Project AM-FED DataSet

Jens Röwekamp, Paul Velthuis, Zahin Azher Rashid
rowekamp@kth.se, velthuis@kth.se, zarashid@kth.se

Abstract

This project is about detecting facial expressions from videos. It involves a dataset with 242 videos with labelled facial expressions. For this project we use the dataset: Afectiva-MIT Facial Expression Dataset (AM-FED): Naturalistic and Spontaneous Facial Expressions Collected In-the-Wild Paper. In this dataset there is labelled data available about gestures people make during the video. In this project we analyse if people are smiling or not based on the data from this dataset.

Introduction

As mentioned this project involves analysing the Afectiva-MIT Facial Expression Dataset (AM-FED): Naturalistic and Spontaneous Facial Expressions Collected In-the-Wild Paper. This paper provides 242 labelled videos. In this paper we provide information about how we use deep learning techniques to detect facial expressions of the people in the videos. The dataset provides information about different gestures. In the next section, we provide information about the method and approach we used to accomplish this project. We will then provide the requirements and the setup code for installing each requirement on a Linux environment. This project cannot be run on windows due to some limitations.

What we are going to do

Extract frames/images from videos and then analyse them on emotions, for example happy and not happy, sad and angry. We save our code and documentation on our github repository: https://github.com/PaulVelthuis93/Scalable_Deep_Learning

Requirements

To analyse the videos for emotions we will perform deep learning. It provides the possibility to detect smiles. In order to perform deep learning there are several requirements which are described below.
Tensorflow (latest version 0.12) is an open source library released by Google. Tensorflow provides the ability to perform deep learning on images. It can make use of the GPU to perform image processing. With image processing it is possible to recognize images, and thus to recognize gestures.
Tensorflow requires Python 2.7 or higher to run. Since Python 2.7 is widely supported we decided to use Python 2.7.

On Linux Python is often pre-installed, with the following command, you can see which version you have:

```
$ python --version  #When reading such code you always type the text after the $ in your terminal
```

If Python is not installed:

```
$ sudo apt-get install python2.7
```

To install Tensorflow there are installation guidelines to be found in the following link:

https://www.tensorflow.org/get_started/os_setup

In order to install Tensorflow the pip package management system of Python needs to be installed. If not you can install it on Linux with:

```
$ sudo apt-get install python-pip python-dev
```

Tensorflow performs better on the GPU than the CPU. To install version 0.12 the current latest version on a computer or laptop without GPU you type:

```
$ export TF_BINARY_URL=https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.12.0rc1-cp27-none-linux_x86_64.whl
```

If you have a GPU on your computer or laptop you install with the following command:

```
$ export TF_BINARY_URL=https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-0.12.0rc1-cp27-none-linux_x86_64.whl
```

Tensorflow can only operate on image data, for this reason we need to extract the images from the videos. For this we use OpenCV (Version 2.4.9.1), with which it is possible to capture the image from a certain time in the video. To install OpenCV type:

```
$ sudo apt-get install python-opencv
```

We use the Python Image Library (PIL) to extract the information from an image and to transform images into grayscale. To install the PIL library type:

```
$ sudo apt-get install python-imaging
```

We make a numpy array of the pixels to analyse the images, so it is also required. Usage for numpy is available at the following link: https://www.scipy.org/install.html

Numpy can be installed with the following command:

```
$ sudo apt-get install python-numpy
```

Steps taken

1. To start with analysing the videos, we first take the image from the videos. The videos contain a CSV file which contains information about which gestures were shown at which moment in time. So we can extract labelled images with for example the information whether the person is smiling or not. Thus our first step is to extract the labeled images from the videos.
2. Next, analyses of the images was rendered using Tensorflow. In the coding section, we explain more about how our Tensorflow algorithm works.

