Machine Learning Engineer Nanodegree

Facial Expression Recognition

Luiz Picanço
April 18th, 2019
Capstone Project

I. Definition

Project Overview

The identification of emotions in facial expressions is an intrinsically human trait, very important in social interactions and communication. Although a task considered easy for humans, it is considered very difficult for software to perform.

Software capable of performing this task satisfactorily has many applications, such as evaluating customer satisfaction in a market research and evaluation of a driver's mental fatigue driving a motor vehicle.

I consider this an interesting problem to solve, being in the field of computer vision, an area that I have a lot of interest.

This project and the datasets are based on this papers:

- Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution[1]
- Challenges in Representation Learning: A report on three machine learning contests[2]

Problem Statement

The problem to be solved by this project is the identification of which type of emotion is associated with a human face. The types of emotions that can be identified are: neutral, happiness, surprise, sadness, anger, disgust, fear and contempt.

The strategy to solve this problem is to train a neural network through a dataset with images of faces with the corresponding emotion identified. After training, this neural network should be able to predict emotions associated with facial expressions.

Metrics

For the evaluation metrics, I used a confusion matrix to identify false positives, false negatives, true positives, and true negatives. The F1 score was also used to measure the performance of the model[3]. For the training phase, I used a categorical cross-entropy function between predictions and targets to make the evaluation of the learning.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = 2(\frac{recall * precision}{precision + recall})$$

II. Analysis

Data Exploration

To solve the problem, 2 datasets were used:

FER2013 - Available on Kaggle, from the “Challenges in Representation Learning: Facial Expression Recognition Challenge” competition [4].
This dataset has 35,888 lines, with the following structure:

- emotion - Emotion numeric code associated with the image(0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). It was not be used in this project
- pixels - Image data in grayscale with 48x48 pixels.
- usage - Image utilization type(Training, PrivateTest, PublicTest). It was not be used in this project.

A sample of the first lines of the dataset can be seen below:

emotion	pixels	Usage
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
6	4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

FER+ - Available on Github [5]. This dataset also has 35,888 rows, with the following structure:

- usage - Same meaning of the usage column in FER2013 dataset. Was used to split the dataset into test, training and validation.
- neutral, happiness, surprise, sadness, anger, disgust, fear, contempt, unknown, NF - Number of votes that each emotion received. Unknown and NF(Not a Face) are not emotions, so they were not used.

A sample of the first lines of the dataset can be seen below:

Usage	neutral	happiness	surprise	sadness	anger	disgust	fear	contempt	unknown	NF
Training	4	0	0	1	3	2	0	0	0	0
Training	6	0	1	1	0	0	0	0	2	0
Training	5	0	0	3	1	0	0	0	1	0

Usage	neutral	happiness	surprise	sadness	anger	disgust	fear	contempt	unknown	NF
Training	4	0	0	4	1	0	0	0	1	0
Training	9	0	0	1	0	0	0	0	0	0

The FER+ is an enhanced version of the FER2013 dataset with 2 main advantages:

- Each image has been labeled by 10 crowd-sourced taggers, providing better quality than the original FER labels
- Since each image has 10 votes, each can have more than one emotion associated with it.

Exploratory Visualization

In Fig. 1 we can see the distribution of the emotions in the dataset. Most of the emotions are: neutral(30%), happiness(25.5%), sadness(13.2%) and surprise(11.6%).

