

MIGUEL SARRAF FERREIRA SANTUCCI

**UMA PROPOSTA PARA APRENDIZADO
COMPUTACIONAL INSPIRADA NO
APRENDIZADO E DESENVOLVIMENTO
HUMANO**

São Paulo
2021

MIGUEL SARRAF FERREIRA SANTUCCI

**UMA PROPOSTA PARA APRENDIZADO
COMPUTACIONAL INSPIRADA NO
APRENDIZADO E DESENVOLVIMENTO
HUMANO**

Trabalho apresentado à Escola Politécnica
da Universidade de São Paulo para obtenção
do Título de Engenheiro de Computação.

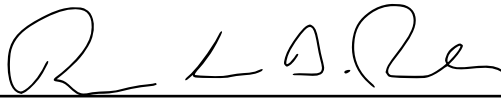
Área de Concentração:

Inteligência Artificial

Orientador:

Prof. Dr. Ricardo Luis A. Rocha

São Paulo
2021

A handwritten signature in black ink, consisting of the letters 'R', 'L', 'A.', and 'R' in a cursive style.

Prof. Dr. Ricardo Luis de Azevedo Rocha

AGRADECIMENTOS

Muitas pessoas foram importantes, direta ou indiretamente, para a realização deste projeto e da minha formação como um todo. Agradeço, inicialmente, à minha família, que sempre me apoiou na minha carreira e sem a qual não teria chegado até aqui.

Os mais sinceros agradecimentos ao Prof. Rogerio Lerner, do Instituto de Psicologia da USP, que foi de grande ajuda nos primeiros passos da modelagem do trabalho, e ao Prof. Ricardo Rocha, meu orientador, que aceitou me ajudar nessa investida para outras ciências e sem quem não teria conseguido terminar tão satisfatoriamente esta pesquisa.

Um imenso agradecimento à Fernanda Bonfatti e Sarah Cerillo, minhas amigas e pacientes professoras de psicologia e neurologia, que dedicaram horas para me ensinar essas áreas que foram tão importantes. Também agradeço a todos os outros amigos que foram criteriosos revisores do texto desta monografia.

Agradeço também ao Prof. Ricardo Tassinari, da UNESP de Marília, que foi muito prestativo durante a busca de algumas referências do trabalho. E a todo o corpo docente da Universidade de São Paulo, principalmente àqueles que me lecionaram no período da graduação e permitiram que conseguisse me formar como engenheiro.

Não posso deixar de agradecer ao Prof. Carlos Rafael Gimenes das Neves, meu amigo e primeiro professor de programação, que 8 anos atrás me apresentou o mundo da computação. Não sei o que estaria fazendo hoje se não tivesse conhecido ele.

Por fim, e em especial, agradeço a todos que me chamaram de louco.

“Truly wonderful, the mind of a child is.”

-- Mestre Yoda [1](#)

“Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child’s? ”

-- Alan Turing [2](#)

RESUMO

Este projeto é fruto de anos de observação do estado da ciência e pensamentos acerca das, e especialmente da falta de, relações entre os diversos ramos do conhecimento. Como prova de conceito, propõe-se, neste Trabalho de Conclusão de Curso (TCC), a experimentação com a miscigenação de diferentes campos do conhecimento, numa investida em direção a um idílico programa Langlands[3] expandido para toda a ciência. O projeto tem como premissa utilizar conceitos e processos de natureza psicológica, de forma a propor um modelo de representação funcional que aproxime algum recorte do processamento no cérebro humano.

O atual estado da literatura ainda mantém forte separação entre os temas pesquisados. Há diversos artigos de psicologia e neurociência que apresentam esforço no sentido de criar maior entendimento sobre a evolução dos processos psicológicos. Já pelo lado computacional, existem vastas investidas com o intuito de criar modelos classificadores e regressores com base em heurísticas aproximativas, área conhecida como Inteligência Artificial, que já apresenta bons resultados atualmente. Estes campos, é claro, tem suas teorias muito bem formadas e desenvolvidas, bem como seus autores clássicos, porém não têm projetos verdadeiramente desenvolvidos os quais se prezem a criar pontes entre esses diversos domínios, numa teoria unificada e funcional.

Algumas referências iniciais, que motivaram e serviram de diretrizes principais para o projeto, valem ser mencionadas. A teoria do desenvolvimento psicológico de Jean Piaget, em especial seu trabalho com Barbel Inhelder[4], que traduz os processos psicológicos em álgebra booleana. Os trabalhos do neurocientista Karl Friston, que propõe uma rede de neurônios como modelagem para respostas cerebrais frente a estímulos[5]. Os livros do matemático Marvin Minsky, em especial seu trabalho com Seymour Papert[6], onde ele desenvolve a teoria sobre perceptrons, base para as redes neurais atuais. Outras referências que incentivam a junção das áreas abordadas também foram consideradas, como o trabalho de David Luxton[7], que mostra diversas oportunidades de pesquisas onde a inteligência artificial pode ser aplicada para resolver problemas de outras áreas da ciência, e o estudo de Eric e Graham Taylor[8], que apresenta uma proposta de abordagem psicológica para a inteligência artificial explicada.

Palavras-Chave – inteligência artificial, psicologia, desenvolvimento, Neurociência.

ABSTRACT

This project comes as a consequence of years of observation of the current state of science and thoughts given on the lack of connection between the many knowledge fields. As a proof of concept, it is proposed, on this Final Paper, the experimentation over the integration of different research areas, in an onslaught aiming an idyllic Langlands program^[3] expanded to accommodate the whole science. The project assumes as a premise the use of psychology originated concepts and processes, in a way of proposing some model of functional representation capable of approximating some snippet of the human brain processing.

At the moment, previous literature maintains a large gap between the researched areas. There are several psychology and neuroscience articles presenting serious efforts in order to develop greater understanding of evolution of the mind's processes. On the computing side, there are large lunges with the intent of building classifying and regressing models based on approximating heuristics, a field known as Artificial Intelligence, which have already shown pretty good results. All of these departments, of course, have their own shaped and developed theories, as well as its classical authors, although there are not truly advanced projects focusing on the task of intertwining these broad domains in an unified and functional theory.

Must be mentioned here some initial references, which served as guidelines and motivators for this project. Jean Piaget's developmental psychology theory, especially his work with Barbel Inhelder^[4], which translated psychological processes into Boolean algebra. The Karl Friston's work, which proposes a neurons network as a model for cerebral responses due to stimuli^[5]. The Marvin Minsky's books, especially his work with Seymour Papert^[6], where the perceptron theory, base for the modern neural networks, is developed. A few other references which encourage the union of these many covered areas were considered, like David Luxton's work^[7], that shows a set of opportunities where artificial intelligence can be applied to solve problems in other areas, and Eric and Graham Taylor's study^[8], that presents an proposal of a psychological approach for explained artificial intelligence.

Key-Words – artificial intelligence, psychology, development, neuroscience

LISTA DE FIGURAS

1	Exemplo de aplicação de One-hot encoding para dez classes	5
2	Exemplo de aplicação da operação de convolução.	8
3	Comparação entre as diferentes funções de ativação	10
4	Primeiro modelo do desenvolvimento	18
5	Áreas funcionais do cérebro [9]	20
6	Modelo dinâmico causal de Friston [5]	22
7	Modelo proposto para desenvolvimento	25
8	Modelo do estágio I	33
9	Modelo do estágio II	33
10	Exemplo de aumento dos dados	35
11	Primeira rede desenvolvida para o estágio I	37
12	Primeira rede desenvolvida para o estágio II	38
13	Perdas para o otimizador Adam	40
14	Acuidades para o otimizador Adam	40
15	Perdas da primeira rede do estágio I	41
16	Acuidades da primeira rede do estágio I	41
17	Novo modelo do estágio I	42
18	Perda para estágio I reelaborado	43
19	Acuidades para estágio I reelaborado	43
20	Perda para estágio II	44
21	Acuidades para estágio II	44
22	Rede final desenvolvida para o estágio I	45
23	Rede final desenvolvida para o estágio II	46

24	Tela inicial do jogo	48
25	Tela de identificação do primeiro jogador	48
26	Tela preenchida pelo primeiro jogador	49
27	Tela apresentada ao segundo jogador	50
28	Matriz de confusão para o modelo do estágio I	53
29	Matriz de confusão para o modelo do estágio II	54

LISTA DE TABELAS

1	Valores de <i>acuidade</i> e <i>pontuação f1</i> para os modelos	52
2	Valores da <i>pontuação de Brier</i> para os modelos	56

SUMÁRIO

1	Introdução	1
1.1	Motivação	1
1.2	Objetivos	2
2	Aspectos conceituais	4
2.1	Dados	4
2.1.1	Representação	4
2.1.2	Aumento	5
2.2	Redes neurais	7
2.2.1	Camadas	7
2.2.2	Funções de ativação	9
2.2.3	Função de perda	11
2.2.4	Otimizadores	12
2.3	O Teste de Turing	14
2.4	Desenvolvimento humano	16
2.5	Organização do cérebro	18
2.6	Trabalhos relacionados	21
3	Metodologia	24
3.1	Modelagem	24
3.2	Tecnologias	26
3.3	Processamento dos dados	27
3.3.1	Conteúdo	27
3.3.2	Formato	28

