A Convolutional Neural Network Solution For Facial Expression Classification

Alexandru Cohal
March, 2018

Prof. Dr.-Ing. Andreas König

Overview

1. Motivation
2. Introduction
3. Objective
4. Data Acquisition
5. Architecture
6. Performance
7. Hyperparameters
8. Results
9. Conclusion

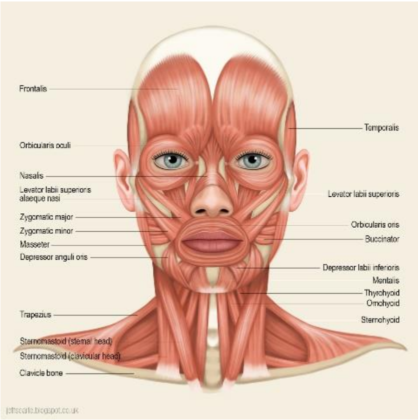
Motivation

- The world's older population is growing (in present, approximately *8.5%* of global population is aged *65 and over*)
- High number of persons with diminished abilities (*1 out of 5* persons has hearing issues, *1 out of 4* persons has vision issues, *1 out of 4* persons has pains in wrists / hands)
- In the USA in *2013*, *17.6%* of people with a disability were employed
- The general current trend → **make technology available to more users with different needs and goals** → simplify people's lives and sustain the economy

http://careforyou.us/making-technology-available-to-everyone/
https://www.dosomething.org/facts/11-facts-about-physical-disability

Introduction

- A **Brain – Computer Interface (BCI)** → a solution for *Human – Computer Interface* → uses signals captured from the brain for controlling an external activity
  - **Electroencephalography (EEG)** → a method for monitoring the electrical neural activity of the brain
  - **Electromyography (EMG)** → monitoring the electrical activity of the muscle tissue → facial expressions can be detected

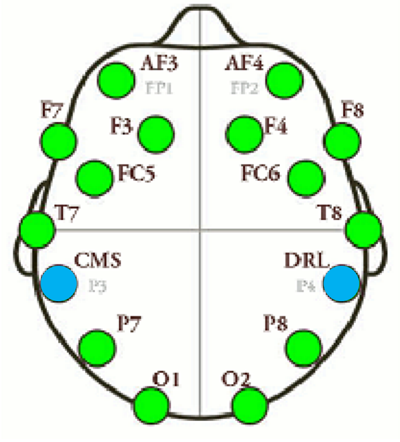


    - Blink
    - Left / Right wink
    - Raise brow
    - Smile

https://jeffsearle.blogspot.de/2015/04/muscles-of-head-and-neck.html

Introduction

- **Emotiv EPOC** ➞ a compact, wireless EEG device, easy to setup and use
- Has *16* sensors (electrodes): *14* for data (µV) and *2* for reference



- Data is acquired with *128 Hz* frequency
- A saline solution has to be used for increasing conductivity

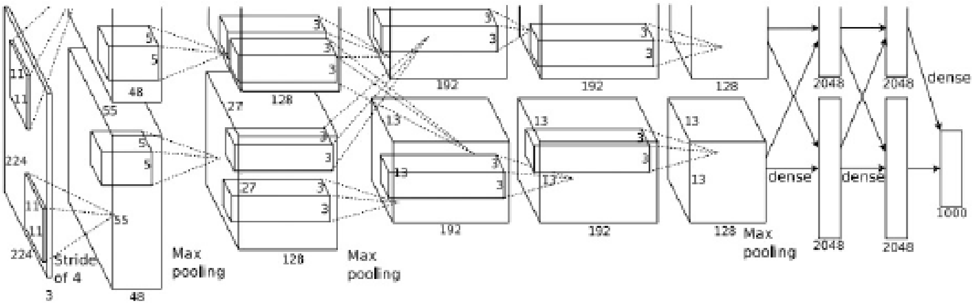
Investigating the Emotiv EPOC for cognitive control in limited training time – M. Lang, T. Mitrovic, *University of Canterbury* (2012)
https://www.emotiv.com/epoc/

Introduction

- **Deep Learning** → a **Machine Learning** family of methods for learning data representations (discover useful features for classification)

- It is composed of multiple processing layers → used to learn representations of data with multiple levels of abstraction

