

## **Team 1 Final Report FER**

Jeff Fossi, Malcolm Nichols, Samson Bezu, Cameron Lauf

Maryville University

DSCI 598: Capstone Project

Dr. Robbie Beane

December 06, 2022

## Abstract

In this paper, we apply deep learning and convolutional neural networks (CNN) to the problem of facial expression recognition. We attempt to predict which one of seven common facial expressions is being expressed using a large set of images. Some of the models included in our analysis include custom CNNs, transfer learning models, and ensemble models using a combination of both. We were able to produce an ensemble model that approached 70% accuracy, while our Meta-Model results produced an accuracy of 68%. These results suggest that deep learning can be an effective application for the problem of facial expression recognition.

## Table of Contents

<b>Abstract(Cameron)</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>Summary Report (Jeff)</b>	<b>4</b>
<b>EDA Notebook (Malcolm)</b>	<b>133</b>
<b>Training Notebooks (ALL)</b>	<b>177</b>
<b>Lesson learned Discord discussions with Dr.Beane.</b>	<b>20</b>
<b>Future Insights and Improvements</b>	<b>22</b>
<b>References</b>	<b>233</b>

## Summary Report (Jeff)

Thousands of years of gradually accelerating progress is leading toward a point in time when a computer can provide greater overall intelligence than a human (Kurzweil, 2019). Team 1 sets how to answer the question: can AI interpret images of facial expressions better than humans? Human performance on this dataset is estimated to be 65.5% (Khairuddin & Chen, n.d.).

This project builds upon the Kaggle competition *Challenges in Representation Learning: Facial Expression Recognition Challenge* originally started in 2013 (Dumitru et al., 2013). In this competition we build image classification models to identify the emotion being expressed in images of human faces. Some important characteristics of this competition are described below.

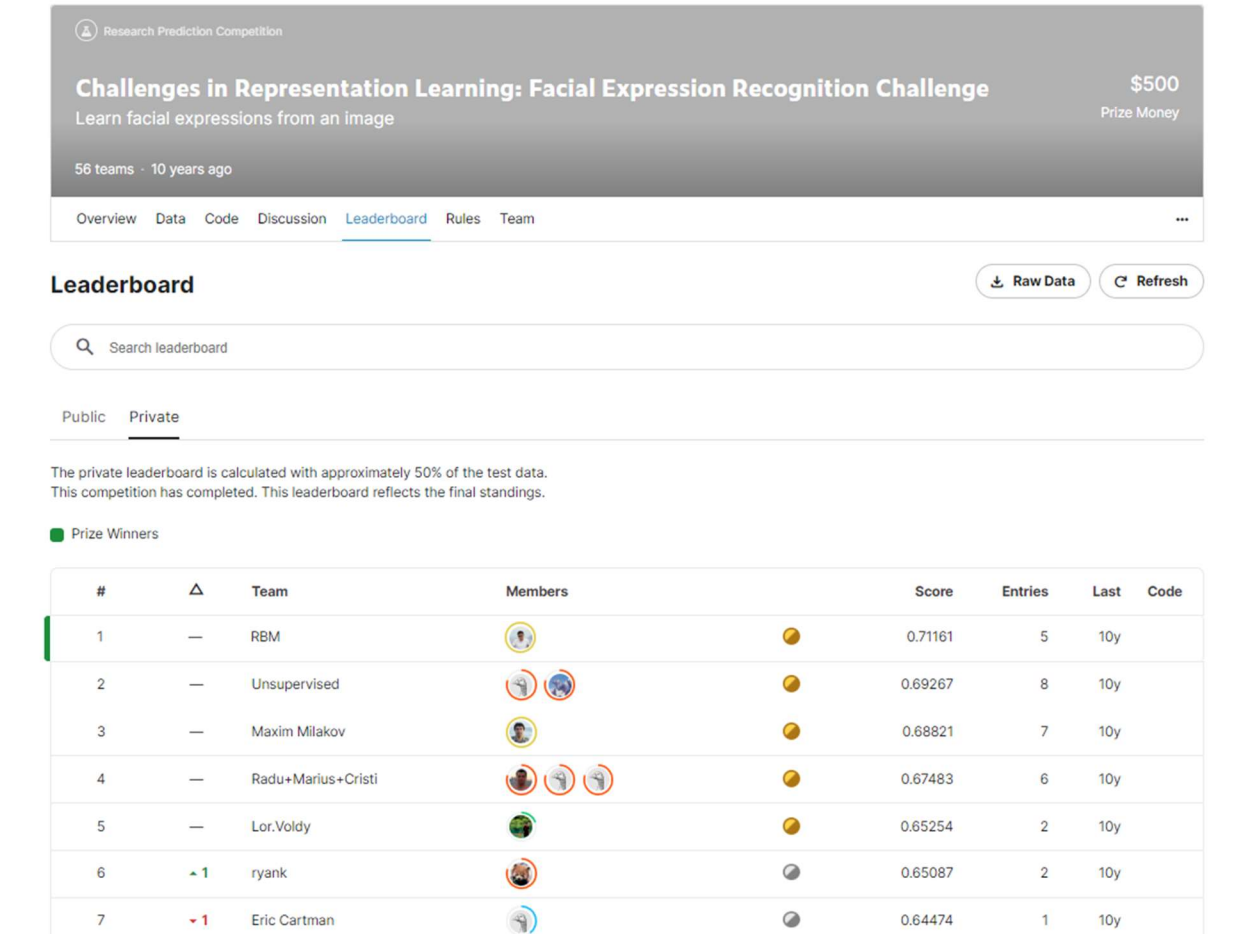
The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string are space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.

