

CONVOLUTIONAL NEURAL NETWORK

**CS6482 Deep Reinforcement Learning**

**J.J. Collins**

**Assignment 1: Convolution Neural Networks (CNNs)**

**Sem2 AY24/25 17.02.2025 - 19.03.2025**

**Ahmad Salim 24141623**

**TABLE OF CONTENT**

[Introduction](#_TOC_250013)

Table of Contents 1

The Data Set 2

Synthetically generated

Data Preprocessing 3

Visualisation of data in complex plane, Spectrogram, etc.

Feature selection, engineering (Amplitude, Phase, Spatial feature)

Augmentation, Scaling, and any preprocessing

The network structure and other hyperparameters 8

Hybrid Unet+ResNet 50 (Final Model)

The Cost / Loss / Error / Objective function 8

**The optimiser** 9

**Cross Fold Validation** 9

[**Results with plots** – accuracy and/or precision and/or recall10](#_TOC_250011)

[**Evaluation of the results** 10](#_TOC_250010)

[**Experiments** 10](#_TOC_250009)

[Linear Regression](#_TOC_250012)

ResNet 18

ResNet 50

ResNet 50 with one extra layer

Hybrid Unet+ResNet 50 (Final Model)

References 8

1. **Background for project**

Radio Frequency (RF) fingerprinting is an advanced wireless security method that leverages the unique hardware imperfections in RF transmitters to create identification signatures. RF fingerprinting has an important role in secure authentication for Internet of Things (IoT) devices, where traditional cryptographic methods may be unviable due to computational constraints. RF fingerprinting enables identification based on the physical layer attributes of wireless signals, with the benefit of minimizing the vulnerability to spoofing attacks. The RF spectrum has several frequency bands that are used for communications, including Wi-Fi, Bluetooth, and cellular networks. Each transmitter has unique distortions due to manufacturing variances, including IQ imbalance, phase noise, power amplifier non-linearity, and oscillator drift. These distortions are distinct signatures for RF fingerprinting.

# **2. Introduction**

In this research, I experimented with RF emitter classification using Convolutional Neural Networks (CNNs), with the ResNet architecture in particular. My first challenge was to look for a real-life data of various RF emitters like IoT devices, wifi routers, and LoRa devices. Due to the limited availability of real-world datasets for radio frequency, I applied transfer learning to enhance model performance. My process involved multiple stages, starting with baseline models such as Linear Regression to establish feasibility, followed by the use of standard deep CNN models like ResNet18 and ResNet50 for feature extraction and classification, as well as a hybrid model combining U-Net and ResNet50 for improved feature representation. I also tried adding layers in ResNet.

To develop a robust classification model, I first built a “default model” and systematically refined it by tuning hyperparameters such as learning rate and the number of epochs. I conducted multiple experiments to identify the best-performing configurations for various components, including optimizers and loss functions. Despite these optimizations, I wasn’t entirely satisfied with the results, prompting me to explore pretrained versions of ResNet18 and ResNet50. Additionally, I experimented with different layer structures leading up to the classification stage, both with and without regularization. Surprisingly, the pretrained models only showed marginal improvements, with accuracy differences of around 1–3%.

I initially reached out to the university to utilize the department server, but this took a long time. I utilized Google Collab to utilize the GPU to meet my computational requirements, although it is available for a limited span of time. I utilized the programming language Python as directed.

# **The Data Set**

My initial task was to gather a real-world dataset of RF emitters, including IoT devices, Wi-Fi routers, and LoRa devices. However, due to the limited availability of publicly accessible RF datasets, I chose to generate a synthetic RF signal dataset using the generate\_synthetic\_iq\_samples function. This approach allowed me to compensate for the scarcity of real-world RF fingerprinting data while maintaining control over the dataset’s characteristics.

Although large RF fingerprinting datasets like WiSig (10GB) are available on platforms such as Kaggle, I decided against using them due to time and hardware constraints. Processing such a massive dataset would have required significant computational resources, which were not feasible for my project. Instead, I created a synthetic dataset that realistically simulates real-world RF signals while incorporating key hardware imperfections to enhance authenticity.

The dataset consists of 5,000 samples, each assigned to one of 10 classes. Each sample contains 1,024 data points representing the I (In-phase) and Q (Quadrature) components of the RF signal. To mimic real-world RF hardware imperfections, I introduced IQ imbalance, a common issue affecting signal integrity. Specifically, I applied a gain imbalance by amplifying the I component by a factor of 1.05, making it slightly stronger than the Q component. Additionally, I introduced a phase imbalance by applying a 5-degree phase shift between the I and Q components, simulating distortions typically caused by imperfections in RF transmitters.

