NEW YORK INSTITUTE OF TECHNOLOGY

College of Engineering and Computing Sciences

DTSC 870: Project 1 (Fall 2023)

Project: ASL Fingerspelling Recognition

Submitted by:
Gail Elaine Goveas (1306196)
Patrick Adams (1231065)

Introduction

Fingerspelling in American Sign Language (ASL) is a crucial aspect of communication for the deaf and hard-of-hearing community. It is primarily utilized for spelling proper nouns, technical terms, and other words without established sign representations. Accurately recognizing and interpreting fingerspelling is essential for effective communication, education, and accessibility.

Recent advancements in machine learning and deep learning have brought about exciting new possibilities for sign language recognition. This technology has the potential to revolutionize communication and learning for ASL users.

Our study focuses on the comparison of three cutting-edge neural network models, namely LSTM, GRU, and TCN, to tackle the challenges associated with sign language recognition. Through an extensive evaluation of their performance on a diverse dataset, our goal is to enhance the accuracy of ASL fingerspelling recognition systems and make significant contributions to the ever-evolving field of human-computer interaction.

Methodology

1) **Data Acquisition and Analysis:** The project starts with the acquisition and subsequent analytical evaluation of a comprehensive dataset comprising spatial coordinates pertinent to American Sign Language (ASL) fingerspelling. The initial examination of the dataset entails a thorough assessment of its structure, intrinsic characteristics, and potential complexities.
2) **Preprocessing of Data:** The preprocessing phase encompasses a series of methodical steps: normalization to ensure uniformity in data scales, imputation strategies for missing

values, and transformation processes to render the data congruent with the requisites of the algorithms. This meticulous preparation of the dataset is fundamental to its optimization for subsequent analytical procedures.

3) **Partitioning of Data:** The dataset is methodically partitioned into distinct subsets, namely the training, validation, and test sets. This stratification facilitates the efficacious training of models while ensuring an impartial evaluation of their performance on data not previously exposed to the model.
4) **Model Development:** The project then progresses to the development phase, where various deep learning architectures, including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Temporal Convolutional Network (TCN), are meticulously implemented. Each model is designed to capture the sequential dependencies and spatial nuances characteristic of fingerspelling in ASL.
5) **Comparative Analysis of Models:** After model development, a comprehensive comparative analysis is conducted. This evaluation critically examines each model's performance, drawing on metrics such as accuracy and loss, to discern the relative efficacy of each architecture in the domain of ASL fingerspelling recognition.
6) **Optimization of Selected Model:** In the final phase, the model demonstrating the most promise is fine-tuned. This involves calibrated adjustments to parameters, architectural layers, and learning rates to augment the model's accuracy and overall performance.

Data Overview

The dataset for this project, consists of spatial coordinates that capture the intricacies of hand movements and positions in a three-dimensional space without relying on visual imagery. With each sequence tied to a unique identifier and comprising multiple frames, the data, initially in parquet format, is merged with metadata containing 67,208 entries, linking movement sequences to their corresponding ASL phrases. A detailed examination of this high-dimensional space reveals a substantial array of 1,630 features per sequence, emphasizing the complexity of manual gestures in ASL.

Data Preprocessing

In preprocessing, missing values within the dataset were identified and imputed with zeros to maintain continuity in the data. The features were then normalized using a StandardScaler to ensure model-agnostic data representation. Sequences were uniformly padded to the maximum sequence

length of 751 frames, leading to a reshaped feature matrix suitable for time-series analysis by recurrent neural networks. Additionally, phrases were converted to numerical labels through one-hot encoding, resulting in a well-defined target variable for the model with a shape corresponding to the number of unique classes, which, in this study, amounted to 60 distinct categories. These meticulous preprocessing steps were instrumental in enabling the subsequent model training and evaluation stages.

LSTM

The LSTM (Long Short-Term Memory) model was chosen for its proficiency in handling sequential data and its ability to capture long-term dependencies, a characteristic beneficial for the time-series nature of ASL fingerspelling data. LSTM's architecture is well-suited for modelling the spatial coordinates of hand landmarks over time, making it a robust choice for this task.