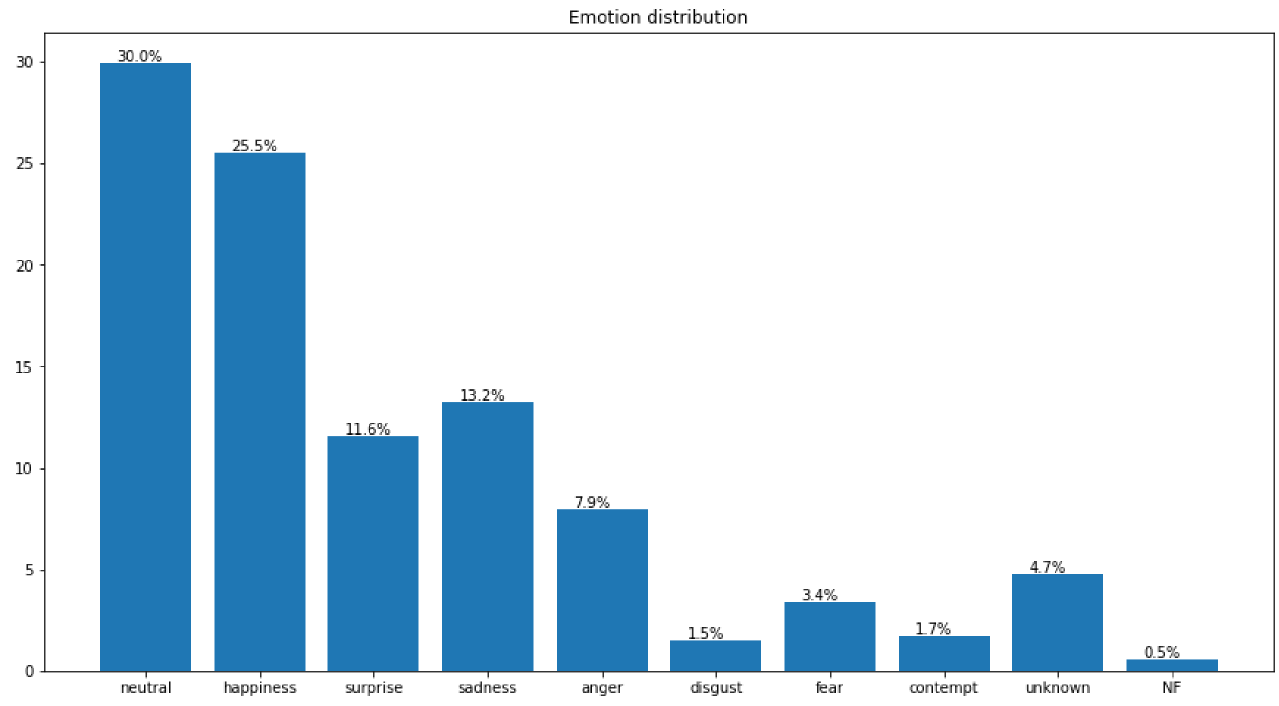


Fig. 1 - Emotion distribution in the dataset

In Fig. 2 we can see some of the faces present in the dataset

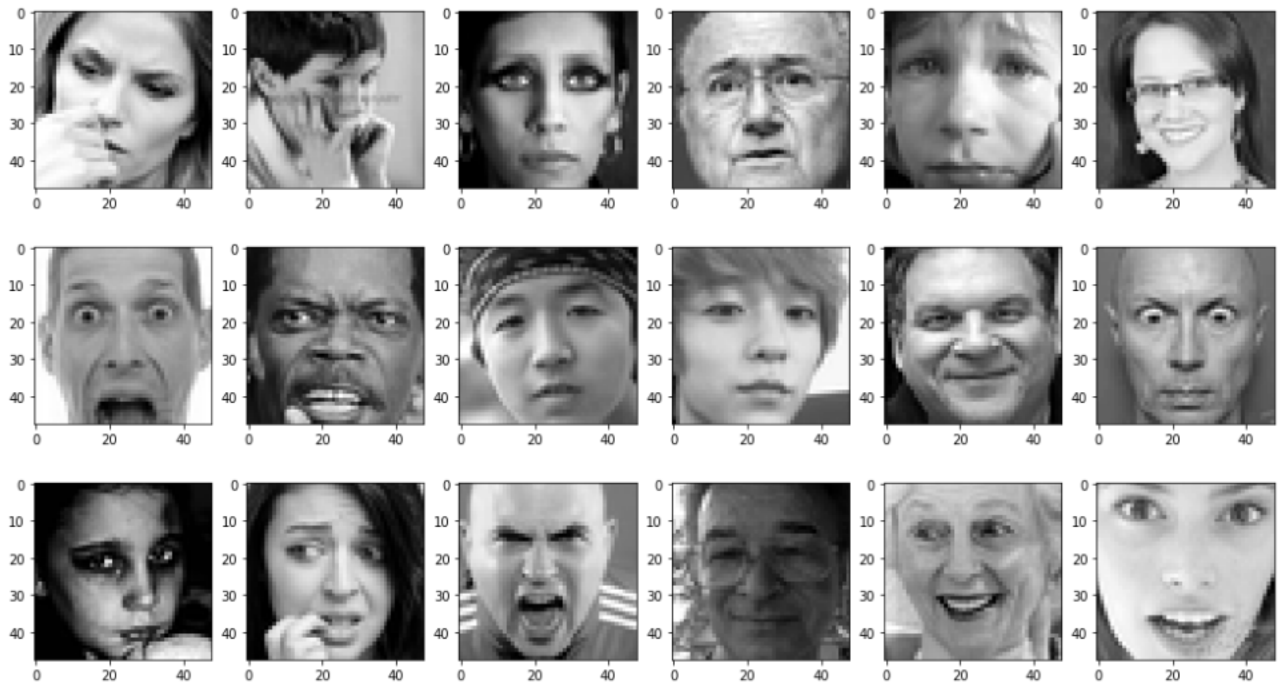


Fig. 2 - FER2013 sample faces

Algorithms and Techniques

For the solution of the proposed problem I combined the two datasets using only the pixels column of the first and the corresponding columns of the emotions of the second dataset. This derived data set was divided into 3 parts: training (80%), validation (10%) and test (10%).

For the construction of the prediction model, I used Deep Learning through the use of CNN (convolutional neural network). CNN is a type of neural network widely used in solving image classification problems.

Since the images in the dataset are 48x48 pixels grayscale, the first layer of the network has an input shape of 48x48x1. In the last layer I created an output with 10 nodes, each representing a different emotion. With softmax as an activation function, we can have more than one output on the network, associated with a probability.

