Explanation of a Neural Network Model for Emotion Detection Based on Facial Expressions

Abstract

In scientifically and industrially focused areas, there's an increasing attention on work to detect feelings or emotions from facial expressions using neural networks. Human emotions displayed on one's face are important in a number of areas, including human-machine interfaces, highly sensitive social inferences, psychological analyses, and security applications.

Through automatic recognition and interpretation of facial expressions, people can be more aware of their own communications and other people's cues by enabling the devices employed (for instance, automated systems) not only to understand human emotions but also to react instinctively. Additionally, recognizing face expression and analyzing it can be of value in assessing and diagnosing mental illness.

Using neural networks are a popular choice in this field and it is considered to be a cutting-edge technology that will rapidly change in the future. This somewhat new technology helps machines to learn things without human supervision and they can recognize complex patterns and features in the large amount of data that human sometimes can't handle well. Research, and development in this field have high significance for further advancing the recognition and interpretation of human emotions.

I choose this topic for my independent laboratory work because I find the topic interesting. I focused on recognizing 7 or 8 basic facial expressions, depending on the available dataset. I worked with numerous datasets, as some emotion categories had disproportionately low data, significantly affecting the model's performance. Nevertheless, I trained the network with some unbalanced datasets to examine how to increase the model's robustness.

Chapter 1

Introduction

Facial recognition with machine learning is certainly a huge research topic. It has many potential applications in areas such as human-computer interaction, healthcare, etc. Being able to understand facial expressions can improve the user experience in any image-related application and can influence the human behavior. Researching facial expression recognition is really a lengthy habit. One exceptional act in this area comes from Paul Ekman's Facial Action Codification Structure (FACS). It gives a precise structure to the actions of facial parts movements, also known as action units (AUs), and is designed for selected emotional varieties.

With the help of the computer vision and machine learning techniques it has made a significant progress facial expression recognition by enabling automated and real-time analysis of facial expressions from images or videos. Traditional methods often relied on custom algorithms and different rules, which were limited in the generalization ability

In this field, in recent decades, deep learning techniques such as convolutional neural networks (CNNs) has demonstrated outstanding performance. CNNs are able to automatically learn spatial and temporal data such as raw pixel data. This is a huge upgrade on what neural networks are capable of because this make more robusts . For example CNN-Cascade, a popular architecture, and models such as Residual Network (ResNet) have been developed. Their performance on simple facial datasets like CK+, FER2013, etc. has reached a new stage. Two-dimensional convolution neural network model for facial expression recognition has some challenges that need addressing including how to deal with pose variability, changes in illumination, occlusion, and individual differences of facial appearance. These challenges are answered by researchers through applying new augmentation techniques, inject spatial and temporal information directly into networks (see CNN-Cascade), and use generative adversarial network (GAN) to amend any problem.

In general, there is a great deal of untapped potential for facial expression recognition in applications, especially with machine learning methods. Until now, our research has been mainly concerned with making facial recognition accuracy and fairness systems achieve the potential of this technology; in addition to this flavor bringing about political repercussions, it must also deal with ethical questions raised by society.

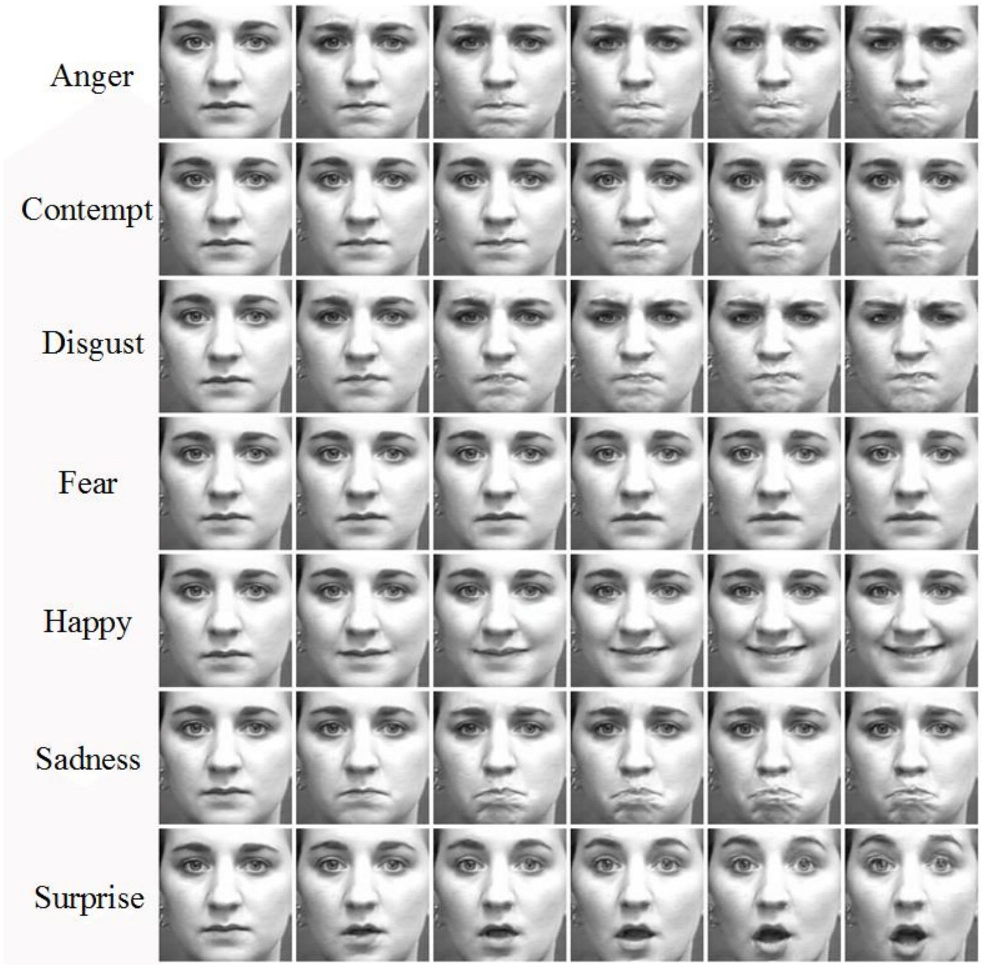


Figure 1: These are the basic emotions that are recognized universally around the world. Anger, Contempt, Disgust, Fear, Happy, Sadness, Surprise. The Neutral is considered a basic emotion as well, but it isn't included in the picture. Studies show we're better at picking up on happiness, sadness, and anger. Emotions like fear, surprise, and disgust can be more subtle and easily confused. Overall, while humans are good at emotional recognition, but they are not perfect. Accuracy can range from around 90%.

Chapter 2

Background

In recent years the advance of the of big data has led machine learning in facial expression recognition to advance rapidly as well and by now offer certain standard features comprehensive better than older conventional methods would be able to achieve Thus machine learning as a whole has transformed expression recognition on the face, leading to more accurate and efficient solutions that can be easily scaled There is no doubt that continuing research and innovation in this field will bring about findings even more beneficial to mankind as well as man-machine interface experiments improved out of all recognition.

Overview of machine learning techniques

Neural networks

Being inspired by human brains, neural networks are computer models that loosely mimic the actual biology of the human brain because it can replicate the intelligence. The neurons consist of interconnected nodules, also called as neurons, which are stacked in layers on top of each other. These layers typically consist of an input layer, one or more hidden layers and an output layer.

Therefore, a single neuron is a building block in a neural network which do some simple math operations like multiplication, addition. Typically, a neural network is made up of multiple neurons organized in layers, but each neuron operates independently behind the scenes and its behavior largely depends on its inputs, weights and bias.

