Deep Learning Models for Realistic Multicomponent Signal Modulation Classification

Agustín M. Galante-Cerviño, Alberto Martínez-Fernández, Adrián Colomer, Valery Naranjo, and Carlos García-Meca

***Abstract*—In this work, we study the application of three recent computer vision architectures to the classification of the modulation type of single- and dual-component signals, making emphasis on their usage in a realistic context by simultaneously considering a wide variety of modulations while varying the number of components. In order to do so, we first generate synthetic signal reception data of 15 modulation types, used for training and testing small variants of these models, chosen such that throughput is maximized and latency minimized, since we consider the context of a time-sensitive application. Given enough training compute, and in the single-component case, all convolutional models obtain an accuracy of 95% or more when signal-to-noise ratio (SNR) is at least 0 dB, and one variant obtains 90% accuracy at -3 dB. In the dual-component case, convolutional models manage upwards of 95% when SNR is at least 12 dB, and more than 90% when SNR is at least 6 dB. Finally, we also measure their throughput and latency as a function of batch size (important parameters for applications such as radar and communication systems), with a convolutional model variant yielding the highest throughput at 14432 samples per second. Based on the obtained results, our work paves the way towards the classification of modern complex modulation schemes and provides selection rules for the most appropriate algorithm depending on the performance feature to be optimized (such as throughput and size).**

***Impact Statement*—Classification of a pulsed signal's modulation type, also called Automatic Modulation Classification (AMC), is crucial in civilian and military contexts. In civilian applications like cognitive radio, AMC enables efficient spectrum usage and interference management. In military scenarios, it supports signals intelligence by identifying communication schemes used by adversaries. AMC faces significant challenges, including handling noise (e.g., thermal noise, frequency mismatches), distinguishing among diverse modulation types and components, and countering advanced spoofing and jamming methods. These challenges justify ongoing research into more effective analysis techniques, driving the adoption of deep learning models for their ability to manage complexity and adapt to diverse conditions. This work evaluates the viability of such models by measuring their throughput and performance under adverse conditions, offering insights into their practical applicability for advancing signal analysis.**

***Index Terms*—Automatic modulation classification, convolutional neural networks, deep learning, signal processing, synthetic data, transformers.**

Manuscript received June 13, 2024; revised December 29, 2024; accepted February 18, 2025. Date of publication MONTH 00, 2025; date of current version MONTH 00, 2025. This work has been supported by the Spanish CDTI through Project CUCO, belonging to the "Misiones Ciencia e Innovación" program, under Grant MIG-20211005.

Agustín M. Galante-Cerviño, and Alberto Martínez-Fernández are with DAS Photonics S.L., Valencia 46022, Spain (e-mail: agalante@dasphotonics.com; albertomartinez@dasphotonics.com).

Adrián Colomer and Valery Naranjo are with the Instituto de Investigación e Innovación en Bioingeniería, Universitat Politècnica de València, Valencia 46022, Spain (e-mail: adcogra@i3b.upv.es; vnaranjo@dcom.upv.es).

Carlos García-Meca is with Monodon, Navantia, S.A., S.M.E., Madrid, 28005, Spain (e-mail: cargarm2@upv.es).

This paragraph will include the Associate Editor who handled your paper.

I. Introduction

THE accuracy of signal processing has been demonstrated to be considerably improved with respect to earlier approaches by making use of deep learning models based mostly on convolutional neural networks (CNNs) [1]–[4], recurrent neural networks (RNNs) [5], [6], and, more recently, transformers [7], [8]. Transformers are a relatively new neural network architecture based on self-attention [9] which has made massive strides in the field of natural language processing [10], speech recognition and synthesis [33], [37], becoming the new state of the art. More recently, a 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 [11]. This spurred further research, yielding several variations of transformer- and convolution-based models, examples of the latter being ConvMLP [12] and ConvNeXt [13].

To expand on the recent advances in signal modulation type classification, we wished to explore ways to best classify signals' modulation types in more realistic contexts, such as having to distinguish between a wide variety of modulation types and correctly classifying each component of a signal simultaneously. Even though multicomponent signals have rarely been classified in the literature [14], this consideration is gaining importance in newer schemes due to its use in recent radar systems [8], [15]. Moreover, the number of modulations considered in previous multicomponent studies was limited to 8 [8]. Here, we consider up to 15 different modulation types, which is more similar to realistic scenarios, which are crowded with a large variety of radar systems.

Furthermore, our focus lies on optimizing the throughput and minimizing latency of the models under consideration for this particular task. If one is in a time-sensitive scenario, it is of paramount importance to not only use a model with high accuracy, but also carefully consider the architecture of the model such that throughput is maximized and latency is minimized. The vast majority of the models explored for this task in literature have an adequate size for the problem at hand [14]. However, reducing the number of trainable parameters does not always correlate to lower latency and inversely correlate to higher throughput, since models may not optimally profit from the parallel nature of GPUs if they are sufficiently deep (too many layers) and narrow (too little operations in parallel; few filters, small kernel size...). The

architectures themselves explored in literature might not be adequate in this aspect either, as older architectures not only have more trainable parameters on average, but also have more limited throughput and latency.

To this end, we study the performance of architectures ConvMLP, ConvNeXt and ViT when applied to this task in detail, having chosen these models on the basis that they improve on more mature CNNs, like ResNet or EfficientNet, when used on other common domains such as photo classification, when trained on ImageNet [18], expecting their superior performance to translate to this problem, which may also be necessary as we are considering a more complex scenario than previous approaches to modulation classification. Said mature CNNs, some of which have been used in this problem, such as LeNet [2] or Inception [3] might not deliver satisfactory performance, especially if we are looking to increase the number of modulations, of components, and maximize model throughput all at the same time, as these newer models we chose are much more efficient [11]–[13]. Few proposals explore the viability of transformers [7], [8], ConvMLP [30] and ConvNeXt [31], [32] in this problem. Note that we have not consider RNNs either due to the sequential nature needed to process the input data, which might lower throughput substantially when compared to CNNs.

Transformers and convolutions are each known to possess unique advantages. It is well-known that convolutions introduce inductive biases, such as equivariance under translation, easing convergence to a solution when training, though these potentially restrict the representational power of the model. ViTs are not subject to these biases as strongly, since the multi-head self-attention layer in transformer encoders is global. We wish to further exploit the flexibility of ViTs by employing the original architecture, without any convolutions, unlike [7], [8]. This comes at the cost of needing to train ViTs on larger datasets to be competitive with CNNs [11]. We intend to train our models using a sizeable number of samples, more than that of datasets employed in other approaches [2], [3], [8], something we do in order to circumvent this disadvantage.

II. Signal Characteristics

In this work, we have made use of a synthetic signal reception dataset, which enables the significant number of samples we need for training ViTs. In order to ensure that the synthetic data we use is realistic, we need to make it as varied as possible. This is done by including many different types of modulations, varying said modulations' signal parameters, including signals with multiple components, and by adding impediments to said signals.

