End-to-end Deep Learning pipeline for Facial Expression Recognition

Thi Huyen Cao

University of Applied Science Hamburg
Berliner Tor 5, 20099 Hamburg, Germany
thihuyen.cao@haw-hamburg.de

Abstract. Deep Learning has recently achieved the state of the art of many challenging tasks including Facial Expression Recognition. This paper describes an end-to-end Deep Learning pipeline for classifying 6 basic emotions (anger, disgust, fear, happiness, sadness, surprise) on short videos. The pipeline contains concrete steps from preparing data, performing training to evaluating model with detailed description and suggestion for the implementation. Result from this experiment will be reported in the follow-up project.

Keywords: Facial Expression Recognition (FER) · Deep Learning · Face Detection · Face Tracking · Neural Networks

1 Introduction

Facial Expression Recognition (FER) is one of the most interesting and challenging researches in Computer Vision field. It refers to the task of understanding the facial expression and furthermore the emotion underneath. Facial expression contains most information about the expressed emotional state besides speech, context and body language. While it is quite natural for human being to perform and interact with facial expression, it is obviously difficult for machine. Identifying the emotion requires machine to find the necessary features, patterns and evidence from the data. [1] summarizes the historical development as well as the state of the art of FER with static image and dynamic image sequence, which is Deep Learning. The art of DL is to use deep neural networks, which are inspired from the human brain to learn the important information from raw data to solve particular task. As that, DL offers the opportunity to build an end-to-end training process.

This paper proposes an end-to-end pipeline for FER using Deep Learning. The implementation and evaluation of this pipeline will be done in the following project. The rest of the paper is organized as follows. Some main concepts involved in FER are introduced in section 2. Section 3 describes the experiment setup in detail including the decision of emotion approach, dataset, tools, infrastructure set up and the construction of the pipeline. Some conclusions are drawn in the last section. Furthermore, this section will give some overview of the future work in the follow-up project.

2 Concepts

2.1 Face Detection and Tracking

FER begins with identifying and localizing faces in order to reduce the redundant information in the surrounding area. According to Dr. Robert Frischholz [2], there are many approaches which have been used for Face Detection such as using typical skin color to find face segments, finding faces by motion, using Edge-Orientation Matching, Hausdorff Distance, using Cascade of classifier, using Histogram of Oriented Gradients (HOG) or Deep Learning.

One of the classical and well-known face detector is Haar Feature-based Cascade Classifiers or Haar Cascade, which was introduced by Paul Viola and Michael Jones in 2001 [3]. It is a a Machine Learning object detection algorithm in images and videos, which is mainly used for detecting face and body parts. The algorithm is trained with a lot of positive and negative images, for example images with and without a face. It uses a boosted algorithm called Adaboost to learn a range of simple or weak features and combine them weightedly to provide a strong classifier. Especially, it proposes the concept of Cascade of Classifier [3] to increase the detection performance as well as reduce rapidly the computation by identifying and skipping negative areas as soon as possible. Cascade of Classifier contains multiple stages, which use classifiers in a hierachy of increasing complexity. It means that early stages use a simple classifier which is built from less features. These stages cost very less computational power and could effectively remove a lot of negative sub-windows. Sub-windows which pass early stages will be processed in later stages with more complex classifiers. To summary, Haar Cascade is robust, computationally efficient and can be used easily with some lines of code from OpenCV [1]. There are a lot of good pre-trained networks documented in [4]. Besides, [5] [6] show how to train such detector with own data to meet the requirement of particular application.

In recent years, many detectors trained with Deep Learning have been exploited and showed remarkable results such as Multi-task Convolutional Neural Network (MTCNN), Tasks-Constrained Deep Convolutional Network (TCDCN), RetinaFace, along with others. Deep Learning has become the state of the art of Face Detection in term of high speed and accuracy. The MTCNN is the most popular one since it performs very well on a range of benchmark datasets and is capable of detecting facial landmarks in addition. It is first introduced in 2016 [7]. The network uses a cascade structure with three sub-networks: P-Net, R-Net and O-Net. Firstly, the image is re-scaled to a range of different sizes (also called the image pyramid). Then the first model (P-Net) does the coarse face detection producing proposal regions, which are fed to the second model (R-Net) for refinement. The last model (O-Net) does the facial landmarking. Non-Maximum Suppression (NMS) algorithm is used in between to reduce the number of overlapping boxes in certain regions. The network architecture is well described in the original paper and there are a lot of pre-trained networks as well as implementation available as open source [8]. Face detector could also work with a