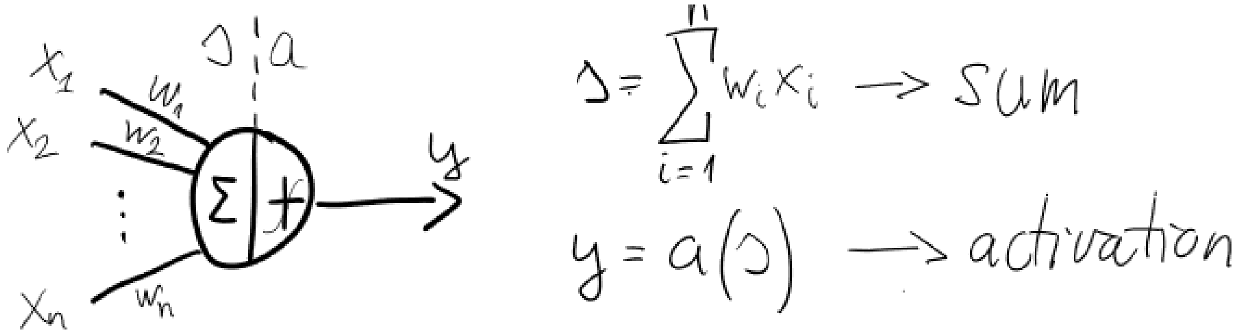

x_1
x_2
x_n
w_1
w_2
w_n
$s \mid a$
Σ
f
y
$s = \sum_{i=1}^{n} w_i x_i \rightarrow$ sum
$y = a(s) \rightarrow$ activation

Figure 2 This is how a single neuron looks like.

A single neuron consists of:

- Inputs (xi): They are represented by the arrows at left with Xn on them. Each Xn is a number. In reality, a single neuron can have many more inputs than those shown in this diagram.
- Weights (wi): They are denoted by the symbols Wi next to each input arrow. The weights have a tremendous impact on how much influence an individual input has over the neuron's output in other words the linkage between feed-forward neuron and its output.
- Sum: This is indicated by the sigma symbol (Σ). Each input is multiplied by its corresponding weight, and then all of these products are added up.
- Bias (b): The notion of this is b. The bias adds a constant value to the weighted sum of the inputs. The main purpose of that is that it allows to adjust the summed value a little bit if there is a large variance. It is not represented in the picture above but the bias is added after the sum operation.
- Activation function (a): The notion of activation function is a(). The activation function often applies a non-linear transformation to the weighted sum and bias but linear can be used also but it is not common nowdays. So this allows the neuron to learn complex patterns in the datas. The mathematical function used for the activation function can vary depending on the application.

Since in a neural network all neurons are interconnected, its architecture is complex and this is how it can learn from input data to make predictions. The connections between neurons allow information to flow through the network, with each neuron receiving input signals, performing computations, and then passing the result on to other neurons in subsequent layers.

Each neuron can't accomplish difficult tasks whether it be big or small; after all most as much effort is put into constructing and training the neural networks that surround them as they have originally. Of course, one neuron represents the most basic unit of operation in this instance, but it is only when many different neurons are working together that a neural network can undertake a variety of machine learning tasks effectively.

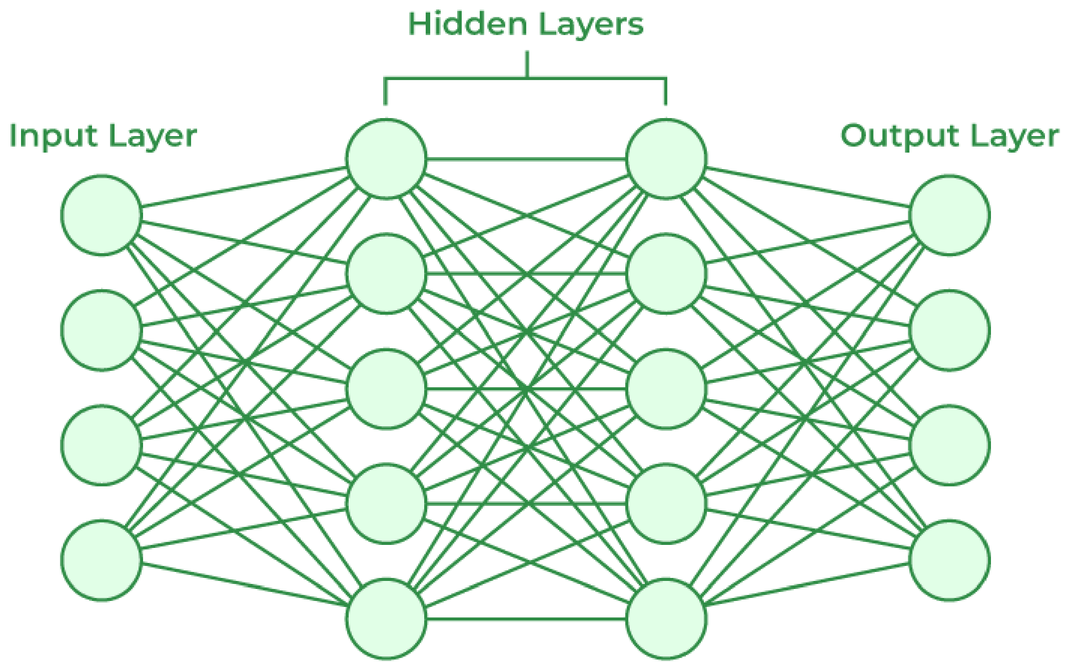


Figure 3 A fully connected neural network

Training phase

In the training process we use the train and the validation dataset to train the model. These two datasets are essential for the purpose of teaching the model and the quality of data has a great impact on this. The purpose of the train dataset is to learn to generalize patterns and relationships from the data, thus the model to make accurate predictions and classifications. The goal of the validation dataset is to see the model's performance on unseen data. The model is learning from the train data and makes a practice prediction on the validation dataset thus the model will generalize to new observations, and this will prepare the model for other kind of unseen data.

Epoch: One epoch is considered to be counted when the entire train dataset passes through the neural network. The train phase usually needs to iterate through a train and validation dataset for couple times so the great number amount of epoch is not uncommon. After one epoch, the model's parameters get updated by the backpropagation algorithm. After that if another epoch will need then it will calculate the dataset on different model weights and bias. The number of epochs is a hyperparameter that can be adjusted.

Batch: Training data sets in a batch refers to a portion of this data set. Besides updating the model parameters every single item in the training samples--which might not be appropriate for large data sets as mentioned above--it is more effective to use batches for running parallel computations on multi-core processors. Indeed, one of the primary purposes of using batches in training neural networks is to deal with datasets that are too large to fit all at once into memory. By dividing the dataset into smaller batches, only a fraction of the data actually needs to be loaded into memory each time. It becomes possible to train on data sets that would otherwise saturate the memory system. This technique makes it possible to use memory in a more efficient fashion and train on large-scale data without running out of memory.

Forward and backward propagation are the main techniques to train neural networks, especially in supervised learning.

Forward Propagation

When passing input data through the neural network for prediction creation or result generation is called forward propagation. In this procedure input values are multiplied with weights, bias are added and the activation function carried out at laid until output is finally obtained. From input to output, it lays out the path of information through the network; and with each layer´s activations becoming the input for layers that follow, you can begin to capture increasingly abstract features or patterns in your data using this technique.

Backward Propagation

After forward propagation, backward propagation calculates the gradient of the loss function by using the chain rule of calculus. With the calculated gradients the backpropagation will starts from the output layers till it reaches the input layers. It will make some adjustments for the weights and biases to minimize the loss function. The learning rate and the gradients the model will make changes to the right direction to get the optimal values.

Activation function:

Activation functions are usually non-linear functions that allow to learn complex patterns and by this it can make the right predictions. In machine learning there are many activation functions that have been defined and some of the basic functions are still popular these days. Each activation function has advantages and disadvantages and its own characteristics which it should be considered before we use them. Each kind of tasks and application might needs different the speficific activation functions depending on what is the most important aspects.

