École Polytechnique

Predicting the relationship between people based on their emotions

Diego Gomez
CSE 303
Vicky Kalogeiton
December 13,2020

Report: Predicting the relationship between people based on their emotions

Diego Gomez
École Polytechnique
diego.gomez@polytechnique.edu

Abstract

In this project we construct a Neural Network(NN) to predict emotions and relationships. Our NN is composed of three sub NN, two of which predict the emotions of two individuals and one that infers relationships between the two people from the predicted emotions. The two sub NN perform well when predicting concrete feelings such as happiness, sadness and anger; but it has trouble with more subtle feelings such as fear and surprise. Nevertheless the final NN did not give adequate results. That is it did not manage to predict the relationship of the characters. We give arguments as to why this happens in the following report. It is my understanding that there is no current research on this topic specifically, i.e predicting human relationships from emotions.

1 Introduction

Facial detection, recognition, and emotion recognition are all important and well-studied topics in computer vision. In this project, we attempt to merge all of these powerful tools to infer more subtle information: human relationships. If developed properly this model could be applied in a variety of fields. One that where in my opinion it could do a big difference is in the fight against bullying. Perhaps one could imagine setting cameras up and studying the relationships of all children to see if there is any kind of conflict. Moreover, this could help the responsible adults act quickly to prevent many of the tragedies caused by social problems, such as suicide, and mass shootings.

In the following report we break down all the important steps that were necessary to achieve the current results. Moreover, we provide a critique to them, while proposing possible next steps.

2 Related work

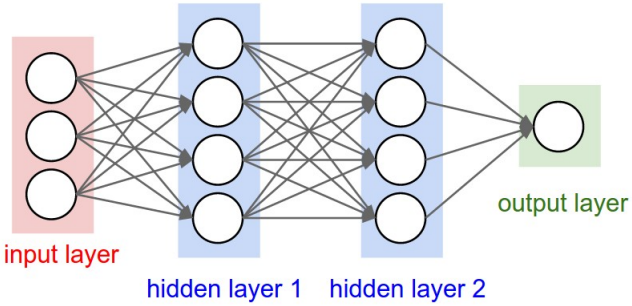
Let us start by defining the crucial elements of this project.

First of all Neural networks (NN) are composed of what we call layers, that is a collection of nodes operating at a given depth of the network. The two types of layers that are mainly used in this project are the following.

Fully Connected layers (FC). An FC layer is a structure of a neural networks, where all of the nodes of a layer are connected to each node of the next one. These types of layers have as input and output, vectors of independent dimension. Nevertheless, the dimension of the output layer of any given layer has to match with the input dimension of the next layer. An important notion that is to be used in this project is the fact that, two FC layers, behave the same as one. This explains the need to add non-linearities, such as the Rectified Linear Unit (ReLU) function, after FC layers. This project uses the ReLU function on several instances, let us state the formula.

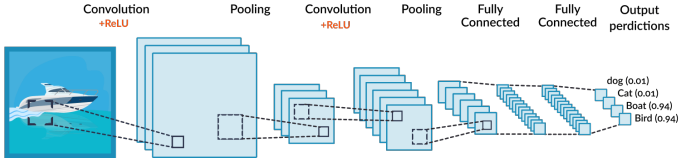
$$ReLU(x) = \max(x, 0)$$

Here is an example of a simple NN composed solely of FC layers.



Convolutional layers. In contrast to FC layers, these kinds of layers can input higher dimensional objects, i.e tensors. The previous fact makes possible to take advantage of information about the structure of the data, information which would be completely or partially lost, by using FC layers. Put in simple terms,

a convolutional layer has a kernel matrix, which we call the filter, and this filter slides through the image spatially (i.e wrt the width and height) computing the dot product. This results in a more "general" representation of the original image. One of the advantages of such layers, is that it helps with translational invariance. We do not make explicit use of these kinds of layers in this project. However, the input to our NN is data that was processed by a structure utilising these kinds of layers. We call the NN composed solely of Convolutional layers CNN. Here is an example of a CNN.



