ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MSC IN BUSINESS AND ANALYTICS

Emotion Detection System

VACHRAM TSAKALIAN - p2822326

GIANNIS KASTRITIS - p2822333

STELIOS ZORMPAS – p2822312

ATHENS, SEPTEMBER 2024

Table of Contents

Chapter 1 Introduction

1.1 Project Overview

In the fast-changing technological world, the incorporation of Emotional Intelligence into software applications has been realized as very key. Our project, Medusa, takes this further to create a desktop application capable of detecting and understanding human emotions through a regular webcam. Accompanying this system is an intelligent machine learning model that identifies real-time emotional information, trained with vast datasets for facial expressions. It can capture and process visual information from the camera to infer happy, sad, angry, and surprised emotional states, thus offering users insight into emotional dynamics that may otherwise go unnoticed.

Medusa is a desktop application that enables easy access and installation on any existing computer. Much more importantly, it's fueled by a finely developed machine learning model trained with varied data to realize higher accuracy and reliability. The application is user-friendly in its interaction and allows seamless integration with other desktop applications and services. Such an approach would underline the potential of emotion detection technology in general, but also the powerful applications that it holds in enhancing user experience and interaction within domains.

The development of Medusa went through several major steps: facial expression data gathering and processing, training of the machine learning model, and desktop application implementation. We intend to prove that the concept of real-time emotion detection is applicable and to offer a solid tool which can be applied in many fields. It could all change everything in the way we relate to devices, making the system so powerful at work and even in personal life, due to the fact that it can analyze and give meaning to an expression of emotions.

1.2 Objectives and Vision

Medusa is developed to provide a very accurate, fast, and desktop-friendly emotion detection tool. We envision something more than functionality—a tool that enriches user interaction with real-time emotional feedback. This is particularly useful in contexts where knowing the emotional nuances may lead to enhanced engagement and responsiveness. Medusa seeks to set a new standard in emotion recognition technology with the newest approaches in machine learning, combining precision and ease of use for an elevated user experience.

We have grand, long-term plans for applications with Medusa in domains such as customer service, mental health, and interactive entertainment. For instance, if feelings of customers are known, then customer service will greatly increase in service quality by having a more empathetic and personalized response to them. It can also help therapeutic interventions in mental health by providing the emotional state of a patient to design more personalized treatment plans. There is huge potential behind Medusa; with that in mind, we work on its continuous development to respond to the demands that will increasingly be made of it by users from various sectors.

We view Medusa as a core technology, making it deployable in many kinds of systems and applications geared toward enhancing the emotional responsiveness of environments. We are fine-tuning our machine learning models and developing ways to consistently update them with user feedback in pursuit not only of bettering emotion detection but contributing to a greater vision: technologies that understand and respond to human emotions. The joining of these two personalities—innovation and user-centered design— commits us to the objective of making Medusa one of the leading tools in the domain of Emotional Intelligence.

1.3 Significance and Applications

This emotion-detection technology has a long way to go within very different industries, providing insights that can improve relations individually and organizationally. For example, in customer service, emotion detection can help support teams measure customer satisfaction levels and thereby modify their approaches for more efficient and personalized service. This can also be used in marketing and sales to know what feelings customers are at a point in time, and adapt campaigns and products to users' needs and preferences.

It can help track student interest and emotional well-being in schools, providing relevant information to educators for the personalized learning experience. Through the analysis of students' emotional responses, educators can detect if some are in need of additional support or adjustments within their learning environment, leading to a more inclusive and responsive educational approach. Emotion detection in therapeutic settings enables the construction of more effective treatments for mental health by providing real-time information on the current status of a patient's emotional state in order to give more accurate and timely support.

This has implications also in the broader sense on societal levels. By developing emotionally intelligent systems, Medusa will enable the production of empathetic, responsive technology that is much more attuned to human emotional needs. Such improvement at the user interaction level is not only useful but also paves the way for future innovations in the complete integration of emotional intelligence into different walks of life, making technology more intuitive and supportive in our personal and professional pursuit.

Chapter 2 Methodology

2.1 Data Collection

Data collection is a basic need when developing a machine learning model, more specifically when the tasks involve emotion detection. In this study, therefore, an impression of FER-2013, among the most known and widely applied datasets in the area of facial emotion categorization, will be in use. The FER-2013 database has a huge and varied collection of facial pictures, providing a benchmark standard for emotion detection systems. The dataset is built for a Kaggle competition and, more importantly, public. Thus, it is an excellent resource for research and development.

FER-2013 expresses facial expressions in grayscale, where each image is of dimension 48x48 pixels. The images were automatically registered so that the faces are roughly centered and roughly constant in size in the images. This preprocessing is very important since it reduces variations in the positioning and size of faces, which allows for more accurate emotion recognition. In doing face normalization, we ensure that the machine learning model is in a good position to learn only those distinct features linked to various emotions, rather than any inconsistency in image quality or placing of faces.

There are two major subsets in this dataset: the training set and the public test set. Examples in the training set total 28,709, hence with a rich dataset to train and tune the model for emotion detection. This is further followed by a public test set with large amounts of 3,589 examples, which should be used only to appraise the model for efficiency and generalization capability. This division helps in testing the accuracy of the model regarding the recognition and categorization of emotional expressions against the provided facial images.

