Emotion Recognition From Facial Expressions: A Preliminary Report

Tanja Jaschkowitz* Leah Kawka* Mahdi Mohammadi* Jiawen Wang*
{Tanja.Jaschkowitz,Leah.Kawka,Mahdi.Mohammadi,Jiawen.Wang}@campus.lmu.de

1. Introduction

Facial emotion recognition (FER) [5] is a topic of significant relevance and ongoing debate, not only in our daily life, but also in the fields of artificial intelligence and computer vision. In this short proposal, we aim to leverage several deep neural networks to analyze and interpret different human facial emotions.

The structure of this report is arranged as follows. In Section 2.1, we provide the datasets we used and the model architecture we implemented. The preliminary evaluation results of our models are given in Section 3. In Section 4 describes the optimization strategies we have already used and plan to investigate in the coming weeks. Finally, an overview of our time schedule for the entire final project is given in Figure 2. Our code and supplementary material are available at https://github.com/werywjw/SEP-CVDL.

2. Approach

2.1. Dataset Acquisition and Processing

To initiate the training, we acquired the databases FER+ [2], RAF-DB [6], CK+ [8], TFEID [3], and the videodatabase DISFA [9], from public institutions and GitHub repositories. Based on these databases we created a dataset by augmentation to increase variety, full details of augmentation is given in Section 4.1. In terms of the illustrating content of the used pictures, we exclusively used human faces representing 6 emotions. We generalized a folder structure annotating the labels 1 (surprise), 2 (fear), 3 (disgust), 4 (happiness), 5 (sadness), and 6 (anger). Besides the original format of images and videos, we set standards for extracting frames from the videos, resizing training pictures to 64x64 pixels, saved them as JPEGs.

The first test results in Figure 1 are aggregated from the database RAF-DB [6]. The images are converted to greyscale with three channels as our original convolutional neural network (CNN) is designed to work with three-channel inputs and add random augmentation. Emotions were assigned tags to each individual picture in a CSV file to facilitate further processing in the model.

The first test results in Figure 1 are aggregated from the database FER+. A custom dataset is a collection of data relating to a specific problem you're working on. In essence, a custom dataset can be comprised of almost anything. PyTorch includes many existing functions to load various custom datasets in the TorchVision, TorchText, TorchAudio and TorchRec domain libraries.

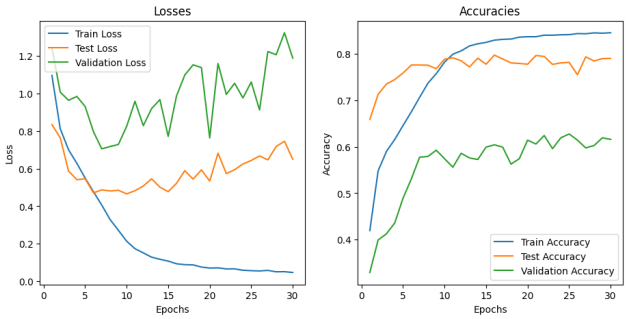

Figure 1. Empirical results in terms of the loss and accuracy on different training epochs

2.2. Model Architecture

We implemented from scratch an emotion classification model with four convolution layers at the very beginning. Following each convolutional layer, batch normalization is applied. This stabilizes learning by normalizing the input to each layer. Then three linear layers are applied to extract features to the final output. We also add a `dropout` layer to prevent overfitting. The activation function used after each layer is Rectified Linear Unit (ReLU), since it introduces the non-linearity into the model, allowing it to learn more complex patterns. In order to find the best hyperparameter configuration (see Table 2 for details) of the model, we utilize the parameter grid from sklearn [1].

3. Preliminary Results

For evaluation, we use the metric accuracy. As seen in Table 1, we report all the training, testing, and validation

[1] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ParameterGrid.html

Models	Accuracy (Train)	Accuracy (Test)	Accuracy (Vali)
CNN (Baseline)	66.3	75.2	52.6
CNN (SE)	74.3	79.9	59.6
CNN (SE-Aug)	84.6	79.5	62.8
CNN (SE+Residual)	71.5	78.9	56.4
ResNet18 [4]	76.8	79.8	60.3

Table 1. Accuracy (%) for different models in our experiments

Hyperparameter	Configuration
Learning rate	{0.1, 0.01, 0.001, 0.0001}
Batch size	{8, 16, 32, 64}
Dropout rate	{0.5}
Epoch	{10, 20, 30, 40}
Early stopping	{`True`, `False`}
Patience	{5}

Table 2. Explored hyperparameter space for our models

accuracy in % to improve the performance of our models. The loss function employed for all models is cross-entropy, which is typically for multi-class classification. The best results of the performance of the model with respect to loss and accuracy are depicted in Figure 1.

