Neural Networks

Lecture 09: Deep Learning

http://www.informatik.uni-hamburg.de/WTM/

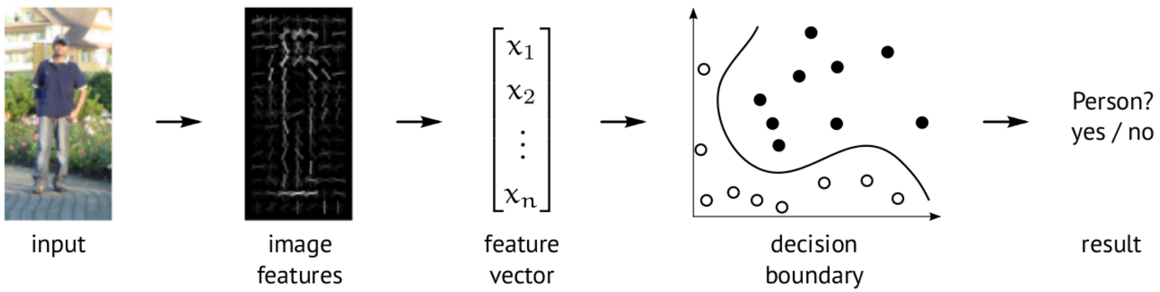
Feature Perception

- People are very good at recognizing patterns
- Our perceptual systems are very good at dealing with *invariances*
  - *translation, rotation, scaling*
  - *pose, deformation, occlusion*
  - *contrast, lighting, color, texture*

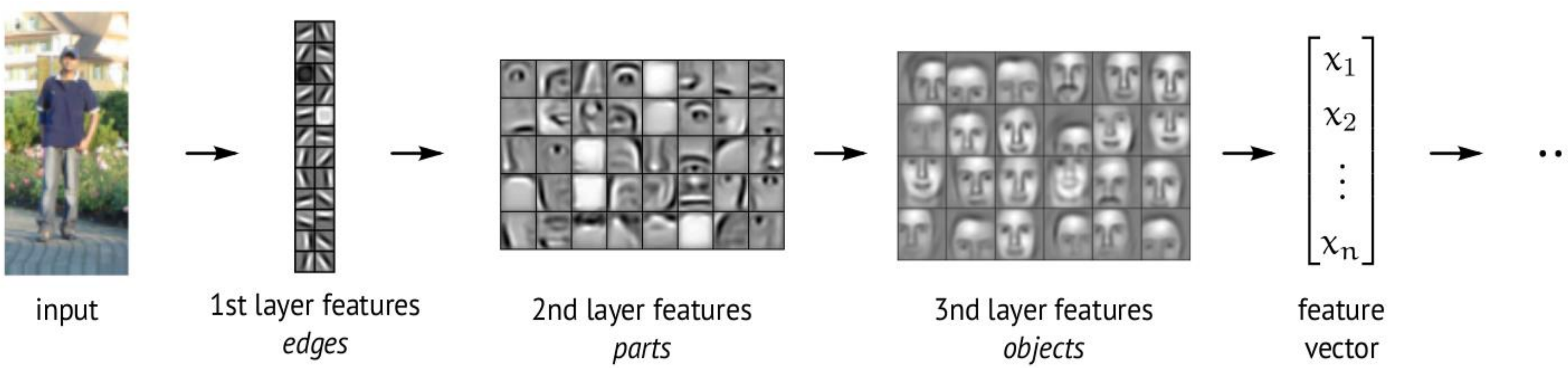
- However, it is too complex to model these systems explicitly (Rule based).
  - Deep Learning: Feature Learning

Feature Engineering vs Feature Learning

- Feature engineering
  - Classical computer vision



- Feature learning
  - Deep Learning



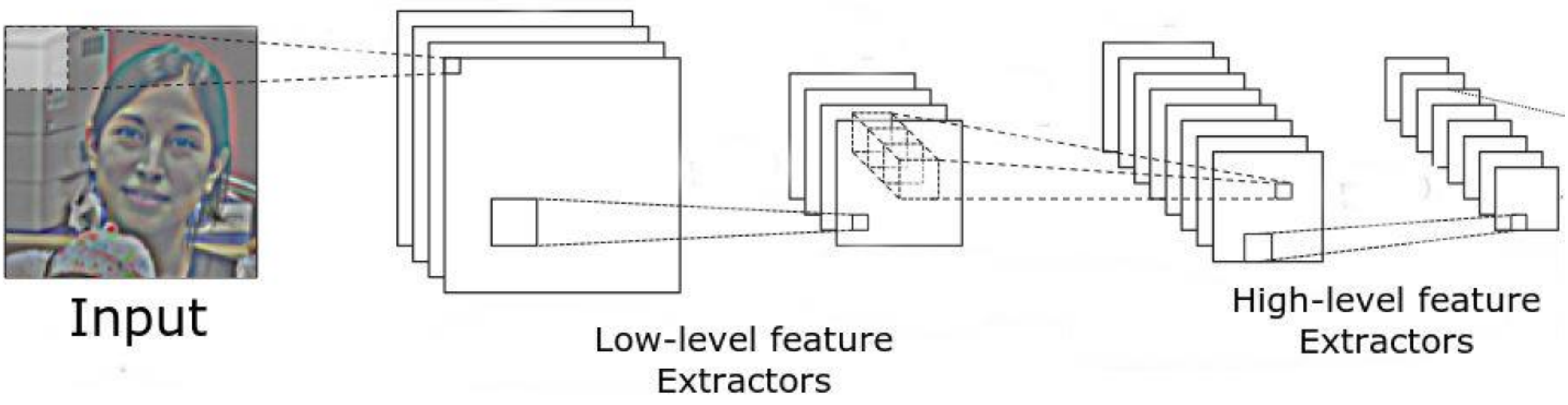
Feature Engineering vs Feature Learning

- Feature engineering
  - Domain-specific
  - Wide research field in the past decades
  - Many approaches and validated results
  - Performance stagnation in the past years

- Feature learning
  - Adapts to the domain
  - Learn specificity of the problem
  - Learning hierarchical features
  - Computationally expensive
  - Diverse applications and active research

Learning hierarchical features: Deep Learning

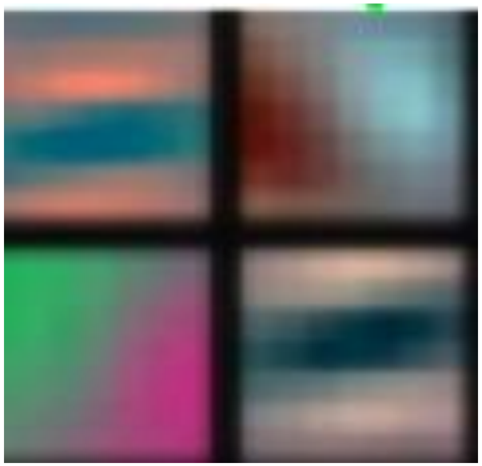
- Use of trained *feature extractors* to model data representation.
- Using sequences of layers to obtain *hierarchical data representation*: from low level features to high level of abstraction.



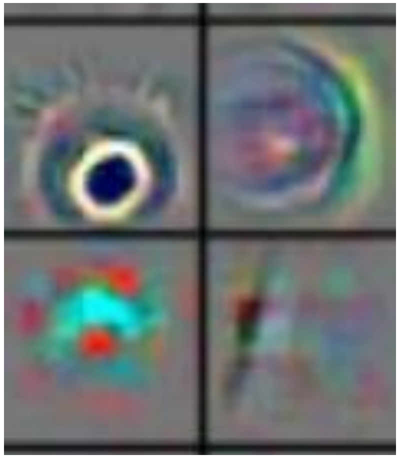
Hierarchical features

- Each layer increases the level of abstraction of the modeled data:
  - Image: pixel → edges → shapes → objects
  - Text: character → word → sentence → story
  - Speech: sound → phone → phoneme → word

