Deep learning

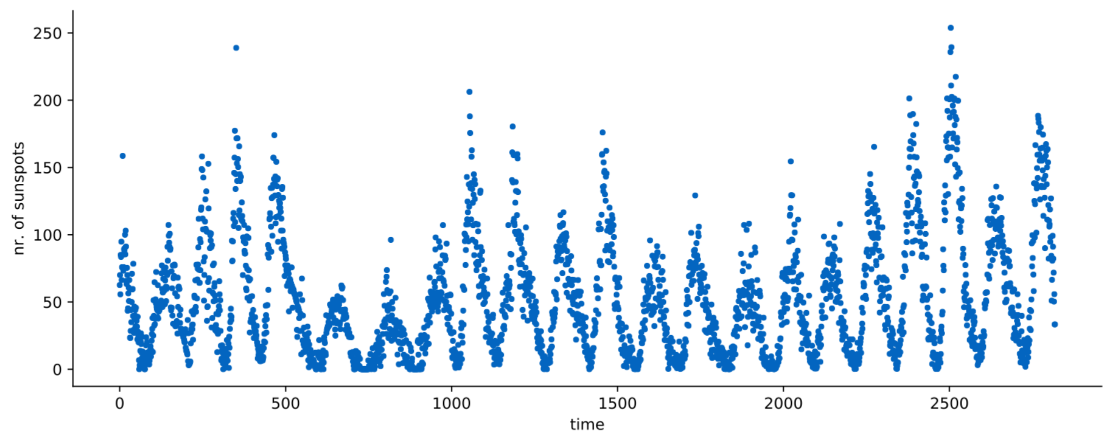
CNNs

Today’s program

- 14:00-14:30 Recap
- 14:30-14:45 Improving RNNs: regularization, stacking, stateful and bi-directional RNNs
- 14:45-15:15 Hands-on: Improved RNNs on temperature prediction
- 15:15-15:45 Image processing, Convolutional Neural Networks (CNN)
- 15:45-16:15 Break
- 16:15-17:00 Hands-on: CNN on fashion MNIST
- 17:00-17:30 Transfer Learning
- 17:30-18:00 Hands-on: transfer learning with pre-trained ResNet-50 layers for fashion MNIST
- 18:00-19:00 Diner
- 19:00-19:30 Unsupervised learning, the (variational) autoencoder (VAE)
- 19:30-20:15 Hands-on: auto-encoder on MNIST
- 20:15-20:45 Generative adversarial networks (GAN)
- 20:45-21:00 Summary, final questions, etc

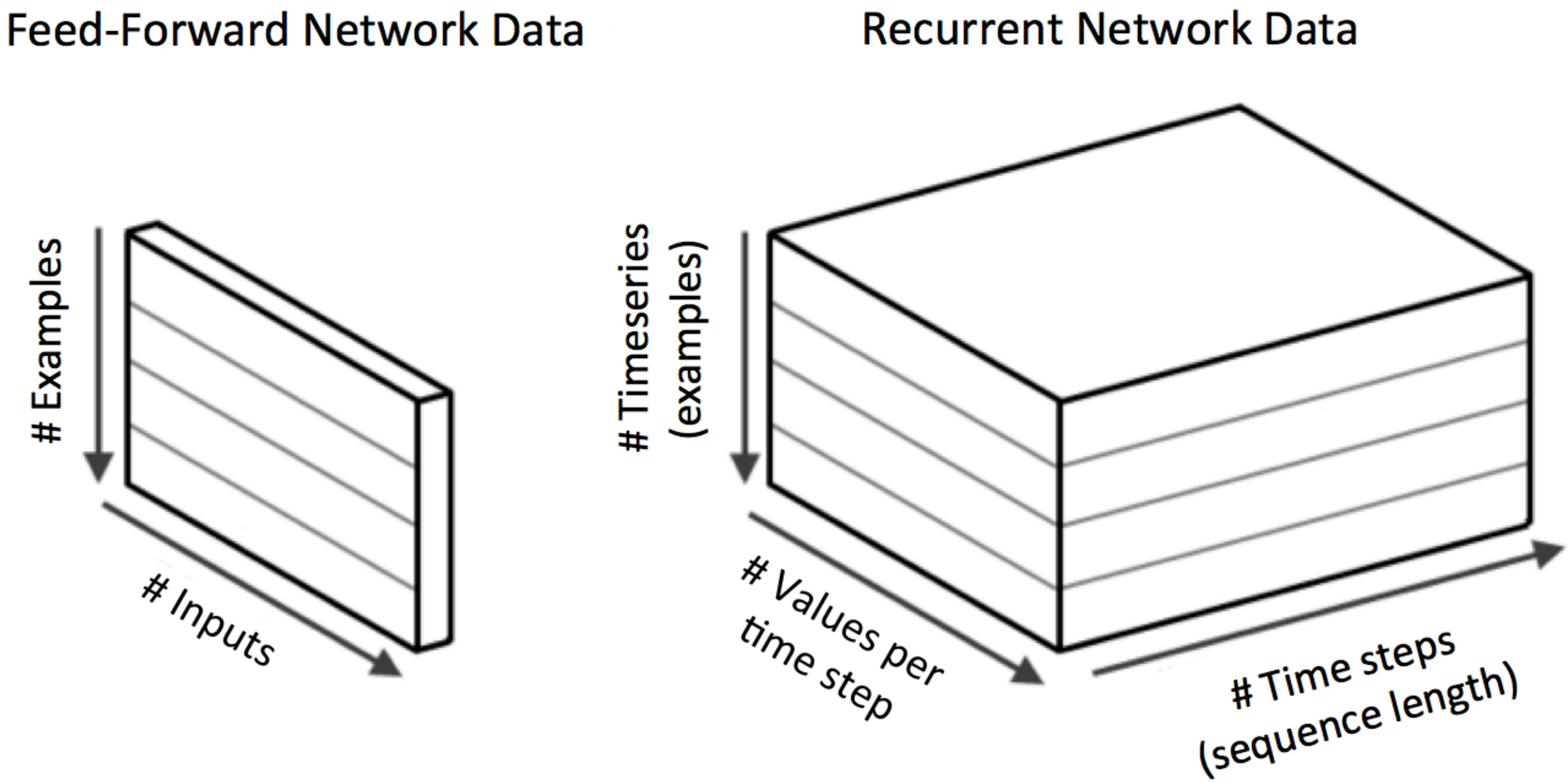
Recap RNNs

- Last time, we looked at analyzing sequential data



the cat sat on the mat .
the book is open .

Recap RNNs



Data = (examples, sequence_length, features)

Source: https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html

Recap RNNs

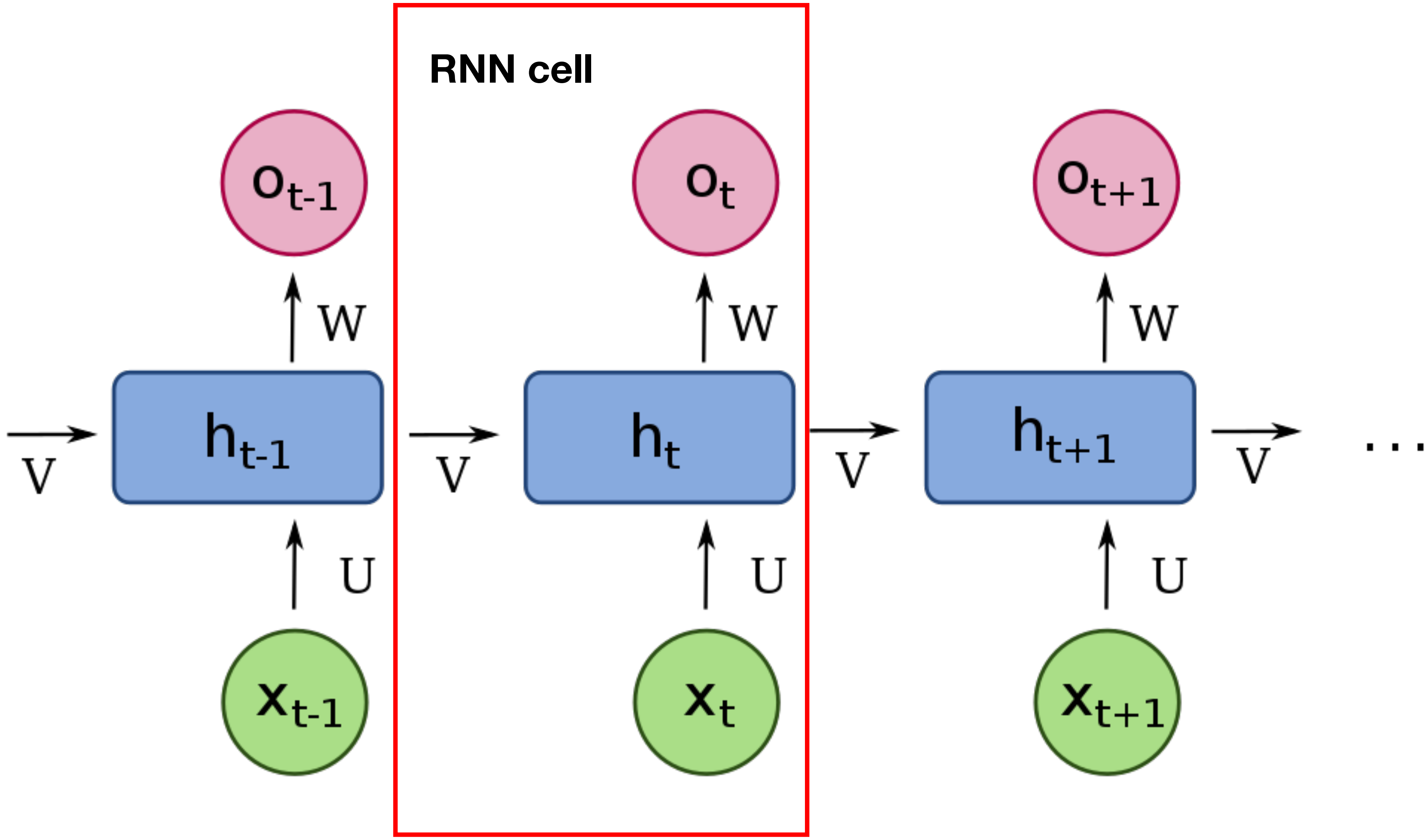
Applications

- Many-to-one, e.g. predict next word
- Many-to-many, e.g. machine translation
- One-to-many, e.g. automatic image captioning

- Note: Many-to-one can also be done iteratively!
    - The
    - The cat
    - The cat is
    - The cat is sitting …

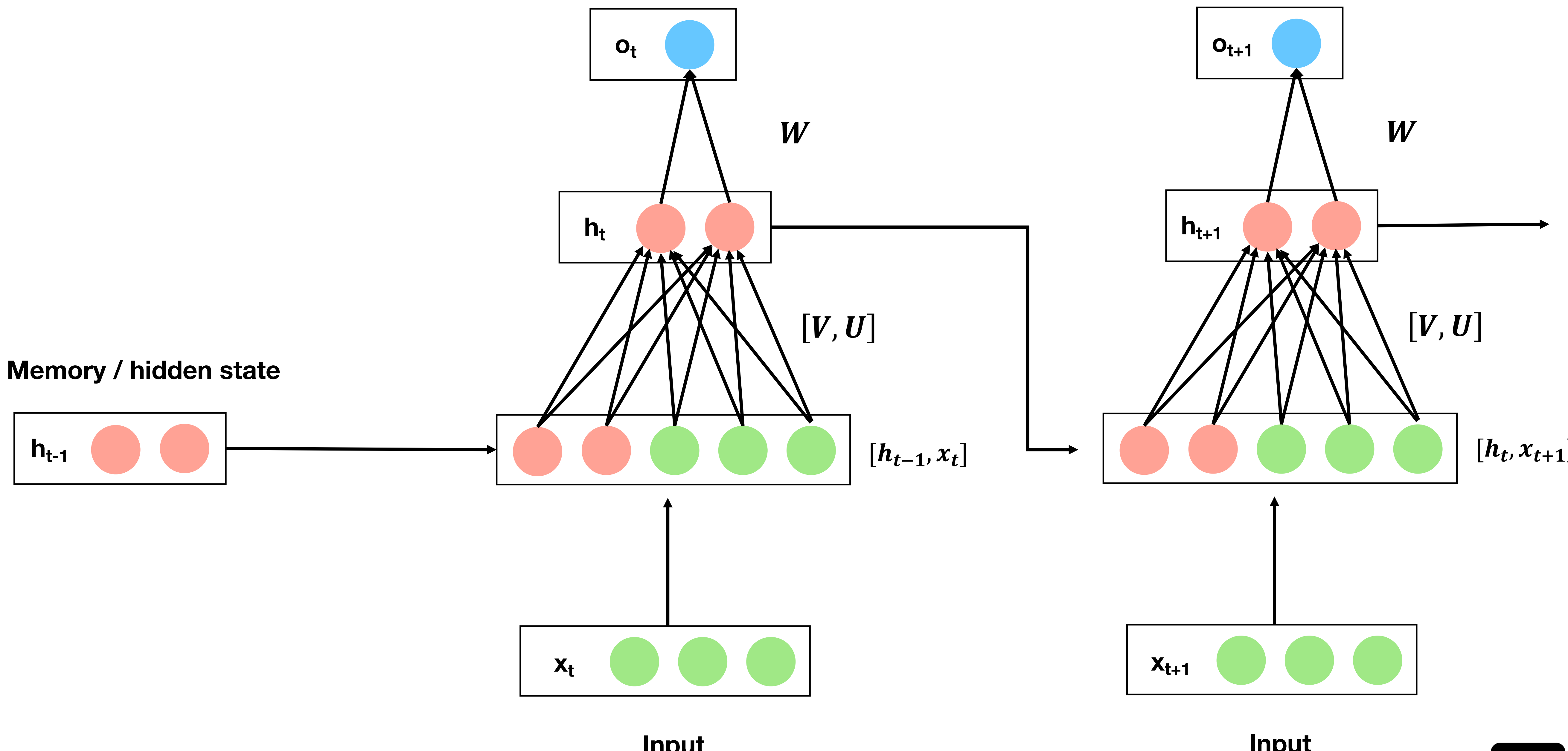
Simple RNN

- U, V, W are the weight matrices
- Weights are reused!



