UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE LUIS BATISTA DA SILVA

PROPOSTA DE SISTEMA ASSISTENTE DE DIAGNÓSTICO DERMATOLÓGICO

RIO DE JANEIRO
2020

ALEXANDRE LUIS BATISTA DA SILVA

PROPOSTA DE SISTEMA ASSISTENTE DE DIAGNÓSTICO DERMATOLÓGICO

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Daniel Sadoc Menasché

RIO DE JANEIRO

2020

CIP - Catalogação na Publicação

SS586p

Silva, Alexandre Luis Batista da
Proposta de Sistema Assistente de Diagnóstico Dermatológico / Alexandre Luis Batista da Silva. -- Rio de Janeiro, 2020.
54 f.

Orientador: Daniel Sadoc Menashé.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2020.

1. Arquitetura de sistemas. 2. Sistemas médicos. 3. Aprendizagem de máquina. 4. Dermatologia. I. Menashé, Daniel Sadoc, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

ALEXANDRE LUIS BATISTA DA SILVA

PROPOSTA DE SISTEMA ASSISTENTE DE DIAGNÓSTICO DERMATOLÓGICO

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 07 de maio de 2020

BANCA EXAMINADORA:

Daniel Sadoc Menasché
Doutorado (University of Massachusetts at Amherst)

Josefino Cabral Melo Lima
Doutorado (Université Pierre et Marie Curie)

Geraldo Zimbrão da Silva
Doutorado (COPPE/UFRJ)

Sergio Assis Rodrigues
Doutorado (COPPE/UFRJ)

Bernard Kac
Graduação (UFRJ)

Dedico este trabalho à minha família, os principais responsáveis por eu ter chegado até aqui.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter permitido que tudo isso fosse possível. A minha família, por todo apoio e orientação que sempre me deram. Aos meus amigos e professores, por toda ajuda em minha trajetória acadêmica.

RESUMO

Com a popularização da inteligência artificial nos últimos anos, seu número de aplicações tem crescido muito. Diversas áreas têm se beneficiado deste processo, sendo que uma das mais eminentes é a medicina. Entre as diferentes especializações médicas, uma cujos sintomas são mais evidentes é a dermatologia. Por este motivo, ela foi escolhida para este trabalho.

Este trabalho apresenta uma arquitetura de sistema assistente de diagnóstico dermatológico. A proposta consiste em que no momento do atendimento de um paciente, o médico deve coletar algumas informações sobre ele, uma foto de sua lesão dermatológica e fornecer ambos ao sistema, que os processa com modelos de árvore de decisão e rede neural convolucional para efetuar uma predição dos possíveis diagnósticos. Após o real diagnóstico ser obtido, o médico o retorna ao sistema para que seus parâmetros possam ser atualizados com esta nova amostra.

No protótipo desenvolvido, o objetivo foi obter uma estrutura funcional, que ilustrasse a motivação deste trabalho, com gerenciamento dos dados, fluxo de informação e execução dos algoritmos de aprendizagem de máquina. Porém, os modelos não foram treinados em uma base de dados dermatológica.

Palavras-chave: desenvolvimento de sistemas. arquitetura de sistemas. sistemas inteligentes. sistemas médicos. inteligência artificial. aprendizagem de máquina. medicina. dermatologia.

ABSTRACT

With the popularization of artificial intelligence in recent years, its number of applications has grown a lot. Several areas have benefited from this process, one of the most eminent being medicine. Among the different medical specialties, one whose symptoms are most evident is dermatology. For this reason, it was chosen for this work.
This work presents an architecture of a diagnosis assistant system. The proposal is that when attending a patient, the doctor should collect some information about him, a photo of his dermatological lesion and provide both to the system, which processes them with decision tree and convolutional neural network models to perform a prediction of possible diagnoses. After the real diagnosis is obtained, the doctor returns it to the system so that its parameters can be updated with this new sample.
In the developed prototype, the objective was to obtain a functional structure, which would illustrate the motivation of this work, with data management, information flow and execution of machine learning algorithms. However, the models were not trained in a dermatological database.

Keywords: systems development. systems architecture. intelligent systems. medical systems. artificial intelligence. machine learning. medicine. dermatology.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO E PROBLEMA

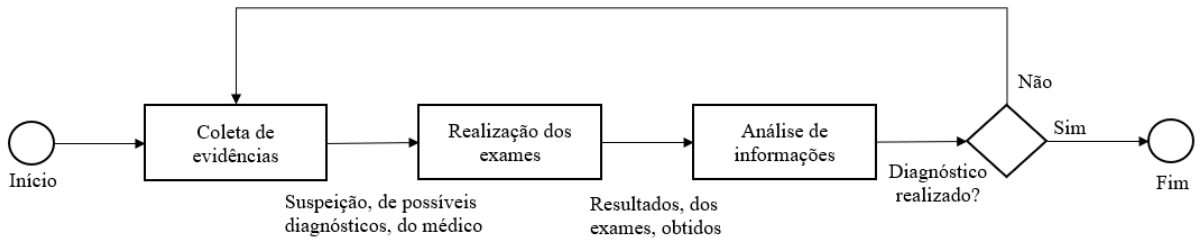
O processo de diagnóstico de diversas doenças de diferentes especializações médicas pode envolver um conjunto de etapas que possuem certa complexidade e que podem levar um tempo considerável para serem concluídas:

- Inicialmente, o médico precisa coletar evidências sobre o paciente, qualquer informação que possa ajudá-lo no atendimento. Isso é feito com objetivo de obter orientação para um certo conjunto de possíveis causas dos sintomas do paciente.
- O médico requisita um certo número de exames, baseado nas informações coletadas na etapa anterior, visando obter alguma comprovação de qual das possíveis causas é a verdadeira.
- Após obter os resultados dos exames, o médico os compara com as informações coletadas e efetua uma análise para verificar se possui o suficiente para fornecer um diagnóstico.

Caso o diagnóstico não tenha sido obtido, os médicos irão repetir estas etapas buscando novas informações, sintomas, realizando novos exames e executando novas análises, com a expectativa de que os novos dados apresentem alguma resposta sobre o diagnóstico do paciente. Este ciclo irá se repetir até que o diagnóstico seja obtido.

Abaixo, temos um diagrama básico que apresenta o fluxo de diagnóstico:

Figura 1 – Fluxo de diagnóstico tradicional



Fonte: O autor (2020)

Para se diagnosticar uma doença, o médico responsável precisa realizar mentalmente um mapeamento entre as variáveis de causas, sintomas e seus respectivos diagnósticos. Esta tarefa se torna complexa devido ao número de valores que cada uma dessas variáveis pode assumir, e consequentemente, o número de mapeamentos diferentes que surgem a

partir disto. E existe a probabilidade de um erro neste processo ocorrer e prejudicar o paciente.

Um aspecto que precisa ser considerado é que algumas dessas doenças exigem uma certa agilidade em sua detecção, para que seu tratamento tenha maior possibilidade de ser bem-sucedido, trazendo o mínimo de prejuízos ao paciente. Desta forma, tempo e precisão se tornam recursos preciosos no processo de diagnóstico e se faz necessário que ele ocorra de modo eficiente e otimizado.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

A tecnologia já trabalha aliada à medicina há décadas. Em especial, os avanços recentes no campo da inteligência artificial ocasionaram uma vasta expansão no seu conjunto de aplicações, nas mais diversas áreas. As consequências se apresentam de variadas formas, como automatizações de tarefas que até então eram inviáveis e ferramentas que oferecem suporte à tomada de decisão.

Neste contexto, o raciocínio natural que surge é a utilização dos benefícios trazidos pela inteligência artificial na área médica. Porém, esta ainda é uma área nova que ainda necessita de muito desenvolvimento para alcançar certa maturidade. Além disso, as diversas aplicações e pesquisas que se encaixam nesta descrição não são muito popularizadas.

Uma aplicação, com o objetivo de auxiliar o processo de diagnóstico minimizando a probabilidade de erro, que realizasse este feito traria diversas vantagens, como: a aceleração na descoberta do diagnóstico de cada paciente, a redução dos prejuízos causados por doenças que necessitam de tratamento rapidamente e o aumento no número de pacientes tratados.

A vertente de especialização médica que foi escolhida como base para este trabalho foi a dermatologia, pelo fato de as evidências e sintomas serem mais facilmente coletados.

Este trabalho se dedica a apresentar uma proposta e protótipo de sistema que atua utilizando inteligência artificial como um assistente de diagnóstico da área de dermatologia.

1.2.2 Objetivos Específicos

Para que o objetivo geral deste trabalho seja concluído, existe uma série de objetivos específicos como requisitos:

- Proposição da arquitetura do sistema.
- Desenvolvimento do protótipo do assistente de diagnóstico:
  - Desenvolvimento do *frontend*.

- Desenvolvimento do *backend*.
- Efetuar comunicação entre *frontend* e *backend*.
- Configuração do banco de dados.
- Execução dos algoritmos de inteligência artificial.

1.3 SISTEMA

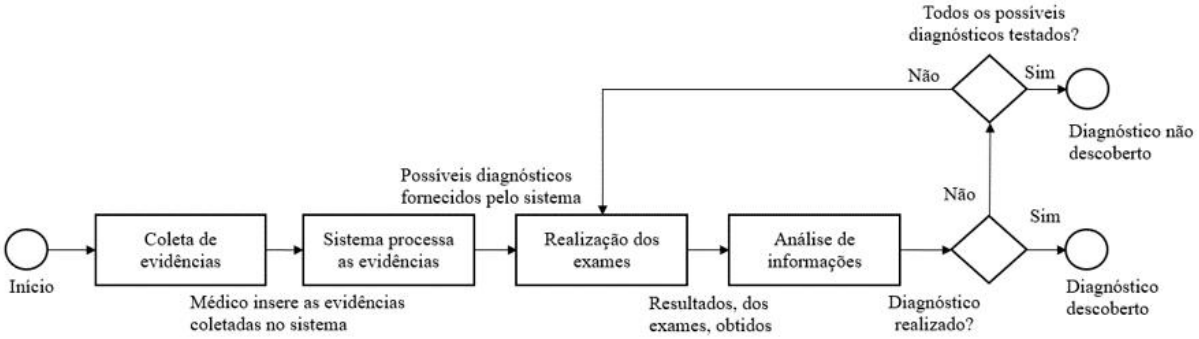
O sistema assistente de diagnóstico arquitetado neste trabalho funciona recebendo as informações do paciente, coletadas pelo médico. Após recebê-las, o sistema executa um processamento de inteligência artificial, e no fim, ele retorna uma lista com uma série de diagnósticos e uma probabilidade associada a cada um deles, que corresponde a chance de ser o verdadeiro diagnóstico do paciente.

A tarefa do mapeamento entre as informações coletadas sobre o paciente e os possíveis diagnóstico é transferida do médico para o sistema. O médico passa a assumir um papel de mais alto nível, com a responsabilidade de gerenciar o processo de descoberta do diagnóstico.

A ideia é que o médico requisite os exames necessários para comprovar qual é o diagnóstico verdadeiro entre os prováveis, fornecidos pelo sistema. Caso não haja comprovação, o médico deve requisitar pela comprovação em exames nos menos prováveis. E se ainda assim, não houver comprovação, então se trata de um diagnóstico sem precedentes estatísticos no banco de dados do sistema.

