GABRIEL STEPHANO SANTOS

ANÁLISE FORMAL DE SEGURANÇA CRÍTICA (*SAFETY*) DE INTELIGÊNCIA ARTIFICIAL EM SISTEMAS DE DETECÇÃO DE ARRITMIAS CARDÍACAS: DIFERENCIAÇÃO ENTRE INDIVÍDUOS SAUDÁVEIS E NÃO SAUDÁVEIS

São Paulo
2024

GABRIEL STEPHANO SANTOS

ANÁLISE FORMAL DE SEGURANÇA CRÍTICA (*SAFETY*) DE INTELIGÊNCIA ARTIFICIAL EM SISTEMAS DE DETECÇÃO DE ARRITMIAS CARDÍACAS: DIFERENCIAÇÃO ENTRE INDIVÍDUOS SAUDÁVEIS E NÃO SAUDÁVEIS

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro de Computação.

São Paulo
2024

Nome: SANTOS, Gabriel Stephano

Título: Análise Formal de Segurança Crítica (*Safety*) de Inteligência Artificial em Sistemas de Detecção de Arritmias Cardíacas: Diferenciação entre Indivíduos Saudáveis e Não Saudáveis.

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro de Computação.

Aprovado em:

Banca Examinadora

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

GABRIEL STEPHANO SANTOS

ANÁLISE FORMAL DE SEGURANÇA CRÍTICA (*SAFETY*) DE INTELIGÊNCIA ARTIFICIAL EM SISTEMAS DE DETECÇÃO DE ARRITMIAS CARDÍACAS: DIFERENCIAÇÃO ENTRE INDIVÍDUOS SAUDÁVEIS E NÃO SAUDÁVEIS

Versão Original

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro de Computação.

Área de Concentração:

Engenharia de Computação

Orientador:

Prof. Dr. Paulo Sérgio Cugnasca.

Co-Orientador:

Prof. Dr. Antonio Vieira da Silva Neto

São Paulo
2024

Catalogação-na-publicação

Santos, Gabriel Stephano
Análise Formal de Segurança Crítica (Safety) de Inteligência Artificial em Sistemas de Detecção de Arritmias Cardíacas: Diferenciação entre Indivíduos Saudáveis e Não Saudáveis / G. S. Santos -- São Paulo, 2024.
73 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Arritmia 2.engenharia de sistemas de computação 3.inteligência artificial 4.redes neurais 5.segurança de software I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Primeiramente gostaria de agradecer à minha Mãe, Simone Stephano de Oliveira Leite, e à minha Vó, Olívia Gouveia Stephano, que me apoiaram nesta jornada e me ampararam nos momentos que eu mais precisei. Sem elas, eu nunca teria conseguido chegar onde estou.

Também gostaria de agradecer aos professores do corpo docente do Departamento de Engenharia de Computação e Sistemas Digitais (PCS) e de toda Escola Politécnica, que foram meus tutores durante a caminhada da graduação e cujos ensinamentos ficaram gravados em mim para todas as minhas jornadas futuras.

Agradecimentos especiais são dedicados aos integrantes do Grupo de Análise de Segurança (GAS) do Departamento de Engenharia de Computação e Sistemas Digitais (PCS) da Escola Politécnica da Universidade de São Paulo pela orientação e colaboração, principalmente ao meu Orientador, Prof. Dr. Paulo Sérgio Cugnasca, ao meu Co-orientador, Prof. Dr. Antonio Vieira da Silva Neto, e à aluna de graduação em Engenharia de Computação Ana Vitória Abreu Murad. Eles não somente foram fundamentais para a execução deste trabalho, como também para o meu desenvolvimento enquanto engenheiro, profissional e ser humano.

Por último, agradeço ao Itaú Unibanco S.A., que apoiou e financiou este trabalho por meio do Programa de Bolsas Itaú (PBI), vinculado ao Centro de Ciência de Dados (C2D) da Escola Politécnica da Universidade de São Paulo (dados da bolsa: projeto n° 3268 FUSP). Tais agradecimentos também se estendem à Fundação de Apoio à Universidade de São Paulo (FUSP), responsável pela intermediação administrativa dessa bolsa.

"Precisamos fazer o que já foi feito para a segurança aérea, a segurança de carros, a segurança de medicamentos, a segurança de dispositivos médicos. A segurança da IA não é diferente — na verdade, é potencialmente ainda mais perigosa."

(Richard Blumenthal)

RESUMO

A evolução da capacidade computacional e a abundância de dados têm intensificado o uso de Inteligência Artificial (IA) para contribuições em diversas áreas, tais como em aplicações críticas em relação à segurança crítica (*safety*). Entre as questões relevantes para projetos nesse tema estão métodos para avaliar se sistemas com IA são suficientemente seguros para serem usados nessas aplicações. O presente projeto busca aprofundar uma avaliação de segurança em um sistema baseado em redes neurais profundas que foi desenvolvido para auxiliar na detecção de arritmias cardíacas em eletrocardiogramas. Para isso, o projeto tem como objetivo construir uma arquitetura de verificação formal que possa descrever a aplicação em análise e averiguar a garantia de segurança crítica. Essa arquitetura prevê três etapas fundamentais para a análise formal de segurança: (i.) traduzir a rede neural para um padrão *de facto* denominado ONNX (*Open Neural Network Exchange*), (ii.) realizar a sobreaproximação da rede por um modelo mais simples e que é sabidamente mais inseguro do que o modelo original e (iii.) verificar o grau de satisfação das restrições de segurança do modelo sobreaproximado. Devido a limitações técnicas identificadas ao longo do desenvolvimento nas ferramentas de sobreaproximação em tratarem a rede neural do sistema analisado, utilizou-se, em contrapartida, uma estratégia baseada em ataques adversários. Essa estratégia, embora não exaustiva para fins de análise formal, possibilitou avaliar a robustez do sistema em situações críticas explorando-se potenciais vulnerabilidades do modelo e gerando-se resultados iniciais sobre a segurança da aplicação que contribuem para a avaliação formal do sistema. Por último, o projeto também prevê a continuação do seu desenvolvimento em futuro trabalho de Mestrado do próprio autor, expandindo a análise para englobar as outras classificações de arritmia, lidar com as limitações técnicas que impediram a execução plena da análise formal de segurança e comparar os resultados obtidos com as normas da área.

Palavras-chave: Arritmia, engenharia de sistemas de computação, inteligência artificial, redes neurais, segurança de software.

ABSTRACT

The evolution of computational capacity and the abundance of data have intensified the usage of Artificial Intelligence (AI) in several application domains, including safety-critical applications. One of the key issues involving safety-critical systems with AI is the need for methods to assess whether these systems are indeed sufficiently safe for usage in their respective applications. The aim of this project is to deepen the safety assurance of a deep neural network-based system that assists in detecting cardiac arrhythmias in electrocardiograms. To achieve this, the objective is to build a formal safety verification architecture that can describe the analyzed application and verify whether the system formally meets its safety restrictions. This architecture has three major formal safety analysis steps: (i.) translating the neural network to a *de facto* standard called ONNX (Open Neural Network Exchange), (ii.) performing the overapproximation of the neural network by means of a simpler model that is known to be more unsafe than the original one, and (iii.) verifying to what extent the safety constraints of the overapproximated model are indeed met.. Due to unanticipated technical limitations identified in the overapproximation tools during the project development, an adversarial attack-based strategy was employed instead. Even though this strategy lacks the needed exhaustive for formal analysis purposes, it still allowed evaluating the system robustness in critical situations by exploring potential vulnerabilities of the model and generating initial safety assurance results that contribute to the formal safety assurance of the system. Finally, the project also envisions continuing its development in a future Master's thesis by the author himself. These future work comprise expanding the analysis to encompass other arrhythmia classifications, addressing the technical limitations that hindered the full formal safety assurance, and comparing the results obtained with industry standards.

Keywords: Arrhythmia, computer systems engineering, artificial intelligence, neural networks, software safety.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIAÇÕES

Auto-LiRPA	*Automatic Linear Relaxation based Perturbation Analysis for Neural Networks* (Análise de Perturbação baseada em Relaxamento Linear Automático para Redes Neurais)
BRL	Real Brasileiro
C2D	Centro de Ciência de Dados
CAISAR	*Characterizing Artificial Intelligence Safety and Robustness* (Caracterizando Robustez e Segurança de Inteligência Artificial)
DDR	*Double Data Rate* (Taxa de Dados Dupla)
ECG	Eletrocardiograma
ERAN	*ETH Robustness Analyzer for Neural Networks* (Analisador de Robustez para Redes Neurais do ETH)
ETH	*Eidgenössische Technische Hochschule Zürich* (Instituto Federal de Tecnologia de Zurique)
FUSP	Fundação de Apoio à Universidade de São Paulo
HP	Hora-Pessoa
HUDD	*Heatmap-Based Unsupervised Debugging of Neural Networks* (Depuração de Redes Neurais não Supervisionada Baseada em Mapa de Calor)
IA	Inteligência Artificial
LDA	*Linear Discriminant Analysis* (Análise Discriminante Linear)
LEB	Laboratório de Engenharia Biomédica
NeVer	*Neural Networks Verifier* (Verificador de Redes Neurais)
nnenum	*Neural Network Enumerator* (Enumerador de Redes Neurais)
NNV	*Matlab Toolbox for Neural Network Verification (Caixa de Ferramentas do Matlab para Verificação de Redes Neurais)*
ONNX	*Open Neural Network Exchange* (Intercâmbio Aberto de Redes Neurais)
PBI	Programa de Bolsas do Itaú
PC	*Personal Computer* (Computador Pessoal)
PCA	*Principal Component Analysis* (Análise de Componentes Principais)
PEREGRiNN	*Penalized-Relaxation Greedy Neural Network Verifier* (Verificador Guloso de Rede Neural com Relaxamento Penalizado)
PGD	*Projected Gradient Descent* (Gradiente Descendente Projetado)
PN	Notação Polonesa (*Polish Notation*)

Polar	*Polynomial Arithmetic-Based Framework* (Arcabouço Baseado em Aritmética Polinomial)
PTC	Departamento de Engenharia de Telecomunicações e Controle
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
ReachNN*	*Reachability of Neural Networks* (Alcançabilidade de Redes Neurais)
ResNet	Rede Neural Residual (*Residual Neural Network*)
SIICUSP	Simpósio Internacional de Iniciação Científica e Tecnológica da Universidade de São Paulo
SMOTE	*Synthetic Minority Oversampling Technique* (Técnica de Sobreamostragem de Minoria Sintética)
SMT	*Satisfiability Modulo Theory* (Teoria Módulo Satisfatibilidade)
SMT-LIB	*Satisfiability Modulo Theories Library* (Biblioteca de Teorias Módulo Satisfatibilidade)
SSD	*Solid-State Drive* (Unidade de Estado Sólido)
TAK	*Title, Abstract, Keywords* (Título, Resumo, Palavras-Chave)
UMCE	*Undersampled Majority Class Ensemple* (Comitê de Classe Majoritária Subamostrada)
UML	*Unified Modeling Language* (Linguagem de Modelagem Unificada)
V&V	Verificação e Validação
VNN-LIB	*Verification of Neural Networks Library* (Biblioteca de Verificação de Redes Neurais)
VNNCOMP	*International Verification of Neural Networks Competition* (Competição Internacional de Verificação de Redes Neurais)

SUMÁRIO

1. INTRODUÇÃO

Neste capítulo da monografia, apresentam-se definições de termos-chave para o projeto, bem como as motivações, os objetivos e as justificativas do projeto. Por fim, também se detalha a estrutura desta monografia.

1.1 DEFINIÇÕES

Estabelecem-se nesta monografia as seguintes definições para as expressões “segurança crítica (*safety*)” e “inteligência artificial”. Segundo Silva Neto (2024):

> “A segurança crítica (*safety*) de um sistema é matematicamente definida como uma função do tempo que descreve a probabilidade condicional de o sistema desempenhar, em um período de tempo, suas funcionalidades corretamente ou descontinuá-las sem causar mortes, danos à saúde, destruição de propriedades, perda de missão ou impactos ao meio ambiente. O fator condicionante para determinar a segurança crítica (*safety*) de um sistema é a hipótese de que ele esteja funcionando corretamente no instante inicial de sua observação.”

> “(...) o termo “inteligência artificial (IA)” é munido de múltiplas definições na literatura da área. O enfoque adotado (...) relaciona-se a modelos eminentemente probabilísticos que, uma vez projetados e configurados para uma aplicação-alvo, têm a habilidade de processar entradas e gerar saídas para essa finalidade, emulando habilidades de aprendizado e de interação com o ambiente de operação.”

Nesta monografia, referências ao termo “segurança” sem qualquer qualificador adicional devem ser interpretados como “segurança crítica (*safety*) (SILVA NETO, 2024)”, tal qual definido anteriormente nesta seção.

1.2 MOTIVAÇÃO

A evolução da capacidade computacional e a abundância de dados vêm fazendo com que o uso de inteligência artificial (IA) para contribuições em diversas áreas seja intensificado, tais como aplicações críticas em relação à segurança

(*safety*). Entre as questões relevantes para pesquisa nesse tema estão métodos para avaliar se sistemas com IA são suficientemente seguros para serem usados nessas aplicações (SILVA NETO et al., 2022).

Uma das aplicações críticas em que o uso de tecnologias de IA tem avançado é a área médica, principalmente no conceito de diagnóstico assistido por IA, em que a tecnologia cumpre o papel de auxiliar o agente médico a analisar exames, a fim de concluir a condição do paciente. Com a leitura realizada para a caraterização do estado da arte, presente no capítulo 2, é caracterizado que, mesmo em pesquisas que retratam esses sistemas críticos, pouca preocupação é posta na garantia de segurança crítica.

1.3 OBJETIVOS

A pesquisa definida para o projeto de formatura busca aprofundar uma avaliação de segurança crítica previamente feita (SILVA NETO; CUGNASCA, 2023) em um sistema baseado em redes neurais profundas que foi desenvolvido para auxiliar na detecção de arritmias cardíacas em eletrocardiogramas (KOZAL; KSIENIEWICZ, 2019). A pesquisa tem como objetivo sobreaproximar o conjunto imagem da inteligência artificial por meio de regras lógico-aritméticas que descrevem a aplicação em análise e avaliar os resultados para averiguar a garantia de segurança crítica – algo não realizado no trabalho de referência (SILVA NETO; CUGNASCA, 2023).

Assim, o objetivo do trabalho de conclusão do curso é avaliar formalmente a segurança sistema de detecção de arritmias cardíacas, através da diferenciação do domínio dos batimentos cardíacos classificados como saudáveis do domínio das classificações de arritmia. Com isso, o sistema pode ser analisado com maior previsibilidade, e as áreas de maior possibilidade de falha do sistema são identificadas.

O sistema alvo da verificação formal da pesquisa é subdivido em quatro subsistemas de redes neurais residuais (ResNets) que utilizam técnicas diferentes para lidar com desbalanceamento entre as classificações presentes no conjunto de dados em que as redes foram treinadas. Um dos sistemas não utilizou nenhuma

técnica para lidar com o desbalanceamento dos dados e é utilizado como base de referência para os outros modelos. Dois modelos utilizam a técnica *Synthetic Minority Oversampling Technique* (SMOTE), que utiliza dados gerados artificialmente para balancear as classes do conjunto de dados, nivelando pela classe com maior representação nos dados. Em um dos casos, a técnica SMOTE é implementada juntamente com técnicas de aumento de dados (*data_augmentation*). O último modelo utiliza a técnica de *Undersampled Majority Class Ensemble* (UMCE), que consiste em dividir os dados de maneira balanceada, nivelando pela classe com menor representação, treinando vários classificadores com os dados divididos e obtendo os resultados por um sistema de votação do comitê composto por esses classificadores.

O foco deste trabalho de conclusão de curso é a variante do sistema com a técnica SMOTE sem aumento de dados. A justificativa para isso é a de que, entre as quatro variações estudadas por Silva Neto e Cugnasca (2023) foi a que ele atingiu os níveis de segurança mais altos.

1.4 JUSTIFICATIVA

Dado o exposto, é possível afirmar que o trabalho proposto, uma vez que atinja seus objetivos, consegue preencher um espaço na área de avaliação de segurança crítica em sistemas de IA. Além disso, é percebida a importância da continuidade de progresso na área de segurança crítica aplicada em sistemas de IA, pois aplicações desse tipo têm se tornado cada vez mais comuns e, por vezes, sem a preocupação necessária para a garantia de segurança.

Assim, o projeto a ser desenvolvido demonstra-se como viável e justificável, pois visa preencher uma lacuna na análise formal de segurança de sistemas com IA voltados à detecção de arritmias cardíacas.

1.5 ESTRUTURA DA MONOGRAFIA

Esta monografia é subdividida em oito capítulos e um apêndice.

