CSI5180. Machine Learning for Bioinformatics Applications

Deep learning — architectures

by

Marcel Turcotte

Version **May 15, 2021**

Preamble

Preamble

Deep learning — architectures

In this lecture, we focus our attention on the **architecture** of deep learning networks. On the back cover of their book “Deep Learning”, Goodfellow, Bengio and Courville present deep learning as a form of machine learning that enables computers to “**understand the world in terms of a hierarchy of concepts**”. This idea was implicit when we looked at the model behind feed forward networks: $h_{W,b}(X) = f_k(\ldots f_2(f_1(X))\ldots)$, where $f_l(Z) = \phi(W_l Z + b_l)$ for $l = 1 \ldots k$. This idea becomes much more concrete when examining the various architectures; namely, convolution neural networks and recurrent neural networks.

General objective :

- **Discuss** the relationships between the nature of the problems to be solved and the architecture of deep networks.

Learning objectives

- **Discuss** the relationships between the nature of the problems to be solved and the architecture of deep networks.
- **Explain** convolution neural networks (CNN).
- **Describe** recurrent neural networks (RNN).

Reading:

- Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby, An introduction to deep learning on biological sequence data: examples and solutions, *Bioinformatics* **33**:22, 36853690, 2017.
- Seonwoo Min, Byunghan Lee, and Sungroh Yoon, Deep learning in bioinformatics, *Brief Bioinform* **18**:5, 851869, 2017.

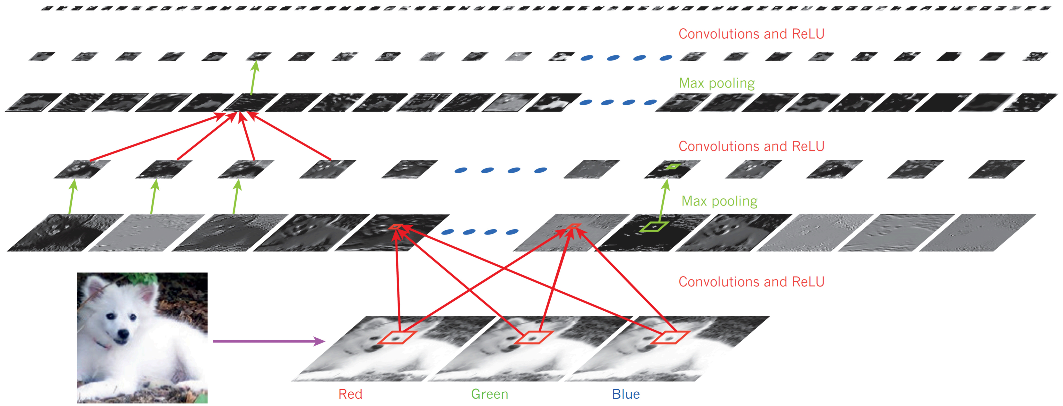
Reading

- Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby, An introduction to deep learning on biological sequence data: examples and solutions, *Bioinformatics* **33**:22, 36853690, 2017.
  - The authors solve **three (3) bioinformatics problems** using deep networks:
    - **Subcellular localization**
    - **Protein secondary structure**
    - **Peptide binding to MHCII molecules**
  - For each problem, they discuss the **pros** and **cons** of **convolutional networks** and **recurrent networks**.
  - `https://github.com/vanessajurtz/lasagne4bio`

Plan

Introduction

Hierarchy of concepts



- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, Deep learning, *Nature* **521**:7553, 43644, 2015.

Hierarchy of concepts

- Each layer detects **patterns** from the output of the **layer preceding it**.

Hierarchy of concepts

- Each layer detects **patterns** from the output of the **layer preceding it**.
  - In other words, proceeding from the input to the output of the network, the network uncovers “**patterns of patterns**”.

Hierarchy of concepts

- Each layer detects **patterns** from the output of the **layer preceding it**.
  - In other words, proceeding from the input to the output of the network, the network uncovers “**patterns of patterns**”.
    - Analyzing an image, the networks first detect simple patterns, such as **vertical**, **horizontal**, **diagonal** lines, **arcs**, etc.