Desta forma, o fluxo de descoberta de diagnóstico é alterado da seguinte maneira:

Figura 2 – Fluxo de diagnóstico com sistema



Fonte: O autor (2020)

1.3.1 Especificações

Um sistema que auxilia no processo de diagnóstico precisa ser intermediado por um médico para evitar que pacientes se autodiagnostiquem. Por este e outros motivos, se

faz necessário que o sistema requisite algum tipo de autenticação do usuário, como consequência disto, o sistema será fornecido aos usuários via *web* através de um servidor. Além disso, surge a necessidade de bancos de dados para manutenção dos usuários.

Como se trata de um sistema que utiliza algoritmos de inteligência artificial, que exige uma quantidade de processamento acima do comum, e que está em constante mudança, é preciso que essas aplicações sejam executadas completamente em algum serviço de nuvem. E novamente, surge a necessidade de bancos de dados para a manutenção dos dados nos quais os modelos de inteligência artificial se baseiam.

Com o objetivo de facilitar e flexibilizar o uso do sistema no momento do atendimento, não seria adequado que o sistema fosse acessado via PC, e sim via aplicativo *mobile*. Desta forma, o celular é suficiente para auxiliar o médico, eliminando a necessidade de um computador.

Assim, o sistema é acessado por aplicativo *mobile* e executado na nuvem, como suas aplicações de inteligência artificial e bancos de dados.

1.3.2 Inteligência

O processo inicial de coleta de evidências sobre o paciente consiste basicamente em fazer determinadas perguntas ao mesmo para que o médico possa obter algum indício dos possíveis diagnósticos. Porém, a eficiência deste processo depende da capacidade do médico conseguir fazer as perguntas certas e mapeá-las aos possíveis diagnósticos. Um recurso que pode ser utilizado neste caso é a modelagem deste problema como uma árvore de decisão.

O sistema usará um algoritmo de árvore de decisão para fornecer ao médico qual a pergunta que deve ser feita ao paciente, e após o médico retornar ao sistema a resposta do paciente, o sistema fornecerá ao médico a nova pergunta. Cada pergunta extra realizada ao paciente corresponde a um nível adicional numa árvore de decisão, na qual cada vértice corresponde a uma pergunta e cada aresta corresponde a uma possível resposta a tal pergunta. A pergunta inicial corresponde a raiz da árvore. As folhas da árvore correspondem a eventuais diagnósticos a serem tomados em função das respostas

Este trabalho é focado na especialização de dermatologia, pelo fato de que eventuais deformidades na pele são visivelmente identificadas. Para tirar vantagem desse aspecto visual, o uso de uma rede neural convolucional é aplicado, com o objetivo de detectar qual diagnóstico está relacionado com a imagem apresentada.

O sistema permitirá que o médico efetue *upload* de uma foto da lesão do paciente, mediante sua autorização. A rede neural convolucional irá processar a imagem e retornar o resultado.

1.3.3 Diagnóstico

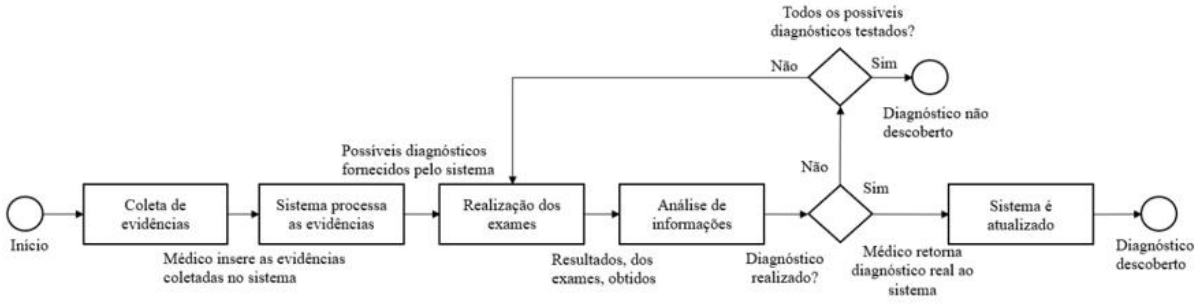
Ao final do processo de coleta de dados sobre o paciente, o sistema fornecerá uma lista com os diagnósticos mais prováveis, que será construída com base nos resultados da árvore de decisão e da rede neural convolucional. Depois, orientado pelos resultados do sistema, irá direcionar o paciente para a realização de exames específicos para verificar qual das opções recebidas representa o diagnóstico verdadeiro.

1.3.4 Aprendizagem

Após fornecer os resultados para um determinado paciente, o sistema os armazenará com o status pendente, isto é, não foi recebido a comprovação de que ele está correto ou não. Quando o médico obtiver o diagnóstico real, com base nos exames pedidos, ele deverá atualizar o seu respectivo status no sistema, informando se o resultado fornecido por ele está correto ou não. E de acordo com isto, o sistema atualizará as estruturas da árvore de decisão e da rede neural convolucional.

Abaixo, temos o fluxo de diagnóstico considerando a atualização do sistema:

Figura 3 – Fluxo de diagnóstico com sistema e atualização



Fonte: O autor (2020)

1.3.5 Desafios

Algoritmos de inteligência artificial necessitam de dados para que os parâmetros de seus modelos possam ser atualizados de forma a fornecer resultados razoáveis para o problema tratado.

No contexto médico, para um sistema que atue como assistente de diagnóstico, seria necessário que se desenvolvesse um projeto de coleta informações e sintomas, por parte dos médicos, de diversos pacientes e as mapeassem para os seus respectivos diagnósticos. O banco de dados resultante seria utilizado como treinamento dos algoritmos de inteligência artificial escolhidos.

Apesar da existência de diversos projetos que visam a coleta de dados médicos para o uso de inteligência artificial, ainda há muitas doenças e especializações com dados insuficientes para que sistemas inteligentes possam ser implementados.

1.4 ORGANIZAÇÃO DO TEXTO

Este trabalho está dividido em 4 capítulos.

O primeiro é responsável por realizar uma contextualização do processo de diagnóstico, apresentação do problema associado a ele, assim como uma breve descrição de um possível sistema que atuaria como solução.

O segundo tem como objetivo a explicação de alguns conceitos, relacionados a área de inteligência artificial, que foram utilizados neste trabalho e que precisam ser apresentados para uma compreensão mais profunda do mesmo.

O terceiro é focado na descrição do sistema de assistência de diagnóstico. Aspectos como tecnologias utilizadas, arquitetura e funcionamento são explicados neste capítulo.

O quarto apresenta as considerações finais do trabalho desenvolvido.

O quinto, e último, capítulo expõe possíveis ideias de aprimoramentos e desdobramentos para trabalhos futuros.

2 CONHECIMENTOS PRÉVIOS

O sistema proposto neste trabalho realiza um processo de classificação. Em outras palavras, baseado em um conjunto de dados com N amostras, com cada elemento i formado por um par (x_i, y_i), onde x_i é um vetor de atributos que caracterizam a amostra i e y_i é a classe de i, ela representa como a amostra é definida em certo aspecto. O objetivo é obter um mapeamento entre x e y, tal que, ao lidar com novas amostras, onde y é desconhecido, ele seja obtido com base em x (MURPHY, 2012).

No contexto dermatológico, o quadro 1 apresenta um exemplo hipotético de conjunto de dados textuais com diagnósticos de três pacientes. Nele, cada paciente possui três dados, que correspondem ao vetor de atributos x, e estão associados ao diagnóstico que foi realizado, que corresponde a y. O objetivo é obter um mapeamento entre os dados de cada paciente e seus respectivos diagnósticos, tal que, ao lidar com novos pacientes, onde o diagnóstico é desconhecido, ele seja obtido com base nos dados do paciente.

Quadro 1 – Exemplo de conjunto de dados de diagnósticos de pacientes

Paciente	Dados do paciente			Diagnóstico
	Lesão elevada?	Mancha de cor diferenciada?	Prurido?	
1	Sim	Sim	Não	Câncer de pele
2	Não	Sim	Não	Vitiligo
3	Não	Sim	Sim	Urticária

Fonte: O autor (2020)

O pares (x_i, y_i) que correspondem a cada paciente apresentado no quadro 1 são:

$$(x_1, y_1) = (['\text{Sim}', '\text{Sim}', '\text{Não}'], '\text{Câncer de Pele}')$$

$$(x_2, y_2) = (['\text{Não}', '\text{Sim}', '\text{Não}'], '\text{Vitiligo}')$$

$$(x_3, y_3) = (['\text{Não}', '\text{Sim}', '\text{Sim}'], '\text{Urticária}')$$

Os dados que auxiliam o médico no processo de diagnóstico dermatológico não se resumem apenas a dados textuais, mas também a imagens das lesões dermatológicas dos pacientes. Assim, é preciso que haja um conjunto de dados com imagens de lesões dermatológicas e seus respectivos diagnósticos associados.

O diagnóstico é fornecido com base em informações textuais do paciente e uma imagem de sua lesão dermatológica. Assim, dois mapeamentos devem ser obtidos referentes aos pares (dados textuais, diagnóstico) e (imagem, diagnóstico). Como são dois problemas diferentes, um associado aos dados textuais e outro associado às imagens, se faz necessário utilizar dois modelos diferentes de aprendizagem de máquina.

Figura 4 – Imagem de um diagnóstico de câncer de pele

Fonte: Sociedade Brasileira de Cirurgia Dermatológica (2020)

Figura 5 – Imagem de um diagnóstico de vitiligo

Fonte: Sociedade Brasileira de Dermatologia (2020)

Figura 6 – Imagem de um diagnóstico de urticária

Fonte: Sociedade Brasileira de Dermatologia (2020)

Árvores de decisão já vêm sendo aplicadas no problema de classificação de doenças em pacientes, pelo fato de suas características se adaptarem bem ao problema (MITCHELL, 1997). Assim, a modelagem de árvore de decisão foi escolhida para trabalhar com o

mapeamento dos sintomas.

Para trabalhar com as imagens, o modelo escolhido foi o de redes neurais convolucionais, por serem especializadas a lidar com dados distribuídos em forma de grade, como uma imagem (GOODFELLOW, 2015).

Existem alguns conceitos que foram utilizados no desenvolvimento da arquitetura do sistema proposto neste trabalho. Eles precisam ser explicados e entendidos para uma compreensão mais profunda do funcionamento do sistema aqui proposto. As ideias centrais que serão apresentadas são a de árvores de decisão e a de redes neurais artificiais. Esta seção se dedica a explicar seus conceitos.

2.1 ÁRVORES DE DECISÃO

2.1.1 Definição

No contexto operacional, árvores de decisão são um modelo hierárquico, representado por um grafo estruturado em árvore, de decisões e suas possíveis consequências. Elas são utilizadas como ferramenta, pelo responsável pelo processo de tomada de decisão, para auxiliar na identificação da melhor estratégia a ser tomada para alcançar um determinado objetivo (ROKACH; MAIMON, 2014).

No contexto computacional, mais especificamente em data mining, segundo Rokach e Maimon (2014, p. 10, tradução nossa):

> [...] a árvore de decisão é utilizada como modelo preditivo, que pode ser usada em tarefas de classificação e regressão [...] Árvores utilizadas em tarefas de classificação são chamadas de árvores de classificação, e as utilizadas em tarefas de regressão são chamadas de árvores de regressão.

Porém, este trabalho se concentra em árvores de classificação e se refere a elas somente como árvores de decisão.