O primeiro capítulo teve como objetivo apresentar a motivação, os objetivos e a justificativa de relevância do trabalho de conclusão de curso para a área de detecção de arritmias cardíacas por IA.

O segundo capítulo versa sobre a revisão sistemática de literatura que caracteriza o estado da arte da análise formal de segurança crítica (*safety*) de sistemas de IA em detectar arritmias cardíacas.

No terceiro capítulo, apresenta-se a especificação de requisitos necessária para a análise formal de segurança crítica (*safety*) a ser coberta no presente trabalho de conclusão de curso.

A arquitetura dessa solução, por sua vez, é explorada no quarto capítulo desta monografia. A descrição arquitetural contempla aspectos estruturais e comportamentais da solução a ser projetada para realizar a análise formal de segurança crítica (*safety*) da IA que diferencia batimentos cardíacos saudáveis e não saudáveis.

No quinto capítulo da monografia, consta o planejamento do projeto. Nele, abordam-se os recursos e custos de desenvolvimento e o cronograma seguido até o momento.

No sexto capítulo, é detalhado o desenvolvimento do sistema. Essa exposição inclui a documentação de todos os passos seguidos e também referencia o Apêndice A, que contém e-mails trocados com a equipe de pesquisadores da ferramenta de análise formal empregada no projeto.

Os resultados associados ao desenvolvimento prévio são, na sequência, apresentados, analisados e discutidos no sétimo capítulo da monografia.

Por último, no oitavo capítulo, constam as conclusões do trabalho. Elas abarcam dois temas: as considerações finais sobre o projeto e potenciais trabalhos futuros.

2. ESTADO DA ARTE DA ANÁLISE FORMAL DE SEGURANÇA CRÍTICA (*SAFETY*) DE DETECÇÃO DE ARRITMIAS CARDÍACAS BASEADA EM INTELIGÊNCIA ARTIFICIAL

Neste capítulo da monografia, expõe-se a revisão de literatura realizada com o propósito de caracterizar o estado da arte na área e servir como motivador e justificativa para o projeto a seguir.

2.1 MÉTODO DE REVISÃO DE LITERATURA

A fim de obter o panorama sobre os trabalhos existentes na área, contextualizar e motivar a pesquisa, foi realizada uma revisão de publicações que englobassem três domínios: (i.) segurança crítica, (ii.) IA e (iii.) detecção de arritmias cardíacas via eletrocardiogramas (ECGs). Para tanto, foi utilizado um método de revisão sistemática de literatura utilizado por Silva Neto et al. (SILVA NETO et al., 2022).

A primeira etapa consiste em definir os termos e as bases de pesquisa. Foram escolhidas as mesmas bases de pesquisa de Silva Neto et al. (2022), pois elas cobrem os temas abordados de maneira ampla: ACM, *Engineering Village*, *Scopus*, *Web of Science* e *Wiley*. Na segunda etapa, voltada à definição dos termos de busca, foram realizadas pesquisas sobre artigos em cada área para extrair as palavras-chave mais relevantes de cada um dos três domínios da pesquisa. Assim, com as expressões de cada domínio estabelecido, os termos foram concatenados em expressões booleanas com operadores “ou” para expressões de um mesmo domínio e operadores “e” entre os domínios utilizando e lógico. Dessa maneira, garante-se que a busca contemple artigos que passem por todos os domínios desejados, com pelo menos um termo de cada domínio.

Em seguida, procedeu-se à busca das publicações utilizando, como referência, suas informações de Título, Resumo e Palavras-Chave (TAK – *Title, Abstract, Keywords*). Após a remoção de duplicatas entre as bases de referências, obtiveram-se 1057 artigos. Pela análise semântica manual de seus resumos, foram descartados artigos que fugiam ao escopo de análise formal de segurança crítica em sistemas de IA que realizam detecção de arritmias cardíacas.

Essa filtragem resultou em 239 referências com potencial de contribuição às áreas da pesquisa. Tais artigos, por sua vez, foram classificados nas categorias da Tabela 1.

Tabela 1 – Sumário das Categorias das Referências Consultadas

Categoria	Análise formal?	Utiliza Inteligência Artificial?	Classifica arritmia cardíaca?	N° de referências	Explicação (*rationale*)
C0	Sim	Sim	Sim	7	Encaixa-se ao escopo completamente
C1	Não	Sim	Sim	35	Aborda-se o tópico sem análise formal.
C2	Não	Sim	Sim	145	Além de não possuir análise formal, não há preocupação de *safety*.
C3	Não	Secundário	Secundário	29	Utiliza-se IA como aplicação secundária.
C4	Não	Indiferente	Não	23	Não se lida com arritmia cardíaca.

Por último, os artigos da categoria C0, foram lidos e analisados em sua completude, a fim de estabelecer a conexão com o projeto proposto e os trabalhos publicados mais relevantes na área.

2.2 Caracterização da Análise Formal de Segurança Crítica (*Safety*) de Detecção de Arritmias Cardíacas Baseada em Inteligência Artificial

Um dos modelos tratados, que possibilita a avaliação formal de segurança crítica de um sistema de IA, é o *neuro-fuzzy* (BENALI; DIB; BEREKSI, 2010), que mistura a técnica de redes neurais com a técnica de lógica nebulosa (*fuzzy*)[1] por meio de árvores de decisão. O achado mais interessante desse artigo foi demonstrar a reconstituição das regras encontradas por meio da árvore de decisão e constatar que

[1] A lógica nebulosa consiste em separar os dados originais em categorias discretas ao invés de usar os dados numéricos como modelo para a IA.

o comportamento dela é próximo do consenso médico. Por outro lado, os autores não se aprofundam na comparação do comportamento do sistema com o consenso médico, nem na maneira com que eles divergem.

Outra abordagem utilizada é o modelo de IA de caixa-branca (*white-box*) (RIEG et al., 2020), que permite a verificação e a interpretação do modelo por meio de árvores de decisão explicáveis. Apesar da explicabilidade, os resultados atingidos por meio desse modelo não são tão eficazes quanto modelos de redes neurais – que são mais complexos e menos explicáveis. Dessa forma, há uma perda de eficiência apesar do ganho da expressividade. No artigo, é reiterado que a explicação das decisões do sistema pode ser utilizada para uma análise formal do sistema, mas esse conceito não é mais aprofundado.

Ademais, outro método apresentado foi o de se utilizar um modelo de IA criado utilizando regras da dinâmica cardíaca, e essas regras são utilizadas para o sistema caracterizar arritmias e rastreá-las nas regras dinâmicas que as justificam (DONG; SI, 2023). Apesar dessa qualidade, o modelo torna necessário definir as regras da dinâmica cardíaca, o que gera uma preocupação de segurança sobre as regras escolhidas e suas motivações.

Houve também uma exploração de um modelo que reside entre redes neurais e árvores de decisão (JO et al., 2021): o modelo de redes neurais explicáveis que utilizam um modelo híbrido, com árvores de decisão auxiliando redes neurais. De maneira semelhante aos demais, há um sacrifício de facilidade de interpretação do modelo e eficácia atingível. Outro problema do modelo é o método para extração de regras do domínio, que é mencionado no artigo, porém sem esclarecimento. Também é necessário destacar que os autores não levantam preocupações com a segurança crítica do sistema, deixando essa análise para trabalhos futuros.

Já no domínio de redes neurais convolucionais, foi explorada a adaptação de domínio (CHAO; LI; ZHANG, 2023), que utiliza uma extrapolação de regras entre domínios diferentes, aqui definidos como cada paciente, para estabelecer um padrão e melhorar a exatidão (*accuracy*) relativa a outras redes neurais. Contudo, não há uma explicação mais profunda sobre as regras selecionadas entre domínios e não houve uma preocupação de análise de segurança crítica nos padrões estabelecidos.

Outro método explorado baseia-se em técnicas de *principal component analysis* (PCA) com *linear discriminant analysis* (LDA) junto de uma rede neural probabilística com classificador Bayesiano (WANG et al., 2011). Essa abordagem permite que a IA seja treinada em dados mais segmentados, tornando o trabalho de análise formal de segurança crítica do sistema mais simples. Apesar dessa vantagem, os autores não exploram essa capacidade e apenas realizam uma comparação entre esse sistema e outros sistemas de classificação baseados em IA.

A última abordagem avaliada consiste em extrair as regras de classificação de arritmias cardíacas com abstração temporal e lógica indutiva de programação (CARRAULT et al., 2003). Essa abordagem pouco explora a IA como sistema para uma aplicação crítica, uma vez que, nela, a IA serve apenas como ferramenta para a abstração das regras de classificação de arritmias. Por isso, foram obtidos resultados satisfatórios de análise formal, uma vez que é possível definir de maneira determinística a saída do sistema. Todavia, perde-se em eficiência em comparação com sistemas de IA mais atuais.

Através desses achados, é estabelecida a conclusão de que, mesmo nas pesquisas que abordam o tema de análise formal de segurança crítica em IA, utilizam-se tecnologias diferentes das redes neurais – que são abordadas neste projeto. Adicionalmente, as pesquisas, por vezes, apenas se atêm em avaliar que a análise formal de segurança é possível, sem apresentar justificativas sobre essa conclusão ou ir executar a análise formal de segurança propriamente dita – i.e., comparar o resultado do sistema com a compreensão médica atual para aferir as possíveis falhas e limitações da IA.

2.3 Conclusão da Revisão Sistemática de Literatura

A partir da exploração do tema, dos trabalhos que servem de base para o projeto e da caracterização do estado da arte no tema de pesquisa, pode-se justificar que este projeto é necessário para o aprofundamento da pesquisa de referência, que trata da garantia de segurança crítica aplicada à IA, mas sem aprofundamento na aplicação de métodos formais (SILVA NETO; CUGNASCA, 2023).

Além disso, constata-se que o tema do projeto não é coberto totalmente por nenhuma pesquisa publicada até o período atual. Dessa maneira, ele tem potencial para preencher lacunas identificadas, até o momento, na área da pesquisa.

2.4 INTRODUÇÃO SOBRE ELETROCARDIOGRAFIA

A fim de contextualizar o sistema, nesta seção é explicado brevemente como se comportam os sinais de eletrocardiogramas normais, para fins de análise dos resultados obtidos e estabelecimento de notações relevantes para o trabalho.

O sinal do eletrocardiograma de um batimento cardíaco normal é caracterizado pela morfologia ilustrada na Figura 1. Seus principais componentes são (i.) o complexo QRS, composto por um vale (Q) seguido de um pico global (R) e sucedido por outro vale (S), e (ii.) as ondas T, U e P (THALER; BURNIER, 2023). A onda P representa o batimento atrial e é sucedida pelo complexo QRS, que corresponde ao batimento ventricular do bombeamento cardíaco. Por fim, as ondas T e U estão associadas à repolarização do músculo cardíaco.

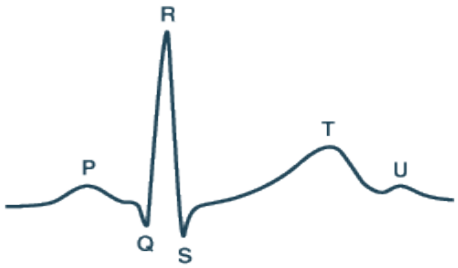

Figura 1 – Sinal de Eletrocardiograma

Fonte: Analog Dialogue (2003)

Devido à forma com que o sinal é capturado pelo sistema de detecção de arritmias baseado em redes neurais (KOZAL; KSIENIEWICZ, 2019), ele se inicia no pico R e é seguido pelo restante do batimento, com o vale S e a onda T. Dessa forma, um batimento cardíaco normal representado como entrada da rede neural é uma série temporal, iniciando em um pico global R e seguido por um vale S. Após isso, ocorre uma onda T e uma onda U – esta última, em alguns casos pode ser insignificante perante a onda T – e no final do sinal uma série de pontos nulos devido ao término do batimento cardíaco. Não há captura ou tratamento nem da onda P e nem do vale Q, os quais, pela morfologia do batimento, deveriam aparecer ao final da série.

3. ESPECIFICAÇÃO DE REQUISITOS

Este capítulo da tese versa sobre a especificação de requisitos do sistema que orienta a análise formal de segurança crítica (*safety*) da IA que diferencia batimentos cardíacos saudáveis de não saudáveis (i.e., com arritmias). A elicitação de requisitos foi conduzida inspirando-se na abordagem proposta por Ericsson, Oberg e Probasco (2000). A caracterização de um requisito compreende as seguintes informações:

- Código de Identificação;
- Nome;
- Descrição;
- Prioridade;
- Estabilidade;
- *Rationale* (informações adicionais);
- Requisitos associados.

Os seis requisitos especificados para o projeto são apresentados entre a Tabela 2 e a Tabela 7.

Tabela 2 – Especificação do Requisito 00

Código: 00	X Funcional ☐ Não Funcional
Requisito: Análise formal do modelo SMT	
Descrição: O sistema deve realizar análise formal de segurança do modelo SMT obtido ao final da sobreaproximação lógico-aritmética.	
Prioridade: X Alta ☐ Média ☐ Baixa	
Estabilidade: X Alta ☐ Média ☐ Baixa	
***Rationale*:** Etapas de “Conjunto de Regras satisfeitas ou não pelo sistema de IA” e “Resultado da análise de segurança” na arquitetura funcional (Figura 2 do capítulo 4) e os pacotes “Alpha-beta-crown” e “analysis” na arquitetura estrutural (Figura 3 do capítulo 4).	
Requisitos associados: #01 e #02	

Tabela 3 – Especificação do Requisito 01

Código: 01	X Funcional ☐ Não Funcional
Requisito: Sobreaproximação da Rede Neural para regras lógico-aritméticas	
Descrição: O sistema deve sobreaproximar o sistema de redes neurais por um conjunto de regras lógico-aritméticas.	
Prioridade: X Alta ☐ Média ☐ Baixa	
Estabilidade: X Alta ☐ Média ☐ Baixa	
***Rationale*:** Etapa necessária para a avaliação formal do sistema de IA. Etapas de “Interface com o sistema de IA”, “Sobreaproximação do modelo utilizando regras lógico-aritméticas” e “Interface do modelo de regras lógico aritméticas” na arquitetura funcional (Figura 2 do capítulo 4) e pacotes “ONNX” e “Alpha-beta-crown” na arquitetura estrutural (Figura 3 do capítulo 4).	
Requisitos associados: #02 e #05	

Tabela 4 – Especificação do Requisito 02

Código: 02	X Funcional ☐ Não Funcional
Requisito: Verificação da Sobreaproximação da Rede Neural	
Descrição: Deve-se verificar se o modelo sobreaproximado é condizente com o modelo de Inteligência artificial de entrada	
Prioridade: X Alta ☐ Média ☐ Baixa	
Estabilidade: X Alta ☐ Média ☐ Baixa	
***Rationale*:** Etapa necessária garantir a validade do projeto. Etapa de “Sobreaproximação do modelo utilizando regras lógico-aritméticas” na arquitetura funcional (Figura 2 do capítulo 4) e pacote “Alpha-beta-crown” na arquitetura estrutural (Figura 3 do capítulo 4).	
Requisitos associados: #01	

Tabela 5 – Especificação do Requisito 03

Código: 03	X Funcional ☐ Não Funcional
Requisito: Tradução do modelo de Keras para ONNX	
Descrição: O sistema deve ser capaz de traduzir o modelo original em Keras para ONNX, a fim de utilizar o arcabouço tf2ONNX.	
Prioridade: X Alta ☐ Média ☐ Baixa	
Estabilidade: ☐ Alta X Média ☐ Baixa	
***Rationale*:** Etapa de "Tradução do modelo de IA para ONNX" na arquitetura funcional (Figura 2 do capítulo 4) e os pacotes "interface Keras", "ONNX" e "tf2ONNX" na arquitetura estrutural (Figura 3 do capítulo 4).	
Requisitos associados: #01, #04	

Tabela 6 – Especificação do Requisito 04

Código: 04	X Funcional ☐ Não Funcional
Requisito: Verificação da tradução do modelo de Keras para ONNX	
Descrição: Deve-se verificar se o modelo traduzido ONNX é equivalente ao modelo Keras de entrada	
Prioridade: X Alta ☐ Média ☐ Baixa	
Estabilidade: ☐ Alta X Média ☐ Baixa	
***Rationale*:** Etapa de "Tradução do modelo de IA para ONNX" na arquitetura funcional (Figura 2 do capítulo 4) e os pacotes "interface Keras", "ONNX" e "tf2ONNX" na arquitetura estrutural (Figura 3 do capítulo 4).	
Requisitos associados: #03	

Tabela 7 – Especificação do Requisito 05

Código: 05	☐ Funcional X Não Funcional
Requisito: O sistema deve ser executável em tempo hábil < 24h	
Descrição: É necessário que o sistema execute todos os processos de tradução e verificação em um tempo que permita a conferência de informações e teste, para garantir a agilidade do desenvolvimento	
Prioridade: ☐ Alta ☐ Média X Baixa	
Estabilidade: ☐ Alta ☐ Média X Baixa	
***Rationale*:** Restrição importante para a viabilidade técnica do desenvolvimento do projeto.	
Requisitos associados: #00, #01, #02, #03, #04	

4. ARQUITETURA DO SISTEMA DE VERIFICAÇÃO FORMAL DE SEGURANÇA

Este capítulo da monografia é destinado ao detalhamento da arquitetura do sistema de verificação formal de segurança. Ela é definida de duas formas: uma estrutural, que define os pacotes de software do sistema de análise formal de segurança, e uma funcional, que estabelece a descrição dinâmica do comportamento do sistema. Esta, por sua vez, é descrita com auxílio de duas representações que possuem níveis de profundidade distintos. A primeira delas, que simboliza o panorama das trocas de dados entre os blocos funcionais do sistema, consiste em um diagrama de blocos. A segunda é composta por diagramas de sequência, que detalham como cada requisito funcional do capítulo 3 deve ser atendido pelos pacotes de software previstos no projeto.

