Escola Tècnica Superior d'Enginyeria ETSE-UV

Master's Degree In Data Science

VNIVERSITAT DE VALÈNCIA

Master's Thesis

Multimodal Transformer for Intrapulse Dual-Component Signal Modulation Classification

AUTHOR:
Agustín Matías Galante Cerviño

TUTORS:
Emilio Soria Olivas
Valero Laparra Pérez Muelas

July, 2023

Abstract In this work, we take on the problem of classifying modulation types of single- and dual-component radar signals. We begin by describing the relevant features of input data, to then proceed to the generation of plentiful and varied synthetic data. After this, said data is used in the training and evaluation of several models found in literature. We used models made to classify radar signals as well as more recent models which have not yet been used for this problem. They are reimplemented in order to compare them with our proposal of a model based on vision transformers and multimodality, observing multiple signal domains at the same time, in order to improve performance with respect to other models found in previous work.

Resumen En este trabajo, se aborda el problema de clasificación de tipos de modulación de señales radar con una y dos componentes. Se comienza describiendo las características relevantes de los datos de entrada, para luego proceder a generar datos sintéticos variados y de calidad. Después, se usan en el entrenamiento y evaluación de varios modelos encontrados en la literatura, tanto modelos hechos para clasificar señales de radar como modelos más recientes, no usados aún en este problema. Se reimplementan con el fin de compararlos con nuestra propuesta de un modelo basado en transformadores de visión y multimodalidad, observando varios dominios de la señal a la vez, con el objetivo de mejorar el rendimiento con respecto a otros modelos encontrados en trabajos anteriores.

Resum En aquest treball, es tracta el problema de classificació de tipus de modulació de senyal radar amb una o dos components. Es comença descrivint les característiques rellevants de les dades d'entrada, per a procedir a generar dades sintètiques variades i de qualitat. Després, aquestes dades se usen en l'entrenament i evaluació de diversos models trobats en la literatura, tant models fets per a classificar senyals de radar com models més recents, encara no usats en aquest problema. Es reimplementen amb la fi de comparar-los amb la nostra proposta de un model basat en transformadors de visió y multimodalitat, observant diversos dominis de la senyal alhora, amb l'objectiu de millorar el rendiment respecte a altres models trobats en treballs anteriors.

Contents

Chapter 1

Introduction and objectives

One of the most evident facts about warfare is that the advancement of technology is a key factor in establishing definite superiority in a strategic, operational and tactical context. A major exponent of this observation is the widespread usage and adoption of the radar (an acronym for **RA**dio **D**etection **A**nd **R**anging). It is a system that allows for the determination of direction, range and velocity of objects, making use of the emission of electromagnetic radiation in the radio or microwave spectrum, and its subsequent reflection. Demostrating its effectiveness in World War II, it went on to become one of the cornerstones of modern military technology. Showing great promise and potential, it has since been adopted in a variety of applications, both military and civilian; air and terrestrial traffic control, radar astronomy, air and missile defense systems, self-driving cars, ground-penetrating radar...

Given its proliferation and usefulness, nations soon began seeking ways to counter such a threat, and an essential part of this is gathering intelligence. Just as a radar system can detect its own reflected emissions, other unrelated third-party recievers are able to intercept these signals and ascertain various revealing characteristics of the emitter. This is known as electronic intelligence (ELINT), which consists in the gathering of actionable information from such electronic signals, and is itself a subset of signals intelligence (SIGINT). Such analysis of radar detection data is however complicated by noisy environments, a shared spectrum with communications and navigation systems, and the ever-increasing usage of sophisticated spoofing methods by emitters, the latter making this a veritable arms race. These considerations merit research into more effective analysis methods.

Indeed, the accuracy of radar signal processing has been demonstrated to be considerably improved with respect to other approaches by making use of deep learning models based mostly on convolutional neural networks (CNNs) [ZDG17; QMD18; Qu+19b; Qu+19a], recurrent neural networks [Nor19; Li+20; LLH20], and, more recently, transformers [YLW22]. Transformers are a relatively new neural network architecture based on self-attention [Vas+17] which has made massive strides lately in the field of natural language processing [Bro+20; Hof+22], becoming the new state-of-the-art. More recently, a slightly modified version of this architecture, the Vision Transformer (ViT) has proven its worth in computer vision, demonstrating that it is able to outperform some CNN architectures given a large training dataset [Dos+20].

In this work, we wish to expand on the recent advancements in this problem, offering a method of classifying the intrapulse modulation type of single- and dual-component pulsed radar signals making use of a model based on a multimodal transformer approach. This is motivated thanks to various concepts shown in literature which we intend to combine into a model that may obtain improvements. One of which is in [Qu+19a], where the signal was apparently input into three squeeze-and-excitation models, each in a different domain; time, frequency, and time-frequency, and their result was combined in a majority vote. The usage of different signal domains may provide improvements stemming from the ability to observe the signal with varying resolutions in both time and frequency. However, we are looking to implement a more efficient approach than using three separate models, so instead, we will combine internal representations of the input data such that we only use one classifier. There is also [YLW22], where a model using convolutions and transformer encoders is introduced, and signals with multiple components are classified.

While radar signals with multiple components have seldom been classified in literature [XLH22], this consideration is gaining importance in these classification schemes because of their usage in newer radar systems [YLW22; Hou+23]. We also wish to explore a different avenue with respect to their point of view on the architecture of the model they showcased, through the further exploitation of the flexibility of transformers by not using convolutional layers. ViTs can learn specific relations to the training data without being held to inductive biases present in CNNs like translation equivariance and locality, showing better results, but their effectiveness becomes clear only when given a large enough number of samples to train on [Dos+20], which is something enabled by our use of synthetic data generation.

Chapter 2

Radar signal characteristics

Let us start by describing the nature of the data from which we wish to extract actionable information. It is usually made up, in its least preprocessed form in the digital domain, of a single real- or complex-valued time series registering voltage induced by electromagnetic radiation on an antenna. Within this series, the signals we wish to characterize are pulsed waves from friendly or enemy emitters, the parameters of which can be modulated in a variety of ways. Each emitter possesses one or several modulation schemes of these parameters, which allows an unrelated, third-party reciever to discern its make and model (or at least its country of origin) and its mode, thus learning its affiliation and intentions.

2.1 Overview

Every parameter of a pulsed radar signal carries important information about what we wish to know. Different modulation types of such parameters are applied by the emitter, depending on various conditions.

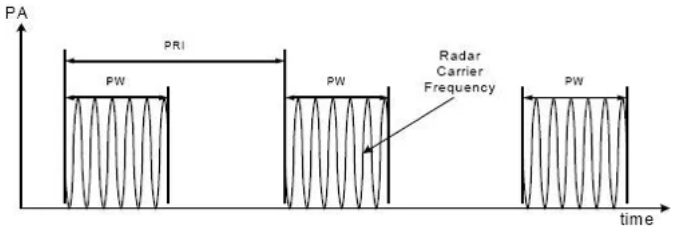


Figure 2.1: Pulsed radar parameters.

Some of the most important features extracted from these signals are the pulse amplitude (PA), pulse frequency (PF), pulse repetition interval (PRI) and the angle of arrival (AOA). The PA is the maximum value of the signal of a single pulse, PF is the value of the frequency of the carrier signal, the PRI is the time passed between the emission of two consecutive pulses, and the AOA is, as its name suggests, the direction from which the emission was recieved[1]. Some examples of useful information which can be gleaned from these parameters is that, through analysis of the PRI, one can reveal the mode the radar is in (searching, target acquisition, or target tracking), or by analyzing PA modulation, it is possible to ascertain the sweep pattern that the radar beam follows, also called its *scan*. In this work, however, we will only classify the intrapulse modulation type of each pulse, which we will detail below in section 2.2.

[1]It is an important feature, but it is not expressed in the time series, but rather a separate feature extracted through other means, like interferometry.

2.2 Intrapulse modulation

Within a pulse, the frequency, amplitude and phase of the signal could each change with time. When done by radars, this is part of a signal processing technique called pulse compression, which is used in order to improve range and velocity resolution. Frequency of a typical emission ranges between 1-10 GHz, and is usually considered to be a real signal. However, these signals are often represented with complex values, since this facilitates many mathematical manipulations, as well as the fact that common sampling schemes such as I/Q sampling[2] directly output complex signals. If sampling occurs on a single channel, then the signal as processed by the emitter is also real. In this case, then turning them into **analytical signals** is sometimes done, which turns a real signal into a complex one that has no negative frequency components. Such a signal holds the original real signal in the real component, and the Hilbert transform[3] of the signal in the imaginary component. This makes up the **analytic representation** of the real signal. This signal is thus usually modeled in the following manner;

$$s(t) = A(t)\mathrm{e}^{\mathrm{i}\varphi(t)}, \tag{2.1}$$

with $A(t)$ being the amplitude modulation, which is not very commonly used in radar systems, and $\varphi(t)$ is the instantaneous phase, which is where frequency and/or phase modulation is contained. Modulation types used here will only affect $\varphi(t)$, so we will focus on frequency and phase modulation. They are usually described using the following signal parameters;

- PW: Pulse width, length in time of the pulse.
- f_s: Sampling frequency.
- f_{min}: Minimum instantaneous frequency.
- Δf: Signal bandwidth. Frequency range of the signal contents.
- f_0: Frequency offset. It lies at the center of the bandwidth. Equal to $f_0 = f_{min} + \Delta f/2$
- N: Order of a given code. It is usually the length in symbols of codes like Costas and Barker, but it is the square root of the length of codes like Frank.

The instantaneous frequency, $\varphi'(t) = \frac{\mathrm{d}}{\mathrm{d}t}\varphi(t)$, of the modulation types used in this work are listed below. The factor of 2π multiplying the left hand side of these expressions is understood. We take $t = 0$ to be the beginning of the pulse, and $t = PW$ to be the end. We first have the simplest case,

- **No modulation (NM)**
  No modulation; frequency does not change.
  $$\varphi'(t) = f_0. \tag{2.2}$$

As well as continuous frequency modulation,

- **Linear frequency modulation (LFM)**
  Frequency varies linearly with time.
  $$\varphi'(t) = \frac{\Delta f}{PW}t + f_{min}. \tag{2.3}$$

- **Dual linear frequency modulation (DLFM)**
  The pulse has a linear, continuous variation in its frequency, but it increases and decreases (or viceversa) such that it appears as a Λ or a V in a spectrogram.
  $$\varphi'(t) = \begin{cases} \pm\frac{2\Delta f}{PW}t + f_{min} + \frac{\Delta f}{2} \mp \frac{\Delta f}{2} & \text{if } t < PW/2 \\ \mp\frac{2\Delta f}{PW}(t - PW) + f_{min} + \frac{\Delta f}{2} \mp \frac{\Delta f}{2} & \text{if } t > PW/2 \end{cases}. \tag{2.4}$$

[2] That is, two analogue-to-digital converters sampling the signal at the same time, but one of them recieves the signal with a phase offset of $\pi/2$ radians.

[3] This effectively shifts the real signal's phase by $\pi/2$ radians as well.

- **Multiple linear frequency modulation (MLFM)**
  The pulse has a linear, continuous variation in its frequency, but it is more like two consecutive LFM pulses. r is the fraction of the pulse width corresponding to each "subpulse".

$$\varphi'(t) = \begin{cases} \pm\frac{\Delta f}{rPW}t + f_{min} + \frac{\Delta f}{2} \mp \frac{\Delta f}{2} & \text{if } t < rPW \\ \pm\frac{\Delta f}{(1-r)PW}(t - rPW) + f_{min} + \frac{\Delta f}{2} \mp \frac{\Delta f}{2} & \text{if } t > rPW \end{cases}. \tag{2.5}$$

- **Even quadratic frequency modulation (EQFM)**
  The pulse has a quadratic, continuous variation in its frequency. The frequency minimum is located on the center of the pulse, so its variation is symmetric about said center.

$$\varphi'(t) = \pm\frac{\Delta f}{PW}\left(t - \frac{PW}{2}\right)^2 + f_{min}. \tag{2.6}$$

- **Sinusoidal frequency modulation (SFM)**
  Frequency of the pulse varies as a sinusoid.

$$\varphi'(t) = \Delta f \sin(2\pi f_{\text{sfm}} t + \phi_{\text{sfm}}) + f_{min}, \tag{2.7}$$

  where f_{sfm} is the frequency with which $\varphi'(t)$ varies, and ϕ_{sfm} is the initial phase.

Discrete frequency modulation,

- **Binary frequency shift keying (BFSK)**
  Pulse frequency shifts between two discrete values. In radar applications, the code used is usually Barker as well.

$$\varphi'(t) = \begin{cases} f_0 + \frac{\Delta f}{2} & \text{if } t \in \text{Symbol 1} \\ f_0 - \frac{\Delta f}{2} & \text{if } t \in \text{Symbol 2} \end{cases}. \tag{2.8}$$

- **Quadruple frequency shift keying (QFSK)**
  Pulse frequency shifts between four discrete values.

$$\varphi'(t) = \begin{cases} f_1 & \text{if } t \in \text{Symbol 1} \\ f_2 & \text{if } t \in \text{Symbol 2} \\ f_3 & \text{if } t \in \text{Symbol 3} \\ f_4 & \text{if } t \in \text{Symbol 4} \end{cases}. \tag{2.9}$$

and finally, discrete and continuous phase modulation,

- **Binary phase shift keying (BPSK)**
  Pulse phase shifts between two discrete values, most often 0 and π. In radar applications, the code used in a BPSK modulated pulse is a Barker code of a certain order.

- **Frank**
  The Frank code has N^2 elements and is usually defined as a matrix,

$$\varphi_{i,j} = \frac{2\pi}{N}(i-1)(j-1), \tag{2.10}$$

  where i and j range from 1 to N. A single pulse of width PW is divided into N segments of equal duration in time. Each of these segments are further divided into N subpulses, each of these having a phase given by equation (2.10). The phase of each subpulse, in order, is this matrix when flattened, by putting its rows side by side according to their order. The fundamental phase increment is $\Delta\varphi = 2\pi/N$, which multiplies this matrix.

- **P1**
  P1 is a discrete phase modulation type, designed similarly to a stepped or discrete linear frequency modulation scheme.

$$\varphi_{i,j} = -\frac{\pi}{N}[N - (2i - 1)][(i - 1)N + (j - 1)]. \tag{2.11}$$

- **P2**
  P2 is a discrete phase modulation type, designed similarly to a stepped or discrete linear frequency modulation scheme, containing an additional phase shift between frequency increments.

$$\varphi_{i,j} = \left[\frac{\pi}{2}\left(\frac{N-1}{N}\right) - \frac{\pi}{N}(j-1)\right][N + 1 - 2i]. \tag{2.12}$$

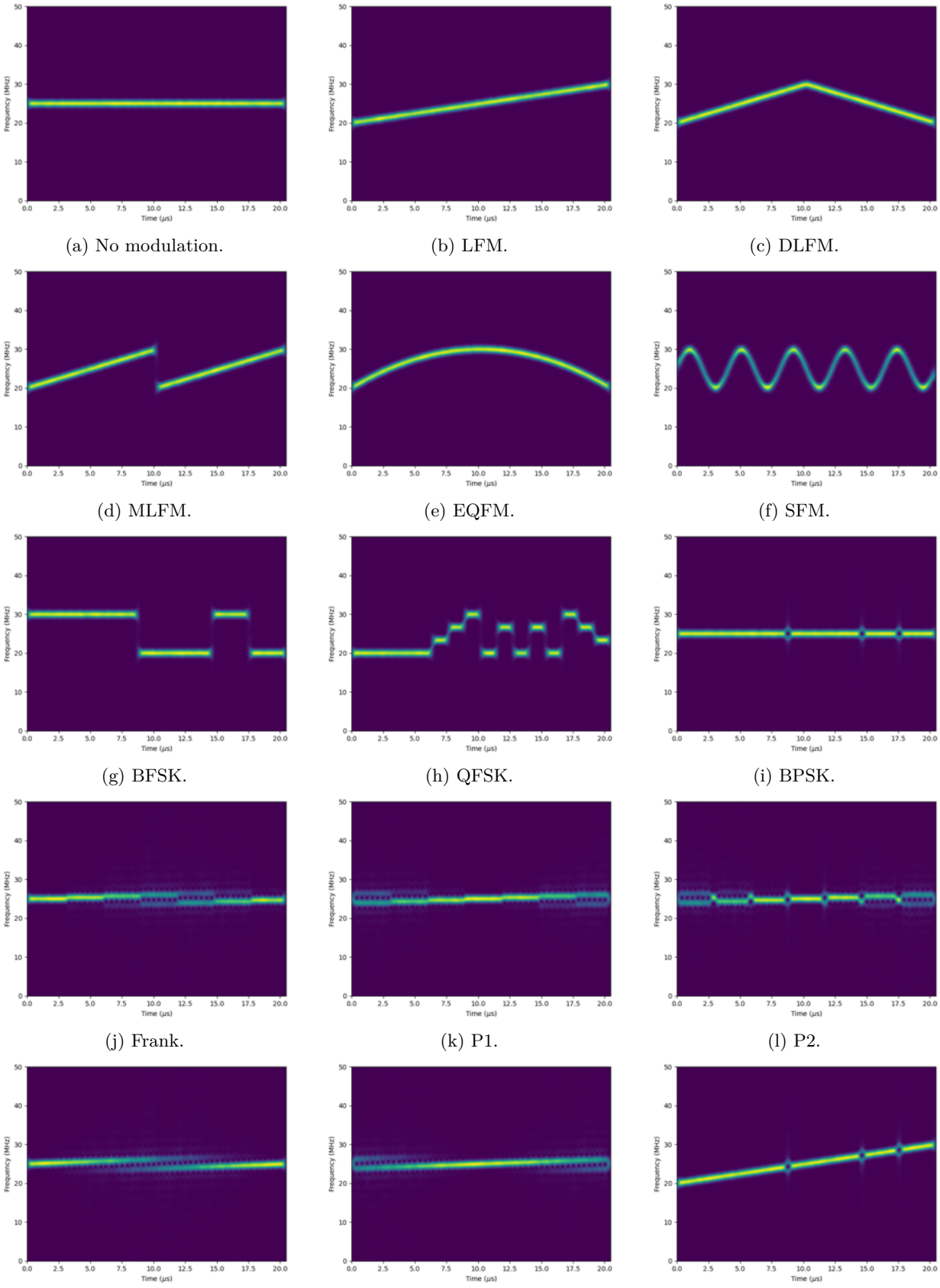
- **P3**
  P3 is a continuous phase modulation type, designed similarly to a dual linear frequency modulation scheme.

$$\varphi(t) = \frac{\pi(t-1)^2}{N}. \tag{2.13}$$

- **P4**
  P4 is a continuous phase modulation type, designed similarly to a linear frequency modulation scheme.

$$\varphi(t) = \frac{\pi(t-1)^2}{N} - \pi(t-1). \tag{2.14}$$

- **LFM-BPSK**
  Both frequency and phase are modulated at the same time. Frequency is modulated linearly and continuously, while phase is modulated in a discrete manner following a code, usually also a Barker one.