The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.

This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project. They have graciously provided the workshop organizers with a preliminary version of their dataset to use for this contest (Dumitru et al., 2013).

We further describe a basic summary of the data set with some visualizations created in several of our models. A screenshot of the leaderboard in the competition is below.



Research Prediction Competition

### Challenges in Representation Learning: Facial Expression Recognition Challenge

Learn facial expressions from an image

\$500 Prize Money

56 teams · 10 years ago

Overview Data Code Discussion **Leaderboard** Rules Team

**Leaderboard** [Raw Data](#) [Refresh](#)

Search leaderboard

Public **Private**

The private leaderboard is calculated with approximately 50% of the test data. This competition has completed. This leaderboard reflects the final standings.

■ Prize Winners

#	Δ	Team	Members	Score	Entries	Last	Code
1	—	RBM		0.71161	5	10y	
2	—	Unsupervised		0.69267	8	10y	
3	—	Maxim Milakov		0.68821	7	10y	
4	—	Radu+Marius+Cristi		0.67483	6	10y	
5	—	Lor.Voldy		0.65254	2	10y	
6	▲ 1	ryank		0.65087	2	10y	
7	▼ 1	Eric Cartman		0.64474	1	10y	

We include links our notebooks created over the period of 5 weeks that contain more complete exploratory analysis and visualizations below:

<https://www.kaggle.com/code/camlauf98/cl-week3-facial-expression>

<https://www.kaggle.com/code/jefffossi/week3-fossi-custom-cnn-v2>

<https://www.kaggle.com/code/jefffossi/week3-fossi-pretrained-vgg16-v1>

<https://www.kaggle.com/code/jefffossi/65-with-conf-matrix-and-cr>

<https://www.kaggle.com/code/malcolmnichols/mn-fer-eda>

<https://www.kaggle.com/code/samsonbezu/week4-fer-tl-and-augmentation-fossi>

<https://www.kaggle.com/code/malcolmnichols/fer-v02>

<https://www.kaggle.com/code/malcolmnichols/fer-ensemble>

<https://www.kaggle.com/code/jefffossi/week-4-icml>

<https://www.kaggle.com/code/malcolmnichols/fer-v04>

<https://www.kaggle.com/code/jefffossi/week-5-fossi-683>

<https://www.kaggle.com/code/malcolmnichols/fer-transfer-learning-v02>

<https://www.kaggle.com/code/malcolmnichols/fer-v06>

<https://www.kaggle.com/code/jefffossi/xception-fossi>

<https://www.kaggle.com/code/jefffossi/week-7-wk3cnn-xception-models>

<https://www.kaggle.com/code/malcolmnichols/fer-meta-model>

[Week4\\_FER\\_TL and Augmentation\\_Fossi | Kaggle](#)

[FER with VGG 16 and VGG 19 | Kaggle](#)

<https://www.kaggle.com/code/camlauf98/cl-week5-resnet50>

We faced many technical challenges in working on this project. A summary of these challenges follows, as well as a summary of any new tools or techniques we learned to address those challenges. The first of these challenges was managing the many notebooks and versions we created. Secondly, we had issues with GPU power needed to analyze the models was limited each week. Finally, trying to beat our previous accuracy numbers week over week.

