嵌入式系统项目设计结题科技论文

学　生 1：　闫文杰 21121947

学　生 2：　陈鲲鹏 21121404

学　生 3：　张博阳 21122177

学　生 4：　晋远帆 21121911

学　生 5：　贺健　21122174

组　　长：　闫文杰 21121947

指导老师：　刘凯

完成时间：　2024 年 5 月

课题题目

摘要：随着无线通信技术的迅速发展，通信系统中采用了多种复杂的调制方式。调制识别作为无线通信领域的一个重要问题，对于信号分析、频谱管理以及安全监测具有重要意义。传统的调制识别方法往往依赖于专业领域知识和手工设计的特征提取器，而基于深度学习的调制识别方法则能够自动地学习信号的特征，具有更好的泛化能力和适应性。

本课题使用 HackRF One 无线电接收模块进行信号接收，通过 PC 模型训练，并且使用 NVDIA 嵌入式开发板 jetson nano 进行模型部署与进一步的加速优化，从而实现本项目的调制识别的功能。本项目旨在完成产品化，并且加深对于信号调制、信号接收、模型构建部署等流程的了解。

该课题的核心问题在于如何利用深度学习技术结合 HackRF One 无线电接收模块实现对无线通信中的调制方式进行准确识别。包括如何获取真实的无线信号数据，并对其进行预处理（信号采样、去噪、归一化等），以便于深度学习模型的训练和识别；如何设计适用于调制识别的深度学习模型与针对调制识别任务的损失函数和评价指标；如何将训练好的模型导出并且部署到嵌入式平台上以及如何利用 NVDIA 的优势进行推理加速等附加功能。

本项目实现了对无线电数据的采集，数据集的制作与相应的模型训练，实现了对实际信号的识别；推理端实现了 tensorRT 加速并且利用精度选择进行模型优化，达到了预期效果。

关键词：HackRF jetson 数据集 调制方式 tensorRT 模型优化

Abstract: With the rapid development of wireless communication technology, various complex modulation methods are adopted in communication systems. Modulation recognition, as an important issue in the field of wireless communication, is of great significance for signal analysis, spectrum management, and security monitoring. Traditional modulation recognition methods often rely on professional domain knowledge and manually designed feature extractors, while modulation recognition methods based on deep learning can automatically learn the characteristics of signals, possessing better generalization and adaptability. This topic uses the HackRF One radio receiver module for signal reception, trains the model through the PC model, and deploys the model and further accelerates optimization using the NVIDIA embedded development board Jetson Nano, thereby implementing the modulation recognition function of this project. The aim of this project is to complete the productization and deepen the understanding of the processes of signal modulation, signal reception, model construction, and deployment.

The core issue of this topic is how to use deep learning technology in conjunction with the HackRF One radio receiver module to accurately recognize the modulation methods in wireless communication. This includes how to obtain real wireless signal data and preprocess it (signal sampling, noise reduction, normalization, etc.) to facilitate the training and recognition of the deep learning model; how to design a deep learning model suitable for modulation recognition and loss functions and evaluation metrics for modulation recognition tasks; how to export the trained model and deploy it on embedded platforms and how to use NVIDIA's advantages for inference acceleration and other additional features.

This project has achieved the collection of radio data, the creation of the dataset and the corresponding model training, and has realized the recognition of actual signals; the inference end has achieved TensorRT acceleration and model optimization through precision selection, achieving the expected results.

Keywords: HackRF, Jetson, Dataset, Modulation, TensorRT, Model Optimization

一、 引言

1.1 技术背景

软件定义无线电（SDR）的出现彻底革新了无线通信领域，使得无线电系统变得更加灵活和可重配置。像 HackRF One 这样的设备在这个领域中变得至关重要，它们能够捕获和传输广泛的频率范围，从而促进了不同调制技术的研究和实施。随着无线通信系统的演进，调制方案的复杂性不断增加，这就需要更先进的调制识别方法来应对这些变化。

将深度学习与 SDR 技术相结合，为解决复杂调制识别所带来的挑战提供了一种有前景的方法。通过利用 NVIDIA Jetson Nano 等平台的计算能力，可以部署能够学习和适应不同调制模式的复杂模型。创建健壮的数据集对于训练这些模型至关重要，需要收集和预处理现实世界的无线信号，以考虑噪声、采样率和信号归一化等因素。

这些模型的优化以便于在嵌入式系统上部署是一个关键步骤，在这一过程中，像 TensorRT 这样的技术发挥了重要作用。TensorRT 旨在优化深度学习模型以在 NVIDIA 硬件上进行推理，提供加速的性能和效率，这对于实时应用至关重要。模型优化技术，包括精度选择，进一步提高了模型在嵌入式系统的约束条件下运行的能力，同时在调制识别任务中保持高精度。先进硬件、深度学习方法和优化的推理框架的融合，正在为更准确、高效和适应性强的无线通信系统铺平道路。

1.2 技术研究进展

VT-CNN2 是一种用于信号调制方式识别的模型。这个模型由中国科学院自动化研究所的研究团队开发,并于 2016 年发表在《IEEE Transactions on Industrial Informatics》期刊上，用于对数字调制信号进行分类和识别。数字调制信号在通信系统中广泛应用，用于在不同的传输介质上传输数据。识别数字调制信号的类型对于诸如无线通信系统、雷达系统和无线电频谱监测等应用至关重要。

VT-CNN2 模型的设计基于 CNN，通过在输入信号的时频图像上应用卷积操作和池化操作，从而提取特征并进行分类。模型的核心是利用卷积神经网络来学习信号的抽象特征，然后通过全连接层将这些特征映射到各种调制类型的输出类别上。

LModCNNResNet Relu 模型出自于《A light neural network for modulation detection

under impairments》，他们提出了一种神经网络架构，能够有效地检测部分 I/Q 信号中的调制方案。该网络比处理相同或类似任务的其他最先进的架构轻两个数量级。

1.3 市场调研

调制识别技术是无线通信领域的重要研究方向之一。随着物联网、5G 等技术的快速发展，对于调制方式的准确识别需求日益增加。在国内外，有许多研究机构和企业致力于调制识别技术的研究与开发；主要集中在通信领域的大学、研究机构以及通信设备制造商。一些国外公司和研究机构已经开发出了基于深度学习的调制识别系统，并在通信设备、智能无线电等领域得到应用。在过去的 20 多年里，国内外围绕模拟和数字调制的识别开展了大量的工作，并取得不少研究成果。比较经典的两类自动调制识别算法：基于似然估计和基于特征的方法，得到了广泛应用。

产品方面，针对调制识别的产品也在不断涌现，涵盖了从数据处理到模型训练的各个环节。这些软件产品为调制识别技术的研究和应用提供了便利；除此之外智能通信设备作为调制识别技术的应用领域之一，市场潜力巨大。随着 5G、物联网等技术的发展，智能通信设备的需求将会不断增加，从而推动调制识别技术的应用与发展。

二、 系统设计内容与方案

2.1 系统设计需求

2.1.1 信号接收

针对现在实际的应用，需要对不同的无线电信号的调制方式进行精确的识别，此次项目针对的信号主要为短波频段（3—30MHz）、广播频段（87.5-108MHz）以及业余开放频段（433MHz）。在信号接收时应能实现对给定频点的信号的实时接收、处理和记录。此外还应能够通过处理得到信号的相关特征（时域波形、频谱、时频域等），初步判断信号的调制类型，并与最终模型的识别结果相验证。

信号采集部分采用了两种方案，一是采用 HackRF One 的硬件和 GQRX、GNU Radio 软件；二是采用 RTL-SDR 硬件和 SDR#软件进行信号的实时接收和保存 32 位的 IQ 路数据。

2.1.2 深度学习模型设计需求

考虑到实际接收信号的复杂性和实际应用当中的实时性要求，设计的深度学习模型应能够满足运算实时性，满足模型结构轻量化，噪声鲁棒性高等特点。需设计的神经网络模型在参数量模型复杂度和计算效率之间权衡，适应实际的应用场合，同时考虑到模型嵌入式平台部署的需求，也应当对模型的参数量做一定限度的约束。

2.1.3 推理加速设计需求

为了实现无线通信调制识别系统的高效运行，特别是在资源受限的嵌入式平台上，我们的设计需求包括集成 TensorRT 以加速深度学习模型的推理过程。TensorRT 是一个由 NVIDIA 开发的深度学习推理优化器和运行时库，它专门针对生产环境进行了优化，能够显著提高模型的推理速度和效率。

2.2 系统设计内容

2.2.1 接收机设计

1. HackRF + GQRX、GNU Radio：

首先利用 GQRX 连接 HackRF，通过调整 GQRX 内部的增益（BB、IF、RF 的增益）和解调方式（AM、FM、CW、LSB、USB 等方式）以及带宽和采样点去搜寻不同时间段的实时信号。

