Maastricht University
Department of Advanced Computer Sciences
Master's Program in Data Science for Decision Making

Computer Vision(KEN4225)
Assignment 2 (Deep Learning based on CNNs)

Christos Koromilas(i6345703) & **Spyridon Giagtzoglou**(i6329021)

May 30, 2023

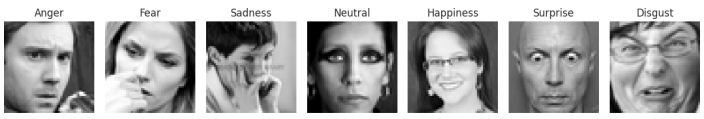
Contents

1 Introduction & Description

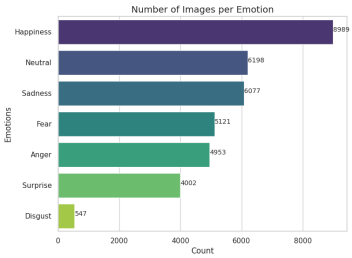
Emotion recognition has gathered significant attention in recent years owing to its extensive application in various fields such as human-computer interaction, robotics, advertising, healthcare, among others. The primary objective of emotion recognition is to decipher the emotional state of a person into one of the basic emotions, which include anger, happiness, surprise, sadness, fear, disgust, or neutral. These emotions are typically recognized based on distinct signals such as facial expressions, body posture, tone of voice, etc. However, this study focuses exclusively on recognizing emotions from facial expressions using Convolutional Neural Networks (CNNs).

1.1 Dataset Description and Preprocessing

In this assignment, we utilized the Facial Expression Recognition (FER) 2013 dataset. This dataset is globally acknowledged and extensively used for emotion recognition research due to its comprehensive and diverse nature. The FER 2013 dataset comprises grayscale images of size 48x48 pixels, each categorized under one of seven basic emotions.



(a) Classes of FER 2013 dataset



(b) Number of examples of each class

Figure 1: Information about the FER 2013 dataset

The dataset preprocessing and splitting began with a visual representation of the recommended split for Training, PublicTest, and PrivateTest. The dataset was divided accordingly and the image data, along with the corresponding emotion labels, were extracted from each subset.

Image pixel data was converted from strings to integers and normalized to a range between 0 and 1. Labels were converted to categorical format using one-hot encoding, creating binary vectors. Pixel data was then reshaped into 4D arrays of 48x48 pixels with a single color channel for grayscale.

Next, the training data underwent a stratified train-validation split. This split maintains a balanced representation of emotion classes.

Finally, the dataset was augmented using horizontal flipping and image rotation techniques. These strategies improve model robustness and generalization by introducing variance and diversity, thus better preparing the model for real-world applications. This process resulted in an enhanced and balanced dataset for model training, validation, and testing.

Furthermore, to evaluate the CNN models in real-life scenarios, we used three videos showing various facial expressions.

In the upcoming sections of this report, we will discuss the design, implementation, and evaluation of four different CNN architectures for the task of emotion recognition. The selection of these architectures and their hyperparameters was guided by the aim to understand the impact of various CNN operations and configurations on model performance. We will also analyze the role of various components such as number of layers, filters, pooling operations, normalization techniques, activation functions, among others, in the model's overall performance.

2 CNN models

We have developed four different Convolutional Neural Networks (CNNs) to perform Facial Expression Recognition (FER).We used crossentropy as a loss function, and the ADAM optimizer to compile CNNs defaults learning rate 0.001.Also we set kernel_size = (3, 3), input_shape=(48, 48, 1) and the pool_size = (2, 2).These models have different structures and complexities and are all designed to take in grayscale 48x48 pixel images as input. The CNN architectures are described as follows:

1. **Simpler_FER_CNN**: This is the most basic CNN model we used. It consists of two convolutional layers, each followed by a max pooling layer for downsampling. The filter sizes for the convolutional layers are 32 and 64, respectively. Following the convolutions, the output is flattened and passed through a dense layer with 128 units. The final layer is a softmax layer with 7 units, corresponding to the seven possible emotion labels.