2.2 Dataset Overview

The FER-2013 dataset is designed for emotion recognition tasks, detailing the structured RGB image features. All the images are 48×48 pixel-rate images of faces in grayscale format; hence, pixel values fully describe different shades of gray. Grayscale format eases the data by removing information that relates to color, which has no use in order to recognize an emotion, effectively focusing on facial expression features.

The core task in this dataset was classifying each facial image into one of the seven inbuilt emotional categories. These categories are 0 for Angry, 1 for Disgust, 2 for Fear, 3 for Happy, 4 for Sad, 5 for Surprise, and 6 for Neutral. These categories are clearly associated with various expression of emotional states, and thus the model would learn to differ between the basic emotional states from facial features.

An adequate structure and label of this dataset gives the best support for the development and evaluation of an affect detection system. The model is able to learn from the varied examples, considering the huge diversity present within the model, including a wide range of facial expressions and emotional states that help in learning and generalizing features. Secondly, a public test set is available to measure objectively how well the model performs when applied to unseen data and assures generalization to new and varying facial expressions. The likelihood of the public being made aware of the FER-2013 dataset and its structured nature will make it the most ideal to use in training and validation of our emotion-detection model, giving an appreciable contribution in the overall accuracy and effectiveness to an application like Medusa.

Figure 1. Training dataset sample.

2.3 Dataset Processing and Augmentation

The effective processing and augmentation of the dataset are crucial preparation steps for machine learning models, as the quality and variability in the data would play a significant role in the model's performance. In this regard, the FER-2013 dataset in the subsequent discussions has been subjected to various preprocessing and augmentation methods that could enhance the robustness and generalization capability of the emotion detection model.

First, raw grayscale images from the FER-2013 dataset were standardized in terms of input to make the data presented consistently. Each image was then reduced to the standard format used in this dataset, 48x48 pixels, and normalized to one common scale to better train the model. Normalization usually means scaling pixel values between 0 and 1; it has a great implication for stabilizing the training process and often improving convergence.

Preprocessing when using the application interface for processing input images involves several steps that align them to the format taken by images in the training dataset. The first step is to convert the images to grayscale since the images in FER-2013 are grayscale. Then, they are resized to 48x48 pixels, which is the size during model training. Finally, the pixel values of these images are normalized to a similar range of 0 to 1. Normalization of input data will scale it to the same range that the model was trained on, hence the model can process and classify new data against learned features effectively.

2.4 Model Selection and Training

For our next task of emotion detection from facial images, we mean to use Convolutional Neural Networks (CNNs) as the tool is already tried and tested in the image recognition domain. Each entry in our dataset will have a grayscale image, 48 pixels in that direction, and each picture will bear one of the seven distinct emotions. Our aim is to construct a model that can train on these images and with this knowledge accurate emotion classification can be secured.

In order to do this, we are going to examine different CNN architectures. Our idea is to design several different networks which would include up to 5 convolutional layers to catch the features in the images in both lower and higher levels of abstraction. Every convolutional layer will be followed by batch normalization, which will scale the outputs from the previous layer. The normalized inputs will make the next layers learn better. Furthermore, we will include max-pooling layers, which will reduce the dimensions of the feature maps of each CNN.

The aim of stabilizing the complexity of the models to avoid both overfitting and underfitting will be pursued through this process. The models that are trying to understand/learn some particular abstract concept are using machines that are equipped with dropout. The dropout will be randomly activating or deactivating a subset of neurons. With this kind of behavior, our models can adapt to new instances easily without getting the exact distributions of the training data. Of course, we will take care also to make the models complex enough to recognize the fine details of the images and thus to prevent any underfitting.

For the last phase of our network design, we will engage the fully connected layers to merge the gathered features and to categorize the images to one of the seven emotion categories. We will utilize a categorical cross-entropy loss function that is perfect for multi-class classification problems. In order to boost training speed, we will opt for the Adam optimizer which alters the learning rate while the training is in progress. While running the training, we will make use of validation data to evaluate the models' outputs and on that basis facilitate the training, ensuring that our best models are saved for later implementation.

Chapter 3 Experiments

3.1 Experimental Setup

This section describes the experimental setup used for emotion recognition with Convolutional Neural Networks (CNNs). The setup includes the data used, preprocessing steps, model architecture, training procedures, and the software and hardware environment.

The dataset used in this experiment consists of facial images categorized into seven emotional classes: angry, disgust, fear, happy, neutral, sad, and surprise. These images are stored in grayscale format with a resolution of 48x48 pixels. The dataset is organized into separate directories for training and testing. The training set comprises images used to train the model, while the testing set is reserved for evaluating model performance. The training set is further divided into training and validation subsets, with 80% of the data used for training and 20% for validation. The split is performed using stratified sampling to ensure that each emotional class is proportionally represented in both subsets.

To prepare the data for model training, several preprocessing steps were applied. All images were resized to 48x48 pixels, matching the input size expected by the CNN models. Pixel values were normalized to a range of [0, 1] by dividing by 255, which ensures that the model can train more efficiently by reducing the range of input values. The class labels were one-hot encoded using a LabelBinarizer, converting them into binary vectors that the model could use for training.

