

# Facial Expression Analysis

Lawrence Pablo

Selden Lin

John V. Lal

## Abstract

The review of novel literature on Facial Expression Analysis that started this project exposed significant variation in the way this problem was tackled. The interest of this project was to evaluate these techniques against an end goal of mapping a subject to an emoji and compile these techniques to form a system that accomplishes this goal. To that end, the project employs the anatomy based Facial Action Coding System (FACS) to identify facial muscle movements through "Action Units". To do this, ResNet-18 was first pretrained on data from the CelebA database with noisy labels from OpenFace 2.0. Following this, the model weights were fine tuned using images from the Cohn-Kanade database with high quality labels. The Action Units extracted by ResNet-18 were then matched to an emoji using Cohen's Kappa coefficient. This report primarily details this approach and the experiments we conducted while implementing it. Preceding that, we explore the facts that informed key decisions while deciding on this approach. The paper ultimately hopes to give the reader an understanding of every detail surrounding this flexible system which can identify human facial expression well with variations in subjects and their environments.

# 1 Logistics

The full source code for this project can be found [here](#). A breakdown of the repo can be found in [section 7](#). Use `ui_download_link.txt` in the repo for instructions on how to run the GUI. There are 2 GUIs: one for loading one image and getting an emoji result back and another for using a webcam to give a real time emoji estimate.





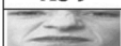



# 2 Introduction and Literature Review

## 2.1 The Facial Action Coding System

The work for this project started with a review of novel approaches to facial expression recognition guided initially by Canedo(2019). The goal was to find the best approach to facial expression analysis as it relates to our end goal of translating a frame to an emoji. With this goal in mind, it becomes clear why a machine learning attempt at mapping a person to one of 6-8 emotions (Canedo, 2019) may not be quite as effective. To illustrate this claim, consider a picture of a happy person. The person may have their eyes open or closed. There are several other variations within the ‘happy’ class that would be lost if we chose to continue with this model. This is a missed opportunity given that we have enough emojis to capture practically all of these variations.

In an effort to capture these variations within emotion labels, a machine learning approach to analyzing facial anatomy was adopted. This approach hopes to map a picture to one or more of 23 Facial Action Units as described by the Facial Action Coding System ( FACS ). The Facial Action Coding System is comprehensive (Farnsworth, 2019), and helps remove the loss of detail associated with relying entirely on emotion. Additionally, an approach based in anatomy has lower reliance on how the subject actually feels and hence improves the quality of the output. To further illustrate this, we can look at an example from Orozco (2018). The paper finds that there may be scenarios where contempt may not have many discernible differences from a neutral face, and hence their model has trouble distinguishing the two. However, the emoji outputs for contempt and neutral may be wildly different. FACS thus ensures that faces with mild differences would not result in significantly different outputs and maintains overall model consistency.

To illustrate how FACS works, we can take a look at what these Action Units are. The attached diagram from Fernando et. al. (2015) show how facial expression can be broken down into these action units. They describe a state in one region of the face. It becomes clear to see now how these Action Units can be much more flexible when we try to map the human face to an emoji. We can detect these Action Units in human faces and combine them to form our emojis.

Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
					
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
					
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
					
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
					
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
					
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

**Figure 1:** Some examples of action units. Only 18/44 action units are used in this project due to limitations in data.

To map these Action Units to emojis, we need to use a subset of these Action Units for each emoji. A breakdown of the emojis used in this project and the Action Units associated with them can be found in [section 6](#). The table illustrates perfectly how good this new model is at capturing facial expressions. We can now even extract more information than can be captured in emojis.

## 2.2 Database Availability and Choice

With this narrow focus on FACS, however, the issue becomes data availability. There are very few datasets with labelled Facial Action Units, in part due to the expert prior knowledge that is involved with labelling subjects accurately. Given this narrow focus, the Extended Cohn-Kanade (CK+) dataset was the best available option. The dataset uses 123 subjects who were instructed by an experimenter to perform a series of 23 facial displays for a total of 593 image sequences. The combination of uniform lighting and the doubly verified labels (Kanade et al., 2010), make Cohn-Kanade the best starting point for our model.

This narrow focus comes with another set of issues as illustrated through analogous studies. Similar models trained on small datasets seem to be extremely sensitive to changes in lighting with significant drop offs in accuracy when dealing with new faces (Orozco, 2018). The best solution to combat these issues is pre-training using noisy data. For this pretraining, we picked images from the celebA dataset labelled using OpenFace 2 (Baltrusaitis et. al, 2018). CelebA, in addition to being reasonably cropped and aligned, gives several attributes including facial landmarks which can be valuable in processing these images efficiently. (Liu, 2016). With this information, the approach for this project was decided to be Facial Action Unit recognition using noisy pretraining and fine tuning using the Cohn Kanade dataset.

## 2.3 Comparable Techniques

There were many variations in techniques for facial expression recognition explored in the research for this project. The aim of this project was to contrast and combine existing techniques to best fit the goal of emoji matching. To that end, the reasons for adopting an approach based on the Facial Action Coding System has already been explored in this section. So has the reasons for choosing a combination of noisy pre-training with CelebA and fine-tuning with the Cohn-Kanade database. Another key decision involved the facial recognition algorithm used in the pre-processing steps. The most frequent facial recognition algorithm used in facial expression recognition seemed to be The Viola-Jones face detector. However, Viola-Jones is most effective in controlled environment databases which is not the case for the pre-training data we use. Furthermore, other face detection algorithms perform better than Viola Jones in several benchmarks while keeping real time performance (Canedo, 2019) and hence, the dLib face detector was used.

