Artificial Intelligence Project

Department of Computing and Informatics – Artificial Intelligence
Bournemouth, United Kingdom
May 2020

Saray Camargo
s5131627@bournemouth.ac.uk

Marco G Possamai
s5223875@bournemouth.ac.uk

Hossein Tamimi
S5223287@bourneouth.ac.uk

Abstract

Many lives are lost every day due to extreme depression. According to the World Health Organisation, 800,000 lives are lost per year due to suicide (Organization, 2020). The lives that are lost to suicide do not have to be vulnerable people that are in unfortunate situations, Suicide can be a terrible thought that can be present in any person's mind regardless of gender, sexuality, ethnicity and appearance. We are proposing a system that possesses the ability to detect a user's face, analyse the user's facial expression which will then be classified into emotion and output a calculation that classifies if a user is depressed by training our model using the Fer(2013) dataset (Verma, 2018). However, this alone is not enough to calculate a user's risk of suicide. This application will impact the medical industry by identifying people who are at risk of suicide by identifying people who are displaying signs of depression which could ultimately lead to suicide.

Table of Contents

Table of Figures

1. Introduction

Suicide is one of the leading causes of death around the world which racks up 800,000 deaths a year (Organization, 2019).Depression is an unfortunate state of the mind that causes changes in mood, thinking motivations and behaviours. The core symptoms of mood changes are sadness or irritable mood, other symptoms may include: difficulties with concentration, sleep disturbances, fatigue and suicidal thoughts because of this, depression is one of the main cause of disability worldwide (Watson, 2020). However, suicidal signs are not straightforward to spot since the complexity of reasons people suicide is vast and it is difficult to pinpoint a common reason people commit suicide. Mental health disorders, including depression and anxiety, are often first reported during adolescence; such conditions normally have implications for present and future health as well as development (England, 2019), as the Millennium cohort study survey suggests, half of all cases of adult mental illness start by the mid-teens, which is why it is important they are diagnosed and treated early; with the right treatment and support, most people can make a full recovery. Therefore, due to the impact that depression might have on society this project aims to recognize as well as predict depression to have early support and treatment. Also, this project can be implemented to help to develop and achieve the ambitious goals of the NHS mental health implementation plan relating to children and adolescent (England, 2019). Suicidal tell signs are difficult to recognize due to the complexity of the human brain, the fact that no one attempts suicide for a specific reason, and different reactions of different people to different situations. However, The usual signs that are displayed by individuals contemplating suicide are excessive sadness or moodiness which basically indicates prolonged sadness and unexpected rage, Hopelessness regarding the future, social isolation by choosing to withdraw from friends, family, and social activities, self-harm such as use of drugs, alcoholism, dangerous driving, and hinting at suicide through conversation with friends or families as a warning sign. However, classifying emotion is not enough to accurately predict the risk of suicide due to the lack of information that can be classified such as a victims actions such as drug abuse, recklessness, and history. (Prevention, n.d.) (Brown, 2016) (Admin, 2012). This project will use the emotional aspects of suicide warning signs to classify a risk of suicide from an image by further exploring the weight of different emotions the model classifies from the image and defines a suicide risk as positive or negative using a formula we will create for functionality purposes that will classify a user as depressed or not depressed.

1.1 Dataset

This problem will require a dataset to be available which will be The Facial Expression Recognition 2013 (FER-2013) (Verma, 2018) that will be used to train the model. This dataset consists of multiple emotion categories which contain images of different people portraying the relevant emotion. These categories will consist of anger, disgust, fear, happiness, sadness, surprise and neutrality. The portraits of these images will be a 48x48 pixel values. The issue at hand is simply focusing on a person's emotion (Kaggle, 2013). However, it is difficult to focus on multiple person's emotion due to inattentiveness of humans and simply the mere size of the human population. The approach is to firstly, create an application that can detect a face through a webcam. Secondly, train our application to detect these emotions by using our available Fer(2013) dataset from Kaggle (Verma, 2018). Moreover, we will gauge the accuracy of our application prior to moving on to linking emotion displayed to depression by presenting foreign images and record the output of the application to measure the successful classifications made by the application. Thirdly, we will need to conduct research into emotions that are present prior to suicide and improvise a link to instances of displayed emotion that will be used to gauge the likelihood of a person feeling suicidal or contemplating suicide (Clinic, 2017). This gauge will be implemented into the application which will compare it to the emotion detection of the uploaded images. Finally, the application will initially compare the emotion detected on the webcam feed and

output the emotion displayed in that image. Then output one of 2 categories which are depressed, and not depressed.

