# Deep Consensus Finding Edge Pipeline for Nanopore Sequencing

Rohan Juneja (A0228349W)
Mohit Upadhyay (A0201104J)

# Problem Statement

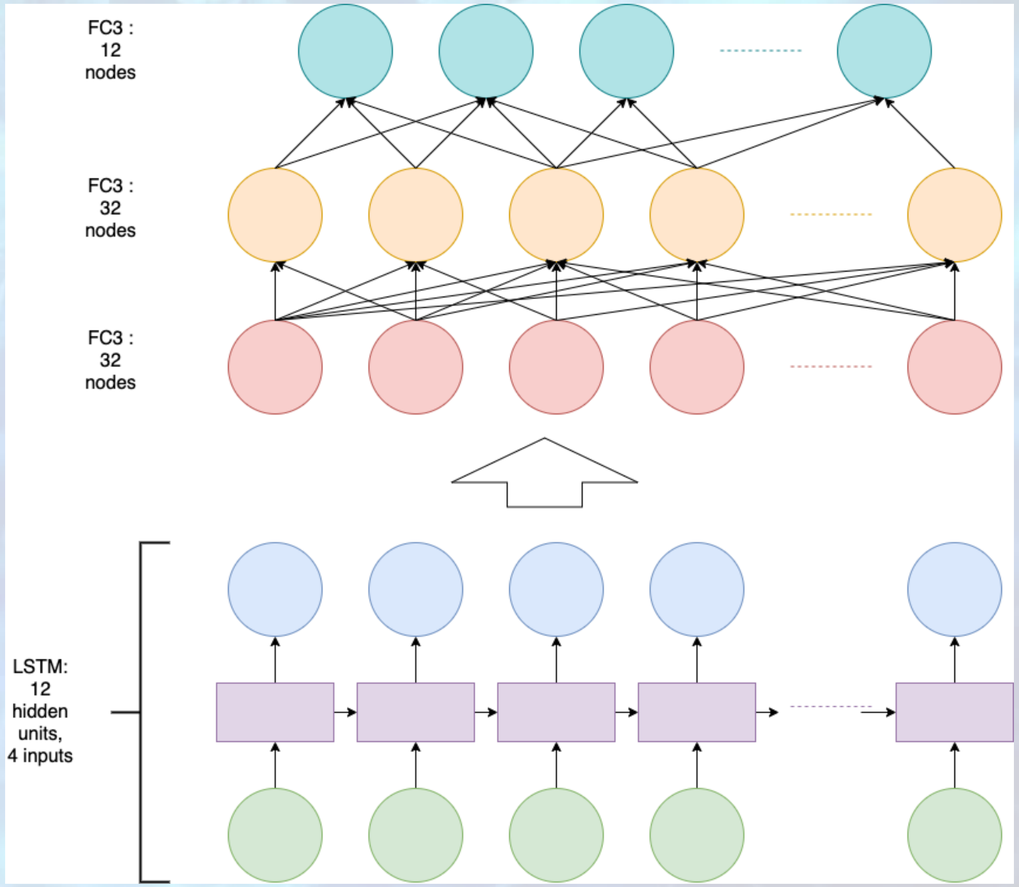

- Deep neural network-based consensus finding
  - DCNet: LSTM-RNN based
  - DeepConsensus+: Transformers for consensus
  - Evaluation on an Edge device (size of an Arduino)
- Software Pipeline around Nanopore Sequencing and DeepConsensus on Edge Accelerators
  - Progressive Clustering

# DCNet: Denoising DNA sequence with LSTM-RNN

- Problem Statement: **Stack** a set of erroneous DNA sequences together to remove all kinds of errors with a neural network!
- We can make a recurrent neural network (RNN) with long short-term memory (LSTM) cells, to aggregate correct sub-sequences and ignore random erroneous events without doing explicit alignments.