3.3.3 Aumento	28
3.4 Métricas	29
4 Desenvolvimento	32
4.1 Simplificação da rede	32
4.2 Implementação	33
4.2.1 Tratamento de dados	33
4.2.2 Redes neurais	34
4.3 Treinamento	39
4.4 Implantação do website	47
5 Resultados	51
5.1 Avaliação das métricas	51
5.1.1 Acuidade e pontuação f1	51
5.1.2 Matriz de confusão	52
5.1.3 Pontuação de Brier modificada	55
5.2 Plataforma web	56
5.3 Implicações dos resultados	58
6 Conclusão	60
Referências	62
Apêndice A – Limites da <i>pontuação de Brier</i>	64
Apêndice B – Distribuição de probabilidades equivalente	67
Anexo A – Dígitos 4 classificados como 9	68
Anexo B – Dígitos 9 classificados como 7	69

Anexo C – Dígitos 9 classificados como 4	70
Anexo D – Dígitos 5 classificados como 3	71
Anexo E – Dígitos 3 classificados como 2	72
Anexo F – Dígitos 7 classificados como 2	73
Anexo G – Dígitos 7 classificados como 9	74
Anexo H – Dígitos 3 classificados como 5	75
Anexo I – Dígitos 8 classificados como 5	76
Anexo J – Dígitos 5 classificados como 6	77
Anexo K – Dígitos 3 classificados como 8	78
Anexo L – Dígitos 8 classificados como 3	79
Anexo M – Dígitos 5 classificados como 9	80
Anexo N – Dígitos 7 classificados como 3	81

1 INTRODUÇÃO

Durante o ano de 2021 foi desenvolvida uma prova de conceito para testar a viabilidade de um modelo computacional de simulação de processos psicológicos para tratamento de dados e classificação em grupos. Para tal, a disciplina “PSA0189 - Psicologia do desenvolvimento”, lecionada em 2020 pelo Prof. Dr. Rogerio Lerner, do Instituto de Psicologia da USP, foi fundamental para compreensão dos conceitos básicos sobre os processos do desenvolvimento e aprendizagem humana, bem como a leitura de diversos livros e artigos sobre o assunto. Esta seção apresentará as motivações e os objetivos principais do trabalho.

1.1 Motivação

John C. Calhoun Jr. disse que “É responsabilidade do engenheiro estar atento às necessidades sociais e decidir como as leis da ciência podem ser melhor adaptadas através da Engenharia a fim de cumprir essas necessidades.” [10]. Na mesma linha de raciocínio, evidencia-se que a Engenharia deve ter sob seu alcance todo o arsenal científico desenvolvido na história e ter capacidade de discriminar, dentre eles, quais são os que mais se adequam aos problemas vigentes, desenhando soluções de forma otimizada em seus contextos sociais.

Em 1956, o termo Inteligência Artificial foi cunhado oficialmente no que ficou conhecido como Conferência de Dartmouth, com a crença de que “todo aspecto da aprendizagem ou qualquer outra característica de inteligência pode ser tão precisamente descrita que uma máquina pode simulá-la”¹[11].

Por outro lado, a psicologia do desenvolvimento tem suas raízes no final do século XIX, buscando uma descrição evolutiva desse processo [12]. Até hoje, os pesquisadores desta área vêm se ocupando de descrever os processos da infância e da adolescência, de

¹Tradução livre de “every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.”

forma a modelar e entender a forma com que as informações são entendidas e incorporadas na mente humana.

Havia, entre os participantes da Conferência de Dartmouth, psicólogos e outros profissionais de áreas correlatas [11], mas a pesquisa sobre Inteligência Artificial, e em especial sobre as Redes Neurais, desconsiderou, em grande parte, as características funcionais dos fenômenos da aprendizagem, estudados por esses profissionais.

Durante os últimos anos, observou-se que essa distanciação entre a área da Inteligência Artificial e as de psicologia e neurociência, que têm paralelos óbvios entre si, está se perpetuando, senão aumentando. Acredita-se que um estudo interdisciplinar do processo e dos modelos de aprendizagem, abarcando estas diversas abordagens já estudadas pela humanidade, é uma necessidade para a continuidade do processo de evolução da ciência.

No atual estado da arte, tanto as teorias de Redes Neurais quanto as de Desenvolvimento Humano estão amplamente desenvolvidas. No mesmo grau em que algoritmos simples e funcionais já existem para as primeiras, conceitos e processos já estão solidificados para as segundas. É completamente inconcebível que duas áreas tão maduras e intrincadas e que, filosoficamente, compartilham sentidos e ligações tão claros, ainda não tenham sido extensivamente trabalhadas de forma a serem compatibilizadas. É com isso em mente, que se pretende desenvolver o presente trabalho.

1.2 Objetivos

O objetivo principal do trabalho é, partindo dos conceitos de inteligência artificial, psicologia do desenvolvimento e neurociência, propor uma representação formal para a aprendizagem do cérebro humano (pelo menos em um de seus estágios) e, com estes dispositivos lógico-formais encontrados, mostrar que é possível criar um algoritmo capaz de “aprender” simulando os processos de entendimento do mundo de uma criança. O enfoque será em simular as rotinas de algumas das fases do desenvolvimento para interpretação e classificação de entradas visuais na forma de imagens.

Essa representação formal propiciaria a definição de modelos de aprendizado de máquina que se assemelharia ao modelo de aprendizado humano em suas diversas fases. As redes neurais, que se prezam a modelar neurônios e sinapses, serão as estruturas básicas a serem utilizadas nessa busca pela representação do conhecimento. Este trabalho pode ajudar na construção de uma formulação mais adequada de métodos computacionais no sentido de compatibilizar esses universos de pesquisa.

Para atingir o que é pretendido, também será necessário o desenvolvimento de rotinas de aumento de dados, de forma a ampliar o conjunto de dados escolhido para o projeto. Esta etapa é, como será visto posteriormente, essencial para o bom desenvolvimento e teste do modelo proposto. Este objetivo, por si só, é um tema profundo e interessante, que exigirá estudo específico e desenvolvimento de rotinas próprias.

Por fim, é interessante vislumbrar a possibilidade de colocar à prova a capacidade dos modelos desenvolvidos de realizar previsões e de se assemelhar ao comportamento esperado de um ser humano real que já aprendeu a tarefa a ser estudada aqui. Esse objetivo, embora secundário, é muito interessante, pois comprovaria a tese de que as diversas áreas estudadas podem ser interligadas.

2 ASPECTOS CONCEITUAIS

Nesta seção serão apresentados os conceitos básicos fundamentais para o desenvolvimento do trabalho e justificativa dos métodos adotados. Serão expostos tanto os aspectos técnicos sobre o que já foi desenvolvido sobre dados e redes neurais e sua vasta aplicação quanto os processos e conceitos conhecidos nas teorias psicológicas e neurais.

Na primeira seção abaixo, estão cobertos os tópicos de uso de dados, na segunda de redes neurais, na terceira do Teste de Turing, na quarta de psicologia, na quinta de neurologia e na sexta estão cobertos trabalhos relacionados.

2.1 Dados

Na abordagem com redes neurais, como a aqui apresentada, o uso de bons dados é extremamente importante e a forma e quantidade em que estes são apresentados são determinantes para obtenção de bons resultados. Os dados que serão utilizados no projeto são de imagens separadas em 10 classes e, para bom aproveitamento destes, existem formas de tratamento específicas. Nesta seção serão apresentadas as formas de representação e de aumento no formato pretendido.

2.1.1 Representação

O formato com que os dados serão apresentados para a rede é fator determinante no seu desempenho, já que são eles que forneceram os valores com os quais o modelo calibra seus parâmetros internos. Como os dados de imagem são bastante simples, sua representação original da forma de matriz é suficiente, mas a informação das classes de cada imagem pode ser representada de diversas formas, segue uma discussão sobre esse formato.

A forma mais simples e imediata de representar a classe da imagem é através de um dígito identificador. Este formato é prático pela sua simplicidade, correspondendo à uma

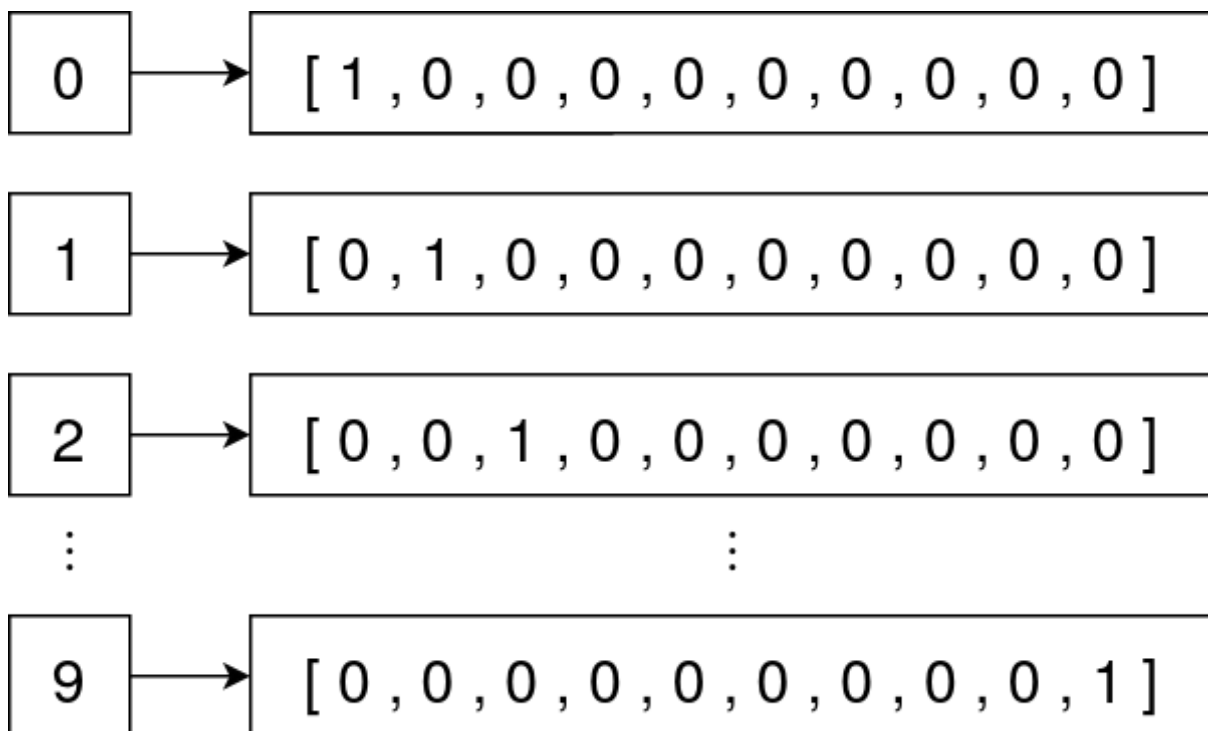


Figura 1: Exemplo de aplicação de One-hot encoding para dez classes

saída única do modelo de predição. Por outro lado, a relação entre as classes fica menos evidente, já que estão todas num único espaço contínuo.