A. Parameters

Within a pulse, the frequency, amplitude and phase of the signal could each change with time, either as part of pulse compression if emitted by radars, or to encode information in communication applications. Such signals are often represented as complex, since this eases mathematical manipulations, as well as the fact that common sampling schemes such as I/Q sampling directly output complex signals. Signals are usually modeled in the following manner;

$$s(t) = A(t)\mathrm{e}^{\mathrm{j}\varphi(t)}, \tag{1}$$

with $A(t)$ being the instantaneous amplitude, where amplitude modulation is held, and $\varphi(t)$ is the instantaneous phase, which is where frequency and/or phase modulation is contained. The modulation types used here 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: Bandwidth. Frequency range of the signal contents.
- f_0: Frequency offset. It lies at the center of the bandwidth. Equal to $f_0 = f_{min} + \Delta f/2$
- r: Fraction of PW where each "subpulse" is. MLFM only.
- f_{SFM}: Frequency at which $\varphi'(t)$ varies. SFM only.
- ϕ_{SFM}: Initial phase of the variation of $\varphi'(t)$. SFM only.
- N: Order of a given code, its length in symbols.

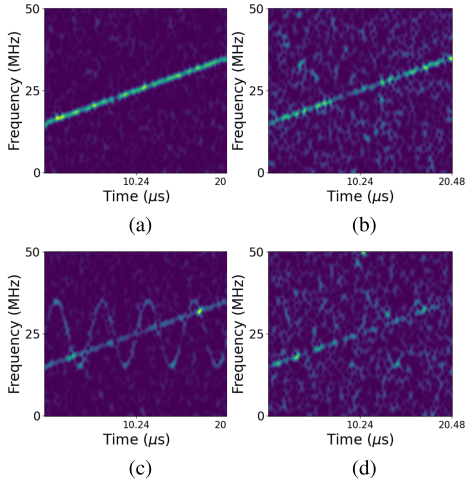

Fig. 1. Spectrograms of a single-component LFM signal with (a) an SNR of 0 dB (b) SNR of -6 dB, (c) a double-component LFM and SFM signal having an SNR of 0 dB and (d) same signal with an SNR of -6 dB.

B. Impediments and multiple components

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

- **Johnson-Nyquist or thermal noise**
  Any electronic device at a non-zero temperature experiences a random movement of its electrons, which induces a random voltage, modeled as additive white Gaussian noise (AWGN) with zero mean. Such noise is quantified using the signal-to-noise ratio (SNR):

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

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

TABLE I
PARAMETERS APPLIED TO SAMPLES OF EACH MODULATION TYPE USED IN THIS WORK.

Modulation type	**Parameters**	**Notes**
NM	$f_0 \sim U(0.1f_s, 0.4f_s)$	
LFM	$f_0 \sim U(0.01f_s, 0.45f_s)$ $\Delta f \sim U(0.05f_s, 0.4f_s)$	Frequency either increases or decreases with time, chosen with equal probability. This is the case for the rest of the mentioned parameters that take discrete values.
DLFM	$f_0 \sim U(0.01f_s, 0.4f_s)$ $\Delta f \sim U(0.05f_s, 0.35f_s)$	
MLFM	$f_0 \sim U(0.15f_s, 0.5f_s)$ $\Delta f \sim U(0.1f_s, 0.35f_s)$ $r \sim U(0.3, 0.7)$	
EQFM	$f_{min} \sim U(0.01f_s, 0.4f_s)$ $\Delta f \sim U(0.05f_s, 0.3f_s)$	
SFM	$f_{min} \sim U(0.01f_s, 0.15f_s)$ $\Delta f \sim U(0.05f_s, 0.35f_s)$ $f_{\mathrm{SFM}} \sim U(\frac{0.75}{PW}, \frac{10}{PW})$ $\phi_{\mathrm{SFM}} \sim U(0, 2\pi)$	
BFSK	$f_1, f_2 \sim U(0.05f_s, 0.45f_s)$ $f_0 = (f_1 + f_2)/2$ $\Delta f = \lvert f_1 - f_2\rvert/2$ $N \in \{5, 7, 11, 13\}$	f_1 and f_2 are sampled independently, but resampling is done in order to keep an 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.
QFSK	$f_i \sim U(0.05f_s, 0.45f_s)$ where $i \in \{1, 2, 3, 4\}$	The absolute difference between these frequencies fulfills the condition $\lvert f_i - f_j\rvert > 0.005\ f_s$, $\forall i, j \in \{1, 2, 3, 4\}$ if $i \neq j$. We use a 16-bit ($N = 4$) Frank code, and is either normal or inverted.
BPSK	$f_0 \sim U(0.05f_s, 0.45f_s)$ $N \in \{5, 7, 11, 13\}$	Each signal employs a normal or an inverted Barker code.
Frank	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	
P1	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	This code is used normal and inverted, which resembles an ascending or a descending chirp respectively.
P2	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	
P3	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	
P4	$f_0 \sim U(0.1f_s, 0.4f_s)$ $N \in \{6, 7, 8\}$	
LFM-BPSK	$f_{min} \sim U(0.05f_s, 0.45f_s)$ $\Delta f \sim U(0.05f_s, 0.4f_s)$ $N \in \{5, 7, 11, 13\}$	Each signal employs a normal or an inverted Barker code.

- **Frequency offset**
  A signal is converted to the digital domain after having shifted its frequency downwards. The new center frequency may not match the emitter's carrier frequency, so the signal's bandwidth could be centered anywhere. This is due to a mismatch between the frequency of the oscillator and the carrier, given by Doppler shift, or due to the reciever not knowing the carrier frequency of the emitter. This is modeled by generating signals with a random frequency offset.

Once these impediments are applied onto the signal, the time series representing a single-component signal interception can be cast as:

$$x(t) = s(t) + n(t) = A(t)\mathrm{e}^{\mathrm{j}(2\pi f_0 t + \varphi(t))} + \mathcal{N}(0, \sigma_n), \quad (3)$$

where $n(t) = \mathcal{N}(0, \sigma_n)$ is the AWGN, and f_0 is the frequency offset. This noise is complex, meaning that both the real and imaginary components of the noise follow an independent distribution, with the same value of σ_n.

A signal containing multiple components means that it is made up of the sum of individual signals of the form of (1), each with its own modulation type. In our case, we are considering signals with one and two components. Each recieved signal we will consider in this work has the form

$$\begin{aligned} x(t) &= s_1(t) + s_2(t) + \mathcal{N}(0, \sigma_n) \\ &= A_1(t)\mathrm{e}^{\mathrm{j}\varphi_1(t)} + A_2(t)\mathrm{e}^{\mathrm{j}\varphi_2(t)} + \mathcal{N}(0, \sigma_n). \end{aligned} \quad (4)$$

If the signal is sampled on a single channel, it is simply the real part of (4), this being our case. We show examples of the signals we used following this description in Fig. 1.

