Emotion Detection with Vision Transformers and Image Features

Stephan Sharkov
Stanford University,
Department of Computer Science
stpshrkv@stanford.edu

Abstract

With computer vision on the rise and new methods such as transformers and finetuning targeting many vision problems, it is important to investigate ones impacting our everyday life and experience of interacting with computers. In this paper, I attempt to improve on problem of emotion detection by using transformers and finetuning on pre-trained models. I also attempt to use image features, particularly HOG, to investigate whether they improve performance for any of the methods. My best model achieves 58.51% accuracy on emotion classification task, using Transformer and original images, improving on previously implemented models. I further discuss qualitative performance of the transformer and how it could compare to humans. I also achieved 71.45% accuracy on sentiment analysis problem, with usage of finetuning and HOG features, showing potential for image features and their importance in computer vision.

1. Introduction

Rising amount of images in the web allows to analyze pictures people post on popular social media websites such as Instagram, Facebook, or Snapchat. While people tag some of those pictures or indicate certain parameters about it, one thing they can't tag is emotion or state of mind they have when making a post. Investigating emotion detection, field of recognising emotion based on an image, has been an ongoing problem, and is currently being improved with more methods arising in Computer Vision. This area of research could be useful in robotics, healthcare, virtual assistance, and many other fields, where machines need to comprehend human emotion and be able to act upon it.

In this project, I am attempting to improve existing methods of emotion detection using Vision Transformers and Finetuning, and investigating whether usage of image features like HOG can improve that performance.

There will be two problems I will be trying to analyse during the project. In the first problem models accept training data of a grayscale image represented in numbers of pixel values or HOG features of the image, and a number indicating emotion in the picture. The final model is supposed to accept a new grayscale image with human face or images' HOG features and be able to identify the emotion by returning back the number indicating the emotion. The emotion categories targeted in this problem are: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral.

The second problem is essentially very similar, except with the sole difference that I am going to perform sentiment analysis of the image. So rather than predicting 1 of the 7 emotions in the image, I am going to perform sentiment analysis and classify data for Negative (Anger, Disgust, Fear, Sad) vs. Neutral-Positive emotion (Happy, Surprise, Neutral).

2. Related Work

A lot of research has been done in the field, starting back from 1972[1] with first non-AI methods of face and emotion detection, but I would like to pay attention to a few important papers relating to this project. Early attempts used methods before CNN and Transformers like ones by Gajarla and Gupta [2], where authors attempted to finetune on existing CNNs for emotion detection and sentiment analysis (positive vs. negative emotion). First of all they collected their own dataset using Flickr API, an approach that seems to be popular in computer vision in other domains as well, including predicting cities or urban setting based on images[3]. Love and Violence were included to represent recent trends of protests and civil action taking place in social media. The authors took on different methods, where most of them included fine-tuning on the existing models of object and scene detection. Authors started with training a One vs. All SVM using object classification model of VGG-ImageNet[4] as a pre-trained model. More specifically they passed the images as input through VGG-ImageNet[4], stored the activations from second-to-last fully connected layer as feature vectors, and then trained SVM for each emotion separately. Coming with the assumption that some images (for example from protests) have many people, authors thought it is hard to analyse emotions in those as there are too many objects. Because of that they tried to fine-tune VGG-Place205[5] using a scene-based dataset, getting accuracy improvement compared to previous method. That gave them rise to the last method they used is fine-tuning on ResNet-50, as that network is trained on both object and scene data, combining the two approaches above. Final results were 41% in emotion classification, which was worse than previously done models and 73% on sentiment analysis task, which was better than earlier results. Authors have done an incredible job considering many limiting and specific for emotion classification factors, as well as changing classes of emotion, but using One vs. All SVM was not the best model choice.

Later on, researchers trained their own CNNs for the problem. In paper by Jaiswal, Raju and Deb [6], authors attempted to train one specifically for the emotion detection, using existing datasets FER2013 and JAFFE. Authors used two submodels which share the same input and both have the same kernel size. Then the outputs from feature extraction are flattened into vectors and concatenated before passing into final output layer for classification. Each submodel contains convolutional layer, local contrast normalization layer, maxpool, flatten. Reasoning behind splitting into submodels came from the fact that authors wanted model to look at features of face like eyes, lips, mouth and etc. They compared their results to a similar model of a previous author, and saw improvement to 70% from 67% on FER2013, and 98.5% from 98% on JAFFE. Authors have done a good job introducing an interesting and working approach of splitting CNN into submodels, however usage of not widely available local contrast normalization prevents from replicating or improving on those results.