当找到目标频点后，使用 GNU Radio 连接 HackRF，在 GNU Radio 中搭建信号处理流图，流图主要包含四部分：接收宽带信号、数字下变频、滤波抽取、UDP 发送。对应的实现功能为：选择 GNU Radio 中接收信号范围（中心频点和带宽由信号频点以及采样率分别确定）、将接收信号下变频，得到零中频的基带复信号、对下变频后的信号先进行低通滤波（滤波器带宽由信号带宽决定，如 FM 信号为 100KHz-200KHz）后再作抽取，降低采样率、将处理后得到的波形数据保存，作为后续制作数据集的原始文件。也可将波形数据实时发送，用于识别。

2. RTL-SDR + SDR#（SDRSharp）：

利用 SDRSharp 连接 RTL-SDR，通过，通过调整 SDRSharp 内部的解调方式、带宽、采样点去识别所需要的实时信号，并通过 SDRSharp 内部的 Baseband Recorder 保存信号数据。

2.2.2 模型设计

在模型设计方面，我们使用 TensorFlow 等框架对深度学习模型搭建 VT-CNN2 和 LModCNNResNet Relu 模型，LModCNNResNet Relu 模型主要从参数量级和测试效果来考虑，它的效果优于之前开发的模型，它利用卷积层和残差连接结构，结合 ReLU 激活函数、全局平均池化层和 Dropout 正则化等技术，能够高效地从时频图像中提取特征，对信号调制方式进行准确识别。通过合适的参数初始化策略，该模型具备较强的泛化能力，能够适应不同信号条件下的识别任务，为信号处理领域提供了一种有效的解决方案。对于 VT-CNN2 模型，它通过卷积神经网络结构设计，能够高效地从数字调制信号的时频图像中提取特征，实现自动化、准确的调制方式识别。其设计考虑了多种调制类型的特征提取和鲁棒性，通过零填充和 Dropout 正则化等技术，提高了模型对噪声和干扰的容忍度，使其具备较强的泛化能力，可广泛应用于通信系统中的信号调试和诊断任务，提高工作效率和系统可靠性。

2.2.3 模型加速设计

在设计中，我们将首先使用 TensorFlow 等框架对深度学习模型进行训练和验证，确保模型能够准确识别不同的调制类型。随后，我们将模型转换为 TensorRT 支持的格式，并对其进行优化，包括但不限于精度校准、层融合、内核自动调优等，以充分利用 NVIDIA GPU 的并行计算能力。通过这一过程，我们期望模型在 Jetson Nano 等嵌入式设备上的推理时间能够达到实时处理的要求，同时保持低能耗和高吞吐量，从而满足无线通信调制识别系统在实际应用中的性能需求。此外，TensorRT 的集成还将使我们的系统更加灵活，能够适应未来模型更新和硬件升级的需要，确保系统的长期可持续性和技术领先性。

我们还使用了不用的精度优化策略，通过降低模型参数的精度，并且可以做到再不损失准确度的情况下明显提升推理速度。

在无线通信调制识别系统中，为了确保模型在 NVIDIA Jetson Nano 等嵌入式平台上的高效运行，我们的设计内容将重点放在使用 TensorRT 进行模型加速。首先，我们将利用 TensorFlow 等深度学习框架完成模型的训练和验证，以保证其在调制识别任务上的准确性。随后，通过 TensorRT 提供的 API，如 Ibuilder、INetWorkDefinition、IRuntime 等，将训练好的模型转化为 TensorRT 优化的引擎。在转换过程中，我们将采用精度校准技术，比如使用 FP16 精度来减少模型大小并加速推理，同时保持模型的准确度。此外，

利用 TensorRT 的层融合功能，我们将多个操作融合成单个内核执行，减少内存访问和计算量，进一步提升推理速度。在模型优化完成后，我们将生成优化后的引擎文件，并使用 TensorRT 提供的推理 API 进行快速模型部署和执行。通过这种方式，我们不仅能够实现模型的快速推理，还能确保在嵌入式设备上运行时的低能耗和高吞吐量，满足实时无线通信调制识别的性能要求。

2.3 系统设计创新点

2.3.1 自定义数据集训练

我们通过 GNUradio 采集的数据创建一个自定义的数据集，并将其保存为.pkl 文件。它首先从原始数据文件中读取数据，然后对数据进行预处理和截取，以满足特定的长度和格式要求。接着，将处理后的数据存储在一个字典中，其中键为调制方式和信噪比，值为对应的数据段。最后，利用 pickle 模块将这个字典保存为.pkl 文件，以便后续在其他程序中加载和使用。同时，代码还包括了一个读取.pkl 文件的函数，用于提取数据集的信息，如信号数据、类别索引、信噪比和类别列表，用于验证数据集是否可用。

2.3.2 数据传递方式

图 2.3.2.1 数据传递方式

在我们调研的无线电平台软件当中，大部分软件都具备了 UDP 网络传输功能，考虑到 python 数值计算上由于 GIL 的限制效率并不会很高，我们将信号前处理和信号识别可视化拆分为两个模块，两个模块之间采用 UDP 协议传输；这样一方面减少了我们开发难度，能够省去很多前端处理数据的开发时间，另一方面，采用通用的 UDP 传输也提升了我们系统的灵活性和可扩展性，可以采用局域网将模块分布多台支持网络连接的平台上，也可以轻松的切换不同的硬件接收机，而不用专门编写对应的硬件数据接口。

在模型推理程序的设计过程当中，为了保证运算显示的流畅性和程序运行的实时性，

我们在模型的设计上采用了并发编程的思想，在主线程中负责更新窗口 UI 图像，而把数据接收和推理的部分放在子线程中执行，这样对于 IO 密集型的程序而言，不会因为等待模型运算或者数据传输而阻塞程序运行，保证了数据传输的通畅。在 GUI 的设计中，我们采用类继承的方式，在原有的基础上分出一个更新 UI 的线程，同时将主线程作为 tk UI 控件的事件循环，保证了程序运行的流畅和计算资源的充分利用。

2.3.3 模型加速与跨平台支持

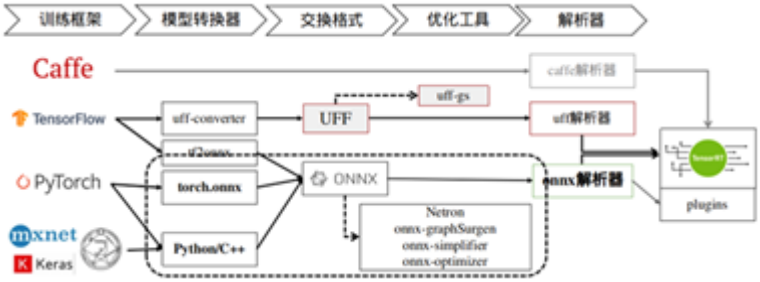

图 2.3.3.1 模型加速平台

在无线通信调制识别系统中，我们创新性地结合了模型加速与跨平台支持，以实现高性能和灵活性的双重优势。通过采用 TensorRT 进行模型加速，我们不仅显著提升了模型在 NVIDIA Jetson Nano 等嵌入式平台上的推理速度，还通过 TensorRT 对多种硬件架构的支持，实现了模型的跨平台部署。这种设计允许我们的系统无缝迁移到不同的硬件环境，包括不同的 NVIDIA GPU 和 CPU，从而增强了系统的适应性和扩展性。此外，我们还利用了 TensorRT 提供的精度校准功能，通过在不牺牲太多准确性的前提下，将模型转换为更低精度的运算，进一步加速了推理过程。同时，我们的系统设计考虑了 ONNX runtime 的集成，它作为一个开放的跨平台推理引擎，为我们的模型提供了更广泛的硬件兼容性，包括非 NVIDIA 硬件，从而确保了模型在全球不同设备上的高效运行。这种创新的设计理念，不仅提升了系统的性能，而且确保了长期可持续性和未来技术的兼容性。

