Facial Sentiment Intensity Analysis

Andy Zhang, Dillion Verma, Yatin Kapur
{aszhang, dverma, ykapur}@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Introduction

"Read the room" is a common expression used during presentations and other social settings. Why is this so important? Reading a room, or understanding others' emotions, enables people to understand how to best interact with others and this is similarly applicable to computers who interact with humans.

Giving computers and AI the ability to "Read the room", or in this case, Facial Expression Recognition (FER), will open a path to opportunities which can be pursued to enhance human computer interactions (HCIs) such as video recruiting, entertainment, and medical robotics. We are interested in facial expressions because studies have shown that 55% of the overall impression of a person is made based on facial expressions (Wu, Lin, and Wei 2014).

Due to the advancement in recruiting practices and how screenings are now done via recorded video interviews, this makes an especially good case for an application which can use computer intervention and provide feedback on the reaction of candidates to certain questions. This can be used for time-series analysis of frame by frame analysis, and both applications will be productive in their own right. 62% of companies use video interviews and the wealth of data being collected can be leveraged with appropriate emotion recognition (Feffer 2019).

A new possible application of facial emotion recognition is for entertainment; for example, Apple's new Animoji app uses emotion detection to simplify the process of finding emoji icons. In this scenario, a user would make an expression to a camera, and the model would predict possible expressions the user is making and suggest emojis.

Being able to recognize human emotions allows better human computer interactions (HCI); one specific example is with medical robotics; by determining the facial expression of patients, medical robots can better adapt their actions. For example, if a patient is in pain, then the robot can halt or slow down their actions.

The purpose of the project is to build a model which best identifies the *intensity* of emotions.

Contributions

The specific research question that is being addressed is how to categorize facial emotion intensity of different head-shots of different people. The methodology being pursued to answer this question is to use supervised learning with deep convolutional neural networks (DCNN).

Previous research has mainly dealt with the classification of the actual emotions themselves while this study is attempting to classify the intensity of the emotions. The hypothesis is that by classifying the emotional intensity, one can increase the accuracy of models which classify the emotions themselves by introducing an extra feature which can be used by emotion classification models. In other words, by determining the emotional intensity of a facial emotion, one can better determine the actual emotion.

Furthermore, emotional expressions are also highly complex and not discrete; this study aims to helps further the current body of research on facial emotional detection by potentially allowing emotions to be characterized on a continuous and more nuanced scale.

The following highlights are the major contributions made by this paper towards the study of intensity of facial expressions in human portraits:

- Prediction of intensity largely depends on how exaggerated the mouth is
- RMSProp is the best optimizer for DCNNs with respect to facial analysis, this is also what was observed in previous studies
- Regardless of emotion, intensity can be observed on a spectrum instead of just grouping it into categories such as "low", "medium", or "high"

Related Work

After reading the work that has been done in this field of research, a few elements became apparent. Firstly, there is one standout method of conducting Facial Emotion Recognition (FER) analysis which yields the best results: Convolutional Neural Network (CNN). Decision trees and Feed Forward Neural Networks (FFNN) were also explored by researchers. The decision tree method and FFNNs showed an accuracy of 30.84% and 17.38%, respectively (Saravanan, Perichetla, and Gayathri 2019). Deep Convolutional Neural

Network (DCNNs), meanwhile, showed a remarkable accuracy up to 95.12% (Abdulsalam, Alhamdani, and Abdullah 2019). There are various factors which can greatly reduce the accuracy of the model outside of laboratory environments, such as illumination variation, head pose, and subject dependence (Samadiani et al. 2019).

A multitude of training methods were used by different researchers, such as RMSProp, Stochastic Gradient Descent, and Batch Gradient Descent.

Beyond quantifiable errors and statistics, the results actually tend to show some indication towards which emotions are more predictable. Happiness and surprised are two emotions which seem to be more predictable than others, this is due to how exaggerated those expressions are when compared to expressions such as neutral, contempt or pride. This is reflected in the test results in many studies; emotions such as neutral, contempt, and pride tend to be less accurate than more exaggerated expressions, such as anger and disgust (Wingenbach, Ashwin, and Brosnan 2016), (SÖNMEZ 2018).

