Anaïs Bellia

Goemotions : emotion classification

Table of content

1) Introduction

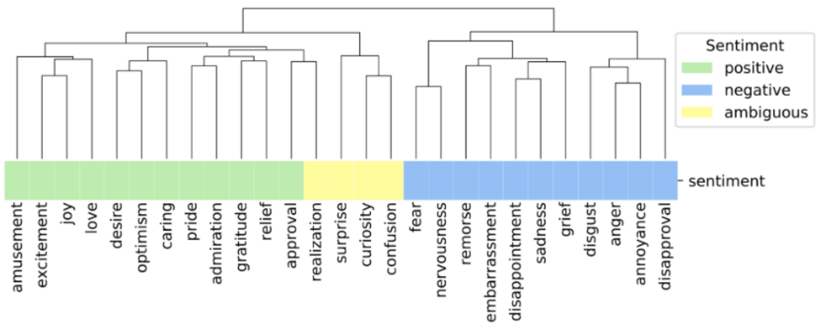
1.1 Dataset description

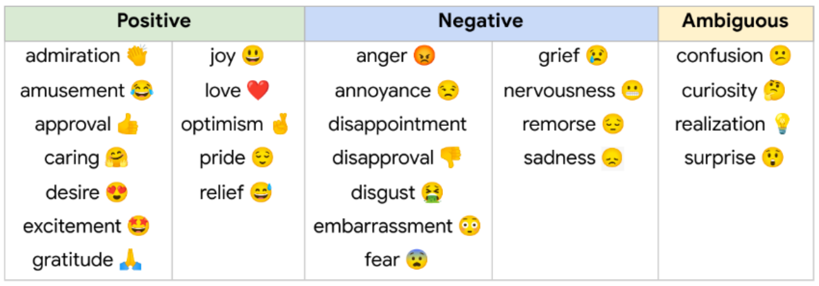
GoEmotions is a corpus of 58k comments extracted from popular English-language subreddit and labeled with 27 emotion categories and neutral.

The dataset was collected by Google and built using Reddit comments from 2005 (the start of Reddit) to January 2019, sourced from subreddits with at least 10k comments, excluding deleted and non-English comments. This dataset is decomposed in 3 csv files and contains:

- Number of examples: 58,009.
- Number of labels: 27 + Neutral.

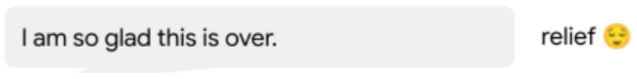
The emotion categories include 12 positive, 11 negative, 4 ambiguous emotion categories and 1 “neutral”.





Positive		Negative		Ambiguous
admiration 👏	joy 😃	anger 😡	grief 😢	confusion 😕
amusement 😂	love ❤️	annoyance 😒	nervousness 😬	curiosity 🤔
approval 👍	optimism 🤞	disappointment	remorse 😔	realization 💡
caring 🤗	pride 😌	disapproval 👎	sadness 😞	surprise 😲
desire 😍	relief 😅	disgust 🤮		
excitement 🤩		embarrassment 😳		
gratitude 🙏		fear 😨		

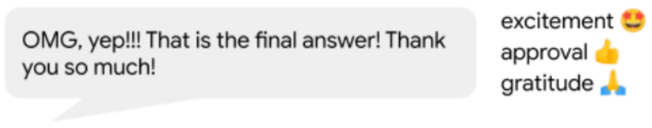
This text expresses relief, a complex emotion conveying both positive and negative sentiment.



I am so glad this is over.