Para fins didáticos, inicia-se a descrição da arquitetura do sistema por meio do diagrama de blocos que representa o panorama da arquitetura funcional (comportamental) do sistema. Esse diagrama é representado na Figura 2.

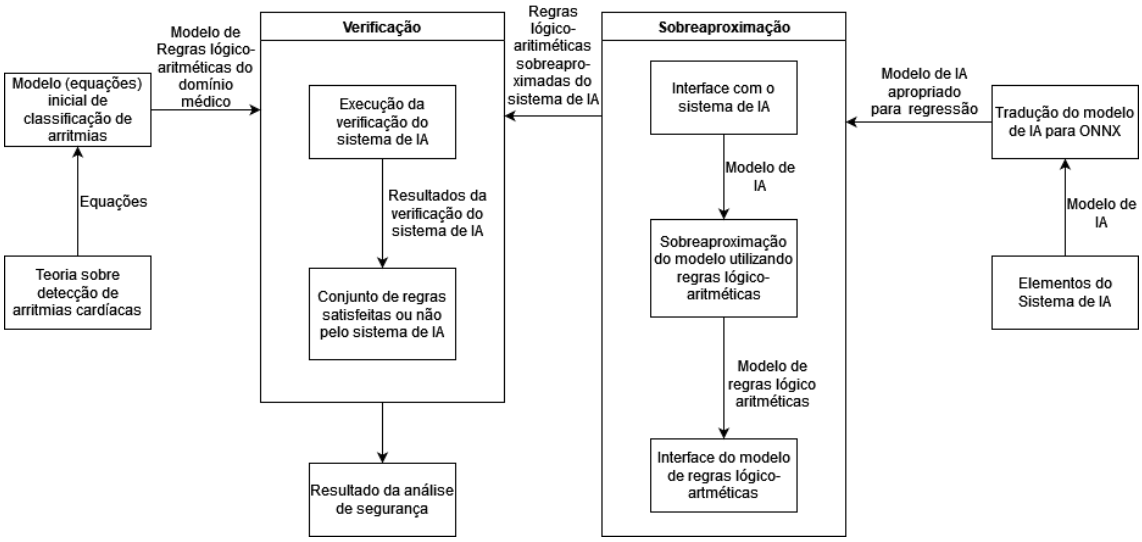

Figura 2 – Diagrama de Arquitetura Funcional / Comportamental do Sistema

O diagrama de arquitetura funcional (comportamental) do sistema da Figura 2 evidencia duas entradas externas, que são integradas a ela em momentos diferentes. Elas compreendem os “elementos do sistema de IA”, advindos do sistema de classificação de arritmias cardíacas, e a “teoria sobre detecção de arritmias cardíacas”, proveniente de conhecimento especialista da área de medicina. A saída

única da solução é justamente o objetivo final de pesquisa, que é o “resultado da análise (formal) de segurança”.

A primeira etapa de processamento dos elementos do sistema de IA é a “tradução do modelo de IA para ONNX”, que realiza a conversão das ResNets de referência para o *framework* ONNX (*Open Neural Network Exchange*), o qual possui melhores ferramentas para o processo de sobreaproximação do que o paradigma originalmente utilizado (*Keras*). Após essa fase, o modelo traduzido é inserido na “interface com o sistema de IA”, que inicia a sobreaproximação dos batimentos cardíacos. Nela, ocorre o processo de “sobreaproximação do modelo utilizando regras lógico-aritméticas” aproveitando a biblioteca tf2ONNX. Por fim, o resultado é inserido na “interface do modelo de regras lógico-aritméticas”, de onde será encaminhado para a próxima etapa da solução: a “Verificação” por meio da interface “regras lógico-aritméticas sobreaproximadas do sistema de IA”.

Na fase de “Verificação”, o modelo sobreaproximado é utilizado, juntamente com o “modelo de regras lógico-aritméticas do domínio médico”, para a análise formal de segurança. Esta está mapeada nos blocos “execução da verificação do sistema de IA” e “conjunto de regras satisfeitas ou não pelo sistema de IA”, que servem para produzir a saída da verificação formal (“resultado da análise (formal) de segurança”).

A representação estrutural da solução é realizada por meio do diagrama de pacotes da UML (*Unified Modeling Language*) presente na Figura 3. Esse diagrama explicita a organização e as dependências entre os pacotes de software que representam a implementação da solução elaborada. Os pacotes representados na Figura 3 são módulos de software independentes que desempenham a dinâmica descrita na Figura 2 com base nas relações hierárquicas de dependência estabelecidas pelas setas da Figura 3. Uma seta direcionada de um pacote “A” para um pacote “B” indica que o pacote “B” utiliza a infraestrutura do pacote “A” para realizar suas funcionalidades.

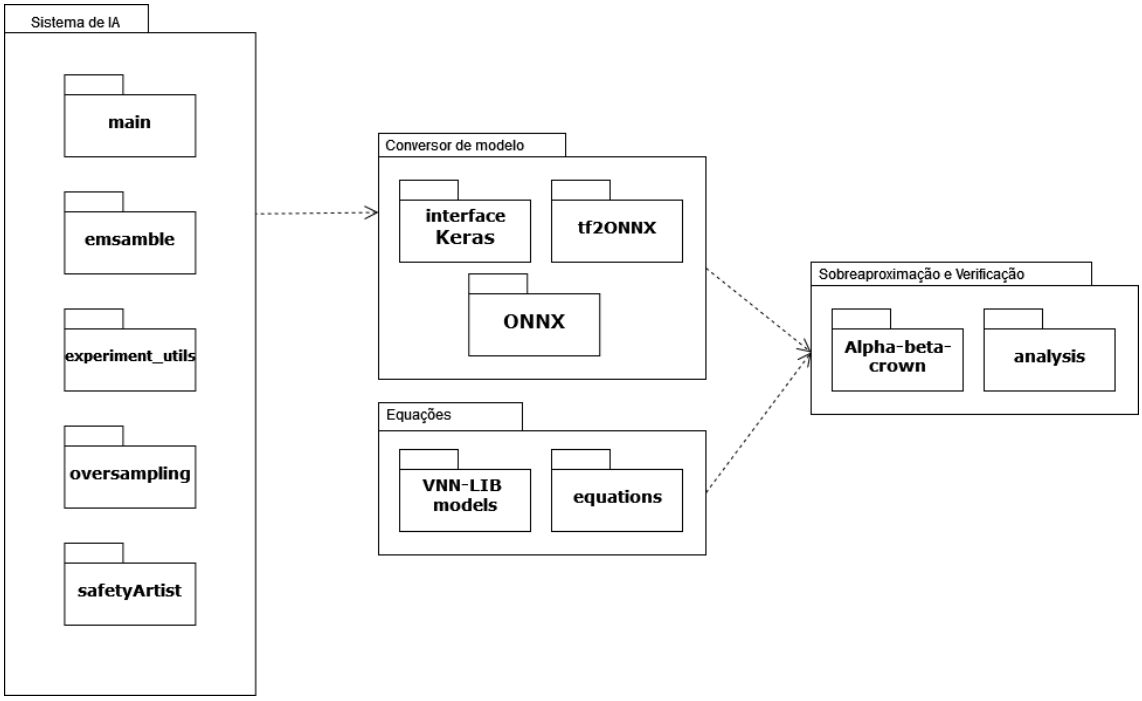

Figura 3 – Diagrama de Pacotes da Arquitetura Estrutural do Sistema

A descrição dos pacotes da Figura 3 é realizada na Tabela 8.

Tabela 8 – Descrição dos Pacotes de Software do Sistema

Pacote	Descrição
Alpha-beta-crown	Módulo que realiza a sobreaproximação e a verificação do sistema de IA em modelo ONNX.
analysis	Módulo que realiza a verificação formal de segurança do modelo lógico-aritmético apresentado, com base nos resultados do módulo Alpha-beta-crown.
emsamble [sic]	Pacote do sistema de IA analisado (SILVA NETO; CUGNASCA, 2023). Contém informações relevantes para a análise em questão.
equations	Material a ser composto pelo conjunto de equações advindas dos modelos teóricos de medicina para classificar arritmias cardíaca. São utilizadas como conhecimento especialista de entrada para modelar o problema.
experiment_utils	Pacote do sistema de IA analisado (SILVA NETO; CUGNASCA, 2023). Contém informações relevantes para a análise em questão.
interface Keras	Módulo que realiza a interface do sistema de IA em Keras para a tradução ONNX.

Pacote	Descrição
Main	Pacote do sistema de IA analisado (SILVA NETO; CUGNASCA, 2023). Contém informações relevantes para a análise em questão
ONNX	Módulo que realiza o processamento de modelos ONNX.
oversampling	Pacote do sistema de IA analisado (SILVA NETO; CUGNASCA, 2023). Contém informações relevantes para a análise em questão
safetyArtist	Pacote projetado em pesquisas anteriores para viabilizar a análise de segurança do sistema de IA (SILVA NETO; CUGNASCA, 2023). Contém informações relevantes para a análise em questão.
tf2ONNX	Módulo que realiza a tradução do modelo Keras para o modelo do tipo ONNX.
VNN-LIB models	Módulo que contém as equações que o sistema de IA deve satisfazer, baseado no módulo “equations”.

Uma vez definidos os pacotes da Figura 3, que integram o sistema de análise formal de segurança, pode-se refinar a descrição comportamental da Figura 2 por meio de diagramas de sequência da UML. Esses diagramas visam representar a dinâmica de troca de informações dos pacotes do sistema, evidenciando como eles contribuem com a dinâmica de processamento prevista para cada requisito funcional.

A ordenação dos diagramas de sequência descritos subsequentemente decorre do fluxo de processamento descrito por meio do diagrama comportamental da Figura 2, começando na fase de “Tradução”, depois “Sobreaproximação” e, por último, “Verificação”. Dessa forma, o detalhamento dos diagramas de sequência é aderente à seguinte ordem de requisitos funcionais: #03, #04, #01, #02 e #00. Os métodos referenciados nos diagramas em sequência são, posteriormente, detalhados na Tabela 9 desta monografia.

O diagrama de sequência do requisito #03 é apresentado na Figura 4. O diagrama indica como a tradução de modelos de IA para o padrão ONNX deve ocorrer por meio dos subpacotes que integram o pacote “Interface Keras” da Figura 3. A sequência desenvolvida para a realização dessa atividade é a passagem do modelo Keras para o pacote “tf2ONNX”, que possui funções para realização dessa função. Na sequência, o modelo convertido é salvo em um arquivo com o uso do pacote “ONNX”.

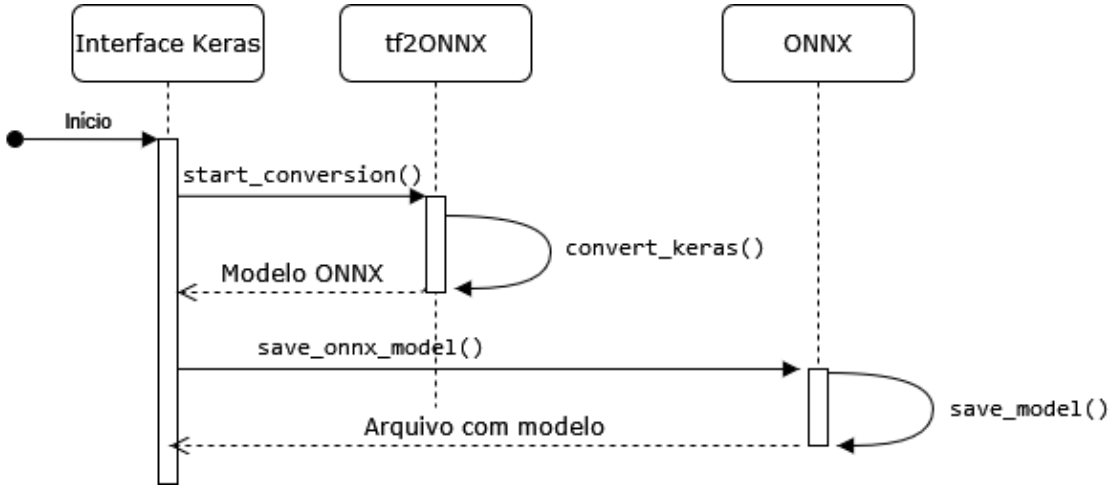

Figura 4 – Diagrama de Sequência para o Requisito #03

O diagrama de sequência do requisito #04 é ilustrado na Figura 5. Nele, é demonstrado o fluxo de verificação da tradução do modelo Keras para o modelo ONNX. Para esse fluxo, é feita a inferência sobre ambos os modelos de IA (Keras e ONNX) e é verificado se eles se demonstram compatíveis. Os pacotes externos do Keras e ONNX são utilizados para realizar a iniciação dos modelos.

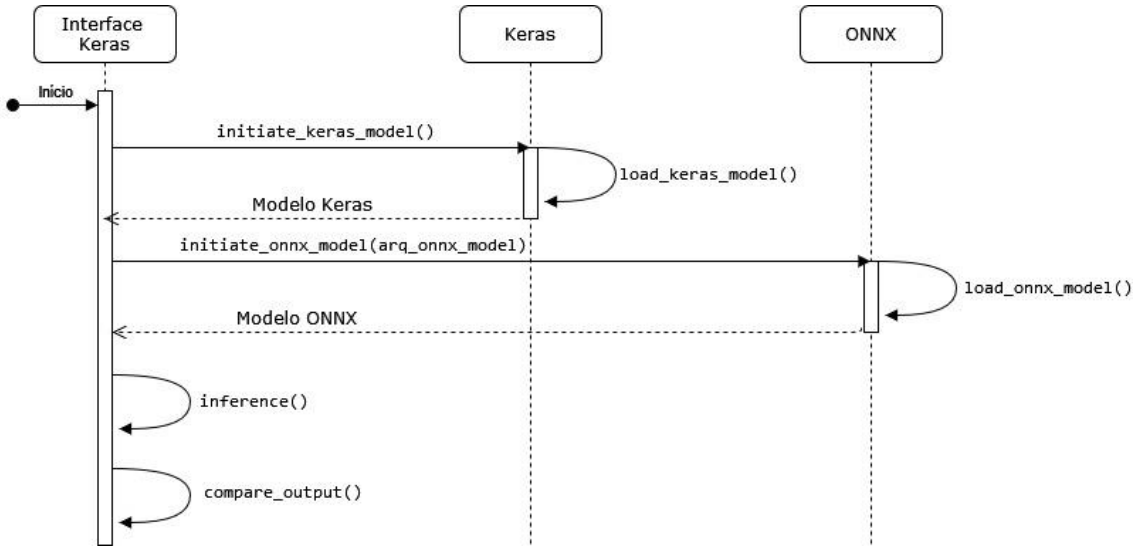

Figura 5 – Diagrama de Sequência para o Requisito #04

O diagrama de sequência dos requisitos #01 e #02 é apresentado na Figura 6. Nele é descrito o fluxo de regressão da IA em modelo ONNX, realizado pelos pacotes “alpha-beta-crown” e “ONNX” da Figura 3. Nesse fluxo, o modelo ONNX é inicialmente carregado para, depois disso, ser regredido e traduzido para regras lógico-aritméticas

descritas em um modelo SMT e verificado utilizando as ferramentas do arcabouço Alpha-beta-crown.