(a) No modulation. (b) LFM. (c) DLFM.

(d) MLFM. (e) EQFM. (f) SFM.

(g) BFSK. (h) QFSK. (i) BPSK.

(j) Frank. (k) P1. (l) P2.

(m) P3. (n) P4. (o) LFM-BPSK.

Figure 2.2: Time-frequency images obtained through the short-time Fourier transform of modulation types used in this work.

2.3 Noise

The recieved signal may be impeded in various ways, as an effect of both imperfect reception and unfavorable transmission conditions. We will include the two most often represented in literature;

- **Thermal noise of electronic components**
  Any electronic device kept at a temperature substantially higher than 0 K experiences a random movement of its electrons in conductors and semiconductors, which in turn induces a random voltage, able to be approximately modeled as additive white[4] Gaussian noise (AWGN) with zero mean. Such noise is usually quantified through the signal-to-noise ratio (SNR) which is expressed logarithmically as

$$SNR = 10 \log_{10} \frac{\sigma_s^2}{\sigma_n^2}, \tag{2.15}$$

  where σ_s^2 is the signal power and σ_n^2 is the noise power.

- **Frequency offset or detuning**
  A signal is converted to the digital domain after having shifted its frequency spectrum towards lower frequencies (downconversion), by multiplying it with a complex exponential having a certain frequency. It does not usually match the emitter's carrier frequency, so the recieved signal's bandwidth could be centered at any frequency. This is modeled by generating samples with a random frequency offset following a uniform distribution.

Once these impediments are applied onto the signal, the time series representing signal interception looks more like

$$x(t) = s(t) + n(t) = A(t)\mathrm{e}^{\mathrm{i}(2\pi f_0 t + \varphi(t))} + \mathcal{N}(0, \sigma_n), \tag{2.16}$$

where $n(t) = \mathcal{N}(0, \sigma_n)$ is the AWGN as described above, and f_0 is the frequency offset. This Gaussian noise is complex, meaning that both the real and imaginary components of the noise follow each an independent Gaussian distribution, with the same σ_n. If the signal is sampled on a single channel, it is simply the real part of 2.16, this being our case.

2.4 Multiple components

In somewhat broad terms, it is said a signal is **multicomponent** when it can be expressed as the sum of various contributions,

$$\begin{aligned} s(t) &= s_1(t) + s_2(t) + \cdots \\ &= A_1(t)\mathrm{e}^{\mathrm{i}\varphi_1(t)} + A_2(t)\mathrm{e}^{\mathrm{i}\varphi_2(t)} + \cdots, \end{aligned} \tag{2.17}$$

however, we will not use just this criterion, as there are an infinite number of ways to express a signal as a sum of parts, one of them famously being Fourier series; any signal can be decomposed into a Fourier series, so does that mean any signal has many or even infinite components?

Thus, we will use the definition described in [Coh92]. Multicomponent signals as seen there are not only made up by a sum of components, but each of these occupy different regions of the time-frequency plane, with every contribution being separate either in time, frequency, or both, while having an instantaneous bandwidth that is less than the distance in frequency between both signals. In more formal terms, and using just two signals for the sake of clarity, it means that

$$\left|\frac{A_1'(t)}{A_1(t)}\right|, \left|\frac{A_2'(t)}{A_2(t)}\right| < |\varphi_2'(t) - \varphi_1'(t)|, \tag{2.18}$$

where $\varphi'(t)$ is the first derivative of the instantaneous phase with respect to time, also called the instantaneous frequency, and $|A'(t)/A(t)|$ is the instantaneous bandwidth. It is necessary to mention that the amplitude of each component in this work will not change with time, so their derivative will be zero. However, due to the uncertainty principle, even if we define a component's frequency perfectly according to a formula, once we apply a time-frequency distribution to the signal, we will still obtain a non-zero bandwidth.

[4] Same power at all frequencies.

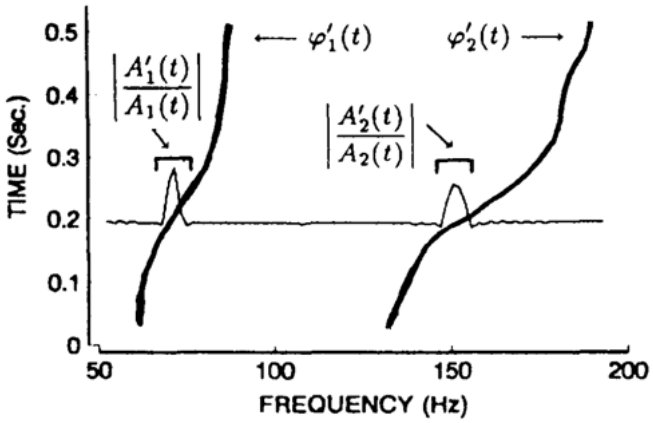


Figure 2.3: Multicomponent signal characteristics. [Coh92]

Since half of each signal's instantaneous bandwidth is included in the interval defined by the right hand side of the inequality, a further requirement is that

$$\frac{1}{2}\left(\left|\frac{A_1'(t)}{A_1(t)}\right| + \left|\frac{A_2'(t)}{A_2(t)}\right|\right) < |\varphi_2'(t) - \varphi_1'(t)|. \tag{2.19}$$

Some of the modulations we use do not have a continuous nor differentiable path through the time-frequency plane; in this case, we must expand the definition to include such modulations. Signals could also have their number of components change over time; however, in our particular case, that will not occur.

Chapter 3

Generating synthetic radar detection data

Given that radar detection data is not easy to come by, as more often than not it is considered a matter of national security, it becomes a necessity to pursue different avenues to acquire a dataset large enough to train models, especially deep learning ones. Even if access to this data would be easier, gathering enough data and labeling it is an arduous and time-consuming task, as is the case with many other datasets used in deep learning. Fortunately, unlike domains like medical imaging, radar detection data has relatively few straightforward characteristics, which can be fathomably encapsulated in code, allowing us to generate a large variety and quantity of realistic synthetic data. This allows us to consider such deep learning models as a whole without as much effort, as well as easing the usage of architectures such as transformers, which usually outperform other types of models given a large amount of samples.

In order to generate synthetic data, a library called `pyradargen` will be used. It contains five functions related to simulation. Four of these generate various characteristics of pulsed radar detection. `toa_gen()` generates times of arrival (TOA) for every pulse. Three other functions, `pw_gen()`, `pf_gen()` and `pa_gen()` then take these TOAs and generate pulse widths for each pulse, pulse frequencies, and pulse amplitudes, respectively.

However, out of these five, `intra_gen()` will be the only one used, which generates the signal of a single pulse according to various parameters, among them the modulation type and SNR, as well as part of the parameters listed previously. More specifically, it generates intrapulse signals used in radar and communication systems, given in the I/Q format (In phase/Quadrature). This format is usually given once the frequency origin is shifted to the center (or close to it) of the frequency contents of the signal; this is the reason why the sampling frequency can be this low (`intra_gen()`'s default is 50 MHz vs >1 GHz for RF signals). We shall take only the I component of the output I/Q signal, therefore using a real signal.

Like this, we will generate a large amount of samples, in the order of hundreds of thousands to millions, each being a signal of a fixed length. The dataset these samples make up will contain, as a whole, a wide range of parameter combinations and amounts of noise. In the following, we shall detail the value of these parameters, our criteria for choosing them, as well as briefly exploring some details of this concrete dataset which would be wise to account for, before we inspect the performance of the models we evaluate in this work.

3.1 Parameter values to be used

The range of the relevant parameters used to completely determine each of these modulation types' characteristics is detailed below in table 3.1, as well as the source for these, having used the definition of the signal parameters we described in section 2.2.

Table 3.1: Parameters applied to samples of each modulation type used in this work.

Modulation type	Parameters	Notes
NM	$f_0 \sim U(0.1f_s, 0.4f_s)$	Sourced from [QMD18]
LFM	$f_0 \sim U(0.01f_s, 0.45f_s)$ $\Delta f \sim U(0.05f_s, 0.4f_s)$	Frequency would either increase or decrease with time, chosen with equal probability. As in, either of these options are picked with a 50% chance, just as done with these other variables, but with discrete values. This will be the case for the rest of the mentioned parameters that take discrete values. Sourced from [QMD18]
DLFM	$f_0 \sim U(0.01f_s, 0.4f_s)$ $\Delta f \sim U(0.05f_s, 0.35f_s)$	Sourced from [QMD18]
MLFM	$f_0 \sim U(0.15f_s, 0.5f_s)$ $\Delta f \sim U(0.1f_s, 0.35f_s)$ $r \sim U(0.3, 0.7)$	Sourced from [QMD18]
EQFM	$f_{min} \sim U(0.01f_s, 0.4f_s)$ $\Delta f \sim U(0.05f_s, 0.3f_s)$ $f_0 = f_{min} + \Delta f/2$	Sourced from [QMD18]
SFM	$f_{min} \sim U(0.01f_s, 0.15f_s)$ $\Delta f \sim U(0.05f_s, 0.35f_s)$ $f_0 = f_{min} + \Delta f/2$ $f_{\text{sfm}} \sim U(\frac{0.75}{PW}, \frac{10}{PW})$ $\phi_{\text{sfm}} \sim U(0, 2\pi)$	Additionally, since this is a sinusoid, selecting frequency and initial phase randomly is needed, something which was apparently not specified in literature. It was chosen to have between 0.75 to 10 periods within the pulse. Sourced most parameters from [QMD18]
BFSK	$f_1,\ f_2 \sim U(0.05f_s, 0.45f_s)$ $f_0 = (f_1 + f_2)/2$ $\Delta f = \lvert f_1 - f_2\rvert/2$ $N \in \{5, 7, 11, 13\}$	Each frequency is sampled independently, but resampling is done sometimes in order to keep absolute difference $\lvert f_1 - f_2\rvert > 0.005f_s$. These are the frequencies of each of the code's symbols. Code is either normal or inverted Barker. Sourced from [YLW22]
QFSK	$f_1,\ f_2,\ f_3,\ f_4 \sim U(0.05f_s, 0.45f_s)$	Absolute difference between any of these frequencies is kept $\lvert f_i - f_j\rvert > 0.005\ f_s\ \forall i, j \in \{1, 2, 3, 4\}$ if $i \neq j$. Code is a 16-bit ($N = 4$) Frank code, and is either normal or inverted. Sourced from [YLW22]
BPSK	$f_0 \sim U(0.05f_s, 0.45f_s)$ $N \in \{5, 7, 11, 13\}$	We used a mixture of parameter choices between [QMD18] and [YLW22]. All that was used from the latter was the fact that these pulses only contain one instance of the code. Code is either a normal or inverted Barker code.
Frank	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	Everything is from [YLW22], except the lower bound of the frequency offset, since if it is too low, the symmetry of the transform will mean that it becomes unrecognizable[1].
P1	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	The order of these codes (P1, P2, P3, P4) is from [Aky+18], but the range of the frequency offset was extended. The code is also either normal or inverted, which resembles an ascending or a descending chirp respectively.
P2	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	Sourced from [Aky+18]
P3	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	Sourced from [Aky+18]
P4	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	Sourced from [Aky+18]
LFM-BPSK	$f_{min} \sim U(0.05f_s, 0.45f_s)$ $\Delta f \sim U(0.05f_s, 0.4f_s)$ $f_0 = f_{min} + \Delta f/2$ $N \in \{5, 7, 11, 13\}$	f_0, in this paper, is said to be between 0.01 and 0.45, but I take it the FM here should be the same as in LFM. Sourced from [QMD18]

$f_1,\ f_2,\ f_3,\ f_4$ refer to the frequencies that correspond to each symbol.

3.2 Dataset

Each of the modulation types as described beforehand will make up one class, so the dataset will contain 15 classes, combining modulation types from different articles in literature, which are believed to be relevant. Each sample will be made up of a real signal, always having 2048 data points, with the signal occupying every point. In turn, each of these points is represented by one 32-bit floating point number. Such a number of points corresponds, for example, to a sampling frequency of 100 MHz and a pulse width of 20.48 μs. This frequency varies widely in literature, between 100 MHz [Aky+18] and 1 GHz, and typical pulse widths are $\mathcal{O}(1 - 100\,\mu s)$ [YLW22]. The pulse amplitude, $A(t)$, will remain constant in all modulations and equal to $A(t) = A_0 = 1$. Additionally, every discrete modulation (FSK and PSK) will contain only one instance of their code, it will not be repeated within a pulse more than once.

In the training set, 800 combinations of parameters are randomly chosen, each sampled from a uniform distribution as mentioned in table 3.1. Samples are impeded using additive white Gaussian noise, with SNR taking values between -12 dB and 18 dB, with increments of 3 dB, so there are 11 noise levels. Each of these levels is applied to each of the parameter combinations, so that the same combination is shown for a wide variety of noisy conditions. The training dataset thus amounts to $800 \cdot 11 \cdot 15 = 132\,000$ single-component samples. A further 200 parameter combinations are sampled for validation and 400 combinations for testing, so $200 \cdot 11 \cdot 15 = 33\,000$ samples in the validation set. As for the testing set, we used more noise levels, including lower SNR values, spanning between -21 dB and 18 dB, making up 14 levels. That means we have $400 \cdot 14 \cdot 15 = 84\,000$ samples in the testing set. Each of these single-component datasets take up 1081 MB, 270 MB and 688 MB, respectively.

In order to obtain dual-component signals, some of these existing single-component signals are combined. The amplitude of each component is not the same; we randomly sample the ratio between parameter combinations (but not among the same combination and different noise levels) in order to keep it between 1:1 and 1:3, similarly to [YLW22]. That is, each recieved signal has the form

$$x(t) = s_1(t) + s_2(t) + \mathcal{N}(0, \sigma_n) = \alpha e^{i\varphi_1(t)} + (1 - \alpha)e^{i\varphi_2(t)} + \mathcal{N}(0, \sigma_n), \tag{3.1}$$

where α is sampled from a uniform distribution between 0.5 and 0.25 in every parameter combination. The clean signal (that is, $s_1(t) + s_2(t)$) is calculated by adding two previously generated single-component clean signals. Then, the noise power, σ_n^2, is calculated by first specifying the desired SNR, and after that, the clean signal power, σ_s^2, is calculated as the power of the combination of these clean single-component signals, so we can solve for σ_n^2 in equation (2.15) and finally obtaining the right distribution in equation (3.1).

