UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

DANIEL DE OLIVEIRA RUBIANO

Inferência Adaptativa para Classificação Automática de Modulação 5G

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Antonio Carlos Schneider Beck Filho
Co-orientador: Guilherme dos Santos Korol

Porto Alegre
2022

“The very nature of science is discoveries, and the best of those discoveries are the ones you don’t expect.”

— NEIL DEGRASSE TYSON

AGRADECIMENTOS

Gostaria de agradecer primeiramente aos meus pais, Osvaldo e Sônia, por estarem ao meu lado desde o início desta jornada e proporcionarem as melhores condições possíveis para que eu pudesse chegar até aqui. Sou grato pelo afeto, pelos ensinamentos e pela dedicação incondicional aos diversos aspectos da minha educação e formação como pessoa. Gostaria de agradecer ao meu orientador Prof. Dr. Antônio Carlos Schneider Beck pela paciência e dedicação constantes ao longo do desenvolvimento deste trabalho. Gostaria de agradecer ao meu co-orientador Guilherme dos Santos Korol, por estar sempre prontamente disposto a ajudar e solucionar minhas dúvidas, e também pelos diversos ensinamentos. Gostaria de agradecer aos amigos e familiares, em especial a minha tia Marisa, que de alguma forma me apoiaram ao longo desta jornada, seja com uma palavra amiga, um conselho ou por simplesmente estarem presentes.

RESUMO

A correta classificação de modulações de sinais de rádio é um aspecto crucial no uso eficiente do espectro de radiofrequência em aplicações modernas, como em sistemas IoT baseados na tecnologia 5G. Soluções estado-da-arte para o problema de classificação de sinais são baseadas em métodos de Aprendizado Profundo (e.g. Redes Neurais Artificiais). No entanto, esses métodos possuem um alto custo computacional e energético, ao ponto de aceleradores dedicados (e.g. FPGA) serem utilizados para desempenhar o processamento. Baseando-se na noção de que a classificação se torna computacionalmente mais fácil ou difícil dependendo da quantidade de ruído que o sinal está submetido (i.e. *Signal-to-Noise Ratio* - SNR), este trabalho propõe um método de inferência adaptativa baseado em Redes Neurais Artificiais para classificar sinais de rádio 5G. Ao desenvolver um sistema de inferência adaptativo embarcado baseado em FPGA, que seleciona o modelo de Rede Neural Artificial mais apropriado de acordo com a qualidade atual do sinal (nível de SNR), foram obtidos ganhos em eficiência energética e desempenho considerando cenários reais quando comparado a soluções estado-da-arte.

Palavras-chave: Classificação de sinais de rádio 5G. redes neurais. aprendizado profundo. FPGA.

Adaptive Inference for 5G Automatic Modulation Classification

ABSTRACT

Correct classification of radio signal modulation is a core enabler to the efficient use of the radio frequency spectrum in modern applications, like 5G-based IoT systems. State-of-the-art solutions to the signal classification problem are based on Deep Learning methods (e.g. Artificial Neural Networks). However, these methods require heavy processing and high energy consumption up to the point that accelerators (e.g., FPGA) are used to carry out such computations. Based on the observation that the classification becomes computationally harder or easier depending on the amount of noise the signal is subject to (i.e., Signal-to-Noise Ratio - SNR), this work proposes an adaptive inference method using Artificial Neural Networks to classify 5G radio signals. By developing a fully adaptive FPGA-based inference system that selects the most appropriate Artificial Neural Network model according to the current signal quality (SNR level), we achieved increased energy efficiency and greater overall performance on practical scenarios compared to a state-of-the-art approach.

Keywords: 5G radio signal classification, machine learning, deep learning, FPGA.

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
AM	Amplitude Modulation
API	Application Programming Interface
ASK	Amplitude Shift-Keying
AWGN	Additive White Gaussian Noise
AXI	Advanced Extensible Interface
BRAM	Block Random Access Memory
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CR	Cognitive Radio
DL	Deep Learning
DRC	Design Rule Check
DSP	Digital Signal Processor
DSA	Dynamic Spectrum Access
FDM	Frequency Division Multiplexing
FF	Flip-Flop
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
HLS	High Level Synthesis
IA	Inteligência Artificial
ICAP	Internal Configuration Access Port
IoT	Internet Of Things
IP	Intellectual Property

LUT	Look-Up Table
ML	Machine Learning
MLP	Multi Layer Perceptron
MPSoC	Multiprocessor System On Chip
OFDM	Orthogonal Frequency Division Multiplexing
OOC	Out Of Context
ONNX	Open Neural Network Exchange
PSK	Phase Shift Keying
QAT	Quantization-Aware Training
QAM	Quadrature Amplitude Modulation
RAM	Random Access Memory
RF	Radio Frequency
RNA	Redes Neurais Artificiais
RTL	Register Transfer Level
SDR	Software-Defined Radio
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
STA	Static Timing Analysis
VGG	Visual Geometry Group
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VRAM	Video Random Access Memory
WANET	Wireless Ad-hoc Network
WLAN	Wireless Local Area Network
5G	5th Generation Mobile Network

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A presença de dispositivos sem fio (*wireless*) no cotidiano não é novidade. Equipamentos como smartphones, redes de monitoramento, dispositivos da Internet of Things (IoT), além de dispositivos de suporte de redes de acesso sem fio como roteadores, utilizam tecnologias baseadas em redes Wireless LAN (Wireless Local Area Network) para enviar e transmitir dados. Segundo um relatório realizado pela Ericsson, estima-se 29 bilhões de dispositivos conectados em 2022, sendo 18 bilhões deles IoT (ERICSSON, 2021). Uma grande parte desses dispositivos também possui a capacidade de utilizar redes ad-hoc descentralizadas (WANETS), as quais utilizam padrões de redes de comunicação móveis sem fio, popularmente conhecidas como 2G, 3G, 4G e, mais recentemente, 5G, para qual são esperados 550 milhões de assinantes (ERICSSON, 2021).

O crescente número de dispositivos 5G, e o aumento no volume de dados por eles provocado, desafia pesquisadores a encontrar soluções de requerimentos de capacidade e infraestrutura nunca vistas (AL-FALAHY; ALANI, 2017; AKPAKWU et al., 2018). Particularmente, as tecnologias de redes sem fio são baseadas em técnicas de modulação de frequência, amplitude ou fase. A escolha dessas técnicas impacta diretamente em aspectos como confiabilidade e energia. Estabelecendo uma analogia com a comunicação humana, tipos distintos de modulação são como linguagens distintas. No momento em que duas pessoas estabelecem uma conversação, ambas se comunicam de forma que sejam corretamente interpretadas, sendo a linguagem estabelecida implicitamente. Assim como na comunicação humana, deve existir um acordo entre as partes na comunicação entre dispositivos. Na prática, o espectro de rádio é como um conjunto de indivíduos que falam diversas línguas se comunicando simultaneamente. Portanto, um aspecto crucial para a transmissão de informações é o acordo entre os agentes comunicantes.

Diferente do que ocorre na interação humana com a linguagem, a modulação não é definida implicitamente pelas partes, havendo a necessidade de identificar suas diferentes variações. Portanto, é necessário um método adequado para a classificação desses sinais de modulação. Um dos principais aspectos a ser considerado como possível complicador na classificação de sinais de Radiofrequência (RF) é a relação sinal-ruído (SNR) em que o sinal está submetido. Sinais com SNR baixo possuem uma quantidade maior de ruído (em relação ao sinal), dificultando uma classificação precisa (WU et al., 2017).

Em situações de maior ruído, portanto, cria-se a necessidade por algoritmos mais complexos (e custosos) capazes de identificar (i.e. classificar) o modulador. Assim, cria-

se margem para otimizar a classificação de sinais no que diz respeito a energia e desempenho. Nesse contexto, é interessante a proposição de uma solução adaptativa que *ajuste* o algoritmo e seus custos às atuais condições de ruído do ambiente.

1.1 Contextualização do Problema

Devido ao crescimento da utilização de tecnologias de redes sem fio, é de extrema importância a definição de regras e estratégias para a divisão do espectro de rádio que propiciem a sua utilização de forma eficiente (KOLODZY, 2004). Nesse contexto, *Dynamic Spectrum Access* (DSA) e *Cognitive Radio* (CR) foram propostos, estabelecendo a necessidade de aperfeiçoar a sensibilidade e identificação de sinais RF, de forma que os dispositivos se tornem capazes de detectar e identificar usuários, protocolos de modulação e interferências de maneira automática e inteligente (O'SHEA; CORGAN; CLANCY, 2016; KOLODZY, 2004; MITOLA; MAGUIRE, 1999).

Ambos DSA e CR dependem de algo em comum: a identificação (ou *classificação*) de padrões na comunicação entre os dispositivos. O reconhecimento da modulação é necessário para distinguir diferentes tipos de dispositivos, como transmissores *broadcast* e de dados, radares e fontes de interferência, os quais possuem diferentes necessidades. Essa identificação é uma das etapas para interpretar o tipo de comunicação, além do próprio emissor (O'SHEA; CORGAN; CLANCY, 2016). Além do mais, a natureza dinâmica das redes sem fio com diferentes ambientes e qualidades de sinal se apresenta como um dificultador adicional. Para lidar com esses desafios, estudos como o de (JAGANNATH, 2019) tratam de esquemas de modulação adaptativa de acordo com a relação de magnitude da razão de sinal ruído (SNR).

Tradicionalmente, o problema de classificação de sinais RF é abordado através do desenvolvimento de algoritmos especializados. Por exemplo, em (HAMEED; DOBRE; POPESCU, 2009) é utilizada uma abordagem de *likelihood*. Recentemente, técnicas de Aprendizado de Máquina, e em especial Aprendizado Profundo, vêm sendo usadas para alcançar resultados estado-da-arte em aplicações de classificação de modulação (O'SHEA; CORGAN; CLANCY, 2016; O'SHEA; ROY; CLANCY, 2018; SOLTANI et al., 2019; HUANG et al., 2019).

Entretanto, a qualidade na classificação de modulação produzida pelos algoritmos de Aprendizado Profundo é obtida com alto custo computacional. Uma alternativa para tornar a execução desse algoritmos mais eficiente é a utilização de aceleradores, em dife-

rentes arquiteturas de *hardware*. A principal vantagem da sua utilização é a alta vazão de processamento, baixa latência e alta eficiência energética se comparado a arquiteturas de propósito geral (e.g., CPU).

Atualmente, empresas como Amazon, Microsoft e Google empregam aceleradores GPUs (AMAZON, 2021), FPGAs (MICROSOFT, 2021) e ASICs (GOOGLE, 2022), para melhorar o processamento das suas aplicações de Aprendizado Profundo. Além disso, em dispositivos IoT, os aceleradores se tornam especialmente atrativos pois permitem a execução desses algoritmos em ambientes restritos em energia e/ou recursos (NVIDIA Corporation, 2022; XILINX, 2020; Xilinx, 2022a).

Portanto, é notável a importância de mecanismos dinâmicos e eficientes para a classificação da modulação. Esses mecanismos irão trabalhar de forma a impulsionar e facilitar o uso das redes sem fio em um cenário de crescente número de dispositivos conectados e de aplicações de alta demanda de carga, como as baseadas em 5G.

1.2 Contribuições

Este trabalho propõe o desenvolvimento de uma solução de inferência adaptativa utilizando Redes Neurais para a classificação de sinais de radiofrequência 5G. Especificamente, um sistema embarcado baseado em FPGA será utilizado para processar as inferências. Primeiro, gera-se um conjunto de diferentes Redes Neurais e aceleradores com diferentes perfis de acurácia, desempenho e custo energético. Depois, considerando a dinamicidade do ambiente, a adaptabilidade é obtida relacionando os diferentes níveis de SNR com a utilização inteligente desse conjunto de redes neurais e aceleradores. Em suma, o trabalho realiza as seguintes contribuições:

- Estabelece novas fronteiras no espaço de projeto dos classificadores de RF baseados em Redes Neurais Artificiais em FPGA ao explorar o compromisso acurácia-energia-desempenho.
- Avança o estado-da-arte quanto a adaptabilidade das aplicações de classificação RF em 5G propondo um sistema adaptativo que melhora a eficiência energética e desempenho em relação a abordagem tradicional.
- Considerando uma aplicação 5G, nossa abordagem melhora a eficiência energética em até 40%, aumentando a capacidade de processamento de inferências em até 70%.

2 REFERENCIAL TEÓRICO E TRABALHOS RELACIONADOS

2.1 Inteligência Artificial e Aprendizado de Máquina

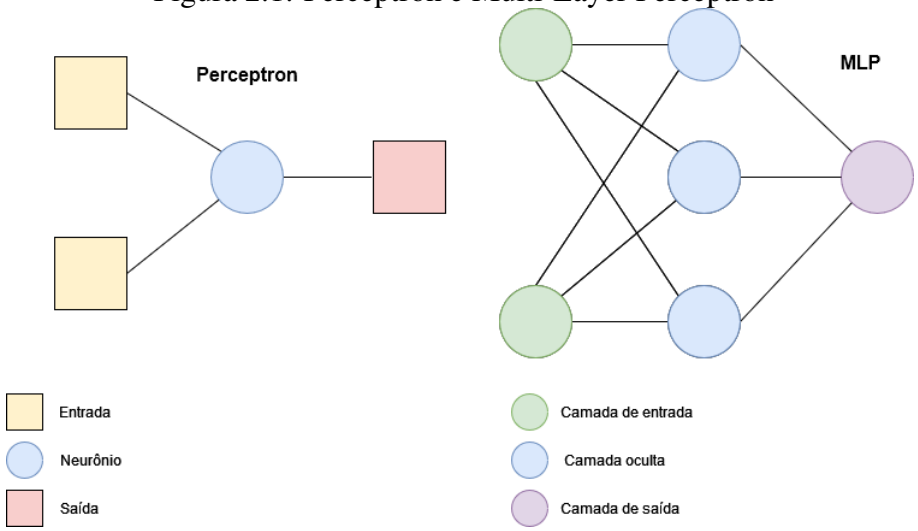
Segundo (LUGER, 1988), o conceito de Inteligência Artificial (IA) pode ser definido como uma vertente da ciência da computação que estuda a automação de comportamentos inteligentes. Mais tarde, (RUSSELL; NORVIG, 1995) definiram IA como o estudo de agentes que existem em um ambiente, os quais têm a habilidade de compreender e agir.

Já o conceito de aprendizado parte do princípio de que a inteligência está relacionada com uma ação racional, e um agente inteligente toma a melhor ação possível em uma dada situação. Além disso, agentes inteligentes são autônomos, no sentido de que seu comportamento deve ser determinado pela sua própria experiência, e não por um conhecimento pré-determinado. O aprendizado é o resultado de interações entre o agente e seu ambiente, tal que suas observações sobre ele contribuam para o processo de tomada de decisão (RUSSELL et al., 2010).

Aprendizado de Máquina (ML da sigla em inglês) é uma subárea da IA nas quais são baseadas as tecnologias atuais que envolvem esses resultados obtidos apenas da iteração entre agente e ambiente. Todo aprendizado está relacionado a uma tarefa. As tarefas podem ser divididas entre preditivas e descritivas. De acordo com a tarefa, podem existir duas abordagens distintas no que diz respeito ao aprendizado, podendo ser Supervisionado ou Não supervisionado (FACELI, 2011).

O aprendizado Supervisionado, que cobre as Redes Neurais Artificiais, tratadas nesse trabalho, é aplicado a dois tipos específicos de tarefas preditivas - classificação e regressão. Em tarefas preditivas, os algoritmos são aplicados a um conjunto de dados, ou *dataset*, de treinamento. Cada elemento pertencente a esse conjunto de dados, é representado pelos valores de seus atributos preditivos e um atributo alvo (rótulo). Esse rótulo é utilizado para treinar o modelo e aprender a relação entre os atributos e rótulos de todo *dataset*. Em tarefas de classificação, o rótulo é discreto, já em tarefas de regressão, é contínuo. O conceito de aprendizado supervisionado surge do fato de existir um supervisor externo que conhece o atributo alvo a ser predito, tendo a função de conduzir o processo de aprendizado de forma de refinar sua tomada de decisão (FACELI, 2011).

Figura 2.1: Perceptron e Multi-Layer Perceptron



Fonte: Os Autores

2.1.1 Aprendizado Profundo

Para lidar com demandas crescentes em *datasets* de maior complexidade, os estudos nas áreas IA e ML buscaram alternativas em outras áreas do conhecimento. Surge então o conceito de Redes Neurais Artificiais (RNAs) inspiradas no processo de aprendizagem do cérebro humano (FACELI, 2011). A arquitetura de uma RNA é baseada na existência de uma unidade de processamento simples, que está interconectada com outras. Tal unidade executa funções matemáticas, e foi denominada *neurônio artificial*.

Segundo (HAYKIN, 1999), um neurônio pode ser descrito através de três elementos principais. Os elos de conexão, ou sinapses, são responsáveis por receber os sinais de entrada. Cada sinapse é caracterizada por seu peso sináptico que multiplica a entrada. Então, as entradas ponderadas são combinadas de acordo com uma função matemática, como por exemplo, um combinador linear (i.e. somatório) e um peso adicional chamado de *bias*. O último elemento do neurônio é responsável por restringir a amplitude da saída e, mais importante, incluir não-linearidade ao neurônio. Seu resultado é tipicamente contido em um intervalo, e representa o estado de ativação do neurônio (FACELI, 2011).

O Aprendizado Profundo, ou Deep Learning (DL), se caracteriza por topologias de redes neurais que possuem ao menos duas camadas intermediárias de neurônios. Essas redes são suportadas por algoritmos específicos de treinamento. Tais redes são compostas por um módulo de aprendizado supervisionado responsável pela extração de características dos dados originais. A utilização de tais redes foi proporcionada por avanços na área de hardware, onde processadores, sejam de propósito geral ou Graphics Processing Units

(GPUs) apresentam desempenho suficiente para suportar redes complexas e permitir que elas tenham um desempenho preditivo superior a algoritmos anteriores (FACELI, 2011).

