עבודת גמר במסגרת הקורס ״פרויקט גמר במדעי המחשב״, 11401

מוגשת לאפקה, המכללה האקדמית להנדסה בתל אביב

שם הפרויקט:
C.U
Emotion's recognition

ע״י

יואב רייצפלד - 205488182

רון יצחק - 204114631

שם המנחה: ד״ר דינה גורן-בר

אייר תשפ״ב תל אביב אפריל 2022

Contents

Acknowledgments....3

Abstract....4

Introduction....5

- Motivation....5
- Problem definition and main goal of the project....5

Lecture review....6

Architectures analysis....8

- Our selection....9
- Competitors....9

System Architecture....10

System Design....11

- Pseudo code....11
- Algorithm inputs....11
- Inputs encoders....11

Alpha product....15

- Algorithms and code description....15

Evaluation....18

- DataSet....18
- Metrics....18

Results....19

Conclusion....25

References....26

Acknowledgments

First, we thank Afeka Tel Aviv Academic College of Engineering for the opportunity to work on such an interesting project. Second, we thank Dr. Diana Goren Bar for her guidance through research and development of this work.

We are grateful to the teams at OpenAI for their work in training the CLIP model.

Another person we would like to mention is Dr. Eyal Katz. His passion, in-depth courses, and hands-on practice helped us choose an advanced project topic.

Finally, we’d also like to thank the developers of the vast software packages use us throughout this project.

Abstract

In our project, we will first search for the right topic, a topic that we desire to find and learn more about. After some deliberation, we decided to focus on emotion recognition. We feel the need for the teachers to recognize student emotions that can identify in images taken in the classroom environment, in which there is a great deal of variation in ambient light, quick changes in student positions, and many obstacles to overcome for the standard recognition model.

We researched the different models of emotion recognition, the challenges they face, and the best way to approach them. After completing our investigation into the architectures of emotion recognition models, we understood the pros and cons of each option. This made it possible for us to make a wise decision about which architecture would be most beneficial and yield the best results in emotion recognition.

We chose the CLIP architecture from OpenAI. It's a newly available model, and using it is revolutionary. Instead of the training model for one task, they combine remarkable text and image features extractors to solve matching problems. It has been trained over 400 million pairs of images and sentences to predict the most relevant text snippet given an image and can be used as a classifier model that can provide good models.

The CLIP model works by comparing an image with its corresponding sentences and calculating for each of them a cosine similarity score, with higher scores indicating image-sentence pairs that are more likely to be semantically related.
We are using a pre-trained model which we have to transfer into the emotion's recognition field. We do this by adding a decision-making layer to extract emotion recognition detections from the similarity of the sentences to the given image. We have also designed a wrapper class to enable multi-sentence input with more than one label. This was our primary exploration of this project. We try to answer two questions:

First, we explored whether any different combinations of sentences affected CLIP's results when detecting emotions. Next, we wanted to see whether extending the seven main classes of the FER competition would give a better detection result.

Our research assumptions were correct. The model with the best metrics in all tests included three-sentence templates, and we expanded the main classes with three synonyms.

Introduction

Motivation

In today's world, the covid-19 virus has changed the way we study, do business, interact with people, and basically changed every aspect of life that we knew until then. So, we have to accept that online meetings will be a central tool for communications, and this technology won't disappear in the future, but there is a problem. According to different surveys, our communication is mainly based on nonverbal facial expressions, these small and rapid changes in the face that sometimes blur due to various problems in online meetings.

When we entrust the future of next-generation education with online classes, we must come up with solutions for a teacher who is away from his students. He must know in real-time what the student is feeling on the other side of the screen. We mustn't forget that it's much easier for a student to lose concentration and interest if he isn't involved in the lesson. Overcoming this problem of missing out on facial expression recognition has significant implications, we need to equip the teacher with tools that will help him understand this easily. This is just one example of an area that is affected by emotion recognition, there are many like in business meetings or streaming platforms. Everyone wants to know what the other side of the screen feels like.

Problem definition and main goal of the project

The main goal is to find an algorithm for recognizing facial emotions that can detect in "difficult" conditions such as: close and large object, lighting conditions and brightness, angles, position, or cover and hiding parts of the face. Unfortunately, collecting and tagging the images is long, exhausting, and resource intensive. Furthermore, the process requires complex copyright protections, and although there are many datasets, most of them are not easily accessible to the public. Because of these issues, there aren't many options to choose from, and one of them has its problems like reflecting a particular group and ideal faces like JAPPE or large datasets like FER2013, whose images are very small and B/W colors. Although training a deep learning model on those datasets already exists, we doubt in their successes in real-time scenarios, they are probably different from what they saw and taught.

From our project scope, there are some problems that arise in recognizing emotions in facial expressions for example there is a similarity in facial expression between certain emotions like fear and surprise, and even one emotion can be described in some ways, a particular feeling like joy and pleasure that describe a happy person. Therefore, we will limit the categories to seven main emotions to test the algorithm, with a list of additional words describing them added to each main category.

Lecture review

There was a vast improvement in the field of facial emotion recognition across the years. The typical way is to use CNN models to extract essential features and recognize the emotions in each image. As part of our work, we tried to search for the more recent papers in this field. As part of the research, we read studies which combined attention mechanisms and tried to detect micro understanding of the image. They succeed in focusing on the region of interest (ROI) instead of the whole image.

In the paper "A visual attention-based ROI detection method for facial expression" [1], an eleven-layered Convolutional Neural Network with Visual Attention is proposed for facial expression recognition. The network is composed of three components. The first part is a stack of ten convolutional layers that extract features of faces. Second, the embedded attention model automatically determined the ROIs. Third, the local features in these regions are aggregated and used to infer the emotional label. This is a small model that can be trained in an end-to-end process, which means the net also learns to recognize the ROIs in the images and score them higher than other parts of the image.

In "frame attention networks for facial expression recognition in videos." [2] The Frame Attention Networks (FAN) is composed of two modules. a CNN, which embeds face images into feature vectors, and self-attention mechanized. They trained him on sequences of images, and his goal was to identify emotions in the video. The frame attention module is learning multiple attention weights. Those weights are used to adaptively aggregate the feature vectors to form a single discriminative video representation.

Another paper proposes new end-to-end network architecture for facial expression recognition with an attention model, At "FERAtt: facial expression recognition with attention net" [3], the innovations they suggested were to use facial image correction and attention, other than just CNN. Combining an encoder-decoder network with a convolutional feature extractor, they succeed in segmenting faces from the image and predicting this instead of the original image. Moreover, the encoder-decoder network obtains a feature attention map.
The researchers classify categories by obtaining an embedded representation and classification of the facial expression.

The paper "Region attention networks for pose and occlusion robust facial expression recognition " [4], for the first time, addresses the real-world pose and occlusion robust FER problem. They proposed a combined solution in several aspects: to simulate real-world occlusions and variant poses. They annotate several in-the-wild FER datasets with pose and occlusion attributes. Secondly, propose a novel Region Attention Network (RAN) to adaptively capture the importance of facial regions for occlusion and pose variant FER. The RAN aggregates and embeds a varied

number of regions features produced by a backbone CNN into a compact fixed-length representation.
Last, inspired by the fact that facial action units mainly define facial expressions, they propose a region-biased loss to encourage high attention weights for the most critical regions.
They train and test the RAN model and the region-biased loss on both annotated datasets to get precise results for pose and occlusion images and four popular datasets to compare with other FER models.

The OPENAI researchers introduced a neural network called CLIP "Contrastive Language-Image Pre-Training" [5] based on language and vision transforms. The intuition behind this work differs from the "traditional" approaches, which use convolution layers to extract the features. Instead, they use an attention mechanism. But one of the conditions when training a transformer is that a vast amount of data is needed. For this purpose, in their study, they have collected an enormous pair number of images and descriptions, a total of 400 million pairs, from the internet.

Furthermore, they trained the model on the Contrastive Learning methodology. Contrastive learning is an approach to formulate the task of finding similar and dissimilar things. They do zero-shot learning to predict the most relevant text snippet, given an image, without directly learning to distinguish between classes. In simple words, they trained a model for matching the photo to is corresponding description, at the same time, reject the rest of the descriptions that didn't describe it. In the article they used transformers to get an embedded representation in both the image and the text and calculated the distance between them. In practice, it is a cosine similarity between image vector and text vector, and while in training demanded that the product score between them be the highest. After training the model and getting the weights, they have a robust model that can tell which sentence from the sentences list best describes the image. By using the same template sentence and switching classes (or words) in it, they manage to use it as a classifier.

After we had done researching of what available architectures exists, we summarized the result in the table below. With that information, we could understand the pros and cons of each option and make a wise decision.