[1] A popular open source library for Computer Vision

frame sequence, in which faces are detected in each frame separately. However, face shape (mouth, eye, cheek etc.) and orientation tends to change over time, which causes a lot of difficulties for face detector. The temporal factor plays a very important role in such frame sequences since the location of face in last frame is essential for finding face in the current frame. Tracker takes exactly advantage of this information and performs in general better than frame-to-frame detecting. [9] pointed out some clearly good reasons why tracker works more efficient than detector on videos:

1. Tracker is faster than detector: while detector always starts finding faces from scratch, tracker can search for face in the surrounding of last position and therefore reduce a lot of computational power.
2. Tracker is preciser than detector: detector fails quite often if the appearance of faces is not typical e.g. in cases of occlusion, too small eye, too big mouth, a good tracker on the other hand can handle those problems quite smoothly.
3. Tracker preserves identity: beside outputting the location of faces, tracker can also report their identities which are held consistently through the video.

The table below summarizes a list of common trackers with their pros and cons. A more detail on each tracker can be further found here [9].

Tracker	Pros	Cons
BOOSTING	None (a decade old)	unreliable failure report
MIL [2]	better than BOOSTING	unreliable failure report, handle occlusion poorly
KCF [3]	better, faster, and reports failure better than BOOSTING and MIL	handle occlusion poorly
TLD [4]	works well under occlusion, over scale changes	high rate of false positive [5]
MedianFlow	excellent failure report, works well when motion is predictable and no occlusion	fails under large of jump in motion
MOSSE [6]	robust to variations in lighting, scale, pose and non-rigid deformations, occlusion and works at higher frame rate 669 fps	-
CSRT	high accuracy	operates at lower frame rate 25 fps

[2] Multiple Instance Learning
[3] Kernelized Correlation Filters
[4] Tracking, learning and detection
[5] a negative sample is mistakenly classified as positive
[6] Minimum Ouput Sum of Squared Error

2.2 Artificial Neural Networks

The word "Deep" in Deep Learning refers to the depth of the used Neural Networks. Deep Neural Networks or in other word Deep Learning has been proven to be able to solve a lot of challenging problems with high complexity such as image recognition, voice recognition, video classification, anomalies prediction and so on. It is believed that Neural Networks are inspired by biological neural system which controls the functionality of the brain. In core, it contains a specific set of algorithms for function approximation in order to map a number of input examples to a number of output classes. It is done by constructing a network containing layers and neurons which hold the useful information for the task. These layers are called data representation and information is passed through layers depending on whether the neurons are activated or not. This whole process is controlled by a system of parameters (weights and bias for each layers, configuration for cost function, optimizer function etc.) which need to be found so that the complex mapping between input and output works best. It is an iterative process and is shortly described in 3 steps: forwardpropagation, cost calculation and backpropagation. The parameters are chosen randomly or due to developer's experience at the beginning. Input value is calculated through layers and a prediction is made at output layer, which is then compared with the ground truth using a specific cost function. Depending on the calculated cost, the parameters will be adapted in the correct direction. It is repeatedly until a suitable set of parameters is found which meets the requirement of the task (accuracy, network size, precision or recall performance and so on). For the purpose of FER, only some topologies of Neural Networks with good performance and high probability of usage for this task are introduced in the following.