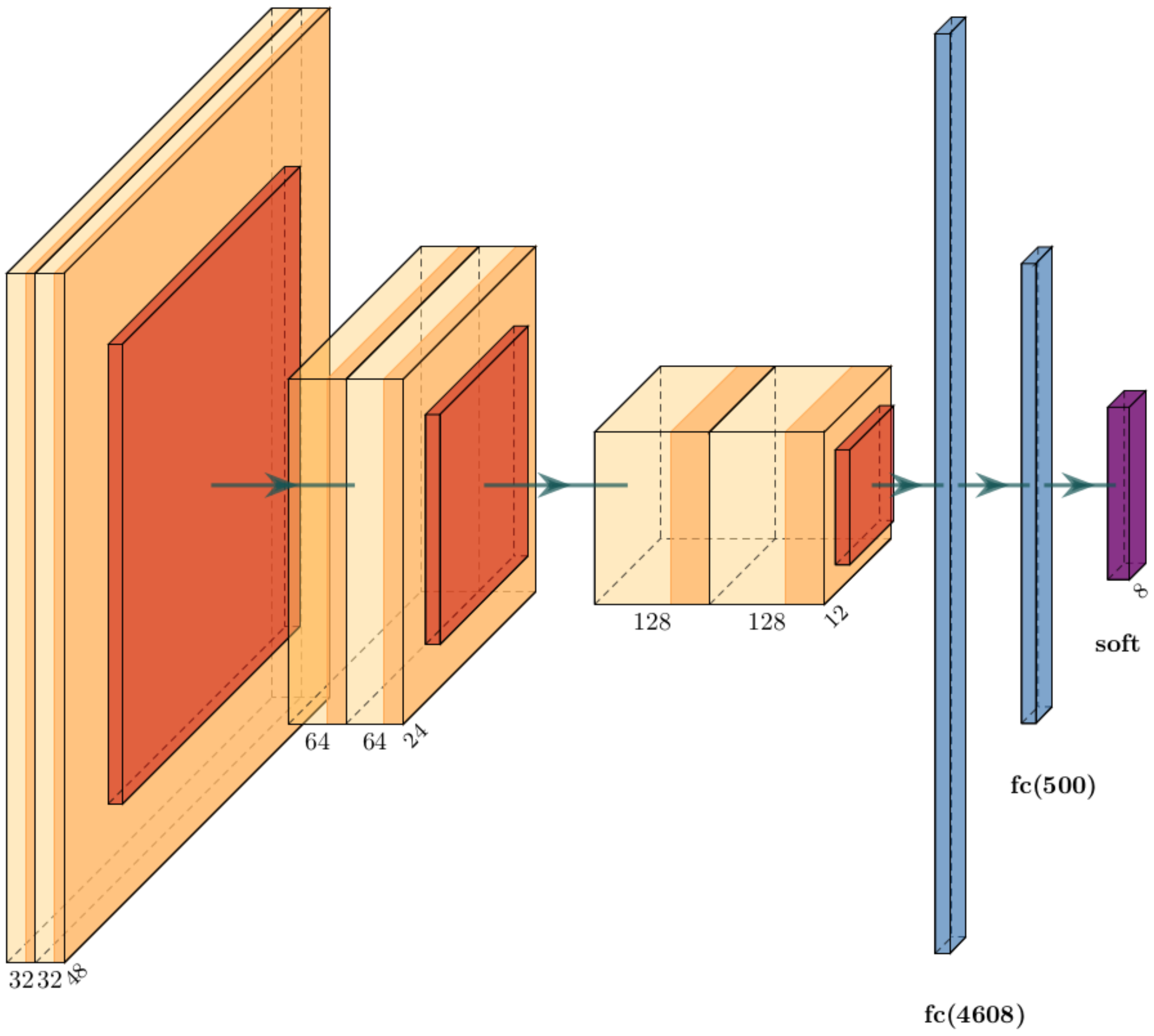


Fig. 3 - CNN model visual representation

Benchmark

Given the image of a human face, the model must predict a human emotion associated with that face. Since 30% of the dataset is associated with a neutral emotion, I will perform a naive approach that all the images in the dataset are related to that emotion.
With this approach, I obtained an acuracy of 32% and 0.06 in f1 score.
The results of the model will be compared with the naive predictor to evaluate the performance.