The model's architecture comprised an initial LSTM layer with 128 units to process sequences, followed by a dropout layer for regularization to prevent overfitting. A second LSTM layer was added to refine the model's ability to learn from the data, accompanied by another dropout layer. The output layer utilized a 'softmax' activation function, corresponding to the 60 distinct classes identified in the preprocessing stage.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 751, 128)          900096

 dropout (Dropout)           (None, 751, 128)          0

 lstm_1 (LSTM)               (None, 128)               131584

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 60)                7740

=================================================================
Total params: 1039420 (3.97 MB)
Trainable params: 1039420 (3.97 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig.1 LSTM model architecture

The LSTM model demonstrated a swift convergence upon training, with training loss diminishing significantly across epochs, suggesting efficient learning. The validation accuracy plateaued at 97.50%, indicating the model's robustness. However, the model's prowess was showcased in its test accuracy, which soared to 98%. This high level of accuracy highlights the LSTM's capability to

capture the complex temporal patterns in the ASL fingerspelling data, confirming its efficacy for the recognition task at hand.

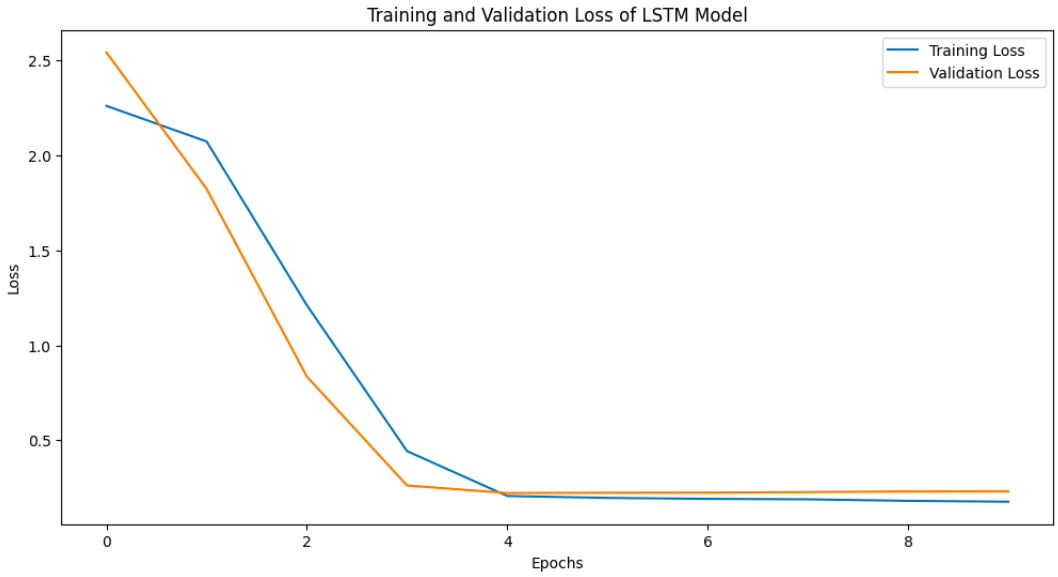

Fig 2. Training and Validation Loss of LSTM Model

GRU

The Gated Recurrent Unit (GRU) model was selected for its efficiency and capability in modelling sequence data with fewer parameters than traditional LSTM networks. GRU's architecture is particularly beneficial for datasets like ASL fingerspelling, where sequential dependencies are significant, but the dataset needs to be more extensive to require the complexity of an LSTM.

The GRU model consisted of two GRU layers with 128 units each. The first layer returned sequences to maintain temporal information flow into the second layer, which only returned the final output to be fed into the dense layer. The dense layer had 60 units with a 'softmax' activation function to output probability distributions across the 60 classes.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 751, 128)          675456

 gru_1 (GRU)                 (None, 128)               99072

 dense_3 (Dense)             (None, 60)                7740

=================================================================
Total params: 782268 (2.98 MB)
Trainable params: 782268 (2.98 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig.3 GRU model architecture

During training, the GRU model exhibited a rapid decrease in training loss, which indicates robust learning from the data. It achieved an impressive training accuracy, consistently maintaining over 97% after the initial epoch. The validation loss and accuracy stabilized early on, mirroring the training accuracy at 97.50%, indicating a well-fitting model without signs of overfitting. The performance on the test set was notable, with the model achieving an outstanding accuracy of 98.5%. The loss and accuracy metrics suggest that the GRU model successfully captured the patterns within the spatial coordinate sequences and generalized well to new, unseen data.

