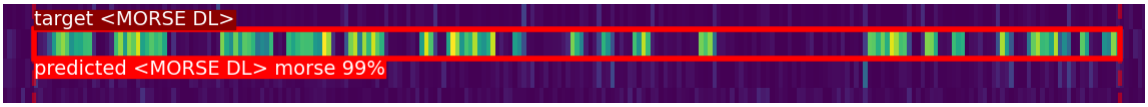


Morse Code Decoder & Detector with Deep Learning

Maor Assayag | Eliraz Kadosh | Eliram Amrusi

April 2024

1. Introduction

Morse code detection methods have evolved from DSP techniques requiring prior signal knowledge to robust deep learning models, addressing issues in traditional time-frequency analysis-based approaches. In this project, we build a DNN to detect morse code in a spectrogram, utilizing the latest advances in Deep Learning object detection and classification. This is a challenging task, as real-world data contains pink\white noise, interferences, and distortions. We investigated two architectures (LSTM-RNN, Faster-RCNN). Multiple approaches have been taking on this task in the past, but focused on those architectures and pipeline that reflect as many techniques from the course as possible, aiming for the object-detection-approach. The variance in labeled morse code data that are needed is hard to come by, thus we synthesized new labeled bursts of morse CW signals. We tested our network output with common metrics in communication decoding, such as a variant of CER (Character Error Rate) vs SNR (Signal-to-Noise-Ratio). The results demonstrate the effectiveness of these approaches in decoding and detection morse code from spectrograms.

2. Ethics Statement

2.1 Stakeholders: Military personnel, rescue organizations, Morse code enthusiasts.

2.2 Implications: Military personnel and rescue organizations may benefit from reliable emergency communication in remote or hostile environments, aiding in covert operations, survival situations, and interoperability with legacy equipment, while also facilitating training and proficiency in Morse code communication.

Morse code enthusiasts may enhance their decoding accuracy, deepening engagement with Morse code communication. On the other hand, they may decode sensitive and classified codes that they were not supposed to reach.

2.3 Ethical Considerations: Implementing robust encryption mechanisms to safeguard intercepted Morse code transmissions.

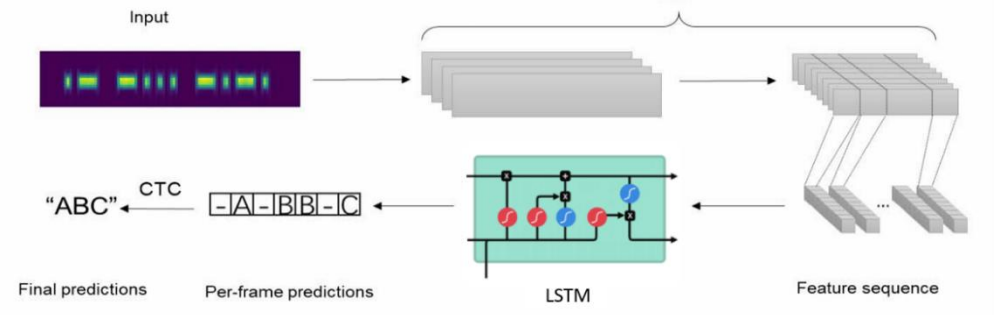
Fostering transparency and accountability by clearly communicating capabilities and risks, establishing mechanisms for addressing misuse.

Implementing a mechanism that identifies transmissions containing offensive/racist content and filtering it according to the type of user.

3. Methods

3.1 Decoding task – LSTM RNN

Implemented in PyTorch, we designed an LSTM RNN model for decoding the morse code content. The network has more than 740,000 trainable parameters, with the ability to decode dynamic spectrogram input length.



3.1.1 Motivation

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture that has been widely used for sequence-to-sequence tasks, such as machine translation, speech recognition, and, in our case, decoding Morse code from a spectrogram.

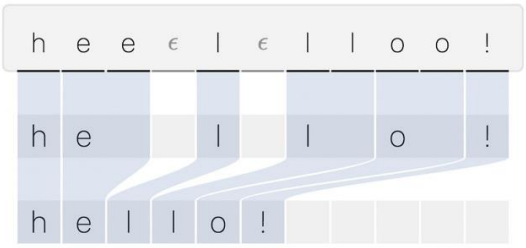
- **Spectrogram Representation** - Before feeding the data into the LSTM, we need to represent the Morse code signal as a spectrogram. A spectrogram is a visual representation of the spectrum of frequencies in a signal as they vary with time. It is a 2D image-like representation, where the x-axis represents time, the y-axis represents frequency, and the intensity of each pixel represents the energy or amplitude of the signal at that particular time and frequency.