Architectures analysis

Architecture	Advantage	Disadvantage
computer vision approaches	• Does not require special devices to detect • Simple to use and well-known techniques	• Can't generalize well • Preprocessing and facial features are generated handcrafted
CNN	• Compared to the classic approaches of computer vision, it does not depend on image construction and hard pre-processing. • Learning accurate patterns and insights from the whole image • Immensely popular and widely used • Can speed up by using GPUs	• Computational power • Complex architecture • Learns from the whole image, which is not necessary to identify emotions. That can easily cause errors because most areas of the face don't change between emotions. • It requires many tagged images and succeeds in a specific task.
RNNs/LSTM + CNN	• Processing a sequence of images instead of only a single image • There is a memory of the previous state that allows to better identify emotion with more focus on changes in the face • Work on videos	• Although expressions are composed as a sequence of face Images, emotions are fully revealed at apex frames for a short amount of time, and the rest of the frames are not essential to classify. • RNN depending on the previous state and because of that cannot be parallelized.
CNN + attention	• classify emotions based on relevant areas in faces and not the whole image • Achieved SOTA results in emotions recognition benchmarks • Does not require additional data for training	• Additional learning parameters • Complexity
CLIP	• Classifies a wide range of categories by learning directly from the raw text about images. It is not trained on a fixed number of categories as a CNN does. • Already trained and ready to use!	• struggles with more abstract and systematic tasks • It will not work if it is not saw the image category before in it is pre-training dataset. Such as detect models of cars, species of flowers, and variants of aircraft.

Our selection

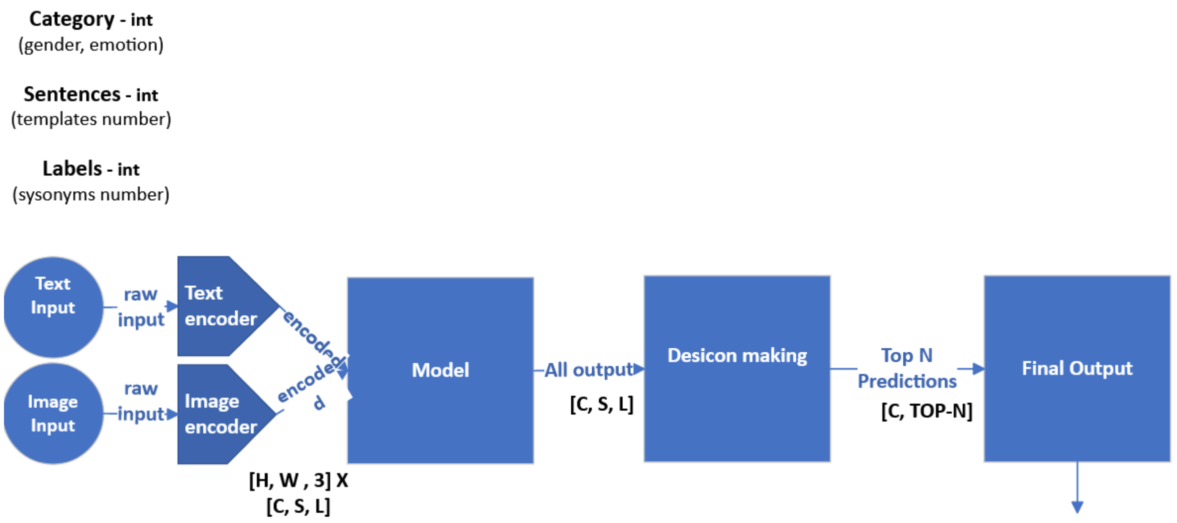
Interestingly, all the architectures we looked at were using CNNs as a backbone, and the improvement between them was when added additional components to the algorithm. This kind of algorithm needs a dataset to train. Still, the existing datasets were not particularly useful. They had drawbacks which they were either too small or the similarity was too big between images and sometimes contained ideal face images which caused the model to be non-general. We want to help teachers recognize student emotions in the classroom. This will allow students to feel more engaged during lessons and be more involved in what their teachers says and teach. Finding a sufficiently large dataset of faces that resemble students' appearances proved to be much harder than what we thought at the beginning. However, the clearest advantage of CLIP architecture was the images utilized in training, the images were taken from the Internet, almost endless dataset. In some of the images, it was even possible to perceive how humans appears today. Others were used to assign an emotional description to images, which is the same thing we would expect to see in a school. The CLIP model is known to find an image's best explanation for given input sentences. by adjusting its input sentences, the model's ability to recognize can be leveraged to emotional detections. The researchers also reported that it achieved state-of-the-art performance on several image classification challenges. We, therefore, decided to use it for the specific goal of facial emotions recognition.

Competitors

MorphCast - MorphCast is a user friendly software solution and also can be used on web, providing SDK solution to mainly detect emotions and also other interactions that are based on facial factors like attention, engagement, age, geolocation, device and other viewer attributes.

Noldus FaceReader – a software program for facial analysis. It is long time in the market and has been used in the more than 300 sites where it is used worldwide, their algorithm also mainly detect facial emotions.

System Architecture



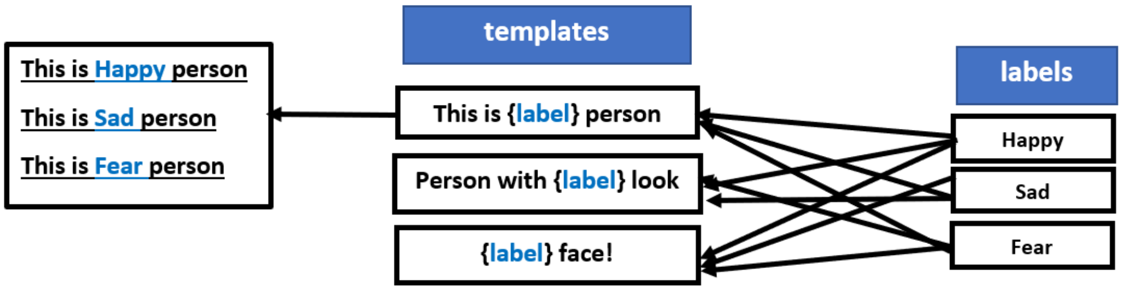
The above architecture describes the main data flow in our suggested algorithm, which uses a dictionary containing information about our labels and templates to generate sentences and images for matching. First, we create a sentences combination list with a size of all sentences list, each sentence generated from a template and using all the labels—that is how we define its possible permutations. The input image needs to be processed in the same way the researchers preprocessed the data during training. After receiving all sentences and the image will proceed to the encoding phase, where we get the embedding representation vectors. Next, the model will compute the similarity score of the sentence with the image to select which sentences present the given image the best. Although the model cannot understand the meaning of what he is doing, we create an index to a link between sentences (or labels) to one of the seven main emotion categories. Briefly, the model spits out a scores vector associated with the embedded sentences vector that goes into the model. We can find the categories that best describe the image using the index (labels-to-7categories) we created. In the decision-making stage, we will determine the top results. However, to ensure that our results are unique, we folded back synonyms and united them into seven main ones. This step is essential since otherwise, we would get the same class in every place in the TOP3.

System Design

Pseudo code

```
// Inner call for generate sentences function
model <- clip(labels, sentences)

model.encode_text()

for image in dataloader:
    model.encode_image(image)

    // Inner call for desicon making function
    result <- model.infer()

    // Parse the results
    gender <- result.get('gender')
    emotion <- result.get('emotion')
```

Algorithm inputs



We pull one image from the dataset, preprocessing and resize it to [224, 224, 3], and then normalizing it with ImageNet values. Example of how the algorithm generate sentences:

Inputs encoders

Word embeddings

Deep learning models trained on natural language processing tasks can process arrays of numbers as input instead of raw data as sentences. Therefore, the first challenge is to convert sentences into formatted arrays. There are several strategies for doing this:

- One-hot encodings - To represent each element of the vocabulary, one-hot encodings create a zeros vector with a length equal to the vocabulary and then place a one in the index corresponding to the word.
- Encode each word with a unique number.
- Word embedding- Word embeddings represent words as vectors, with similar words being encoded to a closer vector distance.

The second alternative, called indices, is more efficient. Instead of a sparse vector representing a dense vector, it uses indexes to define the vocabulary. However, the integer-encoding approach doesn׳t capture the complex relationship between words. Encoding sentences in arbitrary indexing make it difficult for models to interpret, and the fact that the similarity between word pairs is unrelated to their encoding makes this particularly problematic. The third approach embeds words using an embedded layer, which is trained to embed sequences into floating-point mathematical combinations. This allows for higher-dimensional embeddings and richer word associations of the sequence words.