While past researchers have focused mainly on classifying emotions, the goal of this project is to develop a model which best identifies the intensity of emotions. By distinguishing different levels of emotional intensity, one can better address more ambiguous facial expressions, such as neutral and pride; for example, one can first classify the intensity of the emotion, then classify the emotion based on the intensity.

Methodology

The goal is to classify emotions into five intensities on a scale of 1 to 5 with 5 being most intense, using a deep convolutional neural network (DCNN).

Evaluation Method

The problem of recognizing facial emotional intensities is being addressed in this study as a supervised learning problem. Hence, the first step needed to tackle this problem is to collect a labelled dataset of human faces and their accompanying emotions. According to Li and Deng (Li and Deng 2018), who gathered information on multiple public datasets for this domain. Many data sets such as the FER-2013 which classify six expressions in addition to neutral, are publicly available.

The most promising dataset, however, is the Amsterdam Dynamic Facial Expression Set Bath Intensity Variations (ADFES-BIV) which is composed of 1 second long videos of people making facial expressions. ADFES-BIV was chosen as it contains information about emotional intensity of an emotion using three separate videos - low, medium and high intensity. There has also been studies (Abdulsalam, Alhamdani, and Abdullah 2019) where the model was trained on the ADFES-BIV data with great success.

The ADFES-BIV is a companion data set of the original ADFES data set.

Both the ADFES-BIV data set (Wingenbach, Ashwin, and Brosnan 2016) and the ADFES data set (Van Der Schalk et al. 2011) have been validated in various studies. Access to both datasets were granted from the University of Amsterdam.

When the ADFES-BIV data was accessed, the low and medium intensity emotion videos don't show much difference from one another, so for the purposes of this study, a set of five frames will be looked at from various stages in the high intensity video in order to capture these emotions at different stages. The videos of high emotional intensity demonstrate each model's facial expression changing from neutral to high intensity and thus captures the entire emotional spectrum. In other words, the intensity of an emotion at frame 13 will not be as high as the intensity at frame 25.

The ADFES dataset was not used because there were variations of data which does not allow for ease of processing, such as people turning their heads and inconsistent levels of emotional intensity amongst the models.

As mentioned, these videos will be split into frames, which will be processed and fed into the DCNN. The data will be split into training, validation, and testing sets; the model training will be conducted over the training data, the hyperparameters will be tuned over the validation data, and the overall effectiveness of the model will be tested with the test data. The data must be partitioned into the three sets before being split into frames, as data might bleed if different frames from the same video ended up in both the training and validation or test sets.

The first step of this research is to label the images in the dataset with their respective intensities. To achieve this, a scale was developed from 1 to 5, resulting in 5 different intensities. Since every video starts off with an expressionless face escalating to an emotional face, each video was then split into frames and labeled with their respective intensities. In the dataset provided, every video was exactly 25 frames. The chosen frames from each video which correspond to intensities are shown below.

$$Frames = [13, 19, 20, 22, 25]$$

These frames were chosen since they showed the greatest change in emotional intensity among all the videos. Using this method on the ADFES-BIV data set yielded a total of 540 photos, 450 of which are used in training, 45 of which are used in validation, and 45 of which are used in testing.

The second step of this research is data normalization. Normalization in this case entails that the faces are correctly centered and all images have the same dimensions and colors.

In order to achieve this, the first step is to detect a face in the image and crop the image to a 32*32 square with the face centered in it. A size of 32 by 32 pixels was chosen as a hyperparameter after testing a few other sizes since this size clearly depicted the main facial expressions while maintaining a fast model training time.

To achieve face detection, the MTCNN face detector was used. Based on preliminary research and an exploration into the fastest and most accurate face detection algorithms, MTCNN seemed to perform the best in the shortest amount of time. Based on some benchmarks, it was able to detect a face in a 460x259 pixel image in a mere 0.118 seconds. One issue however was that the bounding box returned from this

algorithm has a rectangular shape. Hence, the bounding box was modified such that the side with the longest length was shortened to the size of the shorter length by decreasing both edges of the longer side by half of the difference between the longer and shorter side. This ensures the bounding box is centered on the face. Once there is a bounding box for the image, the image is cropped and scaled down to 32*32 pixels.

