OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

INFORMATIK UND MATHEMATIK

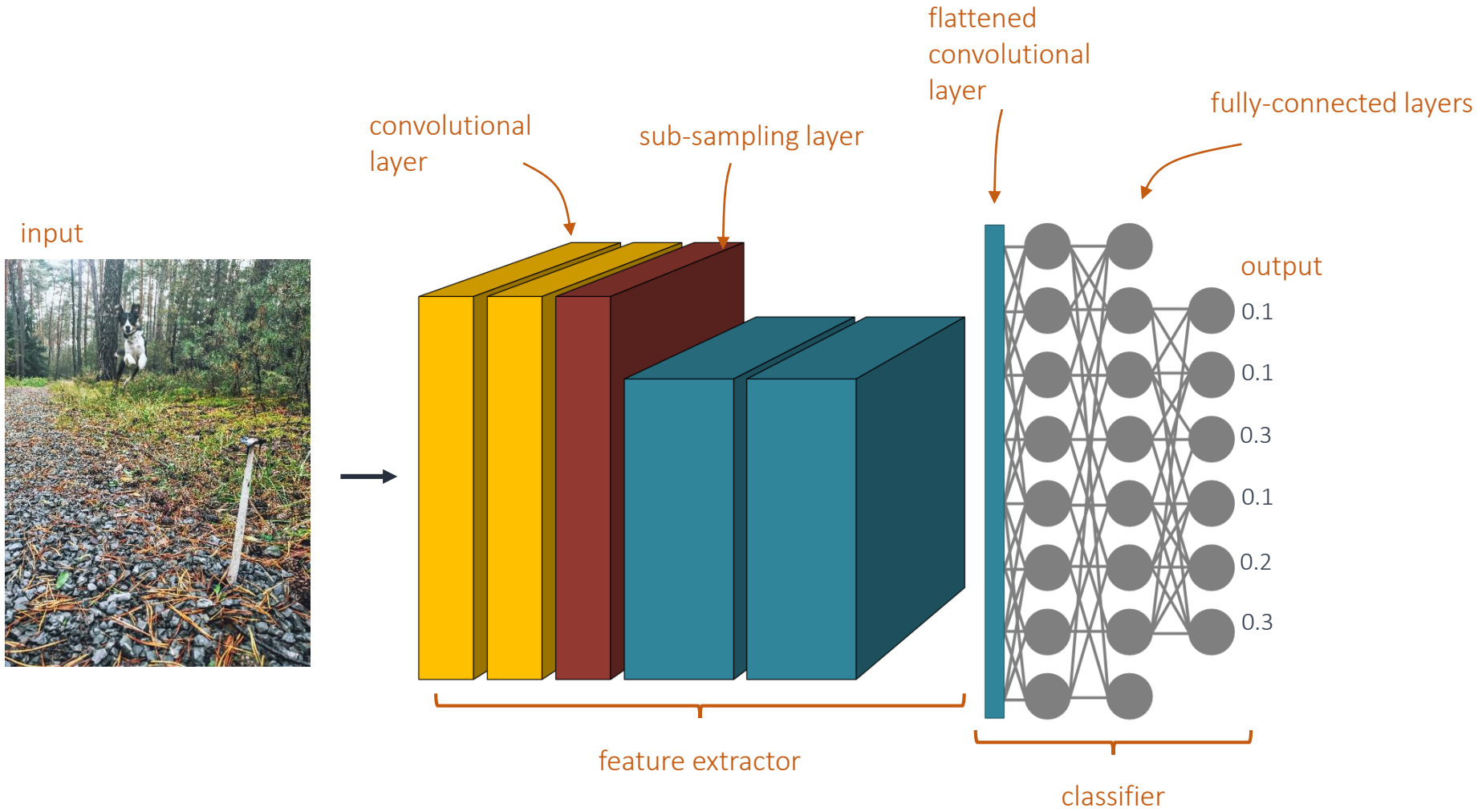
ams OSRAM

Machine Learning Research

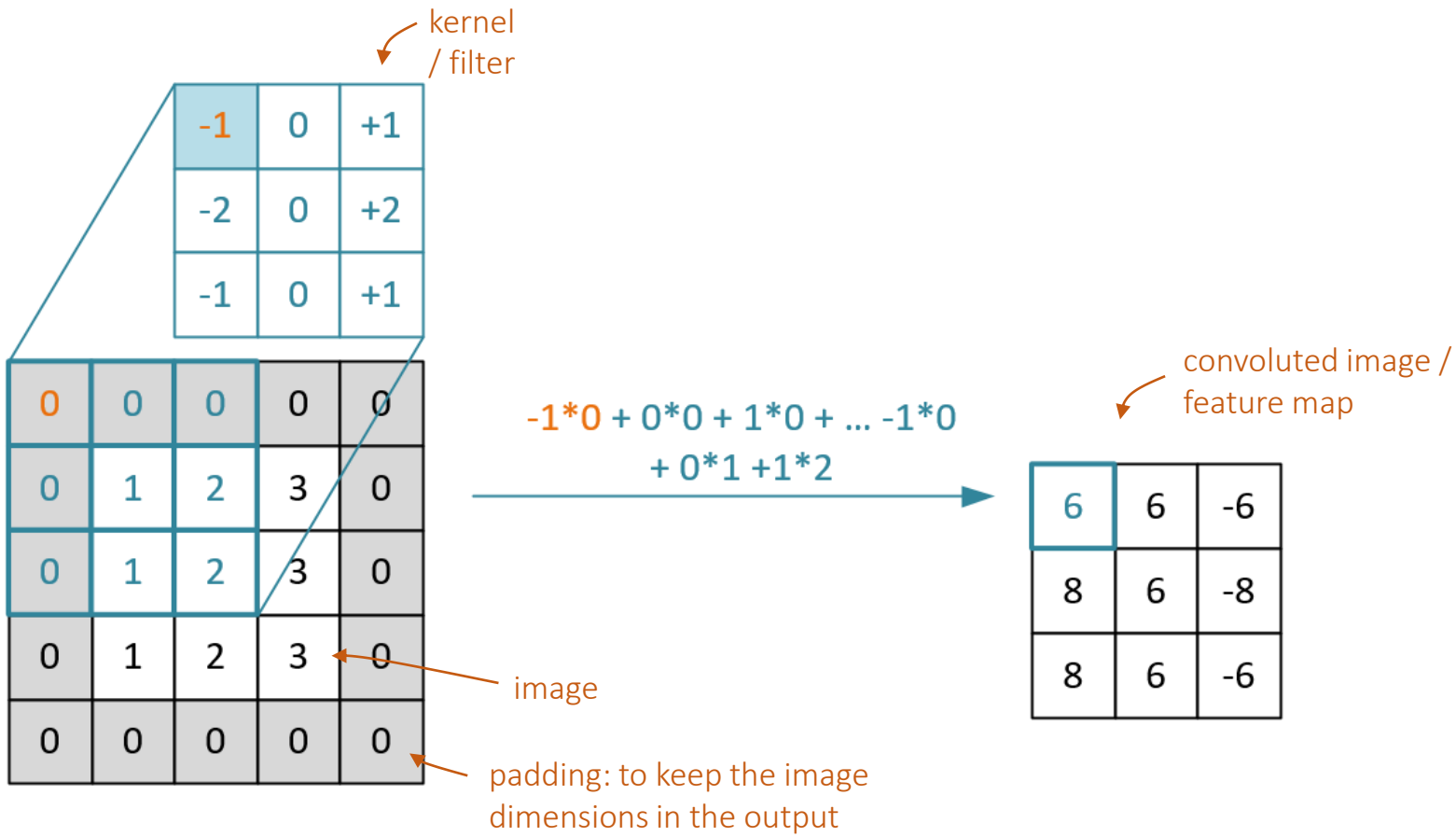
Lecture 6
CNNs & Advanced Network Architectures

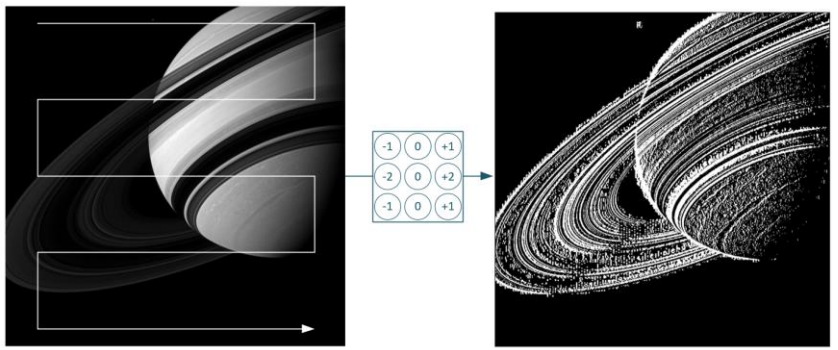
Dr.-Ing. Maike Stern | 19.11.2021

Convolutional neural networks (CNNs)



What is convolution?

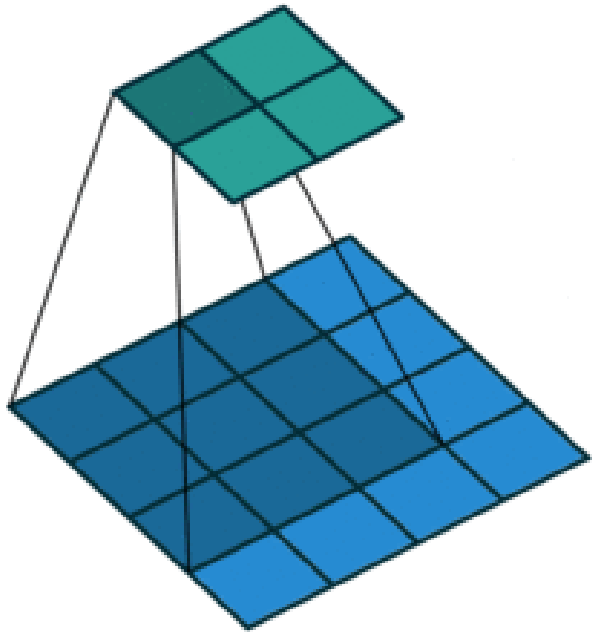




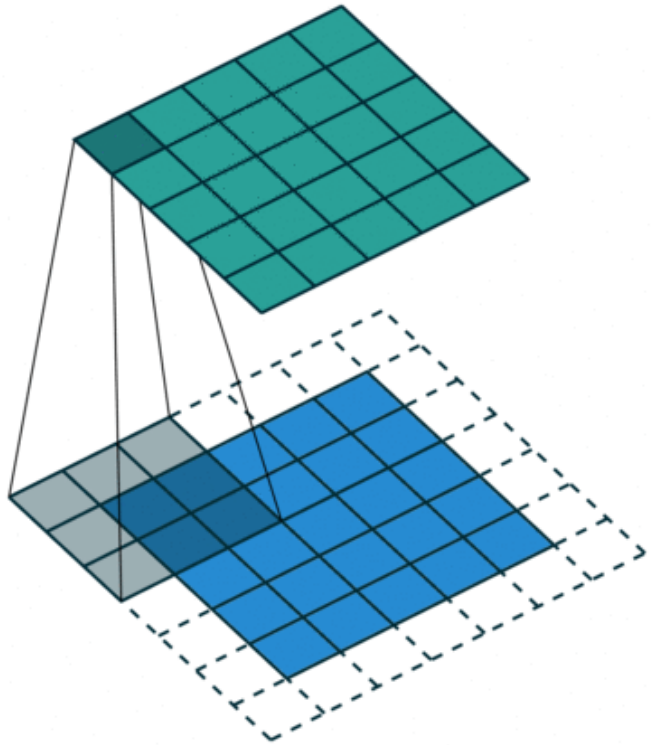
Machine Learning Research

What is convolution?

Convolution with a 3x3 filter



Convolution with padding to keep the dimensions



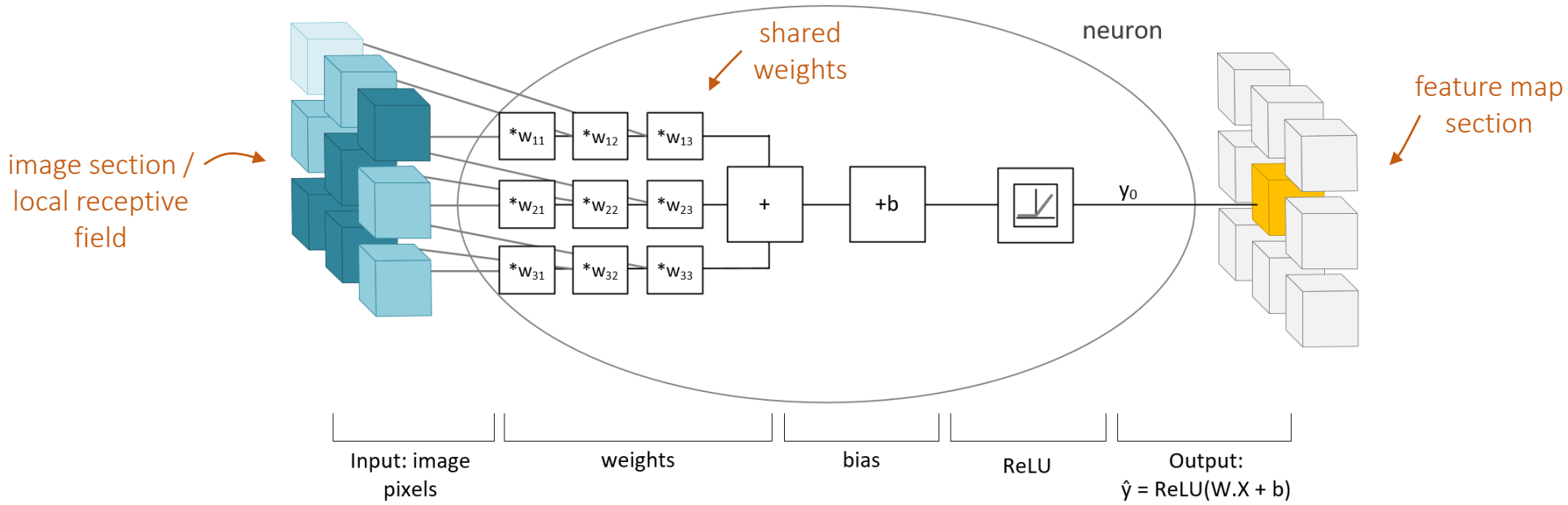
Dumoulin & Visin. A guide to convolution arithmetic for deep learning (2018) https://github.com/vdumoulin/conv_arithmetic

Convolutional neural networks

output, located at the filter's center — bias — input feature maps — kernel height H und width W — input image pixel — kernel weight

Convolution: $z_{i',j',f'} = \mathrm{ReLU}\left(b_{f'} + \sum_{f=1}^{F}\sum_{i=1}^{H}\sum_{j=1}^{W} x_{i'+i-1,j'+j-1,f} \cdot w_{ijff'}\right)$,

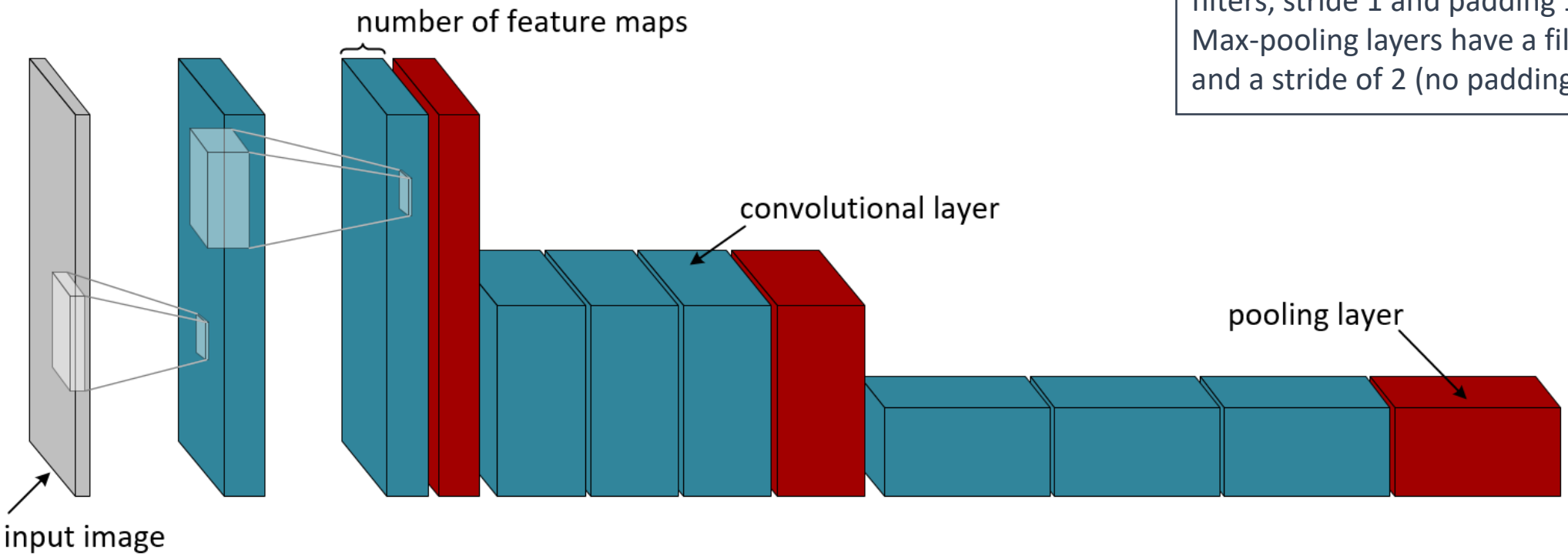
with i', j' feature map indices, i, j kernel indices, F are the input feature maps and f' is the output feature map



Machine Learning Research

Convolutional neural networks

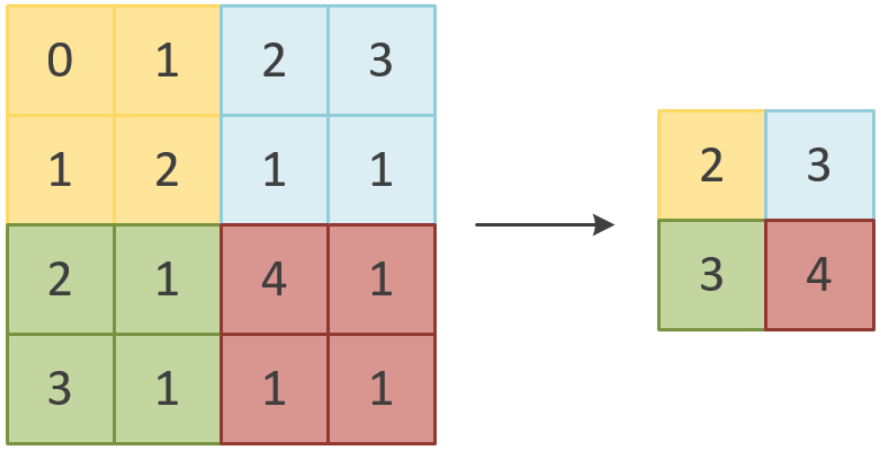
Design rules:
Blocks of two to three convolutional layers followed by a max-pooling layer.
Each layer has 64 to 512 filters, with 3x3 filters, stride 1 and padding 1.
Max-pooling layers have a filter size of 2x2 and a stride of 2 (no padding).