Attention Mechanism

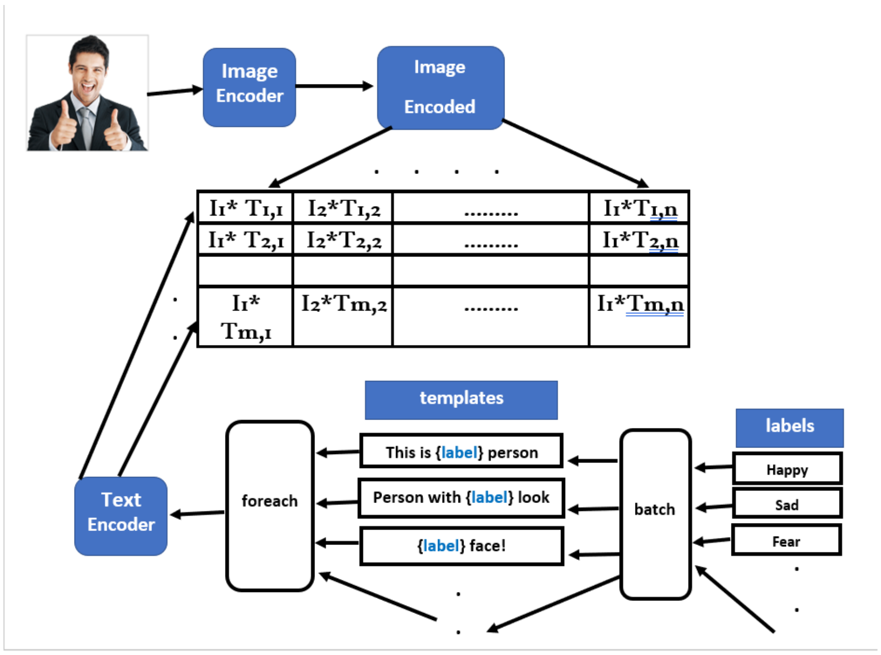
The attention mechanism enhances the crucial parts of the input data. Unfortunately, CNN doesn׳t encode the relative position of different features, so larger filters are required to encode combinations of these features. Increasing the size of the network׳s convolutional filters can increase its representative capacity but doing so also departs from the computational and statistical efficiency attained by using local processing. CNN architectures have limitations. Self-Attention helps model dependencies between image regions. A self-attention layer updates each sequence component by aggregating global information from the complete input sequence. This mechanism produces the encoding for words by a weighted combination of all word embeddings without knowing its position.

Text and vision transformer

Researchers transform the input sequence into a ״bag of words״ and pass it through an embedding transformation into continuous Euclidean space for finding the embedding representation vector. A bag-of-words operation eliminates the order of a sentence׳s words, but embedding words allows similarly structured phrases to be represented closely. The paper used a positional encoding technique, adding small constants to account for word position differences resulting from the same word appearing in different locations in a sentence. These small constants will help the transformer understand the relative position of each token in the sequence.

Furthermore, the transformer architecture can be applied to the field of vision with some modifications. Transformer-based models can process sequential data tokens instead of spatial non-sequential grid-structured data as CNNs do. We can refer to the image as sequence tokens and pass it through transformer blocks by slicing the image into patches and applying the transformer rules.

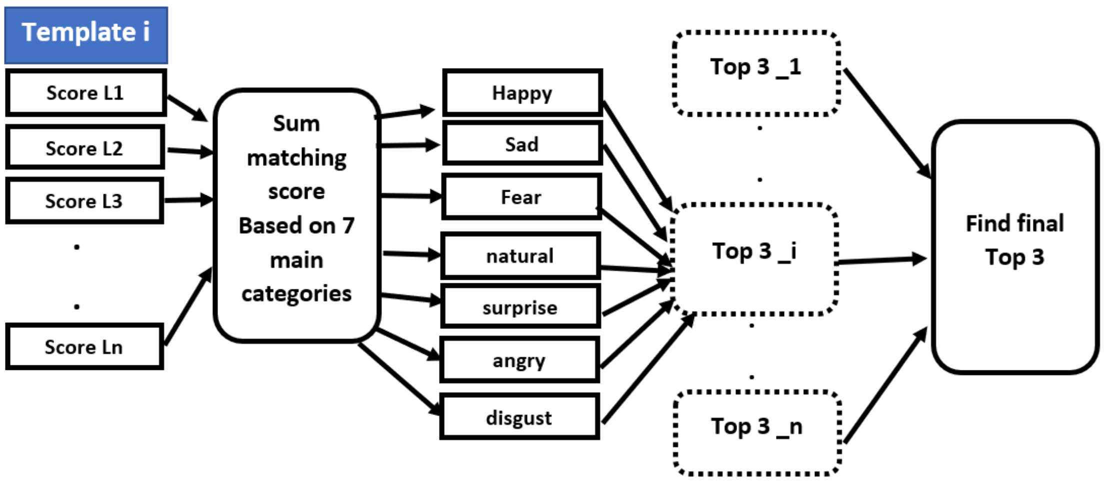
The model



CLIP work aims to train a Deep Learning model to match the best description. The inputs are used to create the embedding representation, performed through an Encoding process. The matching concept is part of the Contrastive Learning approach and calculating the Cosine similarity of each pair is the way the model works. The model is not restricted to a final number of classes and using zero-shot learning means that the model learns multiple concepts from the image through raw texts of the image. To be successful in the classification task (which involves distinguishing between categories), a model must identify the best sentence among all the possible sentences as an accurate description of an image. For the facial emotion recognition challenge, we need to provide a set of emotional labels to give the model options for matching an image.

As part of our study, we considered how varying the input sentence and the word labels changed the model's results. The model's predicted labels that can vary according to the input sentence. We explore whether the best labelling scheme and sentence form yield accurate predictions. Additionally, we explore other properties in images by identifying people's genders. There is a standard classification by the facial-emotions-recognition challenge to divide emotions into seven categories. We think it would be better to describe the feelings people experience in different circumstances using different words. The list is far from complete, but we've tried to use synonyms and grammar changes to expand the conventional list of seven emotions into a more extensive vocabulary that could describe many more feelings.

Decision making



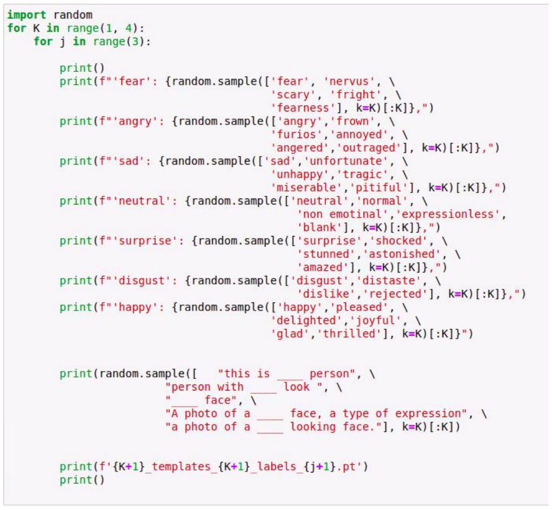
The original purpose of CLIP was to pair a sentence and an image based on a matching score, as opposed to CNN's approach, which used fully-connected layers and a softmax layer for predicting labels. To use the model as a classifier, we must manipulate its results. We changed the architecture to allow for more than one sentence, and each category now has multiple options, but the final predictions are selected from the original seven primary emotions. As shown above, After extending the templates to create sentences with specific labels, we decided which option generated the most proper sentence by summing the matching scores of all possible options for each original seven primary emotions and finding those who got higher scores. We call it the local top 3. After finding all of them (top3), will count how many times each category appeared in each rank (first, second, or third) and select the class with the most appearances at the rank.

Alpha product

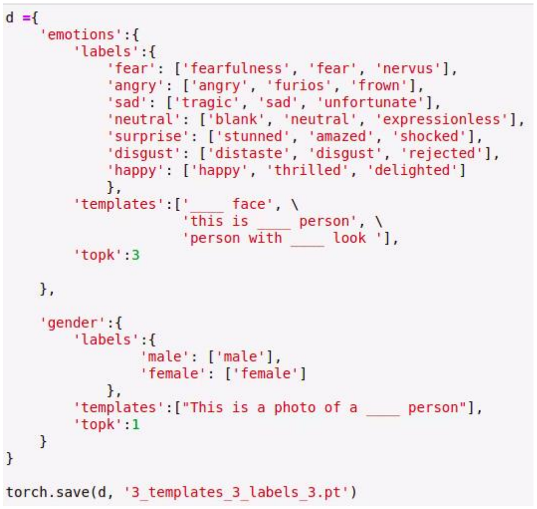
Algorithms and code description

At the beginning of our work, the code was running the simple 7 emtoion labels combined with only one sentence to test the work of the model. To find a way to improve the model work, we enlarge the number of labels and templates and kept it in dictionary data structure that will contain for example in the following screen shot: **3**_templates_**3**_labels_**3**.pt

3-templates for sentences created,3 labels for each category and the last 3 will be the third random option off all the labels inside the category.