2. **FER_CNN**: This model expands on the simple architecture by increasing the filter sizes and adding more convolutional layers. It begins with two 64-filter convolutional layers, followed by two 128-filter convolutional layers, and then two 256-filter layers. Each pair of convolutional layers is followed by a max pooling layer. In addition, this model incorporates batch normalization after each convolution and dropout after each max pooling for regularization.

(a) Simple FER CNN (b) FER CNN

Figure 2: The layers plotted for the first two models

3. **V2_FER_CNN**: This model it has different activation function by using a Leaky ReLU activation function instead of a standard ReLU. This can help mitigate the "dying ReLU" problem, where some neurons can become inactive and stop learning. The kernel initializer "he_normal" is also used for better weight initialization. Additionally, this model uses increased dropout rates after each max pooling layer, going from 0.2 up to 0.4.

4. **regular_FER_CNN**: The final model uses L2 regularization in each convolutional and dense layer, in addition to the batch normalization and dropout used previously. L2 regularization helps prevent overfitting by penalizing large weights in the model. Like the improved model, it has a similar structure to the FER_CNN model, but the use of regularization provides an additional constraint on the model's complexity.

(a) V2 FER CNN (b) REGURALIZED FER CNN

Figure 3: The layers plotted for the third and fourth model

All four models conclude with a dense layer using a softmax activation function, ensuring the model's output can be interpreted as probabilities for each of the seven possible emotion labels. These CNN architectures represent a range of complexity, allowing us to examine the tradeoff between model capacity and the risk of overfitting.

3 Results

The results provide an insight into the performance of each of our four models on a multi-class classification problem. The performance is broken down class by class, which can provide useful information if some classes are more important than others.

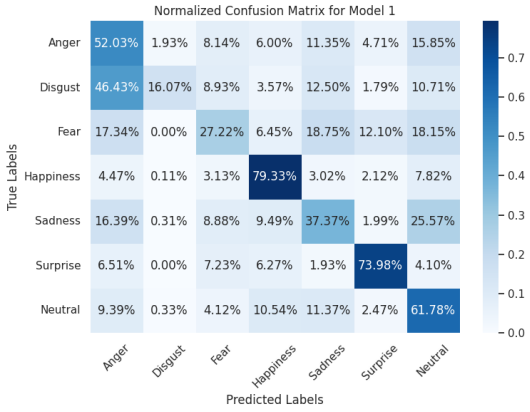
Table 1: Classification Report for all Models

Model	**Class**				**Accuracy**	**Macro Avg**	**Weighted Avg**
	Precision	**Recall**	**F1-Score**	**Support**			
		Model 1			0.56	0.50	0.55
0	0.41	0.52	0.46	467			
1	0.39	0.16	0.23	56			
2	0.42	0.27	0.33	496			
3	0.77	0.79	0.78	895			
4	0.49	0.37	0.42	653			
5	0.70	0.74	0.72	415			
6	0.47	0.62	0.53	607			
		Model 2			0.64	0.59	0.63
0	0.55	0.60	0.57	467			
1	0.61	0.39	0.48	56			
2	0.52	0.27	0.36	496			
3	0.85	0.87	0.86	895			
4	0.55	0.52	0.54	653			
5	0.75	0.77	0.76	415			
6	0.52	0.71	0.60	607			
		Model 3			0.64	0.57	0.62
0	0.57	0.59	0.58	467			
1	0.75	0.27	0.39	56			
2	0.59	0.20	0.30	496			
3	0.85	0.85	0.85	895			
4	0.52	0.51	0.52	653			
5	0.75	0.85	0.79	415			
6	0.49	0.74	0.59	607			
		Model 4			0.55	0.46	0.52
0	0.52	0.38	0.44	467			
1	0.00	0.00	0.00	56			
2	0.33	0.14	0.20	496			
3	0.75	0.83	0.79	895			
4	0.43	0.49	0.46	653			
5	0.53	0.85	0.65	415			
6	0.48	0.51	0.49	607			