The number of combinations is equal to the number of non-zero elements in a triangular matrix, $n(n-1)/2 + n$, so for $n = 15$, it's 120. Thus, if we were to combine every sample belonging to one modulation type with every other sample of a different modulation type in the training set, we would get $800 \cdot 800 \cdot 11 \cdot 120 = 8.448 \cdot 10^8$ dual-component samples, which is impractical to handle[2], so we only mix 25 parameter combinations. In this case, there are $25 \cdot 25 \cdot 11 \cdot 120 = 825\,000$ dual-component samples in the training dataset. Similarly, there are $10 \cdot 10 \cdot 11 \cdot 120 = 132\,000$ such samples in the validation set and $15 \cdot 15 \cdot 14 \cdot 120 = 378\,000$ in the testing set. Each of these dual-component datasets take up 6.758 GB, 1.081 GB and 3.096 GB.

All in all, training, validation and testing sets have 132 000 + 825 000 = 957 000, 33 000 + 132 000 = 165 000, and 84 000 + 378 000 = 462 000 samples, and take up 7.839 GB, 1.351 GB and 3.002 GB respectively, for a total of 9.489 GB. Each of these sets are stored in an HDF5 file.

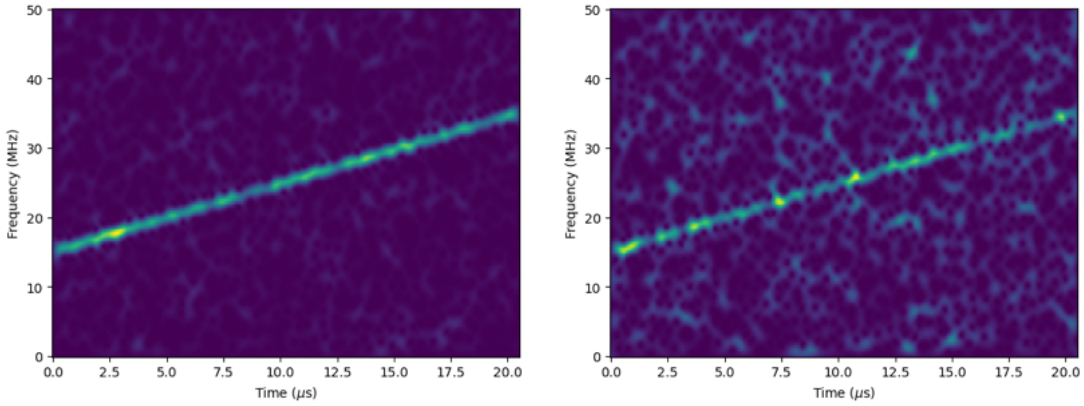
[2]This not only far exceeds the input dimensions and number of parameters in the models, but also takes up 6.92 TB.

3.3 Exploratory analysis

If we examine, in figure 2.2, the path followed by the instantaneous frequency along the time-frequency plane in every modulation type, we quickly conclude that a model using these images as inputs would not have to be as complex as current and previous well-known state-of-the-art models trained on larger and more complex datasets, such as ImageNet [Den+09]. This is because relevant features in our case, when compared to those needed to distinguish between classes in the formerly mentioned dataset, are far fewer and have a very simple morphology, making up just one or more straight lines, parabolas, and circles, which are combined in far fewer ways than, say, a dog; a dog might be oriented in many different ways, having different colors, different backgrounds... and we clearly see that our dataset does not approach the complexity of this typical example of an ImageNet class. Furthermore, our images only have one channel, which is the value of the absolute magnitude squared of the STFT, as opposed to values of red, green and blue. It becomes clear then that the main difficulty of this problem actually resides elsewhere; in the proper recognition of these modulation types in unfavorable conditions, such as noisy reception and poor transmission.

Some of the modulation types we are classifying signals into have shapes fairly similar to continuous frequency modulations, namely polyphase modulations P1, P2, P3, and P4 are very similar to LFM, MLFM, and LFM-BPSK. These modulations, along with Frank, contain small circle-like features stemming from their nature as phase modulations. These features are lesser and more visible in BPSK, which remain after reducing input size, but the fact that many polyphase modulations look a lot like frequency modulations (which is by design, this is not an accident), will hamper the performance of the model when attempting to distinguish between these. This problem may be exacerbated if we used an input size of 64x64 like in [QMD18] and [Qu+19b], and even more so when introducing noise at the same time.

The effect noise has on signal recognition is significant when SNR reaches around and below -10 dB. If we take a look at a single-component signal under different noise levels,



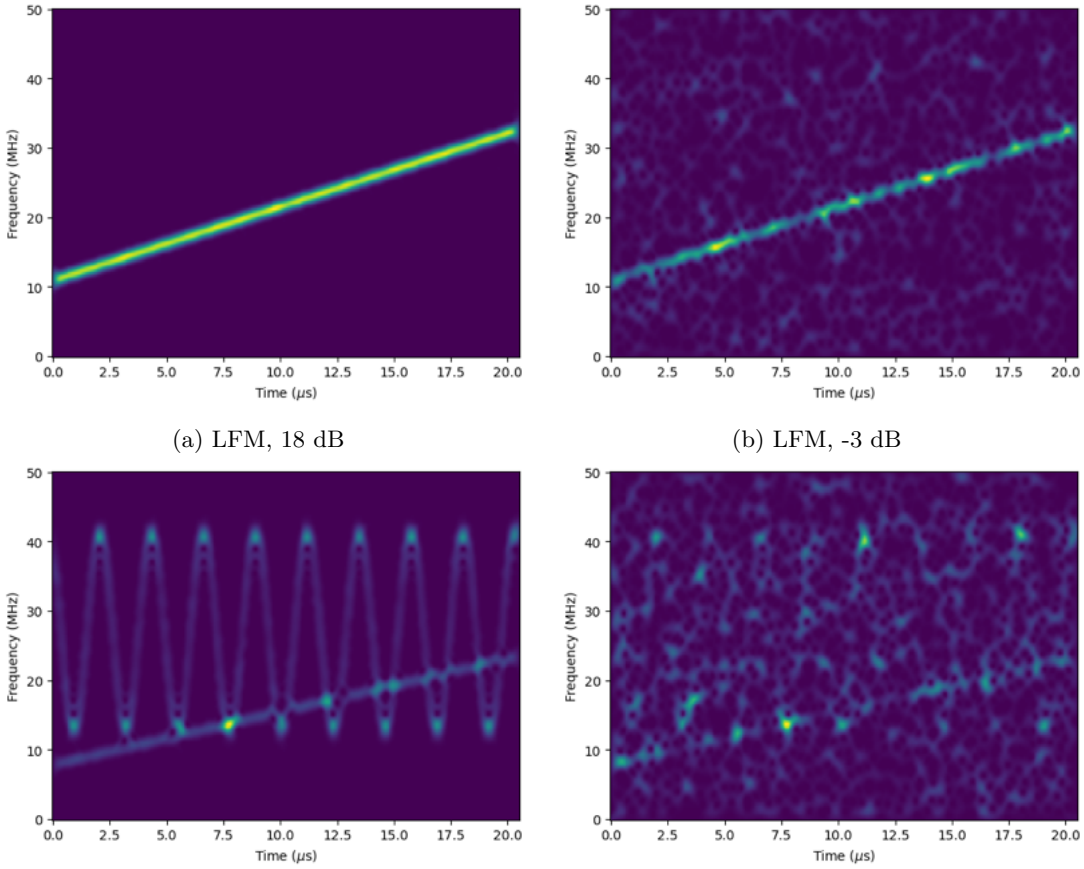
(a) 0 dB

(b) -6 dB

(c) -12 dB

Figure 3.1: Spectrogram of an LFM signal with various SNR values.

we see that it starts to become unrecognizable around -12 dB. Given how distorted its path becomes, we also gather that the confusion between polyphase and frequency modulations mentioned before will be worsened, especially as the white noise introduces circular features which can be confused with what is found in polyphase modulations. Distinguishing between desired and superfluous features is a challenge models will have to overcome. Furthermore, dual-component signals have it even worse, since the power of the signal is shared between both components. This means that, as SNR is lowered, each component will be less distinguishable, as noise begins to overpower both components more easily. Not just this, but there exists the risk of dual-component samples being classified as single-component if the power of one component is larger than the other.



(a) LFM, 18 dB

(b) LFM, -3 dB

(c) LFM and SFM, 18 dB

(d) LFM and SFM, -3 dB

Figure 3.2: Comparison of spectrograms of signals with different modulation types, noise levels and number of components.

Chapter 4

Classification of intrapulse modulation

The intrapulse modulation of a signal is typically classified in literature with CNNs [XLH22], which are usually applied on a time-frequency transform of a signal, since the result of the latter is essentially a readily interpretable image, noise permitting. CNNs have been extensively used successfully in image classification, so they are a logical choice.

Using them on time-frequency images is not a requirement, as deep learning models are well-known to learn how to extract features on their own. Similarly, models could also be rid of the inductive bias introduced by feature extraction through convolutions in CNNs, in our case, by making use of a Vision Transformer-based architecture. This should allow the model to learn specific patterns particular to the problem on its own from the data, without being held down by these biases.

4.1 Preprocessing

Often, but not always, the radar signal is processed in order to extract information about the frequency contents of the signal as they change in time. Transforms that take the signal to the frequency domain like Fourier do not express the location in time of spectral content, so time-frequency distributions (or transforms, or representations) are what intuitively should be used given the nature of this data.

A time-frequency distribution (TFD) [Coh89] is a complex-valued representation of a given signal depending on both time and frequency. The modulus squared of said function represents the instantaneous energy (or power, or intensity) of the signal, and the argument represents the phase. There are many ways to obtain such a function, so we will only look at the methods relevant to the literature seen here. The time-frequency image (TFI) is a real-valued field made up of the modulus squared of the TFD, and is usually the input for most of the models seen in literature.

Linear time-frequency distributions are those that meet the condition, given a composite signal $s_1 + s_2$ made out of the sum of signals s_1 and s_2,

$$F_{s_1+s_2}(t,f) = F_{s_1}(t,f) + F_{s_2}(t,f). \tag{4.1}$$

The most prominent example is the short-time Fourier transform (STFT), a natural first choice, and indeed it is widely used [FLC13], as well as having been used for this problem in particular [Wan+17; Qu+19a]. It's defined as

$$F_s(t,f) \equiv \int_{-\infty}^{\infty} s(\tau) w(\tau - t) \mathrm{e}^{-\mathrm{i}2\pi f\tau} \mathrm{d}\tau. \tag{4.2}$$

It is essentially a Fourier transform with an additional term multiplying the integrand such that when $\tau \to \infty$, $w(\tau - t) \to 0$, carrying out the transform in a limited region, having different values as t changes. $w(t)$ can be chosen arbitrarily. The most intuitive function to use is a rectangular one, however it exhibits the worst distinguishability of frequency values out of all the other usual window functions. Functions like Hann and Hamming are much more common and effective. The discrete version, for a signal with infinite samples, is

$$F_s[m,f] = \sum_{n=-\infty}^{\infty} s[n]\, w[n-m]\, \mathrm{e}^{-\mathrm{i}2\pi f n}, \tag{4.3}$$

and for a finite amount of samples, N, which is our case, it's

$$F_s[m, f] = \sum_{n=0}^{N-1} s[n]\, w[n-m]\, \mathrm{e}^{-\mathrm{i}2\pi f n/N}. \tag{4.4}$$

The STFT requires us to choose the windowing function and its length, which can not be changed once selected, as opposed to other time-frequency distributions. Optimal length of the window seems to depend solely on the number of samples that the signal is made up of. The TFI of the STFT is also called the spectrogram. In the following, the STFT is what we will use to obtain a TFI.

Wavelet transforms are linear as well, however they have not been used as much in this problem. Possible reasons include an increased computational complexity, something which is not adequate for real-time processing of radar signals [Qu+19a], and reduced resolution at higher frequencies [FLC13].

Bilinear time-frequency distributions manipulate a product of the signal with itself (at different points in time) so they are no longer linear; equation (4.1) no longer holds. Composing two signals now looks more like

$$\begin{aligned} F_{s_1+s_2}(t,f) &= F_{s_1}(t,f) + F_{s_2}(t,f) + 2\,\mathrm{Re}\{F_{s_1 s_2}(t,f)\} \\ \text{(WVD)}\quad F_{s_1 s_2}(t,f) &= \int_{-\infty}^{+\infty} s_1\left(t+\frac{\tau}{2}\right) s_2^*\left(t-\frac{\tau}{2}\right) \mathrm{e}^{-\mathrm{i}2\pi f t}\,\mathrm{d}\tau \\ F_{s_1}(t,f) &= \int_{-\infty}^{+\infty} s_1\left(t+\frac{\tau}{2}\right) s_1^*\left(t-\frac{\tau}{2}\right) \mathrm{e}^{-\mathrm{i}2\pi f t}\,\mathrm{d}\tau. \end{aligned} \tag{4.5}$$

The last term is called a crossterm, and the two first terms are called autoterms. Autoterms are easy to interpret, but crossterms are usually undesired, as they arise from the interference between time and frequency due to the quadratic nature of the distribution.

A well-known, but problematic choice of bilinear distribution is the one we detailed in equation (4.5), the Wigner-Ville distribution (also called Wigner function or Wigner distribution function). It is not able to extract phase information, and crossterms impede proper modulation type recognition. Additionally, it has the highest possible time vs. frequency resolution that is mathematically possible, but is very sensitive to noise, as well as being unsuited for multi-component signals due to crossterm interference. Thus, this distribution is not used at all in literature, at least not without modifications [FLC13].

In order to minimize the influence of crossterms while still attempting to have good resolution, a family of transforms called Cohen-class time-frequency distributions (CTFD) are used, the likes of which have the form

$$CTFD_s(t,f) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \Phi(\tau,\nu) A_s(\tau,\nu) \mathrm{e}^{\mathrm{i}2\pi(\nu t - f\tau)} \mathrm{d}\tau\,\mathrm{d}\nu, \tag{4.6}$$

where $A_s(\tau,\nu)$ is the ambiguity of the signal, defined as

$$A_s(\tau,\nu) \equiv \int_{-\infty}^{+\infty} s(t) s(t-\tau) \mathrm{e}^{\mathrm{i}2\pi f t} \mathrm{d}t, \tag{4.7}$$

and $\Phi(\tau,\nu)$ is the kernel or filter in the delay-Doppler domain. It is often a low-pass function, as autoterms, which are what we are usually interested in, concentrate near the origin in said domain, and crossterms appear away from it. The WVD is a trivial CTFD example, with a kernel equal to 1, so absolutely no filtering occurs. Another example of this distribution is in [ZDG17], where the Choi-Williams distribution (CWD) is used. [QMD18] and [Qu+19b] take into account the necessity of maximizing the distinguishability of the classes and noise reduction, so they use a CTFD with a different kernel than what is usual,

$$\Phi(\tau,\nu) = \mathrm{e}^{-(\alpha\tau^2+\beta\nu^2)}. \tag{4.8}$$

All of these kernels manage to keep a decent resolution in both time and frequency while reducing crossterm interference significantly. Such interference can also be reduced when using the Hilbert transform, which transforms a real signal into an analytic one. This is because the Fourier transform of a real signal has a Hermitian symmetry, meaning that $\mathrm{Re}\{\mathcal{F}_x(f)\} = \mathrm{Re}\{\mathcal{F}_x(-f)\}$, $\mathrm{Im}\{\mathcal{F}_x(f)\} = -\mathrm{Im}\{\mathcal{F}_x(-f)\}$. The undesirable consequence of this is that there is also interference between positive and negative frequency components; the Hilbert transform effectively removes every negative frequency component of the signal, thus also reducing this interference when applying a bilinear distribution to an analytic signal versus applying it to a real one.

Linear time-frequency distributions, however, contain no such interference as mentioned previously, and as such, using the Hilbert transform has no bearing on their performance. However, these distributions are still affected by the Hermitian symmetry of real signals' spectrum. Obtaining a TFI from said distribution would contain negative frequency components which mirror positive frequency ones, since the modulus of the signal, which is what's represented in a TFI, has a reflection symmetry about the frequency origin. Therefore, in order to omit redundant information from input data, we shall remove the part of the spectrum belonging to negative frequency components.

4.2 Models

We have selected three models from the available literature on radar signal analysis which have shown considerable performance in the problem at hand in order to compare them against our own, namely those seen in [QMD18], [Qu+19b] and [YLW22]. Aside from these, we also trained two much newer models looking to improve even over transformers from [Li+21] and [Liu+22]. However, extensive modifications to their architectures had to be made to adapt them to the nature of our particular dataset, as well as to compare them in a fair way. In the following, we shall detail their approach, as well as ours, to clearly define and justify where we deviated from them.