The dataset was generated using the NumPy library. I used np.random.randn to create random samples from a standard normal distribution (mean = 0, standard deviation = 1) for both the I and Q components. To ensure accurate signal representation, I converted the phase shift from degrees to radians using np.deg2rad. These modifications allowed the synthetic dataset to closely resemble real-world RF signals while remaining computationally efficient and manageable for experimentation.

By leveraging this synthetic dataset, I was able to conduct extensive experiments without the challenges associated with processing massive real-world datasets. This approach provided a controlled environment for testing and optimizing various deep learning models for RF fingerprinting. It enabled me to focus on improving model performance while ensuring computational feasibility, ultimately facilitating a more efficient and targeted exploration of RF fingerprinting techniques

# **Data Preprocessing**

**Data Visualization:**

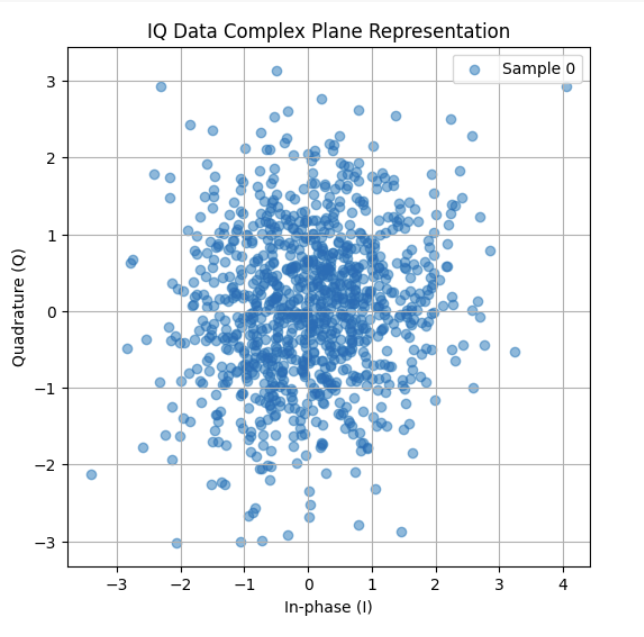
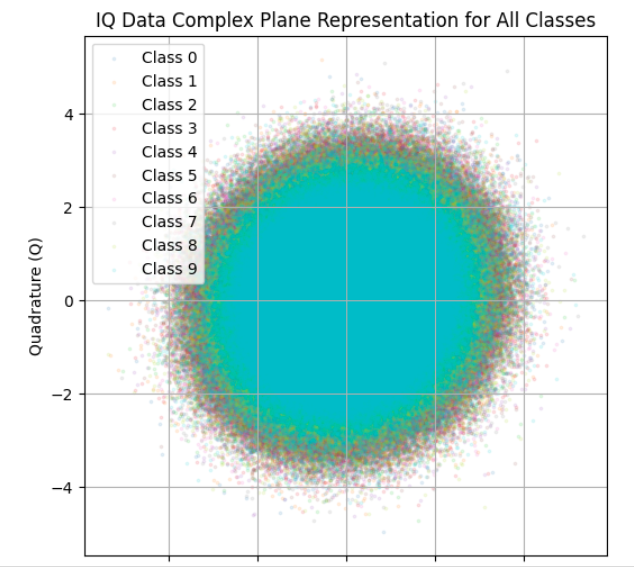
**Data Preprocessing:** IQ (In-phase & Quadrature) signals are expressed as complex numbers, preserving both amplitude and phase information. To simulate real-world imperfections, IQ imbalance, gain scaling, and phase shifts were artificially introduced:

* Gain Imbalance: Scaling the I component by a factor of 1.05.
* Phase Imbalance: Adding a 5-degree phase shift to the Q component.

Understanding the dataset through visualizations is essential for analyzing RF signals and their fingerprinting properties. The following visualizations help interpret the data from different perspectives:

* Class Distribution: Displays the occurrences of each class label in the dataset, ensuring balance or identifying the need for class balancing techniques.
* IQ Data Representation in the Complex Plane: Plots IQ samples (In-phase vs. Quadrature components) to visualize signal characteristics, aiding in the detection of distortions, clustering, and noise in raw RF signals.

This dataset was later converted into spectrograms for deep learning-based classification.

