Memorial University of Newfoundland

Faculty of Engineering and Applied Sciences

Emotion Detection Using Deep Convolutional Neural Network

Course: *COMP 4301-001 Computer Vision*

Group Members:

Muhammad Hammad

Elliott Barnes

Abstract

Human-computer interaction (HCI) is a rapidly evolving field of technology that is accelerating as we integrate more technology into our lives. Emotions play a vital part in human interaction and are therefore an important part in fully developing a smart HCI system. In this paper, we propose a deeply connected Convolutional Neural Network (CNN) to classify the emotions of an individual, in real-time, into two different classes. We use four encoding blocks which consist of convolutional, max-pooling, dropout and ReLU layers; in addition to four decoding blocks which consist of dense layers, to train our sequential neural network model. On the test data, our model achieved accuracy of around ~87% and cross-entropy loss rates of around 0.30. Our results showed that the model was able to predict the emotions accurately with very high confidence in its prediction accuracy.

Table of Contents

1. Introduction

Emotion detection, more commonly known as Facial Expression Recognition (FER) in the scientific community, is a rapidly growing field of research. With humans adopting more technology into their daily lives, like the increasing use of smart IoT devices and personal assistants, the need for better human-computer interaction has become crucial. Humans rely on emotions, and context based on emotions to communicate effectively. This ability to identify, detect and understand human emotions is not inherently available within a computer, which makes human-computer interaction feel less intrinsic and natural. In order to have AI and computers fully understand humans, we need to design programs that will allow computers to successfully identify and understand human emotions.

1.1 Problem Definition

Classifying every human emotion is a topic too complex to solve here. In this paper, we deconstruct the problem into a smaller subset and focus solely on classifying between two emotions: Happy and Neutral. The goal is to create a deep neural network model, capable of identifying between the two unique emotions with a high prediction accuracy.

2. Methodology

Convolutional neural networks (CNN) are composed of multiple convolutional layers, in addition to other hidden layers; these layers perform mathematical operations on the input data, then pass the output into the next layer's input. These layers form a deeply connected neural network, capable of classifying features within data; because these layers are linearly connected, the full model is referred to as a Sequential model. In this section, we will describe the type of model we use, including some of the common layers available to us for creating a neural network, and then explain how we have combined them to create our model.

2.1 Convolutional Layer

As suggested by the name of the neural network, the primary layer in any CNN is the convolutional layer. This layer uses a filter of size MxM to perform a convolution on the input image, sliding the filter across the width and height of the image. The output of this convolution is combined with an activation function to produce a feature map, the feature map can then be used to reduce the number of free parameters during training, allowing the network to be deeper. When feature maps are combined together with other feature maps, they can be used to identify complex features within images, therefore enabling computers to detect and classify these patterns.

2.2 Nonlinear Activation Layer

Nonlinear activation layers are an essential part of any CNN, they are commonly used after other layers that transform the input. The activation layers decide which neurons to fire for the next layer's input, thus transforming the input. The addition of nonlinearity into this process is crucial for our

learning function, especially if it is training to identify complex features within images that aren't linearly defined.

2.2.1 ReLU Function

Rectified Linear Units or ReLU function is a nonlinear activation function that utilizes gradient-based learning and behaves similar to a linear function. ReLU uses a computationally simple function to model the output of a neuron using $f(x) = max(0, x)$, this function produces 0 for all $x < 0$ and x for all $x >= 0$. It is faster and simpler compared to the more commonly used tanh [$f(x) = tanh(x)$] and sigmoid [$s(x) = (1+e^{-x})^{-1}$] activations which require many calculations and are considerably slower while training. ReLU also benefits from being able to produce true zero values, resulting in representational sparsity, thus allowing the model to train using compact data. When comparing a CNN that uses rectified linear units against tanh units, there is an exponential speed increase in the training time for neural networks using ReLUs.