III. DATASET

Each of the modulation types shown in Fig. 2 makes up one class, so the dataset contains 15 classes. Every sample is made up of a real signal that has 2048 data points, with the signal occupying every point. Each of these points is represented by one 32-bit floating-point number. This corresponds, for example, to a sampling frequency of 100 MHz and a pulse width of 20.48 μs. Additionally, every discrete modulation (FSK and PSK) will contain only one instance of their code; it is not repeated within a pulse more than once. In the case of single-component signals, the amplitude will be equal to 1.

In the training set, 800 combinations of signal parameters are chosen, each sampled from a uniform distribution as indicated in Table I, being shown as $U(\cdot, \cdot)$. Signals are impeded using AWGN, with the 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. The training dataset thus contains $800 \cdot 11 \cdot 15 = 132000$ single-component signals. A further 200 parameter combinations are sampled for validation and 400 combinations for testing, so there are $200 \cdot 11 \cdot 15 = 33000$ single-component signals in the validation set. As for the testing set, we used more noise levels, spanning between -21 dB and 18 dB, making up 14 levels. That means we have $400 \cdot 14 \cdot 15 = 84000$ single-component signals in the testing set.

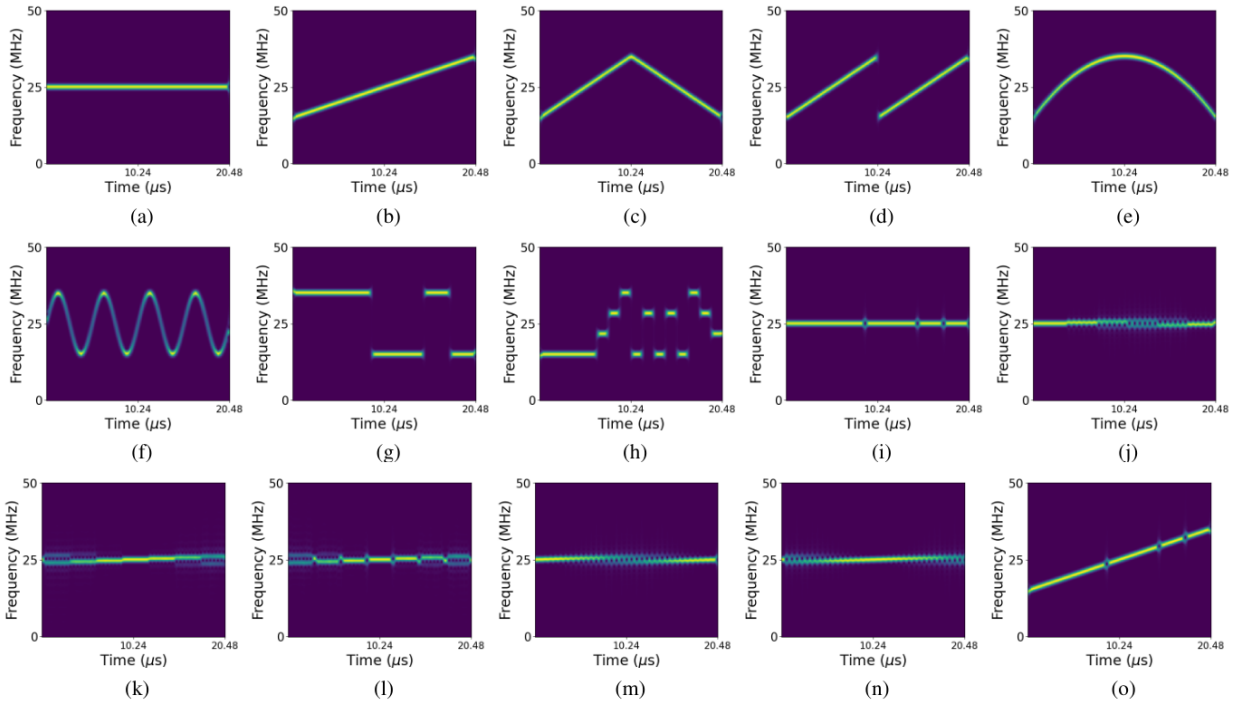

Fig. 2. Spectrograms of modulation types used in this work. These signals have $f_0 = 25$ MHz, $\Delta f = 20$ MHz. (a) NM, (b) LFM, (c) DLFM, (d) MLFM, (e) EQFM, (f) SFM, (g) BFSK, (h) QFSK, (i) BPSK, (j) Frank, (k) P1, (l) P2, (m) P3, (n) P4, (o) LFM-BPSK

In order to obtain dual-component signals, some of these existing single-component signals are combined through addition. Each of their amplitudes is constant with time, so $\mathrm{d}A_1(t)/\mathrm{d}t = \mathrm{d}A_2(t)/\mathrm{d}t = 0$. However, the amplitude of each component is not equal; we sample a parameter, α from a uniform distribution between 0.5 and 0.25 in every parameter combination, such that $A_1(t) = \alpha$ and $A_2(t) = 1 - \alpha$ in (4). After the clean signal, $s_1(t) + s_2(t)$, has been obtained, the noise power, σ_n^2, is calculated. This is done by first specifying the desired SNR. Then the clean signal power, σ_s^2, is calculated as the power of the clean dual-component signal, so we can solve for σ_n^2 in (2) and obtain the distribution in (4).

The number of unique modulation type combinations is $n(n-1)/2 + n$, so for $n = 15$, it is 120. For each of these combinations, we take a certain amount of parameter combinations of one type and mix them with the same amount of combinations from the other type. In the training dataset, we mix 25 parameter combinations. In this case, there are $25 \cdot 25 \cdot 11 \cdot 120 = 825000$ dual-component signals in the training dataset. Similarly, there are $10 \cdot 10 \cdot 11 \cdot 120 = 132000$ such signals in the validation set and $15 \cdot 15 \cdot 14 \cdot 120 = 378000$ in the testing set. Training, validation and testing sets have 132000 + 825000 = 957000, 33000 + 132000 = 165000, and 84000 + 378000 = 462000 signals in total, respectively.

IV. Automatic Identification of Signal Modulations

A. Preprocessing

Often, the signal is processed in order to extract information about its frequency contents as they change in time. The Fourier transform does not express the location in time of spectral content, so time-frequency distributions (TFDs) [16] are what should be used given the nature of this data. Not only that, TFDs usually have the effect of separating the power of the clean signal from the noise. This leads to better distinguishability of the contents of the clean signal itself, as can be seen in Fig. 1, boosting recognizability by the predictive model. Obtaining the TFD is thus a common preprocessing step in the literature [14].

A TFD is a complex-valued function depending on both time and frequency of a given signal. Its modulus squared, also called the time-frequency image (TFI) or spectrogram, represents the instantaneous energy of the signal. The most widely used choice [4], [14], [17] is the short-time Fourier transform (STFT). We use it since it is a TFD whose myriad implementations are optimized thoroughly, and its computational complexity is linear, instead of quadratic as other TFDs like Cohen's class distributions. Besides, we wanted to provide a baseline for the performance of these networks. The discrete version is defined as

$$F_s[m, f] = \sum_{n=0}^{N-1} s[n]\, w[n-m]\, \mathrm{e}^{-\mathrm{j}2\pi f n/N}. \tag{5}$$

where N is the number of samples. It is a Fourier transform with an additional term multiplying the summand such that when $|n| > N_w/2$, $w[n-m] = 0$, transforming in a limited region. $w[n]$ is called the window function, and N_w is the length of the window function. In the following, the STFT is what we use in order to obtain the TFI of the signal, which is the input to the models seen here.

