Emotion Recognition: Comparing Image Classification Models

João Atz Dick
Department of Electrical & Computer Engineering
University of Toronto
joao.dick @mail.utoronto.ca

Chenhao Hong
Department of Electrical & Computer Engineering
University of Toronto
chenhao.hong@mail.utoronto.ca

Pengyu Pan
Department of Electrical & Computer Engineering
University of Toronto
pengyu.pan@mail.utoronto.ca

Sebastian Tampu
Department of Electrical & Computer Engineering
University of Toronto
seb.tampu@mail.utoronto.ca

Assentation of Teamwork

João Atz Dick: ResNet

Chenhao Hong: VGG16

Pengyu Pan: Custom CNN

Sebastian Tampu: AlexNet

Abstract

This project addresses the challenges of recognizing human emotions from images, a task with significant applications across various domains such as mental health, customer service, and entertainment. Leveraging a dataset of approximately 36,000 grayscale images categorized into seven emotions, we employ CNNs for classification. Specifically, we explore and benchmark the performance of four distinct CNN architectures: ResNet-18, AlexNet, VGG-16, and a custom-designed model. Through comparative analysis based on accuracy and F1 scores and employing strategies such as data augmentation and model optimization techniques, we aim to overcome challenges such as the nuanced complexity of emotional expressions and dataset imbalance. The project provides both quantitative metrics and qualitative insights through feature map visualizations, enhancing our understanding of the performance of models in emotion recognition.

1 Introduction

Human emotion recognition from images represents a significant challenge and opportunity in the field of artificial intelligence, with profound implications for various applications, including mental health assessments, customer service, and interactive entertainment. The complexity of human emotions, coupled with the diversity in their expression across different individuals and contexts, necessitates advanced computational techniques for accurate classification and analysis. In this project, we aim to tackle the problem of classifying human emotions into seven distinct categories: fear, disgust, neutral, happy, angry, surprise, and sad, using a dataset of approximately 36,000 grayscale images [1]. We leverage the power of convolutional neural networks (CNNs), employing a diverse set of architectures, namely ResNet-18, AlexNet, VGG-16, and a custom-designed CNN tailored to our specific problem domain. Our approach includes comparing these models based on several performance metrics to identify the most effective architecture for emotion recognition. By addressing the dataset's inherent imbalance through data augmentation and optimizing our models with techniques such as dropout and batch normalization, we aim to enhance the

generalization capability of our solution. The project not only focuses on quantitative metrics such as accuracy, precision, recall, and F1 scores, but also presents qualitative analyses through the visualization of feature maps, offering insights into how different models process emotional cues in images.

2 Problem Specification

The core problem our project addresses is the classification of human emotions from images into one of seven categories: fear, disgust, neutral, happy, angry, surprise, sad. This multi-class classification problem is challenging due to the subtle nuances and complexities inherent in human emotional expressions, as well as the variations in how emotions are expressed across different cultures, genders, and age groups. The uneven distribution of images across the emotion categories in our dataset introduces additional challenges, potentially biasing model performance towards overrepresented classes.

Our objective is to develop a robust model that can accurately recognize and classify these emotions, notwithstanding the challenges. To achieve this, we utilize a dataset of around 36,000 grayscale images, applying data augmentation techniques to mitigate class imbalance and enhance model robustness. The employment of CNN architectures, ResNet-18, AlexNet, VGG-16, and a specially crafted custom CNN, allows us to explore and compare different approaches to emotion recognition. Each model will be adjusted to accept single-channel grayscale images, reflecting our dataset's composition, with the flexibility to adapt to multichannel inputs should the need arise from considering additional datasets.

3 Design Details

In this project, we used 4 different CNN models to perform the classification of the images. The detailed architectures of the 4 models are illustrated below.

A. Custom CNN

For the development of the customized Convolutional Neural Network (CNN) model, the initial structure was composed of three convolutional layers, each accompanied by a max pooling layer, and two fully connected layers. The activation function selected was the Rectified Linear Unit (ReLU), and cross-entropy was employed as the loss function. Considering the dataset comprised exclusively of grayscale images, the input channel was reduced to 1, contrary to the conventional 3 channels. The convolutional layers' output channels were set to 16, 32, and 64, respectively. Through the performance assessment detailed in the Numerical Experiment section, four iterations of the CNN model were explored. The first model's training highlighted pronounced overfitting, leading to the inclusion of batch normalization layers; however, this modification did not effectively mitigate the overfitting problem. In the third model iteration, dropout layers were incorporated, which not only postponed the occurrence of overfitting to later epochs but also resulted in a decrease in training accuracy to a lower level. The ultimate model iteration entailed the addition of extra fully connected and convolutional layers, a decision that proved to be redundant, as it did not markedly improve the model's performance in comparison to the modifications made in the third iteration.