2.3 Batch Normalization Layer

Batch normalization layers are used to standardize the inputs from layer to layer within the CNN. Most deep neural networks rely on back propagation to update their estimate of error, this propagation starts from the output layer to the input layer. During updates the model assumes that the weights and parameters in all the previous layers remain constant during training, which is not true as the parameters are updated epoch to epoch. This slows down learning, as the model attempts to chase a moving target every epoch while the parameters simultaneously change. To help alleviate this issue, batch normalization is used. Batch Normalization standardizes the output from each layer, enabling the model to always have a standardized target, layer to layer and epoch to epoch; regardless of the point at which the model is in during training.

2.4 Pooling Layer

Pooling layers are used to to reduce the number of training parameters within a neural network. This reduction of parameters prevents the model from overfitting by reducing the spatial size of the network, and it also helps reduce the computational complexity of a network.

2.4.1 Max Pooling

Max pooling works by sliding a patch of size MxN over the feature map produced by the convolution and activation layer. The largest value within the patch is pooled down into a new compact feature map, this allows the model to considerably reduce the size of its network, while still maintaining the most prominent/strongest features from the image.

2.5 Dropout Layer

Dropout is a regularization technique that is used with deeply connected neural networks to reduce overfitting. During training, without intervention, a model starts to learn the noise within the dataset along with the features. This results in the model overfitting and an increase in the number of generalization errors as the model starts learning generalized features. To overcome this, the dropout method is used to drop some randomly selected nodes and their connected nodes from training. Using

dropout results in a reduction in the size of the neural network which makes it faster to train the model, and it also prevents overfitting which produces a much more accurate model.

3. Proposed Solution

In order to address the problem of classifying emotions we propose a deeply connected convolutional neural network using a sequential model. Our model is created using an encoding/decoding style of network, it is composed of four encoding and four decoding blocks, each of which consists of multiple hidden layers.

3.1 Encoding Blocks

The encoding blocks in our model follow some basic guidelines. The first layer within each block is a convolutional layer followed by an ReLU activation layer which creates a new feature map using the different sized filters. These layers are then followed by a combination of max pooling, batch normalization and dropout layers to form an encoding block.

The first encoding block starts with a convolutional layer which uses 32 filters, with a window size of 3x3 and a stride of only 1 pixel. As this is the first layer in the neural network the shape of the input for this layer is the same as the image size from the dataset. This layer is complemented with a ReLU activation layer, followed by a max pooling layer with a pool size of 2x2, and a batch normalization layer.

Similarly the second encoding block also starts with a convolutional layer and it uses 64 filters, with a window size of 3x3, followed by ReLU activation and max pooling. The third encoding block starts with another convolutional layer with 64 filters and a window size of 3x3 however, this layer is followed by a dropout layer with a dropout rate of 0.1. This is followed by max pooling and a normalization layer. Our last encoding block is also structured similarly to the other blocks, it starts with a 128 filter convolutional layer followed by a ReLU activation layer and a max pooling layer.

We utilised convolutional and ReLU activation layers at the start of each encoding block to create deeper feature maps every block, this allows the network to learn much more complex features. Moreover, we also used batch normalization in the first and third encoding blocks to ensure that we are normalizing our data every other layer without over generalization. In addition to that, we also used max pooling in each of the encoding blocks to ensure we aren't increasing the size of our network and/or the number of learning parameters during training. Combined with dropout, in the third block, this ensures that our model isn't overfitting by maintaining the size of the network.