- **LSTM Architecture** - The LSTM network is designed to process sequential data, such as the spectrogram. It consists of a series of interconnected memory cells, which are capable of maintaining and updating their internal state over time. This allows the LSTM to selectively remember or forget information from previous time steps, making it well-suited for tasks that require long-term dependencies. Our LSTM architecture consists of the following components:
  1. Input Layer: This layer receives the input data, which in our case is the spectrogram of the Morse code signal
  2. 3 Dense Layers
  3. Dropout Layer after most of the Dense Layers
  4. LSTM Layer: One LSTM layer to process the input sequence
  5. Output Layer: The final layer produces the predicted output sequence, which is the decoded string

- **Training and Decoding** - During training, the LSTM network is fed with pairs of spectrogram representations and their corresponding text transcriptions. The network learns to map the input spectrograms to the correct text sequences by adjusting its weights and biases through backpropagation and optimization algorithms like stochastic gradient descent. During decoding (inference), the trained LSTM model takes a new spectrogram as input and generates the predicted text sequence by iteratively processing the input and updating its internal state. Techniques like beam search or greedy decoding can be employed to find the most likely text sequence given the input spectrogram.

3.1.2 CTC - The Connectionist Temporal Classification loss

Calculates loss between a continuous (unsegmented) time series and a target sequence. CTC Loss sums over the probability of possible alignments of input to target, producing a loss value which is differentiable with respect to each input node. The alignment of input to target is assumed to be “many-to-one”, which limits the length of the target sequence such that it must be ≤ the input length.



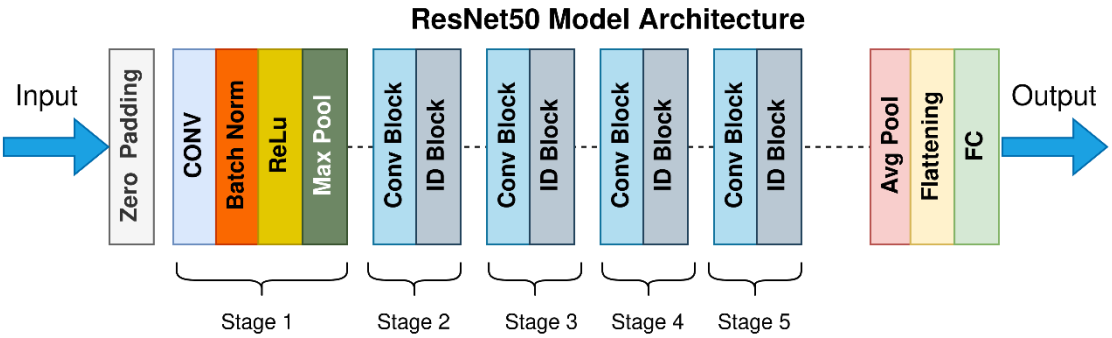
3.1.3 CER (Character Error Rate)

The Character Error is the percentage of characters that have been transcribed incorrectly by the Text Recognition model. For example, a 5% CER means that the text model has correctly transcribed 95 characters out of 100, while it has misread only 5 characters. We incorporated Levenstein Distance in the CER for measuring the model accuracy.

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

3.2 Detection task – Faster RCNN (resnet50)

For the detection objective, we found that the built-in support of torchvision in resnet is suitable for our task. We chose the medium size resnet, RESNT50, with a little over 40,000,000 trainable parameters.



3.2.1 Motivation

Faster R-CNN (Faster Regions with Convolutional Neural Networks) is an object detection algorithm that builds upon the success of its predecessors, R-CNN and Fast R-CNN. It was introduced in the 2015 paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks".

The key innovation in Faster R-CNN is the introduction of a **Region Proposal Network** (RPN), which is a fully convolutional network that generates region proposals (bounding boxes) for objects in the input image. This contrasts with previous approaches that relied on external region proposal algorithms like selective search, which were computationally expensive and slow. The Faster R-CNN architecture consists of two main components:

1. **Region Proposal Network (RPN)**: This is a lightweight convolutional neural network that takes an input image and outputs a set of object proposals or regions of interest (RoIs). The RPN works by sliding a small window over the input feature map and outputting multiple region proposals for each window location. These proposals are then filtered and ranked based on their objectness score, which is the probability that a region contains an object of interest.