Redes denominadas Multilayer Perceptrons (MLP) foram um dos primeiros modelos de DL de sucesso. Essas redes são compostas de uma ou mais camadas intermediárias de neurônios em conjunto com uma camada de saída. Nas MLPs cada neurônio possui uma função específica, onde a função que ele implementa é uma combinação das funções implementadas por neurônios conectados em camadas anteriores. À medida que o processamento percorre as diferentes camadas, ele se torna mais complexo e com fronteiras de decisão distintas (FACELI, 2011). A Figura 2.1 ilustra a diferença entre um perceptron de única camada oculta (de um neurônio) e uma MLP de múltiplas camadas.

Para realizar o treinamento de redes MLP (ou também chamadas de totalmente conectadas) foram propostas diferentes abordagens, sendo a principal, o algoritmo de Back-propagation (RUMELHART, 1986). Em linhas gerais, o treinamento é composto por duas etapas: *forward* e *backward propagation*. A etapa de forward propagation ocorre quando a entrada percorre as diferentes camadas da rede da primeira até a última. A saída da rede é comparada ao valor desejado (rótulo da entrada), indicando o erro cometido pela rede. Esse erro é utilizado na etapa de backward propagation para ajustar os pesos de entrada, percorrendo-os no sentido contrário - da camada de saída até a de entrada (FACELI, 2011). O algoritmo de Back-propagation é baseado na técnica do gradiente descendente proposta por (RUMELHART, 1986).

2.2 Modulação e Classificação de Sinais de Rádio

2.2.1 Modulação de Sinais

A transmissão de dados por cabos e redes sem fio é baseada em técnicas de modulação. Sinais de dados digitais são enviados e recebidos através da aplicação dessas técnicas. Segundo (HAYKIN; MOHER, 2008), a modulação pode ser definida como o processo através do qual uma ou mais características de uma onda portadora são modificadas de acordo com a informação do sinal da mensagem. Dessa forma, o sinal que carrega a informação ou mensagem, é chamado de sinal modulante, e o resultado do processo de modulação é denominado de sinal modulado. Portanto, todo o sinal de dados digital modifica uma onda, usualmente senoidal, denominada portadora. Na comunicação entre dispositivos, o dispositivo transmissor, que realiza o processo de modulação, é

denominado modulador, e o dispositivo receptor que recupera a mensagem é denominado demodulador.

A onda portadora pode ser modificada de diferentes formas. A modulação por amplitude (AM) ocorre quando o sinal que contém a informação modifica a amplitude da onda senoidal portadora, com sua frequência permanecendo constante. Já a modulação por frequência (FM) mantém a amplitude constante, variando a frequência da portadora. Por último, a modulação por fase (PM) mantém ambas frequência e amplitude constantes, porém modifica a fase da onda portadora. Em esquemas mais avançados de modulação, uma ou mais características podem ser moduladas em conjunto (JR., 2012).

A modulação digital é utilizada para codificar os bits de informação através de diferentes símbolos. Então, cada sequência de bits possível possui um comprimento estabelecido e é relacionado a um símbolo predeterminado. O conjunto possível de símbolos é denominado constelação. Esse tipo de modulação é denominado modulação multinível. As diferentes formas de modulação digital são classificadas de acordo com a característica da onda modificada e o tamanho da constelação representada. Por exemplo, a técnica 8-PSK realiza a modulação de fase (*phase*) e possui uma constelação de 8 símbolos, representando um conjunto de 3 bits transmitidos a cada modificação da portadora (FOROUZAN; MOSHARRAF, 2013). A tecnologia 5G utiliza a técnica de modulação denominada Orthogonal Frequency Division Multiplexing (OFDM). Proposta em (CHANG; GIBBY, 1968), ela consiste em combinar as técnicas de modulação de amplitude em quadratura (QAM) com a multiplexação de divisão em frequência (FDM) (WEINSTEIN, 2009). A técnica de modulação QAM se refere a um conjunto de classes de modulação, as quais realizam a modulação de amplitude e fase. Neste trabalho, iremos analisar 24 classes que possuem essas características: OOK, ASK, BPSK, QPSK, PSK, APSK, QAM, AM-SSB-WC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMSK, OQPSK (O'SHEA; ROY; CLANCY, 2018).

2.2.2 Classificação de Sinais de Rádio

A classificação de sinais de rádio pode ser abordada como um problema de decisão. A entrada do classificador é uma amostra do sinal de rádio convertida do domínio analógico para o digital de forma que a frequência da portadora é aproximadamente alinhada com a portadora de interesse. A conversão consiste em obter diversas amostras numa taxa definida onde os componentes de fase e quadratura são amostrados, gerando

um vetor $1 \times N$, onde N é o número de amostras para um dado sinal (O'SHEA; CORGAN; CLANCY, 2016).

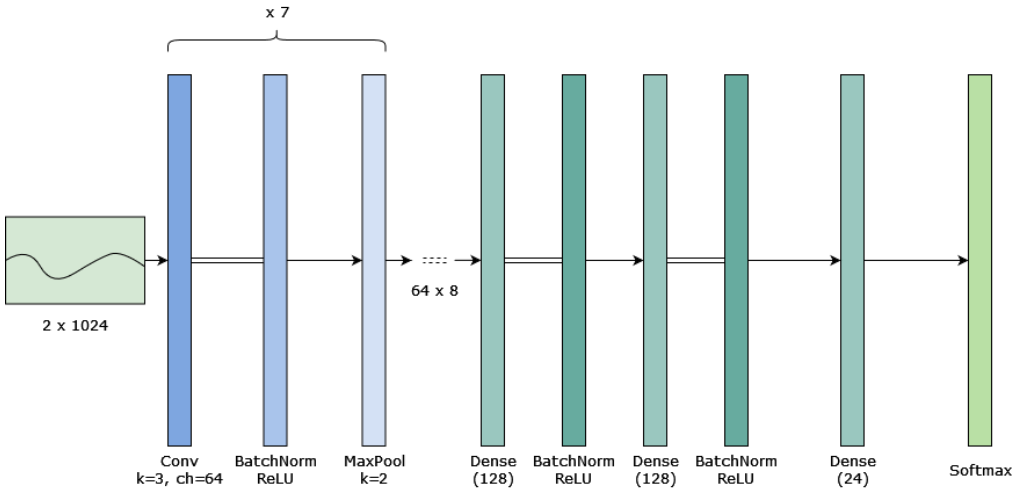
O problema de classificação é tratado tradicionalmente por algoritmos especializados. Uma das abordagens é o *likelihood*, que consiste em transformar a classificação em um problema de teste de hipóteses compostas, onde são reconhecidos padrões para classificar as diferentes modulações através de um teste que infere a razão de probabilidade do sinal ser de uma dada classe (HAMEED; DOBRE; POPESCU, 2009). No entanto, abordagens recentes tratam a classificação como um problema de decisão, e aplicam técnicas e algoritmos de ML para realizar a inferência com maior qualidade que algoritmos especializados (O'SHEA; ROY; CLANCY, 2018; WU et al., 2017).

2.2.3 Aprendizado Profundo para Classificação de Sinais RF

A classificação de sinais RF é uma das aplicações nas quais o uso de DL têm demonstrado excelente desempenho. Uma característica importante em relação ao ML tradicional, é a capacidade superior de extração de características relevantes a partir dos dados brutos. O desempenho dos algoritmos de IA está diretamente relacionado à capacidade do conjunto de treinamento representar o problema a ser tratado (FACELI, 2011). Os algoritmos tradicionais de ML dependem de ferramentas que realizam pré-processamento e engenharia de características. Como mencionado anteriormente, o uso de DL permite que o aprendizado de características ocorra com base nos dados de entrada, além da possibilidade de extrair essas características dado um domínio de dados específico, como textos, imagens ou sons (FACELI, 2011). Segundo (O'SHEA; ROY; CLANCY, 2018) essa habilidade é permitida pelo uso de técnicas de regularização, como o Dropout (SRIVASTAVA et al., 2014) e Batch Normalization (IOFFE; SZEGEDY, 2015) assim como o aperfeiçoamento de técnicas já estabelecidas como o Stochastic Gradient Descent (SGD) (TIELEMAN; HINTON., 2012) e inovações como Redes Neurais Convolucionais (CNNs) (LECUN et al., 1998).

CNNs como AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), LeNet (LECUN et al., 1998) e VGG (SIMONYAN; ZISSERMAN, 2015) combinam as técnicas mencionadas e permitiram que o modelo pudesse evoluir para além da extração de características simplificadas e modelos aproximados até modelos com alto grau de liberdade. Essas evoluções foram fundamentais para permitir o uso de DL em aplicações complexas. Quanto a classificação de sinais RF, as CNNs também demostram bom desempenho

Figura 2.2: Topologia da CNN VGG-10 em (O'SHEA; ROY; CLANCY, 2018).



Fonte: Os Autores

(O'SHEA; ROY; CLANCY, 2018).

Em especial, O'Shea et al. propuseram o uso de CNNs para a classificação de sinais RF (O'SHEA; ROY; CLANCY, 2018). O modelo utilizado é baseado em uma CNN estado-da-arte projetada para reconhecimento de imagens, a VGG-10 (SIMONYAN; ZISSERMAN, 2015). A rede é constituída de sete camadas de convolução com max-pooling e três camadas totalmente conectadas, sendo que a última delas gera o resultado de classificação. Técnicas como as funções de ativação ReLU e Batch Normalization vistas anteriormente também são aplicadas na rede. Neste caso específico, a entrada da CNN é composta por um quadro de 1024 amostras e a saída é a classificação do sinal em uma entre 24 classes de modulação.

2.2.4 O modelo VGG-10

Em (UMUROGLU JENTSCH, 2021), é proposta uma versão quantizada da CNN apresentada em (O'SHEA; ROY; CLANCY, 2018), que será utilizada como estudo de caso nesse trabalho. A técnica de quantização permite que os pesos e ativações, usualmente representados por dados em ponto flutuante, sejam representados por inteiros. Em comparação, operações com inteiros possuem um menor custo computacional e reduzem a utilização de memória. Por consequência, ocorrem reduções na latência e no consumo energético. No entanto, a quantização reduz a precisão da representação dos dados, tendo impacto na acurácia da rede.

Em particular, na versão proposta em (UMUROGLU JENTSCH, 2021), é apli-

cada uma quantização de 8 bits para pesos e ativações de todas as camadas. A Figura 2.2 apresenta um diagrama da sua topologia (ou arquitetura). Assim como a CNN descrita ao fim da seção anterior, a rede proposta consiste de sete conjuntos de camadas de convolução (cada uma com 64 canais e kernel de tamanho 3) seguidas por ReLU, Batch Normalization e MaxPooling (kernel de tamanho 2). Ao final, temos mais três camadas totalmente conectadas (*Dense* com 128 neurônios) e a camada de Softmax final que gera os valores de probabilidade de cada classe.

2.3 FPGAs e Aceleradores de Redes Neurais

Field Programmable Gate Array (FPGA) são dispositivos semicondutores que permitem que a função seja definida após a sua fabricação (i.e. são reconfiguráveis). FPGAs são compostas por uma matriz de portas (ou blocos) lógicos reprogramáveis com interconexões reconfiguráveis. Essa organização possibilita configurar o hardware para desempenhar qualquer função digital específica (XILINX, 2022b).

Dispositivos FPGA são utilizados tradicionalmente para executar tarefas especializadas através do projeto e descrição de uma arquitetura de hardware especializada. No entanto, é crescente seu uso em aplicações avançadas, como na construção de rádios para astronomia, processamento de chips com multiprocessadores, análise de dados físicos como o comportamento de partículas e análise de probabilidade aplicada à finanças (LEONG, 2008).

Além das aplicações tradicionais e avançadas, uma das utilizações de FPGAs são como aceleradores. Aceleradores são dispositivos dedicados a executar uma função de maneira mais eficiente ou mais rápida do que dispositivos propósito geral, ou utilizados em conjunto como apoio destes, a fim de aumentar sua capacidade de processamento e/ou melhorar aspectos como consumo energético. Devido a sua reconfigurabilidade, FPGAs são bons candidatos a desempenhar essa função em um contexto onde novas aplicações e algoritmos surgem em grande velocidade (Aprendizado de Máquina e IoT, por exemplo).

Se comparado as plataformas onde CNNs são executadas tradicionalmente, como CPUs e GPUs, a utilização de FPGAs é mais eficiente energeticamente (que ambas) e apresenta melhor desempenho (do que CPUs). Um dos facilitadores do uso de FPGAs é o surgimento de ferramentas de síntese de alto nível (ou HLS, de High-Level Synthesis) que substituem o método tradicional de desenvolvimento, até então realizado através de linguagens de descrição de hardware (e.g., VHDL e Verilog). A HLS permite transformar

código de linguagens de alto nível (e.g., C) em descrições Register Transfer Level (RTL) de baixo nível (MITTAL, 2020) que podem ser sintetizadas pelas ferramentas de CAD.

2.3.1 Síntese em FPGA

As etapas entre a transformação de uma descrição de hardware em linguagem de alto nível (HLS) em um circuito totalmente implementado em FPGA podem ser descritas através de um fluxo de projeto.

Inicialmente, o processo de HLS resulta em uma descrição de hardware abstraída na forma de um design RTL, que descreve o comportamento do circuito. O design RTL é obtido a partir da compilação do código em linguagem de alto nível, transformando-o em uma descrição equivalente em Verilog. A partir do design RTL é gerado o *netlist*, o qual descreve o design do ponto de vista organizacional, onde os elementos da lógica descrita no design são definidos como células. Uma célula pode representar os elementos básicos que compõem as diferentes funções do hardware, como um LUT (Look-up Table), registrador Flip-Flop, memórias do tipo Random Access Memory (RAM) ou até mesmo elementos que incorporam funções específicas como blocos de Digital Signal Processing (DSP). Todas as instâncias desses elementos são representadas por células. As interligações entre os elementos, chamadas de *nets*, além dos pinos que representam as entradas e saídas de uma célula, ou um conjunto delas, também são descritos no netlist (Xilinx Inc., 2021).

Com o netlist criado, é necessário definir o posicionamento e agrupamento das células, além de realizar otimizações de acordo com a plataforma alvo. Com o posicionamento das células definido, é realizado o roteamento, que define o caminho físico das interligações descritas no netlist. Entre as etapas de geração do netlist e roteamento, podem ser realizadas etapas de otimização intermediárias (Xilinx Inc., 2021).

Após a etapa de roteamento, a síntese do design é validada através da análise STA (Static Timing Analysis). A análise é uma forma de validar o comportamento do modelo em relação a sua temporização, onde são verificados todos os caminhos lógicos possíveis gerados pelo design a fim de confirmar que não existem violações. A validação é realizada através do cálculo do atraso de propagação, verificando se estão dentro das restrições definidas para a síntese (Synopsis, 2022).

Por fim, com o design validado, é realizada a etapa de Design Rule Check (DRC), onde são verificados erros físicos e lógicos, como restrições dimensionais, atribuição de

entradas e saídas incoerentes ou a falta de interligações descritas no netlist. Com o design verificado quanto a temporização, aspectos físicos e lógicos, é gerado um arquivo chamado *bitstream*, análogo a um programa executável, utilizado para configurar o FPGA. O bitstream contém todas as informações necessárias, como a configuração lógica e física dos elementos básicos, além dos dados a serem configurados inicialmente às memórias e Lookup Tables (LUTs) (Xilinx Inc., 2021).

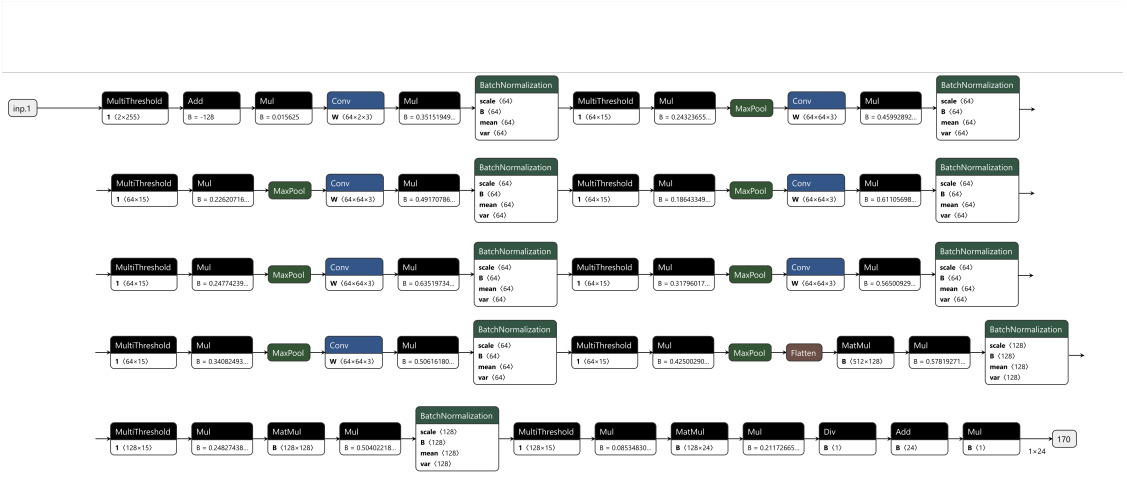
2.3.2 Reconfiguração em Tempo de Execução

Tradicionalmente, a configuração do FPGA é realizada de forma única, ocorrendo uma única vez durante a finalização do projeto quando ocorre programação do bitstream. Por outro lado, a reconfiguração em tempo de execução permite a adaptabilidade dos algoritmos/circuitos criados, o que possibilita a flexibilização do uso do hardware, como a utilização de múltiplas aplicações de funcionalidades distintas ou a alocação dinâmica de recursos. Aplicações como o gerenciamento dinâmico de potência (PAULSSON et al., 2007), a criação de aceleradores para a construção de Software Defined Radios (SDRs) (MCDONALD, 2008) e sistemas modulares (SEDCOLE et al., 2006) são alguns exemplos que se utilizam da reconfiguração da FPGA em tempo de execução.