3.2 Decoding Blocks

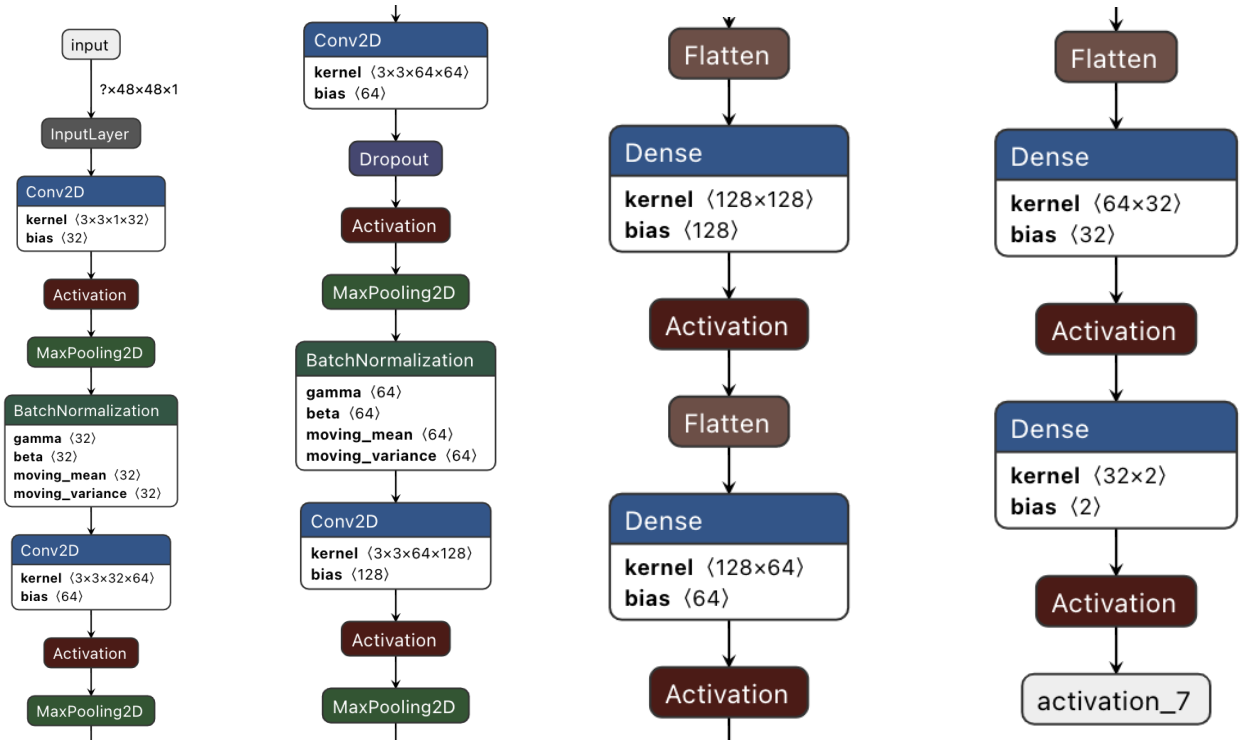
The decoding blocks in our model just like the encoding blocks follow a structure. Our four decoding blocks are made up of flatten, dense and activation layers.

The first decoding block flattens the output and produces a 128 dimensional matrix using the dense and ReLU activation layers. The second block similarly uses 64 units to produce a 64 dimensional matrix and the third layer uses 32 units to produce a 32 dimensional matrix. Lastly, the fourth decoding block uses a dense layer of 2 units to produce a two dimensional matrix followed by a softmax activation function.

The primary function of our decoding blocks is to reduce the parameters of the model down into the number of emotions we are trying to classify. Performing dense and activation functions in multiple blocks ensures that only the strongest parameters are passed onto the next layer without over generalizing. Furthermore, our last decoding block outputs a two dimensional matrix which can easily be used to classify between the two unique emotions. In addition, the last block uses the softmax activation function instead of ReLU and this was specifically selected, as it allows us to transform the unnormalized output of the fully connected last layer into a normalized output, as a probability distribution for each emotion.

3.3 Final Model

Our final model consists of all the decoding layers stacked after all the encoding layers in a linearly connected sequential model. The model trained over 50 epochs with a batch size of 32 images. Moreover, the model accepts images as input, and outputs a probability distribution for each of the two emotions: Happy and Neutral. The model was trained using a dataset with images of size 48x48 pixels as input and it trained on 156,706 trainable parameters.



4. Data

In this section, we describe the data we used to train our proposed model, its composition and how we augmented the dataset to improve our CNN.