# 3 Methodology, Results and Experiments

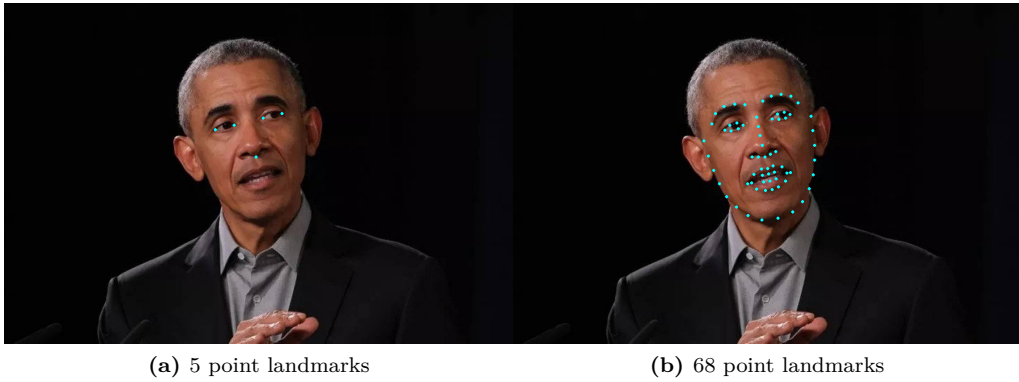
## 3.1 Preprocessing

### 3.1.1 Image cropping and alignment

As Canedo (2019) explains, the best way to maximize model quality is to ensure that the images are as geometrically similar as possible. Classifiers trained on such data are bound to be more reliable. To that end, the faces were cropped and aligned to ensure consistency. The example below perfectly illustrates what we did in our preprocessing steps.



**Figure 2:** Sample pre-processed image



**Figure 3:** Comparison of the two facial landmarks

To do this, We used facial landmarks as illustrated below. Note that although there are two different systems to get facial landmarks, their utility remains the same and hence, the 5 point landmarks were used for this project. We used these points to get a tight bound on the face. Once we had this bound, we used the locations of the pupils to calculate the angle correction required. With these two steps, we ensured that all images fed to the model was cropped and the aligned perfectly with the pupils to maintain the highest level of consistency between them.

As mentioned throughout this report, pretraining was to be done with images from the CelebA database with noisy labels from OpenFace 2.0. One main advantage to using CelebA was the fact that each image from the database came with their 5 point landmarks. This made it much more efficient to crop and align the 140,000 images we used for pre-training. However, the Cohn-Kanade database we used for finetuning the model did not have the same landmark data and hence we used the dlib face detector.

The dlib face detector takes a HOG+SVM approach to face detection. This approach extracts the Histogram of Oriented Gradients (HOG) data, as discussed in class. This data is then fed into a Support Vector Machine to classify the region as a face.

### 3.1.2 Other pre-processing steps

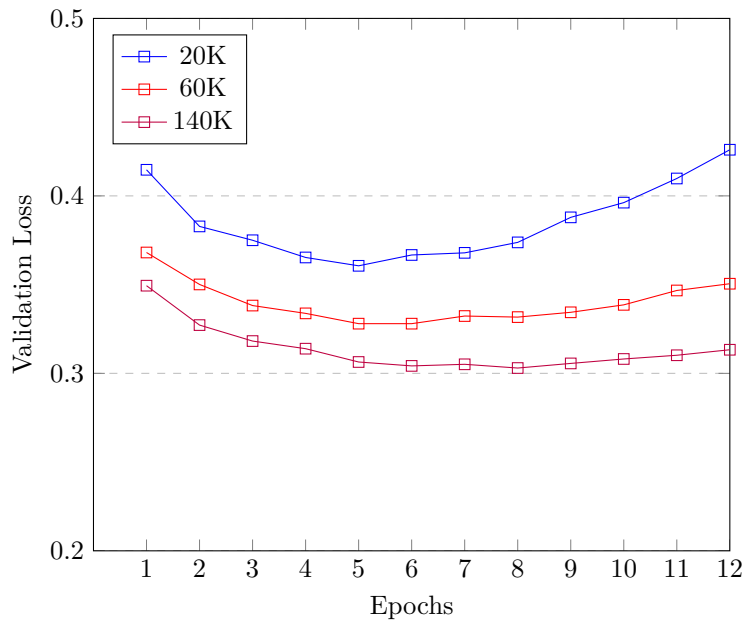
The images were left in color and not converted to grey-scale due to ResNet-18's architecture. Any approach to making ResNet work with greyscale images, whether it included modifying the input or the model, resulted in significant drop offs in accuracy.

The final pre-processing step was mean normalizing the images. The mean and standard deviations for each dataset was manually calculated by iterating through the images. The mean and standard deviation for the 140,000 images from CelebA used for pre-training were (0.4007, 0.4783, 0.6502) and (0.1641, 0.1755, 0.2007). As for the images used for fine-tuning, the mean and standard deviation were (0.3898, 0.3884, 0.3878) and (0.2433, 0.2430, 0.2430).