Mais precisamente, a reconfiguração é coordenada por um co-processador embarcado conectado com o FPGA através de uma interface (e.g. AXI), e que utiliza a porta ICAP (Internal Configuration Access Port) que permite acesso a memória de configuração do FPGA. Portanto, o co-processador é responsável por realizar o controle da reconfiguração e coordenar a execução da FPGA (Xilinx Inc., 2022a).

É importante mencionar que o tempo de reconfiguração do FPGA pode também impactar negativamente o desempenho da aplicação. Por exemplo, com muitas reconfigurações frequentes (ou com reconfigurações lentas), podem ser geradas interrupções que são prejudiciais a aplicação. Esse aspecto é alvo de estudos como (DUHEM; MULLER; LORENZINI, 2011) no qual é proposto um gerenciador de reconfiguração, explorando formas de acelerar e diminuir a sobrecarga de reconfiguração através de técnicas como a compactação dos arquivos de reconfiguração. Os autores também identificaram a transferência do arquivo bitstream como o principal gargalo do tempo de reconfiguração. Isso se deve ao envio dos dados ocorrer em rajadas sequenciais onde o fluxo depende do processamento da rajada anterior.

Figura 2.3: Diagrama da VGG-10 (UMUROGLU JENTSCH, 2021) gerado pelo Netron.



Fonte: Os Autores

2.3.3 Ferramentas de Aprendizado de Máquina em FPGA

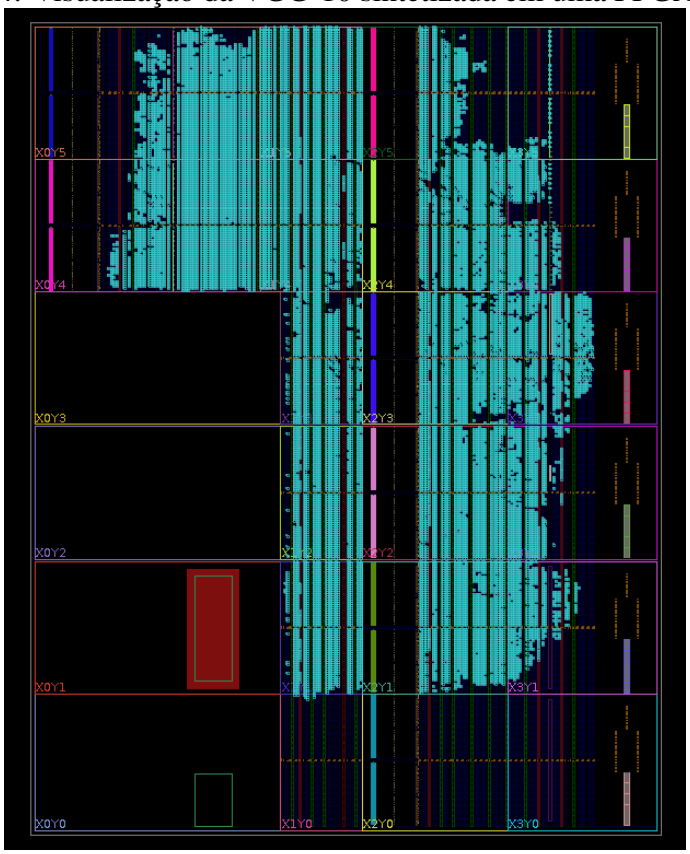
O uso de FPGAs como aceleradores de ML vem sendo popularizado, em grande parte, por ferramentas (ou *frameworks*) de geração automática de *hardware*. Destaca-se o framework FINN da empresa Xilinx (UMUROGLU et al., 2017) que vem sendo utilizado largamente na industria e academia. O FINN é uma ferramenta de código aberto que possibilita a construção de aceleradores de FPGAs de maneira rápida e flexível.

A ferramenta parte do arquivo ONNX (de Open Neural Network eXchange) gerado após o treinamento do modelo. Por exemplo, Figura 2.3 mostra o modelo VGG-10 quantizado apresentado em (UMUROGLU JENTSCH, 2021). A utilização do formato ONNX também possibilita o acesso a ferramentas como o Netron (Lutz Roeder, 2022), onde a figura foi gerada. A visualização permite verificar visualmente a estrutura da rede antes do início da síntese.

Em linhas gerais, o FINN mapeia uma rede neural como a da Figura 2.3 em módulos HLS pré-definidos. Com base nesse mapeamento e a fácil parametrização dos módulos HLS, é possível o ajuste da vazão (paralelismo) e do uso de recursos do FPGA em cada camada da CNN.

O ajuste do paralelismo se dá com a configuração de *folding* no FINN. Por meio de um arquivo JSON, valores de Processing Element (PE) e Single Instruction Multiple Data (SIMD) são definidos para cada módulo, que em conjunto definem o nível de paralelismo de cada módulo do acelerador. Consequentemente, os valores escolhidos têm um impacto

Figura 2.4: Visualização da VGG-10 sintetizada em uma FPGA ZCU104.



Fonte: Os Autores

significativo no desempenho e consumo de recursos de cada modelo. A seguir, detalhamos como ocorre o mapeamento de uma rede neural como as CNNs utilizadas nesse trabalho para um acelerador FPGA.

O fluxo de síntese do FINN pode ser dividido em dezoito etapas (BLOTT et al., 2018). A etapa inicial é a de *tidy-up*, onde o modelo é organizado atribuindo nodos e tensores nomes únicos, realizado o folding de constantes, inferência dos tipos de dados e remoção de entradas estáticas nos grafos. Em seguida, a etapa de *streamlining* reorganiza parâmetros de tipo ponto flutuante juntando os adjacentes em um único parâmetro de quantização. Esses parâmetros quantizam cada valor com base numa escala e deslocamento, e são implementados por um módulo HLS especifico do FINN chamado MultiThreshold. A etapa seguinte converte nodos compatíveis da rede (e.g., camadas convolucionais, totalmente conectadas, pooling, etc.) para os devidas módulos (classes) HLS. Os módulos HLS são então conectados de forma sequencial (seguindo a ordem estabelecida pelas camadas da rede neural), criando o chamado *Streaming Dataflow*. Caso algum alvo de vazão tenha sido especificado, é realizada uma transformação de folding para determinar automaticamente os atributos de paralelismo do acelerador. A próxima etapa é a aplicação desses parâmetros de folding específicos de cada camada/módulo.

Figura 2.5: Esquemático gerado a partir do modelo proposto em (UMUROGLU JENTSCH, 2021).

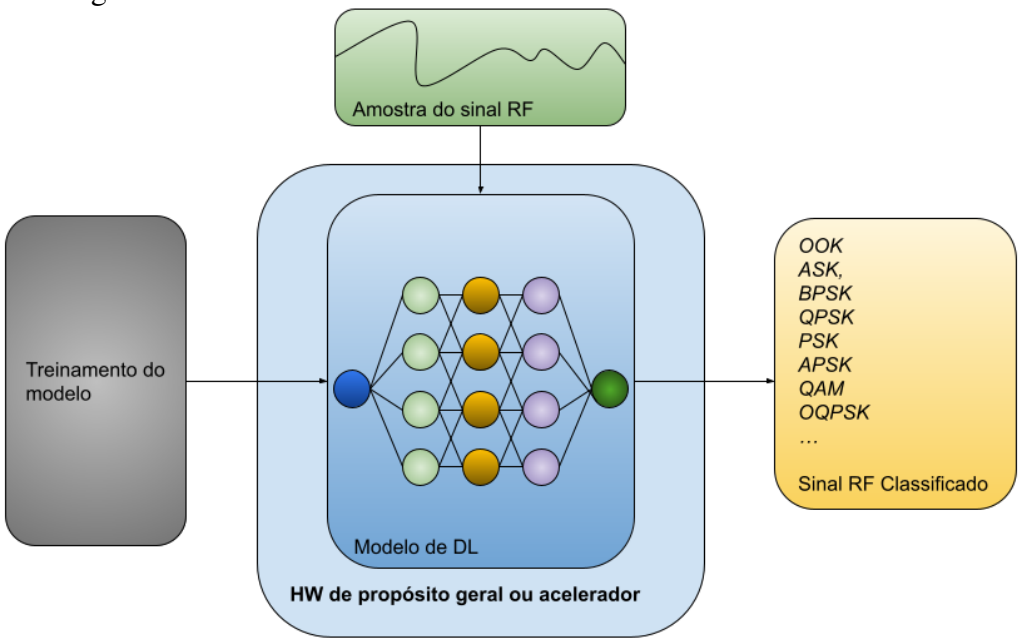
Fonte: Os Autores

Dado que todos os módulos HLS foram corretamente instanciados e configurados, pode-se iniciar a síntese FPGA. Primeiramente, esses módulos são sintetizados individualmente para criação de blocos IP (permitindo sua interconexão e reusabilidade mais facilmente durante o processo de síntese). Veja um exemplo na Figura 2.5 que apresenta o esquemático criado após síntese HLS do modelo VGG-10 proposto em (UMUROGLU JENTSCH, 2021). Adicionalmente, a profundidade de cada fila FIFO (usadas no interfaceamento entre os módulos) é determinada. Com os blocos IP gerados, o acelerador é criado com a instanciação e interconexão dos IPs em um arquivo Verilog único. Finalmente, é realizada a síntese em uma ferramenta de CAD FPGA (i.e., Vivado), gerando o bitstream (veja na Figura 2.4 o floor-plan do bitstream para o modelo VGG-10) para o dispositivo alvo, além de um pacote de implantação contendo o *driver* que executa no co-processador conectado a FPGA.

2.4 Visão geral

Resumimos a seção de referencial teórico com uma visão geral da abordagem tradicional para processamento das inferências em uma aplicação de classificação de RF (Figura 2.6). A abordagem tradicional (O'SHEA; CORGAN; CLANCY, 2016; O'SHEA;

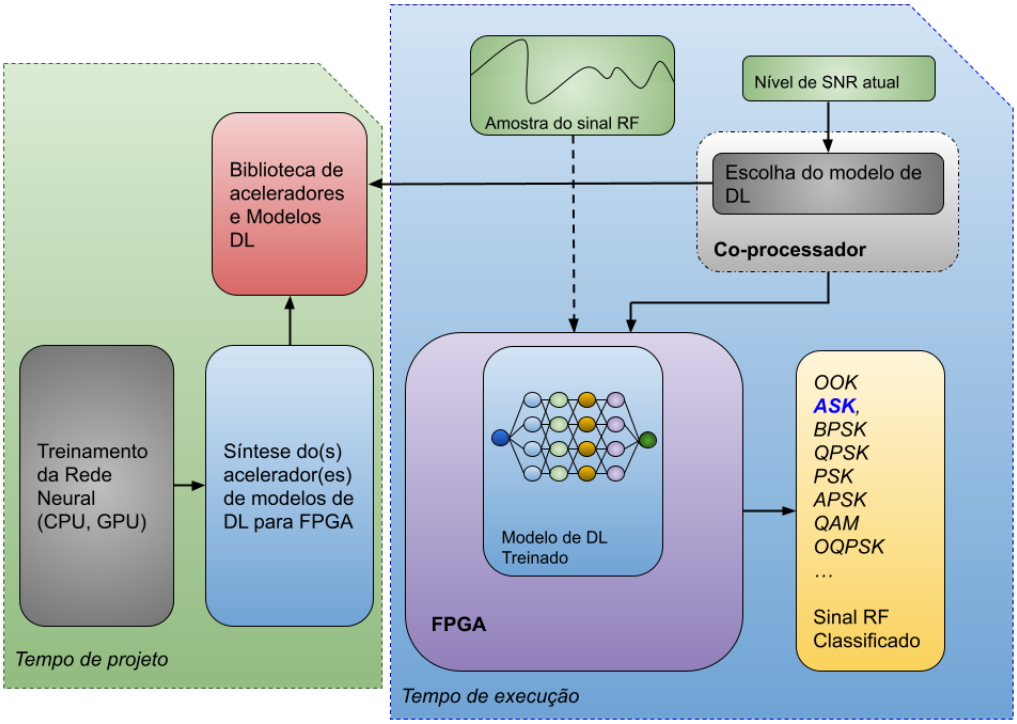
Figura 2.6: Fluxo tradicional dos classificadores RF de Redes Neurais.



Fonte: Os Autores

ROY; CLANCY, 2018; SOLTANI et al., 2019; HUANG et al., 2019; TRIDGELL et al., 2020), não-adaptativa, utiliza um único modelo de DL que foi previamente treinado em tempo de projeto para obter um desempenho satisfatório considerando uma ampla variação de SNR no sinal de entrada (*Treinamento do Modelo*). Em tempo de execução, o modelo treinado já pode executar em um *HW de propósito geral ou acelerador* para processar novas entradas. Ou seja, para cada (*Amostra do Sinal RF*) recebido como entrada pelo modelo, este infere a modulação utilizada no sinal (*Sinal RF Classificado*). A solução proposta nesse trabalho baseia-se nessa abordagem e será detalhada a seguir.

Figura 3.1: Proposta de inferência adaptativa.



Fonte: Os Autores

3 O SISTEMA DE INFERÊNCIA ADAPTATIVO

O trabalho apresenta uma solução de inferência adaptativa embarcada em FPGA, utilizando Redes Neurais para a classificação de sinais RF. Ao adaptar a inferência para diferentes situações de SNR, é possível obter um melhor equilíbrio entre energia, desempenho e custo computacional. Modelos mais simples possuem um consumo menor de energia em sacrifício da acurácia, e o contrário é esperado para modelos mais complexos (e.g., mais profundos - com mais camadas). Em situações onde a classificação é menos dificultosa pela entrada apresentar alto SNR, é possível a utilização de uma rede mais simples e, portanto, economizar energia.

A Figura 3.1 ilustra a proposta desse trabalho. A solução é separada em duas etapas principais: tempo de projeto e tempo de execução. Em tempo de projeto, é realizado o treinamento inicial da Rede Neural (ou qualquer modelo de DL) em uma CPU ou GPU. Em seguida, é realizada a síntese do(s) acelerador(es) FPGA dos modelos de DL treinados para gerar uma biblioteca de modelos. Nessa fase do projeto, a solução explora a geração através da modificação de parâmetros como o tamanho da CNN (i.e., número de canais) e quantização de diferentes modelos de CNN, oferecendo diferentes perfis de custo computacional, desempenho e consumo energético. Já em tempo de execução, a amostra do sinal RF de entrada é enviada para o modelo de DL para inferência. Concorrentemente, o nível

de SNR atual também é enviado para o módulo de escolha do modelo de DL que poderá reconfigurar a FPGA com um novo modelo/acelerador mais apropriado para a condição atual. O módulo de escolha é executado em um co-processador embarcado conectado ao FPGA. O restante da seção apresenta o desenvolvimento da solução e será apresentado em duas etapas: tempo de projeto e tempo de execução.

3.1 Tempo de Projeto

Ao início da etapa de tempo de projeto, é realizada a especificação, treinamento e validação dos modelos de DL, assim como a análise das métricas de desempenho resultantes da inferência do dataset de teste. Em seguida, são necessárias otimizações e a exportação dos modelos treinados, que são armazenados em um formato para modelos de IA compatível com o framework e compilador utilizados. Por fim, é realizada a síntese dos aceleradores, e em seguida, a análise dos relatórios de síntese e simulação resultantes.

3.1.1 Treinamento dos Modelos

As etapas de treinamento, inferência e análise das métricas de desempenho foram realizadas utilizando a infraestrutura disponibilizada pela empresa Xilinx em (UMUROGLU JENTSCH, 2021) que habilita o treino e avaliação de CNNs, utilizando os frameworks PyTorch (PYTORCH, 2022) e Brevitas (PAPPALARDO, 2021) além da biblioteca NumPy (NUMPY, 2022) em um ambiente baseado na plataforma Docker (Docker, 2022).

A infraestrutura mencionada é baseada na linguagem Python, utilizando o recurso de *notebooks* da plataforma de computação interativa Jupyter (JUPYTER, 2022). Para conter as diversas dependências de software, bibliotecas e configurações necessárias para o treinamento, além de permitir que a execução do ambiente seja reproduzível em sistemas distintos, a infraestrutura é organizada através de um *container* do software Docker.

O framework Brevitas permite o Quantization Aware Training (QAT), uma abordagem de treinamento criada para otimização da acurácia em redes limitadas por hardware. Tradicionalmente, modelos de DL são executados em GPUs ou CPUs de propósito geral utilizando números em formato de ponto flutuante com 32-bits de precisão. Em aplicações de borda ou IoT, é frequente a necessidade de redução da representação numé-

rica para inteiros de 8-bits ou menor, como os formatos binários e ternários disponíveis no Brevitas. Todo método de quantização objetiva reduzir a perda da acurácia em redes de formatos numéricos reduzidos. O treinamento pode ser realizado em CPUs ou GPUs, sendo independente da arquitetura alvo (e.g., FPGAs). Devido ao alto custo computacional, foi utilizada a aceleração por GPU suportada pelo Docker através do NVIDIA Container Toolkit (NVIDIA, 2022), realizada por uma GPU NVIDIA GTX 1050 Ti.

O fluxo da etapa de treinamento e avaliação inicia com a instanciação da GPU para a aceleração por hardware e o carregamento do dataset. Em seguida, é definido o modelo, baseado na versão quantizada da rede VGG-10 descrita na Seção 2.2.4. Utilizando o Brevitas, são instanciadas as camadas de convolução, batch normalization, quantização ReLU, quantização linear e MaxPooling, que compõem a rede VGG-10 apresentada, além de especificar seus hiperparâmetros. Os hiperparâmetros são compostos pelo valor de quantização (quantidade de bits de precisão na ativação e pesos dos neurônios), pelo número de canais das camadas convolucionais, e de neurônios nas camadas densas.

3.1.2 Síntese do Acelerador FPGA

Após o treinamento, a etapa seguinte do fluxo é a síntese dos aceleradores FPGA para cada um dos modelos do conjunto. Os modelos foram sintetizados utilizando o framework FINN, apresentado na Seção 2.3.3. O framework fornece um compilador capaz de gerar aceleradores sintetizáveis em FPGA com arquiteturas customizáveis.