Em uma árvore de decisão, cada nó interno, nó que não é folha, representa um atributo, cada aresta que o conecta aos filhos é um valor que ele pode assumir, e as folhas são os valores são as classes. Uma árvore de decisão é uma estrutura baseada em fluxo, que parte da raiz até uma folha, onde o caminho percorrido é uma consequência dos valores assumidos pelos atributos (MITCHELL, 1997). Deve-se identificar quais atributos estão relacionados com as classes, caso contrário, atributos irrelevantes podem ser escolhidos para o processo de classificação.

Uma base de dados é composta por um conjunto de objetos, onde cada um deles é descrito por uma coleção de atributos, que estão contidos em x. Cada atributo representa uma característica importante de um objeto. Além disso, cada objeto está associado a uma determinada classe C_i de um conjunto $\{C_1, C_2, \ldots, C_n\}$ associado à base de dados,

ou seja, y assume um valor de classe C_i. Ao interagir com novos objetos, os valores dos atributos são conhecidos, porém, a classe não (QUINLAN, 1992).

Dado um novo objeto, o objetivo é estimar a classe utilizando os valores dos atributos desta amostra. A base de dados é usada para construir uma relação, representada pela árvore de decisão, entre os valores dos atributos e as classes. Assim, a árvore de decisão é usada para a classificação de novas amostras de dados, orientando o processo de tomada de decisão.

Como a árvore de decisão é utilizada como modelo preditivo, é necessário que ela seja construída através de um processo de aprendizado. De acordo com Mitchell (1997, p. 52, tradução nossa): "O aprendizado de árvore de decisão é um método de aproximação de uma função alvo discreta na qual a função aprendida é representada pela árvore de decisão".

Como o processo de aprendizado é fundamentado em dados (RUSSELL; NORVIG, 2010), consequentemente, uma árvore de decisão precisa ser construída utilizando uma base de dados, que será o conjunto de treinamento.

Em uma base de dados, pode haver vários registros com os mesmos valores dos atributos levando a diferentes valores de classes. Em casos como esse, a classe encontrada ao fim do caminho representa a mais provável, tendo em vista os valores assumidos pelos atributos (QUINLAN, 1992).

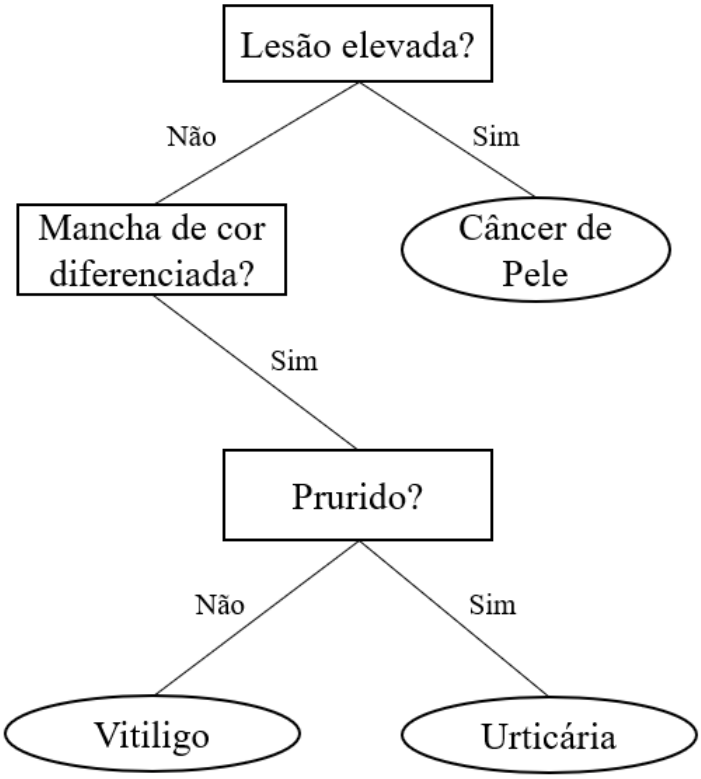
A maioria dos problemas lida com um alto número de atributos e uma grande variação entre os objetos da base de dados, o que resulta no aumento do tamanho da árvore e do conjunto de configurações diferentes. O objetivo ao se construir uma árvore de decisão é encontrar uma estrutura compacta que melhor reflita a base de dados. Porém, o problema de busca da menor árvore de busca consistente com a base de dados é NP-Completo (QUINLAN, 1992). Assim, existem algumas métricas e heurísticas utilizadas na construção da árvore de decisão.

2.1.2 Entropia

A entropia é uma métrica utilizada em várias áreas de estudo e está associada a incerteza. No contexto de inteligência artificial, Mitchell (1997, p. 57, tradução nossa) considera que: "A entropia como uma medida de impureza de um certo conjunto de objetos". Assim, dado um conjunto de objetos S, com n classes, sua entropia é:

$$H(S) = -\sum_{i=1}^{N} P_i \log P_i \tag{2.1}$$

Figura 7 – Árvore de decisão baseada nos dados do quadro 1



Fonte: O autor (2020)

2.1.3 Ganho de Informação

A árvore de decisão é construída através da seleção de qual atributo ocupará cada nó da árvore. Uma vez que o conceito de entropia foi definido, ela pode ser utilizada para avaliar o quão efetivo é um atributo na classificação da base de dados. Esta métrica de avaliação é chamada ganho de informação, ela representa a redução esperada da entropia, causada pelo particionamento da base de dados de acordo com o atributo escolhido (MITCHELL, 1997). Assim, o ganho de informação de um atributo A em um conjunto de objetos S é dado por:

$$G(S, A) = H(S) - H(S|A) \tag{2.2}$$

$$= H(S) - \sum_{v \epsilon valores(A)} \frac{|S_v|}{S} H(S_v) \tag{2.3}$$

Onde:

- $Valores(A)$ é o conjunto de valores que o atributo A pode assumir.
- S_v é o subconjunto de S em que o atributo A possui valor v.

Para cada nó da árvore, calcula-se o ganho de informação fornecido por cada atributo e aquele que possuir maior ganho é escolhido.

2.2 REDES NEURAIS ARTIFICIAIS

2.2.1 Definição

As redes neurais artificiais, comumente chamadas apenas de redes neurais, representam um conjunto de modelos computacionais cujo funcionamento é inspirado no cérebro humano, mais especificamente em seus neurônios e suas intercomunicações (HAYKIN, 2009). Matematicamente falando, de acordo com Mitchell (1997, p. 81, tradução nossa): "Métodos de aprendizagem de redes neurais fornecem uma abordagem robusta para a aproximação de funções objetivo". Em outras palavras, possuímos um determinado conjunto de dados de uma função desconhecida e o fornecemos a uma rede neural para gerar artificialmente uma função que se aproxime da função desconhecida.

O funcionamento de um neurônio ocorre da seguinte maneira: Um neurônio recebe um conjunto de sinais, onde cada um deles possui um determinado estímulo associado, que pode aumentar ou diminuir o respectivo sinal em que ele atua. Após receber estes sinais estimulados como entrada, o neurônio executa um certo processo neles, que resulta em um novo sinal que será transmitido a outros neurônios. Uma rede neural é composta por um conjunto de neurônios (HAYKIN, 2009). Os sinais que cada neurônio recebe são representados por dados de entrada, os estímulos que atuam neles são seus pesos. Além disso, é adicionado um viés ao neurônio, responsável por aumentar seu poder matemático. A saída do neurônio é uma função dos dados de entrada, seus respectivos pesos e viés – esta função é chamada de função de ativação.

Assim, uma rede neural artificial é composta por dados de entrada, seus respectivos pesos, um viés, e uma função de processamento chamada função de ativação.

A seguir, temos o diagrama de um neurônio com 3 entradas. Onde x_1,x_2 e x_3 representam os dados de entrada, w_1, w_2 e w_3 seus respectivos pesos, b o viés do neurônio, e φ a função de ativação.

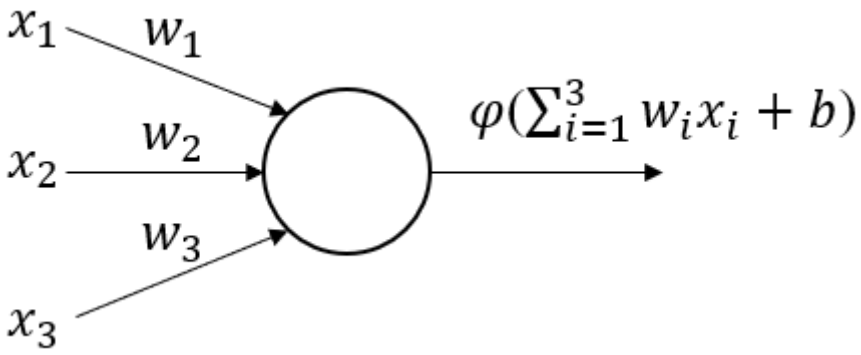
É chamado de perceptron o neurônio cuja função de ativação possui a forma a seguir:

$$\varphi\left(\sum_{i=1}^{m} w_i x_i + b\right) = \begin{cases} 0, \text{ se } \sum_{i=1}^{m} w_i x_i + b \leq 0 \\ 1, \text{ se } \sum_{i=1}^{m} w_i x_i + b > 0 \end{cases}$$

Onde m é o número de entradas do neurônio.

Os pesos e vieses são alterados de forma a melhorar a atuação do neurônio em determinado problema, eles são chamados de hiperparâmetros. Os hiperparâmetros representam aspectos da configuração de um algoritmo de aprendizagem de máquina (GOODFELLOW, 2015). O conjunto de entrada de um neurônio será representado pelo vetor $x = [x_1, x_2, x_3]$, os pesos serão representados pelo vetor $w = [w_1, w_2, w_3]$.

Figura 8 – Arquitetura de um neurônio



Fonte: O autor (2020)

2.2.2 Arquitetura

As redes neurais artificias são geralmente organizadas em camadas, onde cada camada possui um certo número de neurônios. O número de camadas, de neurônios em cada uma delas, e suas respectivas funções de ativação são decisões que caracterizam a arquitetura da rede.

A primeira camada de toda rede corresponde aos dados de entrada, cada nó representa um dado diferente e não neurônios. A última camada é a camada de saída, ela representa o resultado da rede, pode ser um valor previsto ou a probabilidade da entrada pertencer a uma ou mais classes. As camadas intermediárias são chamadas de camadas ocultas e correspondem à maior parte do processamento da rede.