## 3.2 Pre-training

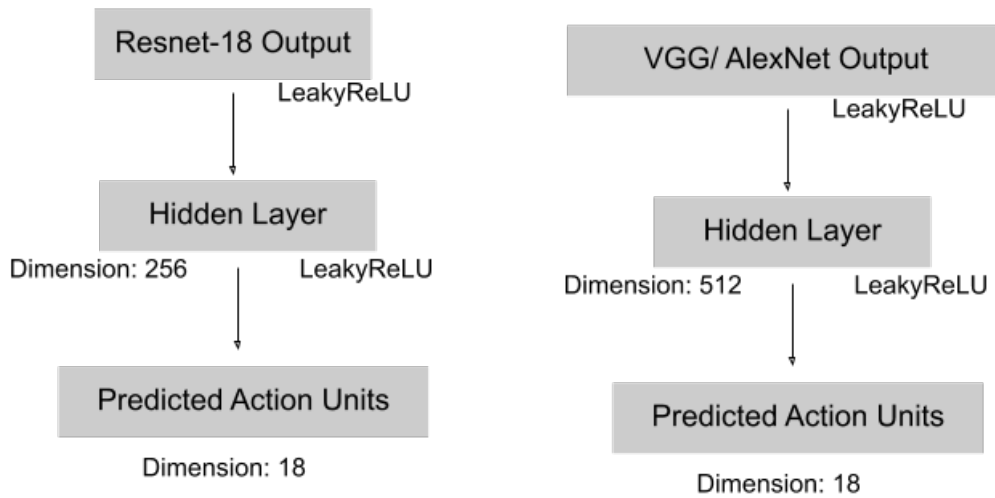
### 3.2.1 Data Size

The noisy pretraining uses cropped and aligned images from the celebA database labelled with Action Units using OpenFace 2.0. This meant that we had access to around 140,000 images with these imperfect labels. A 95/5 training/validation split was used, and was stratified by gender. We recognized that there is a tradeoff between time and accuracy here. The first thing we wanted to establish was that there was a positive correlation between the number of images used and the accuracy. If there wasn't significant improvement in accuracy when using all the data, we could use the minimum amount of data required for our pretraining to significantly save on time. However, as illustrated below, there was enough improvement in model accuracy to justify using all the data.



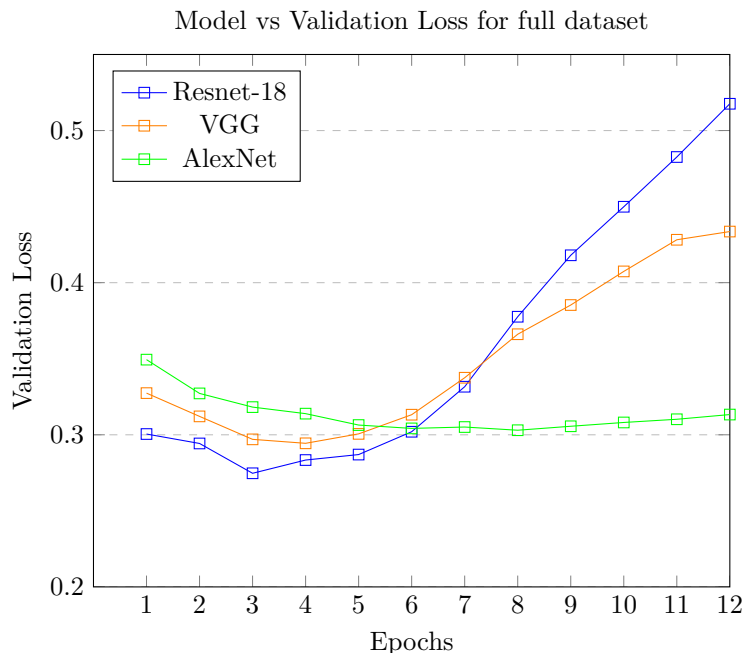
**Figure 4:** *Effect of dataset size on Validation Loss*

### 3.2.2 Model



**Figure 5:** *Resnet’s final fully connected layer has dimension 512 where VGG & AlexNet have a 1000, hence the difference in dimension for the hidden layer.*

Guided in part by Yan (2019) and some experimentation, the final pretrained models we considered for our project were AlexNet, VGG, and Resnet-18. All three models were trained on 1.2 million images from the Imagenet dataset. We used the weights from this training as the starting point for our models. The models were then modified as described above to identify Action Units. The performance of these models while pre training was used as a stand in for real world performance and hence, the model with the lowest pre-training validation loss was picked. Regularization steps as suggested by Yan (2019) were removed due to indications that this was causing underfitting. Additionally, early training seemed to indicate a “dying” ReLU problem and to mitigate that, the LeakyReLU was adopted which seemed to improve validation accuracy.



**Figure 6:** *The models are saved when they reach their lowest validation loss. Even though Resnet starts to overfit significantly in the later epochs, the lowest loss achieved using resnet is significantly lower than the rest and hence we will use that for our model.*

### 3.3 Finetuning

#### 3.3.1 Dataset

To finetune our model, we utilized the Extended Cohn-Kanade Dataset (CK+), which provides full, reliable FACS coding of peak frames for the intensity levels (0 to 5) of 30 AUs (Kanade et al., 2010). Labels for AUs 11, 13, 14, 16, 18, 21, 31, 37, and 39 were not collected due to low frequencies in the dataset which we expect will make the action unit hard to detect for the CNN, as well as due to incompatibilities with the noisy labels of the pretraining dataset. For the remaining action units, peak frames with an intensity level of at least 1 were considered positive examples and others negative examples. In addition to the posed expressions enacted by each subject, a single frame containing a neutral expression by each subject was added to the samples, and is considered a negative example for all action units (no expression). The pre-processing steps outlined above were applied prior to training and inference.