We managed the challenge of different versions of notebooks by posting links to our discord forum regarding the project. GPU usage was split out amongst the different team members to get the most of our allotment for the week. Also not performing a run all sequence during save helps to conserve GPU power. Some members were able to run code in coordination

with CNN models to initiate TPU acceleration adding an extra 20 hours per week. Finally, we were able to see small increments of steady improvement over 5 weeks despite being very minute. Nonetheless, our final models achieved .70 accuracy which exceeded our expectations.

This is a summary of some of the model architectures we tried, including brief descriptions of any prominent models we feel like mentioning. Our first submission during week three was a symbol custom CNN which scored .65. We began experimenting with a pretrained model VGG16 but did not see any improvement, so it was left out. The link to this model is below.

<https://www.kaggle.com/code/camlauf98/cl-week3-facial-expression>

Our week 4 submission notebook is below. This model includes the implementation of image augmentation, more transfer learning techniques, early stopping and ReduceLROnPlateau. We were able to increase our accuracy score to .66 with this submission.

<https://www.kaggle.com/code/malcolmnichols/group1-facial-expression-submission>

Our Week 5 submission we started delving into the idea of ensemble techniques. Our submission is below. This ensemble was the Average () technique that scored .71. However, we suspected this technique was showing data leakage due to the varying methods to create the individual models. This was resulting in abnormally high accuracy levels. This is when we began exploring the stacking technique to combine individual model predictions into one tabular dataset.

<https://www.kaggle.com/jefffossi/week5-team1-ensemble/edit>

Our week 6 consisted of good accuracy scores well above .70. However, we learned these scores were artificially inflated due to errors in how the individual models were trained. We were not using the same split mix, and even mixing up some of the training data. This all resulted in inaccurate ensemble performance. Examples of these notebooks are below.

<https://www.kaggle.com/jefffossi/week6-team1-ensemble-fossi/edit>

<https://www.kaggle.com/jefffossi/84-week5-team1-ensemble/edit>

Our week 7 submission was two submissions. One being the corrected bagging model using MaxVote and the Stacking ensemble. The bagging ensemble was corrected in that the individual models were trained on the same split of training data, eliminating data leakage. This week we also saw an improvement of the stacking ensemble towards .69 accuracy, where the bagging model was at .70. It was deemed the stacking model was a more solid model due to the tie breaking issue with the MaxVal method.