Outra forma é a chamada de *One-hot encoding*, na qual a classe é transformada em um vetor de zeros com apenas um valor um na posição da classe referida [13], como exemplificado na figura 1. Esta mudança faz com que a classe se torne uma distribuição de probabilidade com 100% de certeza da classe à qual pertence o dado. A saída da rede, então, será uma outra distribuição de probabilidade que pode ser comparada à da classe.

2.1.2 Aumento

Nesta seção será discutida uma das técnicas que podem ser utilizadas para resolver o problema do sobreajuste, comum durante o desenvolvimento de redes neurais. O sobreajuste ocorre quando a rede “aprende” tanto com os dados de entrada que, ao invés de obter suas características para classificar, ela passa a identificar perfeitamente os dados de entrada nas suas classes, funcionando quase como uma tabela [14]. Esse fenômeno causa perda de generalidade no modelo, tornando-o inutilizável para uso prático e apresentando altos valores da função de perda até mesmo para os dados de treino [14]. Por essas razões, é crucial para o posterior processo de construção das redes neurais, que seja desenvolvido um bom método de aumento.

O método mostrado para evitar o sobreajuste é o processo conhecido como aumento de dados¹ sobre o conjunto de dados de entrada da rede^[14]. As técnicas de aumento de dados encontradas na literatura são, majoritariamente, voltadas para aplicação sobre imagens, mas quase todas as técnicas são adaptáveis para outros tipos de dados. A seguir estão listadas algumas das técnicas de aumento existentes.

As técnicas de aumento de dados podem ser classificadas em dois grandes grupos: manipulações básicas e abordagem por aprendizagem profunda^[14]. As primeiras aplicam tratamentos sobre os dados de forma a alterar sutilmente algumas qualidades, já as segundas utilizam redes neurais para criar novas imagens, semelhantes às originais e mantendo sua classificação^[14].

Entre as técnicas de manipulação básica, existem aquelas de uso específico em imagens, que realizam mudanças sobre as cores (alterações na escala hue, na saturação, etc.), transformações geométricas (cortes, rotações, etc.) ou recortam pedaços de diferentes dados de uma mesma classe e os misturam^[14]. Outros métodos são generalizáveis, como a aplicação de filtros pré-criados sobre os dados de entrada, o apagamento aleatório de alguns pedaços dos dados de entrada ou a inserção de ruído nos dados^[14].

São três as técnicas envolvendo aprendizagem profunda, treino adversário², baseado em GAN's (Generative Adversarial Network) e transferidor neural de estilos³. Na técnica de treino adversário são criadas duas redes neurais distintas de forma que uma tente criar conjuntos de dados, partindo dos originais, que causem classificação errônea na outra rede^[14]. A técnica de aumento baseada em GAN é uma variação da técnica de treino adversário, onde uma rede cria novos dados e a outra recebe as imagens originais e geradas e deve distinguir quais delas são reais e quais são falsas^[14]. O último tipo é o transferidor neural de estilos, que acompanha as sequências de camadas de uma *CNN*, importando o estilo de desenho de uma imagem para outra^[14].

Isto posto, é evidente que será utilizada ao menos uma técnica de aumento de dados para que os dados originais sejam ampliados de forma conveniente para o treinamento da rede. Como será visto na seção ^{3.3}, o dataset escolhido é muito simples e já é relativamente extenso, assim que não será necessário nenhum tipo de processo muito avançado para o aumento dele. Dessa forma, pretende-se utilizar apenas técnicas de manipulação básica no processo.

¹Do inglês data augmentation

²Do inglês Adversarial training

³Do inglês Neural Style Transfer

2.2 Redes neurais

As redes neurais são uma tentativa muito convincente de representar neurônios humanos e suas interações no intuito de encontrar heurísticas de solução para problemas. Redes neurais se baseiam em um modelo simplificado de neurônio, que realiza uma função simples sobre uma ou mais entradas para gerar uma saída, cujo erro determina a variação de seus parâmetros internos [6, 15].

O processo de treinamento de uma rede neural é conceitualmente simples. Alguns dos dados de entrada são inseridos na rede, obtendo-se alguns dados de saída, que são comparados com os dados de saída esperados através de uma função de perda, utilizada para atualizar os parâmetros internos de cada neurônio [15].

Esse processo de melhora da rede é repetido diversas vezes, sobre conjuntos de dados distintos, que recebe o nome de *lote*⁴. O conjunto de *lotes* que completam a utilização de todos os dados disponíveis é chamado de *época*⁵. Um treinamento usual passa por uma sequência de *épocas* [15].

A seguir, alguns tipos de neurônios e arquiteturas comuns serão citados, bem como métricas e métodos para obtenção do erro da saída da rede.

2.2.1 Camadas

Um modelo de rede neural é composto por uma sequência de camadas pelas quais os dados de entrada são passados para se chegar à saída [15]. Cada camada, por sua vez, pode realizar diferentes tipos de funções a depender do tipo de neurônios dos quais ela é composta [15].

Entre essas camadas também é possível adicionar etapas intermediárias para reestruturar os dados, como concatenação ou mudança no formato, ou aplicar funções simples sobre eles, como as funções de ativação, que serão abordadas na seção [2.2.2]. Segue uma pequena discussão sobre os tipos de neurônios que podem compor as camadas.

O modelo mais simples de neurônio utilizado é o perceptron, que realiza uma operação de soma ponderada das suas entradas, utilizando os parâmetros internos do neurônio como pesos, este resultado é oferecido como saída do perceptron [6, 15]. Um perceptron isolado tem a capacidade de aprender a separar os dados de entrada da forma mais conveniente

⁴Do inglês batch

⁵Do inglês epoch

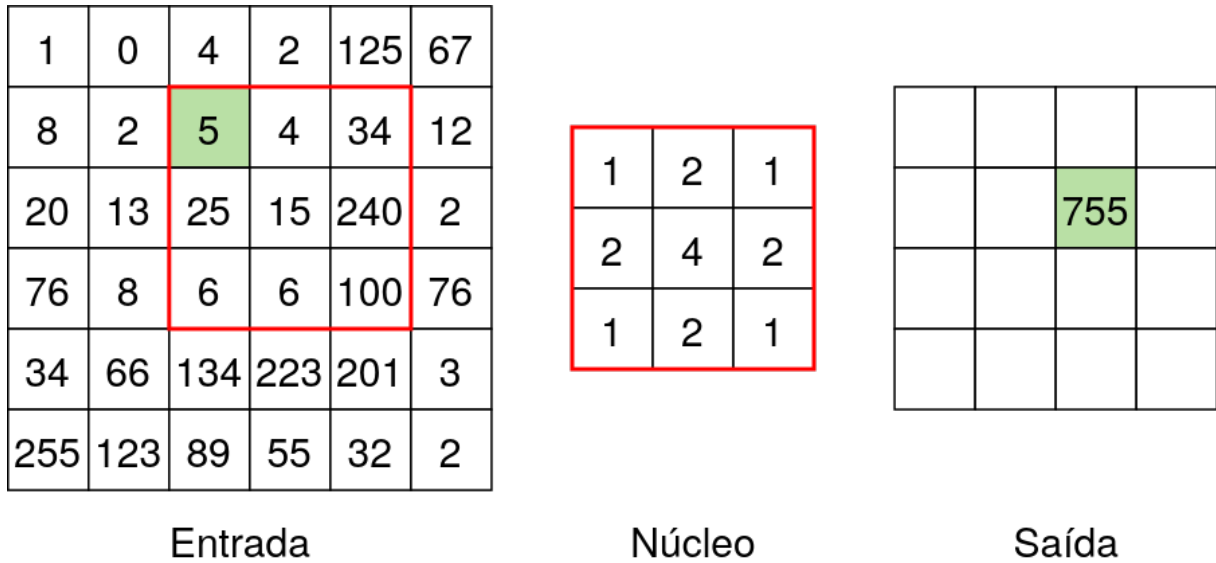


Figura 2: Exemplo de aplicação da operação de convolução.

através de um hiperplano com mesma dimensão da quantidade de neurônios^[6].

