Generative AI Assignment

Muhammad Zaraar Malik
BS - AI Section A
FAST University
Islamabad, Pakistan
i212705@nu.edu.pk

I. Introduction

This is the First Assignment of the Course Generative Artificial Intelligence for Fall-2024. The course instructor is **Dr. Akhtar Jamil**. This Assignment Consists of 2 questions, The first part targets the **Convolutional Neural Networks** and requires the students to Train and Test a CNN for **Signature Classification**. The data of signatures is basically the signatures of each students taking taking this course. The second question focuses on sequence classification. Each student is required to train a **Long-Short-Term-Memory Model** on Shakespear Dataset which has been downloaded from kaggle from the following ling : https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays . Students were responsible for designig the pre-processing steps , model training and model evaluation. Moreover, Students also had to **Deploy this Model** via Fast or Streamlit API and make a standard Web UI where a user would type in text and the UI would recomend/predict the next word based on the previously entered words.

II. Question Number 1

A. *Data Set Information and Pre-Processing Steps*

The Dataset provided to us for the first question was a zip-file containing some images . Each image was contained a photograph of a white paper containing horizantal and vertical boxes where the first box in each row represents the index of the Signature and the next 4 boxes adjacent to the index contained Signatures of individuals who are currently taking the Genrative AI class. The sample of the image has been displayed on the right side of this document. Now, moving towards the Pre-processing step to of the provided dataset. i have used the following pipeline to segment the signatures along with their respective index and in reality the index represents a particular student which has signed the provided document.

- **Gaussian filter:** The main objective initially was to use a smoothing filter inorder to remove the noise present in the images. For that purpose , a Gaussian filter was used with a kernel size of 3 and a standard deviation from the Gaussian kernel.
- **Canny Filter** The next step was to detect the edges of the provided images after smoothing out or removing the noise from the image. For this purpose, I used the Canny Edge Detector to detect the edges. The Canny Edge detector was used with a Lower threshold value of 100 and a upper thresh hold value of 200.
- **Contour Detection** The next step was to detect the contours which will actually help us detect the horizantal boxes inside the provided images and further which we then segment these using a simple mathematical approach.
- **Finding Largest contour Corner Points** The next step is that we then find the corner points of the largest contour detected in the previous step and then from their on forward we slowly segment the image and place it into seperate files before the model training process.

Fig. 1. Signatures of Students

III. Model Architecture

After the preparation of the dataset. The next part of was to design the CNN model which was further going to be used for the Signature Classification Problem . The following is my designed CNN arcchitecture:

- 2x Convolution Layers
- Filter Size (3x3)
- 2x Max Pooling Layers
- Filter Size (2x2)
- 1x Fully Connected Layer
- epochs = 10
- Learning rate = 0.001
- Training Batch Size = 16
- Loss = Cross-Entropy-Loss
- Optimizer Adam
- Relu Activation Function
- Early Stopping to prevent overfitting

The main intuition behind the these parameters is that having a suitable model which is neither too much simple and nor too much complex but a balanced model in terms of performance and arhitecture. I have used 2 Convolution layers to deeply analyze the input data and extract meaningfult information from the provided images using the filters that are defined inside the convolution layer. I have used the Pytorch Module to develop this network. Furthermore, after each convolution operation , a max pooling operation is performed to reduce the spatial feature of the provided images. Lastly, and most importantly comes the Fully Connected Layer. The main purpose of attaching this layer was that to learn the short-commings of the previous layers and to classify the learnt outputs so that during the testing time each input or provided image is classified for a specific index value or in-directly a specified person.

IV. Mathematical Equations used in the Architecture

A. Adam Optimizer

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*
g_t : *Gradient at time t along* ω^j
ν_t : *Exponential Average of gradients along* ω_j
s_t : *Exponential Average of squares of gradients along* ω_j
β_1, β_2 : *Hyperparameters*

Fig. 2. Adam Optimizer for Weight Updation

The above mentioned equation is the Famous Adam Optimizer. This optimizer has been and is still being widely used for Model Training. The main goal is to streamline the weight updation process during the model training part.

B. Rectified Linear Unit

This is the ReLU activation function. This has been widely used in between the layers of the neural network. The main purpose of an activation function is to introduce Non-Linearity in the Model so that the model can learn complex and hidden patterns in the dataset and adjust itself accordingly rather than memorizing the data which further leads to Model OverFiting.

