Facial Expression Recognition (FER) with FER+

# Introduction and Motivation

Facial Expression Recognition (FER) is a machine learning task that attempts to classify human emotions based on facial expressions in still images or video1. The ability for a computer system to determine a person’s emotion has countless applications. For example, FER has been used to detect neuropsychiatric disorders that effect the perception and expression of emotion2. In marketing it has been used to determine a shopper’s intent to buy items in their shopping cart3 and tourist travel satisfaction based on vacation travel4. Needless to say, it is a very hot topic of study due to its many uses.

In this project we use the FER-2013 dataset and FER+ labels to train several models that include the following:

* Support Vector Machine (SVM)
* eXtreme Gradient Boosted Tree (XGBoost)
* Convolutional Neural Network (CNN)
* Classifier-Conditioned Variational Autoencoder (CC-VAE)

Due to the challenging nature of the dataset, we utilized many techniques learned in class, including feature engineering and dimensionality reduction, over and under sampling, transfer learning, and more, in order to get the most out of the unbalanced dataset. Finally, after training an image classifier, we demonstrate how it can be utilized to do real-time emotion detection.

# Dataset and Processing

The following section describes our team’s selected dataset as well as the preprocessing performed and feature engineering.

## FER-2013 and FER+

The Facial Expression Recognition 2013 (FER-2013) dataset consists of 35,685 grayscale images of faces. Each image is labelled and can be one of seven different emotions, namely happiness, neutral, sadness, anger, surprise, disgust, and fear. The dataset was obtained from the Kaggle challenge that was created for the dataset5.



Figure 1: Sample FER-2013 Image

The dataset is partitioned into 3 parts:

1. Private Test: 3,589 images intended to be used to validate the model.
2. Public Test: 3,589 images intended to be used as the test set during training.
3. Training: 28,709 images intended to be used for training models.

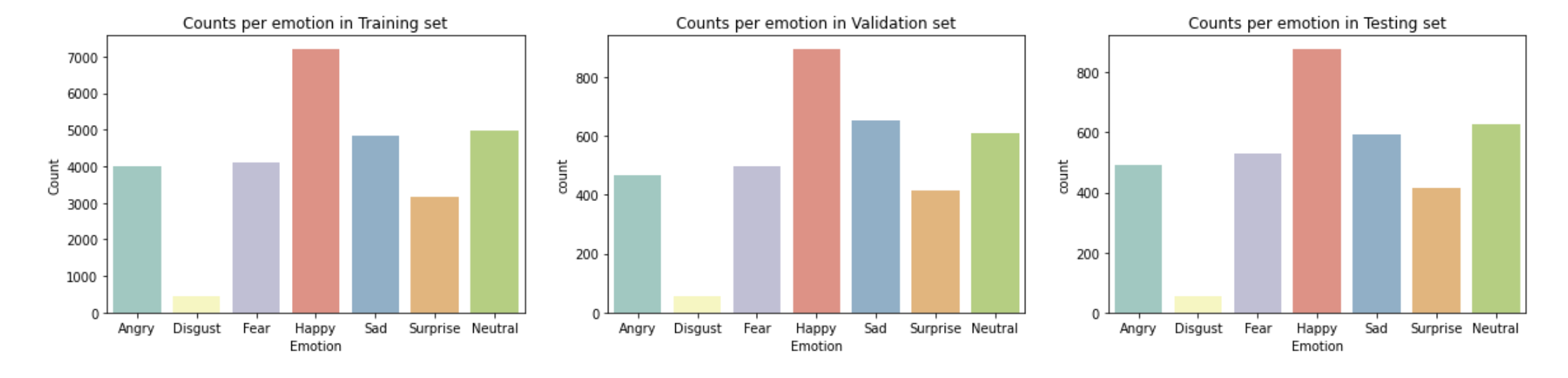


Figure 2: Training, Validation, and Testing Dataset Class Distribution

After our initial attempts to create a generalized classifier for the dataset lead to disappointing results we looked into our misclassification further. Upon inspection of misclassified images, we noticed many “True Labels” were egregiously incorrect in many cases based on our group’s judgement. After investigating further, our team stumbled upon the FER+ (aka FERPlus) dataset which was created by Google and generated by having 10 humans relabel each image6. The data contains the raw vote of each individual for each image in the original set. The labelers also added 3 additional classes, namely Contempt, Unknown, and NF which we dropped from the augmented set since they were not in the original FER-2103 dataset. Using the new labels that more correctly matched the images facial expression we proceeded to train and tune our models with far better results.

