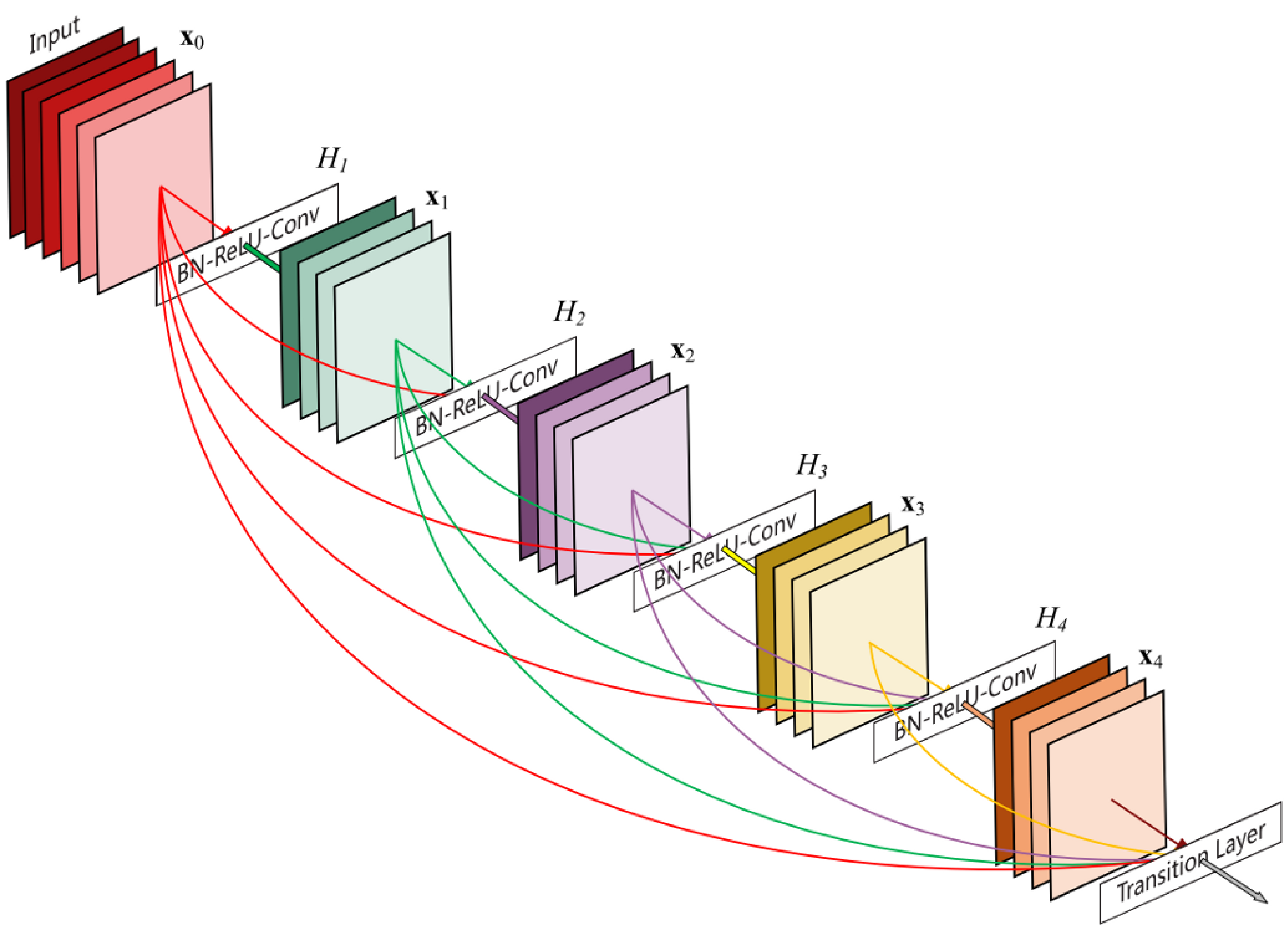
- **Highway networks** (IDSIA) are residual networks which also learn to balance inputs with feature extraction:

$$\mathbf{h}_n = T_{W'} \, f_W (h_{n-1}) + (1 - T_{W'}) \, h_{n-1}$$

- The balance between the **primary** pathway and the **skip** pathway adapts to the task.
- Has been used up to 1000 layers.
- Improved state-of-the-art accuracy on MNIST and CIFAR-10.

Srivastava, Greff and Schmidhuber (2016). Highway networks. ICML15

DenseNets: Dense networks



- **Dense networks** (Cornell University & Facebook AI) are residual networks that can learn **bypasses** between any layer of the network (up to 5).
- 100 layers altogether.
- Improved state-of-the-art accuracy on five major benchmarks.

Huang, Liu and Weinberger (2016). Densely Connected Convolutional Networks. arXiv:1608.06993*

Model zoos

- These famous models are described in their respective papers, you could reimplement them and train them on ImageNet.
- Fortunately, their code is often released on Github by the authors or reimplemented by others.
- Most frameworks maintain **model zoos** of the most popular networks.
- Some models also have **pretrained weights** available, mostly on ImageNet.
- Very useful for **transfer learning** (see later).

- Overview website:

  https://modelzoo.co

- Caffe:

  https://github.com/BVLC/caffe/wiki/Model-Zoo

- Tensorflow:

  https://github.com/tensorflow/models

- Pytorch:

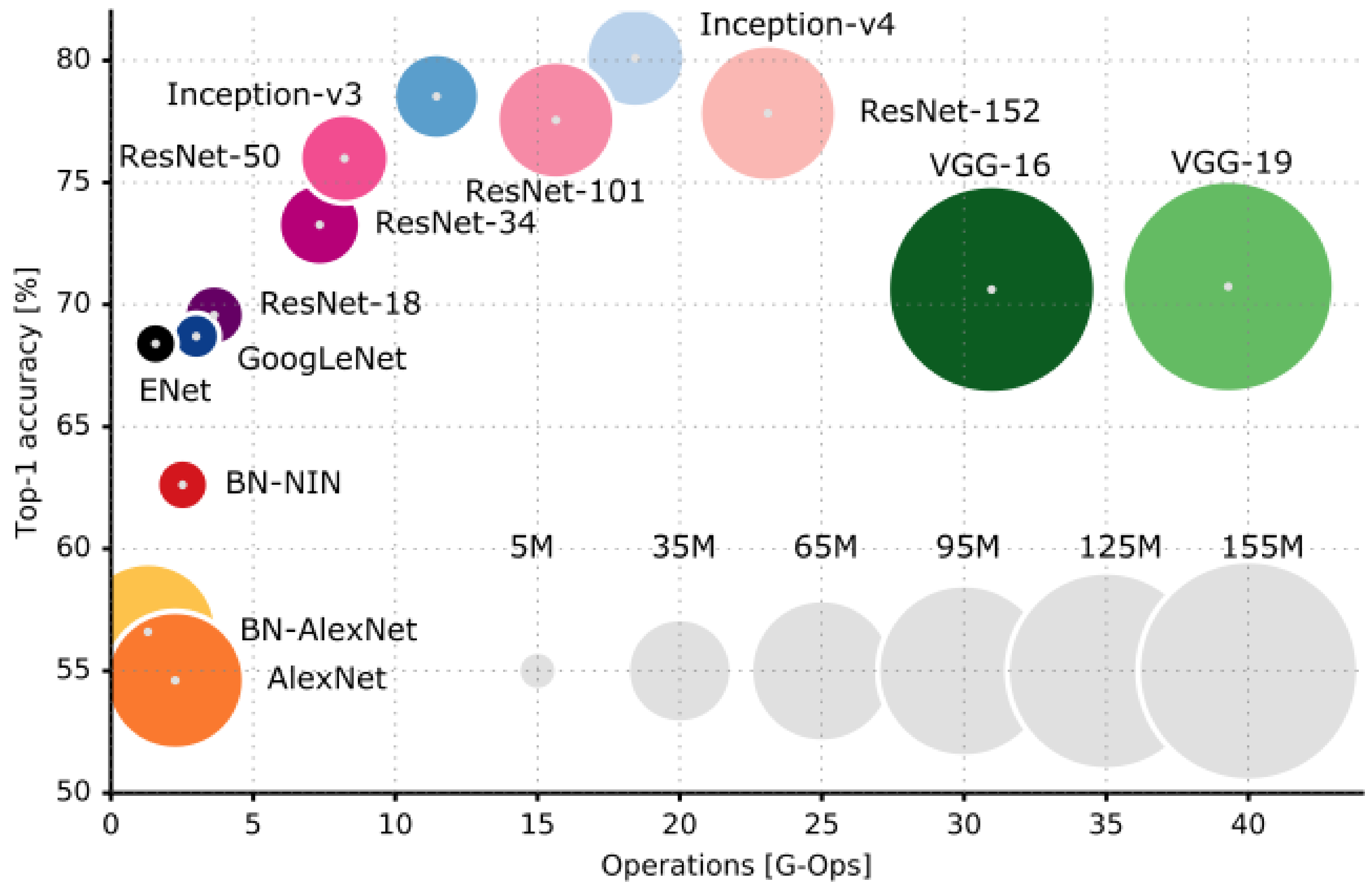
  https://pytorch.org/docs/stable/torchvision/models.html