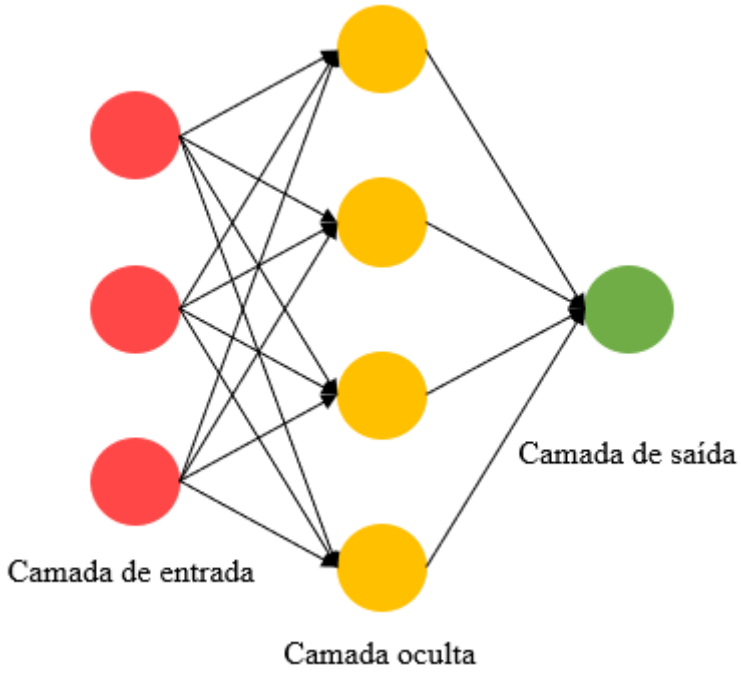
Neste trabalho, utilizaremos apenas redes do tipo *feedforward*. As redes *feedforward* são aquelas cujos neurônios obedecem à restrição de se conectarem apenas com as camadas seguintes, e não com camadas anteriores (HAYKIN, 2009). Um tipo de arquitetura que vale ser citada é a perceptron multicamadas (MLP) que, segundo Haykin (2009, p. 123), possui a seguintes características básicas:

- Cada neurônio possui uma função de ativação não-linear que é diferenciável.
- A rede contém uma ou mais camadas ocultas.
- A rede possui um alto nível de conectividade entre os nós.

2.2.3 Aprendizagem

O processo de aprendizagem ocorre com base em exemplos, que estão contidos no conjunto de treinamento. Neste trabalho, é realizada a aprendizagem supervisionada, ou seja, todos os pares presentes no conjunto de treinamento possuem um y conhecido associado (MURPHY, 2012). Desta forma, o modelo aplicado tem a informação de qual é

Figura 9 – Arquitetura básica de rede neural



Fonte: O autor (2020)

a saída correta para uma determinada entrada, pertencente ao conjunto de treinamento. Cada objeto do conjunto de treinamento é composto por N atributos representados pelo vetor $x = [x_1, x_2, \ldots, x_N]$, que são os dados que caracterizam o objeto, e uma classe y, que corresponde à saída que deve ser dada pelo modelo de rede neural. A ideia é que existe algum mapeamento, representado pela função $f(x)$ que fornece o y de cada objeto do conjunto de treinamento, com base nos atributos deste objeto. O objetivo da rede neural é atuar como um método de aproximação, utilizando θ, que representa todos os hiperparâmetros da rede, ou seja, seus pesos e vieses, fornecendo uma função $f_{aproximada}(x;\theta)$ que se aproxime da função verdadeira $f(x)$.

A função aproximada é construída com base na alteração dos valores dos hiperparâmetros, assim, a melhor configuração de hiperparâmetros fornece a melhor aproximação: $f_{aproximada}(x;\theta) \approx f(x)$. O problema que surge é encontrar esta melhor configuração (BISHOP, 2006).

O aprendizado consiste na atualização dos pesos e vieses da rede de forma a minimizar a função de custo. O algoritmo de atualização de parâmetros e a função de custo são aspectos de arquitetura do modelo.

Quando o treinamento da rede neural é finalizado, é preciso que se avalie o nível de sua precisão. Isso é feito através de um novo conjunto de dados, chamado conjunto de validação. No processo de validação, não ocorre computação da função de perda e nem atualização dos parâmetros do modelo. O objetivo é apenas verificar os acertos e erros

das predições da rede neural (BISHOP, 2006).

2.3 APRENDIZAGEM PROFUNDA

O modelo de redes neurais foi desenvolvido utilizando como base o cérebro humano. Porém, um outro aspecto que foi notado é que o seu funcionamento utiliza vários níveis de processamento. Teoriza-se de que esses diferentes níveis atuem aumentando o grau de abstração da informação que flui através deles. Essa ideia inspirou a criação do modelo de aprendizagem profunda (MURPHY, 2015). Para que o computador incorpore a capacidade de interpretar conceitos complexos, é necessário que ele possua um alto poder representativo. Este poder representativo é fornecido pelo conceito de aprendizagem profunda.

Segundo Goodfellow (2015, p. 1, tradução nossa): "A ideia da aprendizagem profunda é expressar o mundo em termos de uma hierarquia de conceitos, onde cada conceito é definido por sua relação com um conceito mais simples". Estes conceitos são construídos e simplificados ao longo das camadas da rede.

2.4 REDES NEURAIS CONVOLUCIONAIS

De acordo com Haykin (2009, p. 201, tradução nossa), "uma rede neural convolucional (CNN) pode ser descrita como uma rede neural com arquitetura MLP desenvolvida especificamente para o reconhecimento de formas bidimensionais [...]".

A abordagem utilizada no desenvolvimento de uma CNN é a de construir propriedades invariantes na estrutura da rede, de tal forma, que ela seja invariante a certas transformações aplicadas aos dados de entrada (BISHOP, 2006). Assim, uma CNN se torna capaz de detectar padrões mesmo quando estes estão sujeitos a mudanças escalares e translações, por exemplo.

A identificação de uma imagem dada como entrada para uma CNN ocorre a partir da identificação dos atributos presentes nela e como eles estão organizados entre si. Assim, Haykin (2009) divide o funcionamento da CNN possui as seguintes etapas:

1. Extração de atributos ou construção de atributos: Em uma imagem, cada pixel individualmente não fornece muita informação. É preciso que um conjunto de pixels próximos, correspondente à uma certa região da imagem, sejam avaliados para que alguma informação seja obtida (MURPHY, 2015). Uma imagem é dividida em um conjunto de regiões, que recebem o nome de campos receptivos locais. Em cada campo receptivo, atua uma matriz de pesos chamada de *kernel*, ele é responsável por detectar se um determinado padrão ocorre em cada região da imagem. Cada um dos pesos do *kernel* multiplica o seu pixel associado na região que está sendo processada. Depois, os produtos resultantes são somados entre si e a um viés, o

resultado sofre ação de uma função de ativação, gerando um único valor para cada região processada (BISHOP, 2006).

2. Mapeamento de atributos: Após o *kernel* ter sido aplicado em todos os campos receptivos da imagem, o resultado será um conjunto de valores associados a cada uma delas. Estes valores se distribuem em um plano, a posição que os valores assumem nele é decorrente da posição do campo receptivo, na qual eles foram calculados, na imagem. Este plano é chamado de mapa de atributos. A configuração dos pesos do *kernel* é responsável pelas diferentes configurações assumidas pelo mapa de atributos. O conjunto de mapas de atributos, gerado por diferentes campos receptivos, é chamado de camada convolucional (BISHOP, 2006).

3. Subamostragem: Após a camada convolucional, existe uma camada de subamostragem. Nela, os mapas de atributos são utilizados como entrada e passam por um processo que será responsável por reduzir a sua resolução (HAYKIN, 1993). A geração da camada de subamostragem ocorre da mesma forma que a da camada convolucional, porém, a diferença é que os campos receptivos obedecem a restrição de não se sobreporem (BISHOP, 2006).

2.5 COMBINANDO AMBAS AS ABORDAGENS

Dois modelos foram apresentados com o objetivo de se obter o diagnóstico dermatológico de um paciente. Ambos por meios diferentes, um utilizando dados textuais, e o outro, uma imagem da lesão dermatológica do paciente. Por este motivo, cada um deles emite seu próprio diagnóstico, de tal forma, que após o processamento dos dados, existe um diagnóstico associado aos dados textuais e outro à imagem. Porém, o diagnóstico médico não é obtido desta forma, os dados não são analisados separadamente, mas sim em conjunto. Assim, surge a necessidade de integrar os diagnósticos de ambos os modelos.

Um exemplo de uma série de métodos que utilizam múltiplos algoritmos de aprendizagem é o comitê de classificadores. Porém, a forma como ambas as abordagens serão combinadas não é o foco deste trabalho e será deixada para possíveis trabalhos futuros.

3 SISTEMA

A apresentação dos diferentes aspectos que compõe o sistema proposto neste trabalho é realizada nesta seção. Para facilitar o entendimento do mesmo, ela é dividida da seguinte forma:

- Modelos de inteligência artificial: Apresentação do algoritmo de árvore de decisão escolhido para o mapeamento das informações e sintomas do paciente para o diagnóstico, e da arquitetura de rede neural convolucional utilizada para a classificação do diagnóstico da imagem da lesão fornecida.
- Tecnologias e Bibliotecas: Descrição das tecnologias e bibliotecas que foram utilizadas para o desenvolvimento dos diferentes aspectos do sistema.
- Arquitetura: Apresentação de como as tecnologias e bibliotecas utilizadas, como elas se relacionam de forma a suportar o funcionamento do sistema.
- Banco de dados: Descrição da modelagem utilizada no banco de dados.
- Funcionamento: Descrição geral do projeto do sistema, seu uso e de suas funcionalidades.
- Protótipo: Apresentação das telas e funcionalidades que foram implementadas como protótipo do sistema proposto.

3.1 MODELOS DE INTELIGÊNCIA ARTIFICIAL

3.1.1 CART

O algoritmo utilizado para a construção da árvore de decisão é o *CART*, que é baseado nos algoritmos *ID3* e *C4.5*. Assim, é necessário que o *ID3* e o *C4.5* sejam apresentados antes do *CART*.

O *ID3* constrói a árvore através da seleção de qual atributo o maior ganho de informação para cada nó da árvore. Após a árvore ser construída e tiver alcançado o tamanho máximo, ela passa por um processo de poda, no qual ramos da árvore são removidos com o objetivo de se obter uma melhor capacidade de generalização para novos dados. Uma restrição do *ID3* é que os atributos do problema precisam ser categóricos.

O *C4.5* é considerado o sucessor do *ID3*, pois nele os atributos não precisam ser categóricos. Ao receber atributos de domínio contínuo, o *C4.5* dinamicamente os discretiza transformando em um conjunto discreto de intervalos. O resultado da execução do *ID3* é uma árvore treinada, o *C4.5* transforma esta árvore em uma série de regras de se-então.

O *CART* é similar ao *C4.5*, porém, ele suporta valores numéricos para y, permitindo assim que ele realize tarefas de regressão. Além disso, o *CART* não computa o conjunto de regras do *C4.5*.

3.1.2 INCEPTION-V3

Inception é o nome dado à uma arquitetura de rede convolucional profunda proposta por Szegedy et al. (2014). Ela foi desenvolvida para a competição *ImageNet Large-Scale Visual Recognition Challenge* 2014 (ILSVRC14), na qual estabeleceu o estado da arte.

A *ImageNet* é uma base de dados de imagens, com cerca de 14 milhões de imagens. Sua organização é orientada pela hierarquia *WordNet*, uma base de dados léxica da língua inglesa. Cada conceito presente na *WordNet* é chamado de *synset*, a *ImageNet* possui cerca de 21 mil *synsets* mapeados às suas respectivas imagens, ou seja, os *synsets* correspondem às categorias das imagens. A ILSVRC consiste na seleção de um subconjunto, de imagens e categorias, da *ImageNet* para avaliar a eficiência de certos algoritmos na realização de tarefas de classificação e detecção no subconjunto disponibilizado.

A importância da *Inception*, segundo Szegedy et al. (2014, p. 1, tradução nossa), é:

> O principal marco alcançado por esta arquitetura foi a melhoria no uso dos recursos computacionais dentro da rede. Isso foi alcançado através de um projeto cuidadosamente desenvolvido para permitir o aumento da profundidade e largura da rede, mantendo o custo computacional constante.