To train a NN means to make small changes to its nodes until it provides satisfactory results. To define what a satisfactory result is we need to define an evaluation function. In the machine learning lingo this is commonly referred to as loss functions. With these elements one can see that one is facing an optimization problem. Therefore we can borrow tools from statistics and mathematics to tackle it. In fact pre-implemented versions of such tools already exist, they are called optimizers. The Adam (Adaptive Moment Estimation) has been proven to be a really good one, let us briefly explain how it works. In simple terms "Adam behaves like a heavy ball with friction, which thus prefers flat minimum in the error surface [ADAM, 2015]. In fact what makes Adam so special is that its implements Adaptive Learning Rate, that is, instead of using a fixed learning rate like the Gradient Descent to "roll" down the slope, having an adaptive learning rate allows us to take big steps in the beginning and small steps at then, which helps us converge much faster. In the following project we will be using the Adam optimizer. Let us be more precise about the loss functions briefly mentioned before. Put in simple term these functions are maps that punish bad predictions and rewards good ones. Let us present the loss functions that we will be using in this project. To begin, we need to explain two important notions, what the $softmax$ function is and what a hot vector is. Here is the formula;

$$\sigma(\mathbf{z})_i = \frac{exp(z_i)}{\sum exp(z_i)}$$

where $\mathbf{z}$ is an $n-$dimensional vector. This function allows to turn vectors into probabilistic ones, i.e where the sum of its components add up to 1.Finally a hot vector is simple a vector where only one component is 1 and the rest are 0.These two tools couple with the Mean Squared(MSE) loss constitute a powerful tool.

Let us state the MSE's formula.

$$MSE(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{n}\sum(Y_i - \hat{Y}_i)^2$$

The purpose of the softmax function is now apparent. Now the second loss function that will be used in this project is the Binary Cross Entropy loss. Let us state the formula. Let $x, y \in \mathbf{R}^n$ and denote the loss function by l,

$$l(x, y) = \{l_1, \dots, l_n\}^T$$

where,

$$l_i = w_i\Big(y_i log(\sigma(x_i)) + (1 - y_i) log(1 - \sigma(x_i))\Big)$$

where w_i denotes the $i-$th component of the weight vector.
In the above formula x and y will be the output of the NN and the ground truth respectively. Let us explain this loss. The formula of l_i is the cross entropy of x and y. Put into simple terms this will measure how different the two inputs are.
Evaluating NN that output hot vectors is rather simple. If the $argmax$ of the output matches the label it is correct. However, when it comes to NN that output multilabel vectors it is not that straight forward. To evaluate these kinds of NN in this project we will use the Average Precision Score function provided by sci-kit learn.
First of all let us explain the notion of true positive (TP), true negative (TN), false positive (FP), false negative (FN). We say that an output is a TP is it correctly matches a desired class, similarly a TN is when it correctly rejects the opposite class, and we define inversely FP and FN. From this notions we define precision and recall.

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{FN + TP}$$

Then

$$AP = \sum(R_n - R_{n-1})P_n$$

where P_n and R_n are the Precision and Recall at the $n - th$ threshold.
Now in order to properly tackle the use two major AI subjects is needed, emotion recognition (ER) and object detection.
Let us go over ER. It is interesting to get familiar with some of the most recent advancements and NN used for this specific problem. For example, the network EmoP3D [EmoP3D, 2018], it was inspired by the pyramidal structure of the brain in order to better

detect emotions on video. In fact this neural network outperformed previous models, and matched the accuracy of the neural netowork it was inspired from, 3DPyraNet, while having a 76% speed up (which was achieved by the pyramidal structure since, it implies the use of less neurons).

