HF Radio Signal Classification using Spectrograms and CNN with GA-NAS

Moditha Lekkala*, Kesava Sravan

Introduction

This project focuses on classification of HF radio signals using spectrogram images produced by the Short-Time Fourier Transform (STFT), and a Convolutional Neural Network (CNN).

The workflow converts raw complex IQ samples into 2D time-frequency representations, which are then fed into a CNN classifier. Future work will incorporate a Genetic Algorithm for Neural Architecture Search (GA-NAS) to optimize CNN layer hyperparameters.

Dataset

- Total signal vectors: $172,800$
- Each vector: 2048 complex IQ samples
- Duration per signal: 340 ms
- Sampling frequency: 6 kHz
- SNR levels in dataset: $\{25, 20, 15, 10, 5, 0, -5, -10\}$dB

Short-Time Fourier Transform (STFT)

STFT is used to obtain a time-frequency representation of non-stationary signals by computing the Fourier transform on short, overlapping windows of the signal.

- Window length: nperseg $= 2048$
- Python call example: f, t

  Sxx = stft(iq_samples, fs=fs, nperseg=nperseg, noverlap=noverlap)
- Output: frequency bins (f), time segments (t), STFT complex matrix (Sxx).

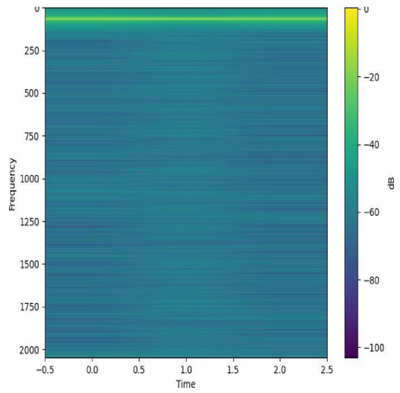
$$X(t,f) = \sum_{n=-\infty}^{\infty} x[n]w[n-t]e^{-j2\pi fn}$$

where:

- $x[n]$: discrete signal (IQ samples),
- $w[n-t]$: window function (e.g., Hamming or Hann window) centered at time t,
- f : frequency variable.

Spectrogram

A spectrogram visualizes the magnitude (or power) of frequency components over time (time on the x-axis, frequency on the y-axis, intensity as color). It is produced by taking the magnitude (or dB) of each STFT column and plotting as an image. Spectrograms are central features for the CNN classifier in this project.[3][4]



CNN-based Classifier (Pipeline)

- The baseline CNN consisted of multiple **convolutional layers** followed by **batch normalization**, **ReLU activations**, and **max pooling**, culminating in fully connected layers with a **softmax output** for multi-class HF signal classification.
- Training used **categorical cross-entropy loss** and **Adam optimization**, with a standard **train/validation/test split (70/15/15)** and **early stopping** to prevent overfitting. [1]

GA-NAS Integration

GA-NAS Process: Core Steps

-> Termination Condition Check

The algorithm checks if the stopping criterion is met. This could be a maximum number of generations (iterations), no improvement in fitness over several iterations, or reaching a target accuracy. If the termination condition is satisfied, the optimal architecture is found.

-> Selection (Survival of the Fittest)

If not terminated, the next step is **selection**, where the fittest individuals (highest-performing networks) are chosen as parents to produce the next generation. Methods like *tournament selection* or *roulette-wheel selection* ensure that better models have higher chances of being chosen while still maintaining diversity.

-> Crossover (Recombination)

In this stage, pairs of parent architectures exchange parts of their chromosomes to create offspring.

Example: Consider the Parent A: $[3,\ 3,\ 3,\ 3,\ 3]$ and the Parent B: $[2,\ 6,\ 4,\ 5,\ 3]$. The the offspring:

$$[3,\ 6,\ 4,\ 3,\ 3]$$

This combines beneficial traits (e.g., effective layer sizes or activation functions) from both parents.

