Technical Review Document

Team Name: Thing 1 & Thing 2
Liam Youm
Kevin Nguyen

Model Description

Our goal is to identify and distinguish between 6 different facial expressions from the FER-2013 dataset from Kaggle: anger, fear, happiness, neutrality, sadness, and surprise. We used the CNN model architecture for our project because it's the architecture best suited to understanding images and derive meaning from them.

We chose to use 4 convolutional layers, with the first layer having 16 filters, the second layer having 32 filters, the third layer having 64 filters and the fourth layer having 128 filters because we know the heuristic that filters tend to increase as layers increase. All the kernel and padding sizes are 3 and 1 respectively on every layer.

We started to set up the number of convolutional layers from 2 which is relatively simple, and then we set up 3 layers to make the model learn more complex patterns. We noticed that the model performance is not improving anymore with other hyperparameters on 3 convolutional layers and we ended up setting up the 4 convolutional layer architecture. We didn't have time to try making the architecture deeper since we had an issue using GPUs to train our model which caused a lack of time to experiment more.

We opted for a batch size of 32 for efficient training, over 30 epochs. We chose 30 epochs because our testing showed that the model's accuracy was still improving after 10 and 20 epochs, and performance increases mostly flattened after 30 epochs.

We set up dropout with a 0.2 rate in the model initialization but didn't use it when we trained the model because it caused model accuracy to decrease. It would've been better if we could use GPUs for training so that we could do more experiments with this hyperparameter.

We used batch normalization after every convolutional layer to make the training stable, and also to avoid overfitting.

Our fully connected layers included 256 neurons in the first layer, 128 in the second, and 6 in the final layer to match the number of classes in the dataset. We used the Adam optimizer with a learning rate of 0.001 because it worked well for our dataset, and cross-entropy loss because it is the standard for multi-class classification tasks like ours.

We used ReLU as our activation function, as it was the main function taught to us and the general standard.

We also used image transformation methods to increase model robustness, different methods on the training set, and test set including a validation set. In the training transformation, we flipped images horizontally with 50% probability, rotated them up to 30 degrees, resized them to 128*128 size, made sure the images were grayscale, converted them to tensor, and normalized them. In the test set, we utilized the same methods we used in the training set except for flipping and rotating.

Dataset Processing & Analysis

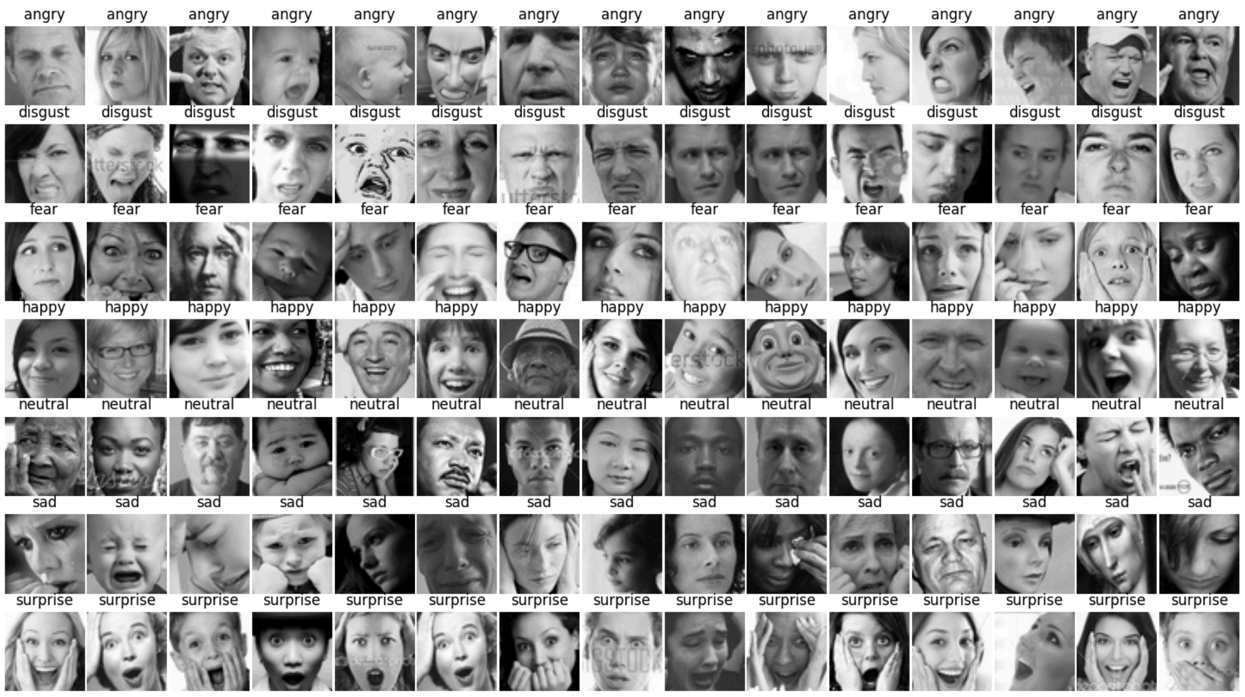
The dataset files are already fairly organized, and as a result, not much cleaning had to be done, at least when it came to adding the folders into our GitHub repository. Though, we initially noticed that there were too many images in the `happy` folder when compared to the other classes: `angry`, `disgust`, `fear`, `neutral`, `sad`, and `surprise`. `happy` had over 8,000 images, while the other classes had about 5,000 on average. As a result, we decided to cut down on `happy` images by about 3,000, with the goal of preventing biases towards happy facial expressions and to provide fair training and testing.

