Generative AI Assignment 1: Signature Image Recognition using CNN and Word Completion using LSTM

Imama Amjad1
BS CS
FAST-NUCES University
Islamabad, Pakistan
i201819@nu.edu.pk

***Abstract*—This report presents the implementation of two deep learning models aimed at solving distinct problems: signature recognition using Convolutional Neural Networks (CNN) and sentence completion using Long Short-Term Memory (LSTM) networks. In the first task, a system was developed to preprocess and classify handwritten signatures. Various image preprocessing techniques, including gamma correction, CLAHE, and contour detection, were employed to enhance the signature images before training the CNN model. The performance of CNN-based feature extraction was compared with manual techniques such as HOG and SIFT, and the evaluation metrics included precision, recall, f-measure, and accuracy.**

The second task involved building a word-level LSTM model to predict the next word in a sequence from Shakespeare's plays. After text tokenization and preprocessing, an LSTM-based architecture was trained to generate word suggestions. The model's output was assessed for coherence and fluency. Additionally, a web-based interface was created to dynamically suggest words as users input text, offering real-time interaction. The effects of different hyperparameters were also explored to optimize performance. Both models were evaluated based on their training and testing performance, and visualizations were included to support the analysis.

***Index Terms*—CNN, LSTM, Signature Recognition, Word Completion, Neural Networks, Image Processing, Natural Language Processing**

I. Introduction

A. Q1: Signature Image Recognition using CNN

The objective of the first part of this assignment is to implement a program that processes the images of signatures, segments them, and classifies them using Convolutional Neural Networks (CNN). The system is also expected to compare the CNN-based feature extraction with manual feature extraction techniques, such as Histogram of Oriented Gradients (HOG) and Scale-Invariant Feature Transform (SIFT). The evaluation of the model performance will be done using metrics like precision, recall, f-measure, and accuracy. Additionally, detailed visualizations like training/test accuracy and confusion matrices are required to support the performance analysis.

B. Q2: Word Completion using LSTM

The second part focuses on building a word-level Long Short-Term Memory (LSTM) model for sentence completion. Using Shakespeare's plays dataset, the model is trained to predict the next word in a sequence. Additionally, a user-friendly interface should be created, allowing users to input partial sentences, after which the model suggests the next word dynamically. The model will be evaluated based on coherence and fluency of generated text, and hyperparameters will be explored for improved performance.

II. Methodology

A. Q1: Signature Image Recognition using CNN

1) Dataset: The dataset contains 184 rows, each row with four columns representing the signatures of individuals. Each row begins with the person's ID, followed by the four signature images.

2) Preprocessing: The following techniques were applied to preprocess the signature images:

- **Image Cropping:** Detected horizontal and vertical lines were used to crop the signature image.
- **Grayscale Conversion:** The images were converted to grayscale to reduce complexity.
- **Gamma Correction:** Applied gamma correction to enhance brightness and contrast.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization):** Enhanced image contrast.
- **Median Blurring:** Used to reduce noise while preserving edges.
- **Sobel Filter:** Applied for edge detection.
- **Adaptive and Otsu Thresholding:** Used for binarization and contour extraction.
- **Contour Detection:** Used to isolate the signature from the background.

3) Model Architecture: In this section, we describe the architectures of the four models used in the task: CNN, ANN, HOG with SVM, and SIFT with SVM. Each model was designed and trained to address the signature recognition

problem using different feature extraction and classification methods.

a) Convolutional Neural Network (CNN): The CNN model consists of the following layers:

- **Input Layer**: Input images were resized to 128x128 with 3 channels (RGB). The input shape for the network was `(128, 128, 3)`.
- **Convolutional Layers**:
  - First convolutional layer with 32 filters of size 3x3, followed by a MaxPooling layer (2x2).
  - Second convolutional layer with 64 filters of size 3x3, followed by a MaxPooling layer (2x2).
  - Third convolutional layer with 128 filters of size 3x3, followed by a MaxPooling layer (2x2).
- **Flatten Layer**: The output from the final MaxPooling layer was flattened into a 1D vector.
- **Fully Connected Layer**: A dense layer with 128 units and ReLU activation, followed by a dropout layer with a 0.5 dropout rate to prevent overfitting.
- **Output Layer**: A final dense layer with softmax activation corresponding to the number of classes in the dataset (determined dynamically based on the training data, i.e., 184 classes).