Hierarchy of concepts

- Each layer detects **patterns** from the output of the **layer preceding it**.
  - In other words, proceeding from the input to the output of the network, the network uncovers “**patterns of patterns**”.
    - Analyzing an image, the networks first detect simple patterns, such as **vertical**, **horizontal**, **diagonal** lines, **arcs**, etc.
    - These are then combined to form **corners**, **crosses**, etc.

Hierarchy of concepts

- Each layer detects **patterns** from the output of the **layer preceding it**.
  - In other words, proceeding from the input to the output of the network, the network uncovers “**patterns of patterns**”.
    - Analyzing an image, the networks first detect simple patterns, such as **vertical**, **horizontal**, **diagonal** lines, **arcs**, etc.
    - These are then combined to form **corners**, **crosses**, etc.
- This explains how **transfer learning** works and why selecting the bottom layers only.

But also . . .

*"An MLP with just **one hidden layer** can theoretically model even the most **complex functions**,*

But also . . .

*"An MLP with just **one hidden layer** can theoretically model even the most **complex functions**, provided it **has enough neurons**.*

But also . . .

*"An MLP with just **one hidden layer** can theoretically model even the most **complex functions**, provided it **has enough neurons**. But for complex problems, **deep networks** have a much **higher parameter efficiency** than shallow ones:*

But also . . .

*“An MLP with just **one hidden layer** can theoretically model even the most **complex functions**, provided it **has enough neurons**. But for complex problems, **deep networks** have a much **higher parameter efficiency** than shallow ones: they can model complex functions **using exponentially fewer neurons** than shallow nets,*

But also . . .

*"An MLP with just **one hidden layer** can theoretically model even the most **complex functions**, provided it **has enough neurons**. But for complex problems, **deep networks** have a much **higher parameter efficiency** than shallow ones: they can model complex functions **using exponentially fewer neurons** than shallow nets, allowing them to reach much **better performance** with the same amount of training data." [4] §10*

How many layers?

- Start with one layer, then **increase the number of layers** until the model starts **overfitting** the training data.
- **Finetune** the model adding regularization (dropout layers, regularization terms, etc.).

The number of neurons and other hyperparameters are determined using a grid search.

Remarks

- Consider a **feed-forward network** (FFN) and its model:

$$h_{W,b}(X) = f_k(\dots f_2(f_1(X))\dots)$$

where

$$f_l(Z) = \phi(W_l Z + b_l)$$

for $l = 1 \dots k$.

Remarks

- Consider a **feed-forward network** (FFN) and its model:

$$h_{W,b}(X) = f_k(\dots f_2(f_1(X))\dots)$$

  where

$$f_l(Z) = \phi(W_l Z + b_l)$$

  for $l = 1 \dots k$.

- The **number of parameters** in grows rapidly:

$$(\text{size of layer}_{l-1} + 1) \times \text{size of layer}_l$$

Remarks

- Consider a **feed-forward network** (FFN) and its model:

$$h_{W,b}(X) = f_k(\dots f_2(f_1(X))\dots)$$

where

$$f_l(Z) = \phi(W_l Z + b_l)$$

for $l = 1 \dots k$.

- The **number of parameters** in grows rapidly:

$$(\text{size of layer}_{l-1} + 1) \times \text{size of layer}_l$$

- **Two layers 1,000-unit** implies **1,000,000** parameters!

Convolutional Neural Network

Convolutional Neural Network (CNN)

- In many applications, the **important information** to detect patterns is **local** - the pixels forming an eye or the nucleotides forming a transcription factor binding site, etc.

Convolutional Neural Network (CNN)

- In many applications, the **important information** to detect patterns is **local** - the pixels forming an eye or the nucleotides forming a transcription factor binding site, etc.
- A **convolution layer** significantly decreases the number of parameters.