Similar to approach taken in Gajarla and Gupta[2], researchers like Xu et. al.[7] used transfer learning but with Convolutional Neural Networks as finetuning to perform visual sentiment analysis task. More specifically they pretrained the same CNN's as did Gajarla and Gupta[2], after which they used the sentiment dataset to make final tuning for the task. These authors also came up with more interesting evaluation metric - Area Under the receiver operating characteristic Curve (AUC). With that metric authors improved on previous models achieving 70% accuracy on the 5-scale sentiment rating prediction, with previous performances of 64% and 67%. AUC helps with imbalanced datasets, which could be great for most emotion datasets, as special emotions like disgust or surprise have way less images existing on the web. Because of the wide imbalance and different choices of which emotions to detect, researchers use a variety of metrics including Precision, Recall, and F-1 score like Asghat et. al. [8].

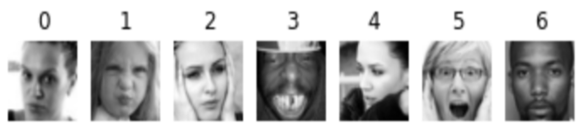
The most recent approaches attempted to use finetuning but with transformers, the method that gave evolution to Natural Language Processing, and now being widely explored in Computer Vision. Ma, Sun, and Li [9] gave the most recent attempt by constructing a transformer with convolution, global and local attention, as well as batch normalization. All of the training of this transformer is done on top of pre-trained model ResNet18, that was trained on MS-Celeb-1M face recognition dataset[10]. Authors improved on all accuracy metrics in the problem they were solving and at the moment their approach represents state of the art method in emotion detection, especially with the fact that their approach generalises better than previous methods. Since their research focused more on generalisation, authors did not explore other pre-trained models, which is a main disadvantage of the approach, however is something I will attempt to improve on in this paper.

3. Data

3.1. Dataset

It took some time to research for a suitable dataset. Many good quality datasets unfortunately are not available to the public, so I ended up going with the FER2013 dataset [11]. That dataset contains 48x48 pixel grayscale images. Representation of it is in a comma-separated value (.csv) file, and dataset is split in two columns, and every row contains two parameters describing an image: first is number representing emotion in the image (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral), and second is a string of 2304(48*48) numbers, where each is a pixel grayscale value of the image.

Figure 1. Examples of data for each emotion



My first job was to convert string to an array of integers and then figure out how to be able to move between number values and create actual images, which was done using PIL[12] libraries of Python. Dataset has a very big disadvantage, low resolution of pictures was not enough to achieve high performance, however, was enough to conduct analysis.

Dataset includes 28709 images for the training set, where each emotion has around 4200 images, except disgust having only 430. Testing data includes 3589 images.

3.2. Train/val/test data

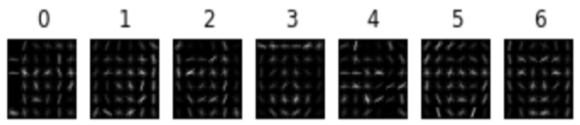
For my experiments I slightly modified the distribution, and split the 28709 training images into 22968(80%) images for training my models, and 5741(20%) for validation set, which models predict on at every epoch.

Moreover, to initially tweak hyper parameters and make sure code works, I worked with smaller subsets of the data. I used 800 training and 200 validation images for training a transformer, and 6400 training and 1600 validation images for finetuning. After tweaking parameters, every model was ran on full dataset. At every point of taking subsets of data, I made sure that every emotion was present in those subsets in proportion similar to those in full dataset.

3.3. Image Features

As one of potential ways to improve on current models, I was considering using different image features as data inputs, which could go separately or be concatenated into a vector together. The feature I focused on is Histogram of Oriented Gradients(HOG) computed using scikit-image[13]. HOG represents distribution of direction of gradients, as it calculates magnitude and direction of gradient at every pixel of the image and stores it as a feature vector of the image. This highlights the edges and corners, as the biggest gradient change happens there. Knowing that distribution, allows us to understand the shape of the face and emotion, as positions of every face part are also recorded, which helps to see unique parts of an emotion.

Figure 2. Examples of HOG data for each emotion



Due to the small resolution of the dataset and the fact that images are cropped just to cover the face, HOG does not demonstrate much difference among emotions. However, looking closely we can see differences in mouth and eye shapes, as even in real life those distinguish emotions the best.

4. Methods

4.1. Baselines

4.1.1 KNN

KNN makes an assumption that similar things are close to each other. It is an algorithm which predicts information about an image based on k number of images closest to it. More specifically it chooses the label that appears the most among those k neighbors, where number k itself can be chosen and adjusted to achieve better accuracy. The decision of how close images are is based on a feature parameter. My KNN is constructed using Numpy[14] and Scikit-learn[15]. Initially I am using pixel values as a feature parameter to compare images, but using HOG representations makes more sense, as we are comparing similarity of shapes of faces and face parts, thus looking more specifically for main characteristics of emotions.