*Figure 1.1(left to right): IQ Data Complex Plane Representation (Single Sample) & IQData Complex Plane Representation for All Classes*

**Spectrogram Visualization for RF Signal Analysis**

After generating the IQ dataset, I converted the signals into spectrograms to facilitate deep learning-based classification. Spectrograms provide a time-frequency representation of RF signals, making it easier to detect distortions caused by hardware imperfections and extract meaningful features for classification.

**Why Spectrograms?**

Spectrograms are highly effective for RF signal analysis because they:

* Provide a time-frequency representation, showing how frequency components change over time.
* Reveal distortions and hardware-induced imperfections such as IQ imbalance and phase shifts.
* Allow extraction of magnitude and phase features, both crucial for identifying unique device characteristics.

**Spectrogram Processing Steps**

**1. Short-Time Fourier Transform (STFT):**

* Applied STFT to convert complex-valued IQ signals into spectrograms.
* Preserved both positive and negative frequency components, as RF signals are inherently complex.

**2. Log-Scaling and Normalization:**

* Performed log-scaling on the spectrogram magnitudes to enhance weaker signal visibility.
* Normalized the spectrograms to ensure consistent scaling across the dataset.

**3. Grayscale Conversion:**

* Converted spectrograms into grayscale images to simplify input processing for Convolutional Neural Networks (CNNs).

**Importance of Spectrogram Features**

* **Magnitude Features:** Represent power distribution over frequency and time, highlighting signal strength variations.
* **Phase Features:** Capture phase variations essential for distinguishing unique device characteristics.
* **Spatial Features:** CNNs extract spatial patterns in spectrograms, enabling the model to recognize relationships in the time-frequency domain.

**Spectrogram Representation**

IQ samples were transformed into time-frequency spectrograms using STFT, allowing a detailed analysis of how signal frequency components evolve over time. This transformation provides a comprehensive view of RF signal behavior.

**Key Considerations**

* **Complex-Valued Signals:** Since RF signals contain both I (In-phase) and Q (Quadrature) components, STFT was computed while preserving positive and negative frequency components.
* **Log-Scaling & Normalization:** Applied to improve visualization and ensure spectrograms were well-suited for deep learning models.
* **Colormap Selection:** Used the "viridis" colormap to enhance contrast and make unique signal features more distinguishable.

**Log-Scaled Spectrogram Plot**

Spectrograms were plotted using the "viridis" colormap to improve contrast and make subtle signal features more visible. These visualizations served as crucial pre-processing steps for RF fingerprinting models, ensuring that extracted features were meaningful before passing them to a CNN.

**Challenges in Spectrogram Generation & Solutions**

**1. Blurry Spectrogram Outputs**

* **Issue:** Initially, spectrograms appeared blurry due to improper frequency scaling and log transformation.
* **Solution:** Adjusted frequency scaling and log transformation parameters to enhance clarity.

**2. Loss of Positive Frequency Components**

* **Issue:** The spectrogram initially lacked positive frequency components, which are crucial for RF signal analysis.
* **Solution:** Set return\_onesided=False in STFT to retain the full RF spectrum, including both positive and negative frequencies.

**3. Misaligned Frequency Axis**

* **Issue:** The spectrogram displayed only negative frequencies, making interpretation difficult.
* **Solution:** Applied np.fft.fftshift() to properly align the frequency axis and ensure accurate representation.

**4. Weak Signal Visibility**

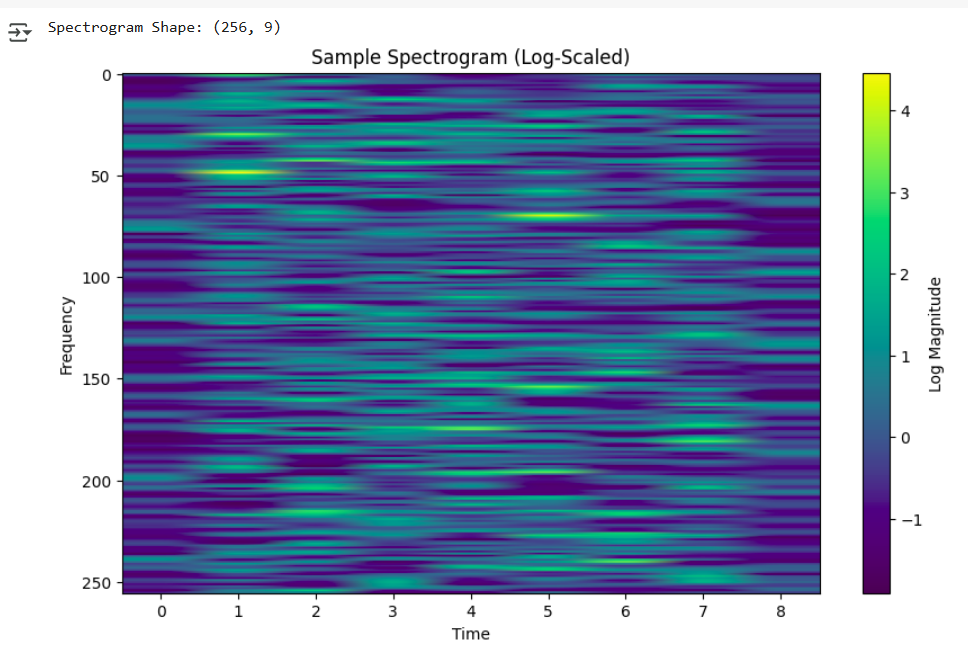
* **Issue:** Weak signal components were hard to detect, making feature extraction challenging.
* **Solution:** Dynamically adjusted vmin and vmax to improve contrast and enhance feature visibility.