## Preprocessing

File Preparation

In order to ensure our group members all worked with a common dataset we created .npy files using common processing file we checked into GitLab for sharing. Using an inner join, we were able to bring the FER-2013 and FER+ sets together and then create a function to get the majority vote of the 10 votes per image to create a new single label for each image. We drop the newly added classes before doing the vote to ensure the undesirable classes do not make it into our final label set.

Under and Oversampling

Due to the unbalanced nature of the classes our team attempted to improve model performance by using *Under Sampling*, for overrepresented classes, and *Oversampling*, on underrepresented classes. We used a One-Sided Selection technique for under sampling which uses Tomek Links and a Condensed Nearest Neighbors (CNN) algorithm to remove datapoints7. For oversampling we utilized the Synthetic Minority Oversampling Technique (SMOTE) which generates new samples for underrepresented classes so we get a more balanced set8. Unfortunately, when applying each of these on their own and together we did not see an improvement of overall accuracy or precision on individual classes so the methods were discarded from all final models.

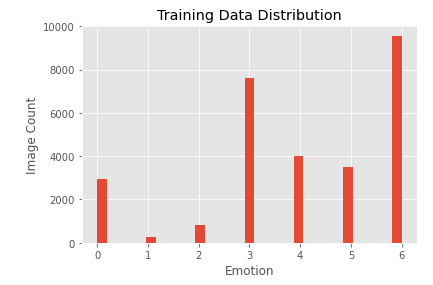
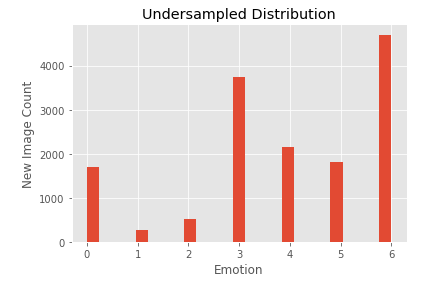
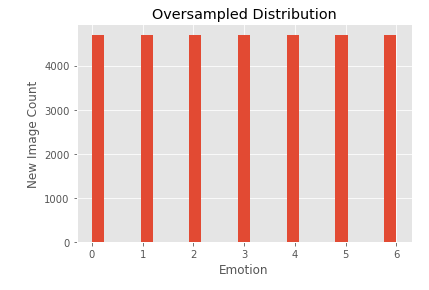
  

Figure 3: Original Distribution (left), Under Sampled (center), and Oversampled (right)

Pixel Unwrap

The SVM and XGBoost libraries do not natively support sending images to a model. This required the team to reformat the pixels for each image before sending to a model for training and predicting. In order to accomplish this, we simply reshaped the image pixels using the built in NumPy functions. We did not normalize the pixels by dividing by 255 since both SVM and XGBoost will utilize Principal Component Analysis (PCA) which already has normalized elements. We discuss our PCA tuning later in the paper.



Figure 4: Image Reshape for SVM and XGBoost

Image Normalization

Two of the models our team created can take actual images as opposed to arrays of feature data. To process this information, we simply normalized the pixels by dividing by 255. This is the only processing done on the images sent to both the VAE and CNN models.

## Feature Engineering

All facial expressions can be broken down into individual components of muscle movement called Action Units (AU)9. These are all described in the Facial Action Coding System (FACS). This motivated our team to try and engineer features from the images to capture key facial features, or landmarks, to help classify the emotions expressed in the facial AUs we detect.

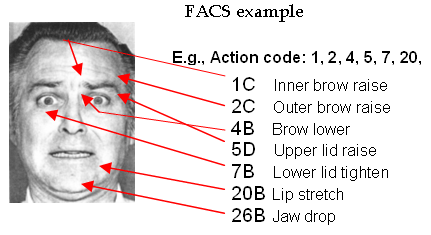


Figure 5: Facial Action Coding System (FACS) Example10

To enable out team to capture the facial landmarks we utilized the DLib library. To extract the landmarks, we first detect all faces in the image using the frontal face detector and then we get the landmarks using the shape predictor. The shape predictor uses a set of pre-trained weights and returns a group of 68 points for each face detected in an image which we will use as new features in our prediction models.