Simple RNN

- Another way to look at it…

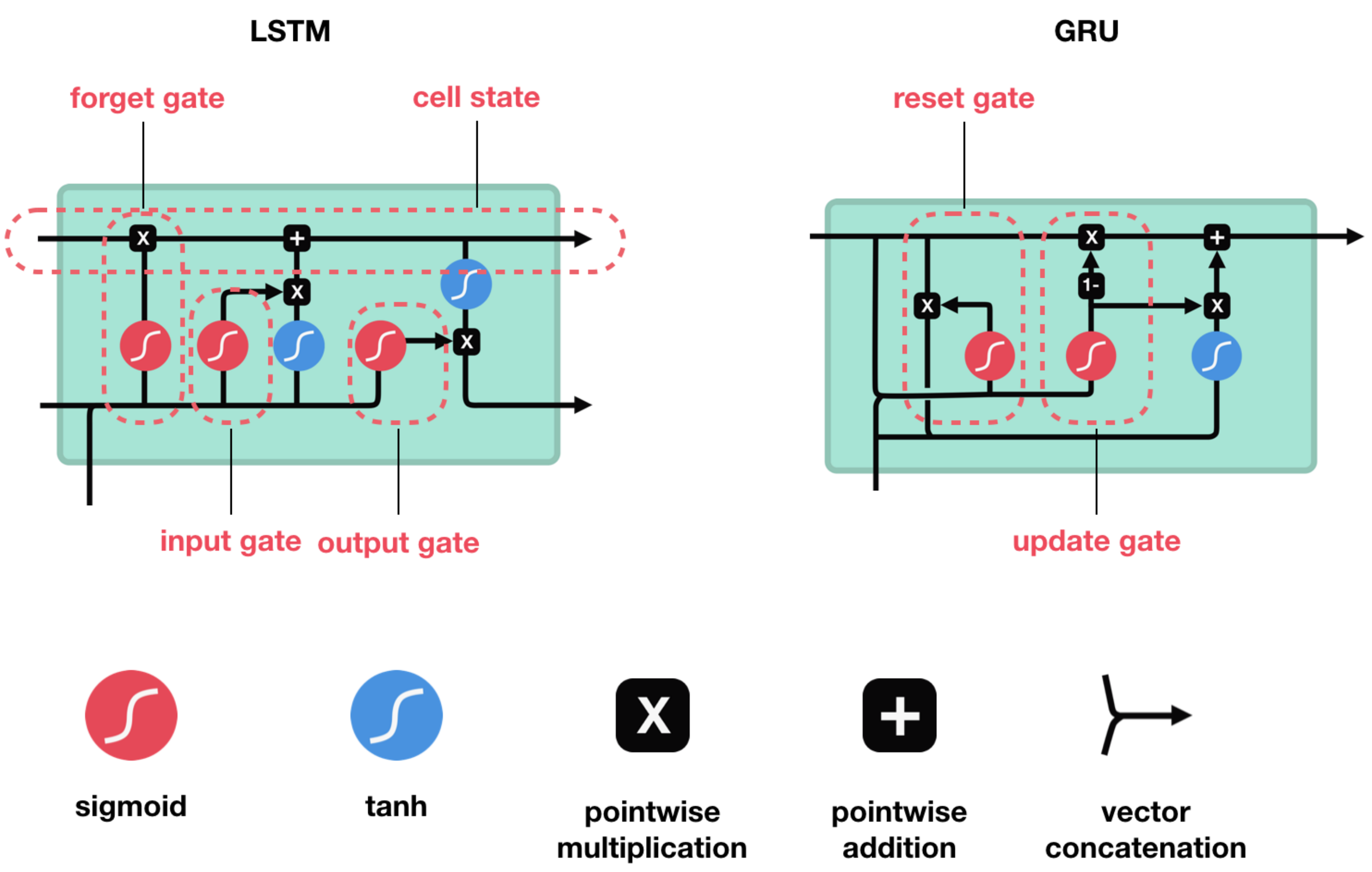


Recap RNNs

Dense networks	**RNN**
Can’t process sequence of arbitrary length	Can process sequence of arbitrary length
Doesn’t account for ordering of input. E.g. hard to learn that *Donald* is likely followed by *Trump*. Would have to learn this for each position in the sequence separately	Does account for ordering of input. It will predict similar outcomes given a similar context. E.g. *Donald* is generally followed by *Trump*, irrespective of position in the sequence.
Lots of parameters	Reuse of parameters for each time step in the sequence

Recap / Questions?

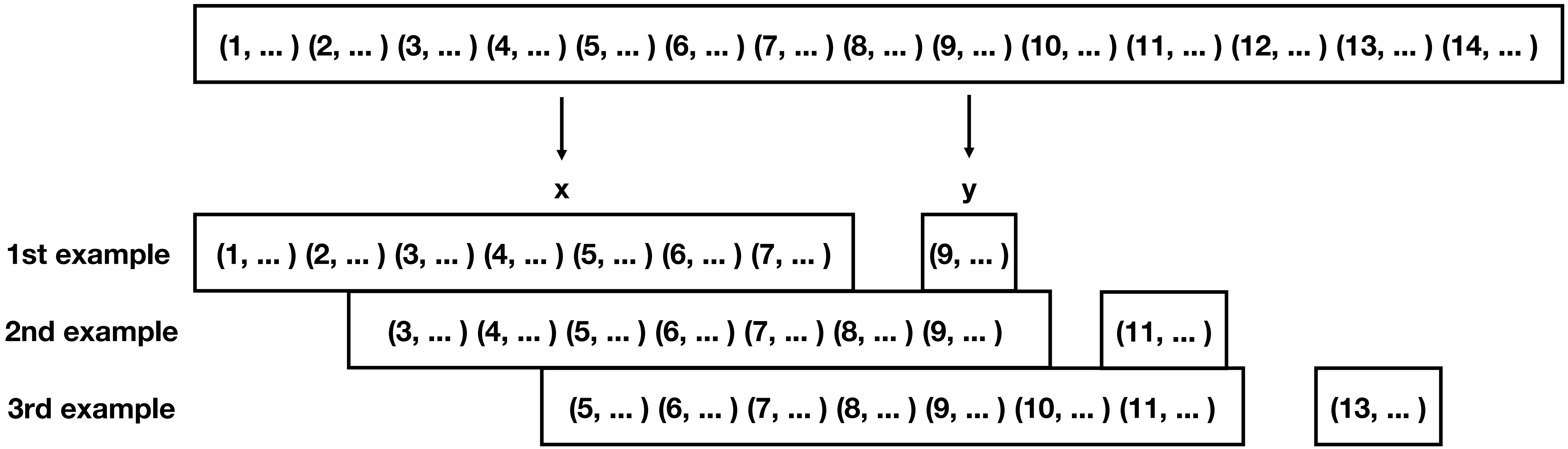
- Basic **RNN** has a very short term memory
- **GRU** and **LSTM** have a longer memory, but are more complex (heavier to train)



Recap / Questions?

- Generating sequences
- Choices between consecutive / overlapping sequences (shift), sequence length, target shift…

Shift approach, using shift = 2, sequence length = 7, target shift = 1



Today’s program

- 14:00-14:30 Recap
- 14:30-14:45 Improving RNNs: regularization, stacking, stateful and bi-directional RNNs
- 14:45-15:15 Hands-on: Improved RNNs on temperature prediction
- 15:15-15:45 Image processing, Convolutional Neural Networks (CNN)
- 15:45-16:15 Break
- 16:15-17:00 Hands-on: CNN on fashion MNIST
- 17:00-17:30 Transfer Learning
- 17:30-18:00 Hands-on: transfer learning with pre-trained ResNet-50 layers for fashion MNIST
- 18:00-19:00 Diner
- 19:00-19:30 Unsupervised learning, the (variational) autoencoder (VAE)
- 19:30-20:15 Hands-on: auto-encoder on MNIST
- 20:15-20:45 Generative adversarial networks (GAN)
- 20:45-21:00 Summary, final questions, etc

Improving RNNs

- Regularisation
  - L1/L2
  - Dropout, recurrent dropout
- Improving RNNs
  - Stacking
  - Stateful
  - Bi-directional

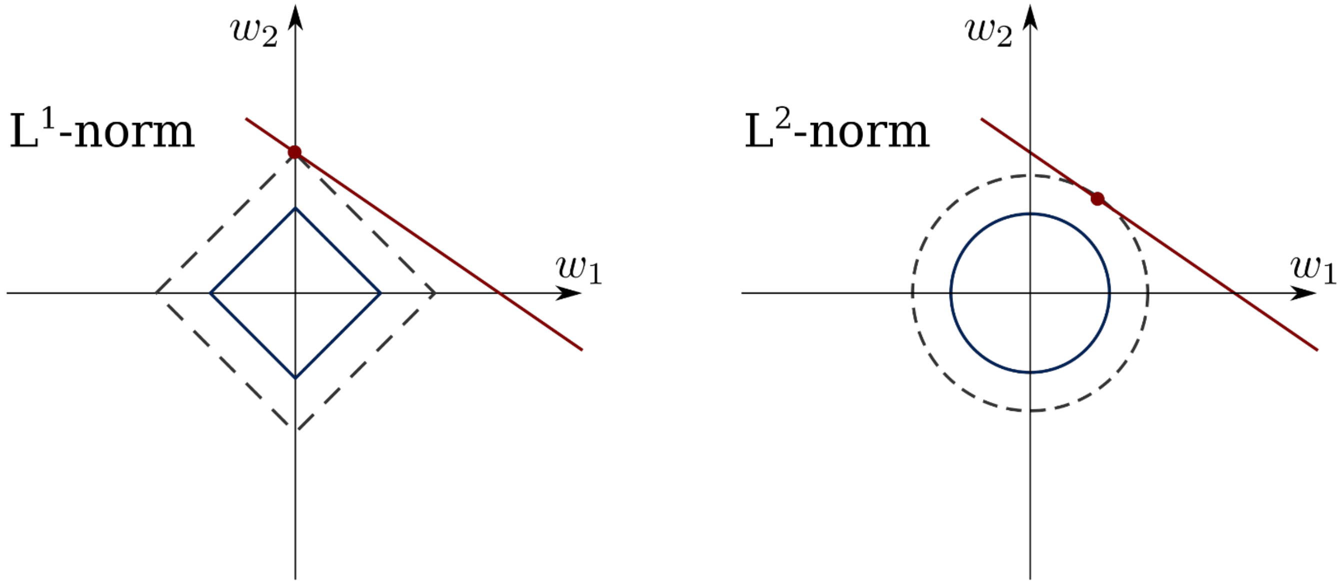
Improving RNNs

L2/L1 regularisation

- Just like with normal dense layers.
- we add L2/L1 regularisation to the weights learnt in the RNN cell.

layer_gru(units = 10, kernel_regularizer = regularizer_l2(l = 0.001))

layer_gru(units = 10, kernel_regularizer = regularizer_l1(l = 0.001))



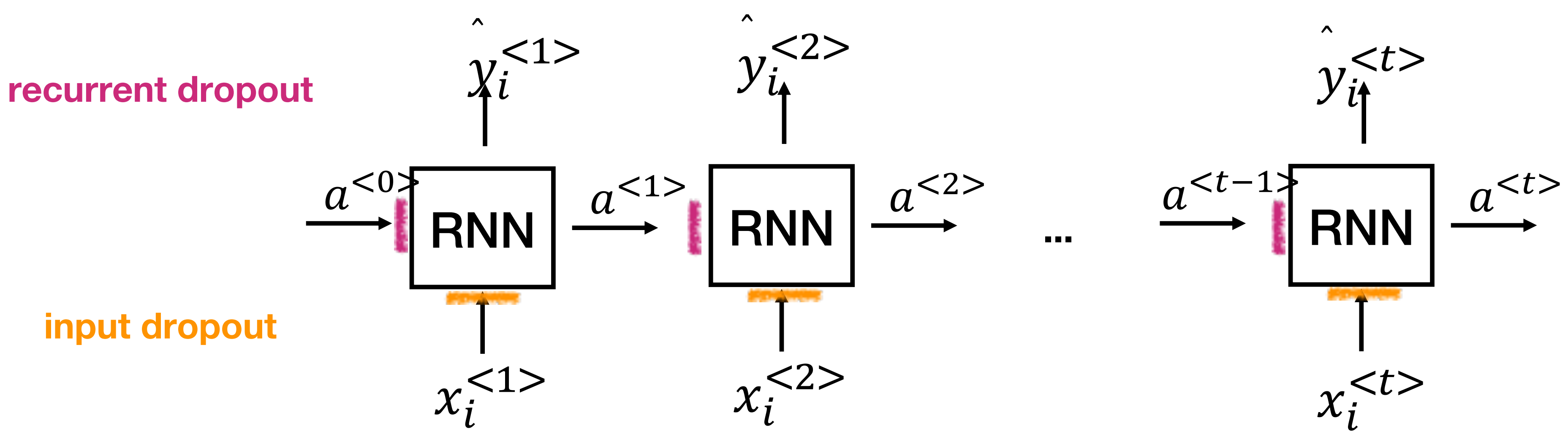
Red line: identical predictions of the NN (too many degrees of freedom)
Red point: solution for (w_1, w_2) preferred by each norm.

Improving RNNs

Dropout

- In RNNs we consider dropouts in two locations.
  - **Input**
  - **Recurrent dropout**

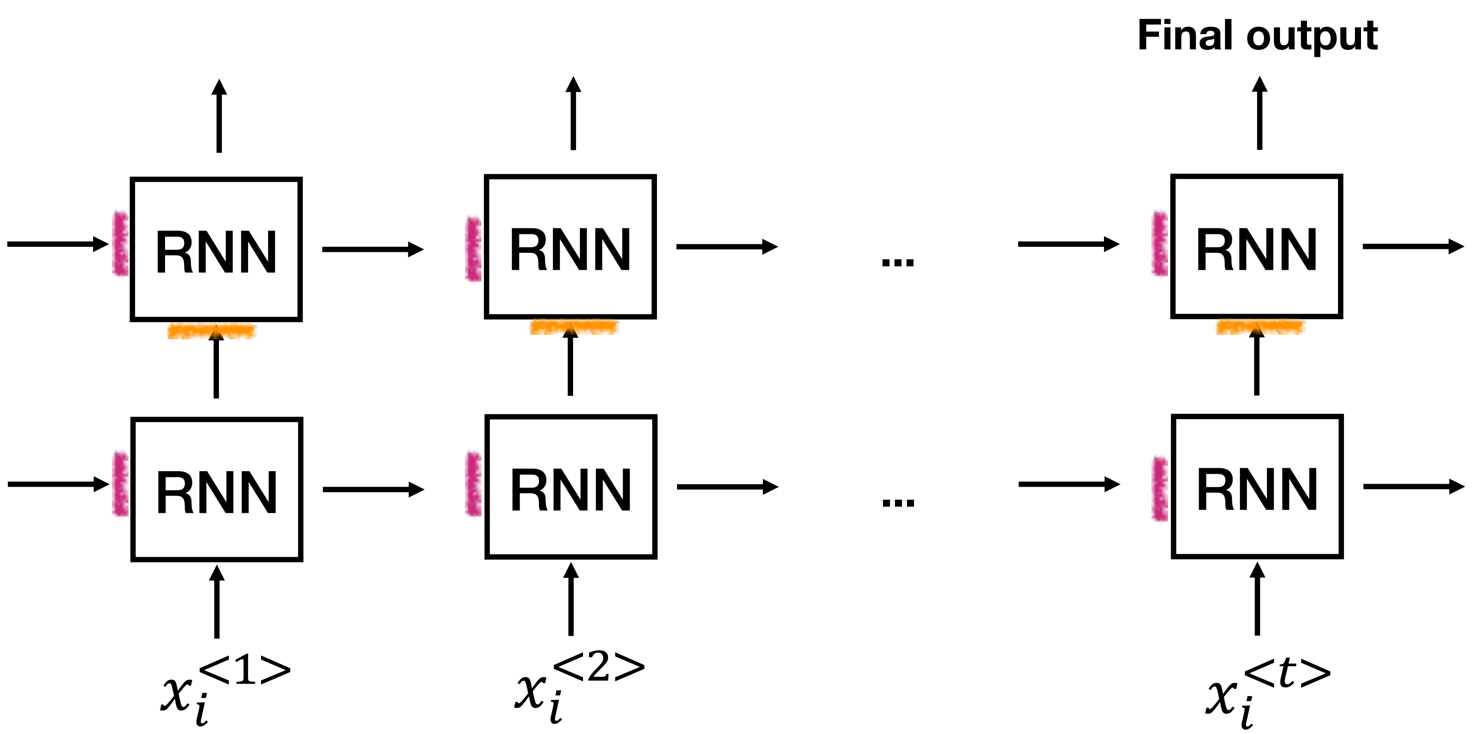
layer_gru(units = 10, dropout = 0.2, recurrent_dropout = 0.3)



Improving RNNs

Stacking RNNs

- Why would we consider input dropout?
    - Maybe in production we might not always get all inputs.
- More likely, we are stacking RNNs.
- Stacking RNNs is like adding additional layers in a dense network.
    - We never go that deep, 1-6 layers. Long training time.