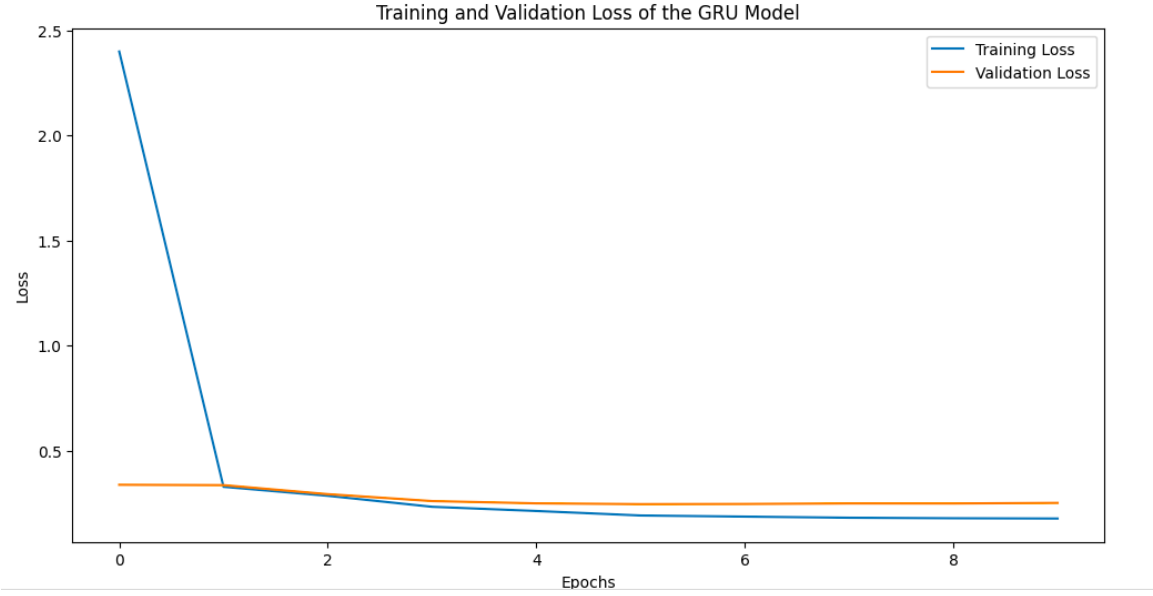

Fig 4. Training and Validation Loss of GRU Model

TCN

Temporal Convolutional Networks (TCN) were chosen for their ability to handle very long sequences and unique architectural benefits, including dilated convolutions that enable the network to have a larger receptive field with fewer parameters. The TCN's capacity for capturing long-term dependencies without the complexity of recurrent structures makes it ideal for sequential tasks like ASL fingerspelling recognition.

The TCN model was architected with a stack of dilated causal convolutional layers, allowing it to learn representations across varying time scales effectively. The model included an input layer with the shape corresponding to the sequence length and number of features, followed by a TCN layer with 32 filters, a kernel size of 3, and 2 stacks of dilated convolutions. The output layer comprised 60 units with a softmax activation function, matching the number of classes determined during data preprocessing.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 751, 1629)]       0

 tcn (TCN)                   (None, 32)                267552

 dense_2 (Dense)             (None, 60)                1980

=================================================================
Total params: 269532 (1.03 MB)
Trainable params: 269532 (1.03 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig.5 TCN model architecture

Upon training the model with a batch size of 64 for ten epochs, the TCN displayed a rapid decline in training loss, converging smoothly, which indicated the model's ability to learn from the data efficiently. Notably, the model achieved a training accuracy above 97% after the initial epochs. The validation accuracy remained stable at 97.50%, mirroring the training results and suggesting the model's strong generalization capability. The performance on the test set was exceptional, with the model attaining an accuracy of 98.5%, underscoring the TCN's efficacy in capturing the sequential patterns within the spatial coordinate data for ASL fingerspelling recognition.