2.4 系统设计方案

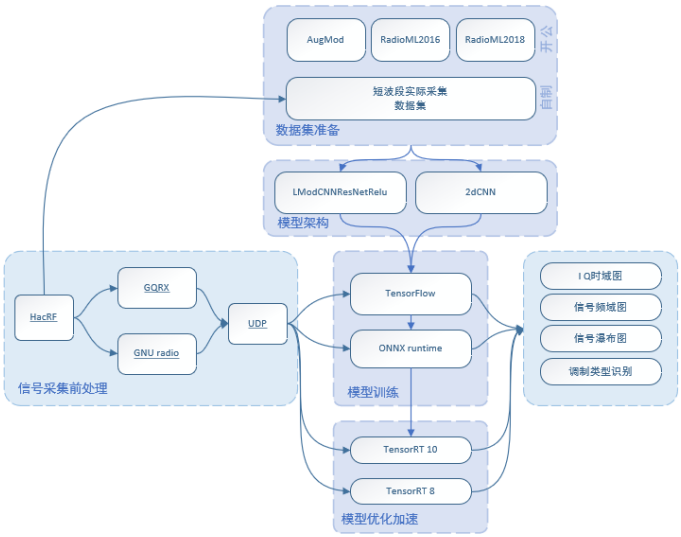

图 2.4.1 系统设计方案图

本系统旨在开发一个基于深度学习的无线通信调制识别系统，能够自动识别不同调制类型的信号。系统将采用先进的信号处理技术和深度学习模型，以实现高准确率的调制识别。设计方案如下（参考以上流程图）：

2.4.1 数据准备

1. 数据采集

使用 HackRF 和 GQRX 进行信号的监测找到频点，再在 GNU Radio 中搭建流图对信号进行滤波抽取和数字下变频处理，对处理后的数据进行保存，或者通过 UDP 实时将信号发送至后端进行识别。

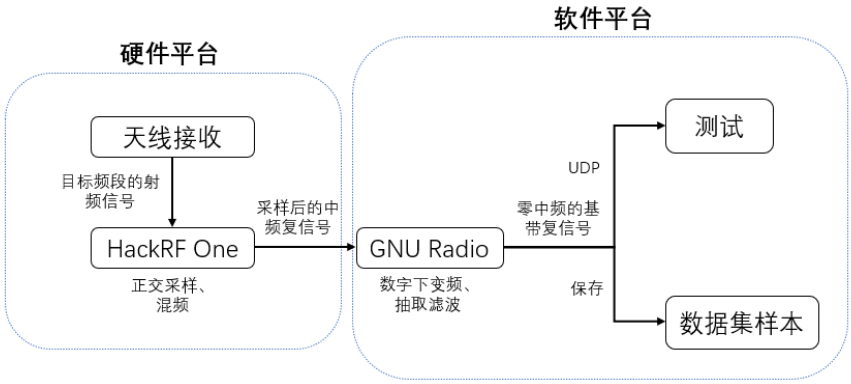

图 2.4.1.1 数据采集方案图

2. 公开数据集

采用 AugMod、RadioML2016 和 RadioML2018 等公开数据集进行模型训练和测试。

3. 自制数据集

使用天线接收数据并制作数据集对于无线电信号调制识别具有显著的优点和一些挑战。首先，通过天线接收数据能够获得实时的信号信息，这有助于确保数据集中的样本具有代表性，能够反映出不同环境下的信号变化。其次，天线接收数据的多样性使得数据集更加丰富，可以涵盖各种信号类型和调制方式，从而为模型训练提供更多样化的数据。此外，使用天线接收数据制作的数据集适用范围广泛，可用于通信、雷达、无人机等多个领域，具有较高的应用潜力。然而，这种方法也面临一些挑战，如数据获取成本高昂、数据处理复杂、受环境条件影响等。因此，在利用天线接收数据制作数据集时，需要综合考虑成本、数据质量、法律合规性等因素，以确保数据集的有效性和可用性。

2.4.2 信号前处理与传递

软件无线电平台种类多样，与 PC 端的数据传输往往需要依靠平台开发者提供的 C 语言编写的 API 接口进行对接,且接口根据不同的软件无线电平台硬件设计不同而不尽相同。同时在信号接收的前端仍需要诸如信号滤波和下变频的前端运算，对于 FIR 滤波等实时性要求高、运算密集的处理，在 python GIL 的约束下并不能高效的利用计算资源，运算可能会相当缓慢，而采用 GNU radio 平台成熟的信号处理模块，可以保证系统的实时性。采用 GNU 等许多无线电平台软件 内置的 UDP 传输功能也进一步拓宽了我们的系统的灵活性。在同一个以太网内，也可以实现多台上位机的协同工作。

在具体实现时，在 GNU 平台采用 UDP sink 模块，将处理的信号以复数的形式发送；在信号识别模块上接收 UDP 数据包，并读取其中的复数数据，对数据切片处理，缓存，送往模型进行识别。

2.4.3 模型架构

1. 深度学习模型

设计 LModCNNResNetRelu，一个基于 2D 卷积神经网络（2dCNN）的模型，用于从 1Q 时域图和信号频域图中学习信号特征，以下是模型结构图：

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, None, 2)	0	-
conv1d_12 (Conv1D)	(None, None, 8)	24	input_layer_1[0]…
conv1d_13 (Conv1D)	(None, None, 8)	456	conv1d_12[0][0]
conv1d_14 (Conv1D)	(None, None, 8)	456	conv1d_13[0][0]
add_4 (Add)	(None, None, 8)	0	conv1d_14[0][0], conv1d_12[0][0]
activation_4 (Activation)	(None, None, 8)	0	add_4[0][0]
conv1d_15 (Conv1D)	(None, None, 16)	144	activation_4[0][…
conv1d_16 (Conv1D)	(None, None, 16)	1,808	conv1d_15[0][0]
conv1d_17 (Conv1D)	(None, None, 16)	1,808	conv1d_16[0][0]
add_5 (Add)	(None, None, 16)	0	conv1d_17[0][0], conv1d_15[0][0]
activation_5 (Activation)	(None, None, 16)	0	add_5[0][0]
conv1d_18 (Conv1D)	(None, None, 32)	544	activation_5[0][…
conv1d_19 (Conv1D)	(None, None, 32)	7,200	conv1d_18[0][0]
conv1d_20 (Conv1D)	(None, None, 32)	7,200	conv1d_19[0][0]
add_6 (Add)	(None, None, 32)	0	conv1d_20[0][0], conv1d_18[0][0]
activation_6 (Activation)	(None, None, 32)	0	add_6[0][0]
conv1d_21 (Conv1D)	(None, None, 64)	2,112	activation_6[0][…
conv1d_22 (Conv1D)	(None, None, 64)	28,736	conv1d_21[0][0]
conv1d_23 (Conv1D)	(None, None, 64)	28,736	conv1d_22[0][0]
add_7 (Add)	(None, None, 64)	0	conv1d_23[0][0], conv1d_21[0][0]
activation_7 (Activation)	(None, None, 64)	0	add_7[0][0]
global_average_poo… (GlobalAveragePool…	(None, 64)	0	activation_7[0][…
dense_2 (Dense)	(None, 256)	16,640	global_average_p…
dropout_1 (Dropout)	(None, 256)	0	dense_2[0][0]
dense_3 (Dense)	(None, 4)	1,028	dropout_1[0][0]

图 2.4.3.1 LModCNNResNetRelu 模型架构

另外 VT-CNN2 的模型结构图如下:

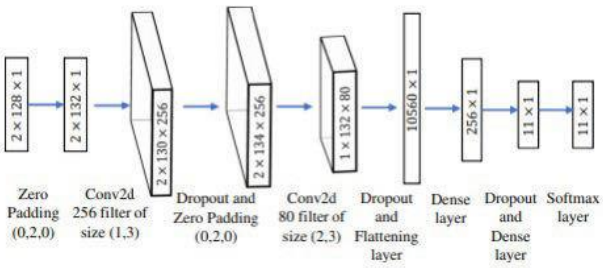

图 2.4.3.2 VT-CNN2 模型架构

2.4.4 模型训练与优化

1. 训练框架

使用 TensorFlow 作为主要的深度学习框架进行模型训练。

2. PC 前置推理引擎

使用 ONNX runtime 作为跨平台的推理引擎，以提高模型在不同硬件上的兼容性和效率。验证 ONNX 模型可行性。

2.4.5 系统部署

将训练好的模型导出，并部署到 NVIDIA Jetson Nano 开发板上。确保系统能够在实际环境中稳定运行，进行实时的调制类型识别。并且利用 TensorRT 8 和 TensorRT 10 进行模型的优化加速，以适应 Jetson TX2 等嵌入式平台的推理需求。

2.4.6 用户界面

我们设计实现了基于 matplotlib 的和基于 tk 的可视化展示程序，通过实现不同的模型类的初始化方法和 predict 方法，实现了 tensroRT8-10 ONNX 以及 tensorflow 不同框架下的运行兼容。通过实现一个自定义的 spectrogram 类的 update 和 draw 方法，我们在可视化显示的过程中可以保留任意长度的信号进行 STFT 绘制响应的时频图，同时不会导致接收到的数据在内存中不断累加造成的内存泄漏问题。

