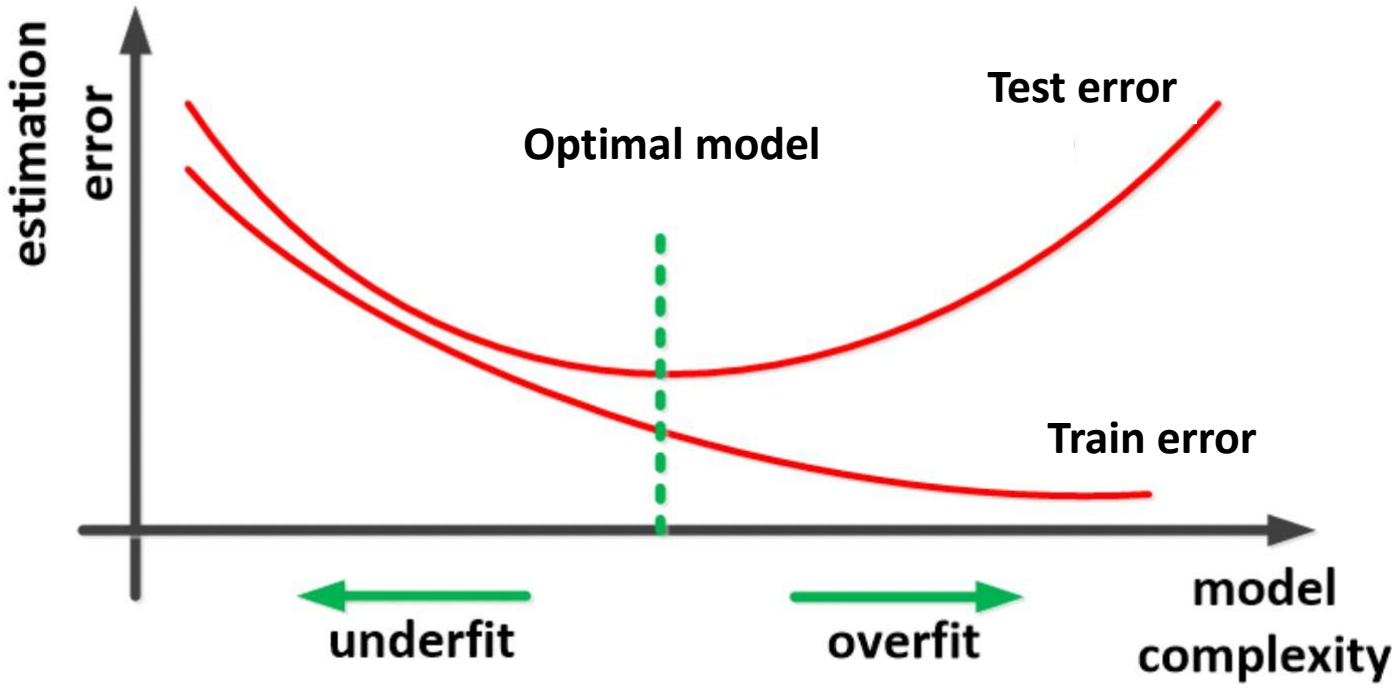
Neural Networks 3

Anders Krogh
Center for Health Data Science
and Department of Computer Science
University of Copenhagen

Performance evaluation

- Neural networks should be evaluated like other machine learning methods on independent test data
- Additional danger with neural networks: Selection of network type/architecture using test set leads to biassed test results

- Use **validation set** for model selection and "early stopping"
- Use **test set once** in the very end of the project