2. **Object Detection Network**: This is a Fast R-CNN network that takes the proposed RoIs from the RPN and performs object classification and bounding box regression. It extracts features from each RoI using a RoI pooling layer and then passes these features through fully connected layers to predict the class and refine the bounding box coordinates.

We will use 'fasterrcnn_resnet50_fpn' variant (original model), read more here "PyTorch FASTER R-CNN".

3.2.2 Loss Function

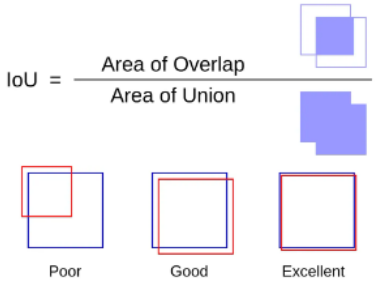
The RPN loss function typically consists of two components:

1. **Classification Loss**: measures the accuracy of the region proposals in terms of objectness. Each anchor box (a set of predefined bounding boxes of different sizes and aspect ratios) is classified as either containing an object (positive) or not containing an object (negative). The classification loss, often computed using binary cross-entropy loss, penalizes incorrect objectness predictions. In our case, we only have 2 classes – 'morse' and 'background'.

2. **Regression Loss**: In addition to predicting objectness, the RPN also predicts adjustments to the anchor box coordinates to better align with the ground truth bounding boxes. The regression loss, often computed using IoU metrics, measures the discrepancy between the predicted box coordinates and the ground truth box coordinates.

3.2.3 IoU-x

IoU measures the overlap between the predicted region and the ground truth region, helping to quantify how well an algorithm's output aligns with the actual object or region in the image. Our focus is to detect the start and end of data in the signal frame. Thus, we focus on x-axis accuracy using a custom IoU metric we call IoU-x.



4. Experiments and Results

4.1 Dataset

Our models require supervised learning, and labeled morse code datasets that fits our needs are not publicly available. In addition, it's common for tasks with unique data (such as communication, military, driving etc.') to build a physical model that simulate the real data as much as possible, while incorporating.

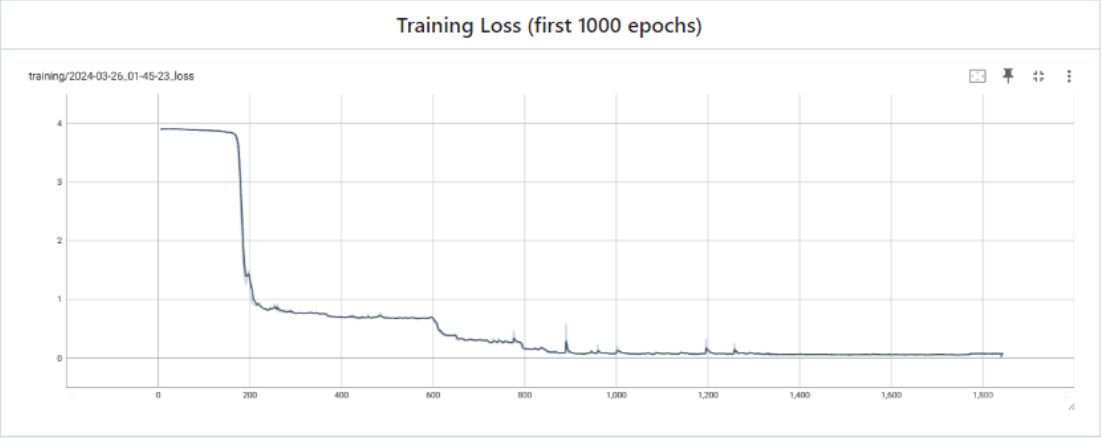
We produce time domain signals in real-time (for each iteration that required a new batch, we override the *_next_item* function of the dataset to generate new signals) with randomized parameters values.