4.1.2 CNN

CNN's have been the most popular approach in vision for a long time, and many different networks were created to achieve the best performance. I implemented a simple CNN using Numpy[14] and PyTorch[16] inspired by lectures, assignment 2[17], and papers in section 3. This CNN starts with batch-normalization, to pre-process and normalize the data. This appeared to be the most important stage that I settled on after trying alternative things such as Local Response Normalization, as well as making dataset more balanced. Without batch-normalization CNN could not learn anything and was just predicting the class that is most frequent in the training set. After that, I applied convolutional Layer with 32 filters, ReLU activation, another convolutional layer with 16 filters, ReLU, and MaxPool.

4.2. Main Methods

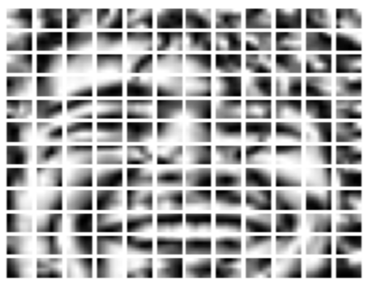
4.2.1 Training a Transformer

I implemented a Transformer using Tenserflow[18] and Keras[19] inspired by class material and guides from Keras[19] library.

Transformer starts with creating different data augmentations for each of the images, including Resizing to 72x72 pixel image, RandomRotation and RandomZoom. All of those help to add more data and assist transformer in recognising different rotations and zooming of each of the faces in the dataset, since everyone's head can be positioned differently in the picture.

After that each image is being split into 144 patches, and those patches are being encoded with position embedding, so that the system knows the relative position of patches. We can see an example below:

Figure 3. Patch splitting



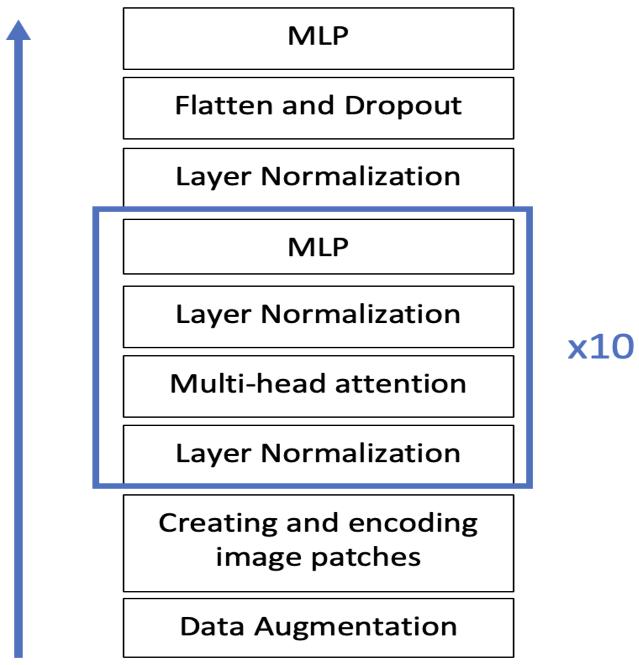
This is being done so that transformer has pieces of images to work with since it requires sequences of data for it to function. Transformer learns how those pieces relate to each other, and how those relations indicate the emotion. It helps the system to learn more concretely about mouth shape, eye shape and other face features indicating emotion.

Next most important step is the main part of the transformer is the transformer block of layers. The number of those blocks could vary, but in my model I set on 10, which performed the best. Each implemented block consists of Layer Normalization, followed by Multi-Head Attention, another Layer Normalization and MLP layers. MLP layer consisted of 2 layers of Dense layer with ReLU activation and Dropout with probability 0.5 and 0.7 for both problems respectively. Dense layer is essentially a Keras[19] method of combining a connected layer with activation function.

The transformer finishes and gives final results after another Layer Normalization, Flatten layer, another Dropout and last MLP with the same structure.

The structure of the transformer is drawn out below:

Figure 4. Structure of the transformer



4.2.2 Finetuning on pre-trained models

Besides training a transformer for the emotion detection, I have considered to explore approaches of transfer learning, similar to the ones done with CNN, using Keras[19] and Pandas[21]. For the start I had to choose a pre-trained model from which to begin finetuning. My options were limited to pre-trained models of Keras[19] library, which used ImageNet dataset that includes a lot of different data for classifying many diverse objects, whereas other researchers used datasets for specific problems like face detection which are more related to emotion detection. Among the different models I performed comparisons of accuracy after first 20 epochs and ended on ResNet101, which is 101 layer network. Important to note the network performed better than ResNet18, that was used in [9], suggesting that deeper networks could be more helpful for complicated task of emotion classification.