The model was compiled using the Adam optimizer, with categorical cross-entropy as the loss function, and accuracy as the evaluation metric. Early stopping was implemented to prevent overfitting, and the best model was saved during training. The total number of parameters in the CNN model was approximately 278,000.

b) Artificial Neural Network (ANN): The ANN model was structured as follows:

- **Input Layer**: The input images were flattened into a 1D vector of size `(128 * 128 * 3)`.
- **Dense Layers**:
  - First dense layer with 128 units and ReLU activation, followed by a dropout layer with a 0.5 dropout rate.
  - Second dense layer with 64 units and ReLU activation, followed by another dropout layer with a 0.5 dropout rate.
- **Output Layer**: A dense output layer with softmax activation corresponding to the number of classes in the dataset (184 classes).

The ANN model was also compiled using the Adam optimizer, with categorical cross-entropy as the loss function, and accuracy as the evaluation metric. The model included 221,056 parameters, and early stopping was implemented to prevent overfitting.

c) HOG with SVM Classifier: For the Histogram of Oriented Gradients (HOG) with SVM approach, the following process was used:

- **HOG Feature Extraction**: HOG features were extracted from each grayscale image, resized to 128x128. The following parameters were used for HOG:
  - 9 orientations
  - 8x8 pixels per cell
  - 2x2 cells per block
  - Block normalization using the `L2-Hys` method
- **SVM Classifier**: The extracted HOG features were fed into a Support Vector Machine (SVM) with a linear kernel. A pipeline with standardization was applied to normalize the feature vectors before classification. The SVM was trained on 184 classes, corresponding to the different signatures.

This method did not involve deep learning, so there were no backpropagation-based parameter updates. The total number of support vectors in the SVM model varied depending on the feature extraction but typically amounted to several thousand.

d) SIFT with SVM Classifier: The Scale-Invariant Feature Transform (SIFT) with SVM approach followed these steps:

- **SIFT Feature Extraction**: SIFT keypoints and descriptors were extracted from each grayscale image. The SIFT descriptors were averaged across keypoints to create a fixed-length feature vector for each image.
- **SVM Classifier**: Similar to the HOG approach, the SIFT descriptors were fed into a Support Vector Machine (SVM) with a linear kernel. Standardization was applied to the SIFT features before classification.

This approach leveraged the robustness of SIFT in capturing scale-invariant and rotation-invariant features. The SVM was trained on the same 184 classes, and accuracy was computed using standard classification metrics.

Each of these models was evaluated using the testing data, and performance metrics such as accuracy, precision, recall, and confusion matrices were recorded to compare their efficacy in recognizing signatures.

B. Q2: Word Completion using LSTM

1) Dataset: Shakespeare's plays dataset was used, containing lines of text from various works of Shakespeare.

2) Preprocessing:

- **Tokenization:** The text was tokenized into individual words.
- **Vocabulary Mapping:** A vocabulary of unique words was created, mapping each word to an integer.
- **Sequence Generation:** The text was converted into overlapping sequences of fixed length (e.g., 5 words per sequence), with the target being the next word in the sequence.
- **Cleaning:** Punctuation, numbers, and stop words were removed from the text, and all characters were converted to lowercase.

3) Model Architecture: The architecture for the word-level LSTM model is as follows:

- **Embedding Layer:** The input sequences are passed through an embedding layer that converts the integer-encoded words into dense vectors of fixed size.
- **LSTM Layer:** A Long Short-Term Memory (LSTM) layer with 100 units is used to capture temporal dependencies between words.

- **Dropout:** Dropout with a rate of 0.4 is added to prevent overfitting during training.
- **Dense Layer:** A fully connected layer with a softmax activation function is used as the output layer, predicting the next word in the sequence based on the vocabulary size.
- **Compilation:** The model is compiled with the Adam optimizer, using categorical cross-entropy as the loss function and accuracy as the evaluation metric.
- **Callbacks:** Early stopping and learning rate reduction on plateau were applied to prevent overfitting and optimize the learning process.

III. Results

A. Q1: Signature Image Recognition using CNN

This subsection presents the results for each of the four models: CNN, ANN, HOG with SVM, and SIFT with SVM. Training and validation accuracy, loss, and observations are provided for each model along with visualizations.

1) CNN (Convolutional Neural Network): The CNN model was trained with early stopping at epoch 5, with the best epoch being epoch 2. The final accuracy was approximately 1%. CNNs generally perform well on large datasets, but the small size of this dataset limited the model's ability to generalize effectively. The model showed overfitting after early epochs, as indicated by the drop in accuracy and increase in validation loss.

Why it performed poorly: CNNs require large datasets to learn meaningful patterns from image data, and the small dataset size led to early overfitting. Although the model quickly learned basic features by epoch 2, it failed to generalize, resulting in poor performance.

Possible Improvements:

- Applying data augmentation techniques like rotation, shifts, flips, and zooms to introduce variability into the dataset.
- Using transfer learning, where a pre-trained CNN such as ResNet or VGG is fine-tuned on the small dataset.