#### 3.3.2 Model

The model architecture from the pretraining state was preserved, and its weights used as starting points for the training on the fine-tuning dataset. Experimentation was done to determine which layers of the CNN should be kept frozen so as to minimize overfitting due to a limited amount of data and a relatively large number of trainable parameters. The model’s performance on a held-out dataset in the different configurations also demonstrate the effectiveness of pretraining in finding useful features in the image for detecting the presence of action units.

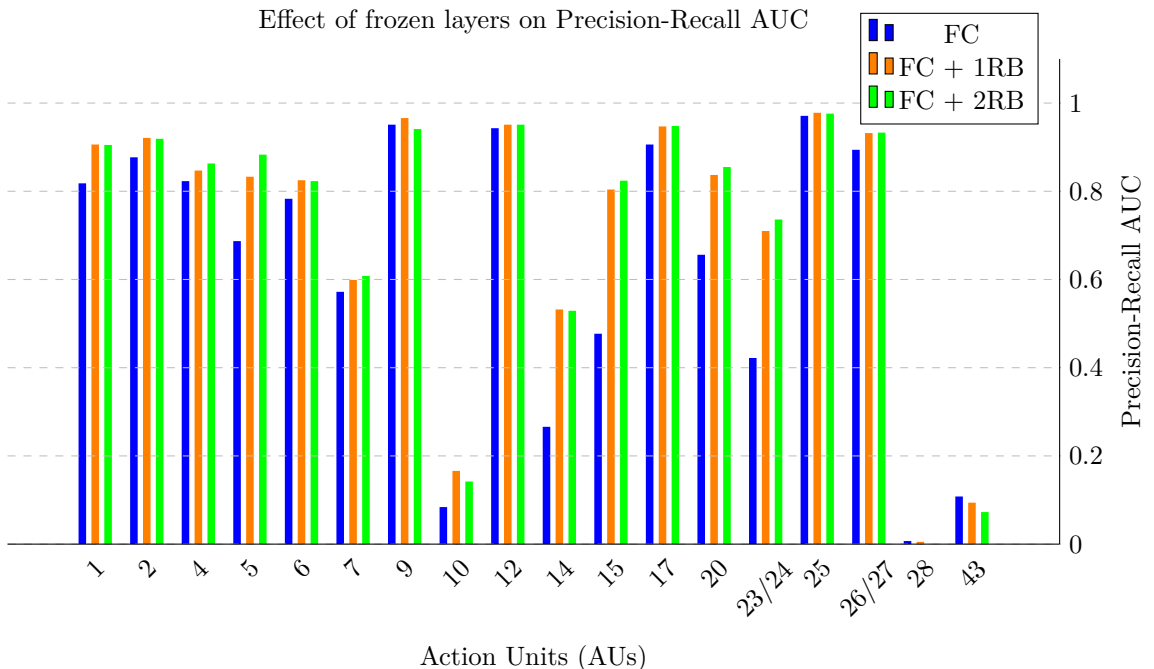
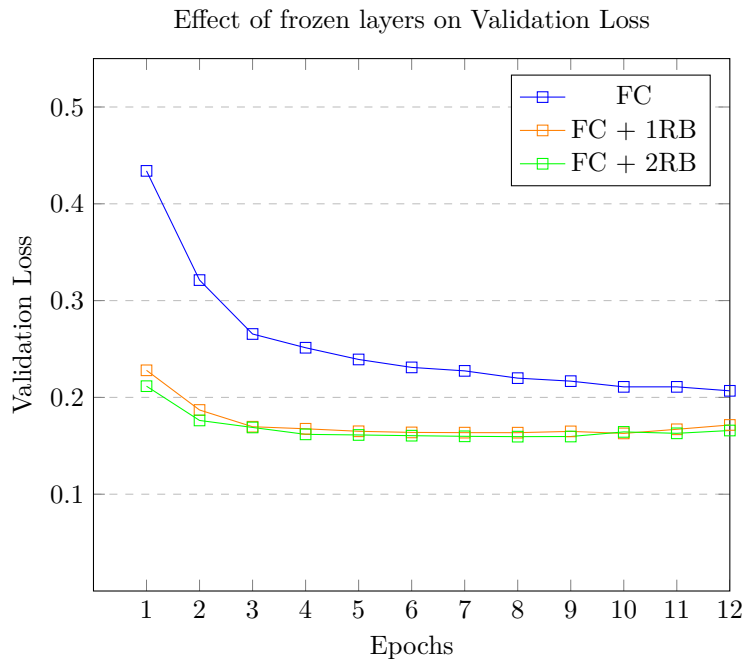
Three settings were considered after all model weights were frozen: unfreeze only the fully connected layers, unfreeze the fully connected layers and the last residual block, and unfreeze the fully connected layers and the last two residual blocks. To evaluate the model’s performance on unseen data in these different settings, we use the validation loss during training time so that we can examine convergence, and we use the Precision-Recall AUC to capture overall model performance with a highly skewed dataset (Davis and Goadrich, 2006). Due to a limited amount of data in the fine-tuning dataset, applying a three-way hold out set up will make our estimates of validation loss and PR-AUC quite sensitive to our choice of splits (Raschka, 2016), so we use a 3-fold subject exclusive cross-validation set up to reduce the variance of our estimates.