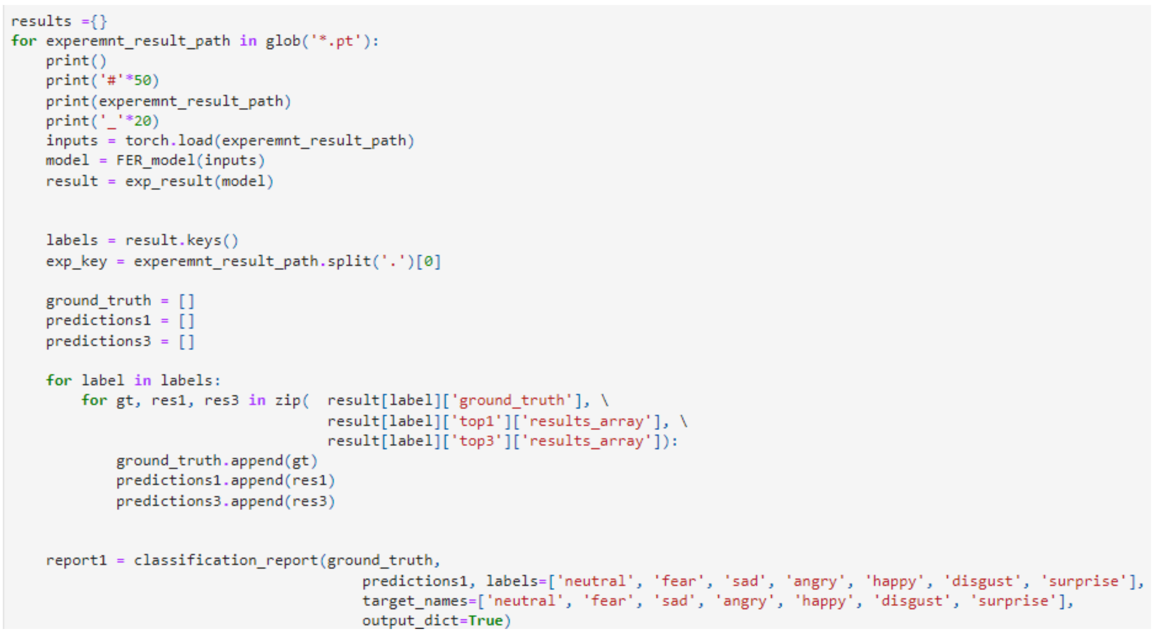
```python
import random
for K in range(1, 4):
    for j in range(3):

        print()
        print(f"'fear': {random.sample(['fear', 'nervus', \
                                      'scary', 'fright', \
                                      'fearness'], k=K)[:K]},")
        print(f"'angry': {random.sample(['angry','frown', \
                                      'furios','annoyed', \
                                      'angered','outraged'], k=K)[:K]},")
        print(f"'sad': {random.sample(['sad','unfortunate', \
                                      'unhappy','tragic', \
                                      'miserable','pitiful'], k=K)[:K]},")
        print(f"'neutral': {random.sample(['neutral','normal', \
                                         'non emotinal','expressionless',
                                         'blank'], k=K)[:K]},")
        print(f"'surprise': {random.sample(['surprise','shocked', \
                                         'stunned','astonished', \
                                         'amazed'], k=K)[:K]},")
        print(f"'disgust': {random.sample(['disgust','distaste', \
                                         'dislike','rejected'], k=K)[:K]},")
        print(f"'happy': {random.sample(['happy','pleased', \
                                      'delighted','joyful', \
                                      'glad','thrilled'], k=K)[:K]}")


        print(random.sample([   "this is ____ person", \
                            "person with ____ look ", \
                            "____ face", \
                            "A photo of a ____ face, a type of expression", \
                            "a photo of a ____ looking face."], k=K)[:K])


        print(f'{K+1}_templates_{K+1}_labels_{j+1}.pt')
        print()
```



```python
d ={
    'emotions':{
        'labels':{
            'fear': ['fearfulness', 'fear', 'nervus'],
            'angry': ['angry', 'furios', 'frown'],
            'sad': ['tragic', 'sad', 'unfortunate'],
            'neutral': ['blank', 'neutral', 'expressionless'],
            'surprise': ['stunned', 'amazed', 'shocked'],
            'disgust': ['distaste', 'disgust', 'rejected'],
            'happy': ['happy', 'thrilled', 'delighted']
        },
        'templates':['____ face', \
                     'this is ____ person', \
                     'person with ____ look '],
        'topk':3

    },

    'gender':{
        'labels':{
                'male': ['male'],
                'female': ['female']
        },
        'templates':["This is a photo of a ____ person"],
        'topk':1
    }
}

torch.save(d, '3_templates_3_labels_3.pt')
```

Creating the experiment with repeat of all the structures in the folder

The following screenshot will demonstrate the import of the data structure to the model and the running of it:



```python
results ={}
for experemnt_result_path in glob('*.pt'):
    print()
    print('#'*50)
    print(experemnt_result_path)
    print('_'*20)
    inputs = torch.load(experemnt_result_path)
    model = FER_model(inputs)
    result = exp_result(model)


    labels = result.keys()
    exp_key = experemnt_result_path.split('.')[0]

    ground_truth = []
    predictions1 = []
    predictions3 = []

    for label in labels:
        for gt, res1, res3 in zip(  result[label]['ground_truth'], \
                                    result[label]['top1']['results_array'], \
                                    result[label]['top3']['results_array']):
            ground_truth.append(gt)
            predictions1.append(res1)
            predictions3.append(res3)


    report1 = classification_report(ground_truth,
                                    predictions1, labels=['neutral', 'fear', 'sad', 'angry', 'happy', 'disgust', 'surprise'],
                                    target_names=['neutral', 'fear', 'sad', 'angry', 'happy', 'disgust', 'surprise'],
                                    output_dict=True)
```

Then inside the model the inputs that brought to the model get all generated to proper sentences that after will be encoded with embedding and saved as part of the model for the next steps to work with the given images.

How sentences get generated inside the model:

```
    def generate_sentences(self, index2label, templates):
        all_sentences={}

        all_labels = [item for _, vals in index2label.items() for item in vals] #list with all labels

        for i, template in enumerate(templates):
            all_sentences[i] = [self.put_name(template, label) for label in  all_labels]

        return all_sentences, all_labels
```

```
emotions:
sentence 0:
        - this is frightened person
        - this is fear person
        - this is angered person
        - this is frown person
        - this is pitiful person
        - this is tragic person
        - this is non emotinal person
        - this is neutral person
        - this is stunned person
        - this is astonished person
        - this is disgust person
        - this is dislike person
        - this is delighted person
        - this is joyful person
sentence 1:
        - A photo of a frightened face, a type of expression
        - A photo of a fear face, a type of expression
        - A photo of a angered face, a type of expression
        - A photo of a frown face, a type of expression
        - A photo of a pitiful face, a type of expression
        - A photo of a tragic face, a type of expression
        - A photo of a non emotinal face, a type of expression
        - A photo of a neutral face, a type of expression
        - A photo of a stunned face, a type of expression
        - A photo of a astonished face, a type of expression
        - A photo of a disgust face, a type of expression
        - A photo of a dislike face, a type of expression
        - A photo of a delighted face, a type of expression
        - A photo of a joyful face, a type of expression
gender:
sentence 0:
        - This is a photo of a male person
        - This is a photo of a female person
```

How sentences get encoded inside the model:

```
    def text_encoder(self, descriptions):
        text_tokens = clip.tokenize(descriptions).to(self.device)
        with torch.no_grad():
            text_features = self.model.encode_text(text_tokens).float()
            text_features /= text_features.norm(dim=-1, keepdim=True)
        return text_features
```

Inference stage of the model is where the model checks for cosine similarity between the encoded generated texts and preprocessed image inserted to the model (it happens to all of the given images in the path):

```
def infer(self, image):
    preprocess_images = []
```

Then the image is being encoded and features extracted:

```
preprocess_images.append(self.preprocess(image))
image_input =  torch.tensor(np.stack(preprocess_images)).to(self.device)

with torch.no_grad():
    image_features = self.model.encode_image(image_input).float()

image_features /= image_features.norm(dim=-1, keepdim=True)
```

cosine similarity - image features and text features gives the result:

```
final_result = {}
for cls in self.text_encoded.keys():
    self.text_encoded[cls]['encoded']['results'] = {}

    for i, text_features in self.text_encoded[cls]['encoded']['sentences'].items():

        text_probs = (100.0 * image_features @ text_features.T)

        result = {}
        for t, v in enumerate(text_probs.cpu().numpy()[0]):
            decoded_label = self.text_encoded[cls]['labels'][t]
            for label, labels in self.text_encoded[cls]['label2labels'].items():
                if decoded_label in labels:
                    try:
                        result[label] += v
                    except:
                        result[label] = v
        sorted_result = sorted(result, key=result.get, reverse=True)
        indexes = sorted_result[:self.text_encoded[cls]['encoded']['topk']]
```

Then the model sort the results with decision making and finds top 3 results and return them:

```
        if cls == 'emotions':
            final_result[cls] = self.decision_making(self.text_encoded[cls]['encoded']['results'])
        else:
            final_result[cls] = self.text_encoded[cls]['encoded']['results']
    return final_result
```

Main part of our algorithm is the decision_making function where we get the top scores and return it to the model to give final result of the test:

```
def decision_making(self, outputs):

    tmp_result = {0:[], 1:[], 2:[]}
    for i in range(len(outputs)):
        first, second, third = list(outputs[i].keys())[:3]
        tmp_result[0].append(first)
        tmp_result[1].append(second)
        tmp_result[2].append(third)

    final_result = {}

    for i in range(3):
        counter_labels = Counter(tmp_result[i])
        key = next(iter(counter_labels)) # extract the largest value
        apperences = counter_labels[key] # get the number of apperences
        apperences = apperences/sum(counter_labels.values()) #
        final_result[i] = (key, apperences)

    return final_result
```

Evaluation

DataSet

The FER2013 dataset is a standard benchmark for the FER task and the largest available dataset for the Facial Expression Recognition task. It contains 35,000 facial expression images for seven emotion classes, including anger, disgust, fear, happy, sad, surprise, and neutral. All face images are grayscale and aligned to a 48x48 pixel grid. Attached EDA notebook (ipynb) for further analysis of him. while we examined the training set, we observed a number of things: The training set images featured a variety of people at different angles and positions. Some individuals wore eyeglasses and scarves. This variety helps to test the model on various options and ensures that his predictions lead to reliable results.