The final iteration of the model comprised four convolutional layers and four fully connected layers, each equipped with batch normalization and a dropout rate of 0.2. Observing the slow rate of improvement in training accuracy, the decision was made to extend the training duration to 50 epochs for this final experiment, as opposed to the 20 epochs designated for the initial three iterations. This adjustment aimed to provide sufficient time for the model to converge and potentially enhance performance.

B. AlexNet

AlexNet is a CNN architecture that marked a pivotal moment in the field of deep learning and computer vision. Developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, AlexNet was introduced in 2012 when it won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a large margin [2]. This competition involves classifying images into 1,000 categories, and AlexNet's performance demonstrated the potential of deep learning models for computer vision tasks, significantly outperforming traditional image recognition methods at the time.

The architecture of AlexNet is distinguished by several key innovations that have significantly influenced the development of CNNs for image recognition tasks. It contains the following components:

- Five Convolutional Layers
- ReLu Activation
- Overlapping Pooling
- Local Response Normalization (LRN)
- Three Fully Connected Layers
- Dropout

The specifics can be seen in Figure 1.

Of course, several changes had to be made for this model to work with the chosen dataset. This refers to the usage of grayscale photos when AlexNet expects RGB photos or 3 channels. There is also the issue of image size where the dataset consists of 48x48 pixel images, while the model was built for 224x224 pixel images. Lastly, a change had to be made from making predictions for 1,000 classes to only 7 classes.

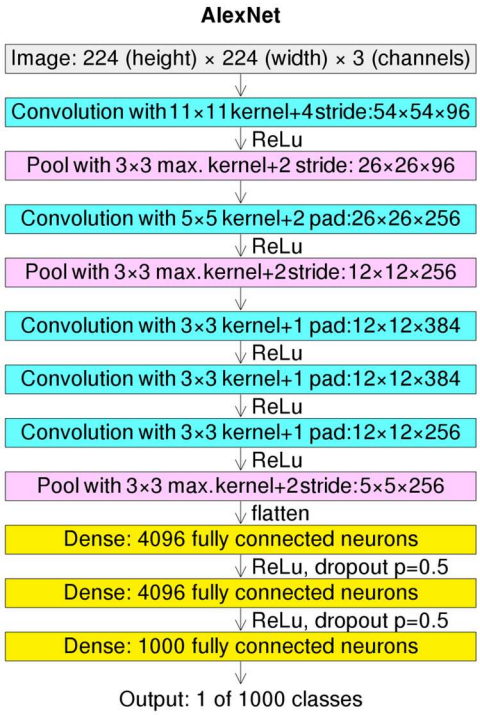

Figure 1: Specifics of AlexNet Architecture [3]

AlexNet was used with support from PyTorch libraries and modules for the task of emotion recognition from images. The *torch* library provided essential functions for tensor operations, device management, and neural network construction, including *torch.device* for computation device selection, *torch.nn* for neural network layers like *nn.Linear* which adapted AlexNet's classifier for a seven-class output, and *torch.optim* for optimization algorithms such as Adam and learning rate schedulers. Data handling was facilitated by *torch.utils.data* with tools like *DataLoader* and *random_split*.

Considering AlexNet's specification for 224x224 pixel RGB images, we applied transformations to adapt our grayscale images for this model. The *transforms.Compose* function enabled the combination of multiple image transformations, such as converting grayscale images to three-channel format using *transforms.Grayscale* for AlexNet compatibility. Additional transformations like *RandomResizedCrop* and *Normalize*, aligned with ImageNet dataset standards, prepared our data for the model. Data augmentation techniques, including resizing, cropping, flipping, rotation, and color jittering, were used to improve model robustness and training vairability. Simpler augmentation was applied to the validation and test sets.

Weighted loss and oversampling were experimented with to address data imbalance but found no significant performance improvement, leaving the focus on the AlexNet architecture and data augmentation.

Gradient computation in forward and backward passes was controlled by *torch.set_grad_enabled,* complemented by optimizers and learning rate schedulers. The model was trained and tested several times changing various hyper parameters such as initial learning rate, batch size, number of epochs, and the scheduler step size and gamma values. This helped in choosing the ideal configuration for both accuracy and efficiency.