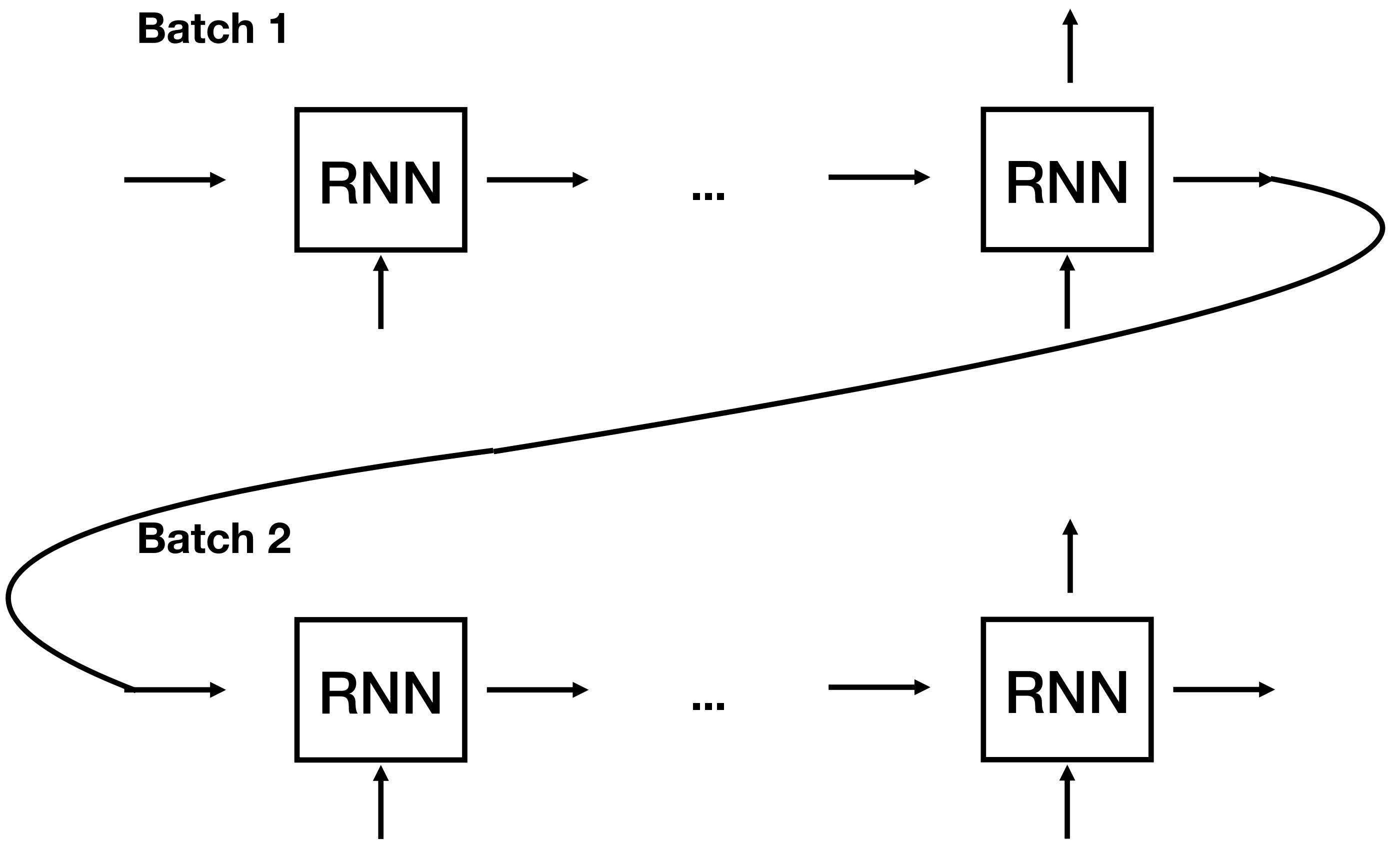



```
layer_gru(units = 10, return_sequences = TRUE) %>%
  layers_gru(units = 10)
```

Improving RNNs

Stateful

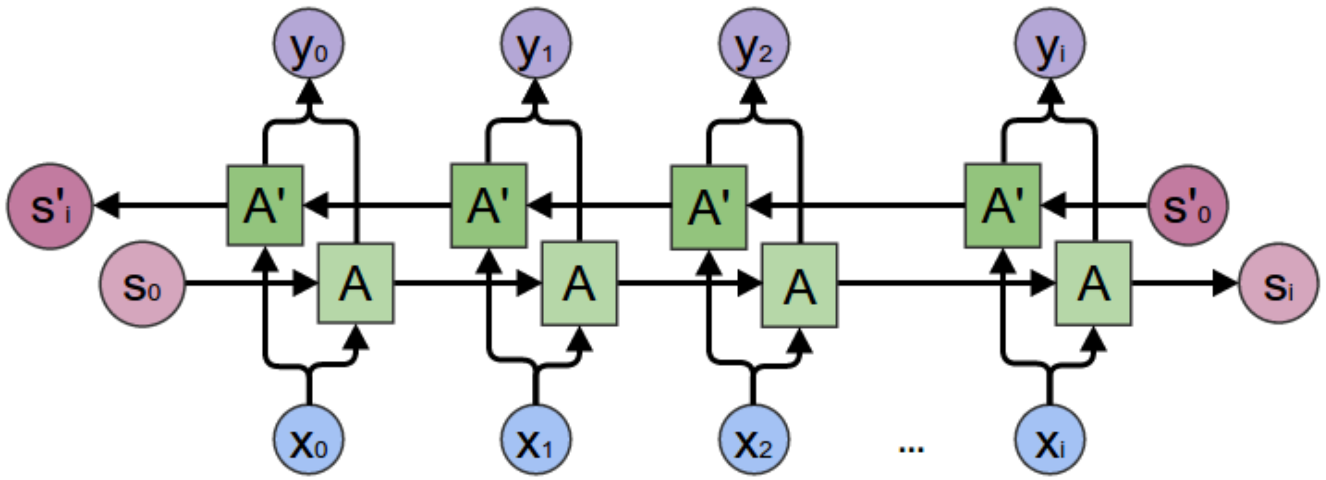
- A stateful RNN passes the last state of the previous batch to as an initial state to the next batch.
  - Otherwise the initial state is "all zeroes".
- This is useful if there is some connection between batches.
  - For example, the batches are in sequence.



Improving RNNs

Bi-directional

- We process the sequence in both directions.
- Very helpful for example in named entity recognition, in which we classify every word as a "person", "place", ...
  - He said, "Teddy Roosevelt ...
  - He said, "Teddy bears ...



Hands-on

Go to https://jupyter.lisa.surfsara.nl:8000/

Or https://dba.projects.sda.surfsara.nl/

Notebook: `05b-rnns-improved.ipynb`

14:45-15:15

Today’s program

- 14:00-14:30 Recap
- 14:30-14:45 Improving RNNs: regularization, stacking, stateful and bi-directional RNNs
- 14:45-15:15 Hands-on: Improved RNNs on temperature prediction
- 15:15-15:45 Image processing, Convolutional Neural Networks (CNN)
- 15:45-16:15 Break
- 16:15-17:00 Hands-on: CNN on fashion MNIST
- 17:00-17:30 Transfer Learning
- 17:30-18:00 Hands-on: transfer learning with pre-trained ResNet-50 layers for fashion MNIST
- 18:00-19:00 Diner
- 19:00-19:30 Unsupervised learning, the (variational) autoencoder (VAE)
- 19:30-20:15 Hands-on: auto-encoder on MNIST
- 20:15-20:45 Generative adversarial networks (GAN)
- 20:45-21:00 Summary, final questions, etc

Image data

An image is…

- A matrix of intensity values
- Range of intensities depend on *color depth* (#bits / pixel)
- Can have 1 (greyscale) or multiple (color) channels
- E.g. 8-bit greyscale, intensity 0-255

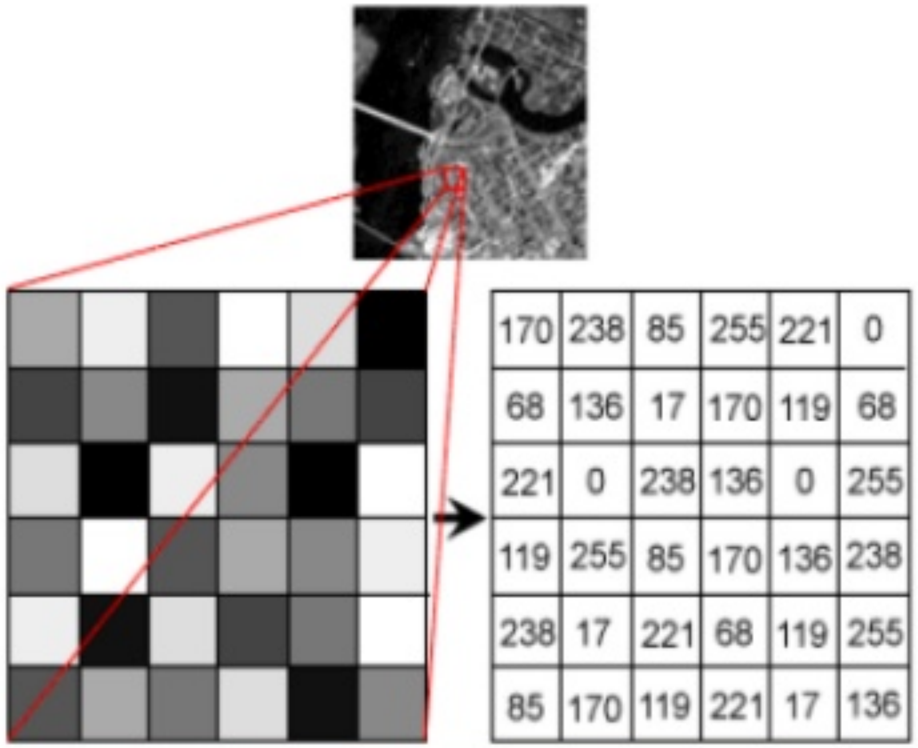


Image data

An image is…

- A matrix of intensity values
- Range of intensities depend on *color depth* (#bits / pixel)
- Can have 1 (greyscale) or multiple (color) channels
- E.g. 8-bit RGB image, intensity 0-255

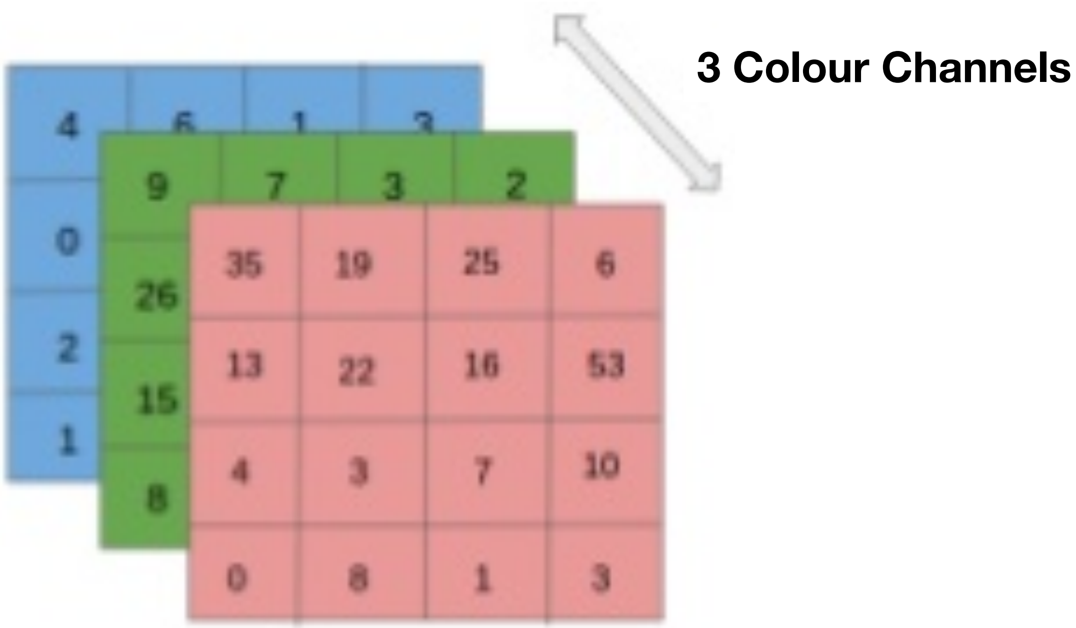


Image data

Analyzing image data with a dense network

Remember notebook 02b, exc 4

```
model <- keras_model_sequential() %>%
   layer_flatten(input_shape = c(28, 28)) %>%
   layer_dense(units = 4, activation = "relu") %>%
   layer_dense(units = 1, activation = "sigmoid")
```

5	2	6	8	2	0	1	2
4	3	4	5	1	9	6	3
3	9	2	4	7	7	6	9
1	3	4	6	8	2	2	1
8	4	6	2	3	1	8	8
5	8	9	0	1	0	2	3
9	2	6	6	3	6	2	1
9	8	8	2	6	3	4	5