<https://www.kaggle.com/code/jefffossi/fer-maxvote-v01>

<https://www.kaggle.com/code/malcolmnichols/fer-meta-model>

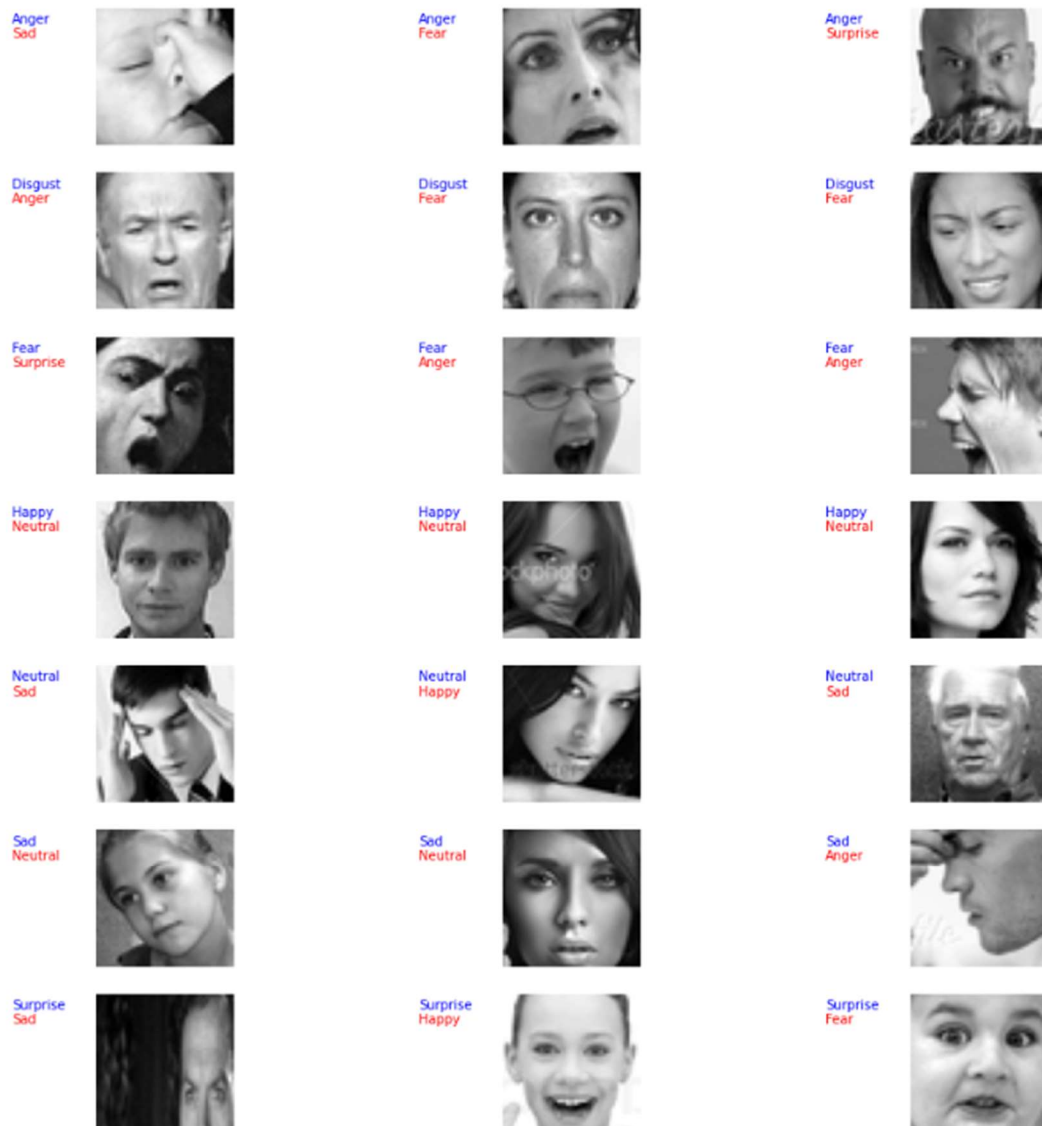
This is a description of our final model and a summary of its performance. Towards the end of our project, we were working on two separate ensemble techniques. One of these methods was a bagging technique, the other a stacking technique. Two separate bagging ensembles were tried, one being an average() technique (youtube) and the other being a MaxVote or ArgMax technique. The Average technique was determined to be unsatisfactory due to a low accuracy score of .65 when compared to the separate models. The MaxVote technique made it as a finalist due to an accuracy score of .70. However, this model does not take into account lower numbers classes getting a disproportionate number of wins. Future research will include a more



sophisticated model which includes looking at the total probabilities for each of the tied classes to break the ties.

The stacking technique evolved out of the deficiencies of the ensemble modeling techniques described above. The benefit of the stacking technique over the other techniques for this dataset are as follows. First, it allows for more advanced methods of aggregating the data over bagging techniques. With bagging we need to rely on averages where every model gets the same influence as the other. The stacking technique allowed us to use more advanced rules such as Decision Tree, Random Forest, Extra Forest, and Gradient Boosted Trees. Future research will use an automated hyperparameter tuning framework Optuna. And LightGBM. Towards the end of this project the stacking meta model also approached .70 accuracy.

To conclude, despite dozens of model training notebooks in which most accuracy scores were between .65 and .67 using ensemble methods we were able to obtain accuracy levels at or near .70. In the context of the Kaggle contest this would position us in 2nd place. These are not bad results. In the end we were able to prove deep learning models could at the very least match human levels of identifying facial expressions. Some of the confusing mismatched labels provided by our models are provided below. We would argue even humans would have difficulty classifying some of these images. The correct label is provided in blue font, the predicted label is in red.