The configuration for training is as follows. The number of epochs is 50 with a batch size of 64. The initial learning rate is 0.01, which then decreases by a factor of 10% every 10 steps. As mentioned earlier,

Adam is used as the optimizer, and cross entropy as the loss given the multi-class classification nature of the problem.

C. VGG-16

VGG-16 is developed by Karen Simonyan and Andrew Zisserman [4] from University of Oxford. The main contribution of this model is using small filter size (3 * 3) with higher depth (16, 19 layers). The original model was applied to the ImageNet Challenge 2014 and won first and second places at that competition.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2: Specifics of VGG Architecture [4]

The figure above shows overall structure of the VGG model. In this model, we use VGG-16 which is the D model in the figure.

The original model is for 244 * 244 RGB images but our dataset is grey image with 48 * 48 size. To fit the data, the initial input channel of the model is changed to 1 and the stride of the last pooling layer is set to 1 instead of 2 for the rest of the pooling layers. The last FC layer in our model outputs 7 classes instead of 1000 since our data consists 7 classes.

In addition to the layers outlined above, The learning rate is set to 0.0001 and the optimizator we used is Adam. The loss function is cross entropy. We train the model for 50 epoches and 64 batch size. We add dropout with rate 0.2 and batch normalization after each actication functions to avoid the overfitting. The overfitting problems still exists and will be discussed in the numerical experiment section.