The results of our experiment shown in Figure 4 demonstrate that unfreezing the first residual block allows the model to converge more quickly to the minimum validation

loss. Unfreezing another layer however results in diminishing returns as we only see a slight decrease in validation loss while also increasing the risk of overfitting to the dataset. In addition, the model’s generalization ability increases quite a bit when the first residual block is unfrozen, suggesting that the pretraining stage was important for extracting more low level features while high level feature extraction must be optimized for the fine-tuning dataset. On the other hand, for some action units the model loses a bit of generalization ability while also slightly improving for others, which may suggest that the model begins to overfit when another layer becomes unfrozen. As a result, before proceeding with training, all weights and biases except those in the fully connected layers and the last residual block were frozen.

### 3.3.3 Training

For training, 90/10 training and test split was used, with the training set being further split up for subject exclusive 3-fold cross validation for hyperparameter tuning. The model was then retrained on the entire training set using the Adam optimizer with an initial learning rate of 0.0001,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and batch size of 32. Early stopping was used to ensure that the model did not fit on the training set.



**Figure 7:** *Top:* A plot of training curves for the model trained under the three settings of frozen layers. *Bottom:* PR-AUC scores for the model trained under the three settings of frozen layers.

### 3.3.4 Results

For each individual action unit, we measure the networks ability to detect its presence through precision, recall, and F1-score when using thresholds  $[0.9, 0.6, 0.75, 0.5, 0.5, 0.5, 0.25, 0.2, 0.75, 0.45, 0.7, 0.65, 0.75, 0.3, 0.8, 0.5, 0.9, 0.1]$ . Table 1 shows the results of the performance metrics on a subject exclusive hold out set of CK+. Note that due to the size of the test set, the scores for some AUs (in particular, AU 10 and 43) are inflated as positive examples appear only once or twice, and the network was able to guess correctly each time. The overall average precision, recall, F1-Score and PR AUC obtained on the hold out set was **0.867**, **0.726**, **0.772** and **0.856** respectively, which suggests that the model is able to generalize pretty well to unseen examples of CK+.

	Precision	Recall	F1-Score	PR-AUC
AU 1	0.867	0.867	0.867	0.971
AU 2	0.900	0.750	0.818	0.874
AU 4	0.875	0.824	0.848	0.927
AU 5	0.900	0.750	0.818	0.882
AU 6	1.000	0.600	0.750	0.789
AU 7	0.600	0.750	0.667	0.723
AU 9	1.000	0.800	0.889	1.000
AU 10	1.000	1.000	1.000	1.000
AU 12	0.875	0.700	0.778	0.808
AU 14	1.000	0.143	0.250	0.690
AU 15	0.889	0.727	0.800	0.956
AU 17	0.889	0.842	0.865	0.958
AU 20	1.000	0.750	0.857	1.000
AU 23/24	1.000	0.875	0.933	0.944
AU 25	0.923	0.889	0.906	0.948
AU 26/27	0.889	0.800	0.842	0.930
AU 28	0.000	0.000	0.000	0.000
AU 43	1.000	1.000	1.000	1.000
Mean	<b>0.867</b>	<b>0.726</b>	<b>0.772</b>	<b>0.856</b>

**Table 1:** Precision, recall, F1-score and PR AUC for each individual AU

### 3.3.5 Emoji Matching

Let  $\mathbf{E} = \{E_1, E_2, \dots, E_n\}$  be the set of emojis that we wish to match to a facial expression, each with label vectors  $\ell(E_1), \ell(E_2), \dots, \ell(E_n)$  and  $\hat{\mathbf{y}}$  the vector of AUs detected by the network. Given an agreement metric  $\alpha$ , we will return the emoji  $E_i$  that satisfies

$$E_i = \operatorname{argmax}_{E_i \in \mathbf{E}} \alpha(\hat{\mathbf{y}}, \ell(E_i))$$

The agreement metric used to match emojis to the predictions made by the network was Cohen’s Kappa coefficient:

$$\alpha(\hat{\mathbf{y}}, \ell(E_i)) = \kappa = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}$$

where  $p_0$  is the relative observed agreement between the network’s predictions  $\hat{\mathbf{y}}$  and the labels corresponding to an emoji  $\ell(E_i)$  (equivalent to accuracy of network’s predictions if  $\ell(E_i)$  is considered to be the ground truth labels)