1.2 Benefits

The positive outcome of this endeavour would be to use this project to detect a face through a webcam feed, classify the facial expression output into an emotion, and calculate the risk of suicide by using a formula that we have improvised for functionality purposes that will analyse the level of emotions detected in the image and output one of two categories indicating whether a user is depressed or not depressed.

1.3 Ethics

This project will require some sensitivity due to the nature of the topic we have selected. Artificial intelligence is an emerging field that still has not been fully explored yet. There are certain ethics that must be followed when undertaking a project that implements the use of AI, especially when it comes to a medical application of AI. The main concern of this project would be the protection of user data. The general data protection regulation act states that a person controls how their information is used. Given the sensitivity of this project, results displayed must be accessible by authorized entities with the user's consent. (Union, 2018) (Etzioni & Etzioni, 2017) (Frankish & Ramsey, 2014) (Fonseka, et al., 2019).

1.4 Constraints

This endeavour to detect the risk of suicide will have some constraints attached to it. Firstly, a person does not necessarily express emotions as freely as others due to past abuse, lack of affection growing up, etc. Secondly, the complexity of humanity is extraordinary. Like an individual's DNA, emotions displayed by humans varies from each person and is difficult to categorize the behaviours leading to suicide due to the uniqueness of every person on earth. Lastly, some vital information that could be useful to help make a decision about an individual will be guarded by laws in place to protect an individual's data (Union, 2018).

2. Methodology

In the following sub chapters, we will be discussing the methods that we have used to design and implement the Suicidal Risk Evaluation project. Firstly, we will explain the project development lifecycle which we used as a reference to help us during this endeavour. Finally, we will explain in depth every step that was taken during the setup and coding stages.

2.1 Traditional Development Lifecycle

The duration of this endeavour will be planned, monitored, and referenced by using the SDLC waterfall model as a reference to the project's progress during each stage. In the following subheadings, we will discuss the process of this project from start to end (Point, n.d.).

2.2 Jupyter Notebook

Jupyter notebook is a non-profit that was founded by Fernando Perez and Brian Granger during the early months of 2015. The main goal of Jupyter notebook is to aid and accommodate data science across all high-level programming languages (Anon., 2020). This project will use prerequisites such as pandas, matplotlib, and Keras which is an open source neural network library written in python to build the model, as shown in Figure 2.1

```
import numpy as np
import pandas as pd
from tensorflow import keras
from keras.models import Sequential, model_from_json
from keras.layers import Conv2D, BatchNormalization, MaxPooling2D, Flatten, Dense, Dropout
```

Figure 2. 1 Imports

2.3 Design

The Convolution Neural Network algorithm has many filters that will be ideal for our project which will detect facial emotion. Moreover, CNN will analyse facial features in detail. We will be using a CNN with 3 convolutional layers which is most used in previous implementations. The model will assign a probability for each class of emotion. Finally, the model will attempt to recognize the features on a collection of images who output emotions associated with depression.

2.4 Quality Measure

We will be using 'accuracy' as a quality measure which will indicate the state of being correct for after the model has been tested on the test dataset.

2.5 Reshaping Dataset

The initial part of the implementation was loading the dataset into Jupyter notebook. Secondly, we did not need to split the data into training, and testing because, the data was already split into two categories from the source by allocating two csv files named train, and test, as shown in Figure 2.2

Read data file and convert to dataframe:

```
data = pd.read_csv('train.csv')
df_data = pd.DataFrame(data.values, columns=['Emotion', 'Pixels'])
```

Convert 'Emotion' column to array:

```
labels = np.array(df_data['Emotion'], dtype=np.float32)
```