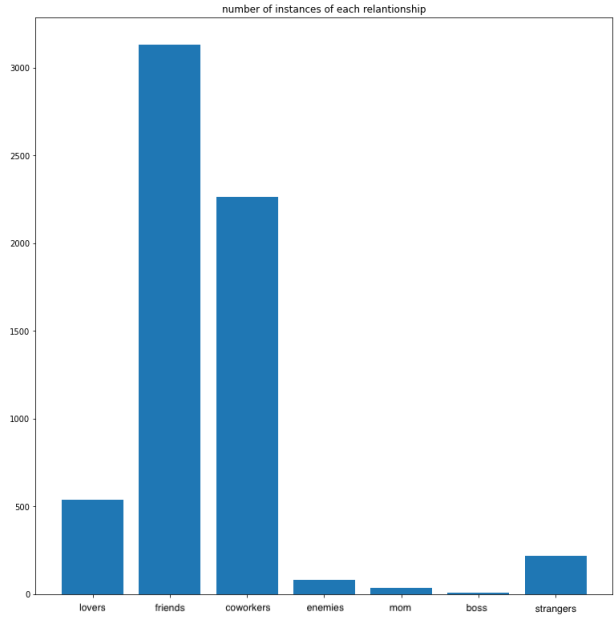
On the other hand, we are actually dealing with a problem that requires just a subset of object recognition, since we are just interested in detecting humans. To get familiar with this topic we used dectetron2 developed in Pytorch by the Facebook research team. We summarize the solution to the object detection problem. Briefly stated, the solution to the above problem inputs an image and outputs a set of bounding boxes with their corresponding class number.In the case of FasterR-CNN, an object detection neural network, this is done by first running an image through a CNN, from the result of this (that we call the feature space) we choose a set of Regions of interest, thanks to another CNN. Then we use these regions of interest to detect the objects and we classify them. Finally we get rid of the duplicate detections we, while keeping the best suited bounding boxes, thanks to IoU and the Non-max suppression algorithm.

Finally let us define some vocabulary that is frequently use when tackling video related problems.

A track is a sequence of bounding boxes that follow an object along the video as time passes. A shot is a continuous sequence of frames where the camera was not stopped. A balanced dataset is a dataset where all of the classes have similar proportions.

3 Annotations

In this project a dataset was built in order to train and test the NN. Since we are dealing with emotion and relationship recognition, a sitcom is the perfect source of information. We choose to study the first 6 episodes of "The Big Bang Theory" show because it provides a variety of relationships that we estimated were relatively simple to predict. The relationships that the NN will atempt to predict are the following: lovers, friends, coworkers, enemies, mom, boss, strangers (Note that mom and boss are still symmetric relationships for the sake of simplicity).



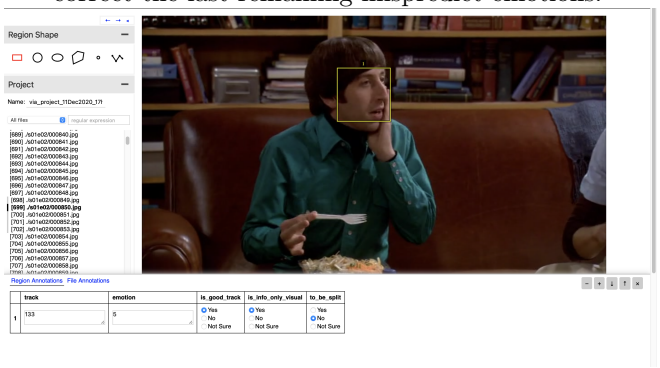
Above one can see a bar plot representing all of the instances of each relationship throughout the 6 episodes. In fact annotating the relationship of the characters is quite simple. That is because their relationship remains constant throughout these episodes. The characters that we will be focusing on are the following: doug, gablehauser, howard, kurt, leonard, leslie, mary, penny , raj, sheldon, summer. Thus there being 11 characters, and the relationships being symmetric, there was only a total of 55 pairs to annotate. It is important to note that this data set is very unbalanced. Nevertheless it was decided to keep it this way for two reasons. In real life these kinds of proportions might be maintained. There is much more people that are friends and coworkers than there is mothers and bosses. And because due to time constraints it was not possible to perform data augmentation.