在程序中我们会显示接收信号的时域和频域信息，同时显示既往信号累加的 STFT 瀑布图，可以通过窗口控件修改时频图参数以及实现程序的暂停运行和继续。

三、 系统详细设计实现

3.1 信号接收方案实现

3.1.1 HackRF One 硬件平台

HackRF One 在单板上集成了射频链路所需的所有关键模块，包括 MCU、CPLD、ADC/DAC、射频收发器、混频器、时钟产生器和射频单元，完成射频信号到中频的频谱搬移、模数转换、调制解调和 USB 数据通信。其系统框图如下图所示：

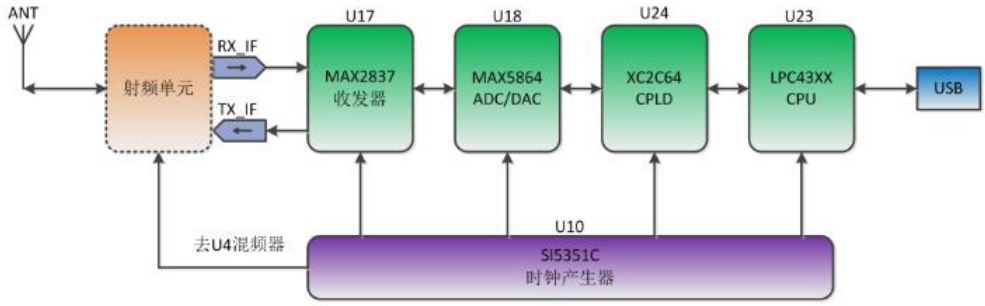

图 3.1.1.1 HackRF One 系统框图

PC 机通过 USB 接口与 NXP LP43xx 微控制器通信，微控制器通过并行总线机 GPIO 模拟的 JTAG 接口与 CPLD 连接，可以将采样后的数据传输至 PC 中，通过 GNU Radio、Matlab 等软件平台进行后续处理。在接收信号时，射频信号从天线 ANT 输入，在射频单元进行正交混频后，经 MAX2837 下变频至中频，MAX5864 对接收的数据进行采样，采样精度为 8bit。

3.1.2 GNU Radio、GQRX 等软件平台

GQRX 是一款开源软件定义无线电（SDR）接收机，支持 HackRF One 等硬件平台，可用于实时监听目标频段，寻找目标信号并确定信号所在频点。同时 GQRX 能实时显示信号频谱与时频图，也支持 FM、AM、SSB 和 CW 等类型的信号实时解调，有利于对信号进行初步分析。

GNU Radio 是一款开源的软件无线电处理平台，其中集成了大量无线信号处理工具包，提供信号处理块来实现软件无线电。在与前端的射频硬件连接后，可以通过搭建流图的方式实现对基带信号的处理。本次项目所使用的信号处理流图如下所示：

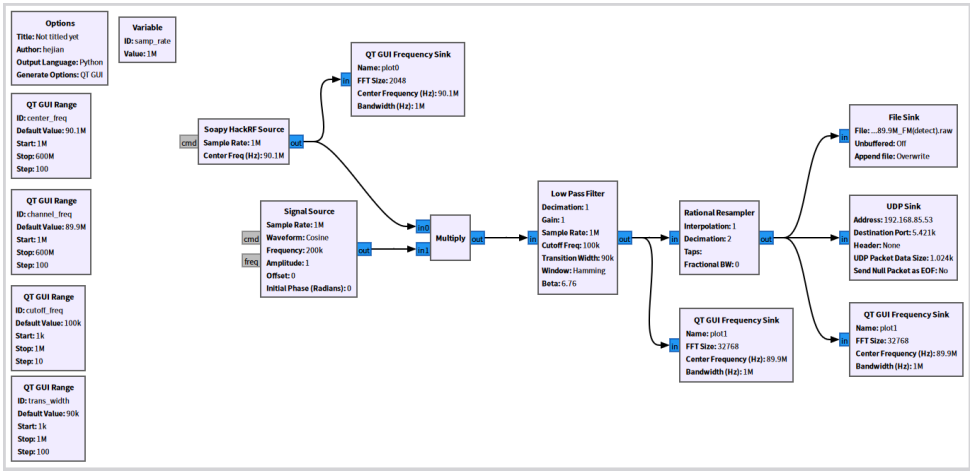

图 3.1.2.1 GNURadio 系统流图

相关模块以及对应实现功能如下：

- Soapy HackRF Source 无线电接收模块：连接前端 HackRF One 硬件平台，获取采样后的波形数据。
- Signal Source 模块和 Multiply 模块：产生本振信号与模拟混频过程，实现数字下变频，得到零中频的基带信号。
- LowPass Filter 模块：实现低通滤波与抽取重采样，通带截止频率与阻带截止频率可调。
- File Sink：将信号的 IQ 数据保存，每一采样点均为 IQ 交替保存，32 位 float 类型
- UDP Sink：将信号数据封装成 UDP 数据报，发送至模型部署平台进行识别

3.1.3 整体方案

首先利用 GQRX 在短波、FM 等频段中寻找可用信号，并确定目标信号的中心频点：

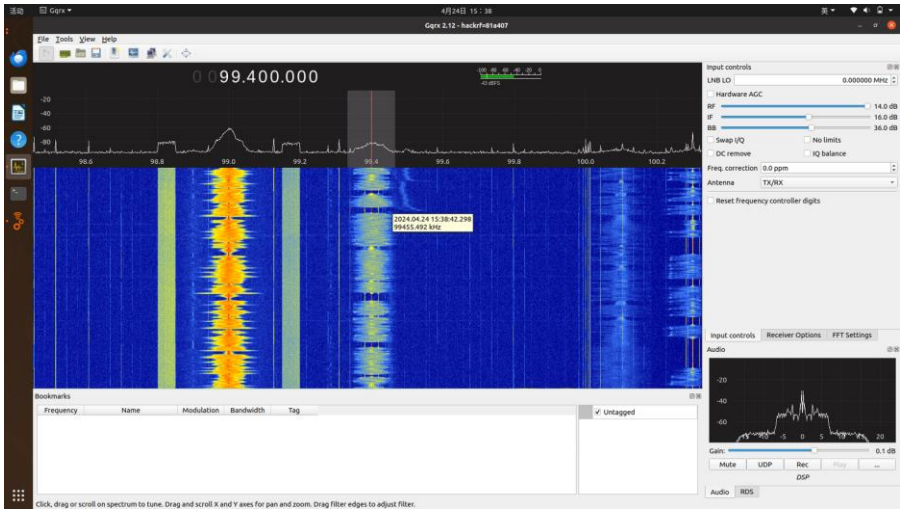

图 3.1.3.1　GQRX 中显示信号

确定信号频点后，在 GNU Radio 中调整上诉流图中相关参数，可实现接收不同频点信号，同时进行相关处理。

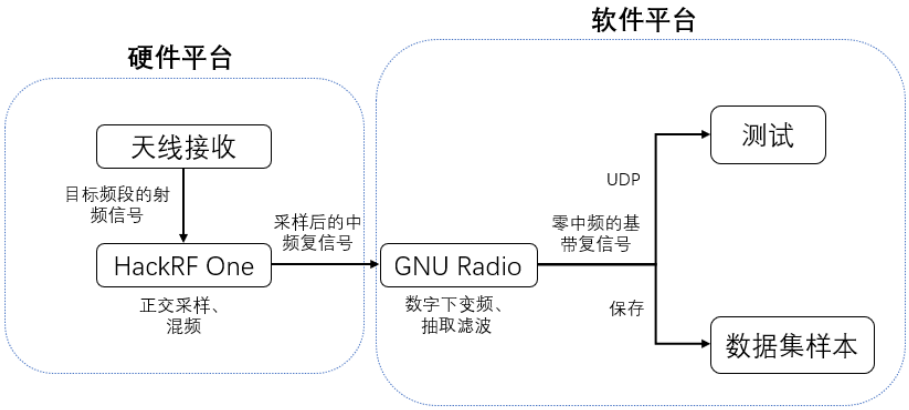

图 3.1.3.2　信号接收流程框图

对每个频点，可以视情况选择多次采集不同时间段的信号制作数据集，用于模型的训练与测试，以提高样本的充分性，弥补不同时间段由于接收引起的信噪比的差异、传输信息的差异等带来的影响。