Lastly, the training data is converted from its RGB channels to grayscale with a pixel value between 0 and 255. The purpose of this transformation is that the chosen problem does not require us to analyze colors and the extra two channels would increase model training time. A simple linear transformation function is used to achieve this conversion. The most standard methodology is the ITU-R BT.601 conversion algorithm. This algorithm is presently used in standard-definition televisions and uses the following equation under the hood where R, G and B are the corresponding color channel pixel values:

$$L = R * 0.299 + G * 0.587 + B * 0.114$$

Algorithms

The algorithm step is to actually build the convolutional neural network. The library used to conduct this research is TensorFlow2.0 (TF) with Keras. This is an extensive library which is popular for it's use cases in deep learning, which aligns well with this research study in mind. TF provides an interface to interact with the CNN model in python, which provides the ability to add layers, pools, specify densities, and use any activation function of choice.

Since this exact research topic has not been undertaken before, coming up with a CNN for this project is very much an experiment as much as it is an analytical task. Thus, to determine the structure and hyperparameters of this neural network, several tests will be carried out to see which set of hyperparameter values yields an acceptable validation accuracy.

Since to inputs to the net is images, 2D convolutional layers (Convolutional2D) will be used. In order to determine the best number of Convolutional2D layers, an experiment will be carried out with a range of values. The proposed set of values are from one to nine Convolutional2D layers. For each of them, the input shape will remain the same, at (None, 32, 32, 1), and the other hyperparameters will also stay constant. Given that the input is an image of a face, nine layers should be more than enough to isolate the necessary features to get an accurate model to represent the different intensities of the parts of a face.

The activation function of choice for each hidden node in the net is the Rectified Linear Unit (ReLu) function namely due to its ease of computation and speed in practice. Theoretically this would help produce outputs faster than using the sigmoid or similar activation functions.

The next steps are to determine the remaining hyperparameters including epochs, batch size, dropout, filters, the optimizer, and the optimizer's learning rates.

The goal of tuning the number of epochs used is to arrive at a number after which there is not a significant betterment in the model's validation accuracy. This goes hand-in-hand with the learning rates; as learning rates decrease, the model must take a longer time to learn this information, which translates to a need of higher epochs. Given that this is a small dataset, the batch sizes to test will range from 10 to 90, and this will be done in conjunction with epoch values ranging from 10 to 60.

As for the optimizer, there are a few optimizer's which are suited to this type of a classification task. Namely, the following will be explored:

- **Stochastic Gradient Descent (SGD)**; A commonly used method to optimize an objective function; potential learning rates: 0.001, 0.005, 0.01, 0.05
- **Adaptive Gradient Descent Algorithm (Adagrad)**: Eliminates the need to manually tune the learning rate, well suited when there is sparse data as in large scale neural network; potential learning rates: 0.01, 0.05, 0.1, 0.5
- **Root Mean Square Propagation (RMSProp)**: Similar to Adagrad, but it utilizes the magnitude of the recent gradient descents to normalize the gradient; potential learning rates: 0.001, 0.005, 0.01, 0.05

Given that this is a multi-class classification problem, a good candidate for the loss function is the Sparse Categorical Cross-Entropy loss; which is typically used for problems which classify in more than two classes. The classes are the range of emotions, ranging from 1-5 on a scale which says that 1 is lowest intensity, and 5 is the highest. As is the case with the algorithm, this loss function can also be used directly through the TensorFlow2.0 library, and does not require re-implementation.

Dropout regularizes a neural network and prevents overfitting. Different positions and thresholds of dropout will be tested to find the optimal validation result.