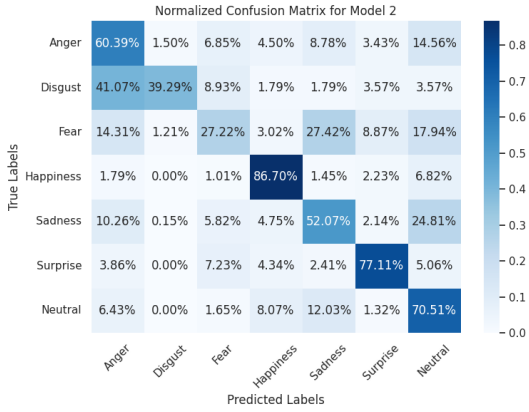
A brief analysis:

- Model 1: The accuracy of this model is 56%. It has the highest F1-score on class 3 and class 5, suggesting good performance on these classes. However, its performance on class 1 and class 2 is considerably lower.
- Model 2: The accuracy of this model improves to 64%, a substantial increase over Model 1. It shows strong performance on class 3 and class 5, with a noticeable improvement on class 0, class 4, and class 6 as well. However, the F1-score for class 2 still remains low, indicating potential areas for improvement.

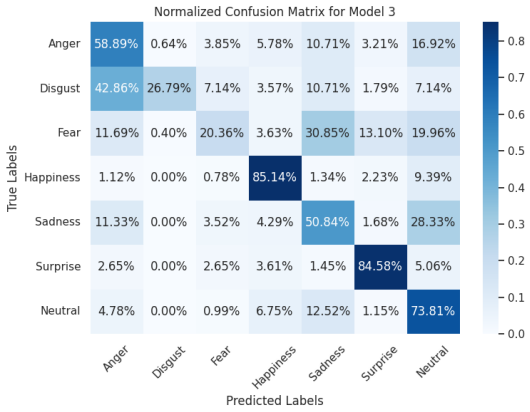
- Model 3: The accuracy remains the same as Model 2, i.e., 64%. This model significantly improves on class 0 compared to Model 2, but there's a decrease in F1-score for class 1 and class 2. Class 3 and class 5 continue to have a high F1-score, while class 6 sees a slight drop compared to Model 2.
- Model 4: This model's accuracy drops slightly to 55%. The F1-scores for most classes drop as well, but there's a significant improvement in F1-score for class 5 compared to other models. This model fails to correctly classify any instances of class 1.



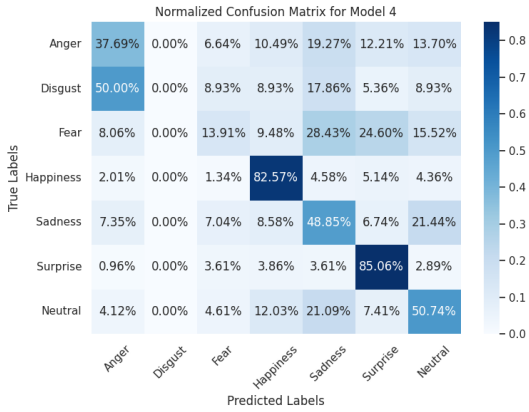
(a) Normalised Confusion Matrix for Model1



