IMPERIAL

Hand-writing Recognition With IMU and Force Sensors

NeverLatex Final Presentation

Antonio Tarizzo
Tuna Kisaaga
Fajar Kenichi Kusumah Putra
21/03/2025

Team

Antonio Tarizzo

Tuna Kısaağa

Fajar Kenichi Kusumah Putra

Agenda

1 Introduction
2 Literature Review
3 Hardware Design
4 Methodology
5 Experimental Results
6 Workplan
7 Conclusion
8 Video
9 References

Introduction

Scope and Objectives, Achievements, Background and Context

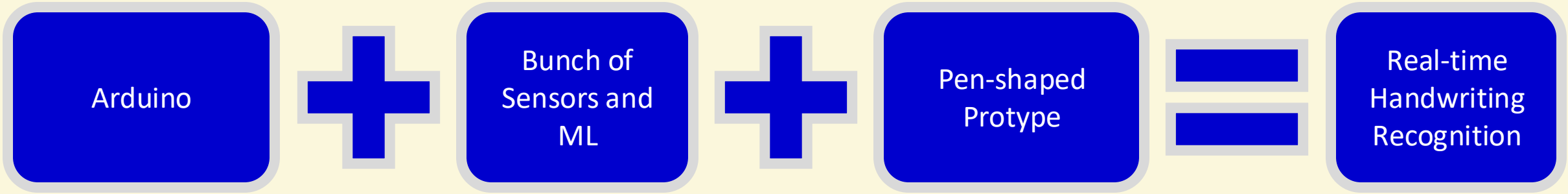
Background and Context

In General

This project proposes developing a **sensor-equipped handwriting recognition system** that **converts handwritten text into LaTeX format in real-time**.

Using Arduino, a pen with IMU (Inertial Measurement Units), and force strip sensors, we aim to provide a low-cost solution that eliminates the time spent to transfer information from paper to digital documents.

The project leverages machine learning to ensure high accuracy in detecting handwritten digits, text, equations and more.



Scope, Objectives and Achievements

Scope

Develop a handwriting recognition system that converts handwritten text (letters, digits) and mathematical notation into LaTeX.

Objectives

Objectives

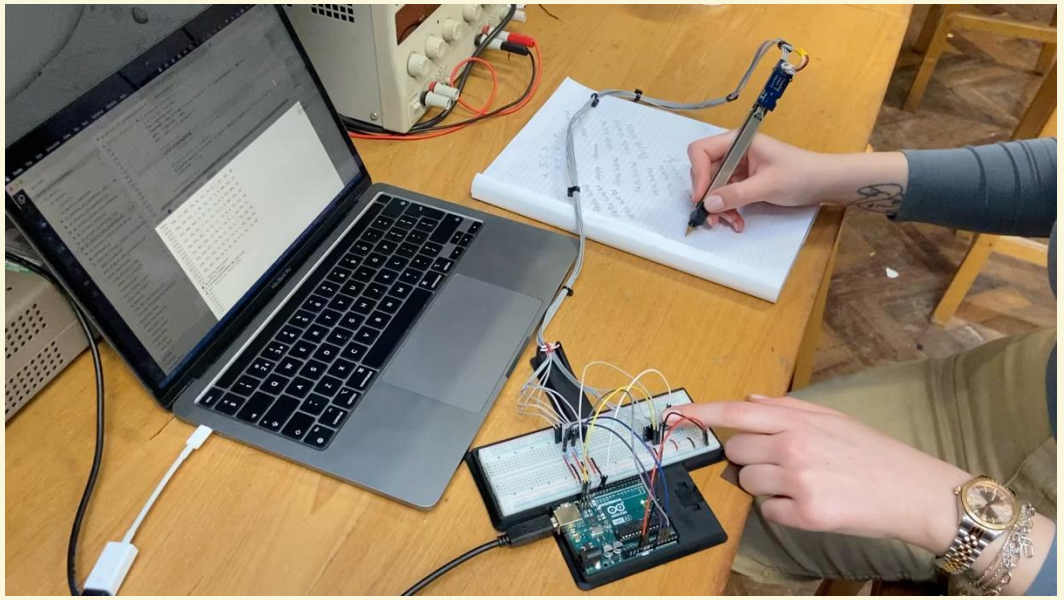
1.Handwriting Recognition: Accurately identify digits, letters (including case), and mathematical expressions.

2.LaTeX Document Generation: Automatically produce formatted LaTeX documents with sections, subsections, bulleted lists, and math environments.

3.Adaptive Learning & Performance: Improve recognition accuracy and speed over time, aiming for a 10% accuracy boost and 30% speed increase in Phase 2.

4.Erasure & Correction: Detect and reflect erasures in real time, with potential use of LLMs for post-recognition correction.

Achievements



Accuracy Achieved: *68.78%*

Literature Review

Reviewed Papers and Sources

Literature Review Key Findings

Things To Note

Input Data:

- Spatial data from IMU Sensors [6]
- Force data from Force Sensors [7]

Model Types:

- CNN [5]
- LSTM [7]
- CLDNN [5]

Loss Functions:

- Categorical Cross Entropy [3]
- Connectionist Temporal Classification (CTC) [5]

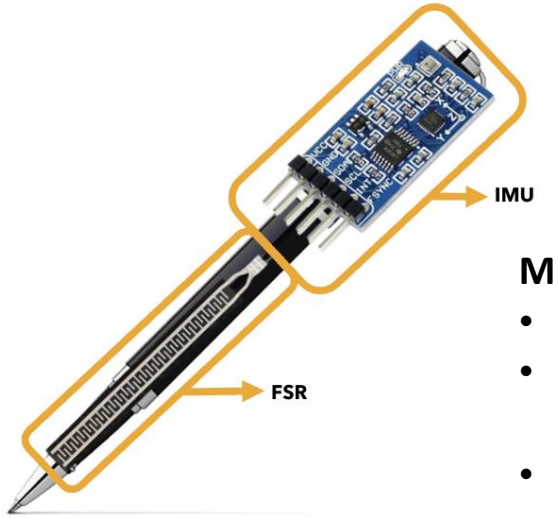
Action Based on Findings

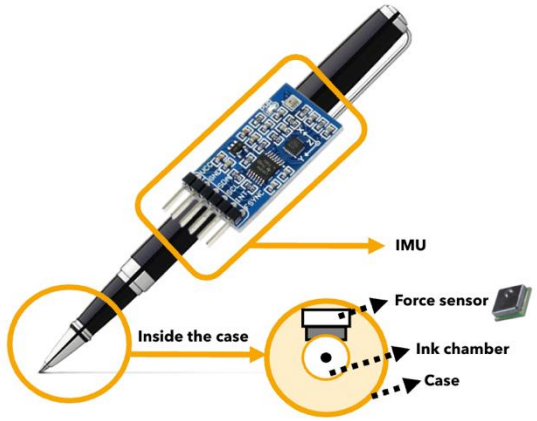
1. Implement a CNN and CLDNN based model on ONHW dataset.
2. Finetune the model on that dataset.
3. Create initial hardware prototype.
4. Collect data using the prototype.
5. Overall test (model, prototype, data collected).
6. Crosscheck results.
7. Repeat the process using the final product.

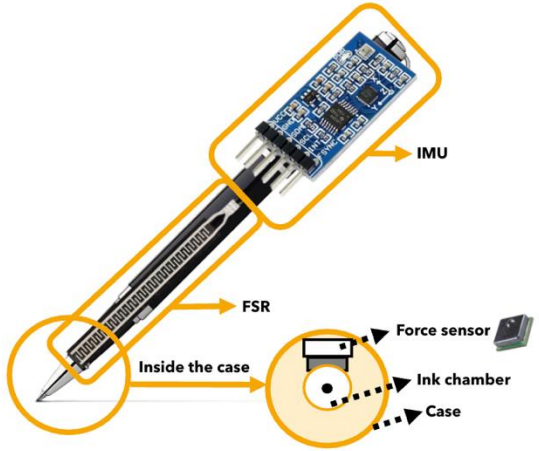
Hardware Design

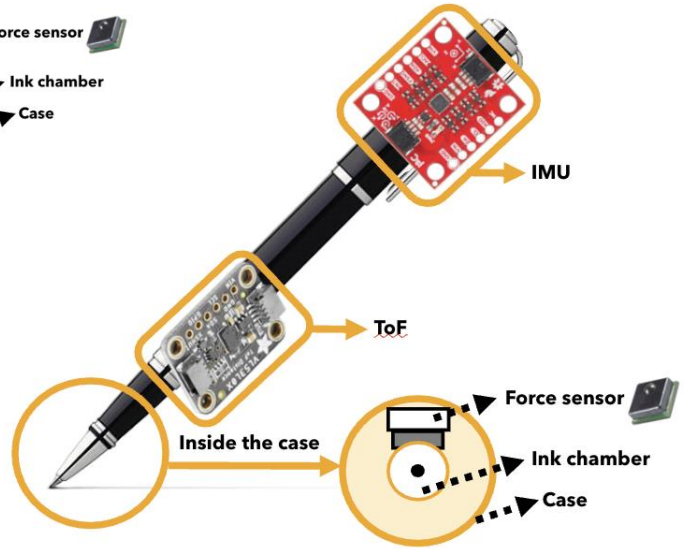
Design Choices, Challenges and Constraints