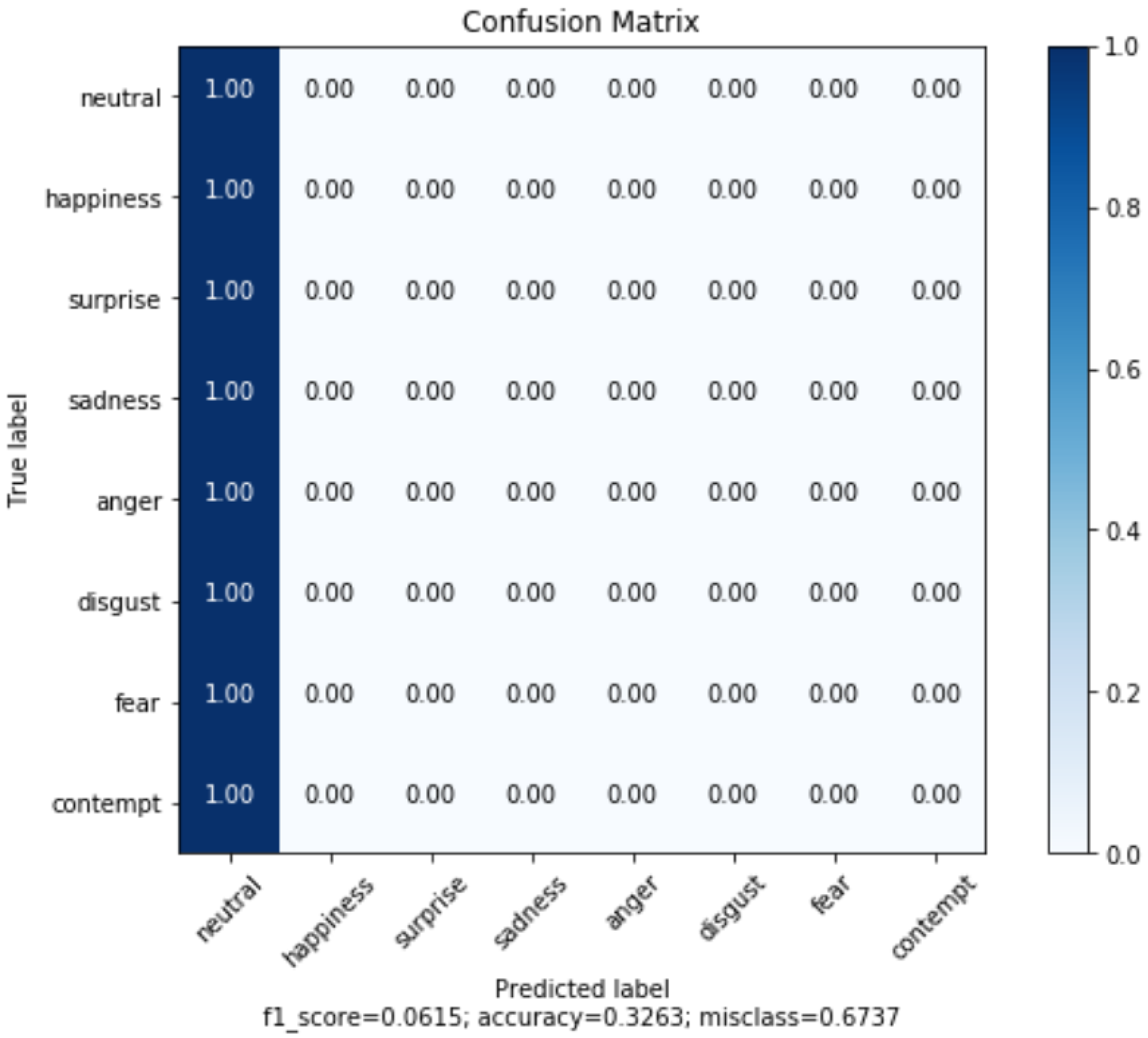


Fig. 4 - Naive prediction confusion matrix

III. Methodology

Data Preprocessing

Several steps were necessary in the pre-processing phase, with the objective of performing cleaning and transformations in the images to improve the training:

1. All rows in which only the “unknown” and “NF” columns received votes were removed. This is done to delete images that do not represent unknown faces or emotions
2. The 2 datasets were combined using the “pixels” column of the FER2013 and the “neutral”, “happiness”, “surprise”, “sadness”, “anger”, “disgust”, “fear”, “contempt” and “usage” of the FER+
3. The number of votes that each emotion received were transformed, being between 0 and 1
4. Using the “usage” column, the data set was splitted in 3 parts: Train, Validation and Test
5. The “pixels” column is a string with a list of integers between 0 to 255. This column was transformed into an array and each element of that array was divided by 255 to be in the range of 0 and 1. Also the array was converted into a 64x64x1 matrix, so as to fit the input of the neural network
6. In order to increase the variety of images during the training, image augmentation techniques was used, adding small random alterations on the images like rotation, inversion and zoom

7. The emotion associated with each image was determined randomly at the end of each training epoch, with probability relative to the number of votes received

Implementation

The implementation of the solution was divided into 4 parts: model construction, training, testing and prediction.

Model construction

In this phase of implementation a Sequential model was built with a input shape of 48x48x1, some convolutional layers, and an output shape of 8 with softmax activation.

Each of the 8 outputs represents an emotion and with the softmax function, this representation is the probability of the emotion being related to the input image. The combined probabilities of the outputs sum to 1.

CNNModel is a class that encapsulate the CNN model

```
class CNNModel:
    def __init__(self, input_shape, num_classes):
        self.input_shape = input_shape
        self.num_classes = num_classes

    def build_model(self):
        model = Sequential()
        model.add(Conv2D(32, (3, 3), padding='same', input_shape=self.input_shape))
        model.add(BatchNormalization(axis=-1))
        model.add(Activation('relu'))
        model.add(Conv2D(32, (3, 3), padding='same'))
        model.add(BatchNormalization(axis=-1))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Conv2D(64,(3, 3), padding='same'))
        model.add(BatchNormalization(axis=-1))
        model.add(Activation('relu'))
        model.add(Conv2D(64, (3, 3), padding='same'))
        model.add(BatchNormalization(axis=-1))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Dropout(0.25))
        model.add(Conv2D(128,(3, 3), padding='same'))
        model.add(BatchNormalization(axis=-1))
        model.add(Activation('relu'))
        model.add(Conv2D(128, (3, 3), padding='same'))
        model.add(BatchNormalization(axis=-1))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))

        model.add(Flatten())

        model.add(Dense(500))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(Dropout(0.25))
        model.add(Dense(self.num_classes))
        model.add(Activation('softmax'))
        return model
```