- Papers with code:

  https://paperswithcode.com/

Comparison of the most popular networks

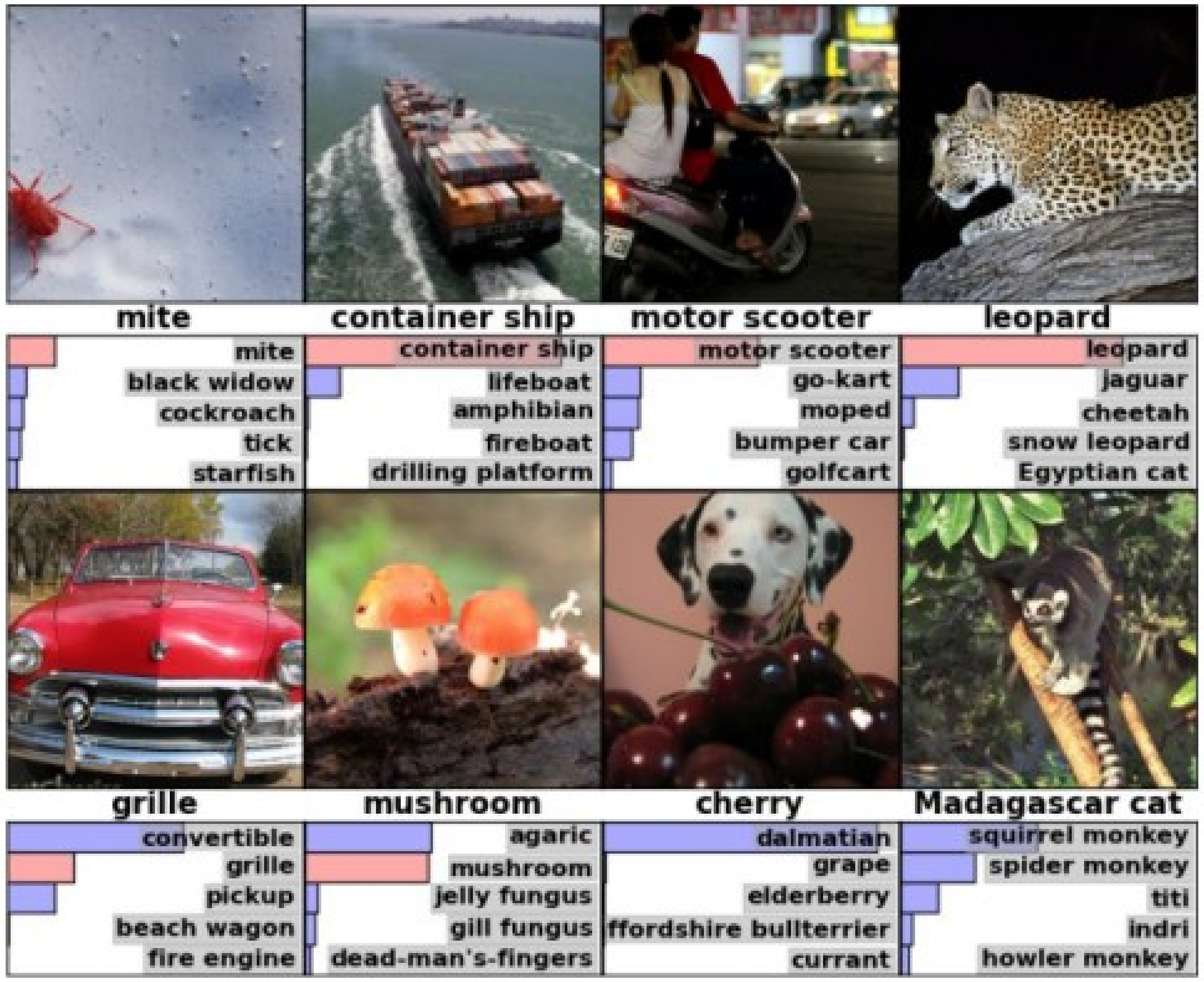
- Several criteria have to be considered when choosing an architecture:
  - Accuracy on ImageNet.
  - Number of parameters (RAM consumption).
  - Speed (flops).



Source: https://dataconomy.com/2017/04/history-neural-networks

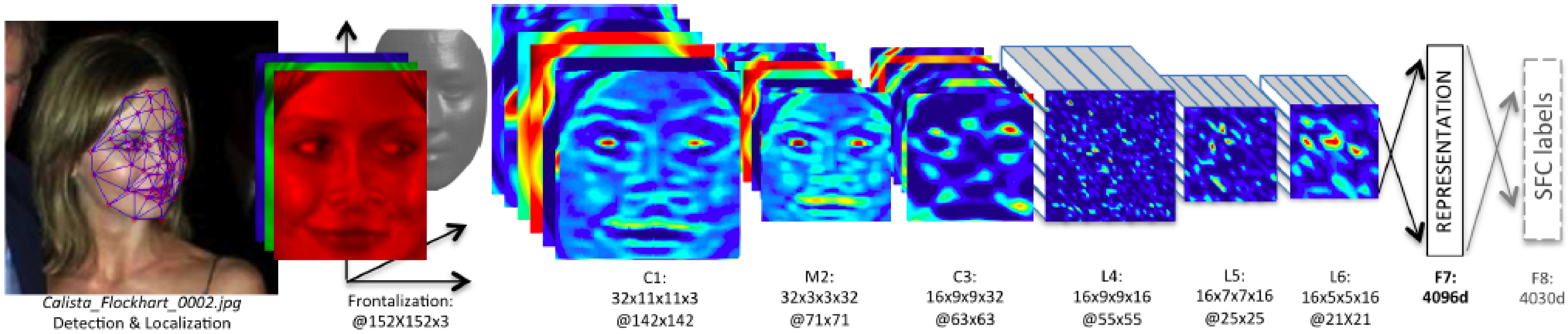
3 - Applications

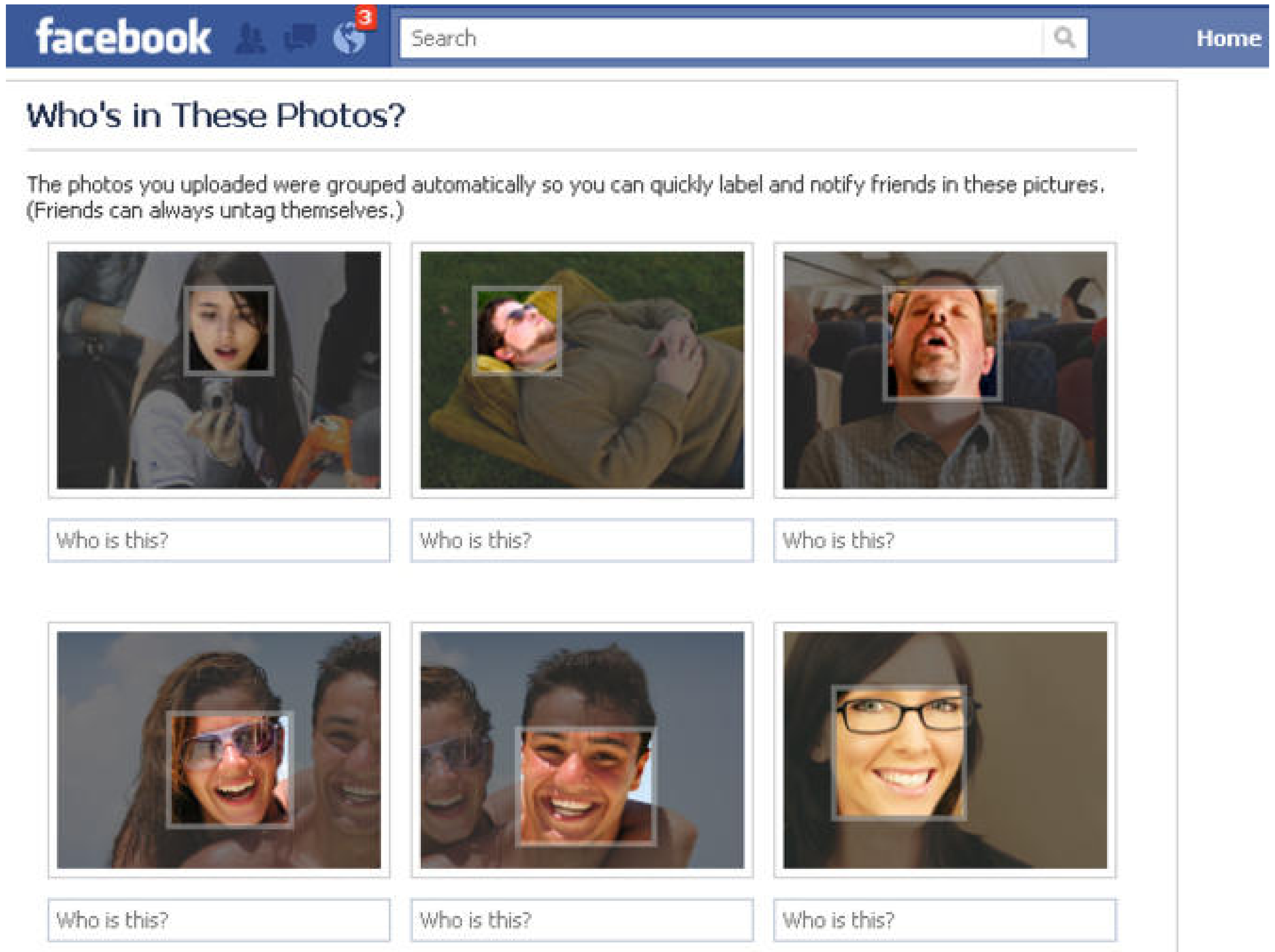
Object recognition



- **Object recognition** has become very easy, each image is associated to a label.
- With huge datasets like **ImageNet** (14 millions images), a CNN can learn to recognize 1000 classes of objects with a better accuracy than humans.
- Just get enough examples of an object and it can be recognized.