relief 😌

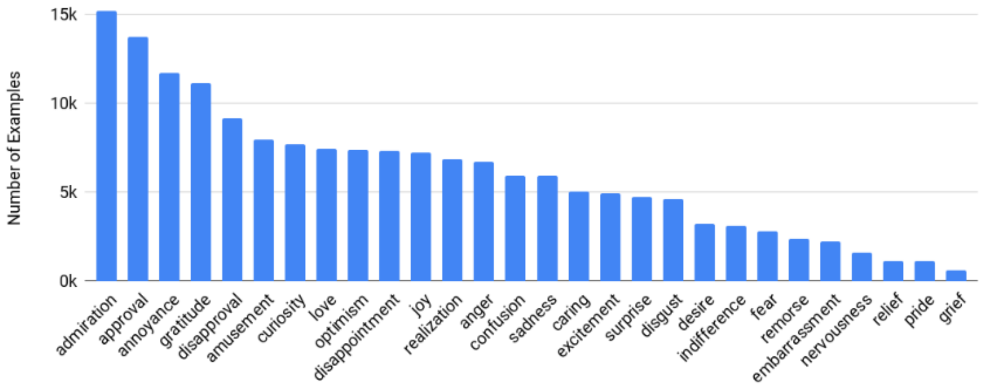
This text expresses several emotions at once, including excitement, approval, and gratitude.



OMG, yep!!! That is the final answer! Thank you so much!

excitement 🤩
approval 👍
gratitude 🙏

Emotions are not distributed uniformly in the GoEmotions dataset. We can see that there is a high frequency of labeled text for “admiration”, “approval”. Therefore, the Dataset is imbalanced.



1.1.b What is the task targeted by the experiments

The goal here is to analyse human sentiment analysis and predict a future text’s emotions. This can be challenging knowing that there are ambiguous emotions.

1.2 Objectives and what achievements goal with the experiment

The objective would be to predict the emotion of a text that was not initially trained.

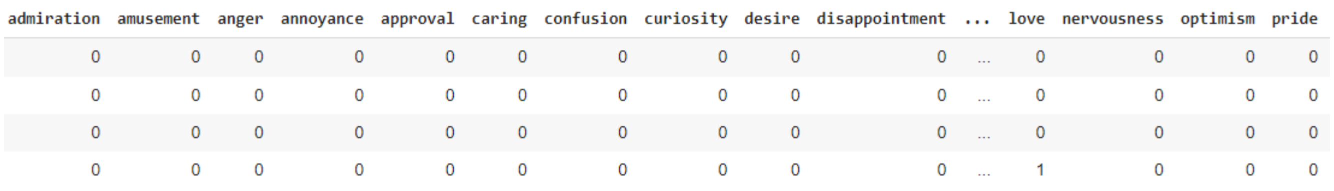
The probability of having the correct emotion label to the associated text is 1/28 = 0.035.

The aim would be to increase this probability much higher with a model that can process the bias from the dataset

2) Proposed pipeline

2.1 Description of the base model used

A one hot encoding is applied on the dataset.



admiration	amusement	anger	annoyance	approval	caring	confusion	curiosity	desire	disappointment	...	love	nervousness	optimism	pride
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
0	0	0	0	0	0	0	0	0	0	...	1	0	0	0

The first model is a sequential model build with :

- An Embedding layer with (vocab_size, embedding_dim, input_length=max_length)
- GlobalAveragePooling1D(),
- Dense(128, activation='relu'),
- Dense(28 , activation='sigmoid')

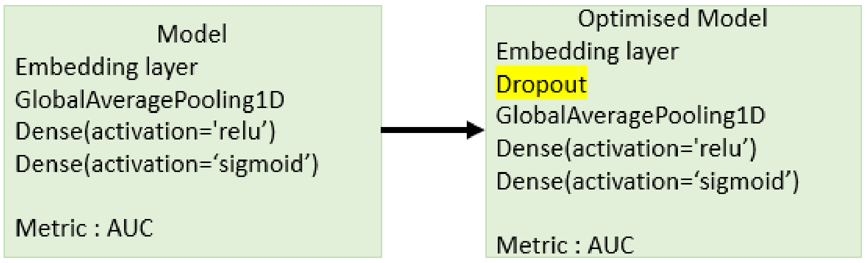
The layers were selected as to be the best choice based on researches on the internet for sentiment analysis.

- It is a one hot encoding label so we pass the prediction as a matrix.
- The loss function : binary classification model. The binary_crossentropy function computes the cross-entropy loss between true labels and predicted labels.

2.1.a Description of baseline model and the more complex model

The model gives good result, although adding regularization techniques would improve the results.

Among the regularization techniques dropout was used to optimize the model.