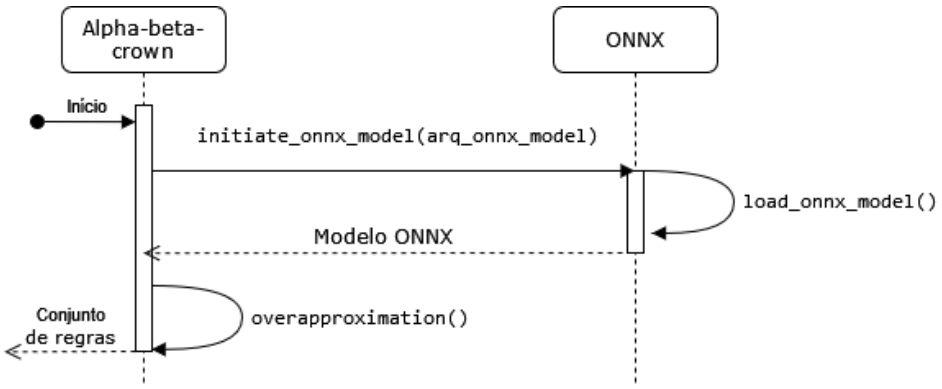

Figura 6 – Diagrama de Sequência para os Requisitos #01 e #02

O diagrama de sequência da Figura 7 representa o requisito #00. Nele é demonstrada a etapa final do sistema – a verificação SMT do sistema, para a análise formal de segurança. Essa verificação é realizada pelo pacote “alpha-beta-crown” da Figura 3, que recebe as informações necessárias para verificação do modelo ONNX e devolve o resultado de satisfatibilidade calculado para o sistema.

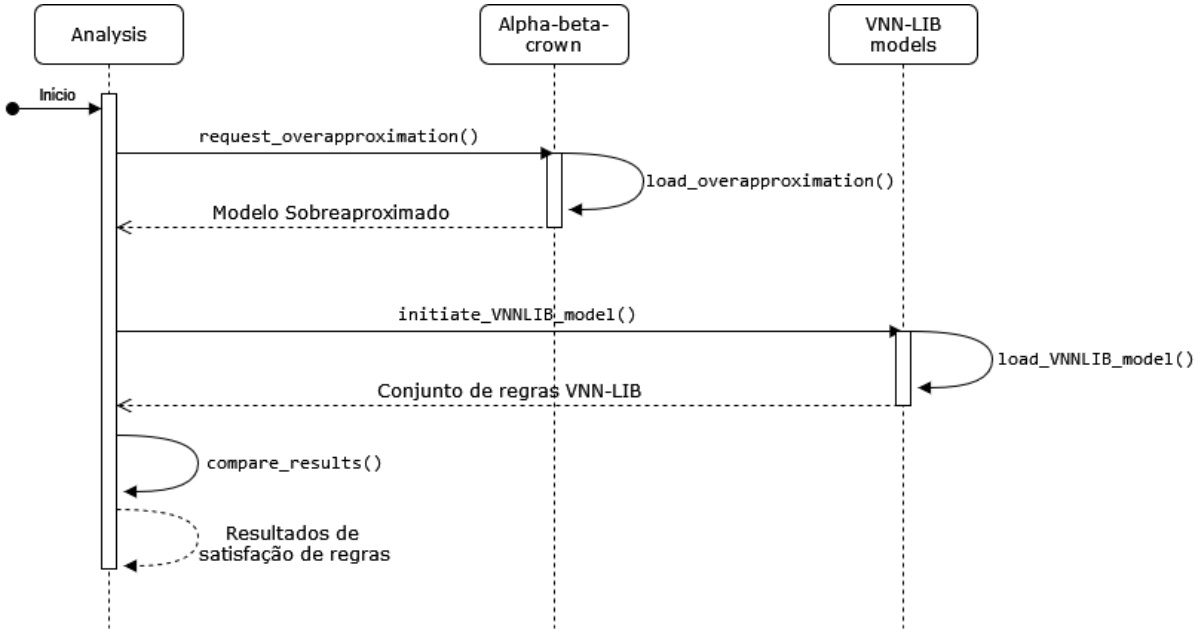

Figura 7 – Diagrama de Sequência para o Requisito #00

Os métodos referenciados nos diagramas de sequência da Figura 4 até a Figura 7 são descritos na Tabela 9. Os métodos foram ordenados de acordo com a ordem de suas primeiras referências entre a Figura 4 e a Figura 7.

Tabela 9 – Descrição dos Métodos Apresentados nos Diagramas de Sequência

Método	Descrição
start_conversion()	Elemento que inicia a conversão da IA de modelo Keras para modelo ONNX
convert_keras()	Elemento do pacote tf2ONNX, implementado com o arcabouço tf2ONNX (THE LINUX FOUNDATION, 2024a) que possui componentes que convertem um modelo Keras em ONNX.
save_onnx_model()	Elemento que inicia o armazenamento da representação ONNX em um arquivo.
save_model()	Elemento do pacote ONNX (THE LINUX FOUNDATION, 2024a) que salva a representação ONNX em um arquivo.
initiate_keras_model()	Elemento que inicia a instanciação da IA no modelo Keras.
load_keras_model()	Elemento do pacote Keras (KERAS TEAM, 2024) que carrega o modelo Keras do sistema original.
initiate_onnx_model (arq_onnx_model)	Elemento que inicia a instanciação da IA no modelo ONNX.
load_onnx_model()	Elemento do pacote ONNX (THE LINUX FOUNDATION, 2024a) que carrega o modelo ONNX extraído da tradução do Keras.
inference()	Elemento que realiza a instanciação de dados e inferência de ambos os modelos ONNX e Keras carregados previamente.
compare_output()	Elemento que avalia se os resultados de ambos os modelos são equivalentes.
overapproximation()	Elemento que realiza a chamada para a sobreaproximação do modelo ONNX.
request_overapproximation()	Elemento do pacote “analysis” utilizado para coletar o resultado de uma sobreaproximação realizada pelo pacote “Alpha-beta-crown”.
load_overapproximation()	Elemento do pacote “Alpha-beta-crown” que devolve o resultado de uma sobreaproximação.
initiate_VNNLIB_model()	Elemento do pacote “analysis” utilizado para requisitar regras lógico-aritméticas de referência do pacote “VNN-LIB models”.

5. PLANEJAMENTO DO PROJETO

Este capítulo da monografia contempla o planejamento do desenvolvimento do projeto e inclui (i.) o cronograma das atividades do projeto e (ii.) os recursos utilizados em seu desenvolvimento.

5.1 CRONOGRAMA DE ATIVIDADES DO PROJETO

O cronograma da Tabela 10 identifica a distribuição mensal das atividades previstas para o projeto, que perdurou desde Dezembro/2023 até Dezembro/2024. As células marcadas com um "**X"** indicam que a atividade de uma linha foi realizada no mês da coluna correspondente (1 – Janeiro até 12 – Dezembro).

Tabela 10 – Cronograma de Atividades do Projeto

Atividade	2023	2024											
	12	01	02	03	04	05	06	07	08	09	10	11	12
I. Familiarização com Temas Introdutórios e Referências do Projeto	X	X											
II. Revisão Sistemática de Literatura		X	X	X	X	X	X	X	X	X	X	X	
III. Especificação do Projeto		X	X	X	X					X	X		
IV. Desenvolvimento do Sistema						X	X	X	X	X	X	X	
IV-a. Implementação do Requisito #00											X	X	
IV-b. V&V do Requisito #00											X	X	
IV-c. Implementação do Requisito #01											X	X	
IV-d. V&V do Requisito #01											X	X	
IV-e.. Implementação do Requisito #02											X	X	
IV-f. V&V do Requisito #02											X	X	
IV-g. Implementação do Requisito #03						X	X	X	X	X			
IV-h. V&V do Requisito #03						X	X	X	X	X			
IV-i. Implementação do Requisito #04						X	X	X	X	X			
IV-j. V&V do Requisito #04						X	X	X	X	X			
IV-k. Implementação do Requisito #05						X	X	X	X	X	X	X	X
IV-l. V&V do Requisito #05						X	X	X	X	X	X	X	X
IV-m. V&V do Sistema Integrado												X	X
V. Documentação Técnica do Projeto	X	X	X	X	X	X	X	X	X	X	X	X	X
VI. Participação no 32º SIICUSP 2024										X	X		
VII. Apresentação Final do Projeto													X

A estratégia de implementação prevista no cronograma consistiu em tratar os requisitos no desenvolvimento pela ordem dos diagramas de sequência apresentados entre a Figura 4 e a Figura 7. Essa estratégia serviu, primeiramente, para facilitar as atividades de V&V (Verificação e Validação) dos requisitos, uma vez que as entradas de algumas partes do sistema dependem de saídas de partes anteriores. Além disso, a estratégia também foi balizada, indiretamente, pelo nível de complexidade das funcionalidades, pois, quanto mais avançado o componente é no fluxo de processamento do sistema, mais complexa tende a ser sua ação.

A estratégia de V&V foi guiada, principalmente, pelos requisitos #02 e #04, que orientam a verificação e a validação de cada bloco funcional do sistema e relacionam-se a funcionalidades e características especificadas nos demais requisitos. Para facilitar o processo de V&V, o desenvolvimento foi ordenado de maneira em que requisitos dependentes começassem a ser desenvolvidos após as suas respectivas dependências já terem sido validadas. Ao final do ciclo de desenvolvimento, também houve uma etapa de V&V integrada para avaliar todos os componentes funcionando em conjunto.

Sobre o requisito não funcional #05, salienta-se que sua asserção somente ocorreu ao final do desenvolvimento. Porém, como ele trata de uma qualidade do sistema como um todo, ele foi ser mantido em vista durante todo o ciclo de vida do sistema, conferindo-lhe desenvolvimento progressivo e contínuo – i.e., em paralelo aos demais requisitos.

Também se destaca que o aluno participou do 32º Simpósio Internacional de Iniciação Científica e Tecnológica da Universidade de São Paulo (SIICUSP) em 23 de outubro de 2024. Essa participação serviu para divulgar o projeto desenvolvido e satisfazer requisitos adjuvantes ao Programa de Pré-Mestrado em Engenharia de Computação, no qual o aluno esteve inscrito durante toda a realização do Trabalho de Conclusão de Curso.

Avalia-se que o cronograma inicialmente idealizado para o projeto requereu ajustes à medida que o desenvolvimento progrediu desde Maio/2024. Os requisitos #03 e #04 tiveram seu desenvolvimento prolongado em dois meses devido à necessidade de aprofundamento de estudos sobre ferramentas de verificação formal

e a atividades administrativas associadas ao programa de Pré-Mestrado. Por conseguinte, a implementação dos requisitos #00 e #01 foi iniciada mais tardiamente e também se estenderam em um mês, atingindo o início de Dezembro/2024.

Avalia-se que, apesar das alterações de cronograma apresentadas previamente, o restante do planejamento foi seguido sem alterações durante todo o desenvolvimento do projeto. Ao final do projeto, e por questões extrínsecas a aspectos controláveis durante sua execução, culminou-se no cumprimento parcial dos objetivos estabelecidos no projeto dentro dos prazos definidos para essa finalidade. Mais detalhes sobre essa análise constam nos capítulos 6 (a partir da seção 6.6), 7 e 8 desta monografia.

5.2 Recursos Técnicos Utilizados no Projeto

A lista a seguir elenca os recursos utilizados no desenvolvimento do projeto. Nesta lista são especificados itens como hardware, bases de dados, documentos técnicos e artefatos de software, como arcabouços (*frameworks*), bibliotecas, linguagens de programação e ambientes de desenvolvimento.

- Biblioteca tf2ONNX (THE LINUX FOUNDATION, 2024b);
- Repositório do Modelo de IA a ser analisado (KOZAL; KSIENIEWICZ, 2019);
- Arcabouço *Alpha-Beta-Crown* (VERIFIED INTELLIGENCE, 2024);
- Arcabouço SMT-LIB (THE SMT-LIB INITIATIVE, 2024);
- Biblioteca ONNX (THE LINUX FOUNDATION, 2024a);
- Biblioteca Keras (KERAS TEAM, 2024);
- Python 3.6 (PYTHON SOFTWARE FOUNDATION, 2016);
- Visual Studio Code (MICROSOFT CORPORATION, 2024);
- Hardware do computador pessoal do aluno:
    - Processador: Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz;
    - Memória RAM DDR4 16 GB;
    - Sistema Operacional: Windows 11 Home 64 bits;
    - Armazenamento: 1 TB SSD.

- Servidores do C2D disponíveis aos alunos bolsistas (autor e colaboradora de Iniciação Científica)[2];
- Base de dados original (KACHUEE; FAZELI; SARRAFZADEH, 2018);
- Repositório e documentação técnica do estudo de caso de referência (SILVA NETO; CUGNASCA, 2023).

Avalia-se que os recursos disponíveis no PC (computador pessoal) do aluno e nos servidores do C2D proveem capacidade computacional suficiente para a realização do projeto.

5.3 CUSTOS DO PROJETO

A Tabela 11 apresenta a descrição dos custos do desenvolvimento do projeto, que inclui a quantificação dos recursos empregados e as respectivas justificativas.

Tabela 11 – Descrição dos Custos do Projeto

Parâmetro	Valor	Justificativa
Esforço de Trabalho (Hora-Pessoa – HP)	Autor Principal: 20HP semanais por 12,5 meses (1000 HP total) Pesquisadora de Iniciação Científica: 10HP semanais por 3,5 meses (140 HP total)	Custo de tempo de desenvolvimento do sistema.
Custo Financeiro para Recursos do Projeto (Reais Brasileiros – BRL)	Nulo	A computação utilizada no projeto advêm de computadores pessoais ou dos servidores do C2D, o que não incrementa os custos do projeto por serem providos gratuitamente. As ferramentas de software utilizadas são *Open Source* e, portanto, não possuem custo de licença.

[2] Acesso garantido aos servidores disponíveis no C2D devido à vinculação do aluno ao Programa de Bolsas do Itaú (PBI) como bolsista de Pré-Mestrado.

Comparativamente ao planejamento inicial, avalia-se que o esforço exigido do autor principal da monografia foi incrementado em 33%, passando de 15HP semanais para 20HP semanais. A pesquisadora de Iniciação Científica, por sua vez, não teve ajustes de sua carga horária. As justificativas já detalhadas na seção 5.1 respaldam o incremento do esforço demandado do autor principal para a realização do projeto.

6. DESENVOLVIMENTO DO PROJETO

Nesta seção da monografia, são relatados os passos executados ao longo do desenvolvimento do projeto.

6.1 MÉTODO DE DESENVOLVIMENTO

O fluxo de desenvolvimento do projeto foi iniciado pela tradução da rede neural do sistema de detecção de arritmias cardíacas para um formato que fosse aceito pelas ferramentas de verificação formal em redes neurais. Após essa etapa, foi realizada a verificação de que seus resultados eram compatíveis com o modelo original.

Na sequência, foi conduzida uma pesquisa para definir uma ferramenta que viabilizaria a verificação formal das redes neurais, tal como previsto para a continuação do desenvolvimento do fluxo de trabalho. Com a escolha dessa ferramenta, procedeu-se à modelagem matemática das restrições de segurança a serem observadas e ao início dos ensaios de análise formal de segurança.

Durante esse processo, constatou-se que a ferramenta selecionada descumpria características da rede neural do sistema exercitado e que essas limitações não haviam sido identificadas ou documentadas pelos seus desenvolvedores. Em razão desse contexto, adotou-se uma estratégia alternativa que, apesar de não exaustiva para a análise formal de segurança, é eficaz em fornecer resultados que identifiquem situações potencialmente inseguras no sistema analisado.

Cada um dos passos prévios do desenvolvimento é explorado em uma das seções subsequentes deste capítulo.

6.2 TRADUÇÃO DO MODELO DE IA PARA ONNX

Como primeira etapa do desenvolvimento do projeto, foi implementado o componente de tradução do modelo para a linguagem de descrição ONNX, que consiste em um padrão *de facto* para a representação de redes neurais para verificações formais de segurança. Essa implementação foi realizada com ferramentas do arcabouço tf2ONNX e consistiu em: realizar o treinamento da rede

neural, salvar em um arquivo *keras_model.keras* o modelo resultante, utilizar a ferramenta *convert.from_keras* do arcabouço e salvar o resultado no arquivo *onnx_model.onnx*.

Ademaisessa conversão foi realizada somente para a rede neural treinada com a técnica de *oversampling* SMOTE, sem o tratamento de *data augmentation,* que apresentou melhor desempenho geral entre os modelos. Essa decisão foi tomada devido ao elevado custo computacional deste passo, o que resultaria em um grande consumo de tempo, sem maiores benefícios nesta etapa do projeto, se aplicado a sistemas sabidamente de desempenho e segurança inferiores ao escolhido.

Os resultados estão presentes no repositório público do projeto no GitHub (STEPHANO SANTOS; MURAD, 2024).

6.3 VERIFICAÇÃO DA TRADUÇÃO DO MODELO DE IA PARA ONNX

Como passo subsequente da tradução do modelo de IA para ONNX, verificou-se se o modelo traduzido realmente corresponde ao modelo original. Para tanto, foram analisadas as características principais de uma rede neural: a arquitetura da rede (quantidade e tipos de camadas), os tensores que representam os pesos dos neurônios e as métricas de desempenho da rede para dado conjunto de teste.