Techniques

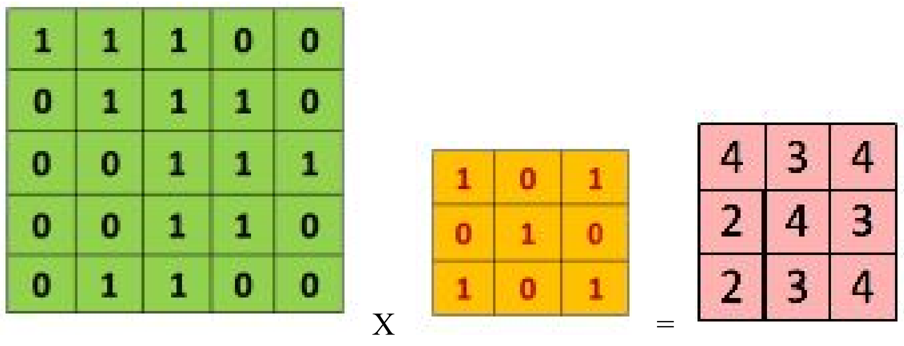
As earlier discussed we will use Tensorflow to detect gestures in images, like smiling.
We used Tensorflow model for deep learning. Other options we could use were Theano or Caffe but we choose Tensorflow because it has support by Google and this framework has been properly tested and used by millions of people.
Within Tensorflow we will use Convolutional Neural Networks to detect these gestures in the images. We also use regularization to prevent overfitting in the images. We explain the dropout technique below. We use Rectified Linear Unit (ReLU) to transform negative values, this is also explained below. We also use learning rate decay.

Steps we couldn’t perform

1. The Spearman brown test discussed in the paper for the dataset could not be performed. All the problems are further discussed in the Problems encountered section.

Convolutional Neural Networks

A convolution is a mathematical operation on two functions and it results in a third function.
We use pooling to shrink the image stack, here we can go in smaller windows over the image and calculate for each window the max value. So we can create from for example a 5x5 image to a 3x3 image. For a better explanation see the pictures below. The 3x3 image slides over the 5x5 image and adding the multiplicative result at each index to generate the output matrix.



Rectified Linear Unit (ReLU)
A Relu Layer is being used to transform all negatives values from the image analysis to zeros and all the positive values remain the same.

Regularization (Dropout)
We used dropout to make sure that the images are not trained on similar images to avoid the problem of overfitting.

Explanation AM-FED dataset

Each video in the AM-Fed dataset has a csv file which contains the probabilities of all gestures at a particular instant of time and the time itself:

Time	AU04	AU02	AU15	TrackerFail	AU18	AU09	AU10	Expressive	AU12	AU14	AU05

Code files

GenerateLabeledData and GenerateLabeledDataMoreFrames

The GenerateLabeledDataMoreFrames script generates labeled frames from videos with a corresponding csv file. These frames are labeled by emotion, for example Happy, not Happy, Sad, Contempt or Angry. To do this the most important function of this script is being called. This function is called process_video. Here, we walk through the directory and read the csv file for every video then for every row in the csv file we first get the header information, so the time, the smile and the AU labels. We then get the content for each row. For that content we then check for the emotion e.g. happy or sad. We then generate a frame for each of these emotions. If the emotion continues for a longer time period e.g. 2 seconds then we can take multiple frames pictures. There can be a maximum of 14 frames per second taken
The functions names and the descriptions is provided in the Appendix section.

TensorFlowModelClassifySadOrHappy

This script is used to classify happy and not happy gestures.

To achieve this labelled pictures are loaded into our model.

These are then transformed into numpy array dataSets with their labels accordingly. So we get a trainingSet and a testingSet.

After the testingSet and trainingSet are created with their sets of labels as well, the TensorFlowModel is executed.

In the TensorFlowModel the actual learning process starts to happen.

We have a placeholder for the numpy images and a placeholder for the labels.

We use a convolutional layer, a max pooling layer and we use a dropout function in the end.

Then we apply softmax and cross entropy.

We use the AdamOptimizer to learn with a certain learning rate and we minimize with the cross entropy which is the loss. We also use learning rate decay.