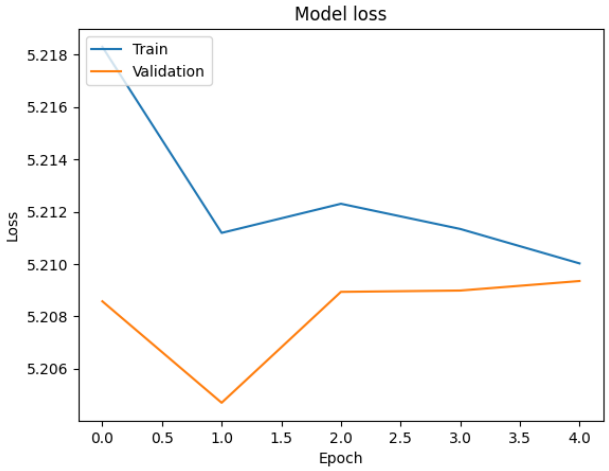

Fig. 1. CNN: Accuracy

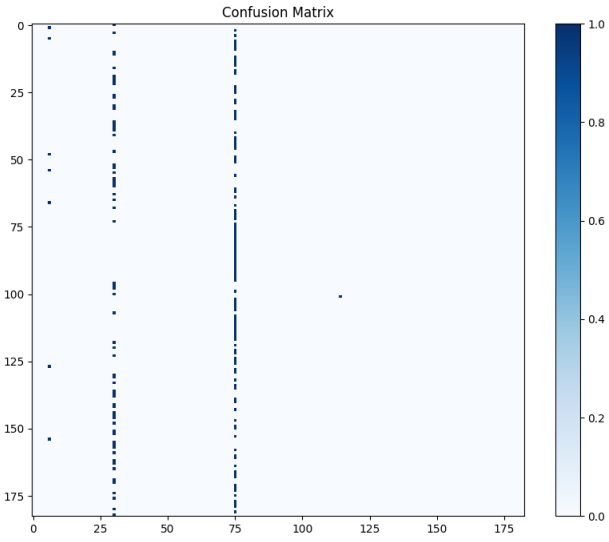

Fig. 2. CNN: Confusion Matrix

B. ANN (Artificial Neural Network)

The ANN model reached a maximum accuracy of 1.32% at epoch 7, after which overfitting occurred. The small dataset led to the model failing to generalize well, and the accuracy dropped after epoch 7. ANN models typically require larger datasets to extract meaningful patterns, and in this case, the model struggled due to the limited dataset size.

Why it performed poorly: ANN models tend to underfit or overfit when trained on small datasets, and the limited data did not provide enough variation for the network to learn high-dimensional patterns effectively.

Possible Improvements:

- Data augmentation could help increase the dataset size and improve generalization.
- Applying regularization techniques such as L2 regularization or dropout to prevent overfitting.
- Transfer learning can be used to leverage pre-trained models on larger datasets.

C. HOG with SVM

The HOG + SVM model achieved an accuracy of 31%. HOG (Histogram of Oriented Gradients) captures gradient information and works well for object detection tasks. However, for signature recognition, it underperformed compared to SIFT as it may not have captured sufficient local features to distinguish signatures as effectively.

Why it underperformed: HOG captures shape and texture information, which might not have been enough for signature variation in this dataset. The method works well in object detection but might miss finer details in signatures that SIFT captures.

Possible Improvements:

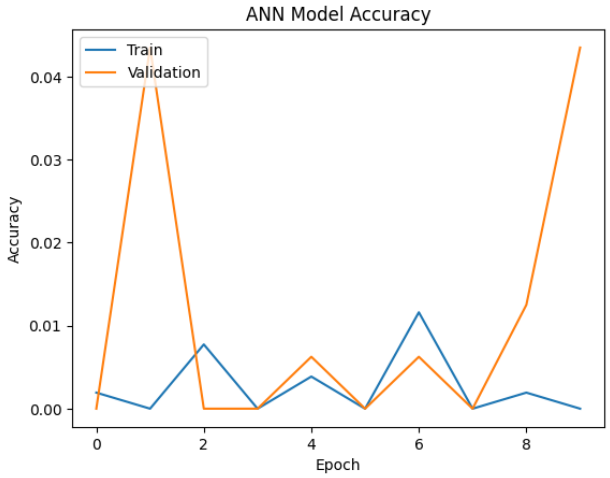

Fig. 3. ANN: Accuracy

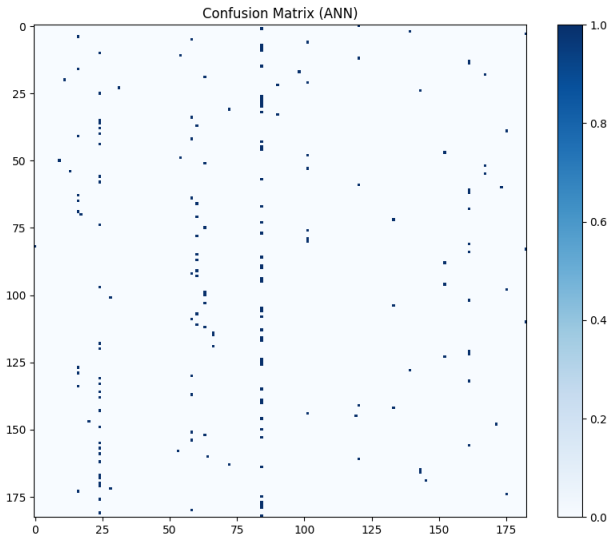

Fig. 4. ANN: Confusion Matrix

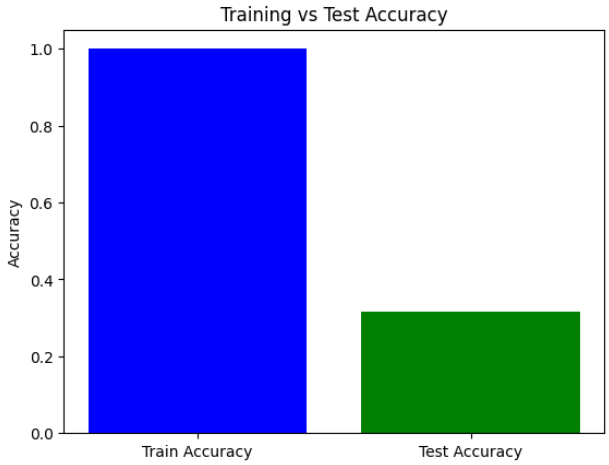

Fig. 5. HOG + SVM: Accuracy

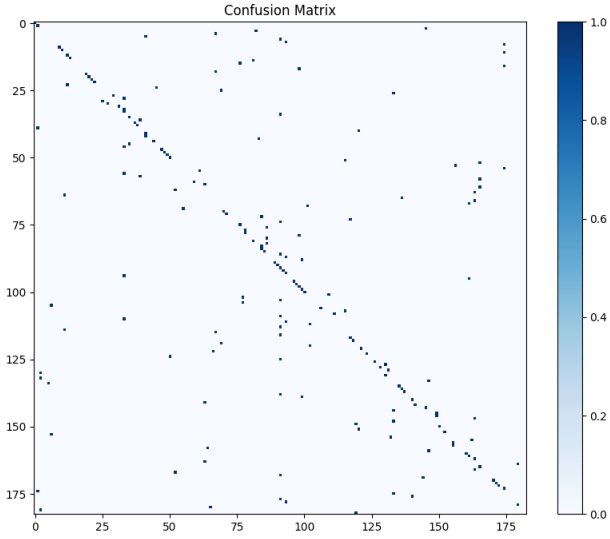

Fig. 6. HOG + SVM: Confusion Matrix

- Spatial pyramid matching with HOG to capture hierarchical patterns.
- Combining HOG with another feature extraction method, such as SIFT or ORB, to complement both global and local features.

D. SIFT with SVM

The SIFT + SVM model performed the best, achieving 33.3% accuracy. SIFT (Scale-Invariant Feature Transform) effectively captures keypoints and local patterns, making it ideal for signature recognition, where keypoint-based feature extraction helps differentiate between subtle variations. The SVM classifier also works well with small datasets and high-dimensional feature vectors, further enhancing performance.

Why it worked best: SIFT's keypoint-based approach captured critical local patterns, and SVM's robustness to small datasets made this combination the best performer on the given dataset.

Possible Improvements:

- Fine-tuning the SVM kernel (try RBF or polynomial) for better classification.
- Using Dense SIFT to capture more dense information and improve feature extraction.

E. Q2: Word Completion using LSTM

The LSTM model trained for the word completion task yielded poor performance, achieving a low accuracy of 3.45% after 30 epochs, with a training loss of 7.42 and a validation loss of 8.06. The learning rate was reduced to 0.00025 after epoch 30 due to the learning rate reduction mechanism (`ReduceLROnPlateau`), indicating that the model had plateaued in terms of learning. Model weights were restored from the end of epoch 27 as this was determined to be the best epoch during training.