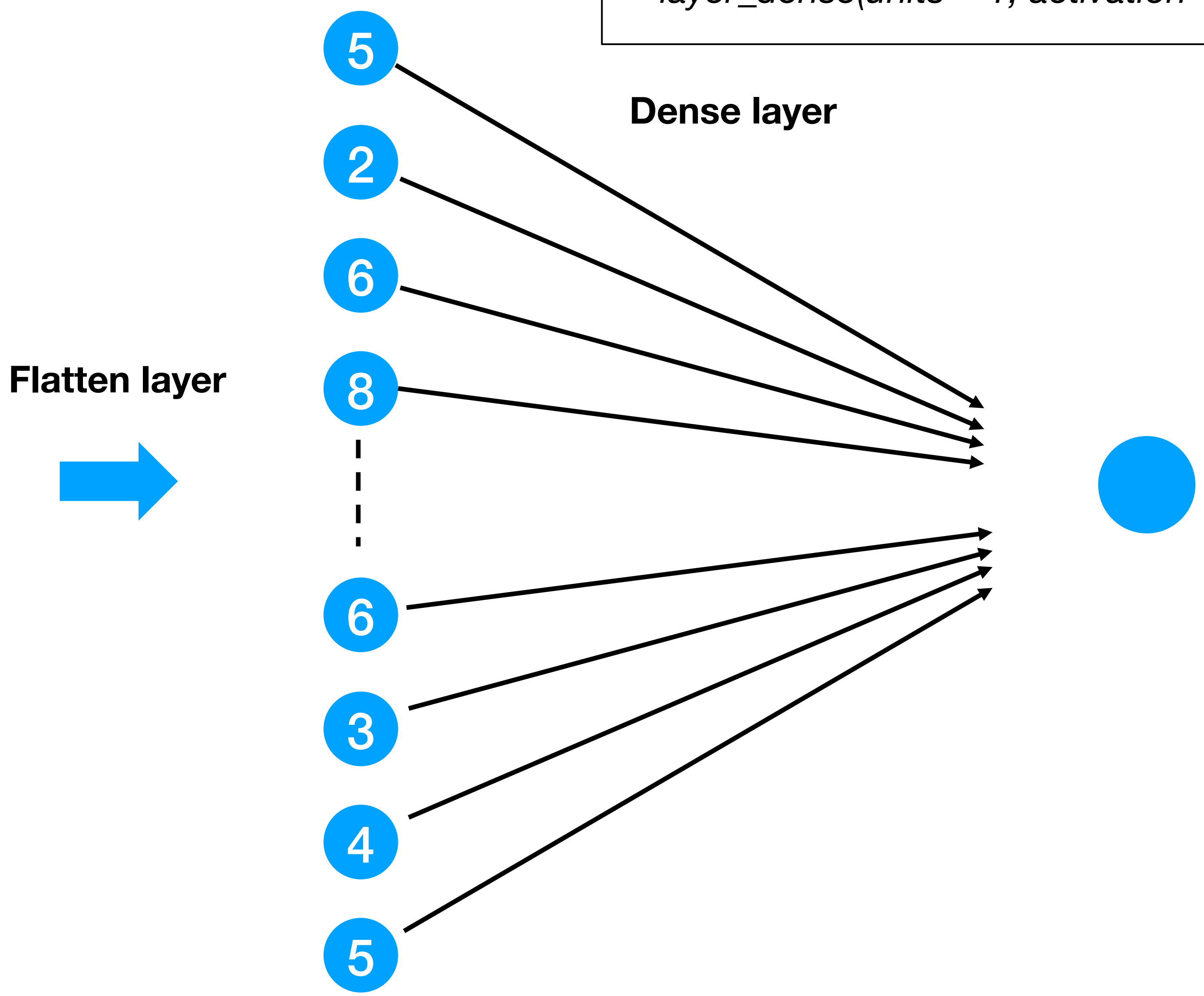
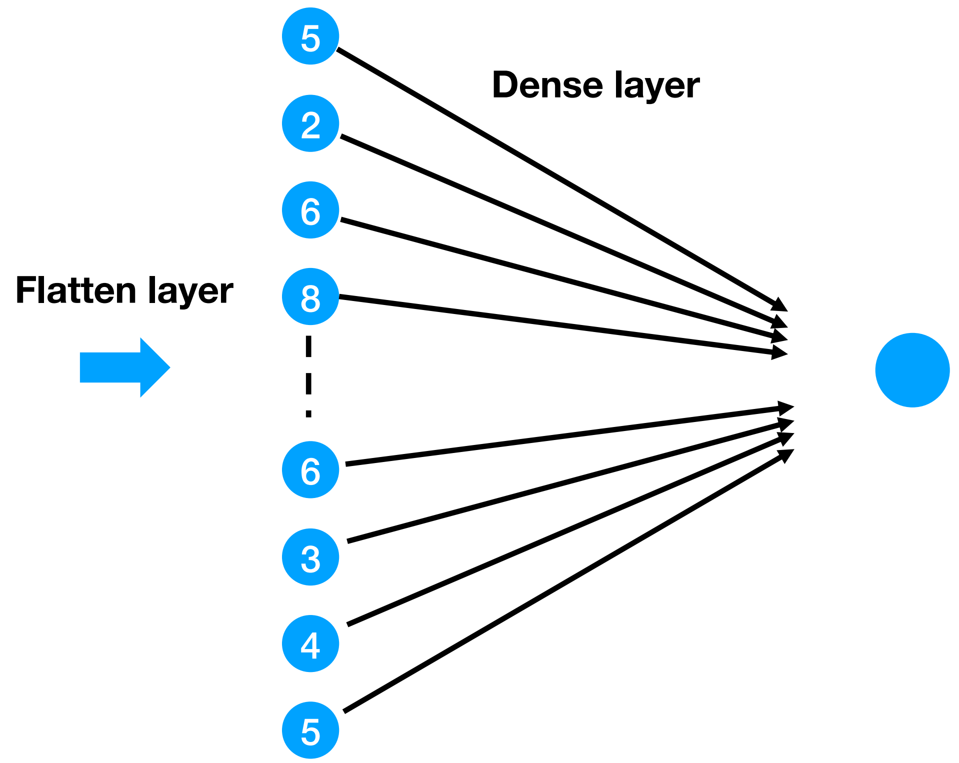


Image data

The problem in analyzing images with dense networks:

- Our first step is to <u>forget</u> pixel locations!
- But pixel locations are important to detect lines, curves, etc!
- We need an architecture that preservers this spatial relationship…

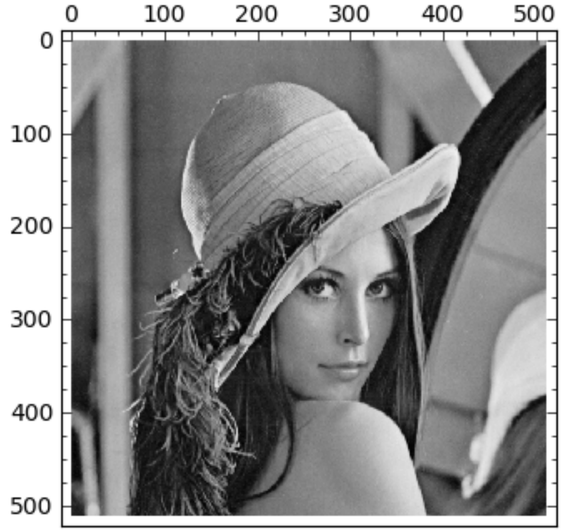
5	2	6	8	2	0	1	2
4	3	4	5	1	9	6	3
3	9	2	4	7	7	6	9
1	3	4	6	8	2	2	1
8	4	6	2	3	1	8	8
5	8	9	0	1	0	2	3
9	2	6	6	3	6	2	1
9	8	8	2	6	3	4	5

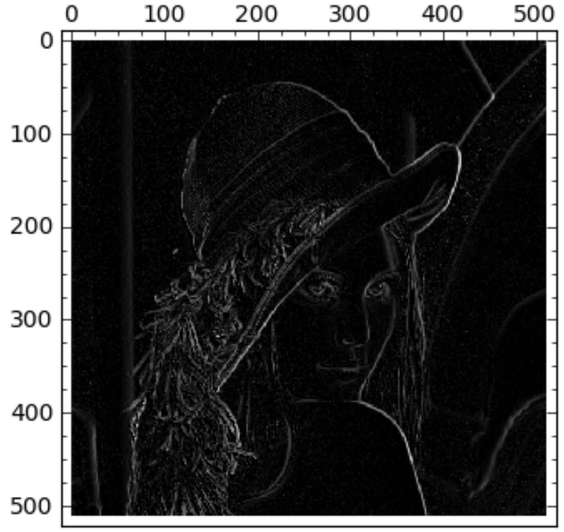


Filtering

- In digital image processing, **filters** are used to achieve certain effects, things like blurring, sharpening, edge detection, etc

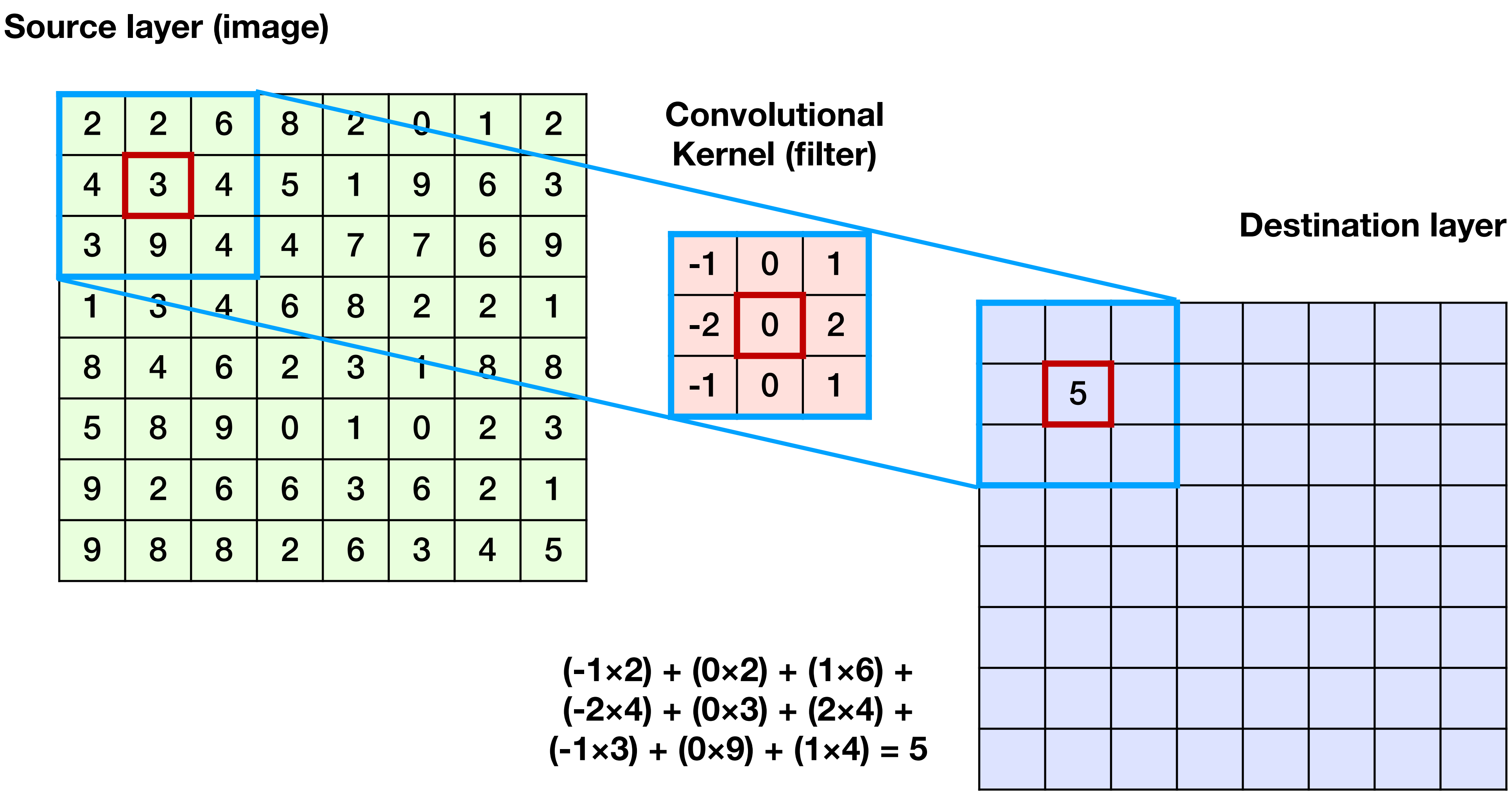
- Filters apply **convolutional** operations on an image





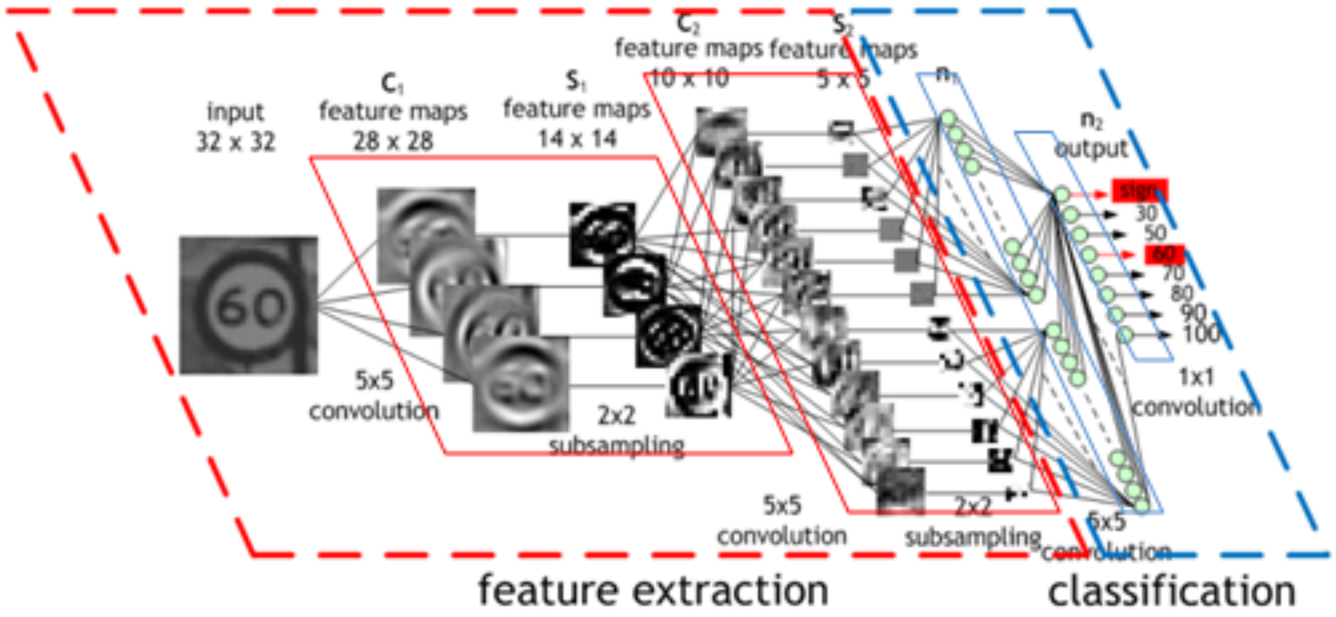
Convolutions

- Convolutional filters process parts of the image and return a high signal when they detect the filter pattern.