– **CNN, Residual Network and C3D**: Convolutional Neural Network (CNN) has gained a lot of success in the field of Computer Vision beginning with LeNet, which was proposed in 1998 to read zip-code, digit along with others, following by AlexNet, ZF-Net, GoogLeNet, VGG, ResNet, ResNeXt, SENet which won the ImageNet Image Large Scale Visual Recognition Challenge ILSVRC on millions of images over thousands of categories in the period of 2012-2017. CNN makes use of the mathematical operation convolution, which expresses the overlap of a function shifting over another. It uses in other word called filter kernel or convolutional kernel to extract feature from data by sliding the kernel through all possible locations in an image. Typical kernels by observing successful architecture are 3x3 and 5x5. On one hand, it captures very well the relationship among pixels in a neighborhood which is especially useful for image recognition. On the other hand, it reduces significantly number of parameters thank to the share weights among regions and as a result computational power. The typical layers of CNN are convolutional layer (CONV), pooling layer (POOL) and fully connected layer (FC). CONV is the core block of CNN and also called feature map since it serves as feature extraction of input. It is built by scanning the filter kernel through the image, shifting by s pixels. As a consequence, it continuously shrinks the width and height dimension of feature map in deeper layer,

which can be easily solved by padding. POOL is also called downsampling layer. It helps reduce the spatial size while keeping the depth. Is is believed that POOL helps compress the important information in a less volume and as that avoids overfitting and reduces computing cost. FC serves mostly as classification layer. It has been showed in a lot of works that the first layers of CNN extract general information such as edge, curve and the deep layers more complex feature such as eye, mouth. While 2-dimension CNN or C2D works well with image (grayscale or color), it does not perform well on video since it is not able to capture temporal feature. C3D was proposed in 2010 as a solution, also known as 3D ConvNets [10]. It uses a 3D kernel to extract not only feature in space but also motion information encoded in multiple adjacent frames, produces a 3D feature map instead of 2D as in original CNN. [11] investigated further the problems of training and optimizing deeper CNN and proposed a new CNN architecture called Residual Network. Residual Network contains basically CNN or feed forward layers with shortcut connections or also called residual connection. Shortcut connections are those skipping one or more layers. A firm argument why Residual Network can train way more layers than other network architecture is that layer with shortcuts can realize identity mapping and can not theoretically cause the training worse. Another explanation for the quality of Residual Network is that it behaves as an ensemble of multiple sub-networks [12]. DenseNet is similar to Residual Network with more connections and is proven to be capable of training even more layers. In each dense-block, there are connections among all input and output from each layer. To do FER, understanding what appear in the image is essential. It is exactly where CNN can contribute to this task.

– **RNN and LSTM**: While CNN performs well on visual data, Recurrent Neural Network (RNN) is known as a good architecture for processing time series (data with variable length) such as stock prices, weather data, logs, feedbacks and so on. RNN processes data timestep by timestep e.g. word by word in a sentence. It has an internal memory, which allows it to hold the information of the sequence it has seen so far. A RNN layer of n neurons can be unrolled in time as n feed forward layers with shared weights. The concept of backward training for RNN is called backpropagation through time (BPTT). The loss is calculated across all timesteps and summed up. Weights are then updated in a backward direction. This way of training can be very slow if the input is a very long sequence. In addition, too big or too small gradient can lead to vanishing or exploding gradients problem. Truncated BPTT came as a solution, which limits the number of timesteps used on forward- and backpropagation. Long Short Term Memory (LSTM) is a special type of RNN, which is designed to remember long-term dependencies as its default behavior. It was first proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [13]. It introduces the concept of gate control which manages the information flow. A LSTM block contains an input unit, 3 gate control (input, output and forget) and a memory cell state. Information can be added, updated and removed from cell state carefully by 3 gate control.

Gate control is built from a sigmoid layer with value in range from 0 to 1, which represents how much information should be passed through the gate. The art of LSTM is to learn how to learn, to learn how to pass, keep and forget the given information so that the useful information can be found at the end in order to solve the task. FER for video requires knowledge about the relationship among frames, about how the facial expression is changed over time. RNN can help at this point.

– **Hybrid**: Each network architecture has its own advantage and disadvantage. CNN extracts well visual features while RNN understands feature in time. Hybrid is an ensemble of multiple networks, which combines different architecture and as a result accumulate all their advantages. A well-known hydrid network is called Long-term Recurrent Convolutional Network (LRCNN), which stacks CNN layers as frontend and LSTM layers as backend. According to [14], LRCNN is "doubly deep": deep in space thank to CNN to extract spatial feature in each frame and deep in time due to LSTM to extract temporal feature among frames. Such network is a very potential candidate for task like FER.