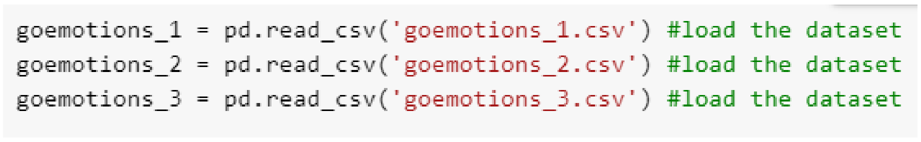


2.2 Describe and justify any data pre-processing you had to do

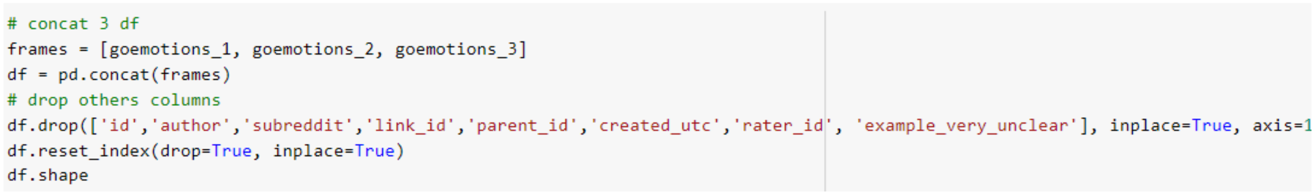
Load and transform the data

We concatenate the 3 dataframes so we can deal with a single one. We drop all the useless columns in the dataframe and only keep the sentiment and the tweet associated to it.

We also reset the index because we concatenated the 3 goemotions so without that we would have 3 rows with the same index



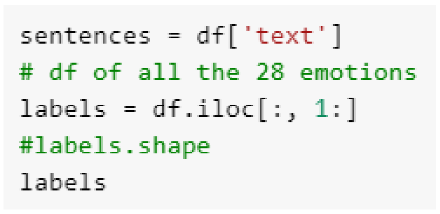
```
goemotions_1 = pd.read_csv('goemotions_1.csv') #load the dataset
goemotions_2 = pd.read_csv('goemotions_2.csv') #load the dataset
goemotions_3 = pd.read_csv('goemotions_3.csv') #load the dataset
```



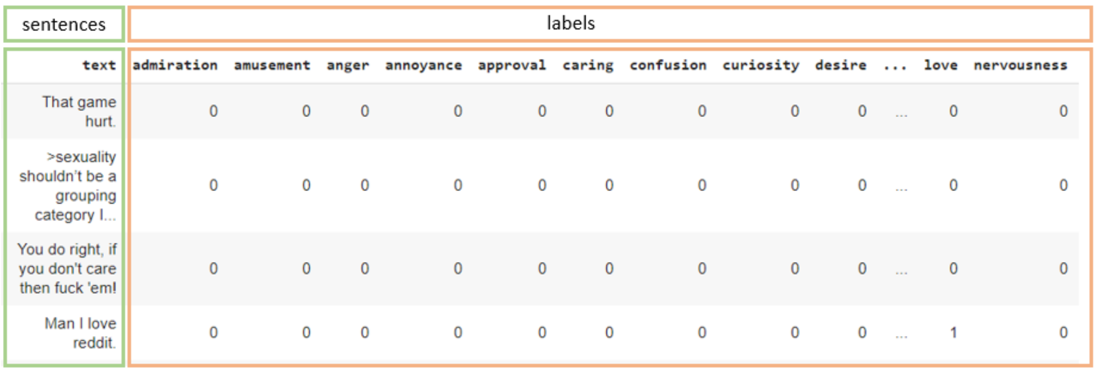
```
# concat 3 df
frames = [goemotions_1, goemotions_2, goemotions_3]
df = pd.concat(frames)
# drop others columns
df.drop(['id','author','subreddit','link_id','parent_id','created_utc','rater_id', 'example_very_unclear'], inplace=True, axis=1
df.reset_index(drop=True, inplace=True)
df.shape
```

We assign the following variables :