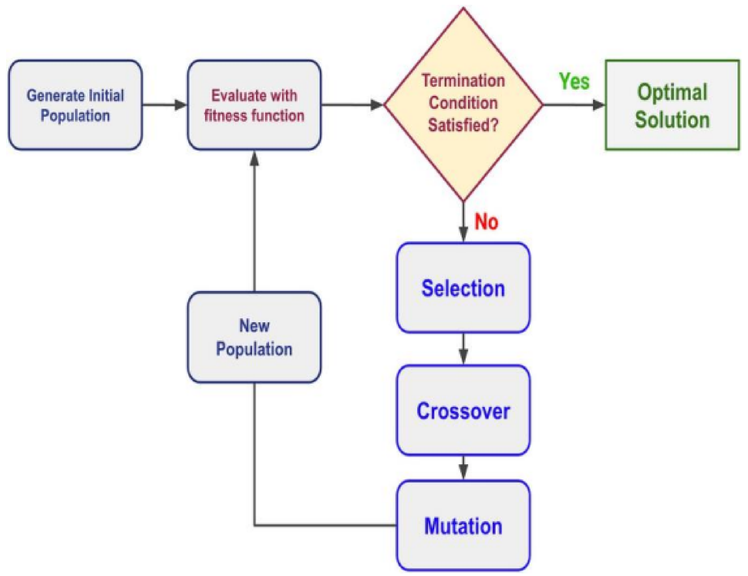
-> Mutation

To introduce variation and prevent stagnation, some offspring undergo random small changes in their parameters.

Example mutations:

- Changing an activation function (ReLU $\rightarrow$ GELU)
- Adjusting kernel size ($3 \rightarrow 5$)
- Modifying dropout rate ($0.5 \rightarrow 0.1$)

Mutation ensures continuous exploration of new architectures that might outperform the current best models.



Why GA-NAS Improves CNN Accuracy

1. Large Search Space Coverage

The space of possible CNN architectures is enormous (layers, filters, kernel sizes, activation functions, skip connections). GA-NAS searches globally and does not get stuck in local optima early, allowing it to discover unconventional but high-performing architectures.

2. Survival of the Fittest (Performance-Driven Search)

Only the best-performing CNN models survive each generation. This ensures that architectures evolve toward higher classification accuracy over time.

3. Crossover Combines Strong Traits

When two strong CNN architectures are recombined, the offspring often inherit effective structural features (e.g., deeper layers, better kernel configurations). This accelerates the discovery of architectures that outperform handcrafted models.

4. Mutation Introduces Novelty

Small random changes—such as altering kernel size or adding dropout—help explore new architectural possibilities that would not arise through human design alone. Some of these novel architectures significantly improve generalization and accuracy.

5. Automated Optimization of Hyperparameters and Structure

GA-NAS simultaneously optimizes multiple CNN components, such as depth, width, filter sizes, activation functions, and regularization layers. Optimizing these together leads to CNNs that are better matched to the dataset's complexity.[2]

6. Robustness Against Overfitting

The evolutionary process naturally encourages architectures that generalize well, because poorly generalizing models have lower fitness and are removed. This makes the final CNN more robust and accurate on unseen data.

Comparison Table

Aspect	Original Architecture	GA-NAS Architecture
Number of Layers	5	5
Neurons per Layer	[32, 32, 64, 64, 128]	[128, 64, 256, 128, 64]
Kernel Sizes	[3, 3, 3, 3, 3]	[2, 6, 4, 5, 3]
Activation Functions	ReLU, ReLU, ReLU, ReLU, ReLU	GELU, Tanh, ReLU, ReLU, Sigmoid
Dropout Rates	0.5	0.1
Overall Accuracy	93.77%	94.33%
Trainable Parameters	8,527,698	962,514
Network Size	97.7 MB	11.1 MB

Conclusions

- Spectrograms derived from Short-Time Fourier Transform (STFT) efficiently capture time-frequency features for HF signal classification.
- A CNN classifier trained on spectrogram images performs well; performance varies across SNR levels and benefits from data augmentation.
- GA-NAS can reduce model size while improving or matching accuracy by searching architectural hyperparameters using selection, crossover, and mutation.[5]

References

1. S. Scholl, *Classification of Radio Signals and HF Transmission Modes with Deep Learning*, 2019. Available: https://arxiv.org/abs/1906.04459
2. CMU ECE, "STFT Notes ADSP," Carnegie Mellon University. Available: https://course.ece.cmu.edu/ece491/lectures/L25/STFT_Notes_ADSP.pdf
3. Mathuranathan, "Spectrogram Analysis using Python," 2022. [Online]. Available: https://www.gaussianwaves.com/2022/03/spectrogram-analysis-using-python/
4. Demirkir and Sankur, "Object Detection Using Haar Feature Selection Optimization," *2006 IEEE 14th Signal Processing and Communications Applications*, Antalya, Turkey, 2006, pp. 1–4, doi: 10.1109/SIU.2006.1659787.
5. Xin Wang, "Moving window-based double Haar wavelet transform for image processing," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2771–2779, Sept. 2006, doi: 10.1109/TIP.2006.877316.

modithalekkala@gmail.com