3 Experiment Setup

3.1 Emotion Approach

There are 2 common emotion approaches: categorical and dimensional. Dimensional approach, also known as continuous emotion consider emotions as points in a continuous space and emphasizes that there are certain factors which exist in every single emotion. Therefore, each emotion can be represented by a point in n-factor coordinate system e.g. 2D of arousal and valence, 3D of pleasantness, attention and level of activation. It covers a huge range of emotion and as a result illustrates the complexity of human emotion. Categorical approach, also called discrete emotion on the other hand relies on the theory of universal facial behavior. One of the original source can be recalled to the book "The Expression of the Emotions in Man and Animals" by C. Darwin in 1872. A further experiment by Ekman and Friesen [15] determines that there are indeed 6 basic emotion which are anger, disgust, fear, happiness, sadness, surprise. Many other models of more or less emotions were created in the later years. There are continuous discussions around these 2 approaches. However, categorical approach is often chosen over dimensional with firm arguments of complexity, application context and goal. The complexity of dimensional approach makes it very hard to create qualitative ground truth. Annotators from crowd-sourcing services like Amazon Mechanical Turk are indeed unreliable for this task because of the lack of knowledge on the general concept. The landscape of application for this approach is also too less discussed so far. Therefore, the author believes that categorical approach is the right choice for the current state of FER. A classification task of 6 basic emotions mentioned by Ekman and Friesen will be implemented in this project.

3.2 Dataset

There are a lot of datasets in laboratory and in the wild available for public and research, both static image and video. Emotion is expressed naturally by the change of facial expression in time. For this reason, even though image of emotion with high intensity can be a good resource, the author decided to go with a video dataset. University Binghamton constructs a range of datasets for analyzing facial expressions and emotions, especially in three dimensional space or with multimodal sensing and releases them for public [16]. For this experiment, BU-4DFE is used. It is a high-resolution 3D dynamic facial expression database, recording on 101 objects with a variety of ethnic/racial ancestries, including Asian, Black, Hispanic/Latino and White. Each object performs 6 basic emotions (anger, disgust, happiness, fear, sadness, and surprise) with the instruction from professional psychologist. A detail on the process of creating the dataset can be read further in [17].

3.3 Pipeline

The standard algorithmic pipeline for FER consists of 4 major steps as showed in Figure 1. The trained and validated model is the result of this pipeline. This pipeline focuses strongly on building a model and will ignore detail on how the model is optimized and deployed in real application or re-trained at run time. Since Neural Network is capable of learning feature from raw data, the step of extracting feature can be skipped.

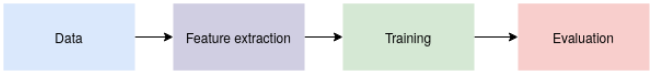

Fig. 1. Standard pipeline for FER

Step 1 is called Data or "procure" data. This step is about preparing data for step Training which includes choosing available datasets or creating a new one by collecting data from different resources or augmenting data to multiple the available data, annotation, clean-up mismatched data, preprocessing, storing data in disk or cloud. Since no feature engineering is neccessary, the next step called Training or in other word feature learning is about building the model which meets the requirement of particular task using the available resources (computing, human resources, budget). It contains decision about network architecture, platform for training (local or cloud), libraries (TF-Keras, Pytorch, etc.) and the process of hyperparameter optimization. This step is empirical and requires a good understanding of training techniques such as regularization, normalization, transfer learning, cost function, optimizer and so on in order to tune the parameters against overfitting and underfitting. In addition, a good evaluation matrix during training will help visualize the current situation clearly and

helps developer make good decision at modifying hyperparameters corresponding. Last step is about evaluating the trained model on a test set and detailed analysis about model performance and error. Figure 2 shows the pipeline for FER derived from the one mention above with an overview of subtasks and used tools. In the following, each step will be reviewed in detail.