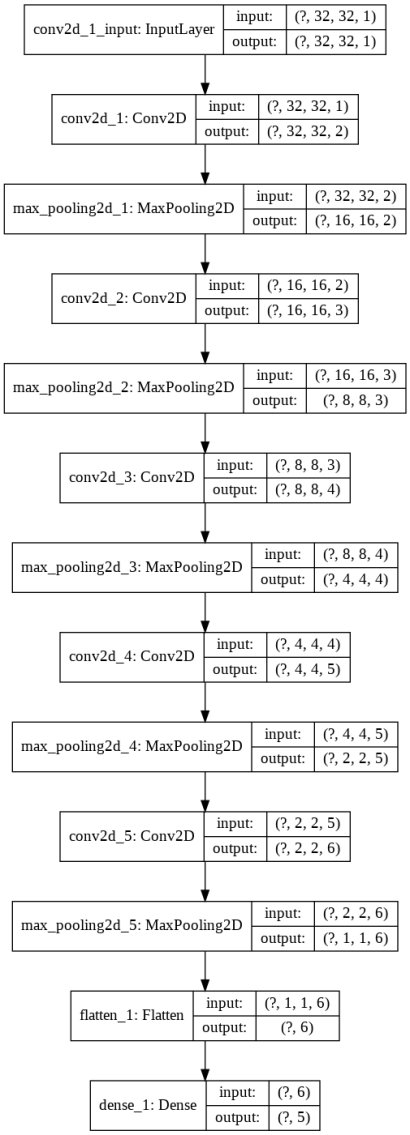
Another essential part of the network is the configuration of the numbers of filters applied to each layer. The number of filters represents essentially the number of feature detectors for a CNN, much like the number of neurons of a traditional neural network; a feature map is generated by applying filters on an input. Features can include lines, edges, and other factors of an image. The larger the filter being employed in a layer, the fewer filters will be used and the more abstracted the feature being mapped is. In other words, the smaller the filter is, the more granular the feature that it is detecting. The size of filters will be altered and tested as hyperparameters. The size of the filter will be strictly increasing as one goes deeper in a CNN, as the deeper layers detect more abstracted features (and eventually the entire image will be detected on the last layer). The experiment will test for two features of filtering; the first is to test for the increment to the size of the filter, how much the size of a filter should increase after each layer, and the second is to test for the size of the first filter. The first test will test values 1 to 5 and the second test will test values 2 to 5 (since a filter of size 1 does not detect any novel features).

The optimal network from the experiments is a network with the following properties:

- filters in n^{th} layer = $n + 1$

- layers = 5
- optimizer = RMSProp
- learning rate = 0.001
- epoch = 50
- batch size = 75
- no dropout

Figure 1: Full Neural Network



Results

For the experiments, different combinations of hyperparameters were experimented upon to determine the best resulting model in the time given. The list of hyperparameters being tested are as follows:

- number of layers
- filter size in first layer
- optimizer
- learning rate
- increment for filters in each layer
- batch size
- epoch
- dropout

It is important to note that the reasoning behind the number of layers being tested first is to get a better sense of structure of the neural network and to reduce the amount of technical tinkering that would be needed later on in order to find the ideal fit. The other hyperparameters are most likely less impactful than the number of layers.

That said, learning rates and optimizers are two of the most important parameters for a neural network, and tuning them before the others also saves a lot of experimentation later in the research (Greff et al. 2016). As a result, these were chosen to be tuned before the other parameters. The order of the other parameters is really subjective and it depends on the particular practitioner what order they want to use. The previous research shows that there are slight differences in results when the order of testing them is changed. So accordingly, learning rates and the optimizer are chosen second, then the other hyperparameters such as epochs and batch size are tuned.

Layers

First, the optimal number of convolutional layers was tested; for each number of layers, all other hyperparameters were kept constant. Each Convolutional2D layer has kernel size 2, the 'same' padding, and 'ReLu' activation (described further in Keras documentation).

Tuning a parameter such as the number of layers used in a CNN is a task that cannot be analytically answered using a set method. Given that no one has looked at this exact research problem before, it is hard to get ideas off of fundamental research about what the ideal number of layers is. The approach taken to overcome this problem is to start with one layer, and to continuously add layers one at a time.