The hardest part comes when annotating the emotions of each track. To construct this dataset the episodes were first split into frames and shots, and face recognition software was used to define the tracks of each character. Pre-existing emotion recogntion software[1] was applied to these tracks in order to help with the annotating process. Nevertheless this emotion recognition software is not perfect. In order to greatly improve the accuracy of the ER software a form of context was introduced to the prediction process, i.e the fact that human emotions are stable in the short term. This means that if a given character is feeling a certain emotion on any given frame it is likely that

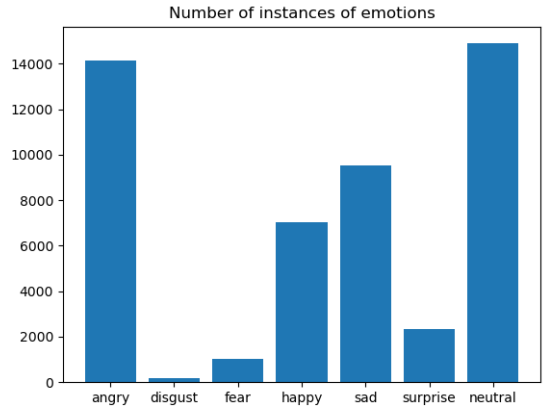
[1] github.com/priya-dwivedi/face_and_emotion_detection.git

they will continue to feel it for the next few frames. We implemented two different methods, one that would take the average emotion of each shot, and another one that would take the maximum. When both techniques agreed on the emotion the prediction was accepted, otherwise a hybrid of these techniques was used as a last attempt to correctly predict the emotion. This makes it harder for the Neural Network to change rapidly from one emotion to another.

Once this process was done, the VIA software[2] to correct the last remaining mispredict emotions.



After correcting the mispredicted graphs this was the result.



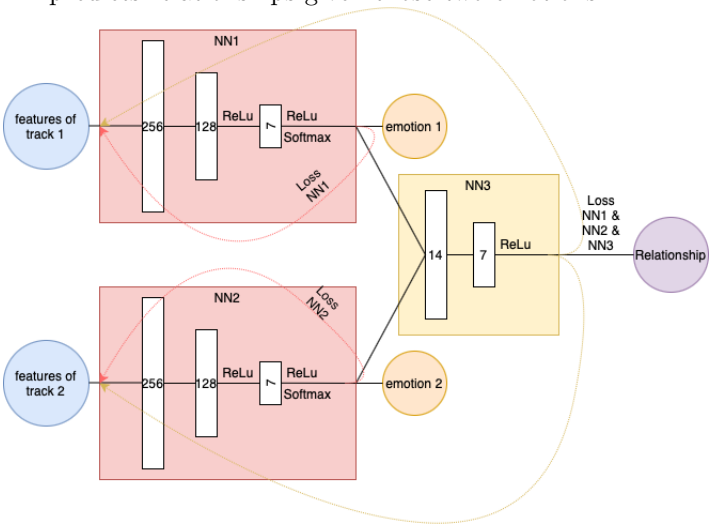
However it important to note that this dataset remains to be far from perfect too. For instance, note that the dataset is unbalanced. The proportions of feelings such as "disgust" are too low in comparison to the rest of the dataset. In order to tackle this problem data augmentation could have been used. Nevertheless, once again due to time constraints this was not possible to implement. Moreover it is also important to take into account that because this dataset is limited to the use of 6 emotions and since it was annotated by one person it is inevitable to introduce human bias.

Now that we are aware of our biases and the flaws in our dataset we can continue to the training and testing process.

[2]https://www.robots.ox.ac.uk/ vgg/software/via/

4 Structure

Below we have the structure of the NN that was used in this project. It is a combination of three separate NNs, two of them predict emotions and the last predicts relationships given those two emotions.