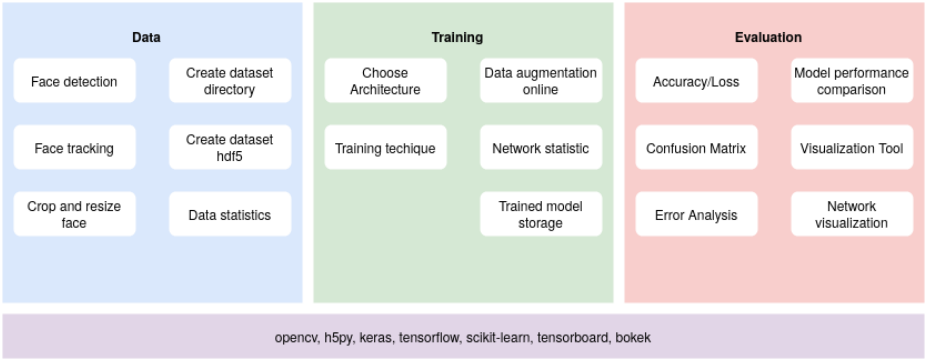

Fig. 2. Implemented pipeline for FER

1. Data: This part is mostly about preprocessing because an available dataset is used. First step is to detect face in the video. As the dataset is recorded in a setup environment, it is easy to localize face. For this reason, Haar Cascade is used as face detector because of its good performance at frontal face detection and efficient computation. Face is then tracked through the video by a tracker, which will be chosen after the method "trial and error". Different trackers should be tested on a number of examples and evaluated on accuracy and processing time. Detected face is then resized to a reasonable shape. The output will be stored in a directory after this structure /person/[videos]. Data will then be divided into train, validation and test set. The split into 3 datasets ensures that model can be evaluated on an unseen dataset during training (validation set) and the final model on a complete new set (test set). Preprocessed data can be written to hdf5 file [7] and is ready for training. Moreover, some statistics can be run through the whole dataset to gain some insights e.g. average video length

2. Training: the very first step at training is choosing network architecture. One thing to keep in mind is the characteristic of FER which requires understanding of both visual appearance (feature in space) and temporal dependency among frames (feature in time). [1] summarizes a range of proposals from different papers, which results in 3 main approaches: frame aggregation, expression intensity and spatio-temporal network. Among those 3, deep spatio-temporal networks are widely deployed on evaluated benchmarks and have proven to im-

[7] HDF5 (Hierarchical Data Format v5) is an open-source technology to store and manage extremely large and complex data collections

prove the performance further in comparison to the others. Table 3 shows again all discussed method for FER on dynamic data in regards to the capability of representing spatial and temporal information, the requirement on training data size and frame length (variable or fixed), the computational efficiency and the performance [1]. It is clearly to see that Cascaded Network (CN) can extract both

Network type		data	spatial	temporal	frame length	accuracy	efficiency
Frame aggregation		low	good	no	depends	fair	high
Expression intensity		fair	good	low	fixed	fair	varies
Spatio-temporal network	RNN	low	low	good	variable	low	fair
	C3D	high	good	fair	fixed	low	fair
	$\mathcal{FLT}$	fair	fair	fair	fixed	low	high
	$\mathcal{CN}$	high	good	good	variable	good	fair
	$\mathcal{NE}$	low	good	good	fixed	good	low

Fig. 3. Comparison of different types of methods for dynamic image sequences in terms of data size requirement, representability of spatial and temporal information, requirement on frame length, performance, and computational efficiency. FLT = Facial Landmark Trajectory; CN = Cascaded Network; NE = Network Ensemble. [1]

spatial and temporal features and can work on sequence with variable length. Despite the requirement of large datasets and fair computational efficiency, it is a good starting point for this experiment though. Cascaded Networks combines multiple networks, for FER mostly CNN with RNN. A good candidate is the well-known hybrid network LRCN mentioned in Section 2.2, which accumulates the powerful capability of extracting visual data from CNN and the strength of RNN e.g. LSTM for variable-length inputs and outputs (see Figure 4). It will be used as base model to start training. The concrete architecture will be reviewed in the follow-up project. The training process is mainly to find the set of parameter which generalizes the mapping between input and output best. It is complex in term of huge parameters to tune including number of layers, number of neurons in each layer, type of layer, learning rate, optimizer, weight initialization, activation function for each layer, amount of regularization and so on. Unfortunately, there is no guideline which guarantees 100% good result. Gridsearch is a method to test all combinations from a set of parameters. It can help as an orientation for picking up parameters. Ultimately, developer has to understand the current model performance. 2 useful concepts are Overfitting and Underfitting [21]. Underfitting is the case that model does not learn well so far or misses some important features, which leads to poor performance in both training and validation set. It implies that the model does not have the capacity or time to capture enough features. Treatment for this situation is training longer or using bigger and more complex network. A more common situation is Overfitting where the training seems to be fine, but the validation on the contrary. It is a symbol that the model tends to learn by heart the mapping and not really