Then we iterate in batches through the whole trainingSet. Each batch get's trained and there is a test batch.

In the end, the results are plotted in graphs to visualize the accuracy and loss of the testingSet and the trainingSet.

The functions names and the descriptions are provided in the Appendix section.

Results

Shuffling the images in the training and test datasets has a huge impact on the accuracy and avoids overfitting. Figure 1 below shows the results where the nonHappy images were appended to the happy images in the training set.

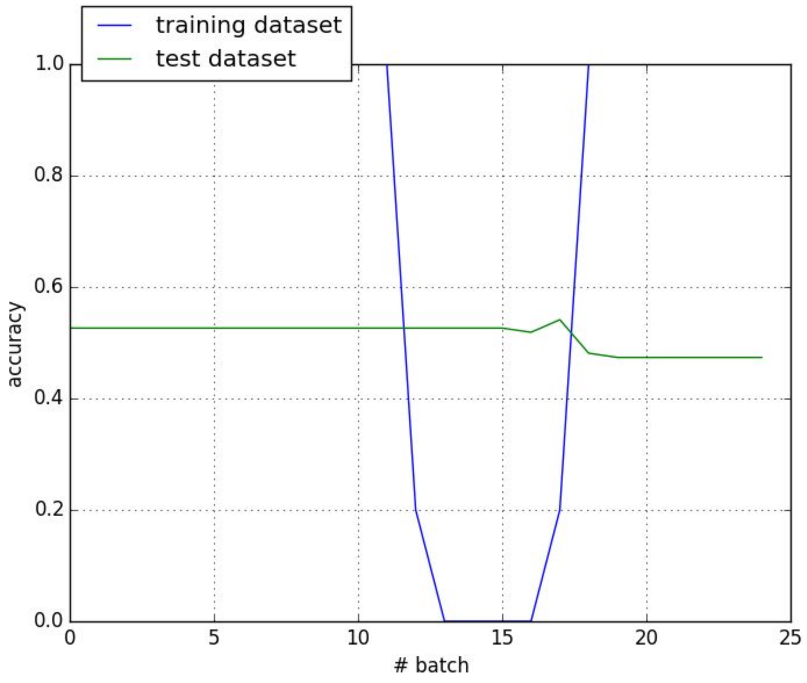


Figure 1: Overfitting due to appended non happy images

In the figure below we give the results for our test on a 30 GB memory machine, we ran it on the CPU. We analysed 9898 images from our dataset with happy and nonHappy images. 4949 sad images and 4949 happy images.
The batches corresponds to the number of iterations with 50 images in each batch.
The testing set consisted of 5% of the initial images, which resulted in a testing set of 511 images and a training set of 9387 images.