- epochs = 50
- batch size = 45
- filters in n^{th} layer = $4n$
- optimizer = RMSProp
- learning rate = 0.001

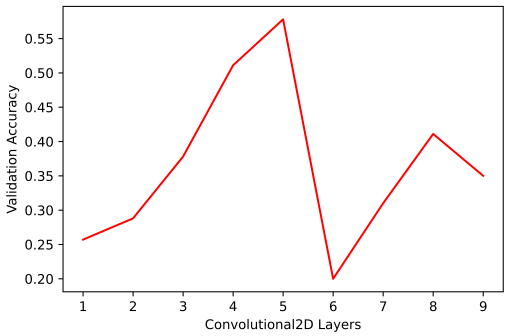


Figure 2: How different Convolutional2D layers impact validation accuracy

From the results, it appears that the best number of layers to use for this problem is 5. Up until 5, the validation accuracy increases as new layers are added, but there is a steep drop-off when the 6th layer is added. The model is most likely underfitting before the 5th layer and overfitting after the 5th layer. In other words, going up to 5 layers, each additional layer allowed the network to pick up more details, but after the 5th layer, the network was picking up details that were too specific to the training data, which means it likely did not generalize well with the validation data.

Optimizers and Learning Rates

At this point in the experiments, the best model currently has five layers. As listed in the algorithms section, there was a choice between three optimizers, and the learning rates for each of them. Keeping the epochs at 50, the batch size at 45, and the number of filters in each layer increments by 4, so that the 5th layer has 20 filters. SGD (Badrinarayanan, Kendall, and Cipolla 2017), Adagrad (Bera and Shrivastava 2020), and (Howard et al. 2017) were chosen based on their effective prior use in research involving CNNs and images. The following are the results of tests conducted on three optimizers with the above kept constant:

The model structure for this test is:

- epochs = 50
- batch size = 45
- filters in n^{th} layer = $4n$
- layers = 5

Optimizer	Learning Rate	Validation Accuracy
SGD	0.001	0.1111
SGD	0.05	0.3556
SGD	0.1	0.4222
SGD	0.5	0.4000
Adagrad	0.001	0.1111
Adagrad	0.005	0.1333
Adagrad	0.01	0.2667
Adagrad	0.05	0.4444
RMSProp	**0.001**	**0.5778**
RMSProp	0.005	0.2000
RMSProp	0.01	0.0000
RMSProp	0.05	0.0000

Table 1: Exploring different optimizers and learning rates.

Based on the results above, the RMSProp optimizer seemed to yield the highest accuracy with a learning rate of 0.001.

SGD is an iterative method for optimizing an objective function. It appears that as the learning rate increases, the validation accuracy initially increases, then decreases.

Adagrad is an algorithm designed to adapt the learning rate based on the parameters; it seeks to perform smaller updates for parameters associated with frequently occurring features and larger updates for parameters associated with infrequent features (Duchi, Hazan, and Singer 2011). The accuracy for Adagrad was strictly increasing as the learning rate increased.

RMSProp is a variation of Adagrad that seeks to address the problem of radically diminishing learning rates. The validation accuracy for RMSProp was strictly descending as the learning rate rose, to the point where the validation accuracy was 0 for learning rates of 0.01 and up. RMSProp coupled with a learning rate of 0.001, however, performed the best and thus will be used in the model.

Filters

The next hyperparameter which was tested is the configuration of the number of filters at each layer in the network. The filters worked best when the number of filters increased after each layer (the first layer has filter size x and the subsequent layer has filter size y, where y $>$ x), as previously mentioned in the algorithm section.

The model structure for this test is:

- epochs = 50
- batch size = 45
- layers = 5
- optimizer = RMSProp
- learning rate = 0.001

Increment for filters in each layer	Validation Accuracy
1	**0.5556**
2	0.0889
3	0.4222
4	**0.5778**
5	0.2889

Table 2: Number of filters compared with validation accuracy.