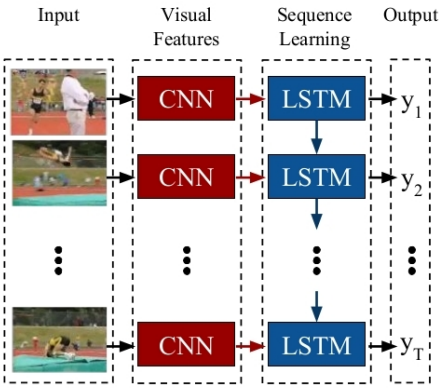

Fig. 4. LRCN Model [14]

learn to generalize, which leads to very poor result when validating on unseen data. This unwanted circumstance can be solved by more data (data augmentation), regularization (L1, L2, Dropout), new architecture or early stopping. The evaluation of which training techniques are working will also be covered in the next project. As the dataset is quite small, the danger of Overfitting is quite big.

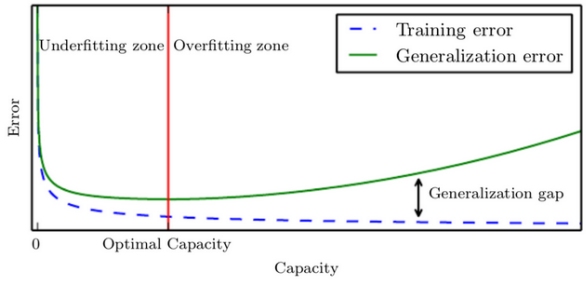

Fig. 5. Overfitting and Underfitting under the error view [21]

Collecting more similar data might be difficult. Data Augmentation will be very helpful here. The art of Data Augmentation is to invent more data by transform the current data using a variety of methods. Thus, the process needs to be done carefully so that no important part of face is cut, cropped or moved to a weird angle. Whether augmenting data should be done offline or online depends on the capacity of the training infrastructure. Finally, the parameters from trained model will be stored respectfully in hdf5 format. Some network statistic will be done as well such as visualization the architecture, summary of number of parameter in each layer to have a detailed view on the chosen network.

3. Evaluation: accuracy is the most important evaluation matrix.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ data} \tag{1}$$

A high accuracy represents in general a good classifier. Nevertheless, there are still use cases where it is not for the reason of unbalanced performance (only good in some particular classes). It's necessary to do further analysis. Confusion matrix (see Figure 6) is a common tool to illustrate all classes and their predictions. The diagonal elements represent the number of correctly predicted examples while others are those mislabeled. In case of class imbalance, normalization the Confusion Matrix with respect to the number of data in each class will provide a better view of model performance. Confusion Matrix is a great tool to gain more insight from the result. Class "surprise" and "fear" have a very high potential of being mislabeled because of their similar facial expression. And if it's really the case, it's easy to find out in the Confusion Matrix.

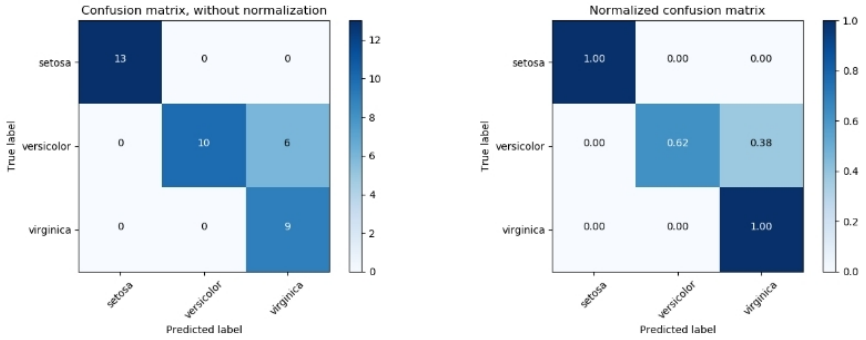

Fig. 6. Confusion Matrix [18]

Another tool is Error Analysis, which is a manual job to take a closer look at each wrongly predicted example and find out the reason why it is mislabeled. Error Analysis is time-consuming but returns in general very precious information. Furthermore, a comparison among different network architecture will be done. Optional is the visualization of how the network learn and where the network focus on by heat map, saliency map and so on. Also optional will be the development of a sub-application to view the result in a 6-dimension coordinate based on the probability predicted at output layer.

