COMP449 HW2 Report

Dong Shu
Ding Zhang
Yuchen Wang
Chen Si

1. (5.0points)Implement and train your autoencoder on the subset of the Emoji dataset that you selected and augmented:

a. Describe your dataset and the steps that you used to create it:

- The dataset is called ‘valhalla/emoji-dataset’. It is a dataset that related to emojis
- Steps:
    a. We load the dataset. We filter out the emoji that is not related to ‘face’ or ‘superheros’
    b. We split the dataset into validation sets, training sets and test sets.
    c. We use data augmentation to expand these sets to 600, 200, and 200 images.
    d. Additionally, we resize the images to a resolution of 64 x 64 x 4 (RGBA) and organize all the data into a Torch Dataset object.

b. Provide a summary of your architecture(see Adversarial Examples Notebook)

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 224, 224]           3,232
              ReLU-2         [-1, 32, 224, 224]               0
         MaxPool2d-3         [-1, 32, 112, 112]               0
            Conv2d-4         [-1, 64, 112, 112]          51,264
              ReLU-5         [-1, 64, 112, 112]               0
         MaxPool2d-6           [-1, 64, 56, 56]               0
            Conv2d-7          [-1, 128, 56, 56]         204,928
              ReLU-8          [-1, 128, 56, 56]               0
         MaxPool2d-9          [-1, 128, 28, 28]               0
           Conv2d-10          [-1, 256, 28, 28]         819,456
             ReLU-11          [-1, 256, 28, 28]               0
        MaxPool2d-12          [-1, 256, 14, 14]               0
  ConvTranspose2d-13          [-1, 128, 14, 14]         819,328
             ReLU-14          [-1, 128, 14, 14]               0
         Upsample-15          [-1, 128, 28, 28]               0
  ConvTranspose2d-16           [-1, 64, 28, 28]         204,864
             ReLU-17           [-1, 64, 28, 28]               0
         Upsample-18           [-1, 64, 56, 56]               0
  ConvTranspose2d-19           [-1, 32, 56, 56]          51,232
             ReLU-20           [-1, 32, 56, 56]               0
         Upsample-21         [-1, 32, 112, 112]               0
  ConvTranspose2d-22          [-1, 4, 112, 112]           3,204
         Upsample-23          [-1, 4, 224, 224]               0
          Sigmoid-24          [-1, 4, 224, 224]               0
================================================================
Total params: 2,157,508
Trainable params: 2,157,508
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.77
Forward/backward pass size (MB): 63.16
Params size (MB): 8.23
Estimated Total Size (MB): 72.16
----------------------------------------------------------------
```

c. Discuss and explain your design choices,

- **For the encoder:** the feature channel number increases from 4 to 256. This is designed to learn complex features extracted. 4 is for R(red),G(green),B(blue) and A(alpha).

  We also design the dimensionality reduction by implementing max pooling and strided convolutions. This is to reduce the spatial dimensions and the computational complexity.

- **For the decoder:** The feature channel number decreases from 256 to 4. This is designed to make the encoder and decoder be symmetric. This design helps in reconstructing the original input.

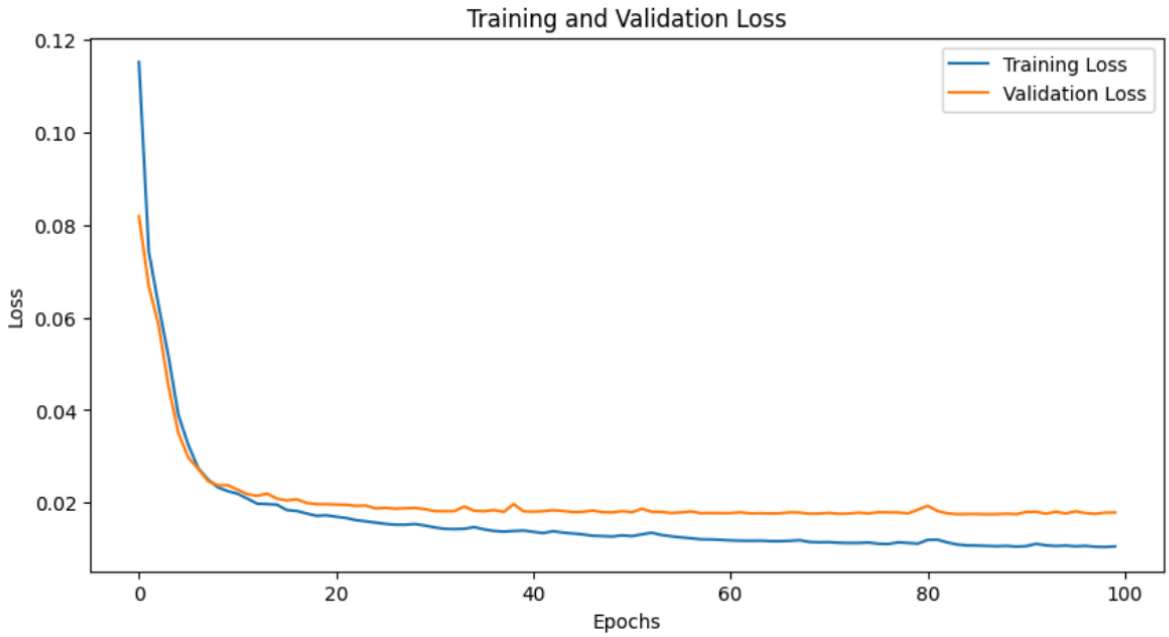
  The design of implementing convolutional transpose layers are to reconstruct the original input.

  The final activation function is used in the last layer to constrain the output values to the range[0,1].

d. List hyper-parameters used in model:

- Learning rate = 0.001
- Number of epochs = 100
- Batch size=32

e. Plot learning curves for training and validation loss as a function of training epochs,



f. Provide the final average error of your autoencoder on your test set, and

- 0.0175

g. Discuss any decisions or observations that you find relevant.

- In the plot graph, we can see that validation loss stops decreasing around round 15. However, training loss is still gradually decreasing until round 70

2. (5.0 points) Separate your dataset into two or more classes using Emoji descriptions and assign labels. Repeat Step 1 adding image classification as an auxiliary task to MSE with a λ of your choosing. You can choose any classification technique.