Convolutional Neural Network (CNN)

- In many applications, the **important information** to detect patterns is **local** - the pixels forming an eye or the nucleotides forming a transcription factor binding site, etc.
- A **convolution layer** significantly decreases the number of parameters.
  - **Unlike a dense layer**, a neuron (unit) in a **convolution layer** is **not** connected to all the units of the **previous layer**.

Convolutional Neural Network (CNN)

- In many applications, the **important information** to detect patterns is **local** - the pixels forming an eye or the nucleotides forming a transcription factor binding site, etc.
- A **convolution layer** significantly decreases the number of parameters.
  - **Unlike a dense layer**, a neuron (unit) in a **convolution layer** is **not** connected to all the units of the **previous layer**.
    - Each unit is connected to neurons in its **receptive fields** (a rectangular region).

Convolutional Neural Network (CNN)

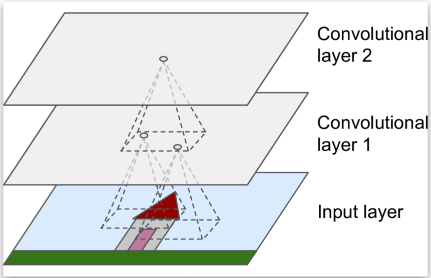
- In many applications, the **important information** to detect patterns is **local** - the pixels forming an eye or the nucleotides forming a transcription factor binding site, etc.
- A **convolution layer** significantly decreases the number of parameters.
  - **Unlike a dense layer**, a neuron (unit) in a **convolution layer** is **not** connected to all the units of the **previous layer**.
    - Each unit is connected to neurons in its **receptive fields** (a rectangular region).

Convolutional Neural Network (CNN)

- In many applications, the **important information** to detect patterns is **local** - the pixels forming an eye or the nucleotides forming a transcription factor binding site, etc.
- A **convolution layer** significantly decreases the number of parameters.
  - **Unlike a dense layer**, a neuron (unit) in a **convolution layer** is **not** connected to all the units of the **previous layer**.
    - Each unit is connected to neurons in its **receptive fields** (a rectangular region).

Convolutional networks come from the field of **machine vision**. Hence their close connection to **grid-like inputs**.

CNN - receptive field



Source: [4] Figure 14.2

- Each unit is connected to neurons in its **receptive fields**.
  - Unit i, j in layer l is connected to the units i to $i + f_h - 1$, j to $j + f_w - 1$ of the layer $l - 1$, where f_h and f_w are respectively the **height** and **width** of the **receptive field**.

CNN - receptive field

- **Zero padding**. In order to have layers of the same size, the grid can be padded with zeros.

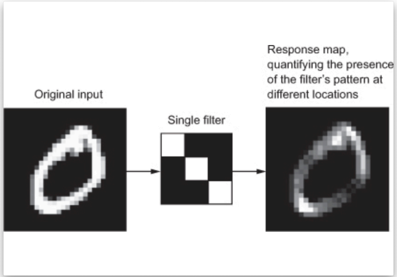
CNN - receptive field

- **Zero padding**. In order to have layers of the same size, the grid can be padded with zeros.
- **Stride**. It is possible to connect a large larger $(l-1)$ to a smaller one (l) by skipping units. The number of units skipped is called **stride**, s_h and s_w.

CNN - receptive field

- **Zero padding**. In order to have layers of the same size, the grid can be padded with zeros.
- **Stride**. It is possible to connect a large larger $(l-1)$ to a smaller one (l) by skipping units. The number of units skipped is called **stride**, s_h and s_w.
  - Unit i, j in layer l is connected to the units $i \times s_h$ to $i \times s_h + f_h - 1$, $j \times s_w$ to $j \times s_w + f_w - 1$ of the layer $l-1$, where f_h and f_w are respectively the **height** and **width** of the **receptive field**, s_h and s_w are respectively the **height** and **width strides**.