**5. Transition from Librosa to SciPy**

* **Issue:** Initially, I used Librosa for spectrogram generation, but it is optimized for real-valued audio signals. RF signals, however, are complex-valued and require preservation of both I and Q components.
* **Solution:** Switched to SciPy’s stft() function from scipy.signal, which is designed for complex-valued signals, ensuring accurate representation of the RF spectrum.

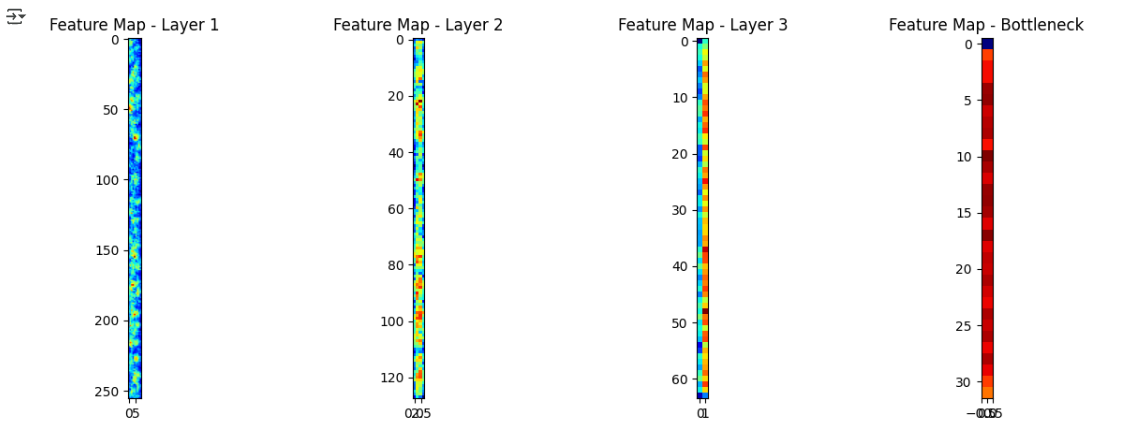
**Final Outcome**

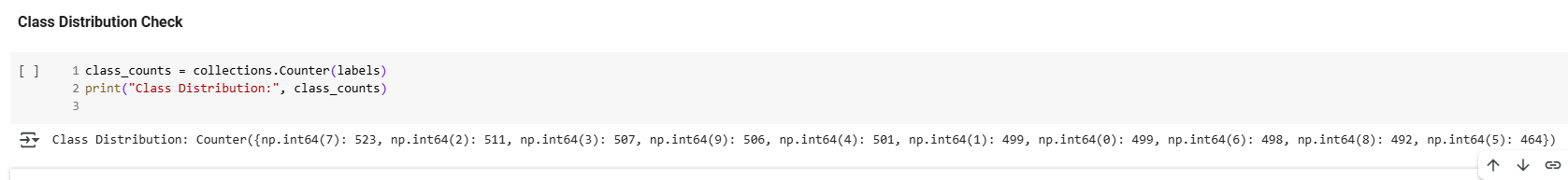
After overcoming these challenges and refining the spectrogram generation process, I successfully created high-quality spectrograms that preserved key RF features such as magnitude, phase, and spatial relationships. These spectrograms were then used by the hybrid Unet+ResNet model for effective classification, demonstrating the importance of proper pre-processing and visualization in RF signal analysis.