Convolutions

- We then applying a convolution upon a convolution.
- Allows us to create filters from many other filters.
- Then we can detect complex patterns over a larger area.

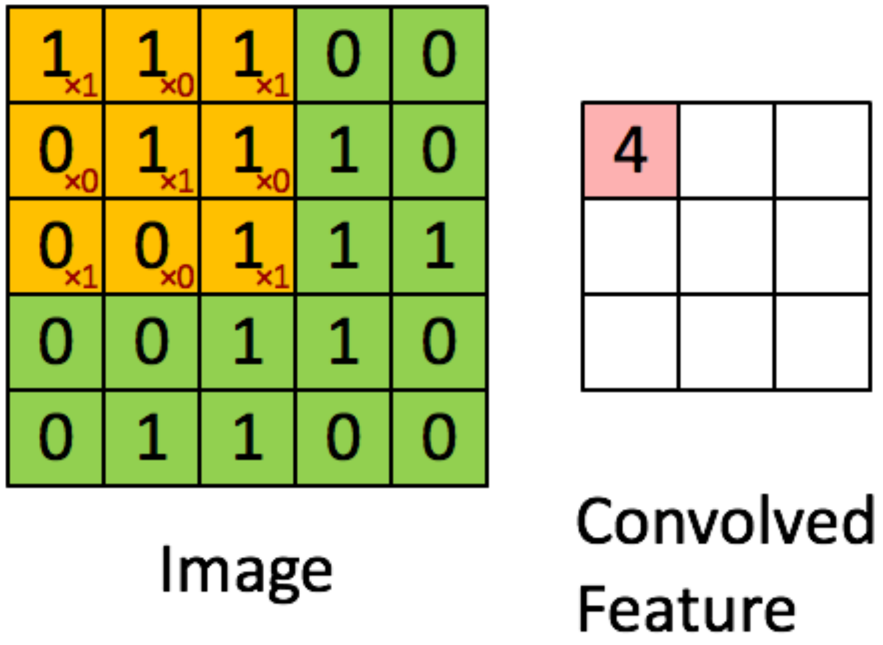


Source: https://developer.nvidia.com/discover/convolutional-neural-network

Convolutions

Technicalities

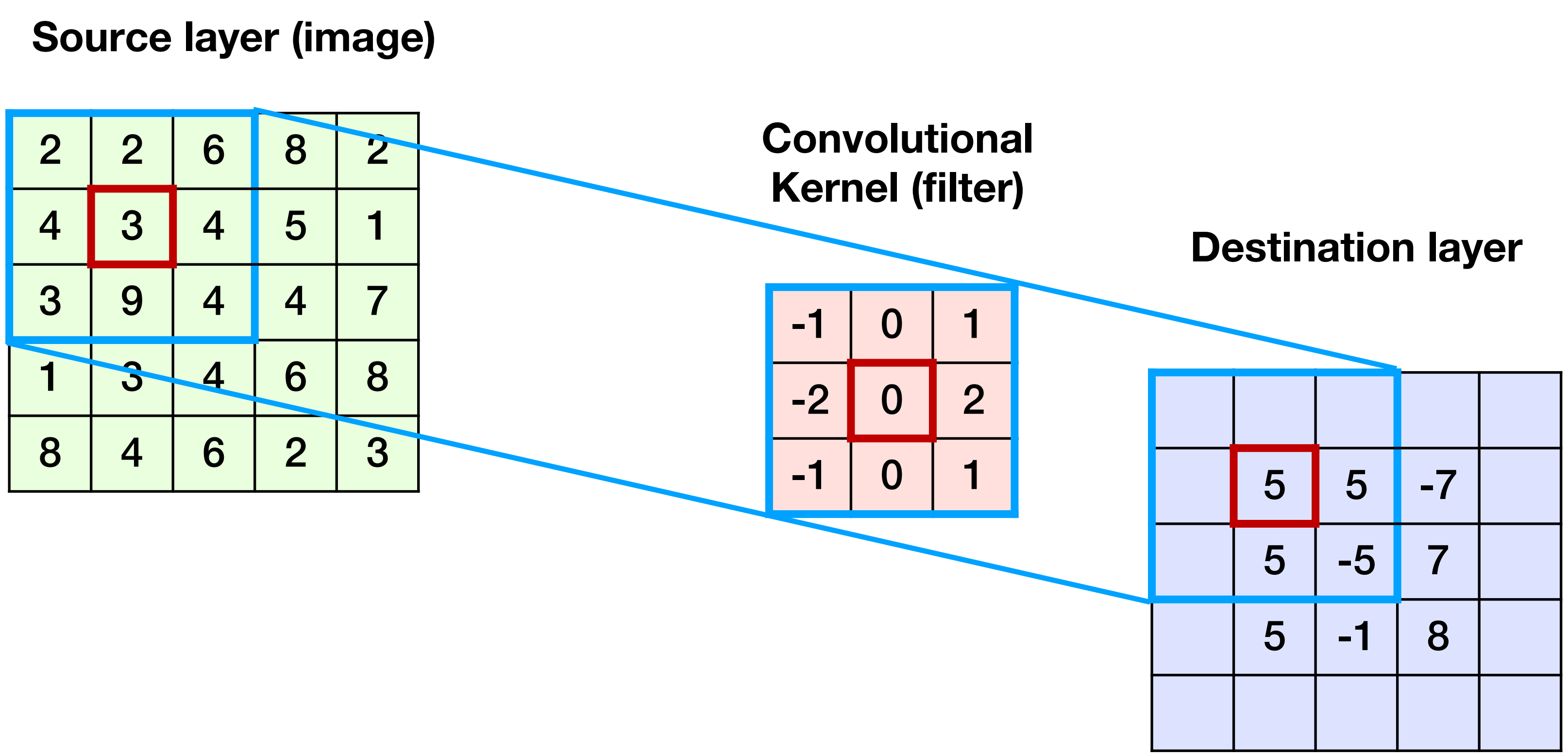
- The kernel is shifted over the image, and computes output for each position



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

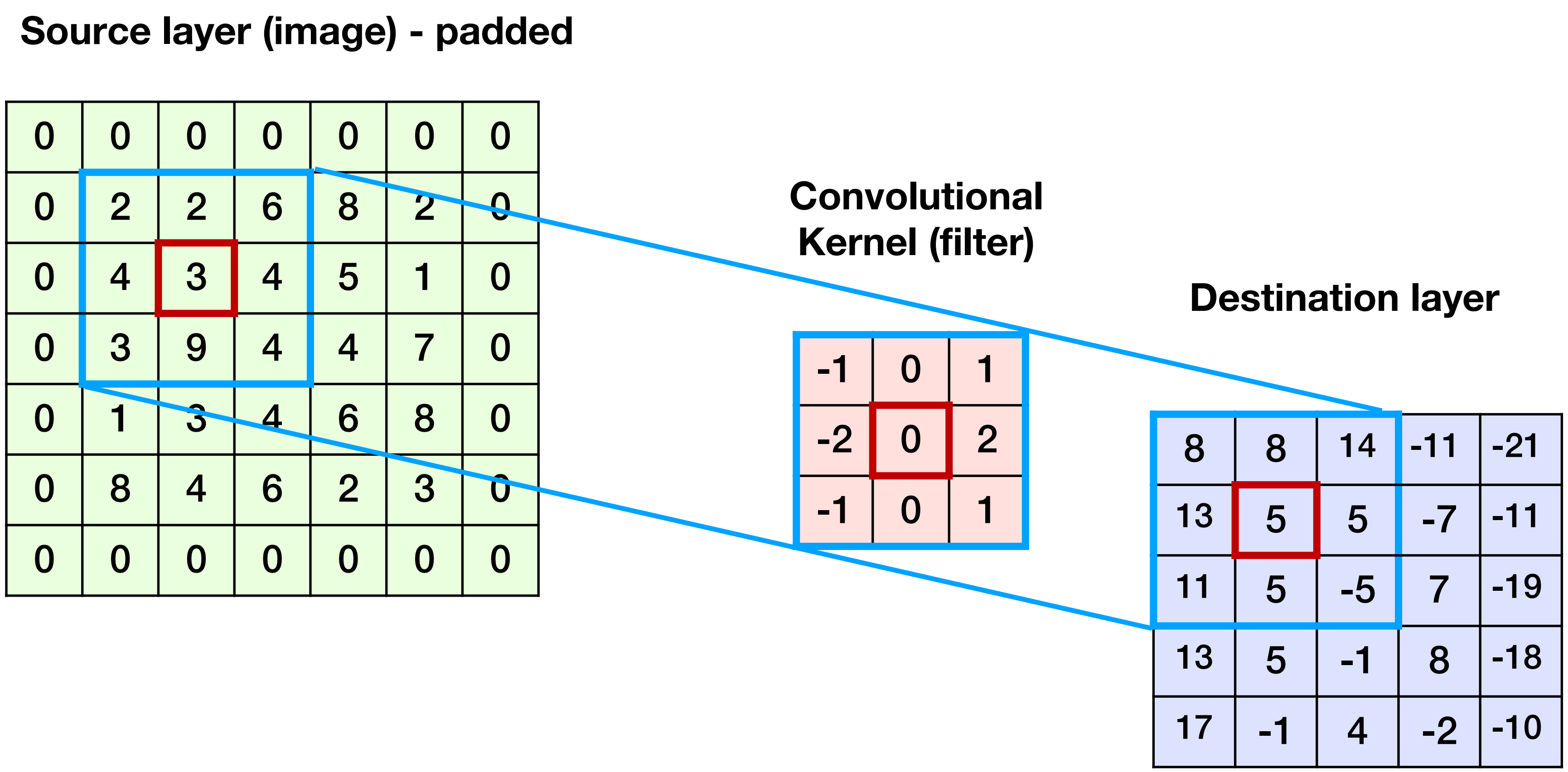
Convolutions

- Output has smaller dimension than input (why?)
- $\dim(output) = \dim(source) - (\dim(kernel) - 1)$
- I.e. a $3{\times}3$ kernel reduces dimension by 2 along each axis, $5{\times}5$ would reduce by 4, etc



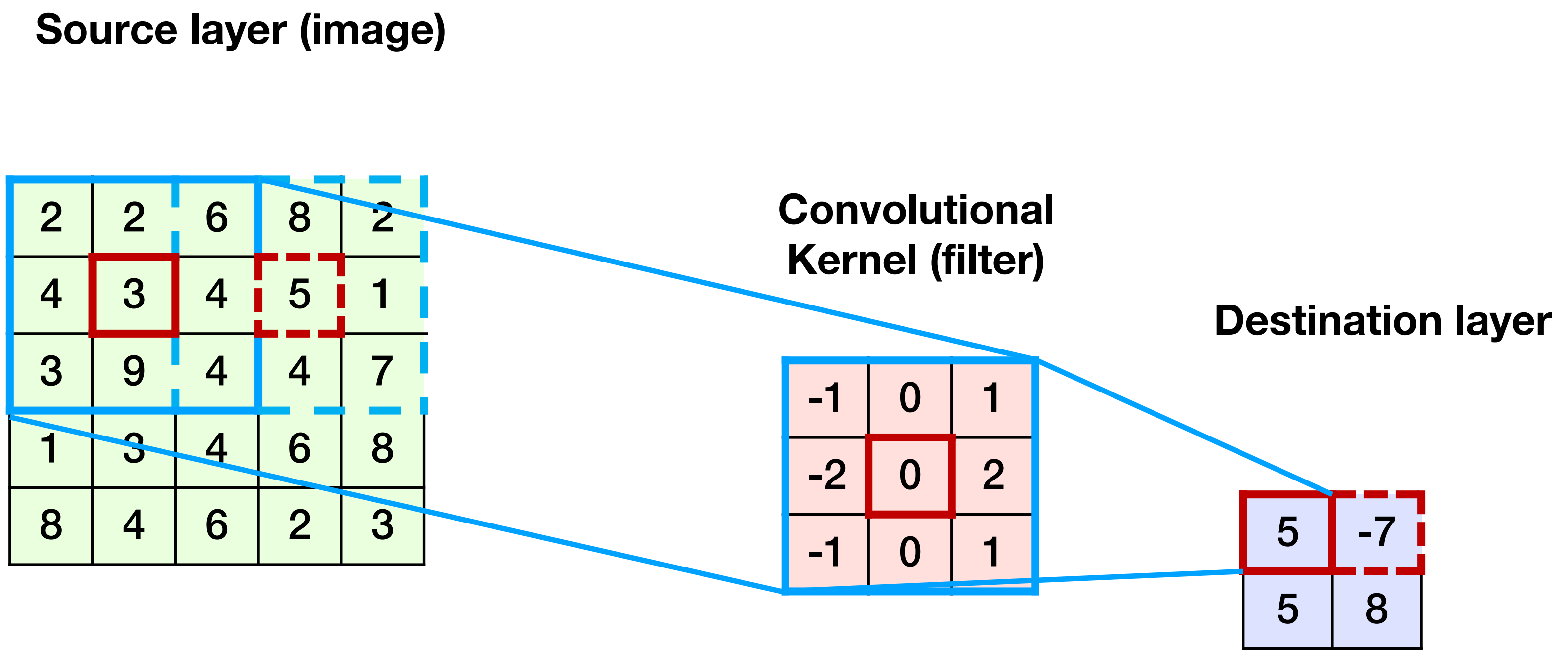
Convolutions

- Output has smaller dimension than input
- **Padding** can be used, this makes the input image 'bigger' by using synthetic data (typically 0-padding)
- Padding 'same' typically preserves the original input dimension