D. ResNet

The model architecture is inspired by the deep residual convolutional model ResNet. A sequence of several interpolated Conv2d, BatchNorm2d, and *MaxPool2d* layers is combined with the ReLU activation function. At the end, a Flatten2d layer is followed by a final Linear classification layer. The Figure below displays the residual network architecture. We can observe that the number of trainable parameters is equal to 6,573,192.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [64, 64, 48, 48]             640
       BatchNorm2d-2           [64, 64, 48, 48]             128
              ReLU-3           [64, 64, 48, 48]               0
            Conv2d-4          [64, 128, 48, 48]          73,856
       BatchNorm2d-5          [64, 128, 48, 48]             256
              ReLU-6          [64, 128, 48, 48]               0
         MaxPool2d-7          [64, 128, 24, 24]               0
            Conv2d-8          [64, 128, 24, 24]         147,584
       BatchNorm2d-9          [64, 128, 24, 24]             256
             ReLU-10          [64, 128, 24, 24]               0
           Conv2d-11          [64, 128, 24, 24]         147,584
      BatchNorm2d-12          [64, 128, 24, 24]             256
             ReLU-13          [64, 128, 24, 24]               0
           Conv2d-14          [64, 256, 24, 24]         295,168
      BatchNorm2d-15          [64, 256, 24, 24]             512
             ReLU-16          [64, 256, 24, 24]               0
        MaxPool2d-17          [64, 256, 12, 12]               0
           Conv2d-18          [64, 512, 12, 12]       1,180,160
      BatchNorm2d-19          [64, 512, 12, 12]           1,024
             ReLU-20          [64, 512, 12, 12]               0
        MaxPool2d-21            [64, 512, 6, 6]               0
           Conv2d-22            [64, 512, 6, 6]       2,359,808
      BatchNorm2d-23            [64, 512, 6, 6]           1,024
             ReLU-24            [64, 512, 6, 6]               0
           Conv2d-25            [64, 512, 6, 6]       2,359,808
      BatchNorm2d-26            [64, 512, 6, 6]           1,024
             ReLU-27            [64, 512, 6, 6]               0
        MaxPool2d-28            [64, 512, 1, 1]               0
          Flatten-29                  [64, 512]               0
           Linear-30                    [64, 8]           4,104
================================================================
Total params: 6,573,192
Trainable params: 6,573,192
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.56
Forward/backward pass size (MB): 1305.50
Params size (MB): 25.07
Estimated Total Size (MB): 1331.14
----------------------------------------------------------------
```

Figure 3: Specifics of ResNet Architecture

The Adam optimizer was utilized with learning rate = $8.05\times10-05$, $\beta1 = 0.85$, and $\beta2 = 0.999689$. The chosen loss function was cross entropy. Training was performed for 50 epochs.

4 Numerical Experiments

The original dataset was pre-split into training and validation sections of 80% and 20% respectively. The test set is taken from half of the validation set.

A. Scores

Given the imbalanced nature of the dataset, the usual metrics of test accuracy and test loss are not enough to judge the quality of the models. In addition to those, the F1 score will also be used. The results are summarized in Table 1.

Table 1: Model Test Scores

Model	Accuracy	Loss	F1 Score
Custom CNN	0.616	1.14	0.140
AlexNet	0.258	1.81	0.105
VGG-16	0.645	1.49	0.143
ResNet	0.681	0.997	0.145

The final iteration of the Custom CNN model resulted in an accuracy of 0.616 under 50 epochs of training. The F1 score is low. Interestingly, the loss of Custom CNN outperformed AlexNet and VGG-16.