Below are the links to our final ensemble submissions:

<https://www.kaggle.com/code/malcolmnichols/fer-meta-model>

<https://www.kaggle.com/code/jefffossi/fer-maxvote-v01>

Below Is a summary and classification report for both of the ensemble models.

Fer-Meta-Model:

Total Wrong Predictions: 2305

	precision	recall	f1-score	support
0	0.60	0.60	0.60	958
1	0.81	0.45	0.58	111
2	0.55	0.42	0.48	1024
3	0.87	0.89	0.88	1774
4	0.54	0.60	0.57	1247
5	0.79	0.80	0.79	831
6	0.63	0.67	0.65	1233
accuracy			0.68	7178
macro avg	0.68	0.63	0.65	7178
weighted avg	0.68	0.68	0.68	7178

MaxVote Model:

	precision	recall	f1-score	support
Anger	0.58	0.68	0.63	491
Disgust	0.86	0.65	0.74	55
Fear	0.59	0.47	0.52	528
Happy	0.87	0.90	0.89	879
Neutral	0.66	0.72	0.69	626
Sad	0.56	0.53	0.55	594
Surprise	0.81	0.78	0.79	416
accuracy			0.70	3589
macro avg	0.70	0.68	0.69	3589
weighted avg	0.70	0.70	0.69	3589

We can see from the precision and recall numbers from the above classification reports, the model performs best when classifying the Happy and Surprise emotion. This is good considering the happy emotion is the most frequent in our dataset. We can also see from the support numbers, the MaxVote model uses only the PublicTest dataset, while the Meta model uses both the public and private dataset. While the final accuracies are close, it is interesting to see the differences in precision and recall based on the testing data used. For example, we see a much higher precision and recall for the Sad emotion with the Meta model. This makes sense since as sample size increases, the precision should also increase, and larger sample sizes lead to more statistically significant results.

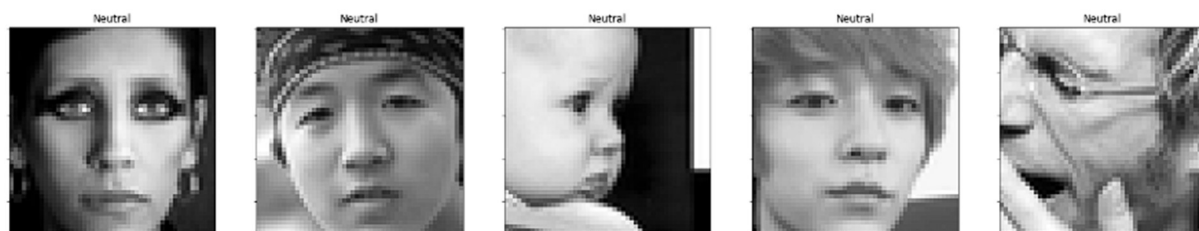
## EDA Notebook (Malcolm)

This Exploratory Data Analysis EDA notebook is a thorough look into what kinds of images we are going to be training and testing our models from. We first started by loading the training data. We decided to load the training data because this is what the model is going to see and train on. So, we will need to understand and look at the images that the models are going to be looking at. What you can see below is a general print out of some labels and their corresponding images. We must know the difference between Happy, Sad, Angry, Fear, Neutral, Surprise, and Disgust. We took a random sample out of the data to produce these images using a for loop, matplotlib for visualizations, and subplots to plot on a grid.