Figure 6: DLib Frontal Detector and Face Landmarks

In addition to the x and y values of each facial landmark we also compute the magnitude of each point as well as direction or bearing to the midpoint of the face11. This led to our final landmark feature set as the following.

## Dimensionality Reduction

After processing our input data for our SVM model and creating our new training features, our team decided use PCA to reduce the dimensionality of the reshaped pixel data and facial landmark features which are sent to the SVM model. All other model’s performance decreased when we eliminated features, but SVM in particular got better so we applied PCA to all data sent to this model. In order to determine how many components to use we ran a loop that considers 10-200 components and we select the one with the best performance. This was found to be 110 for the reshaped pixel data and 80 for facial landmark data.

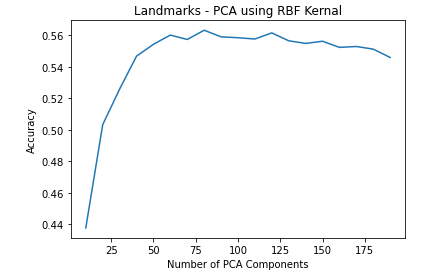
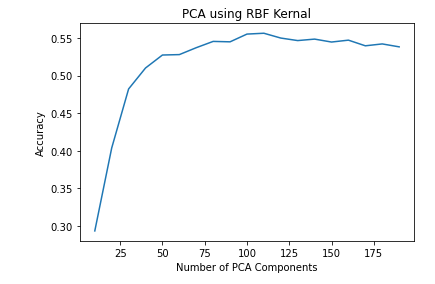


Figure 7: PCA Tuning Accuracy

The resulting eigenvectors that make up PCA components can be visualized in what we call eignefaces14. These eigenfaces can be combined to represent any face in our dataset similar to the eigenvalues in a non-image based problem.