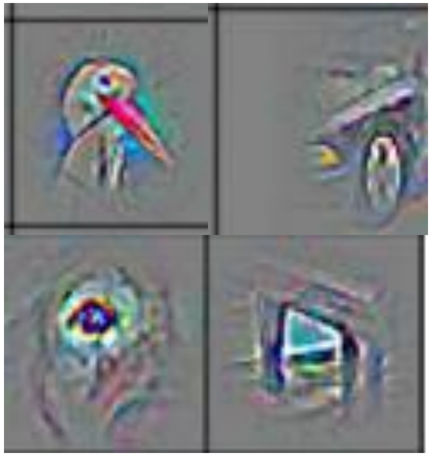
Edges



Shapes



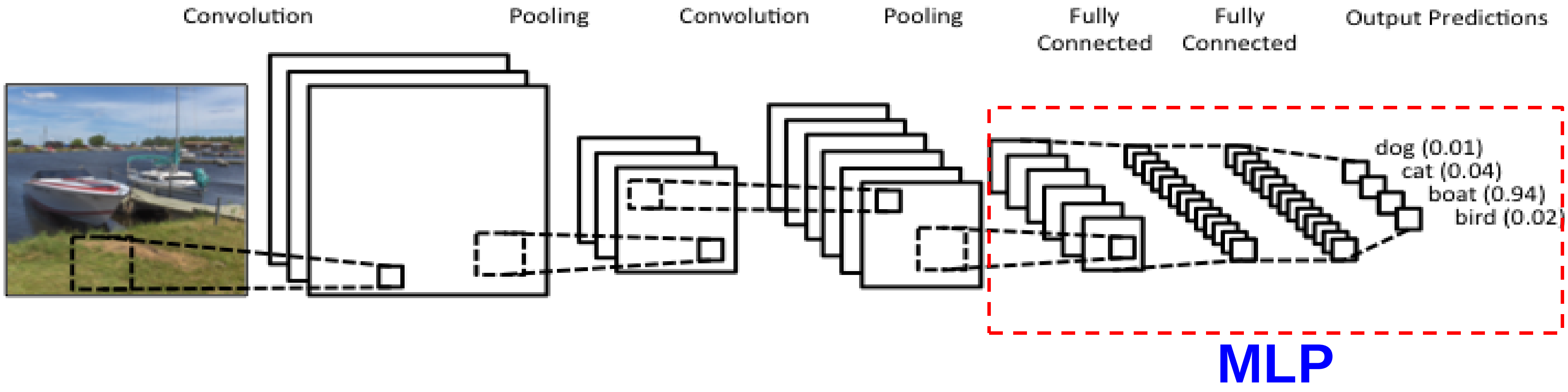
Objects



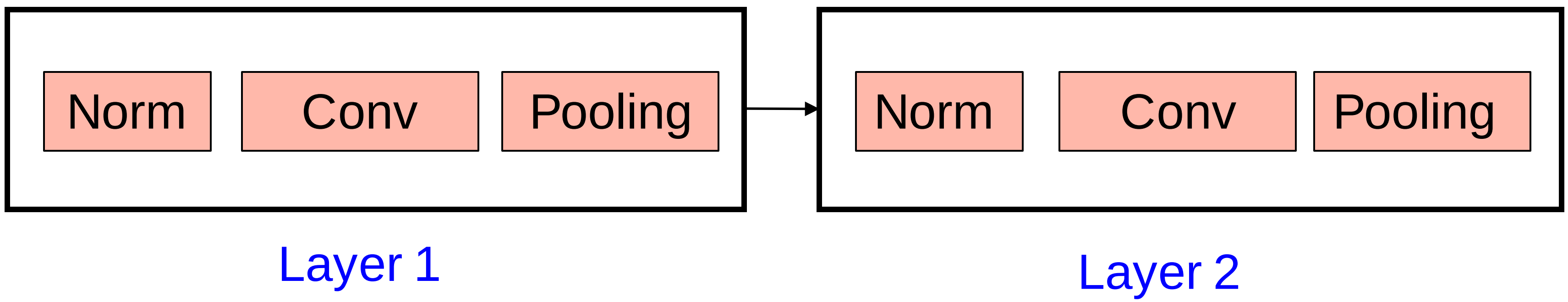
Convolutional Neural Networks (CNNs)

- Stack of convolution and pooling layers
  - Learn **filters** which code feature extractors
  - Increasing abstraction through layers

- Connected to a fully connected MLP



Convolutional Neural Networks (CNNs)



- Each layer is composed of:
  - **Normalization:** for outlier removal, high-pass filtering, smoothing over noise.
  - **Conv:** Filtering and activation function.
  - **Pooling:** dimensionality reduction.

Input Normalization

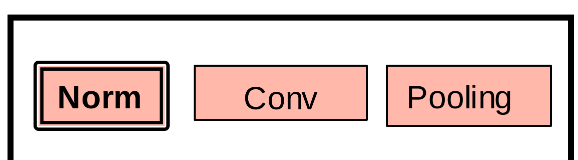
- Local contrast normalization (LCN)

  Subtraction of local mean followed by division with local standard deviation

Original images

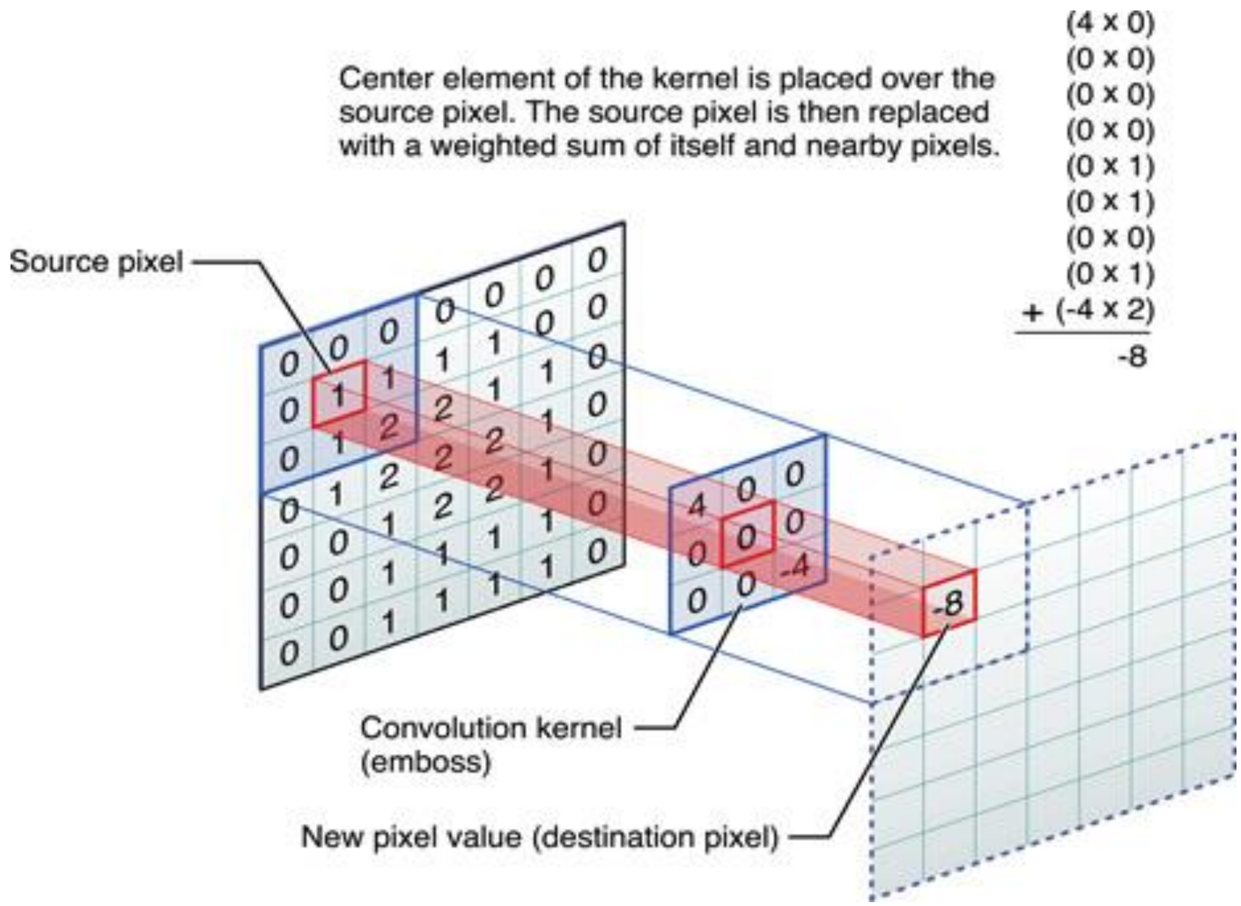
$$h_{x,y} = \frac{h_{x,y} - m_{N(x,y)}}{\sigma_{N(x,y)}}$$