A primeira etapa consiste em caracterizar a arquitetura das redes neurais em ambas as especificações, para garantir que as mesmas operações matemáticas são realizadas igualmente em ambas. Para tanto, foi utilizado o aplicativo Netron (ROEDER, 2024) a fim de ilustrar as camadas de cada especificação. Na Figura 8 e na Figura 9 são apresentadas as estruturas de camadas que estão presentes na rede original em Keras e as representações equivalentes na tradução ONNX. As relações de equivalência são descritas após as figuras.

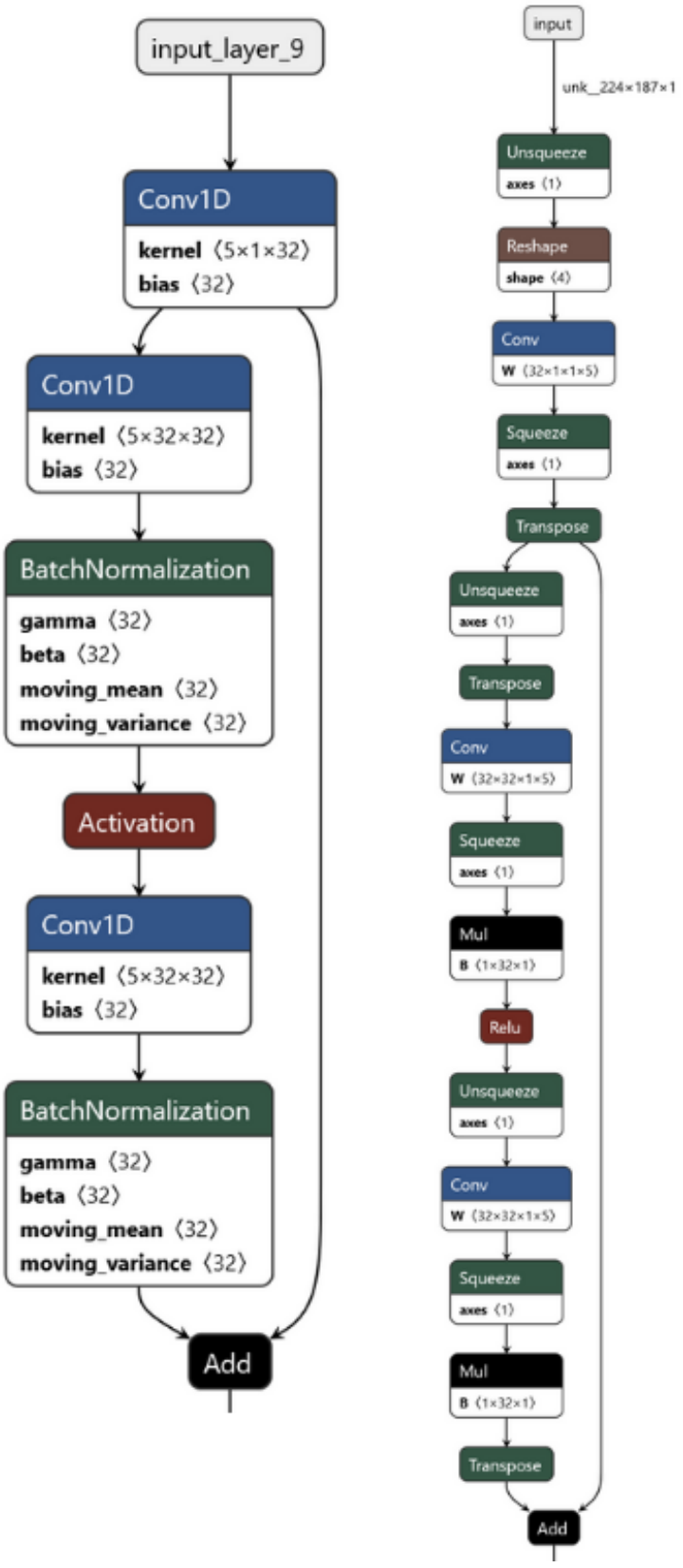

Figura 8 – Arquitetura das Camadas Menos Profundas do Modelo de IA Keras (esquerda) e ONNX (direita)

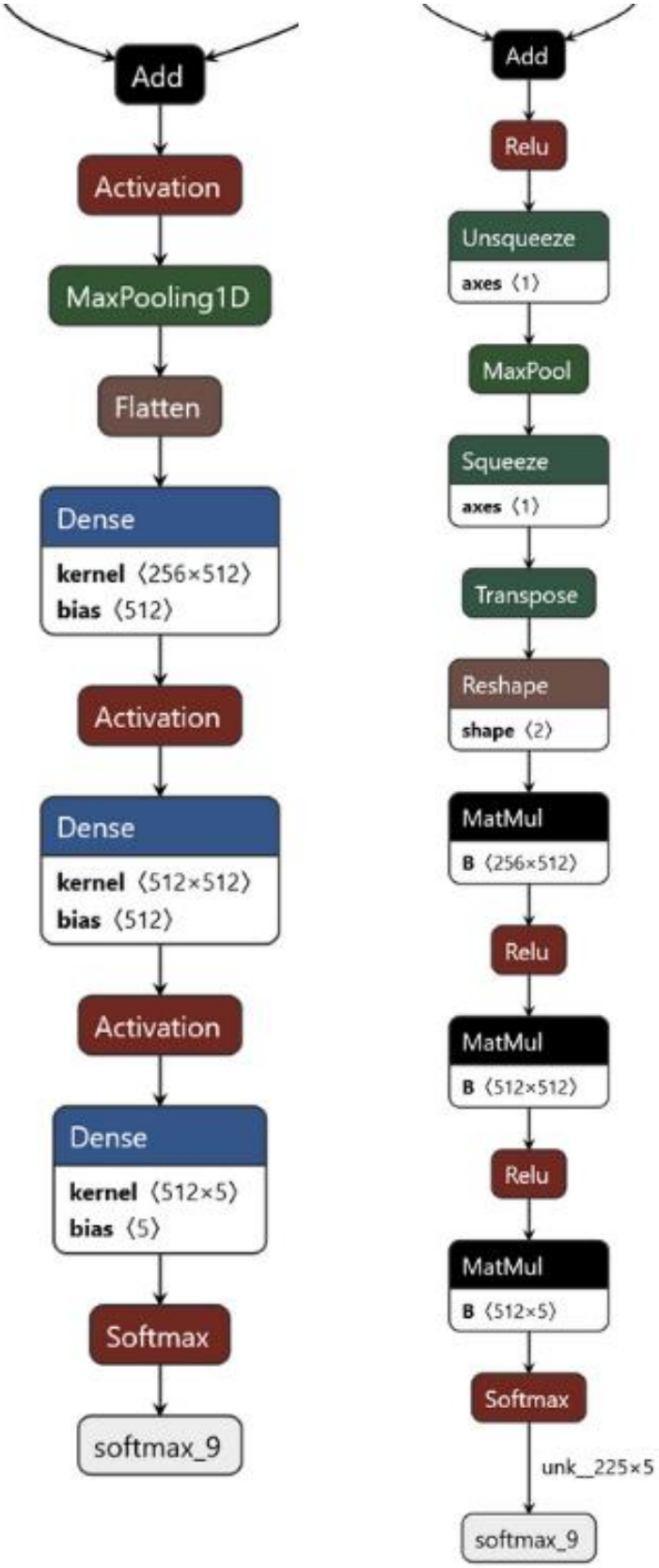

Figura 9 – Arquitetura das Camadas Mais Profundas do Modelo de IA Keras (esquerda) e ONNX (direita)

Aprofundando o detalhamento, pode-se perceber que uma camada *Conv1D* do Keras é convertida em um conjunto *Unsqueeze Reshape Conv Squeeze Transpose* na especificação ONNX, efetivamente realizando a mesma operação com os mesmos parâmetros. Além disso, também pode-se destacar que a operação ONNX *Mul* é equivalente à operação Keras *Batch Normalization,* que a operação ONNX *ReLu* é equivalente à operação Keras *Activation* e que conexões entre camadas pernameceram idênticas entre os modelos.

Por último, a parte final das redes, com camadas densamente conectadas, ativação *ReLu* e operação *Softmax*, também são explicitas em ambas as especificações. A operação *MatMul* do ONNX pode ser utilizada como equivalente para o resultado de uma camada densa, pois ambas podem ser matematicamente descritas pela operação $y = W * x + b$, em que x é o vetor de entrada, y é vetor de saída, W é a matriz com os pesos dos neurônios e o b é o *bias*, componente de adição da operação.

Em vista dessas equivalências analisadas, é possível afirmar que ambas as arquiteturas são homomórficas, assegurando que esse passo foi realizado corretamente.

Após isso, pode-se ler o conteúdo dos tensores de camada para verificar se as características dos neurônios de cada camada também são equivalentes entre os modelos Keras e ONNX. Esses pesos são armazenados nos arquivos em que os modelos são salvos e podem ser acessados e lidos através de mecanismos dos arcabouços, como é demonstrado na Figura 10 para o modelo Keras e na Figura 11 para o modelo ONNX. Dessa forma, foi desenvolvida uma ferramenta que converte esses valores para um mesmo formato e os compara para verificar se os valores são coerentes entre os modelos. Essa ferramenta está disponível no diretório *weights_test* do repositório GitHub do projeto (STEPHANO SANTOS; MURAD, 2024).

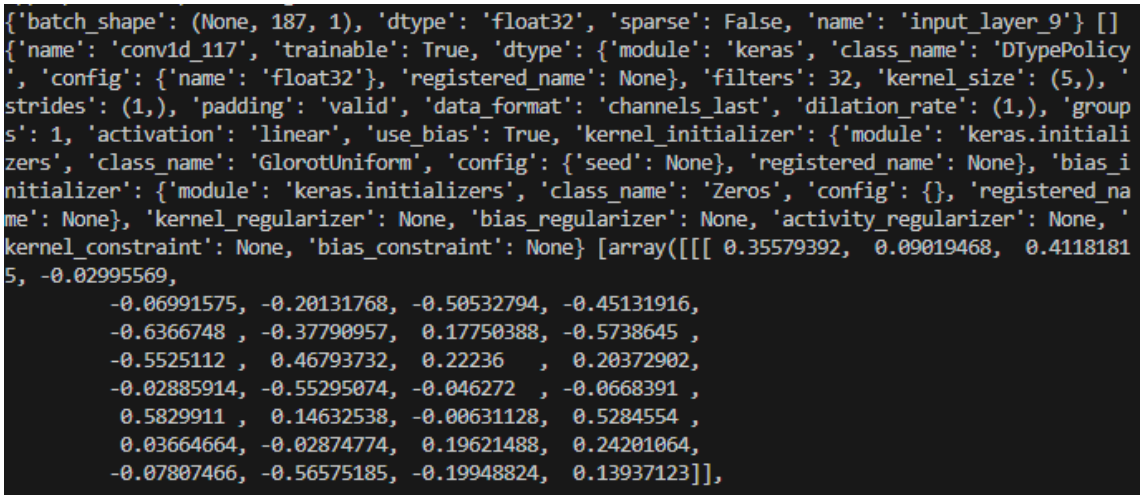

Figura 10 – Exemplo de Representação dos Tensores do Modelo Keras

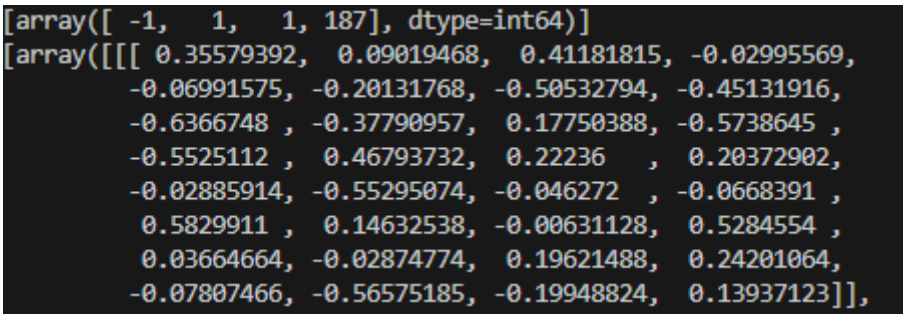

Figura 11 – Exemplo de Representação dos Tensores do Modelo ONNX

Utilizando as ferramentas de extração dos pesos das camadas, os vetores de pesos gerados foram comparados, e os resultados dessa atividade demonstraram que os pesos das camadas de neurônios, costumeiramente chamados de tensores, foram conservados entre a conversão de modelos. Logo, o fluxo desenvolvido passou na segunda camada de verificação.

Por último, verificou-se se as inferências em ambas as redes possuem resultados equivalentes, i.e., se os resultados produzidos pelas duas redes são coerentes dado um conjunto de dados de testes, presente no diretório *translation_tests* do repositório GitHub (STEPHANO SANTOS; MURAD, 2024). Esse conjunto de testes é gerado a mesma técnica de SMOTE com separação em *folds* utilizada durante o treinamento. Para tanto, foi realizada a inferência no conjunto dos dados foi é verificado se os modelos apresentaram, sem grandes mudanças, as mesmas classificações. É possível avaliar essa informação através das métricas de exatidão (*accuracy*), precisão (*precision*) e revocação (*recall*), dado que elas avaliam justamente a qualidade das classificações realizadas.

O resultado deste último teste, apresentado no diretório *accuracy_tests_results* do repositório GitHub (STEPHANO SANTOS; MURAD, 2024), foi o de que, para as métricas avaliadas, ambas as redes apresentavam exatamente os mesmos índices até a sexta casa decimal, evidenciando a coerência dos resultados entre os modelos Keras e ONNX traduzido. Por conseguinte, os resultados dos três passos prévios indicam que o processo de tradução não causou alterações no modelo de redes neurais, sugerindo a equivalência funcional do modelo ONNX traduzido e a viabilidade de uso dele no restante do projeto.

6.4 Análise Comparativa de Ferramentas para Verificação Formal

Juntamente com a realização da tradução do modelo de redes neurais para a especificação ONNX, foi realizada uma pesquisa para definir as ferramentas de verificação formal em redes neurais mais adequadas para o projeto. O detalhamento dessa atividade foi compilado em um relatório disponibilizado na plataforma *Zenodo* (STEPHANO SANTOS; SILVA NETO; CUGNASCA, 2024) e comentado sucintamente neste texto.

Essa análise foi iniciada com uma revisão de literatura pautada em referências do trabalho inspirador da pesquisa (SILVA NETO; CUGNASCA, 2023), com uma busca de termos relevantes em bases indexadoras de trabalhos científicos (STEPHANO SANTOS; SILVA NETO; CUGNASCA, 2024) e a avaliação da maior conferência de verificação em redes neurais, a *International Verification of Neural Networks Competition* (VNNCOMP) (BRIX et al., 2023). Após essa etapa, foram delimitados os requisitos que as ferramentas deveriam atender para se adequar ao projeto e quais características seriam diferenciais desejáveis, mas não obrigatórios.

Dessa forma, foi realizada uma análise comparativa entre as ferramentas identificadas na busca, separando-as em classificações a respeito da utilidade da ferramenta para o projeto de pesquisa como indicado na Tabela 12, que resume os achados da atividade. Entre as ferramentas analisadas, *alpha-beta-crown* foi a que se mais se destacou, sendo escolhida para utilização para as atividades seguintes de verificação formal.

Tabela 12 – Resultados da Análise Comparativa de Ferramentas para Verificação Formal em Redes Neurais

Ferramenta	Classificação	Justificativa
Alpha-Beta-Crown (VERIFIED INTELLIGENCE, 2024)	C0	Vencedora todas as edições da VNNCOMP e apresenta todos os requisitos necessários para utilização no projeto, além de demonstrar maior eficiência computacional do que a competição e compatibilidade com as tecnologias utilizadas no sistema.
NNV (VERIVITAL, 2024)	C1	Ferramenta atinge os requisitos necessários para sistema e são substitutas viáveis à ferramenta C0 para a utilização no sistema, sendo as melhores de suas categorias.
Polar (HUANG et al., 2023)		
Verisig (IVANOV et al., 2019)	C2	Ferramenta atinge os requisitos básicos, porém tem menor prioridade, pois há ferramentas que conseguiriam preencher seu papel de maneira mais efetiva.
NeVer (DEMARCHI et al., 2024)		
CAISAR (CAISAR, 2024)		
QNNVerifier (SONG et al., 2022)		
ERAN (MÜLLER et al., 2022)		
Neurodiff (PAULSEN et al., 2020)		
DeepPoly (POBITZER; FRISCHKNECHT, 2022)	C3	Ferramenta não atinge os requisitos básicos e possui algum interesse limitado em uso complementar – como, por exemplo, em comparação de métodos de verificação
HUDD (FAHMY; PASTORE; BRIAND, 2022)		
DualApp (XUE et al., 2023)		
Auto-LiRPA (XU et al., 2021)		
*ReachNN** (FAN et al., 2020)	C4	Ferramenta não atinge os requisitos básicos do projeto.