4.1 Dataset

We train our model with, *Facial Expression Recognition Challenge 2013,* a dataset available from Kaggle. The original dataset consists of data for seven emotion classes, however, for our paper we have filtered out only the happy and neutral emotions.

4.1.1 Dataset Composition and Subsets

We will further breakdown the dataset and describe the different subsets of data. The dataset consists of three subsets: a training set, a test set and a validation set.

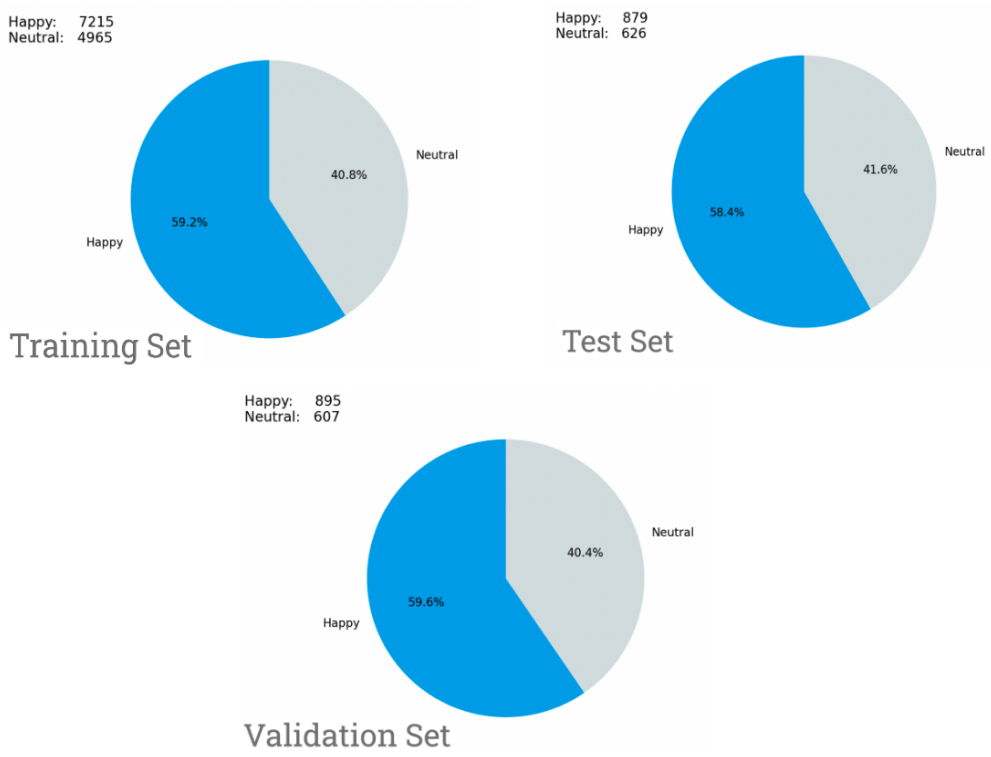
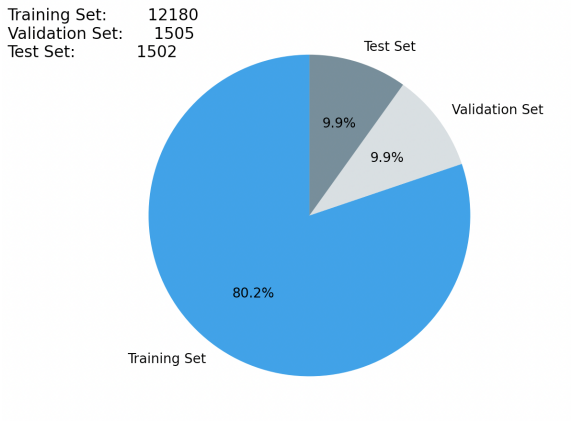


Figure 4.1

The first subset of data is the training set, this set contains the data that is used to train our neural network. This set contains 7215 and 4965 images for the happy and neutral emotions respectively, with a distribution of 59.2% and 40.8%.