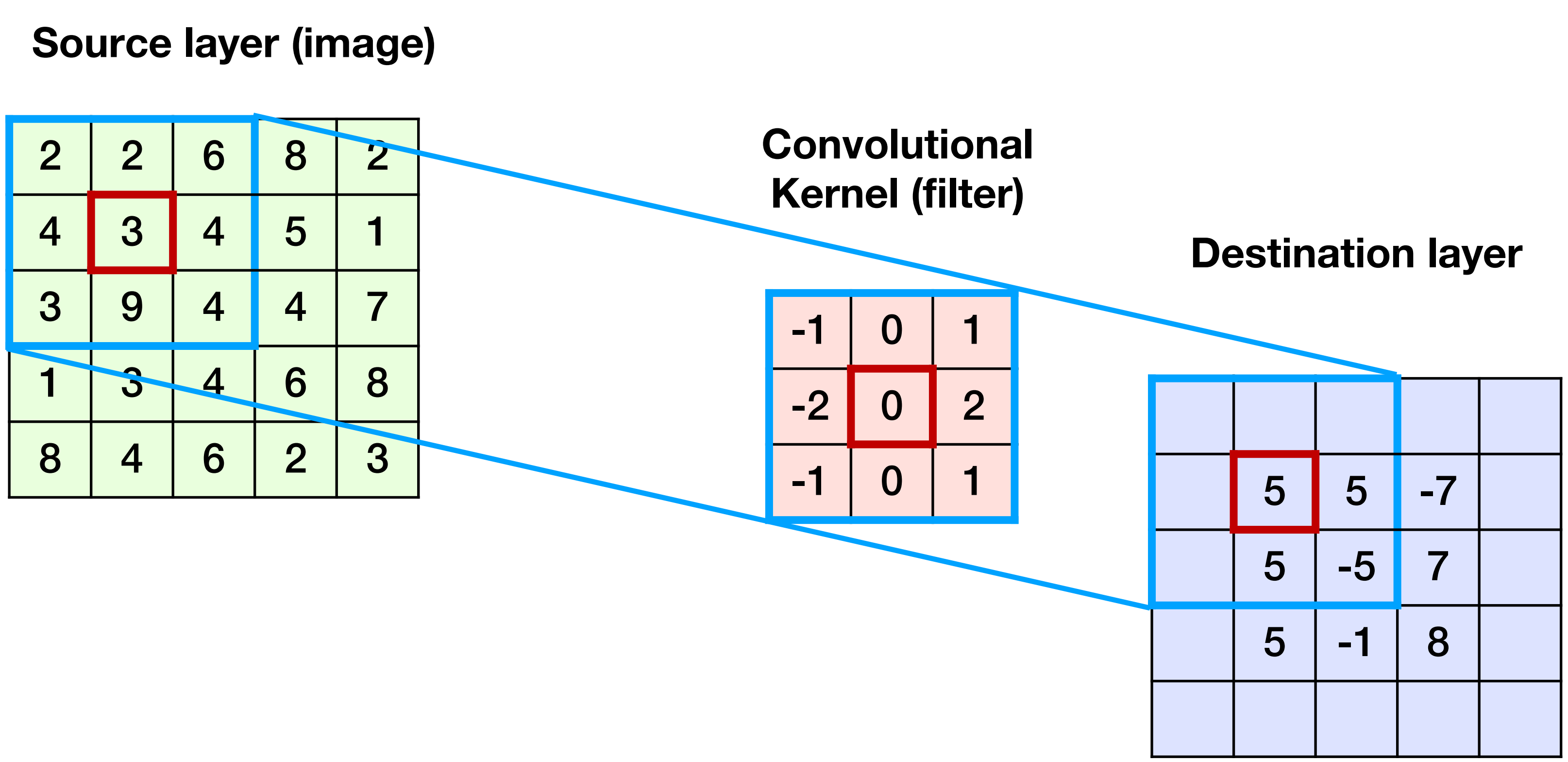
Convolutions

- Filters can also take bigger ‘steps’. This step size is known as the **stride**.
- Stride *also* affects the output dimension
- E.g. stride of 2, without padding, on 5×5 input results in 2×2 output



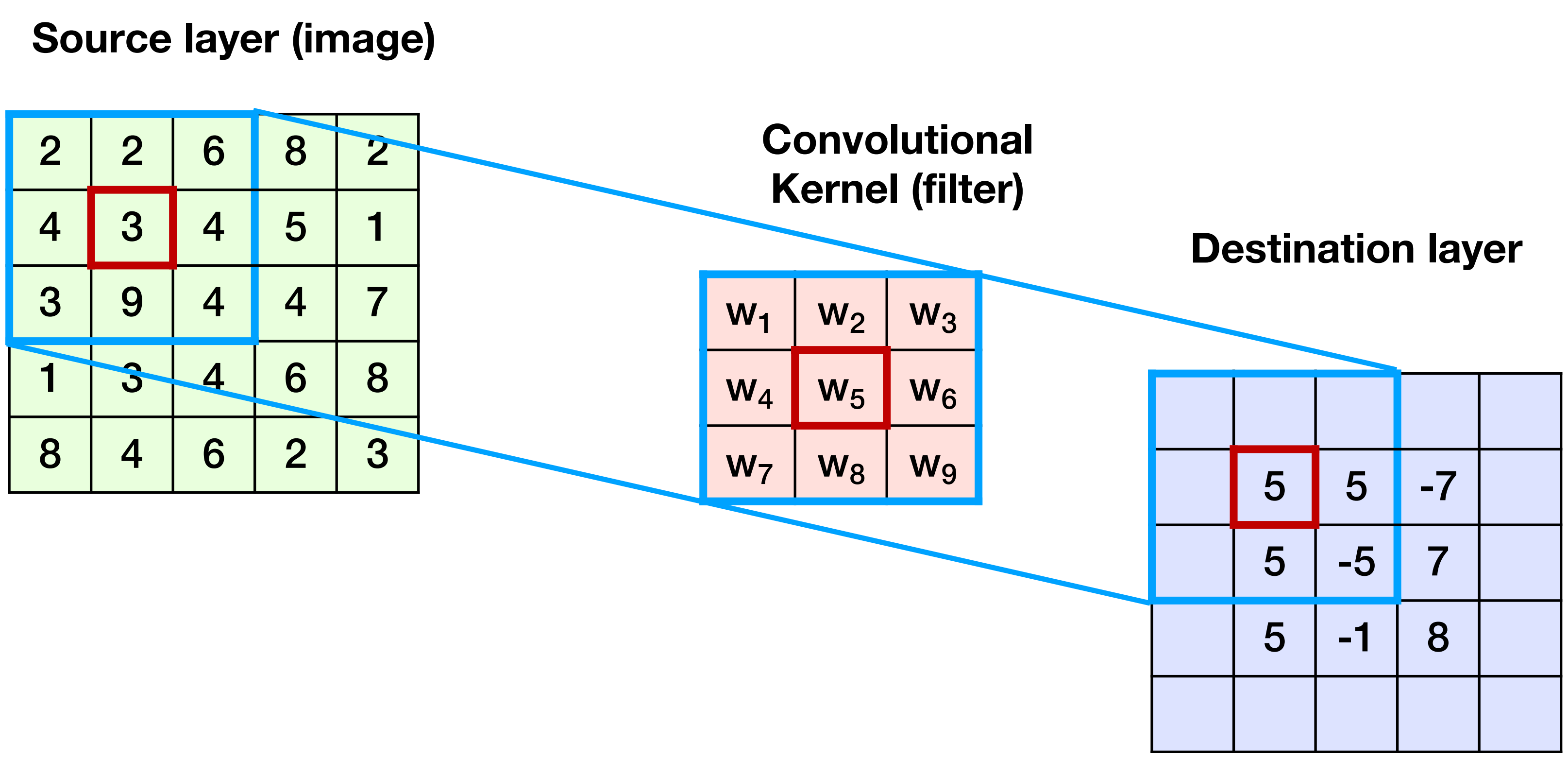
Convolutions

- A traditional filter uses a kernel with fixed values, e.g. the vertical edge detection filter shown below



Convolutions

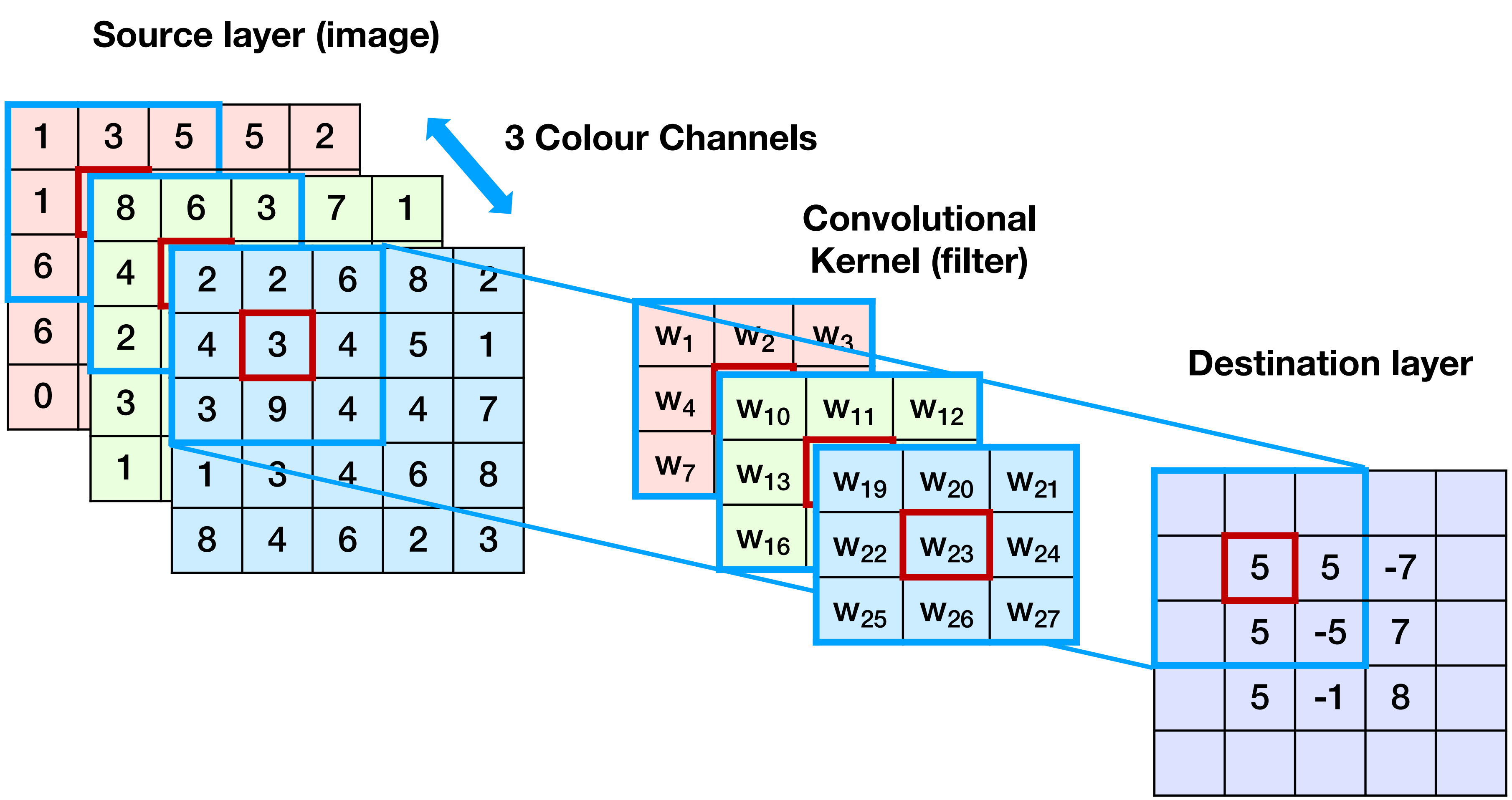
- A traditional filter uses a kernel with fixed values, e.g. the vertical edge detection filter shown below
- In deep learning, the kernel consists of **trainable weights**
- Thus, instead of making an expert decision, the network **learns** which visual features (such as vertical edges) it should detect



Convolutions

For color images, we have multiple input channels

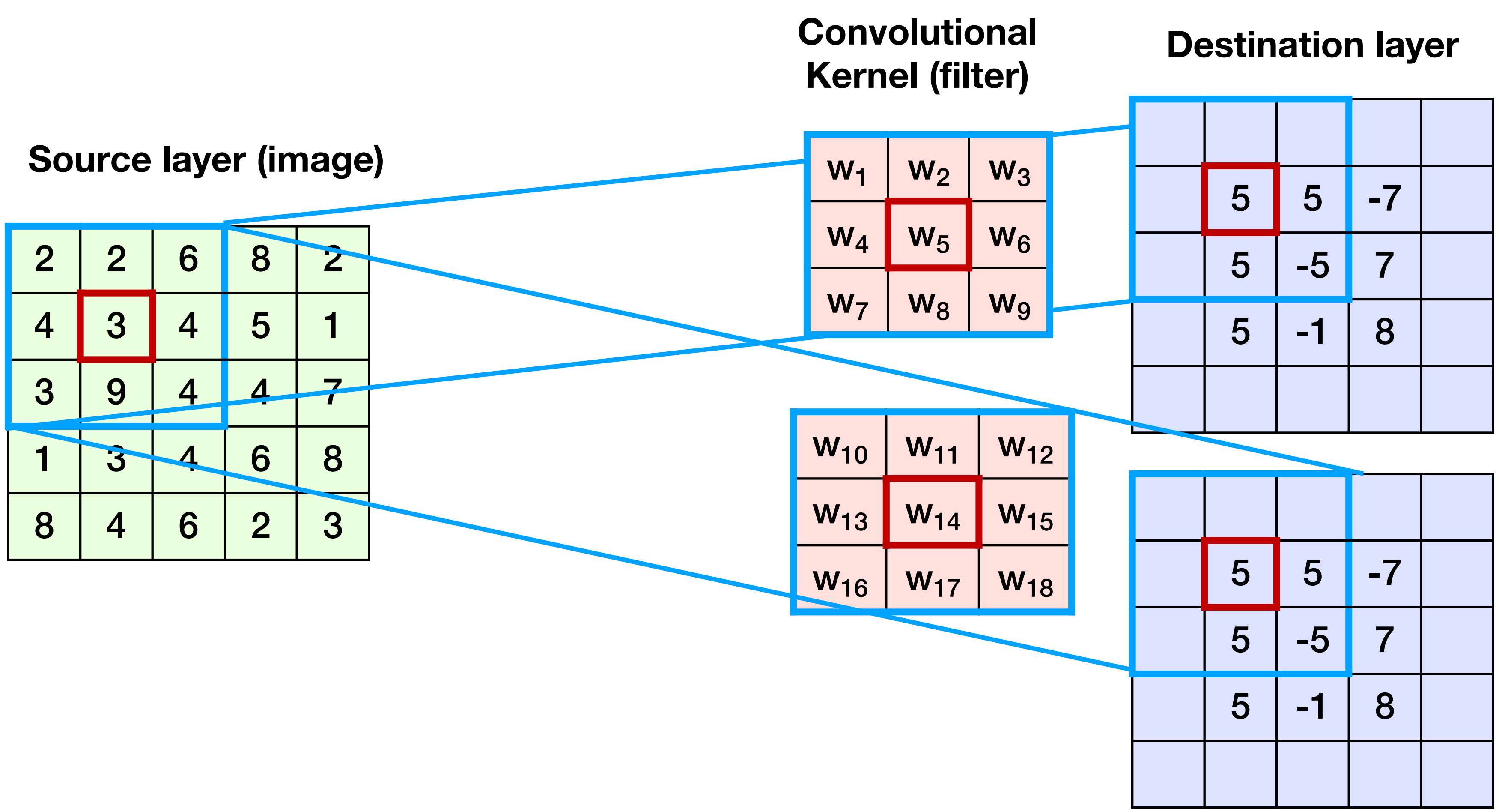
- Kernel is now $3 \times 3 \times 3$, i.e. 27 weights
- Scans over all channels simultaneously