After that I added a multilayer perceptron(MLP) for the finetuning part that would run on the FER2013 data. Final MLP ended up consisting of 4 layers of Dense layer with ReLU activation and Dropout with probabilities dependent on the problem and features.

And finally I combined all the parts together inspired by the code from [22]. That part essentially passes the image inputs through ResNet101 and gets the final features after all 101 layers, and those features are passed into the MLP so we can get from 1000 classes of ResNet101 to 7 or 2 classes specific to our problems. After all of that I apply Softmax activation function to get final predictions.

5. Experiments

In this section I cover my derivation of the best Transformer and best Finetune model I was able to achieve, and how their performance compares to the previous work and baselines.

For my experiments I decided to use Categorical Crossentropy loss[19] which is a common choice for loss functions in classification problems with many classes. Choice of optimizer fell on Adam[19] optimizer, as I used it in assignments for this class, and Adam always performed better than others.

To compare results and performance of my methods and baselines, I will be using loss and different accuracy metrics: training/validation/testing accuracies and differences among those. While numbers themselves demonstrate performance and success of models, differences demonstrate which models overfitted more, and which learned more universally.

5.1. Hyperparameter tuning

For my baselines I have not implemented much hyperparameter tuning, as there were not many parameters to change, especially with the fact that my baseline CNN came from already existing structures. For KNN I only chose the best k out of 1 to 100 and reported it in the tables following. However, for Transformer and Finetune there were a lot of options.

5.1.1 Tuning Transformers

All hyperparameter tuning for transformers was done on a small subset of the dataset with 800 training and 200 validation images and training was done for 10 epochs. The metric of best performance was validation accuracy, and in case of tie I also calculated top-3 accuracy which accounts for correct label appearing in the highest 3 predictions of the model.

I decided to choose the parameters that seemed the most important and influential. I tweaked learning rate(LR), batch size(BS), number of transformer layers(TL), and dropout rate(DR) for MLPs. I also looked at other parameters like number of heads in MLP(default is 2) and parameters for patches such as patch size (default is 6) and resized image size (default is 72), however those did not seem to anyhow affect the performance, thus they are omitted from here.

Table 1: Hyperparameter Tuning for Transformer on original images for emotion classification

	LR	BS	TL	DR	Val Accuracy
1	0.001	256	8	0.5	0.21
2	0.01	256	8	0.5	0.06
3	0.0001	256	8	0.5	0.28
4	0.0001	128	8	0.5	0.26
5	0.0001	64	8	0.5	0.19
6	0.0001	256	6	0.5	0.23
7	**0.0001**	**256**	**10**	**0.5**	**0.28**
8	0.0001	256	10	0.7	0.181
9	0.0001	256	10	0.3	0.21

As we can see model 7 and 3 performed the best with 28% accuracy, however model 7 had better top-3 accuracy.

The exact same process was conducted for the sentiment analysis task, but top-3 accuracy was not computed here. The results are presented below:

Table 2: Hyperparameter Tuning for Transformer on original images for sentiment analysis

	LR	BS	TL	DR	Val Accuracy
1	0.001	256	10	0.3	0.55
2	0.01	256	10	0.3	0.53
3	0.0001	256	10	0.3	0.57
4	0.0001	128	10	0.3	0.58
5	0.0001	64	10	0.3	0.57
6	0.0001	128	8	0.3	0.54
7	0.0001	128	6	0.5	0.51
8	0.0001	128	10	0.5	0.60
9	**0.0001**	**128**	**10**	**0.7**	**0.63**

As I attempted to perform similar process on Transformers but using HOG features, the model was not learning anything and accuracy was the same for different parameters, because model predicted the same class for all validation images. This is due to the fact that Transformers analyse images as a sequential data, and HOG images are too similar to each other with minor differences in orientation of some of the gradients. Because of that it is hard for the Transformers to use HOG features for training, thus I did not proceed with using HOG in this part.

5.1.2 Tuning Finetuned pre-trained models

Proceeding with usage of ResNet101 model pretrained on ImageNet, I had to tweak parameters of MLP that were responsible for the finetuning part: dropout rates(DR) and number of MLP heads(NH). I attempted to work with batch size, however changing it from default(128) only reduced the performance, thus not reported in the table.

Since this part was able to run relatively faster than transformer, I was able to use bigger portions of training(6400 images) and validation(1600 images) sets of the dataset. Each finetune was originally done for just 20 epochs to compare different parameters and choose the best ones. The metric of best performance again was validation accuracy.