(b) Normalised Confusion Matrix for Model2

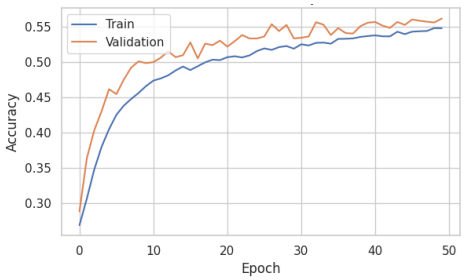


(c) Normalised Confusion Matrix for Model3

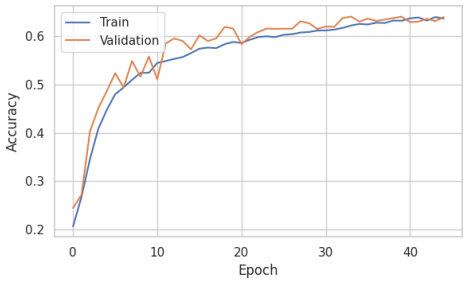


(d) Normalised Confusion Matrix for Model4

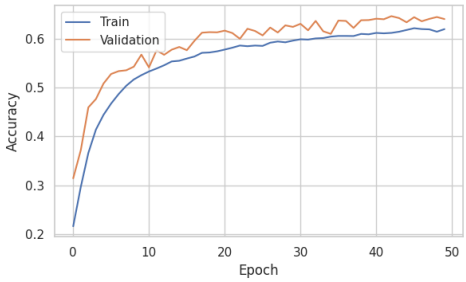
Figure 4: Confusion Matrices



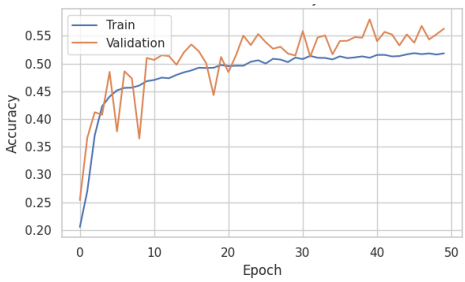
(a) Accuracy for Model1



(b) Accuracy for Model2

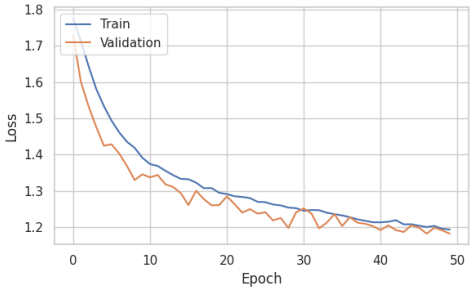


(c) Accuracy for Model3

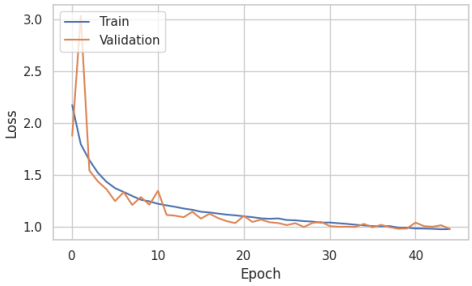


(d) Accuracy for Model4

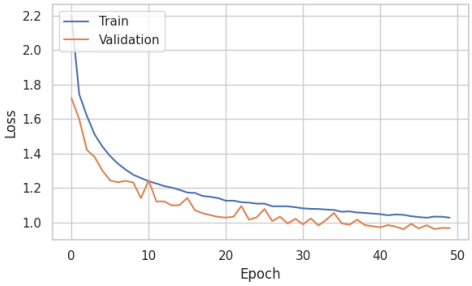
Figure 5: Plots of Accuracy for Models



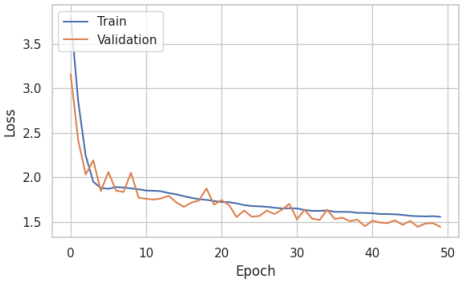
(a) Loss for Model1



(b) Loss for Model2



(c) Loss for Model3



(d) Loss for Model4

Figure 6: Plots of Val Loss for Models

Each of the models in this analysis has its own strengths and areas for improvement, and the optimal choice depends heavily on the specific requirements of the problem at hand.