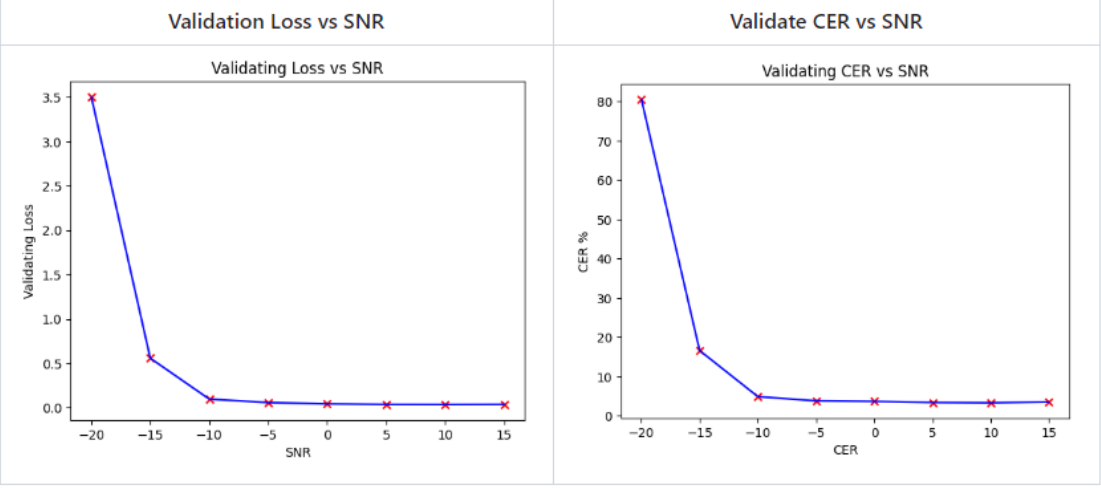
These parameters make the morse code to vary in **pitch** (main frequency), **SNR** (Signal to Noise Ration, from -5dB up to 10 dB), Signal **Amplitude**, Morse String **Length** (from 1 char to 20 chars), and inner-symbol duration that simulates Doppler effect.

4.2 Decoding task – LSTM RNN

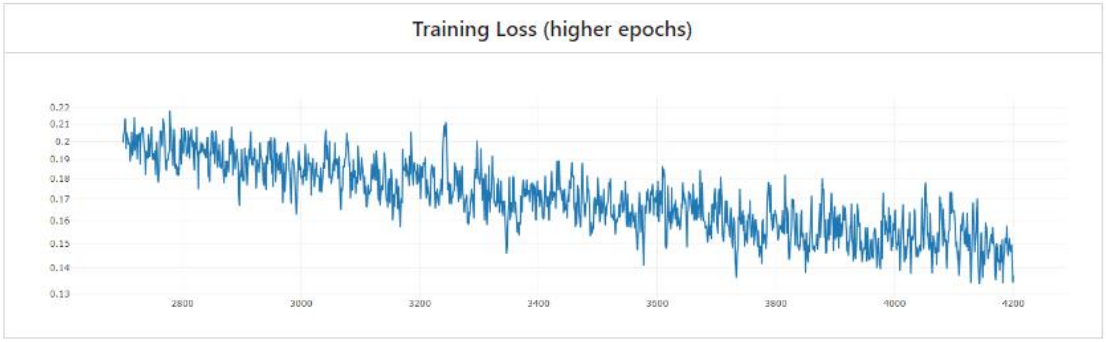
We trained a bunch of LSTM models with different hyperparameters. The final model was trained for ~4000 epochs, translating to around 50,000,000 unique morse chars in different conditions. Two main observations:

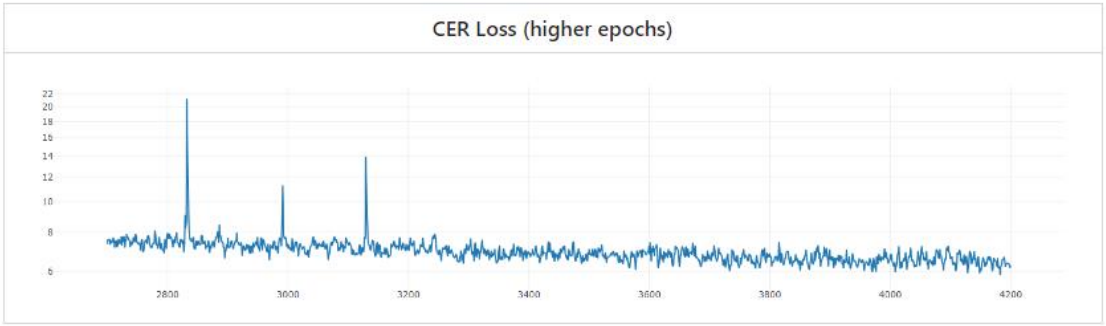
- At 200 and 600 epoch we can see large drop in the model loss, which is around 6 hour of GPU training.
- For signals above -10dB we get less than 3% CER.