Considerations of Hardware Design



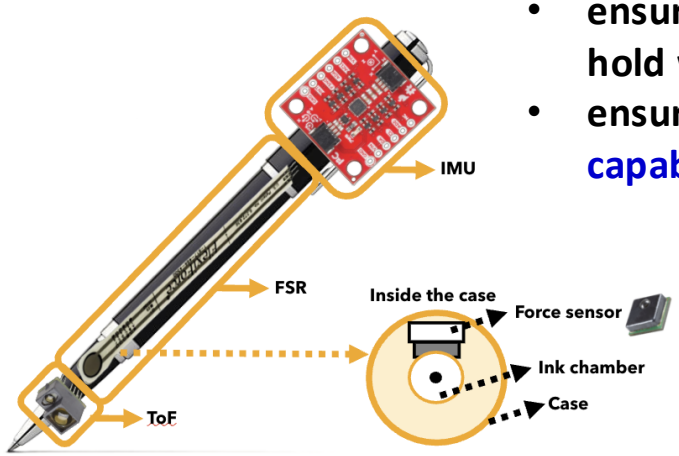


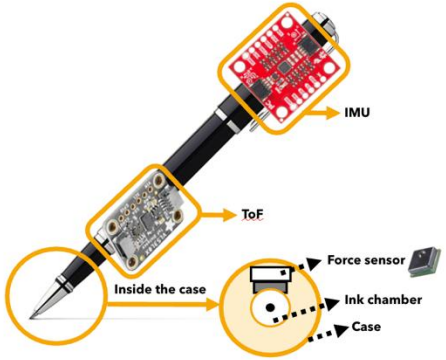


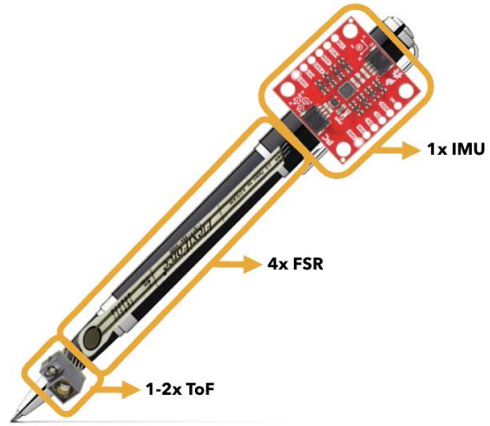


Multiple iterations of hardware design to ensure that we

- **stay within budget,**
- **avoid unnecessary complexity by eliminating non-value-adding components,**
- **select the most suitable sensors for our handwriting recognition task.**
- **ensure the sensors' sensitivity levels are accurate,**
- **ensure the pen remains comfortable and practical for the writer to hold while writing,**
- **ensure the overall design remains within the scope of our technical capabilities to successfully build and implement it.**



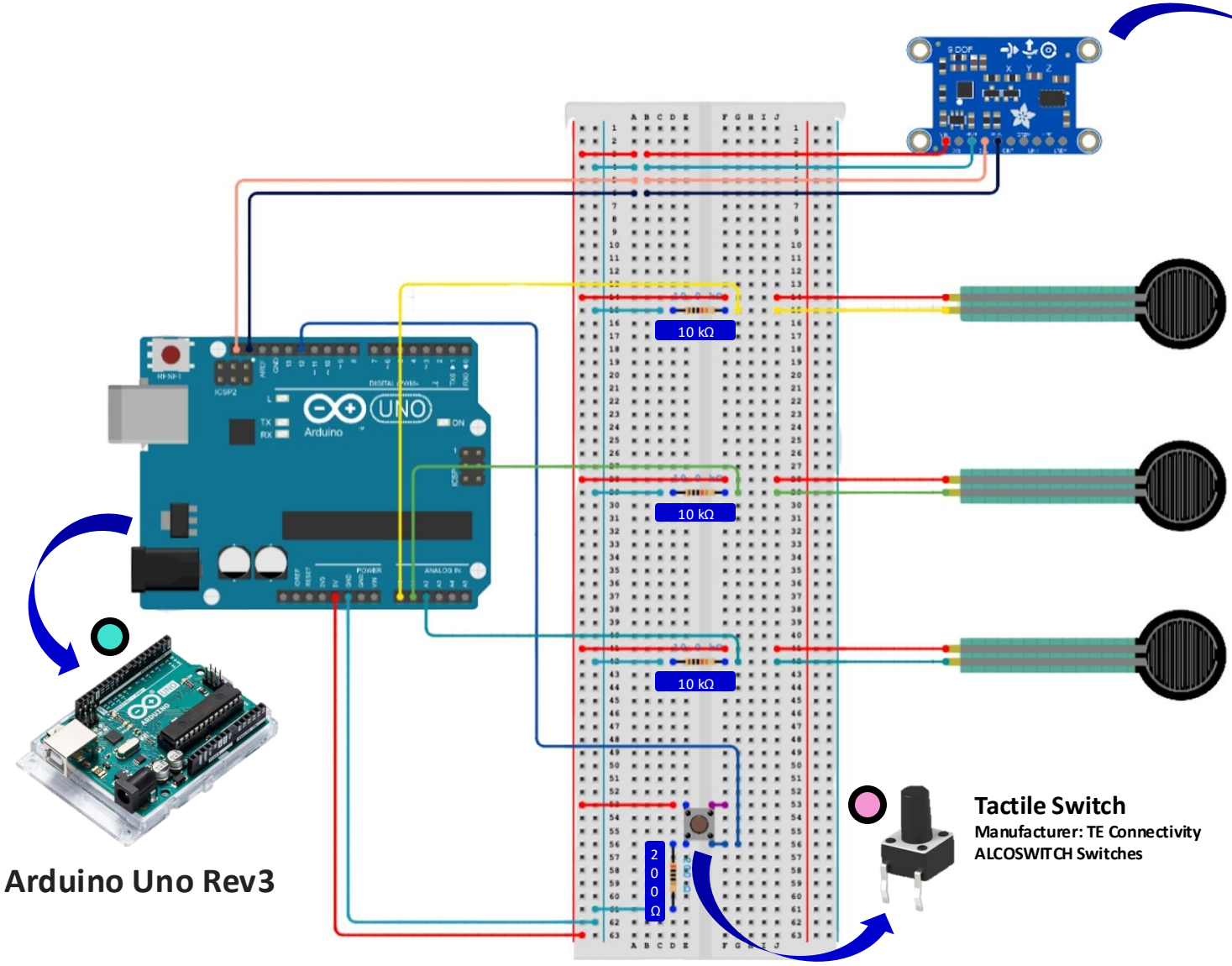




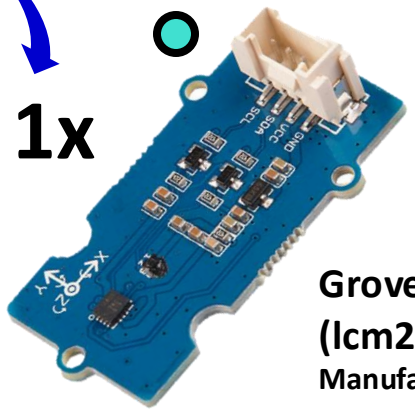
Electronic Design Choices

Data Collection, Training & Real-time Prediction

Data Collection Only



1x

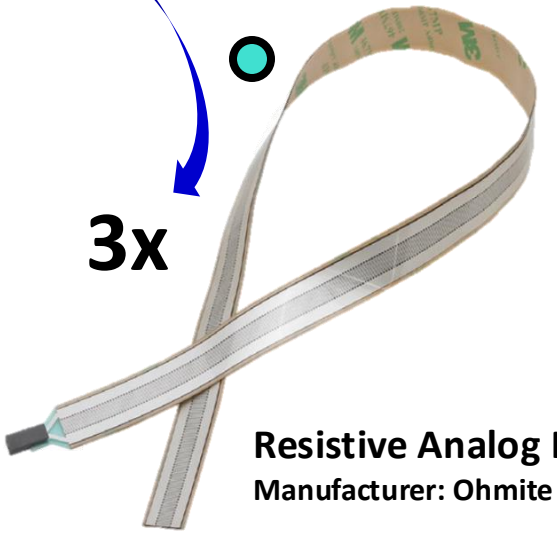


Grove - IMU 9DOF (lcm20600+AK09918)
Manufacturer: Seeed Technology Co., Ltd

Arduino Uno Rev3

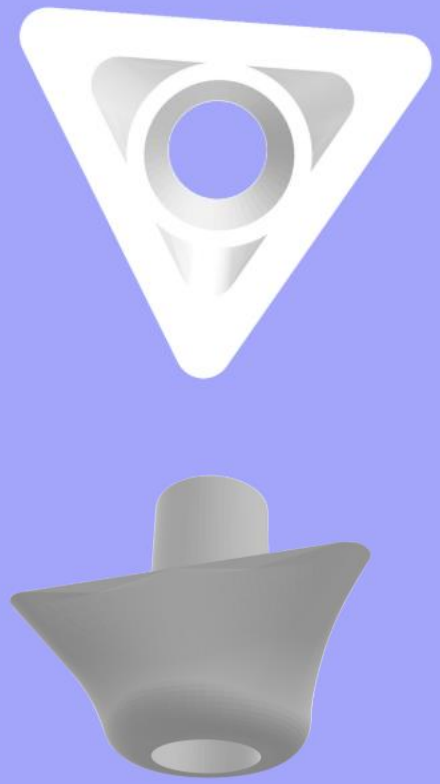
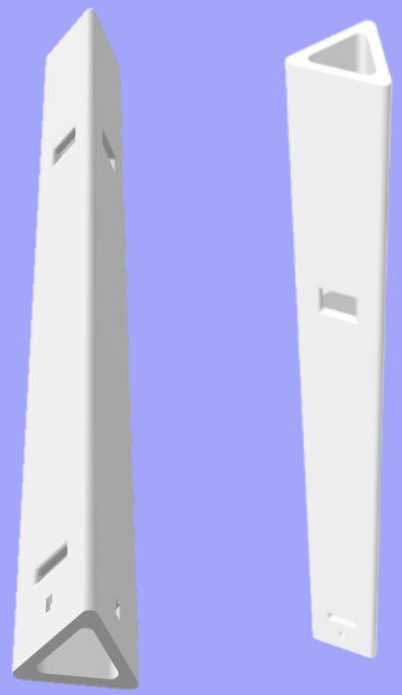
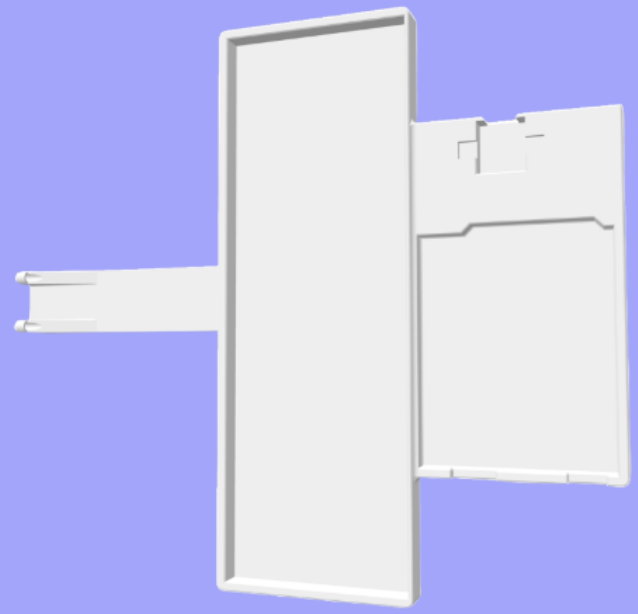
Tactile Switch
Manufacturer: TE Connectivity
ALCOSWITCH Switches