Fig. 7. SIFT + SVM: Accuracy

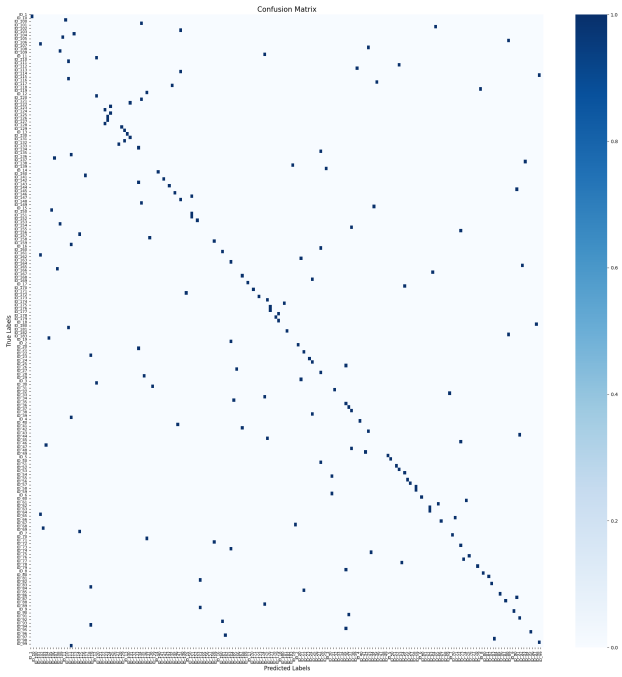

Fig. 8. SIFT + SVM: Confusion Matrix

Batches and Hardware Limitations: Due to hardware constraints, padded sequences were stored in batches on Google Drive and read sequentially during training using Python's data generator. This significantly affected the model's ability to learn effectively, as batches had to be processed one at a time. The RAM available on the system was insufficient to load the entire dataset into memory at once, which likely contributed to the overall underperformance of the model and affected the coherence of sentence predictions.

1) Why the Model Underperformed: Several factors contributed to the poor performance and lack of coherence in the predicted sentences:

- **Insufficient RAM and Hardware Limitations**: The need to process data sequentially in batches due to RAM limitations hindered the model's ability to train efficiently. The continuous reading from disk, rather than keeping data in memory, slowed down training and affected convergence, leading to incomplete learning of sentence structures.
- **Incoherent Sentence Predictions**: Due to the low accuracy of 3.45%, the model struggled to predict coherent sentences. The words predicted by the LSTM were often irrelevant or nonsensical, failing to match the context of the preceding words. This is a direct result of the model not learning the long-range dependencies in the text, which are crucial for generating meaningful and grammatically correct sentences.
- **Overfitting and Lack of Generalization**: The validation accuracy plateaued and even decreased during the latter stages of training, indicating the model may have overfit to the training data. The complexity of Shakespeare's text and the small dataset size made it difficult for the model to generalize well, further contributing to incoherent sentence predictions.
- **Learning Rate Plateau**: By epoch 30, the learning rate was reduced to 0.00025, signaling that the model had reached a point of diminishing returns in learning. The small adjustments made by the reduced learning rate were not sufficient to improve the model's performance, and the predictions remained largely incoherent.

2) Possible Improvements: Several improvements can be applied to the LSTM model and the overall approach to achieve better performance and more coherent sentence predictions:

- **Transfer Learning or Pre-trained Language Models**: Rather than training an LSTM from scratch, using a pre-trained language model such as GPT or BERT, which has been fine-tuned for sentence completion tasks, could significantly improve performance. These models can better handle long-range dependencies and generate more coherent sentences.
- **Data Augmentation and Larger Dataset**: Expanding the dataset by including other texts or augmenting the existing data (e.g., through paraphrasing or using synonyms) could help the model learn more varied sentence structures, improving both accuracy and coherence.
- **Increase Model Capacity**: Increasing the LSTM layers and units may allow the model to learn more complex patterns in the dataset. More sophisticated architectures, such as bidirectional LSTMs or attention mechanisms, could help capture long-range dependencies more effectively, leading to more coherent sentence completions.
- **Hardware Upgrade**: Utilizing a system with more RAM or a faster storage solution would allow the model to train on the entire dataset at once rather than in batches, improving training speed and reducing memory bottlenecks. This would allow the model to better learn the full sentence structure, improving coherence.

IV. Discussion

A. Q1: Signature Image Recognition using CNN

In Q1, the task involved recognizing signatures from a dataset consisting of 184 classes, with each class having four signatures. This dataset presented a challenge due to its relatively small size, which affected the performance of both Convolutional Neural Networks (CNN) and Artificial Neural Networks (ANN). With such a small number of samples, it was difficult for the models to generalize effectively, which led to suboptimal results.

1) Challenges in Data Extraction and Contour Detection: The dataset posed significant challenges during data extraction and contour detection of the signatures. A major issue was the extra row splitting and failure to detect valid signatures in certain rows. In many cases, the signatures were not properly isolated from the surrounding noise, resulting in incomplete or poorly detected contours.