FERReader is a class that encapsulate the reading and pre-processing of the FER2013 and FER+ datasets

```python
class FERReader:
    def __init__(self, fer_path, fer_plus_path):
        self.fer_path = fer_path
        self.fer_plus_path = fer_plus_path
        self.emotion_columns = ["neutral","happiness","surprise","sadness",\
            "anger","disgust","fear","contempt"]
        self.emotion_count = len(self.emotion_columns)

    def read(self):
        df_fer2013 = pd.read_csv(self.fer_path)
        df_ferplus = pd.read_csv(self.fer_plus_path)

        df = self.__join(df_fer2013, df_ferplus)
        df = self.__clean(df)
        self.train_set, self.test_set, self.validation_set = self.__split(df)

        X_train = self.__convert_pixels(self.train_set)
        X_validation = self.__convert_pixels(self.validation_set)
        X_test = self.__convert_pixels(self.test_set)
        return (X_train, X_validation, X_test)

    def generate_emotions(self, df):
        return self.__encode_emotions(df)

    def __join(self, df_fer2013, df_ferplus):
        df_ferplus_columns = self.emotion_columns + ["Usage"]
        df_joined_data = pd.concat([df_fer2013.pixels, df_ferplus[df_ferplus_columns]], axis=1)
        return pd.DataFrame(columns=["pixels"] + df_ferplus_columns, data=df_joined_data)

    def __clean(self, df):
        num_votes = df[self.emotion_columns].sum(axis=1)

        #Removing rows where the total votes are <1.
        df.drop(df[num_votes < 1].index, inplace=True)

        # transform votes to probability(between 0 and 1)
        df[self.emotion_columns] = df[self.emotion_columns].div(num_votes, axis=0)

        return df

    def __split(self, df):
        train_set = df[df.Usage == 'Training']
        test_set = df[df.Usage == 'PrivateTest']
        validation_set = df[df.Usage == 'PublicTest']
        return (train_set, test_set, validation_set)

    def __convert_pixels(self, df):
        imgs = [np.asarray(row.split(" "), dtype=np.float32) / 255 \
            for _, row in enumerate(df.pixels)]
        return np.array(imgs).reshape(len(imgs),48,48,1)

    def __encode_emotions(self, df):
        emotions = [to_categorical(np.random.choice(self.emotion_count, p=row), \
            self.emotion_count) for _, row in df[self.emotion_columns].iterrows()]
        return np.array(emotions).reshape(len(emotions), self.emotion_count)
```

FERGenerator is a class that provide training data with image augmentation to the CNN

```
class FERGenerator(Sequence):
    def __init__(self, fer_reader, x_set, y_set, batch_size):
        self.fer_reader = fer_reader
        self.x = x_set
        self.y_set = y_set
        self.batch_size = batch_size
        self.y = fer_reader.generate_emotions(y_set)
        self.datagen = ImageDataGenerator(featurewise_center=False, rotation_range=10, \
            featurewise_std_normalization=False, rotation_range=10, width_shift_range=0.1, \
            height_shift_range=0.1, zoom_range=0.1, horizontal_flip=True)
        self.datagen.fit(self.x)

    def __len__(self):
        return int(np.ceil(len(self.x) / float(self.batch_size)))

    def __getitem__(self, idx):
        batch_x = self.x[idx * self.batch_size:(idx + 1) * self.batch_size]
        batch_y = self.y[idx * self.batch_size:(idx + 1) * self.batch_size]

        batch_x_modified = [self.datagen.random_transform(item) for item in batch_x]
        return np.array(batch_x_modified), batch_y

    def on_epoch_end(self):
        self.y = self.fer_reader.generate_emotions(self.y_set)
```

Training

In this phase of implementation the model was trained using the training set and the validation set.

The compilation parameters of the model that obtained the best results were the following:

- loss: categorical_crossentropy
- optimizer: adam

The following checkpoints were used:

- ModelCheckpoint - Save to disk the best performing model among training epochs. The validation set is used to make this evaluation
- EarlyStopping - Interrupts the training when the model stops learning after some epochs
- ReduceLROnPlateau - Reduces the rate of learning when the model stops learning after some epochs

Trainer is the class responsible for conducting the CNN training

```
class Trainer:
    def __init__(self, model_path, epochs=5000, patience=100, verbose=0, batch_size=32):
        self.epochs = epochs
        self.patience = patience
        self.model_path = model_path
        self.verbose = verbose
        self.batch_size = batch_size

    def train(self, model, fer_reader, X_train, X_validation):
        train_generator = FERGenerator(fer_reader, X_train, fer_reader.train_set,
            self.batch_size)
        validation_generator = FERGenerator(fer_reader, X_validation,
            fer_reader.validation_set, self.batch_size)
        model_checkpoint = ModelCheckpoint(self.model_path, verbose=self.verbose,
            save_best_only=True)
        early_stopping = EarlyStopping(patience=self.patience, verbose=self.verbose,
            restore_best_weights=True)
        reduce_lr = ReduceLROnPlateau(factor=0.1, patience=int(self.patience/5),
            verbose=self.verbose)

        callbacks = [model_checkpoint, History(), early_stopping, reduce_lr]
        return model.fit_generator(train_generator, validation_data=validation_generator,
            verbose=self.verbose, epochs=self.epochs,
            steps_per_epoch=len(X_train) / self.batch_size, callbacks=callbacks)
```

We can start the CNN training with the code below

```
    reader = FERReader("../datasets/fer2013.csv", "../datasets/fer2013new.csv")
    X_train, X_validation, X_test = reader.read()

    model = CNNModel((48, 48, 1), reader.emotion_count).build_model()
    model.compile(loss=keras.losses.categorical_crossentropy, optimizer="adam", metrics=['acc'])

    trainer = Trainer("../model/best_model.hdf5", verbose=1, batch_size=512)
    trainer.train(model, reader, X_train, X_validation)
```

Testing

In this phase the accuracy, f1 score and confusion matrix of the trained model was calculated

```
score = model.evaluate(X_test, y_test)
y_pred = model.predict(X_test)
plot_confusion_matrix(y_test, y_pred, reader.emotion_columns, normalize=True)
```