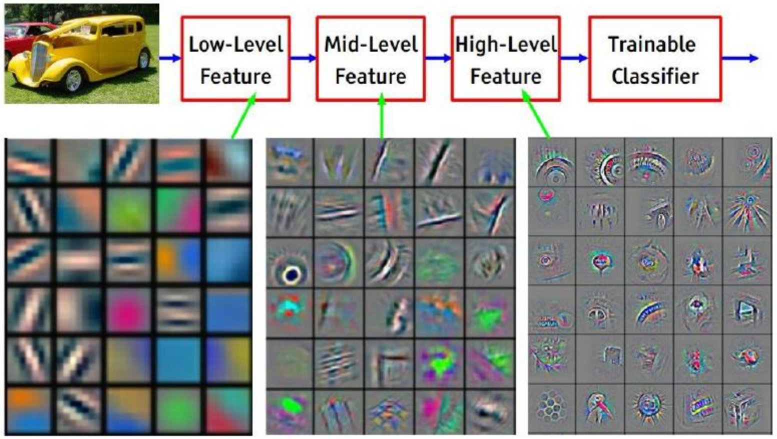
- These methods have obtained very good results → speech recognition, visual object recognition, object detection, etc.



Deep Learning - Y. LeCun et al., *http://www.bioinfo.org.cn/~casp/temp/DeepLearning.pdf*
ImageNet Classification with Deep Convolutional Neural Networks – A. Krizhevsky et al., Advances in Neural Information Processing Systems 25 (2012)

Introduction

- **Classic classification** ➡ manual engineering of features ➡ fed into a machine learning algorithm ➡ high understanding of the domain is required
- **Deep Learning** ➡ does not require manual engineering of features ➡ it learns during training multiple "filters" with increasing complexity as the layers get deeper ➡ uses them in a final classifier



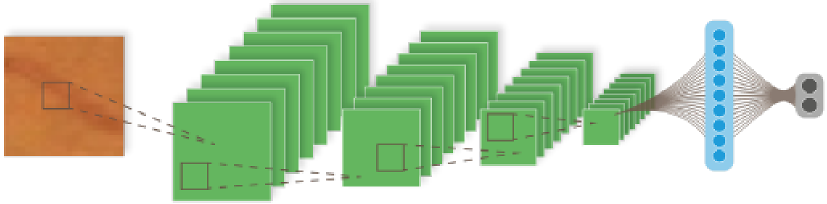
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Deep Learning - Y. LeCun et al., *http://www.bioinfo.org.cn/~casp/temp/DeepLearning.pdf*
https://burakhimmetoglu.com/2017/08/22/time-series-classification-with-tensorflow/

Introduction

- **Convolutional Neural Networks (CNN)** ➡ is a class of Deep Learning
- It is similar to a classic Artificial Neural Network ➡ only that it exploits spatially-local correlation

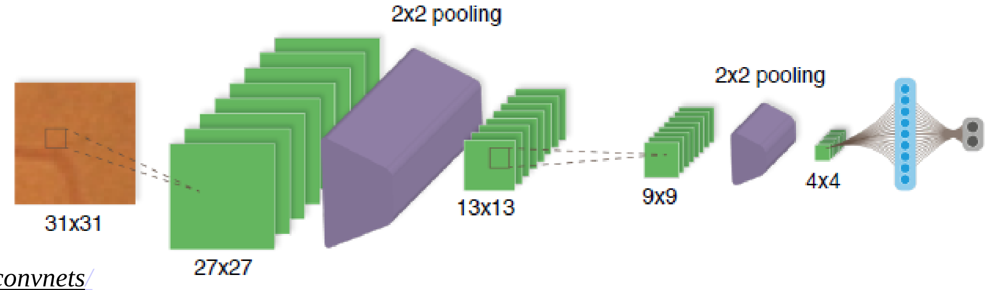
- A **convolutional layer** ➡ convolves the data with multiple small kernels (rolls the filter over the data) ➡ these kernels are learned, not predefined
- By stacking multiple convolutional layers ➡ more and more complex features can be learned



http://deeplearning.net/tutorial/lenet.html
***"Computer Vision" course** – F. Lindseth, NTNU, 2017*

Introduction

- **Convolutional Neural Networks (CNN)** → different layers:
  - ***Input Layer*** – holds the raw input data
  - ***Convolution Layer*** – computes the convolution (output of neurons) between the data and the kernel → multiple kernels are used → the output is a volume, not only a single layer
  - ***ReLU Layer*** (usually included in the Convolution Layer) – applies the ReLU actiation function → indroduces non-linearity
  - ***Pooling Layer*** – downsamples along the spatial dimensions → a smaller volume results
  - ***Fully connected layer*** – computes the score for each class