Three different CNN architectures were explored in this experiment. Three different CNN architectures were explored in this experiment. The first was based on the LeNet-5 architecture, a simpler and classic design with two convolutional layers followed by average pooling, effective for smaller datasets and less complex tasks. The second model was inspired by GoogLeNet, featuring a deeper network with multiple Inception modules, which are designed for complex feature extraction and capture diverse features at different scales. The third model was a custom CNN with six convolutional layers, each followed by batch normalization, max pooling, and dropout layers. This network culminates in fully connected dense layers, leading to a softmax output layer. All models were compiled using the Adam optimizer with a learning rate of 0.0001 and categorical cross-entropy loss, suitable for the multi-class classification task. All models were

compiled using the Adam optimizer with a learning rate of 0.0001 and categorical cross-entropy loss, which is suitable for the multi-class classification task.

The models were trained for 50 epochs with a batch size of 64. Early stopping was implemented to prevent overfitting, monitoring the validation loss with a patience of three epochs. The best-performing model weights were restored at the end of training. During training, the models were evaluated on the validation set at each epoch to monitor performance and ensure that they were generalizing well to unseen data.

The experimental setup was executed using TensorFlow and Keras for building and training the CNN models, Scikit-learn for data preprocessing, label encoding, and evaluation metrics, and Matplotlib and Seaborn for visualizing the confusion matrix and training history.

3.1.1 LeNet-5 Model

The model starts with a convolutional layer that applies six 5x5 filters to the input image, detecting basic features like edges and textures. This is followed by a ReLU activation function, which introduces non-linearity to capture complex patterns. Padding is used to maintain the input size throughout this layer.

An average pooling layer then reduces the spatial dimensions by averaging pixel values in a 2x2 window, which decreases computational load and retains essential features.

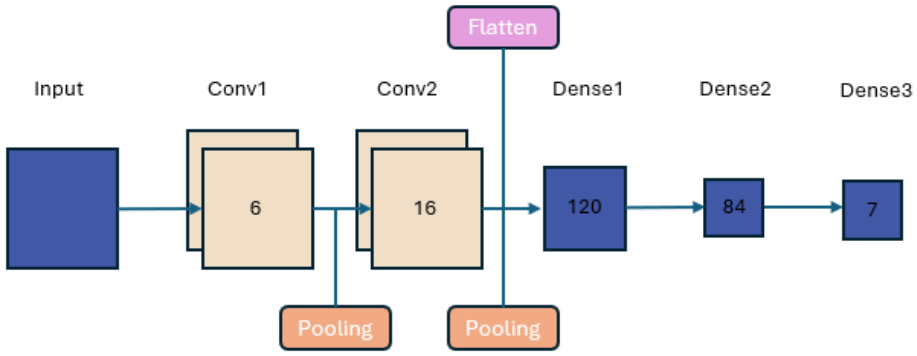

Figure 2. Model Architecture of LeNet-5

The next convolutional layer applies sixteen 5x5 filters, detecting more complex features based on the initial ones. This is also followed by a pooling layer that further reduces the spatial dimensions, abstracting the feature maps. The model transitions to fully connected layers, where the features are combined and processed to identify higher-level patterns. The first fully connected layer has 120 neurons, and the second has 84 neurons, each using ReLU activation. The final fully connected layer outputs probabilities for classification, with a number of neurons equal to the number of classes, using the softmax activation function to make categorical predictions. The model is compiled with categorical cross-entropy as the loss function and the Adam optimizer for efficient training, aiming to achieve high classification accuracy.

3.1.2 GoogleNet Model

First, the model is initialized with a convolutional layer that would work out the basic features from the input images. In it, 32 filters of size 3x3 are used with ReLU activation, followed by Batch Normalization for the stabilization and acceleration of training. Further feature extraction is done through another convolutional layer with 64 filters of size 3x3, using Batch Normalization.

Additional layers include max pooling and dropout. Max pooling reduces the feature maps' spatial dimensions by returning the maximum of all values within a 2x2 window; this helps to make the model more invariant to small translations. After that, dropout randomly sets a fraction of the input units to zero at each training forward pass to prevent overfitting.

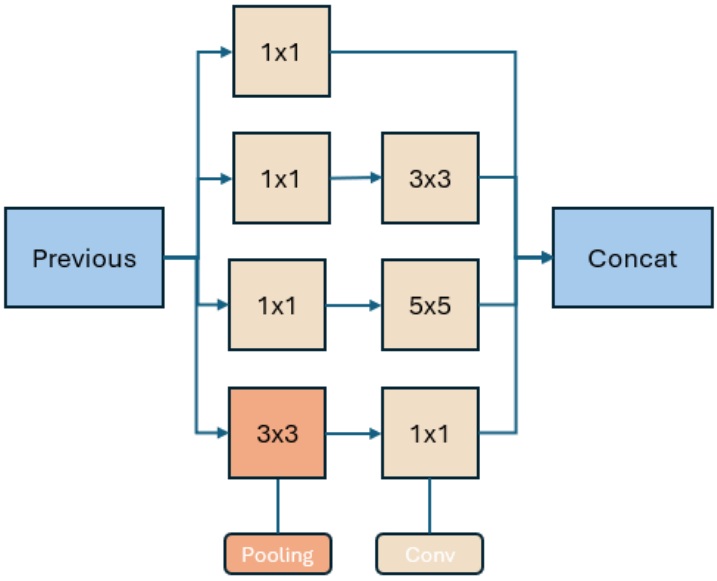

Figure 3. Inception Module of GoogleNet

This model contains more convolutional layers; it increases the number of filters progressively up to 128 and then up to 256. Therefore, it allows the model to learn features from inputted images that are more complicated and more abstract. Each of these layers is followed by Batch Normalization and pooling for stabilizing learning and reducing the risk of overfitting.