Sub-sampling | Max-pooling

Sub-sampling reduces the feature map size

- Reduces the sensitivity of the output to shifts and distortions by introducting invariance to small translations
- Reduces the computational cost and thus allows the typical bi-pyramid:
  decreasing feature map sizes & increasing number of filters with increasing network depth
- Mostly: max-pooling, also possible: average pooling, L2 pooling



Input size: $W_1 \; x \; H_1 \; x \; D_1$
New size: $W_2 \; x \; H_2 \; x \; D_2$, with:

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1$$

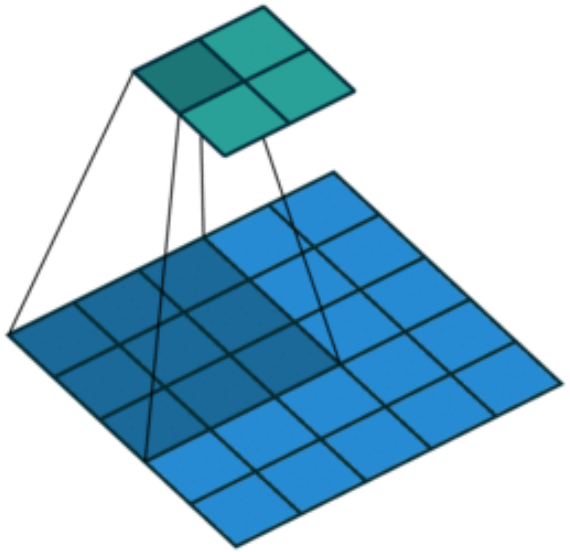
$$D_2 = D_1$$

Where F is the pooling kernel size, here 2, and S is the stride, here 2

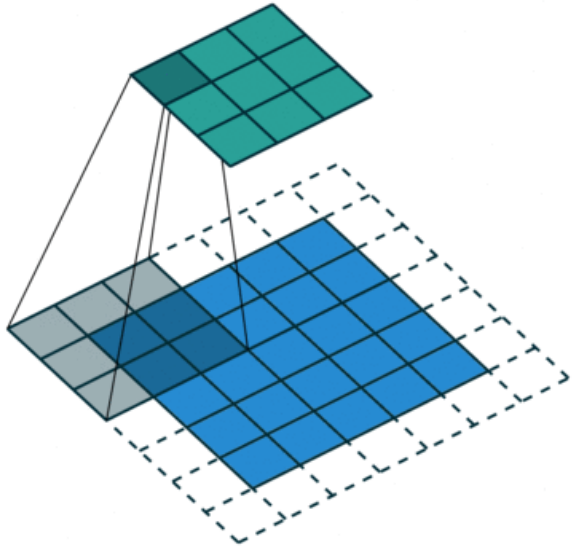
Getting rid of max pooling

- Max-pooling may result in a loss of accurate spatial information
- Instead of sub-sampling with max pooling, we can also use convolutional layers, with a stride of 2 (or bigger)
- Often used in GANs and variational autoencoders

Convolution with a 3x3 filter and stride 2



Convolution with a 3x3 filter, padding, and stride 2



Springenberg et al. Striving for Simplicity: The All Convolutional Net (2014)

From convolution to fully-connected layers

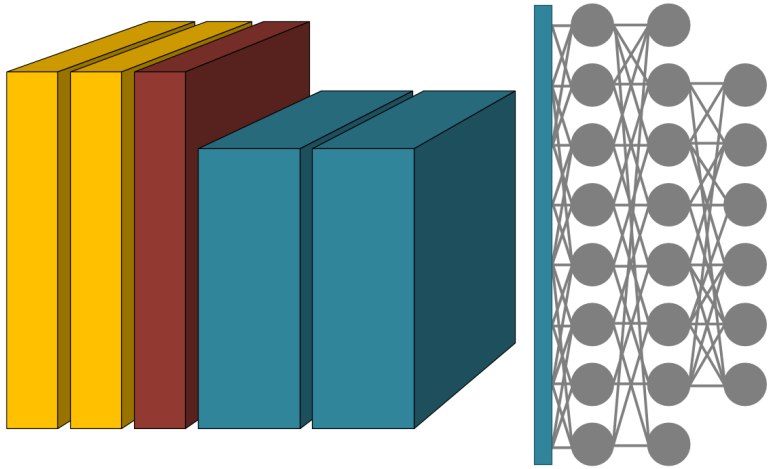
To output a classification, convolutional neural networks transition from the convolutional part (feature detection) into the classifier part, consisting of fully-connected layers

There are two ways to connect both layer types:

- Flatten
  - Use the flattened last convolutional layer as input for the first fully-connected layer (batch_dim, H, W, num_channels) → (batch_dim, H*W*num_channels)
- Global average pooling
  - Computes the mean of each feature map (batch_dim, H, W, num_channels) → (batch_dim, num_channels)
  - Also available as max operation

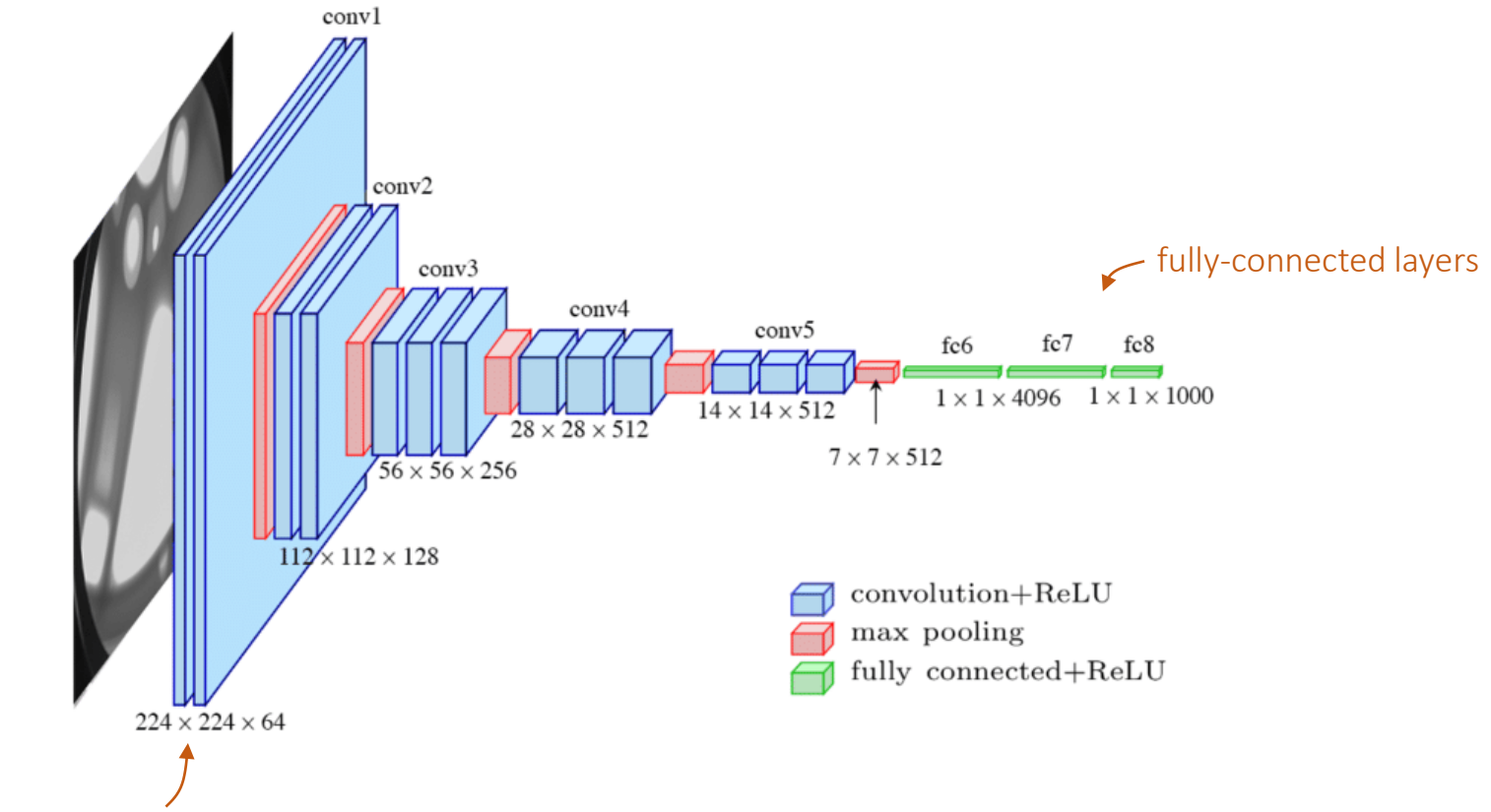
the resulting number of units depends on the input image size

the resulting number of units depends number of feature maps in the last convolutional layer



Setup of a convolutional neural network

The VGG-16 network architecture

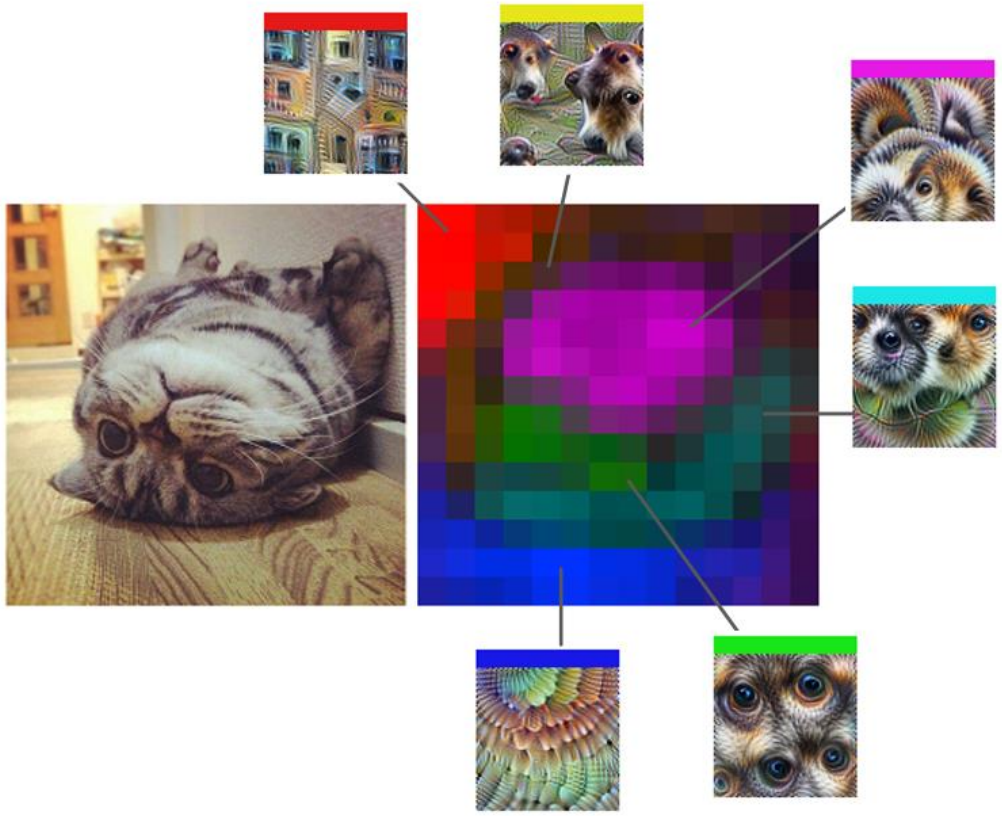


VGG 16: Simonyan & Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition (2014)
Figure source: Ferguson et al. Automatic localization of casting defects with convolutional neural networks (2017)

What a network learns to see

What a convolutional network learns to see

- With self-learned filters, the network learns to distinguish objects that were present in the training data
- Visualising the filters with the highest activation shows how the network analysis an image



https://distill.pub/2018/building-blocks/

What a convolutional network learns to see

- Visualising filters corresponding to a certain class category, here „labrador retriever“ (orange) and „tiger cat“ (blue), shows how the network collects evidence for each hypothesis
- Deeper layers further evaluate previous representations and thus become more confident

https://distill.pub/2018/building-blocks/
https://distill.pub/2019/activation-atlas/

What a convolutional network learns to see

Jason Yosinski. Deep Visualisation Toolbox (2015)
https://www.youtube.com/watch?v=AgkfIQ4IGaM

Convolutional Neural Network Architectures

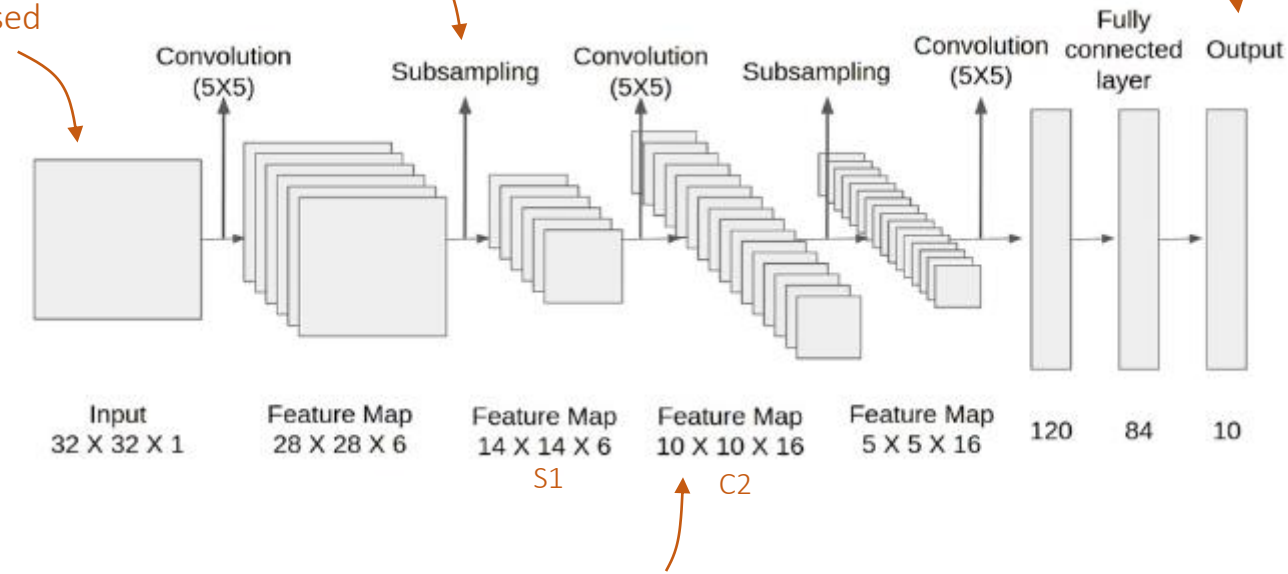
LeNet

Target dataset: MNIST

Input images are padded (from 28x28) and normalised

Average pooling with 2x2 kernels and stride 2

Computes the square of the Euclidean distance between the output's input vector and the weight vector and measures how much the image belongs to a certain class



Feature maps are not padded, thus feature map sizes shrink with every convolution

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Feature maps of S1 combined by each C2 feature map

- Reduces the number of connections
- Forces a break of symmetry → more diversity in the learned features

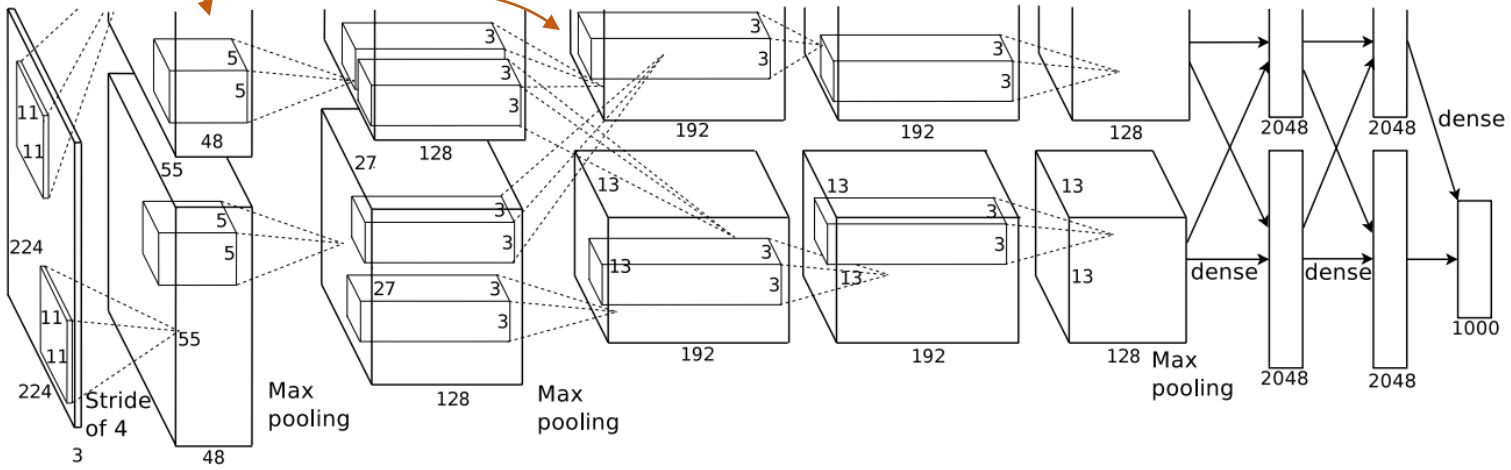
LeCun et al. Gradient-based learning applied to document recognition (1998) http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

AlexNet

- Target dataset: ImageNet challenge dataset
- 50% dropout rate in the fully-connected layers
- Data augmentation

Local response normalisation (LRN): the most strongly activated neurons inhibit other neurons located at the same position in neighbouring feature maps, thus encouraging more diverse feature maps

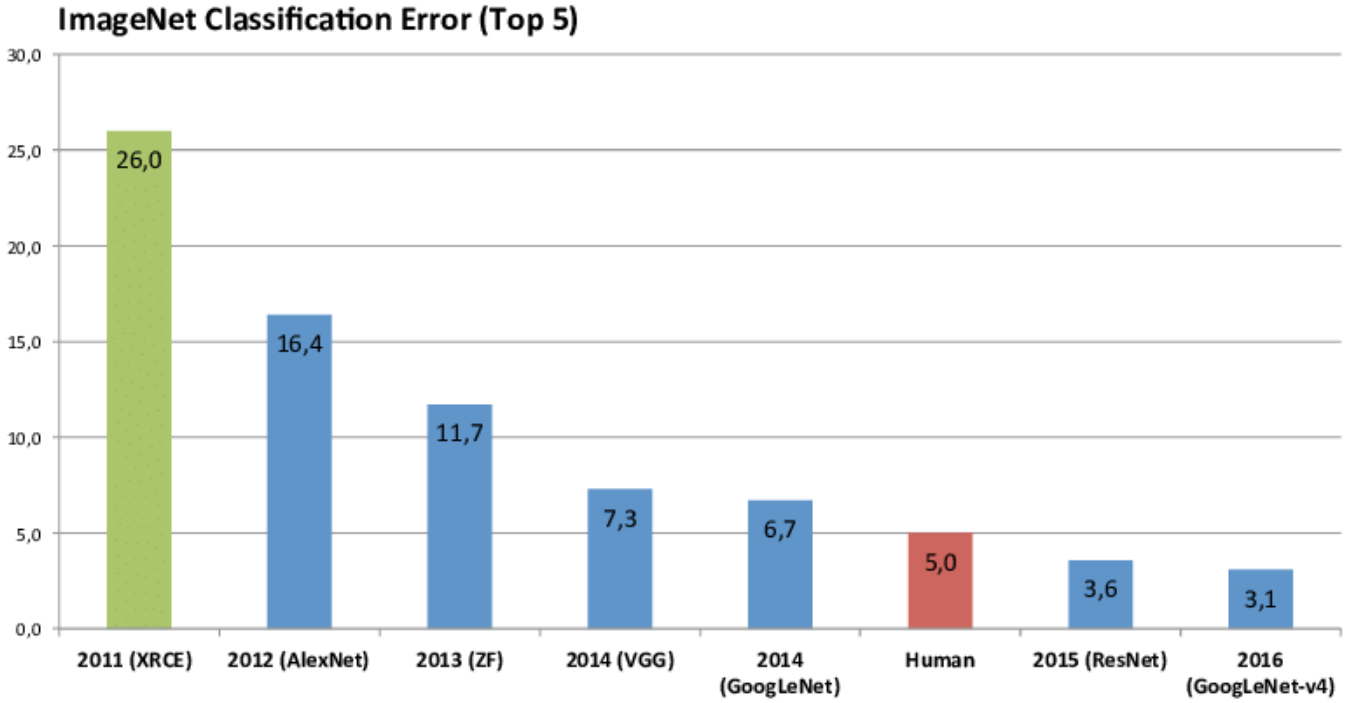
Stacked convolutional layers



Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks (2012)
https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