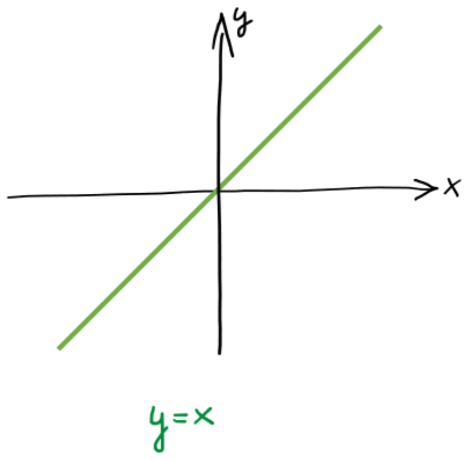


Figure 4 Linear

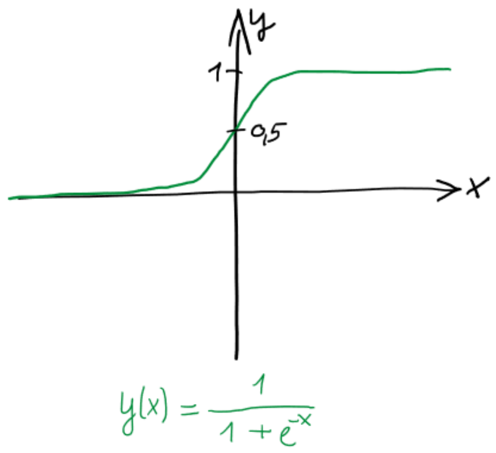


Figure 5 Softmax

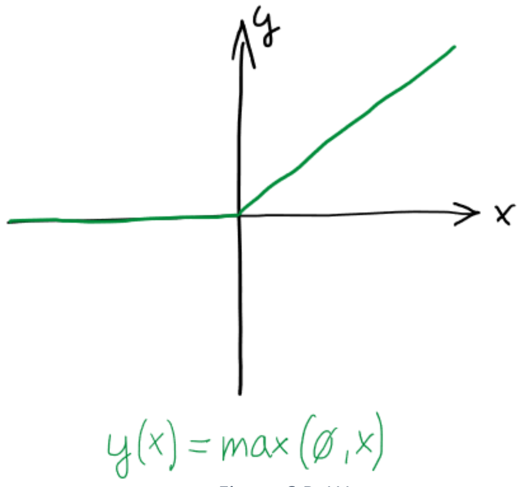


Figure 6 ReLU

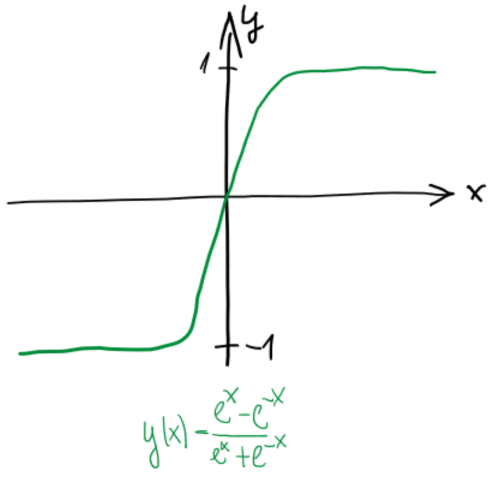


Figure 7 Tanh

1. Linear:

- Functionality: The most simplest activation function, the input value is match with the output value.
- Advantages: Fast computation.
- Disadvantages: Linear so this limiting the ability to learn complex structures and relations

2. ReLU (Rectified Linear Unit):

- Functionality: It is zero if the input is negative , else it will be linear.
- Advantages: Fast convergence and computation almost as fast as linear and it can use to solve vanishing gradient problem.
- Disadvantages: Sometimes the "Dying ReLU" problem can occur, where some neurons become inactive for negative inputs. Because of these some weights and bias won't be updated and these neurons will not contribute to the predictions.

3. Sigmoid:

- Functionality: Often called S-shaped function and put the inputs between 0 and 1.
- Advantages: Great use for binary classification
- Disadvantages: The vanishing gradient problem is common, and this can cause slow and expensive calculation and some weights might not be updated.

4. Tanh (Hyperbolic Tangent):

- Functionality: It squashes input values between -1 and 1.
- Advantages: A non-linearity function and have stronger gradients than sigmoid.
- Disadvantages: Vanishing gradient problem is common like in the sigmoid

Optimizers

Optimizers serve as the backbone of neural network training. These help to update the weights into the optimal values but each of these algorithms has different unique method to adjust them. There are two optimizers that are often recommended to use in the machine learning community: SGD and Adam. The SGD is a very basic optimizer that updates weights witch each of the training sample. It's usually converging slower than most of the optimizers but it is useful for getting the most accurate results.

Adam Optimizer:

The most used optimizer is called Adam. It is an adaptive optimization algorithm that combines Momentum and RMSProp. The purpose of Momentum is to get out from the local minima during the training phrase, so the the weights are not stuck to a non-optimal value. The RMSProp is better version of ADAGRAD which includes the previous updated weight values with the "moving average" method. Adam is often converging faster than the SGD. For basics applications and projects, the Adam is a great choice, and it usually recommended as a default optimizer.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Figure 8 Formula of Adam optimizer

Regularization

Regularization techniques are essential tools for improving the generalization performance of any of the machine learning models. Regularization helps prevent overfitting leading to more reliable and effective models for real-world applications. So, regulating the weights helps to learn data that haven't seen and make it reliable for any of the real-world data.

Here are some commonly used weights regularization techniques:

L1 (Lasso)

This regularization is adding the absolute weight values to the lost function, and they are counted as loss. Due to the model that want to minimize the loss function it will want the weights to be as close as possible to the 0 value. In extreme cases it will eliminate the weights, but it can view a good thing since it can remove the least necessary features and make the calculation faster .

L2 (Ridge)

This regularization is adding the squared weight values to the lost function, and they are counted as loss Like the L1 but this won't make the weights to 0. It reduces the values but it not as significant as L1 and it will keep every weight values.

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij}W_j\right)^2 + \lambda\sum_{j=0}^{M}|W_j|$$

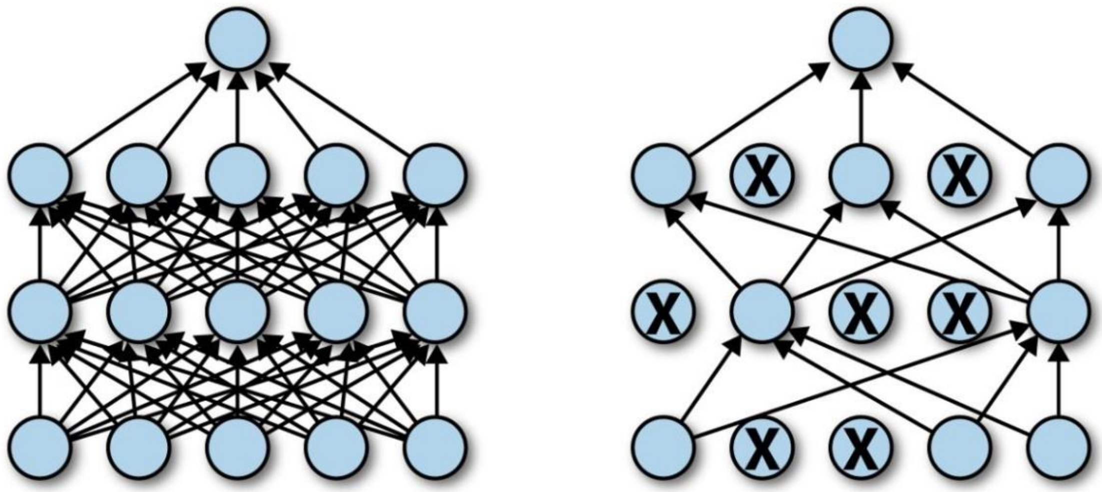
L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij}W_j\right)^2}_{\text{Loss function}} + \lambda\underbrace{\sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Figure 9 Formula of L1 and L2 regularization