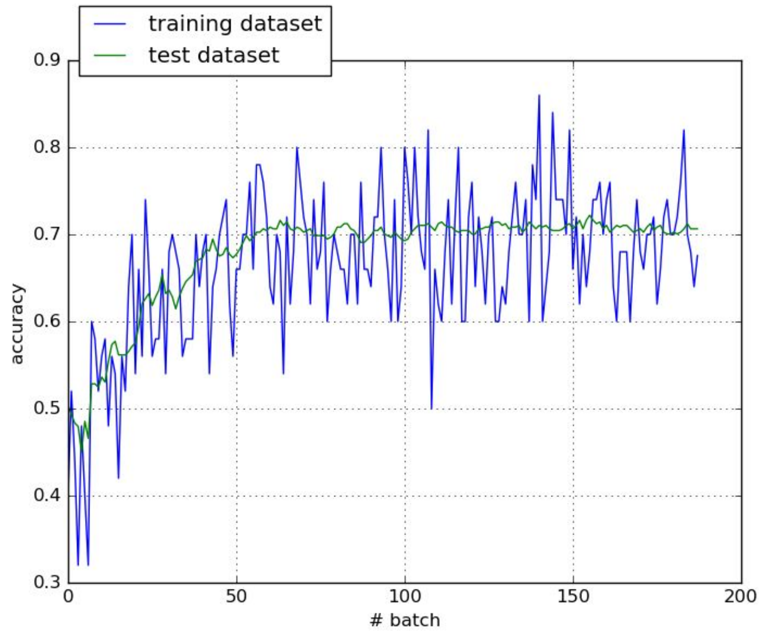


Figure 2: Accuracy equal size of happy and non-happy images

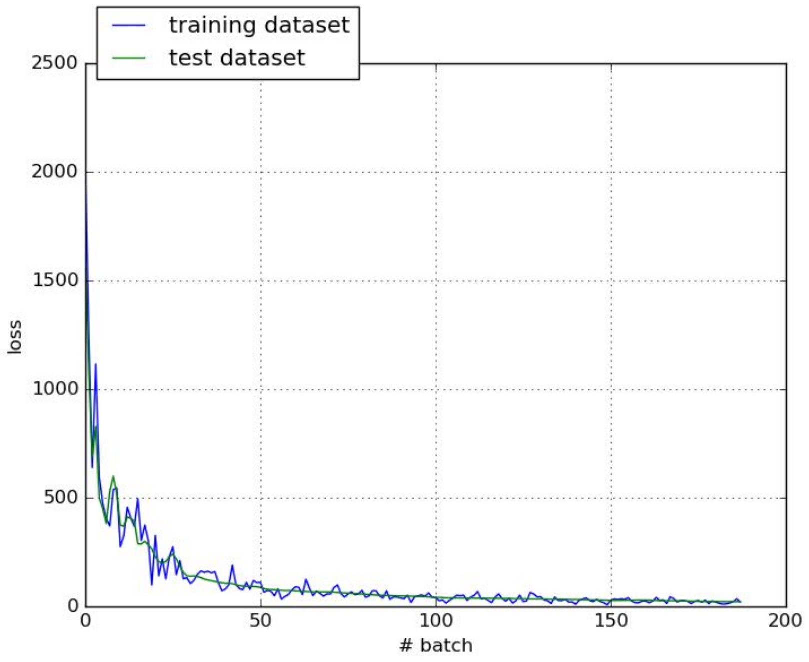


Figure 3: Loss equal size of happy and non-happy images

In the next experiment we had 100 images per batch, and we see the effect of the accuracy.

Testing split is 3% of the total images. The full batch test took around an hour to complete. Nonetheless, the amount of happy and non-happy images wasn't equal any more. We trained in total on 4949 happy images and 13722 non-happy images.