The model embeds an Inception module to enhance its feature extraction capability. This performs several convolutional operations in parallel with different filter sizes (1×1, 3×3, and 5×5) and a max pooling operation. These resultant outputs are further concatenated across the depth dimension so that the network captures a diverse set of features at several scales.

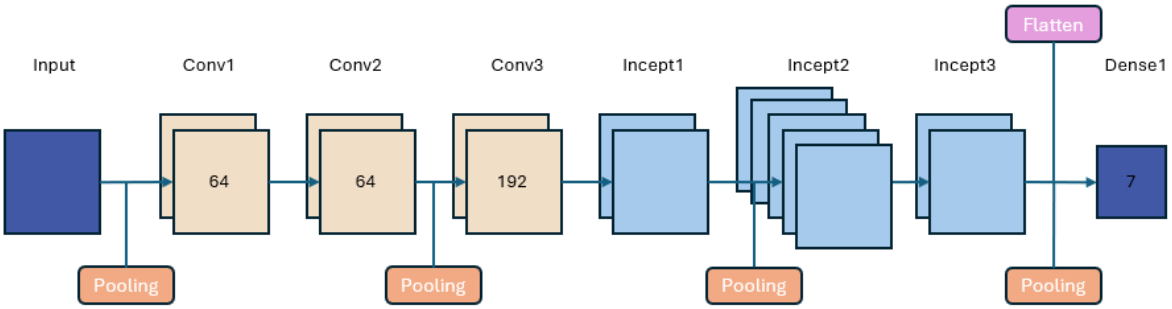

Figure 4. Model Architecture of GoogleNet

In convolutional layers, Inception module pass features that are then flattened into a one-dimensional vector. Next, a fully connected layer of 256 neurons with the ReLU activation is used to integrate features for the final prediction. Dropout is also applied at this layer to further avoid overfitting.

The last layer is made up of another dense layer with the number of neurons equal to the number of classes in the output, using the softmax activation function to get a probability score across each class.

3.1.3 Custom Model

The CNN architecture begins with the input layer, where it gets feed data from the input image. For grayscale and color images, their typical representations are a 2-D grid and a 3-D grid, respectively; each of the dimensions captures width, height, and depth.

First, after the input layer, Conv1 applies 32 different filters to the input image. The filters are small grids, say 3x3 or 5x5 in size, sliding over the whole image and producing 32 feature maps that detect edges and textures as low-level features of the image. Following that is the second convolutional layer, Conv2, which contains more filters-64 in total-that give 64 feature maps extracting higher-level features from the output of Conv1.

After Conv2, a pooling layer is applied based on the max-pooling mode to reduce the spatial dimensions of feature maps. The goal of this pooling operation was to introduce an efficiency in the network by reducing the computational load; hence, it produces a set of downsampled 64 feature maps.

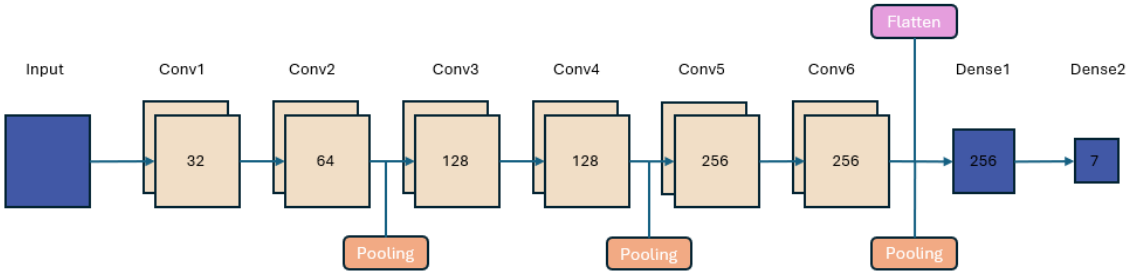

Figure 5. Model Architecture of CNN

It proceeds with the third convolutional layer, using 128 filters on the output to get 128 feature maps that detect higher complicated features. After that, it is followed by the next convolutional layer, Conv4, also with 128 filters, keeping the same feature complexity level intact. Further, another pooling layer is introduced that reduces the spatial dimensions of the 128 feature maps.

In the next level of abstraction, the network is deepened by Conv5, which applies 256 filters to produce 256 feature maps highlighting fine details in the image. This is followed by Conv6, also comprising 256 filters to further tune the feature extraction. The result after the sixth convolutional layer is flattened into a vector of values. This decreases the dimensionality from 256 feature maps into one single-dimensional vector, preparing the data for the fully connected layers ahead. Sometimes, there is a final pooling done here, in order to further reduce the dimensionality; in some architectures, this can often be excluded.

It is further fed into the first fully connected layer, Dense1, which is made of 256 neurons. Every neuron in this layer is connected to every neuron in the flattened vector, allowing the network to combine features that it extracts and make more complex decisions. Finally, the second fully-connected layer, Dense2, is composed of 7 neurons, representing the 7 possible output classes, which give probabilities of the input image belonging to each class.

The CNN architecture processes the input image through successive convolution and pooling layers, each progressively extracting and refining the features. These features then get fed to the fully connected layers at the end, for classifying the image into one of its predefined classes.

3.2 Results and Quantitative Analysis

This section presents the results and quantitative analysis of three distinct neural network models: GoogleNet, LeNet-5, and a custom model. Each model's performance is evaluated based on the evolution of accuracy and loss metrics during training and validation over several epochs. The following paragraphs delve into the detailed examination of these metrics, offering insights into the strengths and weaknesses of each model in terms of learning efficiency, generalization capability, and potential overfitting.