Prediction

In this phase the model is ready to make predictions
The opencv library[6] was used to crop the face part of the input photo before performing the prediction through the model

```
emotions = ["neutral","happiness","surprise","sadness","anger","disgust","fear","contempt"]

def predict(model_file, image_file):
    faces = process_image(image_file)
    model = load_model(model_file)
    predictions = model.predict(faces)
    print("Predictions probabilities:", predictions)
    prediction_labels = [emotions[np.argmax(predictions[i])] for i in range(len(predictions))]
    return prediction_labels

def process_image(image_file):
    face_cascade = cv.CascadeClassifier("haarcascade_frontalface_default.xml")

    image_gray = cv.cvtColor(cv.imread(image_file), cv.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(image_gray, 1.1, 6)
    print ("Found {0} faces in image".format(len(faces)))

    processed_faces = []
    for (x, y, w, h) in faces:
        cropped = image_gray[y : y+h, x : x+w]
        resized = cv.resize(cropped,(48,48))
        scaled = resized.reshape(48,48,1) / 255
        processed_faces.append(scaled)

    return np.array(processed_faces)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("model", help="trained model file")
    parser.add_argument("image", help="image file to predict")
    args = parser.parse_args()

    np.set_printoptions(suppress=True)

    predictions = predict(args.model, args.image)
    print("Predicted emotions: ", predictions)
```

The complete source code can be found at the project repository[7].

Refinement

After the initial development, some modifications to the solution were necessary to improve the results:

- Image augmentation - Using this technique through the use of the `ImageDataGenerator` class improved the results
- Changing the emotion associated with the image at each training epoch - Initially, the emotions associated with each image of the training set were being determined before the start of the training:

```
class FERReader:
    def read(self):
        y_train = self.__encode_emotions(df_train)

    def __convert_pixels(self, df):
        imgs = [np.asarray(row.split(" "), dtype=np.float32) / 255
            for _, row in enumerate(df.pixels)]
        return np.array(imgs).reshape(len(imgs),48,48,1)
```

Changing the emotion associated with each image at each training epoch, significantly improved the training results.

IV. Results

Model Evaluation and Validation

The representation of the final model can be seen in Fig. 3. We can see it in more detail below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 32)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
```