4.2.1 Qu et al. 2018 CNN

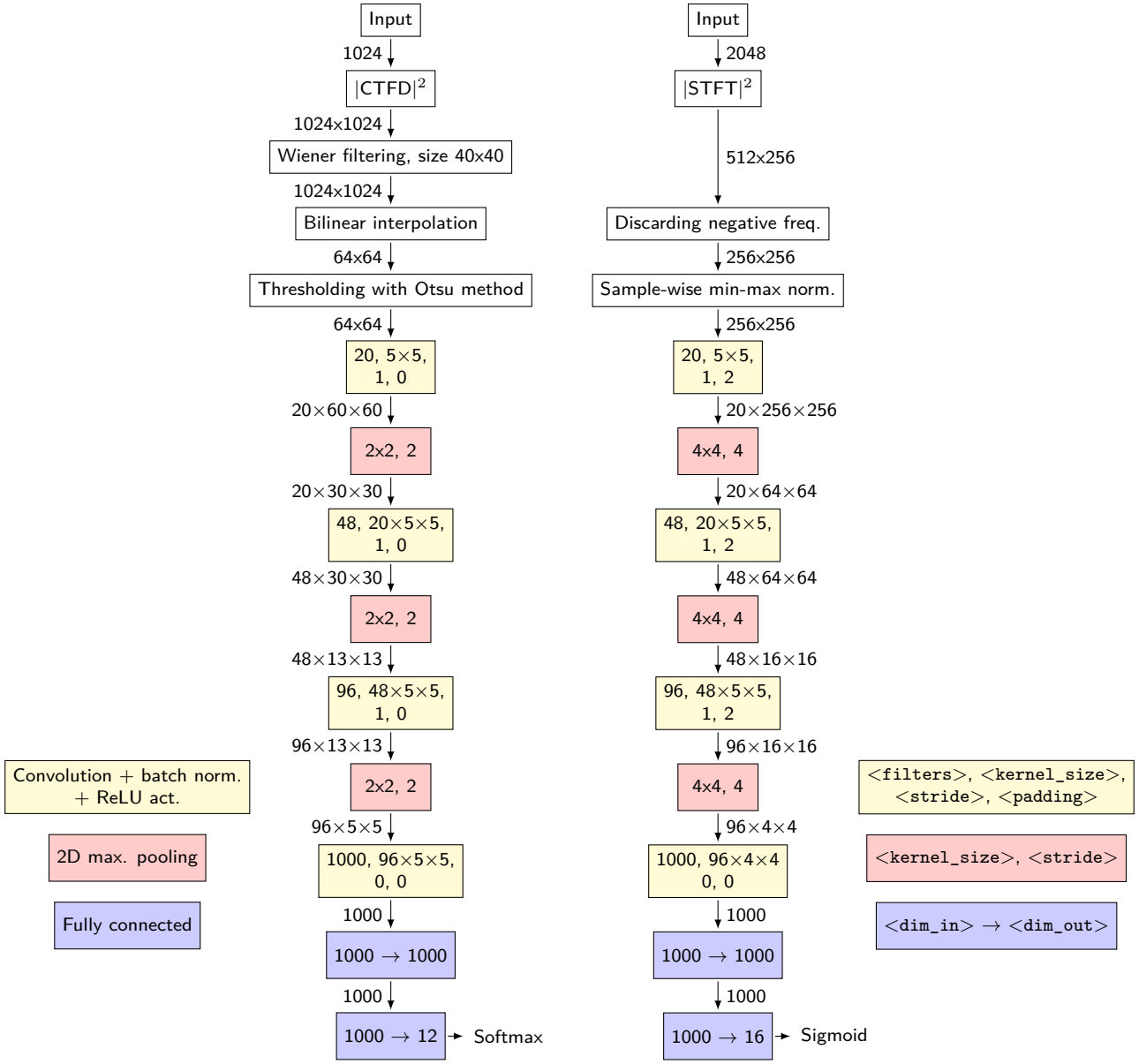


Figure 4.1: Model seen in [QMD18] (left) compared to our customized implementation (right). Beside the arrows, the size of the resulting feature maps is shown.

The model seen in [QMD18] has, as its input, a complex signal with I/Q sampling, containing 1024 data points. They then employ a CTFD which transforms it into a two-dimensional array with size 1024x1024, take its absolute value squared, applying a Wiener filter in order to remove noise, interpolating it down to 64x64, and finally applying

Otsu thresholding to set every input feature to 0 or 1. This seems to be a good compromise between accuracy and inference time, as they apparently obtain decent accuracy in their test datasets, however we are including additional modulation types in our work which are not seen in theirs, like P1, P2, P3, and P4. As we described back in section 3.3, the fact that input size is 64x64, as well as introducing noise at the same time, may impede proper recognition of these modulation types. Hence, we decided to increase the input size of our implementation of this paper's model to 256x256, owing to the particular nature of our dataset.

In the preprocessing stage, our implemenation has removed Wiener filtering and Otsu thresholding. The former was removed since we wanted to make the comparison as fair as possible between models, seeing how well they could ignore the noise on their own, since other models do not use such filters. The latter was removed for much the same reason; an even ground was sought, and Otsu may remove valuable information from input data which other models would possess. Aside from these, we have decided to use the STFT instead of the CTFD suggested here, mostly due to time constraints, as the latter would have had to be implemented for this work, while the former is readily available in the libraries used here. Instead of thresholding, which in part works as a normalization procedure, we used sample-wise min-max scaling, using a minimum of 0, already assumed from the nature of taking the absolute value of the transform. The maximum value, as is standard, is taken equal to 1.

The CNN itself as seen in this paper contains several sequential convolutional and pooling layers, similarly to AlexNet [KSH17] or VGG [SZ14] architectures, though as opposed to these, this model in particular is much less deep than these two. As we have mentioned, the input size they used was 64x64, with a kernel size of 5x5 in every layer. If we were to use an equivalent receptive field at this layer, the kernel size would have to be 20x20 for our input size of 256x256. However, convolutions of that size are known to be more computationally expensive and can only capture bigger features. One might think of reducing the kernel size, owing to this complaint; AlexNet's example, containing an 11x11 convolution and stride 4, is a good starting point. However, 11x11 is still relatively big. VGG instead uses 3x3 convolutions at the first layers. We decided to strike a middle ground, so we ended up keeping the convolution kernel size equal. The receptive field at the first layers is then smaller in our case, but this we can justify as well, by pointing out the fact that we have new modulations not seen in their work containing smaller features, which warrant the usage of smaller receptive fields. We also increased the size of the pooling from 2x2 to 4x4, as we were incurring in a parameter count far too large for the task at hand, largely at the last convolutional layer, as it is used as a linear combination of the feature maps to compose a 1D array. We sought to match the number of feature maps that the paper had, so that could not be changed, and we did not wish to change the convolution size either. Options to trim model parameters were then using strided convolutions, dilated convolutions, or more aggressive pooling, as well as the combinations thereof. We decided on just using the latter. Lastly, we used batch normalization right after every convolutional layer and before activation, while they did not. This makes up yet another decision done to make this model more competitive, as the rest of the models we considered also make use of batch normalization, which is a standard feature in these models nowadays. This model, as we have implemented it, has 2 698 208 parameters.

4.2.2 Qu et al. 2019 CNN

This model has the same preprocessing as before, with the exception of the removal of Wiener filtering and Otsu thresholding. The CNN itself uses different versions of Inception modules laid out sequentially in order to extract features, before applying a single fully connected layer as a classifier head. What is most interesting about their approach in this case, is that they do not employ a single training stage; rather, they split this into pre-training and fine-tuning. Pre-training takes the first convolutional layer, as well as the first four Inception modules, and couples them to four sequential deconvolutions. They then make the model act as a convolutional autoencoder, pre-training the first convolutional layer and first four Inception modules by feeding the autoencoder noisy signals, and using clean signals as the target. This forces the model to extract the right features and making it more robust to noise, in principle. Fine-tuning then takes place by removing the deconvolutions, applying an Inception v3 (b) module, a fully connected layer and softmax normalization, to then train the model by feeding it noisy signals and then using one-hot encoded vectors as targets. Similarly to other decisions taken before, we will not use this training method; we will only carry out one training stage, consisting of training the model from scratch using the second stage seen in [Qu+19b] and updating all model parameters at once.

As for architectural considerations, we will keep the Inception modules mostly untouched, as those were specifically engineered for an input size similar to ours, and the features that they were designed to capture (ImageNet) also range in size widely. The size of these feature maps will be larger, but to compensate, we will only increase the latest maximum pooling layer from 2x2 to 4x4 in order to get a reasonably-sized vector with 4096 features before applying the classifier. Our implementation of this model has 263 712 parameters.

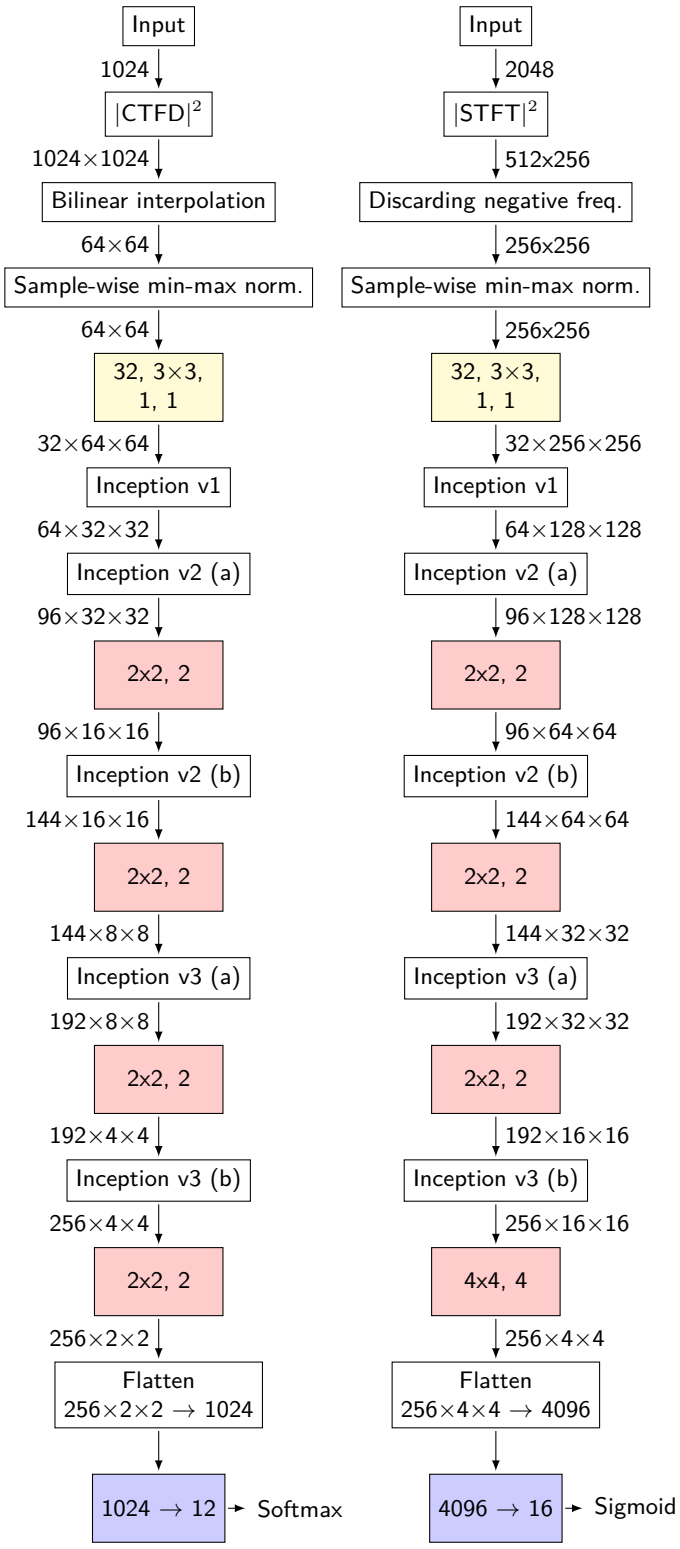


Figure 4.2: Model seen in [Qu+19b] (left) compared to our customized implementation (right).

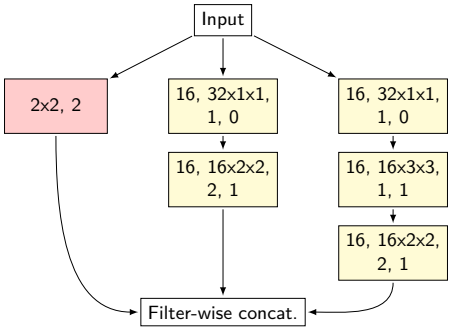
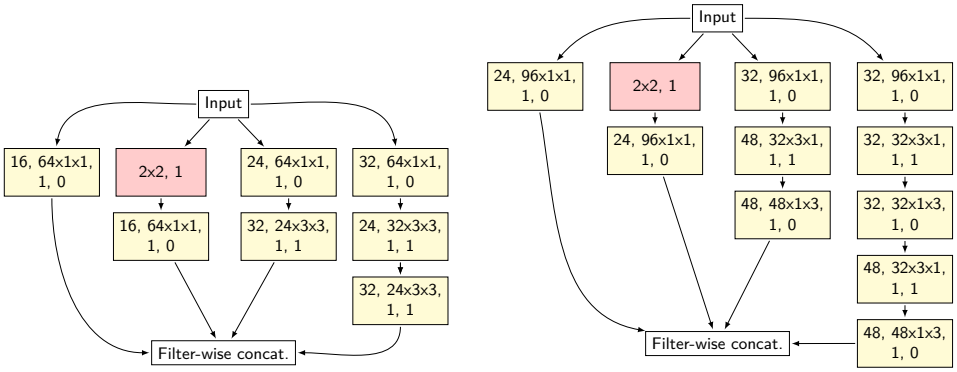
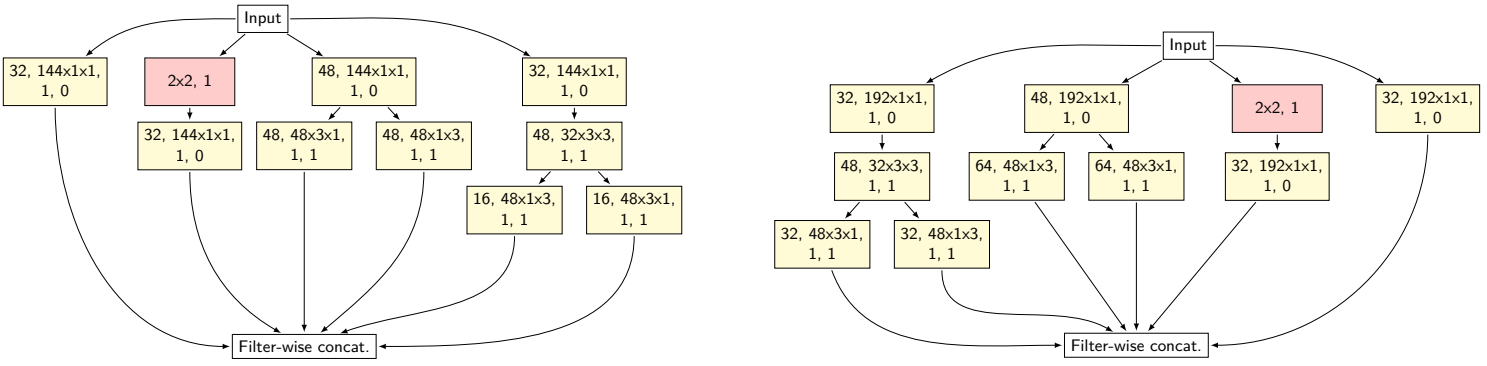


Figure 4.3: Inception v1 module.



(a) v2 (a) (b) v2 (b)

Figure 4.4: Inception v2 modules.



(a) v3 (a) (b) v3 (b)

Figure 4.5: Inception v3 modules.

4.2.3 ConvMLP

Aside from implementing models tailor-made for this problem in particular, we have implemented more cutting-edge models such as ConvMLP [Li+21], which, to our knowledge, have not been used to explore this problem in literature. This model's architecture consists of the following: first, a convolutional tokenizer is used to extract the initial feature map. To reduce computation and improve spatial connections, tokenization is followed with a pure convolutional stage, then followed by three ConvMLP stages before applying a global average pooling layer across to the output feature map and finally a single fully connected layer as the classifier head. Each ConvMLP stage includes multiple ConvMLP blocks, while each ConvMLP block has one channel MLP followed by a depth-wise convolutional layer, succeeded by another channel MLP. Residual connections are introduced, and layer normalization is applied to inputs in the block. Each channel MLP consists of two fully connected layers with a GELU activation and dropout.