Facial recognition



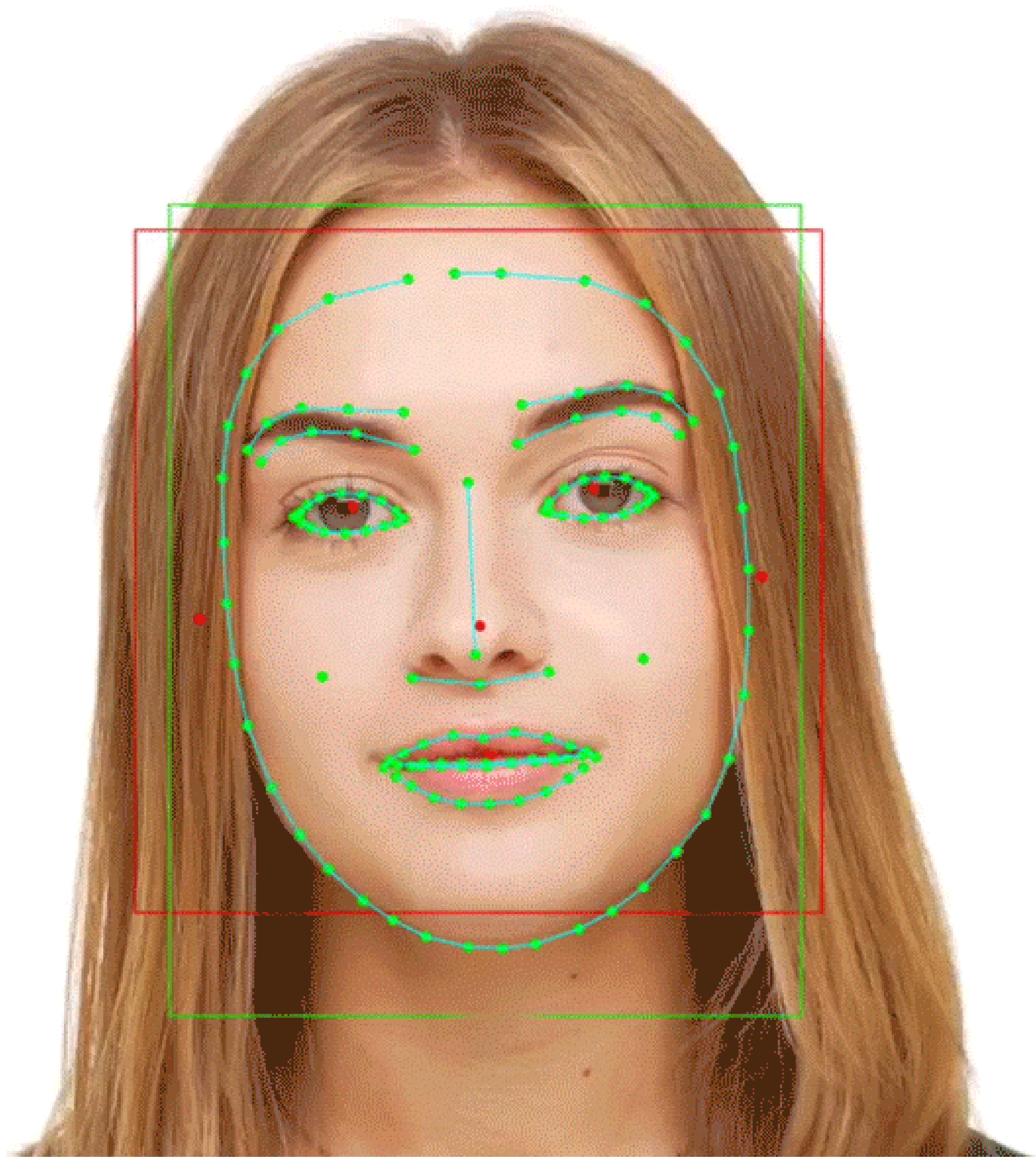


- Facebook used 4.4 million annotated faces from 4030 users to train **DeepFace**.
- Accuracy of 97.35% for recognizing faces, on par with humans.
- Used now to recognize new faces from single examples (transfer learning, one-shot learning).

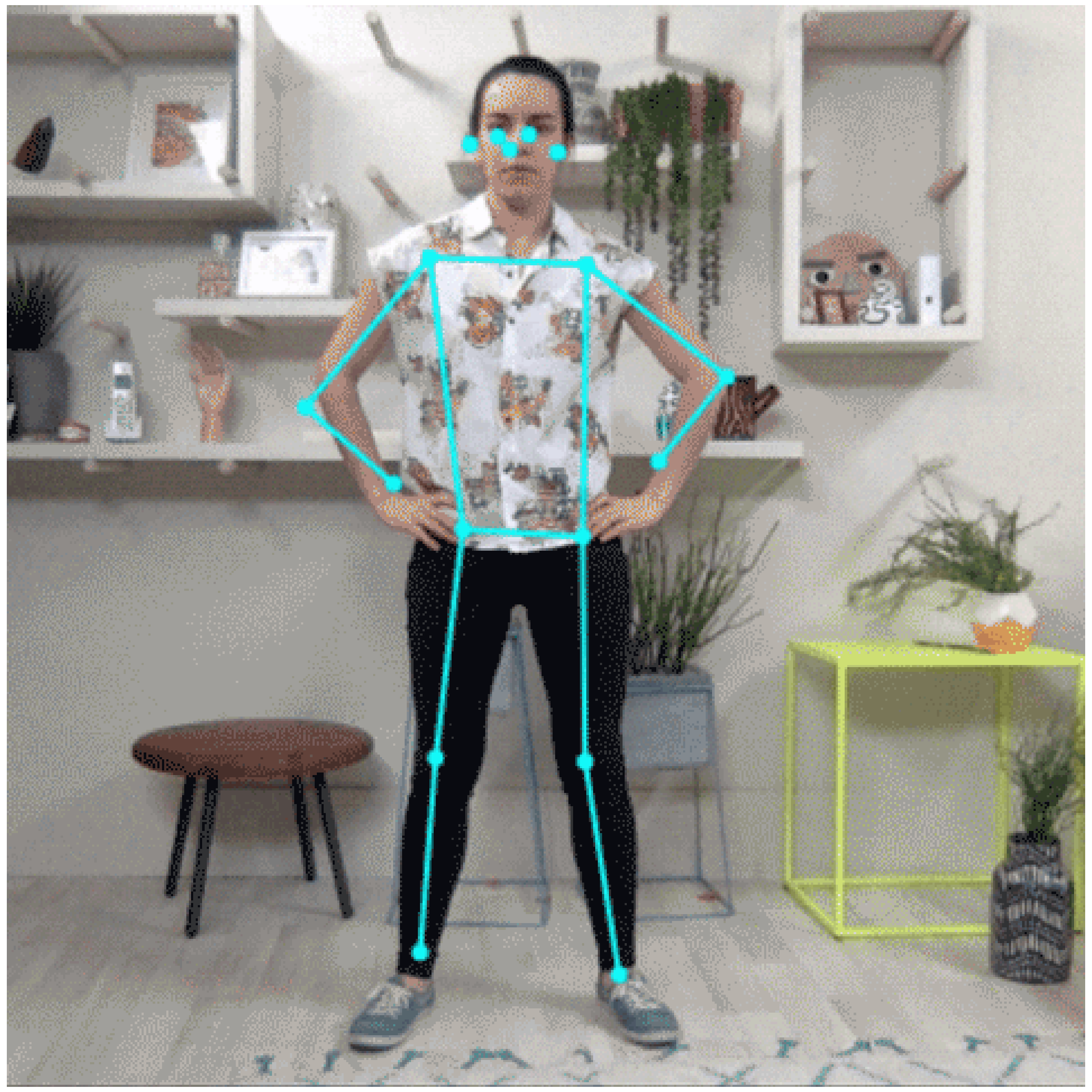
Yaniv Taigman; Ming Yang; Marc'Aurelio Ranzato; Lior Wolf (2014), "DeepFace: Closing the Gap to Human-Level Performance in Face Verification", CVPR.

Pose estimation

- **PoseNet** is a Inception-based CNN able to predict 3D information from 2D images.
- It can be for example the calibration matrix of a camera, 3D coordinates of joints or facial features.
- There is a free tensorflow.js implementation that can be used in the browser.



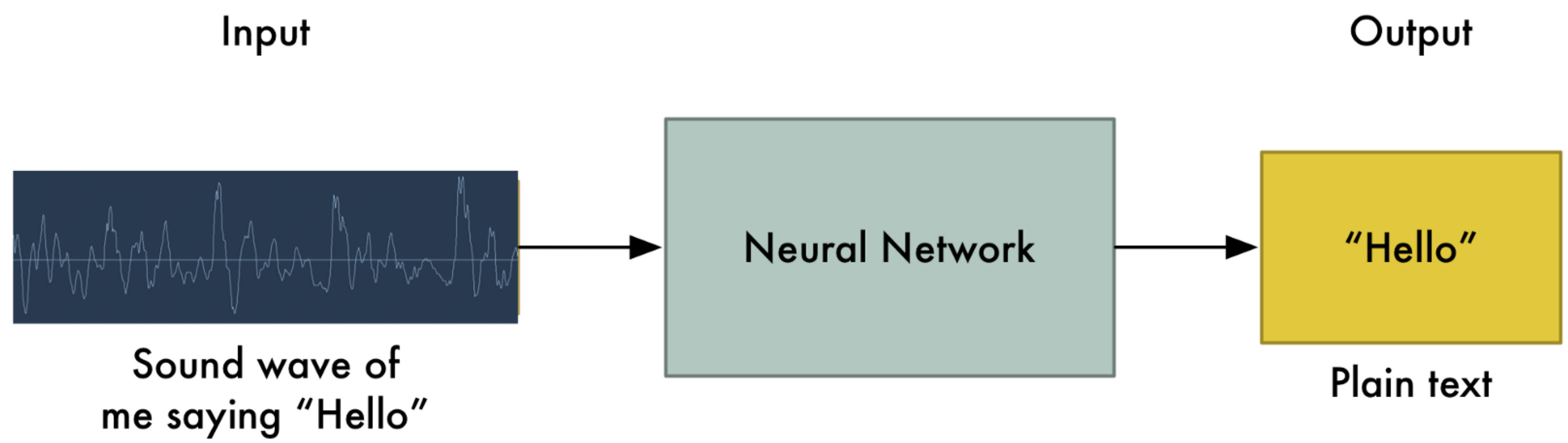
Source: https://blog.tensorflow.org/2019/01/tensorflow-lite-now-faster-with-mobile.html