Metrics

The objective of this project was to evaluate the quality of a classifier by determining which combination of sentences and labels would get us higher accuracy. The split to train and test sets are not necessary anymore because we didn't train any model, so we combined the training and test set and measured the quality of the model by using two criteria: accuracy and F1.

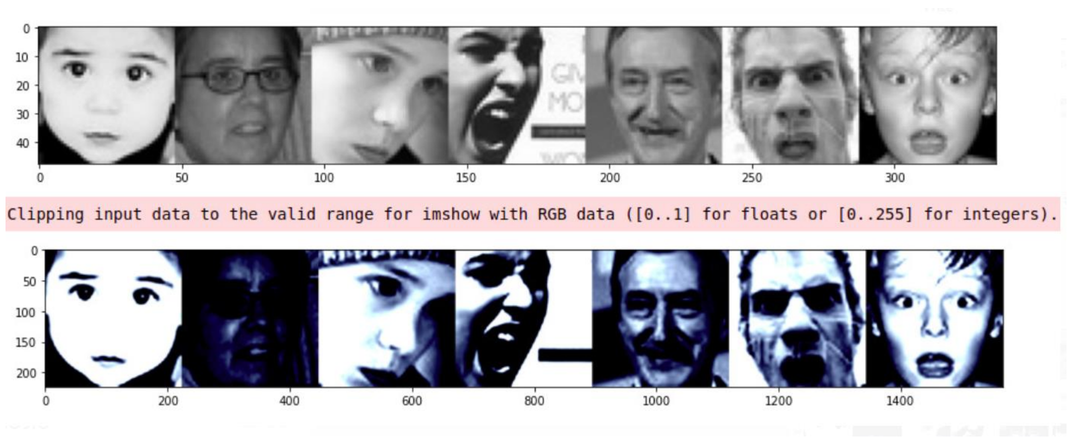
Accuracy is precisely how often the model matched its predictions to the correct labels. It does not account for false positives—cases in which the model incorrectly predicted a label—or false negatives, or cases in which the model failed to predict a present label.

F1 is the weighted average of Precision and Recall, which takes both false positives and false negatives into account. Therefore, F1 is a more accurate measure that can be used to understand the algorithm's results.

In our project, we compute the accuracy of each class (from the main types), which helps us choose a model that will be more accurate at predicting right in the specific classes. But we notice that if we describe a particular class with many words, words that are not in its prototype (like fear with nervus, scary, and fright) will degrade the success of the rest of the classes. To determine the success of the models, we computed F1 scores for the whole set of predictions. However, even with these performance measures, F1 scores fail to provide much insight into how the models were doing. Thus, we visualize these results using a confusion matrix. A confusion matrix lets us see in detail which classes we are successful in and how the model confounds between classes.

Results & insights

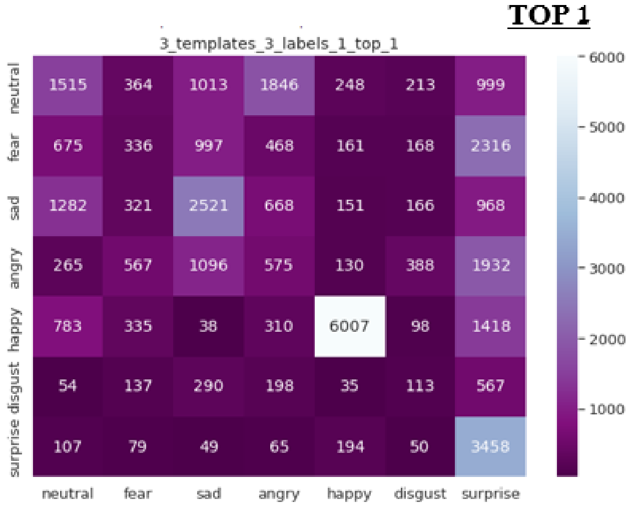
As we aimed to find the best combination of sentence templates and FER labels for our CLIP model, we have collected five-sentence templates. Instead of using the original seven emotions, we extended each category that will contain five synonyms. We randomly selected one template with one label (for every one of the seven categories) and followed this methodology, two templates with two labels and three with three. We are familiar with the model's limitations. Thus, we had to adjust our approach to account for those restrictions. For example, one limitation is its over-reliance on the proper structure. Therefore, we randomly generated three options for each label-template combination to reduce the likelihood of misinterpretations. In total, we generated nine input dictionaries for comparison. Another limitation of the CLIP model is that it was trained on large, color images from the internet. Researchers resize them to a smaller shape (224 X 224) and normalize their pixels using ImageNet mean and standard. However, our dataset differed from the images that CLIP had been trained to recognize. FER2013's images were small, black and white, and came from faces. After we applied those preprocessing steps, we observed the inferior results shown below. It is necessary to mention, with this view, we couldn't resemble the resize effect on the images, and they were stretched and blurry.



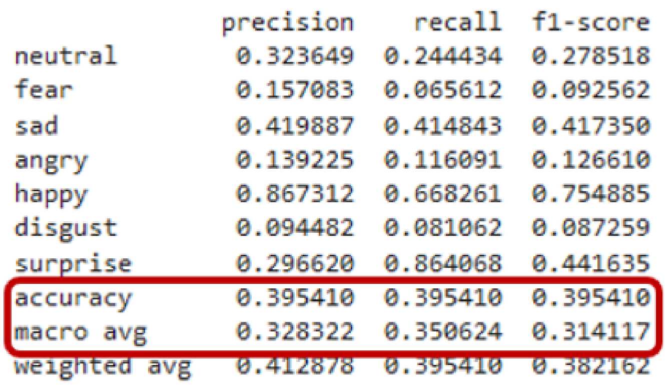
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

The problem occurs when we rely on images that many crucial facial details in them are destroyed by preprocessing. We tested this method, and the performance was terrible. We then checked whether the model achieved a high accuracy rate within the top three results and found that the accuracy was indeed high. This indicates that the model can be used to identify emotions in images, but the images themselves are problematic. We compare the result of top1/top3 of the best model we generated:

TOP 1

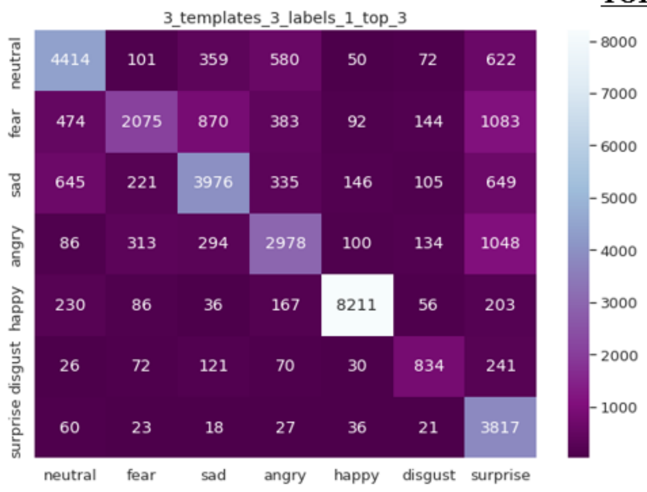


	neutral	fear	sad	angry	happy	disgust	surprise
neutral	1515	364	1013	1846	248	213	999
fear	675	336	997	468	161	168	2316
sad	1282	321	2521	668	151	166	968
angry	265	567	1096	575	130	388	1932
happy	783	335	38	310	6007	98	1418
disgust	54	137	290	198	35	113	567
surprise	107	79	49	65	194	50	3458

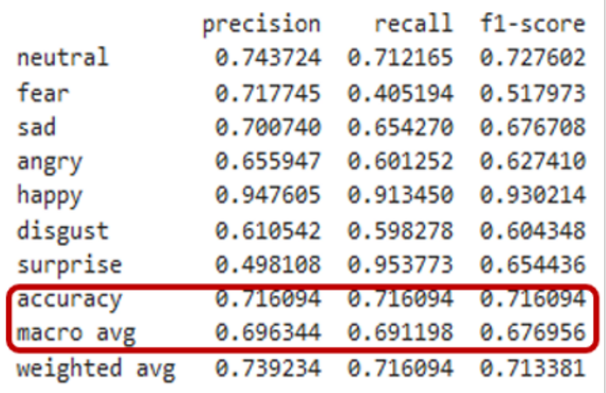


```
              precision    recall  f1-score
neutral        0.323649  0.244434  0.278518
fear           0.157083  0.065612  0.092562
sad            0.419887  0.414843  0.417350
angry          0.139225  0.116091  0.126610
happy          0.867312  0.668261  0.754885
disgust        0.094482  0.081062  0.087259
surprise       0.296620  0.864068  0.441635
accuracy       0.395410  0.395410  0.395410
macro avg      0.328322  0.350624  0.314117
weighted avg   0.412878  0.395410  0.382162
```