ImageNet challenge

- https://www.image-net.org/index.php



IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Russakovsky, Deng, et al. ImageNet Large Scale Visual Recognition Challenge (2015) https://arxiv.org/abs/1409.0575

Machine Learning Research

ImageNet challenge

- https://www.image-net.org/index.php
- 2019: Cleaning of the dataset
  Yang et al. Towards Fairer Datasets: Filtering and Balancing the Distribution of the People Subtree in the ImageNet Hierarchy https://arxiv.org/pdf/1912.07726.pdf
- 2020: New Labels
  Beyer et al. Are we done with ImageNet? (2020) https://arxiv.org/pdf/2006.07159.pdf

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Russakovsky, Deng, et al. ImageNet Large Scale Visual Recognition Challenge (2015) https://arxiv.org/abs/1409.0575

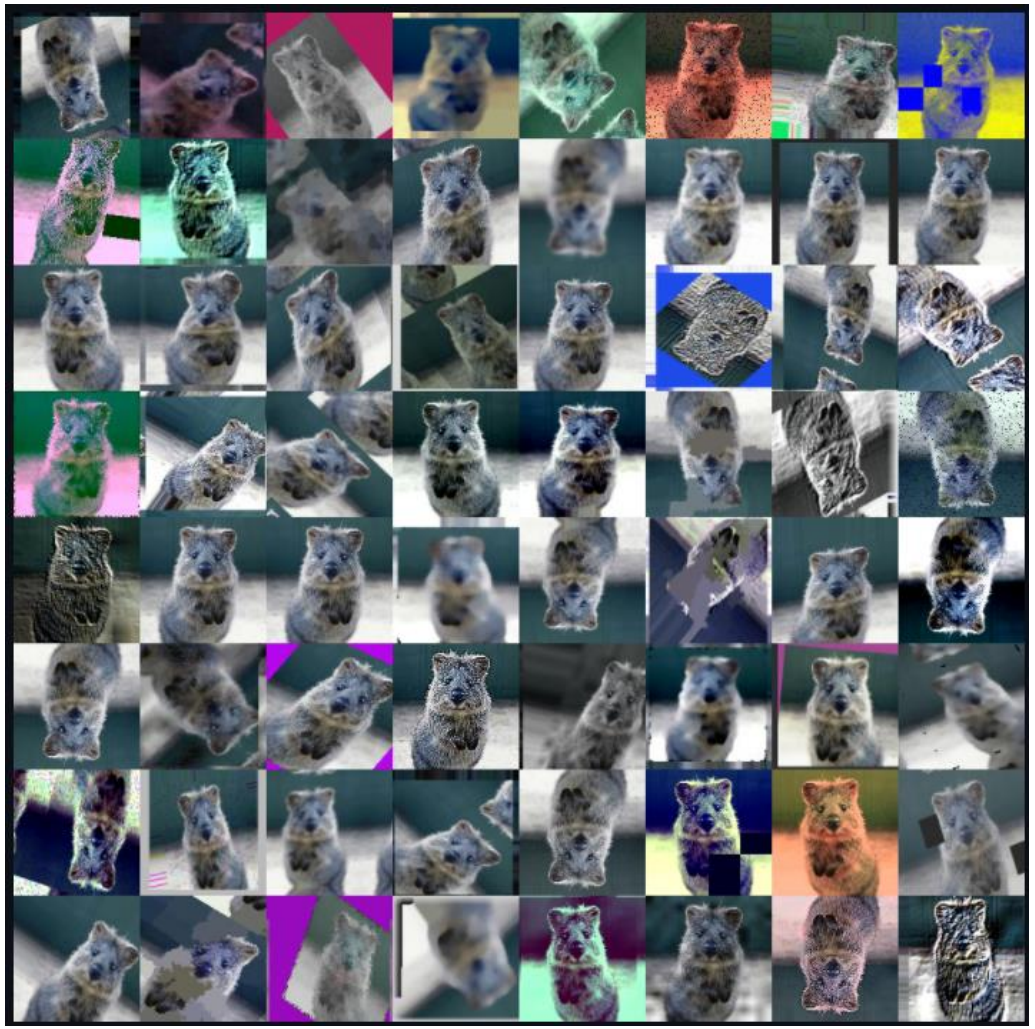
Data Augmentation

Increase the training dataset size by generating many **realistic** variants of the original training samples

- Reduces overfitting by increasing the model's robustness

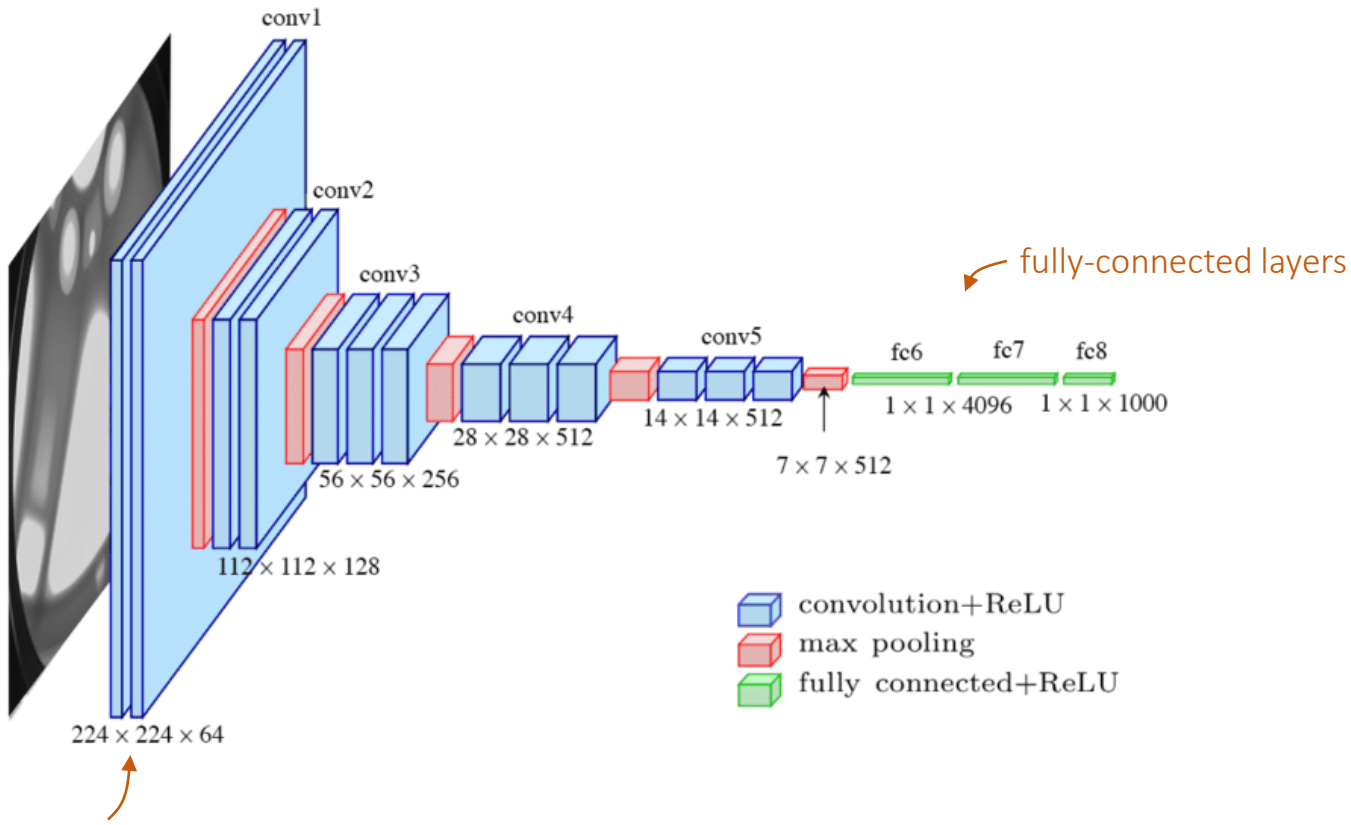
Augmentation libraries:

- https://www.tensorflow.org/tutorials/images/data_augmentation
- https://github.com/aleju/imgaug
- https://ai.facebook.com/blog/augly-a-new-data-augmentation-library-to-help-build-more-robust-ai-models/
- https://albumentations.ai/



VGG-16

- Target dataset: ImageNet
- Second place, ImageNet challenge 2014
- Two networks: VGG-16 and VGG-19
- 3x3 filters throughout the network
- Transfer learning



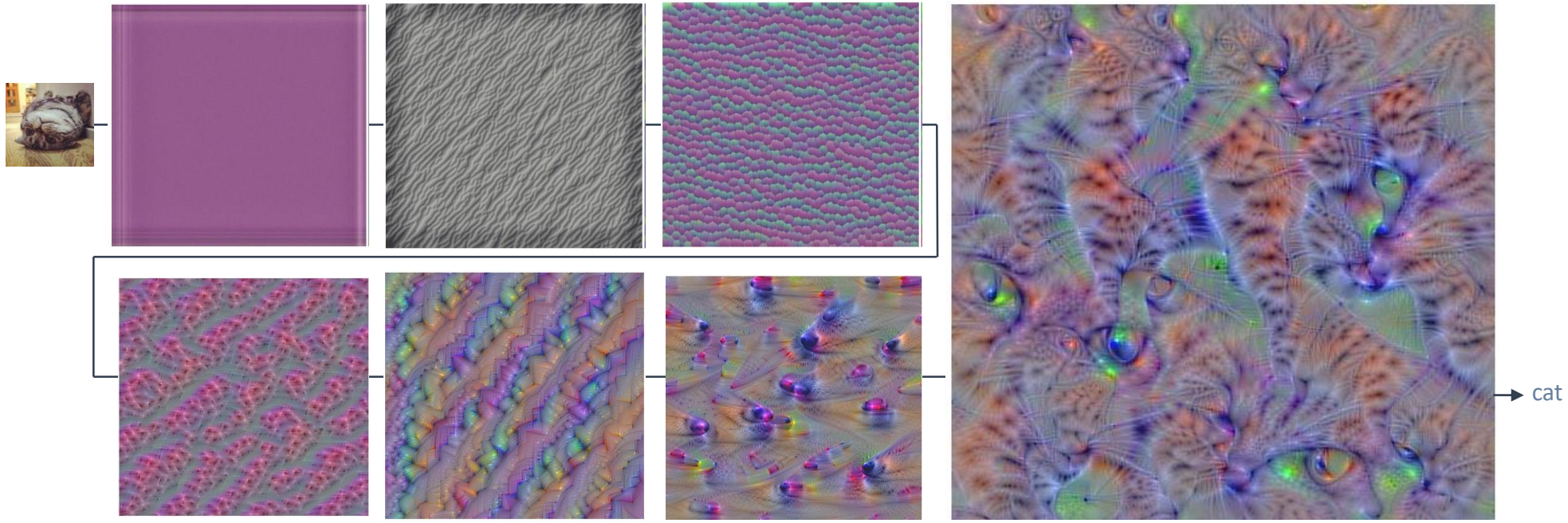
feature map height x width x number of filters

VGG 16: Simonyan & Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition (2014)
Figure source: Ferguson et al. Automatic localization of casting defects with convolutional neural networks (2017)

Transfer learning

- The shallow layers of a CNN learn basic patterns, such as colours, edges, etc.
- With increasing network depth, the learned filters get more specific as they operate on more and more abstracted features of the original input and have a larger receptive field

→ Transfer parameters from a pre-trained network and use them to initialise (parts of) your network



Transfer learning

In case of similar tasks

- Choose a suitable network architecture

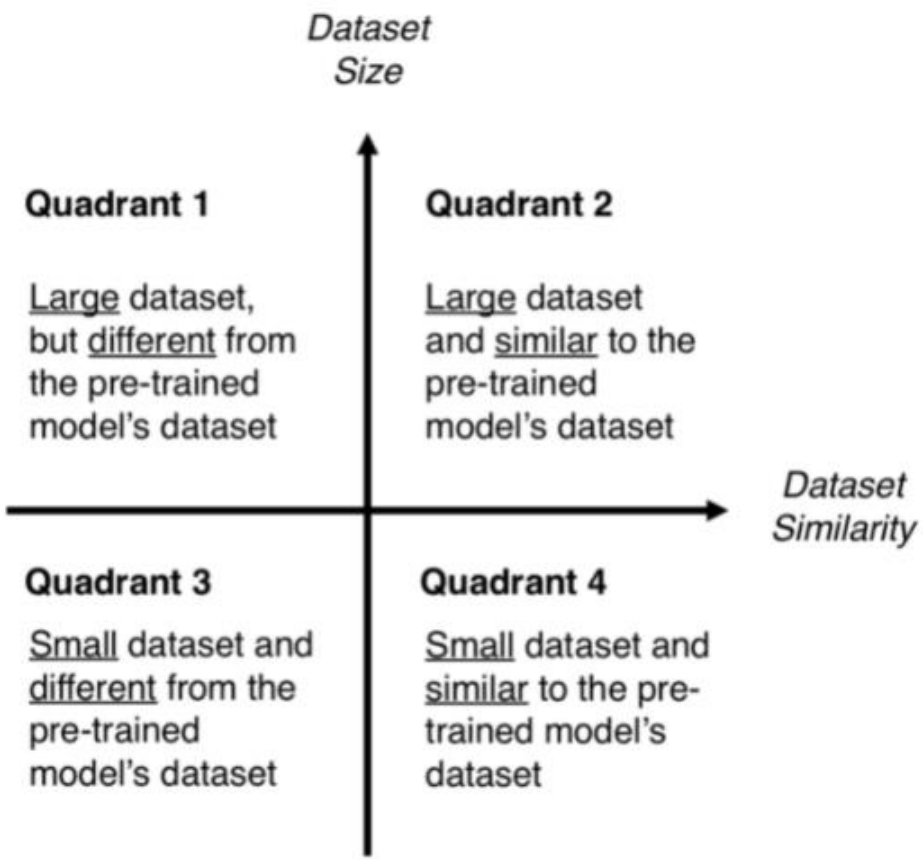
Use weights pre-trained on the ImageNet dataset

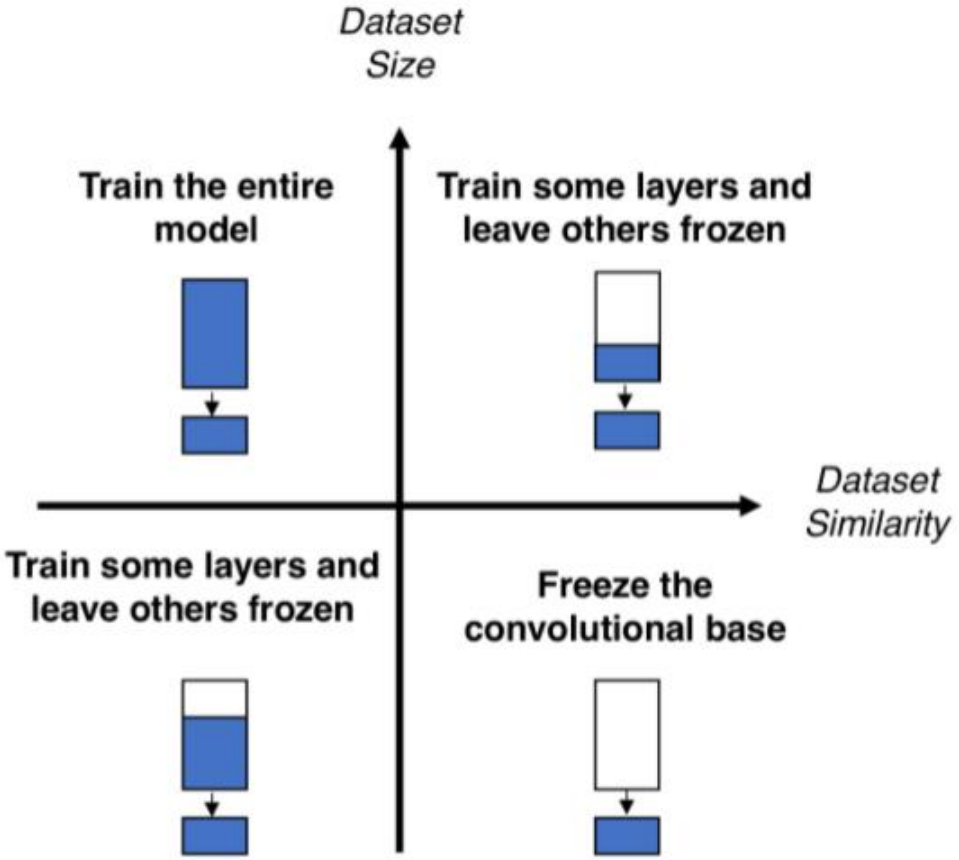
„False“ removes the fully-connected layers

```
tf.keras.applications.vgg16.VGG16(
    include_top=True, weights='imagenet', input_tensor=None,
    input_shape=None, pooling=None, classes=1000,
    classifier_activation='softmax'
)
```

- Add your own classifier (fully-connected layers, randomly initialised)
- Freeze the pre-trained layer parameters
- Train the new layers on your dataset
- If your data is quite different to the dataset the network was trained on:
  - Unfreeze the convolutional layers and fine-tune the whole network using a very small learning rate
  - For very dissimilar data: Transfer only parameters of the shallow convolutional layers and randomly initialise the deep conv layers & fully-connected layers