The increment for filters in each layer represents how much abstraction occurs at each layer. If there is too much abstraction (larger increment for filters in each layer), then the images are not detailed enough to be abstracted to such a degree. If there is not enough abstraction (smaller increment for filters in each layer), then not all of the features of the image will be accounted for by the model.

Since both incrementing by 1 and 4 yielded very similar results, both scenarios will be tested.

Layer 1 Filters	Increment	Validation Accuracy
2	**1**	**0.8889**
2	4	0.2000
3	1	0.3333
3	4	0.6222
4	1	0.5556
4	4	0.5778
5	1	0.4889
5	4	0.4000

Table 3: Number of filters in first layer with increments of 1 and 4.

The best configuration for filters is to start with filters of size 2 in the first layer and increment by 1 in the subsequent layers. Mathematically, $n + 1$ filters for the n^{th} layer.

The likely reason that having filter size of 2 in the first layer yielded the best results is because the network must pick up granular details first, before abstracting to less-granular details from said granular details in later layers. If the first filter is too large, then the granular details will not be picked up by the model, and hence abstraction will not be possible (since there are no granular details to abstract from). For initial filter sizes of larger than 2, the model is most likely not picking up the most granular details; the model is probably abstracting too much.

Batch Size and Epoch

The next hyperparameter which was tested was batch size. Batch size determines how many images are processed before network weights are adjusted again.

The model structure for this test is:

- filters in n^{th} layer = $n + 1$
- layers = 5
- optimizer = RMSProp
- learning rate = 0.001
- epoch = 50

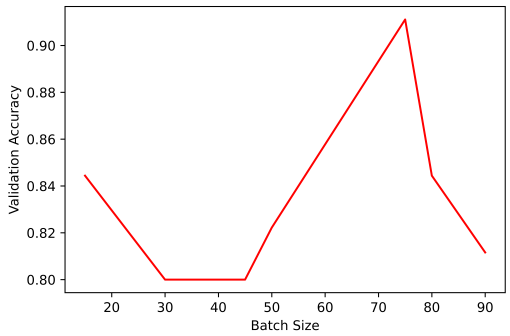


Figure 3: Trend in Batch Size and Validation Accuracy

The validation accuracy is greatest at a batch size of 75, so that was chosen as the value for this parameter in this model.

From the results, it becomes clear that as the batch size increases, the validation accuracy generally increased and then dropped after 75. Two things to note is that the model performed better when the number of training examples was divisible by batch size (this may just be a coincidence), and the smaller the batch size, the more the model is altered each iteration of the epoch relative to the amount of data used. This likely means that for batch sizes less than 75, the model was most likely being overly affected by training examples which cannot be generalized; for batch sizes larger than 75, the model was most likely not being trained enough with the data.

With batch size = 75:

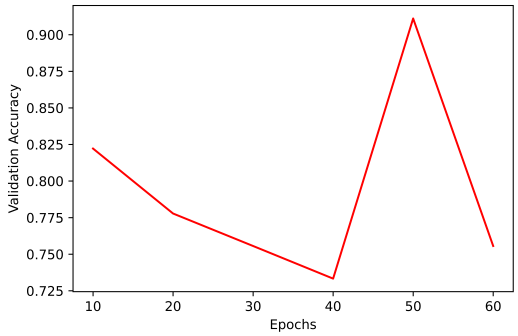


Figure 4: Trend in Epochs and Validation Accuracy

Here, 50 epochs appears to give the highest validation accuracy. There appears to be an upwards trend on validation accuracy as the number of epochs increases; there is a sharp rise in accuracy when the epoch size is 50, then drops drastically afterwards. The number of epoch is a proxy for how much the model was being trained; its likely that for epoch sizes of less than 50, the model was not being trained enough (underfitting) and for epoch sizes of larger than 50, the model was being trained too much (overfitting).