A construção do acelerador e sua síntese foi realizada utilizando exemplos disponíveis no repositório (Xilinx, 2022b). Os exemplos fornecem um esqueleto do processo de síntese através do FINN, sendo possível adequá-los para as necessidades dos modelos de DL explorados e da solução proposta. Similarmente ao ambiente de treinamento, o software e dependências requeridas estão contidos em uma imagem Docker, também disponível no repositório.

O compilador requer que o modelo seja exportado no formato ONNX, reconhecido pelo FINN. O ONNX (The Linux Foundation, 2022) é um formato aberto criado para representar modelos de ML definidos por um conjunto comum de operadores. Os operadores são como blocos de construção, armazenados em um formato único que possibilita a compatibilidade com diversas ferramentas, compiladores e frameworks. As otimizações e exportação do modelo nesse formato foram realizadas através de um ambiente de imagem Docker similar ao de treinamento e de síntese, utilizando-se de scripts específicos

para tal fim e dos mesmos frameworks.

O processo de síntese tem como hardware alvo um Multiprocessor System On Chip (MPSoC) Xilinx da família Zync, modelo ZCU104. O MPSoC possui lógica programável adequada para os requisitos dos modelos, além de um co-processador ARM para aplicações (Xilinx Inc., 2022b).

Para dar início a síntese, é necessária a configuração dos parâmetros de design e a definição dos *constraints* de síntese, como períodos de *clock*, restrições de temporização e configurações de *folding*. Também podem ser necessárias algumas etapas de otimização para garantir a compatibilidade do modelo com o compilador.

Com a síntese de cada modelo concluída, se encerra o desenvolvimento da etapa de projeto. Por fim, todos os aceleradores gerados são organizados em uma biblioteca que vai dar suporte a adaptabilidade em tempo de execução.

3.2 Tempo de Execução

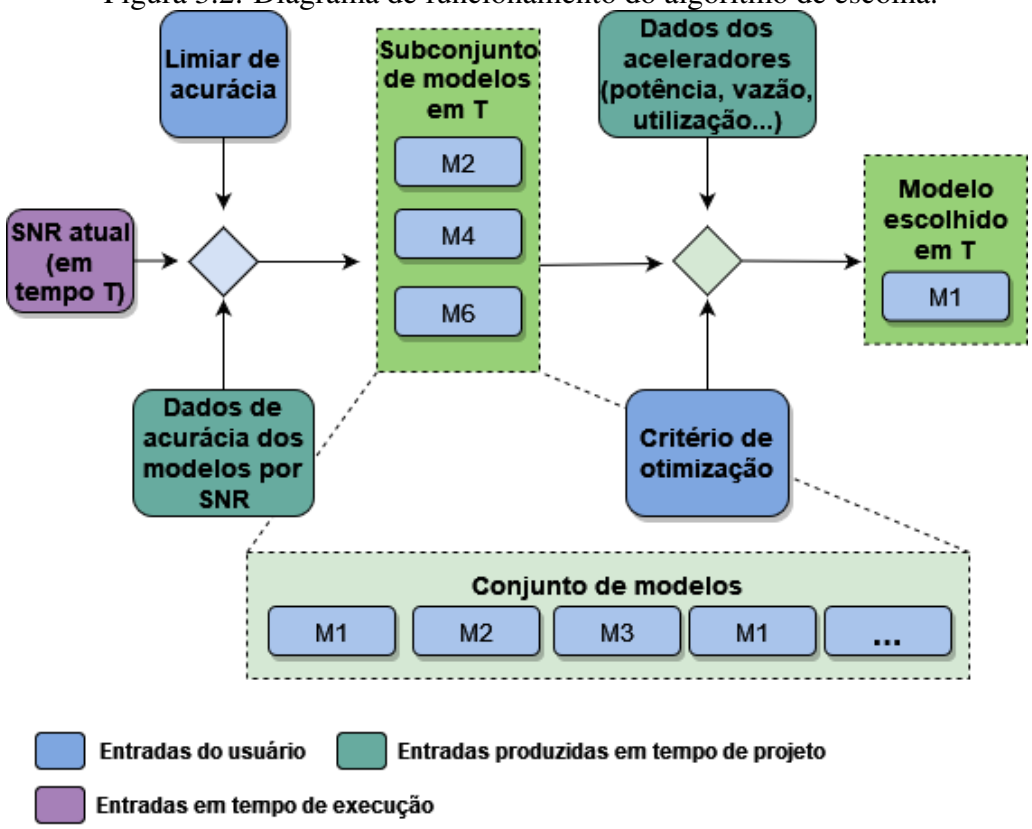
As seções seguintes irão descrever a implementação da solução referente a etapa de tempo de execução. Em especial, será apresentado o módulo de escolha do modelo de DL que adapta o processamento de inferências às condições atuais de ruido.

3.2.1 Módulo de Escolha

O módulo de escolha executa no co-processador embarcado e serve para selecionar, dentre os modelos na biblioteca criada em tempo de projeto, o modelo que melhor se adapta a condição de SNR atual. O algoritmo de escolha proposto considera o valor de SNR atual, um conjunto de entradas, a biblioteca de modelos produzida em tempo de projeto, além de dois parâmetros ajustáveis: limiar de acurácia e critério de otimização.

O limiar tem como objetivo ajustar a perda máxima de acurácia (frente a acurácia original) do modelo selecionado e impacta no grau de liberdade do algoritmo de realizar a escolha dinâmica de diferentes subconjuntos de modelos de acordo com o valor de SNR atual. Já o critério de otimização define a característica de projeto do acelerador que deve ser considerada pelo algoritmo na escolha. Neste trabalho serão abordados os critérios de potência total ou vazão. Entretanto, notamos que a escolha pode facilmente cobrir outras métricas, como potência estática, potência dinâmica, latência ou utilização de recursos

Figura 3.2: Diagrama de funcionamento do algoritmo de escolha.



Fonte: Os Autores

(e.g. LUTs, FFs, BRAMs, FIFOs). As entradas são compostas pelo valor de acurácia de cada modelo para os possíveis valores de SNR e das características de projeto dos aceleradores (e.g. potência, vazão, utilização de recursos). Como explicado anteriormente, esses dados de entrada são obtidos em tempo de projeto através dos resultados de síntese, simulação e treinamento dos modelos.

O funcionamento do algoritmo de escolha pode ser ilustrado na forma de um diagrama, apresentado na Figura 3.2. É importante notar que a escolha do modelo ocorre de forma dinâmica, de forma que o fluxo apresentado é executado para o valor de SNR atual, recebido em um tempo *t*.

Ao receber uma amostra de SNR, o algoritmo analisa o valor de acurácia no SNR atual de cada um dos modelos contidos no conjunto. São então escolhidos os modelos nos quais a queda no valor de acurácia em relação ao modelo de referência (CNN original que retorna acurácia máxima) não supera o limiar definido. Dessa forma, é formado um subconjunto de modelos que atendem esse parâmetro. Considerando esse subconjunto, a decisão pelo modelo mais adequado é tomada de acordo com o critério de otimização. O algoritmo analisa os dados de projeto dos aceleradores (e.g. potência, vazão, utilização) e escolhe o modelo que maximiza os ganhos quanto ao critério escolhido. Por exemplo, ao

definir o critério para otimização de potência, o modelo escolhido será o que possui o menor potência dentre os modelos com acurácia abaixo do limiar. Em suma, considerando o limiar de acurácia e a condição atual de ruído, o algoritmo retorna o modelo que melhor se adapta ao parâmetro de otimização escolhido. E sempre que o modelo escolhido divergir do atualmente carregado na FPGA, esta é reconfigurada com o novo modelo.

4 AVALIAÇÃO E RESULTADOS OBTIDOS

A avaliação da proposta será dividida em duas etapas. Inicialmente, na Seção 4.2 serão apresentadas as oportunidades de exploração criadas em tempo de projeto, descrevendo os modelos e os aceleradores criados. Em seguida, na Seção 4.3, o funcionamento do algoritmo de escolha dos modelos será analisado em tempo de execução.

4.1 Metodologia

O estudo de caso utilizado para classificação de RF 5G foi o dataset RadioML 2018.01A criado por (O'SHEA; ROY; CLANCY, 2018) e disponível em (DEEPSIG, 2018). O dataset contém 2 milhões de exemplos representados por quadros com um tamanho (L) de 1024 amostras. Os exemplos possuem um SNR que pode variar de -20 dB até +30 dB, e devem ser classificados dentre as 24 classes de modulação. O dataset foi preparado e dividido entre as 24 classes de modulação e os 25 valores possíveis de SNR para a análise dos resultados e a realização do treinamento. A divisão dos dados foi definida em 90% para treino e 10% para teste, aplicada uniformemente entre os pares compostos de classe e valor de SNR.

O desempenho do estado-da-arte pode ser estabelecido pelos resultados obtidos em trabalhos onde a implementação dos classificadores é feita em *software* (i.e., CPU ou GPU) - sem a utilização de aceleradores (O'SHEA; CORGAN; CLANCY, 2016; O'SHEA; ROY; CLANCY, 2018; HUANG et al., 2019); e por trabalhos como (TRIDGELL et al., 2020) que utilizam aceleradores FPGA para o dataset apresentado. A avaliação da solução proposta quanto a métricas como desempenho e acurácia tomará como referência os resultados dos trabalhos citados. Implementações em software, como o modelo VGG apresentado em (O'SHEA; ROY; CLANCY, 2018) caracterizam o desempenho do estado-da-arte quanto a acurácia onde os modelos não possuem as limitações em sua complexidade impostas pelo hardware. Implementações que utilizam aceleradores, em especial os resultados apresentados para as redes de topologia e parâmetros similares servem como referência para as métricas de acurácia e desempenho. Em (TRIDGELL et al., 2020) são apresentadas três topologias de redes VGG com 32 bits de precisão nos pesos e ativações.

Tabela 4.1: Hiperparâmetros utilizados para gerar os modelos da biblioteca.

Modelo	*Quantização*	*Canais nas camadas convolucionais*	*Neurônios nas camadas densas*
M1	2-bits	16	32
M2	2-bits	32	64
M3	2-bits	48	96
M4	2-bits	64	128
M5	4-bits	16	32
M6	4-bits	32	64
M7	4-bits	48	96
M8 (baseline)	4-bits	64	128

Fonte: Os Autores

4.1.1 Modelos

A escolha do conjunto de modelos foi guiada pelo treinamento e avaliação com o intuito de observar tendências no comportamento definidos por hiperparâmetros significativamente diferentes e o seu efeito nas métricas de desempenho, e principalmente, nos resultados de acurácia relacionada aos diferentes valores de SNR das amostras. Partindo-se da CNN proposta em (UMUROGLU JENTSCH, 2021) e considerando as limitações do FPGA e do processo de síntese, definiu-se o baseline com quantização em 4 bits e topologia de 7 camadas convolucionais de 64 canais cada, duas camadas densas de 128 neurônios e uma de 24 neurônios (esta última sempre fixa no número de classes) - veja a topologia em detalhe na Figura 2.2. A partir do baseline, foram explorados outros modelos reduzidos, modificando os hiperparâmetros de quantização e número de canais/neurônios.

Especificamente, foi definido um conjunto de oito modelos distintos, onde o número de canais diminui proporcionalmente partindo do baseline em decrementos de 25%. Além disso, o conjunto foi definido para garantir que existam redes de mesmas dimensões, porém com quantizações diferentes, o que permite que a solução explore as características e efeitos da quantização, bem como a diminuição do número de canais. O critério de escolha baseia-se em cobrir o maior conjunto possível de possibilidades dentro das limitações do impostas pelo hardware, para que o sistema adaptativo possua flexibilidade para escolher os modelos mais adequados dentro de cada um dos cenários que serão propostos adiante.

Na tabela 4.1, os modelos são organizados de acordo com sua quantização, número de canais nas camadas de convolução e número de neurônios nas camadas densas. A precisão dos pesos e ativações de cada neurônio é a mesma, sendo indicada apenas por um único valor de quantização. A duração do treinamento foi definida em 20 épocas (UMUROGLU JENTSCH, 2021), e os valores para o tamanho do *batch* (768) e a taxa de

aprendizado (0.01) foram escolhidos com o objetivo de maximizar o desempenho e evitar limitações de memória.

4.1.2 Aceleradores

Para a avaliação dos aceleradores, foram utilizadas ferramentas de simulação RTL e um conjunto de ferramentas de síntese FPGA. Os aceleradores compilados pela ferramenta FINN foram sintetizados no Vivado (XILINX, 2022a) com um período de 4 ns, equivalente a uma frequência de 250 MHz, estabelecendo uma comparação justa com o estado-da-arte (TRIDGELL et al., 2020). Para a correta síntese também foram necessárias as seguintes alterações: adição de um nodo TopK, com k = 1 na saída, inserção do tipo dos dados de entrada, e uma modificação nas dimensões dos tensores, de 3D para 4D. As modificações foram feitas através de um script Python adicional, utilizando recursos do FINN.

Quanto a configuração de *folding* do FINN (que define o nível de paralelismo via valores de PE e SIMD, ver Seção 2.3.3), estes foram ajustados para o máximo denominador comum entre todos modelos treinados, possibilitando que todos fossem sintetizados corretamente e garantindo o mesmo nível de paralelismo entre eles (i.e. com a mesma configuração de *folding*). Com uma configuração única, possibilitamos uma melhor avaliação dos hiperparâmetros, isolando os ganhos de quantização e tamanho do modelos de eventuais diferenças em paralelismo dos aceleradores. Os resultados de potência e utilização de recursos do FPGA foram gerados após a conclusão da síntese pelo Vivado. Todos os modelos escolhidos foram sintetizados com sucesso, respeitando as restrições de temporização e limitações do hardware. A simulação RTL dos aceleradores obtidos é realizada na ferramenta PyVerilator (VERIPOOL, 2022) que fornece os dados de desempenho (e.g. vazão e latência) necessários.

4.1.3 Simulador

Um ambiente de avaliação foi criado com o intuito de verificar e simular o comportamento da proposta em tempo de execução. Para o desenvolvimento do simulador, foi criado um *script* em linguagem Python. O script é responsável por modelar o funcionamento da solução, executando o algoritmo de escolha e salvando os resultados calculados.

A biblioteca de modelos criada é organizada em duas tabelas, a primeira contendo os valores de acurácia obtidos no treinamento, e a segunda contendo as características dos aceleradores. O limiar de acurácia e o critério de otimização são definidos ao início da execução do simulador. A simulação ocorre com um dataset que contém diversas amostras de valores de SNR que variam ao longo do tempo. Dessa forma, é possível simular o funcionamento dinâmico da proposta.

Os modelos escolhidos pelo algoritmo para cada variação de SNR ao longo do tempo são armazenados de forma ordenada e organizados em pares no formato (S, M) contendo o valor de SNR da amostra (S) e o modelo (M) para cada tempo *t*. Dessa forma, o comportamento do algoritmo ao longo do tempo é registrado, possibilitando a sua análise posterior. Através desses dados é possível relacionar o comportamento do algoritmo com os padrões de variação e diferenças de acurácia entre os modelos mencionados anteriormente.

A simulação da solução em tempo de execução e os registros da execução do algoritmo ao longo do tempo permitem calcular o impacto de diferentes valores de limiar e critérios de otimização distintos na otimização. É possível, por exemplo, calcular o consumo energético e o desempenho do algoritmo ao longo do dataset, através dos dados de execução e resultados de potência e vazão dos aceleradores. Através desses resultados, foi possível desenvolver e validar o funcionamento do algoritmo.

O simulador também considera os custos de reconfiguração da FPGA, compostos pelo tempo e potência necessários para sua execução. Dadas as características do FPGA alvo e dos modelos, o tempo de reconfiguração necessário é de 145 ms (DUHEM; MULLER; LORENZINI, 2011), sendo o custo energético definido pelo valor de potência estática do modelo.

4.1.4 Cenário de Avaliação

Para refletir um cenário fiel em tempo de execução de uma aplicação RF, baseamos nossa avaliação no estudo apresentado em (HOOFT et al., 2016), que realizou a medição da qualidade de uma conexão 4G/LTE. Hooft et al. realizaram diversos registros da condição da conexão ao longo de diferentes percursos em uma cidade com o intuito de avaliar uma transmissão de vídeo em redes móveis. O conjunto de registros forma um dataset que contém a vazão da conexão (em Mb/s), o tempo decorrido entre cada uma das amostras (que varia de 700 a 1000 ms), o tempo total e o *timestamp* de cada amostra.

O percurso escolhido para avaliação foi realizado através de um veículo, durante 468 segundos. O dispositivo inicia o trajeto passando por uma área densa em edificações, reduzindo o desempenho da conexão. Com o aumento da distância entre o ponto de acesso (antena) que está conectado e os obstáculos presentes, o desempenho é reduzido até ocorrer a perda da conexão. Em seguida, o dispositivo troca para outra antena com maior cobertura, e a conexão é restabelecida. Finalmente, a medida que o veículo se aproxima de uma área aberta, sem obstáculos, e com a presença de outra antena mais próxima, o desempenho aumenta significativamente até atingir seu máximo ao fim do percurso.