Camadas com este tipo de neurônios são utilizadas para gerar associações entre todos os dados de entrada, extraindo tantas características gerais quantos forem os neurônios na camada^[15]. Quando conectam-se vários perceptrons, forma-se uma rede chamada de Multilayer Perceptron (*MLP*), muito útil para criar inferências de conjuntos de características originais^[15].

Outro tipo de neurônio é o convolucional, no qual a entrada, matricial, é percorrida por uma matriz de parâmetros internos, chamada *núcleo*^[6], menor que a matriz de entrada, aplicando um produto de Hadamard na área sobreposta e somando os elementos da matriz resultado e, assim, gerando cada um dos elementos da matriz de saída do neurônio, como exemplificado na figura^{[27][15]}. Um neurônio convolucional tem a propriedade de conseguir isolar padrões locais tão grandes quanto o *núcleo* das matrizes de entrada.

Camadas convolucionais são utilizadas para identificar padrões relevantes que se apresentem nos dados de entrada. Quando aglutinados em uma rede, esta recebe o nome de Convolutional Neural Network (*CNN*) e são utilizadas para redução de entradas complexas, como imagens, a um tamanho reduzido de informações, concentrando características intrínsecas da entrada^[15].

Um terceiro tipo ainda, é o neurônio recorrente, muito parecido com o perceptron,

⁶Do inglês kernel

⁷O exemplo está feito para apenas uma posição do *núcleo*, deslocado uma unidade para baixo e duas para a direita, mas na rede real a operação é realizada para todas as posições possíveis, preenchendo a matriz de saída.

porém admite também uma entrada vinda da execução anterior, que é constantemente atualizada [15]. Esta componente interna da entrada transmite informações de entradas passadas, criando uma componente de memória no neurônio [15]. A principal funcionalidade adquirida com a inserção desse componente de memória é a possibilidade de gerar dependência entre diferentes entradas do neurônio.

Uma camada recorrente é caracterizada exatamente por essa propriedade, permitindo uma melhor análise de séries temporais. Um conjunto de neurônios deste tipo é conhecido como Recurrent Neural Network (*RNN*) e é utilizada sobre conjuntos de dados onde uma entrada é fortemente dependente de contexto, como sinais de áudio [15].

Vale ressaltar que mais de um tipo de neurônio pode ser misturado na mesma rede, de forma a obter resultados mais elaborados que combinam as capacidades dos tipos de rede. Cada cenário exige o desenvolvimento de redes específicas.

2.2.2 Funções de ativação

Fazendo-se uso apenas dos neurônios do tipo perceptron, apenas funções lineares estariam presentes na composição das camadas, de forma que elas poderiam ser reduzidas a um único neurônio. Esse fenômeno é indesejado, pois significaria que qualquer rede consegue resolver os mesmos problemas que um perceptron isolado [15-17].

Para remediar essa simples linearidade, utilizam-se funções de ativação, que são funções não-lineares aplicadas na saída de cada neurônio, de forma que não se possa mais simplificar as redes [15-17]. A seguir está posta uma breve discussão sobre os tipos de função de ativação mais comuns.

A função de ativação mais simples é a função ReLU, ou unidade retificada linear⁸, que é bastante útil para forçar valores da saída para zero [16]. Essa possibilidade permite que o neurônio sinalize quando determinada propriedade é ou não atendida, funcionando como um interruptor [16]. Esse tipo de função de ativação é muito utilizada, já que as outras funções não conseguem realmente atingir o valor nulo na saída, como será visto adiante, além de outras boas propriedades.

$$ReLU(z) = \max(0, z) \quad (2.1)$$

Outras duas funções também muito utilizadas são as funções sigmóide e tangente

⁸Do inglês Rectified Linear Unit

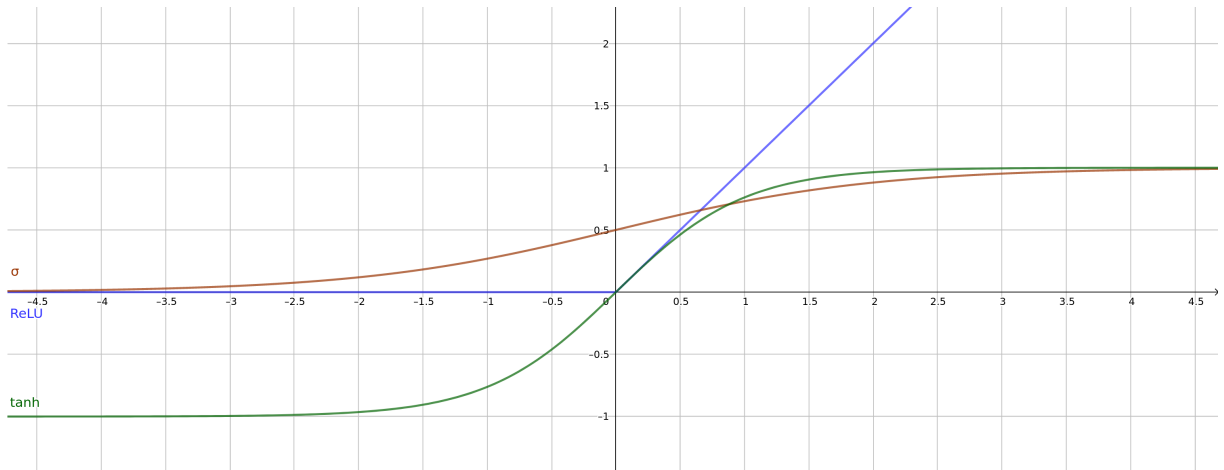


Figura 3: Comparação entre as diferentes funções de ativação

hiperbólica que alteram o domínio dos valores de entrada [16]. Ambas as funções recebem valores do domínio dos números reais, porém a sigmoide retorna valores no domínio $]0, 1[$ e a tangente hiperbólica no domínio $] - 1, 1[$ [16]. A principal característica dessas funções é realizar essa transposição de forma muito suave [16]. Essas funções de ativação são muito úteis para limitar as saídas das redes e prevenir estouro nos dados sendo tratados [16].

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.2)$$

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.3)$$

Na figura 3 podem ser vistas as três funções de ativação discutidas, com suas propriedades de acordo como foram apresentadas.

A última função de ativação dentre essas mais comuns é a conhecida como softmax, que normaliza a sequência de dados de saída de uma camada [16]. Através de exponenciais, cada saída é comparada com as demais da camada, gerando um novo conjunto de valores para a saída [16]. Esse tratamento força os dados a se comportarem como uma função massa de probabilidades, podendo ser utilizado para comparação com dados de saída codificados em *One-hot encoding* [16].

$$\text{softmax}_i(z^L) = \frac{e^{z_i^L}}{\sum_{j=1}^C e^{z_j^L}} \quad (2.4)$$

2.2.3 Função de perda

Como explicado, depois dos dados de um *lote* serem processados, as saídas da rede são comparadas com as saídas esperadas, obtendo-se um valor de erro que é propagado entre as camadas e os diferentes neurônios, atualizando seus parâmetros internos [15]. Esse algoritmo é chamado de retropropagação⁹[15].

Para essa atualização, é computado o gradiente da função de perda e a alteração dos parâmetros se dá no sentido oposto a esse gradiente, ou seja, diminuindo o valor da função de perda e aproximando a saída do esperado [15]. O erro total utilizado na retropropagação é calculado como a esperança dos erros individuais de cada entrada [16].

$$J_{EL} = \mathbb{E}(J(W, b; o, y)) = \frac{1}{M} \sum_{m=1}^M J(W, b; o^m, y^m) \quad (2.5)$$

Uma função de perda muito utilizada é o *erro quadrático médio*¹⁰ (*EQM*), que calcula o erro como sendo o quadrado da diferença entre os valores preditos e esperados [15, 16]. Esta função é extremamente genérica e seu uso produz a aproximação dos vetores de saída e esperado, podendo ser aplicada em praticamente todos os casos.

Devido a essa característica de sempre aproximar em absoluto os vetores resultado, este erro é usualmente utilizado para tarefas de regressão, mas também poderia ser utilizado em uma tarefa de classificação como a proposta pelo trabalho [16].

$$J_{MSE}(W, b; o, y) = \frac{1}{2} \|v^L - y\|^2 \quad (2.6)$$

Outra função de perda que pode ser utilizada é a *divergência de Kullback-Leibler*¹¹ (*DKL*), que é um método utilizado para comparar duas curvas de densidade de probabilidade [17]. Esta divergência calcula a distância relativa entre as curvas de probabilidade ponderadas pela função de referência [17].

Na forma discreta, a *DKL* pode ser utilizada com dados em *One-hot encoding*, de forma a preferenciar saídas da rede que se aproximem mais de distribuições ideais, ou seja, com total certeza do resultado obtido. Para utilizar essa função, deve-se garantir que a saída da rede seja uma função massa de probabilidade.

⁹Do inglês *backpropagation*

¹⁰Do inglês *mean squared error*

¹¹Do inglês *Kullback-Leibler Divergence*

$$J_{KL}(W, b; o, y) = KL(p||q) = - \int_{-\infty}^{\infty} p(x) \ln\left(\frac{q(x)}{p(x)}\right) dx = \sum_{i=1}^C y_i \ln\left(\frac{v_i^L}{y_i}\right) \quad (2.7)$$

12

Por fim, outra função de perda utilizada é a *entropia cruzada*¹³ (*EC*), que é uma simplificação da *DKL* que simplifica a computação dos valores¹⁶. A diferença entre a *DKL* e a *EC* é que a última avalia a massa de probabilidades isoladamente e não em relação ao objetivo.