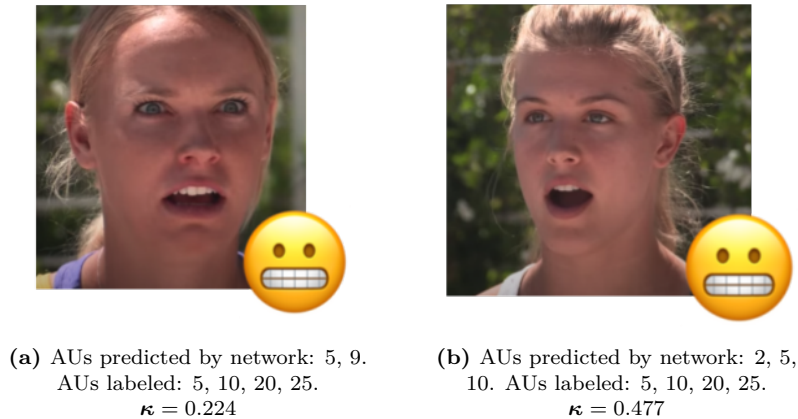
$$p_0 = \frac{1}{18} \sum_{j=1}^{18} 1(\hat{y}_j = \ell(E_i)_j)$$

and where  $p_e$  is the estimated probability of chance agreement, assuming the network predictions and the emoji labels  $\ell(E_i)$  are independent:

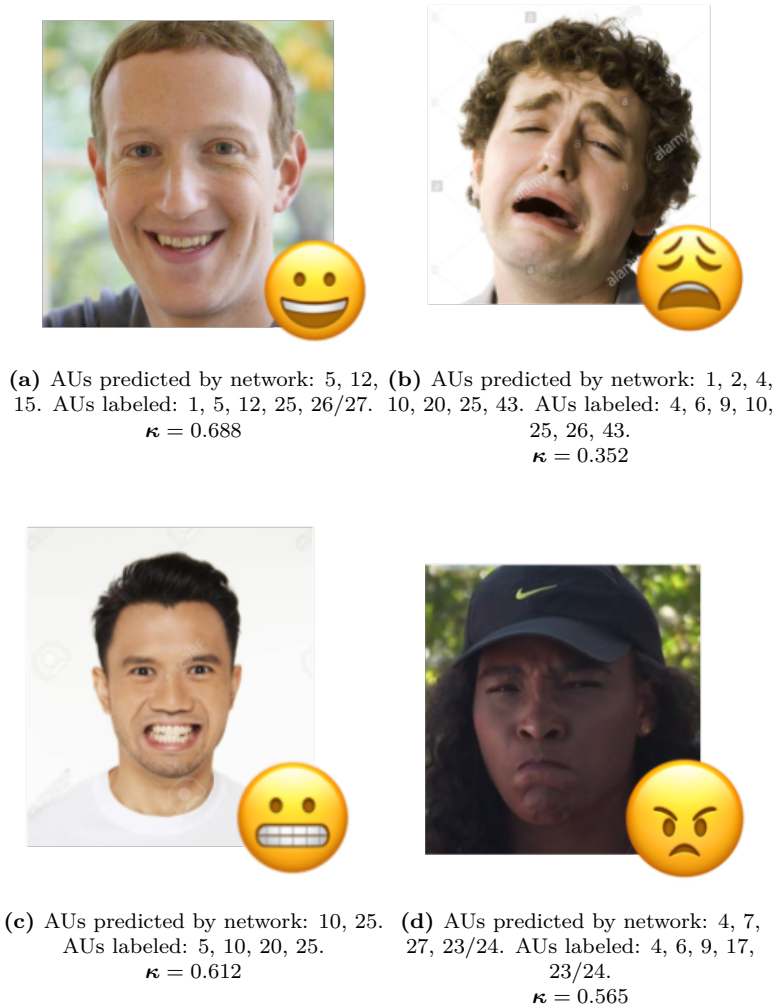
$$\begin{aligned} p_e &= \Pr(\hat{y} = 0, \ell(E_i) = 0) + \Pr(\hat{y} = 1, \ell(E_i) = 1) \\ &= \left[ \left( \frac{1}{18} \sum_{j=1}^{18} 1(\hat{y}_j = 0) \right) \left( \frac{1}{18} \sum_{j=1}^{18} 1(\ell(E_i)_j = 0) \right) + \left( \frac{1}{18} \sum_{j=1}^{18} 1(\hat{y}_j = 1) \right) \left( \frac{1}{18} \sum_{j=1}^{18} 1(\ell(E_i)_j = 1) \right) \right] \end{aligned}$$



If the network’s predictions and the emoji labels are in perfect agreement, then  $\kappa = 1$ , and if there is no agreement between the network’s prediction and the emoji labels other than what would be expected by chance, then  $\kappa = 0$  (“Cohen’s Kappa”, 2020).



**Figure 8:** Unsuccessful matches between network predictions and emoji labels via Cohen’s Kappa



**Figure 9:** Successful matches between network predictions and emoji labels via Cohen’s Kappa

Figure 7 presents examples for which the network and the agreement metric were unable to return the correct emoji for the input facial expression. In cases of failure, the network is generally at fault due to its inability to detect action units either due to facial poses that the network is not used to seeing, poor lighting conditions, or poor performance due to the lack of positive examples in the dataset. Indeed, the model’s performance drops when it encounters angles other than the fully frontal angle that it was solely trained on in the CK+ dataset. Another source of failure is due to errors in the labeling of emojis, where it may be unclear whether the presence of an AU in an emoji comes from the fact that it is present in