Table 3: Hyperparameter Tuning for Finetune on original images for emotion classification

	DR	NH	Val Accuracy
1	0.2	2	0.39
2	**0.2**	**4**	**0.41**
3	0.3	4	0.39
4	0.4	4	0.38
5	0.1	4	0.39

Similar was done for the same problem but using HOG features:

Table 4: Hyperparameter Tuning for Finetune on HOG features for emotion classification

	DR	NH	Val Accuracy
1	0.2	4	0.29
2	0.4	4	0.31
3	**0.6**	**4**	**0.32**
4	0.8	4	0.31
5	0.6	2	0.28

And the same process was conducted for sentiment analysis problem:

Table 5: Hyperparameter Tuning for Finetune on original images for sentiment analysis

	DR	NH	Val Accuracy
1	0.4	2	0.64
2	0.4	4	0.65
3	**0.6**	**4**	**0.67**
4	0.8	4	0.66
5	0.2	4	0.65

Table 6: Hyperparameter Tuning for Finetune on HOG features for sentiment analysis

	DR	NH	Val Accuracy
1	0.2	2	0.59
2	0.2	4	0.6
3	0.8	4	0.6
4	0.6	4	0.58
5	**0.4**	**4**	**0.62**

5.2. Final results

I proceeded to full training for 100 epochs for best models in each of the tasks and features, that are highlighted in each of the tables. And the final results on the test set for the best 6 models can be seen below:

Table 7: Final results for the best models [1]

Model	Task	LS	TA	VL	VA	FA
TR	**EC**	**0.76**	**72.23**	**1.16**	**59.40**	**58.51**
FT	EC	0.02	99.53	4.58	47.64	48.26
FT-HOG	EC	0.03	99.67	5.28	38.17	38.03
TR	SA	0.56	70.21	0.54	72.56	71.08
FT	SA	0.01	99.89	2.6	70.64	68.86
FT-HOG	**SA**	**0.02**	**99.73**	**2.6**	**71.14**	**71.45**

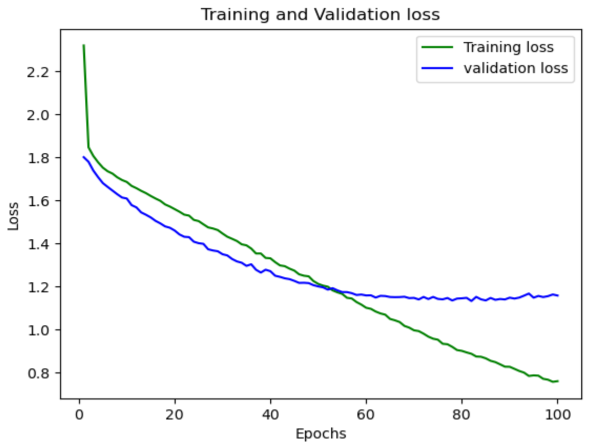
As we can see for emotion classification transformer performed the best, overshooting the final accuracy significantly compared to finetune. However for sentiment analysis finetune on HOG features performed the best, slightly getting above the transformer.

Finetuning has a clear overfitting problem as we can see based on the final loss and the 99% accuracy for training data. This issue appeared earlier during hyperparameter tuning even with low number of epochs (20). I attempted to fix this problem by making a less complex model and increasing regularisation, or in case of Finetuning - the dropout rate, however that did not appear to help at all, as model kept overfitting on the training data. My guess is that size of FER2013 dataset (35,000) is not significant compared to the size of ImageNet(1,000,000) leading to model paying closer attention to training examples, minorly tweaking parameters to work for that data, but not being broad enough for validation.

Transformers however had a relatively stable learning. If we look at loss during emotion classification, we can see that for the first 50 epochs both loses were going down together at the same rate. After 50 epochs decrease in validation loss slowed down, remained stable, but then slowly took an increasing direction by the beginning of epoch 90, indicating that we could have stopped even earlier to fully avoid overfitting, but we were not affected much by it.

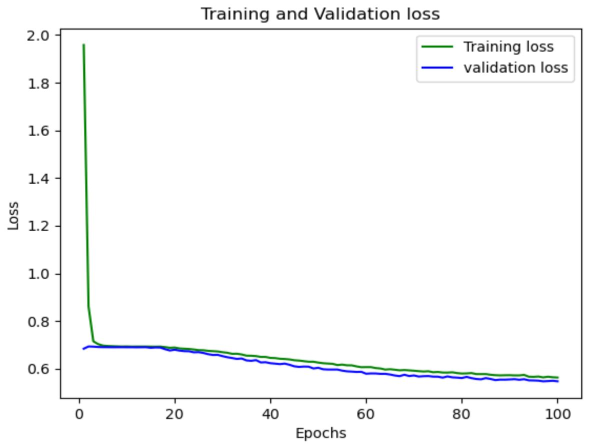
[1]Abbreviations: LS - loss; TA - training accuracy; VL - validation loss; VA - validation accuracy; FA - test accuracy; TR - transformer; FT - finetune; EC - emotion classification; SA - sentiment analysis.