Transfer learning

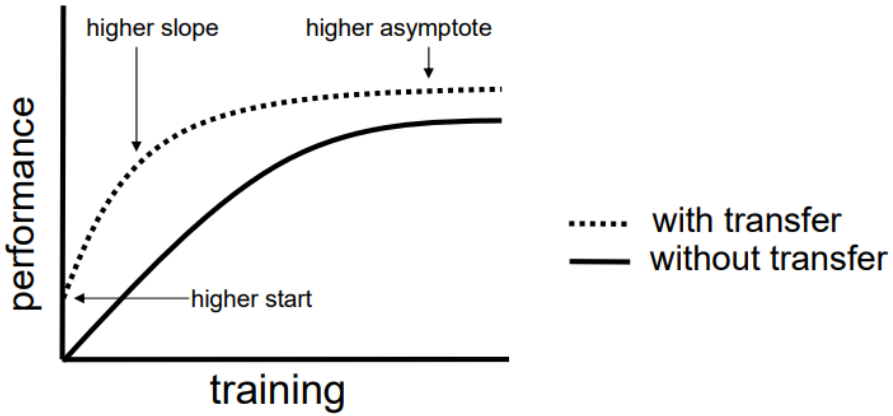




https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

Transfer learning

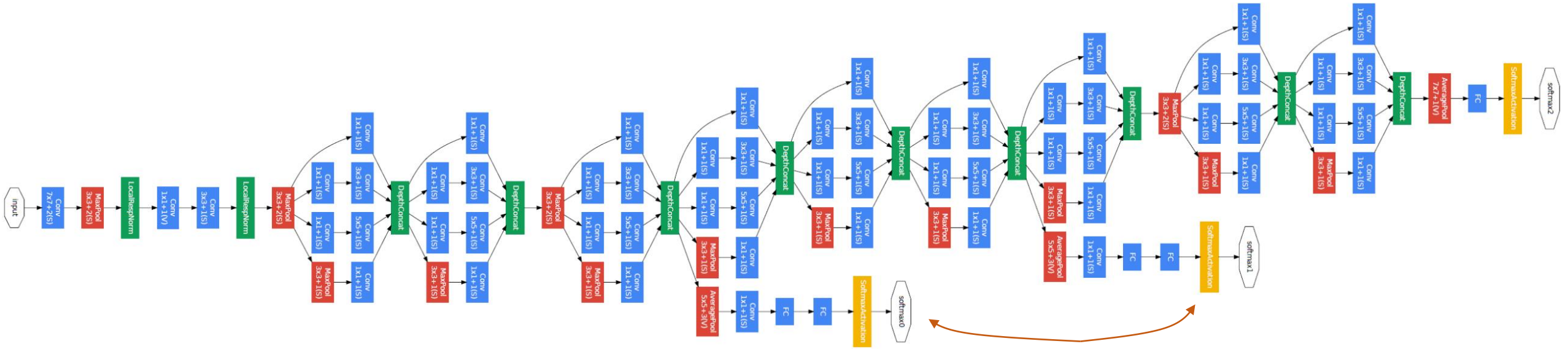
- Exploits the fact that images typically feature the same patterns up to a certain degree (e.g. edges, light-dark transitions, colours, …)
- Hence, transfer learning speeds up the training and requires less data
- Works best on similar data, but can also improve the performance of dissimilar images
- Popular not only in computer vision but also natural language processing (NLP), where large transformer models (ELMO, BERT, GPT-3, etc.) are trained on very, very large datasets A single training of GPT-3 costs >4.5 Million $



Torrey & Shavlik. Transfer Learning (2009) http://vision.stanford.edu/documents/Fei-FeiFergusPerona2006.pdf

GoogLeNet / Inception

- Target dataset: ImageNet
- First place ImageNet challenge 2014 (ensemble of 7 networks)
- 22-layer deep network, but only ≈6 million parameters (AlexNet ≈60 million)

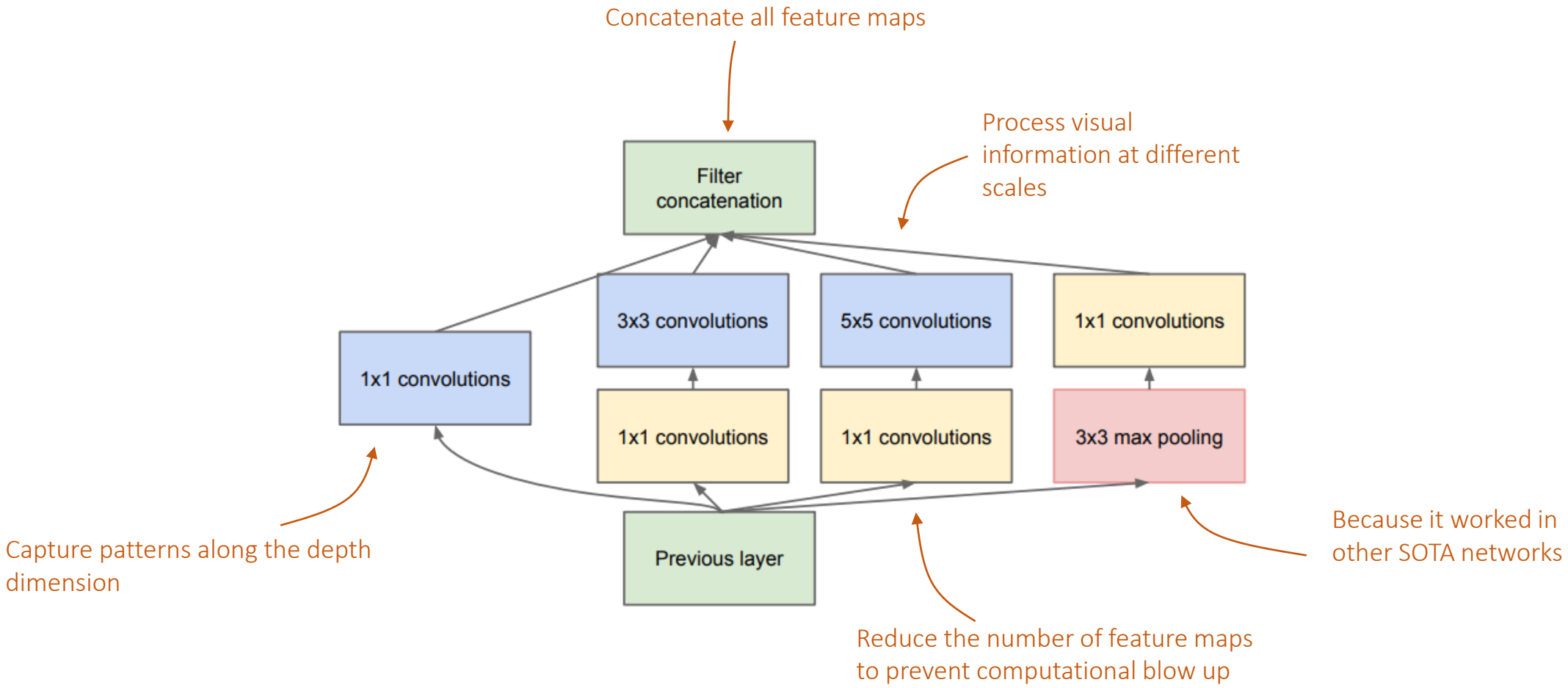


Auxiliary classifiers:

- Improve gradients
- Encourage feature discrimination in shallow layers
- Loss is added to the final loss

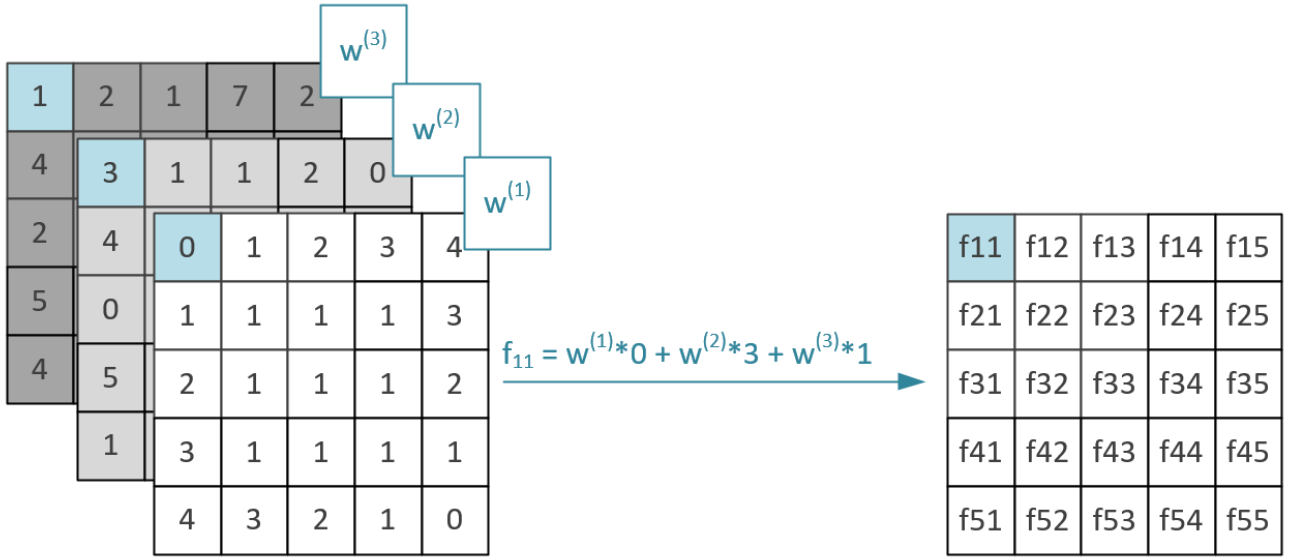
Szegedy et al. Going deeper with convolutions (2014) https://arxiv.org/abs/1409.4842

GoogLeNet
Inception Modules



GoogLeNet
1x1 convolutions

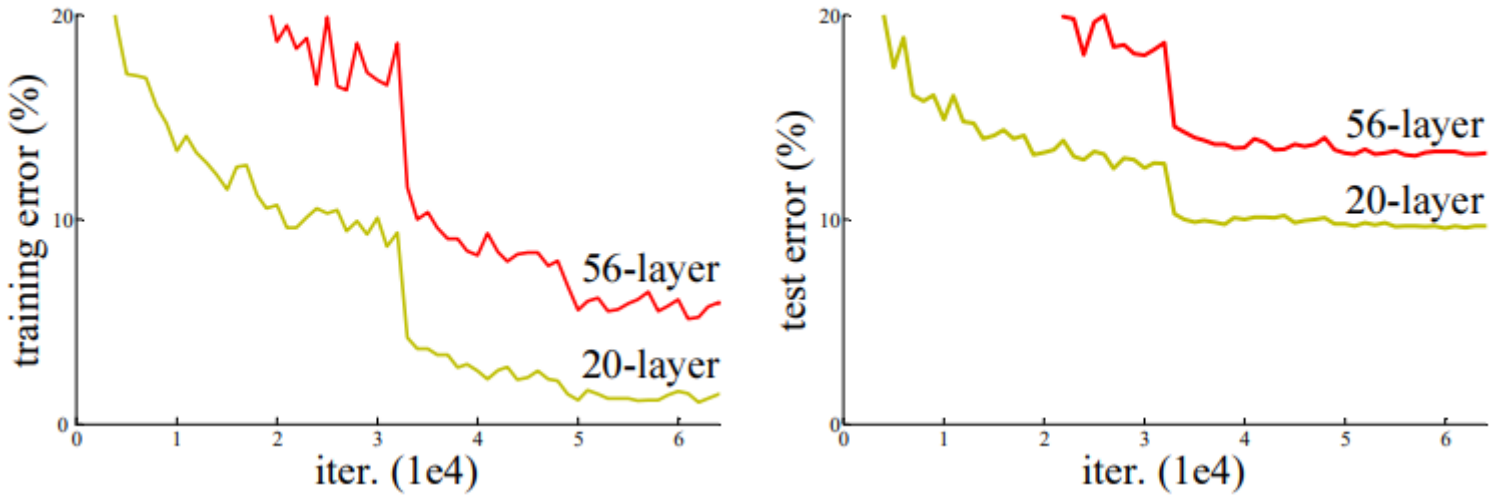
- Also called feature map pooling or projection layer
- Used to reduce the number of feature maps (depth)
- Kind of pixel-wise summary of the input feature maps



ResNet

ResNets adress the so-called degradation problem: with increasing network depth, accuracy saturates and then degrades rapidly

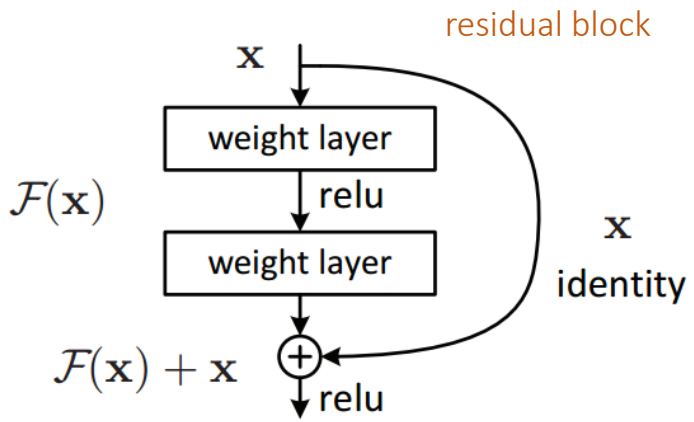
Not overfitting, because both, training error and test error are higher in the deeper network



ResNet
Skip connections

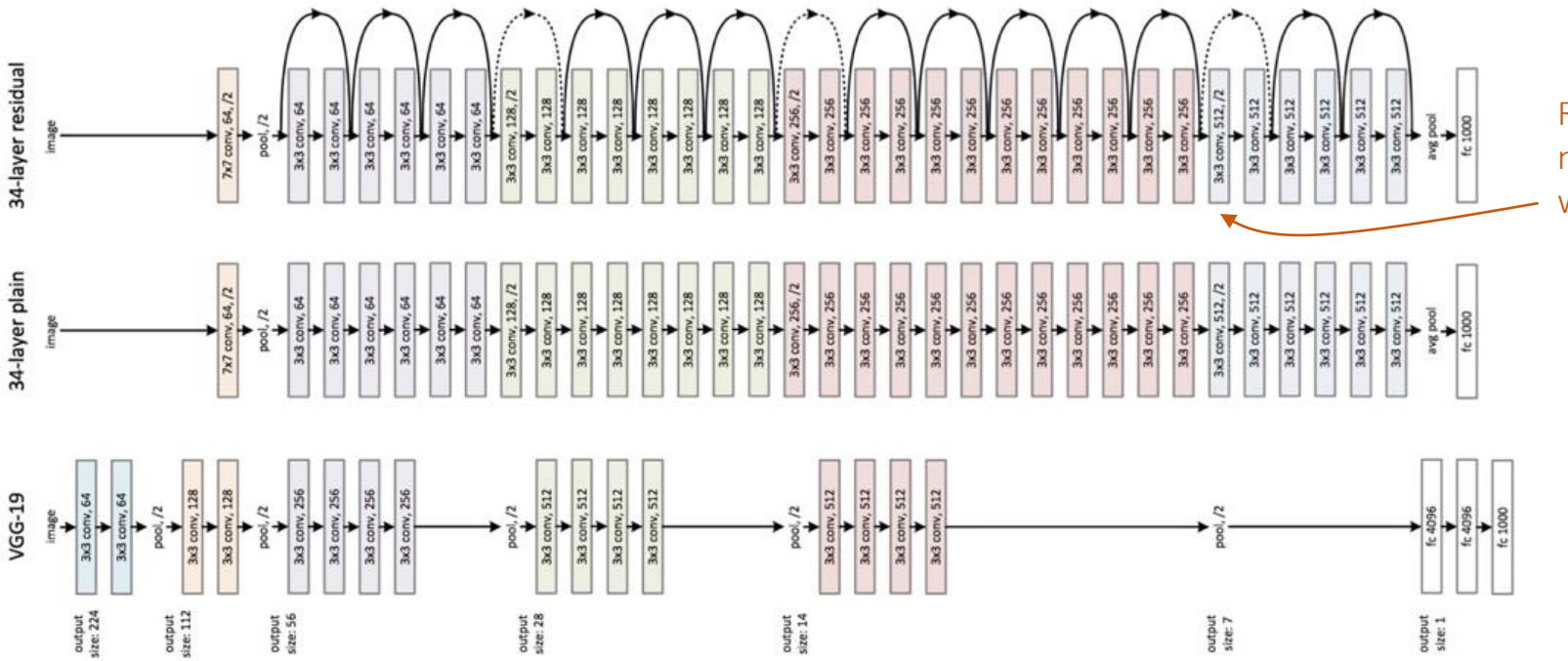
To adress the degradation problem, ResNets introduce residual blocks

- Randomly initialised layers (with standard-normally distributed weights close to zero) output standard-normally distributed values close to zero
- Hence, untrained residual blocks output the identity function, that is they output a copy of their input
- Often, the target function is close to the identity function, hence residual blocks speed up training



ResNet

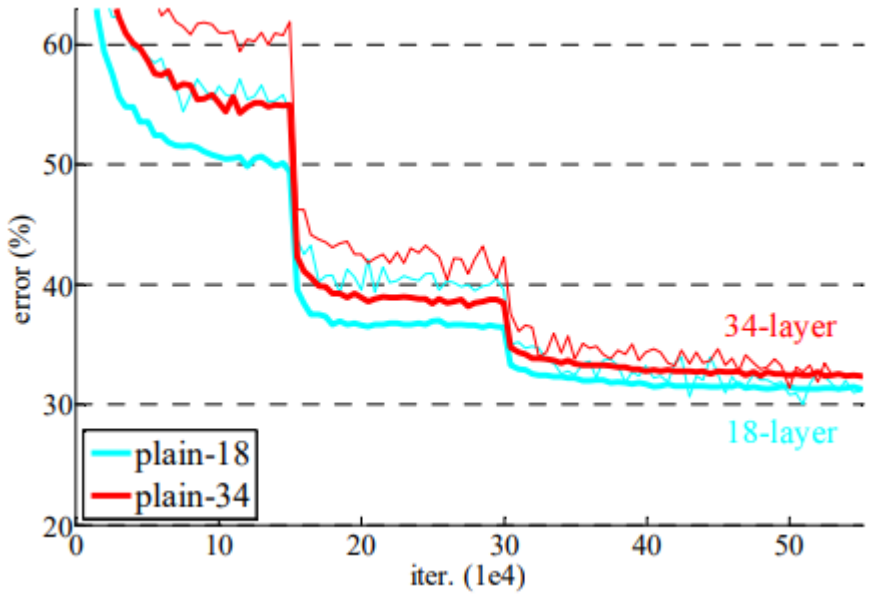
- Target dataset: ImageNet
- First place ImageNet challenge 2015 (152 layers)
- Variants with 34, 50, 101 and 152 layers
- Uses convolutions to reduce feature map sizes

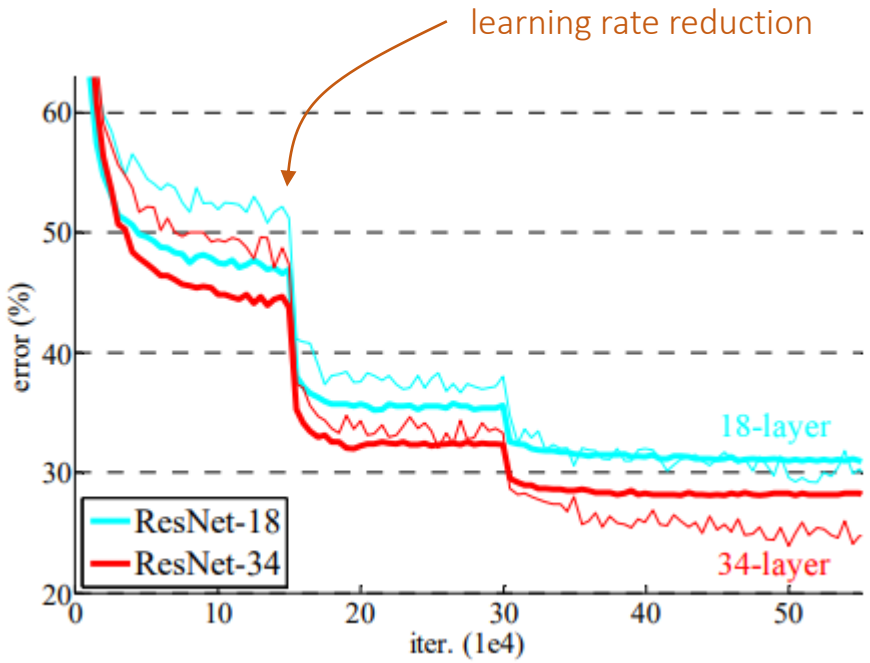


He et al. Deep residual learning for image recognition (2015) https://arxiv.org/abs/1512.03385