Our second subset of data is the test set, this set contains data that is used during training after every epoch to test how the model is performing. It is used to measure the running validation accuracy and the validation loss during training. This subset has a distribution of 879 images for happy and 626 images for neutral emotion.

Our third and last subset of data is the validation set, this set contains data that will be used to validate the accuracy of the model. The data in this set is not seen by the model until after training, this is to ensure that there is no bias in the validation process. This set has a similar distribution as the test set, with 895 images for happy and 607 images for neutral.



The figure on the left shows the distribution of the three different sets of data. The training set contains a total of 12,180 images, the validation set contains 1505 images and the test set contains 1502 images. As evident from the pie chart we approximately have a 80/20 split for our training and validation + test set, which is pretty common and considered balanced.

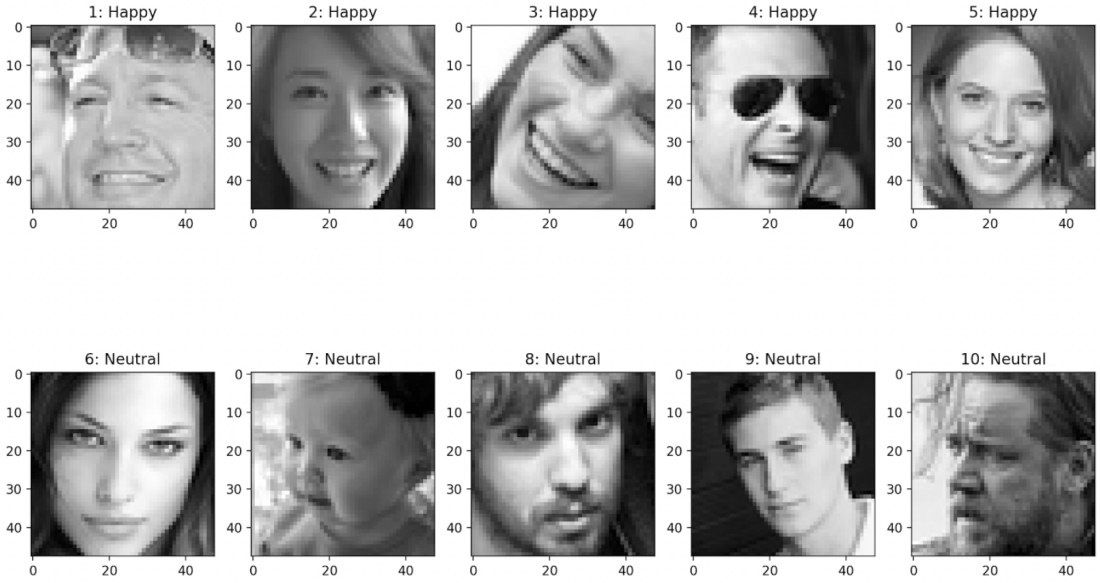


Figure 4.2

Figure 4.2 shows ten randomly sampled images from the training set, row one represents the happy emotion and row two represents the neutral emotion. In addition, as we can see from Figure 4.2, the images within the dataset are of a wide variety, with different backgrounds and points of interest.

4.2 Data Augmentation

Data augmentation is the process of augmenting or modifying the data artificially. All neural networks rely on data to train and learn, however, good data for training is hard to obtain. Datasets need to be varied, representative, and large in order to train a model that does not overfit. A simple and common way to reduce overfitting and meet all the characteristics for a good dataset is to artificially enlarge the dataset with transformations.

We have employed various data augmentation techniques on our data to enlarge it, some of these transformations include: image rotations, shear transformations, zoom, height and width shifts, and horizontal flips (mirroring). Augmentation allows us to increase the size of our training set and provides our model with a much varied set of data. Without this process, our model suffers from substantial overfitting, which would require us to use smaller neural networks and therefore learn more generalized features, resulting in decreased quality and accuracy of our model.