Figure 5. Loss for emotion classification transformer[23]



Looking at the loss for sentiment analysis we can see better performance:

Figure 6. Loss for sentiment analysis transformer[23]



Training and validation loss here are almost identical, however don't decrease by much over the course of training, and converge by the end of it, indicating that 100 epochs is sufficient for this task.

5.3. Comparison to Baselines and Previous Work

5.3.1 Emotion Classification

The results for emotion classification(all 7 categories) can be seen below for all the models:

Table 8: All models for emotion classification

Model	Loss	Accuracy(%)
KNN(k=60)	-	16
KNN-HOG(k=50)	-	35.5
CNN	0.92	47.15
CNN-HOG	0.76	46.93
Finetune[2]	-	41.34
Finetune[9]	-	45.16
Transformer(this)	**0.76**	**58.51**
Finetune(this)	0.02	48.26
Finetune-HOG(this)	0.03	38.03

Both Transformer and Finetune that were implemented are better than any CNN or KNN methods. Finetune-HOG performed significantly worse which was not expected.

I implemented finetuning methods similar to the ones of [2] and [9] that used VGG/ResNet50 and ResNet18 respectively for pretrained models, whereas my finetuning used ResNet101. As can be seen my finetuning performs better, suggesting that more recent and more complex models have potential to improve performance. The reported accuracy is different from the one in papers because of couple factors. Authors used better and newer datasets that are available for professional researchers, whereas FER2013 has lower quality. As well as the fact that authors had access to pre-trained models on more significant data. For example in [9] the authors used 1M faces dataset for pretrained model, which is a closer data to the one used for emotion classification, compared to ImageNet data that was used in my finetuning. This essentially means that I re-implemented authors' approach partially, specifically focusing on pre-trained model in the base, to demonstrate potential improvements.

5.3.2 Sentiment Analysis

Results for sentiment analysis(positive vs. negative emotion) can be seen below:

Table 9: All models for sentiment analysis

Model	Loss	Accuracy(%)
KNN(k=100)	-	49
KNN-HOG(k=10)	-	60.5
CNN	0.58	70.27
CNN-HOG	0.40	69.53
Finetune[2]	-	73.19
CNN[6]	-	70.17
Finetune[9]	**-**	**73.67**
Transformer(this)	0.56	71.08
Finetune(this)	0.01	68.86
Finetune-HOG(this)	0.02	71.45

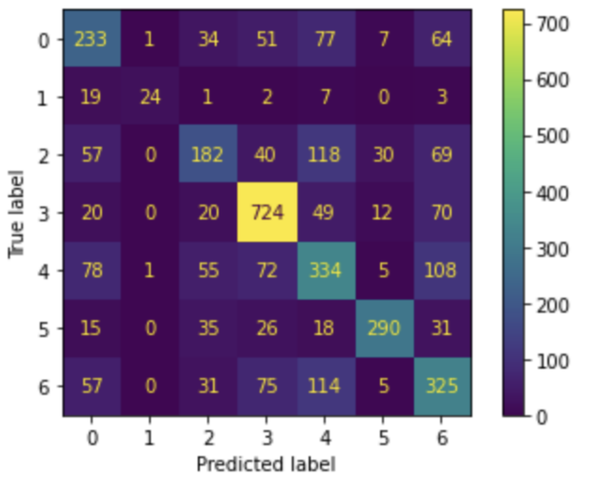
Implemented Transfomer and Finetune-HOG performed better than any baseline models. Implemented Finetune on default images performs worse than CNN, but also worse than Finetune-HOG.

We can see that both Finetunes performed worse than models similar to the ones in [2] and [9], that were implemented the same way as described in the previous section. This indicates that for a simpler problem of sentiment analysis it is not necessary to use more complex pre-trained models like ResNet101. However, with the fact that my Finetune-HOG performed better than my Finetune suggests that usage of image features can improve the existing finetunes further.