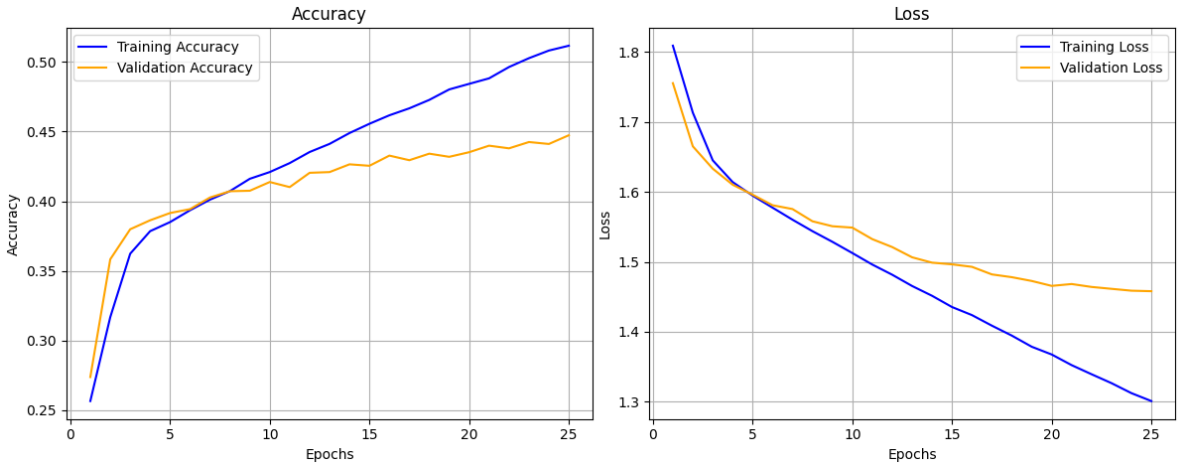

Figure 6. LeNet-5 Metrics

Starting with LeNet-5 model, the accuracy trends show a gradual improvement in both training and validation accuracy, with the training accuracy eventually reaching about 0.50. The validation accuracy follows a similar trajectory but plateaus earlier, leveling off around 0.40. This close similarity that there is between the two curves suggests that the performance of the LeNet-5 model generalizes reasonably well, even though the overall accuracy is modest. This stability is also mirrored in the loss curves of LeNet-5; both the training and validation losses move in a rapidly decreasing fashion. However, the validation loss plateaus at about 1.3, while the training loss drops below 1.0, indicating a slight overfitting tendency as the model keeps learning from what it sees during training.as the model continues to learn from the training data.

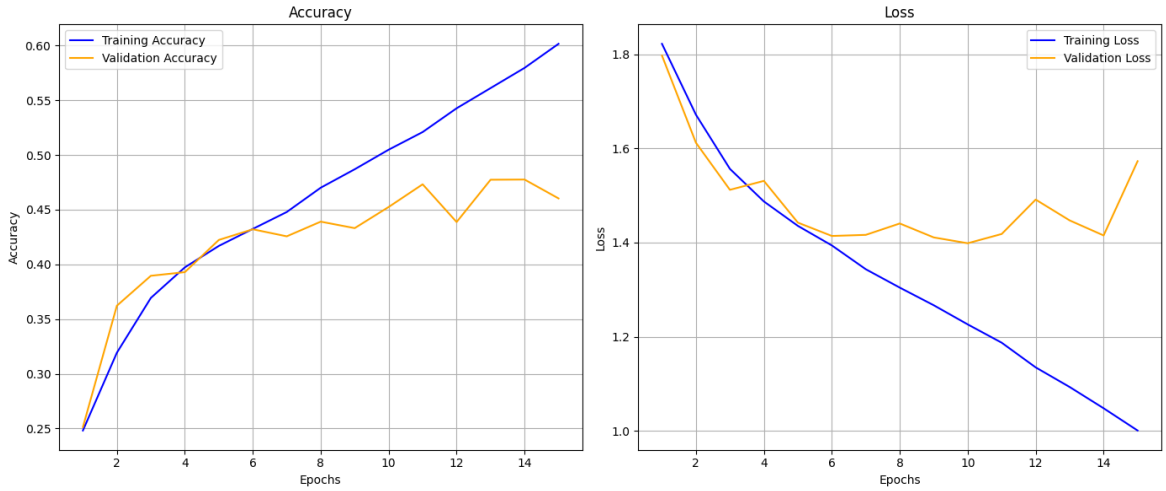

Figure 7. GoogleNet Metrics

About the GoogleNet model, the accuracy graphs clearly show an upward trend for both the training and validation accuracy as the number of epochs increases. The training accuracy increases linearly and reaches a point around 0.60. On the other hand, the validation accuracy improves together with the training accuracy at the beginning, but then it gets flattened around 0.45 to 0.50. This discrepancy between the training and validation accuracies indicates that, while the GoogleNet model is increasingly good at learning from the training data, it does not generalize that well to data it has not seen. This could be indicative of the beginning of overfitting. The training loss drops rapidly and constantly, while the validation loss, tends to jump up and down. This may hint at overfitting against the training data, and that generalization could be a problem.