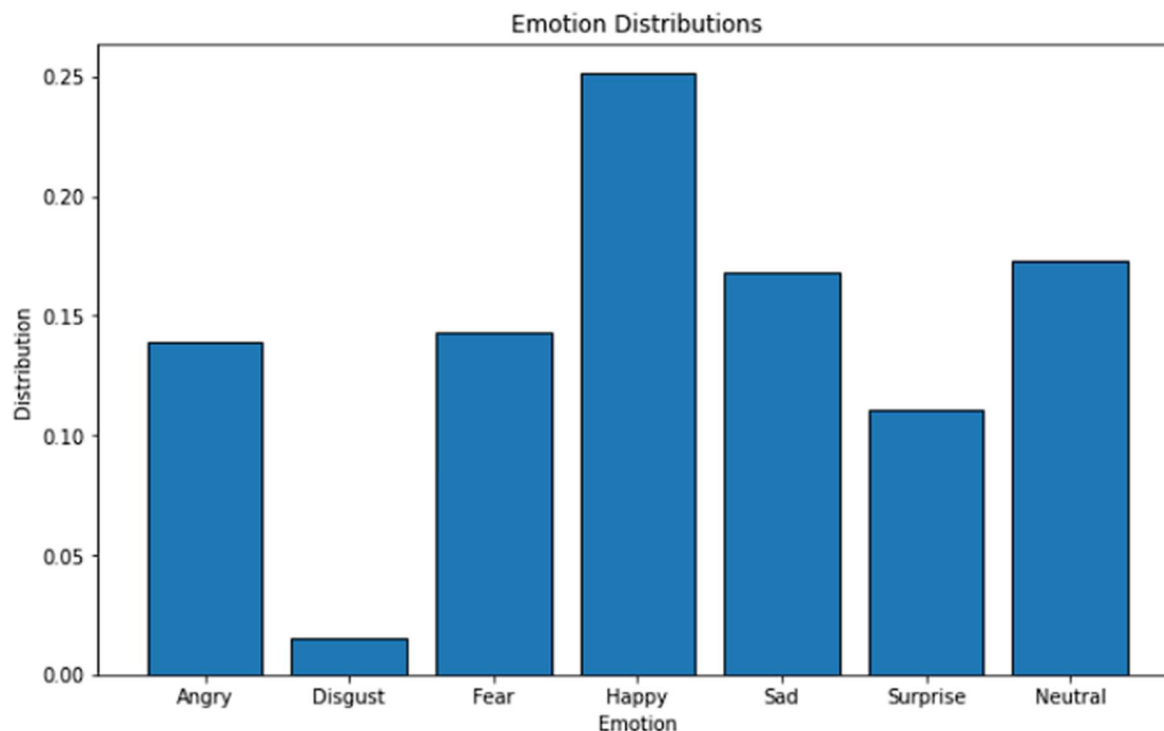
From this image we can get a look at the grayscale images we are working with. From here we can make some guesses as to why the model may or may not misclassify. For instance, in the last row of the second column, we get a cartoon picture. So, we can assume that there are multiple of these images in this dataset. This can be tricky for the model, just like a regular human if we are looking at a cartoon face of say sad, but it is a neutral face. We see this in the final row, final column image where the guy is sad. At first glance we can mistake that for a neutral face. Naturally with this thinking we knew that our models were going to have a hard time performing

high accuracies and needed to think of many ways to overcome this. We also decided to print examples of each label exclusively. Below we will include some examples of that.



Just like with the first image, we get a glance of the images and their labels. We can make a confident claim that some of these images could be labeled incorrectly. Meaning whoever created this dataset, could potentially misclassify some of these images as well. So that told us that the nature of this dataset is a bit difficult. Our professor also mentioned the difficulty of the dataset and claimed that our models would perform under a certain range by themselves for their accuracy (between 60-66%). Once we heard that we were not surprised by our accuracy and the focus was then placed on how to improve with little increments. Next, we wanted to see the

distributions between the labels. We were surprised to find that we are also working with unbalanced data. Meaning there is not an equal representation of the labels.



As shown above, we can clearly see that Disgust contributes little data in this dataset. While happy represents the majority of the data. We can then make a confident statement that we are dealing with imbalanced data and disgust may be mislabeled because the model does not have much data to train that label on. While we are sure that it will not have a problem classifying the happy label because it has a lot of data to train on. From this EDA we can understand the nature of the data, how it was constructed, and how it may affect our models. We got a excellent glimpse that told us everything we needed to know to move forward.



## Training Notebooks (ALL)

**Here we can list some of the individual notebooks we worked on. Not the Ensembles.**

**Does not matter if they performed well....**

<https://www.kaggle.com/code/malcolmnichols/fer-v06> : This notebook dealt in particular with the unbalancing of the data. We decided to try out oversampling/undersampling. The main purpose of this model was to level out the distribution for the labels. Oversampling takes the ‘minority’ labels and adds copies of those images to balance out the data. It is a new technique we have not used before, and I am sure as time progresses and we learn more about it we can optimize it better. I decided to omit this from the final ensemble because it did not do well but it was great to learn overall.