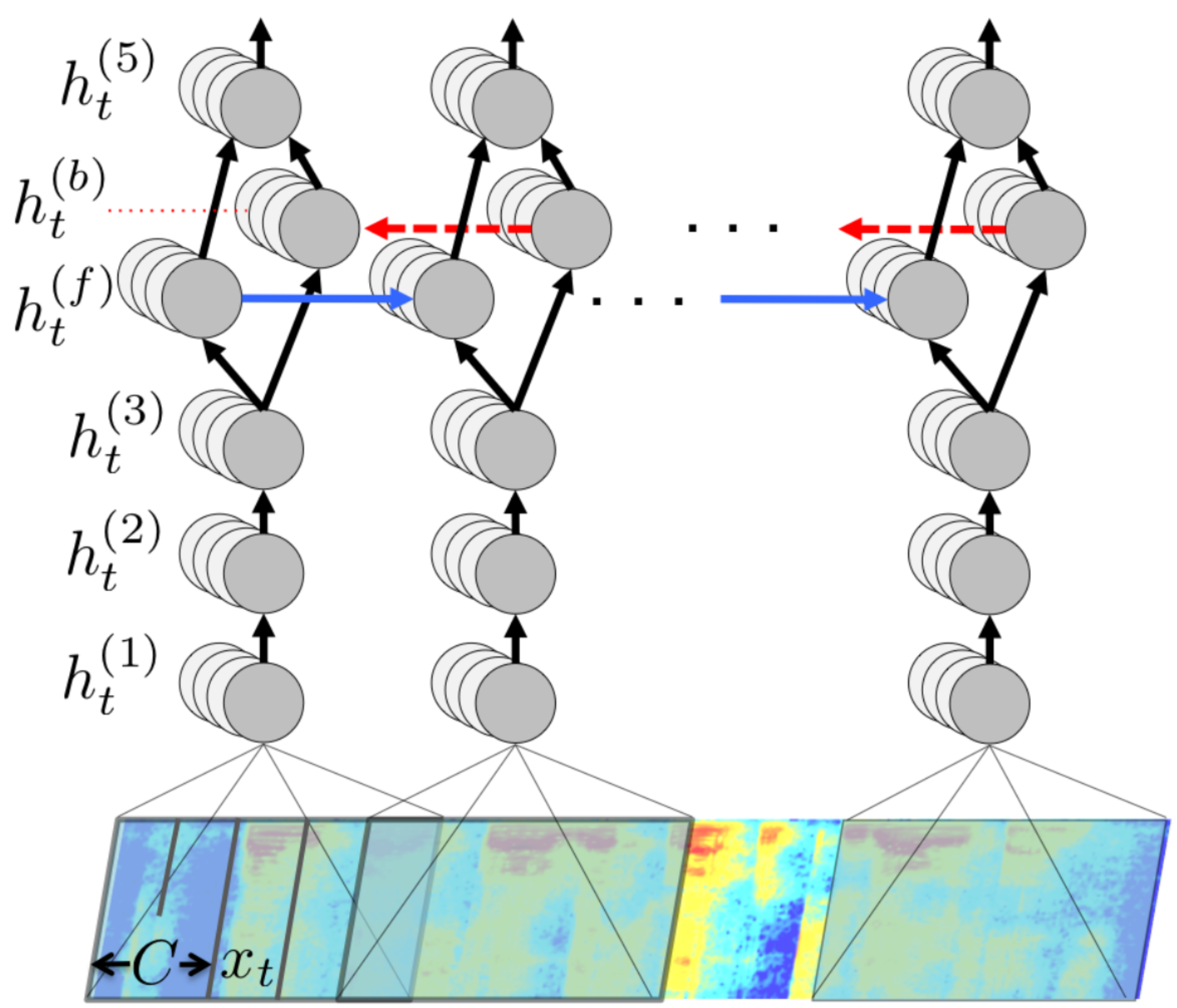


Source: https://www.tensorflow.org/lite/models/pose_estimation/overview

Kendall A, Grimes M, Cipolla R. (2016). PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. arXiv:150507427

Speech recognition

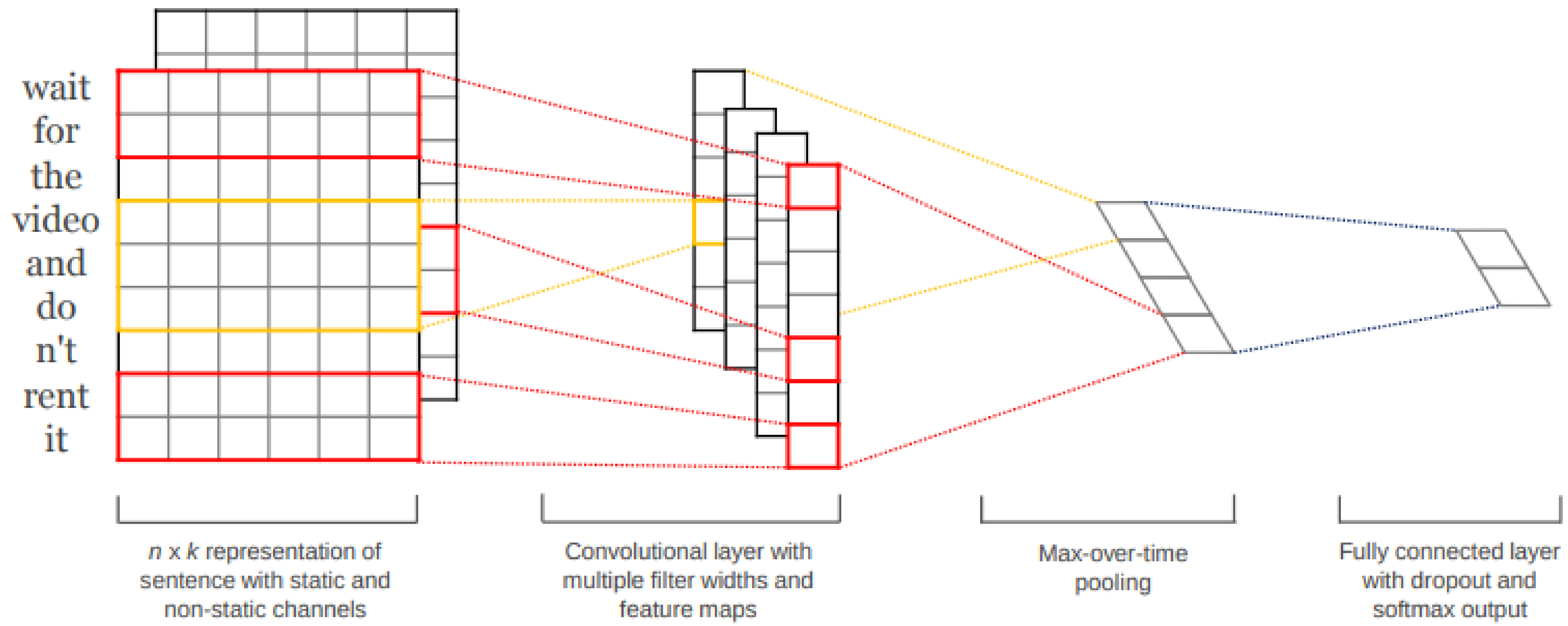




- To perform speech recognition, one could treat speech signals like images: one direction is time, the other are frequencies (e.g. mel spectrum).
- A CNN can learn to associate phonemes to the corresponding signal.
- **DeepSpeech** from Baidu is one of the state-of-the-art approaches.
- Convolutional networks can be used on any signals where early features are local.
- It uses additionally recurrent networks, which we will see later.

Hannun et al (2014). Deep Speech: Scaling up end-to-end speech recognition. arXiv:1412.5567