4. Optimization Strategies

We increase the depth of the network by adding some convolutional layers to learn more complex features. We also add the residual connections to help the training of deeper networks more efficiently, as they allow gradients to flow through the network more easily, improving the training for deep architectures. Moreover, we add squeeze and excitation (SE) blocks to apply channel-wise attention. In the coming weeks, we will focus on the following tasks Section 4.1 and Section 4.2. Further research is orientated on papers engaging similar investigations [7, 10, 11] [2].

4.1. Augmentation

In machine learning and artificial intelligence, augmentation stands as a transformative technique, empowering algorithms to learn from and adapt to a wider range of data. By introducing subtle modifications to existing data points, augmentation effectively expands the dataset, enabling models to generalize better and achieve enhanced performance. As models encounter slightly altered versions of familiar data, they are forced to make more nuanced and robust predictions. With this process, we aim to prevent overfitting, a common pitfall in machine learning. Additionally, we guide the training process to enhance the recognition and handling of real-world variations. During the project, we pursue various approaches. We are implementing different combinations of functions from the `pytorch.transforms` library and testing already established filters that have been developed, in other research contexts. We create various replications of existing photos by randomly altering different properties such as size, brightness, color channels, or perspectives.

[2] https://github.com/maelfabien/Multimodal-Emotion-Recognition

4.2. Explainable AI & Grad-CAM

Class Activation Mapping [12] is a visualization technique designed to highlight the regions of an image or video that contribute the most to the prediction of a specific class by a neural network. Besides proposing a method to visualize the discriminative regions of a classification-trained CNN, the authors also use this method to localize objects without providing the model with any bounding box annotations. The model just learns the classification task with class labels and is then able to localize the object of a specific class in an image.

Class Activation Mapping [12] is a visualization technique designed to highlight the regions of an image or video that contribute the most to the prediction of a specific class by a neural network. Class Activation Mapping (CAM) is a technique used in convolutional neural networks (CNNs) to visualize and understand the regions of an input image that contribute most to a particular class prediction. CAM generates a heatmap that highlights the important regions of the image for the model's decision. Model Architecture: CAM is typically applied to the final convolutional layer of a CNN, just before the fully connected layers. CAM Process: The final convolutional layer produces feature maps, and the GAP layer computes the average value of each feature map. The weights connecting the feature maps to the output class are obtained. The weighted combination of feature maps, representing the importance of each spatial location, is used to generate the CAM heatmap. Application: CAM helps interpret CNN decisions by providing visual cues about the regions that influenced the classification. It aids in understanding the model's behavior and can be useful for model debugging and improvement. The global average pooling (GAP) layer is used to obtain a spatial average of the feature maps. Besides proposing a method to visualize the discriminative regions of a classification-trained convolutional neural network (CNN), the authors also use this method to localize objects without providing the model with any bounding box annotations. The model just learns the classification task with class labels and is then able to localize the object of a specific class in an image.

CAM provides valuable insights into the decision-making process of deep learning models, like CNNs. We follow up Grad-CAM[1], introduced as a technique that is easier to implement with different architectures. This task