3x

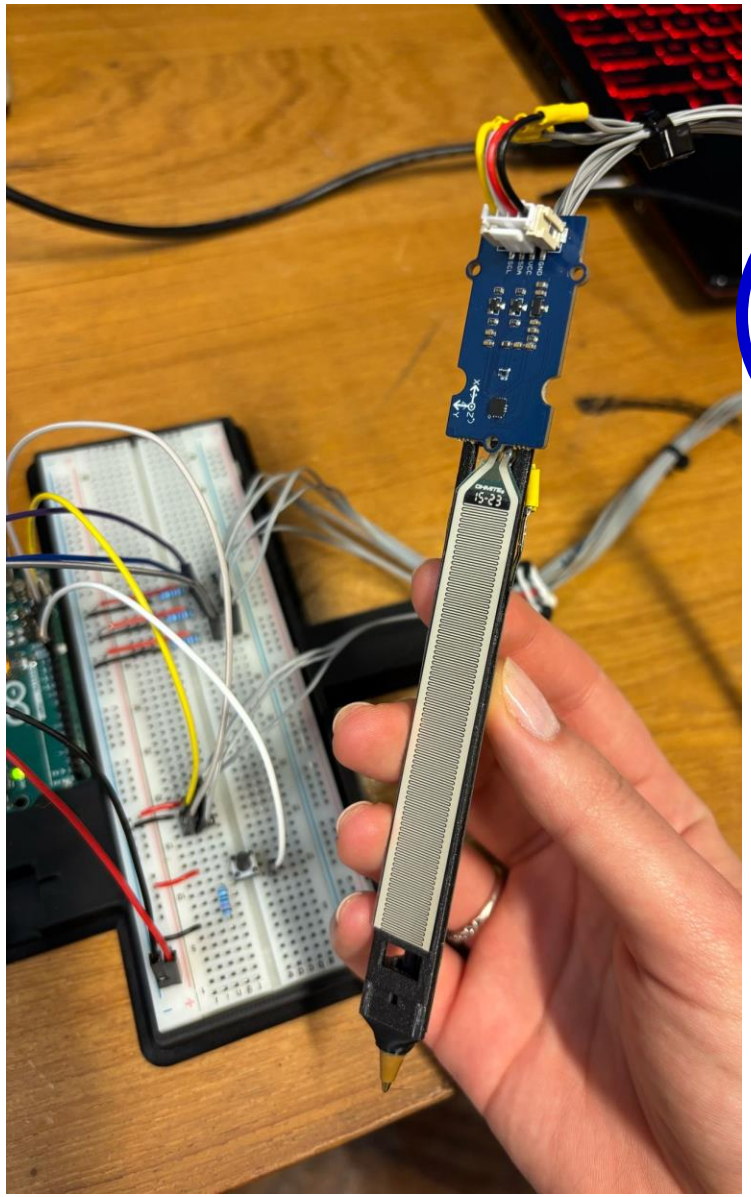


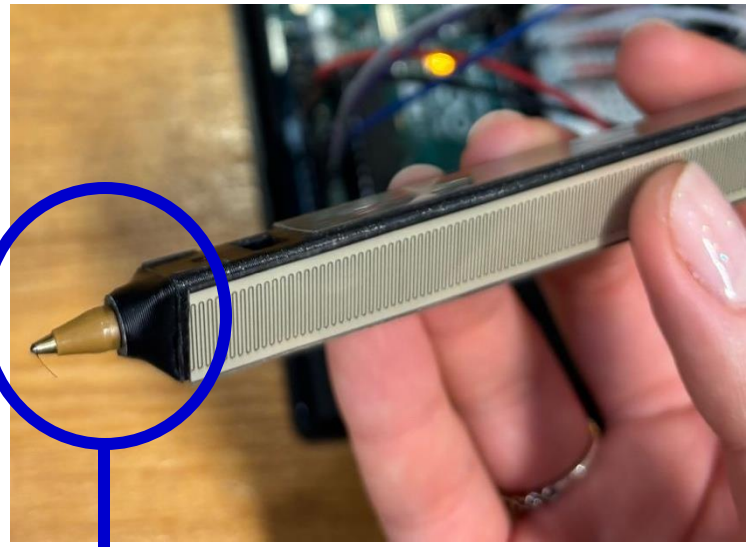
Resistive Analog Force Sensor
Manufacturer: Ohmite

3D Printing Design

Pen Tip (for the Cartridge)	Pen Body	Arduino and Breadboard Holder
		

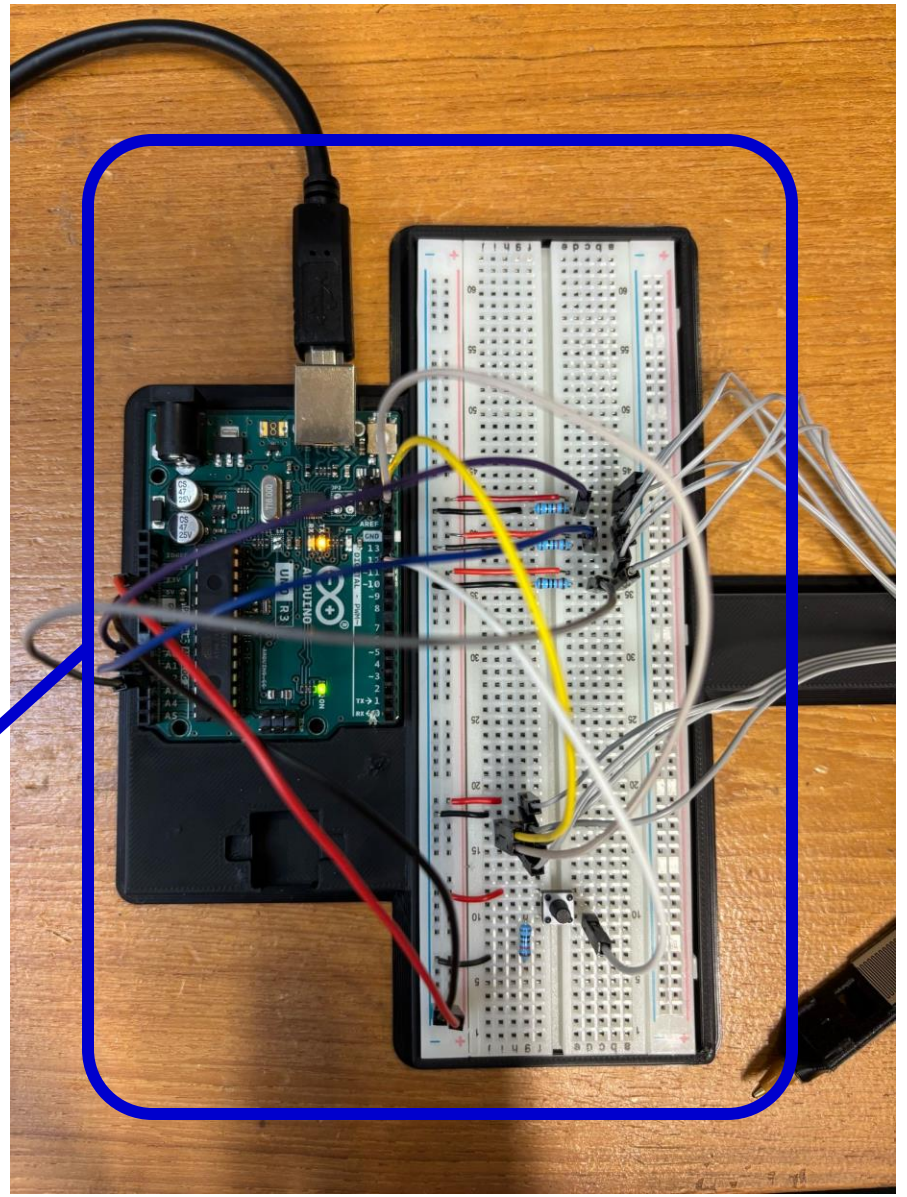
Final Design





Pen cartridge

Arduino and breadhboard holder to immobilise the writing setup



Methodology

Data Collection, Preprocessing, Models and Methods, Real-time Prediction

Data Collection

Target characters, included in data collection

Only data collection

```python
# English Alphabet (Upper and Lower Case)
alphabet = list('ABCDEFGHIJKLMNOPQRSTUVWXYZ') + list('abcdefghijklmnopqrstuvwxyz')

# Digits
digits = list('0123456789')

# Basic Mathematical Symbols
math_symbols = list('+-÷=≠≈<≤>≥±∓∞∑∏√∛∜∫∮∂∇Δπθλμω')

# Special Characters
special_chars = list("!@#%^&*()[]{}|;:'\",.?/~`")

# Logical & Set Theory Symbols
logical_symbols = list('∀∃∈∉∋')

# Currency Symbols
currency_symbols = ['$', '€', '£']

# Words
random_words = [
    'apple', 'banana', 'cherry', 'dog', 'elephant', 'fish', 'grape', 'house', 'ice',
'jungle', 'kite', 'lemon', 'mountain', 'night', 'orange', 'pizza', 'queen', 'river', 'sun', 'tree',
    'umbrella', 'violin', 'water', 'xylophone', 'yacht', 'zebra'
]

# Generate random short phrases
phrases = [
    'Hello World', 'Good Morning', 'Data Science', 'Machine Learning',
    'Time Series', 'Artificial Intelligence', 'Neural Networks',
    'Deep Learning', 'Pattern Recognition', 'Handwriting Analysis',
    'Blue Car', 'Green Dog', 'Yellow Tree', 'Purple Glass', 'Black Phone'
]
```