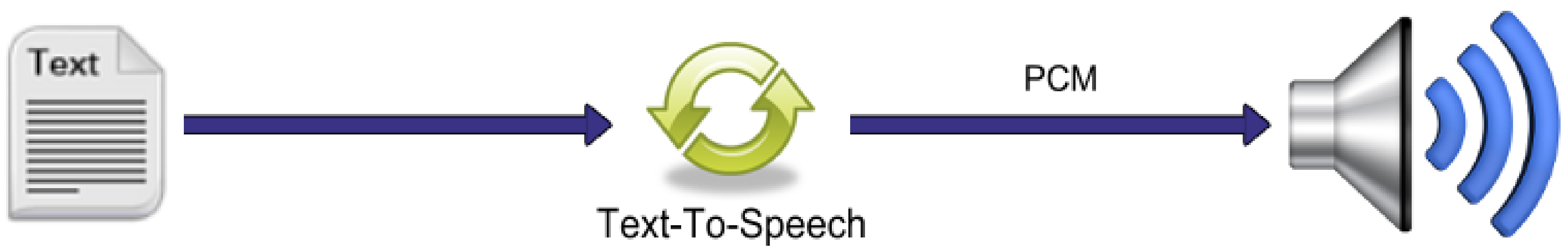
Sentiment analysis



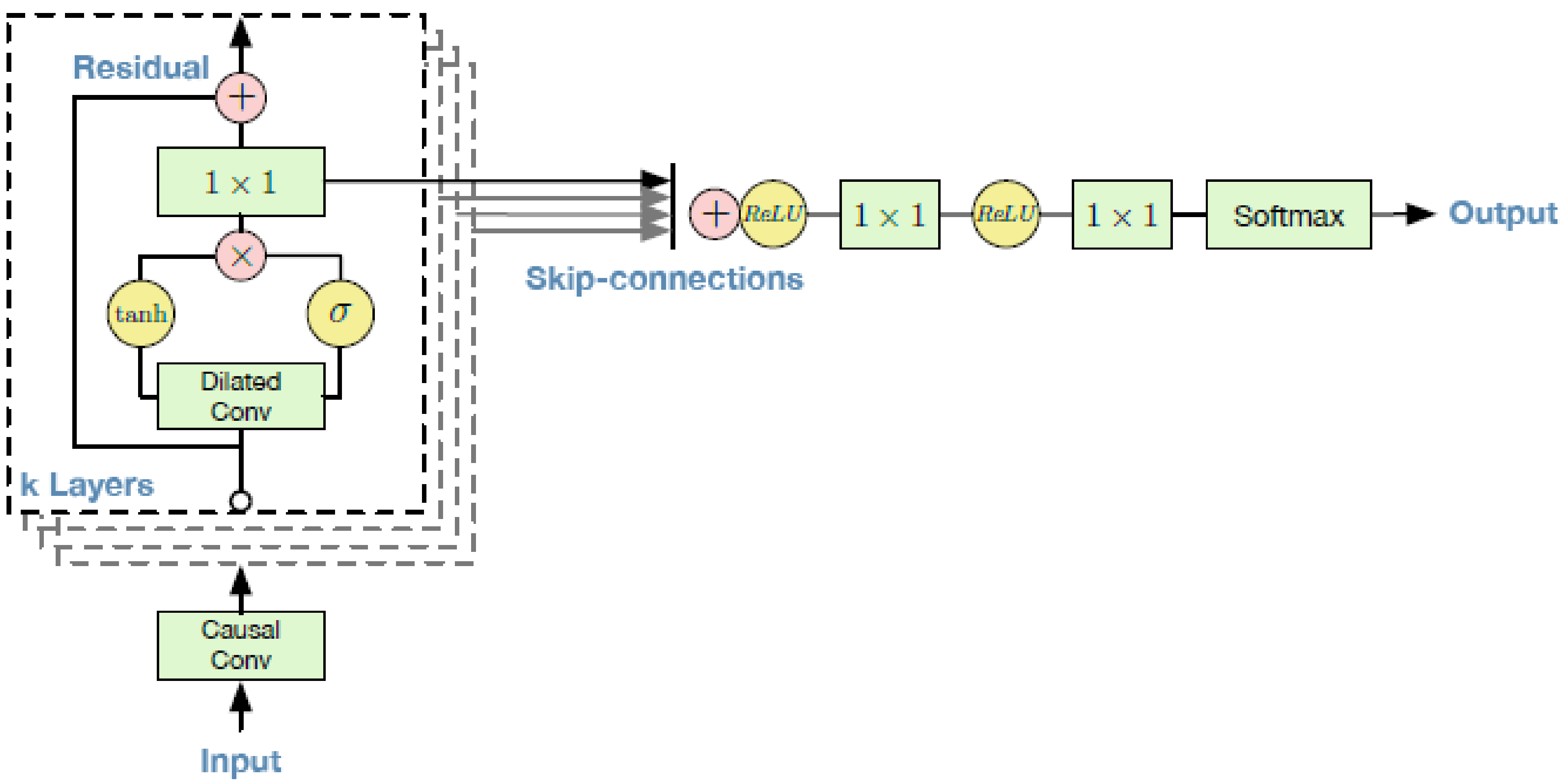
- It is also possible to apply convolutions on text.
- **Sentiment analysis** assigns a positive or negative judgment to sentences.
- Each word is represented by a vector of values (word2vec).
- The convolutional layer can slide over all over words to find out the sentiment of the sentence.

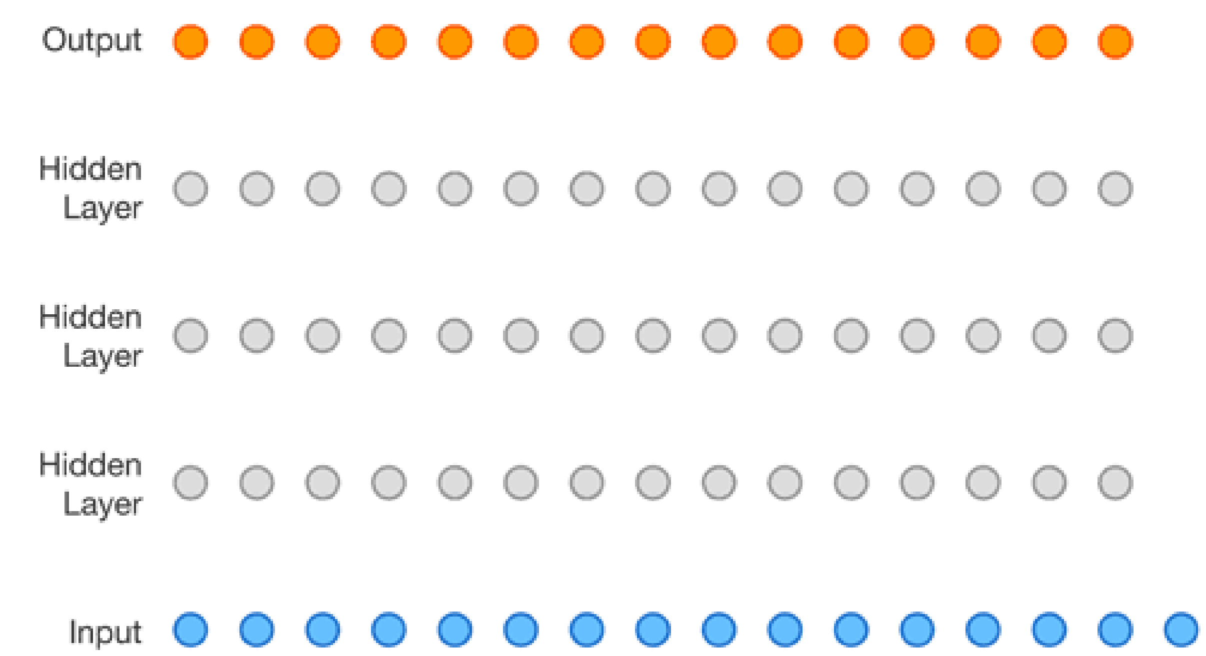
Kim (2014). Convolutional Neural Networks for Sentence Classification. arXiv:1408.5882v2

Wavenet : text-to-speech synthesis



- **Text-To-Speech** (TTS) is also possible using CNNs.
- Google Home relies on **Wavenet**, a complex CNN using *dilated convolutions* to grasp long-term dependencies.



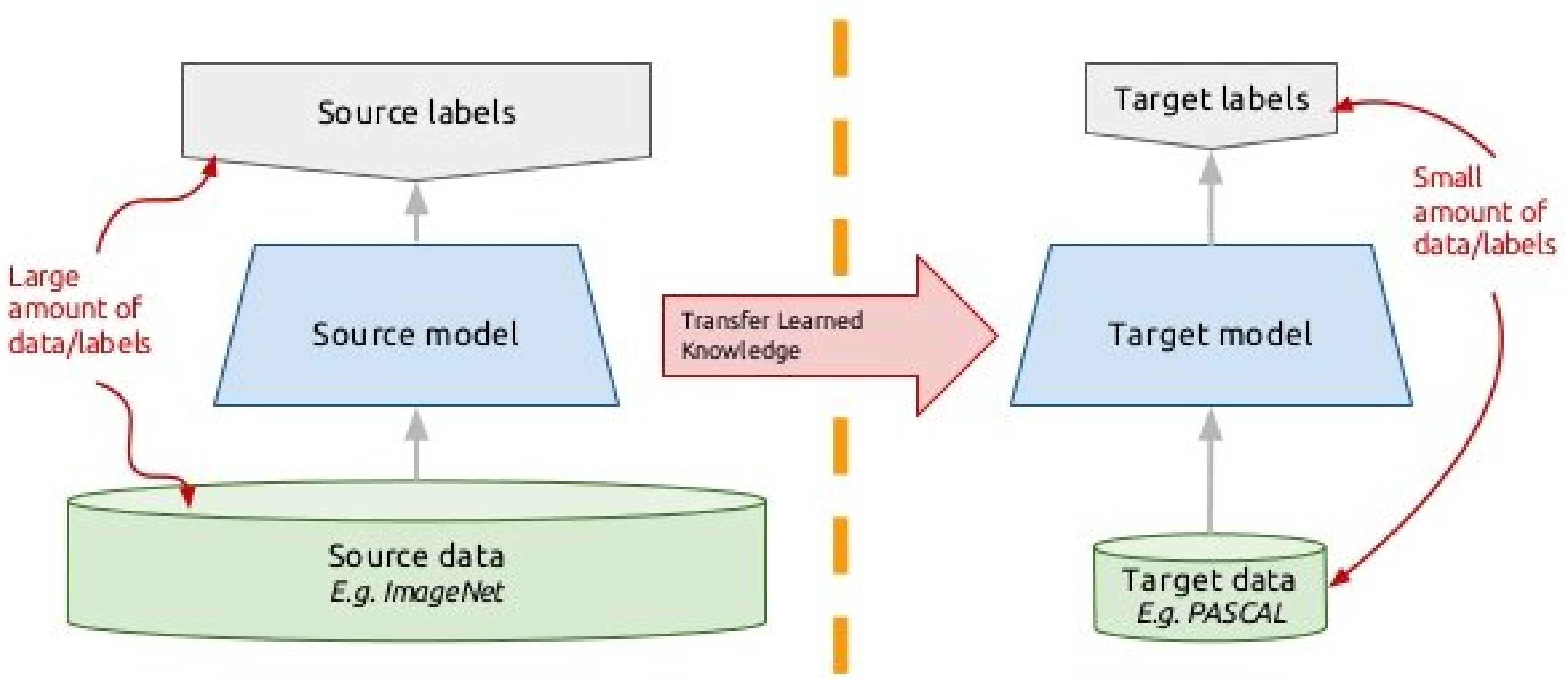


Source: https://deepmind.com/blog/wavenet-generative-model-raw-audio/

4 - Transfer learning

Transfer learning / Domain adaptation

- **Myth:** ones needs at least one million labeled examples to use deep learning.
- This is true if you train the CNN **end-to-end** with randomly initialized weights.
- But there are alternatives:
    1. **Unsupervised learning** (autoencoders) may help extract useful representations using only images.
    2. **Transfer learning** allows to re-use weights obtained from a related task/domain.