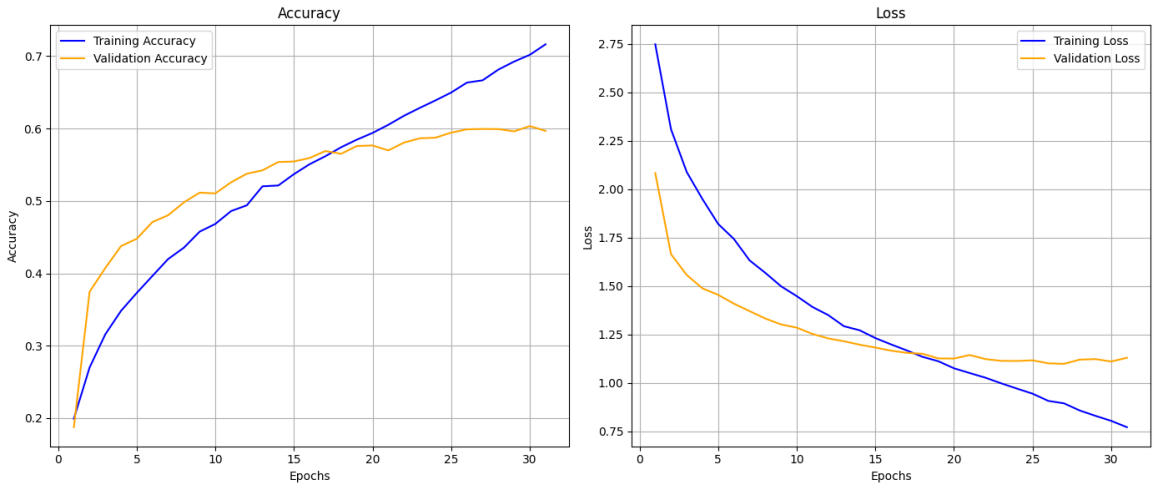

Figure 8. Medusa Metrics

Graphs for the custom model provide evidence that training and validation are doing really well. The accuracy of this model did great improvements during the first few epochs, where the training accuracy finally surpasses 0.70, while the validation accuracy also goes up impressively, although it starts to run flat at approximately 0.60. The proximity of the training accuracy curve to the validation accuracy curve reflects the fact that the customized model generalizes well. This means there is a perfect balance between learning and overfitting. The loss curves confirm this, showing continuous decrease in both training and validation losses. Whereas the loss on validation stabilizes at the end of training, its small fluctuation could signal that overfitting might be about to set in, though not as serious compared to what was observed from the GoogleNet model.

Comparing the three models, it is obvious that all three have some overfitting going on, with the GoogleNet model being the most obvious. Its validation accuracy plateaus while training accuracy rises well past it, showing a gap in the model's capability to generalize well. The custom model seems to yield the best overall results, offering a high accuracy with respect to its counterparts, while keeping a stable relationship between the metrics of training and validation. Meanwhile, the less complex LeNet-5 model has a more smooth and consistent training process but with low total accuracy compared to the other two models. It means that the custom model is quite effective in balancing the trade-off between complexity and performance; nevertheless, it has also hinted at probable overfitting evidenced by fluctuations in the validation loss in the later stages of training.

Overall, each of the models has strengths and weaknesses: The GoogleNet model suffers from generalization due to its architectural complexity, whereas the LeNet-5 model is stable and steadfast, with low accuracy. Custom performed very strongly, but overfitting may result from spending too long training.

3.3 Qualitative Evaluation and Error Analysis

According to the experiment conducted in the work, three models were considered for emotion recognition: the LeNet-5, GoogleNet, and a custom model called Medusa. Each of these models was evaluated on the basis of metrics related to their effectiveness, including training accuracy, training loss, validation accuracy, and validation loss. Among these, the Medusa model had the best performance, yielding 0.72 for training accuracy, 0.75 for training loss, 0.60 for validation accuracy, and 1.13 for validation loss. These scores depict that Medusa generalizes better than LeNet-5 and GoogleNet since it had higher accuracy and lower loss in both training and validation datasets.

	Training accuracy	Training loss	Validation accuracy	Validation loss
LeNet-5	0.51	1.30	0.45	1.45
GoogleNet	0.61	1.00	0.46	1.59
Medusa	0.72	0.75	0.6	1.13

Figure 9. Table of Metrics

Further to understand its performance and possibly the limitations with the Medusa model, we analyzed a confusion matrix generated from the predictions of this model on the test set. This confusion matrix gives a detailed breakdown of how well the model predicts each category of emotions and where it commonly makes errors.

The confusion matrix further shows that the performance of the Medusa model regarding the recognition of the 'happy' emotion is very good, at 1,442 correct classifications. Such high accuracy points out that the features for happiness are well captured by the model and contributes to a high recall for this class. On the other hand, this analysis has shown that there are large difficulties concerning the distinction of the emotion 'fear' from 'angry', 'neutral', and 'sad'. For example, 'fear' was misclassified as 'neutral' 236 times and as 'angry' a total of 136 times. It indicates that the feature set that Medusa uses for detecting fear may somewhat overlap with those from other negative emotions and needs better feature differentiation.

Only 49 instances are rightly detected for the emotion 'disgust'. It easily gets misclassified by the model for 'angry' and 'fear'. This probably has to do more with how disgust features are represented, not being saliently different from other negative emotions. This may be because of either a small

number of samples for 'disgust' during training or the lack of distinctive features which mark it different from emotions like anger and fear.