Obtaining a TFI from a real signal would contain negative frequency components that mirror positive frequency ones,

since the modulus of the signal has a reflection symmetry about the frequency origin. In order to omit redundant information from input data, we shall remove the part of the spectrum belonging to negative frequency components.

B. Predictive models

As we mentioned before, we trained and evaluated three recently developed computer vision architectures; Vision Transformer [11], ConvMLP [12] and ConvNeXt [13]. When developing a model for this problem, speed is a very important consideration. Radar and communication systems emit pulses very often; some radars have a pulse repetition frequency of up to 30 kHz. If we have multiple beams and/or multiple emitters, a reciever might be processing about 10^5 pulses/second.

Therefore, we focused on designing models that are as fast as possible while maintaining acceptable performance, opting to make them as small as the problem allows. This is feasible because the features to extract from the TFIs are few, as can be seen in Fig. 2, given the lower data complexity compared to domains like medical imaging. The primary challenge, as shown in Fig. 1, is overcoming the impact of noise on the signal, which obscures it. We will evaluate model performance against this challenge in Section V.

We have considered models which have the least depth possible, reducing serial computation, as we wish to make good use of the parallel nature of GPUs. We have thus explored model variants with different widths among them, always keeping depth to a minimum. In the following, we shall briefly explain each architecture and the model hyperparameters chosen for the variants we employed;

1) ViT: These models split up the input into ”patches”, i.e. non-overlapping windows, before flattening them and applying a fully connected (FC) layer to each of them, the output of which is the patch embedding. A learned patch is concatenated to this sequence, called a classification token. A learned positional embedding is added to each patch, to then feed them to a series of transformer encoders [9], detailed in Fig. 3(a). After a number of encoders E_T, the class. token is selected and fed to an FC layer. We picked a patch size of 16×16 [11], [19], a patch embedding size of 128, and 4 attention heads, equal for all variants. The rest of the hyperparameters were used to modulate the width through the size of the hidden layer in the 2-layer MLP in each encoder, and their depth through the number of encoders used. We used three variants of this model;

- ViT-femto: $E_T = 4$, hidden size of 256
- ViT-pico: $E_T = 4$, hidden size of 512
- ViT-nano: $E_T = 6$, hidden size of 512

2) ConvMLP: This architecture is made up of a stem, a convolutional stage, and a Conv-MLP stage. The stem is made up of a 3×3 conv., batch norm. and ReLU repeated 3 times, and a max. pooling operation. The convolutional stage has $B_{Conv} = 1$ conv. block and the Conv-MLP stage has $L_{CM} = 3$ levels. In our case, each level has one block, $B_{CM} = \{1, 1, 1\}$. In Fig. 3(b) we can see the structure of the blocks in each part of the network. The downsampling conv.

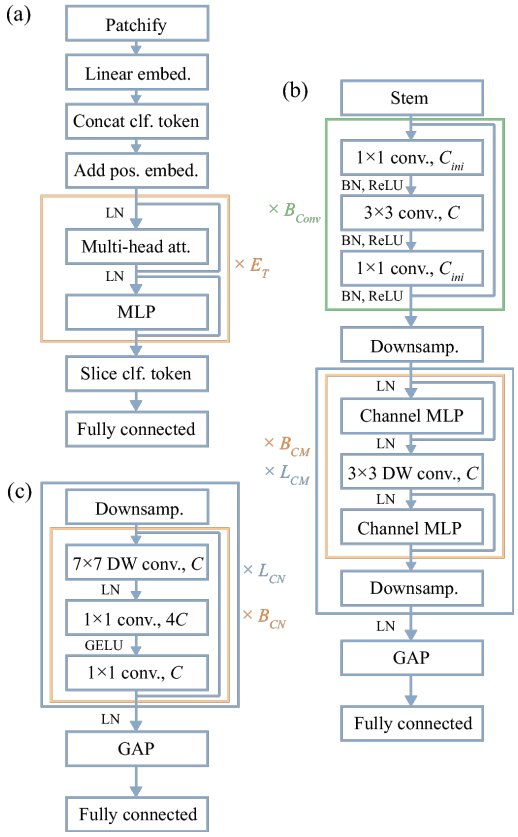

Fig. 3. Deep learning architectures used in this work: (a) Vision Transformer, (b) ConvMLP, (c) ConvNeXt.

has a kernel size of 3, stride of 2 and padding 1. The channel-wise MLP is made up of two layers, with the hidden layer doubling the dimension and a GELU activation. The last layer in a level is a downsamp. conv., though the last level does not downsample, and it has the same value of C as the second to last level. After all of these stages, global average pooling (GAP) is applied, which feeds to an FC layer. We used three ConvMLP variants;

- ConvMLP-atto: $C_{ini} = 16$, $C = \{32, 64, 128\}$
- ConvMLP-zepto: $C_{ini} = 32$, $C = \{48, 64, 96\}$
- ConvMLP-yocto: $C_{ini} = 24$, $C = \{32, 48, 64\}$

3) ConvNeXt: This architecture is made up of $L_{CN} = 4$ levels, each characterized by the number of channels C and the number of blocks B_{CN}, as can be seen in Fig. 3(c). A level starts with a downsampling layer; in the first level, it is a 4×4 conv. with stride 4 and LN, while in the rest, it is an LN and a 2×2 conv. with stride 2. Each level has a number B_{CN} of ConvNeXt blocks [13].

As before, $B_{CN} = \{1, 1, 1, 1\}$. Each block contains a depthwise 7×7 conv., LN, and two 1×1 convs., first increasing and then decreasing channels to the initial amount, then an RC. At the end, GAP and an FC layer are applied. We have picked two ConvNeXt variants;

- ConvNeXt-ronto: $C = \{16, 32, 64, 128\}$
- ConvNeXt-quecto: $C = \{32, 48, 64, 96\}$

C. Implementation

The programming language Python 3.11 is used in order to implement every process. We made use of in-house scripts to generate the dataset employed in this work. The library PyTorch 2.1 [20] is used to implement models, the training and testing procedures, as well as preprocessing. We also use scikit-learn 1.3 [21] to calculate classification metrics, fvcore [22] to obtain models' FLOP (FLoating point OPeration) count and finally, we use Matplotlib 3.8 [23] to plot the results. In contrast to the ViT, we did not write the code needed to implement the CNNs models for this work. Instead, we source these models' definition from each of their GitHub repositories where all the code used in [12] and [13] resided.