LCN images



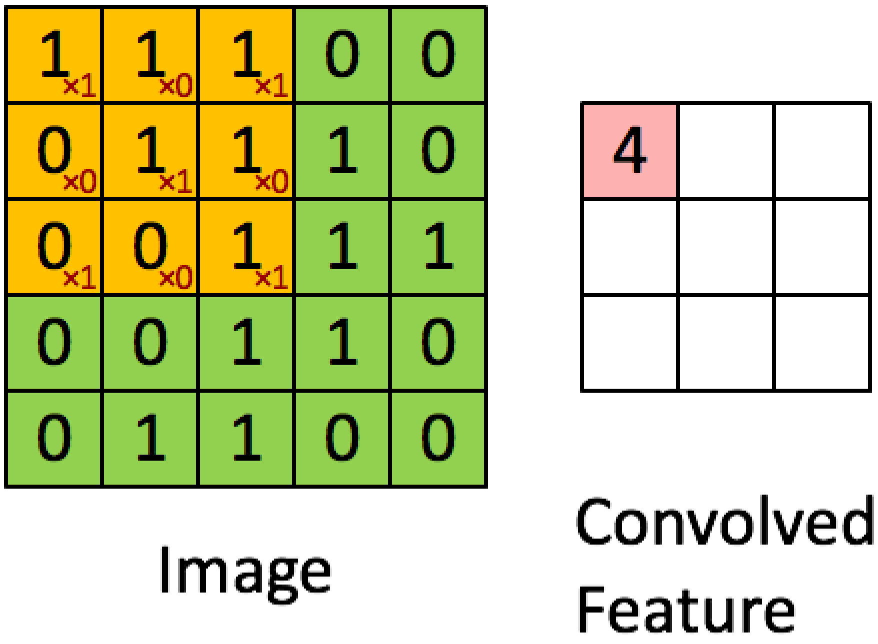
What is Convolution

- 2D convolution
  - Weighted moving sum



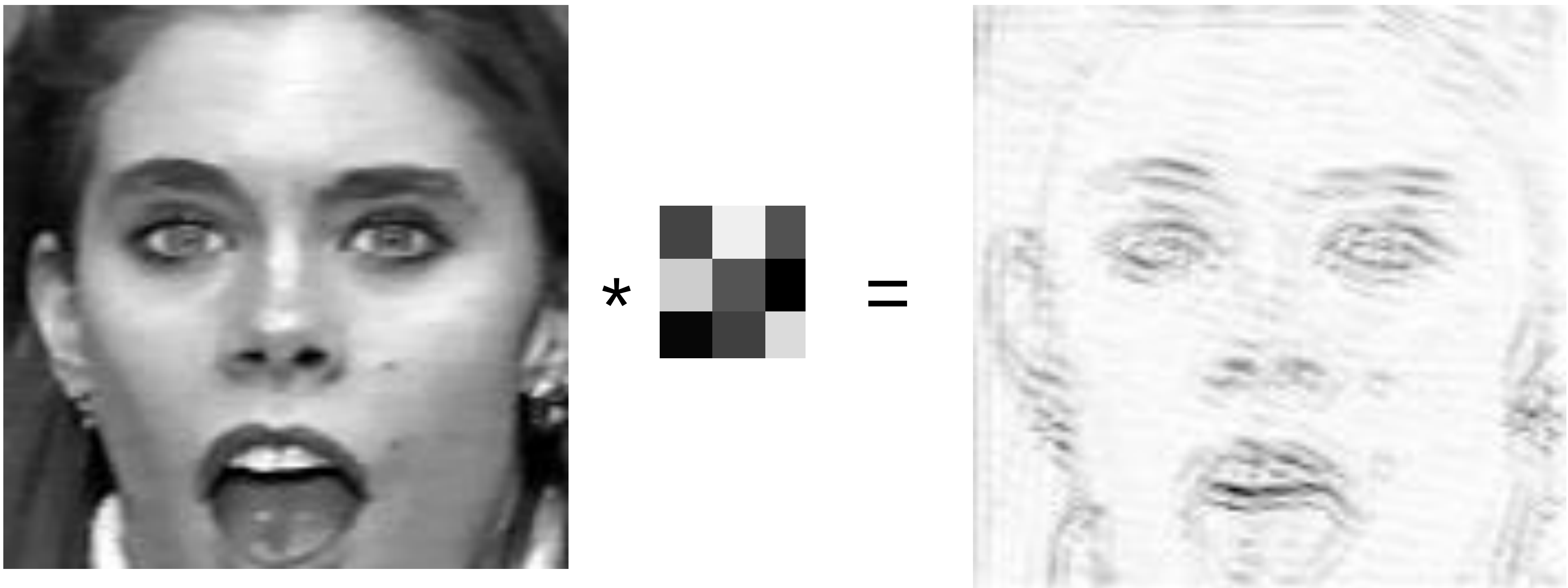
What is Convolution

- 2D convolution
  - Example



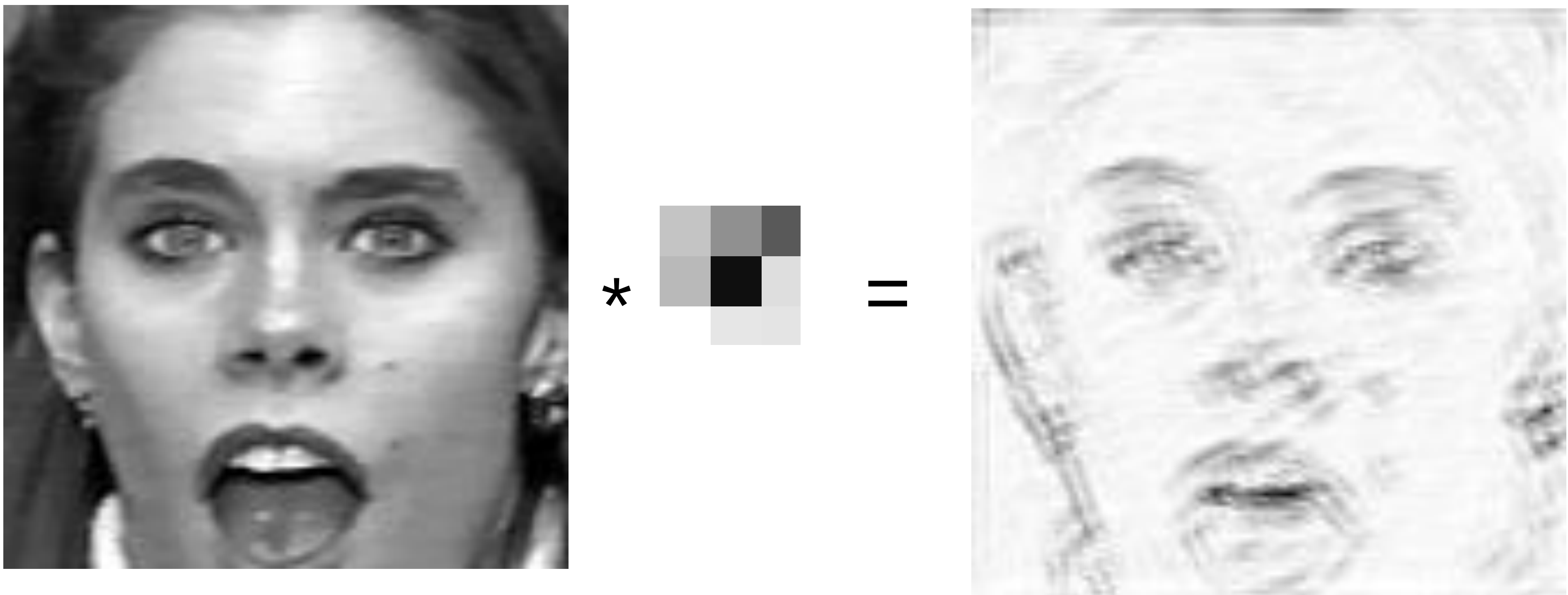
Convolution

- Filters are feature extractors



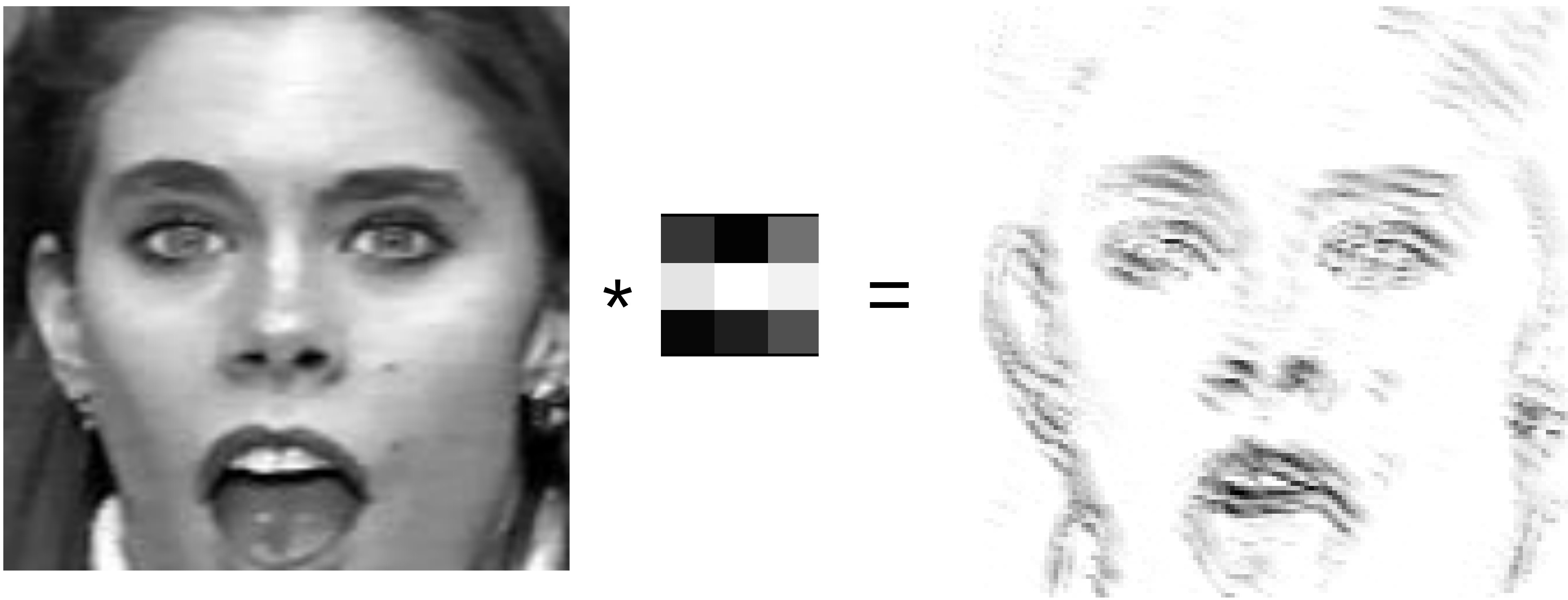
Convolution

- Filters are feature extractors



Convolution

- Filters are feature extractors



Convolutional Layers

- Several different filters per layer
  - *Feature redundancy strategy* to learn different features
- Each filter is applied to the whole image
  - Weights are shared for all image locations
- Captures *spatial structure* of the input
- Use *local weight connections*, instead of fully connected layers

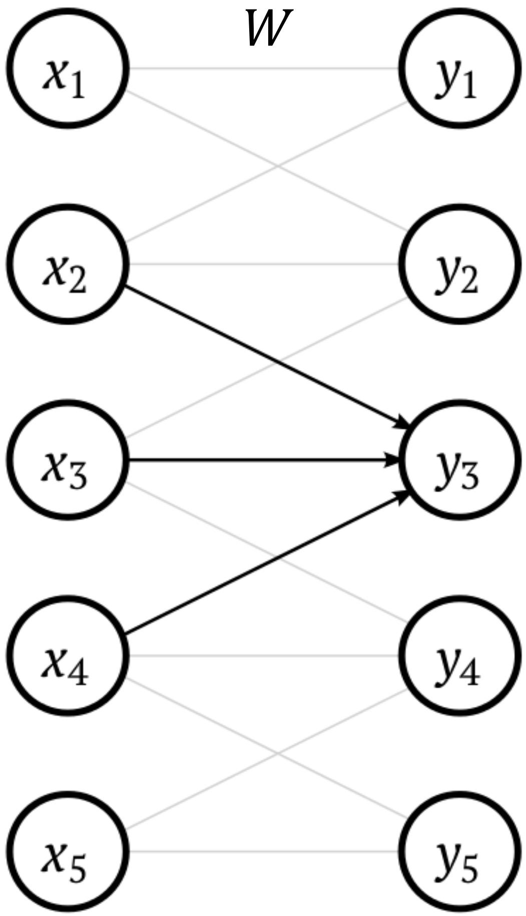
Convolutional Layers

- **x** = input signal
- **y** = filter map
- **W** = kernel (filter)