Convert 'Pixels' column to array:

```
# Convert column to list
pixels = list(df_data['Pixels'])

for i in range(len(pixels)):
    # Split single string into many strings
    pixels[i] = pixels[i].split()

    for j in range(len(pixels[i])):
        # Convert strings to float objects
        pixels[i][j] = float(pixels[i][j])



for i in range(1, len(pixels)):
    # Create list of all pixel values
    ls_pixels = pixels[0]
    ls_pixels.extend(pixels[i])

# Reshape and normalise pixel list
pixels = np.array(ls_pixels,
                  dtype=np.float32).reshape(len(df_data.index),
                                            48,48,1) / 255.0
```

Figure 2. 2 Reshape Data

2.6 Model

The model developed to detect facial emotions of human faces with three convolutional layers that are used to extract relevant characteristics from the presented image. The first convolutional layer uses 64 filters with an input of a greyscale image that has a size of 48x48 in the range [0,1] with a 2x2 kernel matrix as a filter, we also applied the ReLU function in order to introduce non-linearity as well as MaxPooling2D which reduces the dimensionality of each feature whilst retaining the vital information in our model. To speed up the learning process we normalized the input layer by adjusting and scaling the activations with Batch Normalization. Moreover, we also use the max function with a 2x2 pool matrix, and (2,2) stride to help reduce the size of the output volume. The second convolutional layer Conv2D, learns 64 filters with a default stride. The final convolutional layer learns 128 filters with strides of (2,2). Lastly, there are two connected layers that are after flattening. We added a few dense layers that will focus on using the characteristics of the convolutional layers to classify our images. We only made use of 100 neurons from our dataset with a dropout of 20% since we only want to use a modest part of our dataset to reduce the computation time. Furthermore, the second dense layer uses a different activation function called SoftMax, along with 7 neurons representing each class in the Fer(2013) dataset (Verma, 2018), as shown in Figure 2.3

```
model = Sequential()

# First convolutional layer
model.add(Conv2D(input_shape=(48,48,1),
                 filters=64, kernel_size=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D())

#Second convolutional layer
model.add(Conv2D(filters=64, kernel_size=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D())

# Third convolutional layer
model.add(Conv2D(filters=128, kernel_size=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D())

# Flatten from 4D to 2D
model.add(Flatten())

# Dense layer
model.add(Dense(units=100, activation='relu'))

#Apply 20% dropout rate
model.add(Dropout(rate=0.2))

# Dense output layer
model.add(Dense(units=7, activation='softmax'))
```

Figure 2. 3 Model

2.7 Quality Testing

Following the creation of the CNN, we compiled the CNN with Adam as an optimizer, and sparse_categorical_crossentropy as our loss function along with accuracy acting as our main metric. The model was trained using 60 epochs which simply means 60 presentations of the entire training dataset. The model resulted in a 98.5% accuracy, and 84.09% validation accuracy which we deem to be acceptable given the time constraint. Lastly, we save the trained model so that we can load the model when we implement it, as shown in Figure 2.4

```
model.fit(pixels, labels, epochs=epochs,
          batch_size=batch_size, validation_split=0.2)
```

```
3342/3342 [==============================] - 26s 8ms/step - loss: 0.0519 - accuracy: 0.9856 - val_loss: 0.9546 - val_acc
uracy: 0.8409
Epoch 56/60
3342/3342 [==============================] - 28s 9ms/step - loss: 0.0530 - accuracy: 0.9803 - val_loss: 1.0716 - val_acc
uracy: 0.8062

Epoch 57/60
3342/3342 [==============================] - 28s 8ms/step - loss: 0.0713 - accuracy: 0.9782 - val_loss: 1.3906 - val_acc
uracy: 0.7656
Epoch 58/60
3342/3342 [==============================] - 28s 8ms/step - loss: 0.0460 - accuracy: 0.9832 - val_loss: 1.2043 - val_acc
uracy: 0.8337
Epoch 59/60
3342/3342 [==============================] - 28s 8ms/step - loss: 0.0514 - accuracy: 0.9829 - val_loss: 1.3013 - val_acc
uracy: 0.8242
Epoch 60/60
3342/3342 [==============================] - 28s 8ms/step - loss: 0.0453 - accuracy: 0.9850 - val_loss: 1.0828 - val_acc
uracy: 0.8409

<keras.callbacks.callbacks.History at 0x1c71c827fd0>
```