5. Results and Discussion

In this section, we present the results achieved from training our neural network model. The results we achieved show great outcomes with extremely high accuracy in predictions and classification.

5.1 Predictions

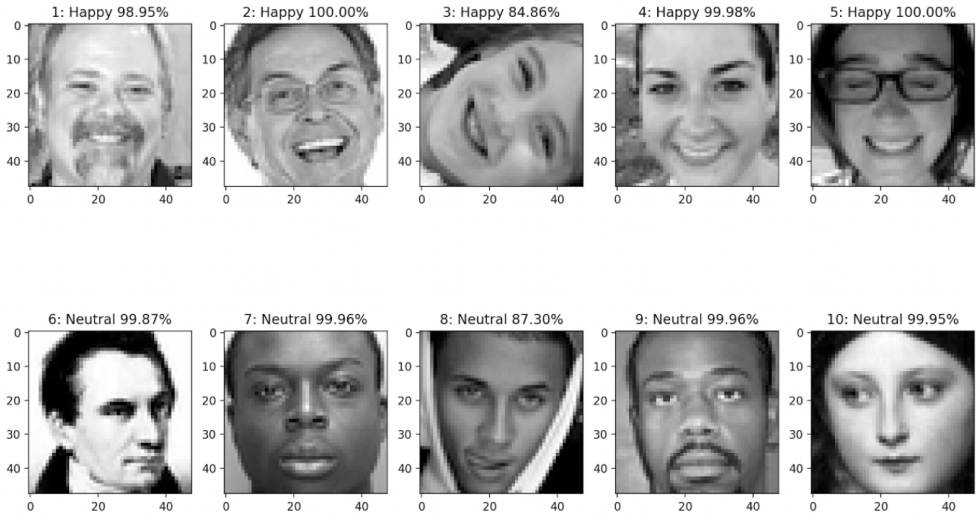


Figure 5.1 Predicted results

To test our model, we ran ten randomly selected images from the validation set against the model and recorded the results as emotion labels and prediction accuracy. Figure 5.1 shows the predicted results,

as we can observe from the images, the model was able to predict the emotion for each of the pictures correctly and with a very high accuracy, represented as a percentage on top of the images. Furthermore, we can also observe that the third image in the first row of Figure 5.1 is rotated by almost 90º however, our model was still able to predict the correct emotion with around 84% accuracy. This is possible due to the augmentation we added into our dataset earlier, which enabled the model to predict the emotion even when the picture was skewed.

5.2 Accuracy vs Loss

In deep learning, two of the most important metrics for measuring the performance are the accuracy and the loss. The accuracy of the model represents the correct predictions the model was able to make whereas, the loss represents errors that the model made during predictions.