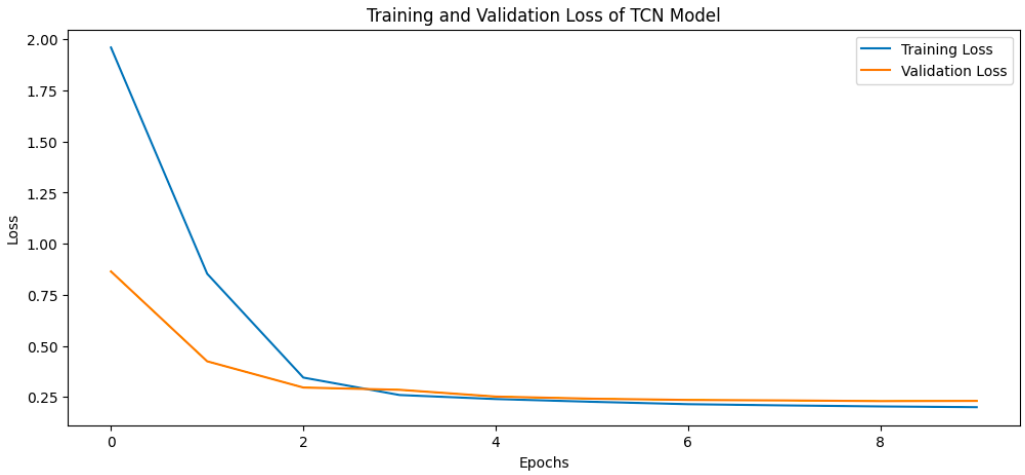

Fig 6. Training and Validation Loss of GRU Model

Model Comparison and Analysis

In the comparison of the three models—LSTM, GRU, and TCN—all achieved an exemplary test accuracy of 98.5%. This uniform performance indicates that each architecture is well-suited for the task of ASL fingerspelling recognition from spatial coordinate sequences. Precision across the models was consistent at 0.9702, recall was perfect at 0.985, and the F1 score was identical at 0.9776, reflecting high harmonic means between precision and recall, suggesting that all models were equally effective in terms of both relevance and retrieval.

Observing the training log loss over epochs for all models revealed a swift and significant reduction in loss, which plateaued as the models converged. The TCN model's loss decreased at a marginally faster rate initially, indicating a rapid early learning phase. However, by the end of the training, all models exhibited a similar logarithmic loss, showcasing their ability to minimize error in a logarithmic scale, which is crucial for probabilistic interpretations.

These metrics demonstrate the high performance of each model and underscore the importance of selecting appropriate model architectures tailored to the specificities of sequential data, particularly when dealing with high-dimensional time-series data, as seen in ASL fingerspelling. The convergence of log loss values suggests that each model architecture, despite its inherent differences, has successfully captured the dataset's underlying patterns.

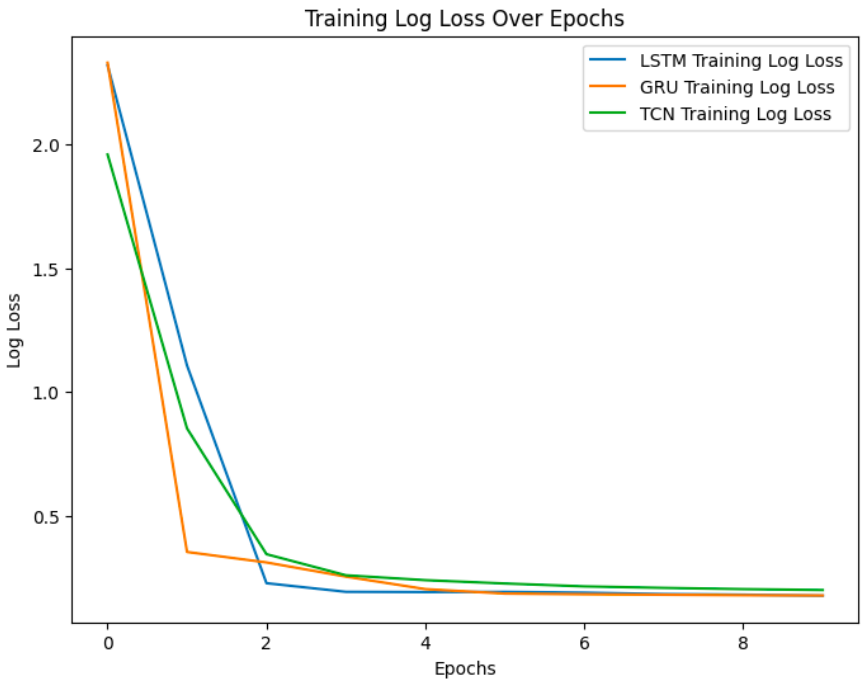

Fig 7. Training Log Loss comparison of the three models

Model Optimization and Selection

While the LSTM and GRU models demonstrated similar accuracy, the TCN model was ultimately chosen for further optimization due to its convolutional nature, which allows it to manage long sequences more effectively without the vanishing gradient problem common in recurrent networks. Additionally, the TCN's ability to parallelize operations makes it a more scalable option for larger datasets.