Save the model:

```
# Serialise model to JSON
model_json = model.to_json()
with open('model.json', 'w') as json_file:
    json_file.write(model_json)
```

```
# Serialise weights to HDF5
model.save_weights('model.h5')
print('Saved trained model to disk')
```

Figure 2. 4 Training Result

2.8 Initial Model Implementation

We add the prerequisites needed to run this notebook and load the model into a new Jupyter notebook that will be used to test the model in real time. Firstly, we load the json model. Secondly, print out the model summary for presentation purposes. Lastly, we load the model weights, as shown in Figure 2.5.

```
import keras
from keras.models import load_model
from keras.utils import CustomObjectScope
import tensorflow as tf
from keras.models import model_from_json
import numpy as np
import os
from keras.initializers import glorot_uniform
import cv2
import numpy as np
```

```
Using TensorFlow backend.
```

Loading in the Model

First, we open the JSON file for the Neural Network structure:

```
with open('model.json', 'r') as json_file:
    json_savedModel= json_file.read()
model = tf.keras.models.model_from_json(json_savedModel)
```

Figure 2. 5 Prerequisites & Loading Model

In order to test the model for emotion detection in real-time using the webcam we use haar cascade which is designed by OpenCV to detect the frontal face of a user. Secondly, we identify the emotion labels, allow python to access the webcam, Draw a frame around the user's face once it is identified, use the model to predict the emotion displayed, calculate the probability of the emotion displayed by multiplying the score by 100, and put the text on the frame that will border the user's face, as shown in Figure 2.6.

```
# Classes for each emotion
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy',
                  'Sad', 'Surprise', 'Neutral']


# Classifier for detecting faces in the video
face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Access to the webcam
cap = cv2.VideoCapture(0)

# Font for the text on the video
font = cv2.FONT_HERSHEY_SIMPLEX
```

Then, we activate the webcam and perform live analysis of emotion:

```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Grayscale the image and detect the face
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray)

    for (x, y, w, h) in faces:
        # Draw a rectangle around the faces
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        # Perform emotion calculation
        img = gray[x:x+w,y:y+h]
        model_img = cv2.resize(img, (48,48))
        input_img = np.reshape(model_img,
                               (1,48,48,1)).astype(np.float32)/255.0
        pred = model.predict(input_img)
        emotion = emotion_labels[np.argmax(pred)]
        score = np.max(pred)
        cv2.putText(frame,
                    emotion+"  "+str(score*100)+'%', (x, y),
                    font, 1, (0, 255, 0), 2)

        # Display the resulting frame
        cv2.imshow('Emotion Classifier', frame)

    # Press 'q' to close the webcam
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
# Once completed, deactivate the webcam and close all windows
cap.release()
cv2.destroyAllWindows()
```

Figure 2. 6 Webcam Testing for Emotion

Upon running the application using a webcam we can observe the emotions detected by the model. We have successfully managed to detect all 7 emotions that the model was trained to detect which are fear, happy, sad, surprise, anger, disgust, and neutral, as shown in Figure 2.7.