Let us briefly explain the structure of the NN. The two NN that predict emotions are identical. The input is the features of a given track (the features were provided by the advisor) and the output is a 7 -dimensional vector. The $argmax$ of the output constitutes the predicted emotion. The networks are composed of two fully connected layers, with activation function $ReLU$. The input dimension is 256 so as to agree with the dimension of the input feature vectors, and the output dimension is 7, as the NN classifies 7 classes. Finally we apply the $softmax$ function to both outputs in order to get a probabilistic vector since we are working under the assumption that there is only one emotion displayed per track.

The last NN predicts the relationship of the two characters. The input of this network is the concatenation of the output of the two previous networks and the output is again a 7-dimensional since we also classify 7 relation classes.

5 Strategy

Let us now explain the strategy we choose to train and test the NN. Note that we utilized 80% of the data to train the NN and 20% to test it.

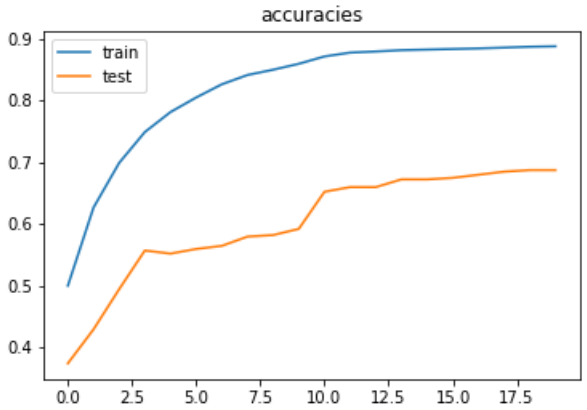
For all of the NN we use the Adam optimizer. In order to train the first two layers we use the MSE loss, NN1 and NN2 only backpropagate with respect to their own parameters. In regards to NN3, we backpropagate with respect to all parameters using the BCElossWithLogits. The back-propagation is illustrated by the dashed red

and yellow arrow. To train we have to be careful since we cannot use any arbitrary pair of tracks. Instead, we must choose a pair of two distinct characters that interact in the same shot. So a function to generate these pairs was created. This function was specially useful to have a shuffled training set at each epoch. In order to test the NN, we use the same function to generate the pairs of characters interacting in a shot from the testing data. It is important not to update the NN while in this process. The output of NN1 and NN2 was considered correct if it was exactly equal to the ground truth. In order to evaluate the output of the NN3 we chose to use the mAP.

6 Results

6.1 Quantitative

Here we will display the quantitative results of the NN. First let us display the accuracy plot.



A total of 20 epochs were done. A learning rate of 10^{-4} was used for the first 10 epochs, and a learning rate of 10^{-5} was used for the rest. The result is a final training accuracy of 90% and a testing accuracy of 70% for the prediction of emotions.

First of all we have the confusion matrix of the emotions. We can see that it does a decent job at predicting the emotions "angry", "happy","sad" and "neutral". While "disgust" is frequently confused with "angry"and eventually "neutral" . This can be explained by the lack of representation this emotion has in our dataset. Nevertheless this is not the only factor. We can see that "fear" and especially "surprise" have very low accuracy scores, even though they are much better represented in our dataset. In fact, the emotions that the NN had troubles predicting are much more prone to be subtle and/or to be subjective. Therefore, although these are not state of the art results, they are sufficient. That is because they allow us to distinguish the most different emotions "angry", "happy", "sad" and "neutral".

Interestingly, we observe that most wrong predictions are attributed to the 'neutral' class in both the correctly predicted classes (such as angry or happy) and the wrongly predicted ones (such as fear or surprise). We speculate that this is due the particularity of the 'neutral' emotion: first, it is very well presented in the dataset (30%), and second it is the emotion *in between* positive and negative ones (i.e. between happy and sad or angry and surprise). We argue that removing the neutral class would have forced the network to make a different decision, thus increasing the performance in the other classes.