- Sliding the filters over the input signal

- y_i is a dot product of the local neighborhood and the filter weights

- Apply an activation function on each filter map

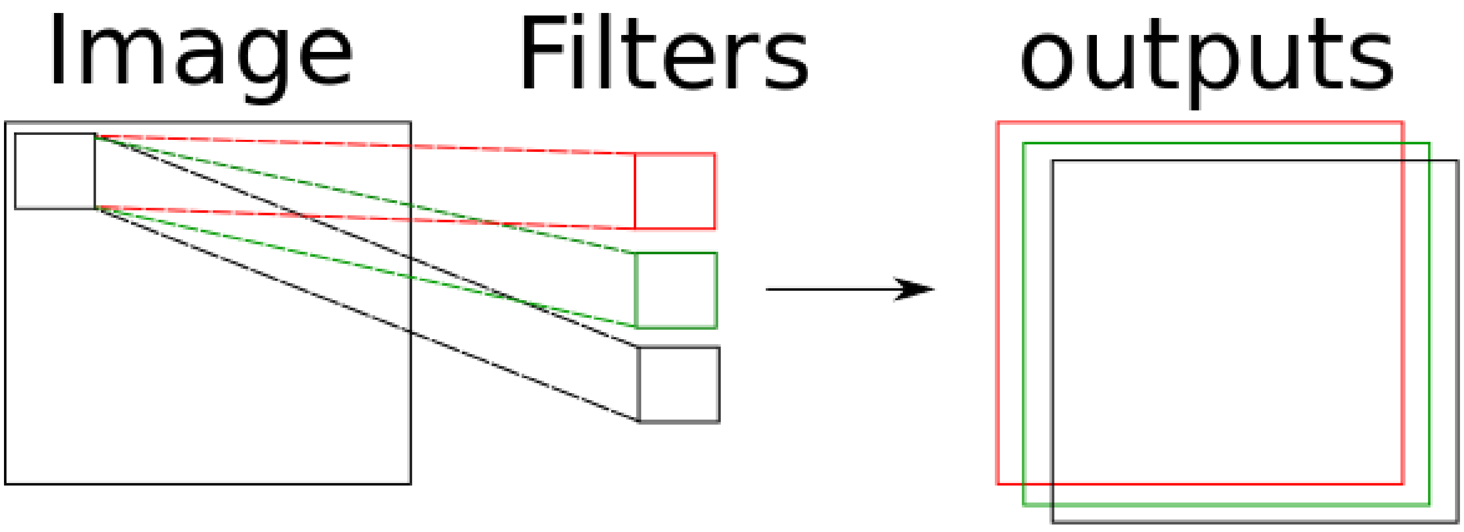


Convolutional Layers

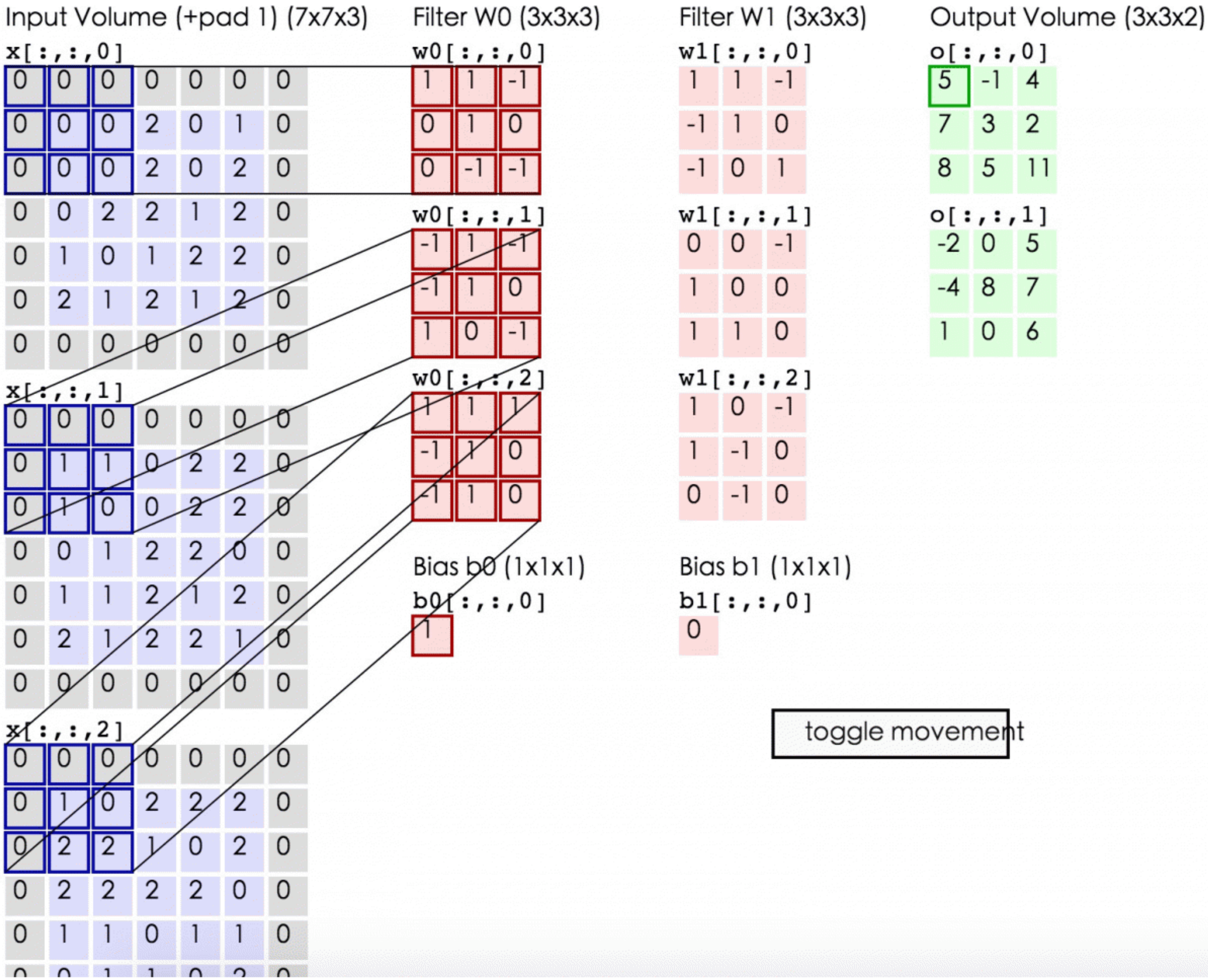
- **2D convolution**

$$Conv = \sum_{m=1}^{M} \sum_{h=1}^{H} \sum_{w=1}^{W} w_{(c-1)m}^{hw} u_{(m-1)}^{(h)(w)}$$

- Convolve the filters (M) weights (w) with the input (u) using the specified window (h,w)



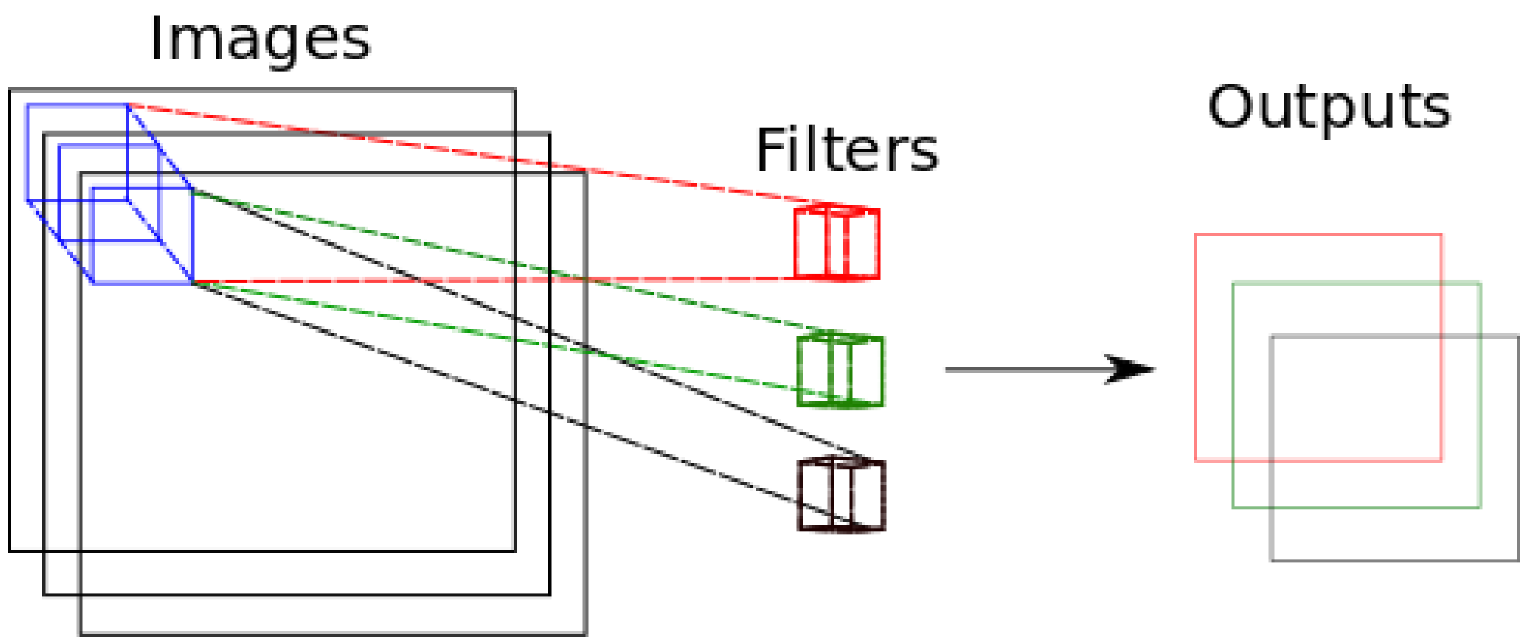
Convolutional Layers



Convolutional Layers

- **3D convolution**
- Extract spatial-temporal structures
- Apply a 3D kernel (h,w,z) over a stack of images
- Slide the kernel in all three dimensions

$$Conv3D = \sum_{m=1}^{M} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{z=1}^{Z} w_{(c-1)m}^{hwz} u_{(m-1)}^{hwz}$$