Meanwhile, though the model Medusa does a fairly reasonable job with the 'surprise' emotion, classifying 603 instances correctly, there is still quite a bit of confusion between 'surprise' and 'fear', as it misclassifies 79 'surprise' instances as 'fear'. The confusion could arise because of the similarity in facial expressions for surprise and fear, like wide eyes and an open mouth, which are perhaps interpreted similarly by the model.

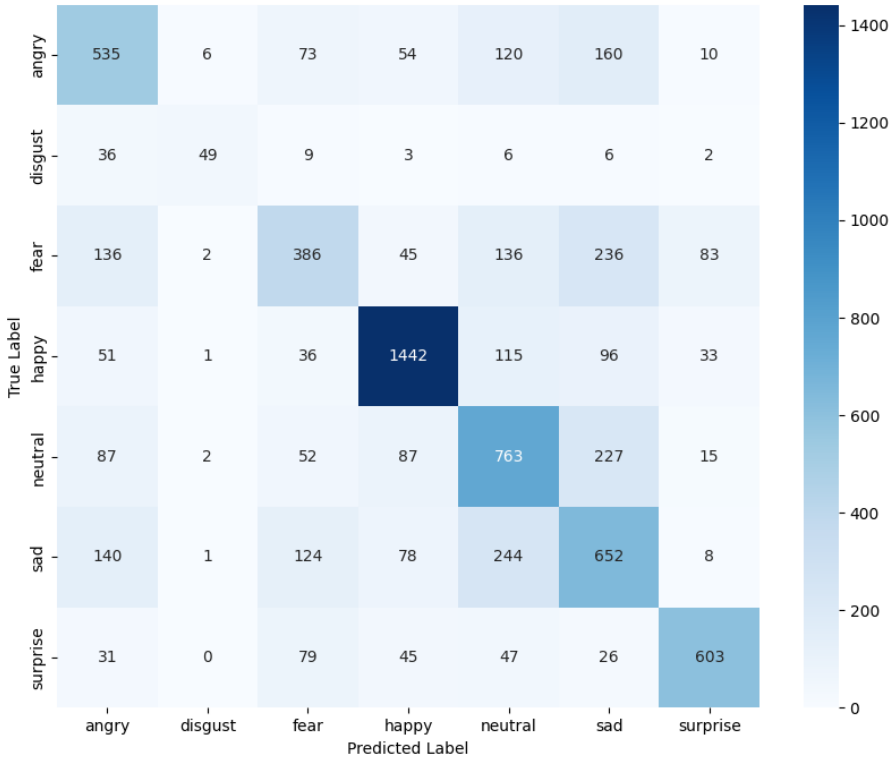

Figure 10. Confusion Matrix of Medusa Model

More interestingly, the frequent misclassification between 'angry' and 'neutral' can be spotted. The model mislabeled 120 'angry' cases as 'neutral' and vice-versa, 87 'neutral' cases as 'angry'. That could be because these two classes sometimes show subtle differences in expression, especially in situations when the expression of anger is not exactly intensive.

Medusa works quite well in general, especially on the recognition of the emotion 'happy', but it does poorly on those whose facial features are somewhat similar, such as 'fear' and 'neutral' or 'disgust' and 'angry'. While the model has learned to extract discriminative features for some

classes, its ability to capture the fine differences among similar emotions may need to be further improved. This may involve future work on more powerful features, including a greater variety of training data or leveraging state-of-the-art methodologies such as attention mechanisms that could allow the model to better discriminate between these similar emotional expressions. Qualitatively, this can be reflected by the confusion matrix, which allows the indication of future improvements and increases the chances of robustness and accuracy of the Medusa model in real-world applications.

Chapter 4 Conclusion

4.1 Summary of Findings

Therefore, against this background, the experimental setup and results in this work show that CNNs remain very effective for emotion recognition studies. It explores three different deep learning models based on CNN architectures: LeNet-5, GoogleNet, and a custom-designed model, which all add something different to the performance and capability insights of deep learning models with regard to emotion classification. The dataset consisted of grayscale facial images labeled over seven classes of emotions and was preprocessed by a sequence of standard preprocessing steps that made the images suitable for the CNN architectures. The results show large differences in generalization performance and emotion recognition accuracy, depending on the model complexity and applied regularization techniques.

Having in mind that this model had a simpler architecture and more classic, the performance of LeNet-5 on this task was good enough, considering it has fewer layers and fewer parameters compared with the other two models. It was outranked by the more complex models, GoogleNet and custom model, which were able to grasp from the inputted images more detailed features due to their deeper networks and advanced layers as Inception modules and multiple convolutional layers with batch normalization and dropout. Whereas the performance of the GoogleNet model proved especially capable, this should be noted because it is able to model a diverse set of features over multiple scales through its Inception modules. By contrast, the custom CNN model contained six convolutional layers in a balanced approach that combined a variety of regularization techniques to avoid overfitting, while still maintaining capacity for extracting complex features.

In order to tune the training process for all models, the authors have used the Adam optimizer with a learning rate of 0.0001 and categorical cross-entropy loss. Early stopping has been implemented to avoid overfitting of the model; this rigorous training process along with validation in real time is made sure of high accuracy on test data while generalizing well to unseen data. According to the chosen model performances, the best performed as per their validation, and upon deeper investigations using confusion matrices and classification reports, the strengths and improvements that had to be made by those became quite clear. In summary, though deeper and more complex models like GoogleNet tend to do better, there is a lot of value still lying in custom models.