<https://www.kaggle.com/code/malcolmnichols/mn-facial-expression-recognition-v01> : The first model I worked on. It was the introduction to this dataset, and I was wanting to put something out there to understand what we are working with. This model did not perform well, but it was a good indicator for me to pursue understanding the data better and learning how to better optimize/create models.

<https://www.kaggle.com/code/malcolmnichols/fer-size-increase-v01> : This notebook I was experimenting with increasing the size of the images. Because we are working with 48x48 images I wanted to try to see the performance in increasing the size. I have also used some image augmentation like in my other notebooks. I increased the size using `tf.image.resize` from 48x48 to 192x192. Overall, I am sure it can prove very useful in future implementations, but as time was running short I only did a basic setup.

Week 5 Fossi .683 v1 was ran in the 2020 environment of Kaggle notebook.

<https://www.kaggle.com/code/jefffossi/week-5-fossi-683>

Then I tried to recreate in the latest environment in the following notebook.

<https://www.kaggle.com/code/jefffossi/fer-v04>

Here I tried experimenting with some older models. These models included an oversampling method and a smoothing method increasing pixel size to 224x224x3 (rgb) from a model I found from 2020. I could not recreate the results from this week 5 fossi .683 version when I incorporated it into our latest model. It is my understanding that VGGFace(model='resnet50') is outdated so I could not get it to run in our latest environment model. I did try using just resnet50 and I could not even approach .68 so I tried a few other pretrained models with similar dismal results. So, I then reverted back to our custom CNN model we submitted previously and I got crazy high val\_accuracy. This was a good example of overfitting and different versions of notebooks not translating well for use later.

### [FER with VGG 16 and VGG 19 | Kaggle](#)

This notebook is to experiment with the VGG models. VGG models are a type of CNN Architecture proposed by Karen Simonyan & Andrew Zisserman of Visual Geometry Group (VGG), Oxford University, which brought remarkable results for the ImageNet Challenge. The two most popular models are VGG16 and VGG19. VGG models are trained on 3 channels.

However, the dataset we are working with is 1 channel. We overcome this challenge using the NumPy repeat method to replicate the pixel values of the grayscale image. Overall, the validation accuracy was lower than the custom-built CNN models. This result can be an indication that we can perform better by training a model from scratch on our dataset rather than using transfer learning.

<https://www.kaggle.com/code/camlauf98/cl-week5-resnet50>

This notebook was created to test the ResNet-50 pre-trained deep neural network. ResNet-50 is a convolutional neural network that is 50 layers deep and has been successfully applied to many image classification problems. The first training run consistently produced validation accuracy scores of about 0.85. Although, when this was applied to the test dataset for our competition, the accuracy dropped significantly to about 0.65, indicating that this model could be overfitting to the training data. The inspiration for testing this model on our data comes from the successful application in this paper written by a group of students at Stanford University.

[http://cs230.stanford.edu/projects\\_winter\\_2020/reports/32610274.pdf](http://cs230.stanford.edu/projects_winter_2020/reports/32610274.pdf)[http://cs230.stanford.edu/projects\\_winter\\_2020/reports/32610274.pdf](http://cs230.stanford.edu/projects_winter_2020/reports/32610274.pdf)

## Lesson learned Discord discussions with Dr.Beane.

The following are the critical points discussed during our team conversation with Dr. Beane.

- Shuffle parameter within the flow () method
- Early stopping
- Bagging approach steps to create ensembles
- Stacking approach steps to create ensembles
- Comparison between the Stacking and Bagging approach
- Why do we use 50/50 data split for the Stacking approach?

### **Shuffle parameter within the flow () method :**

When we ensemble the models for our final submission, we used the flow() method. The flow()method has a shuffle parameter with a default value of True, which means that in each new Epoch, the values are shuffled before being separated into batches. This is good for the training dataset but not necessary for the validation set. To avoid error, we set shuffle=False in theflow()method for the validation set.