Pela mesma razão que o caso anterior, essa função de perda tem muitas vantagens ao ser utilizada junto com a técnica de *One-hot encoding*. Computacionalmente, a *EC* é preferencial à *DKL* pois evita o problema da divisão por zero.

$$J_{CE}(W, b; o, y) = - \sum_{i=1}^C y_i \log v_i^L \quad (2.8)$$

A *DKL* e a *EC* são muito similares em forma e significado e o uso de ambas beneficia situações em que existe alta probabilidade nas classes corretas. Sabe-se também que minimizar a *entropia cruzada* equivale a minimizar a *divergência de Kullback-Leibler*¹⁶.

Nas fórmulas apresentadas nessa seção, a variável W representa os pesos das arestas das redes, b o viés de cada nó delas, o cada uma das entrada, y as saídas esperadas, v^L as saídas obtidas pela rede, M o número total de observações, C o número total de classes, o operador \mathbb{E} representa a esperança de um conjunto de dados e as funções $p(x)$ e $q(x)$ são funções densidade de probabilidade.

2.2.4 Otimizadores

Depois de calculado o erro em uma rede neural, inicia-se o processo de retropropagação desse erro, quando são atualizados os parâmetros da rede, de forma a minimizar o valor do erro da rede, como já foi discutido¹⁵⁻¹⁸. Todo o processo de obter o gradiente da função de perda e aplicar retroativamente seu oposto nas camadas da rede, encaminhando a solução para um estado de menor valor da função de perda, é o algoritmo conhecido como *gradiente descendente*¹⁴⁻¹⁵⁻¹⁸.

¹²Fórmula adaptada e expandida de ¹⁷ de acordo com notas de aula da disciplina “PSI3501 - Processamento de Voz e Aprendizagem de Máquina”, lecionada no 2º semestre de 2021, e de forma a se adequar à notação empregada em ¹⁶

¹³Do inglês cross-entropy

¹⁴Do inglês gradient descent

A aplicação deste algoritmo puramente, porém, leva a algumas dificuldades durante a aprendizagem da rede, que pode ser limitada^[18]. Quando a perda para o problema analisado apresentar mínimos locais, platôs ou regiões planas ou muito íngremes e algumas outras particularidades, o gradiente descendente se torna improdutivo^[18]. Estas condições topológicas do problema fazem com que o modelo se estabilize fora do mínimo global^[18].

Platôs, regiões planas e mínimos locais, estes últimos que podem aumentar exponencialmente com a dimensão do problema^[19], fazem com que o algoritmo acredite ter encontrado a solução ideal e estagnar^[18]. Já regiões muito íngremes fazem com que a variação da solução seja muito grande, eventualmente fazendo a rede se afastar da solução ideal^[18].

Para mitigar estes problemas são utilizados mecanismos conhecidos como otimizadores, que implementam variações no *gradiente descendente* no sentido de contornar esses problemas^[18]. Diversos algoritmos de otimizadores são propostos na literatura, com maior ou menor complexidade e efetividade, alguns deles estão expostos a seguir.

O algoritmo *gradiente descendente estocástico* (*SGD*^[15]) altera o algoritmo original no intuito de inserir pequenas variações que impedem o gradiente de se tornar excessivamente pequeno ou nulo^[18]. Esse pequeno ruído é inserido utilizando uma amostragem aleatória de algumas entradas do conjunto de dados de treino a cada iteração para gerar o gradiente^[18].

O *SGD* é muito utilizado por sua simplicidade e por garantir bons resultados e convergência, conseguindo contornar os problemas mencionados com elegância. Essa técnica, porém, nem sempre traz bons resultados e pode se mostrar bastante lenta durante o processo de treino^[18].

O algoritmo do *momento*^[16] é uma alternativa para acelerar aprendizado nos casos de grandes curvaturas e gradientes pequenos ou ruidosos^[18]. A ideia fundamental aplicada por essa técnica é de manter um registro dos gradientes já aplicados de forma que o novo gradiente se mantenha no mesmo sentido geral^[18].

Uma variação do otimizador por *momento* é o *momento de Nesterov*^[17], que faz uma pequena alteração nos parâmetros do modelo antes de calcular o gradiente a ser aplicado sobre a rede montada^[18]. Essa mínima alteração faz com que o processo de treino tenha convergência mais rápida que se arquitetado de outra forma^[18].

¹⁵Do inglês stochastic gradient descent

¹⁶Do inglês momentum

¹⁷Do inglês Nesterov momentum

Estes dois otimizadores fazem uso do histórico do gradiente para manter a atualização dos parâmetros seguindo na mesma direção [18]. Esse mecanismo é útil para sobre gradientes ruidosos, pois as muitas variações provenientes do ruído podem fazer com que a rede oscile muito ao passar pelas etapas de retropropagação.

O algoritmo *AdaGrad*, por sua vez, aplica uma correção sobre cada gradiente calculado, de forma a reduzir as variações numa rede quanto mais ela for treinada [18]. O conceito principal é aplicar sobre o novo gradiente um fator de redução proporcional ao inverso da soma dos quadrados dos gradientes anteriores, forçando uma diminuição no valor absoluto da variação dos parâmetros [18].

Uma variante do *AdaGrad*, o *RMSProp*, aplica um fator temporal sobre o fator aplicado no gradiente, de forma que os gradientes mais antigos causam menor impacto na redução do gradiente sendo calculado [18]. Essa pequena adição ao modelo faz com que a variação na mudança da rede dependa mais de eventos mais recentes da aprendizagem do que dos mais antigos [18].

O *AdaGrad* se mostra proveitoso em alguns modelos apenas, funcionando para melhorar o ganho de alguns cenários específicos. Por outro lado, o *RMSProp* é efetivo em muitos outros casos e bastante prático devido a essa propriedade de esquecimento [18].

Por fim, o otimizador *Adam* é uma combinação entre os algoritmos *RMSProp* e *momento* e, embora sua motivação teórica não seja muito clara, combina as vantagens desses dois algoritmos, como explicadas acima [18]. Este otimizador se provou bastante robusto, com necessidade de cuidado nas escolhas de seus parâmetros internos.

2.3 O Teste de Turing

O Teste de Turing é um artifício conceitual que visa determinar, através de um teste simples, se uma determinada máquina se comporta de maneira inteligente. Este teste, proposto inicialmente em 1950, parte do princípio de comparar respostas dadas pela máquina em análise com as dadas por um ser humano. Nesta seção, será brevemente explicado como funciona um Teste de Turing.

Originalmente, este mecanismo foi chamado de Jogo da Imitação¹⁸ e foi rebatizado para Teste de Turing em homenagem a seu idealizador, Alan Turing [2]. No artigo, Turing propunha uma discussão sobre a capacidade de uma máquina pensar e explicava o Jogo da Imitação como método de fazer essa conferência [2].

¹⁸Do inglês Imitation Game

O Jogo da Imitação parte de um jogo de três pessoas, uma mulher A, um homem B e um terceiro jogador (homem ou mulher) C [2]. No jogo, C, que não consegue ver nem A nem B e os chama de X e Y, deve tentar adivinhar se A é X e B é Y ou se A é Y e B é X fazendo uso de perguntas não diretas [2]. Porém, A tem o objetivo de enganar C e deve dar suas respostas de forma a induzir C a pensar em ela é B, que tem o objetivo contrário, ou seja, ajudar C a chegar na resposta correta [2]. Ou seja, os dois competidores, A e B, têm o objetivo de convencer C de que são homens. Entende-se que, neste cenário, A deve dar suas respostas da forma mais parecida possível com B respondendo [2].

O Jogo da Imitação de verdade acontece quando se substitui a jogadora A pela máquina que se deseja testar [2]. Então a máquina deve fornecer suas respostas de forma a confundir C e fazer com que este pense que a máquina se trata do jogador B [2].

Dessa forma, vê-se que a máquina é aprovada no Jogo da Imitação quando ela se torna capaz de fornecer respostas a problemas da mesma forma que o jogador B e provar à C que ela é um homem [2]. Como todo ser humano é igualmente inteligente, o problema se reduz à necessidade da máquina conseguir responder às perguntas como qualquer pessoa.

Num sentido mais amplo, o Teste de Turing simboliza qualquer cenário em que coloque-se uma máquina e um ser humano para competir em problemas iguais ou análogos e um outro humano deve tentar descobrir qual é a máquina [20]. Se o avaliador (o equivalente de C) não for capaz de descobrir qual competidor é a máquina ou se achar que consegue apontar o competidor errado, diz-se que a máquina passou no Teste de Turing [20].

O Teste de Turing apresenta um dilema filosófico profundo, uma vez que coloca regras claras e atingíveis para se dizer que uma máquina pode agir de forma inteligente [2, 20]. A inteligência é uma característica usualmente relacionada aos humanos e não existem conclusões definitivas sobre a sua natureza. Assim, a possibilidade de identificar um fator não-orgânico como inteligente é passível de grande estranhamento.

Como Turing ressalta em seu artigo, muitos problemas ainda rondam a inteligência humana, como a tomada de consciência entre outros, mas o comportamento inteligente, que é proposto ser analisado no Teste, pode abstrair este tipo de questão e portanto mantém a validade do Teste de Turing e o dilema da capacidade das máquina de agir como seres humanos [2, 20].

Embora seja muito interessante e simbolize uma hipótese válida para o significado de inteligência, o Teste de Turing tem caráter puramente comportamental e, assim, permite a construção de críticas. O fato de o Teste não analisar a consciência por trás das ações, bem como as perguntas mostradas aos competidores não buscarem características profundas

da humanidade, são argumentos que desafiam a validade dele.