TOP 3

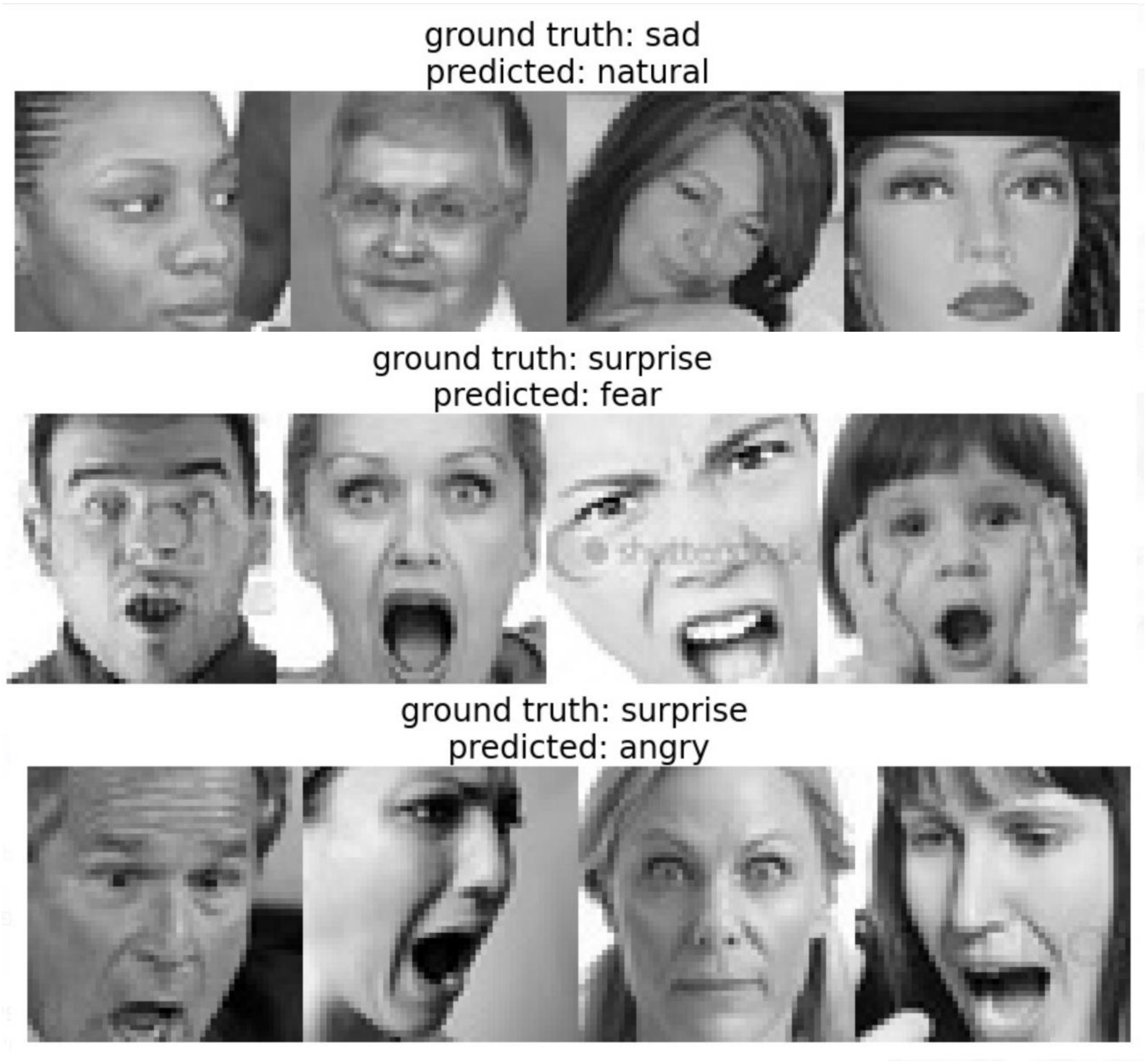


	neutral	fear	sad	angry	happy	disgust	surprise
neutral	4414	101	359	580	50	72	622
fear	474	2075	870	383	92	144	1083
sad	645	221	3976	335	146	105	649
angry	86	313	294	2978	100	134	1048
happy	230	86	36	167	8211	56	203
disgust	26	72	121	70	30	834	241
surprise	60	23	18	27	36	21	3817



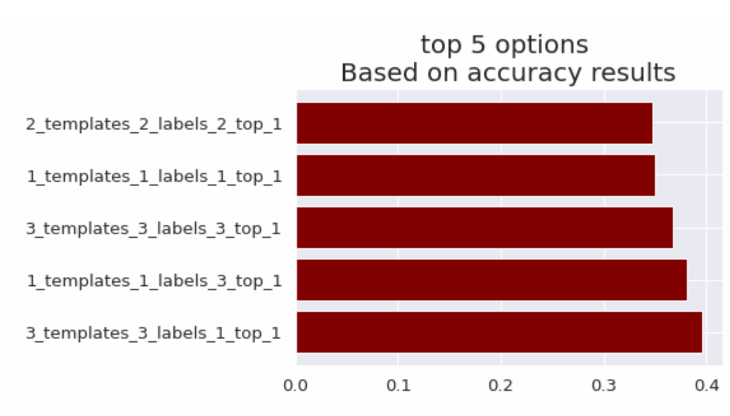
```
              precision    recall  f1-score
neutral        0.743724  0.712165  0.727602
fear           0.717745  0.405194  0.517973
sad            0.700740  0.654270  0.676708
angry          0.655947  0.601252  0.627410
happy          0.947605  0.913450  0.930214
disgust        0.610542  0.598278  0.604348
surprise       0.498108  0.953773  0.654436
accuracy       0.716094  0.716094  0.716094
macro avg      0.696344  0.691198  0.676956
weighted avg   0.739234  0.716094  0.713381
```

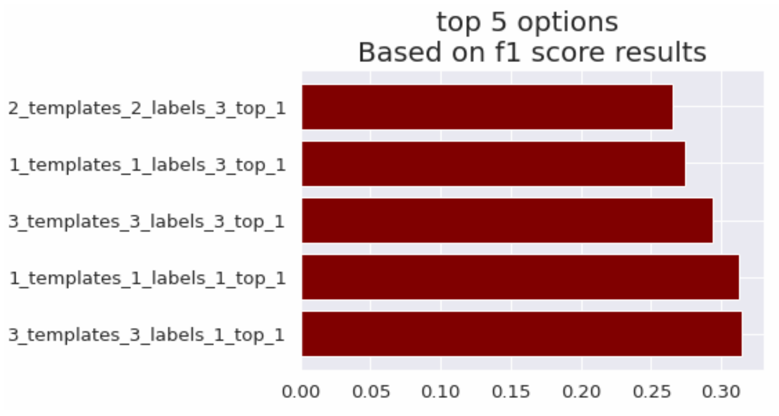
Yet there is some confusion between some emotion categories, such as surprise-fear, surprise-anger, and sadness-fear. To better understand what happened there, we extracted some images as shown below:

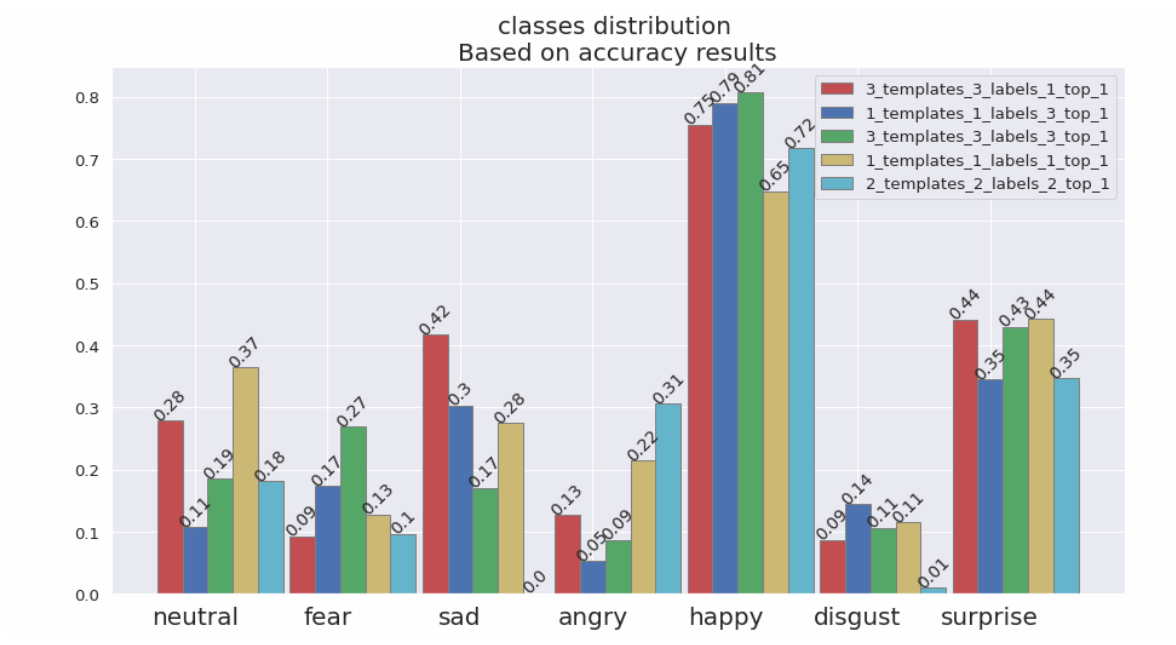


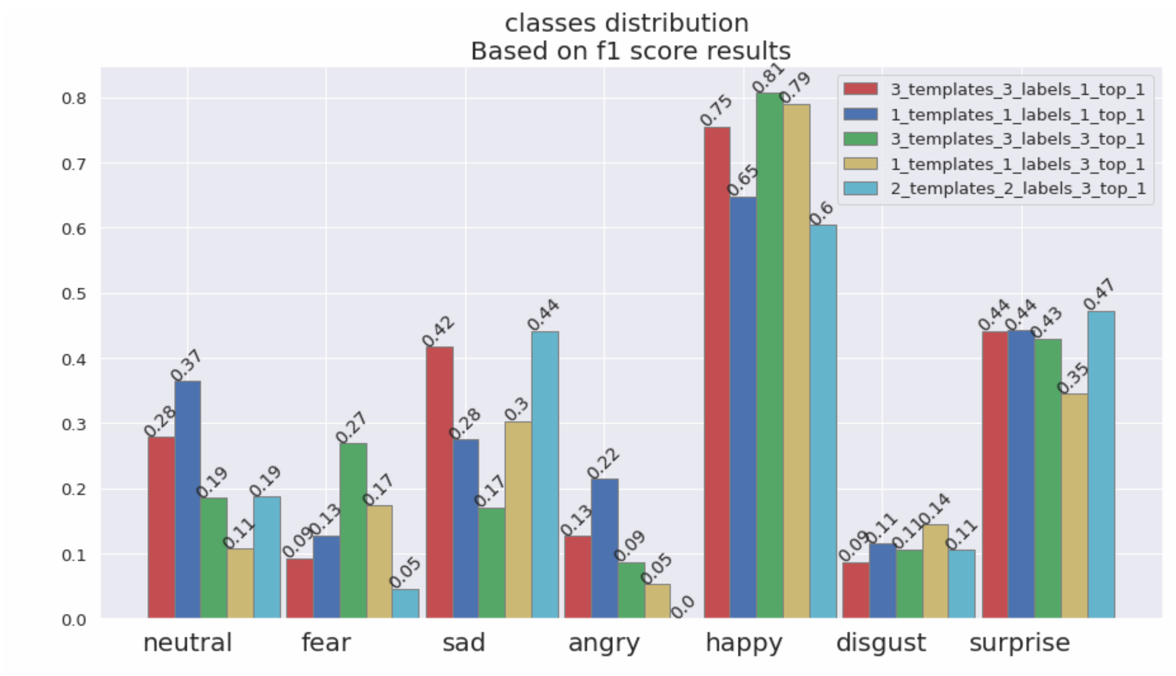
That result indicates some problems with the tagging of the FER2013 dataset. Some labeled images won't look like the label they attached them to. After understanding one model's results and determining why it was behaving unexpectedly, we tested the rest of our models. Finally, we filtered the better models by comparing their accuracy and F1 score to one another on the entire dataset and plotting the results by label.