# DCNet: Neural Network Architecture

- LSTM-based model with 3 fully-connected layers
- All three FC have dropout to reduce overfitting
- Has a quite low accuracy

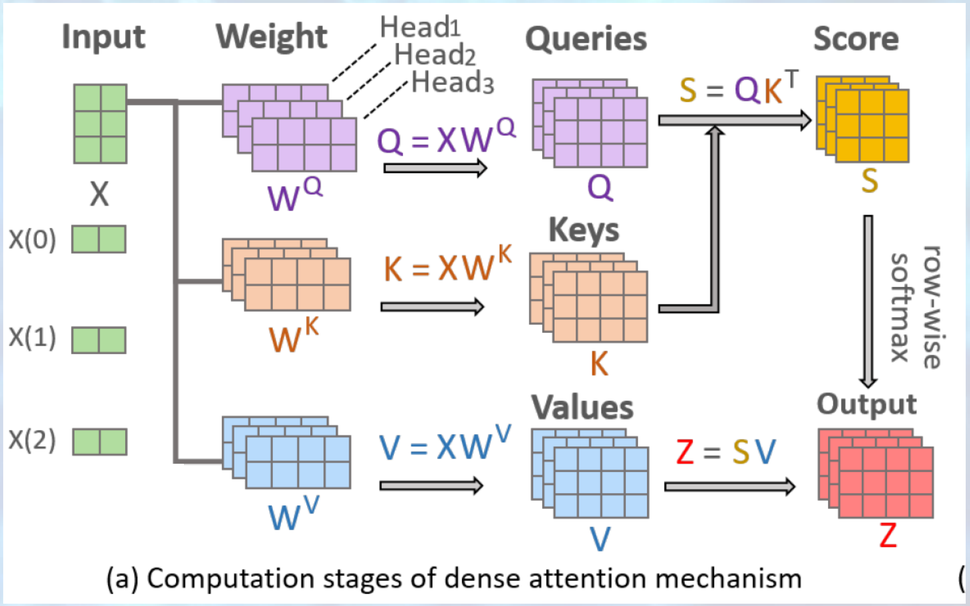


# Methodology

- DCNet was appended with a dropout layer for the last fully-connected layer.
  - Helps to reduce overfitting
- Both models were trained on 10,000 clusters with 10 noisy copies each for epochs. [Training Dataset]
- For accuracy computation, 1000 clusters (with 10 noisy copies each) were used. [Test Dataset]