Um inconsistência do Teste de Turing pode ser vista ao se analisar o argumento do quarto Chinês¹⁹ de John Searle, no qual diz-se que, se um humano não falante de Chinês, de posse de um livro de regras de construção de respostas nesta língua, receber um símbolo de entrada, ele será capaz de construir uma cadeia de saída que parecerá natural^[21].

Este operador dos símbolos, então, nada difere de uma máquina realizando a mesma tarefa e não pode ser considerado inteligente, uma vez que não entende o significado do que está escrevendo^[21]. Se este mesmo desafio fosse aplicado no formato de Teste de Turing, um competidor, humano ou máquina, de posse do suposto livro de regras, é considerado inteligente.

Outra crítica possível ao Teste de Turing se refere à qualidade dos problemas apresentados aos competidores. Sartre aponta que problemas não relacionados com traços humanos, que essencialmente seriam as emoções, qualquer teste não teria validade para analisar se um competidor é inteligente ou não^[22].

Nesta visão, para ter a capacidade de analisar a humanidade, o teste proposto teria que indicar se o competidor foi capaz de invocar alguma emoção ou não^[22]. Como até mesmo crianças pequenas, desprovidas de inteligência, apresentam emoções básicas, como raiva e felicidade, devem ser analisadas emoções mais sofisticadas como a angústia (originalmente proposta por Sartre) ou empatia (como presente no teste de Voigt-Kampff da série *Blade Runner*)^[22].

2.4 Desenvolvimento humano

O desenvolvimento humano é a disciplina da psicologia que estuda a forma com que o cérebro evolui e desenvolve suas estruturas internas durante a infância e a adolescência até que esteja plenamente constituído em todas as suas facetas.

Existem muitos autores que contribuíram para essa área ao longo da história, um dos mais famosos deles é Jean Piaget, cujas ideias sobre o pensamento lógico-formal são base para este trabalho e estão resumidas a seguir.

De acordo com a teoria desenvolvida por Piaget e Inhelder, o desenvolvimento do pensamento lógico-formal no ser humano pode ser dividido em três etapas: pré-operatório, operacional concreto e hipotético-dedutivo^[4].

¹⁹Do inglês Chinese room argument

Cada uma das etapas apresenta um caráter distinto da capacidade de representação de ideias e mostram crescimento da capacidade cognitiva, desde a total falta de representação até a formação de conceitos abstratos e de fenômenos de causalidade e a capacidade de inferência [4].

Durante o estágio pré-operatório (estágio I), que dura do nascimento até sete ou oito anos de idade, quase não existe representação do mundo exterior na cabeça da criança [4]. Durante este estágio, todas as atitudes tomadas pela criança estão muito relacionadas ao estado emocional do momento [4].

Todas as ações e entendimentos no estágio I estão condicionadas às emoções e não existe nenhuma forma de dissociação entre o que é observado do meio externo e o que é resultado de alguma ação do próprio sujeito [4]. Dessa forma, é impossível que uma criança nesse estágio seja capaz de realizar qualquer tipo de pensamento formal sobre o mundo, os objetos e as ações vivenciadas.

Já no estágio operacional concreto (estágio II), que dura do final do estágio I até onze ou doze anos de idade, as primeiras noções de representação começam a surgir na forma de classificações [4]. Nesse estágio o mundo externo se torna elemento de observação para a criança, que busca as diferenças entre os objetos e padrões por ela observáveis [4].

Nesta fase, as informações sensoriais adquiridas passam a ser armazenadas e classificadas de acordo com suas características pertinentes sobre um grupo de informações, sendo esta classificação mais refinada quanto mais avançada estiver a criança dentro do estágio [4]. Com isso, então, aparece a capacidade de separar e discernir os diversos elementos que são encontrados.

Por fim, no estágio hipotético-dedutivo (estágio III), também conhecido como operacional formal, que se alonga do término do estágio anterior até os quatorze ou quinze anos, é quando acontecem os maiores desenvolvimentos do ponto de vista da formalização do pensamento [4]. Durante este estágio, o mundo se torna fonte de conhecimento e a criança aprende a formular hipóteses e experimentos para testá-las.

É durante esse período que o sujeito se torna capaz de associar informações adquiridas no intuito de desenvolver regras gerais de funcionamento de determinado sistema que está sob análise [4]. Portanto, durante este terceiro estágio, se desenvolve a capacidade de criação de formulações da realidade usando as entradas externas como fonte de testes.

Estes dois pesquisadores também apontam a existência de sub-estágios dentro dos estágios expostos e os nomearam adicionando os sufixos A e B aos estágios [4]. Esses sub-

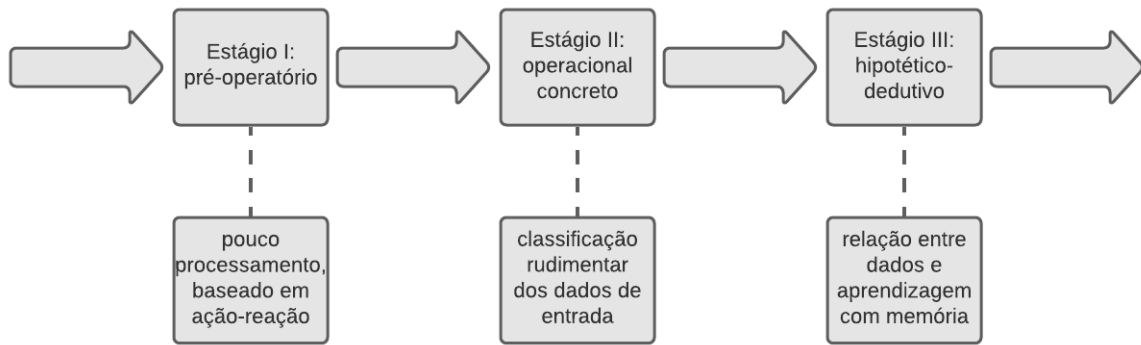


Figura 4: Primeiro modelo do desenvolvimento

estágios apresentam evoluções intermediárias dos processos apresentados, evidenciando que a aprendizagem é um processo contínuo e as etapas são apenas um mecanismo para auxiliar o estudo [4]. Para as finalidades deste estudo, serão ignorados os sub-estágios.

Tomando por base os processos considerados nos três estágios de Piaget e Inhelder, apresenta-se na figura 4 um esquema simplificado dos processos abordados.

Vale ressaltar que Piaget, em outros estudos, separou o desenvolvimento em quatro etapas, ao invés das três apresentadas, incluindo, antes de todas as mencionadas, o estágio sensorio-motor, durando do nascimento aos 2 anos de idade, aproximadamente, e apenas se caracteriza por reações espontâneas á estímulos [23]. Esta etapa será ignorada doravante devido à sua insignificância no contexto do desenvolvimento do pensamento formal.

2.5 Organização do cérebro

A modelagem de inteligência artificial que se pretende fazer do processo de desenvolvimento será, a priori, baseada na organização neurológica da mente humana, de forma que o modelo possa representar as estruturas do cérebro de forma computacional. Para tal, é importante entender, em linhas gerais, como funciona o cérebro humano, assim que uma breve discussão nesse sentido está posta a seguir.

O cérebro humano é um órgão extremamente complexo na sua organização e funcionamento, sendo merecedor de um estudo isolado e profundo, mas que não pôde ser empregado devido ao tempo disponível para execução do projeto. É importante ressaltar que ainda existem muitas lacunas no conhecimento do funcionamento do cérebro, sendo impossível uma análise completa.

O cérebro é a parte principal do sistema nervoso, sendo responsável pelo processa-

mento das informações captadas pelos órgãos sensores [9]. Além de muitas outras funções, o cérebro é responsável por armazenar todo o conhecimento adquirido, realizar todas as tomadas de decisão e executar todas as ações, salvo as respostas impulsivas, assim que é parte fundamental de tudo que acontece diariamente [9].

A grande diversidade de funções exercidas pelo cérebro, característica determinante para a evolução do ser humano, foi propiciada por uma alta diferenciação das células pelo tecido cerebral, resultando em diversas áreas identificáveis [9]. Essas diversas áreas dividem entre si as várias funcionalidades que o cérebro deve realizar [9].

O cérebro pode ser dividido em cinco macrorregiões, quatro lóbulos, separados por fissuras [20] e a Ínsula, uma região interna [9]. Em 1909, o cientista K. Brodmann publicou um livro no qual dividia o cérebro em 45 áreas distintas de acordo com o arranjo de células nelas presentes [24].

Outras organizações, mais funcionais, foram elaboradas desde então, e são facilmente mapeadas nas *áreas de Brodmann* (BA [21]), como ficaram conhecidas essas 45 regiões por ele identificadas [9].

As *áreas de Brodmann* são separadas em três categorias: áreas sensoras, áreas motoras e áreas associativas, as primeiras recebem os sinais sensitivos do corpo e as tratam, as segundas controlam os movimentos do corpo e as terceiras são responsáveis pelo pensamento, ou seja, memória, julgamento, emoções, personalidade, raciocínio, etc. [9]. A figura 5 apresenta estas áreas e suas posições no cérebro.

As áreas associativas são as mais presentes por todo o cérebro, cada uma das suas subáreas é responsável por uma parte diferente do pensamento e das associações e abrange uma ou algumas BA's [9]. Já as áreas sensoriais são divididas de acordo com a fonte de estímulo, cada um dos cinco sentidos [9]. E as áreas motoras, ocupando apenas 3 das BA's, tem uma sub-área específica para controle da fala e outra para o restante das funções motoras [9].