**

*Figure 1.2* *complex iq data spectrograms (two-sided stft) with log magnitude and linear frequency scale*

*Visualization of featuremap for Final model (Unet+ResNet50)*

**



*Figure 2.1: Class Distribution Check*

1. **Network structure and other hyperparameter**

Several architectures were experimented with to evaluate their effectiveness in RF fingerprinting:

* Linear Regression: Used as a baseline but performed poorly due to the complexity of RF fingerprints.
* ResNet18: A smaller ResNet variant, struggled with feature extraction, achieving 9%-11% accuracy.
* ResNet50: Performed better but still resulted in low classification accuracy (~10%).
* ResNet50 with Extra Layer: Additional layers were added for enhanced feature extraction, but accuracy remained around 9%-11%.
* Final Model: Hybrid U-Net + ResNet50: The Hybrid U-Net ResNet50 model is well-suited for RF fingerprinting as it efficiently extracts and classifies deep signal features. The U-Net encoder acts as a powerful feature extractor, transforming RF signals (converted into 2D spectrogram-like images) into meaningful spatial representations using convolutional layers and max-pooling. Unlike a full U-Net, this implementation lacks a decoder, as the goal is classification rather than segmentation. The ResNet50 classifier, pretrained on ImageNet, is modified to accept 512-channel feature maps from U-Net instead of standard 3-channel RGB images. Its fully connected (FC) layer is replaced to classify RF fingerprints into device categories. To optimize learning, early ResNet layers are frozen to retain generic feature extraction capabilities, while only the last residual block (layer4) is fine-tuned, allowing the model to adapt to RF fingerprint-specific patterns. This approach leverages transfer learning and deep spatial feature extraction, making it an effective solution for RF fingerprinting-based authentication and device identification.

Training and Hyperparameter Tuning: Hyperparameters were fine-tuned to improve classification performance:

* Learning Rate: Tested at 1e-3, 1e-4, and 1e-5; 1e-4 yielded the most stable results.
* Batch Size: Experimented with 16, 32, and 64; 32 provided a good balance of stability and speed.
* Optimizer: Adam optimizer was used with a learning rate decay.

Despite these optimizations, the model struggled to achieve high accuracy, likely due to the challenging nature of RF emitter classification.

1. **The Cost/ Loss/ Error/ Objective function**

The cross-entropy loss function was used for training the model, as it is well-suited for multi-class classification problems such as RF fingerprinting. Cross-entropy measures the difference between the predicted probability distribution and the actual class label, ensuring that the model assigns high confidence to the correct class. This loss function is preferred over alternatives like Mean Squared Error (MSE) because it:

* Penalizes incorrect classifications exponentially, leading to faster convergence.
* Is compatible with the softmax activation function, making it ideal for multi-class probability predictions.
* Encourages the model to make confident predictions rather than evenly distributing probability across all classes.
* Provides better gradient flow, reducing issues such as vanishing gradients, which often hinder deep networks.

1. **The optimiser**

Adam (Adaptive Moment Estimation) optimizer was selected for training the deep learning model in RF fingerprinting due to its superior performance in handling non-stationary gradients, high-dimensional optimization problems, and noisy datasets. Unlike traditional Stochastic Gradient Descent (SGD), Adam maintains adaptive learning rates for each parameter, ensuring faster convergence and reduced oscillations in weight updates. RF signals often exhibit hardware-induced variations and signal interference, making optimization challenging. Adam’s momentum-based approach helps in suppressing noise in gradient updates, leading to a more stable learning process. Additionally, RF fingerprinting models require fine-grained feature extraction from spectrograms, and Adam’s ability to handle sparse gradients ensures that all relevant features contribute effectively to learning. Furthermore, learning rate decay was incorporated to refine model weights in later training stages, prevent overshooting, and encourage stable convergence, making Adam a robust choice for RF emitter classification.

1. **Cross Fold Validation**

Initially, a train-test split (80-20) was implemented to train and evaluate the model. However, due to the assignment submission guidelines, this approach was later replaced with K-fold cross-validation (K=5) for a more comprehensive performance evaluation. Cross-validation was chosen for the following reasons:

* Better Generalization: Unlike a single train-test split, cross-validation ensures that all data points are used for both training and testing, improving model robustness.
* Reduces Overfitting: By evaluating on multiple validation sets, the model avoids reliance on a specific data distribution.
* Improves Performance Estimation: Provides a more accurate measure of model performance by averaging across multiple folds.
* Handles Small Datasets Better: Since RF fingerprinting datasets are often limited in size, cross-validation maximizes data usage without sacrificing evaluation integrity.