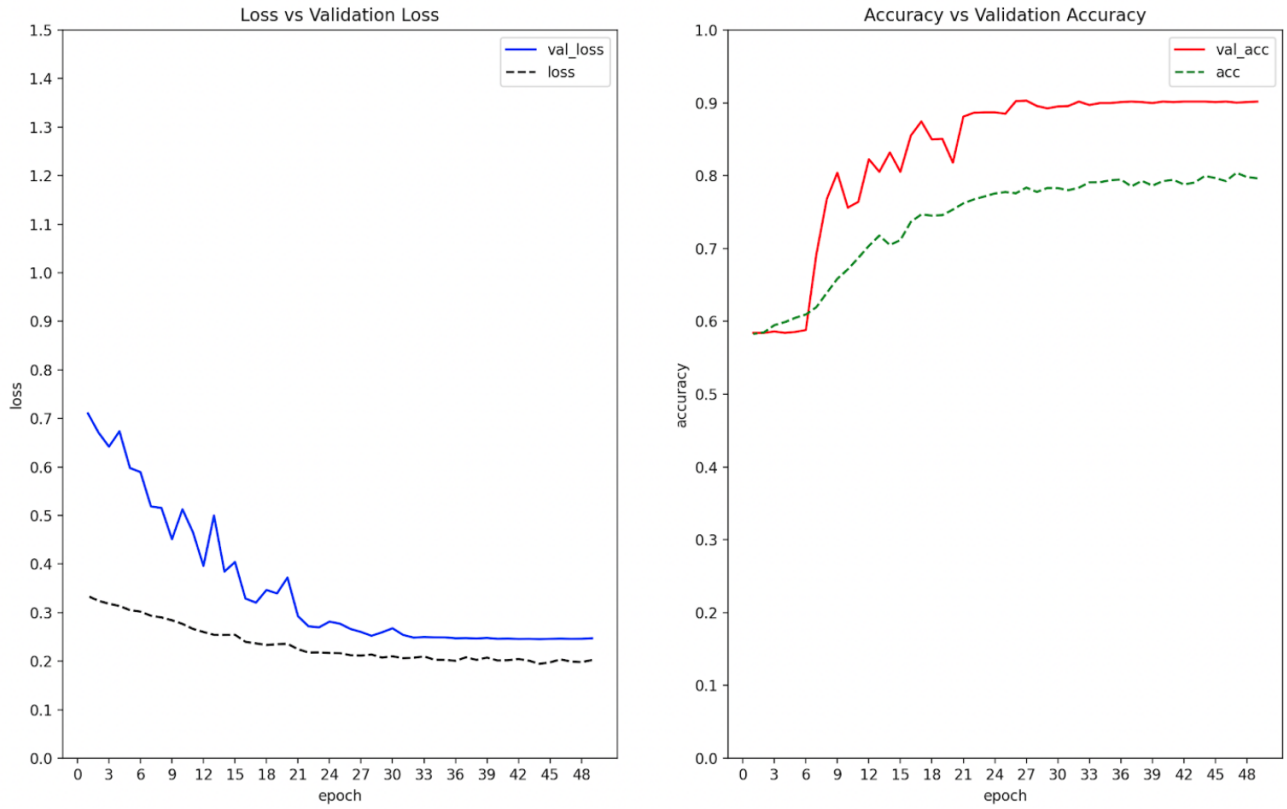


Figure 5.2

Figure 5.2 shows our model's accuracy and loss charts. The subplot on the left displays the Loss vs Validation Loss of the model during training for every epoch. On the other hand, the subplot of the right shows the Accuracy vs the Validation Accuracy of the model during training for every epoch. As observed, we can see that our model starts with a high cross-entropy loss of around 0.7. From epochs 17-20, our model's learning improved significantly; the errors and loss significantly dropped to a cross-entropy loss of approximately 0.3-0.4. Our model's learning stagnates towards the end and the validation loss bottoms out at around 0.30. Conversely, our accuracy started very low, the model's

prediction accuracy was under 60% during training in the first 5 epochs. Then, from epoch 6-9, the accuracy sharply increased and reached peaks of 80%. This upwards trend continues until epoch 27 -30, when the learning rate starts to reduce. The model's validation accuracy maxes out at 87% at the end of training.