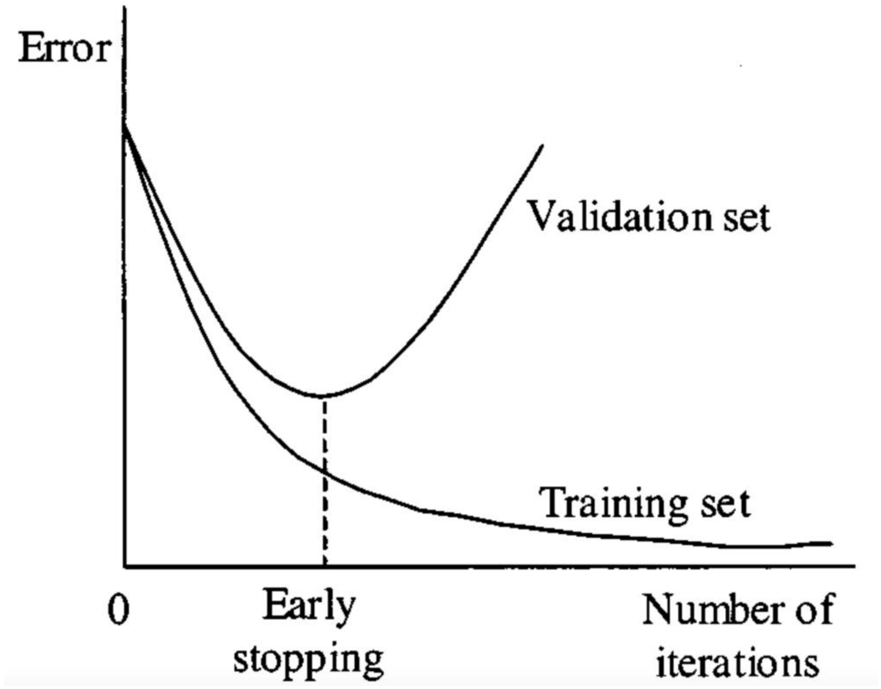
Dropout:

This regularization is used in neural networks to prevent overfitting and improve generalization. It works by randomly deactivating (or "dropping out") a given proportion of neurons in the network during training. These inactive neurons will not contribute to the forward or backward propagation during a particular iteration.



Early Stopping

This regularization is also important to get the model best weights. During training we can keep track of the train and the validation loss. The validation loss is the indicator of the model generalization ability, and it should be kept as low as possible. There is one point in the training phrase that the validation loss is going to increase, and the early stopping will store the earlier best weight depending on the patience value.

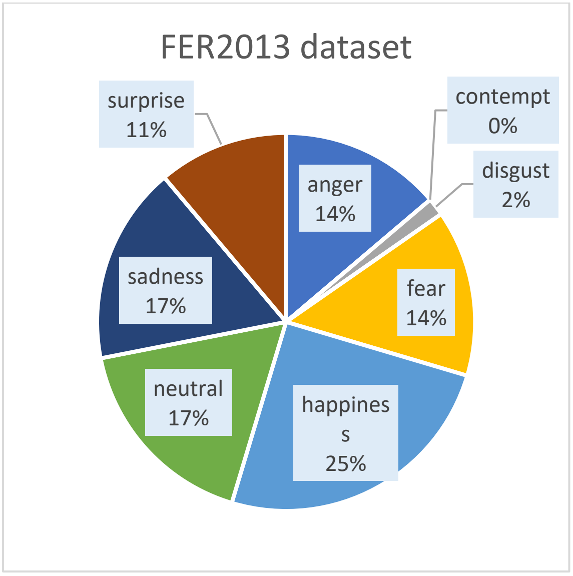


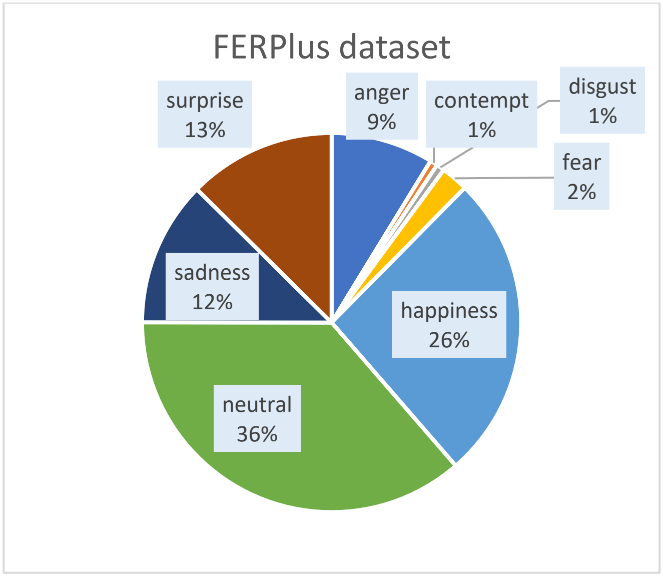
Chapter 3

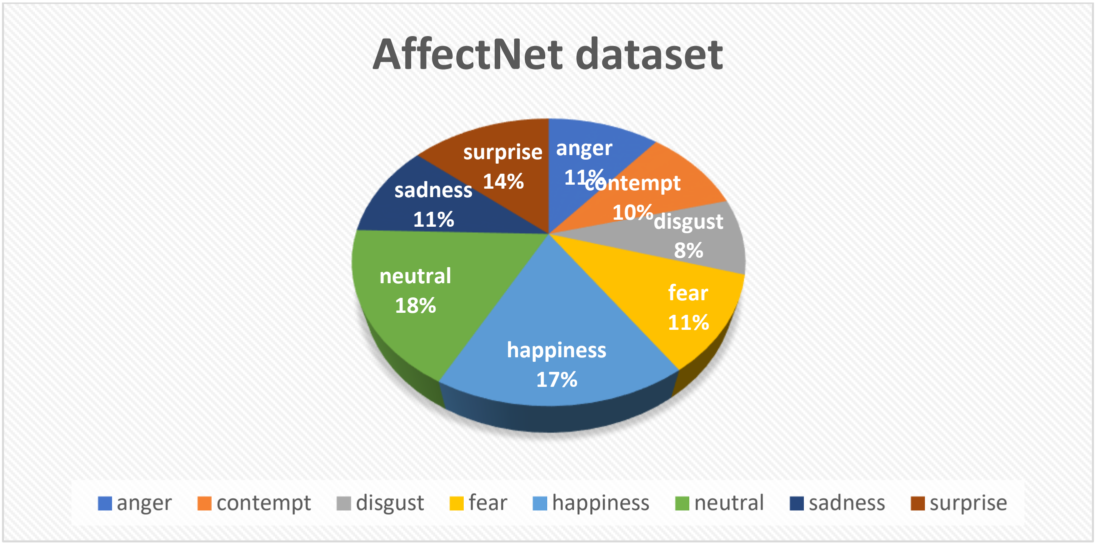
My implementation and work

Dataset

In the beginning of my work with different dataset that contain face images like FER2013, FERPlus and Affectnet. Since the FER2013 labels have not classified dataset very accurately I decided to classify them with the FERPlus labels. During my work I tried to use a small portion Affectnet dataset, but I had challenges with them. They had 96 x 96 pixel color images, while the FER dataset uses simpler image that in only 48 x 48 pixel resolution and only black and white themed. Unfortunately, I didn't have the best GPU on my computer, so I decided to use the FER dataset to save some computational times and resources. Both of these datasets have at least 25000 images.







Originally the FER2013 has only 7 categories (happy, neutral, disgust, anger, surprise, fear, sad) but the FERPlus expanded it to 8 with the contempt category. The FER2013 have a somewhat good distribution of the emotions but after relabeling process the images the distribution became a bit unbalanced.

The most noticeable changes between the FER dataset is that the FER+ drop the ratio of the fear emotion, while it made the neutral significantly larger than the original dataset. It also worth noting that contempt, disgust and fear together only takes the 4% of the dataset and the other 5 facial emotions have 96%.

The model architecture and transfer learning

I worked on the images thus I needed to use a neural network that take advantages of time and space information of the data. The traditional neural network lacks the ability to recognize pattern effectively so I decided to use the CNN techniques, but I considered the Vision Transformers too.

Since there are many exiting CNN architectures are on the Keras library in Python I decided to use one of them with the transfer learning technique

Transfer learning is a technique that allow us to use an existing model architecture and weights for our specific unique task and this will save us time and computational resources. Obviously, the weights needed to change for our specific task, thus we have to retrain the whole model again with a little effort. Most of the defined CNN models have some essential layers like max pooling and convolutional layer. These layers are specifically designed for image data and the output can be represented as an image data.

Transfer learning has the base model and the classification head layer (transfer model). During the fine-tuning process the base model should be frozen, and the head layer should be trained. After all of this process we can train the whole model together and this will improve the model robustness.

