CMPE 597 Sp. Tp. Deep Learning
Spring 2025 Assignment 1
Due March 30th by midnight

Data

In this assignment, the Quick Draw dataset is going to be used. The dataset originally comprises 50 million drawings of 345 categories collected from the players of the game called Quick Draw. For this assignment, a subset of the dataset was downloaded from Hugging Face [1]. You may download the subset prepared for this assignment `https://drive.google.com/file/d/1oGOHnQEb7VVwpdYsn8rShfAQ8Vq4hA9E/view?usp=sharing`. The subset has 20k grayscale images of size 28×28 in the training set and 5k in the test set. An equal number of images were selected from 5 categories in training and test sets. A README is provided in the dataset folder for how to load and visualize the images.

Tasks

I. Classification with Cross-Entropy

In this task, you need to implement a multi-layer perceptron to classify images into their categories. Your network's input will be a 784-dimensional vector after flattening the images, and the output will be 5-dimensional. The last layer's activation should be softmax, as given below:

$$\text{Softmax}\left(\mathbf{h}\right)_i = \frac{\exp\left(\mathbf{h}_i\right)}{\sum_{j=1}^{C} \exp\left(\mathbf{h}_j\right)}$$

where $\mathbf{h}_i$ is i-th dimension of the last layers output before activation and C is the number of categories for this task.

You will train your network to minimize the categorical cross-entropy loss given below:

$$\mathcal{L} = -\sum_{n=1}^{N}\sum_{c=1}^{C} y_{nc} \log \hat{y}_{nc}$$

where N is the size of the mini-batch, C is the number of categories, and y_{nc} and $\hat{y}_{nc}$ are the ground truth and your network's prediction, respectively. Here, the ground truth y is a one-hot representation of the category.

1. **Implementation from Scratch:** You cannot use any deep learning library in this part. You must implement a data loader, network architecture, forward propagation algorithm, backpropagation algorithm, and optimization framework. You are allowed to use libraries such as numpy.

[1] `https://huggingface.co/datasets/google/quickdraw`

(a) Design a multi-layer perceptron for this task. Please explain how you chose the depth and width of the network and activation functions.

(b) Implement stochastic gradient descent with momentum. You may set the momentum value to 0.9. Please feel free to tune it if this value does not work for you.

(c) Evaluate the performance of your network on the test set provided to you. You may use classification accuracy, precision, recall, the area under the ROC curve, and the area under precision-recall curve metric implementations of scikit-learn.

(d) Please plot the changes in loss, training, and validation accuracies during training. You may spare a random subset from the training set for validation.

(e) Clearly explain how you chose the hyperparameters.

(f) Discuss what you tried to improve your network's performance.

2. **Implementation with Deep Learning Libraries:**

   (a) Now implement the same architecture and train your model with the same optimizer and the hyperparameters in part 1 using a deep learning library.

   (b) Compare the gradients you obtain using your backpropagation and the library's implementation for the first iteration.

   (c) Compare the test classification accuracies and loss plots for both implementations.

II. Classification with Word Embeddings

In this task, the goal is again to categorize the images. However, you will not use cross-entropy loss this time. Please implement two multi-layer perceptrons: one learns image embeddings, and one maps the word embeddings of the category names to the same dimensional space as image embeddings. Thus, the loss function should minimize the discrepancy between the image and the category embeddings in their shared space if the image belongs to that category. To evaluate the classification accuracy, you may rank the similarities or distances between the image embedding and the category embeddings and predict the top similar category.

The names of the categories are 0: rabbit, 1: yoga, 2: hand, 3: snowman, and 4: motorbike. You may use Glove [2] word embeddings.

Please use the same training and validation splits you randomly sampled in part I.

1. **Implementation from Scratch:** As in the first task, you cannot use any deep learning library. However, you can use a library to extract the word embeddings of the categories.

   (a) Design two multi-layer perceptrons for this task. Please explain how you chose the depth and width of the networks and activation functions.

[2] `https://nlp.stanford.edu/projects/glove/`

(b) Implement stochastic gradient descent with momentum. You may set the momentum value to 0.9. Please feel free to tune it if this value does not work for you.

(c) Please explain how you choose the loss function. Feel free to explore and add more than one penalty term if it helps.

(d) Evaluate the performance of your network on the test set provided to you. You may use classification accuracy, precision, recall, the area under the ROC curve, and the area under the precision-recall curve metric implementations of sci-kit-learn.

(e) Please plot the changes in loss, training, and validation accuracies during training. If you have multiple loss terms, also plot them separately.

(f) Clearly explain how you chose the hyperparameters.

(g) Could you train your network for this task? Please discuss the challenges you faced and how you overcame them.

2. **Implementation with Deep Learning Libraries:**

   (a) Now implement the same architecture and train your model with the same optimizer and the hyperparameters using a deep learning library.

   (b) Compare the test classification accuracies and loss plots for both implementations.

Submission

- A PDF report addressing the above points, including your name, student number, references, and drive link to the code, should be submitted.
- Please include the link to your code on the first page of your report.
- Your experiments should be reproducible.
- A readme should be provided.

IMPORTANT NOTES:

- There is no report template. However, your reports should address all the bullet points above.
- You should not forget to cite your references. If you followed a GitHub repository, do not forget to cite it.
- Please note that you should be ready to answer questions regarding your code.
- Grading of this assignment will not be based on performance. However, you should be able to show that your networks can learn. In Part I, you can get a classification accuracy of at least around 60%. In Part II, you may get a lower accuracy than Part I, which is alright.