English Alphabet
Upper and Lower Case Letters

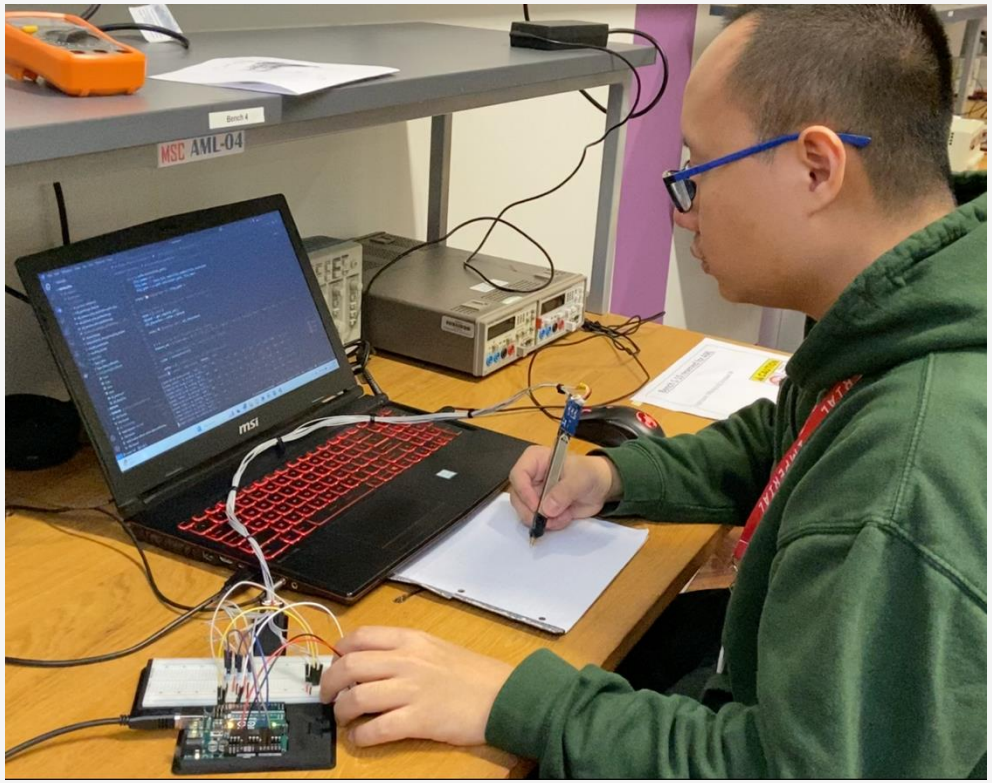
Digits

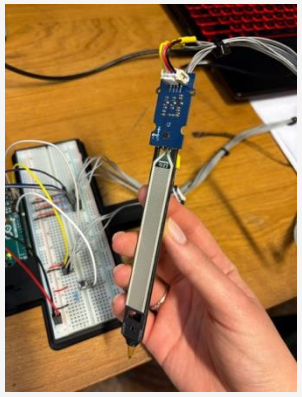
Maths Symbols

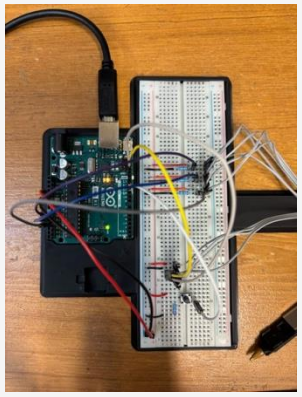
Special Characters

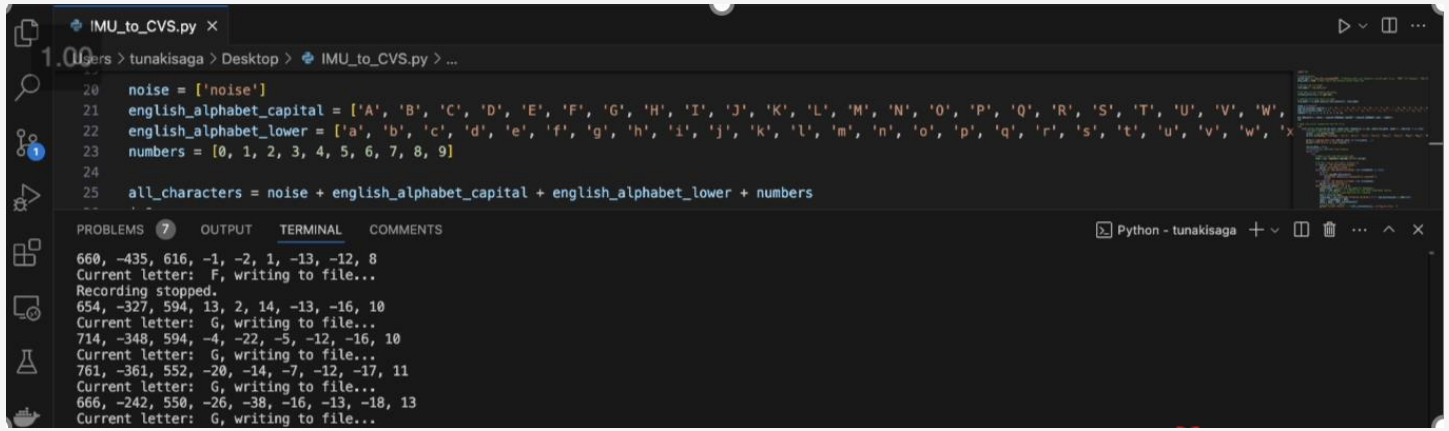
Words & Phrases

Data Collection









Dataset Collection Protocol

- Use the final sensor-equipped pen and normal white paper on a flat, **horizontal surface**.
- Participants should sit comfortably at a table and maintain a consistent **pen orientation (IMU facing upwards)**.
- All English letters (uppercase and lowercase), digits, special characters and mathematical symbols should be included in the target label set.
- Each written character should be immediately **labeled** to ensure accurate ground truth.
- Each character should be written **naturally**, without strict size constraints.
- **Thresholding** should be applied to trim unnecessary sensor data at the start/end of strokes.
- The sensor data, timestamps, and labels should be **stored** in a structured format (e.g., CSV).

Data Preprocessing

1. Remove non-writing data

Filter out non-writing sequences of data to ensure clean input signals for the CLDNN model. Non-writing sequences were determined using the force readings.

```
# Filter out noise
filtered_data = combined_data[
    combined_data["Letter"] != "noise"]
```

2. Scale the data

Use StandardScaler to normalize IMU and force sensor readings to zero mean and unit variance, ensuring stable training dynamics in the CLDNN model.

```
# Normalize sensor data (Z-score normalization)
scaler = StandardScaler()
filtered_data[sensor_columns] = scaler.fit_transform(
    filtered_data[sensor_columns])
```

3. Create sequences

Convert raw time-series data into sequences of sensor data and label the sequences.

```
for _, row in data.iterrows():
    if row["Letter"] != current_label:
        # Save the current sequence if it's not empty
        if current_sequence:
            sequences.append(np.array(current_sequence))
            labels.append(current_label)
        # Start a new sequence
        current_label = row["Letter"]
        current_sequence = []

    # Append sensor values to the current sequence
    current_sequence.append(row[sensor_columns].values)
```

4. Map characters

Utilize a StringLookup layer to convert character labels into integer-encoded sequences for the CTC loss function.

```
# Define a blank token
blank_token = 'BLANK'

# Update the StringLookup layers
char_to_num = layers.StringLookup(
    vocabulary=list(characters),
    mask_token=None,
    oov_token=blank_token
)
```

5. Introduce the blank token

Insert a dedicated blank index required by CTC to handle spacing and transitions between characters. Use ‘BLANK’ as the blank token.

```
# Define a blank token
blank_token = 'BLANK'
```

6. Pad labels and sequences

Apply padding to sensor data sequences and corresponding labels to ensure uniform input size before feeding into the CLDNN model.

```
# Pad sequences to the same length
padded_sequences = pad_sequences(
    filtered_sequences,
    maxlen=max_length,
    padding='post',
    dtype='float32',
    value=0)
```

7. Filter the sequences

Remove sequences that are shorter than the minimum length to maintain dataset consistency.