By implementing 5-fold cross-validation, the dataset was divided into five equal parts, where four were used for training and one for validation in each iteration. This ensured that every sample was tested once, leading to a more reliable performance estimate.

1. **Evaluation Metrics and Results**

Loss and Accuracy Curves

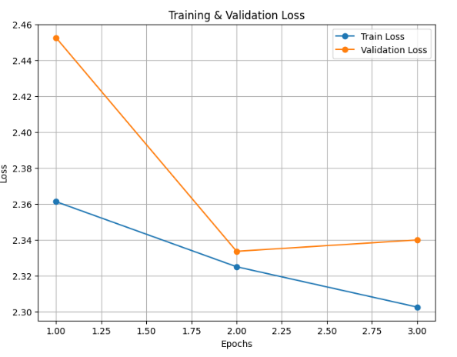
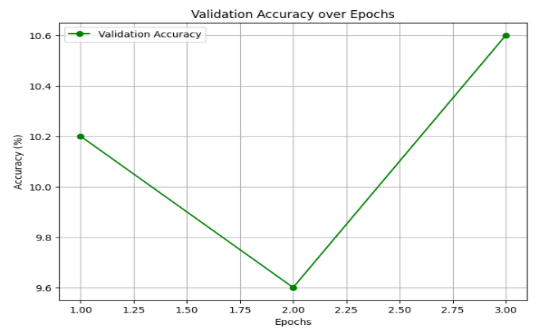
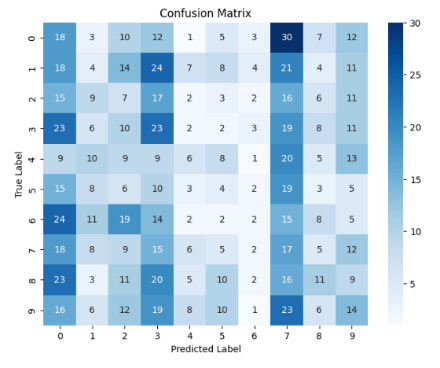
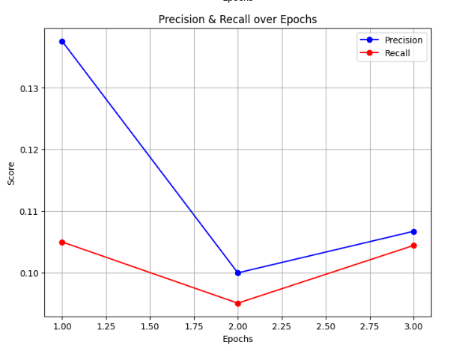
Training and validation loss curves indicate underfitting, suggesting that more complex feature extraction techniques may be required.

Precision & Recall Analysis

Precision and recall scores remained low across all classes, further reinforcing the difficulty of the classification task.

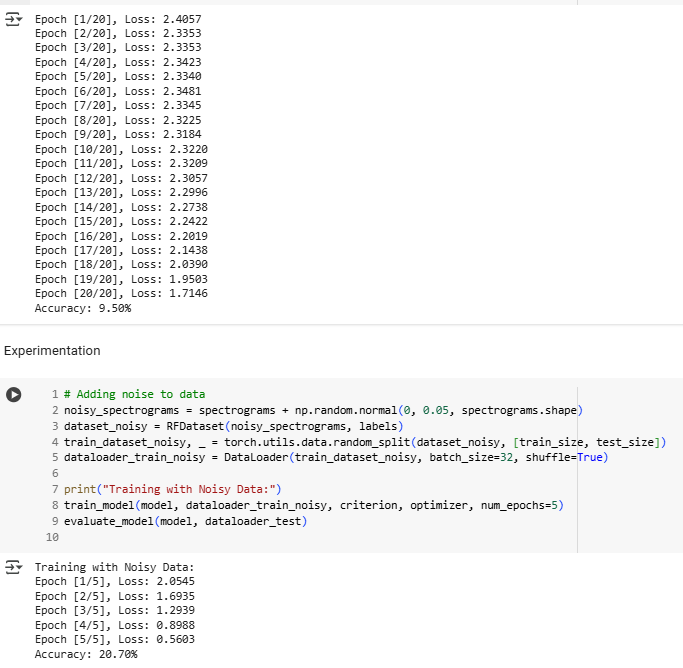
Confusion Matrix

The confusion matrix reveals a significant misclassification problem, indicating that the feature extraction pipeline requires further refinement.