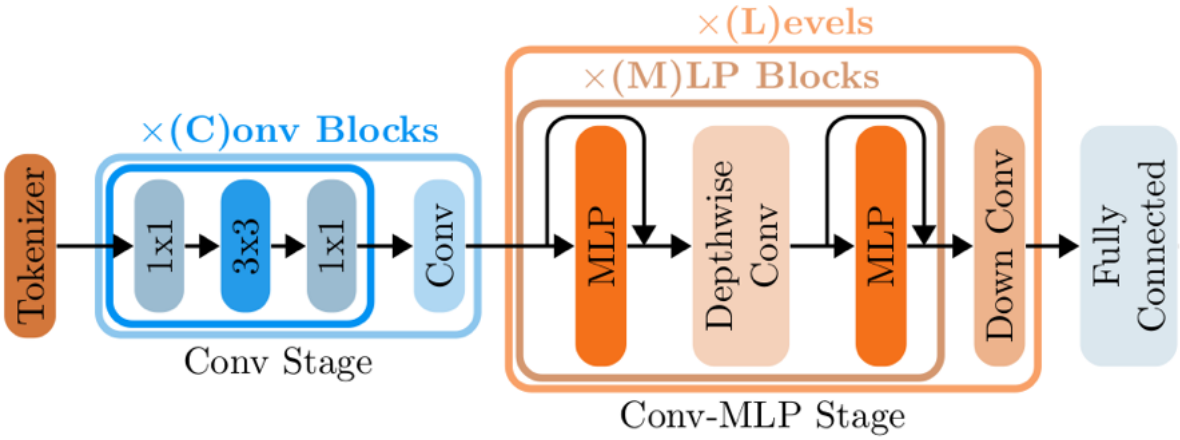


Figure 4.6: ConvMLP architecture [Li+21].

We have used hyperparameters which have left us with a model similar in size to that which was found in our implementation of [Qu+19b], and is far smaller than ConvMLP-S as described in [Li+21] (0.3 million parameters versus 9 million). We kept the same overall structure as the original paper; a convolutional tokenizer which extracts the initial feature map, then one convolutional stage, and after that, three ConvMLP stages, which then feed the classifier head. The convolutional tokenizer first takes our input image, having gone through the same preprocessing as the two previous models, and outputs the first feature map with 16 channels. Then, this is fed into the convolutional stage, which contained 2 convolutional blocks, increasing number of channels to 32. After that, the three ConvMLP stages follow, with the first increasing the number of channels to 64, the second to 128, and the last one leaving the feature map with the same number of channels, before finally feeding a fully connected layer. Each one of these stages has one ConvMLP block. The scaling ratio of hidden layers in the MLPs of these blocks is 2. The model we trained, according to these hyperparameters, has 298 712 parameters.

4.2.4 ConvNeXt

ConvNeXt is another novel convolution-based architecture [Liu+22], also not used yet on radar signal analysis to our knowledge, which was built by gradually "modernizing" a ResNet [He+16] following the hierarchy of Vision Transformers, looking to analyze how design decisions in ViTs influence CNNs' performance. They made several additions and changes to ResNet, such as: added a "patchifying stem" at the start of the net, using 4x4 non-overlapping convolutions, using grouped convolutions similarly to ResNeXt [Xie+17], using pointwise and depthwise convolutions, creating inverse bottlenecks, larger kernel sizes and moving the depthwise convolution to the beginning of the block. Other minor changes were also been made with respect to previous designs, such as replacing ReLU activation with GELU, using less activation functions, less normalization layers, and replacing batch normalization for layer normalization.

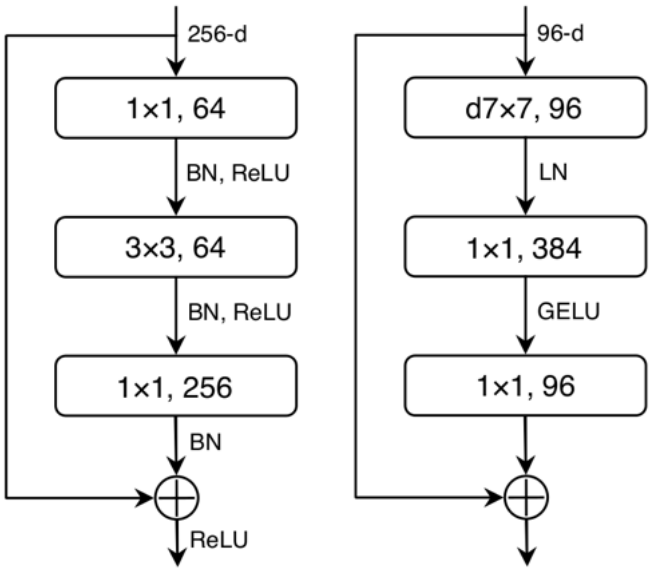


Figure 4.7: ResNet block (left) versus ConvNeXt block (right) [Liu+22].

The model then contains the following; the stem is made up of 16 4x4 convolutions with stride 4, therefore outputting an initial feature map of size 16x64x64. After this, four stages made up of several blocks as seen in figure 4.7 operate on this feature map. Our concrete choice of hyperparameters consisted in having one block in the first two stages, three blocks in the third stage, and one block in the last stage. The first stage had 16 convolutions, the second 32, the third 64 and lastly the fourth had 128, similarly to ConvMLP. The model we have trained has 306 672 parameters.

4.2.5 Yuan et al. 2022 CNN+ViT

The last model from literature related to radar analysis [YLW22] holds many considerable differences from the models we described in subsections 4.2.1 and 4.2.2. These start at the nature of the dataset they employed; the input length of the signal is much larger than the other two models, having 32768 data points instead of 1024, as well as ours at 2048. They use single-channel sampling, resulting in a real signal as opposed to a complex one. The last notable discrepancy with respect to all other approaches seen here which is related to data is that the signals they use do not always occupy the entire length of the input, they opted for a variable length between 5 000 and 30 000 data points, and then zero padding to the end of the input to 32 768. After that, they apply a Fourier transform to the signal, obtain the absolute value of said transform, and then apply sample-wise min-max scaling to it as a normalization procedure, with minimum and maximum values equal to what was used before.

Notably, however, after inspecting their procedure as they describe it, they do not seem to discard values of the Fourier transform of the signal corresponding to negative frequencies before feeding it to the model. Since they use real signals, this means that the spectrum is symmetric about the origin, therefore there is redundant information that can, and probably should, be excluded from the input. As before, our implementation has taken this consideration into account, and did remove the part of the spectrum belonging to said frequencies.

After preprocessing, the signal is passed through four sequential 1D convolutional layers with the same size and stride. We have not been able to replicate the size of their receptive field with relation to the input size, as the number of data points our model intakes is just too low. After these convolutions, the feature map is fed to four sequential transformer encoders, along with a classification token, to then feed this token to a fully connected layer. This model, as trained here, has a total parameter count of 862 096.

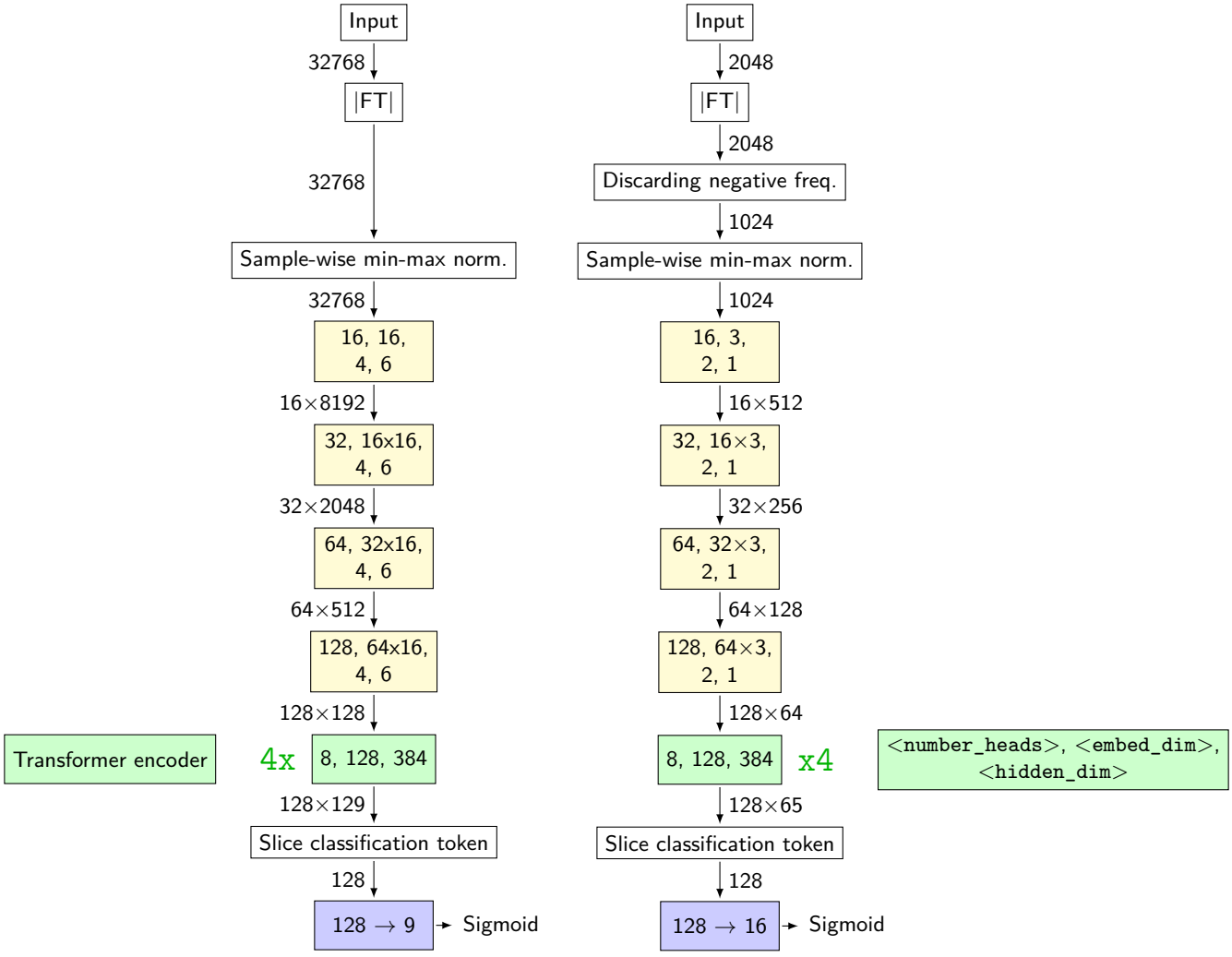


Figure 4.8: Model seen in [YLW22] (left) compared to our customized implementation (right)

4.2.6 Multimodal ViT

As mentioned in chapter 1, our model makes use of three signal domains; time, frequency, and time-frequency. It is well-known that even bilinear time-frequency distributions cannot determine the location of a signal perfectly in both time and frequency simultaneously, due to the uncertainty principle present between these variables. Our intuition is that, by allowing our model to observe all three domains at once, it would be able to better determine the modulation type of a signal.

Preprocessing was almost identical to previous models. We first take the time domain signal and apply both a Fourier transform and a short-time Fourier transform in parallel, thus obtaining the frequency and time-frequency domain signals, respectively. After this, in both of these domains, we eliminate signal values corresponding to negative frequencies. We take the absolute value of the FT and the absolute value squared of the STFT, and then, on both of these domains, we carry out sample-wise min-max scaling, as well as in the time domain.

Each domain is now handled by a typical ViT architecture, almost entirely based on [Dos+20]. Both the TFI and the 1D signals are resized each into a sequence of patches, then these patches are flattened to a 1D array in the case of the TFI, and they are projected into the transformer's patch embedding space through a learned linear transformation. We add a learned positional encoding vector to these patches, as well as concatenating a different learned classification token to each of the sequences. These three sequences of embeddings are each fed to their corresponding series of encoders. After encoding these patches, we slice the encoded classification tokens from this sequence, concatenate them, and feed them to a single fully connected layer acting as the classifier head.

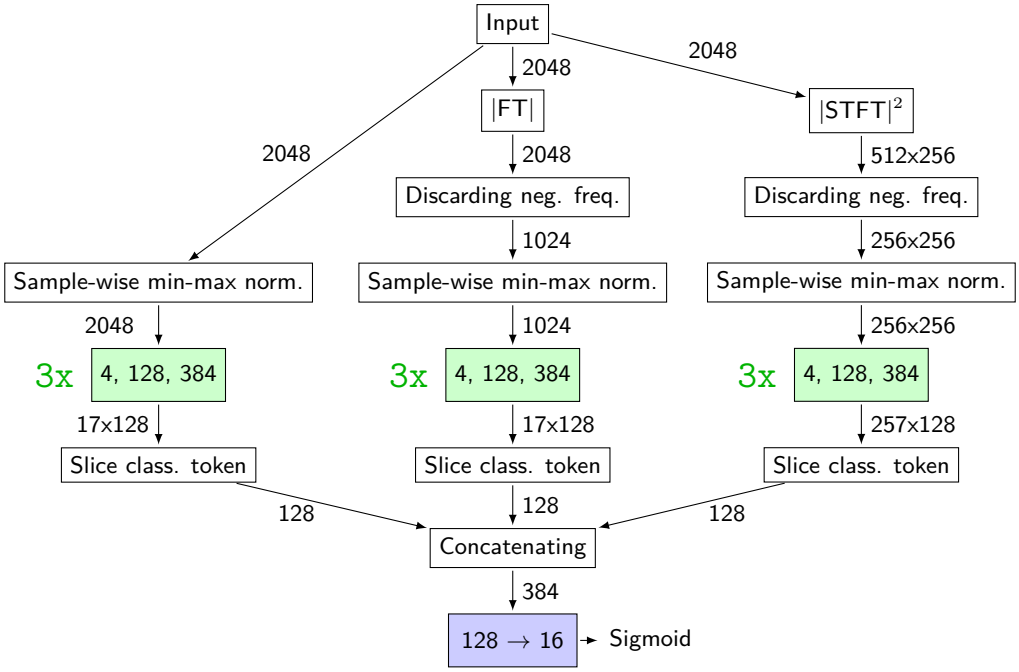


Figure 4.9: Novel multimodal vision transformer proposal.

Transformer encoder hyperparameters are similar to those seen in subsection 4.2.5. We used an embedding space dimension of 128, hidden layer dimension of 384, but 4 heads instead of 8. As we mentioned before, we observed that literature on this problem typically uses models which are as wide as bigger, more general models trained on bigger datasets, but oftentimes they are less than half as deep. Following this trend, instead of completely guiding ourselves by what was seen in [YLW22], we decided to use some of the hyperparameters seen in the ViT-Ti/16 model in [Ste+22].

There is a piece of intuition that is worth mentioning, as it influenced decisions made when defining our model. A different number of dimensions among every domain's encoded classification token implies a different number of parameters in the classifier head relating them to the output vector. This may have the undesired effect of not letting the model properly weigh all three domains equally and fairly. Therefore, we should impose that the embedding dimension in each domain be equal. However, this is not a straightforward proposition.

Let us elaborate on this, starting from the choice of patch size in every domain. Ideally, we would use a patch size of 32 and 16 data points in the time and frequency domains respectively, which would take advantage of the higher resolution in each domain, given its small size with respect to the input. If we were to attempt to match this number of points in the TFI encoders though, a patch would have a size of 4x4, which would increase computational requirements significantly. Increasing patch size to a standard value such as 16x16 leads to each flattened patch having 256 dimensions. If the dimensionality of the patch embeddings is to be the same in all encoders, then this means that the learned linear embedding found in the time-frequency encoders would compress information too much, going from 256 to 16 dimensions. This might lead to overreliance on the two other domains, potentially leading to suboptimal performance. Conversely, making the embedding dimension 128 would mean that a linear embedding transforms the flattened patch with dimension 16 to 128, which could potentially make the embedding, and possibly the model, sparse, leading to overfitting and worse performance.[1] The conclusion is that making embedding dimensions equal naïvely would not be quite as beneficial.

It becomes clear that we must make a compromise, as this is not immediately solvable; rememeber what the input size of each domain is. We have a time domain signal with 2048 data points, a frequency domain signal with 1024 and a TFI with 256x256 = 65536. Our inputs are quite disparate in size. We opted for a standard patch size of 16x16 in the time-frequency encoders, 64 for the frequency domain and 128 for the time domain. This means that we have 256 patches in the sequence fed to the time-frequency encoders, and 16 patches in both time and frequency encoders. A simple workaround may also be explored in future work, inspired by convolutions and the STFT; making patches in the time and frequency domains overlap each other. This, however, is not an ideal solution either, since this means that patches contain redundant information. This model has a total parameter count of 2 086 032.