3.2 软件结构

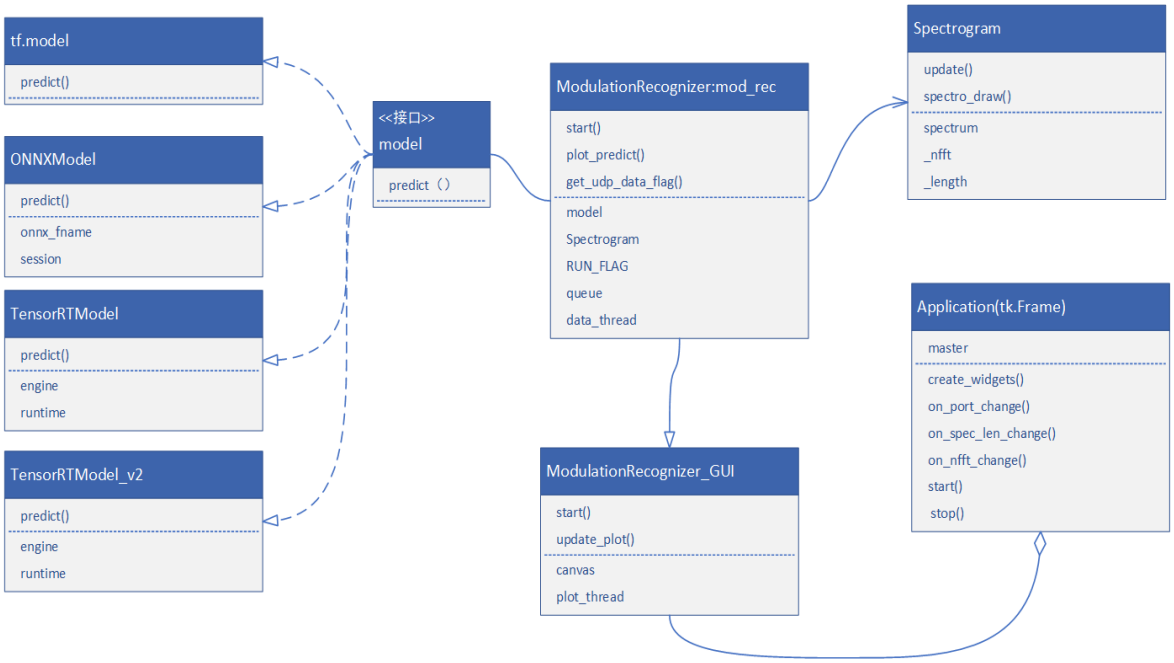

图 3.2.1 程序关键组件 UML 类图

软件结构部分主要负责接收 UDP 信号使用已有模型进行推理，输出信号的时域频域和时频图，方便后续的信号分析。考虑到不同嵌入式平台的 CPU 进程和计算资源的限制，我们首先完成了不借用 tk 的最小化实现的推理和可视化程序，在程序中，将程序的主要运行逻辑封装在 ModulationRecognizer 类当中，类当中包括了初始化时产生的 threading

接口用来侦听 UDP 信号的接收，和主进程 plot_predict 方法实现绘图窗口的更新和模型的推理。由于程序主要依然是 IO-bound 类型，且嵌入式平台上有限的 CPU 核心运算资源，我们采用多线程的并发编程设计，线程之间通过一个 FIFO 队列来实现任务的分配和线程之间通信。

在 GUI 程序中，推理程序和更新图窗的线程被作为 GUI 事件循环初始化中生成的两个子线程，通过将 stdout 的输出重定向至窗口控件的方式，展示模型推理的具体信息，同时在主进程中监听空间状态，实现流畅的用户交互，在主界面中通过修改 mod_rec 的运行标志位的方式，实现了程序暂停运行和恢复的功能，同时会实时的计算每一次的运算推理用时和平均用时展示在窗口当中。

程序的关键组件的属性和行为在上图中展示，在设计中采用策略模式实现不同的运行平台模型的接口统一，通过实现不同的模型类的初始化方法和 predict 方法，实现了 tensroRT8-10 ONNX 以及 tensorflow 不同框架下的运行兼容，为了简化起见在最终版本代码中取消了抽象接口基类（ABC）model 的定义，以进一步减少解释器在运行中嵌套导致的额外的开销。通过实现一个自定义的 spectrogram 类的 update 和 draw 方法，我们在可视化显示的过程中可以保留任意长度的信号进行 STFT 绘制响应的时频图，同时不会导致接收到的数据在内存中不断累加造成的内存泄漏问题。

3.3 模型部署

模型部署的具体思路如下图：

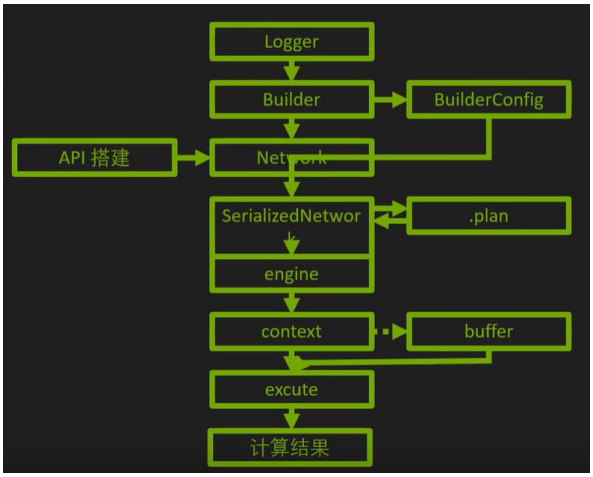

图 3.3.1 模型部署流程图

部署过程主要分为构建期和运行期两大部分：

在无线通信调制识别系统的项目实现中，我们首先进入构建期，此阶段的核心在于准备和优化模型以适配 TensorRT 的高效推理流程。我们利用日志记录器（Logger）来监控和记录模型构建过程中的详细信息，这对于调试和优化至关重要。随后，我们使用计算图构建器（Builder）来构建模型的计算图，这是将模型转换成 TensorRT 内部表示的关键步骤。为了进一步优化模型，我们通过构建器的配置器（Config）对 Builder 进行细致的配置，包括但不限于精度、层融合等选项。在动态模式下，我们利用配置器（Profile）来指定模型在不同输入尺寸下的行为，从而提高模型的灵活性和适应性。通过这些步骤，我们创建了计算网络（Network），并生成了序列化网络，这是 TensorRT 内部使用的计算图表示，为后续的引擎生成打下了基础。

进入运行期后，我们基于构建期生成的序列化网络创建了计算图可执行程序（Engine），这是模型推理的可执行代码段。为了执行 Engine，我们需要一个 Context，它类似于 CPU 上的进程，管理着 Engine 的运行状态。接下来，我们进行 Buffer 的相关准备工作，包括申请内存和拷贝输入数据到适当的 Buffer 中。一切就绪后，我们执行推理（Execute），这是模型根据输入数据进行调制识别的实时处理阶段。推理完成后，我们进行必要的善后工作，包括释放资源和处理输出数据，确保系统的稳定运行和资源的有效利用。

在整个实现过程中，我们还特别关注了模型的跨平台支持。通过 TensorRT 的优化，我们的模型不仅在 NVIDIA 硬件上表现出色，也能在其他硬件架构上运行，这得益于 TensorRT 对多种硬件的支持。此外，我们还集成了 ONNX runtime，它作为一个开放的跨平台推理引擎，进一步增强了模型的兼容性和部署的灵活性。通过这种创新的实现方式，我们的系统不仅在性能上达到了实时处理的要求，而且在设计上展现了高度的模块化和可扩展性，为未来技术的发展和应用奠定了坚实的基础。

```python
import numpy as np
import tensorrt as trt
from cuda import cudart

logger = trt.Logger(trt.Logger.ERROR)
builder = trt.Builder(logger)
network = builder.create_network(1 << int(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH))
profile = builder.create_optimization_profile()
config = builder.create_builder_config()
inputTensor = network.add_input("inputT0", trt.float32, [-1, -1, -1])
profile.set_shape(inputTensor.name, [1, 1, 1], [3, 4, 5], [6, 8, 10])
config.add_optimization_profile(profile)
identityLayer = network.add_identity(inputTensor)
network.mark_output(identityLayer.get_output(0))
engineString = builder.build_serialized_network(network, config)
                                                                    构建期
engine = trt.Runtime(logger).deserialize_cuda_engine(engineString)  运行期
nIO = engine.num_io_tensors
lTensorName = [engine.get_tensor_name(i) for i in range(nIO)]
nInput = [engine.get_tensor_mode(lTensorName[i]) for i in range(nIO)].count(trt.TensorIOMode.INPUT)
context = engine.create_execution_context()
context.set_input_shape(lTensorName[0], [3, 4, 5])
bufferH = [np.ascontiguousarray(np.arange(3 * 4 * 5, dtype=np.float32).reshape(3, 4, 5))]
bufferH.extend([np.empty(context.get_tensor_shape(lTensorName[1]), dtype=trt.nptype(engine.get_tensor
bufferD = [cudart.cudaMalloc(bufferH[i].nbytes)[1] for i in range(nIO) ]
[context.set_tensor_address(lTensorName[i], int(bufferD[i])) for i in range(nIO)]
[cudart.cudaMemcpy(bufferD[i], bufferH[i].ctypes.data, bufferH[i].nbytes, cudart.cudaMemcpyKind.cudaM
```