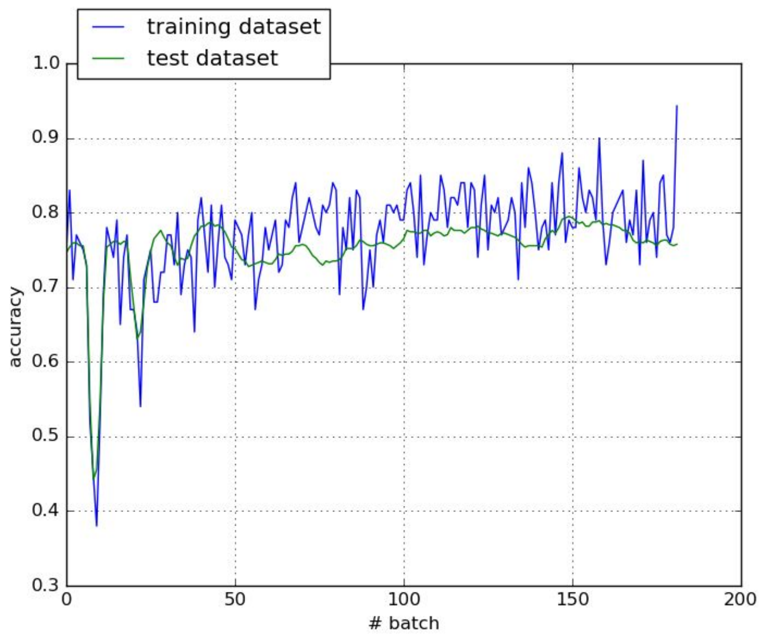


Figure 4: accuracy 4949 happy and 13722 non-happy images

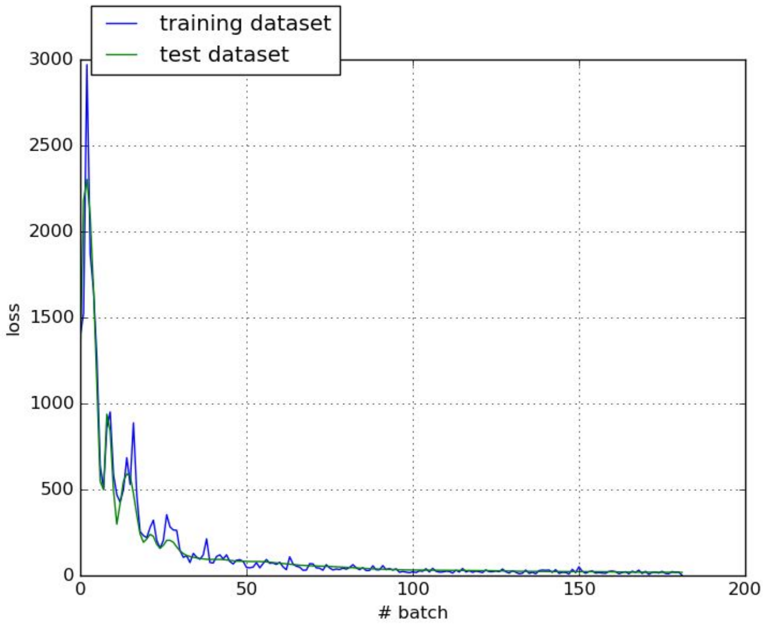


Figure 5: loss 4949 happy and 13722 non-happy images

We did another test with sad, happy, angry and contempt emotions using the script TensorFlowModelClassifySadHappyAngry.py. Here with a testing split of 25% we had an almost random result (cf. figure 6). This is partly due to the fact that we had only 98 sad and angry pictures. So our dataset could only contain around 300 images.