As seen in Table 1, the scores for AlexNet demonstrate that the model has not learned well from the data. The F1 score is very poor, while the accuracy shows that the model does not do much better than guessing from the 7 categories. Training and validation accuracy scores were not much better and did not improve much as training went on. This is perhaps due to trying to fit the data to the model rather than the other way around. Feature maps after the first two convolutional layers are presented in Figure 3 for more analysis.

Based on the table, the overall performance for VGG16 is poor. The F1 score is low, and the accuracy is only about 0.65, indicating that the model could not predict the correct classes overall. The reason for this performance is discussed below.

Resnet presents the best overall results for accuracy, Loss, and F1 score. Nevertheless, the model is also not able to achieve a satisfactory performance, with accuracy around 0.68. Further training, data

B. Other Results

Custom CNN

For the first version of Custom CNN, the model overfitted after short epochs:

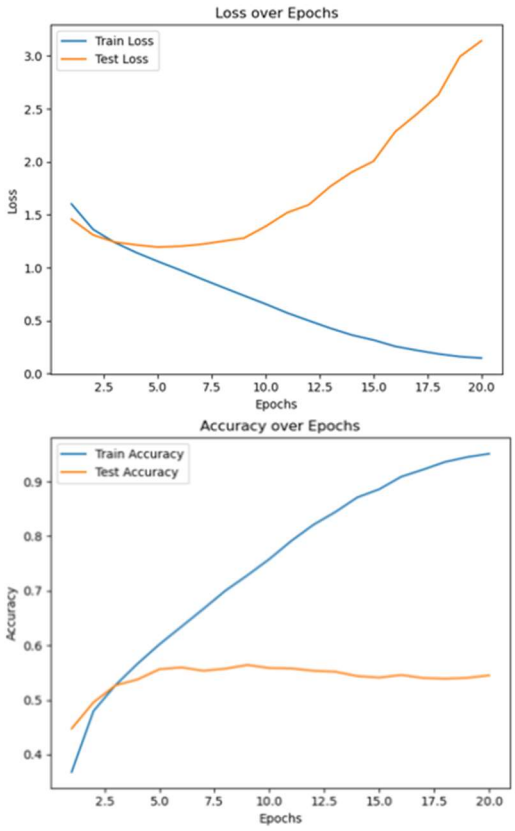

Fig4.1: First version of Custom CNN

After adding batch normalization, the result didn't change much, but after adding the dropout layers, the ascendant of train accuracy slowed heavily, through 20 epochs of training, the train accuracy only reached 0.639, even though it's still increasing, the speed is slow. This is visible in Figure 4.2.

As seen in Fig 4.3, the final version of the Custom CNN performed like an enhanced version of version 3, with more training epochs, the train accuracy further increased to 0.653, the test accuracy fluctuated between 0.616 and 0.601, suggesting the model loss reaching a local minimum.

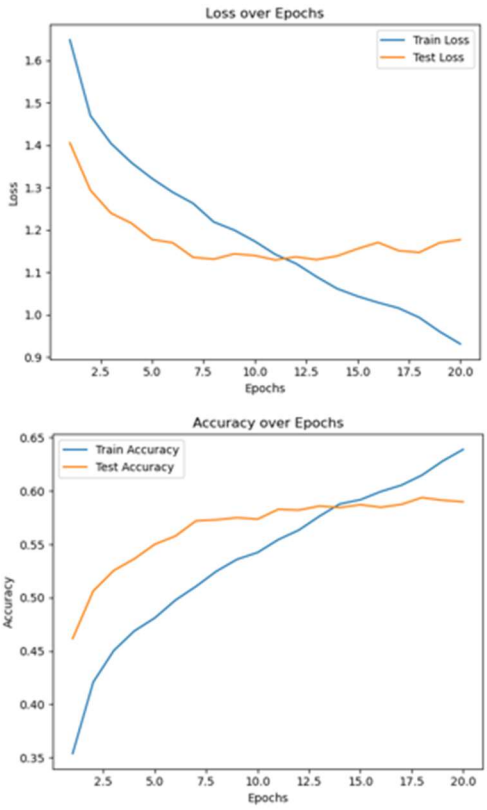

Fig4.2: Third version of Custom CNN

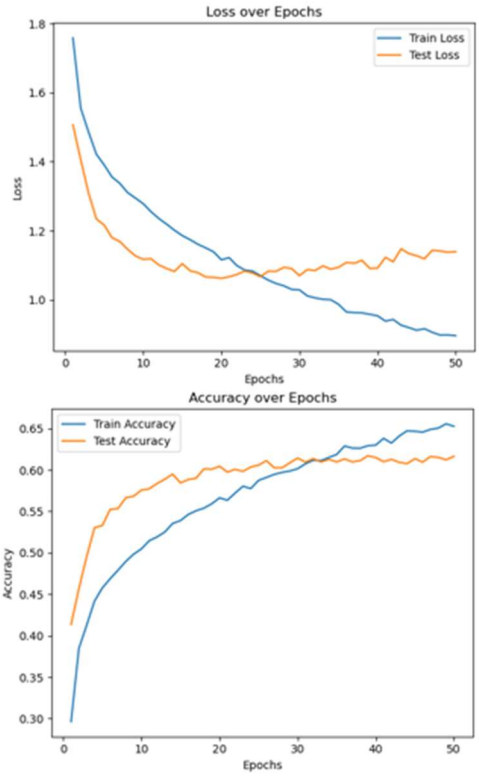

Fig4.3: Fourth iteration of Custom CNN

We suspect the poor performance of the model is due to the unbalanced distribution of the dataset, since some classes have more samples than others, causing insufficient data for certain classes. The greyscale images could be another reason for the overfitting, since it only contributes 1 channel for the initial CNN input, hence providing less features for models to learn. The reason for the drop of train accuracy after adding dropout layers could be that given the small nature of the Custom CNN model, dropping neurons on each layer left the network less weights to pass, thus slowing the training.

AlexNet

As mentioned, AlexNet did not score well on any metrics, and so it is important to visualize how the model was learning the features of the images. An example is presented in Figure 6. Here we see the original image in the first square, followed by two feature maps after the first convolutional layer in the second and third squares. The final square presents what all the feature maps looked like after the second convolutional layer, a black square.

Figure 5: Sample Feature Maps of AlexNet Architecture

Potential issues that would lead to this might be from data preprocessing, although the data was normalized. Vanishing gradients should not be an issue given the simplicity of the architecture. Lastly, the learning rate is also low enough to allow for proper learning. The assumption then is that the data and the model are a poor match. AlexNet's ability to generalize to other types of images, other than the ImageNet it was built for, was something that we wanted to test. As such, changes were not made to the architecture to suit the images, leading to the usage of poor network architecture and parameters for the task.

VGG16

The main reason for the poor performance of VGG16 is overfitting. Figure below clearly indicates the existence of overfitting. Inside the model, there is Dropout and batch normalization after all

the activation functions. Besides, we have tuned the dropout rate from 0.2 to 0.8 and none of them solves the overfitting problems. The test risk always increases after around 20 epochs and the test accuracy stays at 0.65.