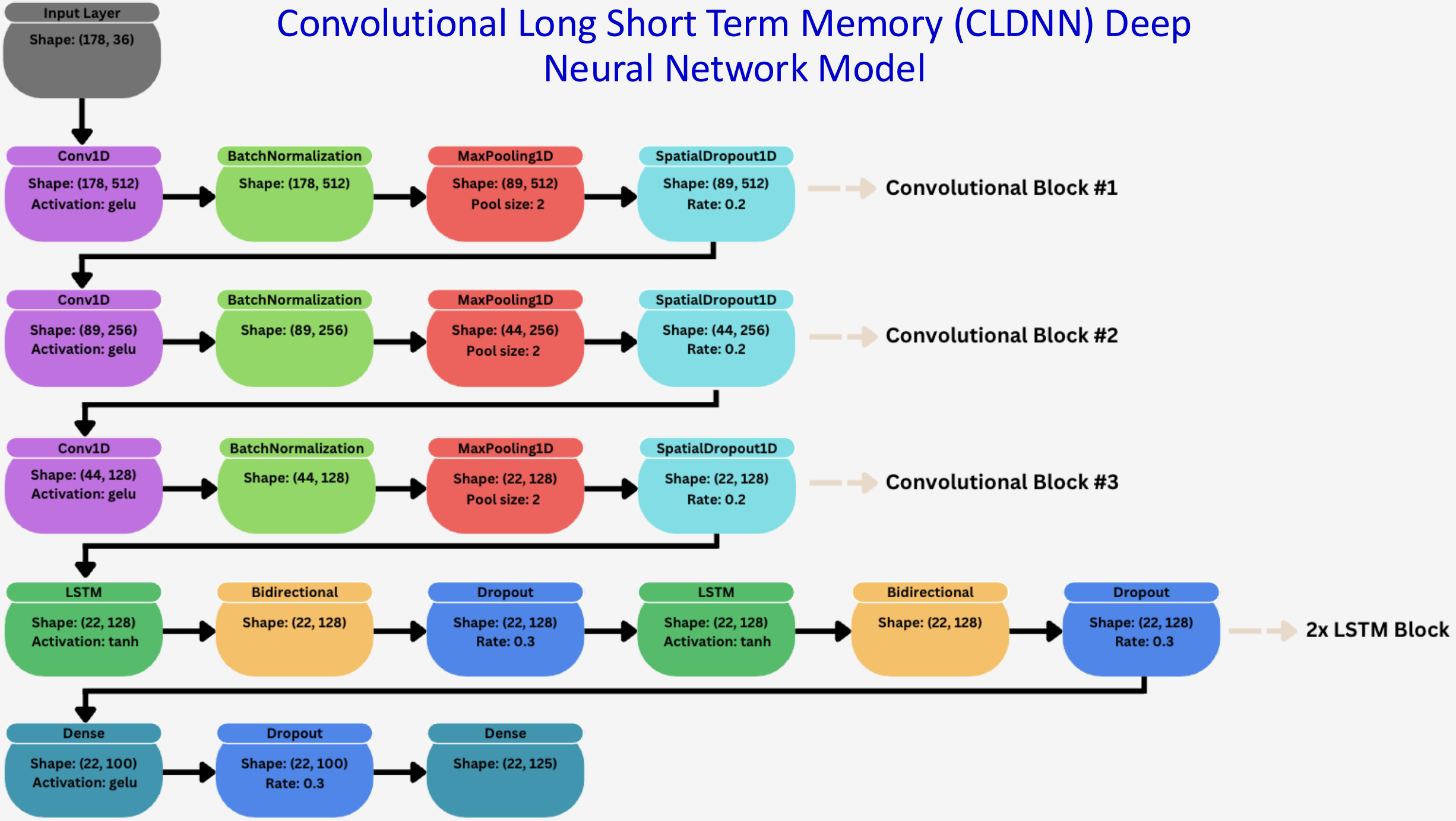
```
# Filter out very short sequences
filtered_sequences = [seq for seq in sequences
                      if seq.shape[0] >= min_length]
filtered_labels = [label for seq, label in zip(
    sequences, padded_labels) if
                   seq.shape[0] >= min_length]
```

8. Split the data

Partition the processed dataset into train, test and validation sets using the designated function. The dataset is split into 70% training, 20% validation, and 10% test sets..

```
# Apply the fixed split_data function
X_train, X_test, y_train, y_test = split_data(
    padded_sequences_tensor,
    filtered_labels)
```

Convolutional Long Short Term Memory (CLDNN) Deep Neural Network Model



Model

Pretraining

STABILO The OnHW Datasets

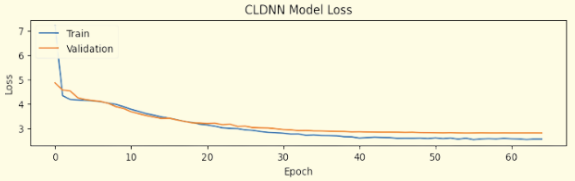
- Front accelerometer (STM LSM6DSL)
- Gyroscope (STM LSM6DSL)
- Rear accelerometer (Freescale MMA8451Q)
- Magnetometer (ALPS HSCDTD008A)
- Force sensor (ALPS HSFPAR003A)

1. Train the model with the OnHW dataset, only including the IMU sensor data [6]
2. When training the model again with our data, transfer the weights from the first model to the IMU-related layers of the new model

Very Small Set of Data

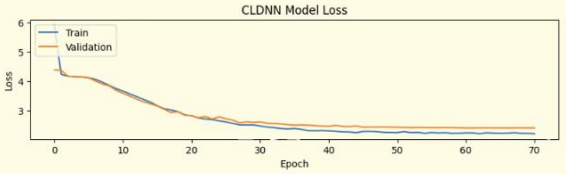
Without Pretraining

Sequence Accuracy for CLDNN: 18.41%



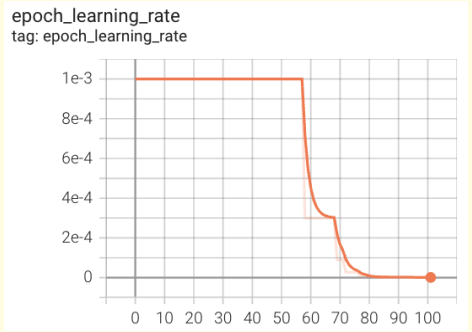
With Pretraining

Sequence Accuracy for CLDNN: 29.06%



Learning rate scheduling

Adjusts the learning rate dynamically during training (using ReduceLROnPlateau) to optimize convergence and prevent overshooting.



Early stopper

Monitors validation loss and halts training if no improvement is observed for a set number of epochs, preventing overfitting.

```
Epoch 81/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 89ms/step - loss: 5.5545 - val_loss: 8.8750
Epoch 82/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 91ms/step - loss: 5.4869 - val_loss: 8.8620
Epoch 83/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 106ms/step - loss: 5.5524 - val_loss: 8.8595
Epoch 84/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 92ms/step - loss: 5.4954 - val_loss: 8.8613
Epoch 85/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 91ms/step - loss: 5.5793 - val_loss: 8.8739
Epoch 86/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 91ms/step - loss: 5.4945 - val_loss: 8.8721
Epoch 87/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 89ms/step - loss: 5.6217 - val_loss: 8.8716
Epoch 88/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 89ms/step - loss: 5.5147 - val_loss: 8.8722
Epoch 89/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 122ms/step - loss: 5.4756 - val_loss: 8.8701
Epoch 90/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 93ms/step - loss: 5.4861 - val_loss: 8.8711
Epoch 91/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 90ms/step - loss: 5.5345 - val_loss: 8.8709
Epoch 92/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 113ms/step - loss: 5.5207 - val_loss: 8.8712
Epoch 93/500
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 91ms/step - loss: 5.5483 - val_loss: 8.8708
```

Custom CTC loss

Title

```python
def ctc_loss(y_true, y_pred):

    # Transpose y_pred
    y_pred = tf.transpose(y_pred, [1, 0, 2])

    # Ensure y_true is of type int32
    y_true = tf.cast(
      y_true, tf.int32)

    # Calculate input length (logit length)
    # and label length
    logit_length = tf.fill(
      [tf.shape(y_pred)[1]],
      tf.shape(y_pred)[0])  # shape:
(batch_size,)
    label_length = tf.reduce_sum(
      tf.cast(
        tf.not_equal(y_true, 0), tf.int32
      ), axis=1)  # shape: (batch_size,)

    # Compute the CTC loss
    # using tf.nn.ctc_loss
    loss = tf.nn.ctc_loss(
        labels=y_true,
        logits=y_pred,
        label_length=label_length,
        logit_length=logit_length,
        logits_time_major=True,
        blank_index=0
    )

    return loss
```

Model Optimisation

1

Before

Conv1D and Dense layers:

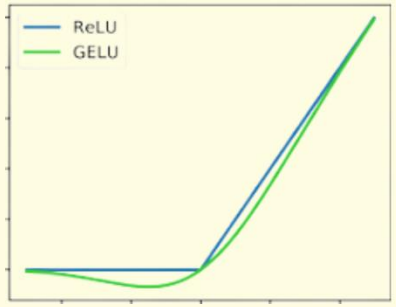
```
x = layers.Conv1D(256, 3, activation='relu', padding='same')(x)
```

After

Conv1D and Dense layers:

```
x = layers.Conv1D(256, 3, activation='gelu', padding='same')(x)
```

Reason



GELU provides smoother activation than ReLU and helps prevent dying neurons.

2

Before

Convolutional blocks:

```
x = layers.Dropout(0.3)(x)
```

After

Convolutional blocks:

```
x = layers.SpatialDropout1D(0.2)(x)
```

Reason

Standard Dropout randomly removes individual units, while **SpatialDropout1D** drops entire feature maps along the channel axis. This helps preserve spatial dependencies in time-series data

3

Before

```
# Compile the model with CTC loss
cldnn_model.compile(optimizer='adam', loss=ctc_loss)
```

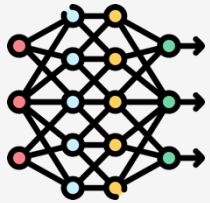
After

```
# Compile the model with CTC loss
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001, clipnorm=1.0)
cldnn_model.compile(optimizer=optimizer, loss=ctc_loss)
```

Reason

Gradient clipping (clipnorm=1.0) stabilizes training by preventing large updates that could disrupt convergence.
This enhances model robustness and prevents exploding gradients.