Convolutions

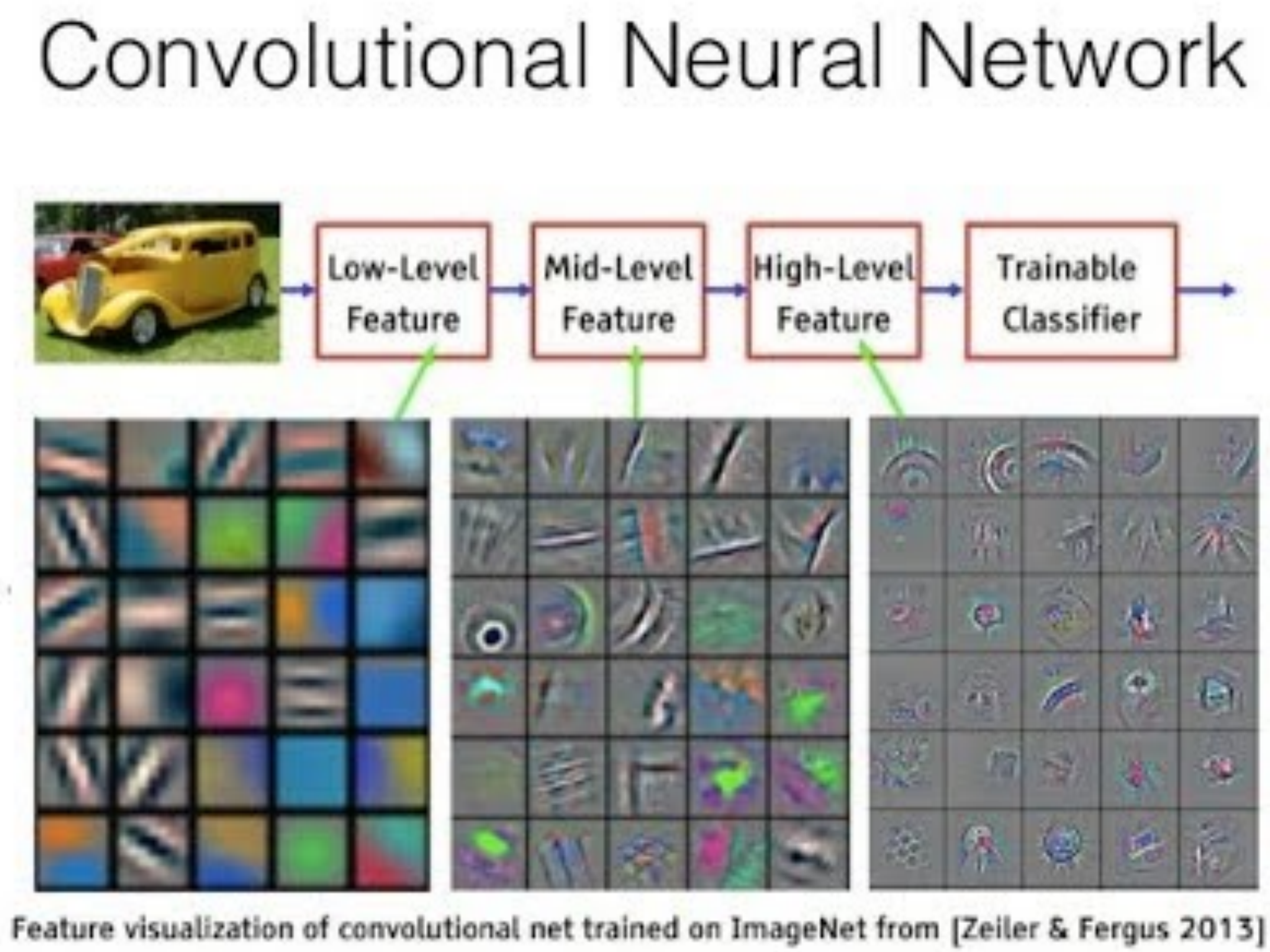
Typically, we want our network to be able to abstract multiple features in each layer (e.g. learn to detect horizontal lines, vertical lines, diagonal lines, etc)

- Each filter produces its own destination layer, known as a **feature map** or **activation map**



Convolutions

- Example showing three convolutional layers, each with 30 filters:



Convolutions

Thus, for a convolutional layer, we typically specify

- Size of the kernel, e.g. 3×3
- Padding, e.g. ‘same’
- Stride, e.g. 1
- Number of **filters** (or: **output channels**)
- For some frameworks: number of **input channels** (though Keras infers this from the previous layer)

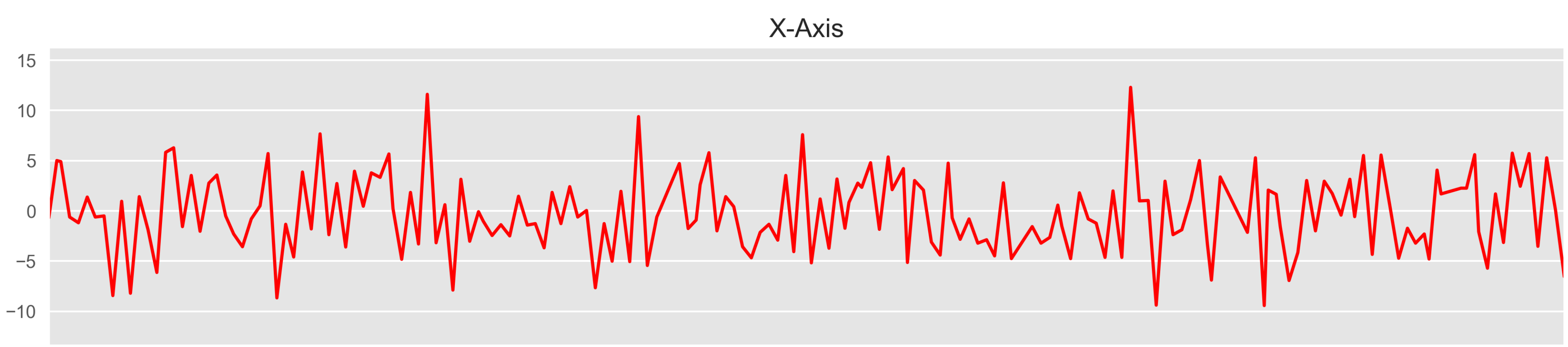
CNNs: not only images

CNNs are most commonly applied on images, but can also work well on other data in which …

- *closeness* / *spatial location* of data points is meaningful
- simple, low level patterns (in *nearby* data points) combine into more complex, high level patterns (across *further* data points)

E.g. Timeseries, sound data (1D convolution)

1D convolution example: activity classification based on smartphone accelerometer



Source: https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf

SURF

Hands-on

Go to https://jupyter.lisa.surfsara.nl:8000

or https://dba.projects.sda.surfsara.nl/

Notebook: `06a-cnns.ipynb`

`16:15-17:00`

Today's program

- 14:00-14:30 Recap
- 14:30-14:45 Improving RNNs: regularization, stacking, stateful and bi-directional RNNs
- 14:45-15:15 Hands-on: Improved RNNs on temperature prediction
- 15:15-15:45 Image processing, Convolutional Neural Networks (CNN)
- 15:45-16:15 Break
- 16:15-17:00 Hands-on: CNN on fashion MNIST
- 17:00-17:30 Transfer Learning
- 17:30-18:00 Hands-on: transfer learning with pre-trained ResNet-50 layers for fashion MNIST
- 18:00-19:00 Diner
- 19:00-19:30 Unsupervised learning, the (variational) autoencoder (VAE)
- 19:30-20:15 Hands-on: auto-encoder on MNIST
- 20:15-20:45 Generative adversarial networks (GAN)
- 20:45-21:00 Summary, final questions, etc

CNNs

Reducing dimensions of activation maps

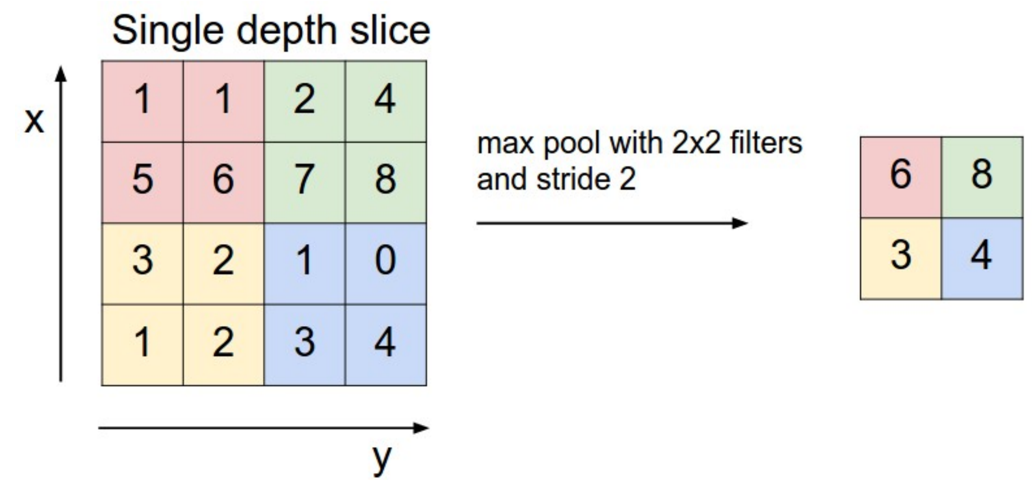
Filter maps are sometimes very large. We typically try to reduce dimensions towards deeper layers. Reducing dimensionality helps

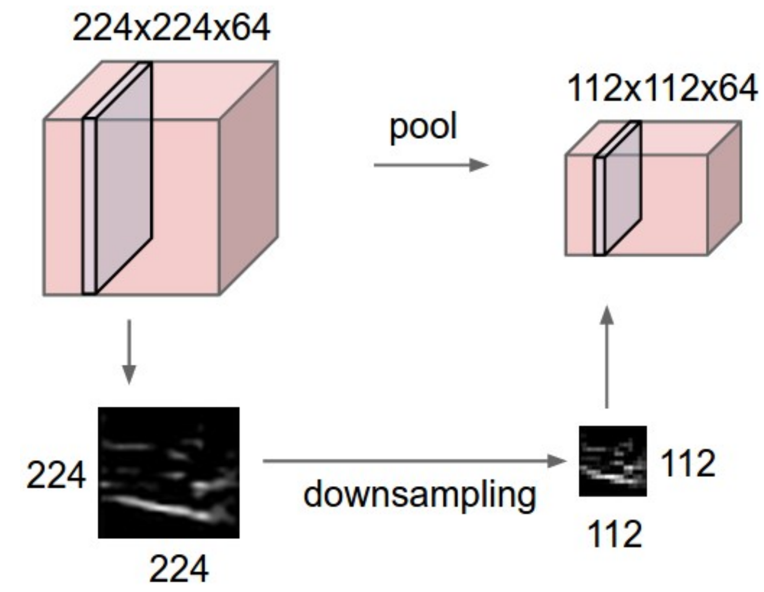
- … limit the amount of computation
- … increase performance of the network by picking only

CNNs

Reducing dimensions of activation maps

- We've seen: **stride** > 1 reduces dimensions
    - E.g. stride = 2 approximately halves the size)
- Another way is to use **pooling.**
    - E.g. **Max pooling**, take the max.
    - E.g. **Average pooling**, take the average.
    - Pooling is applied after convolutions





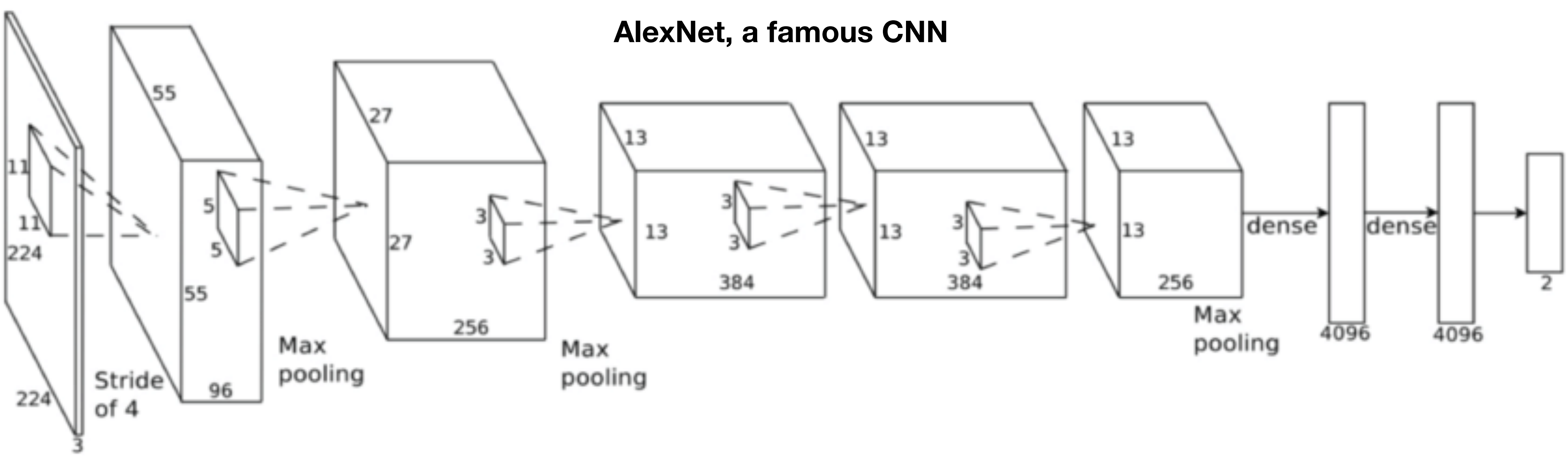
Source: http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html

CNNs

CNN architecture

- We create a CNN by chaining together convolutional layers
- Typically, the dimensionality reduces the deeper we go
- Typically, the number of filters increases, the deeper we go
  - Limited number of low-level features, e.g. horizontal/vertical/diagonal lines (detected in first layers)
  - Many possible high-level features (all kinds of circular / rectangular shapes, pyramids, cones)
- Typically end with one or more dense layers that ‘take the decision’ (e.g. ‘I detect 2 wheels and a person: classify this as a bike’).