To overcome these challenges, several image preprocessing techniques were applied:

- **Adaptive Thresholding:** This technique was used to binarize the image and enhance the contrast between the signature and the background, making it easier to detect contours.
- **Gamma Correction:** Gamma correction was applied to adjust the brightness and contrast of the signature images, improving the visibility of finer details.
- **Hough Line Transform:** This method was used to detect the largest horizontal and vertical lines, which were critical for segmenting the signature images. By detecting these lines, the signature areas could be accurately cropped.
- **Gaussian Blur:** Gaussian blur was applied to reduce noise and smooth the image, making it easier to detect edges and contours in subsequent steps.

Despite these enhancements, extracting clean signatures from the dataset was still difficult due to the variability in the quality of the signatures. The application of techniques like Sobel filtering, CLAHE (Contrast Limited Adaptive Histogram Equalization), and Median Blurring helped mitigate some of these issues by enhancing the edges and improving contrast.

2) Model Performance: The small dataset size was a limiting factor for both CNN and ANN models. Typically, deep learning models perform best when trained on large datasets, but the limited number of samples (184 classes, each with only four signatures) resulted in overfitting during training. The models were able to learn the training data relatively well but struggled to generalize to unseen data.

To address this, several adjustments were made during training:

- **Data Augmentation:**: Techniques such as rotation, scaling, and flipping were applied to artificially increase the dataset size, helping to improve the generalization of the models.
- **Early Stopping:**: Early stopping was used to prevent the models from overfitting by monitoring the validation loss and stopping the training process once the loss plateaued.

Even with these adjustments, the model performance remained limited due to the inherent constraints of the dataset. The CNN-based feature extraction methods outperformed the manual methods (HOG and SIFT), but the overall accuracy, precision, recall, and f-measure scores were still not ideal for practical applications.

In conclusion, while the preprocessing techniques greatly improved the quality of the input data, the small dataset size was a significant limiting factor in achieving high accuracy with both CNN and ANN models. Future work could explore expanding the dataset or experimenting with transfer learning approaches to improve the model's ability to generalize.

B. Q2: Word Completion using LSTM

The goal of Q2 was to develop a word-level Long Short-Term Memory (LSTM) model for sentence completion using Shakespeare's plays dataset. However, the model encountered several challenges, particularly with sentence coherence and accuracy.

1) Sentence Coherence and Model Performance: The model's overall accuracy after 30 epochs was 3.42%, which is quite low for a task requiring language understanding. The loss during training was 7.42, and the validation loss reached 8.06, indicating that the model struggled to generalize well to unseen data. As a result, the sentences generated by the LSTM were often incoherent and lacked meaningful word suggestions. This low accuracy can be attributed to several factors:

- **Training Data Complexity:** The Shakespearean text is rich with archaic language, complex sentence structures, and varying word usage. This likely made it harder for the LSTM to capture meaningful patterns during training. The dataset's vocabulary size of 27,230 further complicated the task, making it difficult for the model to learn accurate word predictions.
- **Limited Training Epochs:** While the model was trained for 30 epochs, the ReduceLROnPlateau callback decreased the learning rate to a very low value by the end of training, suggesting the learning progress had plateaued. This indicates that additional training might not have helped improve the model significantly due to the lack of sufficient signal for learning.
- **Coherence Issues:** Since the model was only able to achieve a small accuracy, it struggled to form coherent sentences. The output words were mostly irrelevant or nonsensical in the context of the preceding words. This may also be linked to the inability of the model to learn long-range dependencies in text effectively, which is crucial for sentence completion.

2) Challenges with Resource Usage and Training Strategy: The low coherence and accuracy could also be attributed to hardware limitations. Despite having access to 83.5GB of RAM on Colab Pro, memory constraints were a persistent

challenge. The vocabulary size and the padded sequences required substantial memory to store and process efficiently.

- **RAM Usage:** Due to memory limitations, padded sequences were saved in batches and uploaded sequentially for training. This batch-processing approach, while necessary to avoid memory overflows, negatively impacted the overall training process. Training the model in smaller batches increased the computational overhead and potentially affected the ability of the model to optimize parameters effectively.
- **Suboptimal Hyperparameters:** The inability to fully optimize the model may have been linked to resource limitations. With batch-processing, it became difficult to find the best set of hyperparameters (such as learning rate, batch size, or LSTM units), leading to suboptimal training. As a result, the LSTM model could not reach its full potential in terms of both accuracy and coherence.
- **Slow Training:** The training process was slower than expected due to the need to save and reload batches of padded sequences. This limited the model's capacity to train on larger batch sizes, which could have otherwise sped up convergence and improved the model's learning.