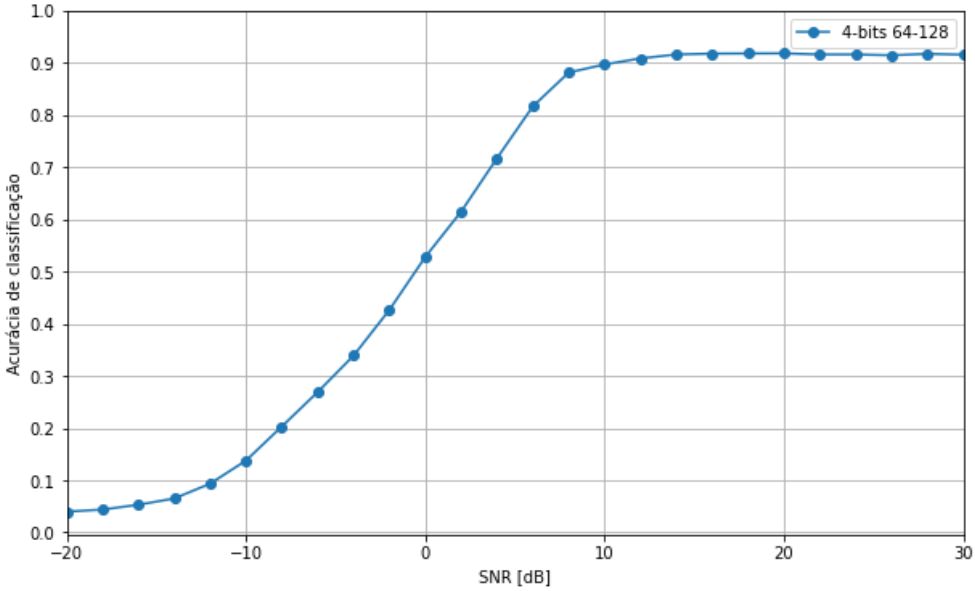
Para utilizar o dataset no nosso trabalho, foi necessário relacionar os valores de vazão a condição da conexão em termos de SNR. Para tal, utilizamos o teorema de Shannon-Hartley (SHANNON, 1949) que define a taxa máxima na qual dados podem ser transmitidos através de um canal de comunicação, considerando sua largura de banda e a presença de ruído. De maneira geral, podemos definir a capacidade de um canal ruidoso segundo a equação $C = B * log_2(1 + SNR)$, onde C é a capacidade do canal em bits/segundo e B é a largura de banda em Hertz.

Aplicando o teorema, podemos calcular o valor de SNR correspondente a vazão para cada amostra, definindo a largura de banda apropriada e normalizando os resultados de acordo com o valor de vazão correspondente aos valores de máximos e mínimos definidos pelo dataset de treinamento. Dessa forma, temos os valores de SNR ao longo do tempo para cada percurso do dataset em (HOOFT et al., 2016).

Quanto a taxa de amostragem de RF a qual a o sistema vai ser simulado, iremos considerar duas opções do estado-da-arte onde os autores definiram taxas distintas (TRIDGELL et al., 2020; IVERSEN et al., 2006). Considerando essas referências, foram escolhidos dois valores - 50 e 100 MHz.

A taxa de amostragem representa quantas inferências por segundo precisam ser processadas, onde cada inferência lê um quadro de L amostras. Para uma taxa de amostragem de 50MHz, e uma entrada com L=1024 amostras, temos uma taxa de 48.828 inferências por segundo. Já para 100 MHz, temos 97.556 classificações por segundo. Ao introduzir esse aspecto para avaliar o desempenho da proposta, podemos definir a sua capacidade de operar em um cenário de carga de trabalho real onde é desejável minimizar a perda de amostras.

Figura 4.1: Acurácia do baseline em relação ao SNR.



Fonte: Os Autores

4.2 Oportunidades para Exploração em Tempo de Projeto

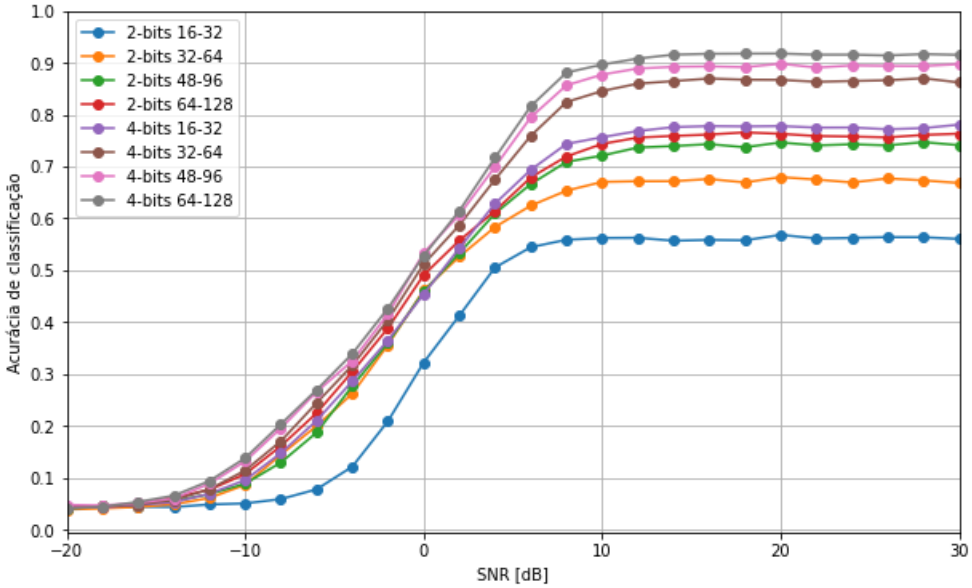
Nas seções seguintes iremos abordar as variações possíveis entre os modelos definidos anteriormente e os aceleradores sintetizados. Essas variações representam as oportunidades para a adaptação do algoritmo em tempo de execução.

4.2.1 Modelos

Quanto ao Baseline. A Figura 4.1 apresenta a relação entre os valores de acurácia (eixo y) e SNR (em decibel, eixo x) para o baseline (modelo M8) apresentado na Seção 3.1.1. Os valores de SNR possíveis variam de -20 a +30 dB em incrementos de 2 dB. O gráfico pode ser dividido visualmente em três intervalos distintos ao longo do eixo X. O primeiro intervalo abrange os valores entre -20 e -10 dB SNR, o segundo entre -10 e +10 dB SNR, e por fim, o terceiro, de +10 até +30 db SNR.

O terceiro intervalo demonstra uma acurácia praticamente constante, onde os valores de SNR são considerados altos. Como visto anteriormente, isso ocorre devido a relação entre a quantidade de sinal-ruido da amostra conter uma parcela comparativamente menor de ruido, facilitando a identificação do modulador, e consequentemente, aumentando o desempenho do classificador. Podemos concluir que o modelo apresenta seu desempenho máximo ao classificar amostras que apresentam um SNR superior a +10

Figura 4.2: Acurácia de classificação dos modelos em relação ao SNR.



Fonte: Os Autores

dB.

Em contrapartida, o primeiro intervalo apresenta valores de SNR considerados significativamente baixos, onde a amostra está submetida a níveis extremos de ruido. Nesse cenário, o classificador apresenta valores de acurácia consideravelmente menores. Podemos considerar que o classificador não obtém um desempenho satisfatório ao inferir amostras com valores de SNR contidos nesse intervalo. No entanto, essa característica é comum aos classificadores estado-da-arte mencionados nos trabalhos relacionados.

O segundo intervalo, onde estão presentes os valores de SNR intermediários, demonstra uma rápida transição no desempenho do classificador a medida que a potência do sinal se torna maior do que a potência do ruído. Essa relação entre as potências é demonstrada por valores superiores a 0 dB SNR. Quando submetido a essas condições, o classificador atinge rapidamente valores próximos ao seu desempenho máximo.

Quanto aos Modelos Disponíveis. Agora podemos comparar com o baseline a influência dos hiperparâmetros de quantização e do número de canais utilizando a mesma relação acurácia-SNR descrita anteriormente. A Figura 4.2 ilustra graficamente essa influência. O baseline é representado pela curva indicada como 4-bits 64-128. A legenda (X-bits Y-Z) indica a quantização de X bits (2 ou 4), o número de Y canais nas camadas de convolução e Z neurônios nas camadas densas.

Podemos verificar inicialmente que o comportamento geral da curva de acurácia é semelhante entre os modelos, seguindo o comportamento estabelecido pelo baseline. Ao comparar os modelos de 4 bits, é possível visualizar de forma isolada o efeito da

redução do número de canais. Neste caso, nota-se que o impacto na redução da acurácia é mais significativo em regiões de SNR acima de 0 dB, onde a potência do sinal é superior ao ruido. Analogamente, em regiões de SNR negativo, a redução é menos acentuada. Ao comparar os modelos com a mesma quantidade de canais (e.g. 4-bits 16-32 vs. 2-bits 16-32), é possível perceber a influência da redução na quantização. Similarmente a comparação anterior, a queda na acurácia segue o mesmo comportamento, sendo ainda mais acentuada em valores de SNR positivos. Então, podemos concluir que o impacto na acurácia é mais significativo ao reduzir a quantização em comparação a redução no número de canais.

Mais especificamente, ao analisar a região entre +10 e +30 dB SNR, podemos perceber visualmente a formação de dois conjuntos de modelos cujas acurácias possuem uma variação menor do que 10%. Ambos os conjuntos possuem em comum o valor de quantização, quatro e dois bits, respectivamente, sendo o primeiro formado pelos modelos M6 (4-bits 32-64), M7 (4-bits 48-96) e M8 (4-bits 64-128), e o segundo formado pelos modelos M3, M4 e M5. Ao comparar o primeiro conjunto (composto por M6, M7 e M8) com o modelo M5 (4-bits 16-32), podemos perceber uma queda significativa na acurácia, mesmo possuindo o mesmo valor de quantização. Portanto, é possível concluir que o impacto da redução no número de canais é significativamente maior quando ela é igual ou superior a 75% do tamanho base, considerando uma quantização de quatro bits.

O segundo grupo possui modelos com diferentes quantizações, e valores de acurácia ainda mais próximos se comparado ao primeiro. Podemos concluir que o impacto na acurácia é semelhante ao comparar a redução em 75% do número de canais com a redução da precisão dos pesos e ativações pela metade. Considerando os modelos com quantização de dois bits, o efeito de redução na acurácia visto anteriormente ao reduzir o número de canais ocorre com uma redução de 50% do tamanho base, sendo ainda mais acentuado quando o modelo é reduzido em 75%.

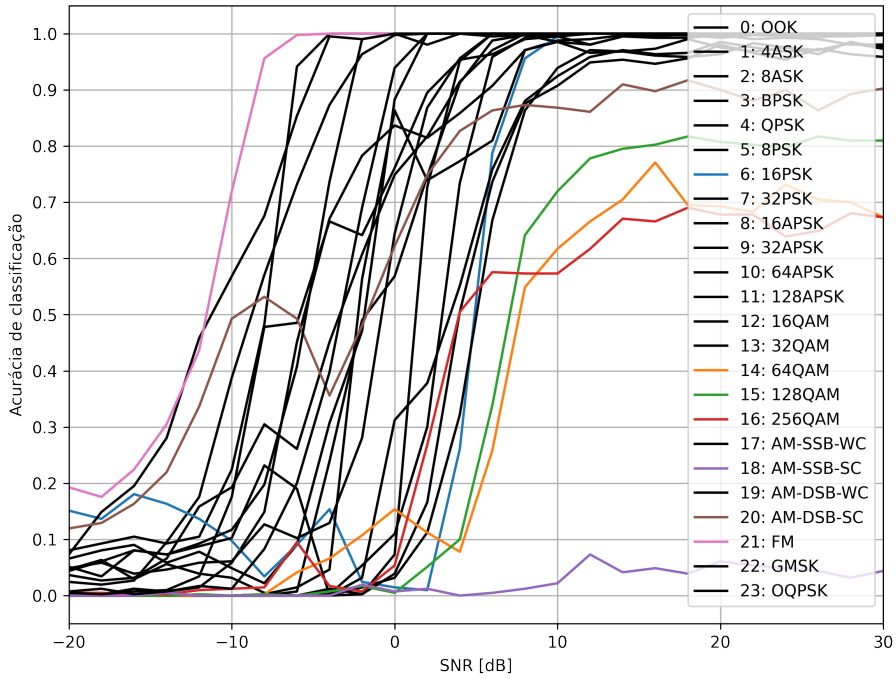
A região entre -10 dB e 0 dB SNR demonstra uma redução significativa nas diferenças de acurácia entre todos os modelos, sendo de aproximadamente 10% em seu máximo. Apesar da acurácia semelhante, as estatísticas de consumo e utilização de recursos que serão vistas adiante, caracterizarão os modelos de forma mais completa. Ao classificar amostras que apresentam magnitudes de SNR significativamente baixas, todos os modelos tendem a convergir para o mesmo valor de acurácia à medida que elas se aproximam de -20 dB SNR. Como esperado, o desempenho de nenhum dos classificadores é satisfatório nessa região.

Tabela 4.2: Valores de acurácia de classificação dos modelos da biblioteca.

Modelo	*Acurácia média*	*Acurácia em +30 dB SNR*	*Acurácia em -20 dB SNR*
M1	0.355859	0.560061	0.043801
M2	0.443203	0.667988	0.038313
M3	0.477036	0.740854	0.040244
M4	0.494524	0.763008	0.042480
M5	0.497088	0.780488	0.041870
M6	0.552709	0.861789	0.041972
M7	0.573952	0.897154	0.047256
M8	0.587441	0.915041	0.039837

Fonte: Os Autores

Figura 4.3: Acurácia por classes de modulação do modelo M8 (baseline).

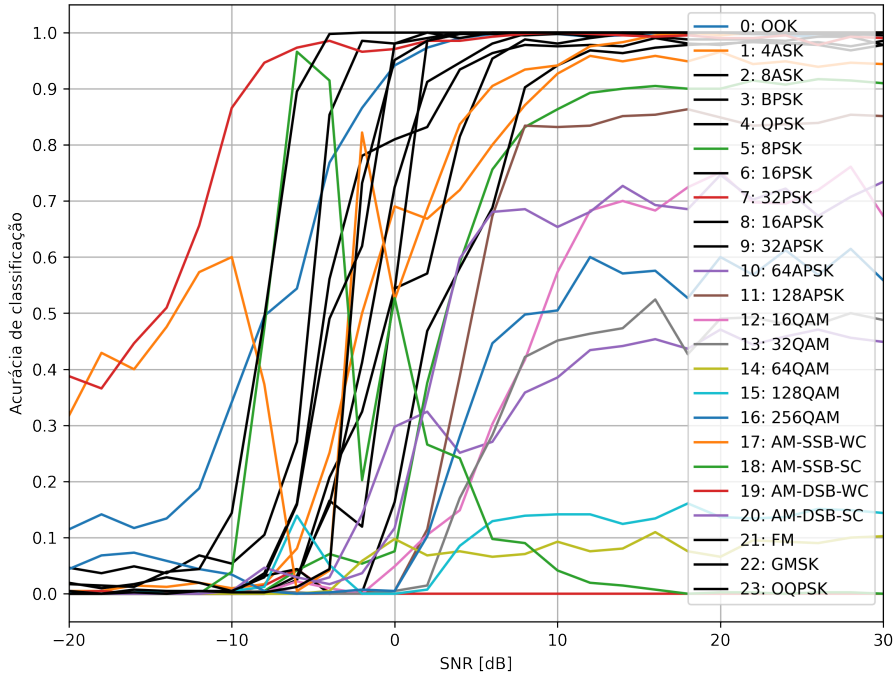


Fonte: Os Autores

A Tabela 4.2 apresenta os valores de acurácia média (ao longo de toda faixa de SNR) e acurácia nos SNRs extremos, permitindo visualizar as diferenças numéricas em maior detalhe e fornecendo uma visão geral do desempenho de cada modelo.

Além do desempenho caracterizado pela relação de acurácia-SNR, podemos avaliar os modelos de acordo com outras métricas, como a matriz de confusão e os valores de acurácia para cada uma das 24 classes de modulação. Com o intuito de observar de forma geral o impacto na acurácia visto anteriormente, agora utilizando outras métricas, podemos comparar o baseline a um modelo intermediário, que possui menor quantização e número de canais. As Figuras 4.3 e 4.4, apresentam as acurácias por classe para o baseline e modelo M3, respectivamente. E as Figuras 4.5 e 4.6 apresentam as matrizes de confusão para os mesmos modelos.

Figura 4.4: Acurácia por classes de modulação do modelo M3.

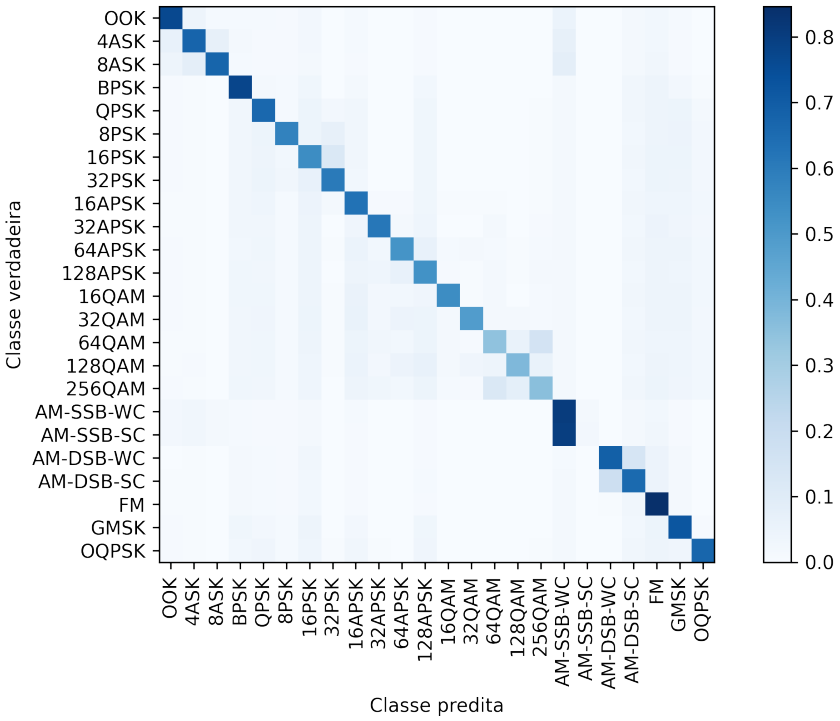


Fonte: Os Autores

A acurácia por modulação oferece uma perspectiva diferente para caracterizar o desempenho do classificador, introduzindo outra variável. Por essa perspectiva, é possível verificar que o seu desempenho não depende somente do valor de SNR, mas também do tipo de modulação a ser detectado. Por exemplo, na Figura 4.3, temos que entre aproximadamente -8 e +10 dB SNR, o classificador baseline apresenta uma variação ampla em seu desempenho, onde modulações tradicionais como AM (Amplitude Modulation) e FM (Frequency Modulation) são detectadas mais facilmente até mesmo em condições de SNR negativo, onde o ruido possui maior potência do que o sinal. Analogamente, modulações de alta ordem, que empregam técnicas mais elaboradas como Phase Shift Keying (PSK) e Quadrature Amplitude Modulation (QAM) requerem valores positivos de SNR para obter um desempenho aceitável e serem distinguidas. É importante notar que em situações onde os valores de SNR estão acima de +10 dB a grande maioria das modulações são identificadas corretamente, com acurácias acima de 80% para o baseline.