- sentences : the column "text" of df that refers to all the tweets in the dataframe
- labels : sentiment get dummies of the dataframe, all the sentiment columns so from the index 1 (index 0 being the text) to the last column) There are 28 sentiments in the dataframe



```
sentences = df['text']
# df of all the 28 emotions
labels = df.iloc[:, 1:]
#labels.shape
labels
```



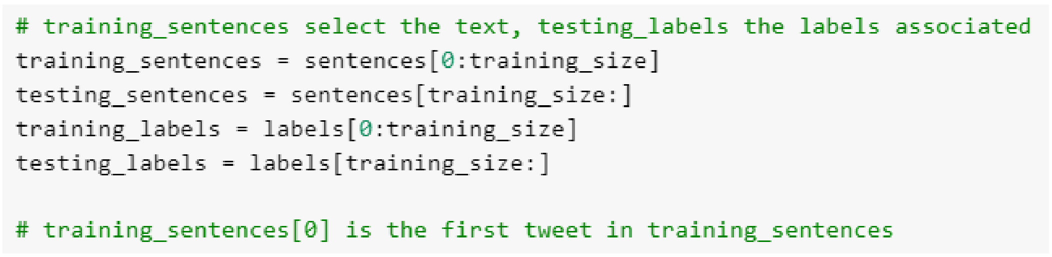
sentences	labels											
text	admiration	amusement	anger	annoyance	approval	caring	confusion	curiosity	desire	...	love	nervousness
That game hurt.	0	0	0	0	0	0	0	0	0	...	0	0
>sexuality shouldn't be a grouping category I...	0	0	0	0	0	0	0	0	0	...	0	0
You do right, if you don't care then fuck 'em!	0	0	0	0	0	0	0	0	0	...	0	0
Man I love reddit.	0	0	0	0	0	0	0	0	0	...	1	0

Create train and test

We associate to training_sentences the tweets for the training size so from 0 to 200 000

The testing_sentences is all the tweet from 200 000 to the end of the dataframe

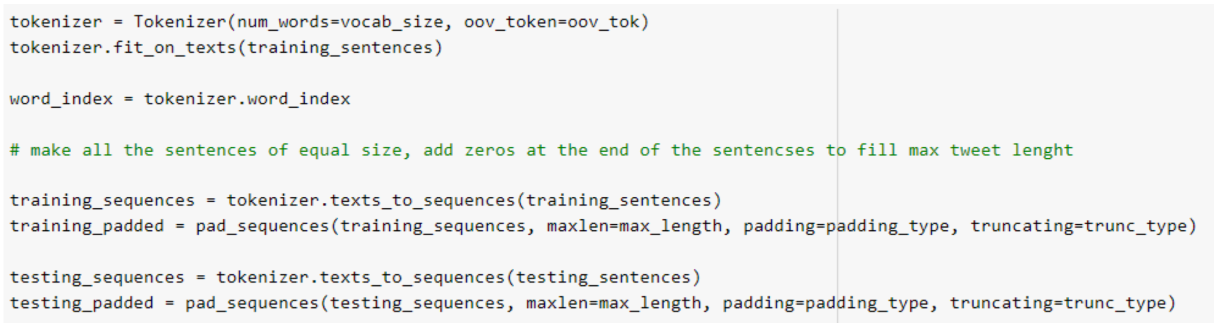
The training_labels takes all the labels associated to the tweets for the training size (0 to 200 0000) The testing_labels takes the ramaining labels



```
# training_sentences select the text, testing_labels the labels associated
training_sentences = sentences[0:training_size]
testing_sentences = sentences[training_size:]
training_labels = labels[0:training_size]
testing_labels = labels[training_size:]

# training_sentences[0] is the first tweet in training_sentences
```

We tokenize training_sentences and testing_sentences



```
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)

word_index = tokenizer.word_index

# make all the sentences of equal size, add zeros at the end of the sentencses to fill max tweet lenght

training_sequences = tokenizer.texts_to_sequences(training_sentences)
training_padded = pad_sequences(training_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)

testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)
```