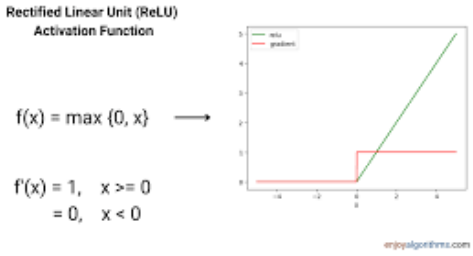


Fig. 3. ReLU Activation Function

C. Convolution Operation

This is a simple example of a convolution neural network. The green box represents the 2D image, the orange box represents the filter present inside the Convolution layer and the blue image is basically a convolved image. The backward pass updates the weights of the filter which basically means the values inside the blue box are learned by the model during the Training event.

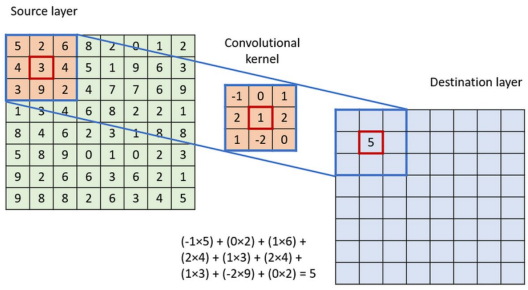


Fig. 4. convolution operation kernel size 3x3

V. Testing and Evaluaion Metrics

In order to test the trained model , I used the the classifiction report metric from Sci-kit Learn Module. This metric clearly shows the following items:

- F1-Score
- Macro Avg
- Weight Avg

Furthermore, From the expcept for the classification report, we can also Generate the Confusion Matrix from the Sci-Kit Learn Module.

VI. Current Possible Issues

A. Dataset Issues

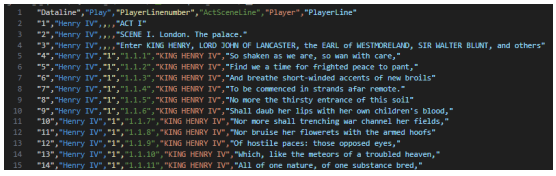
The main and only issue for this question was that the provided dataset was very less. There were 184 total students

and each student had 4 signatures. This means that if I split the dataset for training, 3 Training samples of each student in the Training data and 1 testing sample for the testing data. Which is quite and very low for a CNN Model that has to predict 184 classes by the end of the day. This has been the main issue and is the reason for a low score during the testing phase. Another major issue is that, each student has a different style of doing a signature. So for such a complex data, we should have at least reasonable amount of data in which the model can be trained properly.

VII. Question Number 2

A. Dataset Information and PreProcessin Steps

The next question is related to word prediction using LSTM. A link to a kaggle dataset file was provided which was basically the dataset. Shakespear Plays were used to train the LSTM model. I only used the "PlayerLine" in the dataset and removed all the other information from the dataset. Following is a small part of the dataset



```
"Dataline","Play","PlayerLinenumber","ActSceneLine","Player","PlayerLine"
"1","Henry IV",,,,"ACT I"
"2","Henry IV",,,,"SCENE I. London. The palace."
"3","Henry IV",,,,"Enter KING HENRY, LORD JOHN OF LANCASTER, the EARL of WESTMORELAND, SIR WALTER BLUNT, and others"
"4","Henry IV","1","1.1.1","KING HENRY IV","So shaken as we are, so wan with care,"
"5","Henry IV","1","1.1.2","KING HENRY IV","Find we a time for frighted peace to pant,"
"6","Henry IV","1","1.1.3","KING HENRY IV","And breathe short-winded accents of new broils"
"7","Henry IV","1","1.1.4","KING HENRY IV","To be commenced in strands afar remote."
"8","Henry IV","1","1.1.5","KING HENRY IV","No more the thirsty entrance of this soil"
"9","Henry IV","1","1.1.6","KING HENRY IV","Shall daub her lips with her own children's blood,"
"10","Henry IV","1","1.1.7","KING HENRY IV","Nor more shall trenching war channel her fields,"
"11","Henry IV","1","1.1.8","KING HENRY IV","Nor bruise her flowerets with the armed hoofs"
"12","Henry IV","1","1.1.9","KING HENRY IV","Of hostile paces: those opposed eyes,"
"13","Henry IV","1","1.1.10","KING HENRY IV","Which, like the meteors of a troubled heaven,"
"14","Henry IV","1","1.1.11","KING HENRY IV","All of one nature, of one substance bred,"
```