5.4. Qualitative Analysis

For this section I chose to analyse the Transfomer I implemented for emotion classification since it performed the best among all implemented methods. Below we can see the confusion matrix constructed with scikit-learn[15] for the model's predictions:

Figure 7. Confusion matrix



True label \ Predicted label	0	1	2	3	4	5	6
0	233	1	34	51	77	7	64
1	19	24	1	2	7	0	3
2	57	0	182	40	118	30	69
3	20	0	20	724	49	12	70
4	78	1	55	72	334	5	108
5	15	0	35	26	18	290	31
6	57	0	31	75	114	5	325

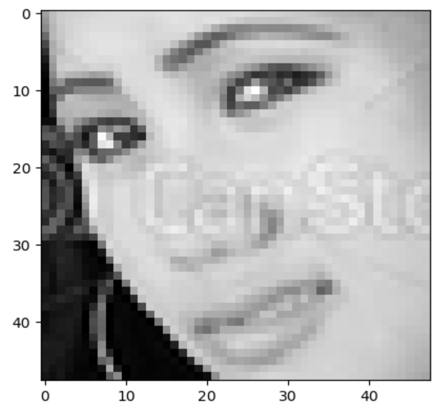
We can see that almost half of disgust(1) images were predicted as anger(0) indicating that model had hard time differentiating between the two when the disgust(1) image was given. Both emotions are quite similar as in real life they have similar eyebrow and mouth shapes that are very expressive.

Significant portion of images with fear(2) and neutral(6) emotions were predicted as sad(4), indicating that for the fear and neutral categories, sadness is another likely emotion. Again this is also expected since emotions are quite similar, and sometimes even in real life it is hard differentiate between those 3 for some people.

While happy(3) emotion had the best performance due to prevalence in data, among less represented emotions, surprise(5) emotion had the least error. This indicates that system was significantly certain when predicting image as surprise(5), as well as other labels seem very unlikely for those type of images. This result is not expected, as I thought happiness(3) and fear(2) would be similar to it due to similarity of face features, and while those 2 were the next choices, they are still not comparable to correct surprise(5) predictions.

Here we can see an example of a correctly classified image, which represents disgust(1).

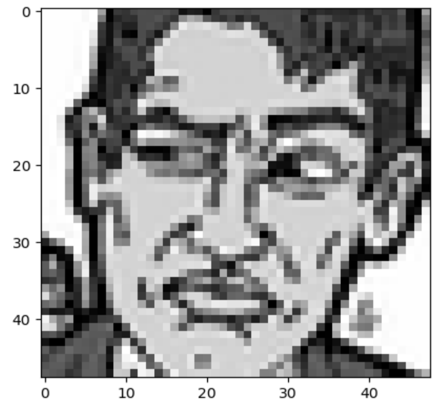
Figure 8. Correctly classified image



Personally I think this is a very hard example to classify, as this emotion could almost pass for happiness(3) with the not obvious shape of mouth we can see. That means that model was able to capture significant differences in shapes of mouth, like here the mouth is slightly closed, potentially differentiating it from happiness(3).

Here we can see an incorrectly classified image, which represents anger(0), however system predicted fear(2) for it.

Figure 9. Incorrectly classified image



This example is interesting because the image is animated rather than of real person, which can affect performance depending on the quality of animation. Clearly here this was able to confuse the system. The left side of the image does look like anger(0), but the right side is almost a surprise(5)/fear(2) resulting in the prediction made by the system.

6. Conclusion

Transformers have confirmed to still remain state of the art method in context of emotion detection. The trained transformer performed the best in emotion classification task, and had accuracy close to best model in the sentiment analysis task.

Transformers showed to have steady learning process that eventually converges to great results, but takes longer time compared to finetuned models. However such steady learning process resulted in a good advantage over finetuning in terms of overfitting on training data. As we saw accuracy and loss were similar for both validation and training data for transformers, whereas finetuning on pre-trained models significantly overfitted. However, for future work I would like to investigate this further, potentially with a larger and higher quality dataset, as finetuning shows to perform well in other fields.

This project has shown that image features like HOG have potential for emotion detection with finetuning on pre-trained models, particularly in the sentiment analysis task. However, the features did not prove to be useful at all for emotion classification, and especially during transformer training. For future work it is definetely worth exploring other image features, and considering which ones could be useful for transformers and emotion classification.

Another thing important to explore is representation within the dataset. As we saw model incorrectly predicted for the animated image, but there could be other factors that can influence the model such as race, sex or age. If there is not enough representation for different groups of people or image types, then training process could be biased and not work for underrepresented groups.