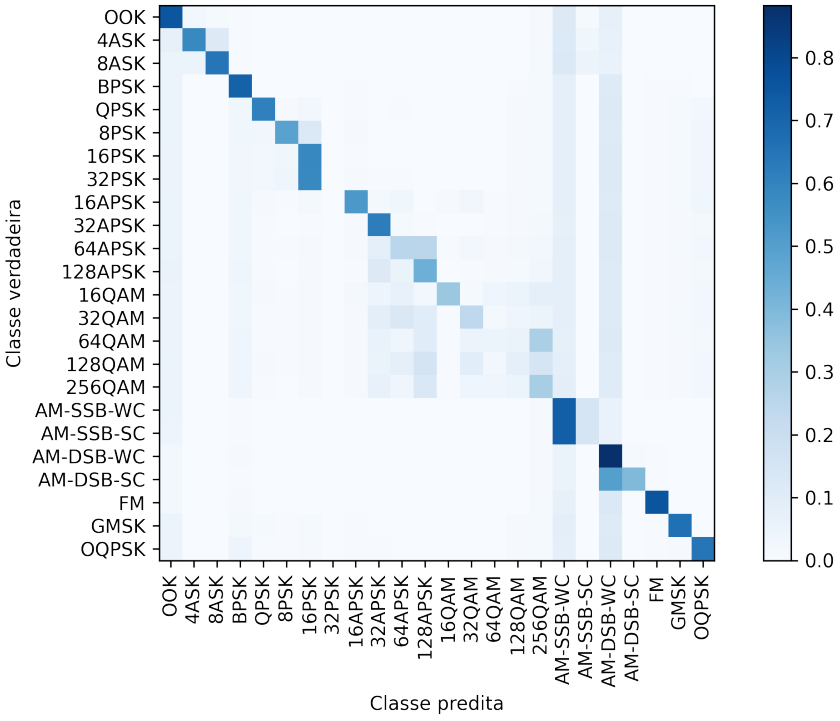
Ao comparar o M3 (2 bits 48-96) com o baseline, podemos verificar que o maior impacto na redução de acurácia ocorre ao identificar modulações de alta ordem como 32-PSK e 64/128/256-QAM (Figura 4.4). Em classes onde o baseline apresentava maior dificuldade (64/128/256-QAM) o impacto na redução de acurácia é significativamente maior, onde o modelo M3 passa a não apresentar um desempenho satisfatório, independente das condições de SNR. No entanto, o padrão de acurácia para modulações de baixa

Figura 4.5: Matriz de confusão para o modelo M8 (baseline).



Fonte: Os Autores

Figura 4.6: Matriz de confusão para o modelo M3.



Fonte: Os Autores

ordem se manteve, com uma grande parte delas obtendo acurácias acima de 70% em valores superiores a +10 dB SNR.

A matriz de confusão das figuras 4.5 e 4.6 mostram as acurácias por modulação

Tabela 4.3: Características de potência e desempenho.

Modelo	*Potência total (W)*	*Potência din. (W)*	*Potência est. (W)*	*Vazão (class/s)*	*Latência (ciclos)*
M1	0.716	0.124	0.592	238790	1688
M2	0.841	0.248	0.593	53753	15498
M3	0.891	0.297	0.593	35167	30545
M4	0.948	0.354	0.594	26141	52569
M5	0.876	0.286	0.593	238755	1676
M6	1.059	0.465	0.594	53752	15499
M7	1.245	0.649	0.596	35167	30546
M8	1.416	0.819	0.597	26141	52570

Fonte: Os Autores

Tabela 4.4: Características de utilização de recursos.

Modelo (util.)	*CLBs (util.)*	*LUTs (util.)*	*FFs (util.)*	*BRAMs*
M1	4109 (14.27%)	16355 (7.10%)	21769 (4.72%)	1 (0.32%)
M2	7670 (26.63%)	32407 (14.07%)	41073 (8.91%)	10 (3.21%)
M3	8277 (28.74%)	33994 (14.75%)	44506 (9.66%)	22.5 (7.21%)
M4	8593 (29.84%)	35161 (15.26%)	47724 (10.36%)	39 (12.5%)
M5	9900 (34.38%)	45755 (19.86%)	41458 (9.00%)	2 (0.66%)
M6	16511 (57.33%)	84738 (36.78%)	71565 (15.53%)	22 (7.05%)
M7	16477 (57.21%)	85313 (37.03%)	77485 (16.82%)	35 (11.22%)
M8	17787 (61.76%)	87654 (38.04%)	81349 (17.65%)	83 (26.6%)

Fonte: Os Autores

considerando valores de SNR iguais ou superiores a zero. Ao estabelecer a correlação entre a classe predita, classe verdadeira e a acurácia (representada pela escala de cores) é possível identificar as classes que são falsamente identificadas como outras, e a probabilidade dessas ocorrências. Como esperado, as modulações mais complexas e de alta ordem possuem maior probabilidade de serem confundidas.

Ao comparar a matriz de confusão do baseline com o M3, verificamos o mesmo impacto visto anteriormente. Na região da matriz onde estão presentes as modulações mais complexas, ocorre um aumento nas ocorrências de predição incorreta nas classes próximas, verificado pelo aumento dos valores de acurácia nessa região.

4.2.2 Aceleradores

A Tabela 4.3 apresenta os valores de potência, latência e vazão dos aceleradores sintetizados na biblioteca. Em seguida, a Tabela 4.4 apresenta os dados de utilização de recursos da FPGA, em números totais e percentuais de utilização. Os aceleradores apre-

sentados seguem um comportamento esperado quanto as suas especificações e modelos. Temos que a utilização de recursos e potência aumenta proporcionalmente a medida que o número de canais cresce e o número de bits de quantização aumenta. Isso ocorre pois os requisitos de memória e quantidade de unidades lógicas são maiores, devido ao aumento da precisão na representação dos pesos e ativações, assim como na profundidade da rede.

Podemos retomar as comparações entre o número de canais e a quantização feitas na seção anterior, agora considerando as características de projeto dos aceleradores. Ao comparar os pares de modelos com a mesma quantidade de canais (e.g M8-M4, M7-M3) verificamos que a vazão e a latência não são impactadas pela redução na quantização. Isso ocorre pois a quantização não interfere no grau de paralelismo do acelerador. Como todos os modelos possuem mesma topologia (número de camadas convolucionais), todos os aceleradores possuem os mesmos estágios de pipeline, configurados da mesma forma (número de PEs e SIMD). Por outro lado, a quantização proporciona uma redução considerável de potência, de aproximadamente 33%, ao comparar o baseline (M8) com seu equivalente quantizado em 2 bits (M4). Além disso, é notável o impacto na utilização de recursos (veja as diferenças de percentual ao comparar modelos de mesmo número de canais na Tabela 4.4). Em contrapartida, como visto anteriormente, a quantização impõe custos significativos em acurácia, oferecendo diferentes compromissos entre as métricas.

4.2.3 Conclusões

Na Seção 4.2.1 foi apresentado o conjunto de modelos que constituem a biblioteca produzida na primeira fase da solução, em tempo de projeto. As diferentes versões de modelo oferecem um conjunto de possibilidades para otimizar o compromisso entre a acurácia e a complexidade da rede (ditada pela quantização e o número de canais) ao longo dos valores de SNR. Os aceleradores sintetizados, apresentados na Seção 4.2.2, consequentemente possuem diferentes características de projeto, como potência, desempenho e utilização de recursos, que estão diretamente relacionadas a complexidade dos modelos. A partir dessas diferenças, é possível adaptar a solução conforme a variação do SNR, através de um conjunto de aceleradores que possui diferentes características de acurácia e de projeto. Relacionando as diferentes características apresentadas nas seções anteriores (e.g. potência e acurácia), é possível prever a adaptabilidade do modelo e seu comportamento em cenários de otimização de consumo energético. Similarmente, para a acurácia e latência, por exemplo, é possível analisar a adaptabilidade em cenários de

otimização de desempenho. Essas relações são importantes para interpretar os resultados e verificar o funcionamento da proposta em tempo de execução nas seções seguintes.

4.3 Adaptabilidade em Tempo de Execução

Esta seção irá avaliar a adaptabilidade da proposta em tempo de execução. Inicialmente, proposta e baseline serão comparados quanto acurácia, energia, EDP, e vazão. Em seguida, iremos contextualizar os ganhos apresentados segundo a aplicação. Os resultados seguintes utilizam o simulador apresentado na Seção 4.1.3 que modela o sistema adaptativo em tempo de execução, segundo o cenário apresentado na Seção 4.1.4.

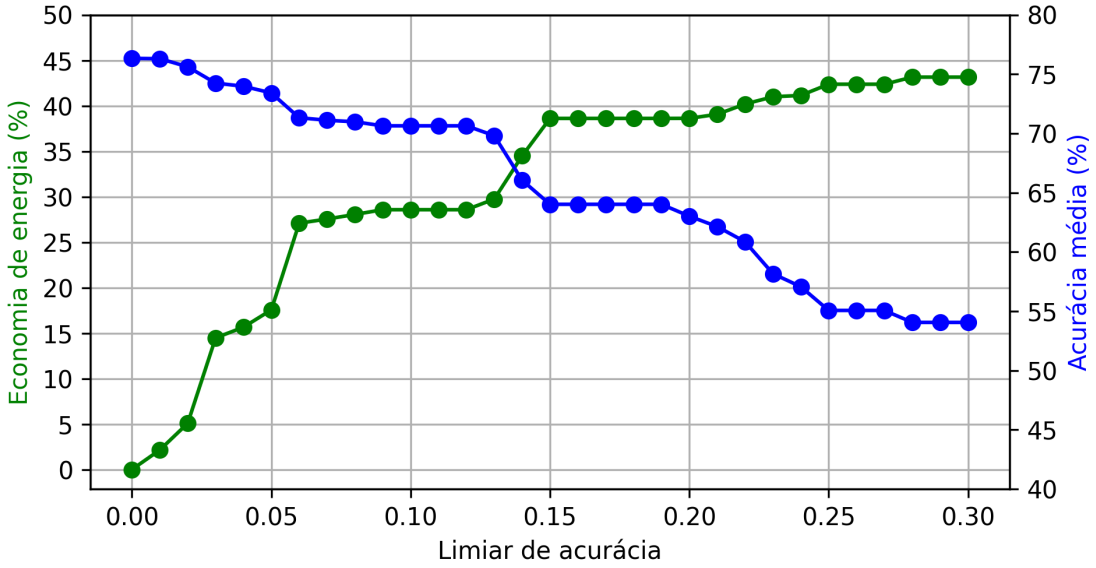
4.3.1 Energia

A Figura 4.7 mostra a economia de energia (eixo y da esquerda) frente ao baseline para valores de limiar acurácia que variam de 0 a 30%, em incrementos de 1% (eixo x, de 0.00 até 0.30). Os valores abrangem ambos extremos, explorando o comportamento do baseline, isto é, sem a troca de modelos (limiar igual a zero, modelo M8 - ver Tabela 4.1), até situações onde o limiar garante alta liberdade de otimização para a escolha do modelo. Além disso, a figura apresenta a acurácia média (eixo y da direita) ao longo do percurso, possibilitando visualizar o compromisso entre os ganhos de economia e a acurácia.

Notamos que a redução do consumo energético varia de aproximadamente 2%, com o limiar de acurácia em 1%, até atingir um máximo de 43% com um limiar igual a 28%. Mesmo em valores de limiar baixos, menores que 5%, o algoritmo já possui grau de liberdade suficiente para demonstrar ganhos, mesmo utilizando uma variedade comparativamente pequena de modelos ao longo do percurso. Isso ocorre pois os modelos que possuem uma queda de acurácia pequena em relação ao baseline, já demonstram uma redução de potência significativa.

Podemos perceber também um aumento significativo de economia ao atingir um limiar igual a 3%, atingindo 14.5%, enquanto a acurácia média sofre uma redução de apenas 2%. O mesmo ocorre entre os limiares de 5% e 6%. Em ambos os pontos, um pequeno aumento no limiar possibilita que novos modelos, que possuem menor potência, sejam escolhidos para mais níveis de SNR. Concorrentemente, o aumento proporciona uma maior otimização ao longo do percurso, pois o limiar começa a superar as quedas de

Figura 4.7: Economia de energia frente ao baseline para limiares de acurácia de 0 a 30%.



Fonte: Os Autores

acurácia em mais regiões. O aumento no conjunto de modelos possíveis e a possibilidade de otimização em mais pontos do percurso proporciona o aumento instantâneo visto no gráfico. O efeito contrário ocorre em regiões onde o aumento do limiar não possibilita que novos modelos sejam escolhidos, onde a economia de energia se mantém constante. Nesse caso, a queda de acurácia também se mantém constante.

Em valores acima de 20% os ganhos em economia de energia tendem a crescer em menor quantidade para cada incremento no limiar. Nessa situação o algoritmo possui um alto grau de liberdade para a escolha, tendendo a escolher os modelos menos complexos do conjunto (i.e. M1 e M2) em mais pontos do percurso. A medida que o limiar se aproxima de 30%, o algoritmo tende a efetuar mais trocas, priorizando o modelo de menor potência entre eles. A preferência por esses modelos ocasiona uma constante queda na acurácia média, ao mesmo tempo que o aumento na quantidade de trocas recua o aumento na economia de energia devido ao seu custo.

O melhor compromisso entre economia e acurácia ocorre até o limiar atingir 15%. Nessa região, os decrementos em acurácia são comparativamente menores do que os incrementos em economia, sendo de 10% a maior queda de acurácia média, ao mesmo tempo que o percentual de economia atinge valores de até 40%. Isso ocorre pois ao longo desses valores de limiar, o algoritmo utiliza um conjunto adequado de modelos distintos ao longo de todos os níveis de SNR, não sendo restrito por um limiar baixo ou apresentando o comportamento descrito anteriormente em limiares altos.

A utilização da solução adaptativa se mostra vantajosa em relação a abordagem estática tradicional, especialmente em cenários onde o consumo energético é um aspecto de projeto importante, como em aplicações de classificadores em dispositivos de borda (WANG et al., 2020). É obtido um equilíbrio satisfatório entre a economia de energia e o seu impacto na acurácia, mesmo em situações onde se permite uma pequena margem de compromisso ao desempenho do classificador. Restringindo o impacto na acurácia média a 5%, foi possível obter uma economia de até 30% de energia utilizando a solução proposta.

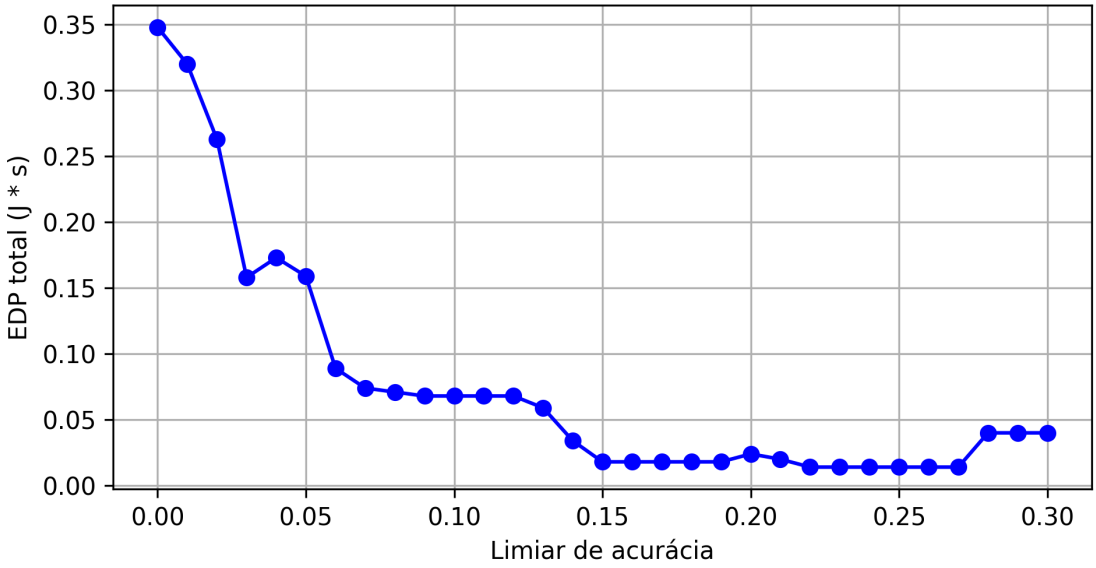
4.3.2 EDP

O Energy Delay Product (EDP) é uma métrica comumente utilizada para representar o compromisso entre desempenho (e.g. atraso) e energia (LAROS et al., 2013). Ao relacionar a energia consumida com o atraso é possível determinar a eficiência da solução para cada incremento no limiar. Uma diminuição nos valores de EDP acumulados ao longo do trajeto representa uma melhora, e demonstra que a redução de energia vista anteriormente é acompanhada de uma diminuição na latência, caracterizando um aumento na eficiência. A Figura 4.8 apresenta o EDP total acumulado ao longo do trajeto para limiares de acurácia de 0% (baseline) a 30%. Neste caso, o algoritmo foi executado com o critério de otimização definido para potência.

Analisando a figura, podemos verificar que a maior queda no valor de EDP ocorre entre 1% e 3% de acurácia. Essa queda pode ser verificada na Figura 4.7 onde ocorre um aumento significativo na economia de energia. A redução no EDP demonstra que a otimização de energia mostrada anteriormente foi acompanhada por uma redução na latência. Em seguida, é apresentado um leve aumento onde o limiar é igual a 4%. Neste caso, considerando o critério de otimização para potência, a diferença de comportamento se dá pela diferença de energia entre modelos que possuem latências similares (e.g M2 e M6), ocasionando um aumento no valor de EDP caso o modelo de maior potência seja escolhido.