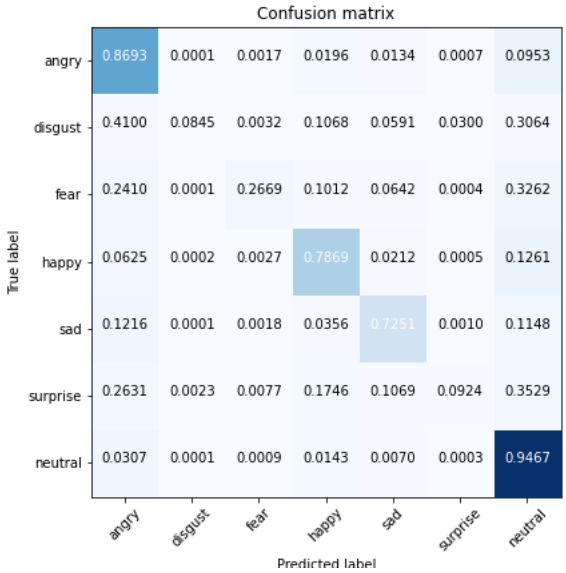


Figure 1: The darkness of the color represents how many instances we had of that given case

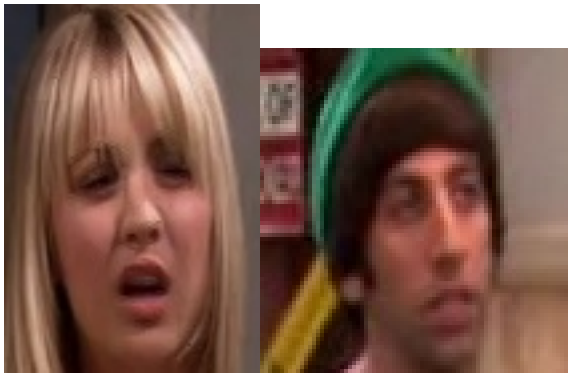
6.2 Qualitative

Let us present some of the qualitative results of the two sub NN. First we show some good predictions.

(a) angry Leonard (b) surprised Leonard

(c) disgusted Sheldon

(d) sad Penny (e) happy Howard



(f) Disgusted Penny (g) Neutral Howard

Figure 2: Good Predictions

We can see that these emotions are very unambiguous, which is why the model has no problem predicting them.

Now let us treat the bad predictions. Moreover, we will see that this behaviour is not completely surprising.

(a) predicts angry, GT: disgust (b) predicts neutral, GT: happy

(c) predicts neutral,GT: angry (d) predicts neutral, GT: happy

(e) predicts angry, GT: disgust

(f) predicts neutral, GT: fear (g) predicts happy, GT: fear

Figure 3: Bad Predictions, GT: ground truth emotion as annotated

Some of these photos are ambiguous. For instance whether someone is disgusted or angry can be a subtle difference when only being presented with an image. For example, figures 3.a and 3.e were annotated to be disgust, taking into account the context of the scene. However, from this pictures one might agree with the label given by the model. Nonetheless, there are still some pictures that are clearly wrong. Such as figures 3.b, 3.c, and 3.d, which are obviously happy, angry and happy. Now in regards to the last two figures. The full picture was included because context was important in

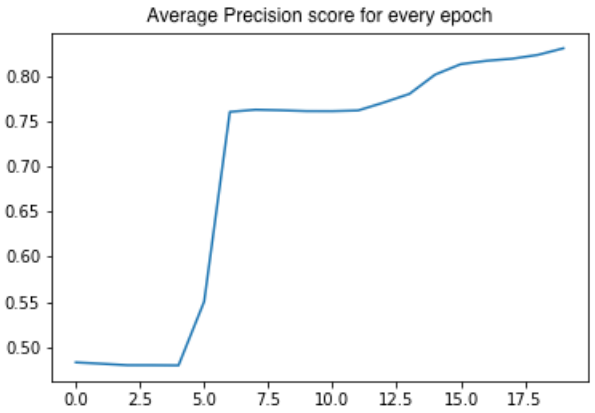
these cases. For the figure 3.f, a human being can deduce mostly from physical behaviour that these two characters are feeling somewhat scared, but this is really hard to say just from studying the faces. Once again, in the figure 3.g, Raj is clearly afraid of Penny, but since the model can only see Raj's face, it lacks the physical information a human being has access to.