```
activation_1 (Activation)    (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 32)        9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
activation_2 (Activation)    (None, 48, 48, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
activation_3 (Activation)    (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
batch_normalization_4 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
activation_4 (Activation)    (None, 24, 24, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
batch_normalization_5 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
activation_5 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 128)       147584
_________________________________________________________________
batch_normalization_6 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
activation_6 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               2304500
_________________________________________________________________
batch_normalization_7 (Batch (None, 500)               2000
_________________________________________________________________
activation_7 (Activation)    (None, 500)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 4008
_________________________________________________________________
activation_8 (Activation)    (None, 8)                 0
=================================================================
Total params: 2,598,732
Trainable params: 2,596,836
Non-trainable params: 1,896
_________________________________________________________________
```

A jupyter notebook was generated with the results of the training performed[8]

After the training, the model achieved an accuracy of 71% and an F1 score of 0.48 with the test dataset.

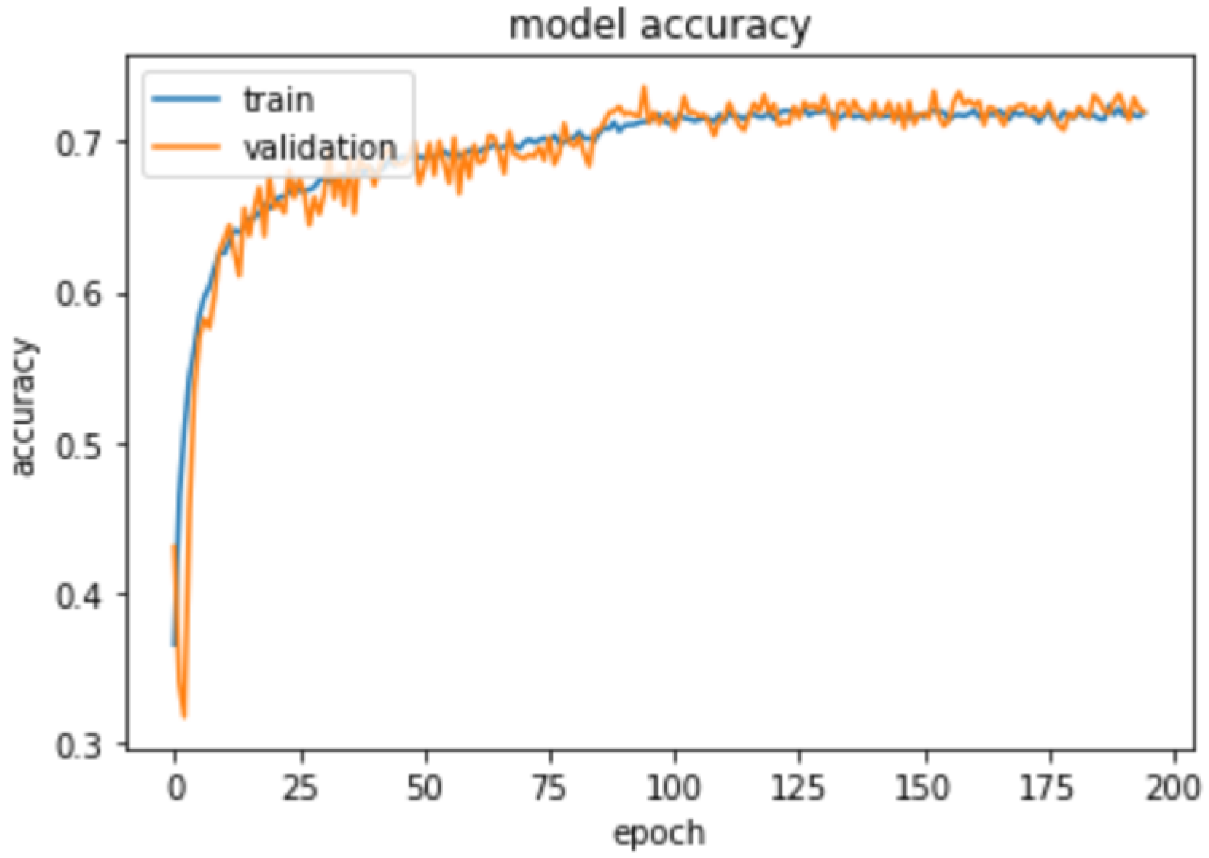


Fig. 5 - Training accuracy

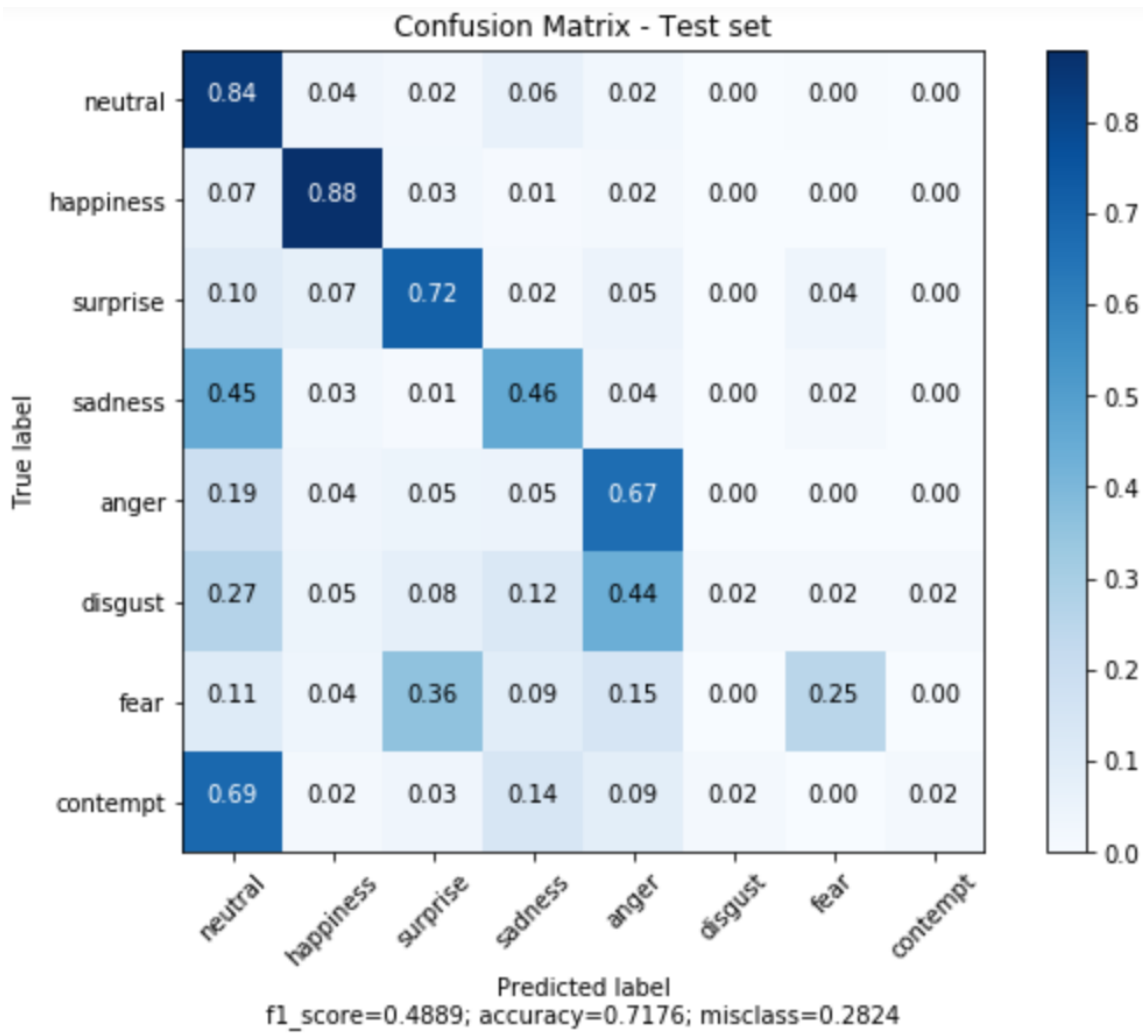


Fig. 6 - Test set Confusion Matrix

The trained model was tested with some images obtained on the internet. The model was able to accurately predict the emotions in all images. Images containing groups of people were also used.

The result of these tests can be seen in Fig 7.

Justification

Comparing Fig. 4 and Fig. 6 we can observe a significant improvement over the naive model:

Model	Accuracy	F1 Score	Misclass
Naive	0.32	0.06	0.67
CNN	0.71	0.48	0.28

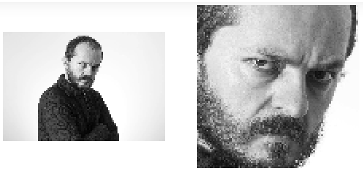
Despite the low performance with some types of emotions such as contempt and disgust, I believe the model is good enough to perform predictions of emotions in photos.

In Fig.7 some predictions are shown using images found on the internet.

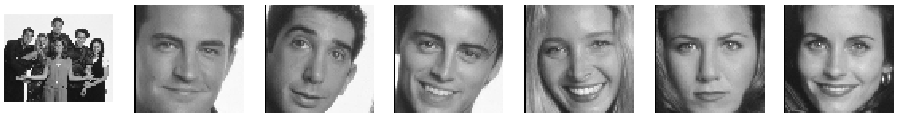
V. Conclusion

Free-Form Visualization

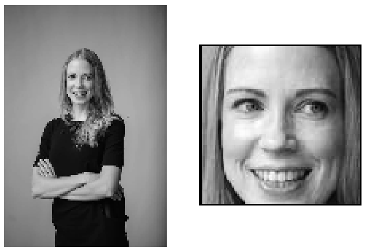
With some images obtained on the internet, the model was able to accurately predict the emotions present in them. An interesting feature is the ability to predict emotions of groups of people.



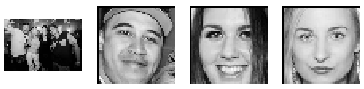
```
Found 1 faces in image
['anger']
```



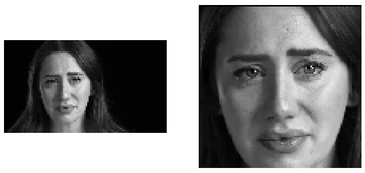
```
Found 6 faces in image
['happiness', 'neutral', 'happiness', 'happiness', 'neutral', 'happiness']
```



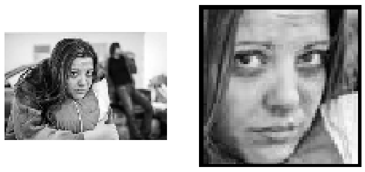
```
Found 1 faces in image
['happiness']
```



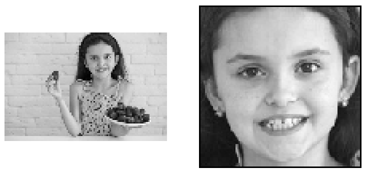
```
Found 3 faces in image
['happiness', 'happiness', 'neutral']
```



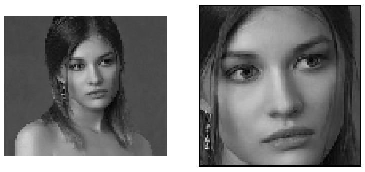
```
Found 1 faces in image
['sadness']
```



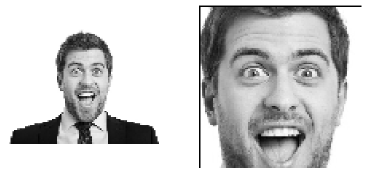
```
Found 1 faces in image
['sadness']
```