# DeepConsensus: Gap-Aware Sequence Transformers for Sequence Correction [2]

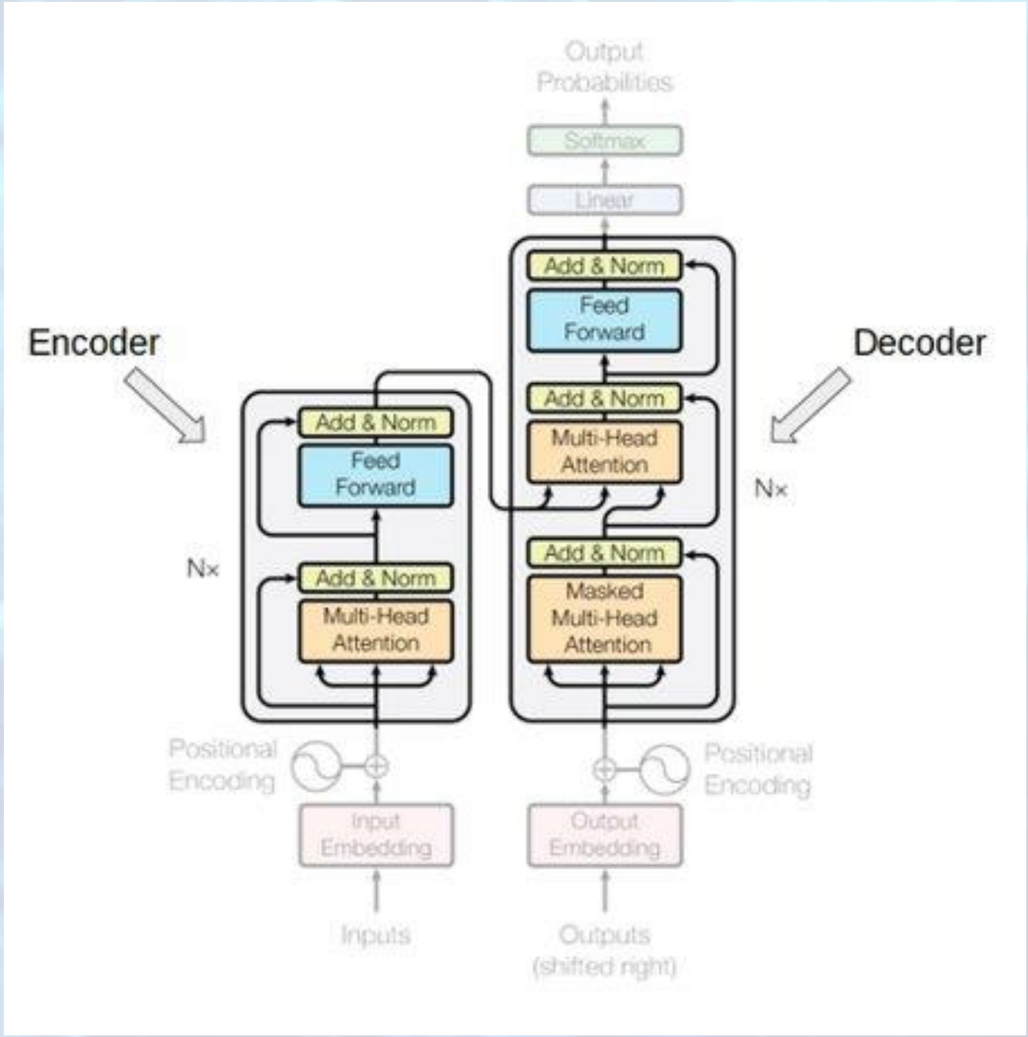
- Developed in collaboration with Google.
- A six-layer encoder-only transformer model with a hidden dimension of 560 and 2 attention heads in each self-attention layer. The inner dimension of the feedforward network in each encoder layer is 2048.
- A custom loss function is used to calculate the error values.



(a) Computation stages of dense attention mechanism

# DeepConsensus: Neural Network Architecture

- Transformer-based model
  - Has 6 encoder layers
  - No decoder layers
  - Encoder outputs fed to a fully connected layer (small dropout probability introduced)
- Cross-entropy loss function used during training



# Methodology

- DeepConsensus+ was appended with a dropout layer for the last fully-connected layer.
  - Helps to reduce overfitting
- Optimized Model: Added extra tokens within transformer model
- Both models were trained on 10,000 clusters with 10 noisy copies each for 20 epochs. [Training Dataset]
- For accuracy computation, 1000 clusters (with 10 noisy copies each) were used. [Test Dataset]

# Results: Per-character Accuracy

| Insertion, Deletion and Substitution individual error rate | Baseline Accuracy (in %) | DeepConsensus+ (in %) |
|---|---|---|
| 1% | 99.98 | 77.883 |
| 2% | 99.54 | 77.959 |
| 3% | 97.39 | 78.01 |
| 4% | 92.09 | 77.965 |
| 5% | 84.81 | 78.133 |
| 6% | 75.22 | 78.031 |
| 7% | 66.61 | 78.024 |
| 8% | 60.2 | 77.674 |
| 9% | 55.22 | 76.57 |

**Accuracy for 1000 clusters for Baseline, DeepConsensus+**

| Number of Clusters | Baseline Accuracy (in %) | DeepConsensus + (in %) |
|---|---|---|
| 10 | 79.5 | 81.03 |
| 100 | 76.7 | 80.25 |
| 1000 | 75.22 | 80.031 |
| 10,000 | 75.34 | 78.06 |