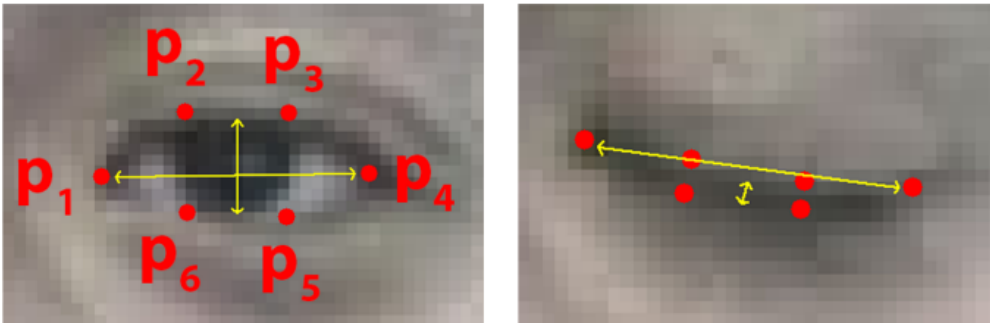
the emoji itself or it is present in a subject’s face when they imitate the emoji’s expression. Furthermore, future work should compare the performance of the Kappa coefficient with other agreement metrics such as accuracy in matching action unit labels. Since action units can be usually grouped to regions in the face, a potential metric may relax requirements for exact matches which treat different action units as completely different entities and instead focus on matches within groups of similar action units.

### 3.4 Issues with eye detection

A limitation of the databases we used, CelebA & Cohn-Kanade, was the number of images of people with their eyes closed. A consequence of this was the fact that our model could not be trained well to recognize AU 43 & AU 46. To fix this issue, we decided to create an independent step outside the CNN we had trained to detect closed eyes. Based on work by Soukupová & Čech( 2016), the hope was to use the 68 point landmark to help determine whether a person had their eyes closed or open. To do so, we describe the Eye Aspect Ratio (EAR):

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_2 - p_4\|}$$

These points are the landmarks around the eye recognized through the 68 point landmark detector, for which we used dlib. The following figure from Soukupová & Čech(2016) illustrates this a bit better relative to the variables used in the equation. Once we have an estimate of the EAR, we can use a threshold on the ratio to determine whether an eye is closed. The paper estimated that any ratio below 0.3 should be considered a blink. However, testing this proved to result in significant numbers of false positives. Hence, a lower threshold of 1.9 was used to detect a closed eye.



**Figure 10:** We calculate the vertical distance relative to horizontal distance to gauge whether the eye is closed

This initial hiccup in our final model, however, proved to further illustrate the flexibility of the FACS based approach we adopted for our project. If at any point we find that our model is not effective at identifying an Action Unit, we can always try to find a better solution. An emotion based approach may not be quite as accurate when the subject has their eyes closed due to the fact that the CNN would have seen very little data of people with their eyes closed. However, with the FACS approach we always have the option to find other ways to compensate for missing data while still being assured that the model stays reasonably accurate.

## 4 Conclusion

In our eventual goal of mapping a subject’s facial expressions to an emoji, the Facial Action Coding System (FACS) allows us to capture activity in multiple independent regions of the face via action unit (AU) constituents. As such, it provides us with a way of encoding a subject’s facial features into a low dimensional vector, from which we can map to a corresponding emoji that also has the same encoding of its depicted facial expression. The ability to encode facial expressions gives us a wider variety of emojis that we can to model, as we can differentiate emojis portraying the same emotion but differ only in subtle facial features.















In the task of mapping a subject’s facial expression to its AU encoding, a convolutional neural network provides an end-to-end framework for both feature extraction and a mapping from features to an AU encoding. With the limited availability of data containing

accurately labeled action units present in a face, we showed that pretraining a network with a large amount of noisy data, obtained from an openly available dataset, can be quite beneficial in learning features that allow the network to discern between different facial expressions. Though the labels generated through OpenFace were generally incorrect, our experiments showed that increasing the number of images for pretraining directly increased model generalization ability as the model was able to learn to ignore uninformative variations in the data after seeing such a wide variety of face poses and types of subjects. After deciding that the ResNet architecture was the most suitable for our task, further experimentation showed that pretraining improved training with the clean dataset as the model had already learned lower level feature extraction methods needed to detect action units. As a result, the model only had to learn very high level feature extraction and the mapping from features to the AU encoding with a limited amount of accurate data. The model can be further improved by extending the pretraining dataset and especially the fine-tuning dataset to include more variety of facial poses, environmental conditions and types of subjects to increase model generalization ability.

The task of retrieving a matching emoji based on presence of action units relied on an agreement metric that measured how similar a person’s facial expressions were to an emoji’s depicted expression. Further improvements can be made by ensuring that the action units labeled for each emoji are more accurate, and making the agreement metric more “aware” of the groups that action units can form over a region of the face to allow matching between different action units within the same group.

In closing, our model is able to take subjects with a frontal angle towards the camera, and output an emoji out of 14 possible emojis with varying facial expressions, and to output what the agreement metric deems to be the closest in terms of action units. However with further improvements, this model should be capable of taking in subjects in varying angles and poses and outputting a corresponding emoji with better accuracy.