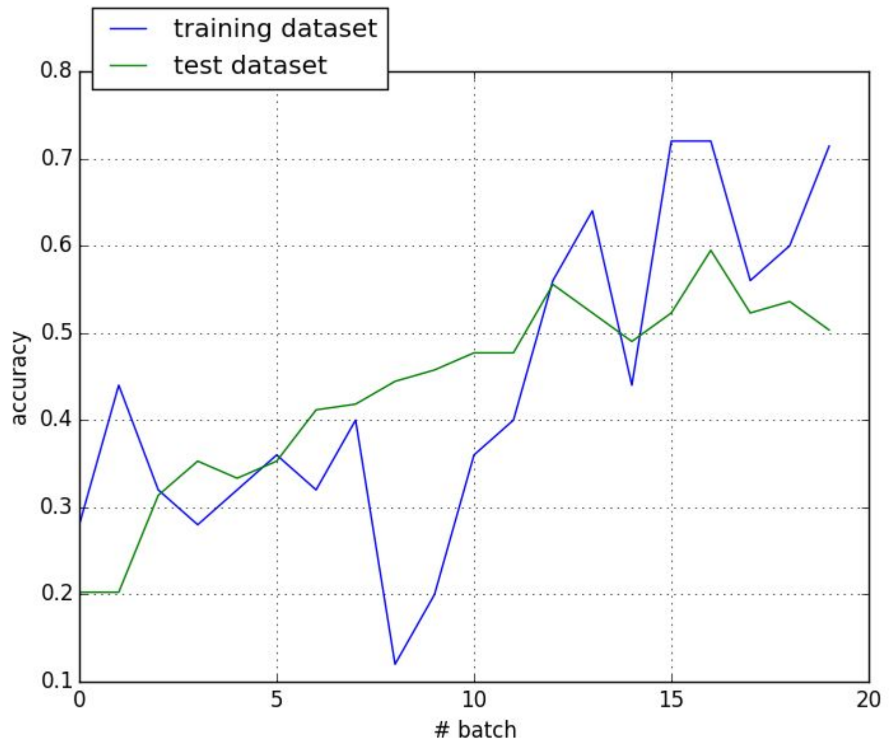


Figure 6: Accuracy sad (non-)happy angry

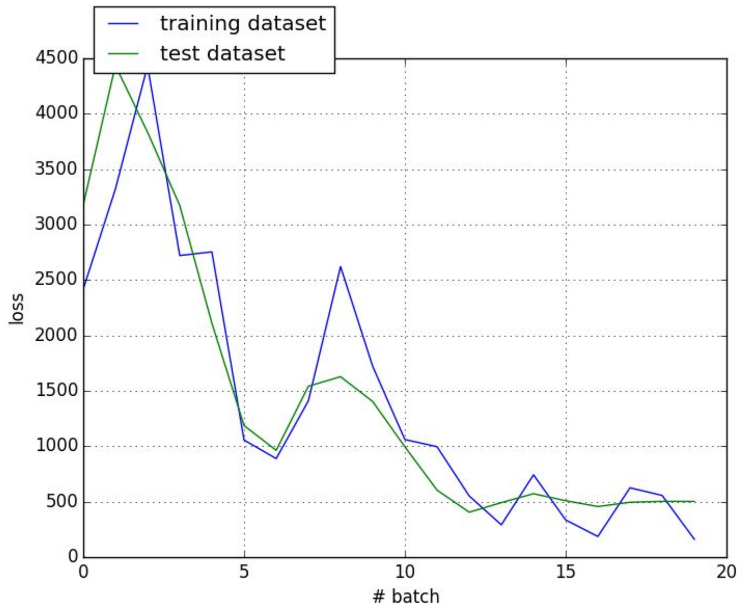


Figure 7: Loss sad (non)Happy angry

In figure 7 we can see the loss which reduces more slowly compared to figure 3, this is partly due to the smaller testing size. In the beginning it even goes up. Another factor here is that we have an extra emotion to analyse, and that the we observed that the sad and angry pictures included some overlapping.

We also compared the accuracy impact of grayscale and RGB images as data source using the script TensorFlowModelClassifySadOrHappyComparisonRGBvsGray.py. Figure 8 shows the accuracy for the testing set. We'll see that there is not much difference between RGB and grayscale. The grayscale images only converge 30 batches earlier than the RGB images.