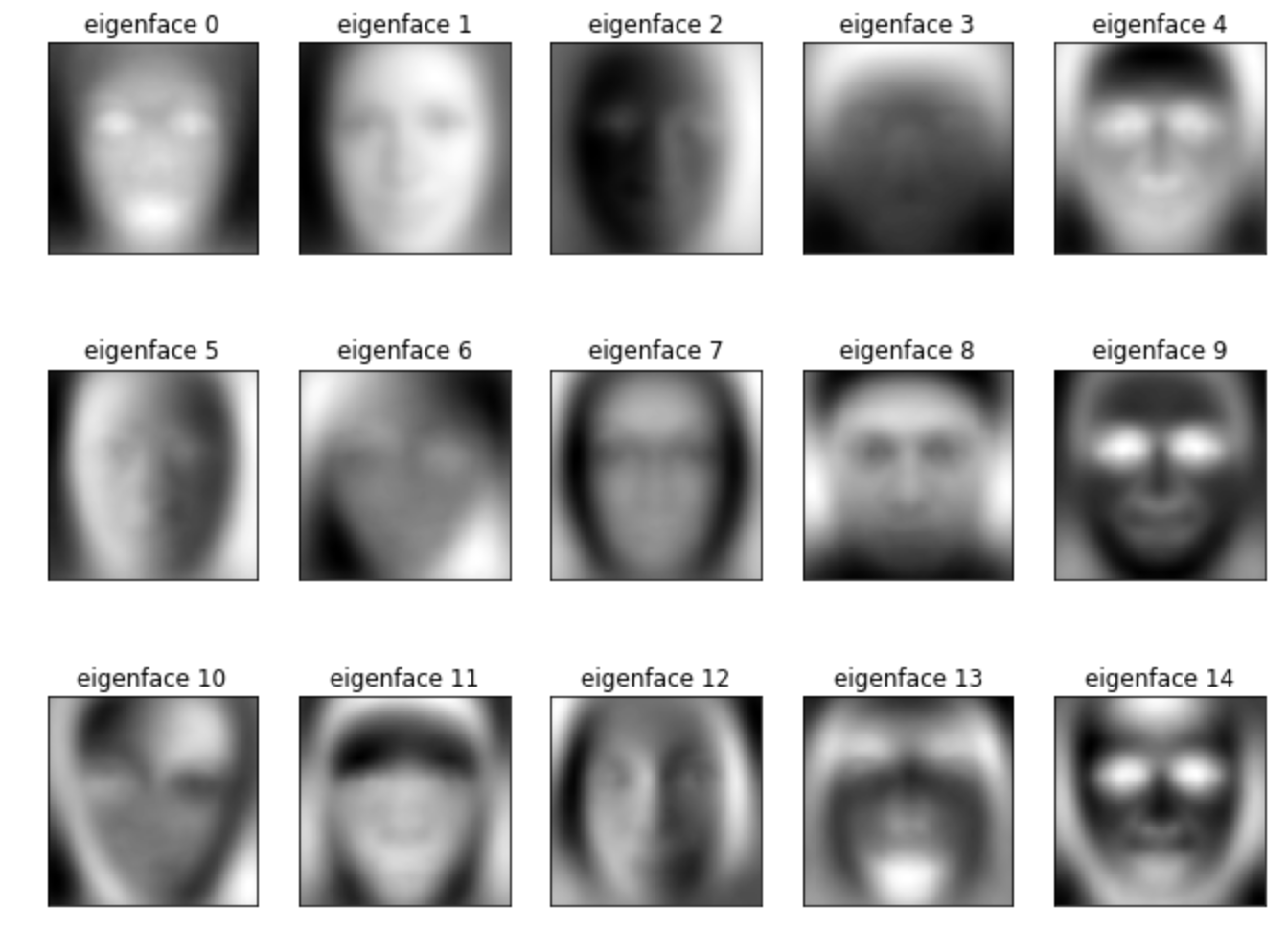


Figure 8: Sample Eigenfaces

# Models

The section below describes our different models as well as any tuning or performance enhancements that have been done beyond what has been discussed previously.

## SVM Models

As we learned in class a Support Vector Machine (SVM) can be a very effective classification model even in a complex multi class problem such as ours. We utilize it here to train two models, one on the pixel dataset and one using the landmark feature set. Our team considered two kernels for the SVM namely the Radial Basis Function (RBF) and the Linear. We ran our PCA tuning in parallel with kernel selection and found that the RBF kernel drastically outperformed the Linear one.

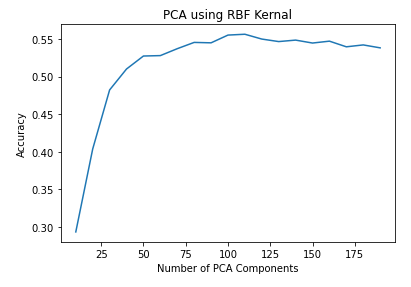
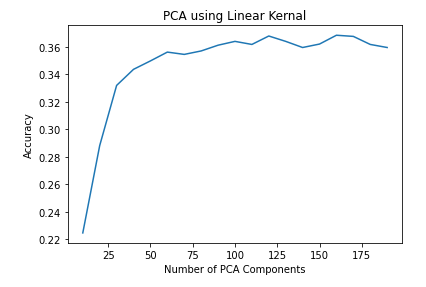


Figure 9: SVM Accuracy - RBF vs. Linear Kernel

Although under and oversampling did not work for our models, using the class\_weights argument set to ‘balanced’ did improve our performance. The balanced value will weight each class inversely related to its prevalence in the dataset12.

## XGBoost Models

eXtreme Gradient Boost (XGBoost) is an ensemble method which implements gradient boosting with decision trees. It is optimized for speed as opposed to AdaBoost which we learned in class. XGBoost is also more tolerant to including weak predicting features13. This is one reason we did not use PCA on data sent to XGBoost for training. Our team actually saw a decrease in accuracy when we eliminated features.

Not much tuning was needed for this model with the exception of selecting the proper objective and evaluation metric for our type of problem, which were multi:softprob and mlogloss respectively. Similar to SVM, we created two XGBoost models using the landmark features and pixel datasets.

## CNN

Convolutional Neural Networks (CNN) are known to be an effective type of model for creating image-based classifiers15. This led our team to explore CNN architectures in order to create a higher performing model for the FER dataset. We started with a basic block with two convolution layers with 32 filters, one batch normalization, one maximum pooling and one dropout layer. We then added the same block three more times and increased the number of filters in each convolutional block, in each subsequent block, to 64, 128, and 256. The convolutional layers are then followed by a flattening layer and one dense layer with 512 nodes before the final output layer. This led to a model with a total of 3.7 million trainable parameters.



Figure 10: CNN Architecture

Our team utilized the Adam optimizer with an initial learning rate of 0.001. We also used callbacks to reduce the learning rate by a factor of 0.2 if our selected metric, accuracy, does not improve for 3 rounds. We also used the early stopping callback which helps with training time as well as overfitting on the training set. Early stopping was set to stop training if our loss metric didn’t improve for any epoch. A summary of our training and test accuracy is summarized below.