A *Inception* se utiliza de uma técnica chamada *Network in Network* (LIN; CHEN; YAN, 2014), cujo objetivo é a redução de dimensionalidade do dado na qual ela é aplicada. Esta técnica consiste no uso de uma convolução 1x1.

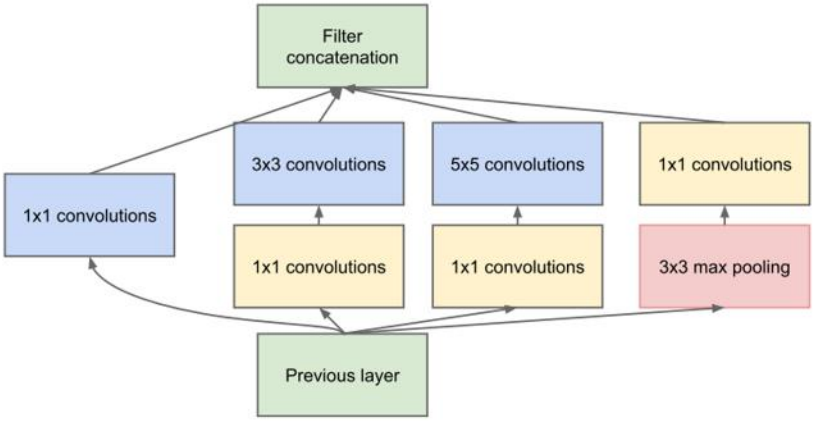
Uma das ideias que orientaram o desenvolvimento da rede foi a de aplicar diferentes tamanhos de *kernels* às imagens. Assim, "a arquitetura da *Inception* se alinha com a intuição de que o processamento de informação visual deve ocorrer em diferentes escalas e então agregadas para que os atributos da imagem possam ser abstraídos" (SZEGEDY et al., 2014, p. 5).

O elemento principal da *Inception* é chamado de módulo *Inception*. A arquitetura basicamente consiste na repetição de módulos *Inception* ao longo da rede. Este módulo, assim como as operações que o compõem, podem ser vistas na imagem a seguir.

A *Inception* possui 3 saídas, ou seja, ela possui 2 classificadores auxiliares. Eles atuam em camadas intermediárias com vários objetivos, dentre eles, aumentar o sinal do gradiente durante a fase de *backpropagation* (SZEGEDY et al., 2014, p. 6).

A escolha da *Inception* como arquitetura para a CNN foi motivada pelo artigo *Detecting Cancer Metastases on Gigapixel Pathology Images*, Liu et al. (2017). Nele, é proposto

Figura 10 – Módulo *Inception*



Fonte: Szegedy et al. (2015)

um *framework* de CNN, para detecção e localização de tumores, que é efetivamente utilizado para a tarefa auxiliar a detecção de metástase de câncer de mama em linfonodos, na qual estabeleceu o estado da arte. A arquitetura de CNN utilizada no *framework* é chamada de *Inception-v3* (LIU et al., 2017). Assim, ao utilizar a *Inception-v3* neste trabalho, o sistema desenvolvido se beneficiaria, da eficiência fornecida pelo *framework* descrito, em aplicações futuras, que se assemelham à detecção e localização de tumores.

As arquiteturas *Inception-v2* e *Inception-v3* foram propostas em *Rethinking the Inception Architecture for Computer Vision*, Szegedy et al. (2015), como aperfeiçoamentos da *Inception* original, aumentando a precisão da rede e diminuindo sua complexidade computacional.

Entre as atualizações implementadas no modelo, pode-se citar:

- Criação de novos módulos *Inception*: O modelo original da *Inception* possui apenas 1 módulo *Inception*, que se repete ao longo da rede. Porém, em sua atualização *Inception-v3*, existem 3 diferentes módulos *Inception*, que se repetem ao longo da rede, assim como no modelo original (SZEGEDY et al., 2015, p. 6).

- Fatoração das convoluções: Na *Inception-v3*, foi aplicada uma técnica chamada fatoração de convolução em algumas operações de convolução. Ela consiste em substituir uma determinada convolução por outras de menor dimensionalidade. Ela é aplicada com o objetivo de diminuir o número de parâmetros do modelo (SZEGEDY et al., 2015, p. 3).

- Retirada de um classificador auxiliar: Na primeira versão da *Inception*, percebeu-se que os classificadores auxiliares não tinham o resultado esperado e a remoção do classificador mais intermediário não traz nenhum prejuízo à rede. Assim, a

Inception-v3 foi desenvolvida com apenas 1 classificador auxiliar (SZEGEDY et al., 2015, p. 4).

3.2 TECNOLOGIAS E BIBLIOTECAS

Nesta seção, são descritas as tecnologias e bibliotecas que foram utilizadas na arquitetura do sistema proposto neste trabalho, elas são apresentadas de acordo com o contexto em que foram aplicadas.

3.2.1 Frontend

- *React Native*: *Framework* criado pelo *Facebook*, com base em *React* (biblioteca *Javascript* para criação de interfaces de usuário), para o desenvolvimento de aplicações mobile, de tal forma que o mesmo código é executado no *Android* e *IOS*.

3.2.2 Backend

- *Python*: Linguagem de programação de alto nível, que permite o desenvolvimento rápido e a integração de sistemas de forma eficiente.
- *Flask*: *Microframework* desenvolvido em *Python*, para aplicações *web*. Ele é responsável por estabelecer a comunicação entre servidores web e suas aplicações.
- *PostgreSQL*: Banco de dados relacional escrito em código aberto, reconhecido pela sua confiabilidade e performance.
- *Psycopg2*: Biblioteca para *Python*, que funciona como um adaptador para o banco de dados *PostgreSQL*, permitindo a interação entre um código em *Python* e um determinado banco de dados *PostgreSQL*.
- *Scikit-Learn*: Biblioteca em código aberto, para *Python*, para aprendizagem de máquina.
- *Keras*: Biblioteca de alto nível, para o desenvolvimento de redes neurais, de código aberto escrita em *Python*. Foi criada com o objetivo de disponibilizar rápida experimentação de redes neurais.
- *Keras Applications*: Seção da *Keras* que contém uma série de modelos de redes neurais profundas implementadas e disponíveis com pesos pré-treinados. Tais modelos podem ser utilizados para tarefas de predição, por exemplo.

3.3 ARQUITETURA

Uma vez que as tecnologias e bibliotecas utilizadas foram apresentadas, o papel que elas assumem na estrutura do sistema, e como interagem entre si, também podem ser apresentados. Esta seção se dedica a descrever estes aspectos.

Existem diversas tecnologias disponíveis para o desenvolvimento do *frontend*, o ideal seria que ele fosse feito utilizando *Kotlin*, no caso do *Android*, e *Swift*, no caso do *IOS*, por serem linguagens que utilizam os recursos nativos do sistema. Porém, visando otimização de tempo e flexibilidade no desenvolvimento, decidiu-se utilizar um *framework*, o escolhido foi o *React Native*, tendo em vista o tamanho e atividade de sua comunidade. Assim, o mesmo código seria executado em ambos os sistemas.

Devido à natureza e arquitetura da aplicação descrita neste trabalho, o seu funcionamento é dependente da troca de informações, obtidas através do *frontend*, entre o usuário e o servidor onde o restante da aplicação está sendo executado, o *backend*. Desta forma, surge a necessidade do uso de uma tecnologia adequada para suportar este intercâmbio de dados.

Neste contexto, o sistema segue a arquitetura *REST*, no sentido de que possui uma estrutura de cliente-servidor, onde o servidor não armazena qualquer informação sobre o estado do cliente e todas as requisições são tratadas como independentes. O cliente envia requisições *HTTP* com as informações necessárias para o processamento pelo servidor e apresenta a resposta ao usuário.

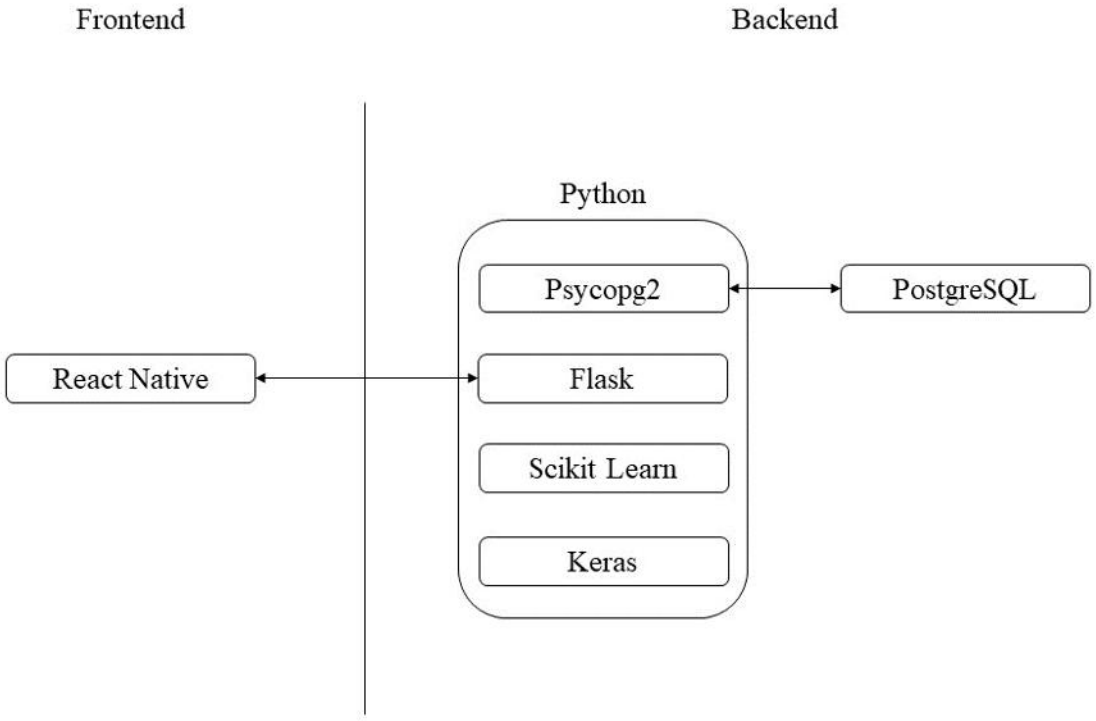
Como o foco do sistema é a execução dos algoritmos de inteligência artificial, que estão escritos em *Python*, visando um ambiente integrado, se definiu que a *API* para disparar tais algoritmos e acessar o banco de dados também deveria ser desenvolvida em *Python*. O *framework Flask* foi escolhido pela sua simplicidade, permitindo o desenvolvimento rápido e fornecendo bastante liberdade no processo. Assim, o *Flask* é o responsável pela interação com o *frontend*.

A aplicação do sistema proposto neste trabalho exige o armazenamento e manutenção de dados. Assim, surge a necessidade de um banco de dados, o escolhido foi o *PostgreSQL*.

Uma vez que a aplicação utiliza um banco de dados, é necessário que a *API* interaja com ela. Como o *Flask* é escrito em *Python*, a biblioteca disponível nesta linguagem é a *Psycopg2*.

Com a estrutura de troca e armazenamento de dados definida, os módulos de inteligência artificial podem ser descritos. Para o algoritmo de árvore de decisão, a biblioteca utilizada é a *Scikit-Learn*. Para a rede neural convolucional, a implementação da *Inception-v3* utilizada foi a disponível na biblioteca *Keras*, pré-treinada na base de dados completa da *ImageNet*.