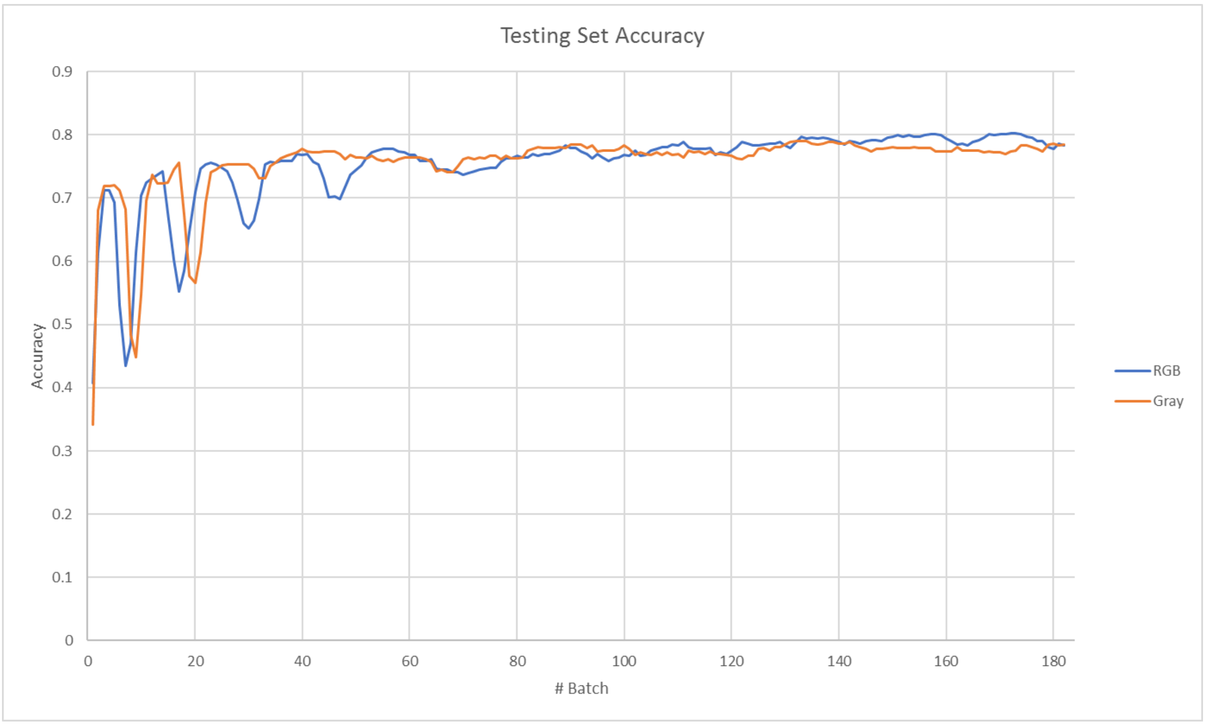


Figure 8: Happy/non-happy testing set accuracy for RGB and grayscale images

Problems encountered

The project didn't work in Windows because PIL library was creating issues extracting information from the images.
Initially we faced some issues creating the binary file for all the images but we were able to generate it later, but still it is unclear how to use the generated binary files.
The hops.site was down and we couldn't get our data on another location like IBM Data center, finally we used a server provided by SICS which we had for another project.
Our computer running with 8GB crashed because the memory overloaded when we used more than 400 images in an batch.
The spearman brown test, mentioned in the initial paper, is a test that is used for reliability but it couldn't be performed because it is not very well explained how to use it. We contacted the authors for guidance but did not get any information on its usability.

Problems faced on Hopsworks

The uploading of our Dataset containing 300 MB of images took 2.5 hours to complete on Hopsworks.
Then we ran into problems in running Hopsworks. The biggest problem was starting the interpreters. If the Python interpreter couldn't be turned on our project therefore couldn't be run on the hopsworks. One time this worked the other times it didn't. Most of the time it didn't so we weren't able to run any evaluations of the dataset. Another problem here was with the libraries that were imported on the hopsworks we ran on IP: t 109.225.89.154 . There we had a problem with the global variable error. This could be solved with replacing it by:

init = tf.initialize_all_variables. But it is only implemented in the latest Tensorflow version (0.12), which was not installed.

Conclusion

By analysing the non happy and happy images we obtained an accuracy of around 75%. We run a test on a small dataset we could obtain with sad, happy, non-happy and angry images. Here we found that the accuracy was only around 50%. The larger the dataset, the lesser the cross entropy. We also evaluated the impact of using RGB or grayscale images for input. Both offered roughly the same amount of accuracy, but the grayscale images converged earlier.

Appendix

Functions and the description

TensorFlowModelClassifySadOrHappy Script

createDataSets

"""Createts the training and test datasets from the images in smilePath and nonSmilePath.
Params: The path the the smiling images, the path to the non smiling images, the size of the dataSet we want to use, and the split value for the testing and training set. The split is set based on the testing split size in percent.
If a testing split of 20 is chosen 20% are going to be test data and 80% training data.
Returns: the testing and training labels, the trainingSet and testingSet
"""

TransformImage

"""Transforming an image into a numpy array
params: The image file
Returns the transformed image as numpy array in the dimension (width, length, color)"""