Fig. 5. Shakespear Dataset

B. Preprocessing Steps

- **Keep only Sentences:** This involved the process to remove all the unnecessary information present inside the lines of the actors such as ACT, EXIT, EXEUNT. I removed all of these lines in the pre-processing function
- **Stopwords:** The next step was the removal of stopwords from the dataset. These are words which are included in a sentence to give it structure but mostly it hinders the training process as the count of stopwords such as "a" , "is" , "and" etc can have a higher influence and confuse the model.
- **Word Tokenization:** After the removal of Stopwords, the next step was to tokenized each of the sentence into words. By doing so, we can make an ngram sequence which will be further used to train the LSTM model for next word prediction
- **Ngram Sequence Generation** The next step was to generate an ngram sequence generation to make a sequentail input for the model. I kept a sequence length = 5. I removed all those sentences from the input sequence which had a lentgh lower than this so that the model is provided with a more contextual-rich sequence
- **Word to Vector** The Ngram Sequence was then next converted to a vector using a vector Dictionary which i previously constructed. It simply assigned a numerical value to the token in the Ngram sequence.
- **Dataset Seperation** The dataset was further divided into Training , Testing and Validation datasets. These were then used to train , validate and test the model

VIII. Model Architecture

After the preparation of the dataset. The next part of was to design the LSTM model which was further going to be used for the Next Word Prediction Problem . The following is my designed LSTM Architecture:

- 1x Embedding Layer
- 1x Hidden Layer (size = 200)
- 1x LSTM Layer
- 1x Fully Connected Layer (size = unique labels in dictionary)
- epochs = 20
- Learning rate = 0.001
- Training Batch Size = 32
- Loss = Cross-Entropy-Loss
- Optimizer Adam
- Early Stopping to prevent overfitting

The main intuition behind the these parameters is that having a suitable model which is neither too much simple and nor too much complex but a balanced model in terms of performance and arhitecture. I have used 1 Embedding Layer to deeply analyze the input data and extract meaningful information from the provided sequence and further map it into a higher dimension. The outputs from this layer are then passed to the hidden layer which further maps the inputs into a higher dimension. The next comes the LSTM Layer where the output of the hidden layer serves as an input to the LSTM layer. The output of this layer is then connected to a Fully Connected Layer which has a output dimension of size equaly to the number of unique labels present in our Vector Dictionary.

IX. Mathematical Equations Used

The equations used in this are the same as used in the previous one with 1 addition that now, an LSTM model is being trained . A simple training visualization can be seen below :

```
import torch
from torch import nn

class LSTM_cell_AI_SUMMER(torch.nn.Module):
    """
    A simple LSTM cell network for educational purposes
    """
    def __init__(self, input_length=10, hidden_length=20):
        super(LSTM_cell_AI_SUMMER, self).__init__()
        self.input_length = input_length
        self.hidden_length = hidden_length
```

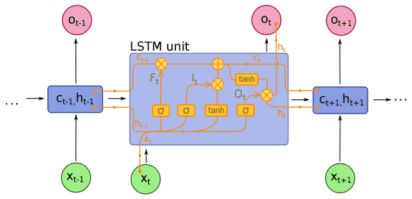


Fig. 6. LSTm Model

X. Testing and Evalution

Once, the LSTM model was trained , I prepared a testing dataset initially and used it to test the model. LSTM training was very very slow and difficult due to the reason that it overfits very easily and my Val Loss resulted very high in many cases. Moreover, the training itself is slow as we are training a model overtime stamps . My each epoch took roughly 3.5 minutes. After the Testing of the model, I saved the model using the ”pickle” module and started working on my Web UI which was also asked in the question .

XI. Web UI and FAST API

I developed a Web UI in HTML and CSS and connected it FAST API to host a website that allows the user to type and while typing the input is being feedback to the model and the model is suggesting what can be the next possible output.

References

[1] https://opencv.org/
[2] https://www.geeksforgeeks.org/
[3] https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays
[4] https://stackoverflow.com/
[5] https://medium.com/analytics-vidhya/introduction-to-long-short-term-memory-lstm-a8052cd0d4cd