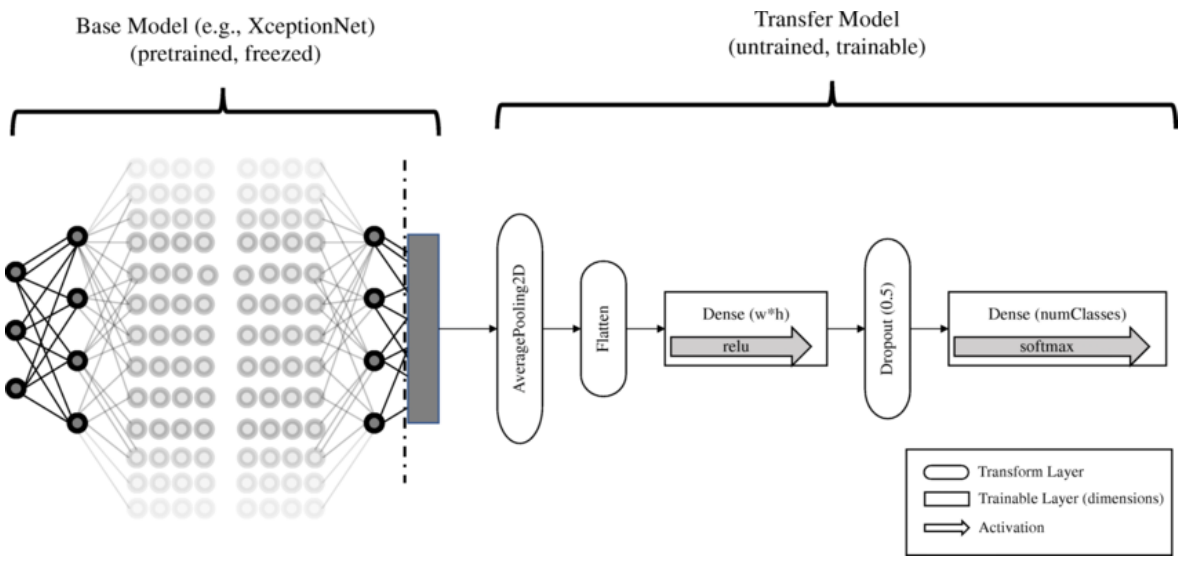


Figure 10 A Transfer Learning model consist of Base Model and Transfer Model

I choose to train my image data on base models like VGG16, ResNet50, InceptionV3 and EfficentNetB0 but I decided to use the VGG16 which gave me solid results. But it is worth noting that the other models did a great job too however the VGG16 overcomes a little bit by having a bit better model accuracy on the test dataset.

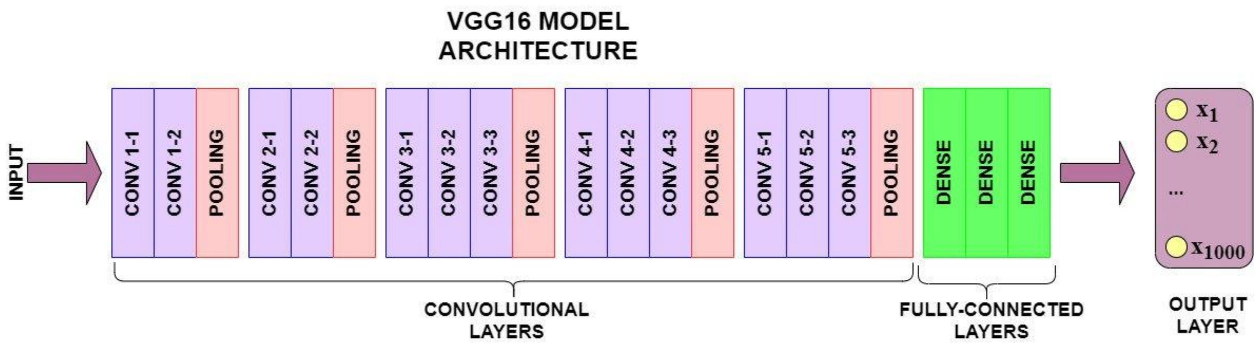


Figure 11 VGG16 model

Explainable Artificial Intelligence with LIME

Local Interpretable Model-agnostic Explanations (LIME) is a XAI tool for explaining the decision of the of the output and it gives human a little insight the black-box model which is hard to understand. It highlights some part image that influenced the output of the model, and it can show which parts were least useful. Segmentations algorithms are also playing a high role and that will create individual superpixels. Superpixels are kind of group of pixel area on the image that somewhat are related to each other.

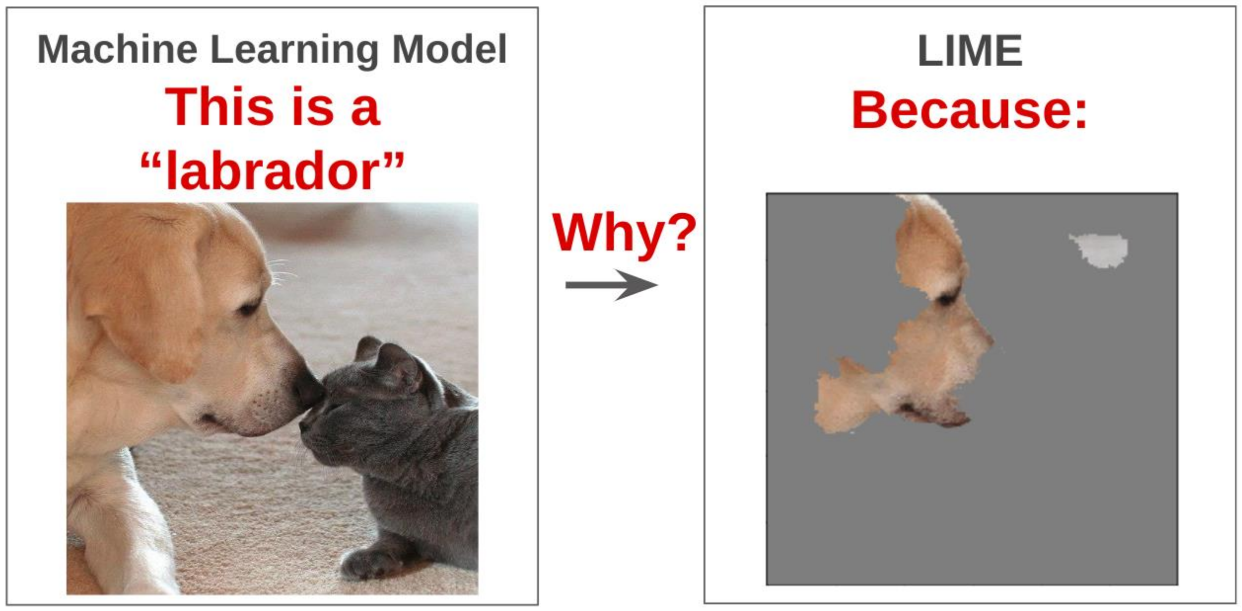


Figure 12 A simple explanation with LIME

Segmentation algorithms

Felzenszwalb

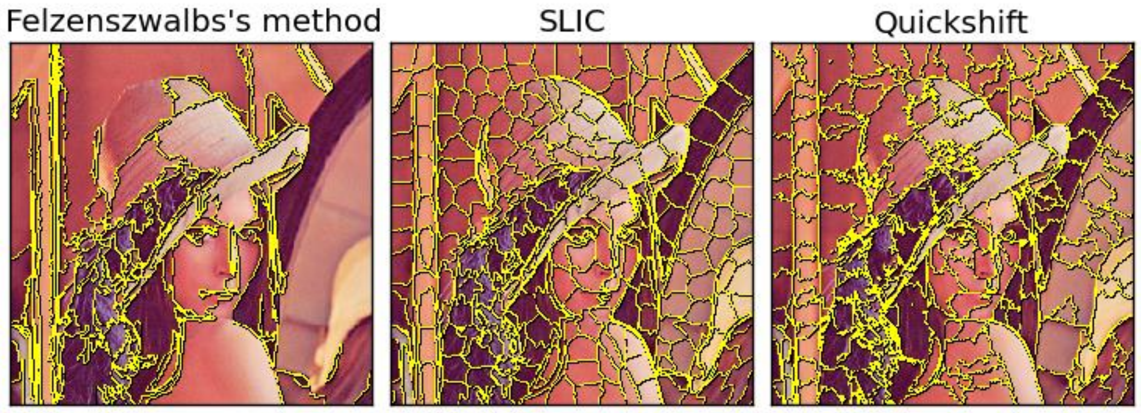
Felzenszwalb is a graph-based segmentation, and it is is a popular method used in image processing. This algorithm is known for its efficiency and ability to produce good segmentations across various images. The algorithm has a segment size (scale) parameter which controls the segmentation sensitivity.