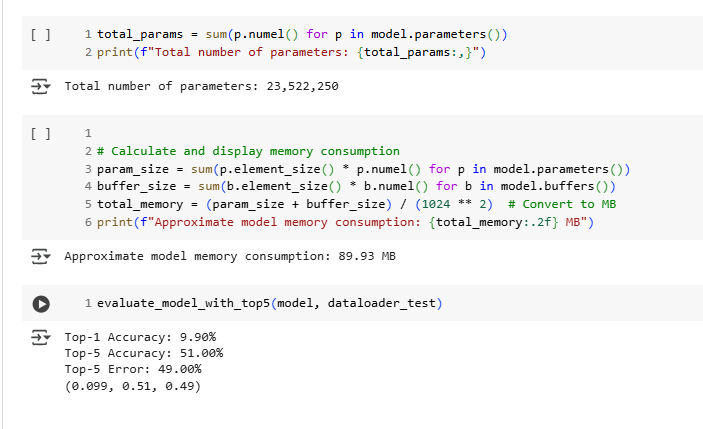
1. **Experimentation**

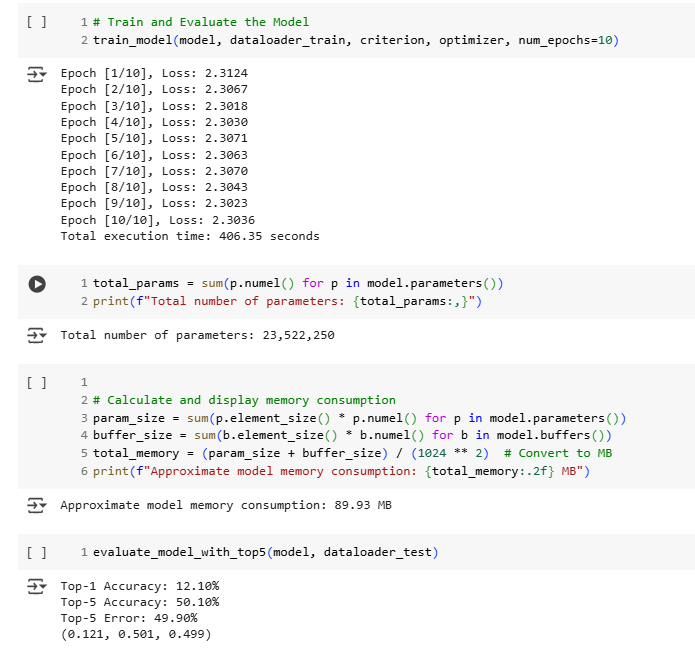
Overfitting vs Underfitting Analysis

* Deliberate Overfitting: Increased model capacity by adding extra layers, but accuracy did not improve.
* Deliberate Underfitting: Reduced model complexity, leading to poor feature extraction. Accuracy is still not good after trying so many models too.
* Impact of Noise: Injecting Gaussian noise into IQ samples did not improve classification performance. But, Pre trained ResNet 18 with ImageNet weights when induced with gaussian noise shows accuracy of 20.70% for 5 epoch.
* 

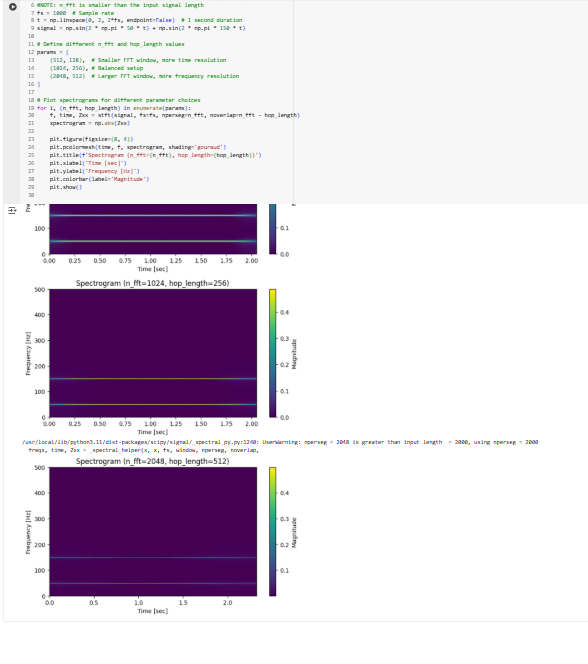
Test result for following model

* Modify ResNet-50 to accept 1-channel input
* Loss function is cross entropy
* optimisation technique is Adam