Figura 11 – Arquitetura da aplicação



Fonte: O autor (2020)

3.4 BANCO DE DADOS

Como sistema precisa realizar autenticação de usuários, controlar os diagnósticos associados a cada médico e armazenar os dados utilizados para o treinamento dos modelos de inteligência artificial, é necessário que exista um banco de dados. As entidades do banco de dados serão simplificadas, tendo em vista o escopo deste trabalho. Elas são:

1. Um médico, o usuário do sistema, que possui *e-mail* e senha, utilizados para acessar a aplicação, CPF e número do CRM.

2. Um paciente, cujas informações são coletadas pelo médico. Os dados que identificam o paciente são o nome e o CPF.

3. Um conjunto de perguntas que são realizadas ao paciente, pelo médico, contendo N campos onde cada um é uma pergunta, que foram representadas por um único atributo multivalorado chamado genericamente de "Pergunta", o valor assumido por cada pergunta é a sua respectiva resposta. Além do conjunto de perguntas, esta entidade também contém a informação do diagnóstico associado a elas.

4. Uma entidade que representa a imagem, com o seu nome e seu respectivo diagnóstico.

Todas as entidades possuem o seu respectivo atributo identificador.

As entidades 3 e 4 são utilizadas nos modelos de inteligência artificial. E além dos campos descritos nelas, há também o campo "diagnóstico pendente" que é responsável por armazenar a informação se o diagnóstico, correspondente ao dado, armazenado foi confirmado pelo médico ou não. Inicialmente, as tabelas referentes a essas entidades são preenchidas somente com o conjunto de treinamento, onde todos os diagnósticos estão confirmados. Ao lidar com um dado novo, o valor que o campo "diagnóstico" assume é o fornecido pelo modelo preditivo, e o campo "diagnóstico pendente" assume o valor "verdadeiro". Após o médico comprovar o diagnóstico, através de exames, ele deve atualizar o valor do "diagnóstico", caso seja diferente, com o diagnóstico real e o "diagnóstico pendente" com falso. Com isto realizado, os modelos podem utilizar este novo dado para atualização de seus parâmetros.

O paciente apenas passa a existir no sistema quando o médico conclui o processo de coleta de evidências, que consiste em realizar o conjunto de perguntas e tirar uma foto da lesão. E os dados referentes ao paciente só são armazenados enquanto o seu diagnóstico não é obtido, para que o médico responsável seja capaz de associar um ao outro, no sistema. Uma vez que seja obtido o diagnóstico, os dados do paciente são deletados, objetivando a sua privacidade.

Se a mesma pessoa passar pelo processo de coleta de evidências várias vezes, ela será representada por uma instância diferente, para cada uma delas, na tabela de pacientes. Em outras palavras, a mesma pessoa pode ser representada por vários pacientes no banco, onde cada um deles tem o mesmo nome e CPF.

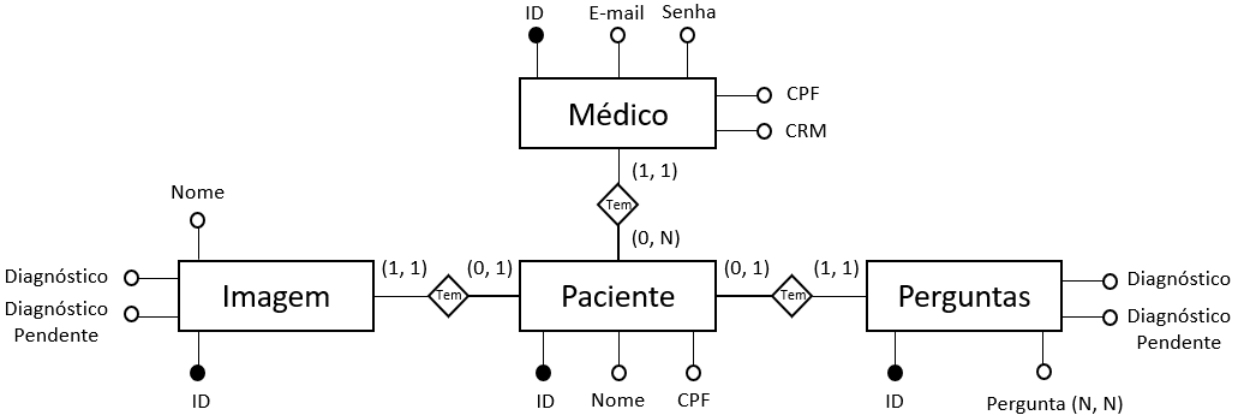
Os relacionamentos entre as entidades são:

- Um médico pode possuir nenhum ou vários pacientes.
- Um paciente só pode estar associado a um único médico, conjunto de perguntas e imagem.
- Tanto o conjunto de perguntas quanto a imagem podem estar associados a nenhum paciente, caso ele já tenha sido excluído do sistema, ou a apenas um.

A criação do paciente no sistema depende da existência de um médico, responsável pelo processo de coleta de informações e diagnóstico, e das informações coletadas, ou seja, o conjunto de perguntas e a imagem.

Todo paciente tem um conjunto de perguntas associado a ele, o número de perguntas neste conjunto é fixo.

Figura 12 – Arquitetura do banco de dados



Fonte: O autor (2020)

3.5 FUNCIONAMENTO

A descrição do funcionamento do sistema será orientada pela organização das diferentes telas que ele possui, pois é desta forma que o usuário interage com ele. As telas da aplicação são: “Login”, “cadastro”, “diagnósticos pendentes”, “novo diagnóstico” e “confirmação de diagnóstico”.

As funcionalidades principais da aplicação estão contidas nas telas de novo diagnóstico e confirmação de diagnóstico. Por este motivo, o foco deste trabalho é dado em ambas.

3.5.1 Login

O acesso ao sistema só pode ser feito aos usuários cadastrados. Assim, a primeira tela do aplicativo é responsável pela autenticação do usuário, permitindo acesso ao sistema, caso ele seja cadastrado, ou permitindo o cadastro, caso não seja.

Esta tela contém dois campos a serem preenchidos pelo usuário: “*e-mail*” e “senha”. E dois botões: “entrar”, “cadastrar-se”.

No primeiro acesso, o usuário sem cadastro deve cadastrar-se. Para que isso seja efetuado, ele deve clicar no botão “cadastrar-se”. Ao fazer isso, o usuário navega à tela de cadastro. Uma vez que o usuário possua cadastro, ele deve inserir o seu *e-mail* e senha nos seus respectivos campos e clicar no botão “entrar”. Neste momento, ocorre um procedimento de autenticação padrão, validando os dados do usuário no banco de dados.

3.5.2 Cadastro

Para acessar a aplicação, é necessário que um cadastro seja efetuado. O objetivo desta tela é fornecer uma interface para receber os dados do usuário, no caso, um médico. Uma

vez que os dados tenham sido inseridos, eles passam por um processo de validação antes de serem inseridos no banco de dados e o usuário ser efetivamente criado.

3.5.3 Diagnósticos Pendentes

Após o usuário efetuar o login, ele será direcionado para esta tela. Ela funciona como um menu principal, que fornece acesso às funcionalidades principais de "novo diagnóstico" e "confirmação de diagnóstico".

Uma vez que o médico tenha atendido um paciente e realizado um novo diagnóstico. Este diagnóstico ainda precisa ser confirmado pelo médico. Assim, o principal objetivo desta tela é apresentar todos os diagnósticos que ainda precisam ser confirmados pelo médico.

3.5.4 Novo Diagnóstico

Esta tela foi desenvolvida para ser acessada pelo médico no momento do atendimento do paciente. Para que o diagnóstico seja efetuado, uma série de informações são necessárias, o médico precisa coletá-las do paciente e fornecê-las ao sistema. Esta tela possui os campos para que elas sejam inseridas.

Para que seja possível associar o paciente ao seu diagnóstico, necessário que haja uma forma de identificar o paciente no sistema. Por este motivo, é necessário que o nome e o CPF do paciente sejam coletados.

Com as informações do paciente coletadas, a coleta de dados para os modelos de inteligência artificial já pode ser iniciada.

A tela de "novo diagnóstico" fornece uma série de perguntas/constatações que devem ser realizadas pelo médico a respeito dos sintomas do paciente. Conforme o médico obtém as respostas, ele deve inseri-las no sistema. Deve-se destacar que tais perguntas são de respostas de sim ou não. É feito desta forma pois tais dados são utilizados para a construção da árvore de decisão, que é binária.

Além das perguntas, a tela de novo diagnóstico também fornece a funcionalidade para o médico tirar uma foto da lesão dermatológica do paciente e inseri-la na aplicação. A foto tirada será fornecida como entrada para a rede neural convolucional.

Após o médico coletar as informações do paciente, ele deve clicar no botão "gerar diagnóstico". Este botão irá disparar a inclusão dos dados coletados pelo médico no banco de dados e a seguir executará os algoritmos de inteligência artificial, que após serem executados, também registrarão seus resultados no banco de dados.

Quando o diagnóstico tiver sido gerado, o usuário será redirecionado para a tela de "diagnósticos pendentes" e o diagnóstico criado ficará disponível para ser acessado nesta tela.

3.5.5 Confirmação de Diagnóstico

Os algoritmos de árvore de decisão e rede neural convolucional propõe cada um seu respectivo diagnóstico, com base nos dados que foram inseridos em cada um dos modelos. Porém, os resultados obtidos podem estar incorretos. E neste caso, é necessário que os modelos sejam corrigidos. Esta é a razão da existência da tela de confirmação de diagnóstico.

O objetivo dos resultados dos prováveis diagnósticos, de um determinado paciente, fornecido pelo sistema, é orientar o médico a descobrir o real diagnóstico de forma mais rápida. E quando ele tiver sido obtido, o médico deve retorná-lo ao sistema.

A tela de confirmação de diagnóstico apresenta ao usuário todas as informações que foram coletadas a respeito do paciente e de seus sintomas. Além disso, a tela também apresenta campos para que o médico confirme se os diagnósticos sugeridos pelos modelos de árvore de decisão e rede neural convolucional estão corretos ou não.

Se os resultados dos modelos estiverem incorretos, então o médico deve inserir o diagnóstico correto no sistema e clicar no botão "confirmar diagnóstico". E uma vez que isso seja realizado, os registros nos bancos de dados serão atualizados e este diagnóstico deixará de ser considerado um diagnóstico pendente. Além disso, após o diagnóstico ter sido confirmado, estes dados poderão ser utilizados como treinamentos para os modelos de inteligência artificial.

3.6 PROTÓTIPO

Na descrição das funcionalidades houve destaque para aquelas relacionadas a inteligência artificial. A implementação do protótipo do sistema também obedece a este mesmo raciocínio. Assim, o desenvolvimento visou obter uma estrutura onde houvesse:

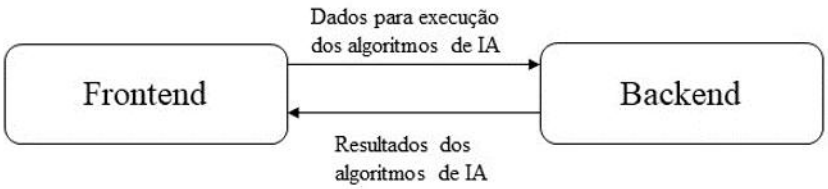
- Entrada de dados, por um usuário, através de um *frontend*.
- Execução dos algoritmos de inteligência artificial utilizando os dados de entrada.
- Apresentação dos resultados, obtidos pelos algoritmos, de volta ao usuário, através do *frontend*.