3.4 Tools

– **OpenCV** is an open source library written in C/C++ for Computer Vision. Its main goal is computationally efficient image/video processing. It contains interfaces in C++, Python, Java and supports multiple OS. OpenCV is mostly used for the preprocessing step.

- **Keras and Tensorflow**: Tensorflow is an open source library for Machine Learning application. Basically it provides efficient numerical computation through a flow of operations with tensors (data unit of n-dimensions). Tensorflow can be deployed in multiple platforms including CPU, GPU or TPU because of its flexible architecture. Keras is a high level API on top of Tensorflow with a focus on enabling fast experimentation. It contains implementation of many up-to-date Neural Network layers, optimizers, activation functions, pre-trained models, datasets, data preprocessing util functions etc. Keras and Tensorflow is used for training. Tensorboard is used for model evaluation during training.
- **Scikit-learn** is a well-known and well documented machine learning library. It contains a variety of supervised and unsupervised learning algorithms, as well as data statistic tool and evaluation. Scikit-learn is used for model evaluation.

3.5 Infrastructure Setup

Figure 7 visualizes the infrastructure setup for training. The training is done in the renderfarm provided by CSTI [19]. Each renderfarm is configured with 2 Intel Xeon E52697V4 2.3 Ghz with 36 threads, 396 GB RAM and 10 NVIDIA Quadro P6000 with 24GB RAM [20], which is dedicated for machine learning computation intensive task like FER. A detailed analysis on hardware of the renderfarms and their performance can be further read in [20]. The training process is containerized in 2 docker containers: one for training (Model trainer) and one for tracking result (Model tracker). Both containers have access to a pre-defined volume containing the dataset, source code, a pre-defined folder for writing logs and another for storing model parameters. The result from Model tracker will be mapped to local machine through ssh.

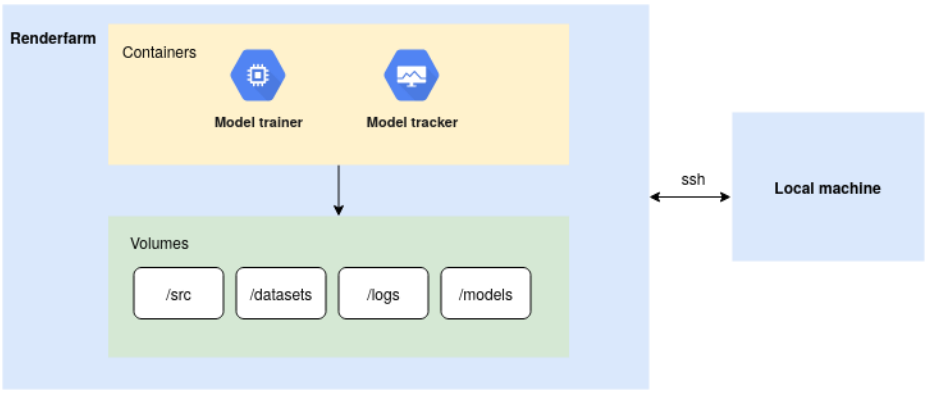

Fig. 7. Infrastructure pipeline

4 Conclusion & Future Work

This paper describes an experiment for the task classification of 6 basic emotion on videos. The chosen dataset is BU-4DFE from university Binghamton, which contains 101 objects x 6 videos (one for each emotion) = 606 videos. It is recorded in laboratory with professional set up. An detailed pipeline from preparing data to training to evaluation is described as a guideline for further work. The network architecture LRCN will be used at starting point with firm argument of its capability to extract spatio-temporal features. Training will be executed in renderfarm provided by CSTI with huge computational power which is especially designed for such computing-intensive tasks like FER. The follow-up project will implement and evaluate this pipeline including the following points

- Report on the implementation of each steps described in the pipeline
- Report on the hyperparameter tuning during training: which techniques works well and which not?
- Report on the chosen network architecture and alternatives if using
- Report on the performance of final trained models
- Detailed analysis on the classification result

An open point is the danger of small dataset chosen for this task. Data Augmentation and a variety of training techniques clearly can help at some level. Thus, ultimately Deep Learning requires enough data to learn. At some point, collecting more data might be a compulsory choice. If so, it's important to pay attention when accumulating data from different sources.