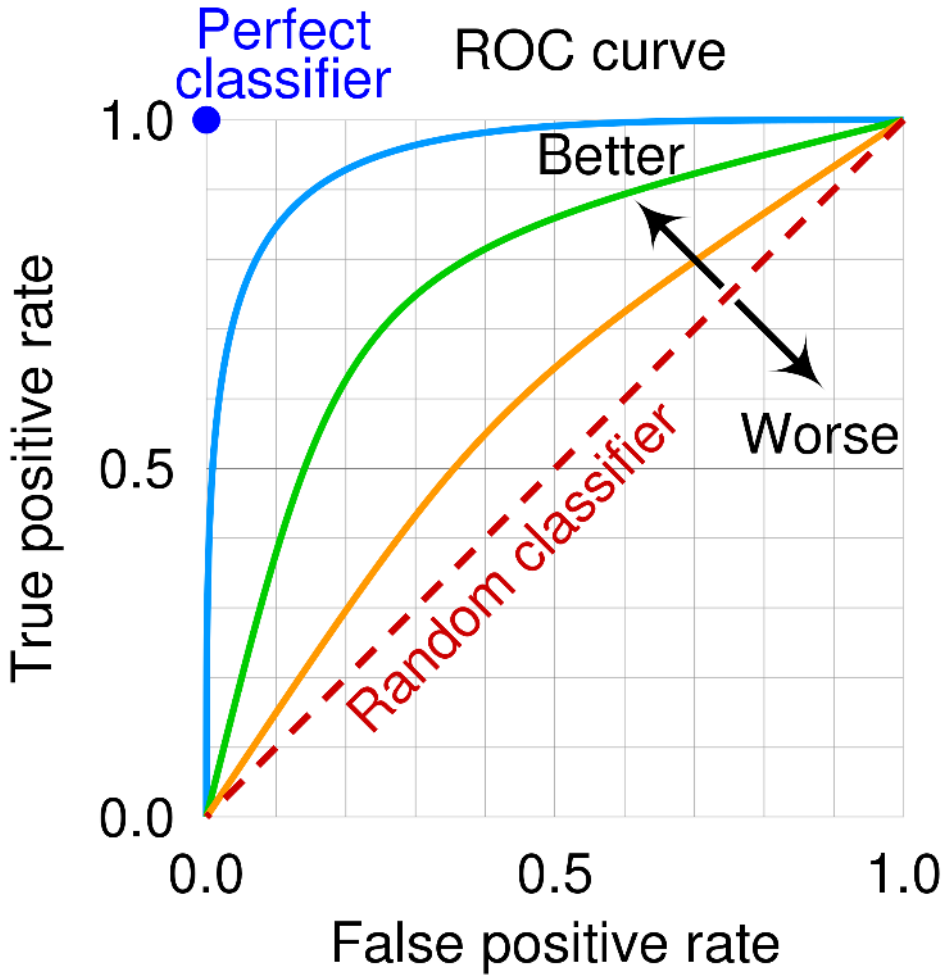


ROC curves for evaluation of classification

- To classify you select a **cut-off** on the output (e.g. 0.5)
- You can then calculate accuracy

- ROC curves give a performance overview **independent of cut-off**

- **AUC**: area under the curve is an overall measure of performance that should be >0.5 and <1.



TP = correctly predicted positives (cats)
TP rate = TP/real_positives (fraction correctly predicted cats)
FP = incorrectly predicted as positives
FP rate = FP/real_negatives

Figure from Wikipedia that has a detailed discussion of ROC curves: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Letters/sequences as input

What if the input DNA or protein sequences – how are they coded?

DNA sequence

AGTCACCACCGCTGGCAAGCCACATGTA

Just number letters

BAD IDEA!

13421221223243321322121434 1

Use **one-hot** encoding

AGTCACCACCGCTGGCAAGCCACATGTA

1000**1**00**1**0000000**11**000**1**0**1**0001
000**1**0**11**0**11**0**1**000**1**000**11**0**1**00000
0**1**0000000**1**00**11**000**1**000000**1**00
00**1**0000000000**1**0000000000**1**0**1**0

2D tensor

Make 1D tensor (flatten)

1000001**0**10000**1**00**1**0000**1**000**1**00**1**0000**1**000**1**00000**1**0010000**1**00**1**0…

A G T C A C C A C C G C T G …

One-hot tensor magic

```
seq='TGTAGCTCTCAG'
onehot2d = one_hot_dna(seq)
print(onehot2d)
print(onehot2d.shape)
```

```
tensor([[0, 0, 0, 1],
        [0, 0, 1, 0],
        [0, 0, 0, 1],
        [1, 0, 0, 0],
        [0, 0, 1, 0],
        [0, 1, 0, 0],
        [0, 0, 0, 1],
        [0, 1, 0, 0],
        [0, 0, 0, 1],
        [0, 1, 0, 0],
        [1, 0, 0, 0],
        [0, 0, 1, 0]])
torch.Size([12, 4])
```

```
# Turn it into a 1D tensor with flatten
onehot1d=onehot2d.flatten()
print(onehot1d)
print(onehot1d.shape)
# Wiew can do the same with view (4*12=48)
print(onehot2d.view(48))
```

```
tensor([0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
        0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0])
torch.Size([48])
tensor([0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
        0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0])
```

The method view() can change the shape of a tensor.