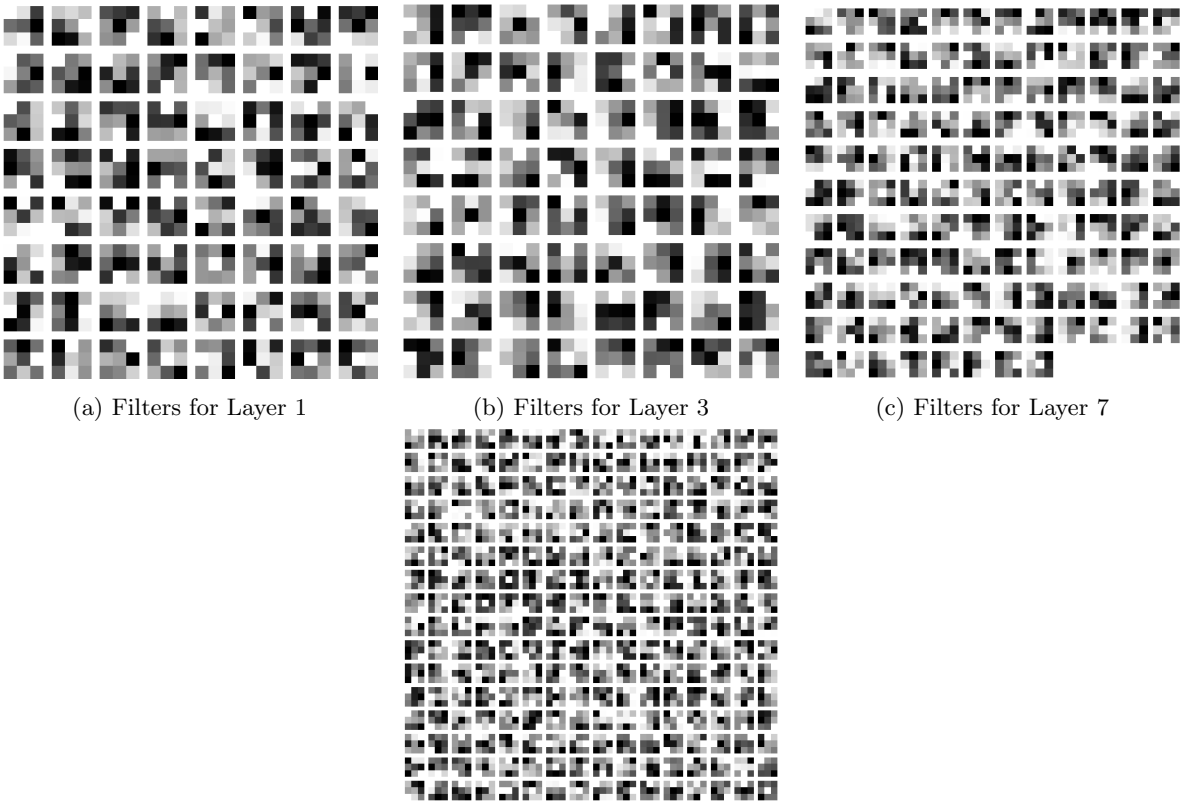
Overall, Model 2 and Model 3 appear to perform the best overall, with accuracy of 64%. However, if we care more about performance on specific classes, you might favor one model over the others.

It's important to remember that overall accuracy is not always the best metric, especially if your classes are imbalanced. You might care more about precision or recall for specific classes, or about the F1-score, which is a combination of precision and recall. It's also worth noting that there's room for improvement in all of the models, as none of them have particularly high performance on class 1 and class 2.

After taking all factors into consideration, Model 2 is chosen as the best model for several reasons. First, it offers the highest overall accuracy, which suggests that it makes the least amount of total mistakes among all models. Second, it exhibits a consistent and balanced performance across different classes, suggesting that it handles the diversity of the data well. Lastly, even though it's not perfect and has room for improvement on class 1 and class 2, its robustness and versatility make the most reliable choice for a wide range of applications.

4 Learned Filters in CNNs

In our experimentation of FER using various CNN architectures, we observed that one model is particularly effective. This specific CNN, distinguished for its superior performance, is characterized by its hierarchical feature learning. The first layer with 64 filters (Figure 7(a)) identifies basic image aspects like edges and color gradients. Another 64-filter layer (Figure 7(b)) continues this process, recognizing complex patterns. The network then discerns intricate textures using 128-filter layers (Figure 7(c)) and high-level visual cues via 256-filter layers (Figure 7(d)). These cues could include facial features indicative of specific emotions. This gradual deepening of comprehension, allowing systematic image breakdown, contributes to the model's superior performance in FER tasks.