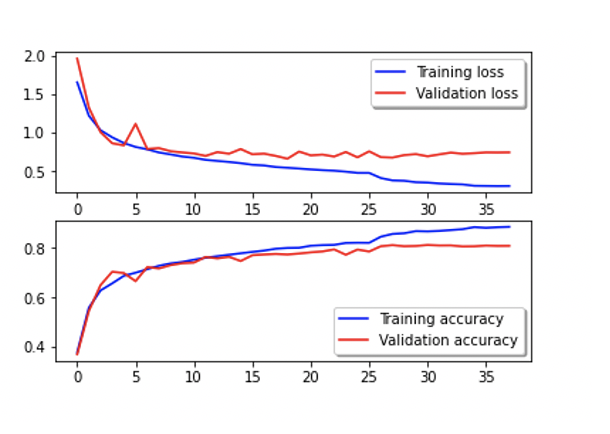
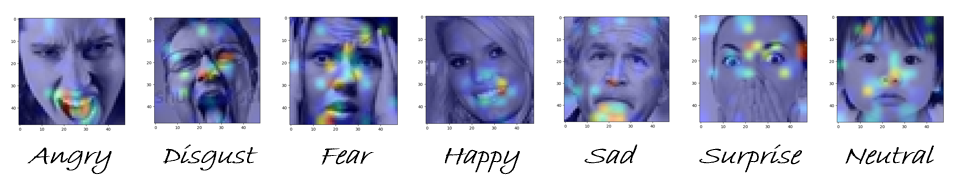


Figure 11: CNN Training and Test Accuracy

Edvin ADD GCAM Images and write up

Grad-CAM is a technique as an effort for model explainability. Typically, an input is passed through the last and trained convolution layer of a model – and the gradient flow though the layer is captured to produce heatmap overlays of the gradient and the input. In our project, we used this technique to visually verify where in the image the convolutional neural network was looking. For the top predicted classes such as angry, happy, and sad - the mouth feature is undeniable clear that is a strong feature to predict the class.

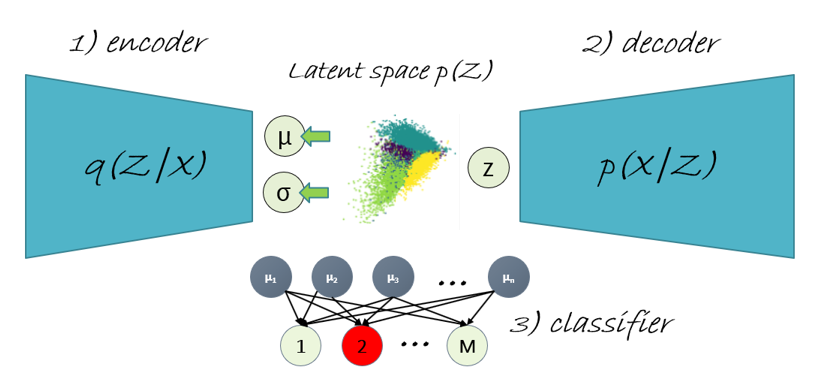


**Figure 1:** Grad-CAM employed on the last convolutional layer of the CNN

## Classifier Conditioned Variational Autoencoder (CC-VAE)

Variational Autoencoders are a neural network architecture typically classified as generative models. The typical components of the network are the encoder, and the decoder. In this specific modification, a third classifier component is added taking inputs from the latent space; hence, the name classifier conditioned VAE. These models are built on two very important statistical methods called Variation Inference and Representation Learning.

Variational inference is a technique used when we encounter an intractable distribution i.e.: an integral that we cannot compute. The core concept of variational inference is to turn an inference problem into an optimization problem. Suppose we are given an intractable distribution , then the technique will optimize over a well-behaved class of distributions in order to find a distribution most similar to.



Let  be the distribution of encoder inputs, and  the distribution of the latent variables. Therefore, the integral denominator of Bayes Rule as shown below cannot be computed

|  |
| --- |
|  |

To solve this problem, we select a function which is the encoder neural network given its parameters to approximate the function  as shown in the figure above. Furthermore, we select a prior to be a Normal distribution then during optimization we attempt to minimize the difference between  and . This is the second term in Eq. (x) below.