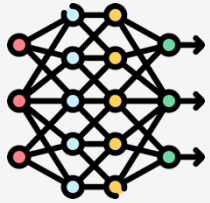
Model Optimisation



CLDNN model [5]

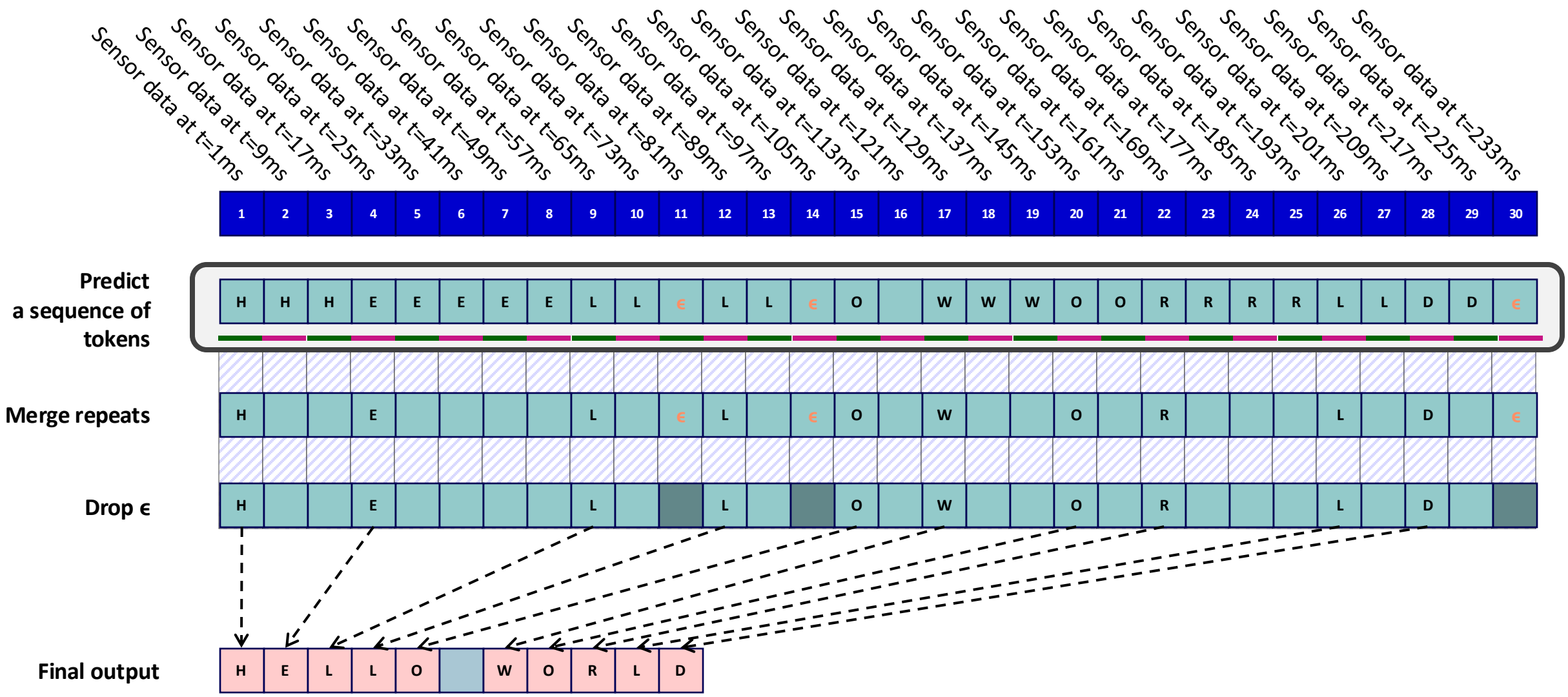
1. Introduce derivatives
2. Replace relu with gelu
3. Set clipnorm to 1
4. Add learning rate scheduler
5. Replace dropout with spatial dropout

Change	Accuracy Before the Change (%)	Accuracy After the Change (%)	Step Improvement (%)	Cumulative Improvement (%)
1	5.63	6.78	1.15	1.15
2	6.78	12.93	6.15	7.3
3	12.93	20.35	7.42	14.72
4	20.35	58.08	37.73	52.45
5	58.08	63.15	5.07	57.52

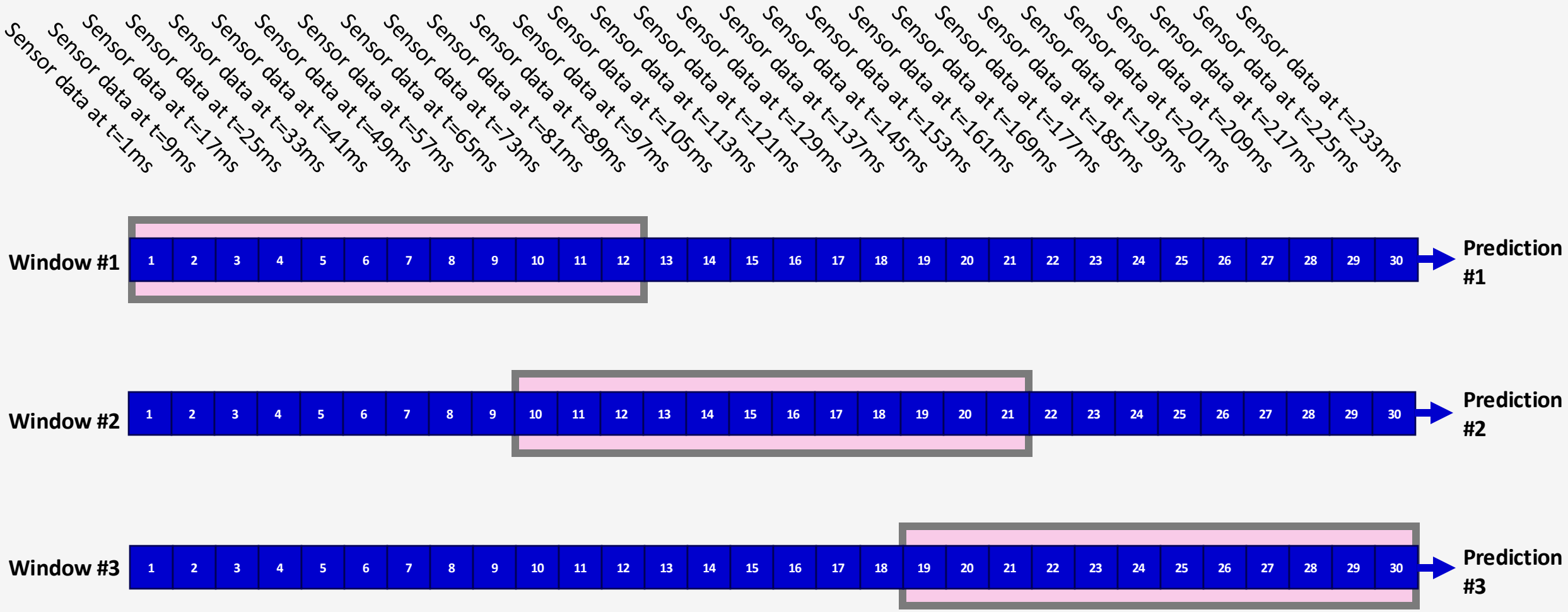


Optimised CLDNN model with improced accuracy

CTC Loss

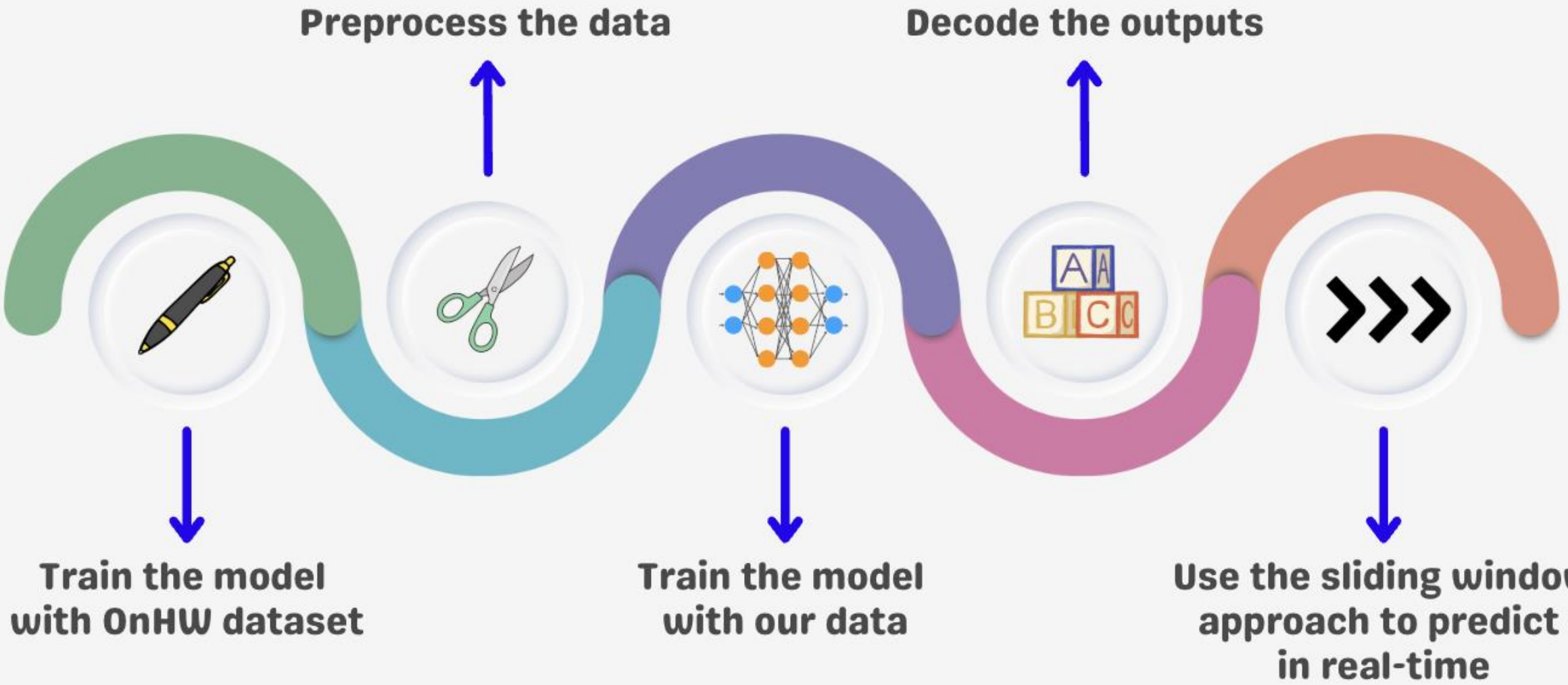

Sensor data at t=1ms
Sensor data at t=9ms
Sensor data at t=17ms
Sensor data at t=25ms
Sensor data at t=33ms
Sensor data at t=41ms
Sensor data at t=49ms
Sensor data at t=57ms
Sensor data at t=65ms
Sensor data at t=73ms
Sensor data at t=81ms
Sensor data at t=89ms
Sensor data at t=97ms
Sensor data at t=105ms
Sensor data at t=113ms
Sensor data at t=121ms
Sensor data at t=129ms
Sensor data at t=137ms
Sensor data at t=145ms
Sensor data at t=153ms
Sensor data at t=161ms
Sensor data at t=169ms
Sensor data at t=177ms
Sensor data at t=185ms
Sensor data at t=193ms
Sensor data at t=201ms
Sensor data at t=209ms
Sensor data at t=217ms
Sensor data at t=225ms
Sensor data at t=233ms
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Predict a sequence of tokens
H H H E E E E E L L ϵ L L ϵ O W W W O O R R R R L L D D ϵ
Merge repeats
H E L ϵ L ϵ O W O R L D ϵ
Drop ϵ
H E L L O W O R L D
Final output
H E L L O W O R L D