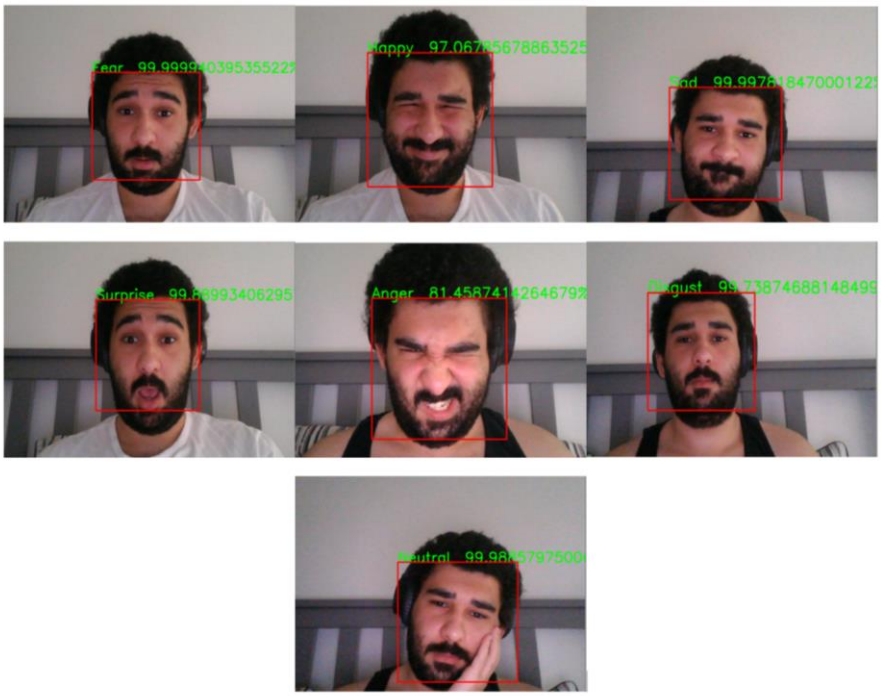

Figure 2. 7 Emotion Output

2.9 Depression Formula

The following formula has been applied to the emotion detection to be able to classify a user as depressed, and not depressed. The formula works by multiplying the coefficients mentioned with the emotion labels as shown anger=0.9, disgust=0, fear=0.2, happy=0, sad=0.8, surprise=0, and neutral=0.9, which is summed together to calculate the total probability. Furthermore, if the total probability is greater than 50%, it will be considered as "depressed", below 50%, will be considered as "not depressed". This formula has been improvised from the following research associated with depression (Admin, 2012), (Clinic, 2017), (Organization, 2020), (Prevention, n.d.), as shown in Figure 2.8.

```
# Classes for each emotion
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy',
                  'Sad', 'Surprise', 'Neutral']


# Classifier for detecting faces in the video
face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Access to the webcam
cap = cv2.VideoCapture(0)

# Font for the text on the video
font = cv2.FONT_HERSHEY_SIMPLEX

# Coefficients for each emotion when calculating the probability of depression
coefficients = [0.9, 0, 0.2, 0, 0.8, 0, 0.9]
```

Figure 2. 8 Depression Formula

2.10 Model Depression Recognition

The depression recognition implementation will require the following prerequisites, loading the model, and model weights into the Jupyter notebook, as shown in Figure. 2.9.

```
import keras
from keras.models import load_model
from keras.utils import CustomObjectScope
import tensorflow as tf
from keras.models import model_from_json
import numpy as np
import os
from keras.initializers import glorot_uniform
import cv2
import numpy as np
```

```
Using TensorFlow backend.
```

Loading in the Model

First, we open the JSON file for the Neural Network structure:

```
with open('model.json', 'r') as json_file:
    json_savedModel= json_file.read()
model = tf.keras.models.model_from_json(json_savedModel)
```

Then, we load in the weights of the model from the H5 file:

```
model.load_weights('model.h5')
```

Figure 2. 9 Loading Model & Weights Depression Implementation

In order to test the model for depression detection in real-time using a webcam we use Haar Cascade as we previously did with emotion testing, we draw a border around the detected faces on the webcam feed. Secondly, we apply our improvised formula to detect depression, as well as, implement code to display the output of the emotion to the side of the window, as shown in Fig 2.10.

```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Grayscale the image and detect the face
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray)

    for (x, y, w, h) in faces:
        # Draw a rectangle around the faces
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        # Perform depression calculation
        img = gray[x:x+w,y:y+h]
        model_img = cv2.resize(img, (48,48))
        input_img = np.reshape(model_img,
                               (1,48,48,1)).astype(np.float32)/255.0
        pred = model.predict(input_img)
        pred = np.around(pred[0], 2)*100
        ds = []
        for j, k in zip(pred, coefficients):
            ds.append(j*k)
        ds = np.sum(ds)
        nd = 100 - ds
        if ds > nd:
            result = 'Depressed'
        else:
            result = 'Not Depressed'

        # Show depression result above faces
        cv2.putText(frame, result, (x,y), font, 1, (255,0,0), 2)


    # List of emotions and probabilities on the side
    for i in range(len(emotion_labels)):
        cv2.putText(frame,
                    emotion_labels[i]+': '+str(pred[i])+'%', (0,100+20*i),
                    font, 0.5, (0,0,255), 1)

    # Display the resulting frame
    cv2.imshow('Depression Classifier', frame)

    # Press 'q' to close the webcam
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
# Once completed, deactivate the webcam and close all windows
cap.release()
cv2.destroyAllWindows()
```

Figure 2. 10 Webcam Testing for Depression

Upon running the application to detect if a user is “depressed”, or “not depressed”. We initially tried to recreate a positive emotional output to induce a “not depressed” result which detected the total sum of emotions multiplied by the mentioned coefficients to be below 50% which does not satisfy our improvised depression detection formula. On the other hand, we attempted to recreate a negative emotional output to induce a “depressed” reading which resulted in the total probability to be above 50% which satisfies our formula and displays a “depressed” state, as shown in Figure 2.11.