O banco de dados do protótipo tem a maioria das características apresentadas na seção de banco de dados. Porém, a tabela de perguntas, onde os dados que alimentam o algoritmo da árvore de decisão são armazenados, merece destaque.

Como o modelo de árvore de decisão escolhido não é pré-treinado, foi necessário que fossem inseridos dados falsos, para que houvesse uma árvore construída no momento de execução da aplicação.

Apenas para os fins de teste do protótipo, se definiu que o número de atributos que a árvore decisão se baseia são 3, ou seja, 3 perguntas devem ser respondidas no sistema, para

Figura 13 – Diagrama de Comunicação entre *frontend* e *backend*

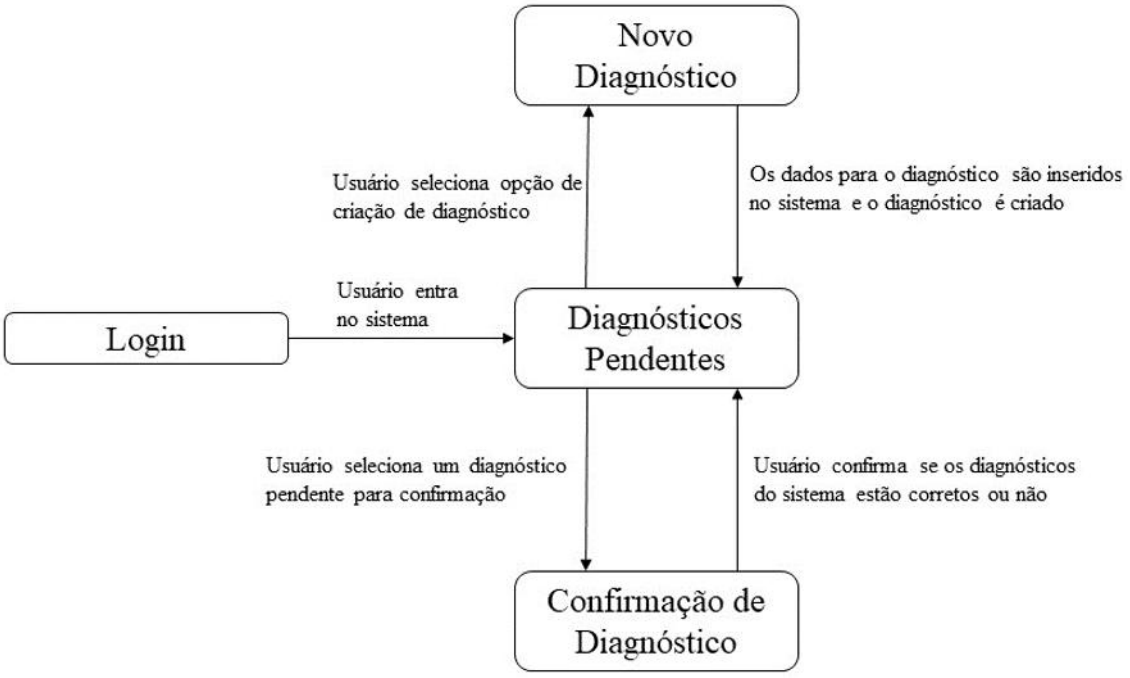


Fonte: O autor (2020)

que uma predição de diagnóstico possa ser realizada. Elas foram definidas genericamente apenas como pergunta 1, 2 e 3.

A forma como o usuário navega nas telas do sistema, utilizando diferentes operações é apresentada no diagrama a seguir.

Figura 14 – Fluxo de navegação de telas



Fonte: O autor (2020)

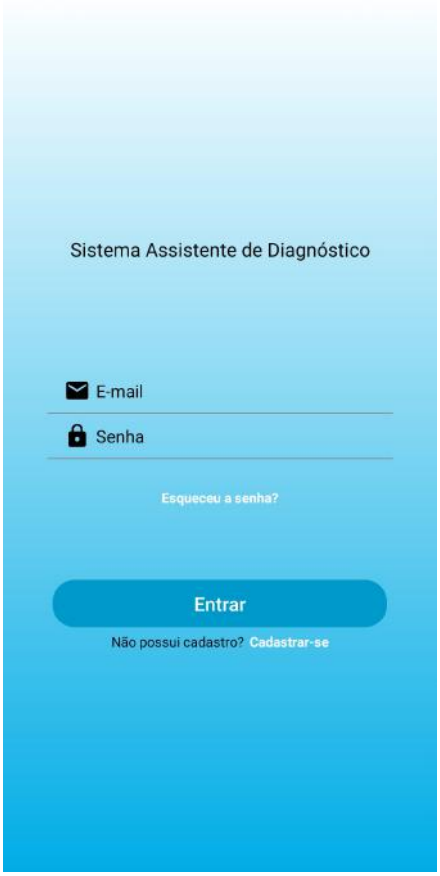
3.6.1 Login

A tela inicial da aplicação é a responsável pela autenticação do usuário, porém, como o foco do protótipo não é este, esta tarefa se restringiu apenas a verificar se os dados inseridos pelo usuário estão contidos no banco de dados do sistema. Assim, funcionalidades como cadastro e recuperação de senha não estão disponíveis. O único usuário do sistema foi inserido diretamente no banco.

Uma vez que o usuário digite seu *e-mail* e senha em seus respectivos campos e clique no botão entrar, o seguinte processo ocorrer:

1. O *frontend* envia uma requisição, contendo o *e-mail* e senha, para o *backend*.
2. O *backend* recebe a requisição
3. Verifica-se se o *e-mail* e a senha recebidos correspondem a um usuário no banco.
4. A resposta da etapa 3 é enviada para o *frontend*.
5. Se a resposta for positiva, o usuário é direcionado para a tela de diagnósticos pendentes.
6. Se não, o usuário recebe uma mensagem, informando que o *e-mail* ou senha estão incorretos.

Figura 15 – Tela de *login*



Fonte: O autor (2020)

3.6.2 Diagnósticos Pendentes

O objetivo desta tela é apresentar ao usuário todos os diagnósticos que ainda precisam ser confirmados por ele. Assim, o médico que está utilizando o sistema apenas precisa receber os diagnósticos pendentes dos pacientes que foram atendidos por ele.

Esta tela também é responsável por direcionar o usuário para as duas principais funcionalidades do sistema, a criação e a confirmação de diagnóstico.

Para criar um diagnóstico, basta clicar no botão '+", que o usuário será direcionado para a tela que fornece a funcionalidade de criação de diagnóstico.

Caso haja diagnósticos pendentes listados nesta tela, o usuário poderá acessar cada um deles através de um clique. Ao fazer isso, ele será direcionado para a tela de confirmação de diagnóstico.

O fluxo de funcionamento da tela de diagnósticos pendentes é apresentado abaixo:

1. Uma requisição é enviada ao *backend*, com o id do usuário que está acessando a aplicação.
2. O *backend* recebe requisição.
3. São consultados todos os pacientes, associados ao id recebido na etapa 1, que possuem diagnóstico pendente igual a verdadeiro.
4. A resposta com o resultado da etapa 3 para o *frontend*.
5. Se houver diagnósticos, o usuário os verá listados na tela.
6. Se não, o usuário receberá uma mensagem informando que não há diagnósticos pendentes..

Figura 16 – Tela de diagnósticos pendentes: sem diagnósticos

Fonte: O autor (2020)

Figura 17 – Tela de diagnósticos pendentes: com diagnóstico

Fonte: O autor (2020)

3.6.3 Novo Diagnóstico

Esta etapa é a mais importante de todo o sistema, pois envolve a execução dos algoritmos de inteligência artificial. Esta tela permite que o usuário insira os dados que são utilizadas para a geração do diagnóstico do paciente.

Os campos desta tela são:

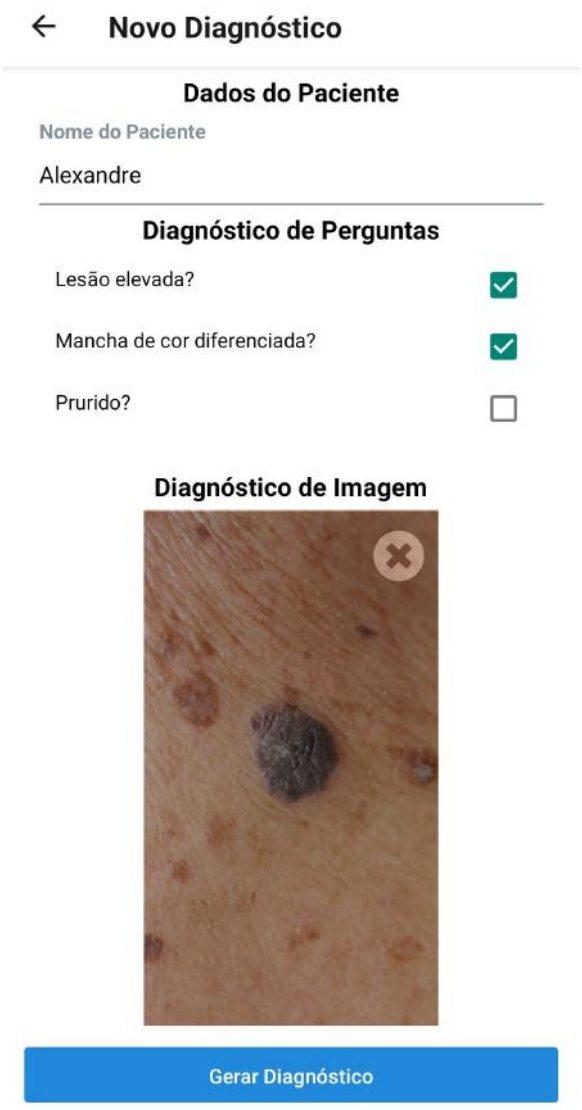
- Nome do paciente: O objetivo deste campo é auxiliar a identificação do paciente, para permitir que os diagnósticos sejam mapeados a ele.

- Pergunta 1, Pergunta 2 e Pergunta 3: Campos que simulam as perguntas que seriam feitas ao paciente, cujas respostas são utilizadas pela árvore de decisão.
- Foto da lesão: Campo que permite que o médico tire uma foto da lesão dermatológica do paciente. A imagem é utilizada pela rede neural convolucional.

No momento em que o usuário termina de inserir as informações desta tela e clica no botão 'gerar diagnóstico', elas são enviadas para o *backend*, salvas no banco de dados e os algoritmos são executados.

1. Uma requisição é enviada ao *backend*, contendo o id do usuário, o nome do paciente, as respostas das perguntas realizadas, e o arquivo de imagem da foto tirada.
2. As respostas das perguntas são convertidas para 1 e 0, e armazenadas com o status de diagnóstico pendente igual a verdadeiro.
3. O algoritmo da árvore de decisão é disparado com base nos dados com o id da instância da etapa 2, e o resultado do algoritmo é inserido no banco.
4. Os dados referentes ao diagnóstico de imagem são instanciados no banco, com o status de diagnóstico pendente igual a verdadeiro.
5. O arquivo de imagem é salvo.
6. O algoritmo de rede neural convolucional é disparado com base nos dados com o id da instância da etapa 4 e o resultado é inserido no banco.
7. Uma instância de paciente é criada, associada aos diagnósticos instanciados em 2 e 4.
8. Uma resposta de conclusão é retornada ao *frontend*.