As for preprocessing, we use the Hann window function in the STFT, and its length is $N_w = 101$ with a stride of 8. We apply zero-padding to windows such that $N_w = 512$, to obtain a TFI having 512 frequency values, before removing values corresponding to negative frequencies and the phase of the transform, resulting in an image of size 256×256 with one channel. Before passing the TFI to the model, we perform sample-wise min-max normalization, such that every feature in the input is divided by the maximum value of that sample.

We use a mini-batch size of 256 samples, and we train for 75 and 150 epochs, to explore the dependence of performance on training compute. We employ the AdamW optimizer [24], with $\beta_1 = 0.9$ and $\beta_2 = 0.995$ in the case of ConvMLP and ConvNeXt, and $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the ViT. A standard training regime was used, using the signals as inputs and the labels as the targets to which they have to converge during training. Samples in each mini-batch are chosen from the training dataset randomly. Model selection is done by choosing the model that has the best overall accuracy over the validation set out of all the epochs.

Regularization is employed in various forms. All models are trained with a weight decay of 0.05. We also use a stochastic depth rate [25] in ConvMLP and ConvNeXt of 0.2. The ViT has a dropout of 0.2 in transformer encoders and 0.2 in the classifier head.

Furthermore, several data augmentation schemes [34] are used, such as random horizontal and vertical flips of each image with a probability of 0.5, randomly cropping and resizing to the original shape, with a scale between 0.08 and 1, and the ratio of the rectangular crop between 0.75 and 1.25. Finally RandomErase is used [26], with a probability of 0.33, a range of proportion of erased area against input image between 0.02 and 0.33, and a range of aspect ratio of erased area between 0.3 and 3.3. These transformations were used as they are well-known standards, are easy to implement and already available in machine learning frameworks such as the one we used, which eases reproducibility of this work.

We encountered significant disparities between the training process of the ViTs and the two CNNs, the former having proved to be difficult to train. Models took between 14 and 60 hours to train, this figure depends on the number of epochs and the concrete model variant considered. ViTs generally took longer to train, between 40 and 60 hours, due to their higher parameter count. Said models were not able to overfit on the training set at first, having a low overall accuracy on both training and validation sets (∼20%), when using the original Adam optimizer [27]. Making use of AdamW managed a large increase in training set accuracy and a smaller increase in validation accuracy. Regularization was attempted next, in the form of weight decay and dropout, which improved validation accuracy. However, this did not solve the problem completely. What made ViTs generalize well was the introduction of data augmentation schemes, first using random horizontal and vertical flips of each spectrogram, which increased it noticeably, however other smaller noticeable improvements could be eked out by cropping said spectrogram randomly and resizing to the original size, and adding RandomErase.

ConvMLP and ConvNeXt worked well without as much tuning, however. Performance was acceptable to a point without data augmentation, though a noticeable disparity between training and validation accuracy was observed. Introducing the same training hyperparameters as what was used with ViTs improved both accuracies substantially.

In our experiments, we found that data augmentation significantly improves performance, despite having a varied dataset. Initially, we assumed it was unnecessary, as prior studies rarely used it [2], [3], [14], [28]. However, it is essential for ViTs due to less inductive biases compared to CNNs, which results in a larger model search space. This differs from [8], where no augmentation was used, fewer signal parameter combinations (400 vs. 800 single-comp., 400 vs. 625 double-comp.), and less noise levels (8 vs. 11) were tested. Said model however used convolutional layers to tokenize before the transformer, lessening the need for augmentation. Despite this, our attempts to reproduce their results with our dataset failed, reinforcing the necessity of augmentation for ViTs.

Every model uses the binary cross-entropy loss function, and their output layer uses a sigmoid activation function, with a threshold of 0.5 for all classes. The output of every model employed here is a one-hot encoded vector of length 16. The 15 first features correspond each to a modulation type, while the last feature is an auxiliary class indicating if a signal is single-component (0) or dual-component (1). This is because we need to clarify if a signal has both components with the same modulation type.

All computations are carried out on a system with an NVIDIA RTX A6000 as its GPU.

V. Results and Discussion

After having experimented with model and training hyperparameters to obtain the configurations described in Subsections IV-B and IV-C respectively, we benchmarked their performance in a variety of noisy conditions, as well as in the case of classifying signals with one or two components, and finally measuring their throughput and latency as a function of batch size.

A. Single-component performance

In Fig. 4, we show the average accuracy over all classes as a function of the SNR in the case of single-component signals for every model we mentioned before. As we see, all

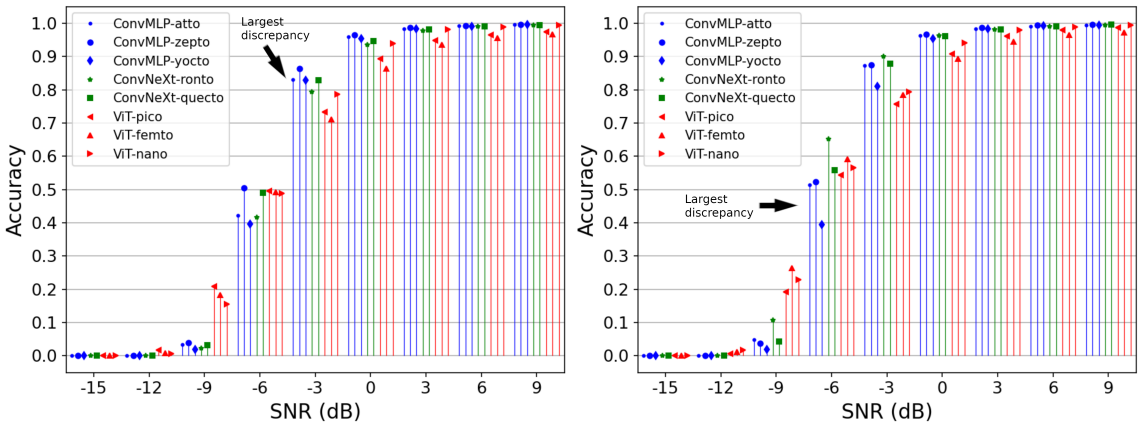

Fig. 4. Average accuracy vs. SNR of all models when classifying single-component signals, with models trained for (a) 75 epochs and for (b) 150 epochs.

convolutional models have very similar performance in the high SNR regime, sporting $\geq$95% accuracy for SNR $\geq$ 3 dB if trained for 75 epochs, and $\geq$95% for SNR $\geq$ 0 dB if trained for 150 epochs. The accuracy drops substantially at -3 dB, and even further at -6 dB.

ConvMLP manages 95% accuracy at 0 dB, whether these variants are trained at 75 or 150 epochs. In the former case, ConvMLP-zepto has the highest performance among ConvMLP variants. However, training it for 150 epochs yields barely no improvements. ConvMLP-yocto also barely improves, and it actually worsens at -3 dB. In contrast, ConvMLP-atto undergoes a sizeable improvement when training for 150 epochs, going from an accuracy of 42% to 51% at -6 dB, and essentially matching ConvMLP-zepto's performance.