"Computer Vision" course *– F. Lindseth, NTNU, 2017*
http://cs231n.github.io/convolutional-networks/
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Introduction

- **Tensorflow** → an open-source machine learning framework
- Uses data flow graphs → the user's written code describes a computation graph → the graph fully describes the desired computation
- It is a portable framework → can run on multiple platforms: CPUs, GPUs, mobile, embedded
- Has interfaces for *C++*, *Java* and *Python*

https://www.tensorflow.org/
https://opensource.com/article/17/11/intro-tensorflow

Objective

- Using the *14* channels data provided by the *Emotiv EPOC* headset, classify using a Convolutional Neural Network (CNN) the *Neutral* state and *5* facial expressions:

  - Left wink / blink (BL)
  - Right wink / blink (BR)
  - Strong blink (BB)
  - Open mouth (OM)
  - Full mouth (FM)

Data Acquisition

- For each class:
  - *20 recording* of *100 samples* each (*0.78 sec*) → the length was observed to be enough for capturing the whole behavior of the considered facial expressions
  - Each recording contains data from all the *14 channels* (sensors)
  - Only from one person → different persons have different behavior of the same facial expressions
  - Only from one session of recording → data is influenced by the position of the headset, quantity of saline solution used

Example of data acquired while performing a "Left Blink" expression

Architecture

- The data acquired from each sensor of the headset is *1D* ➡ **Time-series CNN** is used ➡ the kernels (filters) used are *1D* (not *2D* as in the case of images)

- At each sampling time ➡ Each sensor of the *Emotiv EPOC* headset returns a single values
- The headset has *14 sensors* ➡ at each sample the headset returns an array of *14 values*
- The training is not performed on data from individual time instances (it is irrelevant, can be affected by noise) ➡ windows are used ➡ the input of the CNN will be a *2D* array

Architecture

- The (general) architecture of the CNN is:

Layer	*Data Size*
Input layer	$32\ x\ 14$
1D Convolution layer	$32\ x\ 28$
1D Pooling layer	$16\ x\ 28$
1D Convolution layer	$16\ x\ 56$
1D Pooling layer	$8\ x\ 56$
1D Convolution layer	$8\ x\ 112$
1D Pooling layer	$4\ x\ 112$
1D Convolution layer	$4\ x\ 224$
1D Pooling layer	$2\ x\ 224$
Fully connected layer	$1\ x\ 6$

Convolution layer: each filter has size 2 and moves with stride 1

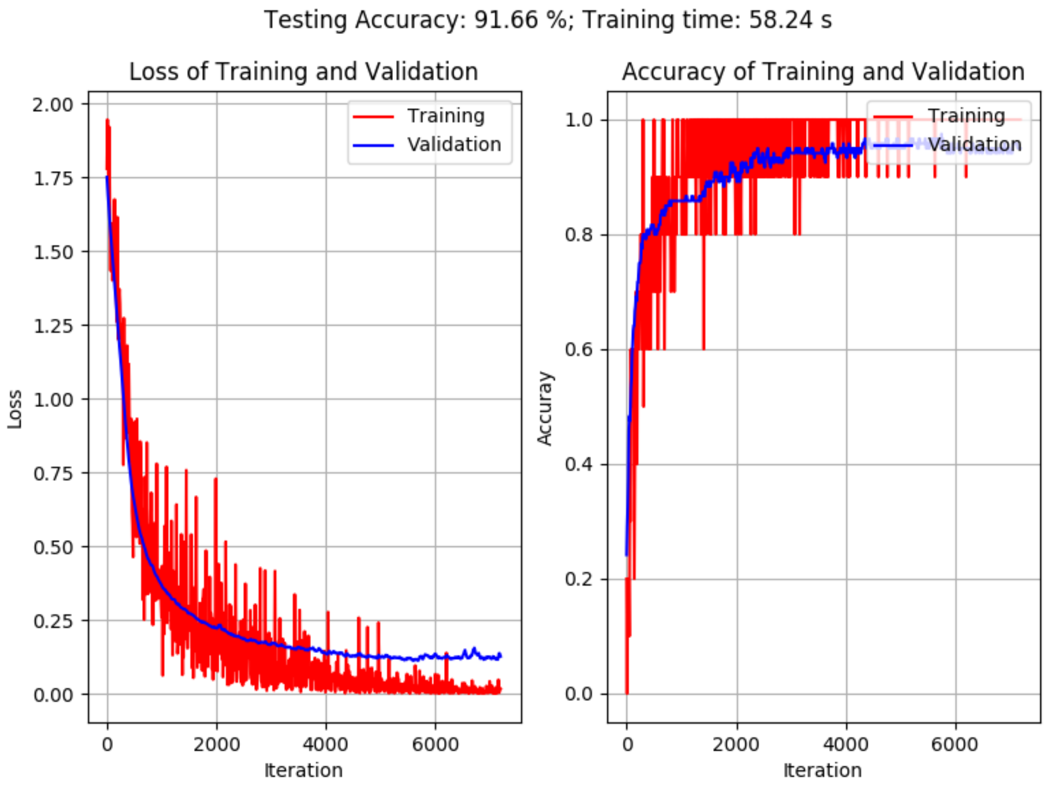
Pooling layer: the pooling size is 2 and the stride is 2