Machine Learning Research

ResNets


error (%)
iter. (1e4)
plain-18
plain-34
34-layer
18-layer


learning rate reduction
error (%)
iter. (1e4)
ResNet-18
ResNet-34
18-layer
34-layer

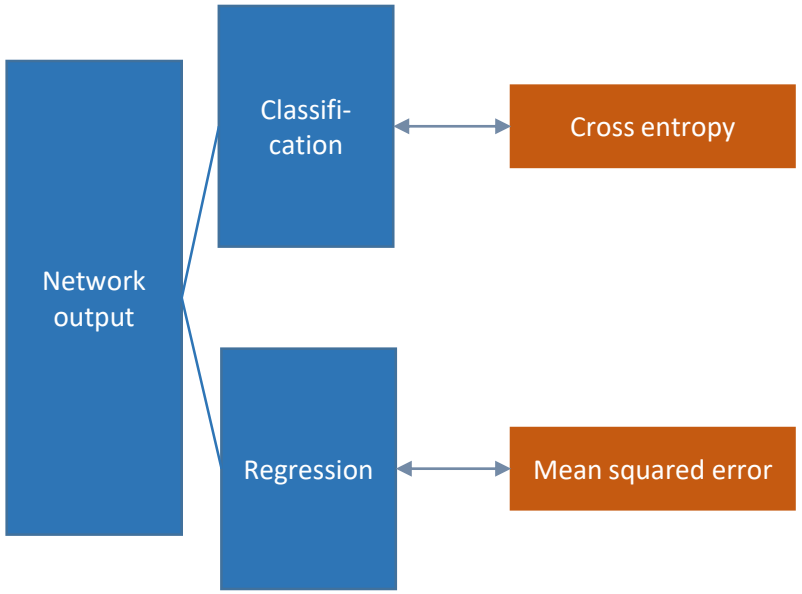
Object Localisation & Detection

Object localisation

- Determine the object position in addition to the object classification
- So-called bounding boxes visualise the object position in the image
- Bounding boxes are comparably cheap to label (in contrast to pixel-wise labels)
- Research dataset: https://cocodataset.org/#home (object detection, segmentation, and captioning)

Object localisation

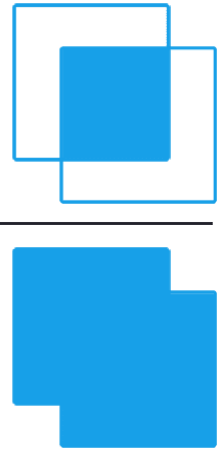
- Determine the object position in addition to the object classification
- So-called bounding boxes visualise the object position in the image
- Add an additional regression output to the network, predicting bounding box coordinates (center (x,y), height, width)



Object localisation

- Determine the object position in addition to the object classification
- So-called bounding boxes visualise the object position in the image
- Add an additional regression output to the network, predicting bounding box coordinates (center (x,y), height, width)
- Performance is typically determined using the intersection over union metric

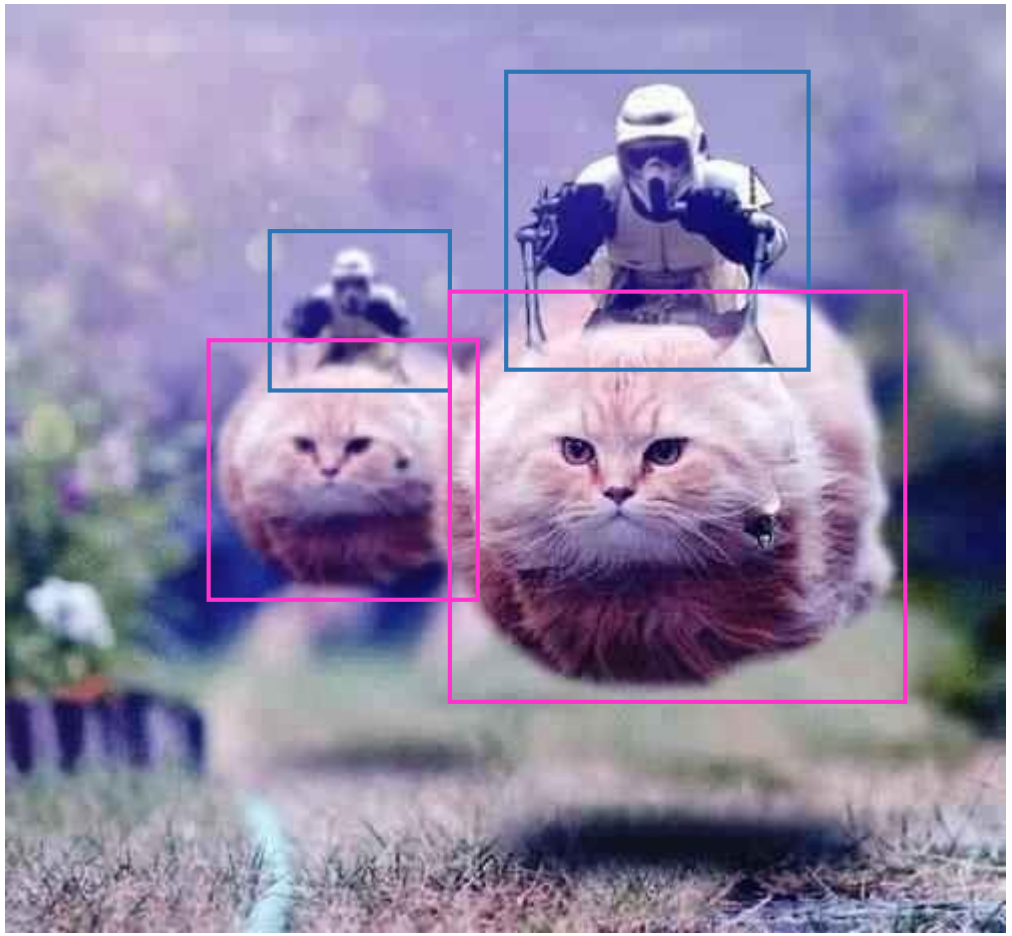
$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



IoU values over 0.5 are usually considered a good prediction

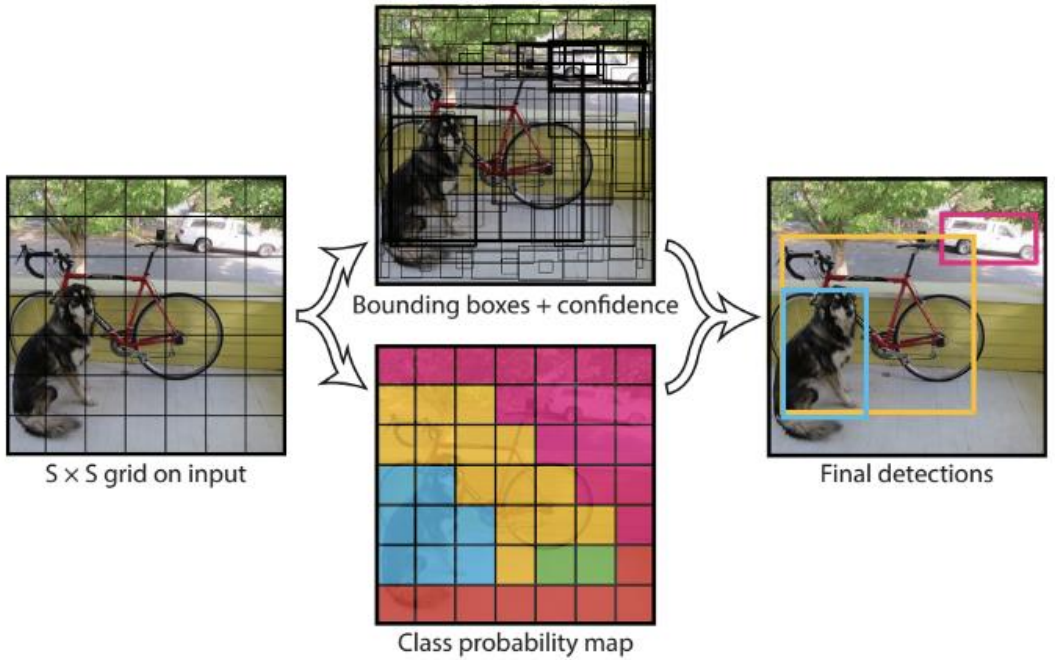
Object detection

- Classify and localise different objects in an image
- The problem: the length of the output layer depends on the variable number of objects in the image



Object detection

- One approach (very simplified) is to split the image into s x s grids and predict m differently sized bounding boxes per grid
- For each of the bounding boxes, an objectness score is computed (prediction confidence) as well as the class probabilities



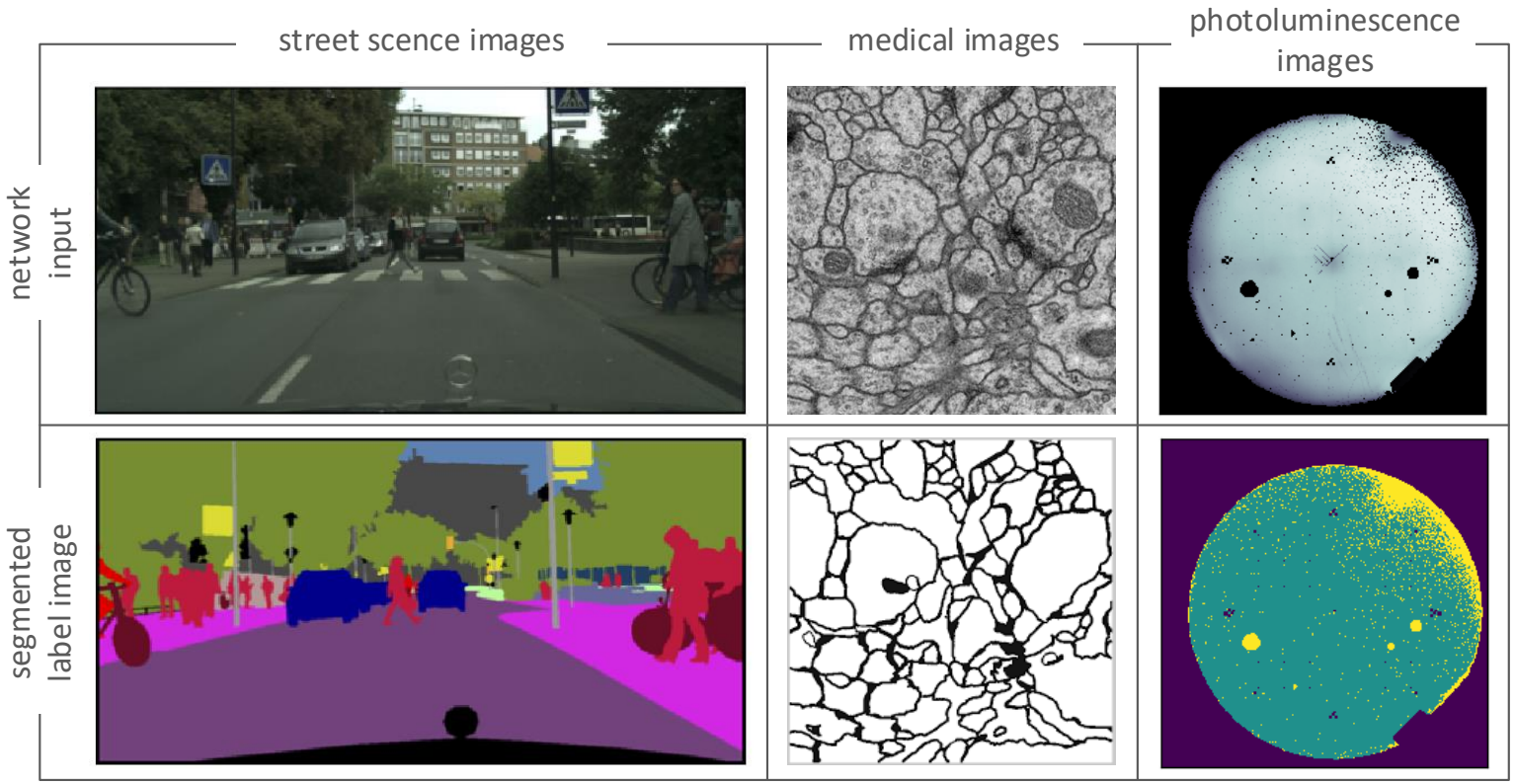
State-of-the-art architectures

- R-CNN, Fast R-CNN and Faster R-CNN
  (https://paperswithcode.com/method/r-cnn)
- YOLO, YOLO9000, YOLOv3(-6)
  (https://paperswithcode.com/paper/yolov3-an-incremental-improvement)
- SSD
  (https://paperswithcode.com/method/ssd)
- RetinaNet
  (Keras: https://keras.io/examples/vision/retinanet/)

https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Semantic Segmentation

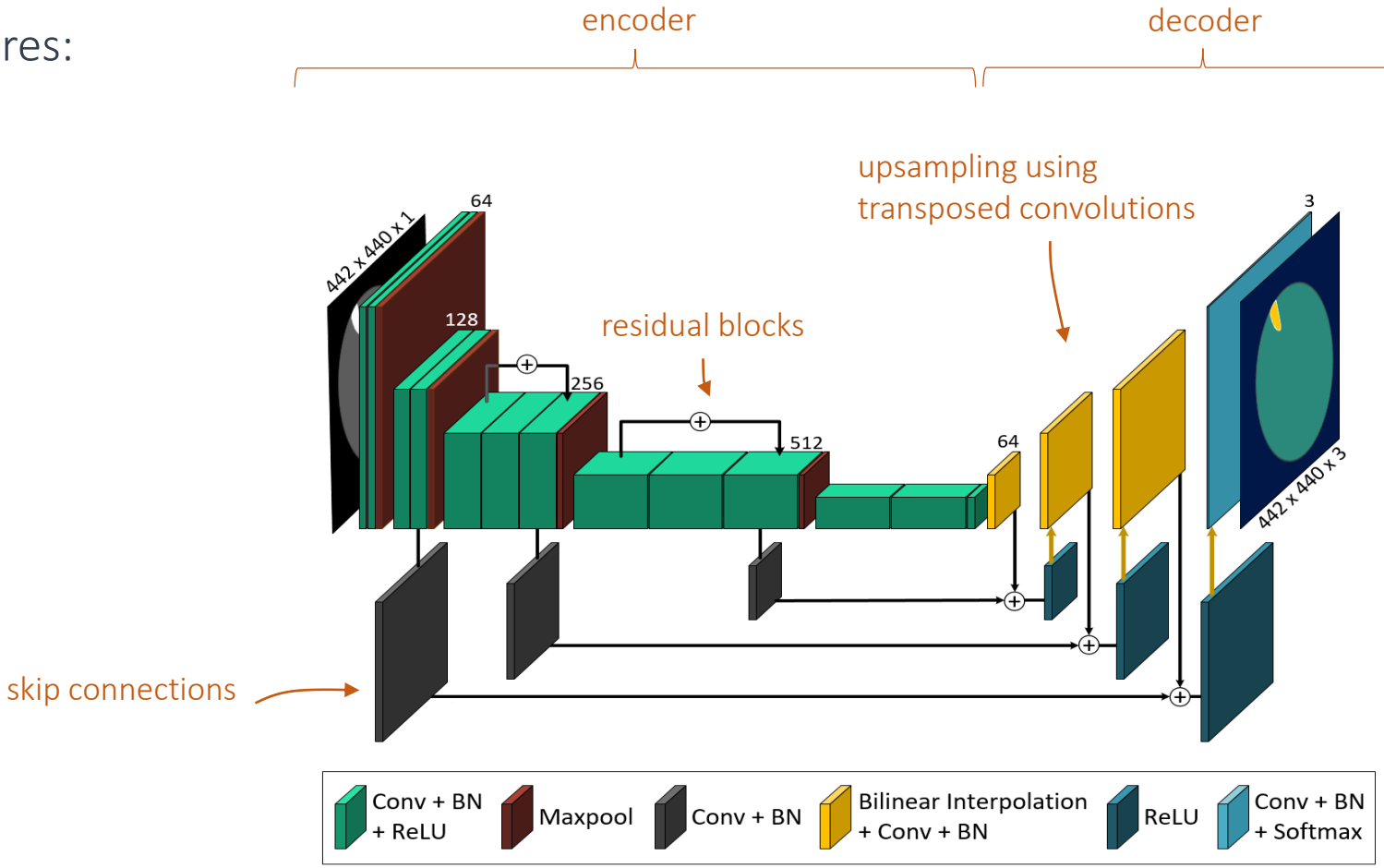
Pixel-wise classification of images



Fully convolutional networks

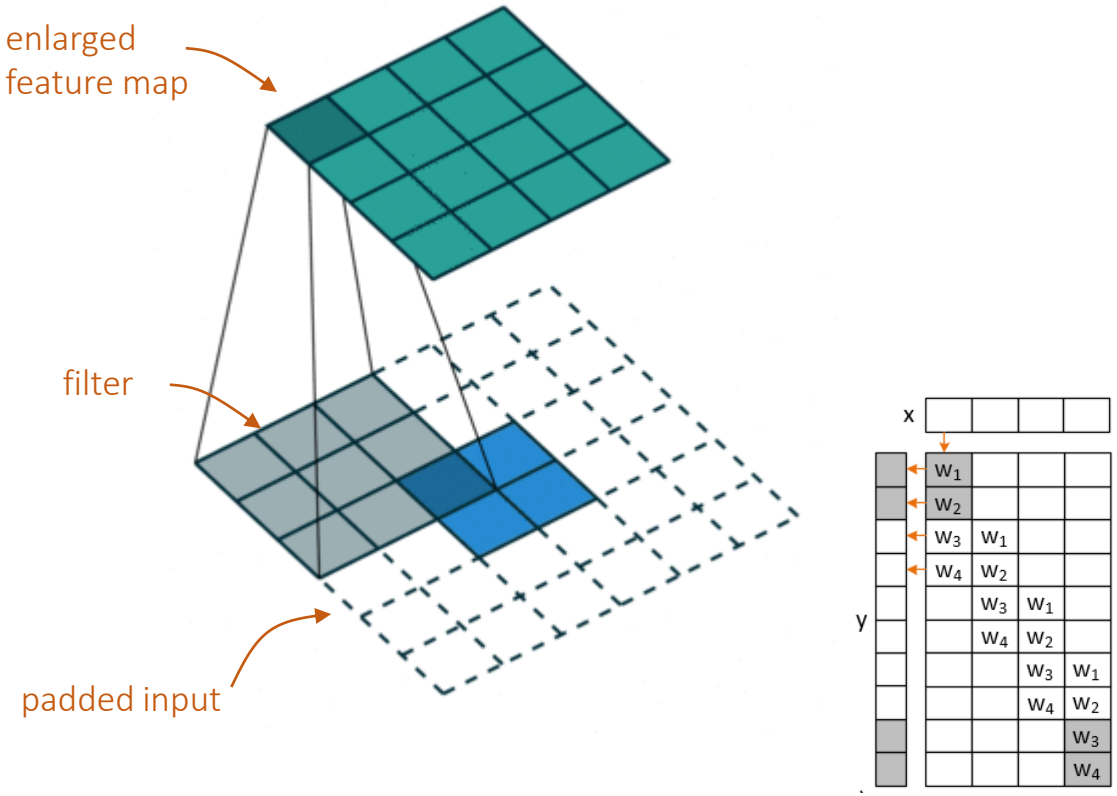
State-of-the-art architectures:

- U-Net
- DeepLab1-3
- Segnet

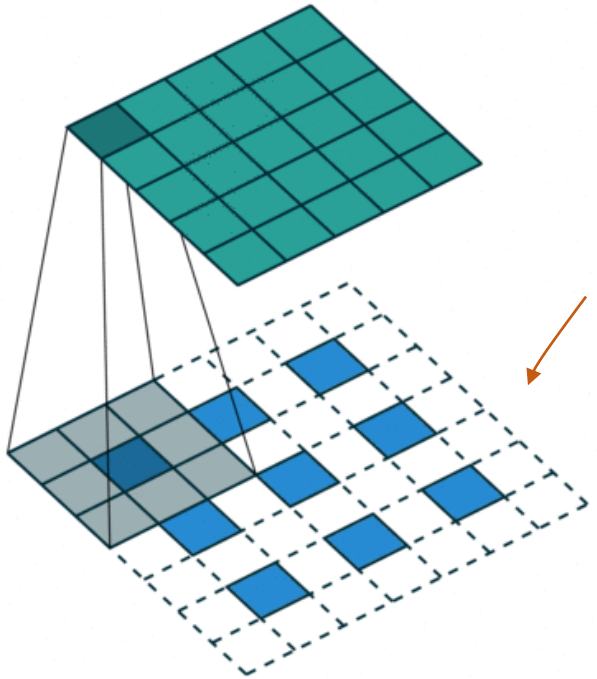


Transposed convolution

Transposed convolution with a 3x3 filter



Transposed convolution with a 3x3 filter and a stride of 2

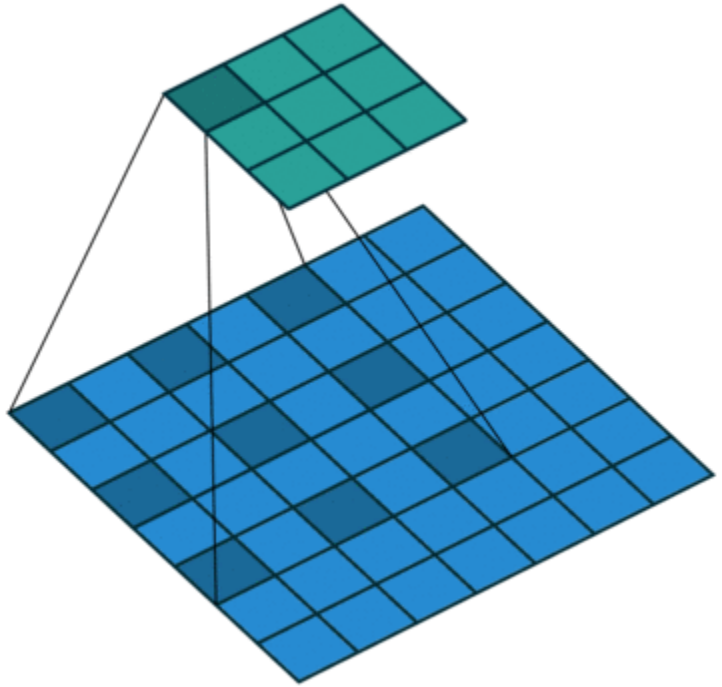


Dumoulin & Visin. A guide to convolution arithmetic for deep learning (2018) https://github.com/vdumoulin/conv_arithmetic

Machine Learning Research

Dilated convolution

Dilated convolution with a
3x3 filter

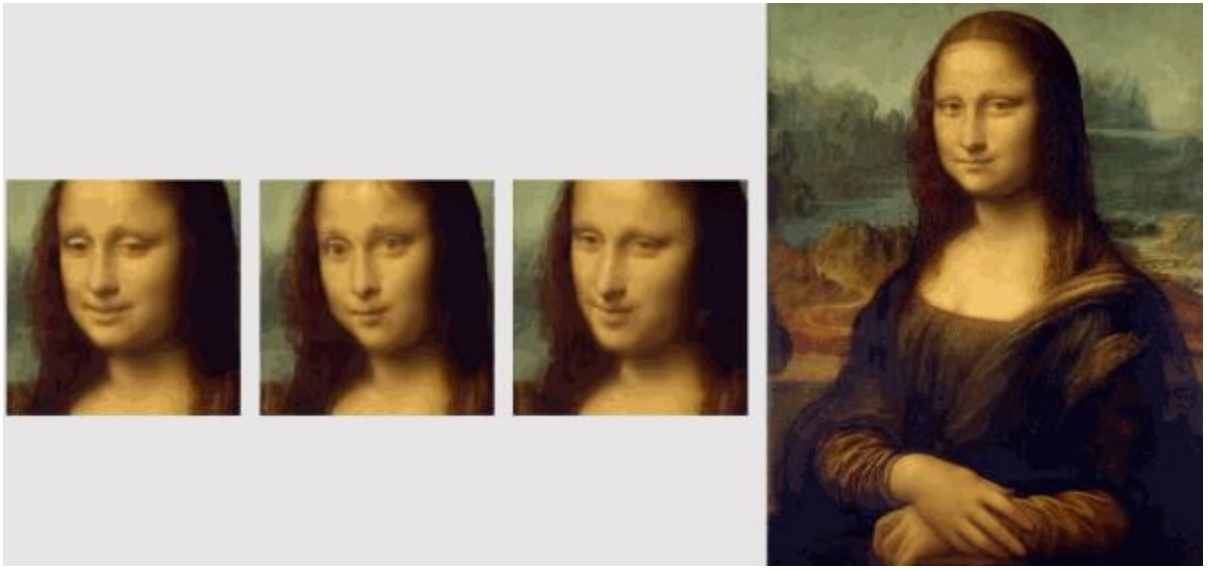


Dumoulin & Visin. A guide to convolution arithmetic for deep learning (2018) https://github.com/vdumoulin/conv_arithmetic

Generative Adversarial Networks

GANs

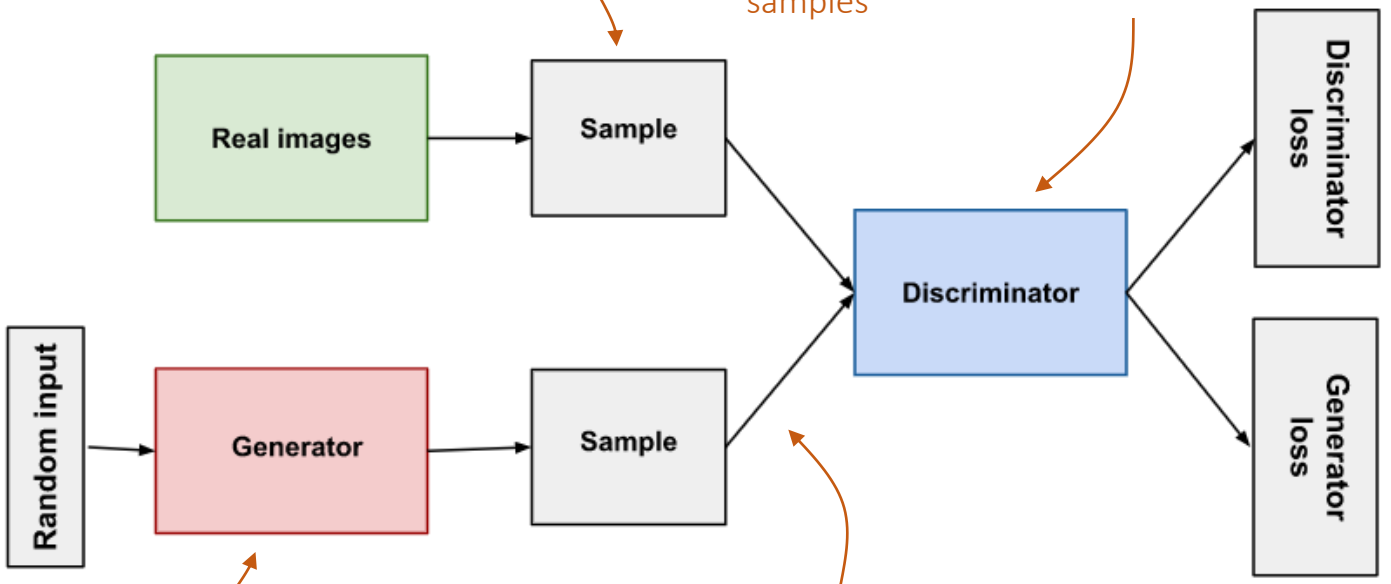
GANs are used to generate new data, e.g. images of non-existent people, new music pieces, or manipulate existing data



GANs

The true samples are the positive class and should be recognised as such by the discriminator

The discriminator (a neural network) learns to distinguish real from fake data and penalises the generator for producing recognisably false samples



The generator is fed with sampled random noise, e.g. noise images

The generator (a neural network) learns to generate plausible data, which the discriminator is not able to distinguish from real data

The generated samples are the negativ training examples for the discriminator

The training goal is to train a generator that produces indistinguishable fakes

https://developers.google.com/machine-learning/gan

Machine Learning Research

Network Training

Conclusion
Network training sanity checks

In general

- If you are building your own architecture, start with a simple design / design that is known to work
- Keep in mind, that CNNs learn hierarchical feature representations, that is the shallow layers learn simple, general patterns/features (e.g. edges, colours, etc.) while the deep layers learn data-specific features → the less information images contain (e.g. measurement images) the fewer layers your network needs
- Run the sanity checks (next slide)
- Monitor the loss / metrics and tune the hyperparameters (slides after the next slide)
  - For tuning (e.g. with Keras tuner) start with a coarse net of the most important hyperparameters and run optimisation for a few epochs only (e.g. 10), which should be sufficient to discover a trend. Then, focus on the promising areas and run a fine-tuning search (with early stopping!)
    - e.g. start with learning rate values: 0.1, 0.01, 0.001, 0.0001, 0.00001
    - If 0.001 achieves the most promision performance, run a tuning search between 0.01 and 0.0001 (in log space!)
  - Tune sensible combinations, e.g. learning rate and regularisation, number of units and dropout, etc.
- Analyse the results, especially wrong predictions to evaulate certain decisions, e.g. the used loss function
- Take a baseline and add more fancy methods to address the shortcomings

https://cs231n.github.io/neural-networks-3/

Run sanity checks

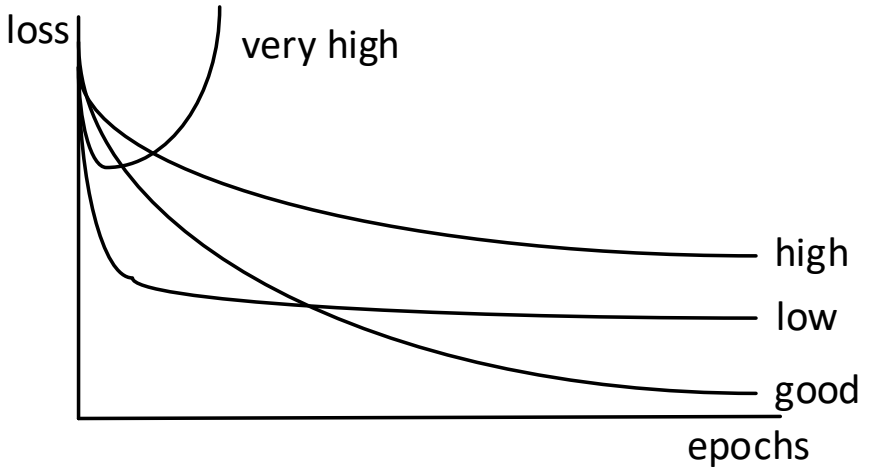
- Compute the expected loss for chance prediction and check if the initial loss is in that range, e.g. classification with 10 class categories and a Softmax classifier:
  - Set the regularisation strength to zero
  - For a randomly intialised network, we expect a diffuse probability of 0.1 for each of the ten class categories
  - The expected cross-entropy loss is then $-\ln(0.1) \approx 2.3$
- Increasing the regularisation strength should increase the loss
- Overfit on a tiny subset of the data, e.g. 20 samples, and try to achieve zero **training** loss
  - Set the regularisation strength to zero

https://cs231n.github.io/neural-networks-3/

Monitor the training loss

- Monitoring the **training** loss indicates how well the network architecture learns in general
- A wiggly (noisy) loss function is a symptom of a small batch size
  - Some noise is normal, but if the loss is very noisy you might want to increase the batch size (if possible)
- The loss also indicates if the learning rate is chosen well
- Also, if the initial **training** loss looks good but the **validation** loss is very high → might indicate a too high learning rate as well
- If network training is driving you nuts: https://lossfunctions.tumblr.com/

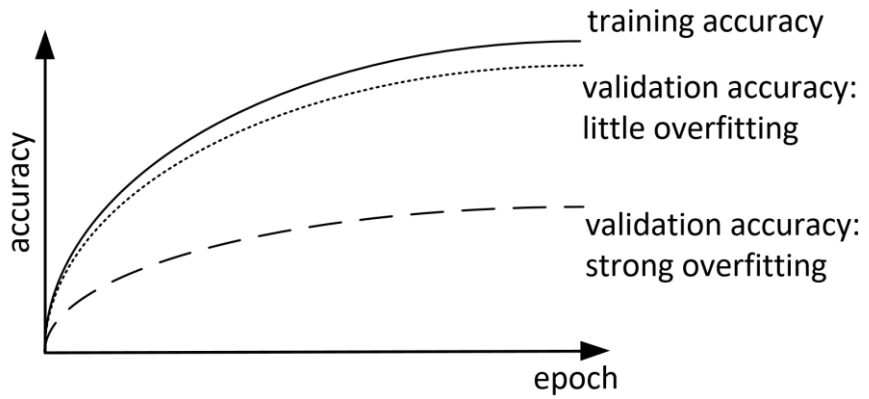
Monitoring the loss may indicate if the learning rate is too high or too low



https://cs231n.github.io/neural-networks-3/

Compare training and validation accuracy

- Monitoring the **training** accuracy with respect to the **validation** accuracy indicates whether the model starts to overfit
- Overfitting: the model learns the training dataset by heart and thus generalises badly to new samples. Thus, we are interested in a network with a high validation / test accuracy
- Overfitting is indicated by a high **training** accuracy in combination with a distinctively lower **validation** accuracy
- Overfitting with a very noisy validation accuracy / loss might also indicate that the training dataset is too small (the training data does not represent the validation data well)
- Reducing overfitting:
  - Early stopping (typically, overfitting increases with increasing training time)
  - Regularisation (Dropout, batch norm, L1/L2, …)
  - More training data
  - Use transfer learning to initialise the shallow layers (or all conv layers, if the dataset is similar)
  - Decrease the number of layers



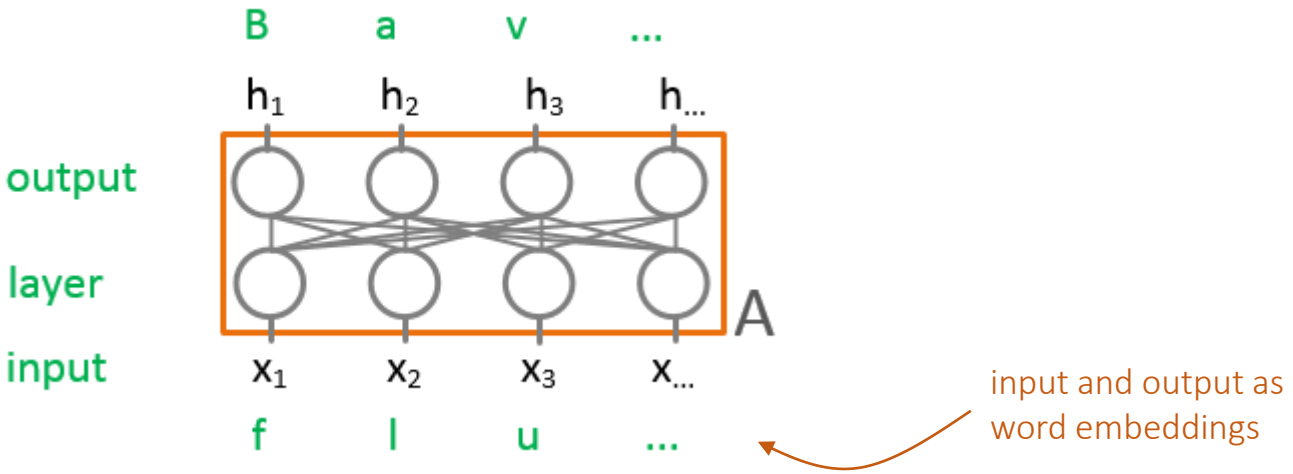
Network Architectures for Natural Language Processing (NLP)

Time series


„I grew up in Bavaria, in the beautiful city of Aschaffenburg with its castle and Pompejanum and many parks. Of course, I speak fluent *Bavarian*“

Task: Predict the next word in the sentence

vanilla neural network architecture

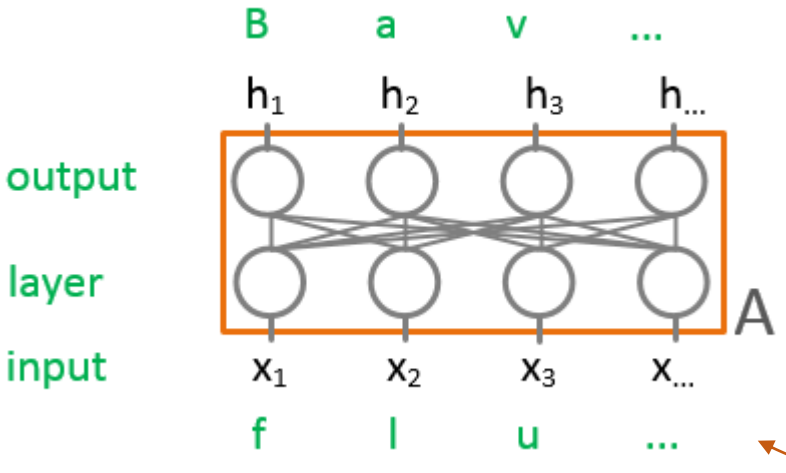


„I grew up in Bavaria, in the beautiful city of Aschaffenburg with its castle and Pompejanum and many parks. Of course, I speak fluent *?*„

Recurrent neural networks

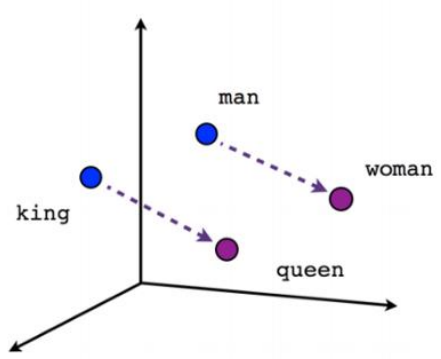
Task: Predict the next word in the sentence

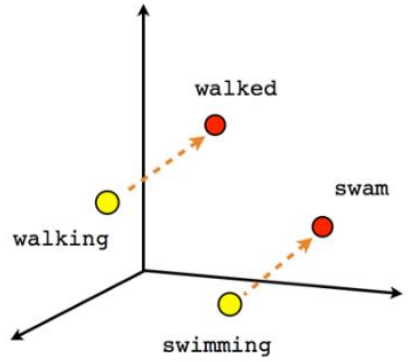
vanilla neural network architecture



input and output as word embeddings

word embeddings (GloVe, Word2Vec)