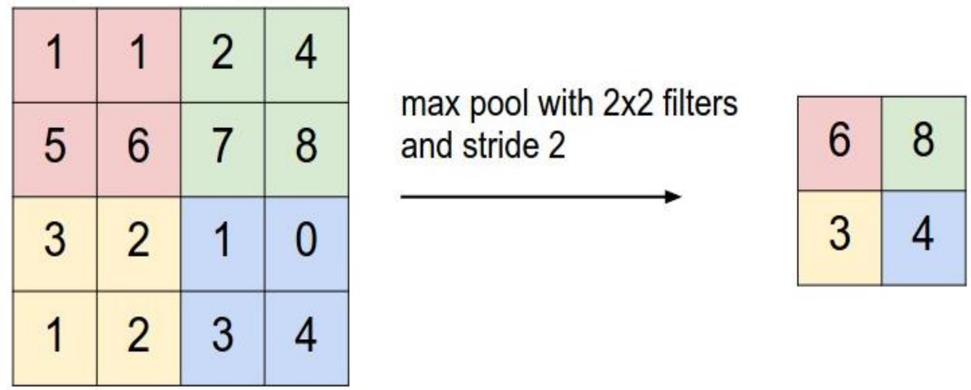
Pooling Layer

- Reduce dimensionality by *subsampling* over a window

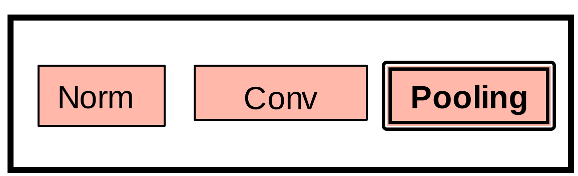
- Applied on each kernel's (filter's) output

- *Translation invariance*

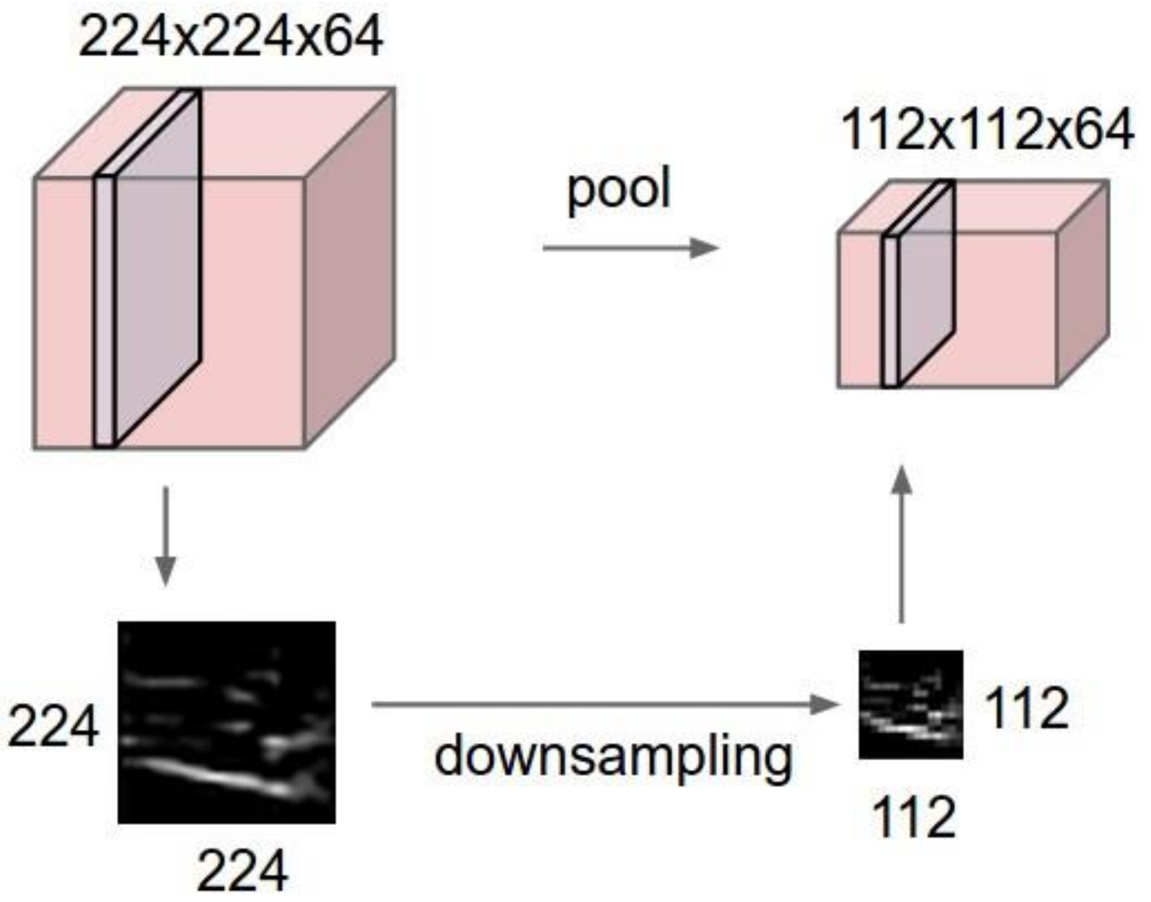
- Usually max or average pooling



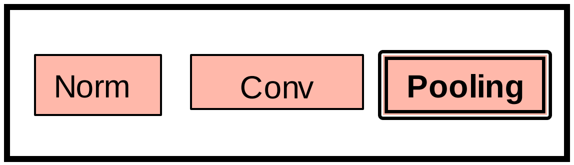
http://cs231n.github.io/convolutional-networks/



Convolution and Pooling Layers



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



Fully Connected MLP

- The outputs of the last convolutional layer are *flattened*
- The resulting feature vector is used as input to a hidden layer
- Usually uses a logistic regression classifier (*softmax*)
  - The *softmax* function squashes the logistic function's k-dimensional output of real values into a vector of probabilities per class, that can be added up to 1.

Training CNNs

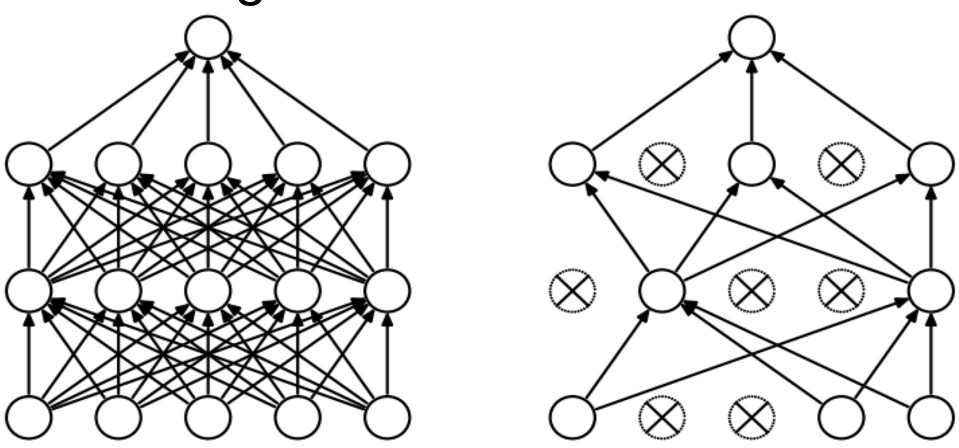
- Use backprop to train Convolution Neural Networks
  - Calculate the gradient of the filters using the *transpose of the filters* (flipping the kernels)

- Usually trained with huge amount of data

- Tendency to *overfit*
  - Due to not enough data and/or high number of parameters
  - Regularizations are necessary in most of the cases

Training CNNs

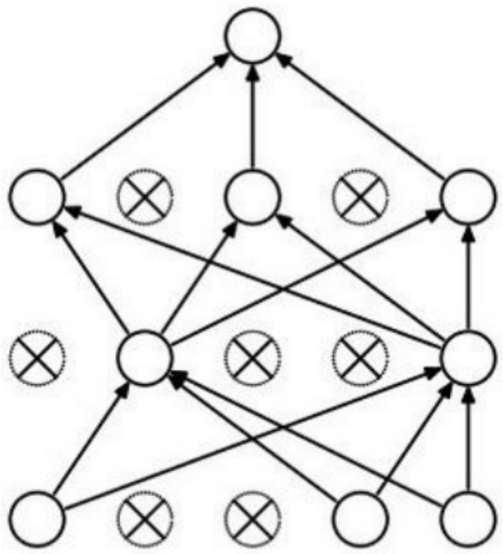
- Batch Normalization
  - Normalizes the activations of the previous layer at each batch. Mean close to 0, standard deviation close to 1.
- Dropout regularization
  - Prevents network overfitting.
  - Temporarily remove the unit and its weights from the network
    - 50% chance of removal for one training epoch.
  - Improves network generalization.



(a) Standard Neural Net (b) After applying dropout.

Dropout

- Forces network to have a redundant representation.