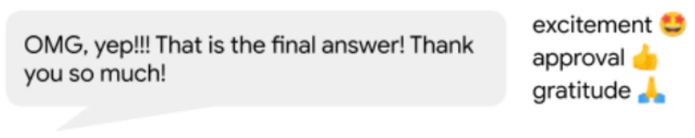
3) Experimental methodology

3.1 Describe and justify the methodology used to test your pipeline

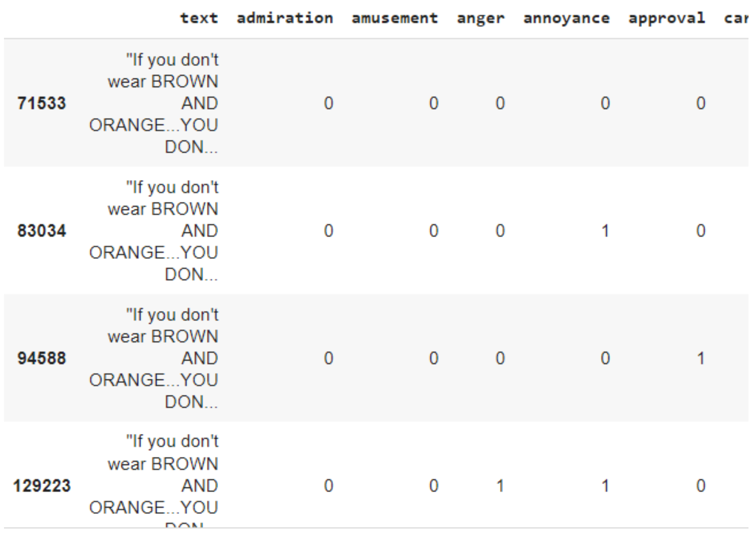
3.1.a Metrics : Area Under Curve (AUC)

The chosen metric for evaluation was the Area Under Curve (AUC). AUC is used for binary classification problem and deals with situations where there is a very skewed sample distribution, and to avoid overfitting to a single class.

In the dataset, there can be a same text with different labels:


OMG, yep!!! That is the final answer! Thank you so much!

excitement 😍
approval 👍
gratitude 🙏



	text	admiration	amusement	anger	annoyance	approval	car
71533	"If you don't wear BROWN AND ORANGE...YOU DON...	0	0	0	0	0	
83034	"If you don't wear BROWN AND ORANGE...YOU DON...	0	0	0	1	0	
94588	"If you don't wear BROWN AND ORANGE...YOU DON...	0	0	0	0	1	
129223	"If you don't wear BROWN AND ORANGE...YOU	0	0	1	1	0	

Therefore, AUC compares True Positive Rate and False Positive Rate. The metric accuracy was used at first but is not a good metric since it only computes the ratio of the number of true predictions over the total labels in the dataset.

3.1.b Split validation method : train test split

The dataset was split into the train and the test set by the training size declared to 20000

```
# training_sentences select the text, testing_labels the labels associated
training_sentences = sentences[0:training_size]
testing_sentences = sentences[training_size:]
training_labels = labels[0:training_size]
testing_labels = labels[training_size:]

# training_sentences[0] is the first tweet in training_sentences
```

3.1.c Hyper-parameter choice criteria

Hyper-parameter set to the framework default

- embedding_dim : the argument for output_dim in the Embedding layer
- trunc_type='post'
- padding_type='post'
- oov_tok = "<OOV>"
- oov_token=oov_tok
- verbose was set to 2

Hyper-parameter tunned

- training_size

The total dataset had 211225 rows and was split into the training dataframe of size 200000 and the testing dataframe with the remaining 11225, so approximately a split of 0.8, 0.2.

- vocab_size : the argument corresponding to input_dim in the Embedding layer

The vocab size is the number of unique words and was set to be bigger than the vocabulary size.

- max_length : This is the length of input sequences, defined for any input. The maximum length of an input text was 33028 but, max_length was set to 100000.
- num_words was set to the vocabulary size

num_words specify the maximum number of vocabulary words to use. For example, if we set num_words=100 when initializing the Tokenizer, it will only use the 100 most frequent words in the vocabulary and filter out the remaining vocabulary words. This can be useful when the text corpus is large and it is needed to limit the vocabulary size to increase training speed or prevent overfitting on infrequent words.