Figura 18 – Tela de novo diagnóstico: câncer de pele



Fonte: O autor (2020)

Figura 19 – Tela de novo diagnóstico: vitiligo

Fonte: O autor (2020)

3.6.4 Confirmação de Diagnóstico

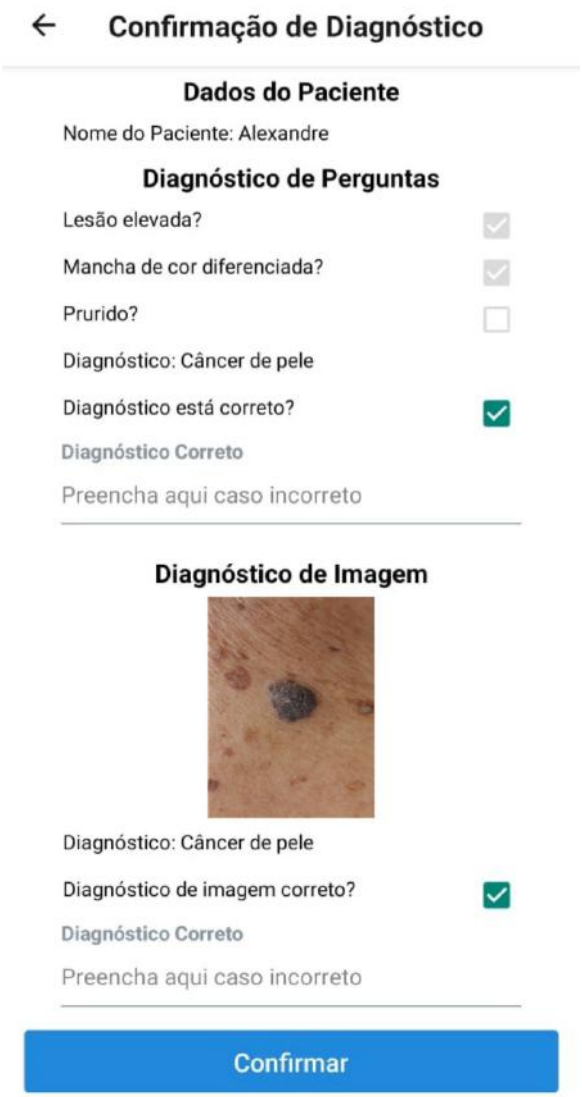
Após exames serem realizados no paciente, é provável que seu verdadeiro diagnóstico tenha sido descoberto. Neste caso, o médico deve retorná-lo ao sistema através da tela de confirmação de diagnóstico.

Na tela de confirmação de diagnóstico, o médico confirma se o diagnóstico fornecido pelo sistema está correto, ou não, através do campo 'diagnóstico correto?'. Este campo está disponível para o diagnóstico de perguntas e para o diagnóstico de imagem. E uma vez que seja informado que um desses diagnósticos está incorreto, o campo 'diagnóstico correto' será habilitado para inserção, e o médico deve inserir o verdadeiro diagnóstico do paciente.

Após realizar a confirmação, o usuário deve clicar no botão 'confirmar', e neste momento o seguinte processo é realizado.

1. Uma requisição é enviada para o *backend*, contendo o id do paciente, o id do diagnóstico das perguntas, id do diagnóstico de imagem, e os valores verdadeiros de ambos os diagnósticos.
2. O *backend* recebe a requisição.
3. O diagnóstico das perguntas é atualizado com o valor enviado na etapa 1 e o diagnóstico pendente passa a ser falso.
4. O diagnóstico de imagem é atualizado com o valor enviado na etapa 1 e o diagnóstico pendente passa a ser falso.
5. Os registros do paciente são removidos.
6. Uma resposta de conclusão é enviada ao *frontend*.
7. O usuário é direcionado para a tela de diagnósticos pendentes.

Figura 20 – Tela de confirmação de diagnóstico: diagnóstico correto



Fonte: O autor (2020)

Figura 21 – Tela de novo Diagnóstico: diagnóstico incorreto

Fonte: O autor (2020)

4 CONCLUSÃO

A inteligência artificial se popularizou muito nos últimos anos, diversas aplicações e avanços têm surgido rapidamente, consequentemente, tarefas complexas tem cada vez mais possibilidade de serem automatizadas. Neste contexto, inúmeras áreas podem se beneficiar, uma das mais importantes é a medicina.

O objetivo deste trabalho foi desenvolver uma aplicação de inteligência artificial em medicina, cuja função é auxiliar no processo de descoberta dos diagnósticos dos pacientes. A especialidade médica escolhida foi a dermatologia, pelo fato dos sintomas e lesões de uma determinada doença serem mais facilmente visualizados.

Assim, a ideia é a de fornecer as informações sobre a presença dos sintomas no paciente e uma foto de sua lesão dermatológica aos modelos de IA e obter os possíveis diagnósticos com base em ambas.

A aplicação proposta possui 2 módulos de IA: um de árvore de decisão, devido ao mapeamento natural que se faz entre os sintomas de um paciente e seus possíveis diagnósticos; e um de rede neural convolucional, por ser o modelo mais eficiente atualmente para se trabalhar com imagens.

Um sistema foi desenvolvido para fornecer suporte à execução dos algoritmos de IA gerenciando os dados utilizados por eles e suas execuções. Além de fornecer uma interface de usuário para simular uma utilização da aplicação no mundo real.

O objetivo foi alcançado, desde que se conseguiu desenvolver a estrutura funcional do sistema e da execução dos módulos de inteligência artificial. Porém, um problema inerente à IA é a necessidade de uma quantidade considerável de dados para o treinamento dos modelos. Tal conjunto de dados precisaria apresentar dois mapeamentos: um entre sintomas e doenças, e outro entre imagens de lesões dermatológicas e suas doenças. Contudo, esta é uma tarefa complexa que necessita de grande cooperação médica.

A ausência de um conjunto de treinamento adequado é o principal desafio da maioria das aplicações de IA, o mesmo ocorre neste trabalho. E é uma possibilidade de pesquisa futura, para que o sistema proposto solucione o problema para o qual ele foi idealizado, que é atuar como um assistente de diagnóstico.

5 TRABALHOS FUTUROS

Além de efetivamente trabalhar na elaboração de um conjunto de dados para tornar a aplicação deste trabalho viável, como trabalhos futuros, pode-se expandir o sistema:

- Combinar ambas as abordagens, de aprendizagem de máquina, utilizadas de forma eficiente.
- Treinar uma rede neural convolucional para identificar se a imagem fornecida como entrada realmente é uma imagem de lesão dermatológica.
- Atender outras especializações médicas, porém, para isso ser realizado, seria necessário a reunião de dados para o treinamento dos algoritmos de IA.
- Uma versão para ser acessada via web pelo PC, utilizando React, por exemplo. O objetivo seria aceitar o upload de exames de imagens, como lâminas. Porém, este aspecto, assim como o anterior, também necessita da reunião de dados para treinamento.
- Utilizar os dados reunidos para o desenvolvimento de uma plataforma para ensinar estudantes de medicina a realizar diagnósticos.

REFERÊNCIAS

MITCHEL, T. M. **Machine Learning**. McGraw-Hill Science/Engineering/Math. 1 mar. 1997.

ROKACH, L.; MAIMON, O. **Data Mining with Decision Trees: Theory and Applications**. 2 ed. WSPC. 3 set. 2014.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3 ed. Prentice Hall. 1 nov. 2010.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. The Mit Press. 1 jan. 2016.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. Morgan Kaufmann. 15 out. 1992.

HAYKIN, S. **NeuralNetworks and Learning Machines**. 3 ed. Prentice Hall. 2009.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. The Mit Press. 24 ago. 2012.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Springer. 2006.

SZEGEDY, C. et al. **Going Deeper with Convolutions**. 2014.

LIN, M.; CHEN, Q.; YAN, S. **Network in Network**. 2014.

LIU, Y. et al. **Detecting Cancer Metastases on Gigapixel Pathology Images**. 2017.

SZEGEDY, C. et al. **Rethinking the Inception Architecture for Computer Vision**. 2015.

APÊNDICES

APÊNDICE A – MÓDULO DE REDE NEURAL CONVOLUCIONAL

```
from keras.applications import inception_v3
from keras.applications.inception_v3 import decode_predictions
from keras.layers import Input

from keras import backend as K

from PIL import Image

import psycopg2

import numpy as np

# Images directory
DEFAULT_DIRECTORY = ''

def imageDiagnosis(queryId):
    # Clear Keras session
    K.clear_session()

    # Database connection
    con = psycopg2.connect(host='localhost', database='DAS',
        user='ai', password='ai1234')
    cur = con.cursor()

    # Get image info
    sql = 'select address from public.image where "ID" =' +
        queryId
    cur.execute(sql)
    result = cur.fetchall()

    # Define neural network input
    input_tensor = Input(shape=(224, 224, 3))
```

```
# Define inception-v3 pre-trained model
model = inception_v3.InceptionV3(input_tensor=input_tensor,
    weights='imagenet', include_top=True)

# Open image
img_path = DEFAULT_DIRECTORY + result[0][0]
img = Image.open(img_path)

# Resize the image
img_resized = img.resize((224,224))

# Converts image to numpy array
data = np.asarray(img_resized)

# Insert a new axis at position 0
data = np.expand_dims(data, axis=0)

# Preprocess input
x = inception_v3.preprocess_input(data)

# Predict the label of x
preds = model.predict(x)

# Decode the imagenet id to its label
diagnosis = decode_predictions(preds, top=1)[0][0][1]

# Update image info on database
sql = 'UPDATE public.image SET diagnosis = \'' + diagnosis +
    '\' where "ID" = ' + queryId
cur.execute(sql)
con.commit()
```

APÊNDICE B – MÓDULO DE ÁRVORE DE DECISÃO

```
from sklearn import tree

import psycopg2

import numpy as np

def questionsDiagnosis(queryId):
    # Database connection
    con = psycopg2.connect(host='localhost', database='DAS',
        user='ai', password='ai1234')
    cur = con.cursor()

    # Get features and labels
    sqlFeatures = 'select question1, question2, question3 from
        public.questions where pendent = false'
    sqlLabels = 'select diagnosis from public.questions where
        pendent = false'

    cur.execute(sqlFeatures)
    features = cur.fetchall()

    cur.execute(sqlLabels)
    labels = cur.fetchall()

    # Create features array
    x = []
    for i in features:
        sample = []
        for j in i:
            sample.append(j)

        x.append(sample)

    # Create labels array
    y = []
```

```
    for i in labels:
        for j in i:
            y.append(j)

    # Create decision tree
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(x, y)

    # Get the features of the sample to be predicted
    sqlPredict = 'select question1, question2, question3 from
       public.questions where pendent = true and "ID" =' + str(
       queryId)
    cur.execute(sqlPredict)
    predict = cur.fetchall()

    # Create features array of the sample to be predicted
    x = []
    for i in predict:
        sample = []
        for j in i:
            sample.append(j)
        x.append(sample)

    # Predict the label of the features sample x and save the
       predicted value on database
    prediction = 'UPDATE public.questions SET diagnosis = \'' +
       clf.predict(x)[0] + '\' where "ID" = ' + str(queryId)
    cur.execute(prediction)
    con.commit()

    return clf.predict(x)[0]
```