A similar behaviour is observed in the other two architectures. The largest ViT variant, ViT-nano, exhibits a small improvement at lower SNR. However the two other variants improve by a larger margin, about 49% to 59% at -6 dB in the case of ViT-femto, and about 71% to 78% at -3 dB. ViT-nano does not improve at this noise level, but its accuracy is still somewhat better than ViT-femto. Curiously, ViT-femto and ViT-pico lag behind all other models, dipping below 95% at 3 dB and 75 epochs, and performing noticeably worse than the rest at 0 and -3 dB. More training improves their performance, comparable to CNN models at 75 epochs, but this widens the gap again. This suggests that small ViT variants trained in this fashion are not competitive with CNNs.

Surprisingly, ViTs are ahead of every model at an SNR of -9 dB, their accuracy falling off more smoothly than CNNs as SNR decreases. We conjecture that, because of the global nature of self-attention, ViTs might be able to better distinguish the signal from noise, given that longer-range relations between regions of the input are needed. ViTs are able to make these relations in the first layer, while CNNs need several layers to widen the receptive field and draw relations between regions of an image.

Lastly, ConvNeXt variants have a similar pattern of performance improvement when adding training compute in the variant with less channels at the start (ConvNeXt-ronto) versus its counterpart (ConvNeXt-quecto). The former, in fact, manages a substantial improvement when doubling training compute, from 42% to 65% at -6 dB, and 79% to 90% at -3 dB, as well as managing to be equal at higher SNR values to ConvNeXt-quecto.

At this small scale, models with more complexity in early layers require less training compute. Convolutional models with exponentially widening feature maps benefit significantly from increased compute. Despite having more parameters, these models are less computationally expensive due to fewer channels in initial layers, allowing higher throughput. For high throughput with sufficient accuracy, models with fewer channels in early layers that increase later are recommended.

In all architectures, increased computational cost, rather than parameter count, appears to ease the process of finding performant model configurations. While more parameters theoretically provide more potential solutions [29], we observe this primarily in ViTs, where both parameters and computational cost grow together, unlike in CNNs. More channels in the initial layers seem to improve convergence more than in later layers, though this has limits; for instance, ConvMLP-yocto underperforms, likely due to limited capacity in its later layers. Overparameterizing the bottom layers appears more effective, even with fewer overall parameters.

B. Dual-component performance

The dual-component case yields significant differences in performance figures, shown in Fig. 5. Power is shared between both components of the signal and with noise, as we can see in Fig. 1, so we expect these signals to be harder to classify. That is, the SNR is calculated with the signal itself being the sum of these two components. So, for the same SNR, a single-component signal concentrates more power in the spectrogram than either of the components in a dual-component signal does. If we also take into account the fact that the power in either component does not have to be equal, as we considered in our dataset, this means that the component with less power will be harder to classify due to being more sensitive to noise.

Indeed, we see a sharp drop in accuracy at every SNR, from nearly 100% to 88-93% (depending on the model variant) at

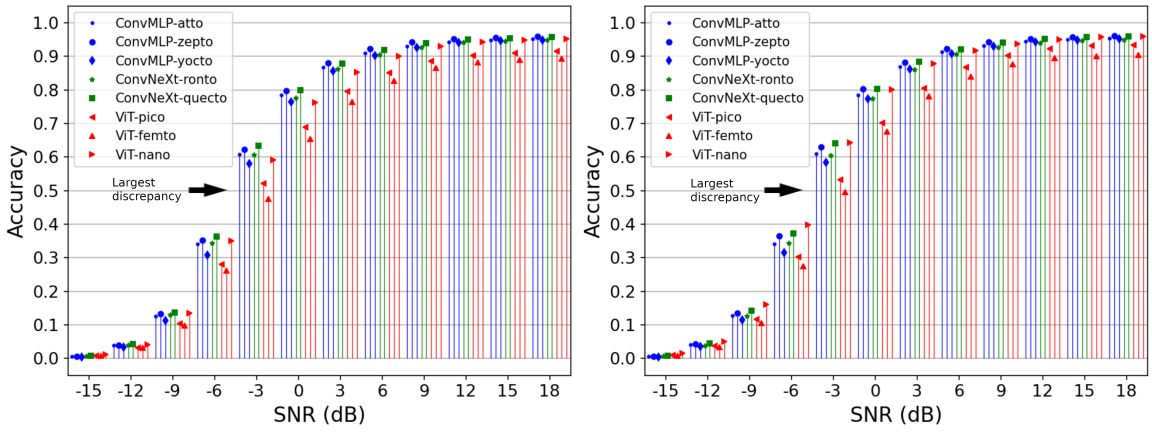

Fig. 5. Average accuracy vs. SNR of all models when classifying dual-component signals, with models trained for (a) 75 epochs and (b) for 150 epochs.

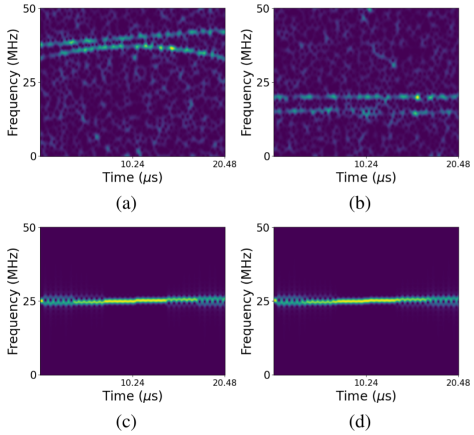

Fig. 6. Classification examples of signals; (a) Dual-component LFM-EQFM, -3 dB SNR. Only ViT-femto classified this signal incorrectly. (b) Dual-component BPSK-P3 (order 5 and order 6 respectively) ConvNeXt-quecto, -ronto and ViT-femto classified it incorrectly, -3 dB SNR. (c) Single-component P1 (order 6), infinite SNR. All ConvMLP variants classified it incorrectly. (d) Single-component P2 (order 6), infinite SNR. All ConvMLP variants classified it incorrectly.

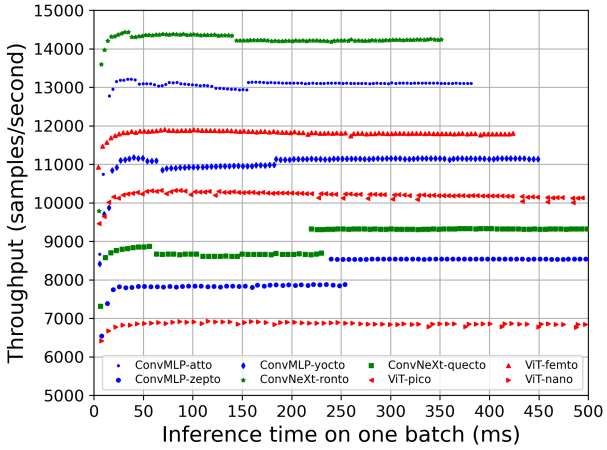

Fig. 7. Throughput vs. latency. Each point corresponds to a different batch size, which ranges from 50 to 5000, in increments of 50.

high SNR, and from 75-90% to 50-64% at -3 dB. All CNN variants manage $\geq$95% when SNR $\geq$ 12 dB, and $\geq$90% when SNR $\geq$ 6 dB. Accuracy falls off more smoothly and does so earlier than in the single-component case, in general. However, unexpectedly, we do see higher accuracy in all models at -15, -12 dB, as well as higher accuracy in convolutional models at -9 dB than in the single-component case. This might be explained by our training dataset being unbalanced with respect to the number of components, we have more than six times more dual-component signals than single-component ones in this set, so models were trained more on the former, better learning how to distinguish such signals from noise.