a. describe how you separated your dataset into classes

- We manually separate the dataset into human faces and non-human faces by detecting the key word from the label (*e.g. horse, zebra, etc*). If the image label contains the key word, then it will be considered as a non-human face with label 1, otherwise, 0.

b. describe your classification technique and hyper-parameters

- batch size: 20. We choose 20 because it can be divided by training size 600, and validation size 200. We found out that if we use another batch size that can not be divided by both sizes, it will raise an error of: "ValueError: Expected input batch_size (XXX) to match target batch_size (XXX)". Learning rate: 0.001, Lambda value = 0.5, and number of epochs = 30.

c. plot learning curves for training and validation loss for MSE and classification accuracy

d. discuss how incorporating classification as an auxiliary tasks impacts the performance of your autoencoder

- As we have seen in part c, the loss curve is very unstable and in general the performance is a lot worse when adding classification as an auxiliary task compared to part 1. The validation loss converges to 0.02 in part 1, while the validation loss in part 2 is very high.

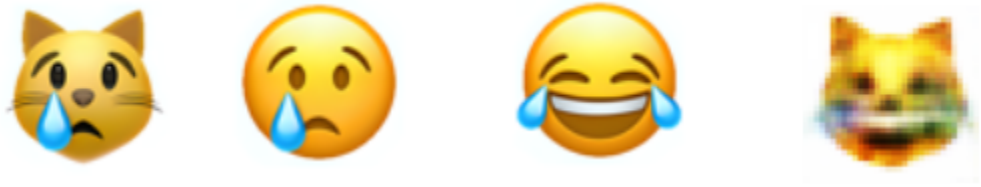
e. speculate why performance changed and recommend (but do not implement) an experiment to confirm or reject your speculation.

- The reason that the performance decreased significantly may be due to several reasons. First of all, the two main objectives are conflicting with each other: the primary objective of an autoencoder is to map the input data into an embedding space that can be used to reconstruct the input as accurately as possible by the decoder. The classification task, however, requires the model to learn the features of the input emojis and categorize them into different categories. These objectives are not aligned with each other, leading the autoencoder struggling to optimize both tasks simultaneously. In addition, the features learned by the autoencoder for reconstruction purposes might not be the ideal ones for classification. This discrepancy may lead to potential weaknesses in the performance.
- One potential solution is simply to treat the problems separately. We can first train the autoencoder on the reconstruction task alone to learn a good embedding space and latent representations of the emojis. We can then fine-tune and train the model on the classification task. Another solution, without separating the tasks, is to use different weights for reconstruction loss and classification loss. We can penalize one to the other to find a balance that allows the model to improve at both tasks simultaneously.

3. (5.0 points) Select an attribute from the Emoji dataset (internal or external to your selected subset) to compose with any image from your selected subset. Use vector arithmetic on latent representations to generate a composite image that expresses the attribute. For example, I chose to add the glasses from "nerd face" to the "face with stuck out tongue"

a. specify which attribute you selected, the vector arithmetic applied and the resulting image(s) as displayed above

- "crying cat face" - "crying face" + "face with tears of joy" = Generated emoji



b. provide a qualitative evaluation of your composite image

- When examining the generated emoji above, we can clearly see that it has the characteristics of cat ears, and although a little bit blurry, the slimy face and the tears of joy are still identifiable. The color of the generated emoji leans towards the face emoji and the shape of the face is round instead of the shape of a cat emoji.

c. discuss ways to improve the quality of your generated image.

- Augmenting our training data with more ways of operations such as cropping and scaling can help the model learn to generate more diverse emojis
- Training the model using a much larger dataset or fine-tuning could help improve its ability to reconstruct and generate emojis for this specific task
- Possibly tweak model layers (adding layers, modify channel size) might also help