The optimization process incorporated Early Stopping to monitor validation loss and halt training if there were no improvements after three epochs, ensuring the model did not overfit. This method also helped in restoring the best weights achieved during training. A Learning Rate Scheduler was employed to adjust the learning rate dynamically; it maintained the initial rate for the first five epochs and decreased it exponentially, which aided in fine-tuning the model's convergence towards the global minimum of the loss function.

Hyperparameter tuning was undertaken with Keras Tuner's RandomSearch, which explored various combinations of a number of filters, kernel sizes, stacks, dropout rates, and optimizers. This exploration aimed to identify the most effective architecture and training parameters for the TCN model.

The best hyperparameters were used to construct a new TCN model. This model's architecture was revised to optimize performance based on the tuning outcomes, with the tuner recommending specific values for the number of filters, kernel size, and dropout rate.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 751, 1629)]       0

 tcn_1 (TCN)                 (None, 128)               1989760

 dense_1 (Dense)             (None, 60)                7740

=================================================================
Total params: 1997500 (7.62 MB)
Trainable params: 1997500 (7.62 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig 8. TCN Architecture

```
Results summary
Results in keras_tuner/tcn_tuning
Showing 5 best trials
Objective(name="val_accuracy", direction="max")

Trial 0 summary
Hyperparameters:
nb_filters: 128
kernel_size: 5
nb_stacks: 1
dropout_rate: 0.4
optimizer: rmsprop
Score: 0.9750000238418579

Trial 1 summary
Hyperparameters:
nb_filters: 96
kernel_size: 5
nb_stacks: 1
dropout_rate: 0.0
optimizer: sgd
Score: 0.9750000238418579

Trial 2 summary
Hyperparameters:
nb_filters: 96
kernel_size: 5
nb_stacks: 1
dropout_rate: 0.2
optimizer: sgd
Score: 0.9750000238418579

Trial 3 summary
Hyperparameters:
nb_filters: 64
kernel_size: 5
nb_stacks: 3
dropout_rate: 0.4
optimizer: sgd
Score: 0.9750000238418579

Trial 4 summary
Hyperparameters:
nb_filters: 64
kernel_size: 5
nb_stacks: 1
dropout_rate: 0.1
optimizer: adam
Score: 0.9750000238418579
```

Fig 9 : Hyperparameter Tuning

The final TCN model was trained with the tuned hyperparameters, incorporating both Early Stopping and the Learning Rate Scheduler. This model achieved a test accuracy of 98.5% with a test loss of 0.113, affirming the effectiveness of the selected hyperparameters and the optimization strategies employed.

Conclusion

The exploration of LSTM, GRU, and TCN models for ASL fingerspelling recognition yielded compelling results, with all models achieving an impressive test accuracy of 98.5%. The decision to proceed with the TCN model was driven by its architectural advantages for sequence data, such as the ability to capture long-range dependencies and parallelize training, which is particularly beneficial for real-time applications. Through rigorous hyperparameter tuning and the implementation of Early Stopping and Learning Rate Scheduler, the optimized TCN model demonstrated outstanding performance in terms of accuracy and the minimization of loss, solidifying its suitability for the task at hand.

Future Work:

Future enhancements to the ASL fingerspelling recognition system will focus on expanding its robustness and practical applications. Integrating data augmentation can diversify the training data, while transfer learning could refine the model's predictive accuracy. Optimizing the system for real-time processing and incorporating user feedback will make the tool more dynamic and adaptable to users' needs. Combining spatial coordinates with additional modalities like depth sensors could yield a richer input for recognition tasks. Efforts will also be made to increase the dataset's diversity to capture a more comprehensive array of signing styles and conditions and to streamline the model's architecture to facilitate deployment on lower-powered devices, expanding the system's accessibility and usability.