- Why is this a good idea?
  - Another interpretation: Dropout is training a large ensemble of models (that share parameters)
  - Each binary mask is one model
- What do we do about the randomness at test time?
  - During testing all neurons are always active
  - Scale activations so that outputs are **equal** during testing and training

Training CNNs

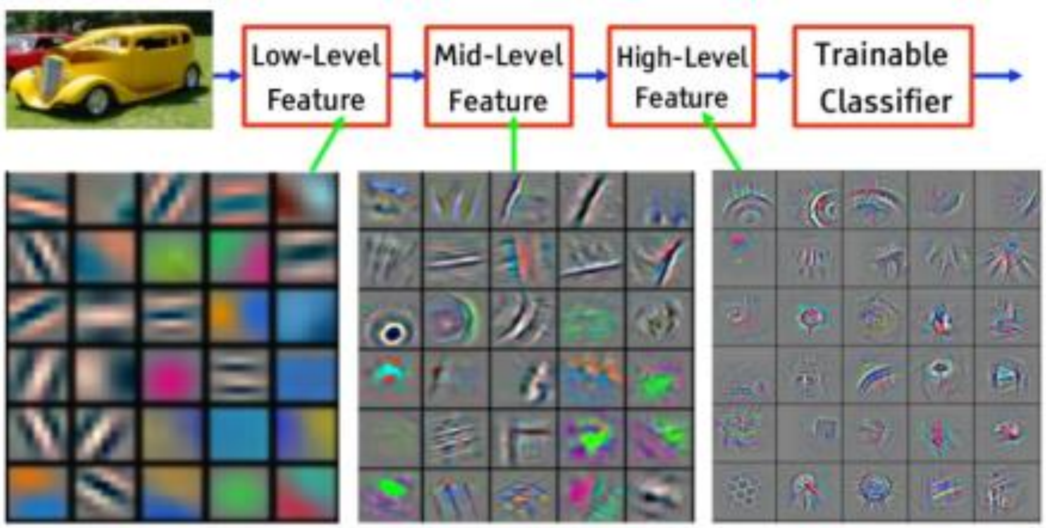
- Shuffle the training samples
- Use *batch learning* and not *online learning*
  - Update the weights of the kernels after a batch (a small part) of the training set is presented

- Calculates the gradient against the samples of each batch, instead of one sample per time

- Usually results in a smoother convergence
  - The gradient computed for each training step uses more training samples, thus the summed gradient is more detailed

Understanding and Visualizing CNNs

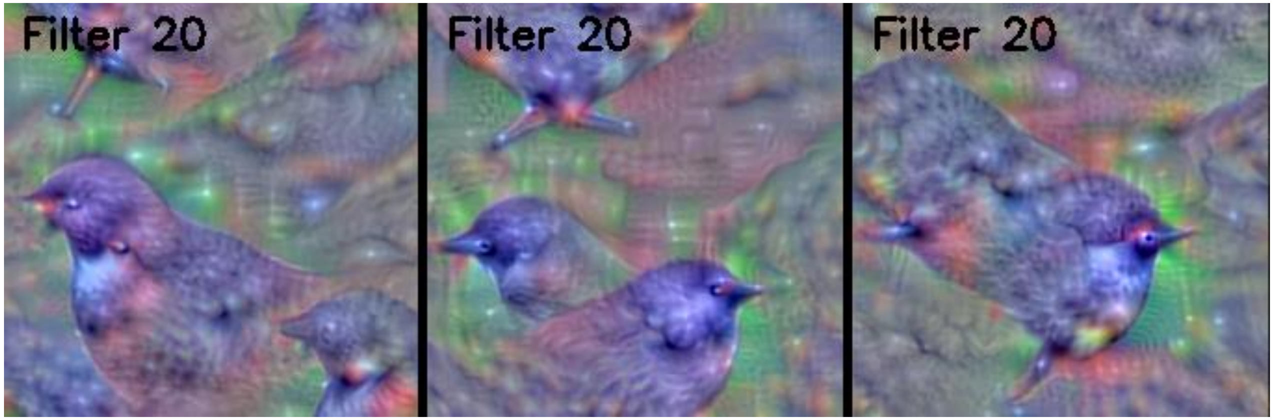
- Neural networks are black boxes
  - Hard to understand what they learn
  - Difficult to evaluate the quality of the learned representations
- How to have an insight on what the network learned?
  - Different visualization techniques
  - Different knowledge



Filters visualization

- Transform the learned representations into high-abstraction information
  - Helpful way to subjectively analyze the effect of different parameters change and regularizations.
- Send a random image (x) to the network and backpropagate the gradient ($\partial a_i(x)/\partial x$) of the activation of specific filter (a_i) to update the input image.

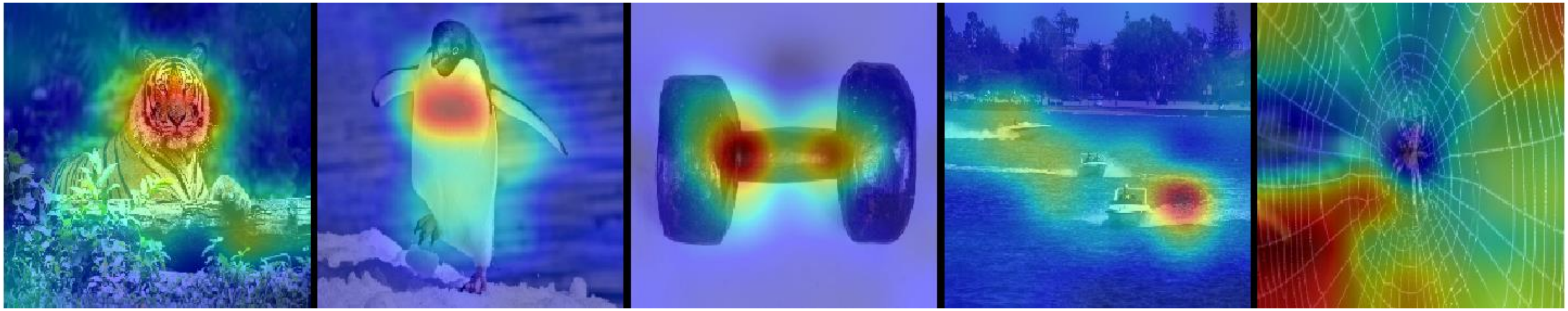
$$x \leftarrow x + \partial a_i(x)/\partial x$$



Attention maps

- Identify regions of the stimuli which activate specific network regions
  - Very popular method for attention-based networks
- Send a selected input image (i) to the network and backpropagate the gradient ($\partial a_i(i)/\partial x$) of the activation of a specific filter (a_i) to generate a heatmap of the region that neuron activates to.

$$i \leftarrow \partial a_i(i)/\partial x$$



Adversarial Learning

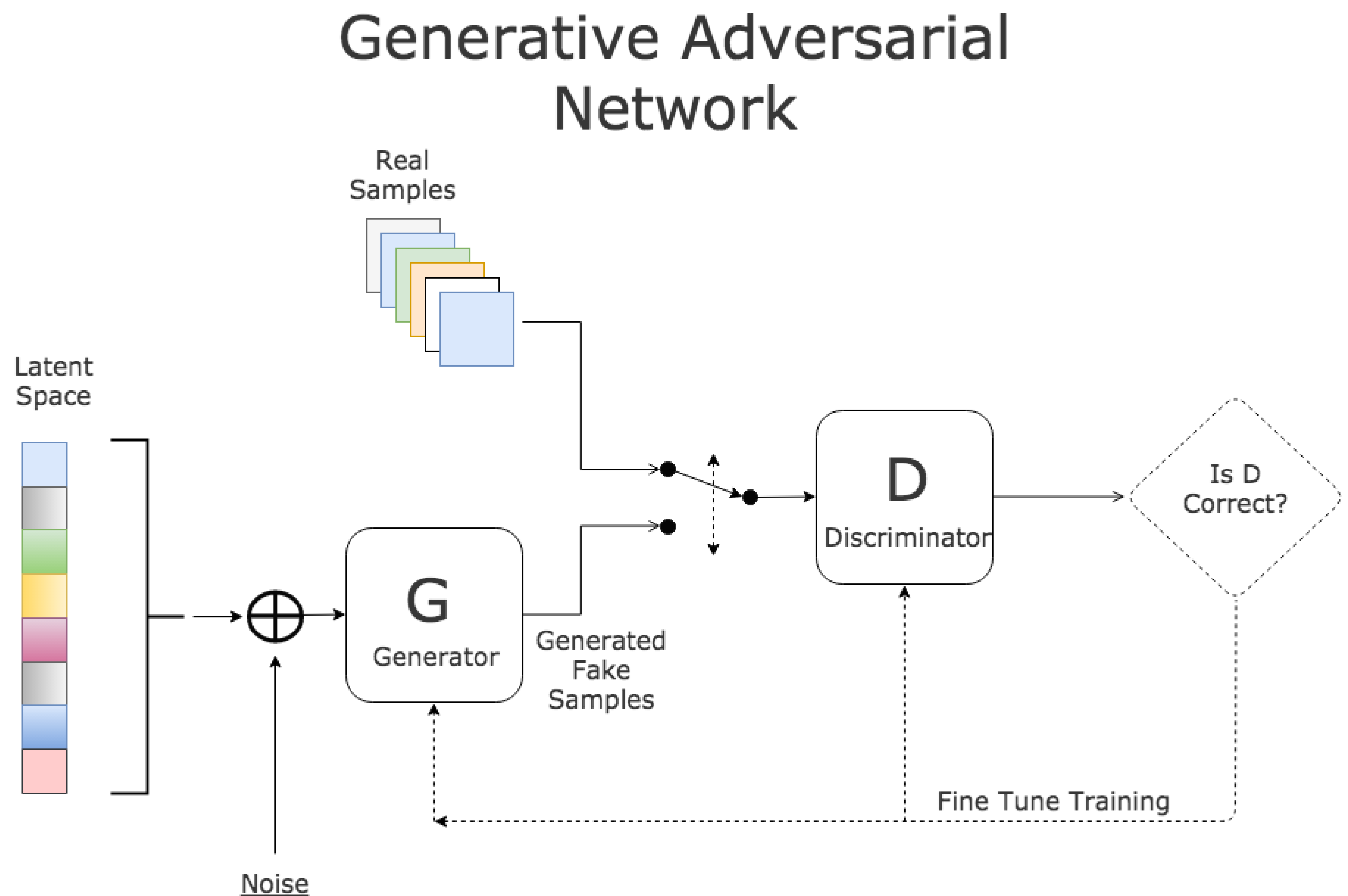
- Train a neural network using an unsupervised method
  - Adversarial Generative Neural Networks (GANs)

- Adversarial training
  - Two networks: generator (G) and discriminator (D)
  - Generator: learns to generate data from noise (z)
  - Discriminator: learns to discriminate real data (r) from generated data.
  - Error function (E) : is the generated image real (t)?