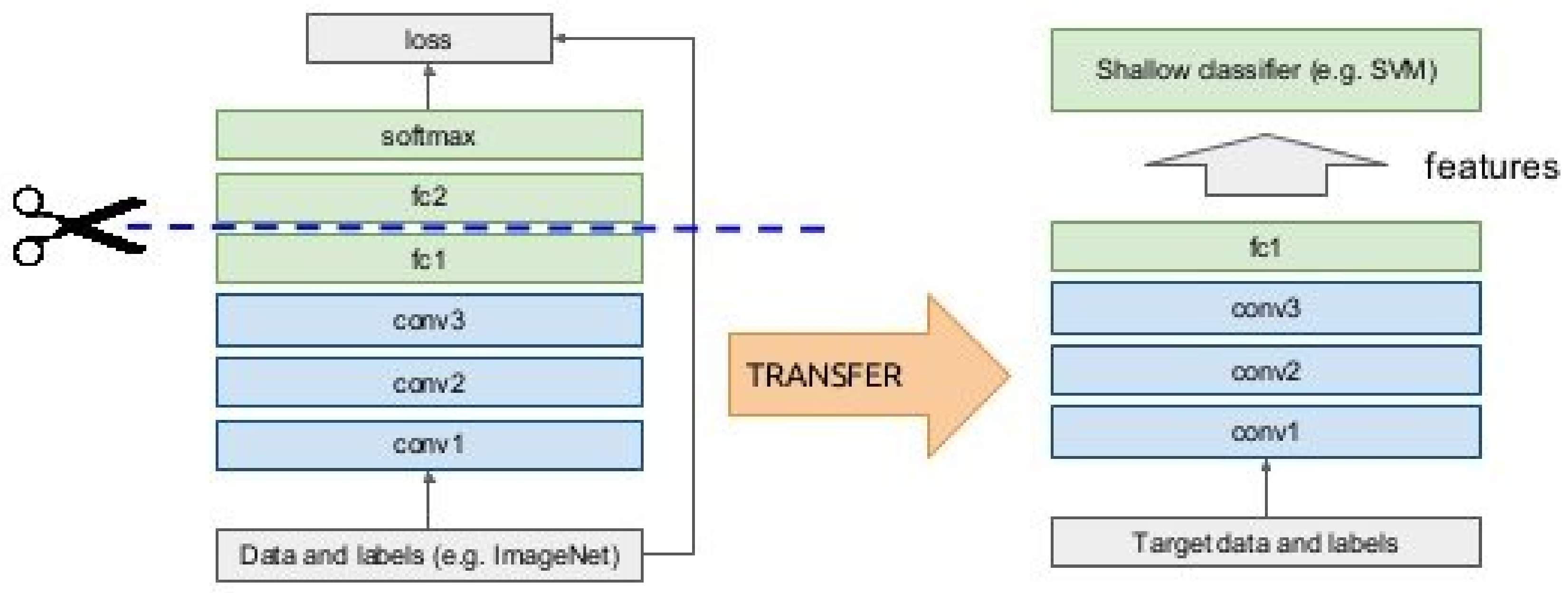
Source: http://imatge-upc.github.io/telecombcn-2016-dlcv

Transfer learning / Domain adaptation

- Take a classical network (VGG-16, Inception, ResNet, etc.) trained on ImageNet (if your task is object recognition).

Off-the-shelf

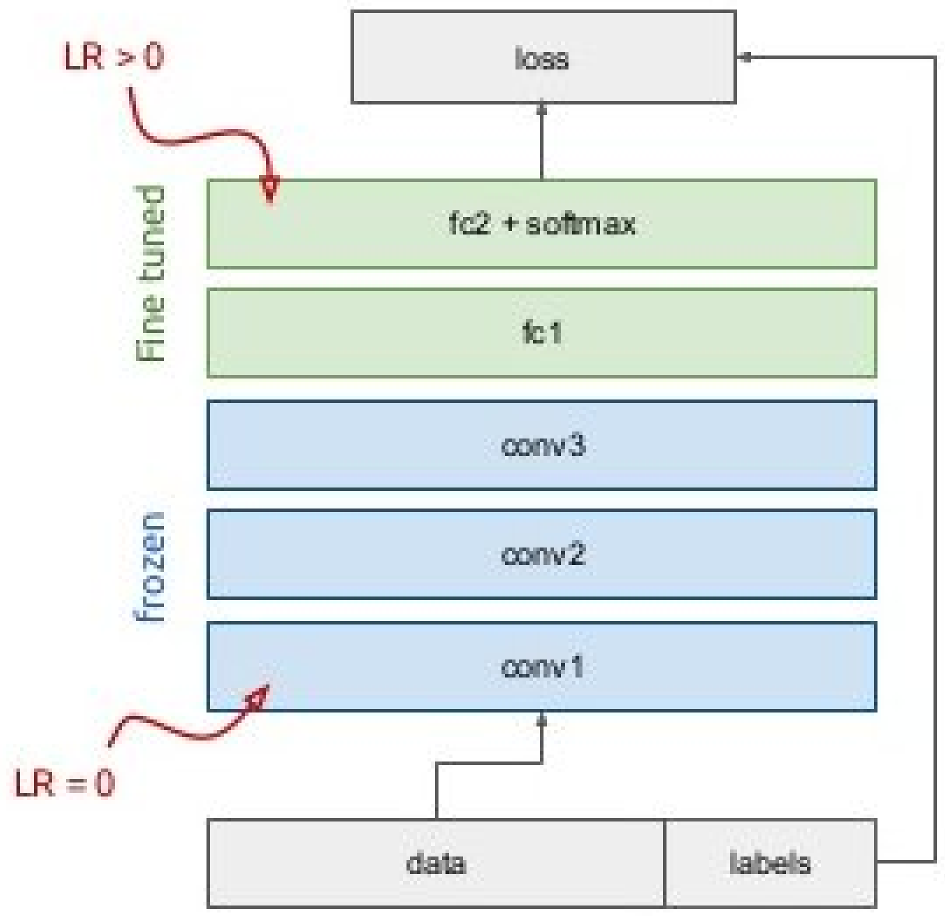
- Cut the network before the last layer and use directly the high-level feature representation.
- Use a shallow classifier directly on these representations (not obligatorily NN).



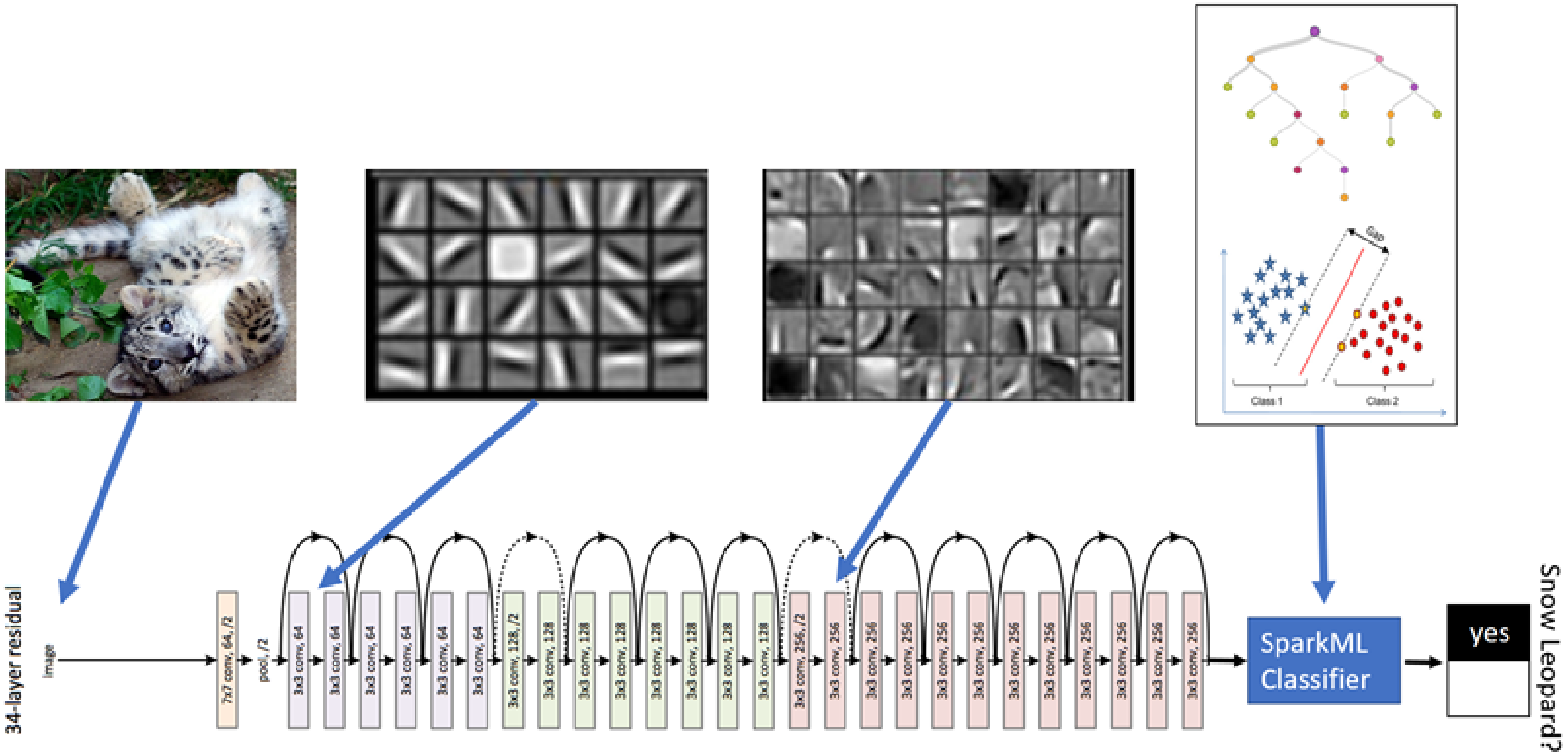
Source: http://imatge-upc.github.io/telecombcn-2016-dlcv

Fine-tuning

- Use the trained weights as initial weight values and re-train the network on your data (often only the last layers, the early ones are frozen).



Example of transfer learning



Source: https://blogs.technet.microsoft.com/machinelearning/2017/06/27/saving-snow-leopards-with-deep-learning-and-computer-vision-on-spark/

- Microsoft wanted a system to automatically detect **snow leopards** into the wild, but there were not enough labelled images to train a deep network **end-to-end**.
- They used a pretrained **ResNet50** as a feature extractor for a simple **logistic regression** classifier.