https://burakhimmetoglu.com/2017/08/22/time-series-classification-with-tensorflow/

Performance

- The performance of the CNN can be evaluated through:
  - ***Loss*** → used to optimize the parameters of the network → sum of errors made for each example → better to be as close as possible to *0*
    - Loss of training dataset
    - Loss of verification dataset
  - ***Accuracy*** → used to evaluate how good are the predictions after the network's parameters are optimized
    - Accuracy of training dataset
    - Accuracy of verification dataset
    - Accuracy of testing dataset
  - ***Training time*** → if the training has to performed multiple times, it is better to be as small as possible

Performance



Hyperparameters

- What can be changed in the CNN in order to obtain better results:

  – ***Training / Validation / Testing datasets ratios***
    - *Training* ➡ the dataset used to adjust the parameters (weights) of the network
    - *Validation* ➡ used to estimate how well the network has trained while training ➡ used for tuning the network hyperparameters (e.g. number of hidden neurons), for early stopping, for observing overfitting
    - *Testing* ➡ the dataset used after the training is done, to prove the predictive capacity of the network

    - The tests performed used the same ratios:
      – Training: *60%*, Validation: *20%*, Testing: *20%*

Hyperparameters

- What can be changed in the CNN in order to obtain better results:

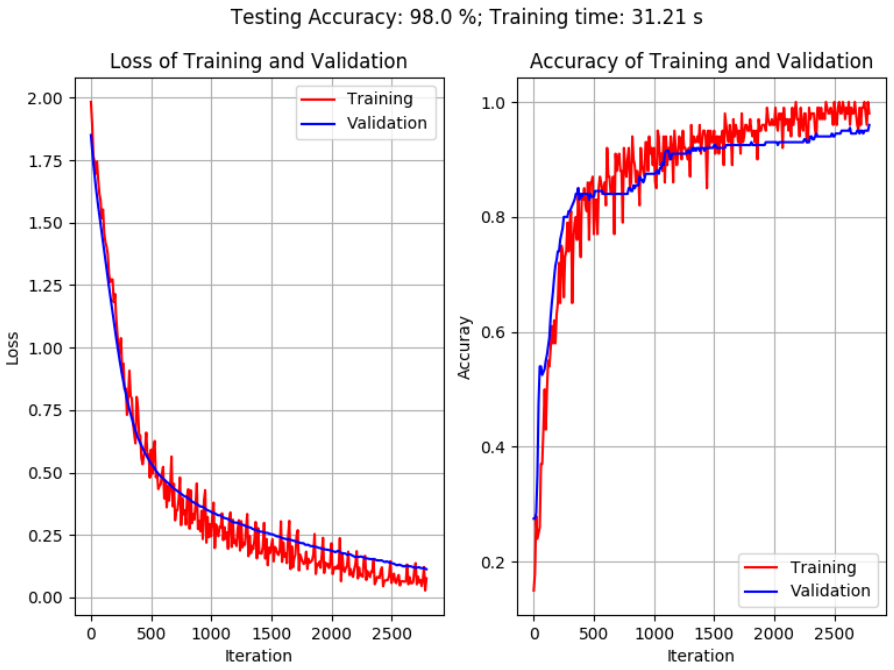
  – ***Window size and Overlapping size of windows***
    - Larger windows → too much information, too less samples for training
    - Smaller windows → too few relevant information (learn noise), too many samples for training, longer training time, not enough time for predicting in real-time
    - Larger overlapping → too many irrelevant samples, longer training time, overfitting
    - Lower overlapping → too few samples, loose the behavior from the border

    - Window sizes used: *16* and *32* samples
    - Overlapping size used: *0*, *8*, *16* and *24* samples

Hyperparameters

- What can be changed in the CNN in order to obtain better results:

  – ***Window size and Overlapping size of windows***



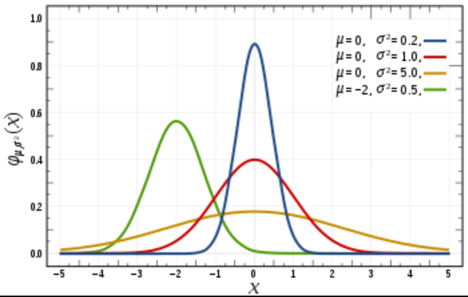
Too small window (16), long training time (400 epochs are not enough)

Hyperparameters

- What can be changed in the CNN in order to obtain better results:

  – ***Artificial input data generation***
    - For each window from the original dataset, multiple windows can be obtained by adding noise
    - Too many artificial data used ➡ overfitting, learn noise
    - Less artificial data used ➡ too less data for training

    - Normal distributed noise with mean *20* and standard deviation *10* was used in order to increase the available data *2*, *3* and *4* times

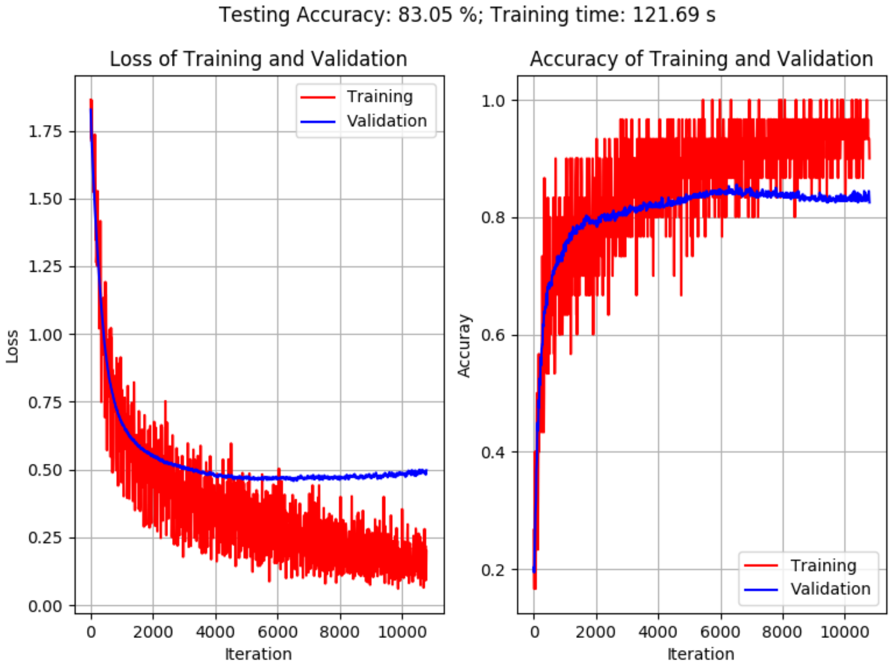


https://en.wikipedia.org/wiki/Normal_distribution

Hyperparameters

- What can be changed in the CNN in order to obtain better results:

  – ***Artificial input data generation***



Too many artificial data used (3 times more than the original data), Overfitting

Hyperparameters

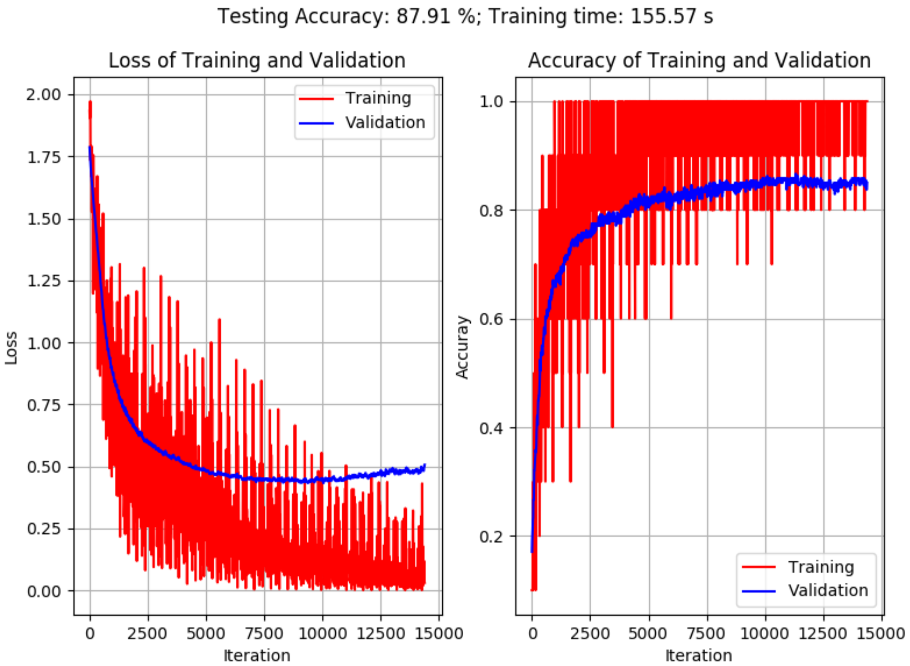
- What can be changed in the CNN in order to obtain better results:

  – ***Batch size***
    - *Batch* ➡ the number of training samples used for one update of the network's parameters
    - Too larger batches ➡ few training batches, computational expensive
    - Too small batches ➡ many training batches, slow, noisy variation of the parameters

    - Batch sizes used: between *10* and *100* samples

Hyperparameters

- What can be changed in the CNN in order to obtain better results:
  - ***Batch size***



Too small batches (10 samples), Noisy behavior, Overfitting

Hyperparameters

- What can be changed in the CNN in order to obtain better results:

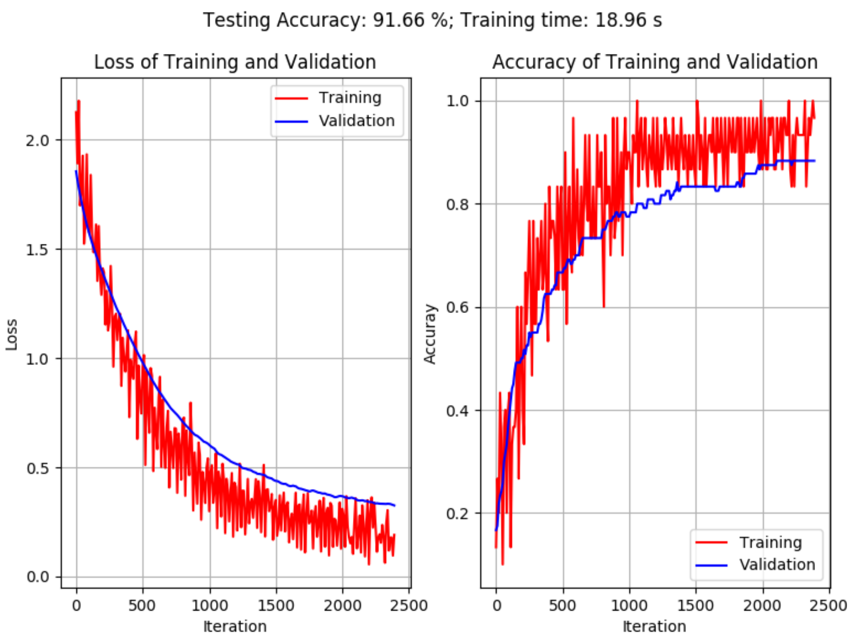
  – ***Number of epochs***
    - Epoch → the number of times the networks sees the entire training dataset → different from ***Iteration*** → the number of times the networks sees a batch
    - Too many epochs → overfitting
    - Too few epochs → underfitting