Sliding Window Approach



Final Prediction = **Prediction #1** + **Prediction #2** + **Prediction #3**

End-to-end Machine Learning Pipeline



Experimental Results

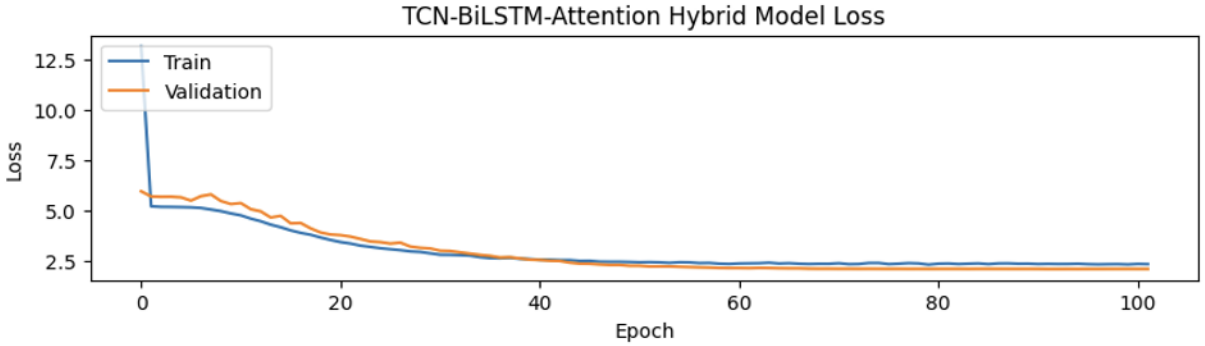
Model Performances, Loss Plots

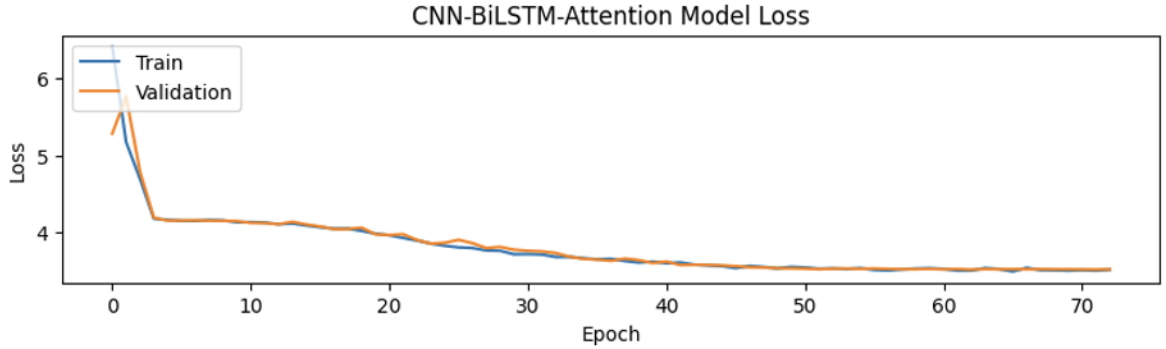
Discussion on Why Some Models Outperform Others

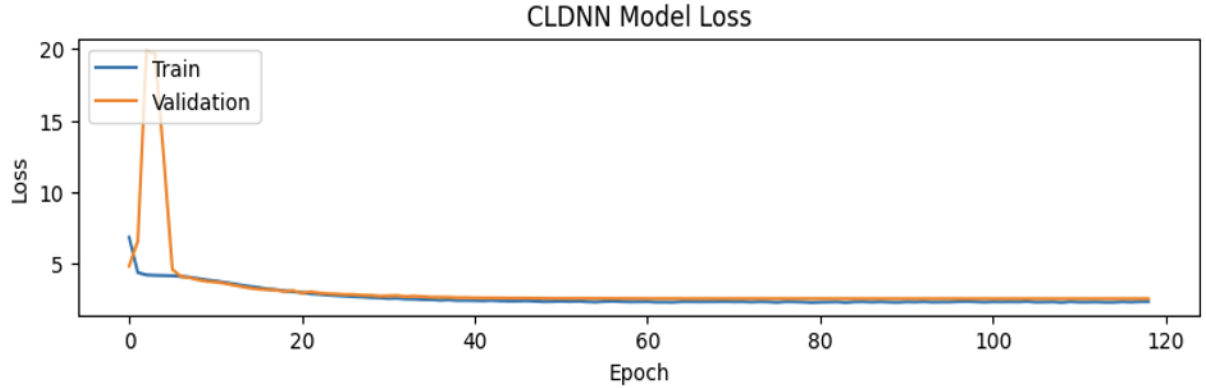
```
# Sort models (highest sequence accuracy first)
sorted_models = sorted(results.items(), key=lambda x: x[1], reverse=True)

# Print
for model_name, accuracy in sorted_models:
    print(f"{model_name}: Sequence Accuracy = {accuracy * 100:.2f}%")
```

```
CLDNN: Sequence Accuracy = 23.04%
TCN-BiLSTM-Attention Hybrid: Sequence Accuracy = 21.53%
CNN-BiLSTM-Attention: Sequence Accuracy = 7.64%
Attention-Based Seq2Seq Model: Sequence Accuracy = 5.81%
InceptionTime: Sequence Accuracy = 4.74%
TCN: Sequence Accuracy = 3.88%
CNN: Sequence Accuracy = 1.29%
BiLSTM: Sequence Accuracy = 0.86%
HMM-RNN: Sequence Accuracy = 0.86%
BLSTM-CTC: Sequence Accuracy = 0.86%
Transformer: Sequence Accuracy = 0.00%
TCN-BiLSTM-Transformer Hybrid: Sequence Accuracy = 0.00%
```

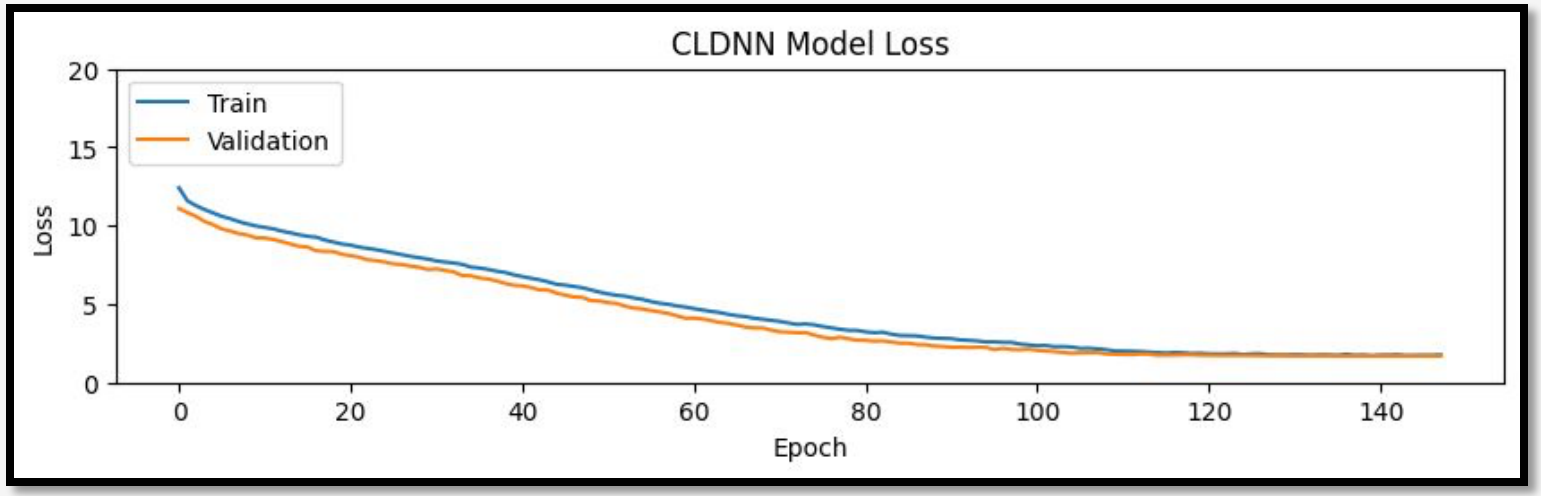
Model	Acc. (%)	Key Strengths
CLDNN	23.04	Strong **feature extraction** (CNN) + **temporal dependencies**, past and future contexts (LSTM)
TCN-BiLSTM-Attention Hybrid	21.53	**Long-term dependencies** without losing information about the **temporal order** (TCN), preceding and succeeding sensor readings (BiLSTM), **critical movements**, such as **sharp turns** in the pen trajectory or pressure changes (attention)
CNN-BiLSTM-Attention	7.64	Effective hybrid approach, but weaker attention