(a) Filters for Layer 1

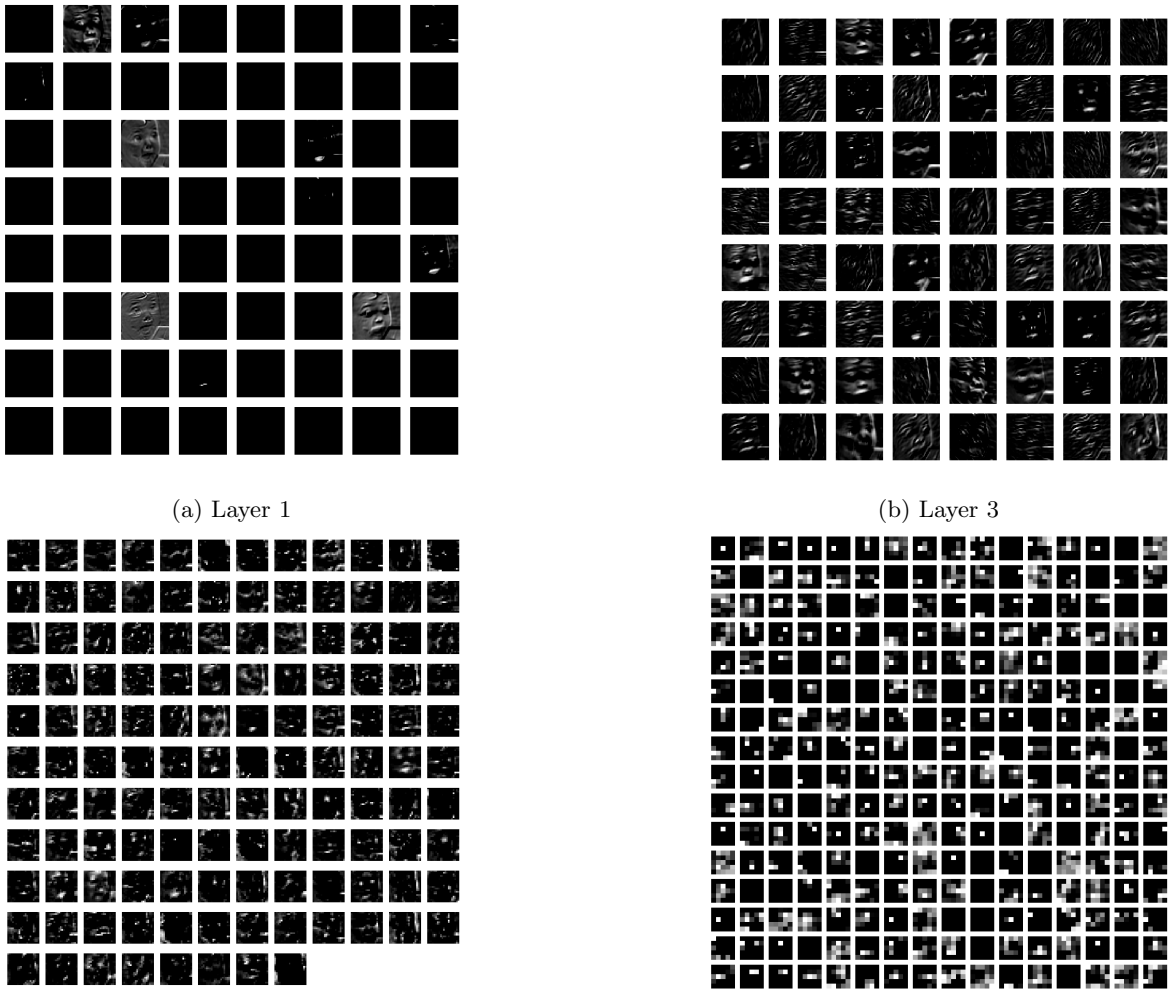
(b) Filters for Layer 3

(c) Filters for Layer 7

(d) Filters for Layer 15

Figure 7: Plots of Filters of Conv2D

The activation visualizations from an example in Figure 8 illustrate how distinct facial features activate different filters.