A análise do EDP apoia os resultados de energia apresentados anteriormente, e demonstra a capacidade de otimização da solução ilustrada pela queda dos valores para cada incremento do limiar.

Figura 4.8: EDP da proposta adaptativa para limiares de acurácia de 0 a 30%.



Fonte: Os Autores

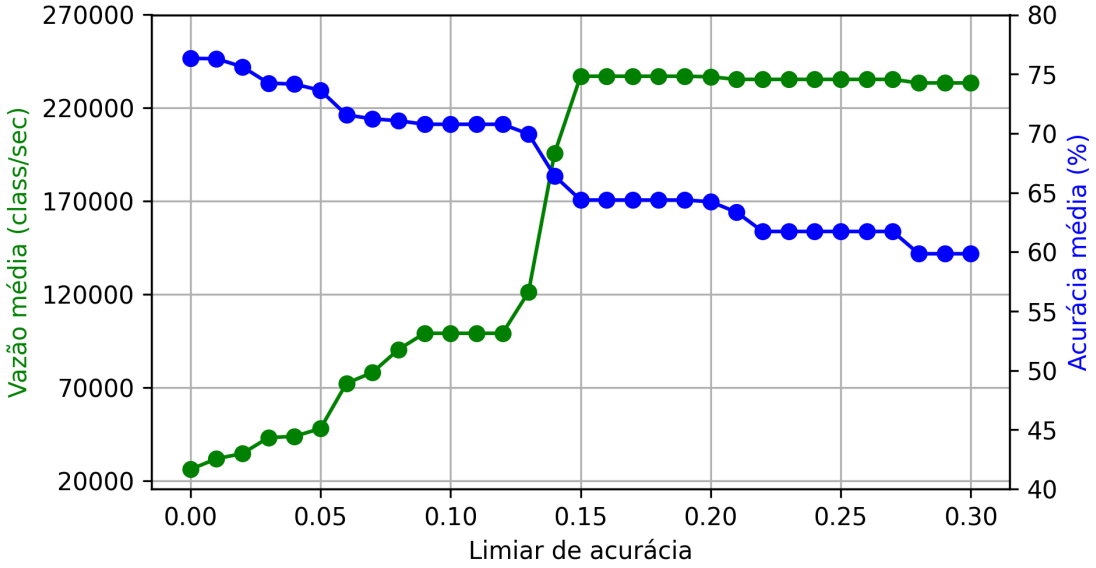
4.3.3 Vazão

Similarmente aos resultados para energia, a Figura 4.9 apresenta os valores de vazão média (eixo y da esquerda) e acurácia (eixo y da direita) ao longo do percurso, para limiares de 0 a 30%. Como ponto de partida, temos a vazão média do baseline, capaz de realizar aproximadamente 26.000 classificações por segundo. Analisando os limiares iniciais, a solução adaptativa apresenta um aumento na vazão mesmo em limiares baixos, similar ao que ocorre com a energia. Esse comportamento é esperado, pois os modelos que apresentam redução de potência também apresentam um aumento de vazão significativo. A vazão média aumenta para aproximadamente 43.000 classificações por segundo com um limiar igual a 3%, apresentando uma redução na acurácia média igual a 2%. Aumentando o limiar até 9% é possível obter ganhos significativos, onde a média aumenta para aproximadamente 99.000 classificações por segundo, com uma queda de acurácia média de 6%.

A aumento de vazão apresentado entre os limiares de 13% e 15% é substancial, rapidamente atingindo o valor máximo. A partir desses valores, o algoritmo é permitido a incluir os modelos M5 e M1, que apresentam ganhos substanciais em vazão. Esses modelos são escolhidos em faixas intermediárias e baixas de SNR, onde a diferença de acurácia entre os modelos é menor.

Diferentemente do que ocorre com a energia, o algoritmo atinge o ganho máximo

Figura 4.9: Vazão média da solução adaptativa em relação ao limiar de acurácia.

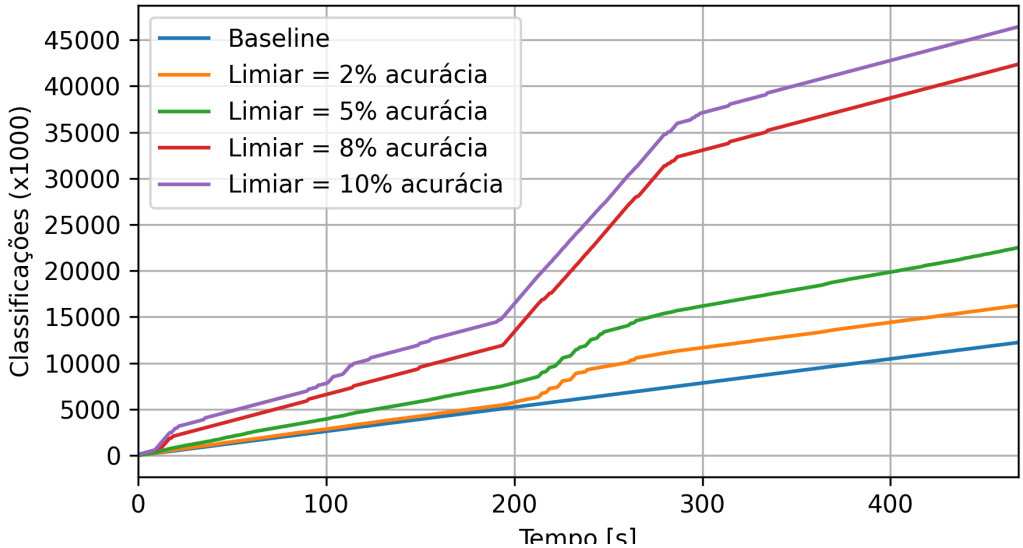


Fonte: Os Autores

de otimização em um limiar significativamente menor, já em 15%. Ao analisar as características de projeto, vimos anteriormente que a vazão é indiferente quanto a quantização, onde os modelos de 4 bits apresentam valores de vazão praticamente idênticos aos de 2 bits. Neste ponto, o algoritmo é permitido a escolher os modelos M7, M6 e M5, que possuem valores de vazão extremamente próximos aos modelos M3, M2 e M1. Mesmo com os incrementos subsequentes no limiar, permitindo que sejam escolhidos os modelos com 2 bits de quantização, o valor de vazão se mantém, já que eles que não apresentam aumentos significativos em relação aos de 4 bits. No entanto, os valores de acurácia média se mantém em queda, já que o sistema tende a escolher os modelos menos complexos, não considerando se o aumento em vazão é significativo perante a redução na acurácia.

É importante notar que ocorre uma leve queda nos valores de vazão quando são atingidos os níveis máximos de limiar testados, entre 0.27 e 0.30. Como visto anteriormente, nessas situações o algoritmo possui um grau de liberdade próximo do seu máximo. Com isso, o algoritmo tende a escolher somente os modelos que possuem maior vazão (M1 e M5) ao longo dos diferentes valores de SNR. A partir deste ponto, um aumento no limiar permite que o modelo M1, com maior queda de acurácia, seja escolhido em cada vez mais pontos do percurso. Com isso, a quantidade de trocas efetuadas ao longo do percurso aumenta, o que consequentemente aumenta o impacto do custo dessas trocas na vazão total. Nesse caso, o aumento na quantidade de trocas possui um impacto maior do que o ganho de vazão obtido entre os modelos, ocasionando a queda de desempenho

Figura 4.10: Inferências processadas ao longo do tempo.



Fonte: Os Autores

apresentada no gráfico.

Além de analisar o impacto da vazão perante o aumento do limiar, podemos observar o comportamento do algoritmo ao longo do tempo através do número de classificações acumuladas durante o percurso. A Figura 4.10 apresenta, além do baseline, quatro limiares de acurácia distintos. No eixo x, temos o tempo percorrido, e no eixo y, o número classificações realizadas para cada valor de tempo. As linhas representam a relação entre tempo e classificações realizadas para cada limiar distinto (2, 4, 8 ou 10%). Dessa forma, é possível observar o comportamento da otimização de vazão para cada limiar ao longo do tempo.

Podemos perceber que o baseline possui o comportamento esperado, mantendo o aumento do número de classificações constante ao longo do tempo, o que é justificado pela ausência de trocas de modelo. De maneira indireta, as curvas ilustram o grau de liberdade do algoritmo para a escolha dos modelos ao longo do tempo. Ao aumentar os valores de limiar, o algoritmo permite ganhos expressivos no período entre 200 e 280 segundos. Recordando o cenário de avaliação, nesse momento o dispositivo está passando por uma área densa em edificações, se afastando gradativamente da antena que está conectado, e portanto a qualidade do sinal passa a ser prejudicada. Portanto, nesse período os valores de SNR tendem a ser menores do que zero, região onde a diferença entre os valores de acurácia entre os modelos é menor, possibilitando a escolha dos modelos com maior vazão.

A inclinação das curvas nas regiões entre 0-100 s e 300-400 s aumenta a medida que o limiar cresce, demonstrando que os ganhos de vazão são obtidos tanto nos momentos do percurso onde o SNR é médio (até 100 segundos) e alto (acima de 400 segundos). Nos momentos iniciais, o veículo está em condições normais de conexão, onde os valores de SNR são intermediários. Ao fim do percurso, o dispositivo se aproxima de uma área sem obstáculos e em seguida, de outra antena, onde as condições da conexão se tornam ideais e os valores de SNR se tornam altos pela melhor qualidade do sinal.

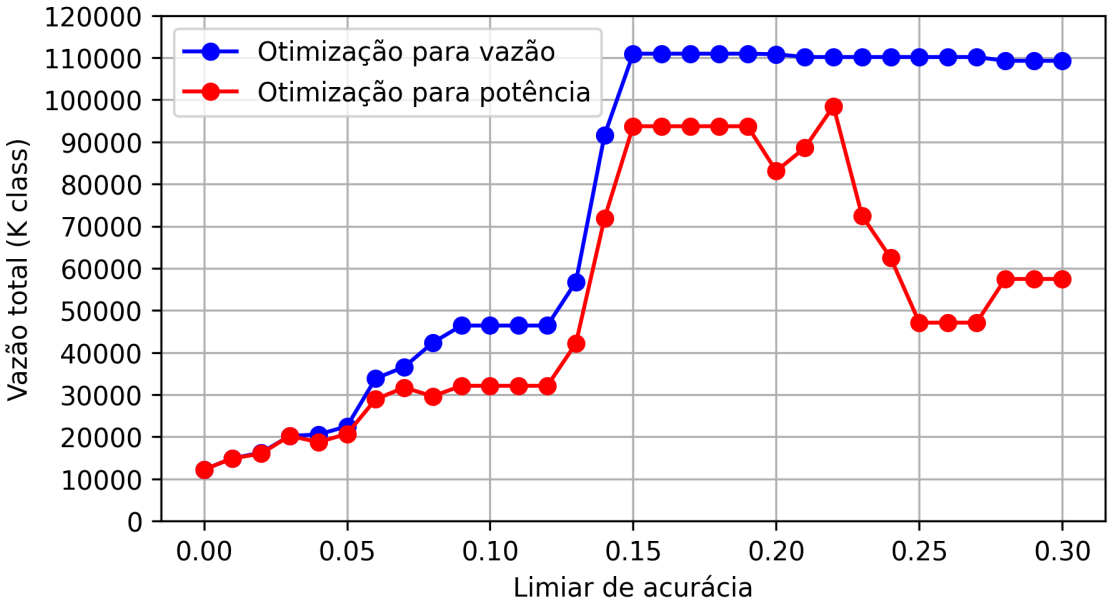
Ao analisar os resultados de vazão, foi possível mensurar o impacto da utilização do sistema adaptativo na capacidade de processamento de inferências do classificador. Em limiares de até 10%, foi observado um aumento de 350% nos valores de vazão total ao longo do percurso. O aumento de vazão ocorre principalmente pelo impacto significativo no desempenho entre os modelos ao reduzir a quantidade de canais, e sua independência da quantização, possibilitando a ausência de um grande impacto na redução de acurácia entre eles. Além disso, a visualização da vazão acumulada ao longo do tempo permitiu verificar a adaptabilidade da solução quanto ao desempenho ao longo dos diferentes níveis de SNR do percurso.

4.3.4 Impacto do Critério de Escolha na Vazão

Para ilustrar o comportamento do algoritmo ao utilizar diferentes critérios de otimização, podemos comparar a sua execução analisando uma das métricas (e.g. vazão) utilizando dois critérios distintos. Ao comparar as execuções utilizando a vazão como métrica, é esperado que os resultados demonstrados para a execução com o critério definido para a potência apresentem valores inferiores ao longo dos incrementos. A Figura 4.11 compara os critérios de otimização segundo a vazão. A linha em vermelho representa a execução com o critério de otimização para potência e em azul para vazão.

Em limiares baixos, o algoritmo não possui grau de liberdade suficiente para escolher modelos que otimizam um ou outro critério. A medida em que o limiar aumenta, a diferença entre o conjunto de modelos escolhidos, por possuir maior liberdade de escolha, também aumenta. A partir deste ponto a escolha entre os modelos é guiada pelo critério de otimização, onde a diferença entre as linhas é ocasionada por situações de escolha entre modelos que possuem grandes diferenças entre os critérios, como por exemplo, os modelos M4 e M5.

Figura 4.11: Vazão para os dois critérios de otimização com limiares de acurácia de 0 a 30%.



Fonte: Os Autores

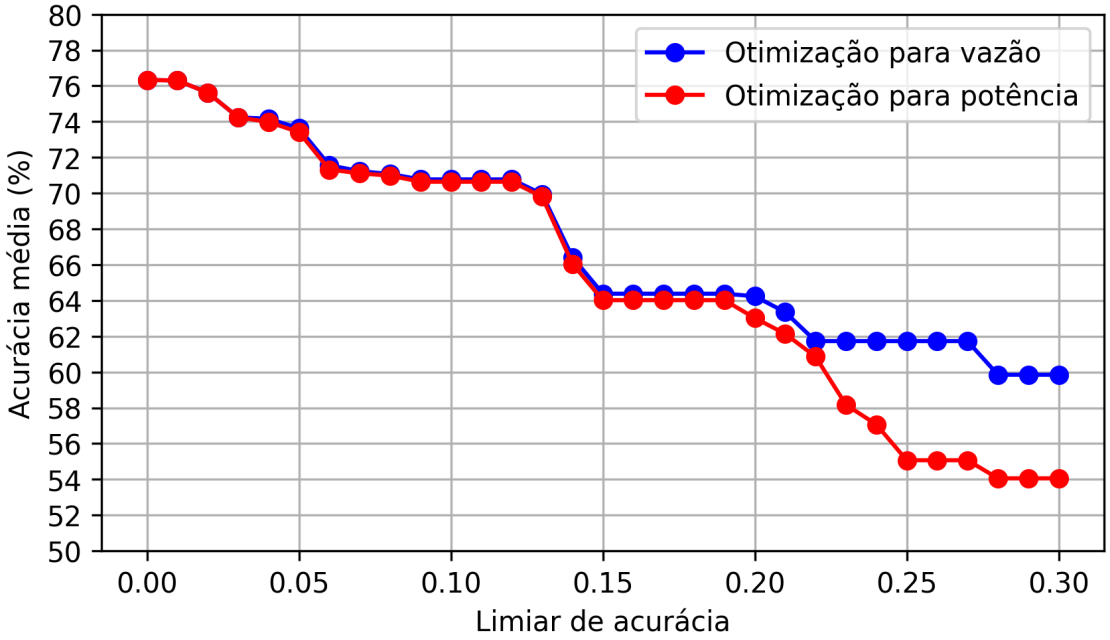
4.3.5 Impacto do Critério de Escolha na Acurácia

Similarmente a comparação levando em consideração a vazão, na Figura 4.12, podemos comparar o comportamento da acurácia. Em grande parte dos valores de limiar o comportamento da acurácia é similar entre os modos. A partir de 20%, as linhas se deslocam, com a otimização de potência ocasionando uma maior diminuição na acurácia. Isso ocorre pois a partir deste ponto, o subgrupo de modelos mais simples (M1-M4), em especial o modelo M1, tende a ser escolhido com mais frequência pelo critério de potência, e consequentemente apresentando menor acurácia.

É importante notar que, para o conjunto de modelos escolhidos, em limiares de até 15%, o comportamento de ambos os critérios é previsível quanto a acurácia. Isso possibilita que aplicações possam usufruir de diferentes alvos de otimização em momentos distintos, sem que a acurácia seja impactada de forma desigual entre eles.

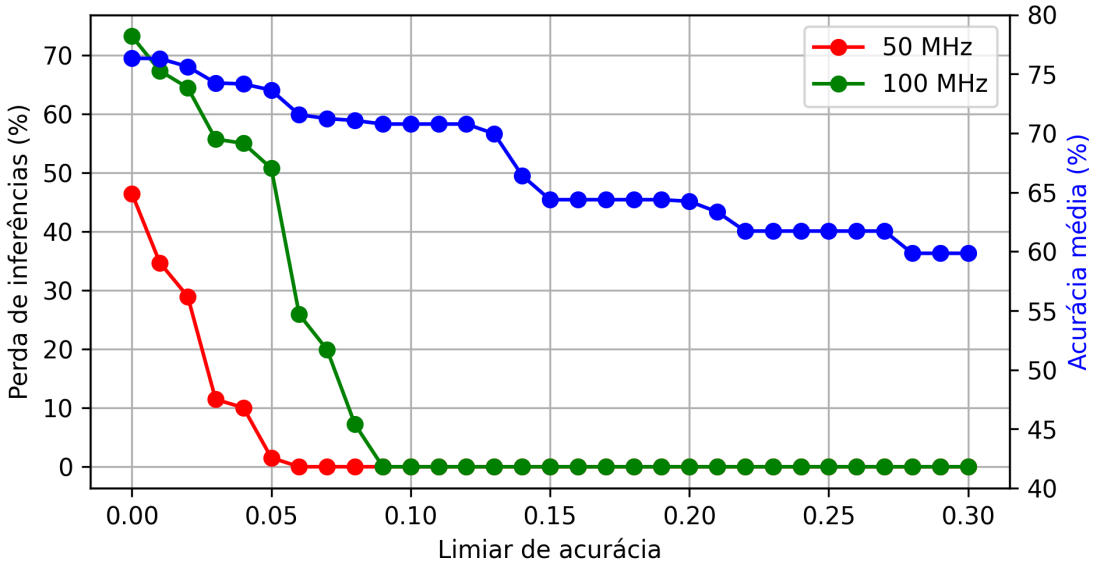
Os resultados demonstram ganhos significativos em consumo energético e desempenho pelo uso da solução adaptativa, obtendo ganhos de otimização ao longo de todos os limiares testados. A seguir, iremos contextualizar os resultados de desempenho da solução considerando o cenário de avaliação estabelecido.