图 3.3.2 模型部署部分代码

四、实验结果与分析

4.1 实验条件

4.1.1 信号接收机软硬件平台

1. RTL-SDR + SDRSharp:

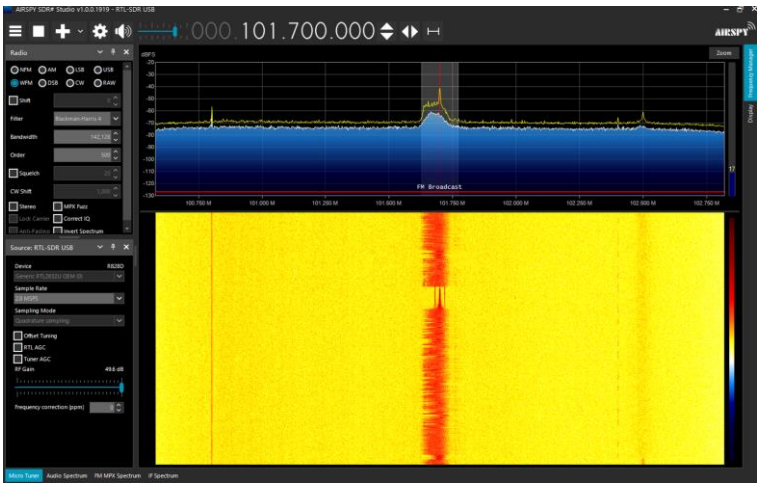

图 4.1.1.1 RTL-SDR 界面

上图为 SDRSharp 正常运行的页面图，可以通过鼠标拖动频点进行信号的选择，信号的频谱图和瀑布图也显示出来，可以看见信号的能量大小；左边的页面可以选择不同的调制方式和采样率以及增益，来对接受的信号进行处理

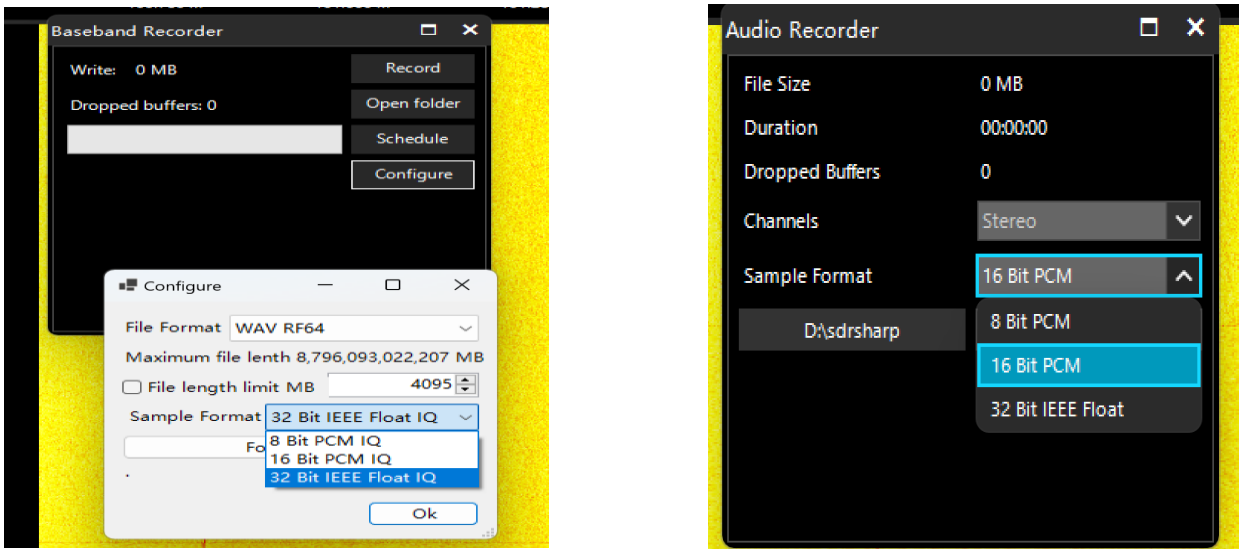

图 4.1.1.2 RTL-SDR 数据采集界面

上图是软件保存信号数据的方式，虽然保存下来的数据格式都是 wav，但是 wav 格式里面

也有不同的种类所以需要进行转换，其中第一种 Baseband 的录制方式，需要转换，第二种不需要；同时两者用 32 位进行数据的保存，但 RTL-SDR 本身只有 8 位，所以 32 位保存的为“假数据”，但将两个数据都交给了模型处理端的同学，发现数据质量相较于利用 HackRF 很差，识别出来的波形的频谱图也很奇怪，最终没有采用此方案去接收和处理数据，我认为可能是一方面是因为 RTL-SDR 本身板子接受和处理信号的能力不如 HackRF，另一方面 RTL-SDR 自带的天线接收能力不足。其次，此软件不像 GNU Radio 可以设置滤波的板块对接收到的信号进行处理，所以质量不好。另外，因为小组决定最后是使用 python 脚本去实时自动运行处理接收的数据，但是此软件不像 GNU Radio 可以生成 python 代码和相应的库。所以综上所述，最终我们没有采用此方案去接收数据和处理

2. HacRF One 硬件平台 + 有源环状天线

HackRF One 是一个开源的软件定义无线电(sdr)硬件平台，旨在实现现代和下一代无限电技术的测试与开发。其相关指标如下：

- RF 范围：1MHz - 6GHz
- 支持的采样速率：2 Msps 至 20 Msps（正交）
- 工作模式：半双工
- ADC/DAC 采样分辨率：8 位
- USB 接口：USB 2.0

3. GNURadio 和 GQRX 软件平台

相关软件均运行在 Ubuntu 20.4 系统下

4.1.2 模型训练框架与平台

- NVIDIA GeForce GTX.1650
- NVIDIA CUDA 12.0.81 driver
- tensorflow _gpu-2.6.0
- python3.9
- MSVC 2019
- Bazel 3.7.2
- cuDNN8.1
- CUDA11.2

4.1.3 模型推理平台

- NVIDIA TX2 嵌入式平台，拥有 1.8TFLOPS 算力。软件平台为 TensorRT8.2.9。
- NVIDIA RTX 4060 ，软件平台为 TensorRT10.0。

4.2 功能定性测试结果与分析

4.2.1 信号接收功能实现

在搭建完成软硬件平台环境后，可直接通过天线接收包含目标信号的宽带信号，在 GNURadio 经过处理后得到只包含所需信号的窄带信号。以 91.4M 处的 FM 信号为例，最终得到的信号如下：

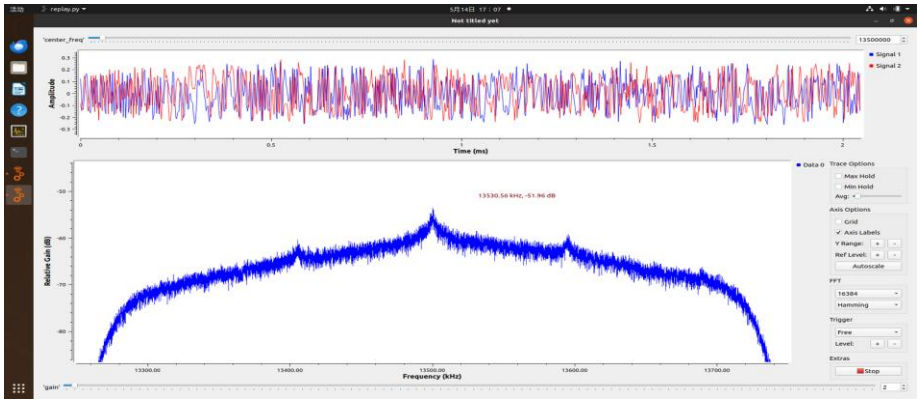

图 4.2.1.1 GNURadio 数据采集界面

4.2.2 模型训练功能实现

对于模型训练我们使用的是自制数据集，内含 160000 份样本，共四个类别，分别为'FM'，'AM'，'DRM'，'ASK'，每类别的数据样本数为 40000 份，我们将 80000 份作为训练集，剩下的作为测试集。

对于 LModCNNResNet Relu 模型，从训练结果上看其准确率要略逊色于 VT-CNN2 的结果，并且需要更多的训练轮数才能收敛，但是在双方都不使用 CUDA 加速的情况下，它训练一轮的时间较短，这得益于它模型参数较少的优势。