In summary, while the model was able to learn to some extent, the hardware limitations and suboptimal hyperparameter settings severely constrained its performance. Future work could explore ways to mitigate these issues, such as using a more efficient model architecture, reducing the vocabulary size, or employing more powerful hardware to support training on larger batches and better memory management.

V. Conclusion

In this report, we developed and trained four different models for the task of signature recognition in Q1, and an LSTM model for word completion in Q2. Each approach provided valuable insights into the challenges and opportunities for improvement in these domains.

For Q1, we explored Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN), and two manual feature extraction methods—SIFT with SVM and HOG with SVM. The results revealed that traditional feature extraction methods combined with a robust classifier like SVM outperformed the deep learning models. The SIFT + SVM approach achieved the highest accuracy of 33.3%, followed closely by HOG + SVM with 31% accuracy. CNN and ANN, while powerful for larger datasets, struggled with the small size of our signature dataset, yielding lower accuracies of 1% and 1.32%, respectively. This highlights the need for either larger datasets or the use of transfer learning when applying deep learning methods to similar tasks.

For Q2, we implemented an LSTM model for the task of word completion using Shakespeare's plays dataset. Despite several optimizations, including learning rate scheduling and early stopping, the model underperformed, achieving a final accuracy of 3.45%. The small dataset and the need to process data in batches due to hardware limitations were major contributors to this underperformance. Additionally, the LSTM model struggled with generating coherent sentences, a limitation that could be addressed with a larger dataset or by utilizing pre-trained language models.

In summary, this work demonstrates the strengths of manual feature extraction techniques like SIFT and HOG when dealing with small, specialized datasets, while deep learning models like CNN and LSTM can excel with larger datasets or through transfer learning. For future work, improvements such as data augmentation, the use of more sophisticated architectures, and leveraging transfer learning for CNN and LSTM models could yield better results. Additionally, deploying models in real-time systems, such as the LSTM model for dynamic word prediction, can be further refined to improve both accuracy and user experience.

VI. Prompts

A. Q1: Signature Image Recognition using CNN

In this conversation, various aspects of data preprocessing, model architectures, and training across different approaches such as CNN, ANN, SIFT, and HOG were discussed. Since the direct links to the GPT chat cannot be shared, a summary of the prompts used in the discussion is provided below.

Data Preprocessing and Setup

- The dataset consisted of segmented signature images stored in subfolders named according to the ID of each individual.
- The dataset was split into train and test directories, with 3 images per ID in the train set and 1 in the test set.
- A script was requested to automate this split based on subfolder structure.

CNN (Convolutional Neural Network)

- The CNN architecture consisted of 3 convolutional layers, followed by max-pooling layers, and a dense layer with dropout for regularization.
- The model was trained using the Adam optimizer and categorical cross-entropy loss.
- Training involved early stopping, with the model stopping at epoch 5 due to overfitting, and the best performance was at epoch 2.
- The accuracy plateaued around 1%, and the model failed to generalize well.
- Confusion matrix, accuracy, and loss were visualized during the process.

ANN (Artificial Neural Network):

- The ANN architecture featured multiple fully connected layers, and a softmax output layer was used for classification.
- Early stopping was applied, and the model performed best at epoch 7 with an accuracy of 1.32%.
- After epoch 7, the model overfitted and failed to learn meaningful patterns.

SIFT (Scale-Invariant Feature Transform) + SVM

- SIFT was applied to extract keypoint-based features, which were averaged and fed into an SVM classifier.
- SIFT + SVM achieved the highest accuracy among the models, reaching 33.3% accuracy.
- The combination of SIFT and SVM worked best due to SIFT's ability to detect distinct local features.
- The confusion matrix, precision, recall, f-measure, and accuracy were visualized, though precision warnings occurred for classes with no predictions.

HOG (Histogram of Oriented Gradients) + Classifier

- HOG was used for feature extraction, capturing gradient-based information from the images.
- The classifier trained on HOG features achieved 31% accuracy, slightly lower than SIFT + SVM.
- Confusion matrix, training/testing errors, precision, recall, and f-measure were visualized for this approach.

Summary of Results

- SIFT + SVM was the best-performing approach, achieving 33.3% accuracy.
- HOG + classifier achieved 31% accuracy, following closely behind SIFT.
- Both CNN and ANN underperformed due to the small dataset size. CNN stopped early at epoch 5, while ANN overfitted after epoch 7.

Suggestions for Improvement

- For CNN and ANN, data augmentation and transfer learning were suggested to improve performance.
- For SIFT and HOG, fine-tuning the classifiers and using techniques such as Dense SIFT or spatial pyramid matching were recommended for enhanced feature extraction.