4.2 Time Plan and Milestones

A structured time plan for the key milestones and deliverables that will be required throughout the project's lifecycle was developed, structured into phases concerning different aspects of the project in each phase: preparation of the initial data, development of the model, evaluation, and final integration.

The first phase involved the collection and necessary pre-processing steps for data. During these weeks, the dataset was organized, and preprocessing techniques such as resizing, normalization, and label encoding were applied. By the end of this phase, the data was fully prepared for model training, with both training and validation sets properly split and ready for use.

The next stage would involve the actual design and implementation of the three CNN architectures, which include LeNet-5, GoogLeNet, and a custom model developed based on certain experiments. In this stage, the models are designed and built. Fine-tuning of architectures is done based on experimentation and preprocessed data. Different regularization techniques and early stopping mechanisms were integrated into the CNN architectures to avoid the problem of overfitting and give good generalization on new data. This phase was completed when the training of all three models was done and the best model was selected out of them based on its performance on the validation accuracy.

In this phase, the trained models were thoroughly evaluated using the test set. This included generating confusion matrices, classification reports, and accuracy metrics for each model. The analysis has been performed to understand where each architecture lies in terms of strengths and weaknesses, and a relative comparison gives one the idea of which will be more effective in recognizing the emotion.

The final stage will cover model adoption within the Medusa system and conducting final tests to implement the whole system with seamless interactions. This involved refining the user interface for smooth interaction, ensuring real-time performance of the system, and ensuring that the various modules worked cohesively. At the end of this phase, the project would be fully integrated and deployed, with all deliverables addressed and the system functioning as intended.

4.3 Team Members and Roles

The Medusa Emotion Detection System was developed through the collaboration of a skilled and dedicated team, each member bringing their expertise to different aspects of the project. Vachram, oversaw the entire development process. He ensured that every phase of the project met its goals and was finished on time. Vachram was the major figure involved in the design, training, and evaluation of the Convolutional Neural Networks (CNNs) that were used for emotion detection. He was also responsible for real-time execution of the model and data accuracy and reliability.

Giannis, was the data-focused tasks lead in the project. The duties were data gathering with the FER-2013 dataset and the overall data check that Giannis did to make sure it fits the preliminary model. Furthermore, Giannis was also in charge of the one of the two functions, which was data processing and augmentation, although the other task, model training, was the common one to all members. The dataset needs to be prepared for training and improved with different techniques, such as making the model stronger and more accurate.

Stelios, led the development of the desktop application. He was responsible for the smooth integration of the emotion detection models into an easy-to-use human interface. Stelios put his efforts into the application's real-time performance optimization as a part of the project, and at the same time, he ensured that the system was compatible with different platforms. In this case, Stelios assured non-undergirded operation.

4.4 Future Work and Improvements

Future work on this project could focus on enhancing the emotion detection model to achieve better accuracy while minimizing overfitting. The current models, which have been analyzed in the previous sections, have shown great potential but also evidence of overfitting-especially when considering more complex architectures such as GoogLeNet. This work may also include better generalization of the model on unseen data by trying different techniques of regularization, early stopping, and data augmentation. Besides, the use of more informative approaches, like ensemble learning or transfer learning, could enhance the performance of the model with no loss in generalization capability.

Yet another direction in which Medusa could be extended is simply in the number of different features it would support; that is, even for emotion detection alone, but also in general, it could serve more broadly. For example, further extensions could integrate more models that can detect attributes such as age, gender, or even specific facial expressions to enhance the model's usefulness. The adoption of the multi-modal approach could potentially enable Medusa to provide even more insight, invaluable in everything from user experience research to security systems.

Also, with the scaling of Medusa, the system would indeed be more powerful and at the same time versatile towards different use cases. From here, age and gender detection would provide the system with the opportunity to adapt the outputs concerning various groups with a particular demographic background. This could become quite useful in industries like marketing where knowing customer segment subtleties is important. Moreover, this may be extended to include other emotion-related features, such as the detection of stress or fatigue, with the aim of expanding the usability of Medusa into health monitoring or workplace safety applications.

This could be the ultimate vision for Medusa-to make it an integrated, complete, and end-to-end product with all these features put together. The product shall offer a value proposition that integrates emotion detection, demographic analysis, and other sophisticated features within one product or solution. An integrated end-to-end system would thus be more user-friendly and effective for processing data, analyzing it, and reporting on one platform. This development would position Medusa as a leading tool in the area of emotion analytics and intelligent human-computer interaction, ready for deployment into a number of commercial and research environments.

Bibliography

1. Goodfellow, I., Bengio, Y., Courville, A. (2016). **Deep Learning**. MIT Press.
2. Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). **ImageNet Classification with Deep Convolutional Neural Networks**. *Advances in Neural Information Processing Systems*, 25, 1097-1105.

3. Barsoum, E., Zhang, C., Ferrer, C. C., & Zhang, Z. (2016). **Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution**.

4. Lundqvist, D., Flykt, A., & Öhman, A. (1998). **The Karolinska Directed Emotional Faces (KDEF)**.

5. Chollet, F. (2017). **Deep Learning with Python**. Manning Publications.

6. **FER-2013 Dataset**. (2013). Kaggle.
7. Abadi, M., et al. (2016). **TensorFlow: A System for Large-Scale Machine Learning**. *OSDI*, 16, 265-283.