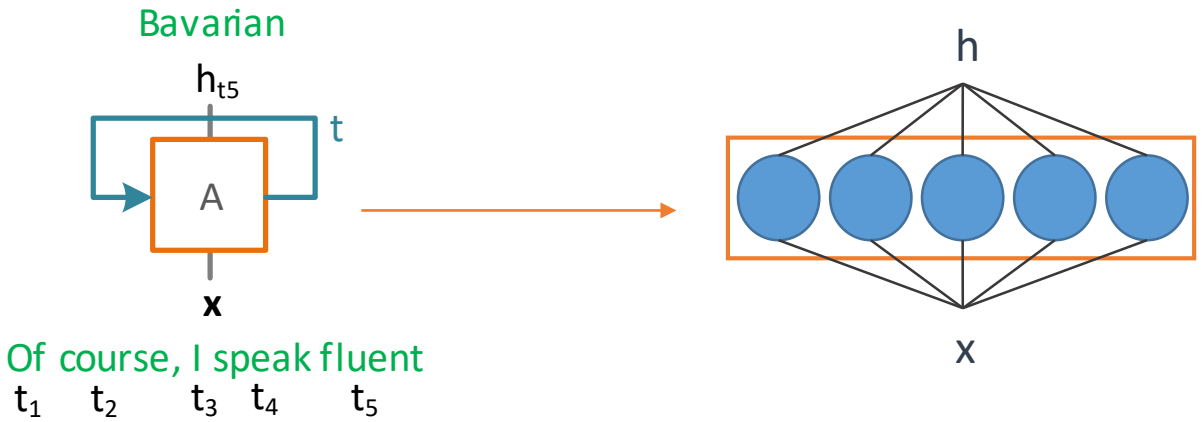
„I grew up in Bavaria, in the beautiful city of Aschaffenburg with its castle and Pompejanum and many parks. Of course, I speak fluent *?*„

https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8

Recurrent neural networks

- Task: Predict the next word in the sentence
- Recurrent neural networks have connections which point backwards in time

recurrent neural network architecture

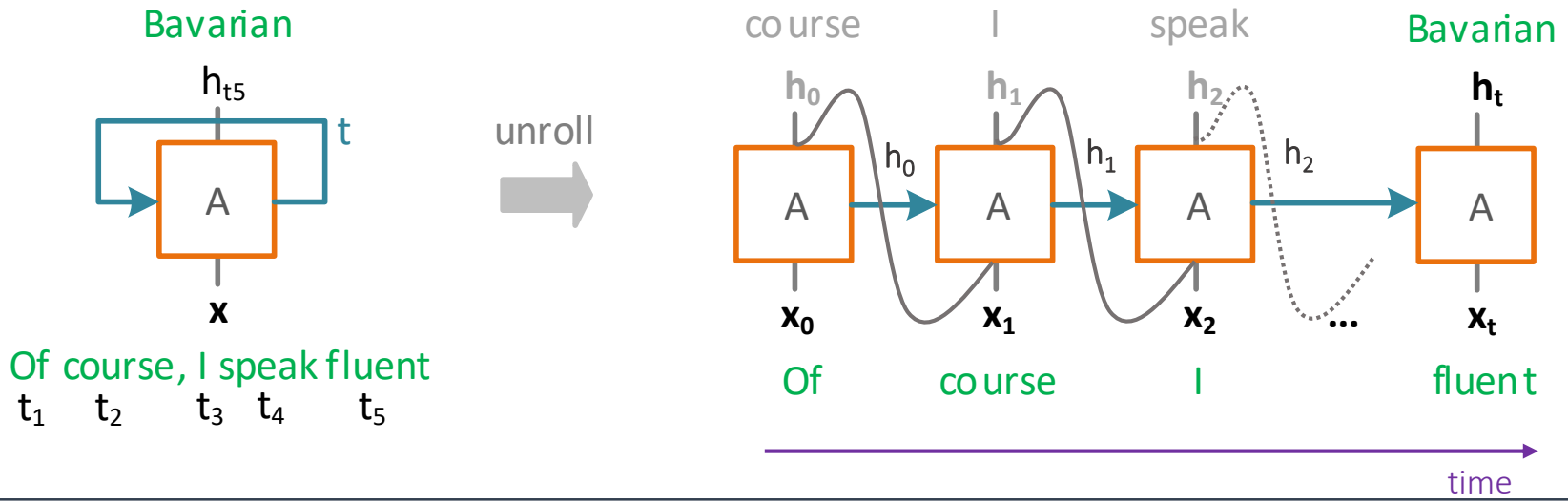


„I grew up in Bavaria, in the beautiful city of Aschaffenburg with its castle and Pompejanum and many parks. Of course, I speak fluent *?*„

Recurrent neural networks

- Task: Predict the next word in the sentence
- Recurrent neural networks have connections which point backwards in time
- At each time step t, a recurrent neuron receives its input $\boldsymbol{x}_t$ and its own output from the previous step, $\boldsymbol{h}_{t-1}$
- Accordingly, each recurrent neuron has two sets of weights, one for the inputs $\boldsymbol{x}_t$ and one for the outputs of the previous step $\boldsymbol{h}_{t-1}$: $h_t = \phi\left(W_x^{\mathsf{T}} x_t + W_h^{\mathsf{T}} h_{t-1} + b\right)$, where is ϕ the tanh or ReLU activation function

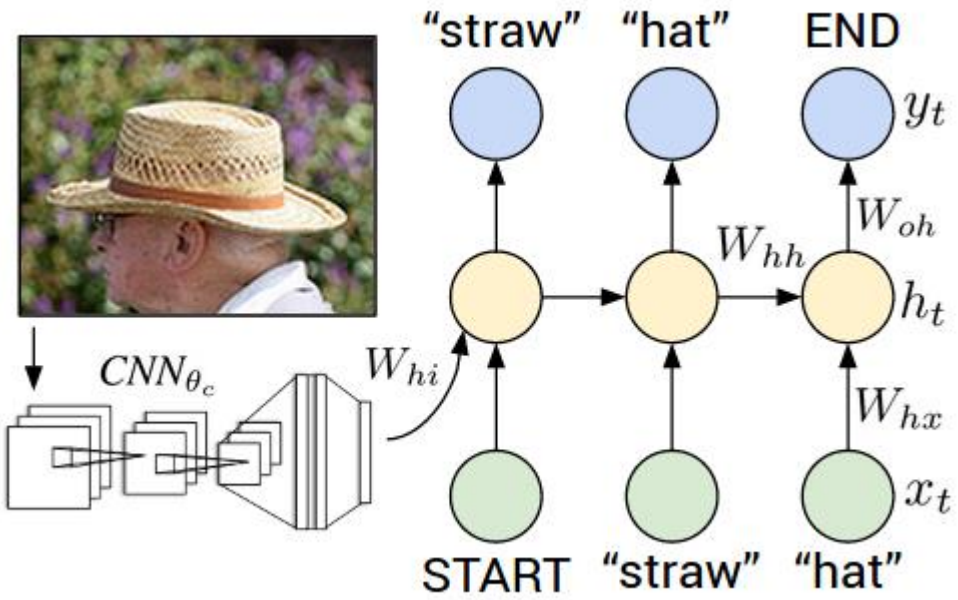
h_t is a function of all the previous inputs since time $t = 0$

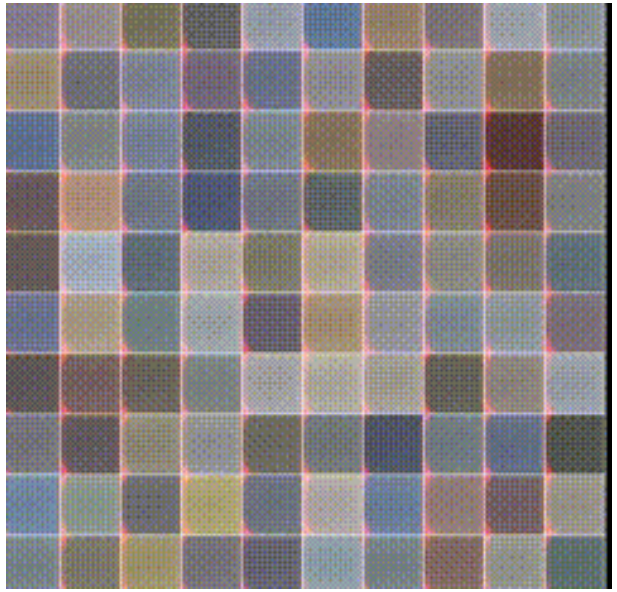


Machine Learning Research

Recurrent neural networks

Recurrent neural networks are the go-to network architecture for speech recognition, language modeling, translation, image captioning, timeseries, …



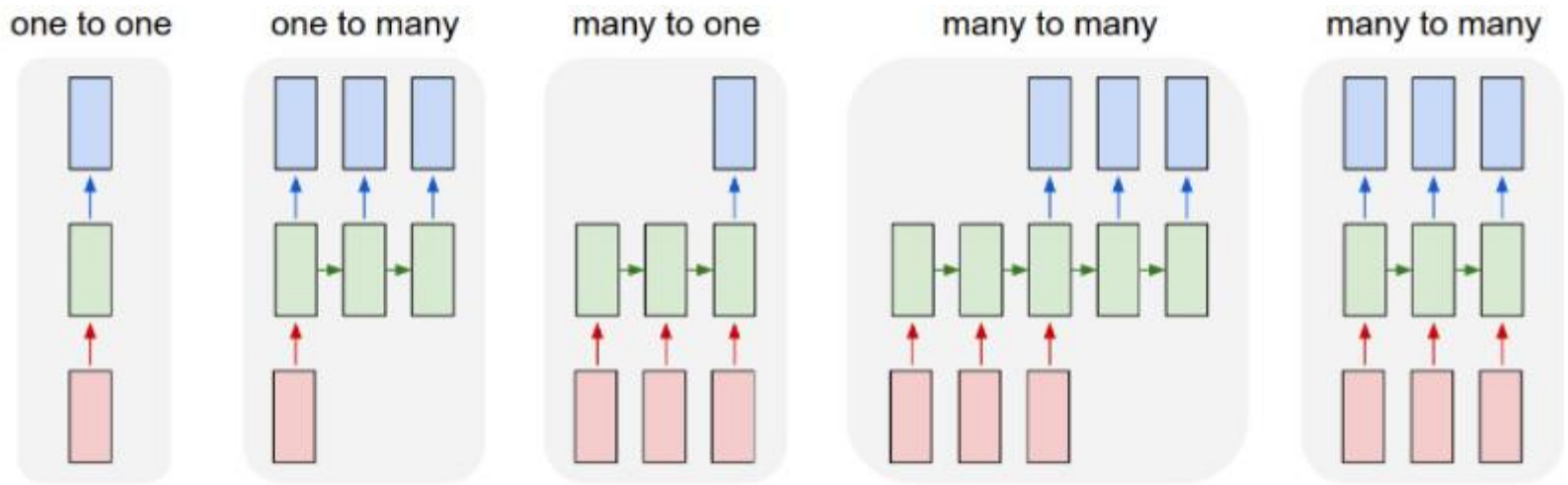


https://cs.stanford.edu/people/karpathy/
https://karpathy.github.io/2015/05/21/rnn-effectiveness/
https://arxiv.org/abs/1502.04623

Machine Learning Research

Recurrent neural networks

… because they are flexible regarding input and output dimensions



Vanilla mode, e.g. CNNs

Sequence output, e.g. image captioning

Sequence input, e.g. sentiment analysis

Sequence input and output, e.g. machine translation

Synced sequence input and output, e.g. video classification where every frame of the video will be labelled

https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent neural networks

Training examples: generate new text character-wise, based on Leo Tolstoy‘s War and Peace

100 iterations

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

700 iterations

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

1200 iterations

```
"Kite vouch!" he repeated by her
door. "But I would be done and quarts, feeling, then, son is people...."
```

2000 iterations

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent neural networks

Training examples: generate new text character-wise, based on Latex source code for an algebraic stacks book

Vanilla RNNs lack long-term memory

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma **??**. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccc} \mathcal{S} & \longrightarrow & \\ \downarrow & & \\ \xi & \longrightarrow & \mathcal{O}_{X'} \\ & & \uparrow \\ \mathrm{gor}_s & & \end{array} \qquad \begin{array}{ccc} & & \\ & \searrow & \\ & & \end{array}$$

$$\begin{array}{ccc} = \alpha' & \longrightarrow & \\ \updownarrow & & \\ = \alpha' & \longrightarrow & \alpha \end{array} \qquad \begin{array}{c} X \\ \downarrow \\ \end{array}$$

$$\mathrm{Spec}(K_\psi) \qquad \mathrm{Mor}_{Sets} \quad \mathrm{d}(\mathcal{O}_{\mathcal{X}_{X/k}}, \mathcal{G})$$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas **??**.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_t}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{w}})$$

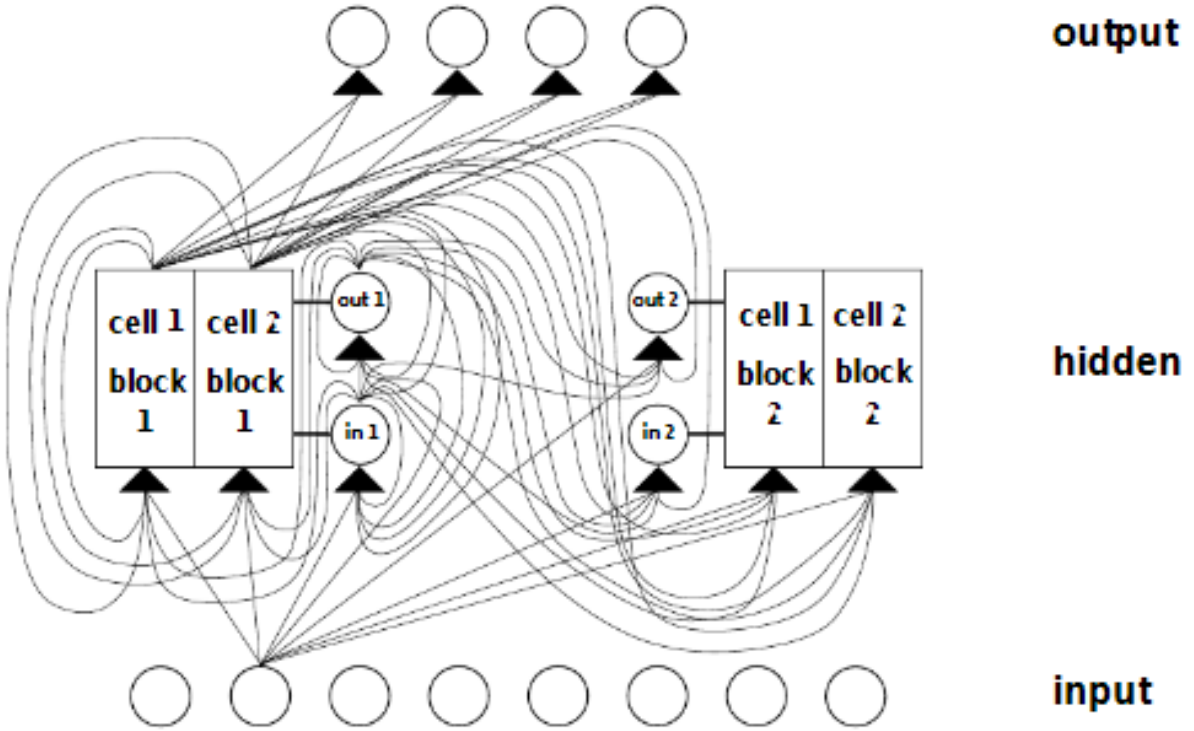
is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition **??** and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma **??**. This is a sequence of $\mathcal{F}$ is a similar morphism.

https://karpathy.github.io/2015/05/21/rnn-effectiveness/

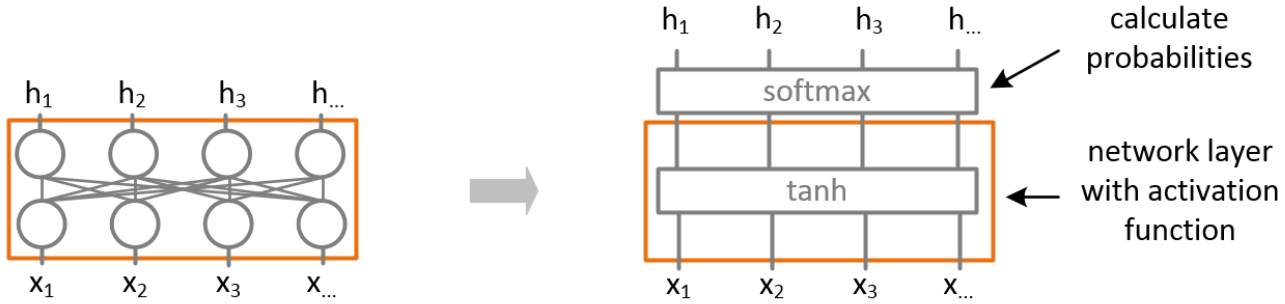
LSTMs

- Long short-term memory networks improve the memory of recurrent networks

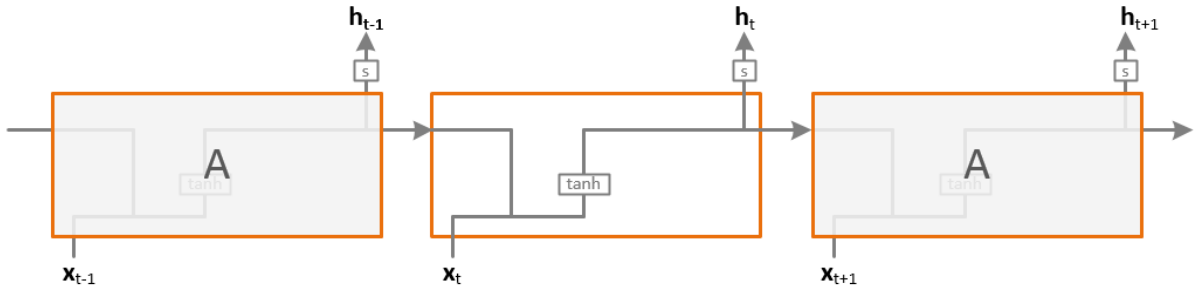


Hochreiter & Schmidhuber. Long short term memory. (1997) https://www.bioinf.jku.at/publications/older/2604.pdf
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