Training and Validation Losses



Decoding Method	Sequence Accuracy (%)
Greedy	61.97
Beam	68.13

```
Sample True vs. Predicted Comparisons:
True: hbd | Predicted: hd
True: l | Predicted: l
True: B | Predicted: B
True: BLANK | Predicted: BLANK
True: f | Predicted: x
True: dkdog ms | Predicted: dnog m
True: XdkknvZSqdd | Predicted: XdknvZSd
True: c | Predicted: T
True: z | Predicted: N
True: bgdqqx | Predicted: bgdqq
True:    | Predicted:
True: C s ZRbhdmbd | Predicted: C  ZRbhdmb
True: kdlnm | Predicted: kdlnm
True: t | Predicted: t
True: S | Predicted: S
True: 4 | Predicted: 4
True: G | Predicted: L
True: uhnkhm | Predicted: mhfkh
True: 4 | Predicted: 4
True: G | Predicted: G
```

Workplan

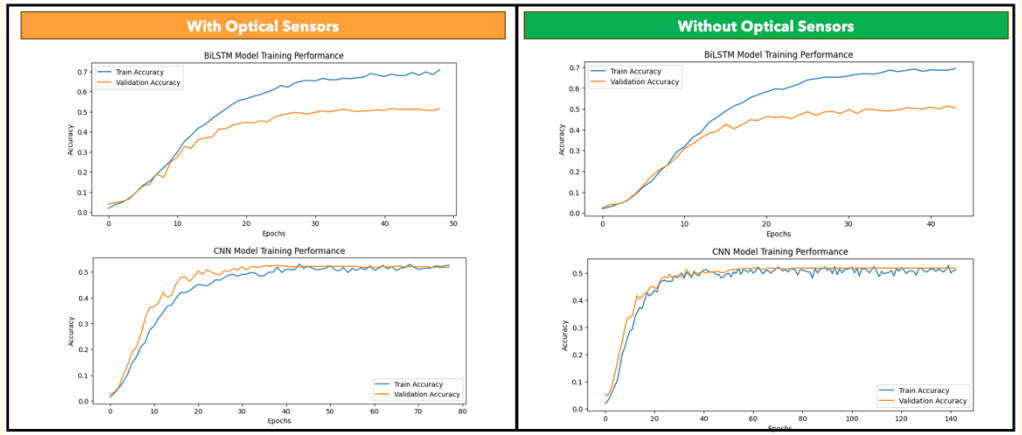
Milestones and Challenges

Project Timeline

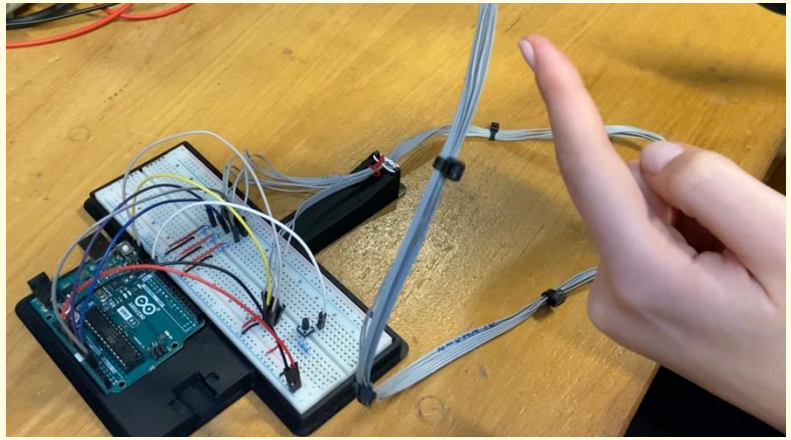
Gantt Chart

Key Challenges and Decisions

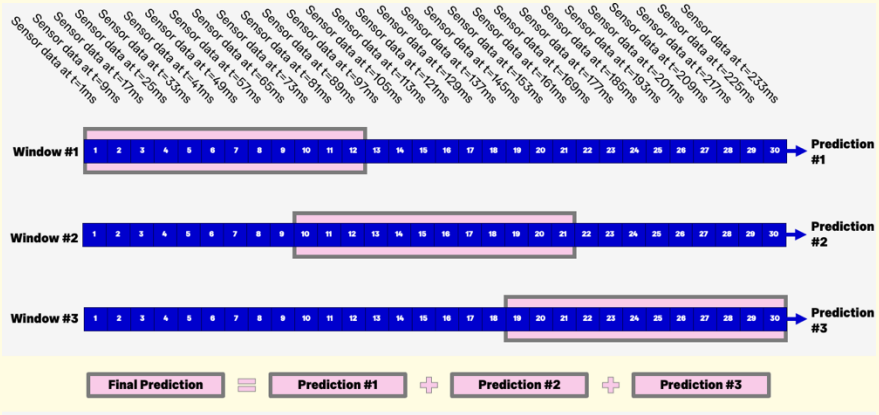
Removing the optical sensor



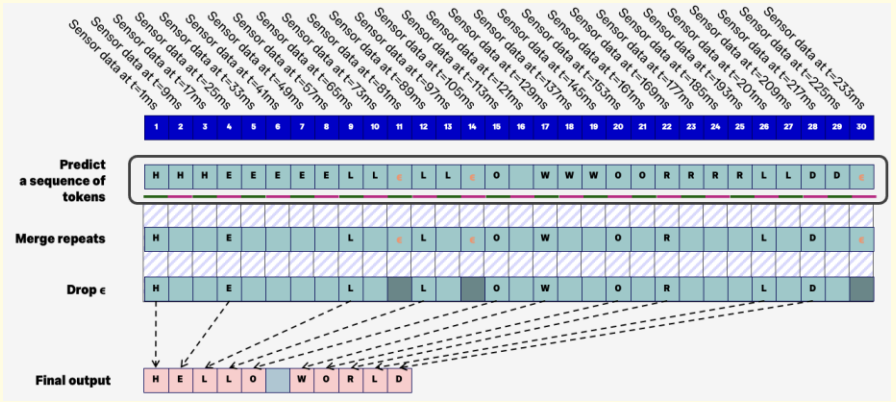
Long wiring



Real time decoding



CTC loss being different from what we've seen so far



Conclusion

Summary, Evaluation, Future Work

Summary & Evaluation

To Summarize What We Have Done:

- Able to implement a CLDNN model for our project
- Able to collect handwriting dataset using the hardware
- Able to have a working final project overall
- Able to create a working smart pen hardware
- Able to optimize the model that used our dataset

Potential Future Work

Extended Dataset Collection
Increase the dataset size by collecting more handwriting samples from diverse participants, including different writing styles, speeds, and orientations.

Context-Aware Recognition
Implement a language model (e.g., LLM or transformer-based post-processing) to improve recognition by leveraging word and sentence context.

Handwriting-to-Editable-Document
Expand beyond LaTeX to support real-time handwriting-to-Word or Markdown conversion for broader usability.

Few-Shot Learning for Personalization
Train models that adapt to individual handwriting styles with minimal additional data using meta-learning techniques.

Erasure and Editing Improvements
Enhance dynamic erasure detection and correction, allowing seamless real-time modifications of handwritten text.

Multi-Language Support
Extend the system to support multiple languages and scripts, adapting character recognition models accordingly.

References

Papers and Other Resources

References

[1] A. Author, B. Author, and C. Author, “A Self-Attention-Based Deep Architecture for Online Handwriting Recognition,” *Proc. Conf. on Pattern Recognition*, 2024.

[2] X. Zhang, Y. Liu, and Z. Wang, “Representing Online Handwriting for Recognition in Large Vision-Language Models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2024.

[3] M. Wehbi, T. Hamann, J. Barth, P. Kaempf, D. Zanca, and B. Eskofier, “Benchmarking Online Sequence-to-Sequence and Character-Based Handwriting Recognition from IMU-Enhanced Pens,” in *Proc. ICDAR*, 2022.

[4] P. Smith and R. Lee, “Improving Accuracy and Explainability of Online Handwriting Recognition,” *Neural Computation*, vol. 34, no. 3, pp. 1234–1250, 2022.

[5] M. Wehbi, T. Hamann, J. Barth, P. Kaempf, D. Zanca, and B. Eskofier, “Towards an IMU-Based Pen Online Handwriting Recognizer,” in *Document Analysis and Recognition – ICDAR 2021*, J. Lladós, D. Lopresti, and S. Uchida, Eds. Cham: Springer, 2021, vol. 12823, Lecture Notes in Computer Science. doi: 10.1007/978-3-030-86334-0_19.

[6] F. Ott, M. Wehbi, T. Hamann, J. Barth, B. Eskofier, and C. Mutschler, “The OnHW Dataset: Online Handwriting Recognition from IMU-Enhanced Ballpoint Pens with Machine Learning,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. (IMWUT)*, vol. 4, no. 3, article 92, pp. 1–20, Sep. 2020, doi: 10.1145/3411842.

References

[7] L. Kim, H. Park, and J. Choi, “Deep-Learning-Based Character Recognition from Handwriting Motion Data Captured Using IMU and Force Sensors,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5231–5243, 2022.
[8] A. Gupta and S. Mehta, “Motion-Based Handwriting Recognition and Word Reconstruction,” *Pattern Recognition Letters*, vol. 151, pp. 12–21, 2021.
[9] Voidful, “Understanding CTC loss for speech recognition,” *Medium*, [Online]. Available: https://voidful.medium.com/understanding-ctc-loss-for-speech-recognition-a16a3ef4da92. [Accessed: Mar. 20, 2025].
[10] Wikipedia, “Rectifier (neural networks),” *Wikipedia*, [Online]. Available: https://en.wikipedia.org/wiki/Rectifier_%28neural_networks%29. [Accessed: Mar. 20, 2025].
[11] Distill, “CTC,” *Distill.pub*, [Online]. Available: https://distill.pub/2017/ctc/. [Accessed: Mar. 20, 2025].

IMPERIAL

Thank you

NeverLateX Final Presentation
21/03/2025