Como o foco deste projeto será classificação de imagens, será dada atenção especial para as *áreas de Brodmann* que se relacionam a este contexto. Nessa linha de pensamento, entende-se que serão abordadas principalmente áreas associativas e, em menor escala, áreas sensoras. Na seção 2.6, a estrutura a ser seguida e as áreas envolvidas serão melhor estruturadas.

²⁰Fissuras são reentrâncias muito profundas formadas nas curvas (gyros) do cérebro

²¹Do inglês Brodmann area

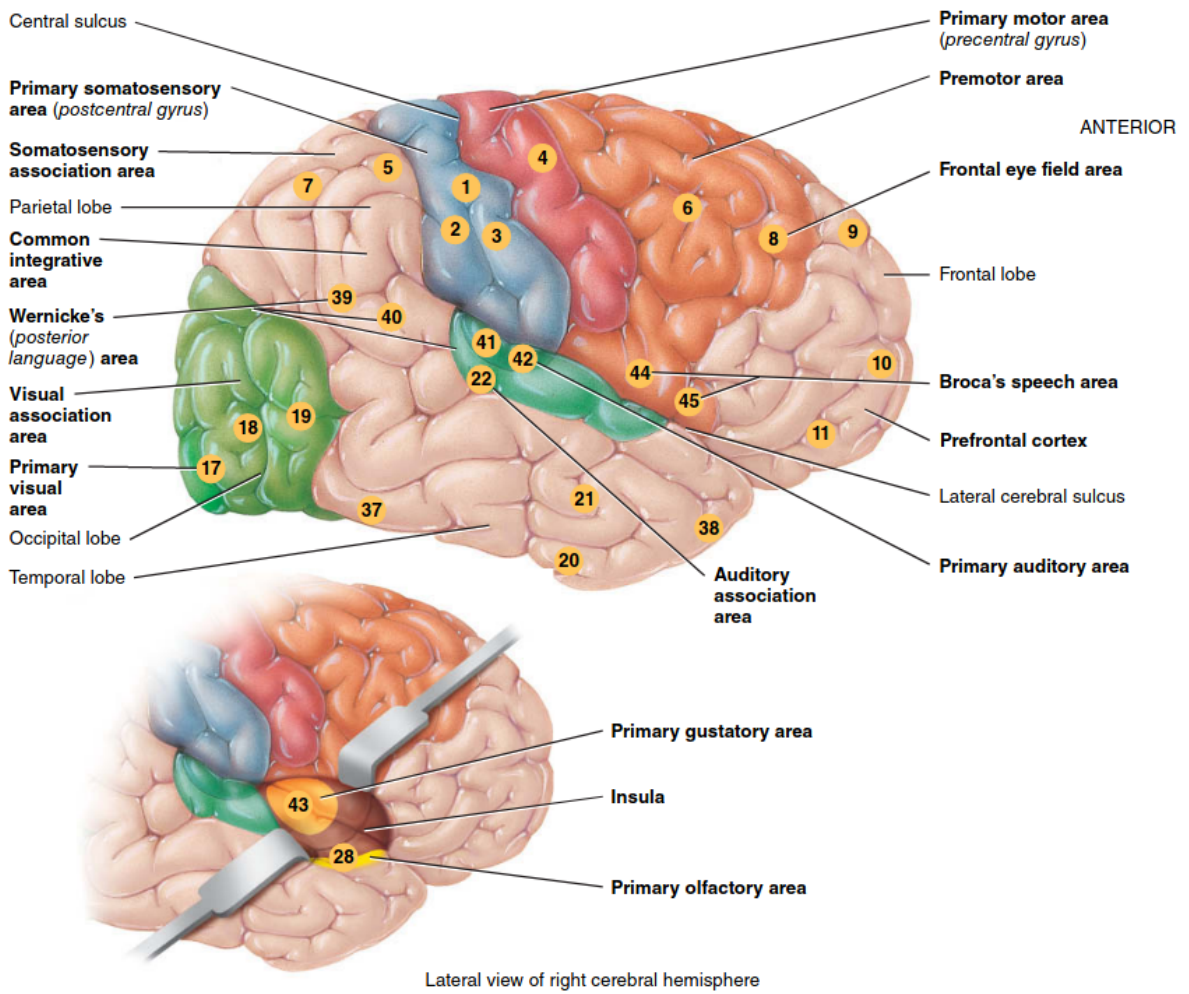


Figura 5: Áreas funcionais do cérebro 9

2.6 Trabalhos relacionados

Existe muito conteúdo relacionado ao tema pesquisado, mas todos os autores encontrados que abordam essas questões o fazem em discussões pertinentes em suas respectivas áreas, mas que são tangentes ao proposto. Praticamente todas as referências abordam o aprendizado humano de um ponto de vista específico, não buscando a interligação das áreas.

Um dos poucos autores que se aproxima da conexão pretendida é Gary Drescher em seu artigo de 1986 [25], mas ele se limita a modelar resultados de processos, sem despende esforços em modelar o processamento da informação como forma de inteligência.

Em um artigo de 2003, o neurocientista Karl Friston descreveu um modelo que busca unificar e prever processamento de dados pelo cérebro [5]. Esse modelo leva em consideração as relações entre as algumas regiões do cérebro, bem como os sinais que se transmitem de uma para outra [5].

Na figura 6 está apresentado um modelo proposto por Friston, onde cada estado representa uma região cerebral e y é uma função de saída que traduz os sinais obtidos em cada estado de forma a representarem métricas pertinentes para o contexto [5].

Como o modelo de Friston foi desenvolvido para simular respostas de ressonância magnética quando submetido a estímulos [5], as equações utilizadas no estudo de Friston não terão aplicação no trabalho aqui realizado, focando-se na arquitetura conceitual proposta.

Acredita-se que substituindo as equações de cada estado por redes neurais com finalidades específicas, a depender da região cerebral, e desenvolvendo outros mecanismos correlatos, como função de ativação e métricas de erro, a serem utilizadas na composição da rede formada pelas redes neurais específicas, será possível obter os resultados esperados.

A principal vantagem percebida e esperada ao se utilizar desse modelo proposto por Friston é que ele modela exatamente o fluxo de processamento de uma entrada visual num cérebro humano adulto. Como essa finalidade é a mesma da aqui estudada, interpretar imagens, seu uso na modelagem pode ser muito benéfico.

O modelo apresentado por Friston [5] mistura algumas notações diferentes, V1 e V4 referem-se à função exercida pelas áreas, classificação funcional; BA37 e BA39 se referem às *áreas de Brodmann*, classificação geográfica; e STG se refere a posição macroscópica no cérebro [5]. Para manter a homogeneidade de notação, todas as áreas serão traduzidas

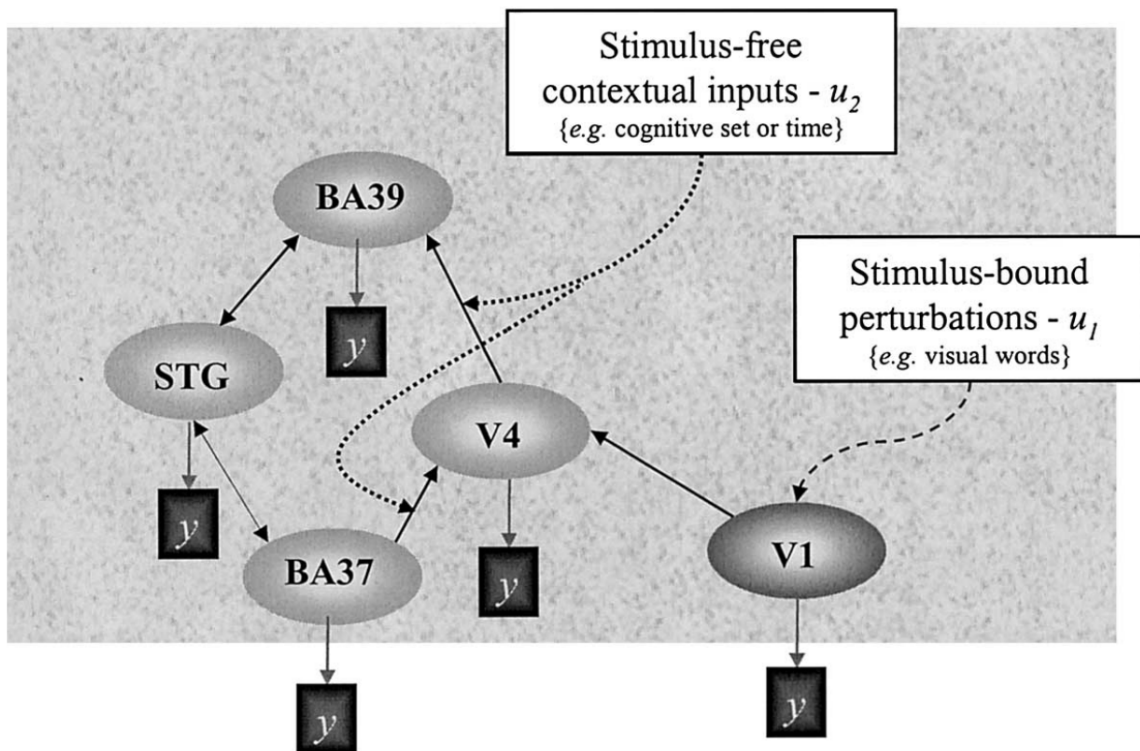


Figura 6: Modelo dinâmico causal de Friston [5]

nas suas respectivas BA 's.