(a) Layer 1

(b) Layer 3

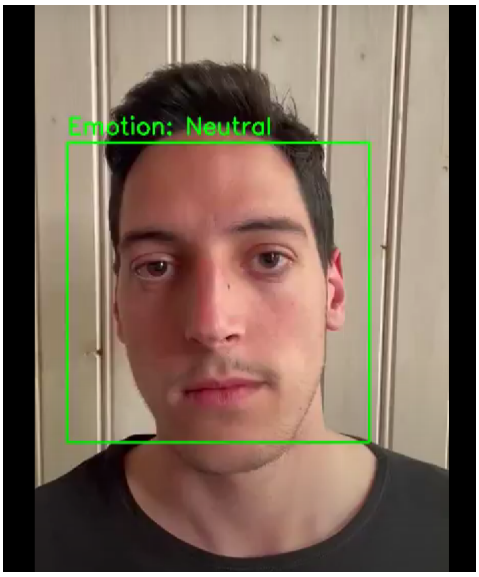
(c) Layer 7

(d) Layer 15

Figure 8: Plots of Layers of Conv2D for one example image

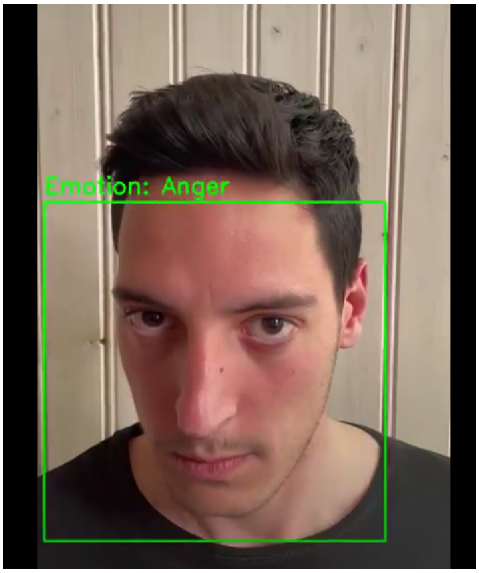
5 Real life videos experiment

We build a script which is effectively utilizes video files for complex emotion recognition through the application of our CNN model FER_CNN. The initial phase employs the Haar cascades classifier, renowned for its proficiency in face detection. Subsequently, the regions of interest, representing detected faces, are extracted, resized, and normalized, priming them for analysis by the FER_CNN. The chosen model provides a prediction of the emotion, which is then annotated on the video frame along with a bounding box highlighting the face. All processed frames are consolidated into a final output video. Additionally, the script creates a slow-motion version of the initial video file, customized to a pre-set size. This slow-motion video is temporarily stored, following which a video widget is constructed to display the resultant slow-motion output.We attach some images of the slow-motion output video.

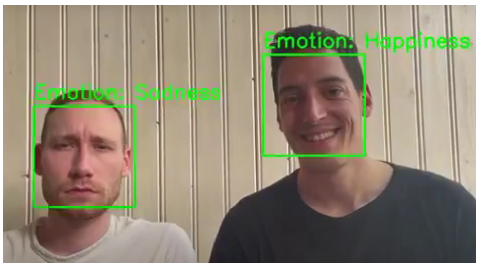


(a) An image from the first video

(b) An image from the first video



(c) An image from the second video



(d) An image from the third video

Figure 9: Samples of the output videos of emotion regocnition

6 Discussion and Conclusion

This experiment offered critical insights into CNN model construction for Facial Expression Recognition (FER). All models used the same loss function, optimizer, kernel size, input shape, and pool size, differing primarily in architecture and complexity.

Model 1, a simple CNN, highlighted the limitations of a basic structure in handling diverse classes. Model 2, with greater complexity, demonstrated enhanced performance, validating the importance of deeper networks in image classification tasks.Model 3, despite using Leaky ReLU and increased dropout rates, showed similar accuracy to Model 2, indicating that while hyperparameters are crucial, they don't guarantee improved accuracy. Model 4 applied L2 regularization, yet saw a decrease in performance, underscoring that excessive regularization can lead to underfitting.

Across all metrics, Model 2 emerged as the most effective, maintaining high accuracy and balanced performance across classes. It managed the data's diversity well, despite challenges with classes 1 and 2. Therefore, this experiment emphasizes the balance of model complexity, judicious hyperparameter selection, and avoiding over/underfitting for effective CNN optimization. Future work should consider these aspects, aligning them with specific class-performance requirements.

In our expirements, we also implemented transfer learning with the **VGG16** pre-trained model, tailored to our FER task with additional custom layers. This model achieved an accuracy of 41%, which is respectable yet comparatively lower than our custom-built CNN Model 2. This could suggest that for certain tasks, particularly domain-specific ones like FER, models specifically crafted to match the complexity of the task could perform optimally. Hence, while the robustness of VGG16 is a significant advantage, it's possible that its complexity exceeds the requirements for this particular task, underscoring the importance of calibrating model complexity according to task specificity for optimal performance.