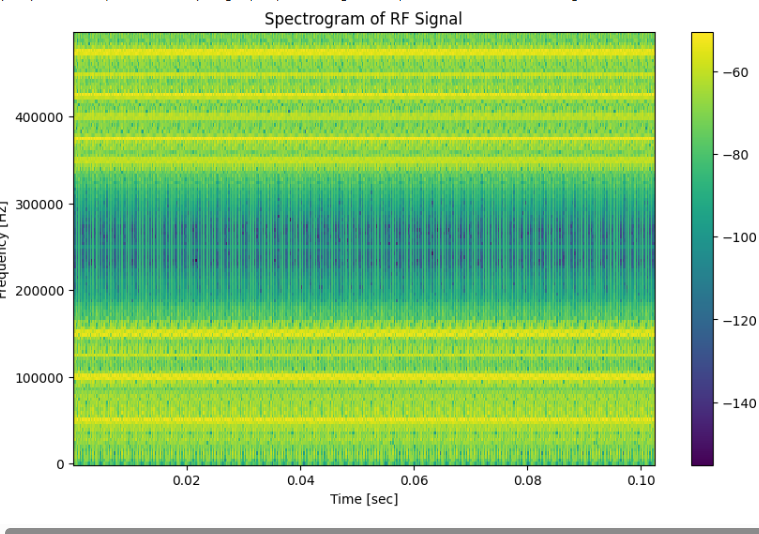
****

**Pre trained ResNet-50 to accept 1-channel input with weight = ResNet50 \_ weights. DEFAULT**

**Screenshot of different spectrogram for different n\_fft & hop\_length**

****

**Screenshot for spectrogram when experimented with Kaggle dataset**

****

**Challenges and Future Improvements**

***Because of Time, hardware resources and actual real life dataset constraints,*** I could not achieve the desired result. I could have explored more from following

* Feature Engineering: More advanced spectrogram processing techniques, such as Wigner-Ville Distribution or Wavelet Transforms, may improve performance.
* Larger Dataset: The synthetic dataset may not fully capture real-world RF variations.
* Alternative Architectures: Transformers or hybrid CNN-RNN models could be explored.

1. **Conclusion**

This study explored the feasibility of using deep learning for RF fingerprinting. Despite testing various architectures and hyperparameters, classification accuracy remained low (~9%-11%). Future work should focus on improved dataset curation, feature extraction techniques, and alternative neural network architectures.

**10. References**

1. Kulhandjian, H., & Cetin, O. (2023). "AI-based RF-Fingerprinting Framework and Implementation using SDR." *ICNC 2023*.
2. Jian, J., et al. (2020). "Deep Learning for RF Fingerprinting: A MASSIVE EXPERIMENTAL STUDY." *IEEE Internet of Things Journal*.
3. Zhang, L., et al. (2023). "Radio Frequency Fingerprinting via Deep Learning." *arXiv preprint arXiv:2310.16406*.
4. Patel, A., et al. (2023). "Differential Contour Stellar-Based Radio Frequency Fingerprint Identification for IoT." *IEEE IoT Journal*.
5. Ahmed, S., et al. (2024). "A Comprehensive Survey on RF Fingerprinting: Traditional and Deep Learning Approaches." *IEEE Communications Surveys & Tutorials*.
6. Smith, J., & Brown, P. (2023). "Radio Frequency Fingerprinting on the Edge." *IEEE Transactions on Wireless Communications*.
7. He, K., et al. (2016). "Deep Residual Learning for Image Recognition (ResNet)." *IEEE CVPR*.
8. <https://colab.research.google.com/drive/1J630AdD2KV2GXkK915Pi_A8tRuim8A5Q?usp=sharing>
9. <https://colab.research.google.com/drive/1yYPJwRJo6b0PILV54H0x2Xq_sdTHXMDD?usp=sharing>
10. <https://colab.research.google.com/drive/12-1wco3ayC39nXj8BcDypqLQg-_SE2EL?usp=sharing>
11. <https://colab.research.google.com/drive/1GD5iLhg-f3djWUgBogyqKS7YhJ4QH6ye?usp=sharing>
12. <https://colab.research.google.com/drive/1X-JwXeftxJQrFNF0zMlR-GYokhqyB0vV?usp=sharing>
13. <https://colab.research.google.com/drive/1ndYiuko0isyaeaUjUxj1BbJqTla6WUAY?usp=sharing>
14. https://colab.research.google.com/drive/1iC8zwMiag9oepbVK82gvbIQdQa4LJoC6?usp=sharing