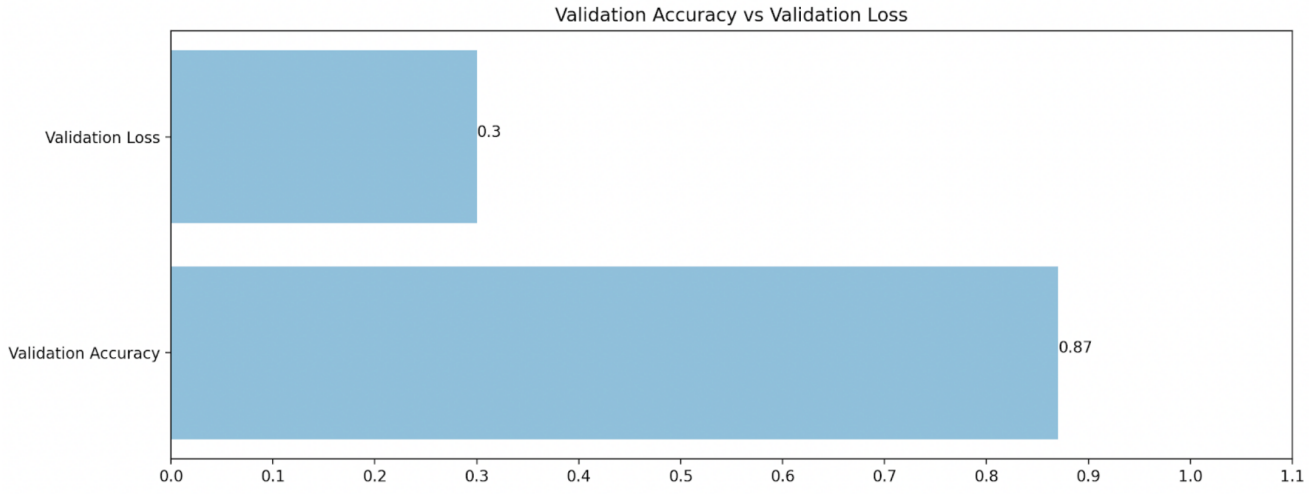


Figure 5.3

The bar graph in Figure 5.3 displays the validation accuracy beside the validation loss, which are ~0.87 & ~0.3 units, respectively; these results were found when running our trained model against the validation set. We find that our results on the validation set match those from the results obtained during training, this indicates that our model is not overfitting and is able to detect emotions correctly even on unseen data.

5.4 Drawbacks and Improvements

Our model is not perfect, it has shortcomings and lots of room for improvement. Our model can be improved to account for faces with facial hair, glasses or even blurrier images. We can do this by using a much larger and more varied dataset for training, larger datasets will allow the model to learn more. Our model can also be improved by reducing the validation loss or improving the accuracy.

6. Conclusion

In conclusion, our results show that a deep convolutional neural network model can be used to accurately and efficiently predict, with near certainty, between two unique emotions. Our CNN model was able to achieve an accuracy of ~87% when ran against unseen images. However, it's also important to note that our model is not perfect and has room for improvements, such as reducing loss, enlarging the dataset, training more without overfitting, etc.

Bibliography

1. Bengio, Y., Simard, P., & Frasconi, P. (1994, March). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157-166. 10.1109/72.279181

2. Brownlee, J. (2019, July 05). *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*. Machine Learning Mastery. Retrieved April 05, 2021, from https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

3. Brownlee, J. (2019, August 06). *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks*. Machine Learning Mastery. Retrieved April 05, 2021, from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

4. Brownlee, J. (2019, December 04). *A Gentle Introduction to Batch Normalization for Deep Neural Networks*. Machine Learning Mastery. Retrieved April 05, 2021, from https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

5. Brownlee, J. (2020, August 20). *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. Machine Learning Mastery. Retrieved April 05, 2021, from https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

6. Carrier, P.-L., & Courville, A. (2013, April 12). *The Facial Expression Recognition 2013 (FER-2013) Dataset*. Kaggle. Retrieved April 05, 2021, from https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

7. Chen, X., Yang, X., Wang, M., & Zou, J. (2017). Convolution neural network for automatic facial expression recognition. *2017 International Conference on Applied System Innovation (ICASI)*, 814-817. 10.1109/ICASI.2017.7988558

8. Li, S., Xing, J., Niu, Z., Shan, S., & Yan, S. (2015). Shape driven kernel adaptation in Convolutional Neural Network for robust facial trait recognition. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 222-230. 10.1109/CVPR.2015.7298618

9. Matsugu, M., Mori, K., Mitari, Y., & Keneda, Y. (2003). Facial expression recognition combined with robust face detection in a convolutional neural network. *Proceedings of the International Joint Conference on Neural Networks*, *3*, 2243-2246. 10.1109/IJCNN.2003.1223759

10. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. 10.1007/s10710-017-9314-z