Transfer learning in keras

- Keras provides pre-trained CNNs that can be used as feature extractors:

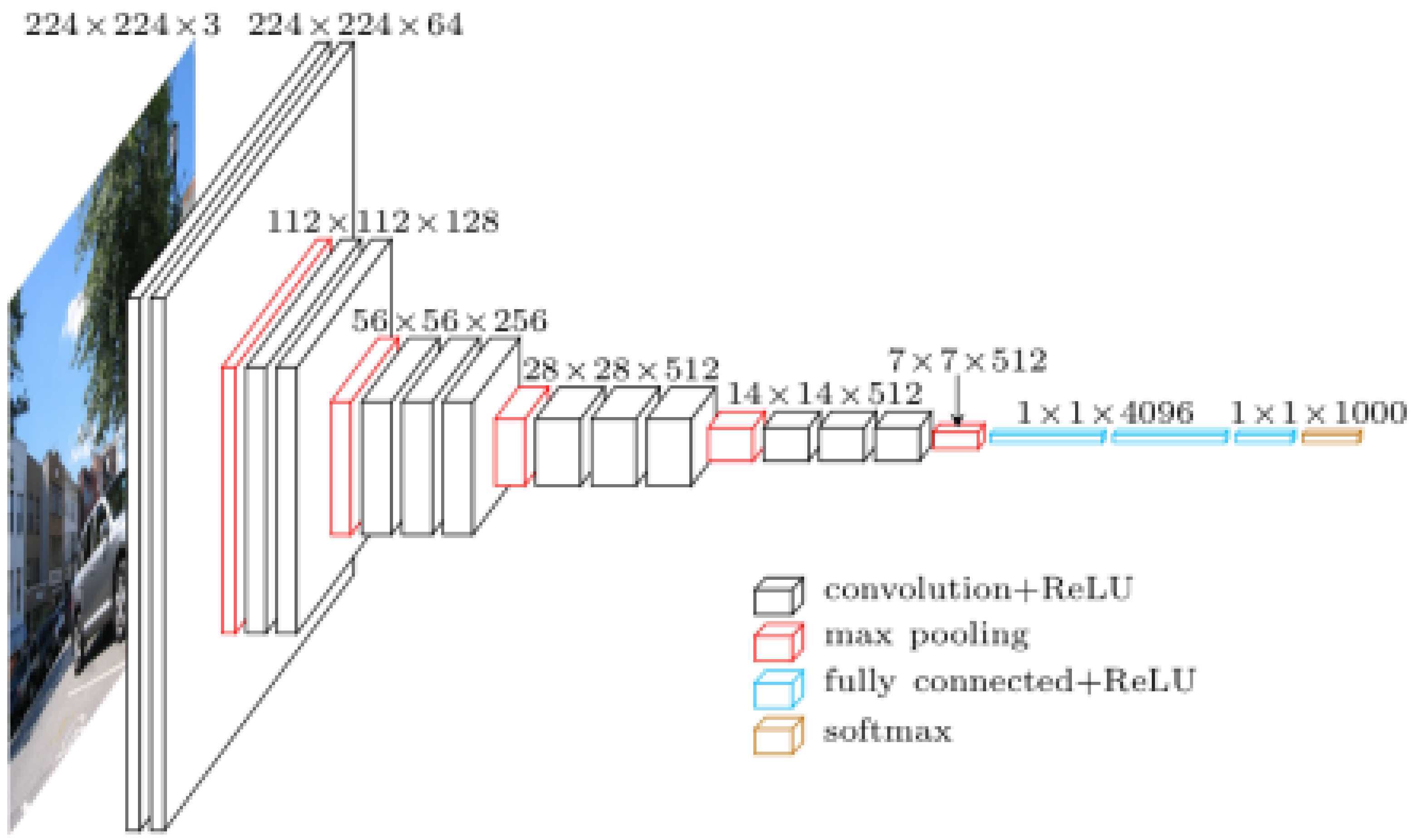
```
from tf.keras.applications.vgg16 import VGG16

# Download VGG without the FC layers
model = VGG16(include_top=False,
              input_shape=(300, 300, 3))

# Freeze learning in VGG16
for layer in model.layers:
    layer.trainable = False

# Add a fresh MLP on top
flat1 = Flatten()(model.layers[-1].output)
class1 = Dense(1024, activation='relu')(flat1)
output = Dense(10, activation='softmax')(class1)

# New model
model = Model(
    inputs=model.inputs, outputs=output)
```



- See https://keras.io/api/applications/ for the full list of pretrained networks.

5 - Ensemble learning

ImageNet recognition challenge: object recognition

- Since 2016, only ensembles of existing networks win the competitions.

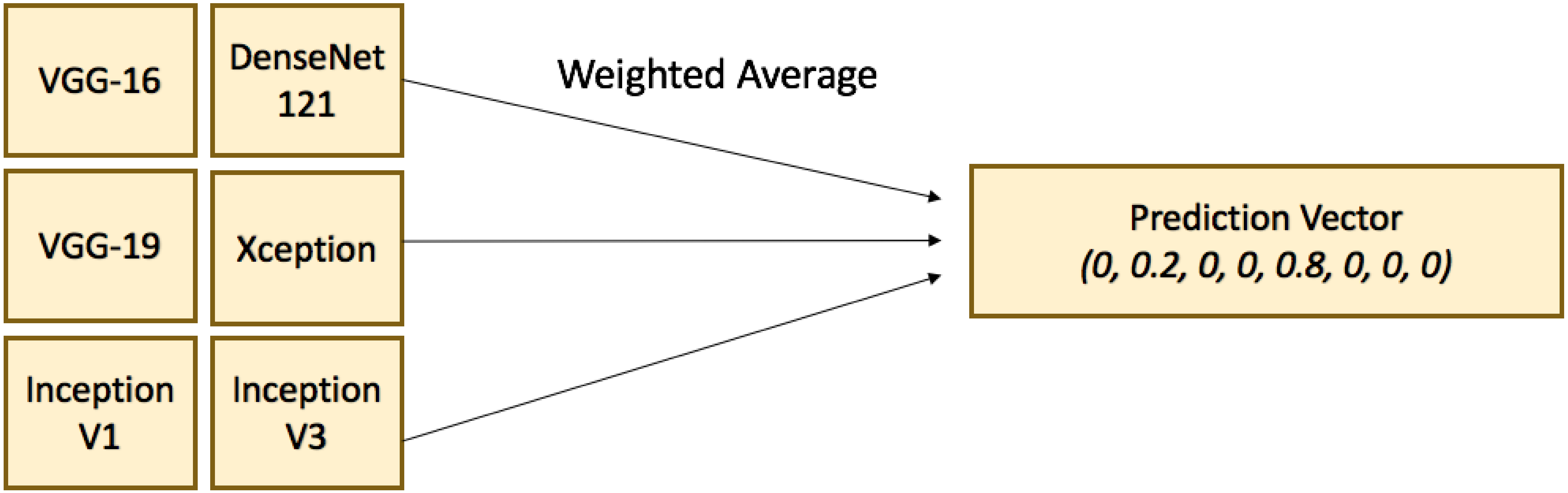
Object detection (DET)[top]

Task 1a: Object detection with provided training data

Ordered by number of categories won

Team name	Entry description	**Number of object categories won**	mean AP
CUImage	Ensemble of 6 models using provided data	109	0.662751
Hikvision	Ensemble A of 3 RPN and 6 FRCN models, mAP is 67 on val2	30	0.652704
Hikvision	Ensemble B of 3 RPN and 5 FRCN models, mean AP is 66.9, median AP is 69.3 on val2	18	0.652003
NUIST	submission_1	15	0.608752
NUIST	submission_2	9	0.607124
Trimps-Soushen	Ensemble 2	8	0.61816
360+MCG-ICT-CAS_DET	9 models ensemble with validation and 2 iterations	4	0.615561
360+MCG-ICT-CAS_DET	Baseline: Faster R-CNN with Res200	4	0.590596
Hikvision	Best single model, mAP is 65.1 on val2	2	0.634003
CIL	Ensemble of 2 Models	1	0.553542
360+MCG-ICT-CAS_DET	9 models ensemble	0	0.613045
360+MCG-ICT-CAS_DET	3 models	0	0.605708

Ensemble of networks



Source https://flyyufelix.github.io/2017/04/16/kaggle-nature-conservancy.html

- **Ensemble learning** is the process of combining multiple independent classifiers together, in order to obtain a better performance.
- As long the individual classifiers do not make mistakes for the same examples, a simple majority vote might be enough to get better approximations.

Ensemble learning

- Let's consider we have three **independent** binary classifiers, each with an accuracy of 70% (P = 0.7 of being correct). When using a majority vote, we get the following cases:

    1. all three models are correct:

        P = 0.7 * 0.7 * 0.7 = 0.3492

    2. two models are correct

        P = (0.7 * 0.7 * 0.3) + (0.7 * 0.3 * 0.7) + (0.3 * 0.7 * 0.7) = 0.4409

    3. two models are wrong

        P = (0.3 * 0.3 * 0.7) + (0.3 * 0.7 * 0.3) + (0.7 * 0.3 * 0.3) = 0.189

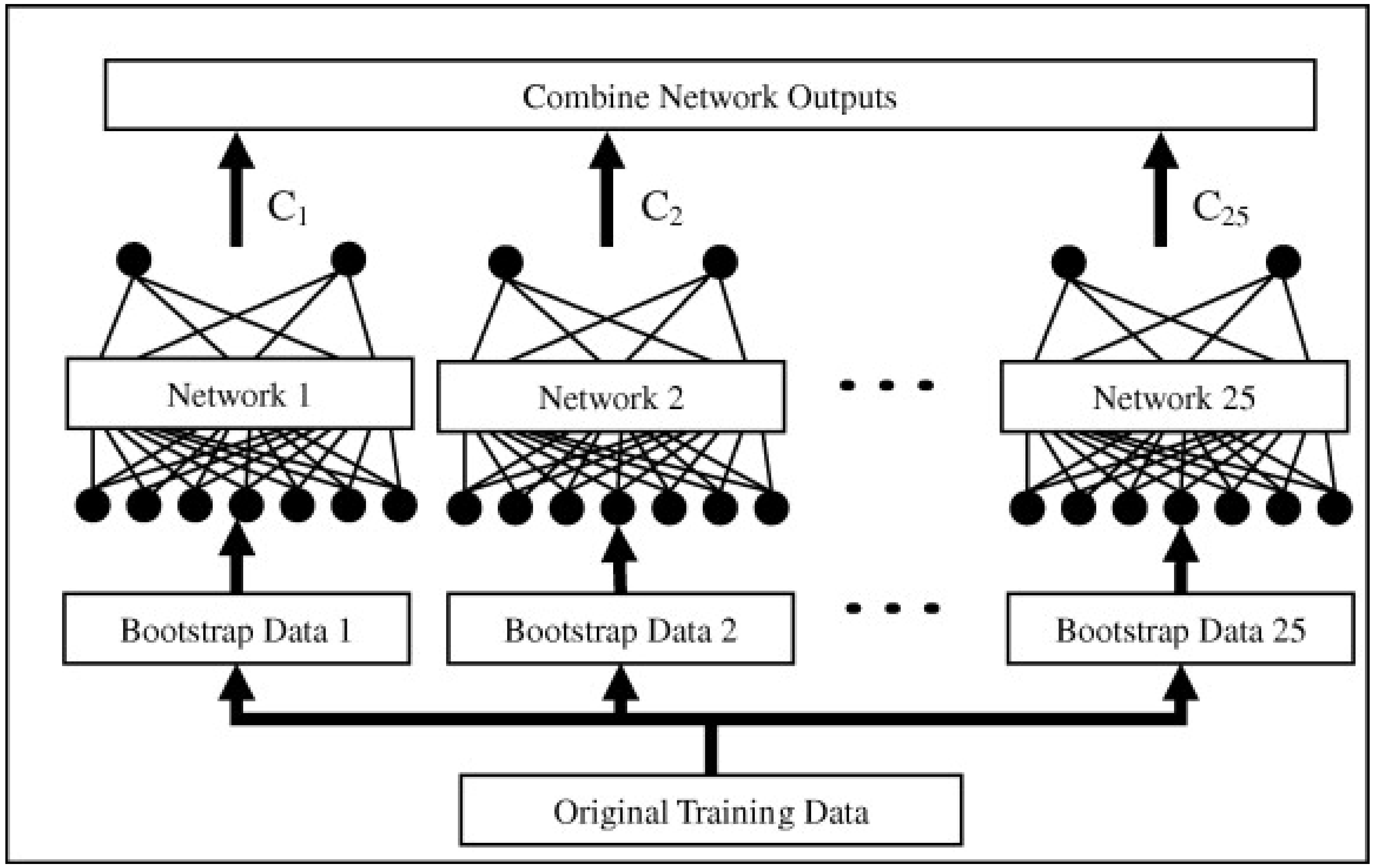
    4. all three models are wrong

        P = 0.3 * 0.3 * 0.3 = 0.027

- The majority vote is correct with a probability of P = 0.3492 + 0.4409 = **0.78 !**
- The individual learners only have to be slightly better than chance, but they **must** be as independent as possible.

Ensemble learning: bagging

- Bagging methods (bootstrap aggregation) trains multiple classifiers on randomly sampled subsets of the data.

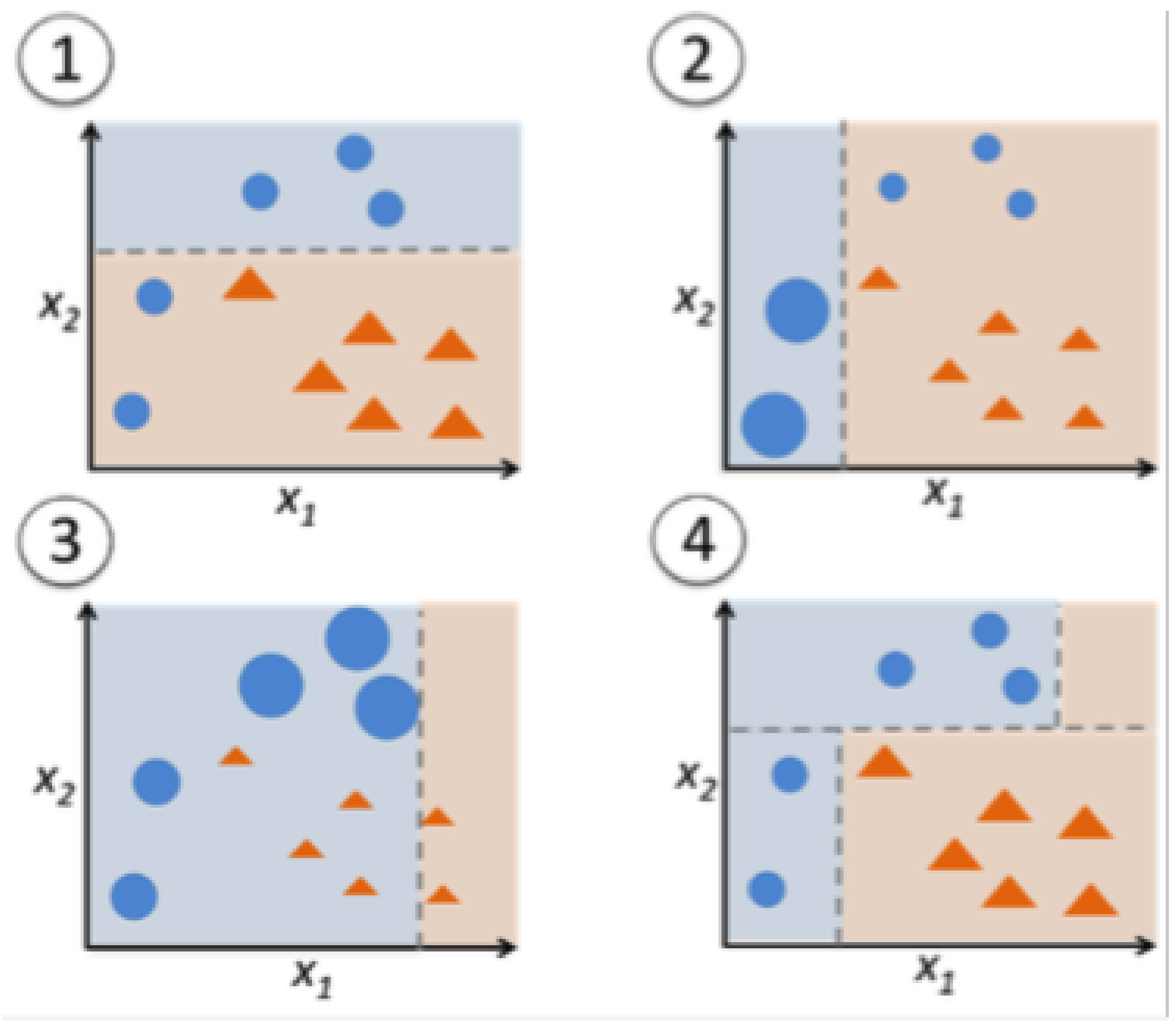


Source: http://www.sciencedirect.com/science/article/pii/S0957417409008781

- A **random forest** is a bagging method for decision trees, where the data and features are sampled..
- One can use majority vote, unweighted average, weighted average or even a meta-learner to form the final decision.

Ensemble learning: boosting

- **Bagging** algorithms aim to reduce the complexity of models that overfit the training data.
- **Boosting** is an approach to increase the complexity of models that suffer from high bias, that is, models that underfit the training data.
  - Algorithms: Adaboost, XGBoost (gradient boosting)…

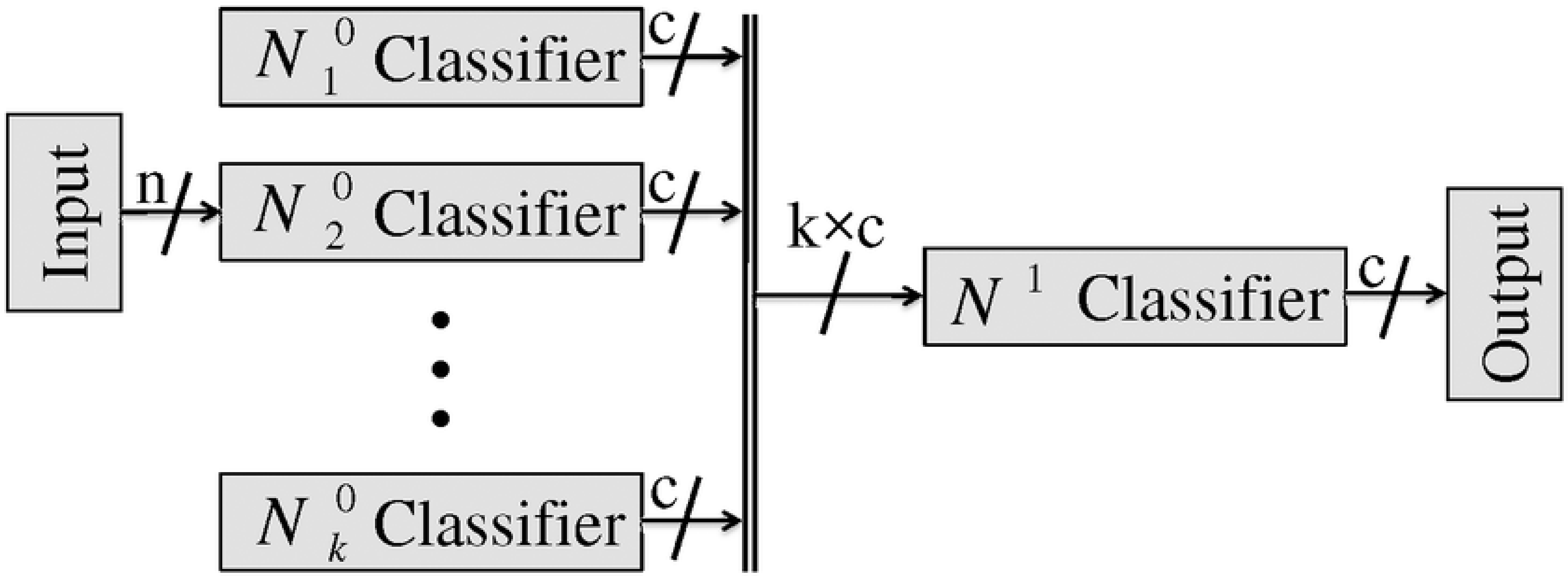


Source: https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/

- Not very useful with deep networks (overfitting), but there are some approaches like SelfieBoost (https://arxiv.org/pdf/1411.3436.pdf).

Ensemble learning: stacking

- **Stacking** is an ensemble learning technique that combines multiple models via a meta-classifier. The meta-model is trained on the outputs of the basic models as features.



Source: doi:10.1371/journal.pone.0024386.g005

- Winning approach of ImageNet 2016 and 2017.
- See https://blog.statsbot.co/ensemble-learning-d1dcd548e936