A dimension with -1 will be filled to fit the remaining dimensions

```
# Turn a 1D tensor into 2D with view
# You can use -1 for one dimension
new2d = onehot1d.view(-1,4)
print(new2d)
```

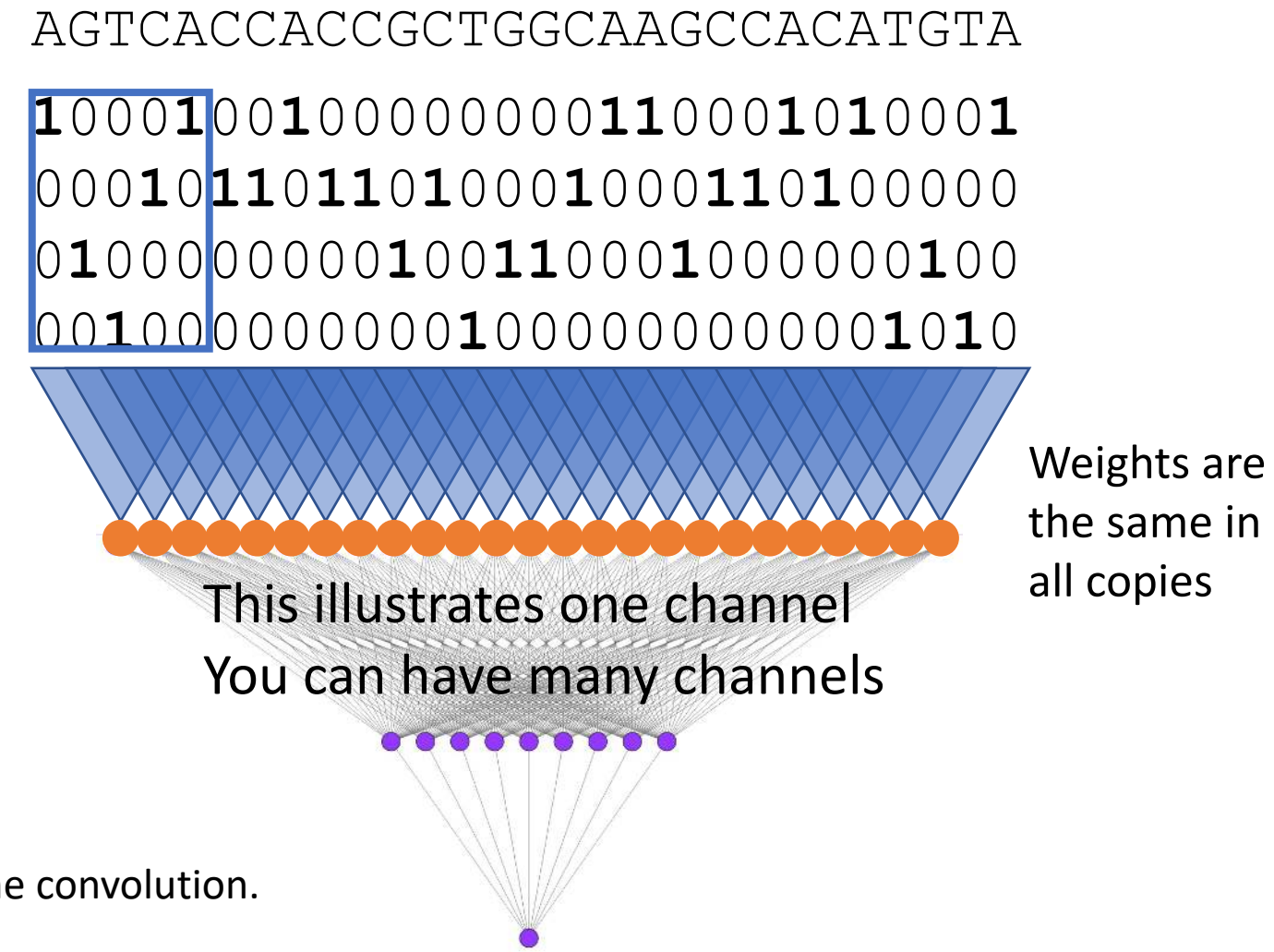
```
tensor([[0, 0, 0, 1],
        [0, 0, 1, 0],
        [0, 0, 0, 1],
        [1, 0, 0, 0],
        [0, 0, 1, 0],
        [0, 1, 0, 0],
        [0, 0, 0, 1],
        [0, 1, 0, 0],
        [0, 0, 0, 1],
```