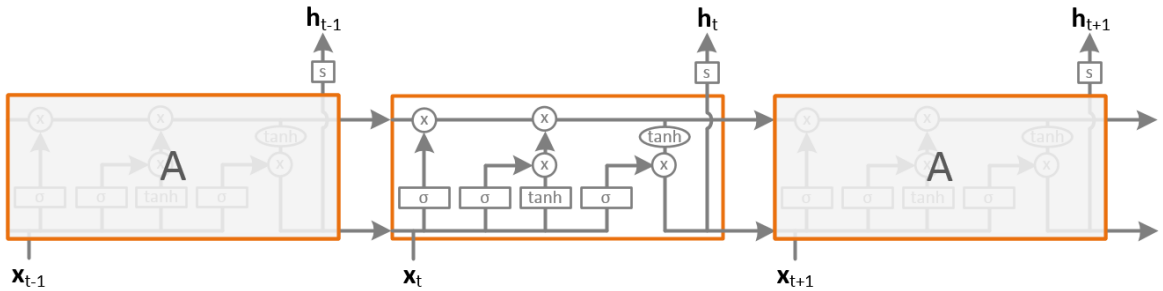
LSTMs



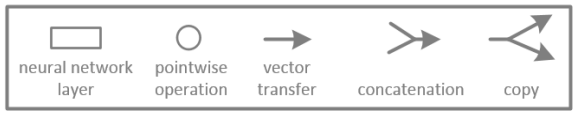
vanilla neural network architecture



vanilla recurrent network architecture



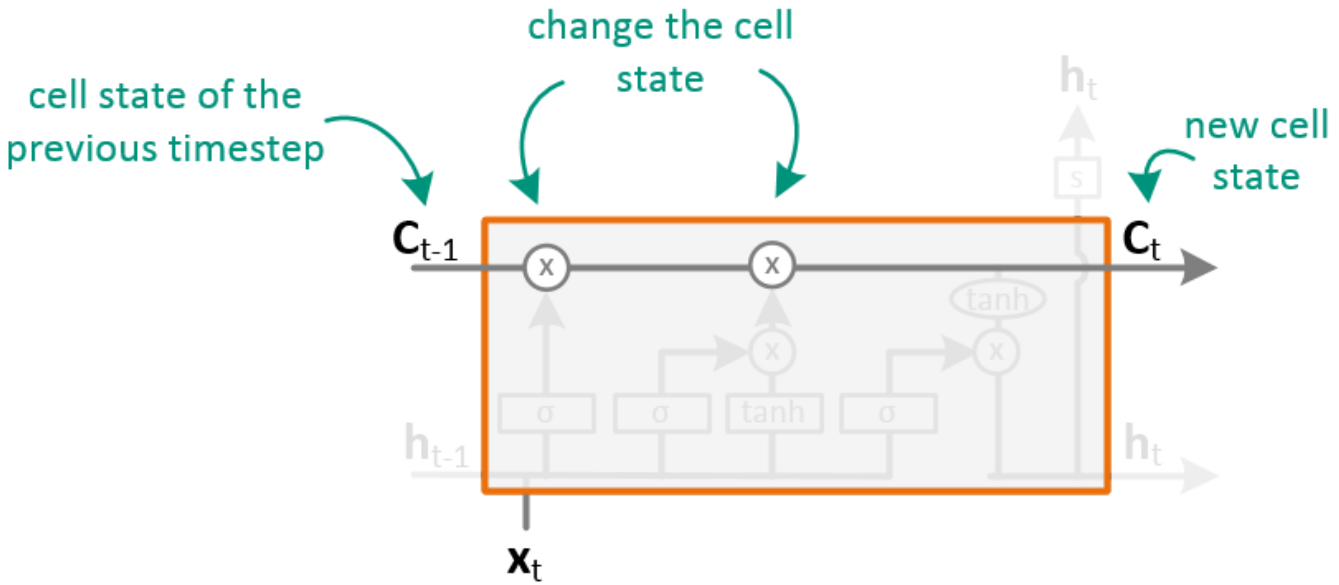
LSTM network architecture



Machine Learning Research

Cell state

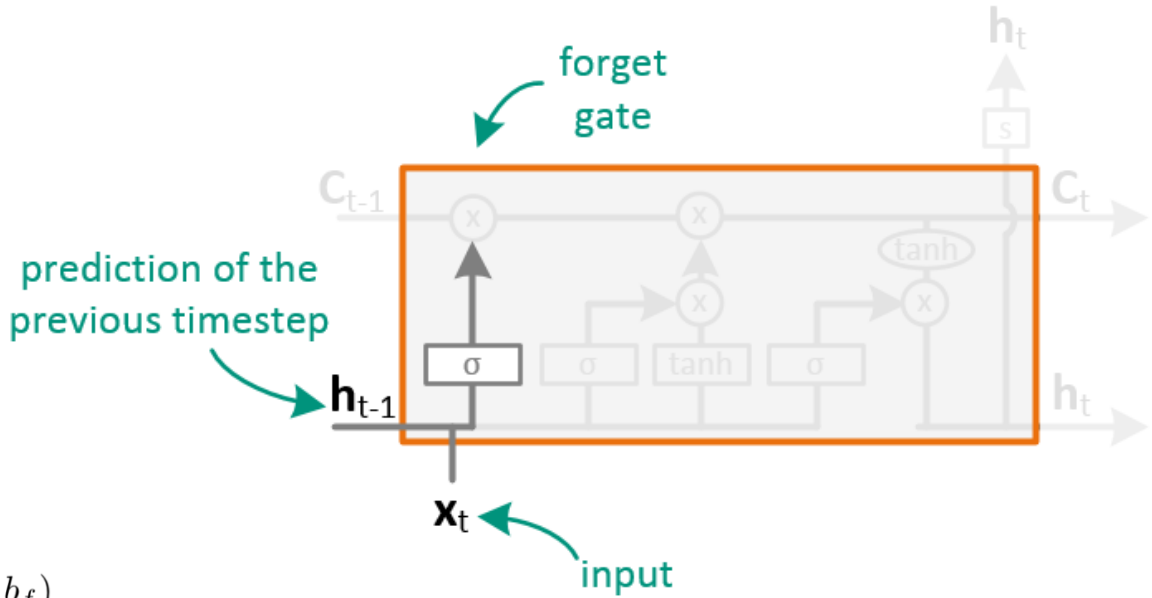
- The cell state stores the information and passes it to the next timestep
- Can be changed by so-called gates



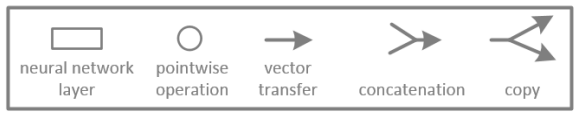
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Forget Gate

- decides which values will be updated based on the current input and previous prediction
- outputs values between 0 (forget everything) and 1 (keep information)



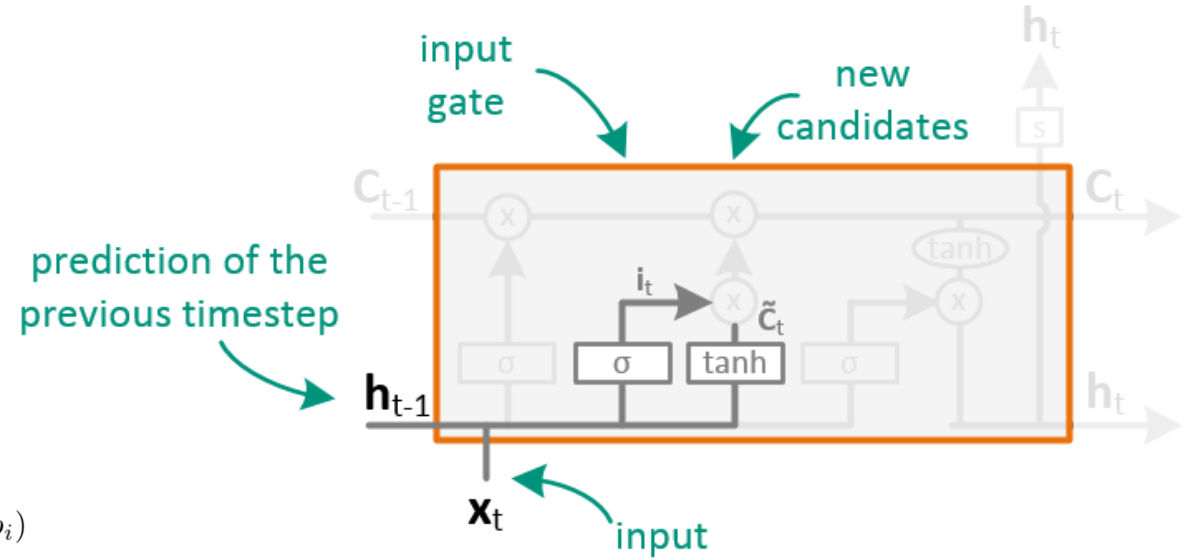
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

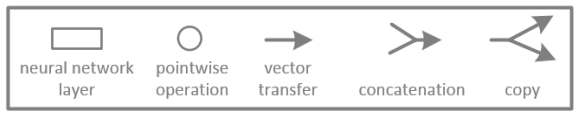
Update Gate, composed of two parts:

- input gate: decides, which information to update and to what scale
- tanh layer: calculates new candidates



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

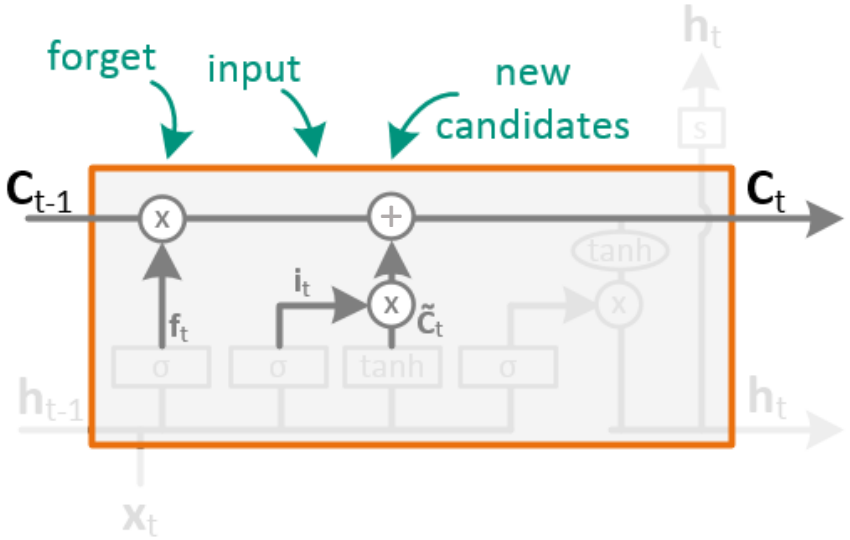
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



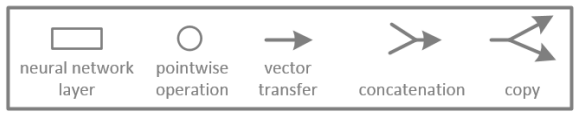
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Update the Cell State

- delete obsolete information from the cell state
- add the selection of new candidates



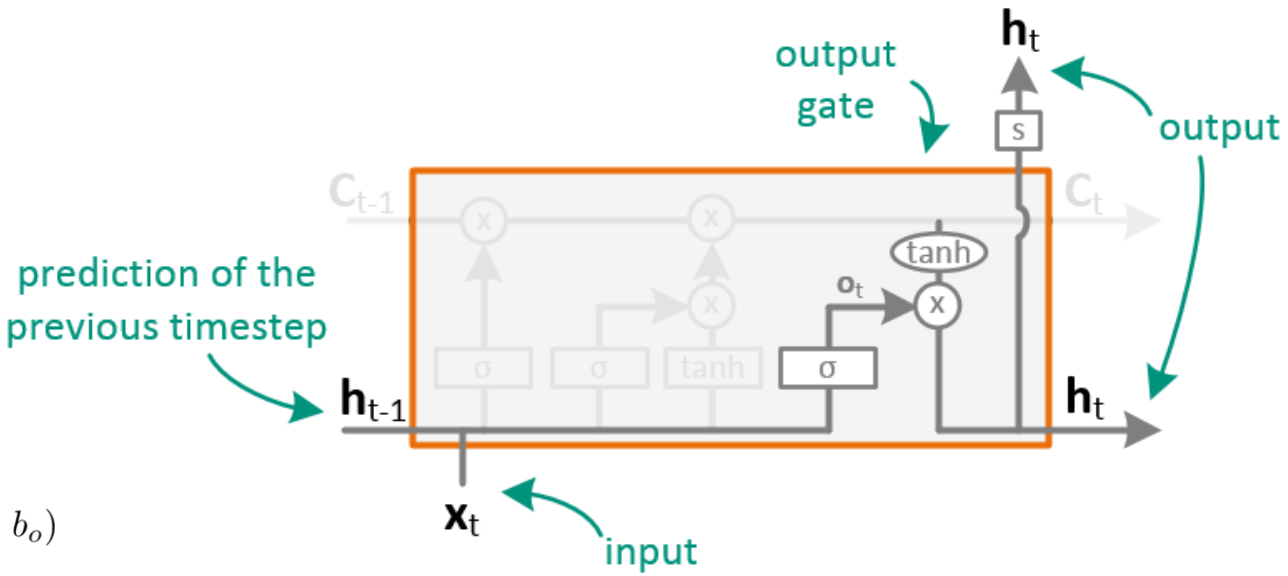
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

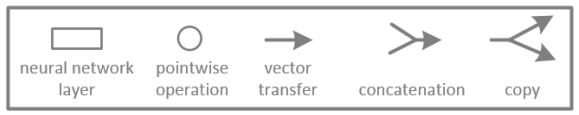
Output Gate

- determines which information to output, based of the previous timestep‘s prediction and the input
- tanh squeezes the cell state information between -1 and 1



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

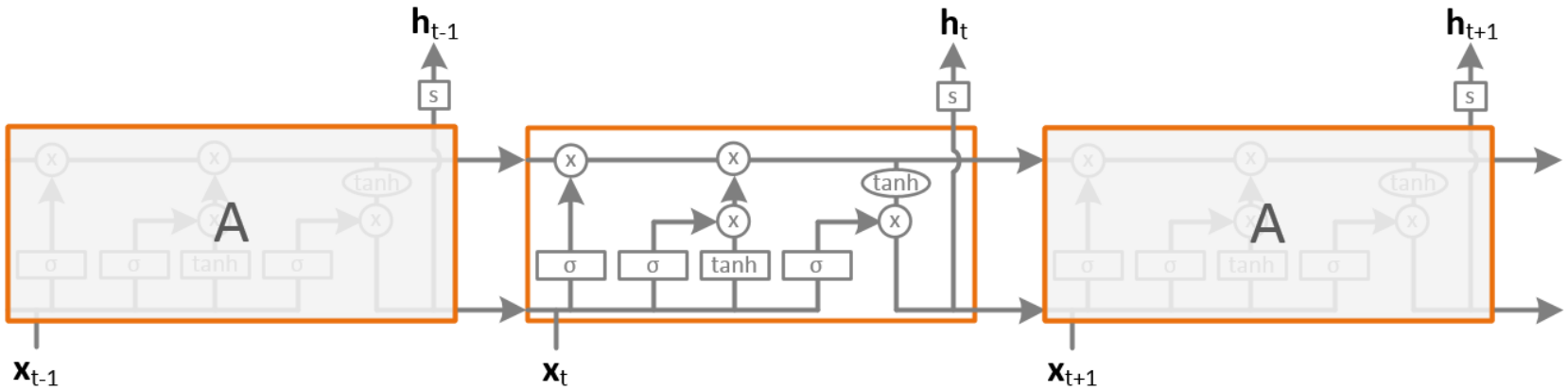
$$h_t = o_t * \tanh\left(C_t\right)$$



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs & Variants

- There are various adaptations of LSTM cells, such as GRU (Gated Recurrent Unit), which simplify LSTMs
- Which works best depends on the dataset



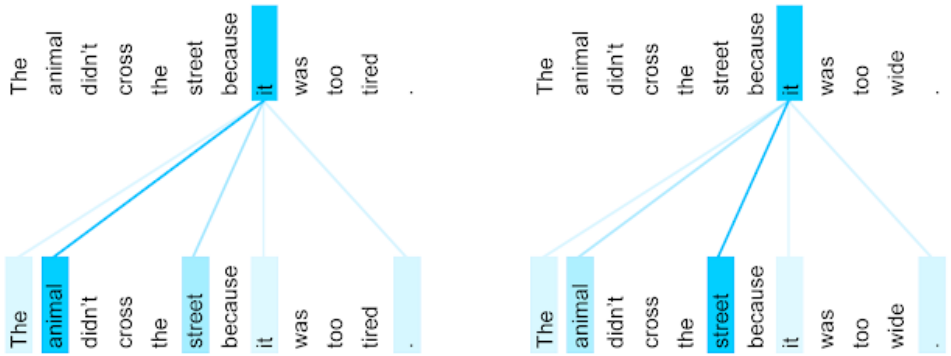
Transformers

Current state-of-the-art networks for natural language processing, e.g. GPT-3, T5, BERT, use neither convolutional layers nor recurrent layers but the Transformer architecture, which improves upon the long-term memory

- GPT-NeoX by Eleuther
  - www.eleuther.ai/
- Google BERT and T5
  - https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb
  - https://github.com/google-research/text-to-text-transfer-transformer
- DistilBERT by Huggingface
  - https://medium.com/huggingface/distilbert-8cf3380435b5
- GPT-3 by OpenAI
  - https://openai.com/blog/openai-api/

Transformer networks

- Transformers are encoder-decoder architectures for sequence-to-sequence (seq2seq) applications, e.g. language translation
- The encoder maps the input (source sentence) into a higher dimensional space (representation)
- This representation is then fed into the decoder, which turns it into an output sentence (target sentence)



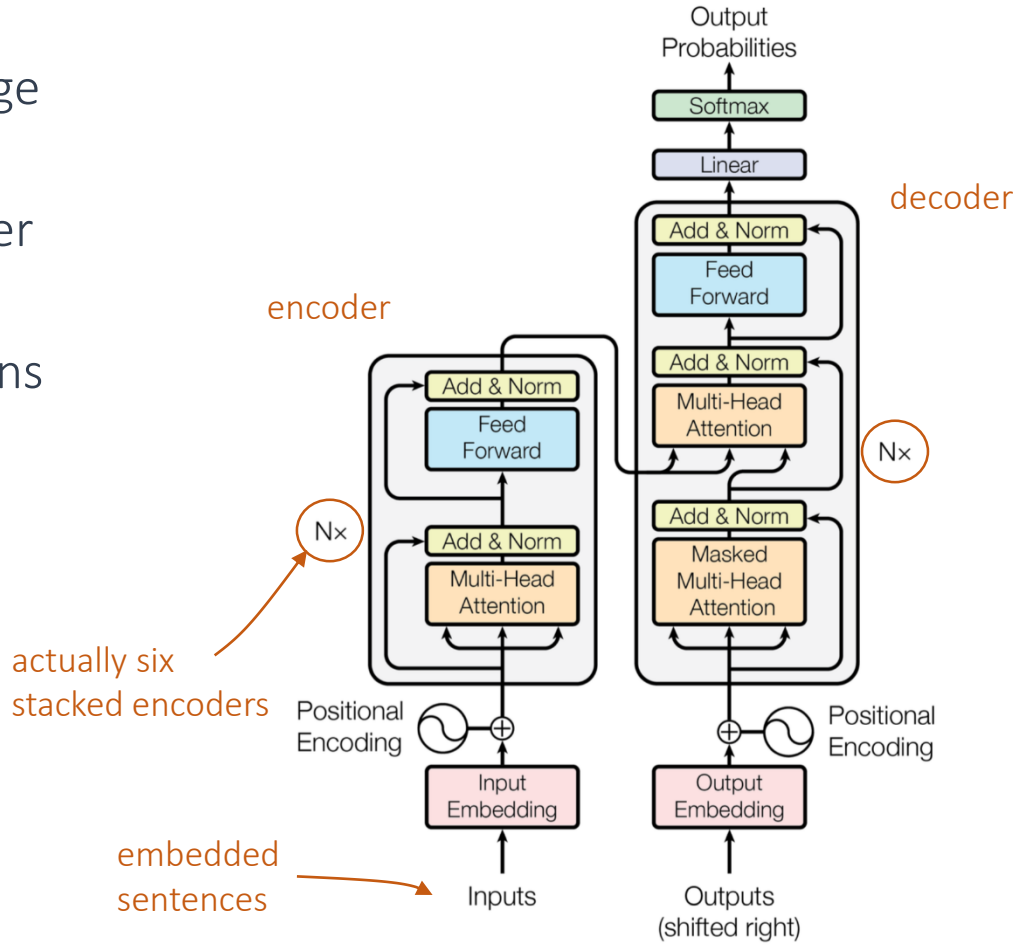


Figure 1: The Transformer - model architecture.

Attention: Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate https://arxiv.org/abs/1409.0473
Transformer: Vaswani et al. Attention is all you need. (2017) https://arxiv.org/abs/1706.03762
https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Resources

CNNs, Explainable AI, RNNs:

- https://karpathy.github.io/
- https://distill.pub/
- https://colah.github.io/

Computer vision:

- http://cs231n.stanford.edu/

Natural Language Processing

- https://web.stanford.edu/class/cs224n/
- https://huggingface.co/course/chapter1