3.2 GitHub repo

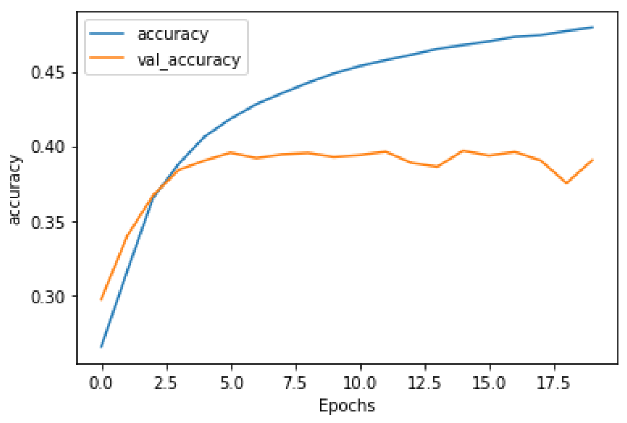
4) Results

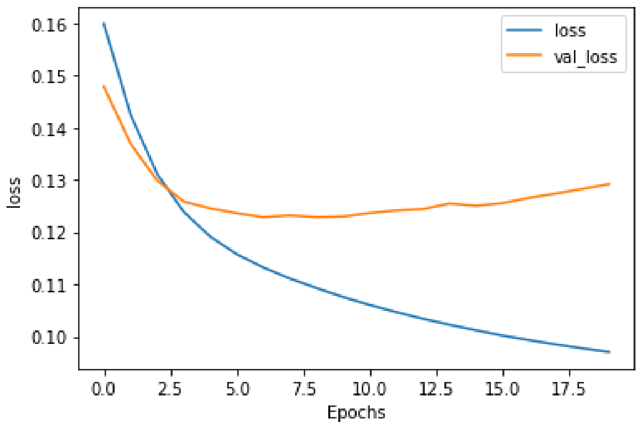
Results of your experiments:

The first experiment is presented to show the difference of performance a model can have when the metric is not appropriate. In this case, the accuracy stops to increase and even start to decrease a little bit. The validation loss increases, learning too much noise leading to overfitting.

Model
Embedding layer
GlobalAveragePooling1D
Dense(activation='relu')
Dense(activation='sigmoid')

Metric : accuracy

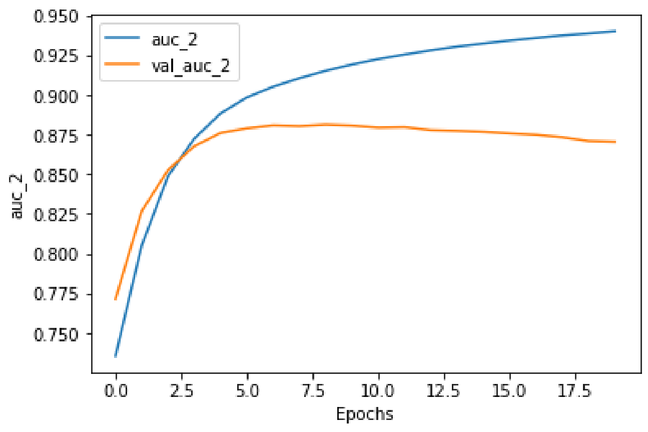


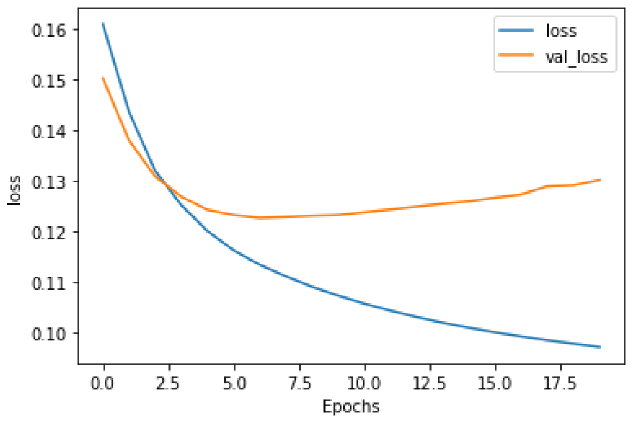


The second case shows the performance with AUC metric, but the model also shows overfitting.