Dropout Threshold

The model structure for this test is:

- filters in n^{th} layer = $n + 1$
- layers = 5
- optimizer = RMSProp
- learning rate = 0.001
- epoch = 50
- batch size = 75

Dropouts at layer (1,2,3,4,5)	Validation Accuracy
(0.00, 0.00, 0.00, 0.00, 0.00)	**0.9111**
(0.00, **0.10**, 0.00, 0.00, 0.00)	0.8000
(0.00, **0.10, 0.10**, 0.00, 0.00)	0.0222
(0.00, **0.10, 0.10, 0.10,** 0.00)	0.1556
(0.00, **0.10, 0.10, 0.10, 0.10**)	0.2222
(0.00, **0.01**, 0.00, 0.00, 0.00)	0.8333
(0.00, **0.01, 0.01**, 0.00, 0.00)	0.0000
(0.00, **0.01, 0.01, 0.01**, 0.00)	0.3556
(0.00, **0.01, 0.01, 0.01, 0.01**)	0.3333

Table 4: Exploring dropouts for overfitting.

From the trials, it is clear that dropout is not necessary for this network, as dropout consistently lowers the validation accuracy.

In particular, the more dropout layers which are used and the higher the thresholds, the worse the validation accuracy.

This is thus indicating that the model most likely does not overfit.

The optimal model is as follows:

- filters in n^{th} layer = $n + 1$
- layers = 5
- optimizer = RMSProp

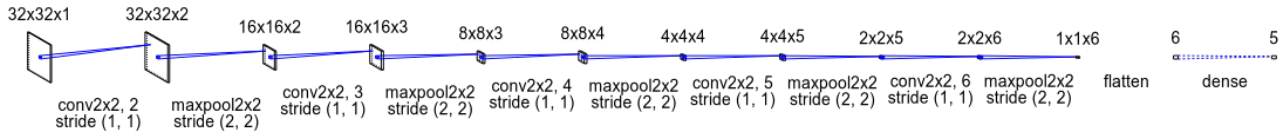


Figure 5: Final model

- learning rate = 0.001
- epoch = 50
- batch size = 75
- no dropout

The accuracy of the model on test data was 86%.

Prediction Example

To illustrate an example, let's walk through how the model processes the image and gives an output.

Figure 6: Initial photo

Figure 7: Processed photo

The model predicts the following probabilities:

Intensity	Model Probability
1	0.20986344
2	0.17544739
3	0.19927052
4	**0.26161286**
5	0.15380572

Table 5: Predicting on real data.

As can be seen, the model fairly accurately predicts that this is likely an intense emotion (highest probability on level 4 intensity). It is interesting to note, however, that the model was not clearly deterministic. There was no clear distinction; this is most likely due to the contradictory facial features. For instance, the mouth is not exaggerated, which on its own, may suggest that the emotion is more neutral.

Conclusion and Future Work

This research project came into fruition because of the question of how AI can become more human-like; more specifically, how AI can view and interpret the world as a human agent. The chosen area of research was in human computer interaction (HCI), more specifically emotional analysis. All previous research on this topic is on identifying the actual emotions themselves. It was also noticed that in many of the prior research reports, the results for more neutral expressions (such as contempt or pride) tend to be less accurate than those for the more intense expressions (such as anger or surprise). The motivation behind this research paper was to determine the emotional intensity of the emotions as opposed to the actual emotions themselves. More specifically, this research aimed to classify emotions along 5 intensity levels; this can be used to enhance regular emotional classification by acting as an extra input factor, or as an extra dimension of analysis. To that end, this research was able to determine a tuned model which accomplishes the feat of detecting facial emotional intensity.

First, numerous facial expression datasets were evaluated and narrowed down to two datasets: the ADFES and the ADFES-BIV. Next, the data was manually inspected for any faults or errors and the training process was constructed. Afterwards, all data was preprocessed using numerous transformations on the raw data set such as selecting key frames, cropping images to faces only, reducing RGB channels to grayscale and converting to tensors. Finally, a simple deep convolutional neural network (DCNN) was created and trained on the dataset. The number of layers and hyperparameters of the network were all chosen using the scientific process and quantitative decision making.