[1] It is not immediately clear whether or not also having the same sequence length among these encoders is optimal for the performance of the model.

4.3 Implementation

The programming language Python 3.9.16 [22] was used in order to implement every single process discussed up until now. Along with it, we made use of several industry-standard libraries to accelerate both the implementation of these procedures and to reduce computation time. We used `pyradargen`, an unreleased and internal use only signal generation library, in order to acquire the dataset employed here. This library was written, as its name suggests, in Python, making heavy use of NumPy 1.24.3 [Har+20]. The library PyTorch 1.13.0 [Pas+19] was used to implement models, the training and testing procedures, as well as the preprocessing steps like obtaining the time-frequency image through the short-time Fourier transform and normalization. We also used scikit-learn 1.2.2 [Ped+11] to calculate classification metrics.

As for preprocessing, we used the Hann function as the window in the STFT, as well as a window length of 101 data points with a stride of 8, as well as having zero padded each window from 101 to 512 points, in order to obtain a spectrogram which has 512 frequency values, before getting rid of negative frequencies. Zero padding adds no extra information, it only has the effect of interpolating between signal values in the frequency domain.

In the interest of reproducibility, and before explaining our training procedure, we will detail a different, but important aspect of the work we made. In contrast to the models seen in subsections 4.2.1, 4.2.2, 4.2.5 and 4.2.6, we have not written the code needed to implement the ConvMLP and ConvNeXt models expressly for this work. Instead, we sourced these models' definition from each of their GitHub repositories where all the code used in [Li+21] and [Liu+22] resided, respectively. We then made a small change to the ConvMLP definition, in order to allow it to intake an image with one channel instead of three, while ConvNeXt underwent no changes. After that, we instantiated an object of its class for training and inference, the same way we did with the rest of the models, using these parameters with ConvMLP: `blocks=(1, 1, 1), dims=(32, 64, 128), mlp_ratios=(2, 2, 2), channels=16, n_conv_blocks=2, classifier_head=True, num_classes=16`. With ConvNeXt, we used these parameters: `in_chans=1, num_classes=16, depths=[1, 1, 3, 1], dims=[16, 32, 64, 128], drop_path_rate=0.1, layer_scale_init_value=0., head_init_scale=1.`

During the training of the CNN models, we used a minibatch size of 256 samples, 25 epochs, with a learning rate of 10^{-4}. The optimizer used was Adam, with $\beta_1 = 0.9$ and $\beta_2 = 0.995$. Transformers, however, underwent a large amount of tests with different hyperparameters, as we explored their ability to learn from the training dataset. The transformer-based models the metrics of which we will discuss have the following set of hyperparameters, which we believed to be adequate taking into account literature and previous models, despite the results we obtained: 70 epochs, with a minibatch size of 256 samples, a learning rate of 10^{-4}, Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and finally learning rate scheduling, reducing the learning rate to 20% every 30 epochs. This means that in the first 30 epochs, we have 10^{-4}, between 30 and 60 it's $2 \cdot 10^{-5}$ and between 60 and 70, $4 \cdot 10^{-6}$.

We employed regularization in various forms; all models had loss penalization through L^2 regularization, with a weight decay of 0.001. We also used weight dropout in most models, however with different probabilities. The VGG-like model used 0.5 in both of its fully connected layers, the Inception-like CNN used 0.1, though ConvMLP and ConvNeXt used no dropout regularization, instead opting for a stochastic depth rate of 0.1 in both models. The convolutional and transformer hybrid had dropout 0.1 in transformer encoders and 0.2 in the classifier head. Our multimodal ViT has 0.2 in transformer encoders and 0.2 in the classifier head. All models used the binary cross-entropy loss function as the criterion, and their output layer uses a sigmoid activation function, with a threshold of 0.5 for all classes. The output of every model seen here is a one-hot encoded vector of length 16. The 15 first features are the classes we described previously, while the last feature is an auxiliary class indicating if a signal is single-component (0) or dual-component (1). This is because, unlike most multilabel classification tasks, we need to clarify whether a class is repeated or not, as we may have a signal with both components from the same modulation type.

All computations were carried out on a system running Windows Server 2022, featuring two Intel Xeon Silver 4314 CPUs, three NVIDIA RTX A6000 GPUs (though only one model per GPU was used at all times) and 512 GB of system RAM.

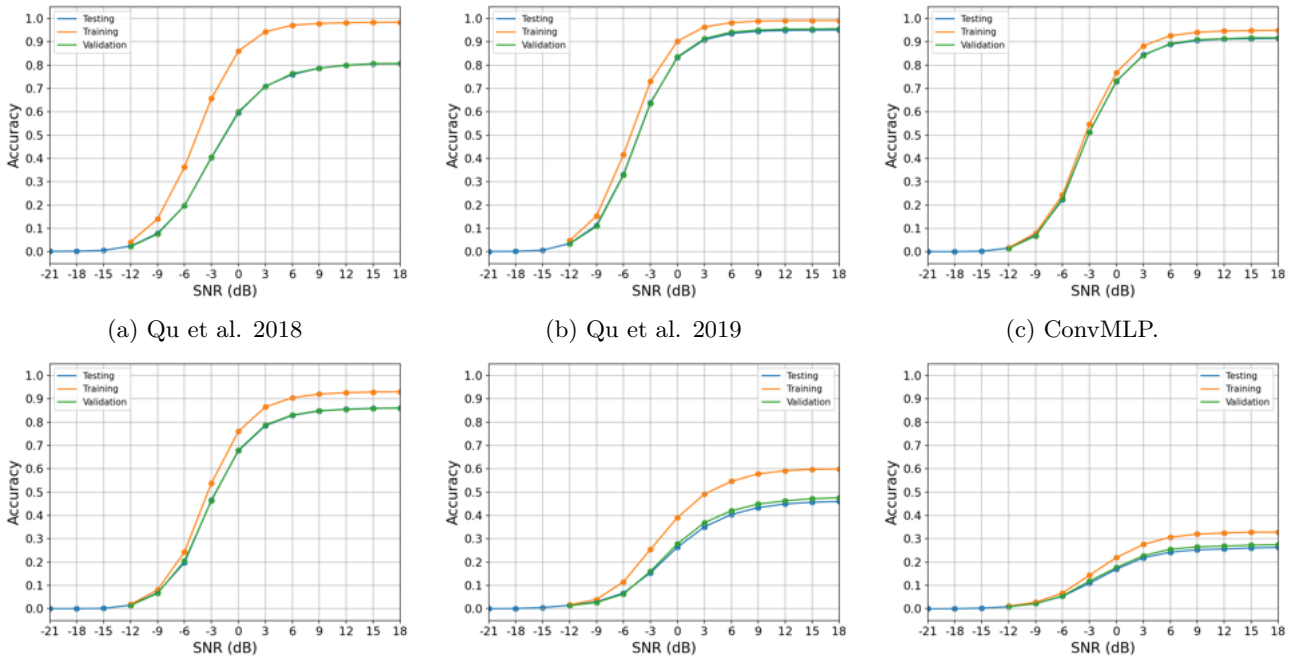
Chapter 5

Results and discussion

During the training process we detailed previously, we already encountered noteworthy differences among CNN-based models versus those incorporating transformer encoders. CNNs were able to attain a reasonable validation accuracy after just 10 epochs, reaching between 55% and 70% accuracy in the validation set, depending on the model.[1] Unfortunately, the same cannot be said for transformers, not just our novel proposal we described in subsection 4.2.6 but also our implementation from scratch of the model from literature in subsection 4.2.5. They have proven to be exceedingly hard to train, something that was in part expected, echoing concerns found in literature [Ste+22; Liu+20; Gon+21]. An extensive array of tests were made to attempt to make the validation and training loss to converge towards 0, however the minimum of this loss hovered around 0.15 and 0.2 for both sets, which translated into an accuracy of about 30% and 20% for the hybrid CNN+ViT model and our proposal respectively. Many different training hyperparameters were modified; learning rate (0.1, 0.01, 0.001, 0.0001...), Adam's β_2 (0.995, 0.999), minibatch size (64, 128, 256), number of epochs (50, 70, 300) as well as regularization hyperparameters: dropout rate of weights (from 0 to 0.5), both in encoders and final classifier head, L^2 regularization (0.1, 0.03, 0.001), gradient clipping to global norm (0.1, 1, 10). Learning rate scheduling was used as well, to no avail, having experimented with exponential decay, triangular periodic learning rate, and decaying learning rate by a certain percentage every epoch. We even ignored, at first, one of the preprocessing steps we mentioned we carry out, which is removing negative frequency values, but including this step made no difference. Our last attempt consisted of swapping the activation function from ReLU to GELU, both in convolutions and transformer encoders, despite the fact that we wished to change as little hyperparameters with respect to literature as possible, however this bore no fruit either.

In the following, we will explore these models' performance by observing their accuracy calculated over several sliced subsets out of the testing set. Let us first plot the accuracy of every model as a function of SNR over the training, validation and testing datasets, in order to explore the level of overfitting or underfitting we have encountered, before making a proper judgement of the results over the testing dataset. We measure this with respect to SNR because, as we'll see, accuracy tends toward 0 as SNR decreases, something which skews accuracy figures downward. We detail this in figure 5.1. From said figure, we can see that some models, like the VGG-based design from subsection 4.2.1, suffer from significantly lower accuracy in the validation dataset compared to the training dataset, which is a sign of overfitting, even after applying stronger regularization than the rest of the models. This might be due to the fact that this model utilizes two fully connected layers instead of one, conversely to every other model seen here, resulting in a difference of about 17% between datasets. The rest of the CNN-based architectures do not have such a strong discrepancy between training and testing, or training and validation accuracies. Both the Inception-based and ConvMLP models have an acceptable difference of about 3%-5% in the most ideal conditions, while ConvNeXt has a slightly larger difference, about 6%. They have comparable differences in their accuracies, so it would be fair to compare them using the testing dataset.

[1] It is necessary to mention that this metric was calculated over all the set, including very noisy samples, so lower accuracy is to be expected. This is why, in the testing stage, we have calculated this metric as a function of SNR.



(a) Qu et al. 2018 (b) Qu et al. 2019 (c) ConvMLP.

(d) ConvNeXt. (e) Yuan et al. 2022. (f) Multimodal ViT.

Figure 5.1: Accuracy as a function of SNR over the training, validation and testing datasets.

We can see that most CNN architectures we trained clearly outperform the transformer-based models, the latter having apparent signs of poor generalization, especially in the case of our proposal. Given this, we have trained these models with weaker and stronger regularization, as well as varying model hyperparameters, however, results were fairly similar to what is shown here in these figures. Besides, the poor performance they both suffer on the training dataset signals deeper issues, so this is not the only aspect that should fall under review. In figure 5.2, we represent the accuracy over the entire testing dataset with every model, in order to observe differences among all models more clearly,

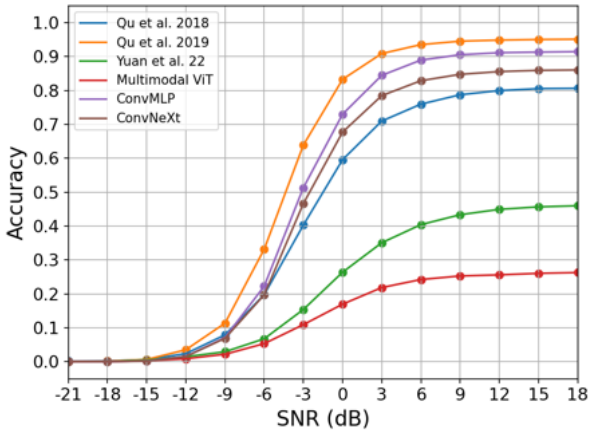


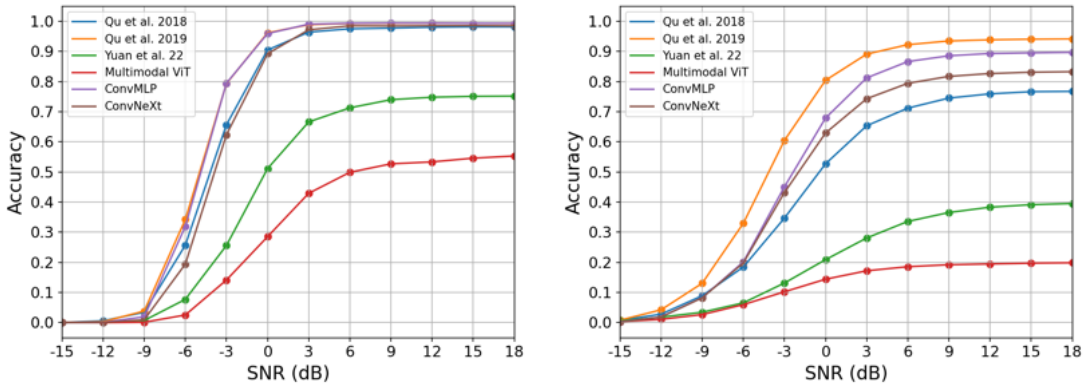
Figure 5.2: Accuracy as a function of SNR over the testing dataset.

The magnitude of the performance difference among transformers and convolutional networks is clearly laid out here. All models start to classify modulation types around -12 dB, but accuracy only starts to reach acceptable levels from 0 dB and higher SNRs. Proper classification seems to be impossible at -21 and -18 dB, as expected, so we will exclude these noise levels from the following figures. Throughout these noisy conditions, the Inception-based model seems to have a clear advantage over every other proposal, reaching beyond 90% accuracy after 3 dB. Interestingly,

it improves even beyond the newer models, which have not been used yet tackle this problem. As we mentioned, we chose a set of hyperparameters that we believed to be appropiate for the task at hand, resulting in these two newer models and the Inception-based model having similar parameter counts. However, this is not a direct indicative of FLOPs or throughput, especially since we are considering significantly different architectures, so measuring these metrics would be necessary to establish which model is ideal when controlling for them. This will be left for future work.

Our proposal seems to have had major difficulties trying to keep up with the rest of the models. Many aspects of the model itself could be the root cause of this behaviour, or even a combination of circumstances could account for this performance. One such aspect could be the inclusion of the time domain into the model's analysis of the signal. In principle, it is reasonable to assume that it is easier to classify a signal if we extract its spectral features as they change in time; however, there are many ways to do so, one of them being just extracting the TFI and manipulating that alone, and this method has proven to work well. In our case, using the time domain without any manipulation, would likely make the model have a harder time learning how to distinguish the features, since noise affects this domain a lot more than the other two. Even if our model did learn how to extract the right ones, it would likely require an increased computational budget in order to do so, and we have assigned the same budget to all domains. Not just that, but it is likely that one domain's path throughout the network somehow dominates over the other two. Given the performance we observe in this figure and in later ones, one possible explanation for this is that it could be that the time or frequency domains are taken more into account than the time-frequency domain. We could also add to this observation that transformers are generally harder to train, so we essentially seem to be facing several difficulties at once. What features are worth extracting the most, as well as choosing the right hyperparameters, are two major questions worth exploring in the future.

Calculating this metric over the entire dataset, which contains both single-component and dual-component samples, is not enough in order to fully grasp the models' behaviour. There is an unequal number of samples with one or two labels (components), and this skews accuracy figures towards the subset containing the most. Not only this, but we have hypothesised the likelihood of models having a harder time classifying modulation types if more than one component is present in section 3.3, so now is the time to put this statement to the test. We break this down in figure 5.3,

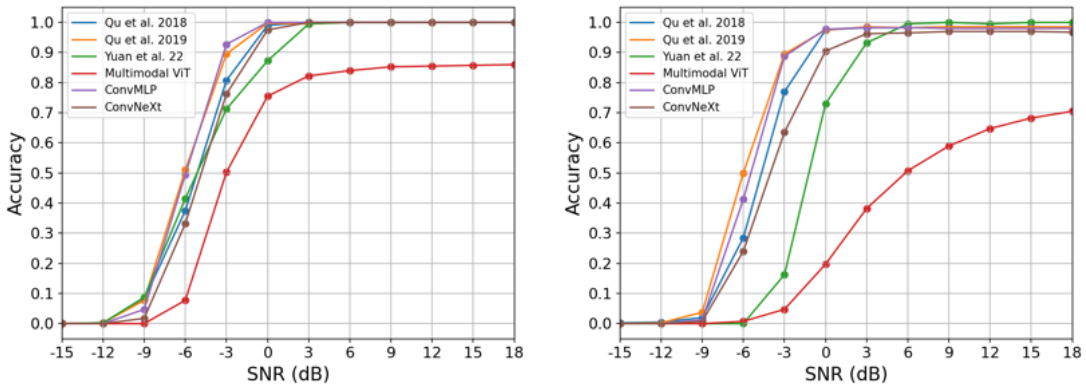


(a) Single-component.