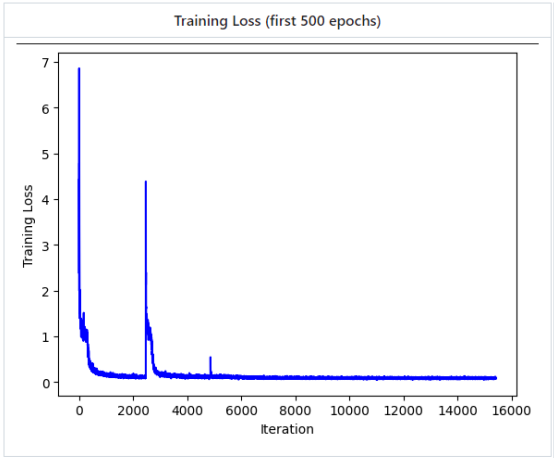
In the higher epochs training range, we can still see improvements in loss and CER, with learning rate of 10^{-4} (for the detection task we implemented adaptive learning rate).

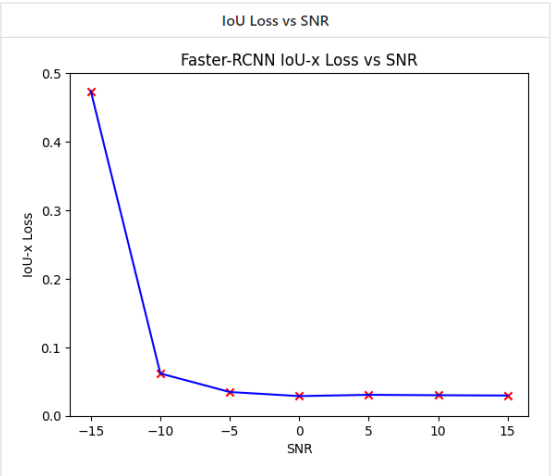




4.3 Detection task – Faster RCNN (resnet50)

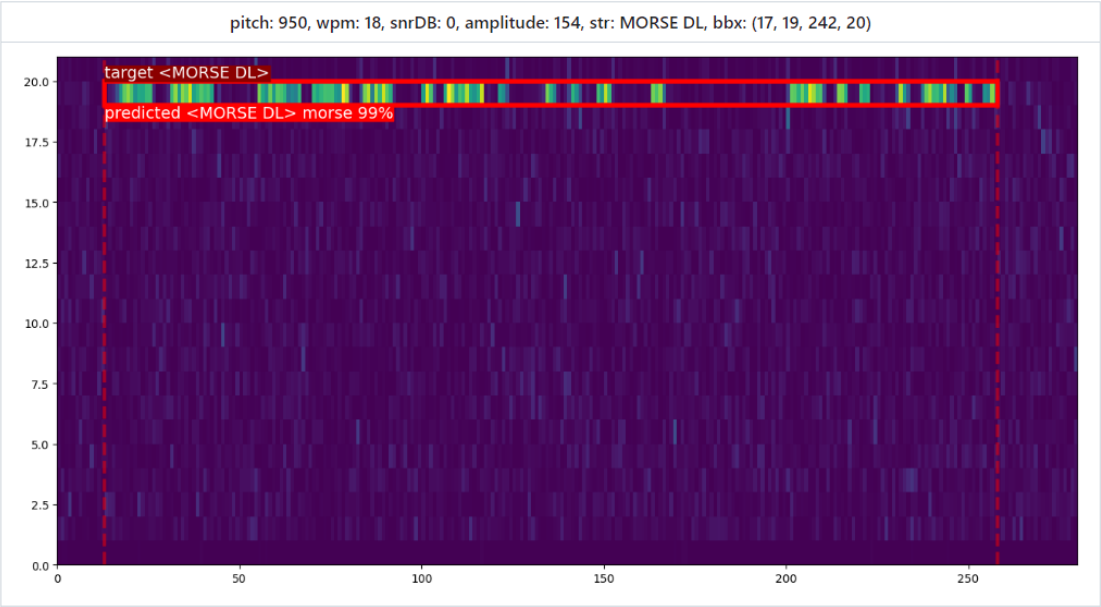
After undergoing 500 epochs of training, encompassing approximately 1,000,000 spectrograms, the model reached convergence. The model's accuracy, specifically concerning IoU-x axis, achieved an impressive 97% accuracy for SNR levels above -10dB.