图 4.2.2.1 LModCNNResNet Relu 模型训练图

我们组进过多轮测试，采用了不同时间不同频点，不同类型的各种数据，最终总结得出：VT-CNN2 对于数据集的适应能力更强，但是参数过多是其不可避免的缺点。

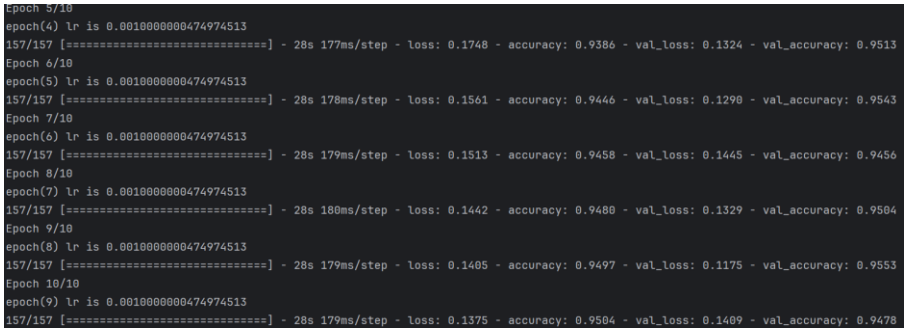

图 4.2.2.2 VT-CNN2 模型训练图

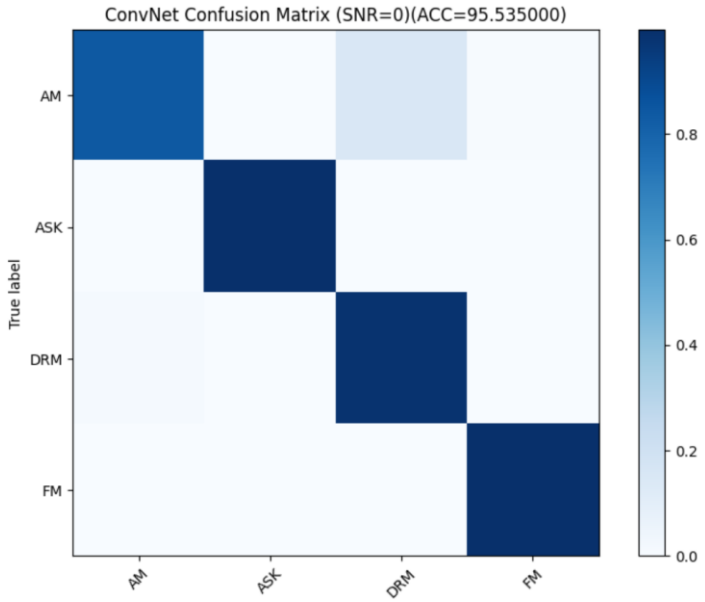

图 4.2.2.3 VT-CNN2 模型混淆矩阵图

4.2.3 模型推理功能实现

使用 argmod 数据集测试，测试条数为数据集中的随机 1000 条，结果如下：

```
input data successful!  time: 1.1818945407867432
FP16: False
INT8: False
BF16: False
FP8: False
TF32: False
build engine successful!  time: 11.680138111114502
build context successful!  time: 0.013208627700805664
binding successful!
acc: 0.7189 need time: 5.090284585952759
```

图 4.2.3.1 模型推理测试

测试结果稳定在 0.72 可能是因为此数据集偏小，采集的随机样本并不具备随机性。

4.3 系统定量测试结果与分析

4.3.1 信号接收

在信号接收过程中主要实现对信号的下变频与抽取滤波，以在 91.4M 处的 FM 信号为例，经过各个部分处理后的信号如下图所示。

1. 接收到的基带信号：

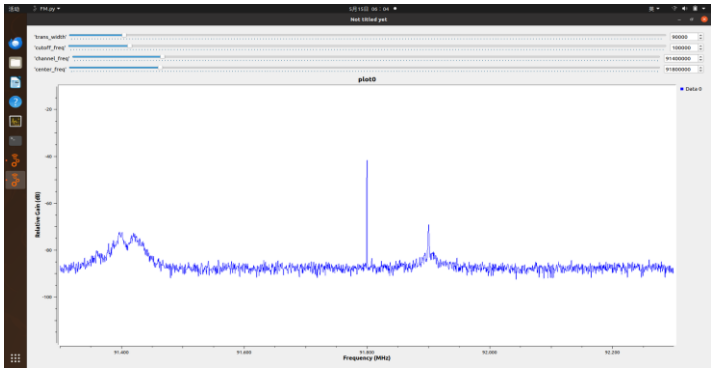

图 4.3.1.1 未经处理的基带信号

其中左侧波峰为需要记录的信号，经过混频后此时位于中心频段的左侧。

2. 下变频、滤波后的信号

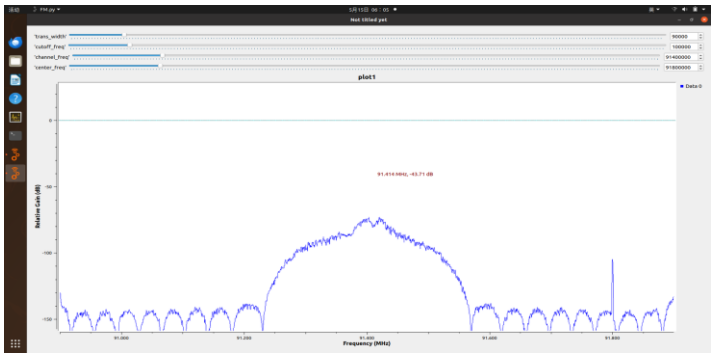

图 4.3.1.2 未经处理的基带信号

信号在下变频后的中心频率位于 0Hz 处，此时在对信号进行低通滤波，可只保留目标频带内的信号，消除其他信号的干扰同时提高信噪比。设置的通带截止频率为 100KHz（最大衰减 10dB），阻带截止频率为 150KHz(最小衰减 40dB)，结果基本吻合

3. 降采样

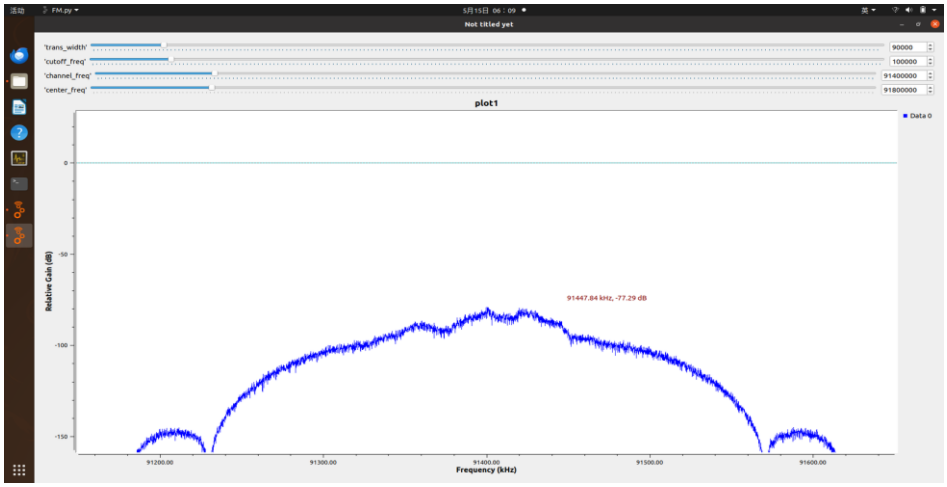

图 4.3.1.3 降采样后的信号

HackRF One 允许的最低接受带宽为 2MHZ（对应最低采样率为 1Mbps),远远大于信号所需采样率。为降低数据传输冗余，提升系统通信效率，有必要对信号进行降采样处理。

4.3.2 模型测试

经过多轮测试，我们组决定将 VT-CNN2 训练好的模型进行部署，以下是不同类别的识别情况，我们使用三阶段验证法来验证模型准确率。首先采集一个时间点的四种类别的信号数据，等待几分钟之后采集第二段四种类别的信号数据，将第一段制作成数据集然后训练，用训练好的模型预测第二段四种类别的数据的调制方式，如果效果较好则用天线接收新来的信号进行测试。