```
Found 1 faces in image
['happiness']
```



```
Found 1 faces in image
['neutral']
```



```
Found 1 faces in image
['surprise']
```

Fig. 7 - Emotions predicted by the trained model

Reflection

The process of building this project began with the search for a reliable dataset with a good variety of images. I found the FER2013 that contains a lot of images but with not very precise classification. A solution to this problem was the FER+: a dataset that complemented the FER2013 with the classification of each image being made by 10 different people.

After getting the datasets, the next challenge was to clean the dataset and transform the images to a format more friendly to processing by a CNN.

Once that stage was over, the next and biggest challenge was to develop a CNN model that could learn to classify emotions correctly. I did some experiments with DenseNet, Inception and VGG16 but didn't get satisfactory results. The best result I had was with the model presented, varying the emotions associated with the images at each training epoch, combined with the image augmentation techniques.

One aspect of this project that I found very interesting was the ability of some CNNs to converge relatively quickly to the expected result, which makes me think of the many practical applications in which they can be used.
My greatest difficulty was to be able to build a compact enough model that could converge satisfactorily to the expected results.

Improvement

I believe that better results can be achieved through the following suggestions:

- Improve the classification of images in the dataset, reclassifying or removing the outliers. Some emotions such “contempt” and “disgust” can benefit from these adjustments, since they have the worst results
- Search for other datasets that have a higher accuracy and variety
- In addition to performing the prediction from an image, also make predictions from videos or a webcam in real time

VI. References

- [1] Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution(https://arxiv.org/abs/1608.01041)
- [2] Challenges in Representation Learning: A report on three machine learning contests[(https://arxiv.org/abs/1307.0414)
- [3] An introduction to ROC analysis(https://people.inf.elte.hu/kiss/11dwhdm/roc.pdf)
- [4] Challenges in Representation Learning: Facial Expression Recognition Challenge (https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data)
- [5] FER+(https://github.com/Microsoft/FERPlus)
- [6] OpenCV library(https://opencv.org)
- [7] Project repository(https://github.com/lpicanco/nanodegree-machine-learning/tree/master/capstone)
- [8] Training results(https://github.com/lpicanco/nanodegree-machine-learning/blob/master/capstone/facial_expression_recognition.ipynb)