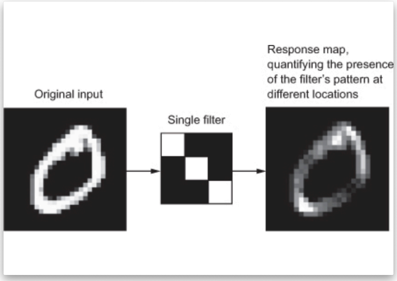
CNN - filters


Original input
Single filter
Response map, quantifying the presence of the filter's pattern at different locations

Source: [3] Figure 4.3

- A **window** of size $f_h \times f_w$ is moved over the output of the layers $l-1$ (this is called the **input feature map**) position by position.

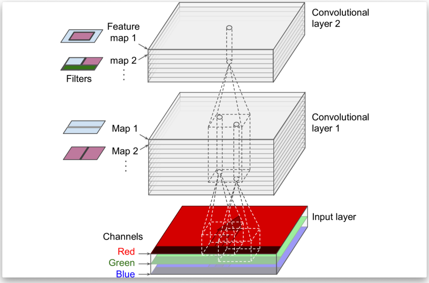
CNN - filters


Response map, quantifying the presence of the filter's pattern at different locations
Original input
Single filter

Source: [3] Figure 4.3

- A **window** of size $f_h \times f_w$ is moved over the output of the layers $l - 1$ (this is called the **input feature map**) position by position.
- **For each location**, it calculates the product between the extracted patch and a matrix of the same size, called a **convolution kernel** or **filter**. The **sum** of the values of the resulting matrix is the **output** for that location.

CNN - model



Source: [4] Figure 14.6

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} + \sum_{v=0}^{f_w-1} + \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k}$$

where $i' = i \times s_h + u$ and $j' = j \times s_w + v$; see [4] §14.

CNN - convolutional layer

[4] §14:

- "Thus, a layer full of neurons using the **same filter** outputs a **feature map**."

CNN - convolutional layer

[4] §14:

- “Thus, a layer full of neurons using the **same filter** outputs a **feature map**.”
- “Of course, you do not have to define the filters manually: instead, **during training the convolutional layer will automatically learn the most useful filters for its task**,”

CNN - convolutional layer

[4] §14:

- "Thus, a layer full of neurons using the **same filter** outputs a **feature map**."
- "Of course, you do not have to define the filters manually: instead, **during training the convolutional layer will automatically learn the most useful filters for its task**,"
- "(. . .) and **the layers above will learn to combine them** into more complex patterns."

CNN - convolutional layer

[4] §14:

- "Thus, a layer full of neurons using the **same filter** outputs a **feature map**."
- "Of course, you do not have to define the filters manually: instead, **during training the convolutional layer will automatically learn the most useful filters for its task**,"
- "(. . .) and **the layers above will learn to combine them** into more complex patterns."
- "The fact that **all neurons in a feature map share the same parameters** dramatically reduces the number of parameters in the model."

CNN - convolution (ressources)

- **A guide to convolution arithmetic for deep learning**
- Vincent Dumoulin and Francesco Visin
- Last revised 11 Jan 2018
    - https://arxiv.org/abs/1603.07285
    - https://github.com/vdumoulin/conv_arithmetic/

Pooling

Pooling

- A **pooling** layer has similar characteristics to a **convolutional layer**.

Pooling

- A **pooling** layer has similar characteristics to a **convolutional layer**.
    - In particular, each neuron is connected to neurons in a **receptive field**.

Pooling

- A **pooling** layer has similar characteristics to a **convolutional layer**.
    - In particular, each neuron is connected to neurons in a **receptive field**.
- However, a pooling layer has no **weights**.

Pooling

- A **pooling** layer has similar characteristics to a **convolutional layer**.
  - In particular, each neuron is connected to neurons in a **receptive field**.
- However, a pooling layer has no **weights**.
  - It returns the result of an aggregating function. Typically **max** or **mean**.

Pooling

- A **pooling** layer has similar characteristics to a **convolutional layer**.
  - In particular, each neuron is connected to neurons in a **receptive field**.
- However, a pooling layer has no **weights**.
  - It returns the result of an aggregating function. Typically **max** or **mean**.
- This (subsampling) has the effect of **shrinking the network**, each window of size $f_h \times f_w$ is reduced to a single value, **max** or **mean** of that window.