We believe that the overfitting occurs because the VGG16 has large capacity compared to the task we aim to do. The structure of the model is initially planned for 1000 classes and trained by millions of images. For our case, the number of classes is much less, and the number of images passes for training may not be adequate.

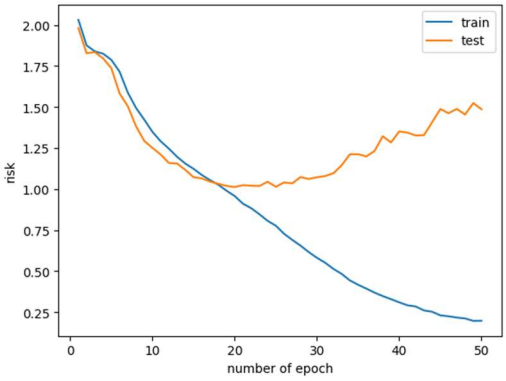

Figure 6: Plot of risk vs epoch

Resnet

Figure 7 displays the per-class accuracy. We can observe that accuracy is highly variable between classes. Going from 80.43% in the surprise class to as low as 34.23% within the disgust class. This suggests that some data cleaning and refactoring may be needed. Figure 8 displays the training loss and validation accuracy during training. We can observe that the model converges as expected.

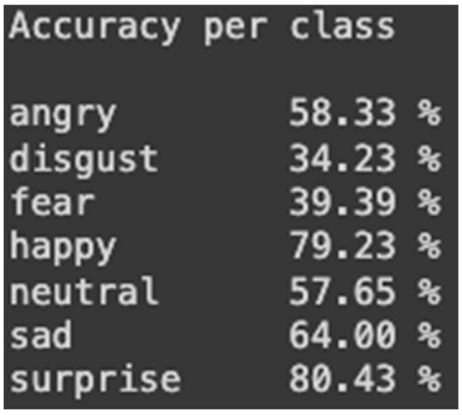

Figure 7: Per-class accuracy.

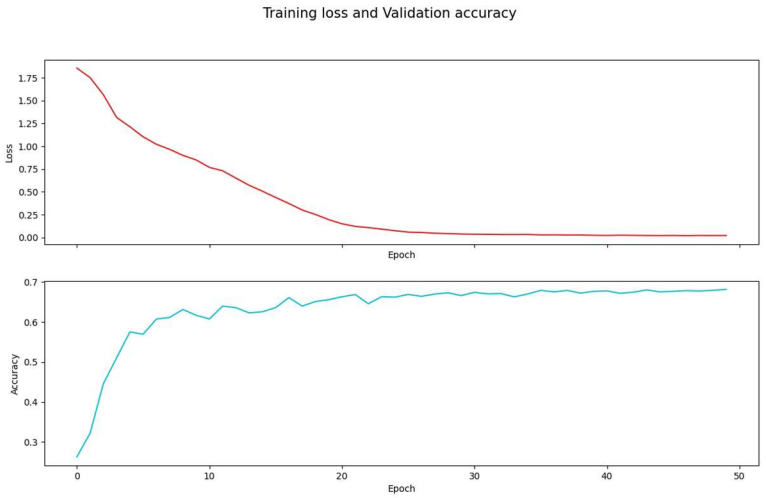

Figure 8: Training loss and validation accuracy.

C. Overall Comparison

In evaluating the performance of our models, it became evident that none achieved the anticipated levels of accuracy. Among the custom CNN, VGG-16, and ResNet models, the highest accuracy recorded was a modest 68.1%, with ResNet leading in performance. This outcome was somewhat expected, considering ResNet's more recent development and the greater focus on its optimization compared to both AlexNet, VGG-16, and our custom-built CNN.

However, the analysis based on accuracy scores only provides a partial view of the models' performance. The F1 scores, which offer a more nuanced reflection of model efficacy, especially in the context of an imbalanced dataset, reveal a significant shortfall. With an ideal benchmark of 0.7 for F1 scores denoting satisfactory model performance, our highest achieved score was only 0.145. This indicates a critical area for improvement, underscoring the need for enhanced strategies in handling imbalanced datasets to ensure more balanced and accurate emotion classification.

5 Conclusions

Through this project, we deepened our expertise in deploying CNN models, enhancing our skills from foundational concepts to advanced implementation. We gained valuable experience in handling and preprocessing image data, constructing and refining models, and adjusting

hyperparameters to optimize performance. The resolution of unexpected challenges, guided by theoretical knowledge acquired from academic studies has notably advanced our proficiency in deep learning.

To elevate our model's accuracy, a pivotal next step is the augmentation of our dataset. Specifically, addressing the challenge of data imbalance, particularly evident in underrepresented classes like disgust, is crucial for achieving more balanced learning and improved accuracy across all emotion categories. Strategic data augmentation techniques tailored to these less represented emotions could significantly bolster model effectiveness.

Moreover, refining our approach by concentrating on binary classifications, such as distinguishing between positive and negative emotions, could streamline the model's predictive accuracy. This is highlighted by the ResNet model's performance in identifying happy and sad emotions. Focusing on binary outcomes could simplify the complexity of emotion classification and potentially lead to more robust overall performance. This endeavor underscores the ongoing journey in machine learning, where continuous learning and adaptation pave the way for innovative solutions to enhance outcomes.

References

[1] J. Ohex, "Face Expression Recognition Dataset," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset. Accessed on: Mar. 3, 2024.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the Neural Information Processing Systems (NIPS)*, 2012.

[3] Alecive, "Comparison image neural networks," Wikimedia Commons, [Online]. Available: https://commons.wikimedia.org/wiki/File:Comparison_image_neural_networks.svg. Accessed on: Accessed on: Apr. 4, 2024.

[4] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).