6.3 Discussion about NN3

We start by displaying the obtained results. We choose for the sake of clarity and meaning to only provide the following table.

Relationship	FP	FN	TP	TN
Lovers	1	516	0	2952
Friends	475	0	2994	0
Coworkers	1283	12	2126	48
Enemies	0	67	0	3402
Mom	2	47	0	3420
Boss	1	13	0	3455
Strangers	0	216	0	3253

From this table we see that the model only predicts that everyone in the show are friends and coworkers. Indeed, given the bar plot of the number of instances of each relationship that we presented earlier this yield an impressive score. This fact is illustrated by the following plot.



Moreover, when testing a similar score is obtained. Nevertheless, this model clearly does not convey any meaningful information.

It is true that this could be due to that fact that the relationship dataset is highly unbalanced. Nonetheless, we will make the hypothesis that using the current NN structure it is impossible to get adequate results. Let us state our arguments.

The first major flaw in our structure is that we are attempting to predict a rather complex phenomenon using very simple inputs. That is, in our structure the input of the NN3 is a 14-dimensional vector and the output is a vector that is only half the dimension of the input. Compare this with NN1 and NN2 where the dimension of the input is over 36 times bigger than the dimension of the output. On top of that, the input of NN1 and NN2 are highly processed data while the input of NN3 is a rather raw data point.

Furthermore, I strongly believe that performing data augmentation will not be enough to yield good results. That is because at the moment, given that we only have 6 emotions at our disposal, it is highly possible that the exact same input should map to two completely different outputs. In fact, when first tackling this problem we were working under the assumption that there exist some hidden patterns to be found between relationships and emotions (see Apendix for graphs). But now it is clear that even if these patterns exist, they are not well enough represented by our mere 6 emotions. Consider the following example, one might argue that the fact that Mary and Sheldon argue so much, and therefore are angry at each other much more often, could be a sign that Mary is Sheldon's mom. However, the fact that Sheldon and Leonard fight consistently also conveys that there is a strong bond between them and thus we can argue that anger could imply that they are friends. Moreover, perhaps coworkers are having a heated debate with respect to a topic. All things considered, we see now that if NN1 and NN2 both output anger, the relationship is not well defined. Therefore in practice what ends up happening in our case is that anything that output that is not friends or coworkers can be viewed as a statistical anomaly by the NN.

Indeed, we refer to the MovieGraphs paper [MovieGraphs, 2018].Even though, they are not dealing with the exact same problem, we see that when attempting to predict something as subtle as hidden meaning or even human relationships there is a need to use an enormous amount of features. Not just emotions, but also spatial information (scenes), and peer-to-peer interaction (interactions).

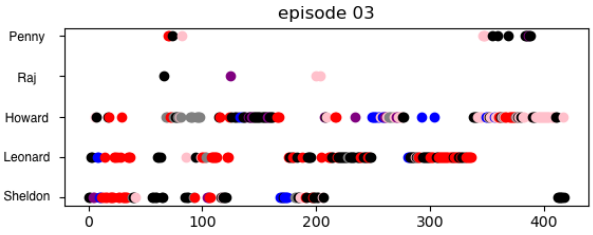
In conclusion, this problem is a highly complex one that would require much bigger datasets and way more information.