## 5 Breakdown of emojis used in project

	Inner Brow Raiser+Upper Lid Raiser+Lip Corner Puller+Lip Parts+Jaw Drop/ Mouth Stretch
	Inner Brow Raiser + Cheek Raiser + Lip Corner Puller + Lip Parts+Jaw Drop/ Mouth Stretch + left eye closed + Right eye closed
	Cheek Raiser + Lid Tightener + Lip Corner Puller + Lip Parts + Right Eye Closed + Left Eye Closed
	Lip Corner Puller
	Inner Brow Raiser + Cheek Raiser + Lip Corner Puller
	[ neutral ]
	Inner Brow Raiser + Outer Brow Raiser + Upper Lid Raiser + Lip Parts + Jaw Drop/ Mouth Stretch
	Inner Brow Raiser+Outer Brow Raiser+Upper Lid Raiser+Lip Parts+Jaw Drop/ Mouth Stretch
	Brow Lowerer+Lip Corner Compressor+Right Eye Closed+Left Eye Closed
	Brow Lowerer+Lip Corner Compressor+Chin Raiser
	Brow Lowerer+Nose Wrinkler+Upper Lip Raiser+Lip Corner Puller+Right Eye Closed+Left Eye Closed
	Brow Lowerer+Cheek Raiser+Nose Wrinkler+Upper Lip Raiser+Lip Parts+Jaw Drop/ Mouth Stretch+Right Eye Closed+Left Eye Closed
	Brow Lowerer+Cheek Raiser+Nose Wrinkler+Chin Raiser+Lip Tightener/ Pressor
	Lip Corner Puller+Right Eye Closed

## 6 Repo Breakdown

```
Repo
├── _ui_download_link.txt Instructions for UI download and setup
├── Ui.py Main UI to run and execute the full project
├── final
│   ├── emoji_labels.csv Mapping from AUs to emojis
│   ├── emoji_matcher.py code to map from AUs to emojis
│   └── eyes_closed_detector.py Manual detector for eyes closed
├── finetuning
│   ├── Experiments.ipynb Experiments with models trained on CK+
│   ├── Finetuning.ipynb Finetuning using CK+
│   ├── align_faces.py Aligns faces using dlib
│   ├── mmod_human_face_detector.dat dat file for dlib face detector
│   ├── model_weight_links.txt links to weights for trained model
│   └── finetune_data_links.txt links to training/validation/test splits
├── pretraining
│   ├── align_faces.py Aligns faces using dlib
│   ├── align_faces_with_landmarks.py Aligns faces using landmarks from
│   │   from CelebA
│   ├── celebAU.py Custom dataset object created for pre training
│   └── pretraining_final.ipynb
└── README.md
```

## 7 References

- Canedo, D., Neves, A. J. R. 2019. Facial Expression Recognition Using Computer Vision: A Systematic Review. *Applied Sciences*, 9(21), 4678. doi:10.3390/app9214678
- Davis, J., Goadrich, M. (2006). The Relationship between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 233–240). Association for Computing Machinery.
- Farnsworth, Bryn. 2019. Facial Action Coding System. iMotions. <https://imotions.com/blog/facial-action-coding-system/>
- Tian, Y., Kanade, T., Cohn, J.F. 2010. Comprehensive Database for Facial Expression Analysis. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, pp. 484–490, Grenoble, France.
- Orozco, D., Lee, C., Arabadzhi, Y., Gupta, D. 2018. Transfer learning For Facial Expression Recognition. UC San Diego. [http://noiselab.ucsd.edu/ECE228\\_2018/Reports/Report7.pdf](http://noiselab.ucsd.edu/ECE228_2018/Reports/Report7.pdf)
- Liu, Z., Luo, P., Wang, X., Tang, X. 2016. Large scale CelebFaces Attributes (CelebA) Dataset. Multimedia Laboratory, The Chinese University of Hong Kong. <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>
- De la Torre, Fernando Chu, Wen-Sheng Xiong, Xuehan Vicente, Francisco Ding, Xiaoyu Cohn, Jeffrey. (2015). IntraFace. 10.1109/FG.2015.7163082.
- Yan, Y., Lu, K. , Jian, X., Pengchengm G., Lyu,J. 2019. FEAFa: A Well-Annotated Dataset for Facial Expression Analysis and 3D Facial Animation. arXiv:1904.01509v1.
- Viola, P., Jones, M.J. 2001. Robust real-time face detection. *International Journal of Computer Vision* 57(2), 137–154.
- Lienhart, R., Maydt, J. 2002. An Extended Set of Haar-like Features for Rapid Object Detection. *Proceedings. International Conference on Image Processing*.
- Farfadi, S.S., Saberian, M.J., Li, L.J. 2015. Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. 643–650: New York, NY, USA.
- Cootes, T.F.; Edwards, G.J.; Taylor, C.J. 2001. Active appearance models. *IEEE Trans. Pattern Analy. Mach. Intell*(23), 681–685.
- Tian, Y., Kanade, T., Cohn, J.F. Facial expression analysis. *Handbook of Face Recognition*. Berlin: Springer; 2008.
- T. Baltrusaitis, A. Zadeh, Y. C. Lim, L. Morency (2018). OpenFace 2.0: Facial Behavior Analysis Toolkit. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)* (pp. 59–66).
- Sebastian Raschka. (2020). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning.
- Cohen’s Kappa. (2020, December 17). In *Wikipedia*. [https://en.wikipedia.org/wiki/Cohen%27s\\_kappa](https://en.wikipedia.org/wiki/Cohen%27s_kappa).
- De la Torre, F., Cohn, J.F. Facial expression analysis. In: Moeslund, T.B.B., Hilton, A., Kruger, V., Sigal, V. *Visual Analysis of Humans*. London: Springer; 2011. p. 377–409.
- Soukupová, T., Čech, J. (2016). Real-Time Eye Blink Detection using Facial Landmarks.