top 1 results

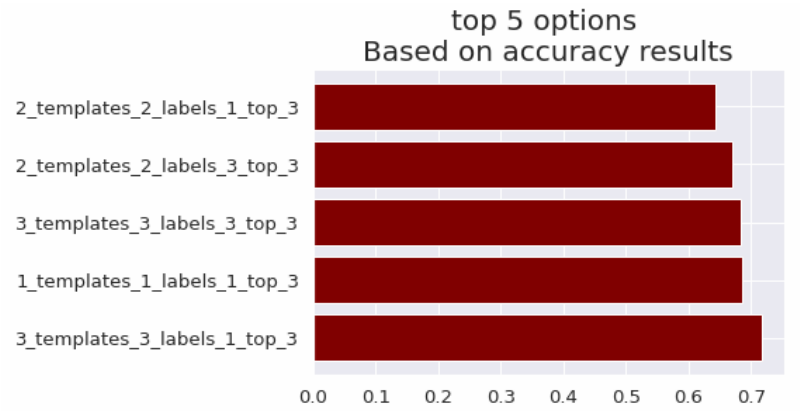


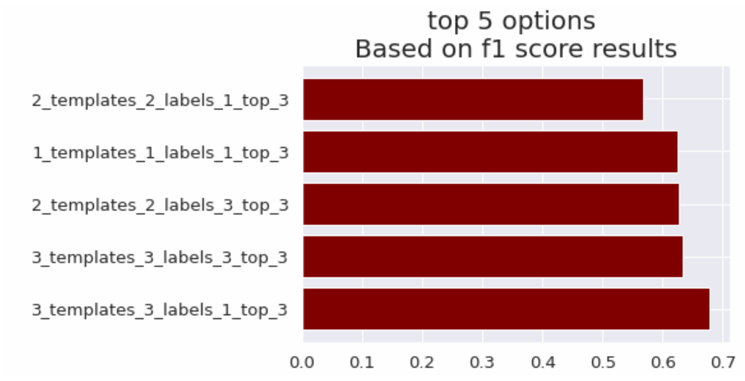


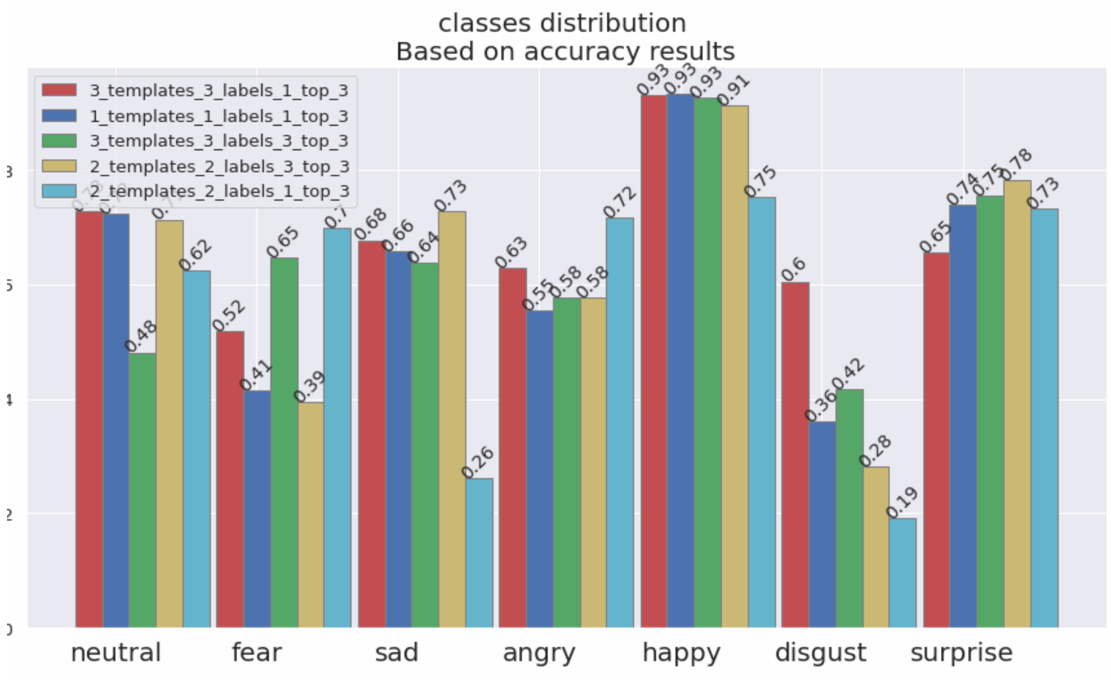


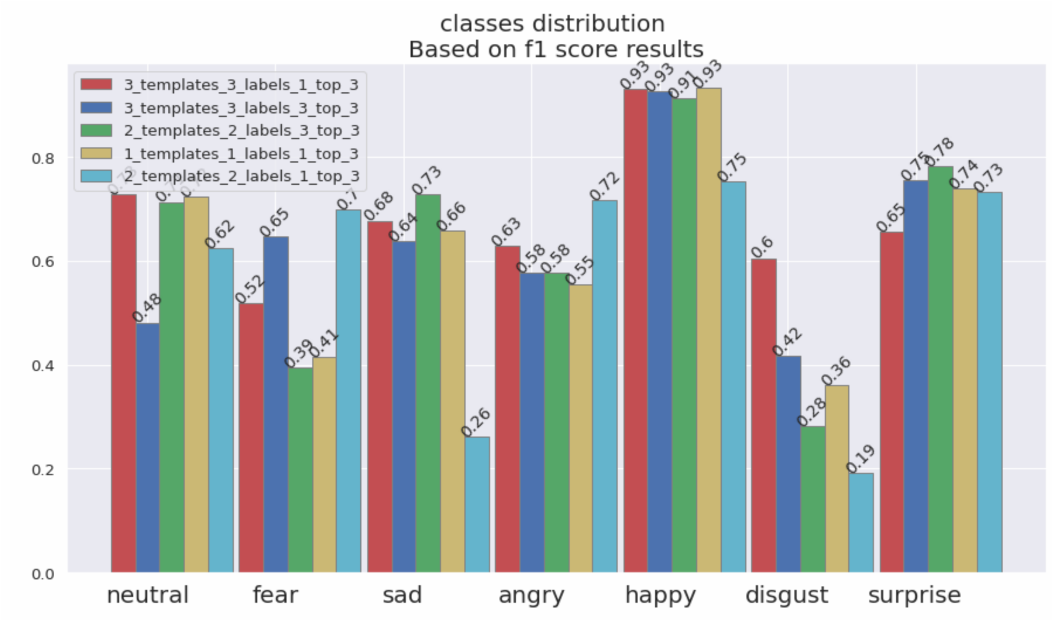


top 3 results

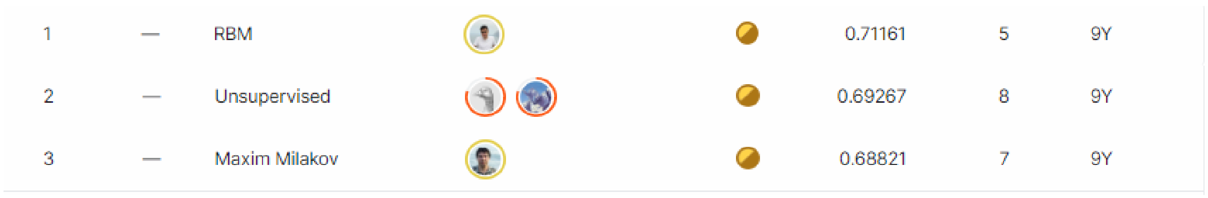






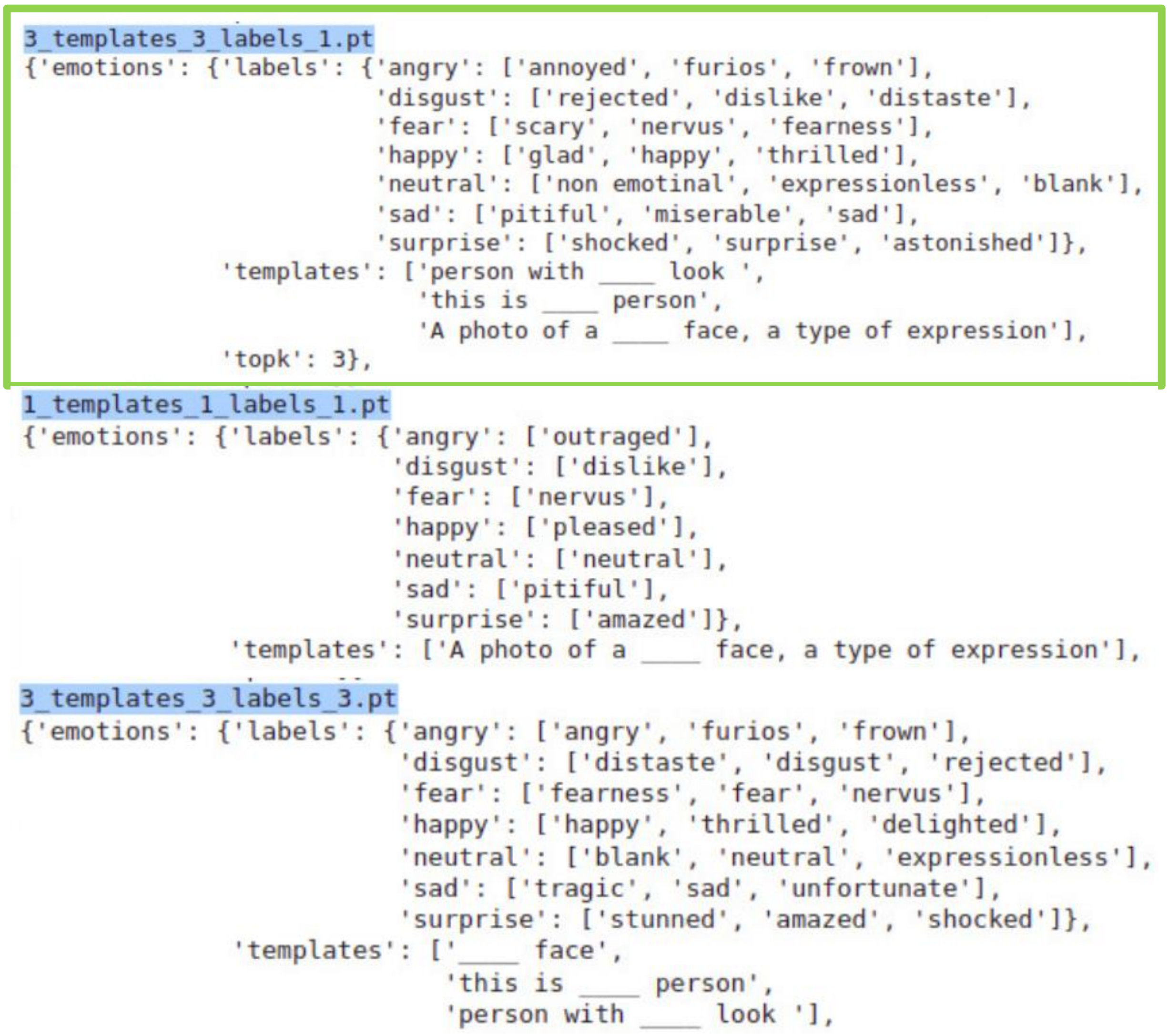


The results indicate that some models are ranked highly by accuracy and F1 score for the top 1 and 3 tables. Furthermore, the best-performing model is rated as best by all given metrics. Even if the top 3 is not a valid option for submitting in the Kaggle competition "Challenges in Representation Learning: Facial Expression Recognition Challenge" which initially introduced the FER2013 dataset, our model achieves an accuracy score of 0.716. You can see the below table of rank and score of competitors, in which our model is listed in first place.



1	—	RBM		●	0.71161	5	9Y
2	—	Unsupervised		●	0.69267	8	9Y
3	—	Maxim Milakov		●	0.68821	7	9Y

Those dictionaries are inputs to the models that appeared in both the top 1 and top 3 tables. The one with a green frame is the best. By looking at his templates, we see that it is the union of the two other templates. We can't find any other similarities between them.



```
3_templates_3_labels_1.pt
{'emotions': {'labels': {'angry': ['annoyed', 'furios', 'frown'],
                         'disgust': ['rejected', 'dislike', 'distaste'],
                         'fear': ['scary', 'nervus', 'fearness'],
                         'happy': ['glad', 'happy', 'thrilled'],
                         'neutral': ['non emotinal', 'expressionless', 'blank'],
                         'sad': ['pitiful', 'miserable', 'sad'],
                         'surprise': ['shocked', 'surprise', 'astonished']},
             'templates': ['person with ____ look ',
                           'this is ____ person',
                           'A photo of a ____ face, a type of expression'],
             'topk': 3},
1_templates_1_labels_1.pt
{'emotions': {'labels': {'angry': ['outraged'],
                         'disgust': ['dislike'],
                         'fear': ['nervus'],
                         'happy': ['pleased'],
                         'neutral': ['neutral'],
                         'sad': ['pitiful'],
                         'surprise': ['amazed']},
             'templates': ['A photo of a ____ face, a type of expression'],
3_templates_3_labels_3.pt
{'emotions': {'labels': {'angry': ['angry', 'furios', 'frown'],
                         'disgust': ['distaste', 'disgust', 'rejected'],
                         'fear': ['fearness', 'fear', 'nervus'],
                         'happy': ['happy', 'thrilled', 'delighted'],
                         'neutral': ['blank', 'neutral', 'expressionless'],
                         'sad': ['tragic', 'sad', 'unfortunate'],
                         'surprise': ['stunned', 'amazed', 'shocked']},
             'templates': ['____ face',
                           'this is ____ person',
                           'person with ____ look '],
```

Conclusion

In conclusion, we feel that this work is unique and innovative, as it allows for a new method of classifying the emotions depicted in photos of the face. By manipulating the input data structure, we have created a classifier that will match performance on facial emotions compared with unique facial emotions recognition models. There are several possible ways to improve this research. We could use a more extensive, better-organized dataset. Our current dataset is not similar enough to what was used to train the CLIP, which could explain why our results were not good enough for the top 1 results. We needed to collect more color images that are well-tagged to test on. It will enrich the image features extraction and reduces the number of misidentifications mentioned in the results section. Another possible improvement would be adjusting the sentence templates and labels that may be necessary. We see that all the models achieved different results, and the input structure controls them.

We have done a lot of work, and we think it has been effective and satisfying. We have reached the results we were looking for, and we made our assumptions validate.

References

1. W. Sun, H. Zhao, Z. Jin.**A visual attention based ROI detection method for facial expression recognition.**Neurocomputing, 296 (2018), pp. 12-22. Link: https://www.sciencedirect.com/science/article/abs/pii/S0925231218303266?via%3Dihub

2. Debin Meng, Xiaojiang Peng, Kai Wang, and Yu Qiao**. frame attention networks for facial expression recognition in videos**. arXiv preprint arXiv: 1907.00193, 2019. Link : https://arxiv.org/abs/1907.00193

3. P.D.M. Fernandez, F.A.G. Peña, T.I. Ren, et al. **FERAtt: facial expression recognition with attention net.** arXiv preprint arXiv: 1902.03284, 2019.Link: https://arxiv.org/abs/1902.03284

4. K. Wang, X. Peng, J. Yang, D. Meng, Y. Qiao. **Region attention networks for pose and occlusion robust facial expression recognition** IEEE Trans. Image Process., 29 (2020), pp. 4057-4069. Link: https://dl.acm.org/doi/abs/10.1109/TIP.2019.2956143

5. Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever. **Learning Transferable Visual Models From Natural Language Supervision** arXiv: 2103.00020 Link: https://arxiv.org/abs/2103.00020