Pooling

- A **pooling** layer has similar characteristics to a **convolutional layer**.
  - In particular, each neuron is connected to neurons in a **receptive field**.
- However, a pooling layer has no **weights**.
  - It returns the result of an aggregating function. Typically **max** or **mean**.
- This (subsampling) has the effect of **shrinking the network**, each window of size $f_h \times f_w$ is reduced to a single value, **max** or **mean** of that window.
- "[A] max pooling layer also introduces some level of **invariance to small translations**." [4] §14

Recurrent Neural Network

Recurrent Neural Network (RNN)

- **Recurrent neural networks** (**RNN**) take as input **sequence data** (time series, text, speech, and biological sequence data).

Recurrent Neural Network (RNN)

- **Recurrent neural networks** (**RNN**) take as input **sequence data** (time series, text, speech, and biological sequence data).
- **Unlike feed forward networks**, **RNN** contain **loops**.

Recurrent Neural Network (RNN)

- **Recurrent neural networks** (**RNN**) take as input **sequence data** (time series, text, speech, and biological sequence data).
- **Unlike feed forward networks**, **RNN** contain **loops**.
- Furthermore, whereas **feed forward networks** are **memory less (stateless**), **RNN store information**.

Recurrent Neural Network (RNN)

- **Recurrent neural networks** (**RNN**) take as input **sequence data** (time series, text, speech, and biological sequence data).
- **Unlike feed forward networks**, **RNN** contain **loops**.
- Furthermore, whereas **feed forward networks** are **memory less (stateless**), **RNN store information**.
- **Each unit** stores information about its **own state**.

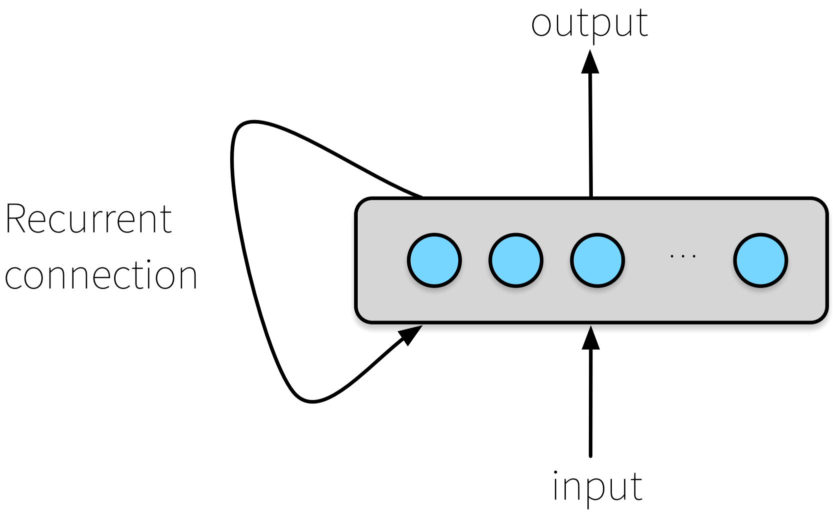
Recurrent Neural Network (RNN)

- **Recurrent neural networks** (**RNN**) take as input **sequence data** (time series, text, speech, and biological sequence data).
- **Unlike feed forward networks**, **RNN** contain **loops**.
- Furthermore, whereas **feed forward networks** are **memory less (stateless)**, **RNN store information**.
- **Each unit** stores information about its **own state**.
  - Let $h_{u,l}$ be the state of the unit u in layer l.