Ferramenta	Classificação	Justificativa
PEREGRiNN (KHEDR; FERLEZ; SHOUKRY, 2021)		
nnenum (BAK, 2021)		
Overt (SIDRANE et al., 2021)		

6.5 Modelagem Matemática do Comportamento Seguro da Rede Neural

Para a realização da modelagem do comportamento seguro da rede neural, devem ser elaborados arquivos ".vnnlib" que possuam as propriedades a serem verificadas na sobreaproximação a ser produzida no arcabouço "*alpha-beta-crown*". A elaboração desses arquivos consiste em duas etapas, a declaração das variáveis de entrada e saída e as afirmações que devem ser satisfeitas pelas variáveis para que a propriedade seja verificada.

Esses arquivos seguem a formatação estabelecida pelo padrão VNN-LIB (GUIDOTTI et al., 2022), utilizado pela VNNCOMP e que serve de referência para os programas identificados na avaliação de ferramentas. A linguagem de especificação de propriedades a serem satisfeitas utilizada é a SMT-LIB, utilizada como padrão para representar problemas que envolvem soluções com SMT, e as propriedades de interesse são definidas utilizando-se operações lógicas e aritméticas (AND, OR, soma, subtração e comparações) em notação prefixada – ou polonesa direta (PN – *Polish Notation*).

Em um primeiro momento, foi desenvolvido um arquivo contando com propriedades básicas das entradas e saídas da rede neural, com as variáveis e afirmações (*asserts*) a serem satisfeitas por elas, tal como declarado no trecho ilustrado na Figura 12. Dessa forma, foi verificado se as entradas e saídas obedecem aos limites numéricos entre 0 e 1, a fim de aferir o funcionamento da ferramenta "*alpha-beta-crown*" no sistema de redes neurais objeto da análise.

Esses limites decorrem do fato de as amostras dos batimentos cardíacos (entradas) serem números no intervalo [0; 1] e de as saídas serem probabilidades de cinco categorias de batimentos cardíacos. Assume-se que as cinco categorias são

disjuntas, portanto, a soma das saídas iguala-se a um. Também cabe reforçar que a propriedade da saída é verificada utilizando lógica negativa, pois a ferramenta *alpha-beta-crown,* ao trabalhar com a verificação por contraprova, requer que as condições de saída sejam complementares às situações corretas.

```
; Definição das entradas (série temporal com 187 entradas)

(declare-const X_0 Real)
; Restrições de entrada (inputs no intervalo [0, 1])

(assert (<= X_0 1))
(assert (>= X_0 0))
```

Figura 12 – Exemplo de Verificação de Propriedade

Para o desenvolvimento das propriedades em si, elas foram primeiramente foram elaboradas com base nas propriedades definidas no trabalho motivador da pesquisa (SILVA NETO; CUGNASCA, 2023). Ressalta-se que, tal como definido nos objetivos deste projeto, as propriedades observadas estão apenas relacionadas às diferenciações entre batimentos cardíacos com a classificação “normal”, representada como classificação “0” pelo sistema de detecção de arritmias, e quaisquer outras classificações de arritmias[3].

6.6 Problemas com a Ferramenta de Verificação Formal

Durante as primeiras iterações do processo de verificação formal, foram identificados problemas de compatibilidade entre a ferramenta de análise *aplha-beta-crown* e a rede neural alvo, além de limitações intrínsecas à ferramenta utilizada. Esses problemas e limitações não foram identificados durante a pesquisa relatada na seção 6.4 porque não constam na documentação técnico-científica da ferramenta, como artigos científicos e relatórios técnicos (VERIFIED INTELLIGENCE, 2024; XU et al., 2021).

Primeiramente, embora a linguagem de especificação VNNLIB ofereça suporte a operações aritméticas para a definição de cláusulas de contraprova, os operadores

[3] Diferenciar arritmias distintas pertence ao escopo de trabalho futuro, a ser elaborado em pesquisa de Mestrado do autor teste Trabalho de Conclusão de Curso a partir de 2025.

de soma e subtração não foram implementados na ferramenta *alpha-beta-crown*. Essa limitação resultou em dificuldades adicionais, exigindo modificações na rede neural, com a adição de alguns vértices (neurônios) de saída para realizar operações aritméticas entre entradas e saídas. Essas alterações viabilizam o processo de verificação e como elas foram realizadas com a adição de novas saídas, inspecionadas como corretas e que não influenciam o fluxo de processamento normal da rede, assegura-se que a alteração não afeta a verificação formal de segurança.

Além disso, características específicas da arquitetura da rede neural analisada dificultaram, em um primeiro momento, a execução da verificação formal. Um exemplo notável é a exigência técnica da ferramenta *alpha-beta-crown* de que, nas camadas convolucionais, os hiperparâmetros *kernel_size* e *stride* sejam iguais, condição que não é atendida pela rede neural alvo da análise. É válido registrar que na documentação atual da ferramenta essa restrição não é citada em nenhum momento.

Esses hiperparâmetros determinam como a operação de convolução é realizada pela camada: *kernel_size* determina o tamanho do filtro de convolução que varre a entrada, enquanto *stride* determina o passo com o qual o filtro de convolução passa pelas entradas. Embora essas hiperparâmetros estejam relacionados um ao outro, inclusive determinando o tamanho da saída da camada, a condição de igualdade entre elas não é usual, pois normalmente esses hiperparâmetros são utilizados para manipular aspectos diferentes da rede. Por exemplo, o *stride* é manipulado para diminuir a dimensionalidade da rede, enquanto o *kernel_size* determina o quanto de contexto de vizinhos é utilizado pela convolução.

Ademais, foi garantido, em comunicação com o grupo responsável pela manutenção da ferramenta, que todas as operações envolvendo ResNets, como é o caso da rede utilizada, deveriam ser cobertas pela ferramenta *alpha-beta-crown*. Porém, a ferramenta gerou um erro ao processar uma operação de *Bounded_mult,* que se trata de uma multiplicação com limites superiores e inferiores utilizada na camada *Mat_Mul* da Figura 9 para multiplicação matricial. Esse erro está em processo de investigação por um integrante da equipe de desenvolvimento da ferramenta *alpha-beta-crown* devido à anomalia detectada, e o histórico dessa comunicação é relatado no Apêndice A.

6.7 ARQUIVOS DE VERIFICAÇÃO INICIAIS

Inicialmente, os arquivos com as restrições de segurança a serem verificadas foram elaborados utilizando as propriedades apontadas pelo trabalho motivador da pesquisa (SILVA NETO; CUGNASCA, 2023). Foram selecionados os critérios A até H da Tabela 13, que representam casos de fronteira de confusões entre a categoria “0” ou “saudável” e outras categorias de arritmias.

Os cenários tratados correspondem a situações tanto em que um batimento cardíaco saudável assemelha-se a uma arritmia quanto em que um batimento cardíaco com arritmia assemelha-se a um batimento cardíaco saudável. Considera-se que essas situações de “confusão” entre categorias são casos de fronteira críticos para a avaliação de segurança.

Tabela 13 – Detalhamento das Condições de Confusão dos Casos de Fronteira Envolvendo Batimentos Cardíacos Saudáveis (categoria “0”) e Arritmias (categorias “1” a “4”)

Fonte: Adaptado de Silva Neto e Cugnasca (2023)

Cenário de Confusão para Caso de Fronteira	Descrição dos Critérios de Análise
A – Batimento da categoria “0” parecido com “1”	As quatro condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Pelo menos uma das amostras 3 a 9 é menor do que 0,2; • Condição B: Pelo menos uma das amostras 30 a 45 é maior do que 0,3; • Condição C: Pelo menos uma das amostras 75 a 90 é maior do que 0,25; • Condição D: O batimento cardíaco deve possuir classificação igual a “0” na base de dados de referência.

Cenário de Confusão para Caso de Fronteira	Descrição dos Critérios de Análise
B – Batimento da categoria “1” parecido com “0”	As quatro condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Nenhuma das amostras 3 a 9 é menor do que 0,2; • Condição B: Nenhuma das amostras 30 a 45 é maior do que 0,3; • Condição C: Nenhuma das amostras 75 a 90 é maior do que 0,25; • Condição D: O batimento cardíaco deve possuir classificação igual a “1” na base de dados de referência.
C – Batimento da categoria “0” parecido com “2”	As três condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Todas as amostras de 1 a 3 são menores do que 0,7; • Condição B: Todas as amostras de 30 a 45 são maiores do que 0,3; • Condição C: O batimento cardíaco deve possuir classificação igual a “0” na base de dados de referência.
D – Batimento da categoria “2” parecido com “0”	As três condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Pelo menos uma das amostras 1 a 3 é maior do que 0,7; • Condição B: Todas as amostras 30 a 45 são menores do que 0,3; • Condição C: O batimento cardíaco deve possuir classificação igual a “2” na base de dados de referência.
E – Batimento da Categoria “0” parecido com “3”	As três condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Todas as amostras de 10 a 25 são menores do que 0,2; • Condição B: Todas as amostras de 50 a 75 são maiores do que 0,29; • Condição C: O batimento cardíaco deve possuir classificação igual a “0” na base de dados de referência.

Cenário de Confusão para Caso de Fronteira	Descrição dos Critérios de Análise
F – Batimento da categoria “3” parecido com “0”	As três condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Todas as amostras de 15 a 26 são maiores do que 0,2; • Condição B: Pelo menos uma das amostras de 50 a 75 é menor do que 0,29; • Condição C: O batimento cardíaco deve possuir classificação igual a “3” na base de dados de referência.
G – Batimento da categoria “0” parecido com “4”	As três condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Todas as amostras de 9 a 14 são menores do que 0,2; • Condição B: Pelo menos 70 das 86 amostras do intervalo [20; 105] são maiores do que 0,3; • Condição C: O batimento cardíaco deve possuir classificação igual a “0” na base de dados de referência.
H – Batimento da categoria “4” parecido com “0”	As três condições a seguir devem ser simultaneamente satisfeitas: • Condição A: Todas as amostras de 3 a 16 são maiores do que 0,16; • Condição B: Menos de 70 das 86 amostras do intervalo [20; 105] são maiores do que 0,3; • Condição C: O batimento cardíaco deve possuir classificação igual a “4” na base de dados de referência.

Na Figura 13, é exemplificada a especificação uma parte da propriedade das entradas utilizada como critério para averiguar a “Condição B” do critério “F” da Tabela 13. É observado que as características seguem a mesma especificação da “Condição B” do critério “F” da Tabela 13.

Na Figura 14, por sua vez, é exemplificada uma propriedade da saída do critério “F”, que corresponde ao batimento cardíaco ter maior probabilidade de ser da arritmia tipo 3 do que de qualquer outro tipo. Nesse caso, como a propriedade especificada deve ser aderente à lógica de contraprova de uma situação insegura, as características especificadas correspondem ao contrário do desejado no critério “F”. Em outras palavras, a lógica especificada na Figura 14 prevê que sejam identificadas

as situações inseguras de a arritmia do tipo 3 (Y_3) ser menos provável do que qualquer outro tipo de batimento cardíaco – seja ele normal (Y_0) ou outra arritmia (Y_1, Y_2 ou Y_4).

```
(assert (or
    (and (<= X_50 0.29))
    (and (<= X_51 0.29))
    (and (<= X_52 0.29))
    (and (<= X_53 0.29))
    (and (<= X_54 0.29))
    (and (<= X_55 0.29))
    (and (<= X_56 0.29))
    (and (<= X_57 0.29))
    (and (<= X_58 0.29))
    (and (<= X_59 0.29))
    (and (<= X_60 0.29))
    (and (<= X_61 0.29))
    (and (<= X_62 0.29))
    (and (<= X_63 0.29))
    (and (<= X_64 0.29))
    (and (<= X_65 0.29))
    (and (<= X_66 0.29))
    (and (<= X_67 0.29))
    (and (<= X_68 0.29))
    (and (<= X_69 0.29))
    (and (<= X_70 0.29))
    (and (<= X_71 0.29))
    (and (<= X_72 0.29))
    (and (<= X_73 0.29))
    (and (<= X_74 0.29))
    (and (<= X_75 0.29))
))
```

Figura 13 – Exemplo de Propriedade Utilizada para Verificar a "Condição B" do Critério F da Tabela 13

```
(assert (or
    (and (>= Y_0 Y_3))
    (and (>= Y_1 Y_3))
    (and (>= Y_2 Y_3))
    (and (>= Y_4 Y_3))
))
```

Figura 14 - Exemplo de Propriedade de Saída para Verificar Contraprovas do Critério F da Tabela 13

Para a elaboração dos critérios G e H da Tabela 13, foi necessário adicionar à rede neural grafos computacionais de saída que recebem diretamente a operação de média de algumas entradas, para que algumas operações pudessem ser realizadas. Essas alterações estão explícitas no diagrama da Figura 15, com a camada de entrada em verde, o fluxo da rede neural em azul e as alterações em vermelho. Assim, é demonstrado que essas modificações são apenas saídas adicionais que aferem operações aritméticas entres as entradas e que, portanto, não representaram mudanças na estrutura da rede e nem afetam a validade dos passos de verificação relatados na seção 6.3. Adicionalmente, as novas saídas também foram inspecionadas e comprovadas como funcionalmente corretas.

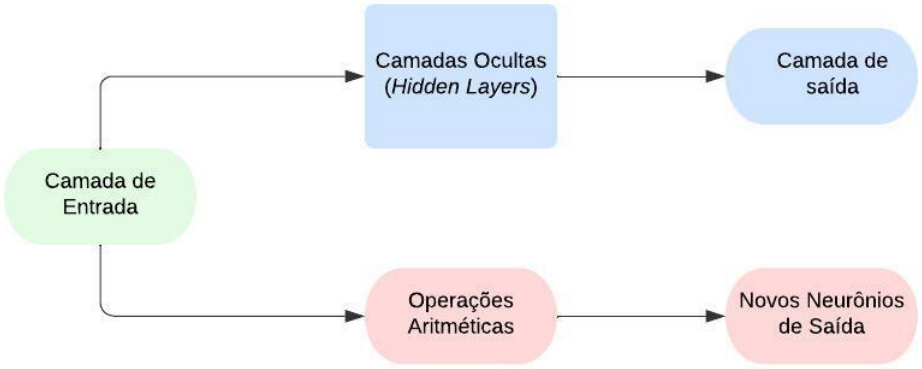

Figura 15 – Diagrama de Fluxo de Processamento da Rede Neural e das Camadas Aritméticas Adicionais para o Tratamento de Condições das Entradas

6.8 VERIFICAÇÃO POR ATAQUES ADVERSÁRIOS

Devido às limitações identificadas na ferramenta de verificação formal *alpha-beta-crown*, cuja resolução depende de interações com a equipe que projeta a ferramenta e é planejada para trabalhos futuros, optou-se por adotar uma estratégia alternativa baseada em ataques adversários. Essa abordagem, também implementada na ferramenta utilizada, permitiu a obtenção de resultados iniciais de análise formal de segurança, apesar de suas limitações.

A técnica de ataques adversários (CARLINI; WAGNER, 2017) consiste em introduzir pequenas perturbações nos dados de entrada do modelo. Embora essas perturbações sejam, idealmente, de baixa significância dentro do domínio das

entradas, elas provocam alterações significativas nas classificações geradas pela rede neural, indicando uma vulnerabilidade em contextos em que essas mudanças não deveriam ocorrer. Essa abordagem é amplamente empregada, por exemplo, para avaliar a robustez de redes neurais convolucionais voltadas para a classificação de imagens, pois utiliza ruídos imperceptíveis ao olho humano para identificar taxas de erro elevadas em situações adversas (CARLINI; WAGNER, 2017).

Na ferramenta *alpha-beta-crown*, o mecanismo utilizado para realizar os ataques adversários é o *Projected Gradient Descent* (PGD) (MADRY et al., 2018). Essa técnica baseia-se no algoritmo de gradiente descendente, amplamente aplicado no aprendizado de máquina, para otimizar a perturbação necessária à indução de erros na rede.

Como resultado do ataque adversário, a ferramenta *alpha-beta-crown* informa se alguma contraprova foi encontrada, apresentando, quando aplicável, os dados de entrada utilizados para gerá-la. Além disso, é produzido um arquivo contendo o registro detalhado dos ataques realizados.

Uma das grandes limitações dessa estratégia de verificação de robustez é que, além de a sua execução exigir elevada quantidade de recursos computacionais, muitas vezes os exemplos gerados não correspondem a cenários que poderiam ocorrer na realidade, justamente pelo fato de as perturbações introduzidas serem aleatórias. Por exemplo, dentro do contexto deste trabalho, possíveis ataques adversários podem correspondem a formas de onda que não representam nenhum tipo de batimento cardíaco esperado para um ser humano vivo, como linhas retas ou com mínimas oscilações.