Figura 4.12: Acurácia para os dois critérios de otimização com limiares de acurácia de 0 a 30%.



Fonte: Os Autores

Figura 4.13: Taxa de perda de inferências para amostragens de 50 e 100 MHz.



Fonte: Os Autores

4.3.6 Ganhos em uma Aplicação 5G

Os resultados de vazão apresentados anteriormente comparam proposta e baseline quanto a vazão máxima dos seus classificadores. Para contextualizar os ganhos de desempenho apresentados, consideremos agora a carga de trabalho em uma aplicação 5G

(apresentada na Seção 4.1) onde fica evidente que, caso a vazão do classificador não seja suficiente para absorver a carga de trabalho imposta pelo ambiente, uma parte dela não é processada, e parte das amostras do sinal recebidas é perdida.

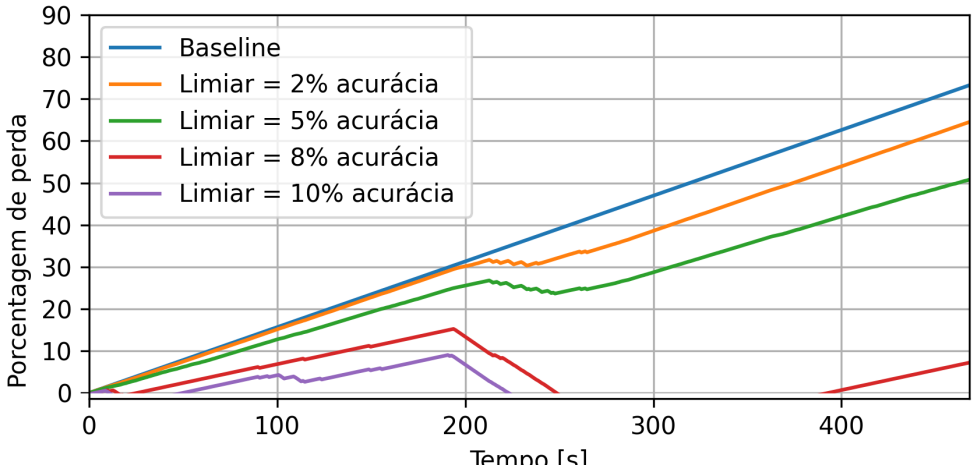
Na Figura 4.13, podemos observar a otimização da vazão através da minimização da taxa de perda de amostras. A linha em vermelho representa a perda considerando uma taxa de amostragem de 50 MHz e a linha em verde, 100 MHz. Assim como nas figuras anteriores, o eixo x apresenta os incrementos do limiar de acurácia. O eixo y da esquerda demonstra a perda de inferências, calculada pela diferença percentual entre o número de classificações por segundo de acordo com a taxa de amostragem (50 ou 100MHz) e os valores de vazão total para cada limiar. Uma perda igual a zero representa que a solução possui vazão suficiente para absorver totalmente a carga de trabalho imposta pelo ambiente. Além disso, o impacto na acurácia média é demonstrado no eixo y da direita.

Verificamos que o baseline, representado pelo limiar igual a zero, não obtém um resultado satisfatório em nenhuma das taxas de amostragem, apresentando uma perda significativa em ambos os casos. A perda é minimizada a medida em que a solução adaptativa consegue utilizar modelos com maior vazão, eventualmente se tornando nula. Como esperado, em cargas maiores, é necessário um maior sacrifício na acurácia média, e portanto um limiar de acurácia maior, para que a perda seja minimizada. No entanto, para ambas as cargas, foi possível obter uma taxa nula com um limiar menor do que 10%.

Considerando os resultados, é possível concluir que a solução opera em sua melhor faixa de compromisso entre desempenho e acurácia, em cargas onde a taxa de perda de inferências é minimizada ao atingir um limiar de acurácia de até 12%. Entre 13% e 15% o a acurácia média sofre um impacto significativo, e em 15% os ganhos de vazão atingem o seu máximo, como visto na Figura 4.9. Além disso, podemos observar o comportamento da otimização ao longo do tempo, ilustrado na Figura 4.14, para uma taxa de 100 MHz. Similarmente a Figura 4.10, o eixo x demonstra o tempo percorrido durante o percurso. Já o eixo y, demonstra a taxa de perda de inferências. Cada curva representa o comportamento da taxa ao longo do tempo para cada um dos limiares escolhidos. É esperado que a taxa de perda seja minimizada a medida que o limiar aumenta, demonstrando uma inclinação menor a cada incremento.

Os resultados apresentam o desempenho da solução em uma perspectiva de classificação sob uma carga de trabalho, e ilustram como a solução adaptativa pode ser utilizada para obter uma redução de perdas de inferência significativa se comparado ao modelo estático. Na Seção 2.3.1 mencionamos que os parâmetros que definem o paralelismo do

Figura 4.14: Taxa de perda de inferências ao longo do tempo para amostragem de 100 MHz.



Fonte: Os Autores

modelo sintetizado foram escolhidos para fornecer uma base de comparação igualitária entre os modelos. Em cenários onde a taxa de amostragem é superior aos valores apresentados e a minimização de perda de amostras é crítica, é possível redimensionar os modelos quanto aos seus hiperparâmetros e definir valores de paralelismo compatíveis que maximizem a vazão do conjunto ou subconjuntos específicos.

5 CONSIDERAÇÕES FINAIS

Os objetivos traçados na Seção 1.2 foram abordados ao longo do trabalho. Na Seção 4.2.1 investigamos os efeitos da modificação das características dos modelos de uma mesma arquitetura de rede, avaliando o impacto da quantização e do número de canais na acurácia do classificador. Desta forma, foi possível contribuir para a verificação de estratégias e critérios de escolha para modelos de DL na aplicação proposta. Na Seção 4.2.2 realizamos as etapas requeridas para transformar o conjunto de modelos em uma biblioteca de aceleradores para o hardware alvo através do processo de síntese. Ao obter os modelos sintetizados foi possível analisar as características de desempenho, consumo e utilização de recursos de cada modelo e o impacto das mudanças exploradas na Seção 4.2.1. Através destas etapas, foi possível definir o espaço de projeto disponível para elaborar a solução, e contribuir para a definição de critérios de otimização dos modelos para aceleradores embarcados em FPGA. Por fim, na Seção 4.3 foram avaliados os resultados segundo diferentes cenários de adaptabilidade e aplicação. Portanto, foi possível validar uma solução que evolui o estado-da-arte em relação a adaptabilidade das aplicações relacionadas, ao propor um sistema adaptativo que permite obter melhorias em eficiência energética e desempenho em relação a abordagem tradicional. A solução adaptativa contribui para possíveis abordagens de otimização de recursos levando em consideração o ambiente de classificação, podendo ser modificada utilizando outros critérios ou para otimizar parâmetros além dos abordados.

O trabalho apresenta novas possibilidades para estudos futuros, onde é possível explorar a adaptabilidade da solução utilizando diferentes métricas ou critérios de escolha para o algoritmo. O desempenho geral pode ser aperfeiçoado utilizando um conjunto de modelos mais amplo e otimizado para cenários específicos. A introdução de um algoritmo que pondera as diferenças de acurácia, vazão e energia para decidir a transição entre modelos pode minimizar o impacto de um desses aspectos mantendo os ganhos obtidos em outros. A adaptabilidade do algoritmo pode ser desenvolvida adiante tornando o critério de decisão dinâmico, ao invés de estático, de acordo com outras condições do ambiente ou até mesmo pela definição de limites de consumo energético ou perda de inferências.

REFERÊNCIAS

AKPAKWU, G. A. et al. A survey on 5g networks for the internet of things: Communication technologies and challenges. **IEEE Access**, v. 6, p. 3619–3647, 2018.

AL-FALAHY, N.; ALANI, O. Y. Technologies for 5g networks: Challenges and opportunities. **IT Professional**, v. 19, n. 1, p. 12–20, 2017.

AMAZON, E. Amazon ec2 p3 instances. **Disponível em: https://aws.amazon.com/ec2/instance-types/p3/ (Novembro 2021)**, 2021.

BLOTT, M. et al. **FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks**. 2018.

CHANG, R.; GIBBY, R. A theoretical study of performance of an orthogonal multiplexing data transmission scheme. **IEEE Transactions on Communication Technology**, v. 16, n. 4, p. 529–540, 1968.

DEEPSIG. **RadioML datasets**. 2018. <https://www.deepsig.ai/datasets>.

Docker. **Docker Website**. 2022. <https://www.docker.com>.

DUHEM, F.; MULLER, F.; LORENZINI, P. Farm: Fast reconfiguration manager for reducing reconfiguration time overhead on fpga. In: . [S.l.: s.n.], 2011. v. 6578, p. 253–260. ISBN 978-3-642-19474-0.

ERICSSON. **Ericsson Mobility Report 2021**. 2021.

FACELI, K. **Inteligência artificial: uma abordagem de aprendizado de máquina**. Grupo Gen - LTC, 2011. ISBN 9788521618805. Available from Internet: <https://books.google.com.br/books?id=4DwelAEACAAJ>.

FOROUZAN, B.; MOSHARRAF, F. **Redes de Computadores: Uma Abordagem Top-Down**. AMGH Editora, 2013. ISBN 9788580551693. Available from Internet: <https://books.google.com.br/books?id=57BIAgAAQBAJ>.

GOOGLE. Cloud tpu: Train and run machine learning models faster than ever before. **Disponível em: https://cloud.google.com/tpu (Fevereiro 2022)**, 2022.

HAMEED, F.; DOBRE, O. A.; POPESCU, D. C. On the likelihood-based approach to modulation classification. **IEEE Transactions on Wireless Communications**, v. 8, n. 12, p. 5884–5892, 2009.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. Prentice Hall, 1999. (International edition). ISBN 9780139083853. Available from Internet: <https://books.google.com.br/books?id=M5abQgAACAAJ>.

HAYKIN, S.; MOHER, M. **Sistemas modernos de comunicações wireless**. Bookman, 2008. ISBN 9788560031993. Available from Internet: <https://books.google.com.br/books?id=HvfuqmkpufwC>.

HOOFT, J. van der et al. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. **IEEE Communications Letters**, IEEE, v. 20, n. 11, p. 2177–2180, 2016.

HUANG, L. et al. **Data Augmentation for Deep Learning-based Radio Modulation Classification**. 2019.

IOFFE, S.; SZEGEDY, C. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**. 2015.

IVERSEN, A. et al. Classification of communication signals and detection of unknown formats using artificial neural networks. p. 21, 12 2006.

JAGANNATH, K. S. S. M. Adaptive modulation scheme with cooperative diversity in wireless systems. In: . [S.l.]: Journal of Signal Processing, 2019.

JR., L. E. F. **Fundamentos de Comunicação Eletrônica: Volume 1: Modulação, Demodulação e Recepção**. [S.l.]: Bookman, 2012. ISBN 8580551374.

JUPYTER. **Project Jupyter Website**. 2022. <https://jupyter.org/>.

KOLODZY, P. J. Dynamic spectrum policies: Promises and challenges. In: . [S.l.]: 2 CommLaw Conspectus 147, 2004.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 60, n. 6, p. 84–90, may 2017. ISSN 0001-0782. Available from Internet: <https://doi.org/10.1145/3065386>.

LAROS, J. et al. Energy delay product. In: ____. [S.l.: s.n.], 2013. p. 51–55. ISBN 978-1-4471-4491-5.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.

LEONG, P. H. W. Recent trends in fpga architectures and applications. In: **4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)**. [S.l.: s.n.], 2008. p. 137–141.

LUGER, G. F. **Artificial Intelligence Structures and Strategies for Complex Problem Solving**. [S.l.]: Addison Hill, 1988.

Lutz Roeder. **Netron repository**. 2022. <https://github.com/lutzroeder/netron>.

MCDONALD, E. Runtime fpga partial reconfiguration. **Aerospace and Electronic Systems Magazine, IEEE**, v. 23, p. 10–15, 08 2008.

MICROSOFT. Project catapult. **Disponível em: https://www.microsoft.com/en-us/research/project/project-catapult/ (Novembro 2021)**, 2021.

MITOLA, J.; MAGUIRE, G. Q. Cognitive radio: making software radios more personal. **IEEE Wirel. Commun.**, v. 6, p. 13–18, 1999.

MITTAL, S. **A survey of FPGA-based accelerators for convolutional neural networks.** 2020.

NUMPY. **NumPy Website**. 2022. <https://numpy.org/>.

NVIDIA. **NVIDIA Container Tookit**. 2022. <https://github.com/NVIDIA/nvidia-docker>.

NVIDIA Corporation. **Advanced AI Embedded Systems**. 2022. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/>.

O'SHEA, T. J.; CORGAN, J.; CLANCY, T. C. Convolutional radio modulation recognition networks. in engineering applications of neural networks. In: . [S.l.]: Springer International Publishing, Cham, 213–226, 2016.

O'SHEA, T. J.; ROY, T.; CLANCY, T. C. Over-the-air deep learning based radio signal classification. **IEEE Journal of Selected Topics in Signal Processing**, v. 12, n. 1, p. 168–179, 2018.

PAPPALARDO, A. **Xilinx/brevitas**. [S.l.]: Zenodo, 2021. <https://doi.org/10.5281/zenodo.3333552>.

PAULSSON, K. et al. Exploitation of run-time partial reconfiguration for dynamic power management in xilinx spartan iii-based systems. In: **ReCoSoC**. [S.l.: s.n.], 2007.

PYTORCH. **PyTorch**. 2022. <https://github.com/pytorch/pytorch>.

RUMELHART, G. E. H. . R. J. W. D. E. Learning representations by back-propagating errors. In: . [S.l.]: Nature 323, 533–536, 1986.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 1995. (Prentice Hall international editions). ISBN 9780131038059. Available from Internet: <https://books.google.com.br/books?id=CUVeMwAACAAJ>.

RUSSELL, S. et al. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 2010. (Prentice Hall series in artificial intelligence). ISBN 9780136042594. Available from Internet: <https://books.google.com.br/books?id=8jZBksh-bUMC>.

SEDCOLE, P. et al. Modular dynamic reconfiguration in virtex fpgas. **Computers and Digital Techniques, IEE Proceedings -**, v. 153(3), p. 157 – 164, 06 2006.

SHANNON, C. Communication in the presence of noise. **Proceedings of the IRE**, v. 37, n. 1, p. 10–21, 1949.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. 2015.

SOLTANI, S. et al. Real-time and embedded deep learning on fpga for rf signal classification. In: **MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)**. [S.l.: s.n.], 2019. p. 1–6.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. **J. Mach. Learn. Res.**, JMLR.org, v. 15, n. 1, p. 1929–1958, jan 2014. ISSN 1532-4435.

Synopsis. **What is Static Timing Analysis (STA)?** 2022. <https://www.synopsys.com/glossary/what-is-static-timing-analysis.html>.

The Linux Foundation. **ONNX Website**. 2022. <https://onnx.ai/>.

TIELEMAN, T.; HINTON., G. **Neural Networks for Machine Learning, 4, 26-31**. 2012.

TRIDGELL, S. et al. Real-time automatic modulation classification using rfsoc. In: **2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.: s.n.], 2020. p. 82–89.

UMUROGLU JENTSCH, P. **Xilinx/brevitas**. 2021. <https://github.com/Xilinx/brevitas-radioml-challenge-21>.

UMUROGLU, Y. et al. Finn. **Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays**, ACM, Feb 2017. Available from Internet: <http://dx.doi.org/10.1145/3020078.3021744>.

VERIPOOL. **Verilator**. 2022. <https://www.veripool.org/verilator/>.

WANG, Y. et al. Distributed learning for automatic modulation classification in edge devices. **IEEE Wireless Communications Letters**, v. 9, n. 12, p. 2177–2181, 2020.

WEINSTEIN, S. B. The history of orthogonal frequency-division multiplexing [history of communications]. **IEEE Communications Magazine**, v. 47, n. 11, p. 26–35, 2009.

WU, Z. et al. Robust automatic modulation classification under varying noise conditions. **IEEE Access**, v. 5, p. 19733–19741, 2017.

XILINX. **Smart World AI Video Analytics: Real-Time Analytics For A Smarter, Safer World**. 2020. <https://www.xilinx.com/applications/data-center/video-imaging/video-ai-analytics.html>.

Xilinx. **Embedded Vision Solutions Powered by Xilinx**. 2022. <https://www.xilinx.com/applications/megatrends/video-vision.html>.

Xilinx. **FINN-examples**. 2022. <https://github.com/Xilinx/finn-examples>.

XILINX. **Vivado**. 2022. <https://www.xilinx.com/products/design-tools/vivado.html>.

XILINX. **What is an FPGA? Field Programmable Gate Array**. 2022. <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>.

Xilinx Inc. **Vivado Design Suite User Guide: High-Level Synthesis (UG902)**. 2021. <https://docs.xilinx.com/v/u/en-US/ug902-vivado-high-level-synthesis>.

Xilinx Inc. **Vivado Partial Reconfiguration Documentation**. 2022. <https://docs.xilinx.com/r/en-US/ug909-vivado-partial-reconfiguration/>.

Xilinx Inc. **Webpage for Zync UltraScale+ MPSoC ZCU104**. 2022. <https://www.xilinx.com/products/boards-and-kits/zcu104.html>.