We then decided to merge the `angry` and `disgust` folders together for a couple of reasons. Firstly, the `disgust` folder did not have many samples, even less than the average 5,000 images of the other classes. Secondly, the facial expressions in `disgust` all looked similar to the facial expressions in `angry`. We also have observed neural network models on the Internet struggling to differentiate between angry and disgusted faces, likely for the reason that was just mentioned. Lastly, anger as an emotion is not far off from disgust. All of these reasons rationalized our decision to merge the folders together.

When adding the images to the repository and observing their shape, we noticed that each image had 3 color channels. This is not what was expected, as every image in the FER-2013 dataset is grayscale. As a result, we used `v2.Grayscale(1)` in our transformation variables to correctly make the images 1 channel.

From the FER-2013 dataset page: "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 training set consists of 28,709 examples and the public test set consists of 3,589 examples." The image sizes and face locations were justified when viewing their shape when converted to tensors and viewing the images directly. With about 25,000 samples after cleaning, this is more than enough images for our model to train on, top of the now 6 distinct classes.

We can display 100 example images to show their diversity and verify their qualities as explained by the dataset page:



Observing the images, we can see that disgust facial expressions are hard to distinguish from angry facial expressions, justifying their merging. Also, the fact that the dataset images are grayscale should benefit our model during training, as it simplifies the training so that the model no longer needs to factor in unique color values when finding the facial expressions, and instead only needs to account for the image brightness.

We then performed data augmentation on our images using `torchvision.transforms`. We believed that rotation and flipping augmentations would make our model the most robust. This is because we believe that our model should be able to still process and predict a person's expression any way they tilt their head, which would mean rotation. Flipping, though likely not a significant augmentation, as facial expressions tend to not vary much when flipped a certain axis, helped our model detect a more diverse set of facial expressions, as it had to account for the facial expressions it trained on *and* their flipped versions. Other augmentations, such as cropping, color, and resizing, seemed unviable for one reason or another: CNNs should not be expected to predict from cropped faces, as they may not have enough key info to make a correct prediction; color is irrelevant when the image is converted into grayscale; and resizing seems unnecessary, as we don't plan on training our CNN on images with varying sizes. Lastly, we made a validation dataset for hyperparameter tuning.

Model Results

We set a goal as 3 goals:

- Bare minimum accuracy: 25%, when the classifier predicts all the output as a class of the largest class in the data set.
- Initial accuracy: 40%, 15% improvement from the previous

- Target accuracy: 60%, 20% improvement from the previous, our actual goal

All the losses in every phases were scattered and fluctuating which means model's learning is not stable. It converges to lower losses but still unstable. Nonetheless, We could achieve our actual goal with this model. Below is the result.
Test Accuracy: approximately 60.3%
Test loss: 1.09
Train accuracy is approximately 66.9% which is a little higher than the test accuracy. It may refer the model is a little overfitted by the train dataset, but the difference is not huge. If we continue to tune the other hyperparameters, I look forward to see even higher accuracy with this model.

Reflection

Starting our project was swift yet messy at first, as we were unaware of each others' working styles. When creating our project proposal document, we performed well in communicating what roles we wanted in each milestone and what sections we would complete in the document. We completed all the required work during our meeting and had no significant problems. The main issue, however, was the distribution between work for the week; Kevin was not able to do as much work as Liam, Kevin mainly set up the GitHub repository and some files, while Liam worked on a majority of the project proposal document and made code to graph our dataset. This would be noted by both members and work distribution would swiftly even out in later weeks.

Our data processing and analysis were much better in terms of work distribution and general collaboration. We both had clear goals and completed them sometime before and during our weekly meeting. We communicated what we wanted to clean in our model and came to a consensus swiftly. The time it took between converting the data to tensors in the main code and creating the code to print 100 sample images was about the same, if not leaning more toward the 100 images. However, this was balanced by one person doing more portions of the milestone document than the other, so ultimately the work was even. There was good and fast communication between what transformations we wanted to add to our model, and feedback on what and why each transformation would be effective. We then discussed possible improvements we could try to our model, which we would later incorporate in future weeks.

Our model creation also went smoothly and well. Each person created a good amount of code and did their part in the milestone document. Throughout the milestones, we tended to follow a general pattern of Kevin laying the foundation for the code and document, while Liam spent his time–sometimes outside of our meeting time–refining and adding new code for the next week and completing parts of the milestone document that required completion of the code to answer. It made for an effective system, as both people could contribute equally in their ways to the project despite the different academic levels between us (freshman and senior). For this portion of the final, this system was especially prominent and was effective in letting each person contribute their thoughts and work.

For model refinement, the majority of the work distribution was towards Liam. The reasoning is that, firstly, we believed that it would be a bit difficult or even tedious to have both people set up the same programs and packages on their respective devices and attempt to collaborate. Secondly, the work was balanced by Kevin completing more of the other objectives such as the writing and presentation portions. Ultimately, Kevin took more of a "backseat" role in model refinement, giving feedback and new ideas to what Liam would then implement. Although at first it may seem like an uneven distribution between work, we believe that overall it was balanced by other portions where someone does the majority of the work. However, next time, we still believe that there should be more collaboration between both members of the code. Otherwise, everything still went well.

Our results analysis went pretty well. We were able to identify what went well and what went wrong in our model and adjust the model accordingly and correctly. We were both able to share an even amount of work between analyzing the data and giving our thoughts on it. There isn't much else to speak about otherwise.

Our presentation slides, as previously mentioned, had the majority of the work distribution go to Kevin, with Liam going over the slides and refining mainly the parts about the model architecture and results. Liam made up for the lack of work on the presentation by focusing more on the model refinement. Otherwise, the process of creating the slides was smooth and was created well before the deadline. Though we didn't practice together, we left speaker notes and practiced our portions in our own time.