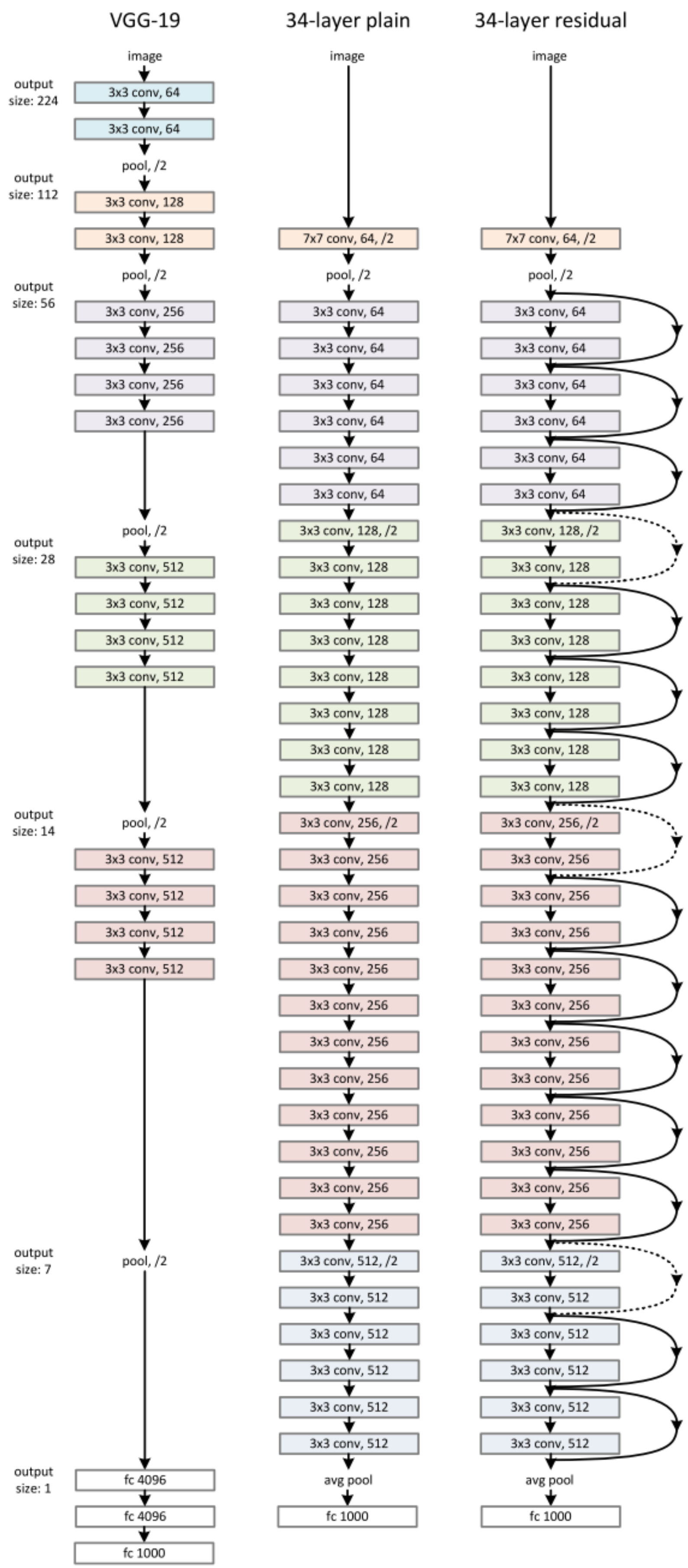
AlexNet, a famous CNN



Source: https://www.jeremyjordan.me/convnet-architectures/

CNNs

Architectures - ResNet (2015)



Source: https://arxiv.org/abs/1512.03385

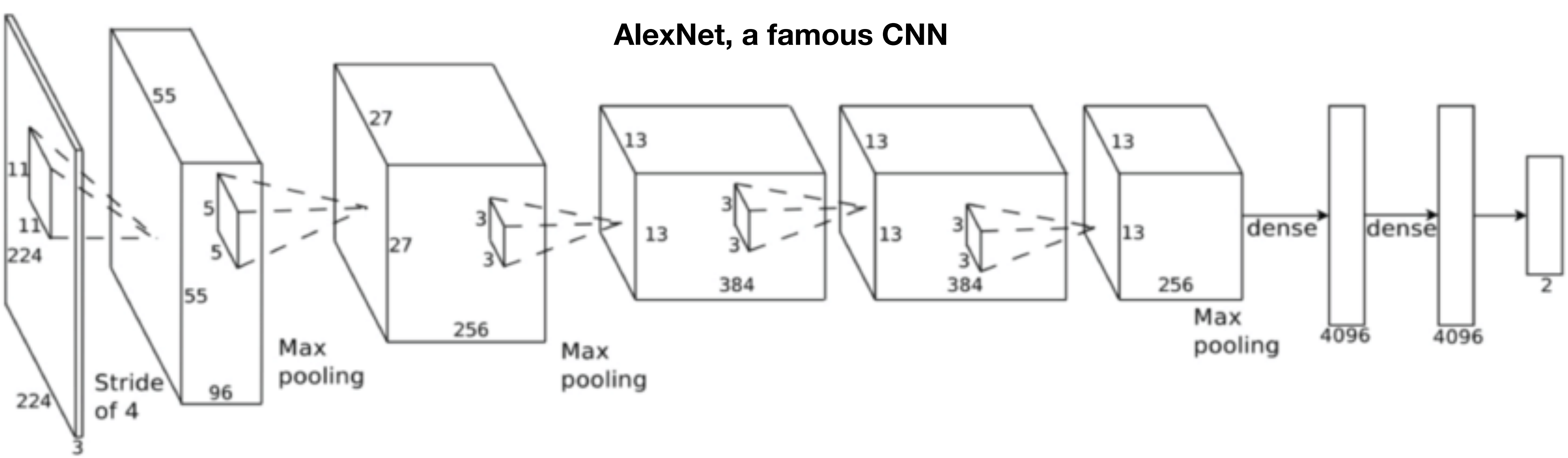
CNNs

Designing architectures

Should we develop these architectures ourselves?

- Typically, we use well-known architectures that we know work well
- We might adapt those architectures to tune to our problem

AlexNet, a famous CNN



Source: https://www.jeremyjordan.me/convnet-architectures/

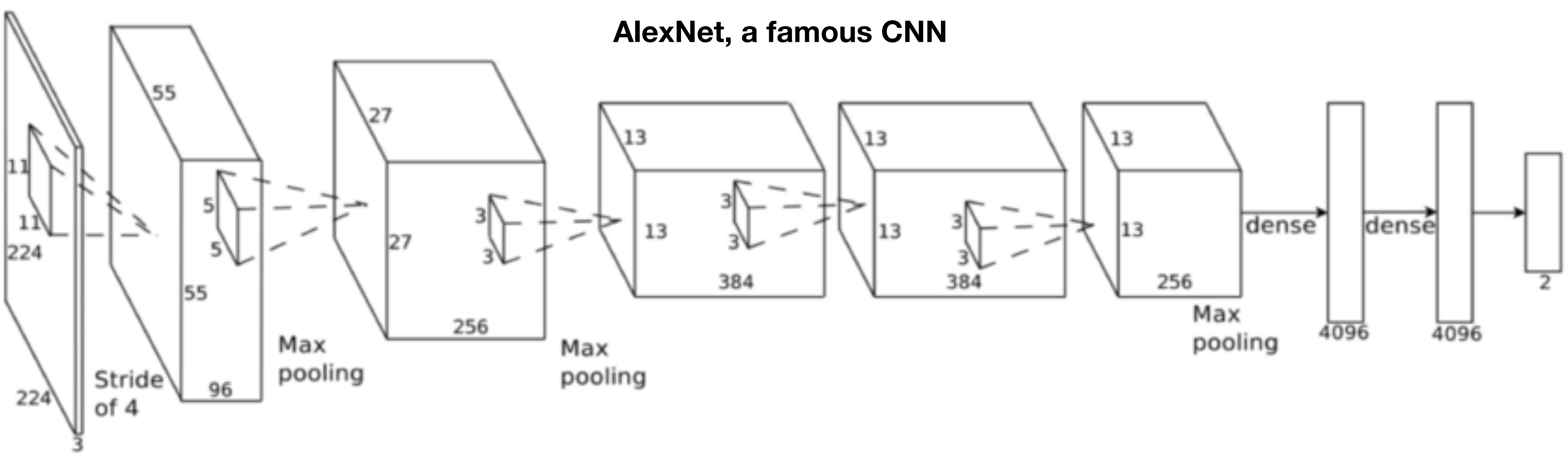
CNNs

Pretrained

Training times can be long, especially for complex architectures

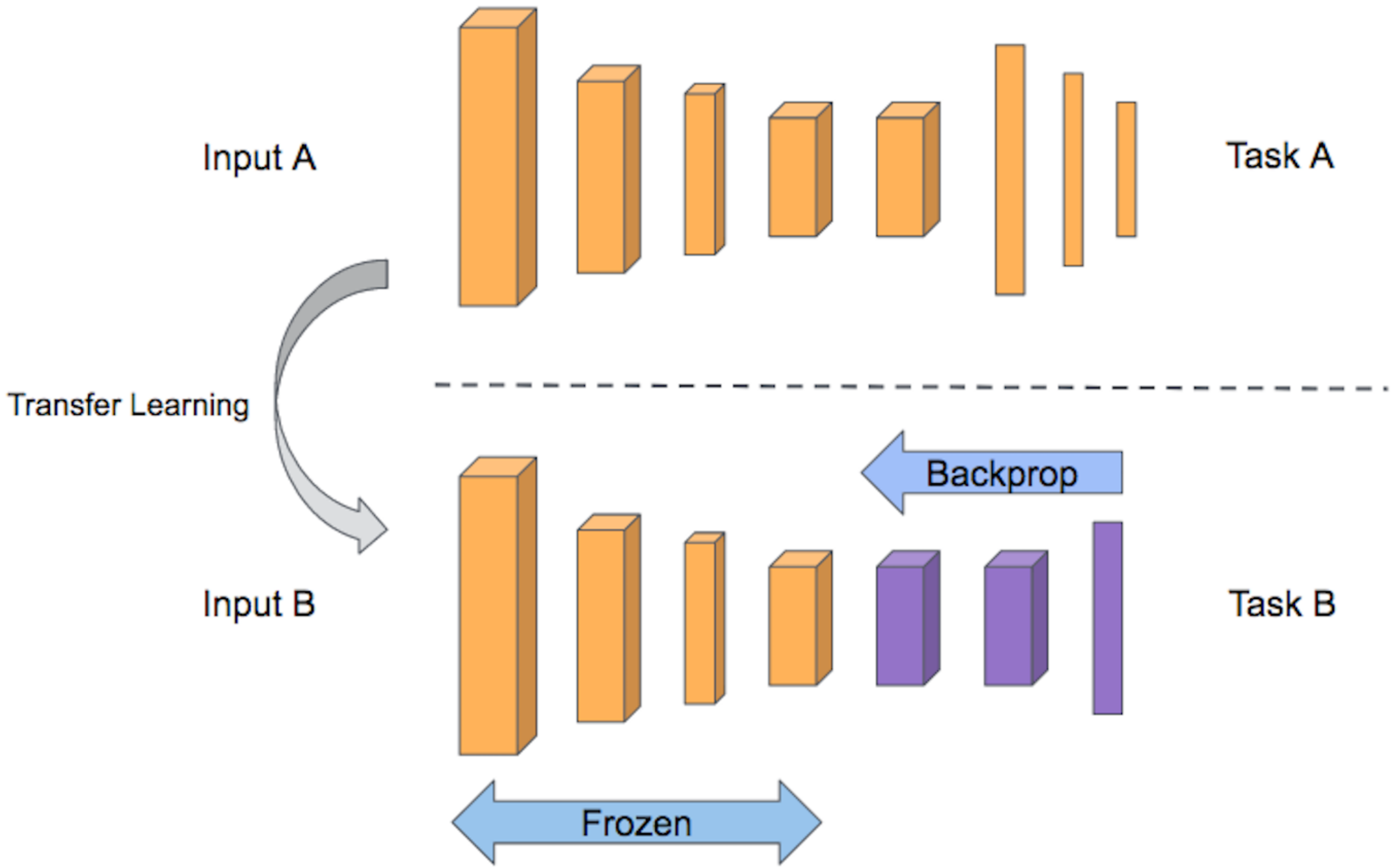
- First layers encode low-level features (e.g. lines)
- Images of plants probably have similar low level features as images of animals, but different high level features.
- We can reuse the first layers of a CNN that was already trained on another dataset
- This is called **transfer learning**.

AlexNet, a famous CNN



Source: https://www.jeremyjordan.me/convnet-architectures/

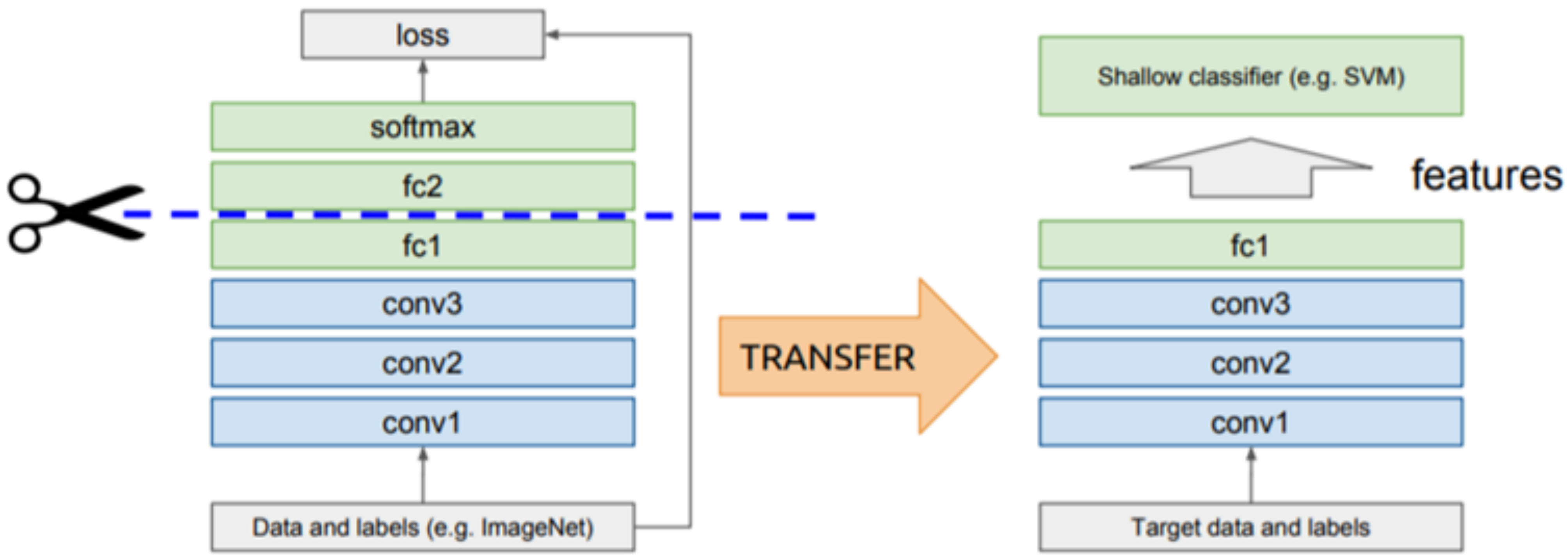
Transfer learning



Transfer learning

Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.

Assumes that $D_S = D_T$



Transfer learning

Caveats

- We need to **download** the network and weights, which can be large (still 1000 times faster than training ourselves).
- We use a **smaller learning rate**.
- To increase speed, many people preprocess the input through the static network and **save the representations to disk** and then train a new network separately.

Hands-on

Go to https://jupyter.lisa.surfsara.nl:8000

or https://dba.projects.sda.surfsara.nl/

Notebook: `06b-cnns-transfer.ipynb`

17:30-18:00