7 Conclusion

To conclude this report we summarize what we have achieved. In this project a NN that predicts emotions given the features of a track was built with adequate results. Moreover the secondary goal of predicting the relationships of the characters was not achieved due to the reasons stated above. It is clear now that the strategy that was chosen at the beginning of the project was not the best one, nevertheless through research like the MovieGraphs dataset [MovieGraphs,

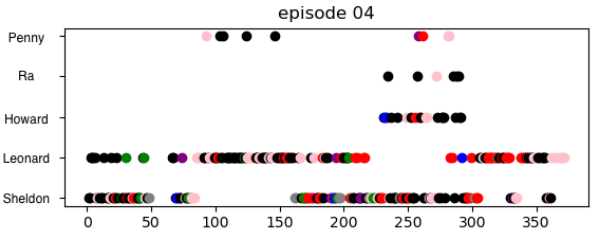
2018] we understand now what direction to take if one chooses to tackle this problem.

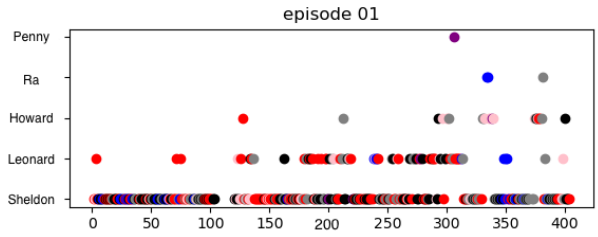
In a personal note this project as a whole has been a great mean to explore the field of computer vision, and to have a brief hands on experience of the research process. I have found this work very stimulating and enriching. Even though I did not manage to get satisfactory results for NN3 I am convinced that this experience has taught me a lot.

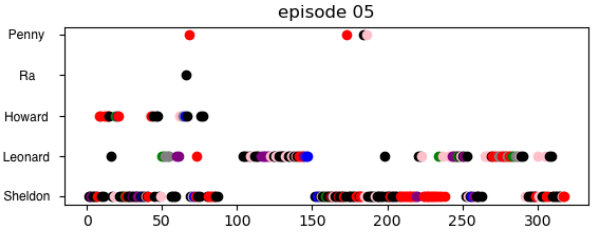


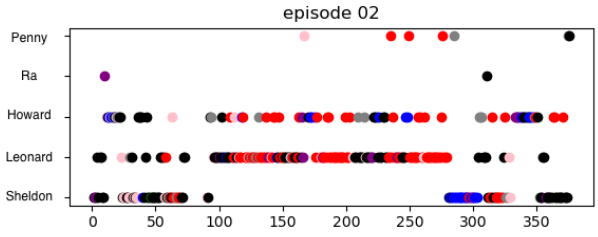
8 Appendix

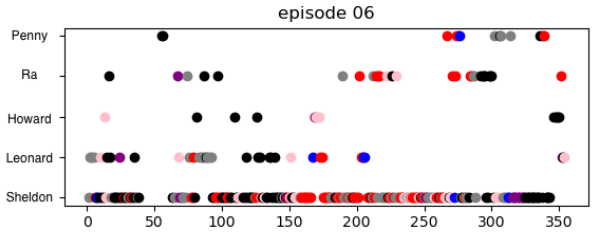
In the graphs below we have the following mapping.
red → angry,green → disgust, blue → fear,
pink →
happy,grey→sad,purple→surprise,black→neutral











This plots were made to attempt to identify hidden patterns, however no relevant ones were found.

References

[Ref, 1920]

[EmoP3D, 2018] Emanuel Di Nardo, Alfredo Petrosino, and Ihsan Ullah: EmoP3D: A Brain like Pyramidal Deep Neural Network for Emotion Recognition, 2018.

[ADAM, 2015] Diederik P. Kingma and Jimmy Lei Ba (2015) ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, [Online] Available:https://arxiv.org/pdf/1412.6980.pdf

[MovieGraphs, 2018] Paul Vicol, Makarand Tapaswi, Lluís Castrejón, Sanja Fidler, University of Toronto Vector Institute and Montreal Institute for Learning Algorithms(2018) MovieGraphs: Towards Understanding Human-Centric Situations from Videos OPTIMIZATION, [Online] Available:https://arxiv.org/pdf/1712.06761.pdf