Situações como a descrita anteriormente podem ser minimizadas, por exemplo, limitando a faixa da perturbação inserida. Contudo, ainda com essa precaução, há a possibilidade de que os ataques simulados não correspondam a situações reais devido à sensibilidade dos ataques adversários às condições aleatórias de iniciação. Por fim, embora o uso do algoritmo PGD reduza significativamente esse impacto, devido à exploração do panorama de perturbações (*perturbation landscape*) que ele realiza inspirado no gradiente descendente, ele ainda está presente (MADRY et al., 2018), o que torna a verificação de segurança incompleta por dois motivos:

a) Uma situação insegura pode ser detectada com um ataque adversário de entrada sem sentido físico da aplicação;
b) Deixa-se de se exercitar ataques adicionais em condições mais próximas das que permeiam o real contexto da aplicação.

Também se salienta o fato de o funcionamento dos ataques adversários ser independente das características da arquitetura da rede neural, o que o torna facilmente generalizável. Isso permite que, mesmo com as restrições da ferramenta *alpha-beta-crown*, essa parte da execução possa ser utilizada para obtenção de resultados de análise formal de segurança da rede neural do sistema de detecção de arritmias cardíacas.

7. RESULTADOS OBTIDOS E ANÁLISE

Neste capítulo da monografia, detalham-se os resultados obtidos após o desenvolvimento do projeto. Esse detalhamento é realizado apresentando-se os resultados atingidos e discutindo-os sob duas óticas: (i.) avaliar a adequação da análise formal de segurança do sistema de detecção de arritmias cardíacas e (ii.) discutir o grau de cumprimento dos requisitos funcionais e não funcionais estabelecidos no capítulo 3.

7.1 Resultados da Análise Formal de Segurança

Inicialmente, o cenário “F” da Tabela 13 (seção 6.7) foi considerado como prova de conceito para os ataques adversários. Ele foi selecionado por ser relativamente simples de representar e por ter conduzido a resultados mais claros do que outros cenários exercitados em caráter experimental. Por restrições de tempo e dos recursos da ferramenta *alpha-beta-crown*, não foi possível desenvolver todos os casos de fronteira com resultados satisfatórios. Tal como comentado posteriormente nesta seção, há ataques adversários que não tiveram resultados compatíveis com batimentos cardíacos reais (i.e., representam problemas de segurança implausíveis em aplicações reais).

Os arquivos resultantes da execução dos ataques adversários estão registrados no repositório do projeto, na pasta “*PGD results*” (STEPHANO SANTOS; MURAD, 2024). Os arquivos de verificação utilizados para produzir esses resultados, a rede neural utilizada e os arquivos de configuração também se encontram no mesmo repositório, nas pastas “*VNNLIB files*”, “*translated model*” e “*config files*”, respectivamente (STEPHANO SANTOS; MURAD, 2024).

Pelos resultados obtidos, são observadas algumas inconsistências entre as saídas obtidas pela rede e os resultados esperados, indicando que, em alguns casos, há o potencial de confusão entre categorias e, por conseguinte, situações inseguras decorrentes da classificação de um indivíduo com arritmia como saudável. Todavia, alguns contraexemplos obtidos se encaixam na categoria de potencialmente não representarem sinais reais.

No exemplo ilustrado na Figura 16, o sinal obtido como contraprova insegura pelo ataque adversário utilizando o critério F da Tabela 13 provocou a troca de classificações na rede, mas é anômalo se comparado a um batimento cardíaco real, com vales e picos retangulares e múltiplas amostras constantes em 0,5. Dessa forma, apesar de ser uma contraprova insegura, o resultado é implausível em um contexto de uso real e evidencia que nem todo ataque adversário traz evidências relevantes para uma análise de segurança crítica *(safety).*

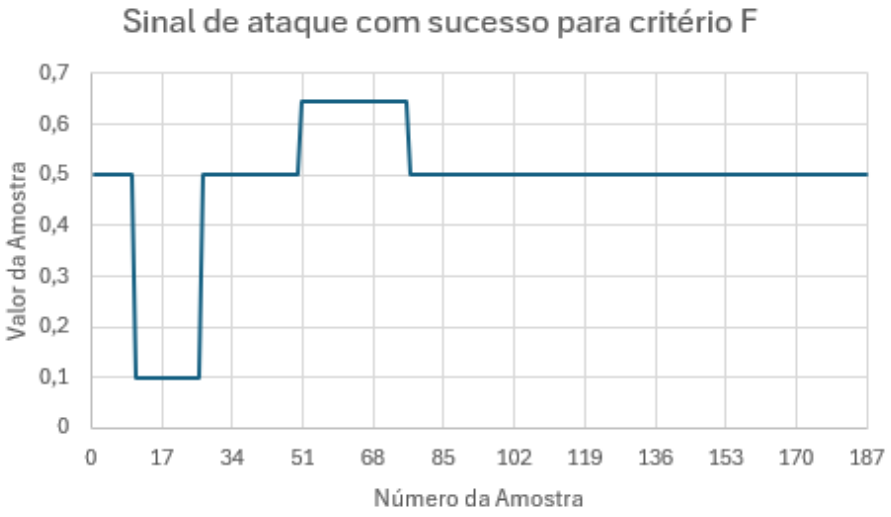

Figura 16 – Gráfico do Sinal Inicial de Ataque com Sucesso para Critério F

Após a obtenção de resultados de batimentos irrealistas, foi aplicada uma estratégia baseada restringir os limites de exploração do sinal de entrada para os limites da distribuição reconhecidos no trabalho de referência (SILVA NETO; CUGNASCA, 2023), que registra, para cada amostra de entrada, o valor médio e o desvio padrão do conjunto de dados. Utilizando os limites de exploração com valor médio mais ou menos o desvio padrão, há uma garantia de que os ataques ocorreram com características mais realista. Essas restrições podem ser aplicadas no arquivo de entrada VNNLIB, tal como demonstrado no exemplo da Figura 17. Nele, são especificadas restrições que são aplicadas para a averiguação da Condição “A” do critério “F” da Tabela 13 (seção 6.7) para forçar o algoritmo de ataques adversários a procurar valores de entrada que sejam mais compatíveis com a realidade.

```
(assert (<= X_45 0.4))
(assert (>= X_45 0.1))

(assert (<= X_46 0.4))
(assert (>= X_46 0.1))

(assert (<= X_47 0.4))
(assert (>= X_47 0.1))

(assert (<= X_48 0.4))
(assert (>= X_48 0.1))

(assert (<= X_49 0.4))
(assert (>= X_49 0.1))

(assert (<= X_50 0.4))
(assert (>= X_50 0.1))

(assert (<= X_150 0.1)
(assert (>= X_150 0))

(assert (<= X_151 0.1)
(assert (>= X_151 0))

(assert (<= X_152 0.1)
(assert (>= X_152 0))
```

Figura 17 – Exemplos de Restrições Aplicadas às Entradas para a Condição “A” do Critério "F"

O resultado obtido processando as entradas da Figura 17 é apresentado na Figura 18. O batimento cardíaco obtido representa uma melhoria significativa em comparação com a Figura 16, que possui a maior parte das entradas fixadas no valor 0,5, pois se assemelha à forma de onda de um batimento cardíaco real. Os arquivos com os gráficos dos sinais de ataque resultantes dessa atividade estão depositados no repositório do trabalho (STEPHANO SANTOS; MURAD, 2024) na pasta “*attacks_results*”.

Figura 18 – Gráfico do Sinal de Ataque com Sucesso para Critério F após Restrições do Domínio das Amostras a Limites de Valor Médio e Desvio Padrão

Para todos os ataques que foram executados, foi encontrado um sinal que exibia uma confusão com sucesso. Esses sinais apresentam um começo no pico Q e têm a presença da onda P, demonstrando coerência com a realidade. Embora essa análise não seja suficiente para garantir a existência ou não de problemas de segurança, a sensibilidade das classificações da rede para pequenas perturbações já indica preliminarmente que existem possíveis pontos de falha críticos em relação à segurança.

7.2 Considerações sobre o Cumprimento dos Requisitos do Projeto

Devido aos problemas técnicos relatados previamente na seção 6.6, não foi possível cumprir todos os requisitos estipulados para o projeto; todavia, com a estratégia de ataques adversários, avalia-se que alguns requisitos – e, portanto, os objetivos do projeto, foram atingidos parcialmente. Na Tabela 14 são detalhados os graus de cumprimento de cada requisito, junto com as respectivas justificativas que atestam se e como cada requisito é cumprido.

Tabela 14 – Análise de Cumprimento dos Requisitos do Projeto

Código do Requisito	Grau de Cumprimento	Justificativa
#00	Nenhum	Não foi possível cumprir devido aos problemas técnicos da ferramenta *alpha-beta-crown*. A análise formal de segurança será explorada em trabalhos futuros.
#01	Parcial	A sobreaproximação não pôde ser realizada, porém a estratégia de ataques adversários substituiu parcialmente seu papel neste trabalho. Além disso, nem todos os critérios da Tabela 13 (seção 6.7) foram aplicados, reforçando o cumprimento apenas parcial deste requisito.
#02	Parcial	Foi verificado que os batimentos cardíacos de saída eram condizentes com a realidade, mas como a sobreaproximação em si não foi implementada, o cumprimento deste requisito é tido como parcial.
#03	Total	A tradução para ONNX foi totalmente implementada.
#04	Total	A tradução para ONNX foi validada utilizando a estratégia de verificação tripla explicada na seção 6.3.
#05	Parcial	Como a sobreaproximação não foi executada, não é possível aferir o cumprimento global do requisito. Com a estratégia alternativa de ataques adversários, verificou-se que eles são executados em ordem de grandeza de minutos nos servidores do C2D e utilizando, nestes, duas placas de vídeo nVIDIA GTX 1080 Ti. A parte que demora mais tempo para ser executada é o treinamento da rede neural. Esse treinamento, porém, não é considerado integrante do requisito porque precede a construção do modelo de análise de segurança.

8. CONCLUSÃO

Neste trabalho foram apresentados: (i.) a arquitetura de uma solução para verificação formal em sistema de redes neurais que classifica arritmias cardíacas; (ii.) a implementação da tradução dos sistemas de redes, com a verificação de equivalência do sistema original; (iii.) uma primeira estratégia de verificação, baseada em ataques adversários; (iv.) as limitações dessa estratégia e (v.) a motivação para a necessidade de uma verificação formal de segurança mais ampla.

Apesar das limitações já comentadas sobre os cenários inseguros considerados e a fidedignidade dos resultados dos ataques adversários a batimentos cardíacos reais, considera-se que o desenvolvimento realizado pavimenta o caminho para a expansão do trabalho em atividades futuras. Foi possível identificar que, mesmo com pequenas perturbações, é possível provocar erros inseguros na classificação do sistema de detecção de arritmias cardíacas baseado em redes neurais, justificando uma análise de segurança mais extensiva.

Para trabalhos futuros, consideram-se necessários ajustes na ferramenta de verificação formal *alpha-beta-crown* para permitir a análise formal completa da rede neural, uma vez que foram identificadas limitações impeditivas para essa avaliação durante o presente trabalho. Além disso, a estratégia de ataques adversários adotada até o momento também possui limitações que restringem a abrangência da análise formal de segurança – sobretudo devido à geração de cenários de entradas impróprias que não caracterizam situações plausíveis para batimentos cardíacos de seres humanos vivos.

Ademais, o escopo de trabalhos futuros também compreende cenários adicionais de batimentos cardíacos e uma expansão também para o domínio das diferentes classificações de arritmias, permitindo que se avance além da diferenciação destas e dos batimentos cardíacos saudáveis. Outro tema a ser explorado futuramente é a comparação dos resultados obtidos da análise formal de segurança das redes neurais com as determinações existentes em normas de segurança e equipamentos de saúde comerciais. Todos os temas para trabalhos futuros pertencem ao escopo de pesquisa de Mestrado a ser realizada pelo próprio autor.

REFERÊNCIAS

ANALOG DIALOGUE. **ECG Front-End Design is Simplified with MicroConverter®**. Disponível em: <https://www.analog.com/en/resources/analog-dialogue/articles/ecg-front-end-design-simplified.html>. Acesso em: 8 dez. 2024.

BAK, S. nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement. *In:* NASA Formal Methods (NFM). **Procedings** [...]..Virtual, 2021. DOI: 10.1007/978-3-030-76384-8_2.

BENALI, R.; DIB, N.; BEREKSI, F. R. Cardiac Arrhythmia Diagnosis Using a Neuro-Fuzzy Approach. **Journal of Mechanics in Medicine and Biology**, v. 10, n. 3, p. 417 – 429, 2010. DOI: 10.1142/S021951941000354X.

BRIX, C. et al. First Three Years of the International Verification of Neural Networks Competition (VNN-COMP). **International Journal on Software Tools for Technology Transfer**, v. 25, p. 329–339, 2023. DOI: 10.1007/s10009-023-00703-4.

CAISAR. **CAISAR**. Disponível em: <https://www.caisar-platform.com/>. Acesso em: 8 dez. 2024.

CARLINI, N.; WAGNER, D. Towards Evaluating the Robustness of Neural Networks. *In:* IEEE Symposium on Security and Privacy (SP). **Proceedings** [...]. San Jose, Califórnia, EUA: 2017. DOI: 10.1109/SP.2017.49.

CARRAULT, G. et al. Temporal Abstraction and Inductive Logic Programming for Arrhythmia Recognition from Electrocardiograms. **Artificial Intelligence in Medicine**, v. 28, n. 3, p. 231–263, 2003. DOI: 10.1016/S0933-3657(03)00066-6.

CHAO, L.; LI, Z.; ZHANG, H. Research on Arrhythmia Classification Based on Domain Adaptation. *In:* 9th International Conference on Control, Automation and Robotics (ICCAR). **Proceedings** [...]. Pequim, China: 2023. DOI: 10.1109/ICCAR57134.2023.10151759.

DEMARCHI, S. et al. **GitHub - NeVer2: Learning and Verification of Neural Networks**. Disponível em: <https://github.com/NeVerTools/NeVer2>. Acesso em: 8 dez. 2024.

DONG, X.; SI, W. Heartbeat Dynamics: A Novel Efficient Interpretable Feature for Arrhythmias Classification. **IEEE Access**, v. 11, p. 87071–87086, 2023. DOI: 10.1109/ACCESS.2023.3305473.

FAHMY, H.; PASTORE, F.; BRIAND, L. HUDD: A Tool to Debug DNNs for Safety Analysis. *In:* IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). **Proceedings** [...]. Pittsburgh, Pensilvânia, EUA: 2022. DOI: 10.1109/ICSE-Companion55297.2022.9793750.

FAN, J. et al. ReachNN*: A Tool for Reachability Analysis of Neural-Network Controlled Systems. *In:* Automated Technology for Verification and Analysis (ATVA). **Proceedings** [...]. Hanoi, Vietnã, 2020. DOI: 10.1007/978-3-030-59152-6_30.

GUIDOTTI, D. et al. **The VNNLIB Standard for Benchmarks**. Sássari e Gênova, Itália, 2022. Disponível em: <https://www.vnnlib.org/assets/doc/standard.pdf>.

HUANG, C. et al. **GitHub - POLAR Official Tool**. Disponível em: <https://github.com/ChaoHuang2018/POLAR_Tool>. Acesso em: 8 dez. 2024.

IVANOV, R. et al. Verisig: Verifying Safety Properties of Hybrid Systems with Neural

Network Controllers. *In:* 22nd ACM International Conference on Hybrid Systems: Computation and Control, (HSCC). **Proceedings** [...]. Montreal, Quebec, Canadá, 2019. DOI: 10.1145/3302504.3311806.

JO, Y.-Y. et al. Detection and Classification of Arrhythmia Using an Explainable Deep Learning Model. **Journal of Electrocardiology**, v. 67, n. July–August 2021, p. 124 – 132, 2021. DOI: 10.1016/j.jelectrocard.2021.06.006.

KACHUEE, M.; FAZELI, S.; SARRAFZADEH, M. ECG Heartbeat Classification: A Deep Transferable Representation. *In:* IEEE International Conference on Healthcare Informatics (ICHI). **Proceedings** [...]. Nova Iorque, NY, EUA, 2018. DOI: 10.1109/ICHI.2018.00092.

KERAS TEAM. **Keras: Deep Learning for Humans**. Disponível em: <https://keras.io/>. Acesso em: 8 dez. 2024.

KHEDR, H.; FERLEZ, J.; SHOUKRY, Y. PEREGRiNN: Penalized-Relaxation Greedy Neural Network Verifier. *In:* 33rd International Conference, (CAV). **Proceedings** [...]. Virtual, 2021. DOI: 10.1007/978-3-030-81685-8_13.

KOZAL, J.; KSIENIEWICZ, P. Imbalance Reduction Techniques Applied to ECG Classification Problem. *In:* Intelligent Data Engineering and Automated Learning (IDEAL). **Proceedings** [...]. Manchester, Reino Unido, 2019. DOI: 10.1007/978-3-030-33617-2_33.