$$E = D\big(r, G(z)\big) - t$$

Unsupervised Learning

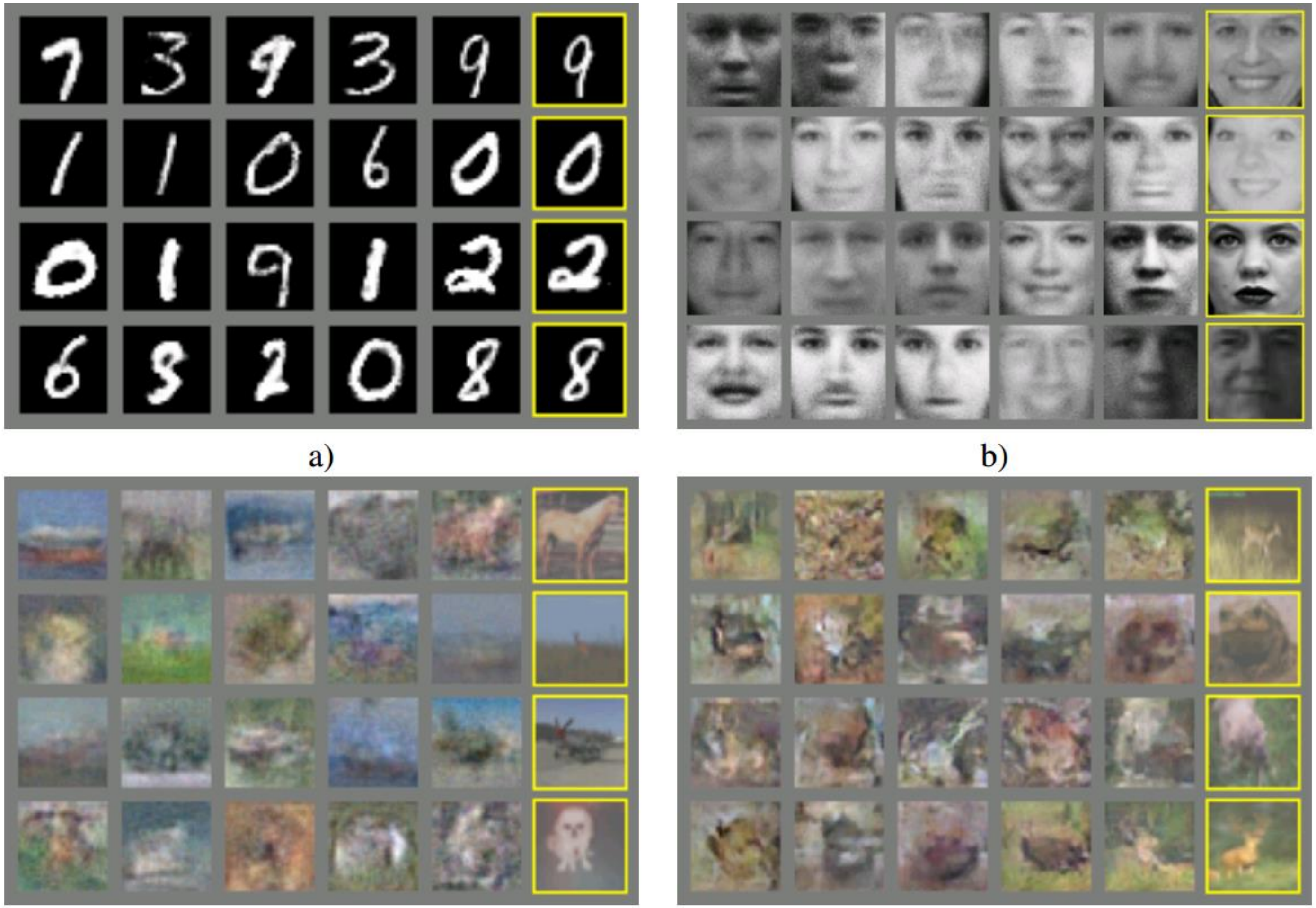
- Adversarial Training



Radford et al. 2015

Unsupervised Learning

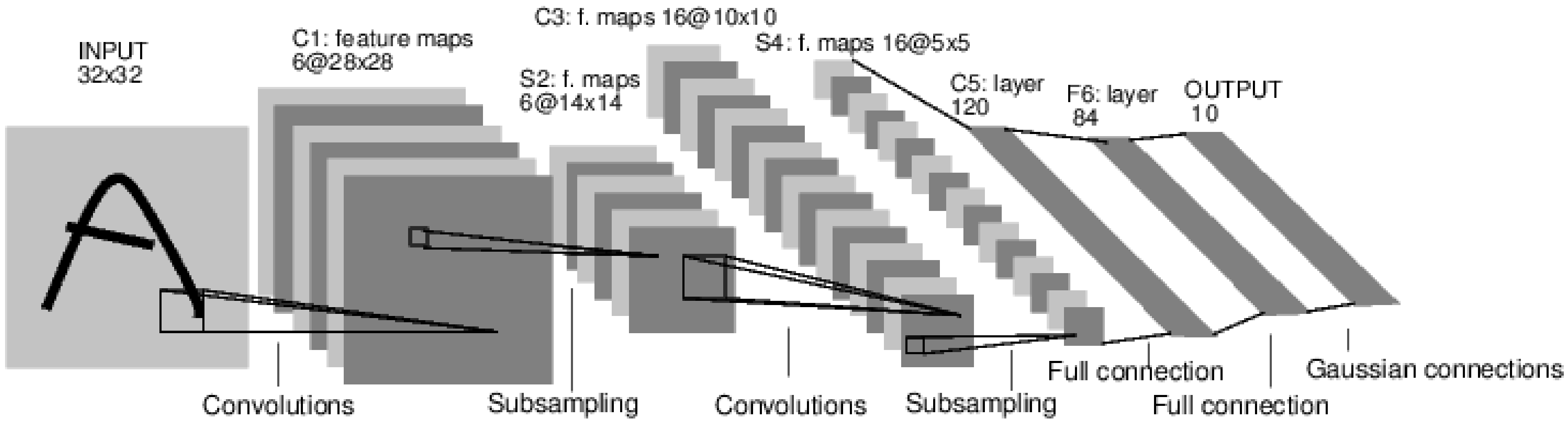
- Generated Images



Goodfellow et al. 2014

Early CNN architectures

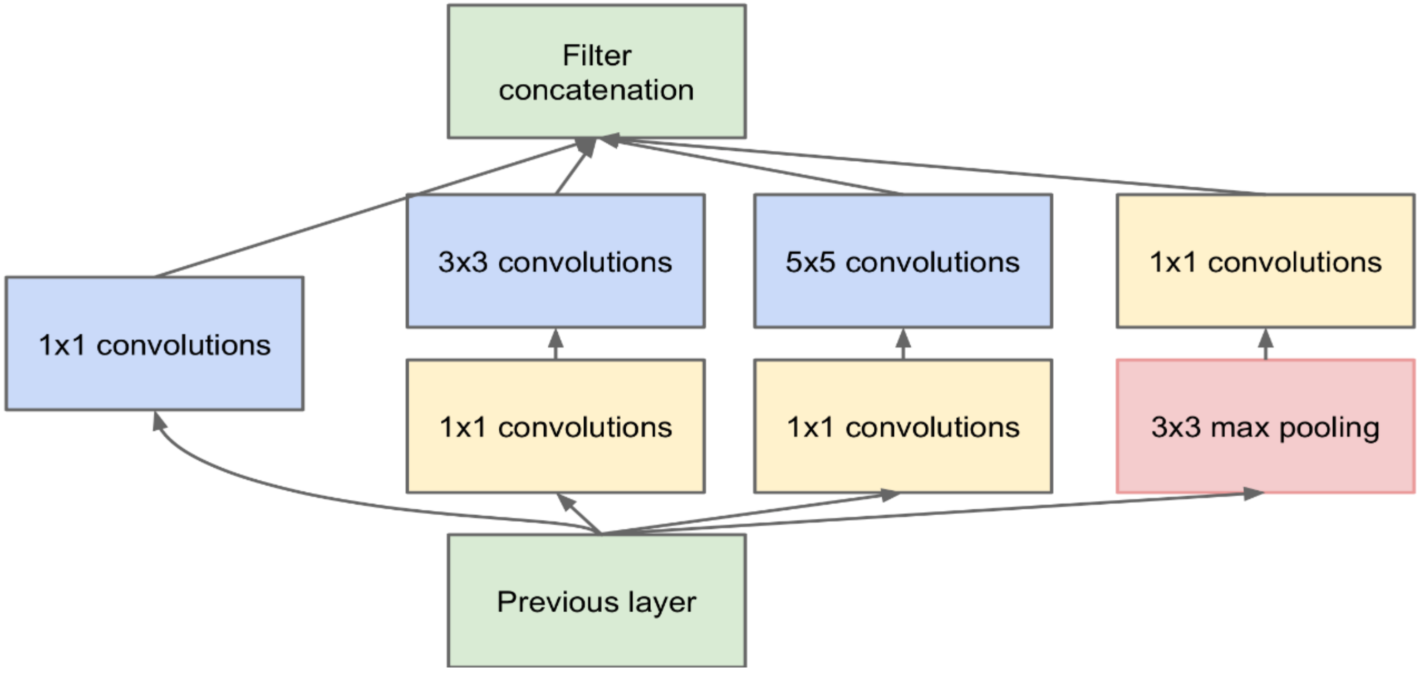
- LeNet-5
  - 2 Conv layers, connected with 2 max-pooling layers
  - 2 fully connected hidden layers
  - Applied to character and handwriting recognition