It is immediate to observe that ConvMLP-atto is ahead of ConvMLP-yocto and ConvNeXt-ronto at lower SNRs, partly in contrast to the single-component case, with the most performant variant among ConvMLP variants being ConvMLP-zepto. The lead among all models, though, is disputed between ConvNeXt-quecto and ViT-nano. Accuracy among CNNs and ViT-nano ranges from less than 1% at higher SNRs to about 7% at -3 dB, with ViT-nano narrowly in the lead, and ConvMLP-yocto falling behind. The two smallest ViTs are much less performant than all the other models, as much as 15% at -3 dB.

Note that the accuracy of the CNNs and the two smallest ViTs barely increases with more training compute in this case. Surprisingly, the largest variant, ViT-nano, achieves the largest increase, contrary to the single-component case. The lack of an improvement on the rest of the models suggests that, given their capacity and this dataset, they are at the limit of what is able to be correctly classified. This contrasts with the findings in the single-component case, where more compute clearly benefited some models. In view of this, we come to the conclusion that larger models were undertrained at 75 epochs on single-component signals, though they seem to be trained enough on dual-component signals.

Improved preprocessing, a larger model, or alternative training regimes could enhance performance, as suggested by ViT-nano's results. Being the largest model, it surpasses ViT-pico in parameters by 34% and outpaces ConvMLP-zepto in FLOPs

TABLE II
TRAINABLE PARAMETERS, COMPUTATIONAL COST (IN MEGAFLOPS), MAXIMUM THROUGHPUT (IN SAMPLES/SECOND), AND SINGLE-COMPONENT ACCURACY AT -3 DB WHEN TRAINED FOR 150 EPOCHS OF ALL CONSIDERED MODEL VARIANTS.

Model	Trainable parameters	Comp. cost	Max. thr.	Sing. comp. acc. at -3 dB
ConvMLP-atto	288 312	113.9	13 209	87.37%
ConvMLP-zepto	259 264	274.8	8 548	87.55%
ConvMLP-yocto	127 132	140.6	11 178	81.05%
ConvNeXt-ronto	233 840	47.5	14 432	90.08%
ConvNeXt-quecto	192 656	89.6	9 333	87.80%
ViT-femto	730 384	144.4	11 900	78.48%
ViT-pico	1 059 344	211.8	10 326	75.82%
ViT-nano	1 455 888	313.5	6 927	79.35%

(see Table II). Doubling training compute makes it competitive with CNNs at higher SNRs and slightly better at lower SNRs, though this significantly reduces throughput, making it the slowest model. Thus, careful balancing of throughput, latency, and performance is crucial for this architecture at a small scale.

In Fig. 6, we explore concrete classification examples by every model. Dual-component examples with high noise levels can be classified reasonably well if they are frequency modulated, as they span significant space in the time-frequency plane, aiding their distinction. However, phase modulations span very little space, and their defining features, such as the rings occurring when phase shifts, are both limited in space and can be easily confused for noise. Since their frequency does not change, it is easy to confuse such modulations among each other and with NM. Hence, the models we trained are less performant in these scenarios. We also illustrate another source of confusion; P1 and P2 modulations are nearly identical when represented on a spectrogram if the order of their codes is 6 or 8, but not 7. This leads to much more confusion among these two modulations, with all models performing poorly when classifying these signals, even with very little noise.

C. Throughput and latency

As we mentioned before, achieving high throughput and low latency is key in a time-sensitive context [35]. Not only is it important if many signals are oncoming, but higher throughput ensures that less powerful devices, with a lower memory bandwidth or less compute, which might be constrained by power or cost, can execute these models at an acceptable speed. We show such metrics in Table II and Fig. 7, where latency is represented by the time taken by the model to do inference on one batch, while letting batch size vary.

ConvNeXt-ronto leads in throughput, achieving inference times below 50 ms, which aligns with its lower computational cost. Larger variants generally offer higher throughput. Using more channels in early layers increases computational cost due to the larger input size of these layers compared to later feature maps. While this boosts performance (as seen in Fig. 4 and Fig. 5), it reduces throughput. ConvNeXt-ronto outpaces ConvMLP-atto by over 10^3 samples/second despite differences in computational cost, as ConvMLP's two FC layers are costlier than ConvNeXt's 1×1 convs., while ViTs lag due to higher computational cost but utilize resources efficiently, like ViT-nano, which achieves half the throughput of ConvNeXt-ronto despite being six times more computationally expensive. This efficiency comes from their wider design compared to CNNs, which typically use fewer channels in early layers.

All CNN variants show abrupt throughput variations with increasing batch size and latency. While some have no negative impact, others delay the latency at which maximum throughput is achieved, particularly in computationally costly models. This may result from quantization effects, causing GPU underutilization due to suboptimal tensor sizes for the GPU's topology and memory access patterns. Adjusting model hyperparameters and batch sizes could uncover optimal values to maximize throughput while minimizing latency.

VI. CONCLUSION

In this work, we have trained several variants of ConvMLP, ConvNeXt and ViT on synthetic pulsed signal reception data, generated specifically for this purpose, in order to compare their performance when classifying signals' modulation types. We have made emphasis on evaluating these models in scenarios closer to a realistic use case, by making use of single- and dual-component signals, each component having one out of 15 modulation types. We also considered a time-constrained context, where high throughput and low latency is of importance, measuring these two runtime metrics.

Our analysis shows that ConvMLP and ConvNeXt have comparable performance and throughput. Variants with fewer channels in the initial layers offer the best throughput and accuracy when trained longer in the single-component case. However, in the dual-component case, variants with more initial channels, fewer parameters, but higher computational cost outperform others. ViTs underperform both CNNs, except for the largest variant, which matches CNNs only with higher training compute, resulting in the highest computational cost and lowest throughput at inference. Additionally, ViTs require careful tuning of training hyperparameters, with AdamW and data augmentation proving essential for generalization, which also benefit CNNs.

To sum up, if looking for performant models at this scale, while using a conventional training regime, ConvMLP and ConvNeXt are preferable to ViTs on this domain. These two CNNs manage high throughput at low latencies coupled with an accuracy about 80-90% with SNR $\geq$ -3 dB in the case of single-component signals, and 75-80% with SNR $\geq$ 0 dB in the dual-component case. ConvMLP-atto and ConvNeXt-ronto possess the highest throughput, while ConvNeXt-quecto, ConvMLP-zepto possess the best performance, with a noticeable throughput hit.