(b) Dual-component.

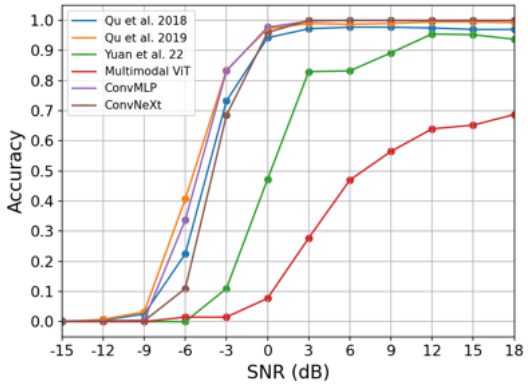
Figure 5.3: Accuracy as a function of SNR over the testing dataset, with varying number of components.

We can see that these figures were very much skewed towards the dual-component case, since we have 84 000 single-component samples versus 378 000 dual-component samples in this dataset. And, corroborating our previous statement, dual-component samples are indeed quite a bit harder to classify, shifting every model's accuracy downward by a large margin, especially that of transformers. This additional difficulty also makes the hierarchy of models' performance much more clear than the single-component case, even in much more favorable conditions where there isn't as much noise. Let us now break down single-component accuracy by class, starting with frequency modulation types, in figure 5.4, and after that, with phase modulation types, in figure 5.5.

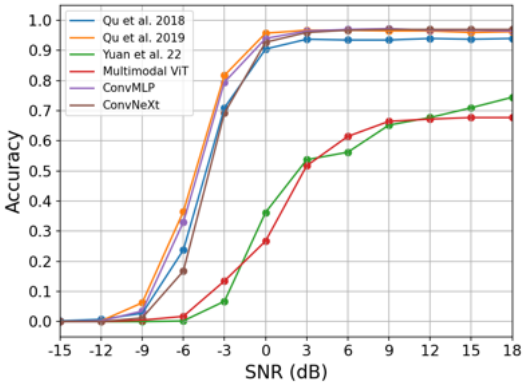


(a) No modulation.

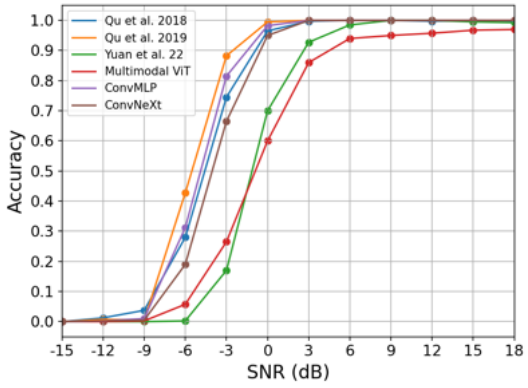
(b) LFM.



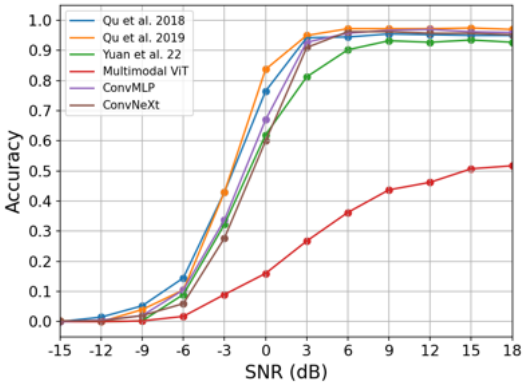
(c) DLFM.



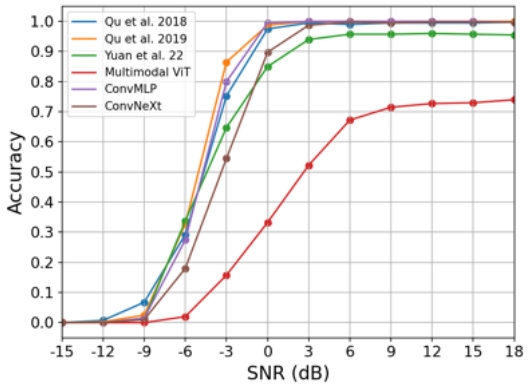
(d) MLFM.



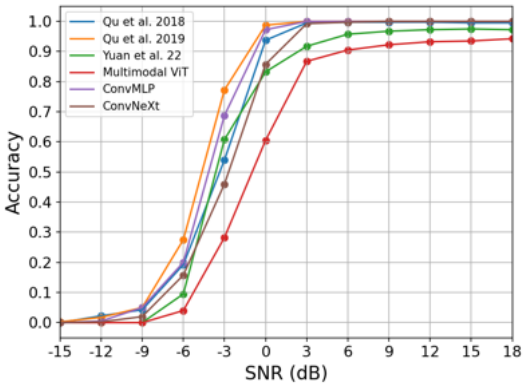
(e) EQFM.



(f) SFM.

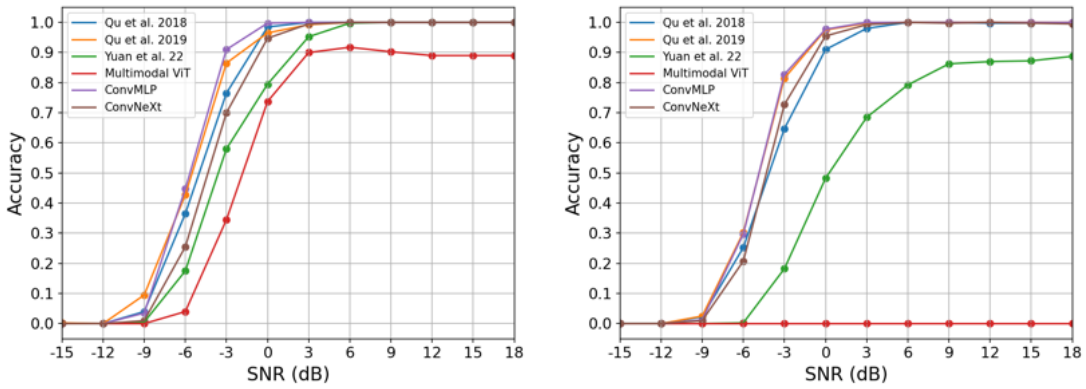


(g) BFSK.



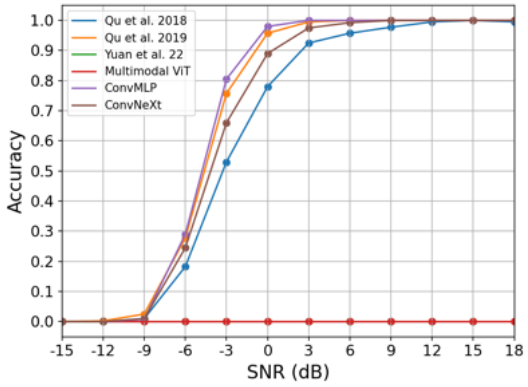
(h) QFSK.

Figure 5.4: Accuracy of single-component samples (with one label) per class as a function of SNR over the testing dataset, frequency modulation types.

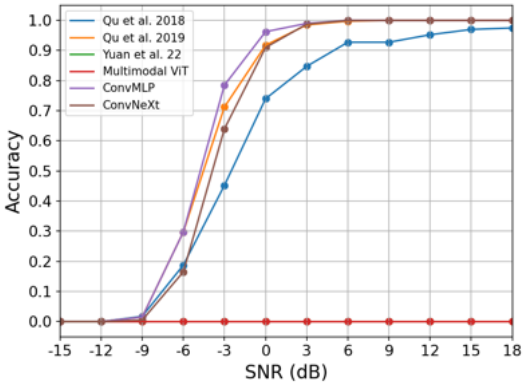


(a) BPSK.

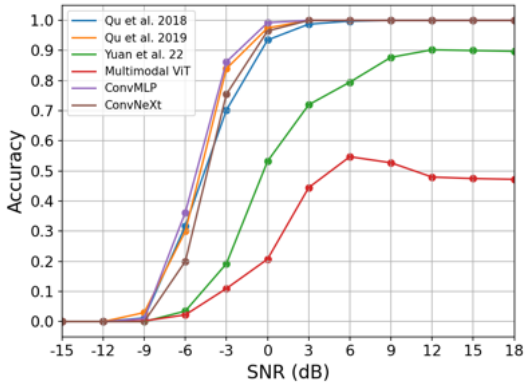
(b) Frank.



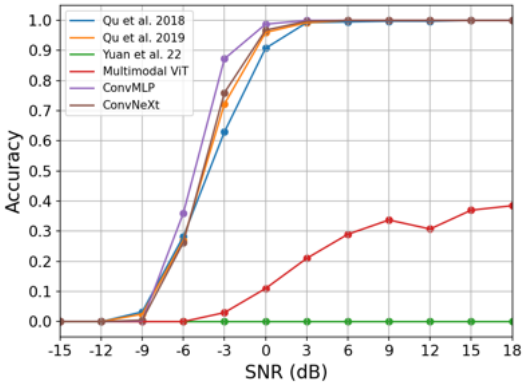
(c) P1.



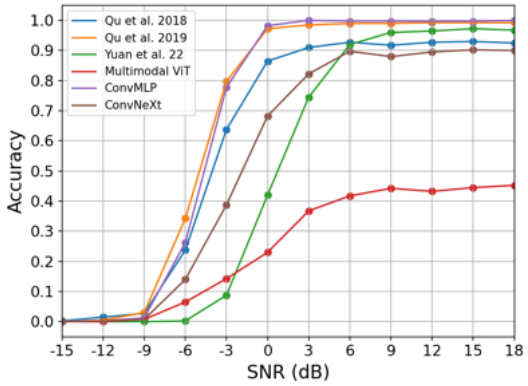
(d) P2.



(e) P3.



(f) P4.



(g) LFM-BPSK.

Figure 5.5: Accuracy of single-component samples (with one label) per class as a function of SNR over the testing dataset, phase modulation types.

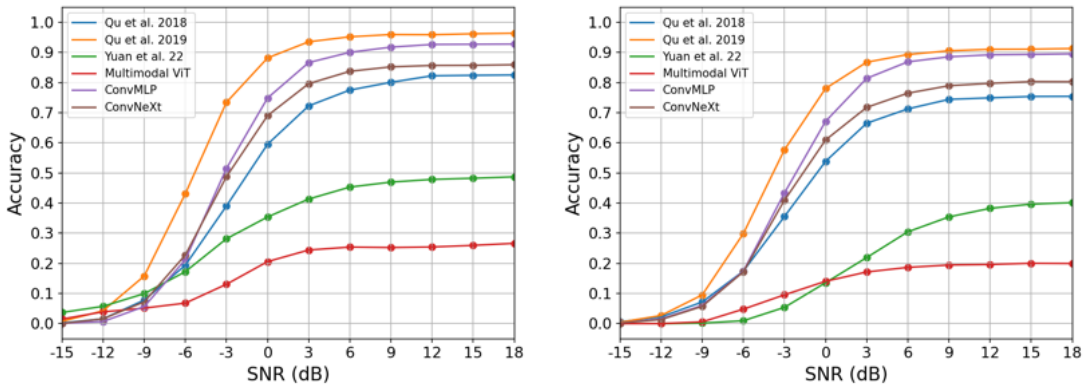
It is immediate to observe that every convolutional model essentially has no problems classifying frequency modulated signals with one component. Expectedly, simple modulation types such as NM are very easily classified, with CNN models having perfect accuracy with an SNR above 0 dB. The rest of the modulation types are not classified so well, yet we still obtain satisfactory results. This set of classes is not free of outliers, however; SFM seems to only be properly classified at a higher SNR threshold than other modulation types.

Interestingly, we see that the CNN+ViT hybrid attains acceptable performance under most of these situations, meaning it did seem to grasp the right features in order to classify these signals, albeit having lower performance than its CNN-based counterparts more often than not, especially at lower SNRs. After all, frequency modulations are more or less distinguishable if only exploring the frequency spectrum, without temporal dependence. Some classes where we can observe this behaviour include SFM, EQFM, and LFM, where it even surpasses CNNs, however this is the only occasion where it seems to take the lead. However, DLFM and MLFM seem to suffer, probably because it is harder to tell apart when frequency decreases or increases in time if only the Fourier transform is used. Discrete frequency modulation types such as BFSK and QFSK are easily recognized by almost every model, which is not very surprising, as they are easy to recognize both with TFIs and Fourier transforms, given that each symbol has a constant frequency, which results in well-defined peaks in the spectrum. Our proposal does not seem to attain similar performance in these cases, likely due to the cause we discussed previously, with frequency and time domains dominating over time-frequency.

Now, we continue with phase modulation types in figure 5.5. Here, we can see some striking results on a few of these modulation types, explaining the poor performance in the single-component accuracy over all classes in figure 5.3(a). Our multimodal ViT proposal did not classify any signals with modulation type Frank, P1, or P2 correctly. This might be due to problems with the model itself, or due to bugs in our implementation, as this failure rate for these classes seems to be too regular. The CNN+ViT hybrid did manage to classify signals as Frank correctly, however it suffered identically to our proposal in P1, P2 and P4. A possible explanation is that phase modulations such as these, when they have codes which are short, usually have a small bandwidth, which hinders distinguishability. As the spectrum changes in time, the spectral contents can be recognized properly if observing the signal in the time-frequency domain, as we saw in figure 2.2. However, the CNN+ViT handles the pure spectrum, and as such, it is likely that integrating the spectrum over all time is inducing confusion in the model.

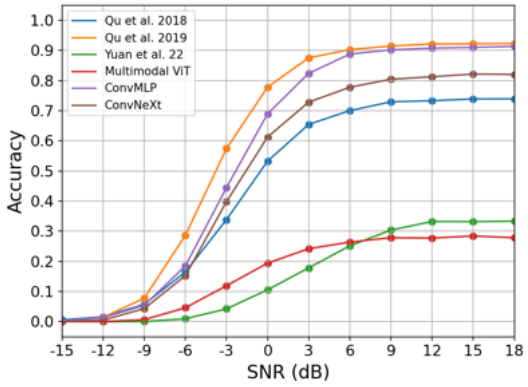
The last figures we will look at contain the dual-component accuracy of combinations of classes which have a given class. That is, for every class, there are 15 combinations of it with every other class, so we have compiled accuracy metrics for every combination of two classes containing this one class. So, for LFM, we have LFM-NM, LFM-LFM, LFM-DLFM, LFM-MLFM... and on, then we took the accuracy of each of these combinations and averaged it for every SNR value. We detail this in figure 5.6 in the case of frequency modulation types and in figure 5.7 in the case of phase modulation types. Both of these figures are, as expected, similar to figure 5.3(b), with accuracies being lower in every SNR value, and with a larger separation among models, as well as the lower performance attained by transformer-based models. We also see similar patterns between figures 5.5 and 5.7, where transformers have abnormally low performance in some phase modulations, as well as the lower performance when classifying SFM, DLFM and MLFM in figures 5.4 and 5.6. Accuracies in dual-component samples are not quite zero in the latter figure probably because these combinations of classes containing phase modulations also contain other modulations which are easier to classify. One of the other noteworthy feature of these figures is that they highlight that, depending on the classes, the hierarchy of model performance seen before does not quite seem to change, but some models have accuracies which approach those of the model that is immediately worse, such as ConvNeXt approaching the VGG-based model in figure 5.6(f), or the Inception-based model matching ConvMLP in figure 5.6(d). What is a bit surprising perhaps is that accuracy does not seem to be affected much when classifying polyphase modulations, so there might not be so much confusion with continuous frequency modulations after all, however we require different metrics to say this with certainty.

Curiously, in some cases, our proposal outperforms the transformer hybrid at lower SNR, to then perform worse at higher SNR in some cases, such as figure 5.6(c). One might think that some of these observations might be attributable to statistical fluctuations due to a low number of samples, but this isn't the case. We have $15 \cdot 15 = 225$ samples per every combination of two modulation types, however, in each of these figures, we are looking at an average over all the other modulation types, so really, we have $225 \cdot 15 = 3375$ samples per SNR level. Compare this to how many of these samples we have in the single-component case, which is 400, and we see it's very much the opposite, we have a lot more samples in this case.