Model
Embedding layer
GlobalAveragePooling1D
Dense(activation='relu')
Dense(activation='sigmoid')

Metric : AUC

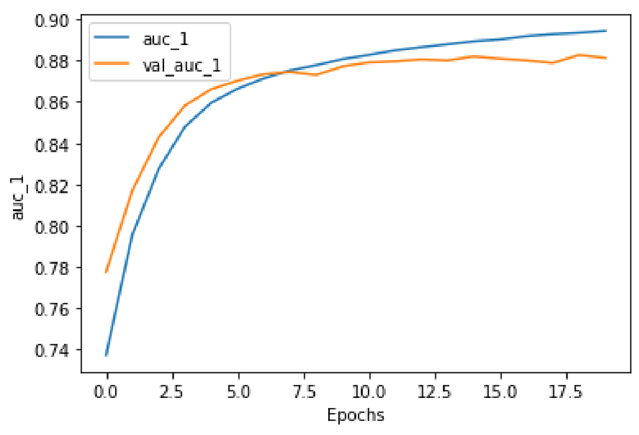


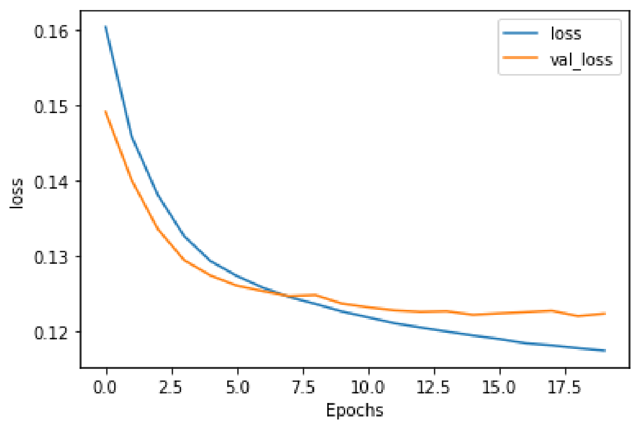


In the optimized model, the dropout regularization prevents overfitting. The training process is working well, with a minimum error and good prediction

Optimised Model
Embedding layer
Dropout
GlobalAveragePooling1D
Dense(activation='relu')
Dense(activation='sigmoid')

Metric : AUC





Epoch = 20	Model	Model	Optimized model
Metric	Accuracy : 0.3943	AUC : 0.8704	AUC : 0.8813
Loss	0.1063	0.1302	0.1223

4.2 Reasoning behind changes to improve performances

At first, the accuracy metric was used, and the results were stagnant with a maximum accuracy reaching 0.39. The results could be considered good because the model can predict the right emotion 40% accuracy (instead of 1/28 = 3%).

The AUC metric was then the right metric to choose for the experiments, allowing the model to reach 88% accuracy but with overfitting.

Finally, a dropout regularization was added to prevents overfitting.

4.3 Comparison of the results between baseline and proposed models

By comparing the results of the models, the last one with a dropout regularization seems to give the best results.

4.4 Main difficulties encountered in running the experiments and/or to improve the models

Finding the right metric with the context of the experiment was a little challenge. With the accuracy metric, it was not increasing and going further than 0.39, having a phenomenon of “plateau”. Different loss were tested such as sparse categorical cross entropy by transforming the one hot encoding dataset to a label encoding, also both categorical cross entropy and binary cross entropy work for binary classification so they had to be evaluated

5) Conclusion and next steps

5.2 Extra experiments or procedures to perform

Other regularization techniques could be tested to see if there are better performances such as L2-norm Regularization, Weight Decay and L1-norm.

Different cross-validation techniques can be tested. The best splitting method must represent the right portion of each emotion and the data must be well distributed in the train and the test set.

6) References

https://androidkt.com/choose-cross-entropy-loss-function-in-keras/#:~:text=Different%20between%20binary%20cross%2Dentropy,(one%2Dhot%20encoding).

https://www.kdnuggets.com/2021/06/beginners-guide-debugging-tensorflow-models.html

https://vitalflux.com/keras-categorical-cross-entropy-loss-function/