Convolution

Convolution is used when data are ordered like

- Letters in a sequence
- Words in a sentence
- Pixels in an image

Cannot be used for gene expression values



- The kernel size is the input geometry for the convolution.
- In the illustration the kerne size is (5,4)
- Stride is the number of inputs you skip. stride=1 here.
- Padding can be used to add zeros to each side of the sequence. padding=0 here.

Convolution for images

Follow the same recipe

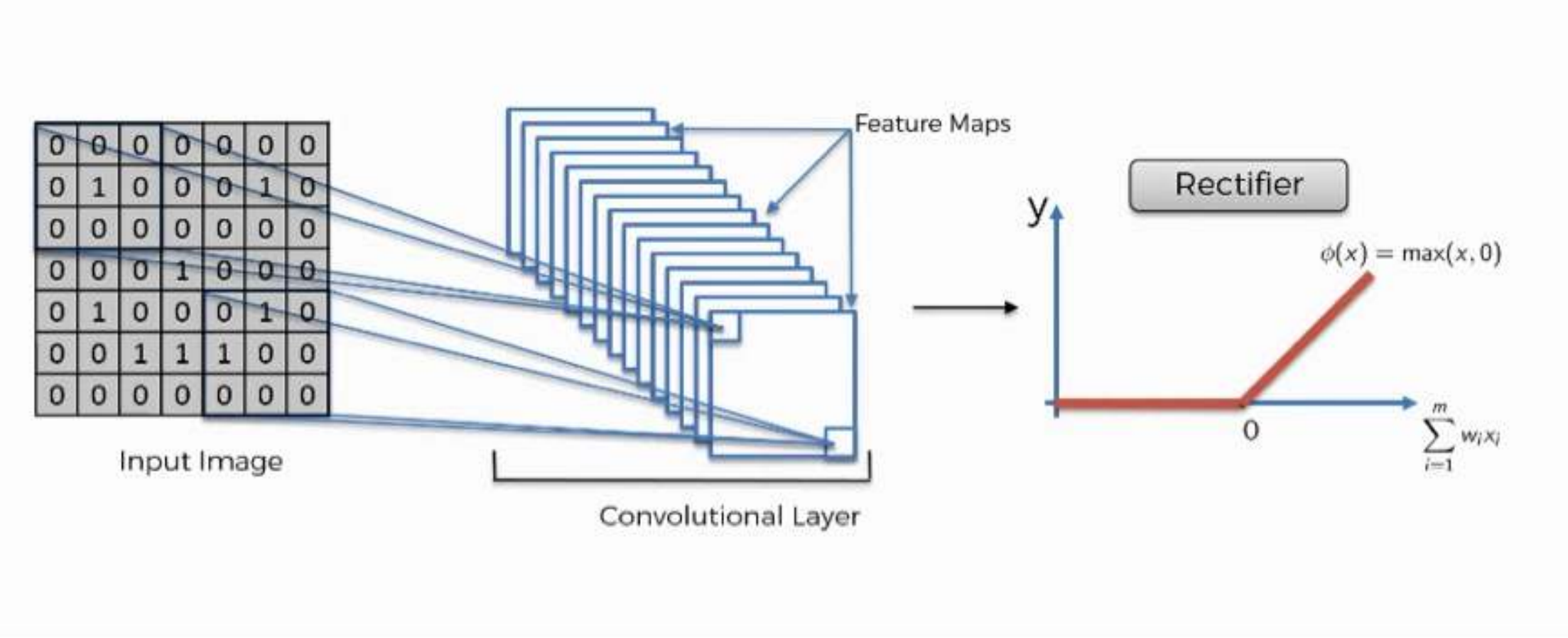


Image from
https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn

Convolution for images

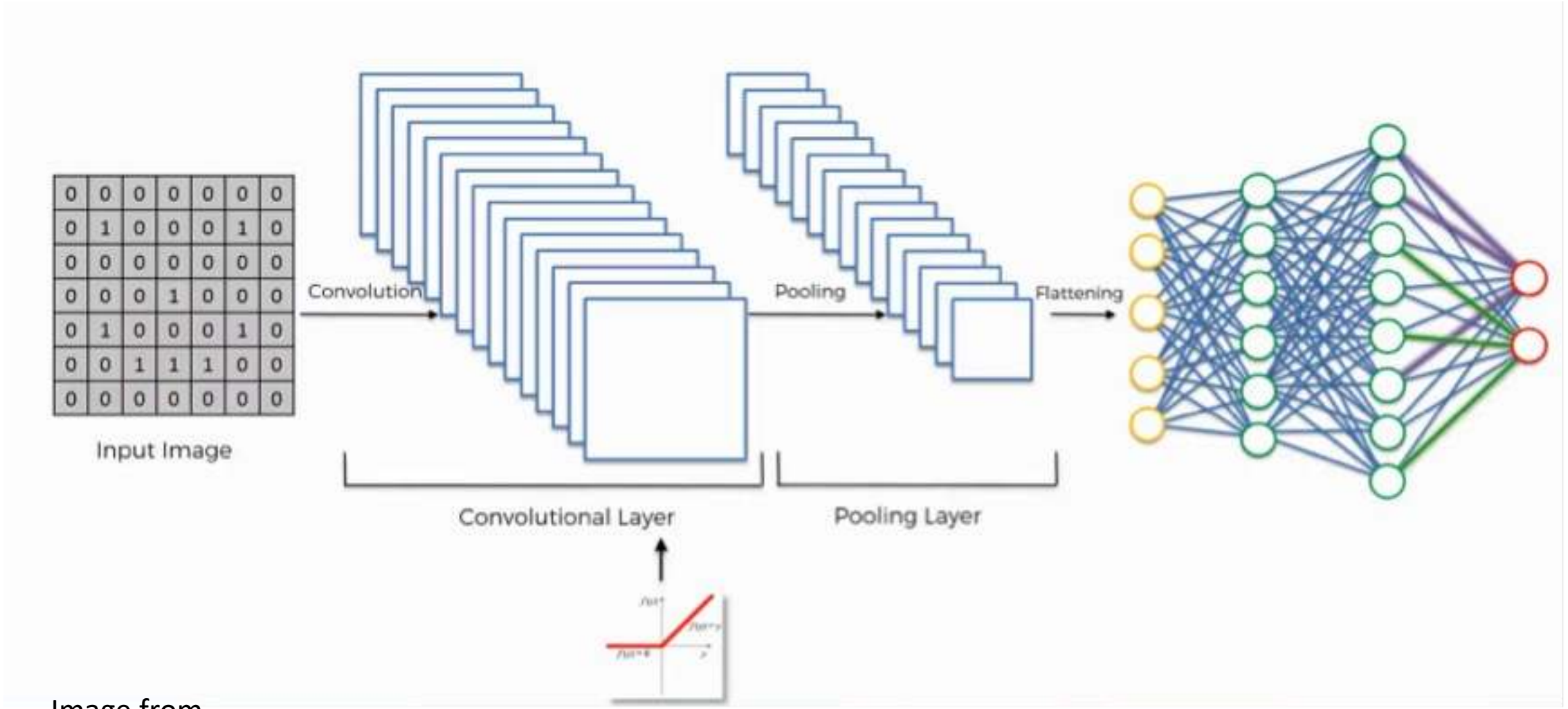
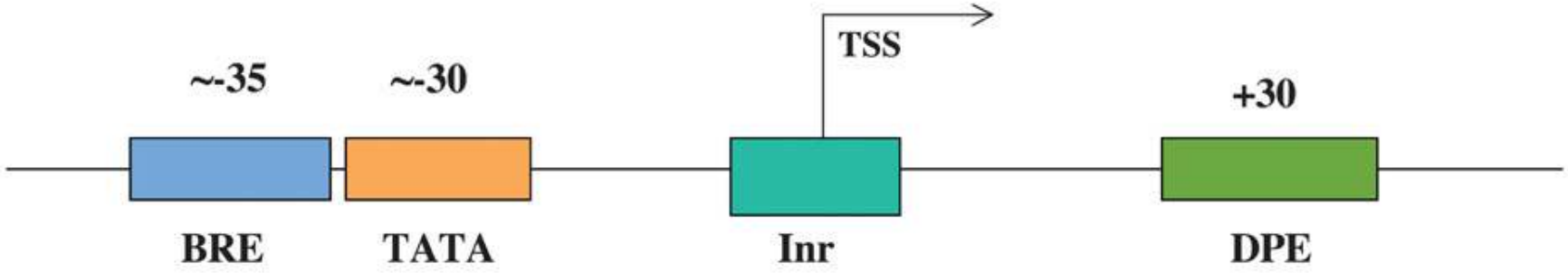