B. Q2: Word Completion Using LSTM

Here's a summary of the key prompts I asked throughout this conversation regarding LSTM model architecture, cleaning, trials and errors, preprocessing, RAM and GPU usage, accuracy, and results:

LSTM Model Architecture

- **Model Layers**: You inquired about building an LSTM model for next-word prediction with various architectures.
  - Input layer for sequence data.
  - Embedding layer for processing text input into dense vectors.
  - LSTM layer (e.g., with 100 or 150 units).
  - Dropout layer to prevent overfitting.
  - Dense layer with `softmax` activation for output prediction.
- **Model Training**: You explored different configurations, including batch sizes, dropout rates, optimizer settings, and gradient clipping to stabilize training.
- **Issues with Model Shape**: You encountered dimensional mismatch issues with the LSTM layer, which were resolved by fixing the sequence preprocessing and tensor shapes.

Cleaning and Preprocessing the Dataset

- **Cleaning Text**: You performed standard text cleaning steps, including removing punctuation, special characters, lowercasing the text, and removing stopwords.
- **Tokenization**: The text was tokenized using `Tokenizer` from Keras, converting the text into sequences of integers, with the tokenizer saved for later reuse during inference.
- **Sequence Padding**: You used `pad_sequences` to ensure uniform input sequence length for the LSTM model.

Trials and Errors

- **Batch Processing Issues**: You faced input dimension issues during batch processing, which were resolved by debugging and adjusting input shapes.
- **Multiple Epochs and Decreasing Accuracy**: You encountered decreasing accuracy during training and adjusted learning rate, dropout, and batch sizes to address this.
- **Early Stopping and Learning Rate Scheduling**: Early stopping and learning rate reduction helped prevent overfitting and stabilize training.

Preprocessing Techniques Used

- **Batching/Unbatching**: You used `unbatch()` and `batch()` from TensorFlow's `tf.data.Dataset` API to manage sequence splitting and batch processing.
- **One-Hot Encoding**: The target words were one-hot encoded for use with categorical cross-entropy loss.
- **Splitting Data**: The dataset was split into training and validation sets (typically 80/20).

RAM and GPU Usage

- **GPU and RAM Issues**: You faced memory issues during training on large datasets, which increased RAM usage (38.1% GPU usage at one point), causing memory optimization steps like batch size adjustments.
- **Model Size**: The model had over 6 million parameters, significantly impacting memory usage during training and inference.

Accuracy and Results

- **Initial Results**: The model achieved low accuracy (around 0.034) and low F1-scores (around 0.0073), with minimal improvement in validation accuracy.
- **Best Epoch**: The model's best performance occurred at epoch 27, with early stopping triggered at epoch 30. Despite accuracy issues, the model produced predictions, and the final model was saved.

Saving and Deployment

- **Model Saving**: The model was initially saved in `.h5` format but later switched to TensorFlow's `.keras` format to address compatibility issues.
- **Deployment**: You inquired about deploying the model locally via API using FastAPI, creating a Flask-based UI for real-time word predictions, and serving model predictions with FastAPI.

C. Report Generation

Here's a summary of the key prompts that were discussed during this conversation regarding report generation, model implementations, and results discussion:

Report Template

- How to create an IEEE conference paper report in LaTeX format.
- How to organize the introduction, methodology, and discussion sections for Q1 and Q2.
- Provide LaTeX code for sections such as results, discussion, and prompts.
- How to include GPT chat links and references in LaTeX.

Model Implementations

- Provide detailed LaTeX code for the architecture of CNN, ANN, HOG with SVM, and SIFT with SVM models based on the given code.
- Extract relevant model details (e.g., layers, parameters) from Python code to describe the architecture in LaTeX.
- How to include images (e.g., accuracy and loss graphs) for each model in LaTeX, both side by side and stacked vertically.

Results Discussion

- Discuss why the LSTM model underperformed in the word completion task, including details on accuracy, loss, and memory limitations.
- Explain possible improvements to the LSTM model for word completion.
- Discussion of results for Q1, including interpretation of performance for CNN, ANN, HOG with SVM, and SIFT with SVM, and recommendations for improvement.
- How to present prompts and provide GPT conversation links in a LaTeX report.

REFERENCES

[1] King Burrito666, "Shakespeare Plays Dataset," *Kaggle*, Available: https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays, Accessed: Sep. 29, 2024.

[2] I. Amjad, "Signature Dataset," *Google Drive*, Available: https://drive.google.com/drive/folders/1q9P42vw61SRdQEfBc9UgaxZctsb7rgSg?usp=drive_link, Accessed: Sep. 29, 2024.

[3] IEEE, "IEEE Conference Template," *Overleaf*, Available: https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn, Accessed: Sep. 29, 2024.