References

[1] ”Emotion in the human face.” P Ekman, W Friesen, P Ellsworth. Pergamon Press New York. 1972. https://www.elsevier.com/books/emotion-in-the-human-face/ekman/978-0-08-016643-8

[2] ”Emotion Detection and Sentiment Analysis of Images.” V Gajarla, A Gupta. Georgia Institute of Technology. 1-4. 2015. https://faculty.cc.gatech.edu/ hays/7476/projects/Aditi_Vasavi.pdf

[3] ”PlaNet - Photo Geolocation with Convolutional Neural Networks.” Weyand, T., Kostrikov, I., Philbin, J.,

Leibe, B., Matas, J., Sebe, N., Welling, M. Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(). vol 9912. Springer, Cham. https://doi.org/10.1007/978-3-319-46484-8_3

[4] ”Very Deep Convolutional Networks for Large-Scale Image Recognition.” K Simonyan, A Zisserman. arXiv. 2014. https://arxiv.org/abs/1409.1556

[5] “Learning Deep Features for Scene Recognition using Places Database.” B Zhou, A Lapedriza, J Xiao, A Torralba, and A Oliva. Advances in Neural Information Processing Systems 27 (NIPS). 2014. http://places.csail.mit.edu/places_NIPS14.pdf

[6] ”Facial Emotion Detection Using Deep Learning.” A Jaiswal, A Krishnama Raju, S Deb. International Conference for Emerging Technology (INCET). 2020. pp. 1-5. https://ieeexplore.ieee.org/document/9154121

[7] ”Visual Sentiment Prediction with Deep Convolutional Neural Networks.” C Xu, S Cetintas, K Lee, L Li. 2014. https://arxiv.org/pdf/1411.5731.pdf

[8] ”Performance Evaluation of Supervised Machine Learning Techniques for Efficient Detection of Emotions from Online Content.” M Asghar, F Subhan, M Imran, F Kundi, A Khan, S Shamshirband, A Mosavi, P Csiba, A Koczy. Computers, Materials & Continua. 2020. vol. 63, pp. 1093-1118. http://www.techscience.com/cmc/v63n3/38864

[9] ”Facial Expression Recognition with Visual Transformers and Attentional Selective Fusion.” F Ma, B Sun, S Li. IEEE Trans. Affective Comput. 2021. 1-1. https://doi.org/10.48550/arXiv.2103.16854

[10] ”MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition.” G Yandong, Z Lei, H Yuxiao, H X, and G Jianfeng. ECCV. 2016. https://doi.org/10.48550/arXiv.1607.08221

[11] FER2013 Dataset: ”Challenges in Representation Learning: A report on three machine learning contests.” I Goodfellow, D Erhan, PL Carrier, A Courville, M Mirza, B Hamner, W Cukierski, Y Tang, DH Lee, Y Zhou, C Ramaiah, F Feng, R Li, X Wang, D Athanasakis, J Shawe-Taylor, M Milakov, J Park, R Ionescu, M Popescu, C Grozea, J Bergstra, J Xie, L Romaszko, B Xu, Z Chuang, and Y. Bengio. arXiv. 2013. http://arxiv.org/abs/1307.0414

[12] ”PIL”. A Clark, and contributors. 2010-2022. https://pillow.readthedocs.io/en/stable/about.html

[13] ”scikit-image: Image processing in Python.” S Walt, J Schönberger, J Nunez-Iglesias, F Boulogne, J Warner, N Yager, E Gouillart, T Yu and the scikit-image contributors. PeerJ 2:e453. 2014. https://doi.org/10.7717/peerj.453

[14] ”Array programming with NumPy.” C Harris, K Millman,S Walt, et al. Nature 585. 2020. 357–362. https://doi.org/10.1038/s41586-020-2649-2

[15] ”Scikit-learn: Machine Learning in Python.” Journal of Machine Learning Research. 2011. vol. 12, pp. 2825-2830. https://scikit-learn.org/stable/index.html

[16] Pytorch: ”Advances in Neural Information Processing Systems 32”. Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and Desmaison, Alban and Kopf, Andreas and Yang, Edward and DeVito, Zachary and Raison, Martin and Tejani, Alykhan and Chilamkurthy, Sasank and Steiner, Benoit and Fang, Lu and Bai, Junjie and Chintala, Soumith. Curran Associates, Inc. 2019. pp. 8024-8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[17] ”CS231N Convolutional Neural Networks for Visual Recognition: Assignment 2.” Stanford University. 2022. https://cs231n.github.io/assignments2022/assignment2/

[18] ”TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.” Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. tensorflow.org

[19] ”Keras.” F Chollet and others. 2015. https://keras.io/api/

[20] ”Image classification with Vision Transformer.” K Salama. Keras.io. 2018. https://keras.io/examples/vision/image_classification_with_vision_transformer/

[21] ”Pandas-dev/pandas: Pandas.” The pandas development team. Zenodo. 2020. https://doi.org/10.5281/zenodo.3509134

[22] ”Transfer learning and the art of using Pre-trained Models in Deep Learning.” D Gupta. Analytics Vidhya. 2017. https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/

[23] ”Matplotlib: A 2D graphics environment.” J Hunter. IEEE COMPUTER SOC. 2007. vol. 9, n. 3, pp. 90-95. https://matplotlib.org/stable/index.html