Quickshift

A more modern 2D picture segmentation technique called Quickshift is based on a kernelized mean-shift approximation. Consequently, it is utilized in the 5D space, which consists of color information and picture position, and it is a member of the family of local mode-seeking algorithms. Quickshift's ability to compute a hierarchical segmentation on several scales at once is one of its advantages.

SLIC (Simple Linear Iterative Clustering)

This technique is closely related to quickshift because it essentially executes K-means in the 5d space of color information and image position. The clustering method is particularly effective because it is less complicated. To get good results, this algorithm must operate in the Lab color space. The algorithm took off rapidly and is currently in widespread use. Similar to Quickshift, the compactness parameter trades off closeness and color similarity, while n_segments determine the number of kmeans centers.



Chapter 4

Evaluation

Results of the model that used FERPlus dataset

I trained a model that was only allowed to use 50 epochs, but it usually finished under 25 epochs with early stopping. The model used VGG16 for transfer learning and used 64 batch sizes. I focused particularly on the categories that had very few samples, like disgust, contempt, and fear. I used a technique to handle this problem by adding weight importance to each category. I luckily found a loss function that helped solve that issue. I used a loss function similar to CategoricalFocalCrossentropy because the old version of TensorFlow I used didn't have this function implemented.

The first model I created didn't account for the unbalanced distribution, and it performed very poorly on the disgust, contempt, and fear emotions. It mainly focused on the other five emotions, which resulted in a high overall accuracy. The model worked great overall, but a balanced main diagonal is the most important aspect for human perception. It had around 80% accuracy on the test dataset, but the main diagonal average percentage was only 54.25%. This could be improved if the neglected categories received more attention.

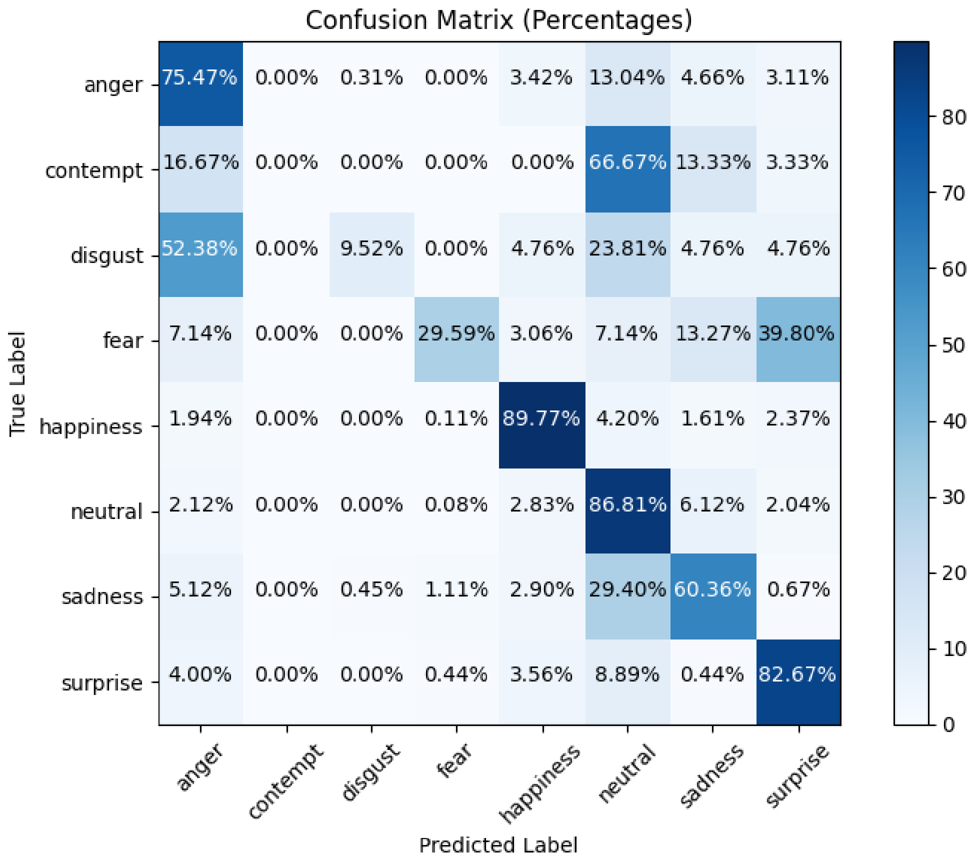
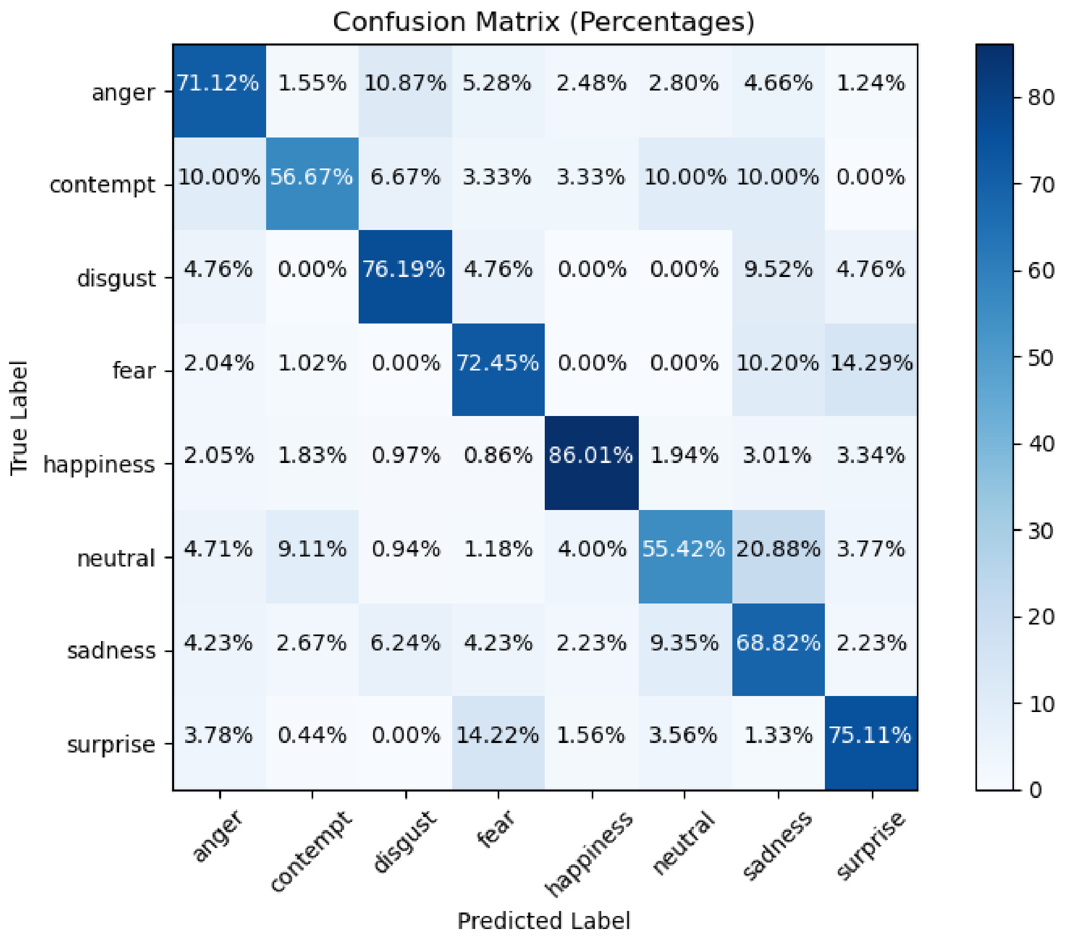


Figure 13The confusion matrix of the first model that didn't handle unbalanced class