4.4 Demo

We see here a demo that expresses both models (LSTM + RESNET50). The target string is shown here, "MORSE DL", compared to the predicted string with the parameters below.



4.5 Comparing to previous methods

4.5.1 Decode task

In comparison to the paper 'Research on Automatic Decoding of Morse Code Based on Deep Learning' [3] where a CER of around 90% is reported at -5dB with CNN, our improvement demonstrates 98% CER. Moreover, while their average accuracy stands at 95%, we elevate it to 98% on average, effectively addressing frequency drift(Doppler effects).

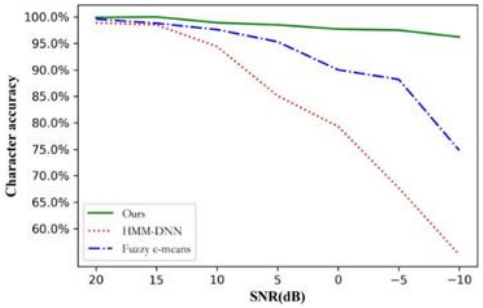


Fig. 3. Decoding accuracy under different SNR

TABLE II. DECODING ACCURACY UNDER DIFFERENT CONDITIONS

	Character Accuracy	Word Accuracy
Original signal	99.85%	99.30%
Frequency drift	95.79%	90.23%
Frequency drift + Manual deviation	94.67%	89.56%

4.5.2 Detection task

In contrast to the study outlined in the paper 'YFDM: YOLO for detecting Morse code' [4] which explores various IoU thresholds and achieves up to 95% accuracy, our analysis differs by focusing solely on the X-axis. They did not provide an analysis against Signal-to-Noise Ratio (SNR). However, for our purposes, we concentrate on the X-axis, neglecting the Y-axis as in practical communication, the main frequency is known, preventing Morse code from shifting in the spectrogram. Consequently, our approach yields an accuracy of 97% or higher.

5. Conclusion and Future Work

5.1 Conclusion

In this project, we addressed the challenge of Morse code detection by leveraging deep learning techniques, specifically LSTM-RNN and Faster-RCNN architectures. These models were trained on spectrogram representations of Morse code signals, enabling us to detect and decode Morse code from noisy and distorted real-world data effectively. Our experiments demonstrated promising results, with both models achieving high accuracy in decoding Morse code signals across varying signal-to-noise ratios.

The LSTM-RNN model, designed for Morse code decoding, exhibited an average Character Error Rate (CER) of less than 2% on signal-to-noise ratios above -5dB. Meanwhile, the Faster-RCNN model, tailored for Morse code detection, achieved an impressive accuracy of 97% in bounding box prediction (IoU-x), showcasing its ability to accurately localize Morse code signals within spectrograms.

5.2 Future Work

Moving forward, several approaches could be explored to further improve Morse code detection and decoding:

5.2.1 Complex Signal Modeling: Develop more sophisticated physical models for Morse signals that account for factors such as harmonics, pink noise, and spectral leakage. By incorporating these aspects into our models, we can better simulate real-world Morse code signals and enhance the robustness of our detection system.

5.2.2 Deeper Networks: Investigate the use of deeper neural network architectures, such as RESNET152, to further improve the performance of Morse code detection and decoding. Additionally, explore the potential benefits of bi-directional LSTM networks and conduct architecture grid-search to optimize model performance.

6. References

1. Ham Radio Blog by AG1LE [link1, link2, link3]
2. Ling YOU, Weihao LI, Wenlin ZHANG, Keren WANG. Automatic Decoding Algorithm of Morse Code Based on Deep Neural Network[J]. Journal of Electronics & Information Technology, 2020, 42(11): 2643-2648. doi: 10.11999/JEIT190658. [link]
3. Li, Weihao and Keren Wang. “Research on Automatic Decoding of Morse Code Based on Deep Learning.” 2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS) (2019): 488-491. [link]
4. Zhenhua Wei, Zijun Li & Siming Han. YFDM: YOLO for detecting Morse code [link]