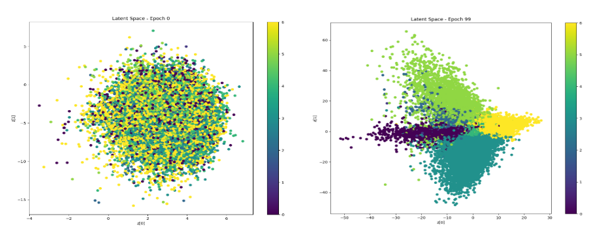
|  |  |
| --- | --- |
|  | (x) |

Equation (x) represents the total loss of the Classifier-Conditioned VAE but only term one and two are found in the vanilla VAE. The reconstruction loss is simply the difference between the encoder input and decoder output, and the third term is the categorical-entropy loss between the predicted class labels and the true labels.

In application, the expert must choose a latent space dimensionality. For example, in our project we experimented with a Multivariate normal distribution of 1, 3, and ten dimensions. The output of the decoder is a n-dimensional vector of the mean, variance, and variate z which is computed in one dimension as  where . In the literature this is know as the reparameterization trick and allows the network to be trained with back-propagation.

|  |
| --- |
|  |

One of advantages of Gaussian modeling is that it allows each class to push itself further apart from other classes. However, this is not always achievable given the complexity of the data set and the similarity of the features that the network is learning. The purpose of the additional classifier is to add extra stress on the network and separate the cluster since its only inputs are the means of the Gaussian features. Furthermore, by tuning the weights and  – further emphasizes the importance of disentangled classes. The result for the 2-dimensional case is shown in the figure below.



**Figure 2**: Latent space after epoch 0 (left) and latent space after epoch 100 (right) Each color represents a class while similar classes are also located closer to each other.

After training the classifier may be used exclusively to classify new latent variables – or coupled with a simple clustering algorithm such as K-Nearest Neighbors to squeeze a bit more performance out of the system. The dimensions in the VAE are in concept similar to principal components where the first few components are the most important. Therefore, each consecutive additional dimension will not return the same performance as previous.

Variation Autoencoder extremely exiting models and topic in continued research for interpretable machine learning algorithms.

## Voting Ensemble aka MegaModel

One final attempt at increasing performance was to combine the final models into a larger voting ensemble our team affectionately calls *MEGAMODEL*. We found that if we used our five weakest models in a hard voting ensemble, we could actually increase performance beyond that of any one base model. We had to implement our own soft voting and hard voting functions since the built-in Sci-Kit Learn functions assume the data is the same going into each model. The team also found that adding the CNN model to the ensembles led to less accuracy that using the CNN alone so it was omitted from the final voting model.

# Results

After training our models and using them to predict the labels on the Private Test set, we observed that the CVAE and CNN out performed all other models. We evaluated our models by looking at the confusion matrix for each as well as the classification report and overall accuracies. These results are summarized below.

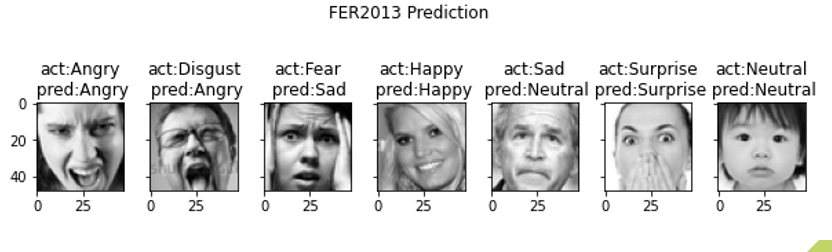


Figure x: Sample Prediction

## Overall Accuracy

The table below summarizes the overall accuracy of each model ran against the private test dataset.

|  |  |
| --- | --- |
| Model | Accuracy |
| SVM with Pixels | 55.6% |
| SVM with Landmarks | 56.3% |
| XGBoost with Pixels | 59.4% |
| XGBoost with Landmarks | 57.4% |
| CNN with Images | 78.9% |
| CVAE with Images |  |

Table 1: Overall Model Accuracies

## Confusion Matrices

|  |  |  |
| --- | --- | --- |
|  |  |  |
| SVM with Pixels | SVM with Landmarks | XGBoost with Pixels |
|  |  | Edvin put image here |
| XGBoost with Landmarks | CNN with Images | CVAE with Images |