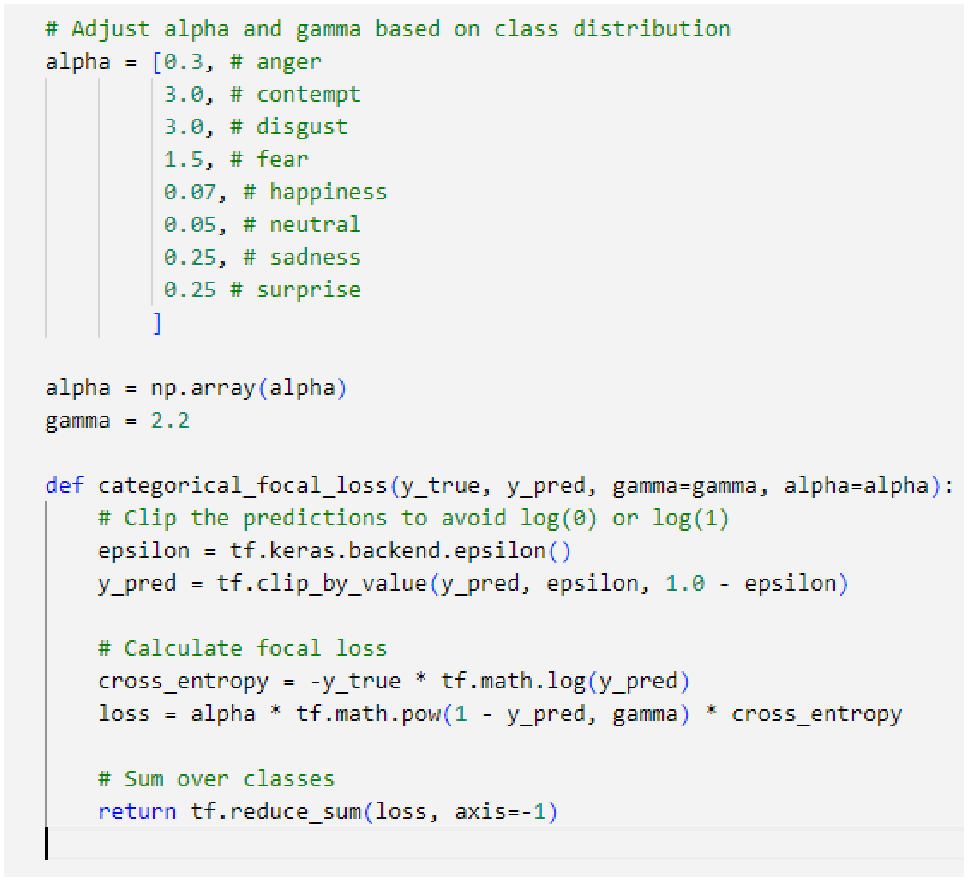
The final model I used could handle the unbalanced distribution, and I intended to reduce the false positive and false negative ratio for every facial emotion. I gave significantly less attention to neutral and happiness compared to the other categories, despite happiness still having the highest accuracy on the confusion matrix. I prioritized contempt, disgust, and fear significantly higher overall. This model had around 70%

accuracy on the test dataset, and the main diagonal average percentage was also 70%.



True Label \ Predicted Label	anger	contempt	disgust	fear	happiness	neutral	sadness	surprise
anger	71.12%	1.55%	10.87%	5.28%	2.48%	2.80%	4.66%	1.24%
contempt	10.00%	56.67%	6.67%	3.33%	3.33%	10.00%	10.00%	0.00%
disgust	4.76%	0.00%	76.19%	4.76%	0.00%	0.00%	9.52%	4.76%
fear	2.04%	1.02%	0.00%	72.45%	0.00%	0.00%	10.20%	14.29%
happiness	2.05%	1.83%	0.97%	0.86%	86.01%	1.94%	3.01%	3.34%
neutral	4.71%	9.11%	0.94%	1.18%	4.00%	55.42%	20.88%	3.77%
sadness	4.23%	2.67%	6.24%	4.23%	2.23%	9.35%	68.82%	2.23%
surprise	3.78%	0.44%	0.00%	14.22%	1.56%	3.56%	1.33%	75.11%

Figure 14 My final model's confusion matrix



```
# Adjust alpha and gamma based on class distribution
alpha = [0.3, # anger
         3.0, # contempt
         3.0, # disgust
         1.5, # fear
         0.07, # happiness
         0.05, # neutral
         0.25, # sadness
         0.25 # surprise
        ]

alpha = np.array(alpha)
gamma = 2.2

def categorical_focal_loss(y_true, y_pred, gamma=gamma, alpha=alpha):
    # Clip the predictions to avoid log(0) or log(1)
    epsilon = tf.keras.backend.epsilon()
    y_pred = tf.clip_by_value(y_pred, epsilon, 1.0 - epsilon)

    # Calculate focal loss
    cross_entropy = -y_true * tf.math.log(y_pred)
    loss = alpha * tf.math.pow(1 - y_pred, gamma) * cross_entropy

    # Sum over classes
    return tf.reduce_sum(loss, axis=-1)
```

Figure 15 The loss function that could handle the unbalanced classes

Results of Explainable Artificial Intelligence with LIME

I used the Felzenszwalb, SLIC, and Quickshift segmentation techniques with LIME to get an explanation of the decision. I wanted to try other segmentation techniques, but I didn't have time to investigate them.

In summary, I got many different results with these segmentation methods on the same faces. I used Paul Ekman's emotional traits to compare with the LIME results. In many cases, I found that the image schemes between the FER images and Paul Ekman's traits did not exactly match. I found many instances where hands in the image covered one of the essential emotion traits, such as surprise and fear.

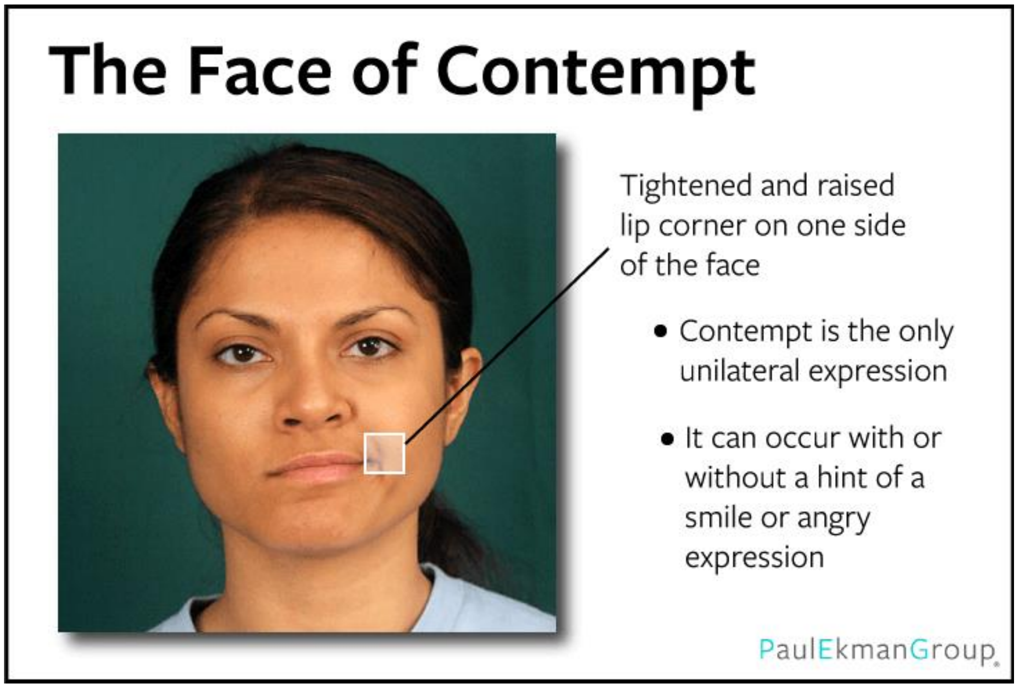

The Face of Contempt
Tightened and raised lip corner on one side of the face
• Contempt is the only unilateral expression
• It can occur with or without a hint of a smile or angry expression
PaulEkmanGroup