Image from
https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn

Exercise with human core promoters

The core promoter is the region around the transcription start site (TSS)



BRE: "B recognition element"
GGGGCGCC
CCAGCGCC

TATA box
TATAA

Initiation signal

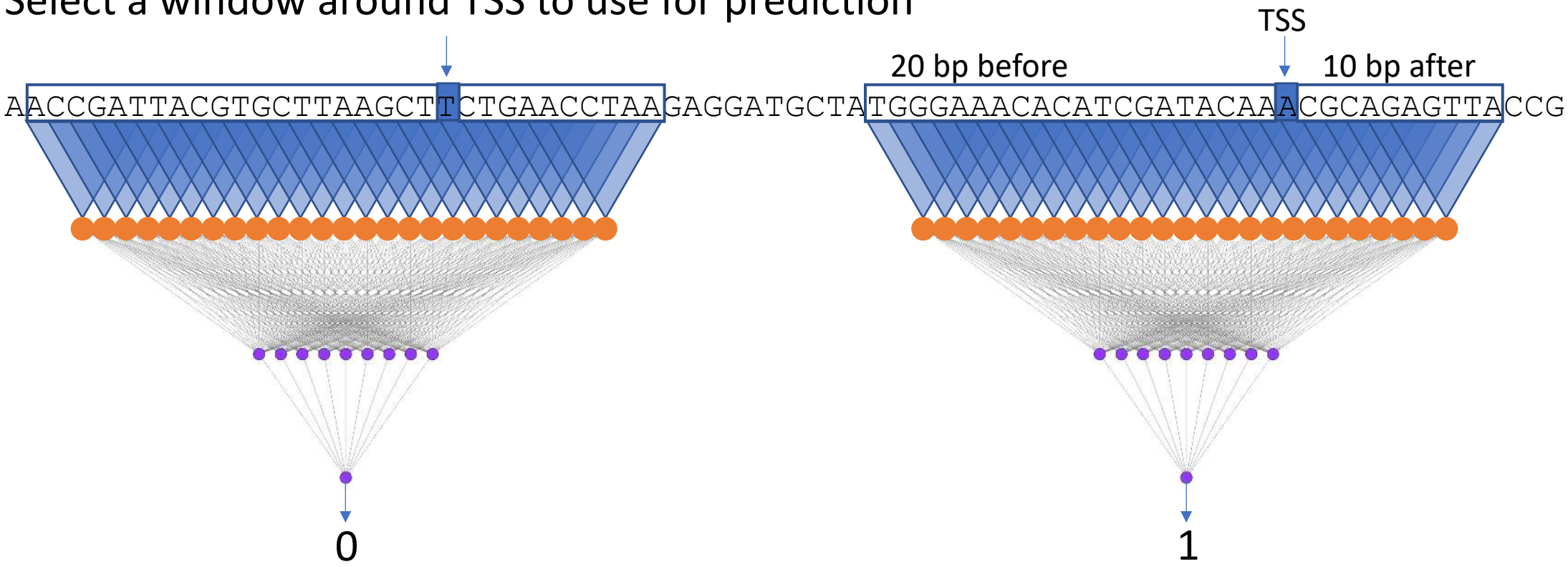
Downstream promoter elements

In the exercise we will try to predict the TSS from the DNA sequence around it

figure from Roy A.L. (2005) Core Promoters. In: Encyclopedic Reference of Genomics and Proteomics in Molecular Medicine. Springer, Berlin, Heidelberg . https://doi.org/10.1007/3-540-29623-9_2210

Predict TSS from DNA

Select a window around TSS to use for prediction



- We use one-hot encoding
- Negative examples are randomly sampled from +/- 1000 bases around TSS
- You can compare convolution and fully connected networks