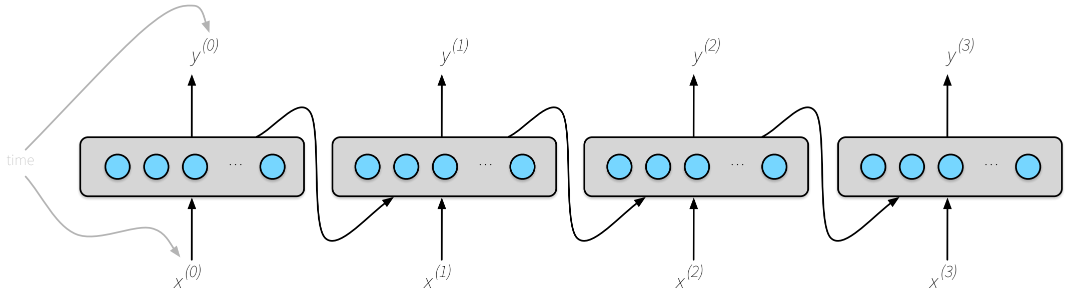
Recurrent Neural Network (RNN)

- **Recurrent neural networks** (**RNN**) take as input **sequence data** (time series, text, speech, and biological sequence data).
- **Unlike feed forward networks**, **RNN** contain **loops**.
- Furthermore, whereas **feed forward networks** are **memory less (stateless**), **RNN store information**.
- **Each unit** stores information about its **own state**.
  - Let $h_{u,l}$ be the state of the unit u in layer l.
- **Each unit** has two inputs, as before the output of the units from the **previous layer** ($l-1$), but also the vector of states for this layer (l) at the **previous time step**.

Recurrent Neural Network



Recurrent Neural Network



$$Y^{(t)} = \phi(X^{(t)} W_X + Y^{(t-1)} W_Y + b)$$

Recurrent Neural Network

- The backpropagation algorithm is adapted and becomes a **backpropagation through time** (**BPTT**).

Recurrent Neural Network

- The backpropagation algorithm is adapted and becomes a **backpropagation through time** (**BPTT**).
- With time, information from a **distant pass is forgotten**.

Recurrent Neural Network

- The backpropagation algorithm is adapted and becomes a **backpropagation through time** (**BPTT**).
- With time, information from a **distant pass is forgotten**.
- **Long Short-Term Memory** (**LSTM**) are networks with long-term memory.

```
model = keras.models.Sequential([
    keras.layers.LSTM(20, return_sequences=True, input_shape=[None, 1]),
    keras.layers.LSTM(20, return_sequences=True),
    keras.layers.TimeDistributed(keras.layers.Dense(10))
])
```

Recurrent Neural Network

- The backpropagation algorithm is adapted and becomes a **backpropagation through time (BPTT)**.
- With time, information from a **distant pass is forgotten**.
- **Long Short-Term Memory (LSTM)** are networks with long-term memory.

```
model = keras.models.Sequential([
    keras.layers.LSTM(20,return_sequences=True,input_shape=[None,1]),
    keras.layers.LSTM(20,return_sequences=True),
    keras.layers.TimeDistributed(keras.layers.Dense(10))
])
```

- **Bidirectional LSTM** cells also exist.

Dropout

Dropout

- **Dropout** layers are **regularization** mechanisms.

Dropout

- **Dropout** layers are **regularization** mechanisms.
- During **training**, each unit in a dropout layer has probability p of being ignored.

Dropout

- **Dropout** layers are **regularization** mechanisms.
- During **training**, each unit in a dropout layer has probability *p* of being ignored.
  - According to [4] §11:

Dropout

- **Dropout** layers are **regularization** mechanisms.
- During **training**, each unit in a dropout layer has probability p of being ignored.
  - According to [4] §11:
    - 20-30% is a typical value of p **convolution networks**;

Dropout

- **Dropout** layers are **regularization** mechanisms.
- During **training**, each unit in a dropout layer has probability p of being ignored.
  - According to [4] §11:
    - 20-30% is a typical value of p **convolution networks**;
    - whereas, 40-50% is a typical of p for **recurrent networks**.

Dropout

- **Dropout** layers are **regularization** mechanisms.
- During **training**, each unit in a dropout layer has probability p of being ignored.
  - According to [4] §11:
    - 20-30% is a typical value of p **convolution networks**;
    - whereas, 40-50% is a typical of p for **recurrent networks**.
- Hinton and colleagues say that they are "**preventing co-adaptation**".