The main result was a network with a validation accuracy of 91% and a prediction accuracy of 86%. Many of the hyperparameters did what one might expect; for instance, adding more layers caused the model to overfit and starting with a filter size that was too large in the first layer caused the model to over-abstract and miss the more granular details. There were some other observations which are more difficult to explain and warrant further research; for instance, the model performed better when the number of training examples was divisible by the batch size. This can be a coincidence or it can be a real pattern. More work can also be done to fully understand the impact of different optimizers and learning rates.

There are several ways this work can be improved upon

in future studies. First of all, the dataset which was used in this research was fairly small and did not have much variation. There were only around twenty-four models per each emotion, evenly split between males and females, and there were only twelve emotions. Clearly, such a small subset cannot completely encompass the vast variety of human emotions produced by all humans in the world. Furthermore, the dataset contained mainly European models; the ensuing CNN may not generalize well amongst other races. One way to approach this problem is to find and use additional rich and diverse datasets.

Secondly, the data pre-processing was not exhaustive and could be vastly improved for better and unbiased results. The data pre-processing used for this study solely consisted of cropping an image onto a subject's face, and applying a series of transformations on it to convert it into a tensor. In future work, it would be helpful to apply a series of random rotations, flips on vertical axis, and slightly different cropping to generate an exponentially larger dataset with different variations of faces. This would help the network better adapt to new faces and predict untrained data with less bias.

Another very important factor affecting the accuracy of results are the hyperparameters. Firstly, different orders of hyperparameter testing should be tried. There may be better results if hyperparameters were experimented in different orders; it is possible that the experiments simply stumbled on a local optima and not the global optima. Secondly, as previously mentioned, more work can be done to investigate more optimizers and learning rates. While the current experiment finds a suitable optimizer and learning rate, there are various other optimizers which have been used with CNNs on images, such as Adam, which were not explored. Furthermore, more learning rates should have been tested to get insights on the trends and reasoning behind the performance of different optimizers. Lastly, the variation of the size of each layer in the neural net is presently uniform, but it can be further tuned so that the pooling and convolutional layers do not follow a pattern; this can potentially further optimize the network.

References

Abdulsalam, W. H.; Alhamdani, R. S.; and Abdullah, M. N. 2019. Facial emotion recognition from videos using deep convolutional neural networks. *International Journal of Machine Learning and Computing* 9(1):6.

Badrinarayanan, V.; Kendall, A.; and Cipolla, R. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 39(12):2481–2495.

Bera, S., and Shrivastava, V. K. 2020. Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing* 41(7):2664–2683.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12(7).

Feffer, M. 2019. How video interviews are steamlining job applications. *insights.dice.com*.

Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; and Schmidhuber, J. 2016. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems* 28(10):2222–2232.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Li, S., and Deng, W. 2018. Deep facial expression recognition: A survey. arxiv 2018. *arXiv preprint arXiv:1804.08348*.

Samadiani, N.; Huang, G.; Cai, B.; Luo, W.; Chi, C.-H.; Xiang, Y.; and He, J. 2019. A review on automatic facial expression recognition systems assisted by multimodal sensor data. *Sensors* 19(8):1863.

Saravanan, A.; Perichetla, G.; and Gayathri, D. K. 2019. Facial emotion recognition using convolutional neural networks. *arXiv preprint arXiv:1910.05602*.

SÖNMEZ, E. B. 2018. An automatic multilevel facial expression recognition system. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi* 22(1):160–165.

Van Der Schalk, J.; Hawk, S. T.; Fischer, A. H.; and Doosje, B. 2011. Moving faces, looking places: validation of the amsterdam dynamic facial expression set (adfes). *Emotion* 11(4):907.

Wingenbach, T. S.; Ashwin, C.; and Brosnan, M. 2016. Validation of the amsterdam dynamic facial expression set–bath intensity variations (adfes-biv): A set of videos expressing low, intermediate, and high intensity emotions. *PloS one* 11(1).

Wu, C.-H.; Lin, J.-C.; and Wei, W.-L. 2014. Survey on audiovisual emotion recognition: databases, features, and data fusion strategies. *APSIPA Transactions on Signal and Information Processing* 3:e12.