    - Number of epochs used: *200*, *300*, *400*, *500* and *600*

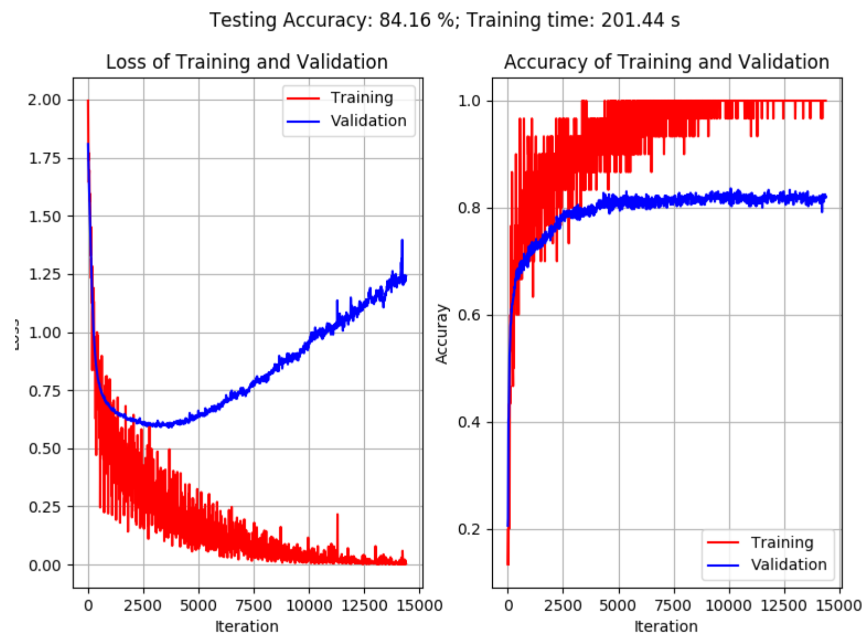
Hyperparameters

- What can be changed in the CNN in order to obtain better results:

  – ***Number of epochs***



Underfitting (200 epochs)



Overfitting (400 epochs)

Hyperparameters

- What can be changed in the CNN in order to obtain better results:

  – ***Learning rate***
    - *Learning rate* → the size of the "step" done in the direction of the negative gradient
    - Too large learning rate → oscillations
    - Too small learning rate → slow convergence, stuck in local minima

    - Learning rate used: *0.0001*

Hyperparameters

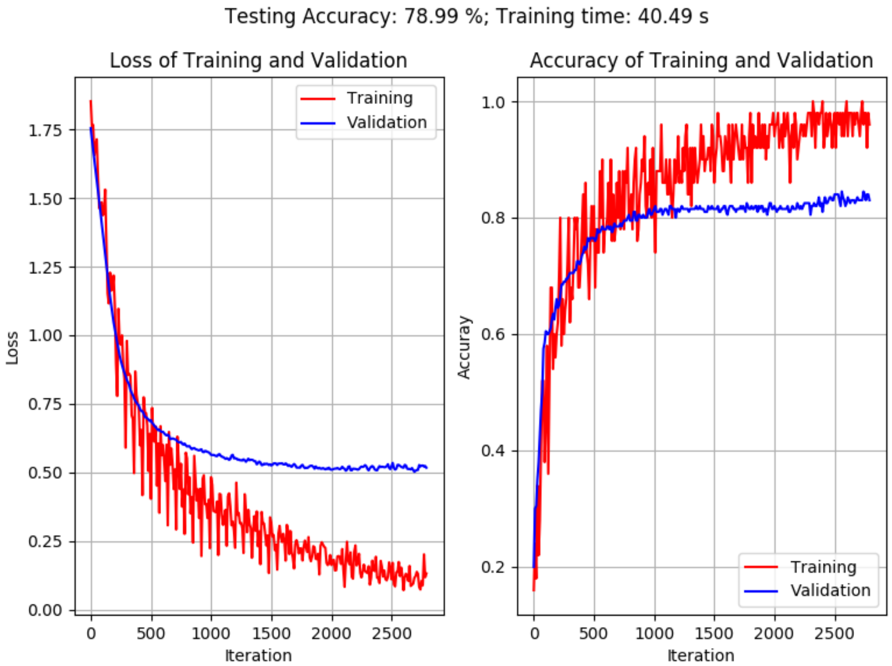
- What can be changed in the CNN in order to obtain better results:

  – ***Network structure***
    - Change of number of layers
    - Change of layers' sizes
    - Too less layers ➡ loose the advantage of Deep Learning (learn representations of data with multiple levels of abstraction)
    - Too many layers ➡ overfitting, vanishing gradient

    - Number of layers used: *1*, *2*, *3* and *4*
    - Layers' sizes were changed when the window size was changed