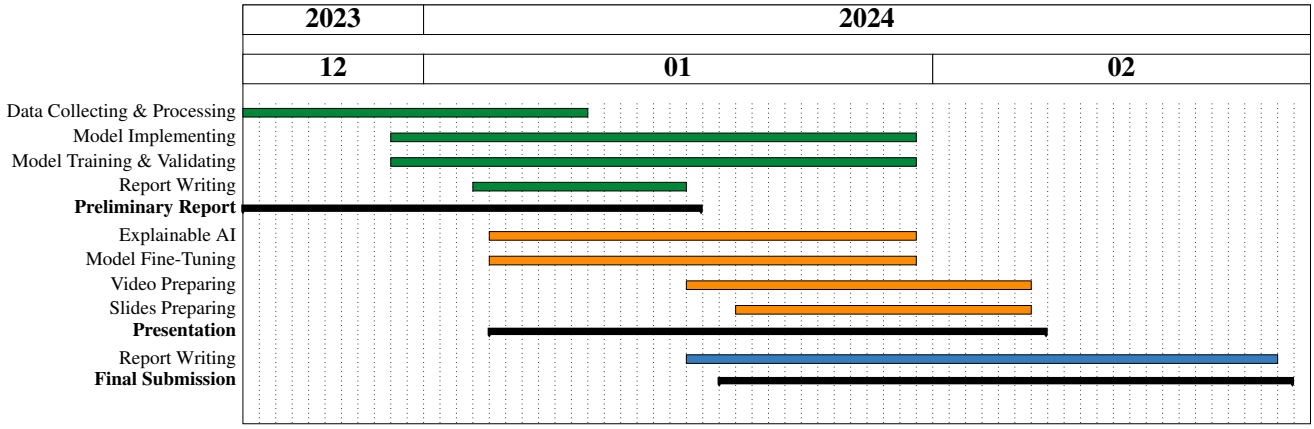

Figure 2. Overview of the schedule for the final project

will be implemented by using the libraries of Pytorch [3] and OpenCV [4].

4.3. Table of Classification scores

We also wrote a script that takes a folder path as input and iterates through the images inside a subfolder. The output is a CSV file representing the corresponding classification scores.

Author Contributions

Equal contributions are listed by alphabetical order of surnames. Every author did the literature research and contributed to the writing of the paper.

- **Tanja Jaschkowitz** implemented the model architecture, training and testing infrastructure, and CSV file aggregations.
- **Leah Kawka** collected the training data, prepared data processing, implemented augmentation, and ran the results. She also takes part in the Explainable AI and Grad-CAM.
- **Mahdi Mohammadi** implemented the augmentation, did the research searching, conclusion reasearching, Data preprocessing, CAM-Images inquiry
- **Jiawen Wang** implemented the model architecture, training and testing infrastructure, and optimization strategies. In the specific writing part, she also draw the figures and tables and improved this report from other team members.

Acknowledgements

We are deeply grateful to our advisors **Johannes Fischer** and **Ming Gui** for their helpful and valuable support during the entire semester. We also thank **Prof. Dr. Björn Ommer** for providing this interesting practical course.

[3] https://medium.com/@stepanulyanin/implementing-grad-cam-in-pytorch-ea0937c31e82

[4] https://opencv.org

References

[1] Vinícius Almeida. Grad-CAM in Pytorch: Use of Forward and Backward Hooks, 2023. 2

[2] Emad Barsoum, Cha Zhang, Cristian Canton-Ferrer, and Zhengyou Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction, ICMI 2016, Tokyo, Japan, November 12-16, 2016*, pages 279–283. ACM, 2016. 1

[3] L.F. Chen and Y.S. Yen. Taiwanese facial expression image database, 2007. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 2

[5] ByoungChul Ko. A brief review of facial emotion recognition based on visual information. *Sensors*, 18(2):401, 2018. 1

[6] Shan Li and Weihong Deng. Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition. *IEEE Transactions on Image Processing*, 28(1):356–370, 2019. 1

[7] Shan Li, Weihong Deng, and Junping Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2584–2593. IEEE Computer Society, 2017. 2

[8] Patrick Lucey, Jeffrey F. Cohn, Takeo Kanade, Jason M. Saragih, Zara Ambadar, and Iain A. Matthews. The extended cohn-kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2010, San Francisco, CA, USA, 13-18 June, 2010*, pages 94–101. IEEE Computer Society, 2010. 1

[9] Seyed Mohammad Mavadati, Mohammad H. Mahoor, Kevin Bartlett, Philip Trinh, and Jeffrey F. Cohn. DISFA: A spon-

taneous facial action intensity database. *IEEE Trans. Affect. Comput.*, 4(2):151–160, 2013. 1

[10] Monu Verma, Murari Mandal, M. Satish Kumar Reddy, Yashwanth Reddy Meedimale, and Santosh Kumar Vipparthi. Efficient neural architecture search for emotion recognition. *Expert Syst. Appl.*, 224:119957, 2023. 2

[11] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, pages 818–833. Springer, 2014. 2

[12] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2921–2929. IEEE Computer Society, 2016. 2