LeCun et al. 1998

Recent CNN architectures

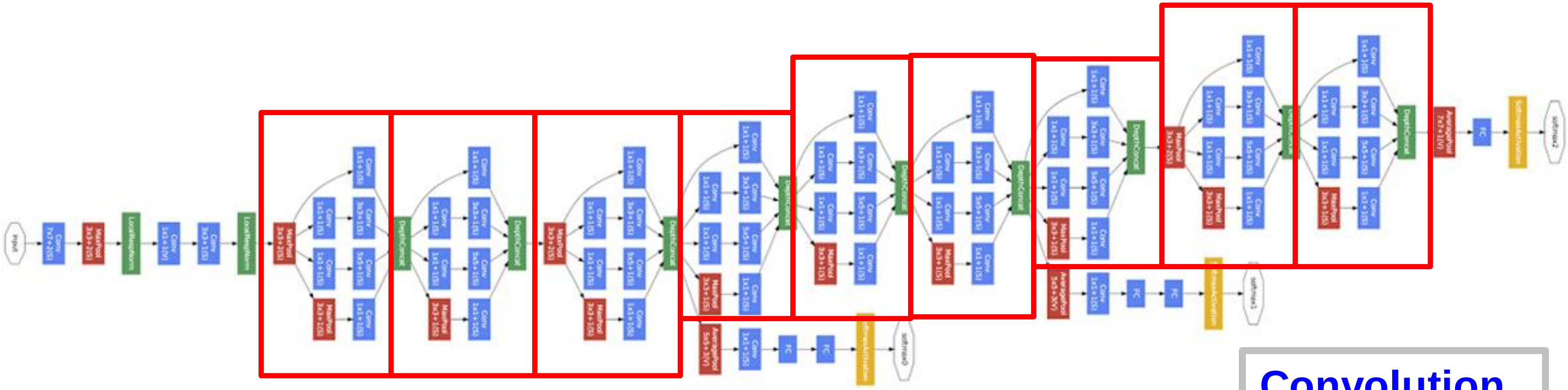
- GoogleLeNet – ImageNet challenge
  - Inception layer: All conv. layers have kernels of different size which are concatenated



Szegedy et al. 2014

Recent CNN architectures

Inception Layers

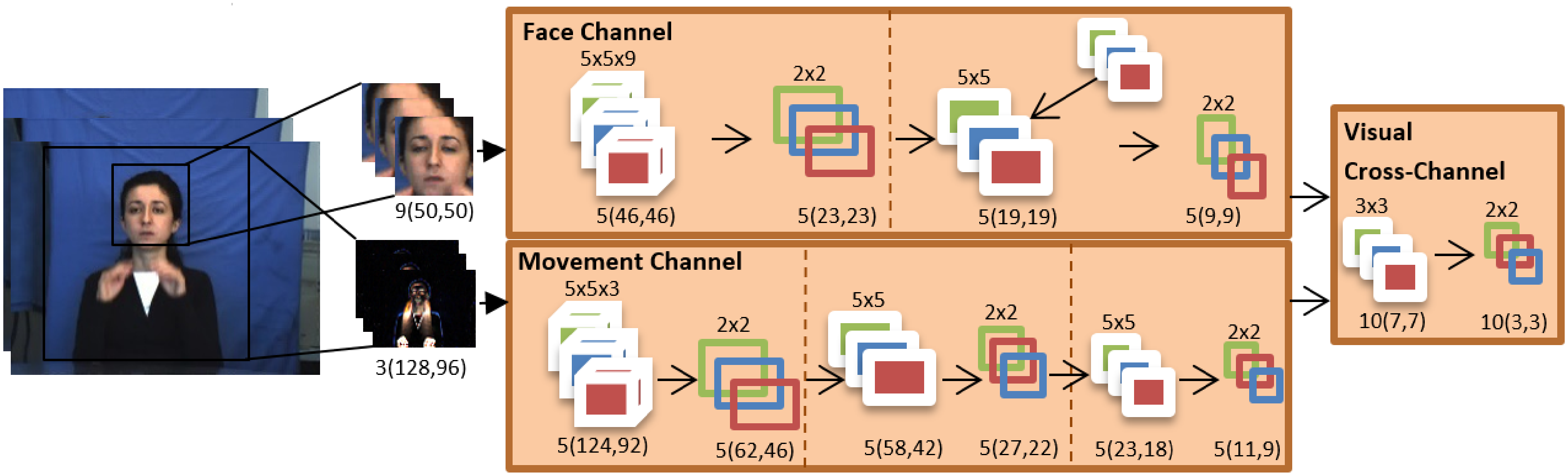


Convolution
Pooling
Softmax
Other

Szegedy et al. 2014

Multimodal Processing

- Cross Channel Convolutional Neural Network (CCCNN)
- Applied to emotion expression recognition
  - One stream dealing with face expression and the other with body posture.
- Use shunting inhibition to force strong features in deeper layers



Barros et al. 2015

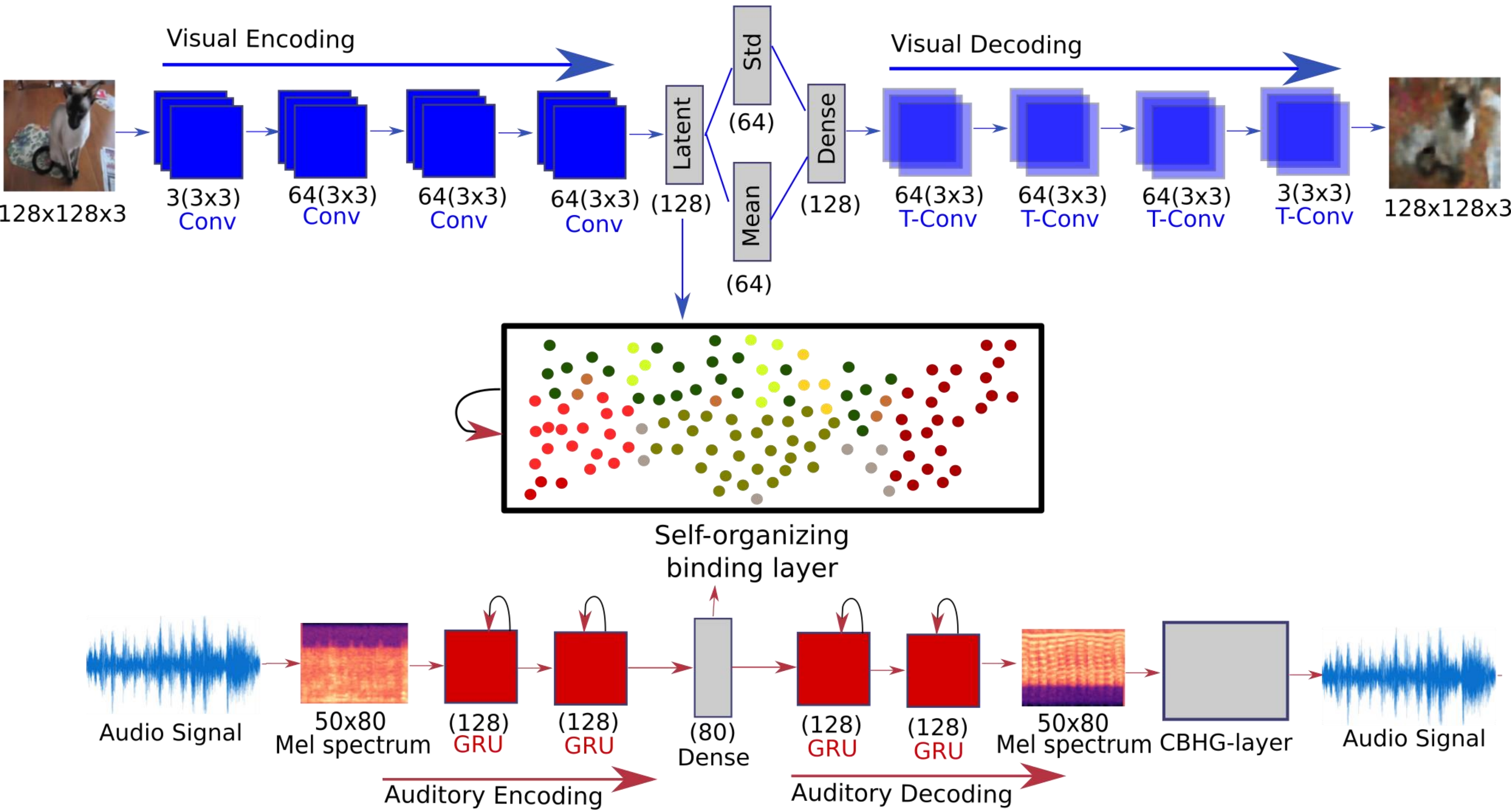
Expectation Learning

- Unsupervised learning of crossmodal representations
  - I see a dog, I expect it to bark.

- Crossmodal convolution autoencoders
  - Encoding and decoding high-level information

- Self-organizing binding layer
  - Bind dog with barking

Expectation Learning



Barros et al. 2018

Expectation Learning

- Perceived vision, generated audio

- Perceived audio, generated vision

Summary

- *Learned features* adapt to domain and provide solutions for very hard tasks
- CNNs are composed of a stack of *convolutional operations* and *pooling layers*, extracting specific structures in each layer and compressing information.
- Usually used: *backpropagation to train the network*. Some regularization techniques are necessary to avoid overfitting
- Several architectures were and still are proposed, it is an open and active research field.

Next lecture:

Training Neural Networks – Best practices and Frameworks

References

[LeCun, 1998] Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P., "*Gradient-based learning applied to document recognition*," in Proceedings of the IEEE , vol.86, no.11, pp.2278-2324, 1998

[Krizhevsky, 2012] A. Krizhevsky, I. Sutskever, and G. Hinton, *“ImageNet Classification with Deep Convolutional Neural Networks”*, NIPS 2012

[Barros, 2015] Barros, Pablo, et al. *"Multimodal emotional state recognition using sequence-dependent deep hierarchical features." Neural Networks* 72 (2015): 140-151.

[Szegedy, 2014], W Liu, Y Jia, P Sermanet, S Reed, D Anguelov, D Erhan, “*Going deeper with convolutions*”. C Szegedy. arXiv preprint arXiv:1409.4842

Call for Experiments

- Want to help us to develop the next generation deep learning models?
- Take part in our human behaviour experiments!
- Send an email to: barros@informatik.uni-hamburg.de

b)

(c)