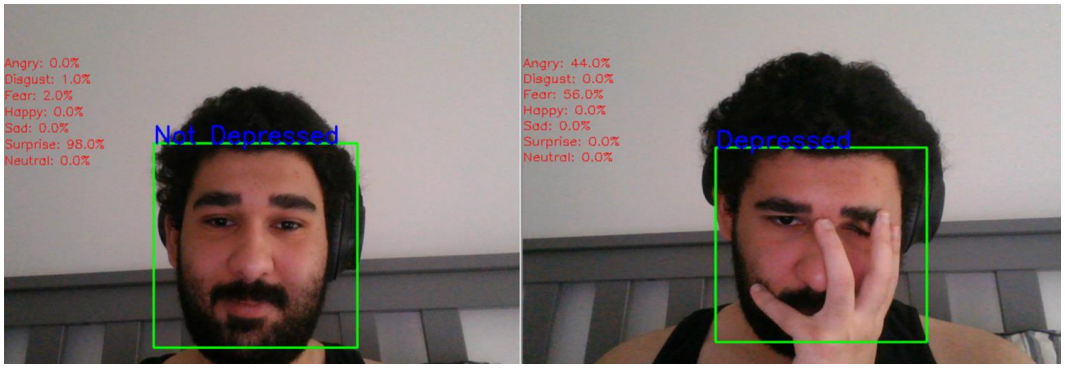

Figure 2. 11 Depressed/Not Depressed Output

3. Result & Discussion

The application testing regarding emotion detection was successful in terms of detecting the 7 emotions that the model was trained to detect. However, we faced some challenges during the testing stage due to the model rapidly adapting to a user’s facial expression which resulted in swift changes in readings which we describe as flickering through the different emotions. Moreover, there were some challenges in detecting a given emotion which is due to the dataset used to train the model which portrayed facial expressions that were not diverse enough to output an absolute result. This can be resolved by retraining the model using more diverse images that portray an emotion using different facial expressions.

The application testing using the improvised depression detection formula successfully detected the different emotion outputs and applied our formula to output if the conditions to be “depressed”, or “not depressed” was satisfied. The main challenge we faced during this phase of the project is the lack of real-world studies conducted to link emotional output with depression. This meant that our formula that was used to detect if a user is depressed or not, is not realistic. However, we have successfully manipulated the model to satisfy our needs in this project by applying a formula that multiplies the emotional weights output by the SoftMax layer by improvised coefficients which will be summed together. Once summed together if the result is above 50% that will indicate a “depressed” state and below 50% will indicate “not depressed” state.

4. Conclusion

In conclusion, the detection of emotion can be considered as a starting point to calculating the risk of suicide by recognizing depression. Furthermore, we applied an improvised formula to use the emotion output to be classified further into 2 states which are “depressed” and “not depressed”. The reasons people suicide is more complex than narrowing it down to emotional display. Emotion detection will detect the emotions that are associated with suicide. However, without analysing a person’s family history, drug use history, economical standing, and current situation in life It will be difficult to appropriately point out the risk of suicide a person emits. Moreover, depending on the individual’s history, they could lack proper expression of emotion that could be because of past abuse, lack of care or love from family, relatives and friends (Dominguez-Garcia & Fernandez-Berrocal, 2018). This project does not intend on making medical professionals redundant in this field. Instead, we aim to further improve our model in the future in the hopes of allowing the model to exceed its own limits as Ernest Hemmingway quoted “There is nothing noble in being superior to your fellow men. True nobility lies in being superior to your former self” (Hemingway, n.d.).