## Classification Reports

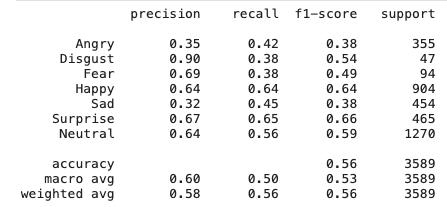


Figure xxx: SVM with Pixels Classification Report

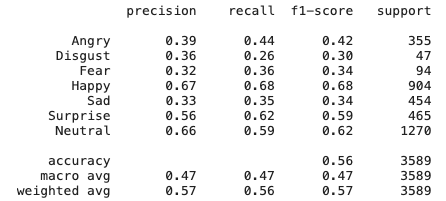


Figure xxx: SVM with Landmarks Classification Report

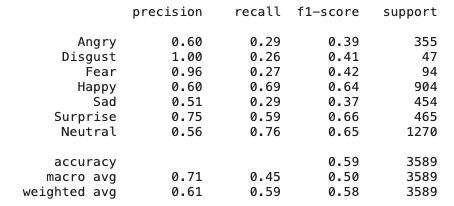


Figure xxx: XGBoost with Pixels Classification Report

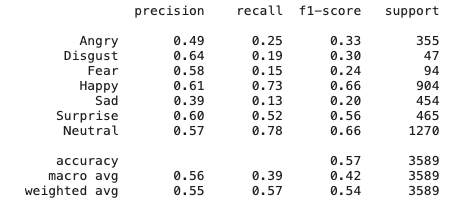


Figure xxx: XGBoost with Landmarks Classification Report

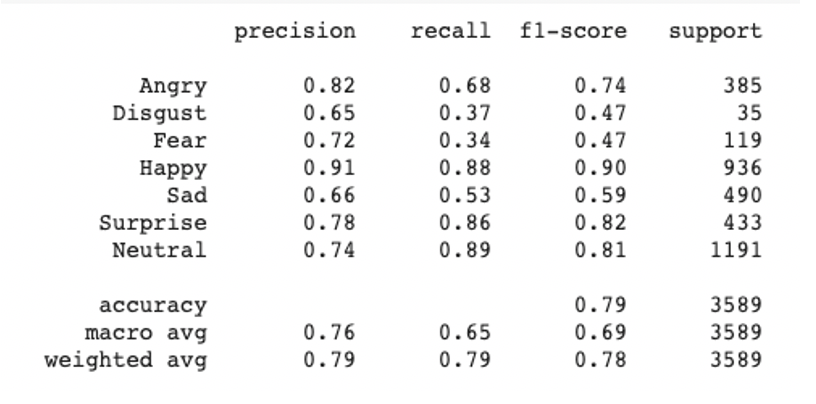


Figure xxx: CNN Classification Report

Edvin put image here

Figure xxx: CVAE Classification Report

# References

1. <https://algorithmia.com/blog/introduction-to-emotion-recognition#:~:text=Facial%20emotion%20recognition%20is%20the,can%20recognize%20emotions%20as%20well>.
2. <https://www.med.upenn.edu/sbia/facial-expressions.html#:~:text=Facial%20Expression%20Analysis%20in%20Neuropsychiatric,and%20psychopathology%20of%20an%20individual.&text=We%20are%20using%20high%20dimensional%20shape%20transformations%20to%20relate%20different%20expression%20states>.

#### **Computerized Sales Assistants: The Application of Computer Technology to Measure Consumer Interest – A Conceptual Framework**

1. <https://www.sciencedirect.com/science/article/abs/pii/S0736585320300630>
2. <https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data>
3. <https://github.com/microsoft/FERPlus/blob/master/README.md>
4. <https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/>
5. <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>
6. <https://www.paulekman.com/facial-action-coding-system/>
7. <https://inc.ucsd.edu/mplab/grants/project1/research/face-detection.html>
8. Facial Expression Recognition using Facial Landmarks: A novel approach
9. <https://scikit-learn.org/dev/glossary.html#term-class-weight>
10. <https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f#:~:text=Compared%20to%20random%20forests%20and,being%20significantly%20slower%20than%20XGBoost>
11. <https://en.wikipedia.org/wiki/Eigenface#:~:text=An%20eigenface%20(%2F%CB%88a%C9%AA%C9%A1,Alex%20Pentland%20in%20face%20classification>.
12. <https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks>

# File List