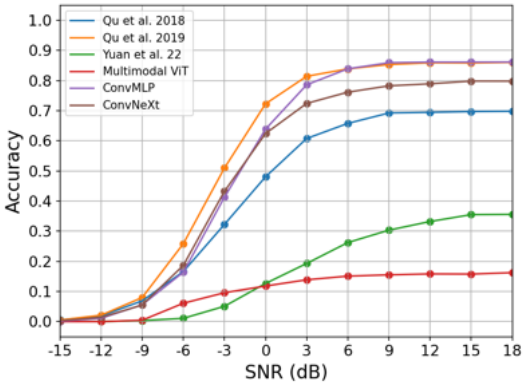


(a) No modulation.

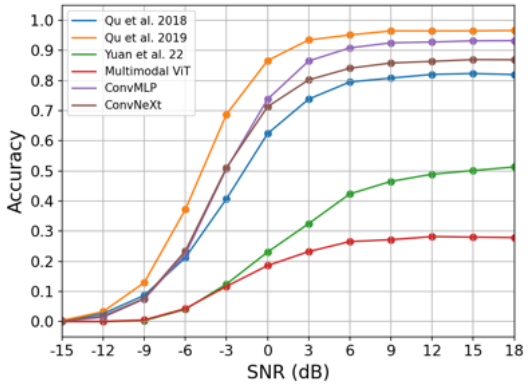
(b) LFM.



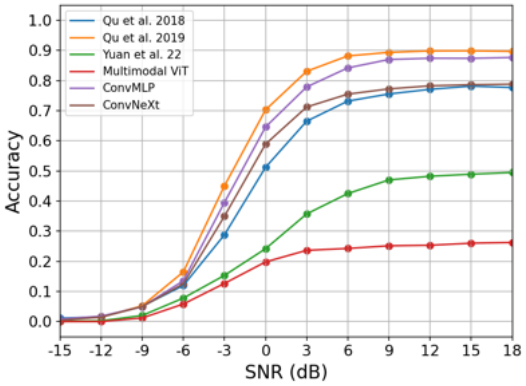
(c) DLFM.



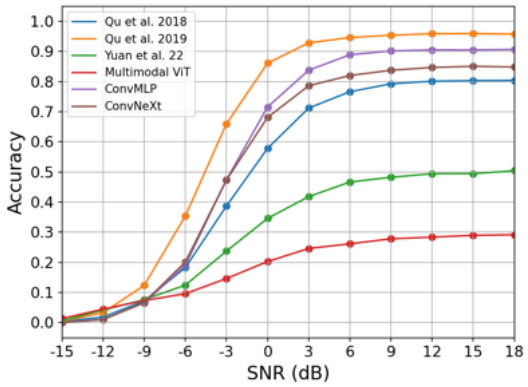
(d) MLFM.



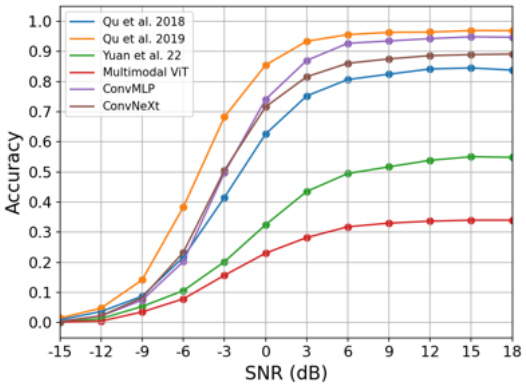
(e) EQFM.



(f) SFM.

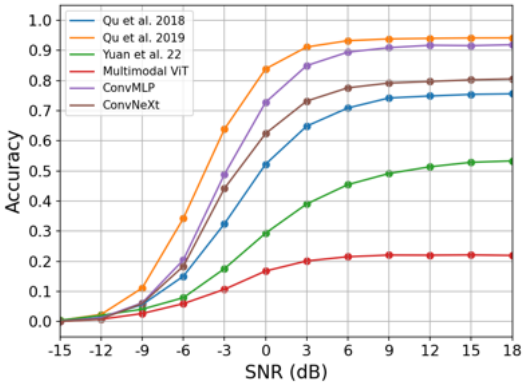


(g) BFSK.

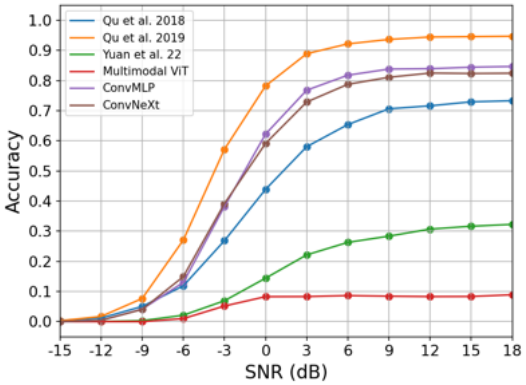


(h) QFSK.

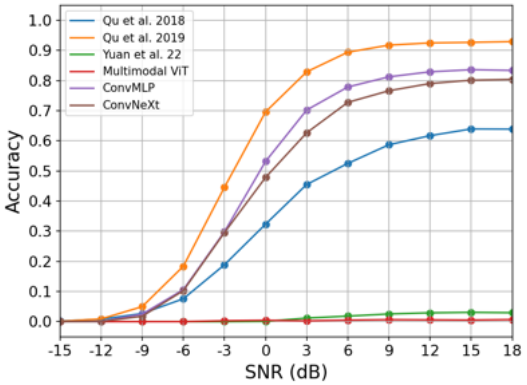
Figure 5.6: Accuracy of dual-component samples (with two labels) per class as a function of SNR over the testing dataset, frequency modulation types.



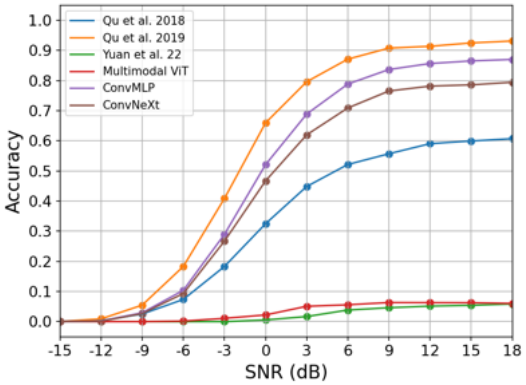
(a) BPSK.



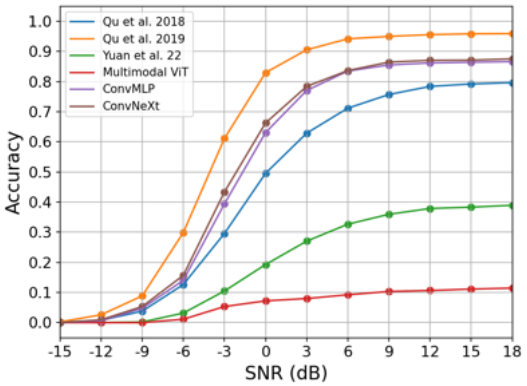
(b) Frank.



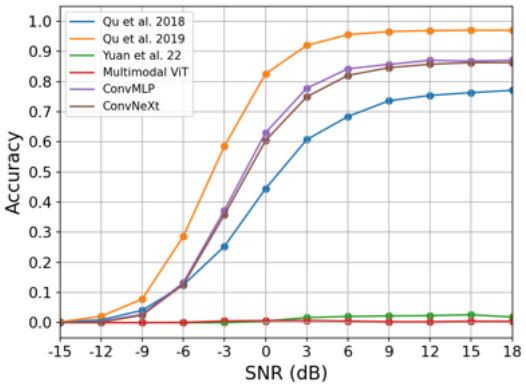
(c) P1.



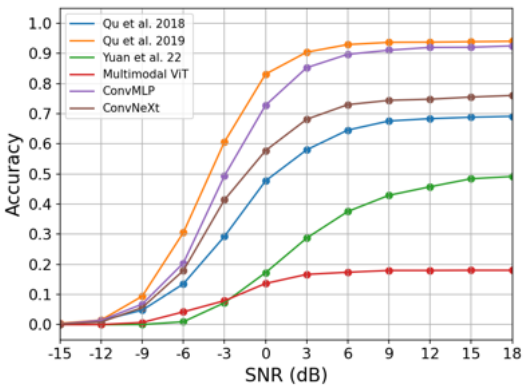
(d) P2.



(e) P3.



(f) P4.



(g) LFM-BPSK.

Figure 5.7: Accuracy of dual-component samples (with two labels) per class as a function of SNR over the testing dataset, phase modulation types.

Chapter 6

Conclusions and future work

In this work, we have reviewed the performance of several models made for pulsed radar signal classification, as well as newer models implemented for the first time to carry out this task, in order to compare them to a novel multimodal architecture we introduced, having trained all of them from scratch, and with few modifications. However, despite following training procedures seen in literature, and making extensive tests, it was not possible to make transformer-based architectures to learn how to classify in every circumstance properly. The convolutional and transformer hybrid we trained performed better than our multimodal proposal in the single-component case, however both of them were shown to have subpar performance classifying phase modulation types and especially classifying dual-component signals. In contrast, every convolution-based model we trained managed to classify signals correctly much more often, having looked at four different architectures, some established, some more recent.

Surprisingly, the Inception-based architecture managed to edge ahead of every other CNN, including the models which are state-of-the-art on ImageNet. These results pose some interesting questions going forward. On one hand, we were assuming, before training models ourselves, that models at this scale (number of layers and/or parameters) would perform follow a hierarchy similar to what is seen when scaling them up and training them with bigger datasets such as ImageNet. However, this has been far from the case; these results point to models residing at this small scale having notably different behaviour than their larger, more general counterparts. We thus wonder if there is another model out there that manages to outperform even this Inception CNN, and believe it would be beneficial to explore the wider range of models available in all of computer vision literature to find an even better performing candidate. We have not attempted any special training regimes or more advanced preprocessing, either, so exploring the effectiveness of these is subject to study as well. Aside from all this, we also believe that these results have much to do with the nature of this dataset, given its relative lack of features to extract and emphasis on rejecting noise, so we hope to keep exploring the unique interaction between these models (especially transformers) and data, using these results as a tentative first foray by our part.

Future work should encompass a variety of improvements within and around the subject matter of this thesis. One of these improvements is the inclusion of further data augmentation. We require obtaining the time when a pulse starts and when it ends, in order to window the signal and feed it to these models; however, methods which retrieve this information are almost never perfect at lower SNR. As such, shifting the signal in time so that it only occupies a portion of the input to the model is another cheap and realistic data augmentation procedure that nevertheless makes a model more robust and capable. Pulses also vary in duration often, not just between radar systems, but also the same radar might have different pulse widths between pulses, so varying this parameter is another consideration to think about.

Simulating multipath fading, as a consequence of imperfect transmission, rather than imperfect processing at the reciever as before, is also a potentially important data augmentation procedure. A signal may travel through different paths and end up at the reciever, where the difference in phase may induce constructive or destructive interference which impedes signal recognition. This is, however, more difficult to implement. Perhaps this is why it was not included in any of the literature studied in anticipation of this work, or maybe authors therein were not concerned about this effect, which may not occur as much in open fields, as opposed to urban, built-up areas.

Another improvement would be to increase the number of classes that the model outputs, by adding polytime (T1 through T4) and Costas modulations, known to be used in radar applications, as well as adding others used in communication and navigation applications, in order to help discerning the nature of an emitter. There is also the addition of classification of both modulation type and order of code, the possibility to classify signals with more than two components, testing the model with real radar detection data, and the improvement of model execution speed in order for its inclusion in real-time classification tasks. Ablation tests are also in order to determine the optimal set of hyperparameters to use in this model as well.

But perhaps, the most important concern of all, as we hinted above, is that these models are not end-to-end. They classify pulses, but a pulse detection scheme is required to be applied first to the time series representing the signal in order to extract the pulsed signals themselves. As such, the coupling of these models with more advanced pulse detection schemes, or even their integration with them, would be one of the more interesting research avenues.

Bibliography

[22] *Python 3.9.16 documentation*. Python Software Foundation. Dec. 27, 2022. URL: https://docs.python.org/release/3.9.16/ (visited on 06/14/2022).

[Aky+18] Fatih Cagatay Akyon et al. "Classification of intra-pulse modulation of radar signals by feature fusion based convolutional neural networks". In: *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE. 2018, pp. 2290–2294. DOI: 10.23919/EUSIPCO.2018.8553176.

[Bro+20] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: (2020). DOI: 10.48550/ARXIV.2005.14165.

[Coh89] Leon Cohen. "Time-frequency distributions-a review". In: *Proceedings of the IEEE* 77.7 (1989), pp. 941–981. DOI: 10.1109/5.30749.

[Coh92] Leon Cohen. "What is a Multicomponent Signal?" In: *ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 5. 1992, 113–116 vol.5. DOI: 10.1109/ICASSP.1992.226645.

[Den+09] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pp. 248–255.

[Dos+20] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). DOI: 10.48550/ARXIV.2010.11929.

[FLC13] Zhipeng Feng, Ming Liang, and Fulei Chu. "Recent advances in time–frequency analysis methods for machinery fault diagnosis: A review with application examples". In: *Mechanical Systems and Signal Processing* 38.1 (2013), pp. 165–205. DOI: 10.1016/j.ymssp.2013.01.017.

[Gon+21] Chengyue Gong et al. *Vision Transformers with Patch Diversification*. 2021. arXiv: 2104.12753 [cs.CV].

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hof+22] Jordan Hoffmann et al. "Training Compute-Optimal Large Language Models". In: (2022). DOI: 10.48550/ARXIV.2203.15556.

[Hou+23] Changbo Hou et al. "The recognition of multi-components signals based on semantic segmentation". In: *Wireless Networks* 29.1 (2023), pp. 147–160. DOI: 10.1007/s11276-022-03086-7.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[Li+20] Gao Li et al. "LSTM-based Frequency Hopping Sequence Prediction". In: *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE. 2020, pp. 472–477. DOI: 10.1109/WCSP49889.2020.9299717.

[Li+21] Jiachen Li et al. *ConvMLP: Hierarchical Convolutional MLPs for Vision*. 2021. arXiv: 2109.04454 [cs.CV].

[Liu+20] Liyuan Liu et al. *Understanding the Difficulty of Training Transformers*. 2020. arXiv: 2004.08249 [cs.LG].

[Liu+22] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV].

[LLH20] Xueqiong Li, Zhangmeng Liu, and Zhitao Huang. "Attention-Based Radar PRI Modulation Recognition With Recurrent Neural Networks". In: *IEEE Access* 8 (2020), pp. 57426–57436. DOI: 10.1109/ACCESS.2020.2982654.

[Nor19] Eric Norgren. "Pulse repetition interval modulation classification using machine learning". 2019.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2826–2830.

[QMD18] Zhiyu Qu, Xiaojie Mao, and Zhian Deng. "Radar signal intra-pulse modulation recognition based on convolutional neural network". In: *IEEE Access* 6 (2018), pp. 43874–43884. DOI: 10.1109/ACCESS.2018.2864347.

[Qu+19a] Qizhe Qu et al. "Radar signal recognition based on squeeze-and-excitation networks". In: *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*. IEEE. 2019, pp. 1–5. DOI: 10.1109/ICSIDP47821.2019.9173345.

[Qu+19b] Zhiyu Qu et al. "Radar signal intra-pulse modulation recognition based on convolutional denoising autoencoder and deep convolutional neural network". In: *IEEE Access* 7 (2019), pp. 112339–112347. DOI: 10.1109/ACCESS.2019.2935247.

[Ste+22] Andreas Steiner et al. *How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers*. 2022. arXiv: 2106.10270 [cs.CV].

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[Vas+17] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). DOI: 10.48550/ARXIV.1706.03762.

[Wan+17] Xuebao Wang et al. "Radar emitter recognition based on the short time Fourier transform and convolutional neural networks". In: *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE. 2017, pp. 1–5. DOI: 10.1109/CISP-BMEI.2017.8302111.

[Xie+17] Saining Xie et al. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[XLH22] Wenshi Xiao, Zhongqiang Luo, and Qian Hu. "A Review of Research on Signal Modulation Recognition Based on Deep Learning". In: *Electronics* 11.17 (2022), p. 2764. DOI: 10.3390/electronics11172764.

[YLW22] Shibo Yuan, Peng Li, and Bin Wu. "Towards Single-Component and Dual-Component Radar Emitter Signal Intra-Pulse Modulation Classification Based on Convolutional Neural Network and Transformer". In: *Remote Sensing* 14.15 (2022), p. 3690. DOI: 10.3390/rs14153690.

[ZDG17] Ming Zhang, Ming Diao, and Limin Guo. "Convolutional Neural Networks for Automatic Cognitive Radio Waveform Recognition". In: *IEEE Access* 5 (2017), pp. 11074–11082. DOI: 10.1109/ACCESS.2017.2716191.