Name: Aymen Ouali
ID: 261051347

Date: November 4th 2022

Project: Manga Character Mood Classifier

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

1. **Problem statement**

   I chose a dataset containing 462 images (faces of characters) and 7 different classes (emotions) to make a deep learning model that can predict the mood of a character or a drawing. My goal is to make a website where the user would input an image or a drawing, and ideally where they could draw themselves and the model would predict while they draw.

2. **Data Preprocessing**

   My dataset contains 462 images of manga characters, and they are classified in 7 different classes which correspond to the moods I want my model to predict. The preprocessing method I will use for now is the preprocessing I found on the PyTorch documentation for ResNet, which basically consists of resizing the images to fit the pretrained model's original training data.

3. **Machine learning model**

   For my project, I will be using the ResNet-50 model. I chose to use a pretrained model instead of making my own model since the ResNet-50 has 50 convolutional neural networks already and it would be difficult to make my own model from scratch with the limited knowledge I currently have. It is also often used for image classification, which is my objective.

   a. I will mainly be using PyTorch to make my project, since I am already familiar with it through my MAIS Hacks project. My model is a pretrained ResNet-50, but the last fully connected layer must be trained separately before training the entire model because I have a specific number of classes (7). ResNet-50 is composed of 48 Convolution layers, 1 MaxPool and 1 Average Pool layer.
   b. I split my data into three parts: a training data set, consisting of 80% of the images, a set of images to be displayed in a grid (to show the effects of the preprocessing) and a testing dataset. For my hyper-parameters, I just set them to 0.001 for the learning rate and the weight-decay for now.
   c. To test my model's accuracy, I calculated the total number of correct guesses over the total amount of data for both the training and testing datasets. My model seems to be overfitting since the accuracy for the training dataset is much higher (~20%) than the accuracy for the testing dataset.

d. My biggest challenge initially was trying to understand the PyTorch documentation, looking at forums and people's code to understand how the model worked and how to preprocess the data. By looking at multiple resources, I ended up understanding that most things that people did were following some sort of recipe/convention, which helped me understand it better (for example, I was confused about why I had to train the last layer separately but found out that it was to finetune the model, so it had the right output size, which is 7 in my case). Another challenge, which is not solved yet, is going to be the testing accuracy, which is still quite low (around 50%).

4. As mentioned in Deliverable 1, the loss function I used is Cross-Entropy loss, since I want to get probabilities between 0 and 1 to classify the images. I also used the same method for accuracy that I mentioned in Deliverable 1, which is counting the number of images that were correctly classified out of all the images.

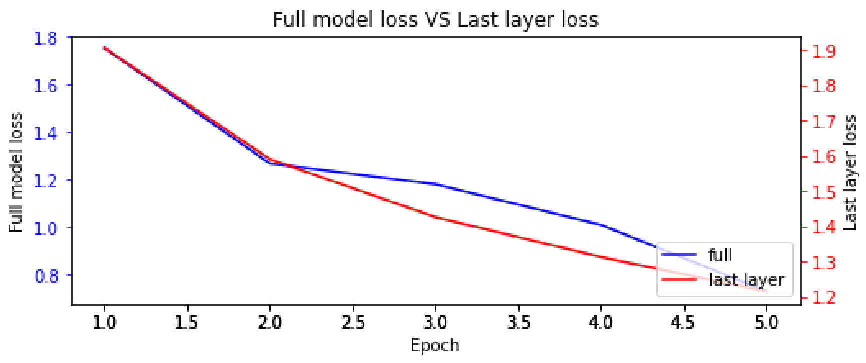

Figure 1: Training loss for full model and training loss for the last layer (by epoch)

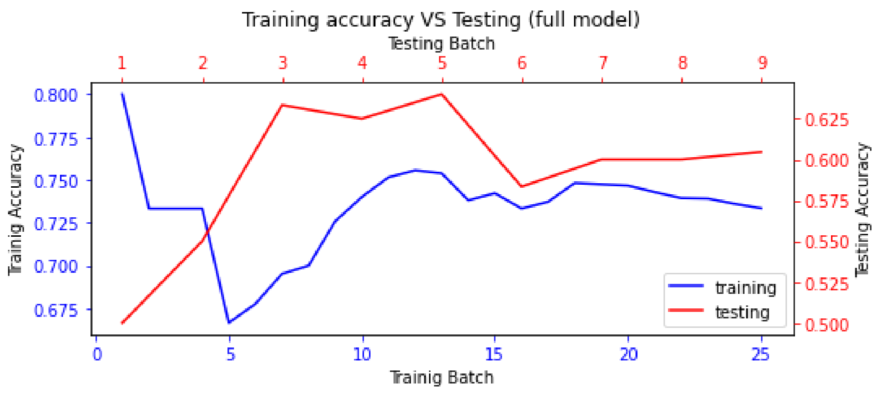

Figure 2: Training accuracy and testing accuracy (by batch)

On Figure 1 is the loss when training the last layer only and the loss for training the full model. Both losses decrease over the epochs (only five for now because training takes a long time) and do not seem to stop decreasing after five epochs. As seen on figure 2, the training accuracy for the full model is around 75% and the testing accuracy is around 57%. Right now, I am not sure about the way I have tested my accuracy (only testing the accuracy after training instead of testing at each epoch) and I am still unsure about the metric itself. However, this accuracy manages to display the feasibility of the project. Also, this accuracy seems to be higher than the accuracy achieved on the project I got inspiration from on Kaggle.

5. My next steps will be:
   - **a.** Confirm the metric I use for calculating the model’s accuracy and adjust it to make it more reliable (look at precision-recall, confusion matrix).
   - **b.** Play with hyperparameters, image preprocessing (sharpening the images, etc.) to increase the model’s accuracy.
   - **c.** Figure out if having more data would be helpful to increase the accuracy.