We expect our work to be a useful and actionable resource for researchers looking to train their own CNN- and ViT-based models on this domain, especially if throughput is a main concern of theirs, as is the case in the scenarios we discussed here. Specifically, we would suggest future efforts be directed towards variations of ViTs, such as Swin Transformer, in order to extract as much performance as possible while keeping throughput competitive with newer CNN models.

We believe it is of great interest in future work to keep researching models at this small scale, and one of the most immediate avenues would be to explore their accuracy when trained and evaluated using reduced or mixed precision floating point arithmetic [36], as well as pruning and quantization. The throughput gains can be substantial, however such gains would only make sense if the accuracy loss is minimal. In order to improve accuracy, using other training regimes such as self-supervised training, data augmentation schemes such as time-stretching, and use of other modalities, like constellation diagrams, more adequate than spectrograms for distinguishing phase modulations, is also worth looking into. Characterizing the performance and throughput of other time-frequency distributions coupled to these models, allowing for better noise rejection and improved resolution, is of interest as well.

REFERENCES

[1] M. Zhang, M. Diao, and L. Guo, "Convolutional Neural Networks for Automatic Cognitive Radio Waveform Recognition," *IEEE Access*, vol. 5, pp. 11 074–11 082, 2017.

[2] Z. Qu, X. Mao, and Z. Deng, "Radar signal intra-pulse modulation recognition based on convolutional neural network," *IEEE Access*, vol. 6, pp. 43 874–43 884, 2018.

[3] Z. Qu, W. Wang, C. Hou, and C. Hou, "Radar signal intra-pulse modulation recognition based on convolutional denoising autoencoder and deep convolutional neural network," *IEEE Access*, vol. 7, pp. 112 339–112 347, 2019.

[4] Q. Qu, S. Wei, H. Su, M. Wang, J. Shi, and X. Hao, "Radar signal recognition based on squeeze-and-excitation networks," in *Proc. Int. Conf. on Sig., Inf. and Data Proc.*, 2019, pp. 1–5.

[5] E. Norgren, "Pulse repetition interval modulation classification using machine learning," M.S. thesis, School of Elec. Engin. and Comp. Sci., Royal Inst. of Tech., Stockholm, Sweden, 2019.

[6] X. Li, Z. Liu, and Z. Huang, "Attention-Based Radar PRI Modulation Recognition With Recurrent Neural Networks," *IEEE Access*, vol. 8, pp. 57 426–57 436, 2020.

[7] L. Boegner et al., "Large Scale Radio Frequency Signal Classification," 2022, *arXiv:2207.09918*.

[8] S. Yuan, P. Li, and B. Wu, "Towards Single-Component and Dual-Component Radar Emitter Signal Intra-Pulse Modulation Classification Based on Convolutional Neural Network and Transformer," *Remote Sensing*, vol. 14, no. 15, p. 3690, Aug. 2022.

[9] A. Vaswani et al., "Attention Is All You Need," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst*, 2017.

[10] T. B. Brown et al., "Language Models are Few-Shot Learners," in *Proc. 33th Int. Conf. Neural Inf. Process. Syst*, 2020.

[11] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," 2020, *arXiv:2010.11929*.

[12] J. Li, A. Hassani, S. Walton, and H. Shi, "ConvMLP: Hierarchical Convolutional MLPs for Vision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6306–6315.

[13] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11976–11986.

[14] W. Xiao, Z. Luo, and Q. Hu, "A Review of Research on Signal Modulation Recognition Based on Deep Learning," *Electronics*, vol. 11, no. 17, p. 2764, 2022.

[15] C. Hou, D. Fu, L. Hua, Y. Lin, G. Liu, and Z. Zhou, "The recognition of multi-components signals based on semantic segmentation," *Wireless Networks*, vol. 29, no. 1, pp. 147–160, 2023.

[16] L. Cohen, "Time-frequency distributions-a review," in *Proceedings of the IEEE*, vol. 77, no. 7, pp. 941–981, July 1989.

[17] H. Wang et al., "A radar waveform recognition method based on ambiguity function generative adversarial network data enhancement under the condition of small samples," *IET Radar, Sonar & Navigation*, vol. 17, no. 1, pp. 86–98, 2022.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[19] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers," *Transactions on Machine Learning Research*, May 2022.

[20] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proc. 32th Int. Conf. Neural Inf. Process. Syst*, 2019, pp. 8024–8035.

[21] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2826–2830, 2011.

[22] "fvcore library," Dec. 2023. [Online]. Available: https://github.com/facebookresearch/fvcore/.

[23] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[24] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *Proc. Int. Conf. on Learn. Rep.*, 2019.

[25] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-Deep Neural Networks without Residuals," in *Proc. Int. Conf. on Learn. Rep.*, 2017.

[26] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," in *Proc. AAAI Conf. on Artif. Intel.*, vol. 34, no. 7, pp. 13001–13008, 2020.

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017, *arXiv:1412.6980*.

[28] F. C. Akyon, Y. K. Alp, G. Gok, and O. Arikan, "Classification of intra-pulse modulation of radar signals by feature fusion based convolutional neural networks," in *Proc. 26th Euro. Sig. Proc. Conf.*, 2018, pp. 2290–2294.

[29] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 033–30 038, Dec. 2020.

[30] H. Fei, et al., "MobileAmcT: A Lightweight Mobile Automatic Modulation Classification Transformer in Drone Communication Systems," *Drones* vol. 8, no. 8, pp. 357-378, 2024.

[31] L. Luo, J. Huang, Y. Yang and D. Hu, "Radar Waveform Recognition with ConvNeXt and Focal Loss," *IEEE Access*.

[32] W. Ma and Z. Cai, "Deep Learning Based Cognitive Radio Modulation Parameter Estimation," *IEEE Access*, vol. 11, pp. 20963-20978, 2023.

[33] A. A. Abdullah, H. Veisi, T. Rashid, "Breaking Walls: Pioneering Automatic Speech Recognition for Central Kurdish: End-to-End Transformer Paradigm," 2024, *arXiv:2406.02561*.

[34] A. A. Abdullah et al., "Advanced Clustering Techniques for Speech Signal Enhancement: A Review and Metanalysis of Fuzzy C-Means, K-Means, and Kernel Fuzzy C-Means Methods," 2024, *arXiv:2409.19448*.

[35] A. A. Abdullah, S. S. Muhamad, H. Veisi, "Enhancing Kurdish Text-to-Speech with Native Corpus Training: A High-Quality WaveGlow Vocoder Approach," 2024, *arXiv:2409.13734*.

[36] A. A. Abdullah, S. Tabibian, H. Veisi, A. Mahmudi, T. Rashid, "End-to-End Transformer-based Automatic Speech Recognition for Northern Kurdish: A Pioneering Approach," 2024, *arXiv:2410.16330*.

[37] S. S. Muhamad, H. Veisi, A. Mahmudi, A. A. Abdullah, F. Rahimi, "Kurdish end-to-end speech synthesis using deep neural networks," *Natural Language Processing Journal*, vol. 8, pp. 100096–100107, 2024.