MADRY, A. et al. Towards Deep Learning Models Resistant to Adversarial Attacks. *In:* 6th International Conference on Learning Representations, (ICLR). **Proceedings** [...]. Vancouver, Colúmbia Britânica, Canadá: 2018. Disponível em: <https://openreview.net/pdf?id=rJzIBfZAb>. Acesso em: 8 dez. 2024.

MICROSOFT CORPORATION. **Visual Studio Code - Code Editing. Redefined**. Disponível em: <https://code.visualstudio.com/>. Acesso em: 8 dez. 2024.

MÜLLER, M. N. et al. **GitHub - ETH Robustness Analyzer for Deep Neural Networks**. Disponível em: <https://github.com/eth-sri/eran>. Acesso em: 8 dez. 2024.

OBERG, R.; PROBASCO, L.; ERICSSON, M. **Applying Requirements Management with Use Cases.Rational Software Corporation**, v.5, p. 1-24, 2000.

PAULSEN, B. et al. NeuroDiff: Scalable Differential Verification of Neural Networks using Fine-Grained Approximation. *In:* 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). **Proceedings** [...]. Virtual: 2020. DOI: 10.1145/3324884.3416560.

POBITZER, M.; FRISCHKNECHT, N. **GitHub - DeepPoly**. Disponível em: <https://github.com/Markus-Pobitzer/RTAI>. Acesso em: 8 dez. 2024.

PYTHON SOFTWARE FOUNDATION. **Python Release Python 3.6.12**. Disponível em: <https://www.python.org/downloads/release/python-3612/>. Acesso em: 8 dez. 2024.

RIEG, T. et al. Demonstration of the Potential of Whitebox Machine Learning Approaches to Gain Insights from Cardiovascular Disease Electrocardiograms. **PLoS ONE**, v. 15, n. 12, p. e0243615 (1-20), 2020. DOI: 10.1371/journal.pone.0243615.

ROEDER, L. **GitHub - Netron**. Disponível em: <https://github.com/lutzroeder/Netron>. Acesso em: 8 dez. 2024.

SIDRANE, C. et al. Safety Verification of Neural Network Control Policies for

Nonlinear Systems OVERT: An Algorithm for Safety Verification of Neural Network Control Policies for Nonlinear Systems. **Journal of Machine Learning Research**, v. 23, n. 117, p. 1-45, 2021. Disponível em: https://www.jmlr.org/papers/volume23/21-0847/21-0847.pdf. Acesso em: 8 dez. 2024.

SILVA NETO, A. V. **Safety ArtISt: Um Método para a Garantia de Segurança Crítica de Sistemas com Inteligência Artificial**. 2024. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo. DOI: 10.11606/T.3.2024.tde-14112024-090238.

SILVA NETO, A. V. et al. Safety Assurance of Artificial Intelligence-Based Systems: A Systematic Literature Review on the State of the Art and Guidelines for Future Work. **IEEE Access**, v. 10, p. 130733–130770, 2022. DOI: 10.1109/ACCESS.2022.3229233.

SILVA NETO, A. V.; CUGNASCA, P. S. **Relatório Técnico de Pesquisa - Desenvolvimento do Estudo de Caso de Sistema de Detecção de Arritmias Cardíacas por Aprendizado Supervisionado Profundo - Versão 3**. São Paulo (SP), p. 1-479, 2023. DOI: 10.5281/zenodo.7485108.

SONG, X. et al. QNNVerifier: A Tool for Verifying Neural Networks using SMT-Based Model Checking. **arXiv [*pre-print*]**, 2022, p.1-4. DOI: 10.48550/arXiv.2111.13110.

STEPHANO SANTOS, G.; MURAD, A. V. A. **GitHub - NN-ECG-classification-Verifier**. Disponível em: <https://github.com/GaStephano-USP/NN-ECG-classification-Verifier>. Acesso em: 8 dez. 2024.

STEPHANO SANTOS, G.; SILVA NETO, A. V.; CUGNASCA, P. S. **Relatório Técnico de Pesquisa - Análise Comparativa de Ferramentas Para Análise Formal em Redes Neurais**. São Paulo (SP), p. 1-15. 2024. DOI: 10.5281/zenodo.13687409.

THALER, M. S.; BURNIER, J. N. T. **ECG Essencial: Eletrocardiograma na Prática Diária - 7a edição**. 10. ed. Porto Alegre, RS, Brasil: Artmed, 2023.

THE LINUX FOUNDATION. **GitHub - onnx/onnx: Open Standard for Machine Learning Interoperability**. Disponível em: <https://github.com/onnx/onnx>. Acesso em: 8 dez. 2024a.

THE LINUX FOUNDATION. **GitHub - tf2ONNX**. Disponível em: <https://github.com/onnx/tensorflow-onnx>. Acesso em: 8 dez. 2024b.

THE SMT-LIB INITIATIVE. **SMT-LIB The Satisfiability Modulo Theories Library**. Disponível em: <https://smtlib.cs.uiowa.edu/>. Acesso em: 8 dez. 2024.

VERIFIED INTELLIGENCE. **GitHub - alpha-beta-CROWN: An Efficient, Scalable and GPU Accelerated Neural Network Verifier**. Disponível em: <https://github.com/Verified-Intelligence/alpha-beta-CROWN>. Acesso em: 8 dez. 2024.

VERIVITAL. **GitHub - Neural Network Verification Software Tool**. Disponível em: <https://github.com/verivital/nnv>. Acesso em: 8 dez. 2024.

WANG, J.-S. et al. An Effective ECG Arrhythmia Classification Algorithm. *In:* 7th International Conference on Intelligent Computing (ICIC). **Proceedings** [...]. Zhengzhou, China: 2011. DOI: 10.1007/978-3-642-24553-4_72.

XU, K. et al. Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers. *In:* 9th International Conference

on Learning Representations, (ICLR). **Proceedings** [...].Virtual: 2021. Disponível em: <https://openreview.net/pdf?id=nVZtXBI6LNn>. Acesso em: 8 dez. 2024

XUE, Z. et al. A Tale of Two Approximations: Tightening Over-Approximation for DNN Robustness Verification via Under-Approximation. *In:* 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA). **Proceedings** [...]. Seattle, WA, USA: Association for Computing Machinery, Inc, 12 jul. 2023. DOI: 10.1145/3597926.3598127.

APÊNDICE

APÊNDICE A – E-Mails Trocados com a Equipe da Ferramenta *Alpha-Beta-Crown*

O presente apêndice tem o objetivo de documentar os e-mails trocados com a equipe responsável pela ferramenta *alpha-beta-crown* a respeito das limitações e dos problemas identificados durante a realização deste trabalho.

Inquiry on Alpha-Beta-CROWN Capabilities for Residual Networks and VNN-LIB Compatibility

Ana Vitoria Abreu Murad <anavitoriamurad@usp.br> 11 de novembro de 2024 às 08:10

Para: huan@huan-zhang.com, zshi@cs.ucla.edu, brix@cs.rwth-aachen.de, kx46@drexel.edu, xiangruzh0915@gmail.com, qiruijin@umich.edu, haoc8@illinois.edu, hx84@duke.edu

Cc: Antonio Vieira da Silva Neto <antonio.vieira88@usp.br>, Gabriel Stephano Santos <gabrielstephano@usp.br>, Paulo Sergio Cugnasca <cugnasca@usp.br> Dear Alpha-Beta-CROWN Development Team,

My name is Ana Vitória Abreu Murad, and I am a Computer Engineering undergraduate student at the University of São Paulo. I am reaching out for assistance
with some challenges we have encountered while using Alpha-Beta-CROWN as a tool within our broader research project on the formal safety verification of neural network-based heart arrhythmia detection systems. Our project is led by Gabriel Stephano, under the guidance of Professors Paulo Cugnasca and Antonio Vieira. While our primary focus is on the safety verification of our own neural networks, we have been using Alpha-Beta-CROWN as a verification tool to explore potential limitations and verify model robustness. We would appreciate your insights on technical issues, and for additional clarity, we have included summarized logs relevant to each issue as attachments.

The main areas of inquiry are as follows:

Compatibility of Alpha-Beta-CROWN with Residual Networks: We are applying Alpha-Beta-CROWN to ResNet models, and during this process, we encountered errors related to a layer type called BoundMul. Upon further investigation, it appears that VNN-LIB files might not support this specific arithmetic operation. We have attached the logs (smote_BoundMul.log and baseline_BoundMul.log) to illustrate these errors. Could you provide any guidance on whether VNN-LIB is compatible with different ResNet variations or if there are recommended workarounds for handling arithmetic operations like BoundMul within the current verification framework?

PGD Attack Behavior: When executing a PGD attack as part of the verification, it fails instantly, raising questions about the interpretation of this result. We have reviewed the counterexamples, which appear to match our specified input and output constraints. The attached logs (smote_attack.log and baseline_attack.log) provide further detail. Could we be misinterpreting the verification outcome? Specifically, does an “unsafe” result solely indicate a mismatch with specifications in the VNN-LIB file, or are there other properties being verified alongside matching these specifications?

Support for Mathematical Operators in VNN-LIB: Our verification scenario requires support for mathematical operations such as addition (e.g., + Y_0 Y_1). We would like to confirm if VNN-LIB files support arithmetic expressions or if they are limited to logical operators. This functionality would be crucial to accurately reflect the mathematical relations within our ResNet models.

Attachments:
smote_attack.log and baseline_attack.log – Summarized logs from the PGD attack experiments for the SMOTE and baseline models, respectively. These logs highlight the instant failures we encountered during verification.
smote_BoundMul.log and baseline_BoundMul.log – Summarized logs showing the BoundMul error encountered when running Alpha-Beta-CROWN on the SMOTE and baseline models.
basic_prop.vnnlib – The VNN-LIB file specifying the input-output constraints we applied in our verification.
smote_basic_prop.counterexample – A sample counterexample generated during the SMOTE experiment, matching the input and output specifications provided in basic_prop.vnnlib.

This should give your team a clear view of the errors and outputs we're seeing. Let me know if the complete logs would be helpful for a deeper review.

Thank you for your time and assistance. Any information or suggestions you could provide would be greatly helpful as we work to optimize the use of Alpha- Beta-CROWN for our research objectives.

Best regards,
Ana Vitória Abreu Murad
Computer Engineering Undergraduate, University of São Paulo

Huan Zhang <huan@huan-zhang.com> 11 de novembro de 2024 às 10:00
Para: Ana Vitoria Abreu Murad <anavitoriamurad@usp.br>
Cc: zshi@cs.ucla.edu, brix@cs.rwth-aachen.de, kx46@drexel.edu, xiangruzh0915@gmail.com, qiruijin@umich.edu, haoc8@illinois.edu, hx84@duke.edu, Antonio Vieira da Silva Neto <antonio.vieira88@usp.br>, Gabriel Stephano Santos <gabrielstephano@usp.br>, Paulo Sergio Cugnasca <cugnasca@usp.br>

Hi Ana,

Thank you for reaching out and I greatly appreciate your interests in alpha-beta-CROWN!

At high level, we support general computation graphs including different variations of ResNets (Zhouxing can take a detailed look at your log and see what is the problem). The VNNLIB specification is restricted to simple logic conditions, but you can encode additional computation into your network rather than in VNNLIB. The VNNLIB eventually checks for the output of your computation graph, and things like "addition" you mentioned can be appended at the last layer of your computation graph.

"Unsafe" means that there exists an input x that satisfies the condition defined in VNNLIB. VNNLIB defines counterexamples.

Zhouxing and Xiangru, I believe many of the questions are common questions.

After you respond to Ana could you also create a FAQ page that collects recent questions that may help other people? We can include this FAQ in our next release.

Thanks,
Huan

Gabriel Stephano Santos <gabrielstephano@usp.br> 11 de novembro de 2024 às 18:47
Para: Huan Zhang <huan@huan-zhang.com>
Cc: Ana Vitoria Abreu Murad <anavitoriamurad@usp.br>, zshi@cs.ucla.edu, brix@cs.rwth-aachen.de, kx46@drexel.edu, xiangruzh0915@gmail.com, qiruijin@umich.edu, haoc8@illinois.edu, hx84@duke.edu, Antonio Vieira da Silva Neto <antonio.vieira88@usp.br>, Paulo Sergio Cugnasca <cugnasca@usp.br>

Hello, Huan,

I hope this message finds you well.

My name is Gabriel Stephano, and I am a Computer Engineering student at the University of São Paulo entering the Master's program and Ana's advisor on this project. I would like to thank you for your prompt response and for your kindness. The tool developed by your team has been immensely valuable for our research, and its availability as an open-source project is a significant contribution to the research area as a whole.

We await further information regarding the BoundMul issue. Also, I would like to offer my assistance with drafting user-oriented questions for the FAQ page, should you find this perspective valuable.
Thank you for your attention.

Best regards,
Gabriel Stephano

Zhouxing Shi <zhouxingshichn@gmail.com> 14 de novembro de 2024 às 01:49
Para: Gabriel Stephano Santos <gabrielstephano@usp.br>
Cc: Huan Zhang <huan@huan-zhang.com>, Ana Vitoria Abreu Murad <anavitoriamurad@usp.br>, brix@cs.rwth-aachen.de, kx46@drexel.edu, xiangruzh0915@gmail.com, qiruijin@umich.edu, haoc8@illinois.edu, hx84@duke.edu, Antonio Vieira da Silva Neto <antonio.vieira88@usp.br>, Paulo Sergio Cugnasca <cugnasca@usp.br>

Hi Ana and Gabriel,

Great to hear that you are trying to apply alpha-beta-CROWN for medical systems. In response to your inquiries:

Compatibility of Alpha-Beta-CROWN with Residual Networks and Support for Mathematical Operators in VNN-LIB:
Huan has already explained that complicated specifications can be encoded in the computational graph even though VNNLIB only supports simple specifications.
For the issue in smote_BoundMul.log, there might be some bug in the verifier or some inconsistency in VNNLIB/model/config. Would you be able to share the model you were using?
For the issue in baseline_BoundMul.log, it looks like we require stride==kernel_size for convolutional layers at this time. Maybe you

could consider modifying the convolutional layers in your model.

PGD Attack Behavior:
If it reports "unsafe", it means that some counterexample exists in your model. If you want to eliminate all the "unsafe" cases, you would need to train or repair your model to remove those counterexamples. The result can be specific to the model and there may not be an issue in the VNNLIB file.
To eliminate counterexamples, you may consider using a counterexample-guided training process (e.g., see https://arxiv.org/abs/2404.07956). I have also been working on a new training method and may share it with you a little later after my revision.

@Huan Zhang Thanks for the suggestion and it sounds helpful to create a FAQ.
@Xiangru Would you like to initiate one?
And thanks to @Gabriel for offering to help on the FAQ from users' perspective. Feel free to suggest more questions for the FAQ.

Best,
Zhouxing

Xiangru Zhong <xiangruzh0915@gmail.com> 14 de novembro de 2024 às 02:00
Para: Zhouxing Shi <zhouxingshichn@gmail.com>
Cc: Gabriel Stephano Santos <gabrielstephano@usp.br>, Huan Zhang <huan@huan-zhang.com>, Ana Vitoria Abreu Murad <anavitoriamurad@usp.br>, brix@cs.rwth-aachen.de, kx46@drexel.edu, qiruijin@umich.edu, haoc8@illinois.edu, hx84@duke.edu, Antonio Vieira da Silva Neto <antonio.vieira88@usp.br>, Paulo Sergio Cugnasca <cugnasca@usp.br>

Hi Zhouxing,
Sure! I'll start writing a FAQ, collecting these recently asked questions.
Best regards,
Xiangru Zhong

Ana Vitoria Abreu Murad <anavitoriamurad@usp.br> 15 de novembro de 2024 às 15:11
Para: Xiangru Zhong <xiangruzh0915@gmail.com>
Cc: Zhouxing Shi <zhouxingshichn@gmail.com>, Gabriel Stephano Santos <gabrielstephano@usp.br>, Huan Zhang <huan@huan-zhang.com>, brix@cs.rwth-aachen.de, kx46@drexel.edu, qiruijin@umich.edu, haoc8@illinois.edu, hx84@duke.edu, Antonio Vieira da Silva Neto <antonio.vieira88@usp.br>, Paulo Sergio Cugnasca <cugnasca@usp.br>

Hi Huan and Zhouxing,

Thank you for your detailed response. We appreciate the insights and suggestions regarding the compatibility of Alpha-Beta-CROWN with residual networks and the PGD attack behavior.As requested, I am sending the model we are currently using for your review. In addition, I'm including the configuration files and the VNNLIB file that passed the PGD attack but encountered issues afterwards. I hope these additional files will help with your investigation into the issues we've faced.

We'll also investigate the stride==kernel_size requirement for the convolutional layers and explore counterexample-guided training to address the "unsafe" cases.
Thank you again for your support, and we look forward to your feedback and any additional recommendations.

Best regards,

Ana