Dropout

- **Dropout** layers are **regularization** mechanisms.
- During **training**, each unit in a dropout layer has probability p of being ignored.
  - According to [4] §11:
    - 20-30% is a typical value of p **convolution networks**;
    - whereas, 40-50% is a typical of p for **recurrent networks**.
- Hinton and colleagues say that they are "**preventing co-adaptation**".
- **Dropout layers** can make the network converging more slowly. However, the resulting network is expected to make **fewer generalization errors**.

Summary

- **Convolutional Neural Network** are able to detect patterns **irrespective** of their location in the input.

Summary

- **Convolutional Neural Network** are able to detect patterns **irrespective** of their location in the input.
  - **Pooling** makes the network less sensitive to small translations.

Summary

- **Convolutional Neural Network** are able to detect patterns **irrespective** of their location in the input.
  - **Pooling** makes the network less sensitive to small translations.
  - In bioinformatics, **CNN** networks are ideally suited to detect local (sequence) motifs, independent of their position within the input (sequence).

Summary

- **Convolutional Neural Network** are able to detect patterns **irrespective** of their location in the input.
  - **Pooling** makes the network less sensitive to small translations.
  - In bioinformatics, **CNN** networks are ideally suited to detect local (sequence) motifs, independent of their position within the input (sequence).
- **Recurrent networks (RNN)** and **LSTM** can input sequences of varying length.

Keras

```
model = keras.models.Sequential([
    keras.layers.Conv2D(64, 7, ..., input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation="softmax")
])
```

[4] §14:

Further considerations

Further considerations

We obviously barely scratched the surface of deep learning. Here are some important concept that we did not consider:

- The **vanishing** and **exploding** gradient.
- Initialization.
- **Data augmentation**.
- **Attention** layer.
- Understanding what the network has learnt.

Prologue

Summary

- Deep networks consisting only of **dense layers** become **computationally intractable** as the number of parameters grows exponentially with each additional layer.

Summary

- Deep networks consisting only of **dense layers** become **computationally intractable** as the number of parameters grows exponentially with each additional layer.
- **Convolutional layers** considerably reduce the number of parameters since each unit is connected to a limited number of neurons from the previous layer, its **receptive field**.

Summary

- Deep networks consisting only of **dense layers** become **computationally intractable** as the number of parameters grows exponentially with each additional layer.
- **Convolutional layers** considerably reduce the number of parameters since each unit is connected to a limited number of neurons from the previous layer, its **receptive field**.
- **CNN** is able to detect patterns in a positon independent manner.

Summary

- Deep networks consisting only of **dense layers** become **computationally intractable** as the number of parameters grows exponentially with each additional layer.
- **Convolutional layers** considerably reduce the number of parameters since each unit is connected to a limited number of neurons from the previous layer, its **receptive field**.
- **CNN** is able to detect patterns in a positon independent manner.
- **RNN** and **LSTM** handle sequence information, where the input sequences can be of different lengths. They can detect patterns along the sequence.

Summary

- Deep networks consisting only of **dense layers** become **computationally intractable** as the number of parameters grows exponentially with each additional layer.
- **Convolutional layers** considerably reduce the number of parameters since each unit is connected to a limited number of neurons from the previous layer, its **receptive field**.
- **CNN** is able to detect patterns in a positon independent manner.
- **RNN** and **LSTM** handle sequence information, where the input sequences can be of different lengths. They can detect patterns along the sequence.
- **Dropout** layers are an effective regularization mechanism.

Next module

- **Concept**- and **rule**-based

References

Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby.
An introduction to deep learning on biological sequence data: examples and solutions.
Bioinformatics, 33(22):3685–3690, Nov 2017.

Seonwoo Min, Byunghan Lee, and Sungroh Yoon.
Deep learning in bioinformatics.
Brief Bioinform, 18(5):851–869, 09 2017.

François Chollet.
Deep learning with Python.
Manning Publications, 2017.

Aurélien Géron.
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.
O'Reilly Media, 2nd edition, 2019.

References

Andriy Burkov.
The Hundred-Page Machine Learning Book.
Andriy Burkov, 2019.

Marcel **Turcotte**
Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EECS)
University of Ottawa