5. Future Work

This endeavour is not meant to outperform humans. Instead, outperform itself by training the model with a greater dataset. This endeavour scratches the surface of possibility in this field. Our imagination is our limit. There is no question that this project will be further developed by future academics due to the benefits this project will yield. This project can be further developed to detect emotion using a live video feed coupled with a voice recognition application of AI that can increase the accuracy of the result to provide instant results by allowing the model to constantly adapt to a facial expression that is constantly changing. Moreover, developing a scale that the model can use to analyse multiple factors that can affect the risk of suicide in a person such as financial situation, family history, individual history, and life satisfaction score (Dominguez-Garcia & Fernandez-Berrocal, 2018).

References

Admin, U. D. /. A. a. M. H. S., 2012. *Preventing Suicide: A Toolkit for High Schools.* Rockville: U.S. DHHS /Substance Abuse and Mental Health Services Admin. (2012).

Anon., 2020. *Project Jupyter.* [En línea]
Available at: https://en.wikipedia.org/wiki/Project_Jupyter
[Último acceso: 28 March 2020].

Brown, M., 2016. *Suicide Prevention in Alaska,* Rockville: Substance Abuse and Mental Health Services Administration.

Clinic, C., 2017. *Recognizing Suicidal Behaviour.* [En línea]
Available at: https://my.clevelandclinic.org/health/articles/11352-recognizing-suicidal-behavior
[Último acceso: 27 March 2020].

Dominguez-Garcia, E. & Fernandez-Berrocal, P., 2018. The Association Between Emotional Intelligence and Suicidal Behavior: A Systematic Review. *Frontiers in Psychology,* Volumen 9, p. 2380.

England, P. H., 2019. *GOV.UK.* [En línea]
Available at: https://www.gov.uk/government/publications/better-mental-health-jsna-toolkit/5-children-and-young-people
[Último acceso: 18 March 2020].

Etzioni, A. & Etzioni, O., 2017. Incorporaing Ethics into Artifcial Intelligence. *The Journal of Ethics,* 21(4), pp. 403-418.

Fonseka, M. T., Bhat, V. & Kennedy, 2019. The utility of artificial intelligence in suicide risk prediction and the management of suicidal behaviours. *Australian & New Zealand Journal of Psychiatry,* 53(10), pp. 954-964.

Frankish, K. & Ramsey, M. W., 2014. The ethics of artifcial intelligence. En: *The Cambridge Handbook of Artifcial Intelligence.* Cambridge: Cambridge University Press, pp. 316-333.

Hemingway, E., s.f. *Ernest Hemingway Quotes.* [En línea]
Available at:
https://www.brainyquote.com/quotes/ernest_hemingway_174758#:~:text=Ernest%20Hemingway%20Quotes&text=There%20is%20nothing%20noble%20in%20being%20superior%20to%20your%20fellow,superior%20to%20your%20former%20self.
[Último acceso: 3 June 2020].

Kaggle, 2013. *Emotion and Identity Detection from face images.* [En línea]
Available at: https://www.kaggle.com/c/facial-keypoints-detector
[Último acceso: 25 March 2020].

Organization, W. H., 2019. *World Health Organization.* [En línea]
Available at: https://www.who.int/en/news-room/fact-sheets/detail/suicide
[Último acceso: 17 May 2020].

Organization, W. H., 2020. *Suicide across the world (2016).* [En línea]
Available at: https://www.who.int/mental_health/prevention/suicide/suicideprevent/en/
[Último acceso: 26 March 2020].

Point, T., s.f. *SDLC-Waterfall Model.* [En línea]
Available at: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
[Último acceso: 15 March 2020].

Prevention, A. F. f. S., s.f. *Risk Factors and warning signs.* [En línea]
Available at: https://afsp.org/risk-factors-and-warning-signs
[Último acceso: 15 May 2020].

Union, E. P. a. C. o. t. E., 2018. *Intersoft Consulting.* [En línea]
Available at: https://gdpr-info.eu/
[Último acceso: 17 May 2020].

Verma, R., 2018. *Kaggle.* [En línea]
Available at: https://www.kaggle.com/deadskull7/fer2013/metadata
[Último acceso: 2 May 2020].

Watson, R., 2020. *Lords Library.* [En línea]
Available at: https://lordslibrary.parliament.uk/research-briefings/lln-2020-0032/.
[Último acceso: 26 March 2020].