### **Early stopping :**

Early stopping is to train efficiently on the training dataset but to stop training when performance on a validation dataset starts to decrease. We can use the loss on a validation dataset as the metric to monitor or the model's accuracy. Once we set our monitoring metrics, we choose a trigger to stop the training process. Training can be stopped when the model's performance on the validation datasets starts to decrease compared to the performance on the validation dataset at the prior training epoch (e.g., an increase in loss).

### **Bagging approach:**

It is a model ensembling method that involves training models separately and then creating predictions according to all the models. The predictions from the various models are aggregated together in some way (such as taking averages, argmax, etc).

### **Steps for Bagging approach:**

- 1) Split the data into training/validation as 50/50.
- 2) Train our models, each using the same training set created in the 50/50 split. Then we will use the validation set to evaluate the models and decide which to keep.
- 3)Select our models to keep. Let's say we will keep "N" number of models.
- 4) Generate training, validation, and test predictions for each model and save them to a CSV file.
- 5) Load all of the prediction files into a notebook. Thus we will have 3\*N of these (3 for each of the N models).

6) For each dataset type (training, validation, test), we will combine the N files into 3D dataset with shape `((num_observations, number of class, num_models))` and then perform the aggregation.

7) Generate training/validation/test predictions based on the aggregated predictions.

8) Finally, we will calculate the training, validation, and test scores for the ensemble.

### **Stacking approach:**

It is a model ensembling method that combines the predictions into one dataset, then is treated as another tabular dataset. Then we will train new models of this prediction dataset. The new "meta" model can be of any type (Logistic Regression, Decision Tree, Random Forest, Neural Network, etc.)

### **Steps for Stacking approach:**

- 1) Split the data 50/50, using the same seed.
- 2) I train the models, each using the same training set created in the 50/50 split.
- 3) I select models to keep. Let's say we will keep "N" number of models.
- 4) Generate training, validation, and test predictions for each model, and save them to a CSV file.
- 5) Load all the prediction files into a notebook. Thus, we will have 3\*N of these (3 for each of the N models).
- 6) For each type of set (training, validation, test), we will combine the N files into one 2D dataset with the shape: `((num_observations, num of class*num_models))`.
- 7) Pick ML models such as logistic regression or decision tree and train the output of several models on the training2 set. Then use cross-validation for scoring models and for performing hyperparameter tuning. Let's say the final model is the "meta model".
- 8) Finally, generate test predictions by combining the test predictions from each of the base models into one dataset (discussed above). Then run that dataset through our meta model.

### **Comparison between Stacking and Bagging**

- Stacking is more advanced. It allows for more flexible ways of aggregating the data and moreover, allows for these to be learned from the data rather than selected by a human.
- With bagging, you combine things into a 3D dataset with shape `((num_observations, num of class, num_models))`. but with Stacking you will combine into a 2D dataset with shape: `((num_observations, num of class*num_models))`.

### **Why do we use 50/50 data split for the Stacking approach?**

- Because we have two rounds of training: one for all the base models, and one for the meta model.

## Future Insights

### **Future Insight/Improvements:**

We do believe that we can make improvements to our model and increase the performance. There are so many ways we did not try to implement and/or explore more of. When the semester is over there is a plan to continue with this project or move on to a similar dataset in order to further our knowledge in this scope. Because it is a massively studied subject and used in our everyday life. It makes sense that it grabs the attention of those seeking some challenge. I do want to improve on the ensemble model as best I can and will share the results as time progresses. Overall, this project has brought great challenges and knowledge. Many resources available online and projects that use different methods to try and solve this with the most efficient way possible.

## References

- Dumitru, Goodfellow, I., & Bengio, Y. (2013). *Challenges in Representation Learning: Facial Expression Recognition Challenge*. Kaggle. Retrieved December 6, 2022, from <https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/>
- Khairuddin, Y., & Chen, Z. (n.d.). Facial Emotion Recognition: State of the Art Performance on FER2013. *Dept. of Electrical and Computer Engineering, Boston University*. Retrieved 12 6, 2022, from <https://arxiv.org/abs/2105.03588>
- Kurzweil, R. (2019). *Danielle: Chronicles of a Superheroine*. WordFire Press LLC.