References

1. Authors, Shan Li and Weihong Deng : Article title. Deep Facial Expression Recognition: A Survey (2018) http://arxiv.org/abs/1804.08348
2. Face Detection and Recognition Homepage by Dr. Robert Frischholz, https://facedetection.com/algorithms/. Last accessed 28 Nov 2020
3. Authors, Paul Viola and Michael Jones: Article title. Rapid Object Detection using a Boosted Cascade of Simple Features (2001)
4. OpenCV Github Page, https://github.com/opencv/opencv/tree/master/data/haarcascades. Last accessed 28 Nov 2020
5. OpenCV Cascade Classifier Training, https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html. Last accessed 28 Nov 2020
6. MathWorks Cascade Classifier Training, https://de.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html. Last accessed 28 Nov 2020
7. Authors, Kaipeng Zhang and Zhanpeng Zhang and Zhifeng Li and Yu Qiao: Article title. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks (2016) https://arxiv.org/pdf/1604.02878.pdf
8. Github Page MTCNN, https://github.com/kpzhang93/MTCNN_face_detection_alignment. Last accessed 28 Nov 2020
9. Authors, Satya Mallick, Object Tracking using OpenCV (C++/Python) https://github.com/kpzhang93/MTCNN_face_detection_alignment. Last accessed 28 Nov 2020

10. Authors, S. Ji and W. Xu and M. Yang and K. Yu: Article title. 3D Convolutional Neural Networks for Human Action Recognition In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 1, pp. 221-231, Jan. 2013, doi: 10.1109/TPAMI.2012.59.
11. Authors, Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun: Article title. Deep Residual Learning for Image Recognition https://arxiv.org/abs/1512.03385
12. Authors, Prof. Dr.-Ing. Andreas Meisel. HAW Deep Learning Project.
13. Authors, Sepp Hochreiter and Jrger Schmidhuber: Article title. Long Short-Term Memory (1997)
14. Authors Jeff Donahue: Article title. Long-term Recurrent Convolutional Networks for Visual Recognition and Description (Nov. 17, 2014). http://arxiv.org/abs/1411.4389v4
15. Authors, P. Ekman and W. V. Friesen: Article title. Constants across cultures in the face and emotion, Journal of personality and social psychology , vol. 17, no. 2, pp. 124129 (1971)
16. University Binghamton: Analyzing Facial Expressions and Emotions in Three Dimensional Space with Multimodal Sensing, http://www.cs.binghamton.edu/~lijun/Research/3DFE/3DFE_Analysis.html. Last accessed 28 Nov 2020
17. Authors, L. Yin and X. Chen and Y. Sun and T. Worm and and M. Reale: Artitcle title. A High-Resolution 3D Dynamic Facial Expression Database, In: The 8th International Conference on Automatic Face and Gesture Recognition (2008), 17-19 September 2008 (Tracking Number: 66). IEEE Computer Society TC PAMI. Amsterdam, The Netherlands. http://www.cs.binghamton.edu/~lijun/Research/3DFE/Yin_FGR08_Paper66.pdf
18. Skicit Learn https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py Last accessed 28 Nov 2020.
19. Homepage Creative Space for Technical Innovation (CSTI), https://csti.haw-hamburg.de/. Last accessed 28 Nov 2020
20. Authors, Matthias Nitsche and Stephan Halbritter: Artitle title. Development of an End-to-End Deep Learning Pipeline (2019) https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2019-proj/nitsche-halbritter.pdf
21. Authors, Dr. Hanhe Lin https://www.mmsp.uni-konstanz.de/typo3temp/secure_downloads/97871/0/36f431c3dcb9d494a8cd5dd4c483d81883c3d2dc/dlp-lec6-upload.pdf (2018) Last accessed 28 Nov 2020