**Accuracy for varied clusters for Baseline, DeepConsensus+ with insertion, deletion and substitution error of 6%**

# Results: Performance

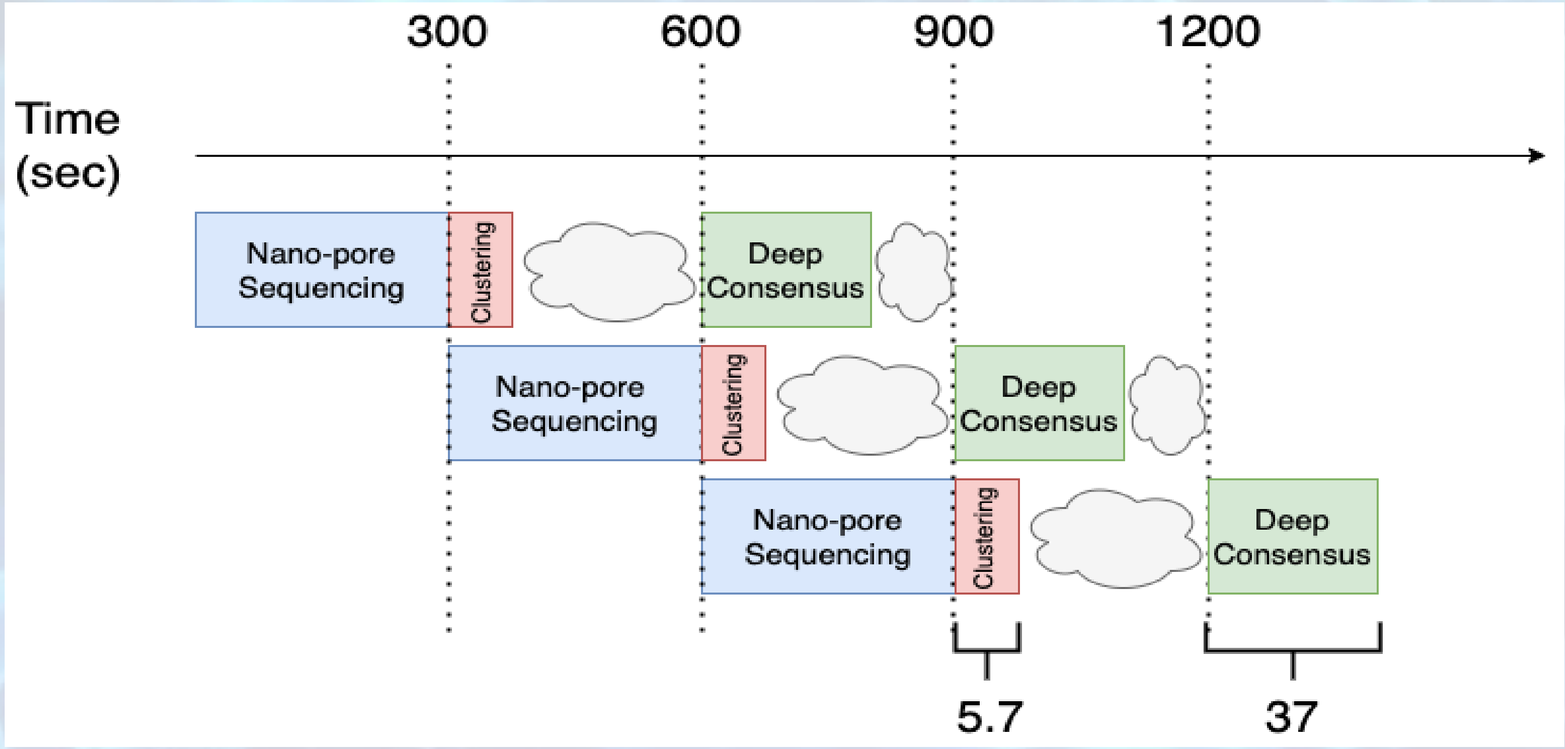
- Execution time is computed for Nvidia Jetson Nano board.