TensorFlowModel

"""The actual tensorflow Model
Here we initialise first all the variables, such as the numpy images, the labels, the learning rate, the weights, the bayes.
We then initialize a convolutional layer and initialize ReLU and max pooling
Then we reshape it for the processing
We perform a dropout to make the analysis perform faster and prevent overfitting
The model will then be trained in batches.
Each batch has a TrainingSet and a testingSet. An iteration will be performed for each training Image
The training session uses the dropout and will be trained, the testing session is not.
After all the batches are finished there will be a train result for the accuracy and the loss.
There is also a testing result for the accuracy and the loss
Params: The trainingSet, trainingLabels, the TestingLabels and the TestingSet, The batch size the model will be trained in
Returns: the training accuracy, the training loss, the testing accuracy, the testing loss
"""

plotResults
"""
Plot and visualise the accuracy and loss
accuracy training vs. testing dataset
Params: the training accuracy, testing accuracy, training loss, testing loss
Returns the plot for the accuracy and the loss as an image
"""
The main function:
"""
In the main function we initialize the datasetSize, the testing split and the batch size
The DataSetSize says how much images from each set we want to use, for example from happy and non happy pictures
The batchsize defines the size used for training
When this is initialized we create the Datasets, here we obtain the training set and the testing set
We run our tensorflow model
We then visualize the outcome of our tensorflow model, by plotting the result
Returns: The plots created in the plotResults function.
"""
GenerateLabeledData and GenerateLabeledDataMoreFrames
In this file we import three packages:
```
import csv  # to read csv files
import cv2  # to generate images from the video
import os    # to get file paths
```

Below we explain the functionality of each functions we used. The code can be viewed in the github repository, where it is documented as well.
change_to_video_name:
The name of the csv and the video is the same, only the suffix is different. With this function we get the video name with the flv suffix.
params: name of csv file and the suffix (like flv)
generate_frame:
Generate Frame is used to generate the frame from a video. The frame is taken from a specific moment in time.
We save the image in the format: video-name_second_label.jpg
In this format we can see from which video the image was, at what moment it was taken, and what the label was, for example whether a person is happy or not.
Params used: the path to the video, the video name, the second on which the image should be taken, the label and the destination folder
check_happiness:
The check_hapiness function is used to check for which time a person is happy and store the time and happy label
The information to see whether a person is extracted from the csv file. If the identification of the smile a person has is above a certain threshold, the label and the time are returned. We store the labels in numbers, so we can easily use them for any calculation. For happiness we use label '5'
Params: the content of each row in the csv file
check_nonHapiness:

The check_nonHapiness function is used to see if a person is not happy.
A person is not happy, when there is no smile and the AU12 baseline is less than 20
For the nonHapiness we use the label 50
Params: the content of the row in the csv file
get_content:
""
The get_content function returns the content of the csv file on the specific row
Params, the header to append the data to, the row number for which the content is wanted.
return: time frames for each AU in video
""
get_header_au
""
Function to return the information for this row
param: The specific row with the information
return: The time, if a person is smiling, and all the AU gestures
""
process_video_happiness & process_video_non_happiness
For the processing of the video to return the happiness and the non happiness frames we have developed two functions. The difference is between the functions, that for the non happiness we check if the people are not happy (sad). For the happiness function we check if a person is happy.
""
we walk through the directory and read the csv file for every video
then for every row in the csv file we first get the header information, the time, the smile and the au labels
We then get the content for each row. For that content we then check for the emotion e.g. happy or sad. We then generate a frame for each of these emotions.
If the emotion continues for a longer time period e.g. 2 seconds then we can take multiple frames pictures.
There can be a maximum of 14 frames per second taken
This function calls the following functions:
get_header_au
get_content
check_happiness or check nonHappiness
generate_frame
change_to_video_name
params: the csv_path, the video path that needs to be given for generating the frame, the destination path for the pictures, the suffix for the video in this case flv.
""

main function:
""
the main function initiates the csv path, the video path, the suffix and the destination path
It calls the process_video_happiness or process_video_nonHapiness
""