Name: Aymen Ouali
ID: 261051347

Date: November 18th 2022

Project: Manga Character Mood Classifier

Dataset: https://www.kaggle.com/datasets/mertkkl/manga-facial-expressions

1. **Final Training Results**

Since the last deliverable, I have made a few changes, the first one being adding data to the dataset. Initially, my dataset was composed of 462 images, but after taking screenshots of characters from multiple series, I bumped it up to 680 images. I also changed my pre-processing method. Initially, all I did was resize and center crop the images to fit the pretrained model's original training data, but using a transformation method found here, I sharpened the images and adjusted the contrast and gamma. This resulted in a slight increase in accuracy (going from around 55% to around 62.5%).

After these tests, I concluded that my method to measure accuracy was not ideal, which is why I am now using a confusion matrix, which I then use to compute the precision and recall using torchmetrics. Using this method to measure accuracy, I got a precision and recall of 0.40, which is less than the accuracy I measured using my former method of just computing right predictions over the total number of images.

```
tensor([[1, 0, 0, 2, 0, 0, 0],
        [0, 0, 2, 2, 1, 0, 1],
        [2, 0, 2, 0, 0, 0, 0],
        [0, 0, 0, 5, 1, 0, 0],
        [0, 0, 1, 1, 3, 0, 1],
        [0, 1, 1, 0, 2, 0, 0],
        [1, 0, 0, 1, 1, 0, 3]])
```

Figure 1: Confusion matrix for the model
Classes in order: angry, crying, embarrassed, happy, pleased, sad, shock

Then, I decided to use data augmentation. To do that, I created 5 additional datasets within my code (horizontally flipped, rotated to the left, rotated to the right, horizontal flip + left rotation, horizontal flip + right rotation). Doing this increased the dataset's size to 4080 images. After

training for 50 epochs and using completely new images (the same images used for the non-augmented model), the model achieved a precision recall of 0.4571.

```
tensor([[1, 0, 0, 1, 0, 0, 1],
        [0, 2, 1, 3, 0, 0, 0],
        [1, 0, 2, 0, 0, 0, 1],
        [0, 1, 0, 3, 1, 0, 1],
        [2, 0, 0, 1, 3, 0, 0],
        [0, 0, 0, 0, 1, 3, 0],
        [1, 0, 0, 1, 2, 0, 2]])
```

Figure 2: Confusion matrix for the data augmented model
Classes in order: angry, crying, embarrassed, happy, pleased, sad, shock

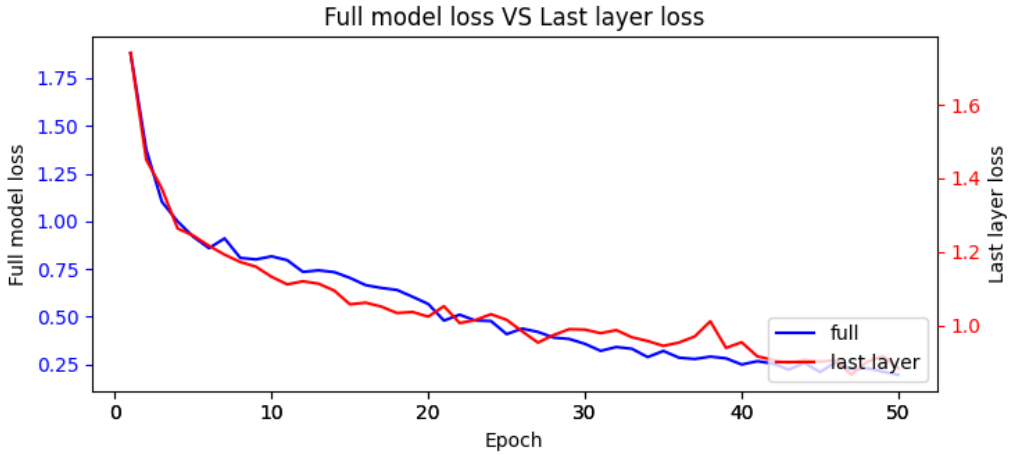

Figure 3: Loss for the last layer VS Loss for the full model

2. Final demonstration proposal

For my demonstration, I am planning on making a website where there will be a canvas the user will draw on. Upon drawing on the canvas and clicking on a button, the website would display the model's prediction for the character's mood. To build this, I will use React for the front end since I am familiar with it (personal projects and internship), and Flask (personal projects) for the back end to make the predictions using the model. To deploy the React front end, I would simply use Netlify, which I've used multiple times to deploy simple projects. The only thing I need is to link it to my GitHub repo and it deploys when I push to the main branch. To deploy the Flask back end, I would use Heroku. However, I faced memory problems in the past using Heroku when I tried to use a computationally expensive program, which makes me doubt the feasibility of the deployment.

I will try using ONNX.js if it ends up being simpler, since it allows me to use my model's script directly on the browser, which would make it the project easier to build and deploy. I do not have any experience with it, which is why I will watch this tutorial to learn about it.