FM 测试：

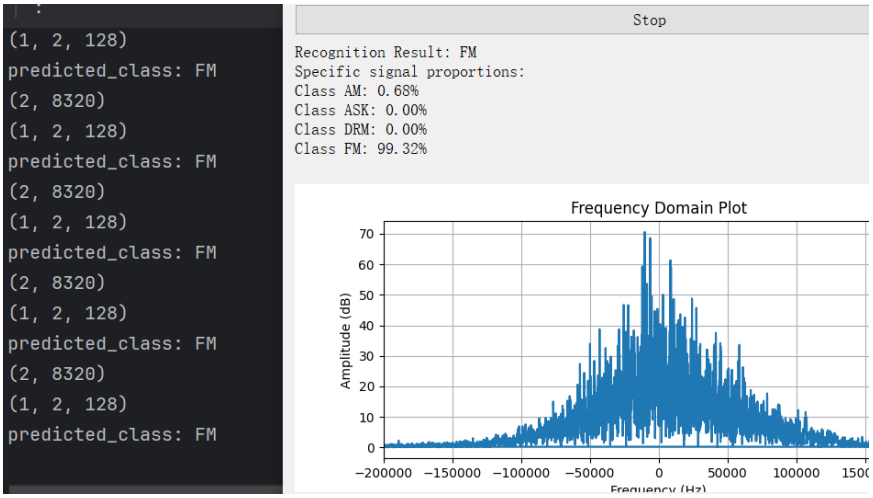

图 4.3.2.1 FM 测试结果图

AM 测试：

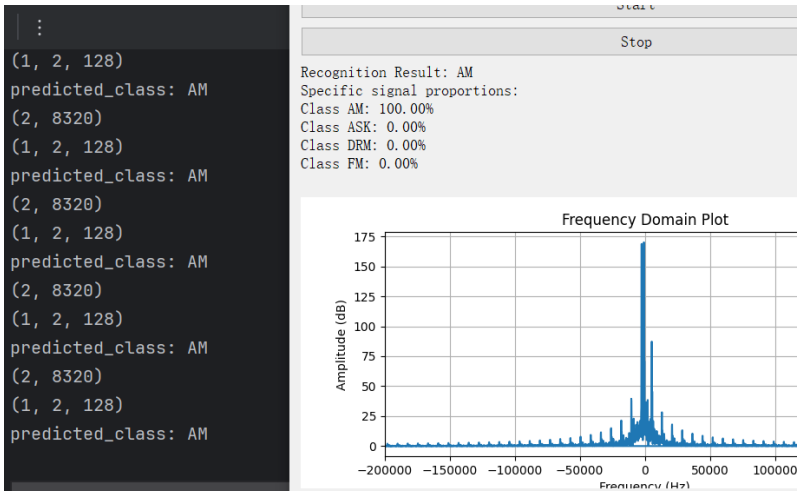

图 4.3.2.1 AM 测试结果图

DRM 测试：

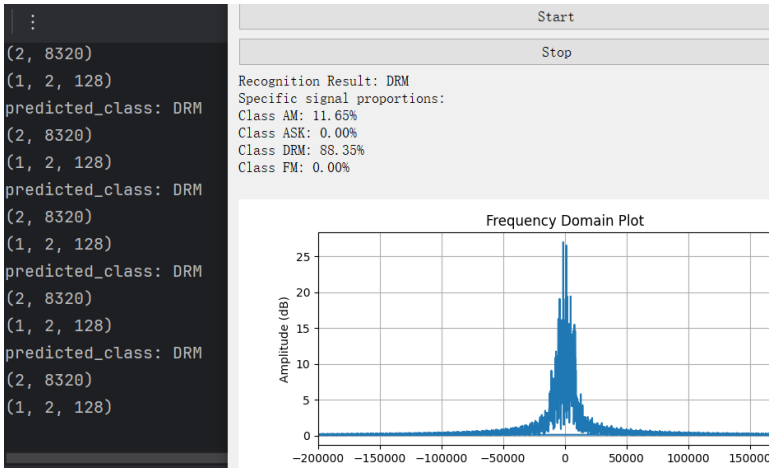

图 4.3.2.1 DRM 测试结果图

ASK 测试：

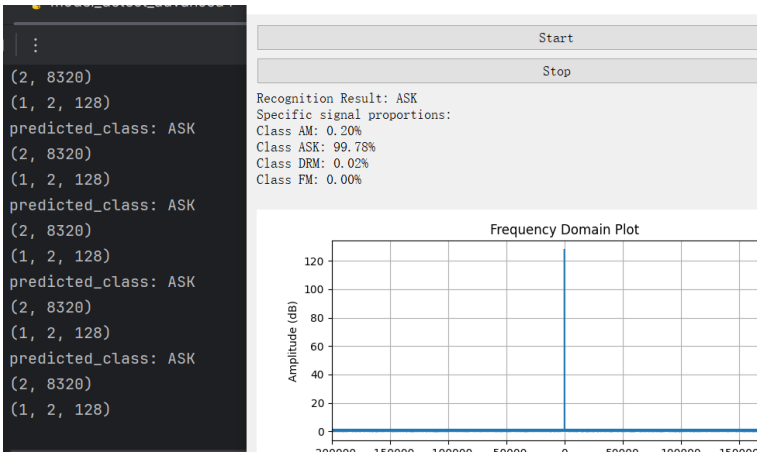

图 4.3.2.1 ASK 测试结果图

从上面的测试结果可用看出，FM、AM 和 ASK 的识别准确率较高，但是 DRM 的识别准确率欠佳，据我们分析应该是 ODFM 中的调制方式并不唯一，而且不同时间段使用的调制方式也不一样才会导致这样的结果。

4.3.3 多精度下的 TensorRT 推理对比

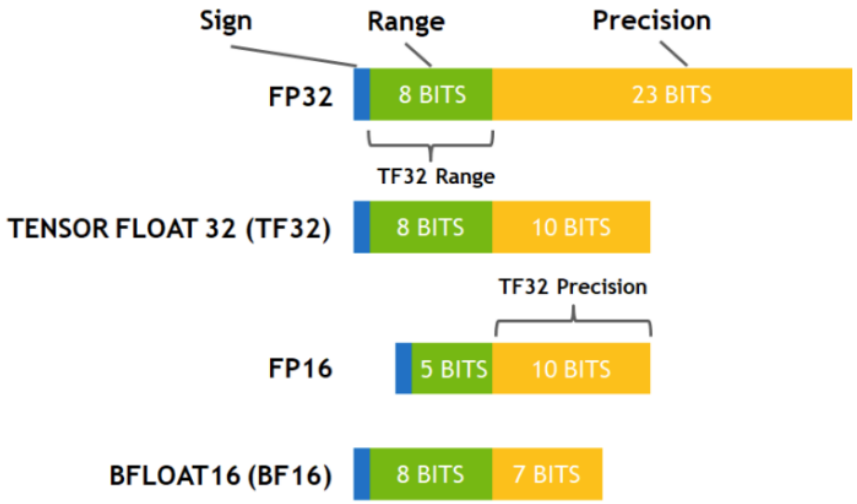

图 4.3.3.1 不同精度的动态范围和精度范围对比

使用精度	准确率	推理时间(10000 次)	构建时间
FP32	0.73	4.95	11
TF32(默	0.72	4.38	12

认)			
FP16	0.72	4.12	28
BF16	0.72	3.96	12

图 4.3.3.2 10000 次推理下推理时间和准确率对比

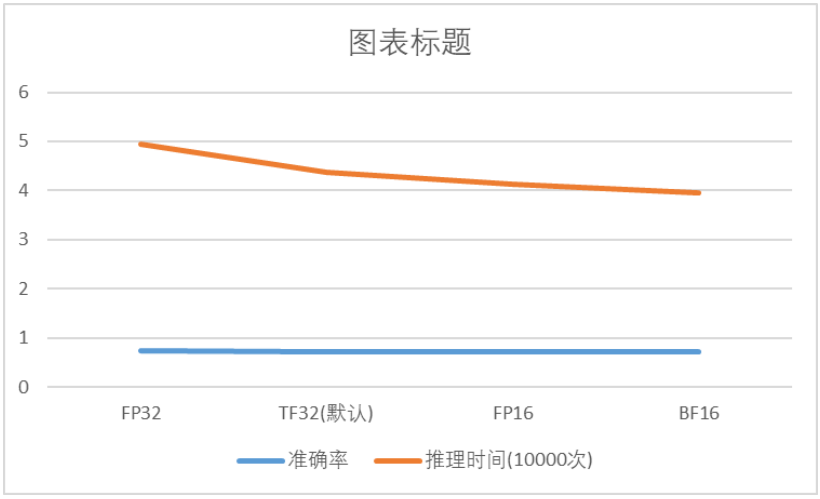

图 4.3.3.3 不同精度的效率定量对比

通过比对发现，在不损失准确率的条件下，进行变量精度优化可以对推理效率有较大提升。

参考文献

[1]Courtat, T., & du Mas des Bourboux, H.M. (2020). A light neural network for modulation detection under impairments. *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 1-7.

[2]O'Shea, T., Corgan, J., & Clancy, T.C. (2016). Convolutional Radio Modulation Recognition Networks. *International Conference on Engineering Applications of Neural Networks*.