| Number of clusters | Baseline (in sec) | DeepConsensus+ (in sec) |
|---|---|---|
| 10 | 0.281 | 0.34 |
| 100 | 2.932 | 3.701 |
| 1000 | 25.199 | 38.017 |
| 10,000 | 237.935 | 370.179 |

# Nanopore Sequencing: Software Pipeline

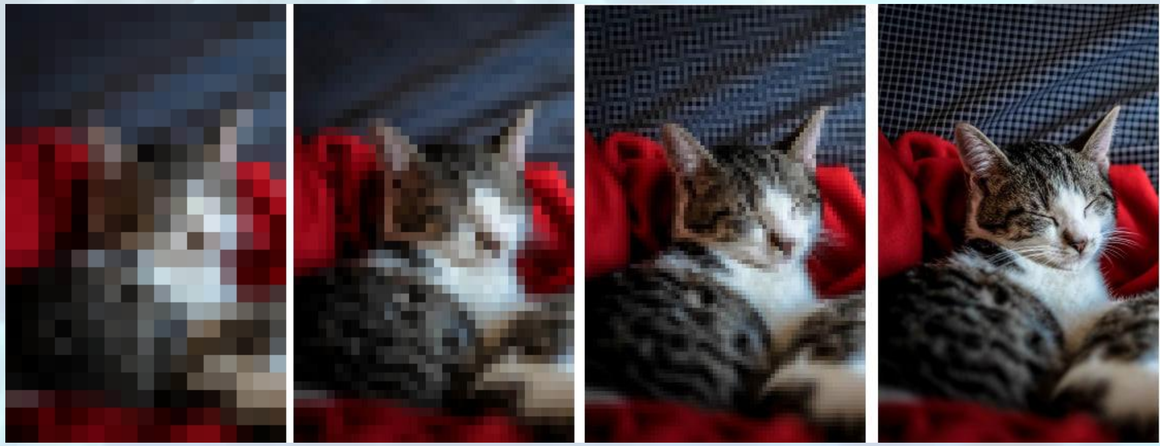
- Proposed software pipeline for nano-pore sequencing at the edge
  - Nano-pore sequencing can be done faster and on smaller device area
  - Proposed pipeline utilizes **streaming** of DNA data from the nanopore sequencing device
- Benchmarking done on Jetson Nano [4] board
  - Deep-Consensus inference done on GPU
- System-based pipeline evaluation (1,000 clusters):
  - Clustering on Jetson Nano (CPU): 5.756 sec
  - Consensus Finding on Jetson Nano (GPU): 37.011 sec
- MinION: Nanopore portable technology (state-of-the-art nano pore sequencing)
  - Can read upto 420 bases per second


# Proposed Software Pipeline




- Nano-pore sequencing is still the bottleneck in the given pipeline
- Possible to add another "cache" in the pipeline to reduce the bottleneck and increase utilization (Future Work)

# Progressive Clustering and Trace Reconstruction



for iterator = 1 to 1000:

cluster1 = sequence 1000 random strands // Take less than 5 min

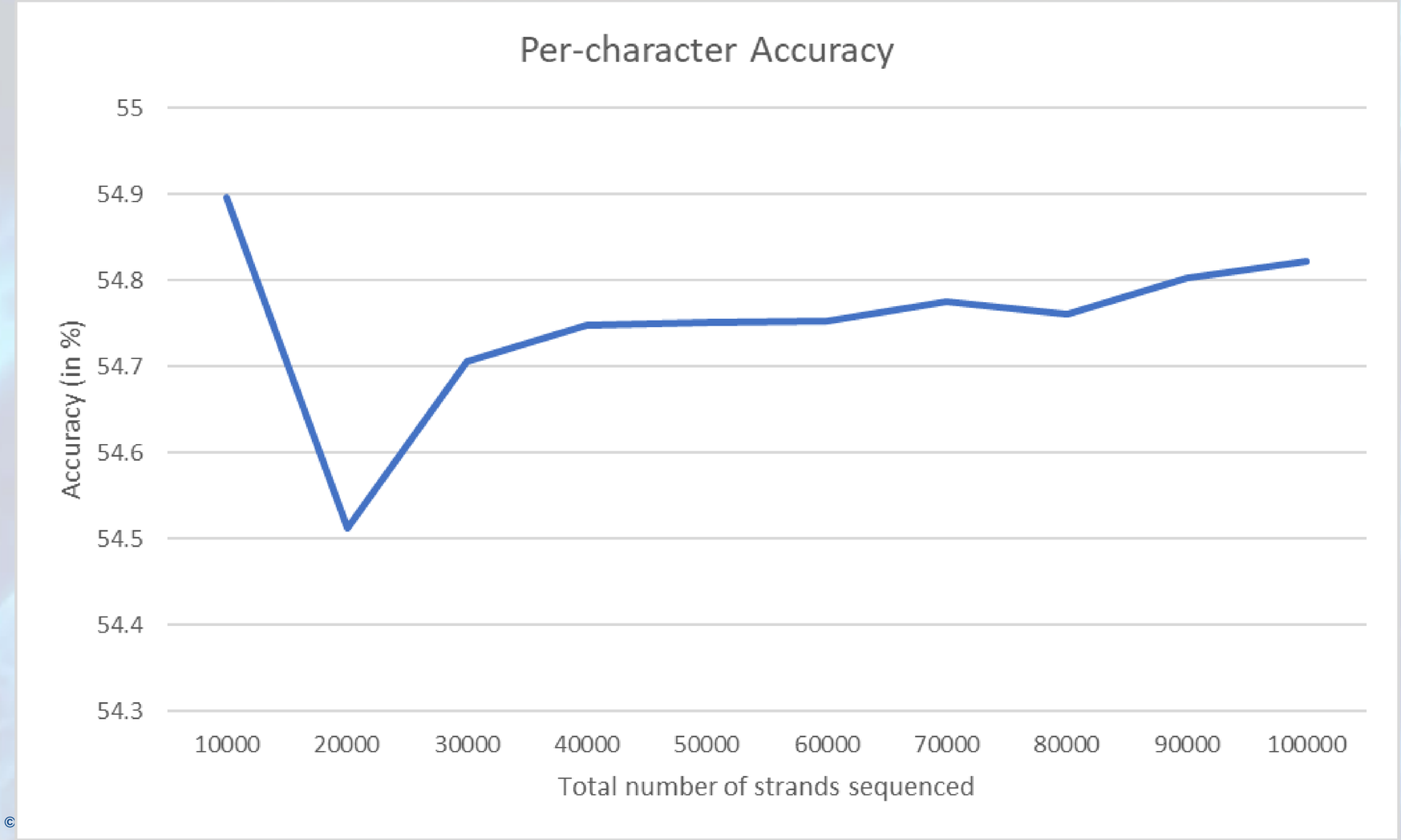
clustering(cluster1)

strands = trace_reconstruction(cluster1)

save_strands = save_strands.append(strands)

final_strands = trace_reconstruction(save_strands)

# Progressive Clustering and Trace Reconstruction



# References

[1] “DCNet — Denoising (DNA) Sequence With a LSTM-RNN and PyTorch”, https://infoecho.medium.com/dcnet-denoising-dna-sequence-with-a-lstm-rnn-and-pytorch-3b454ff727e7

[2] “Reducing the dimensionality of data with neural networks”

[3] Gunjan Baid et al, “DeepConsensus: Gap-Aware Sequence Transformers for Sequence Correction”, https://doi.org/10.1101/2021.08.31.458403

[4] https://developer.nvidia.com/embedded/jetson-nano-developer-kit

# Thanks!!

- Questions?