Figure 16 Paul Ekman's emotional traits for contempt

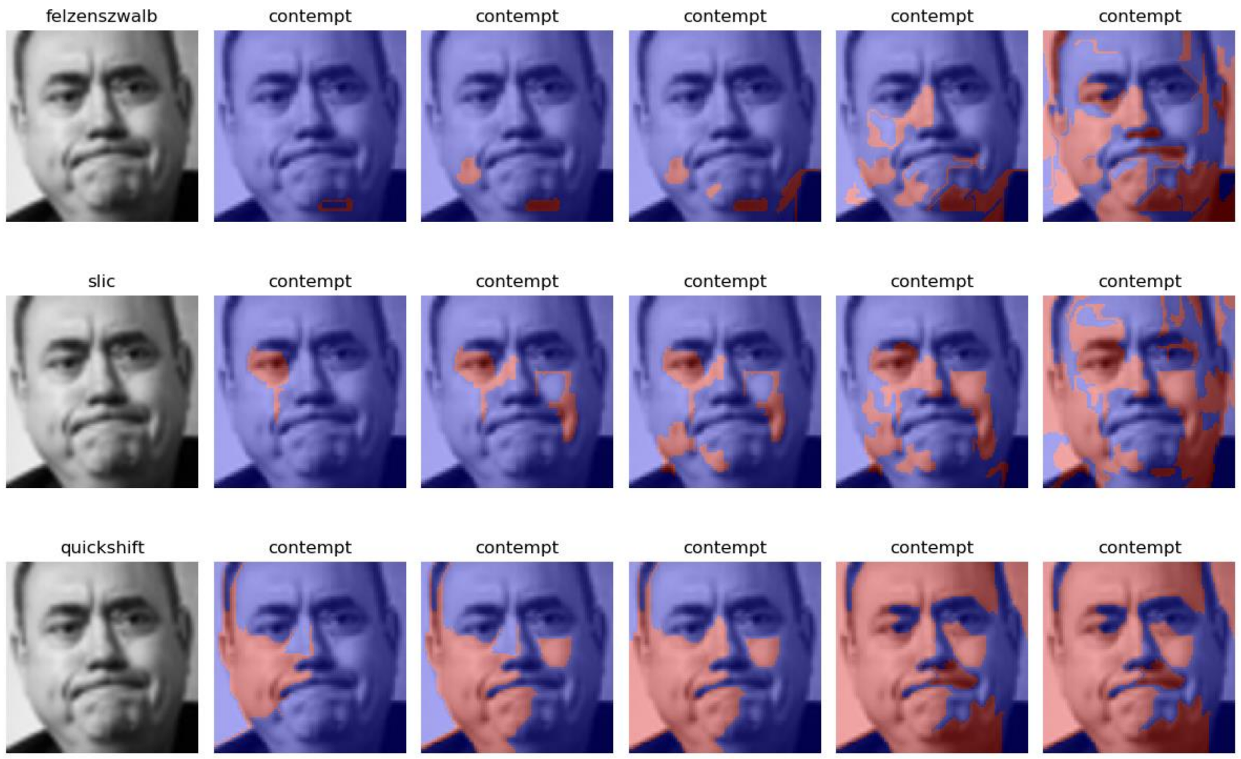
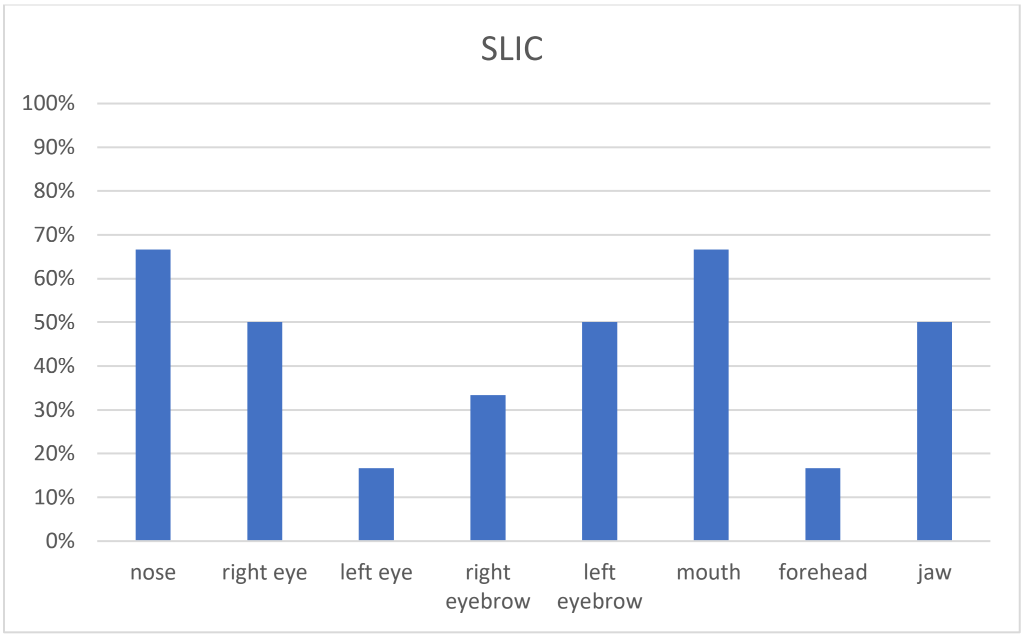


Figure 17 Using LIME for three segmentations algorithm. The first is the original, the second is only show the most significant super pixels, the third show the 3 important superpixel and etc. The red is the positive and the blue is the negative. The last image shows all the positives as red. The Quickshift has only around 15 superpixels and the other two has at least 100 of these.

I also created charts showing the frequency of facial regions for the SLIC and Felzenszwalb segmentation based on 6 randomly chosen images from the Test Dataset for each category. This helped me understand which parts of the face are important for LIME. Obviously, it was not that easy to determine to the regions counted or not. I choose to count if I find that the positives regions cover at least the 50% the face part, but also if they highlight a significant part of the face region then I counted that as well.



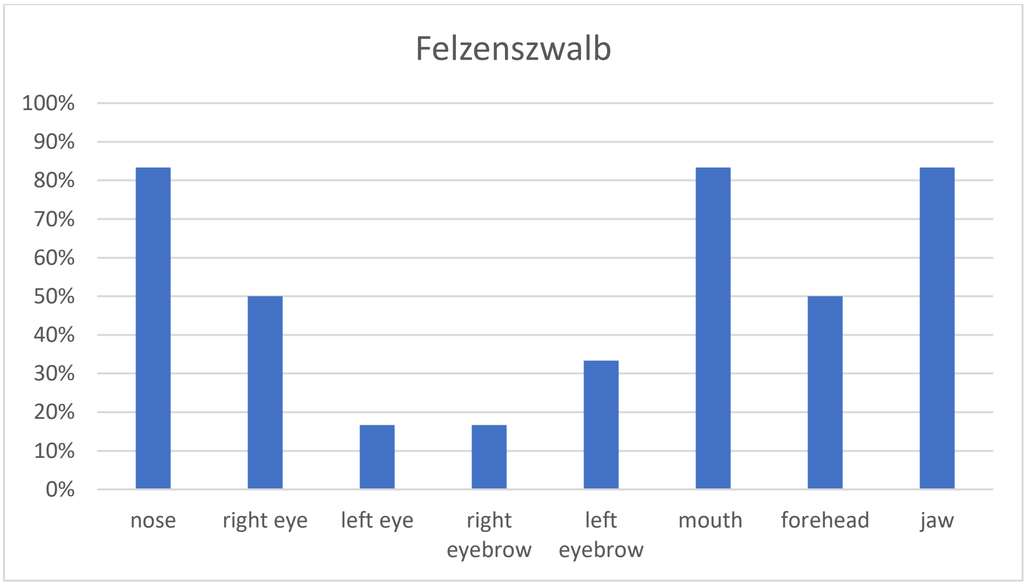


Figure 18 The LIME has almost the same result with SLIC and the Felzenszwalb on the contempt emotion. It can be seen that the mouth and the nose were the most important parts which is right..

Acknowledgment

I would like to express my special thanks of gratitude to my consultant, Dr. Hullám Gábor for the help on this project.