Hyperparameters

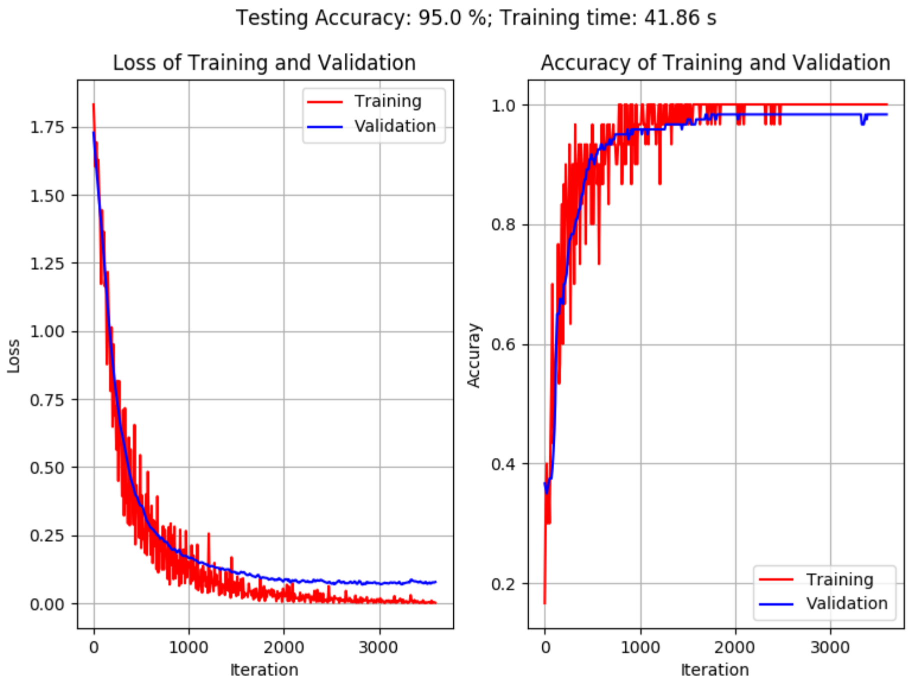
- What can be changed in the CNN in order to obtain better results:
  - ***Network structure***



Too many layers (4), Overfitting

Results

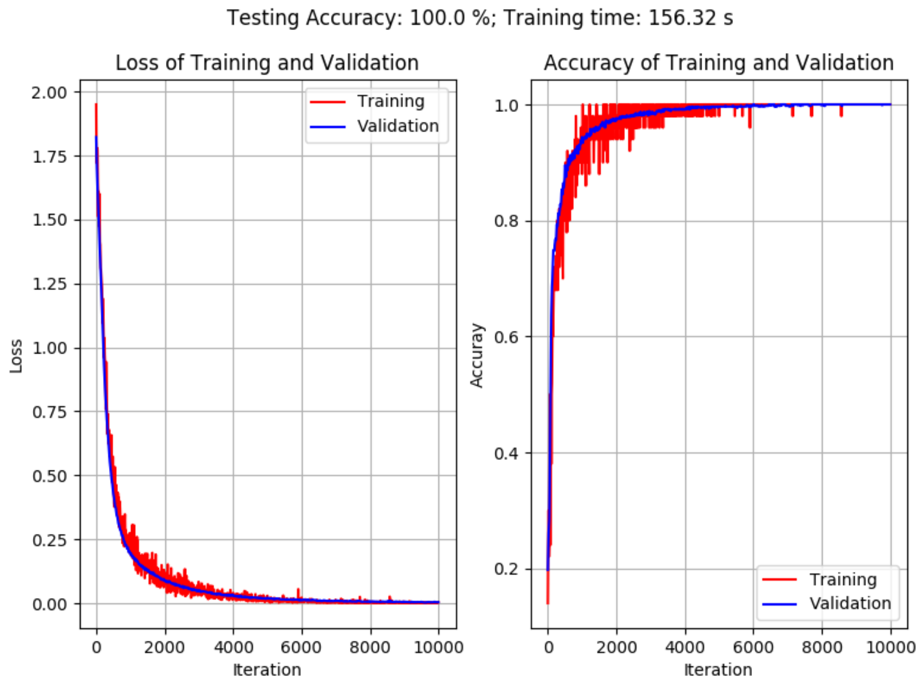
- ***<u>The best result obtained</u>***
  - 4 Layers, Window size: 32 samples, Overlapping: 16 samples, No artificial data, Batch size: 30 samples, Epochs: 300
  - Testing accuracy: ***95%***



Results

- ***Better than the best result obtained***
  - 3 Layers, Window size: 32 samples, Overlapping: 30 samples, No artificial data, Batch size: 50 samples, Epochs: 200
  - Testing accuracy: ***100%***

The evolutions of loss and accuracy are smoother



Not replicable

"Some things in life only happen once, the memories of them lasting forever"

J.M.Darhower

Conclusions

- **Time-series Convolutional Neural Network (CNN)** → can be used for classification of facial expressions based on the signals acquired from the *Emotiv EPOC* headset
- The accuracy obtained was very good → *95%* (even *100%* once) for the testing dataset
- For this case → not a very large dataset was required (*20 recordings* of *0.78 seconds* each for every class) and the training was not long (*42 seconds*)
- These results → only for one person

- The possibility of generalization could be studied → a large dataset will be required (data from multiple persons and for different positions of the headset, quantities of saline solution used, etc.)


THANK
YOU