A área V1, que corresponde à BA17, é a primeira área visual, responsável por criar um mapa de saliências das imagens observadas, isto é, extrair as informações mais relevantes das imagens [9, 26]. Essa área, portanto, representa um primeiro mecanismo de interpretação das entradas sensoriais.

A área V4, correspondendo à BA19, é a quarta área visual e realiza funções similares à V1, porém em níveis mais detalhistas e relacionando a conhecimentos passados [9, 26]. Essa área refina os resultados da BA17, melhorando a representação dos dados da entrada.

A área BA37 pertence à área de reconhecimento facial e permite identificação de pessoas por seus rostos, passa por processos de comparação de experiências passadas e presentes e armazena informações específicas [9]. Essa área é responsável por recuperar dados aprendidos e aplicá-los sobre os dados processados pela BA19.

A área BA39 é ligada à área de Wernick, macrorregião responsável pela associação de informações, e à área integrativa comum, responsável por responder às entradas sensoriais [9]. Assim, essa região assume o papel de traduzir entradas sensoriais em pensamentos, analisando os conteúdos e gerando representações;

O STG (Giro Temporal Superior²²), que corresponde à BA42, é ativado durante os raciocínios que geram momentos de introspecção e de descoberta repentina^[9,27]. Dessa forma, essa área é o principal ponto da rede em que as informações são associadas, para gerar as inferências.

De uma forma geral, o modelo de processamento desenhado por Friston opera de maneira muito simples, as áreas BA17 e BA19 recebem as entradas sensoriais provenientes do meio externo e as reduzem para suas características fundamentais, a área BA39, então, gera modelos mentais completos representativos das entradas e os repassa para a BA42, que cria um processo de inferências relacionando os modelos atuais com os passados, realimentando o BA39 e fornecendo bases para o BA37 definir os padrões de reconhecimento, que retornam aos estados anteriores da rede, refinando o processo.

Acredita-se que, partindo dessa rede proposta por Friston e acima analisada, é possível desenvolver um modelo de inteligência artificial com as propriedades necessárias para resolver o problema proposto de classificação de imagens. No capítulo ^[3], será discutida a forma que será adotada para esse modelo de forma a combinar todos os aspectos apresentados neste capítulo.

²²Do inglês Superior Temporal Gyrus

3 METODOLOGIA

Nesta seção serão tratados os aspectos conceituais e técnicos que foram utilizados no projeto. Serão abordados todos os aspectos relevantes que permitiram uma boa implementação do trabalho, desde a concepção do modelo até detalhes do método de desenvolvimento e dos dados utilizados.

Na primeira parte será coberto o modelo que foi construído, na segunda as tecnologias que foram utilizadas no desenvolvimento, na terceira os dados que foram utilizados para treino do modelo e na quarta as métricas que foram utilizadas para avaliação dos resultados.

3.1 Modelagem

Partindo dos conceitos apresentados no capítulo anterior, estruturou-se a arquitetura geral do modelo desenvolvido e testado. Este modelo computacional teve que respeitar, antes de qualquer outra coisa, as necessidades de cada estágio de Piaget-Inhelder e estar de acordo com os preceitos de neurologia.

O modelo proposto para representar computacionalmente os processos psicológicos deve buscar uma simulação de aprendizagem baseada no modelo mostrado por Friston em [5] e replicado na figura [6]. Este modelo, porém, foi levemente alterado para concordar com as teorias psicológicas estudadas.

Uma das mudanças propostas no modelo consiste basicamente em limitar as entradas ao nó BA17 e a saída ao nó BA39, já que a entrada é apenas a imagem, sem contextos externos e o esperado para a saída é uma classificação dos dados de entrada.

A outra mudança foi trocar a relação entre BA17 e BA19 de unidirecional para bidirecional, pois sabe-se, da psicologia, que crianças pequenas, que não possuem pensamento formal, são capazes de recordar formas e objetos no curto prazo [4, 23].

Com essas alterações feitas, o modelo revisado tem a estrutura mostrada na figura [7].

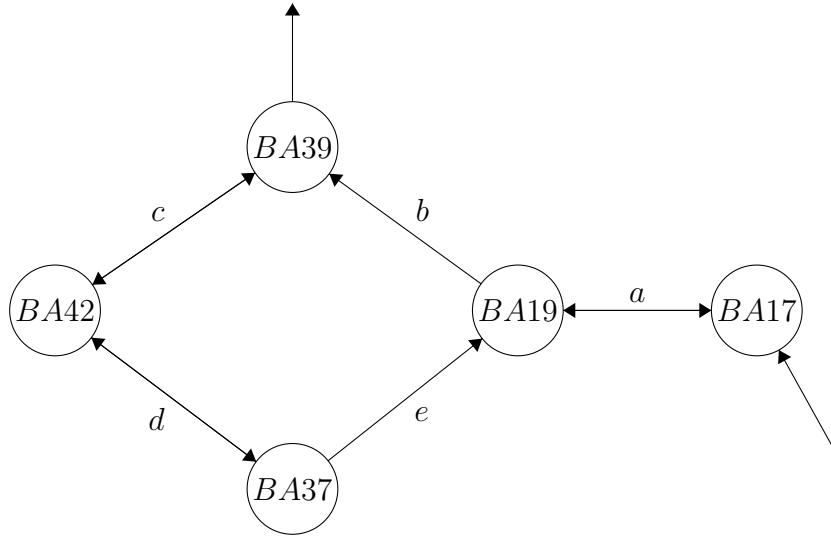


Figura 7: Modelo proposto para desenvolvimento

Cada um dos nós desta rede proposta representa um conjunto de neurônios em uma região cerebral e, portanto, devem ser modelados por camadas de uma rede neural para que seja possível realizar a implementação. Essa correspondência foi feita considerando as características das *áreas de Brodmann* e dos tipos de neurônios apresentados nas seções [2.5](#), [2.6](#) e [2.2.1](#).

O nó BA17 deve receber a imagem de entrada e as informações que retornam do BA19 e extrair, desse conjunto de dados, as características principais da entrada visual. Para realizar esse processo o tipo de neurônio mais indicado é o convolucional, que recortará os detalhes mais importantes da imagem.

O nó BA19, por sua vez, deve receber esses dados manipulados pelo nó BA17 e processar as informações, gerando dados compilados e a saída de retorno que auxilia o BA17. Para esse processo será utilizada uma rede MLP com duas saídas em estados diferentes do processamento, cada uma delas para cada uma das finalidades mencionadas.

Já o nó BA39 tem a função de gerar a classificação final da entrada partindo dos dados compilados pelo nó BA19 e das inferências do nó BA42. Para atingir esse objetivo, o modelo deve ser montado como uma rede decodificadora, que é um tipo de MLP cujas camadas tem sempre número maior de neurônios que a anterior, possibilitando a geração de mais informação.

O nó BA37 incorpora as informações de inferência do BA42 e retorna dados auxiliares para os nós BA42 e BA19 poderem realizar suas funções. A rede para este trabalho será do tipo recorrente, tratando os dados anteriores de forma a relacioná-los aos novos.

Por fim, o nó BA42 é o que deve gerar inferências partindo dos dados apresentados, recebendo os resultados da análise histórica dos dados, do BA37, e as classificações obtidas da imagem em análise, do BA39, e retorna dados do aprendizado construído para os mesmos estados. Essa inferência se dá pelo cruzamento das informações de ambas as entradas e será realizada por uma rede MLP.

Para a boa modelagem de todos os parâmetros de entrada e saída das camadas, é preciso manter muito controle do formato dos dados que são passados. Por isso, definiu-se, em antecedência, que em todos os casos que houver mais de uma entrada em uma camada, elas serão concatenadas antes de serem passadas pelos neurônios.

O modelo completo, como foi definido, representa o processamento completo das entradas visuais, como seria feito por um adulto, ou seja, representa o comportamento do estágio III. Conjectura-se que os estágio I e II podem ser modelados como sub-redes desta rede completa e o controle do estágio modelado foi feito pela ativação das transições entre os nós da rede (representadas por “a”, “b”, “c”, “d” e “e”).

Durante o estágio I, quando não há identificação de características, haverá apenas a transição “a” e a saída do modelo será, temporariamente, fornecido pelo nó BA19. Já no estágio II, quando a criança passa a classificar as entradas, será ativada, além da “a”, a transição “b”.

3.2 Tecnologias

A rede proposta é bastante não-ortodoxa, primeiramente por conter muitos nós de tipos diferentes e a relação entre eles é relativamente complexa, com redes admitindo mais de uma entrada e mais de uma saída, além dos ciclos. Por isso, a forma convencional de desenvolvimento de redes, por sequência de camadas, não pôde ser aplicada.

Para a codificação foi utilizada a linguagem *Python* no interpretador online *Colab* com o módulo *pyTorch* para criação das redes neurais, além de outras bibliotecas pertinentes para organização e tratamento numérico (*Pandas*, *Numpy*, etc.). Já para o desenvolvimento do site de demonstração do trabalho foi utilizado o arcabouço *Streamlit*, também com a linguagem *Python*.

A linguagem *Python* foi escolhida pela sua simplicidade de escrita, com instruções práticas e que facilitam muitos processos, além de existirem muitas bibliotecas para diversas finalidades disponíveis para uso.

O *Colab* é um ambiente virtual de programação da *Google*, ele foi escolhido como plataforma pois as operações a serem realizadas são muito volumosas e seriam demoradas demais para serem executadas em uma máquina local, além da possibilidade de alocação de GPU's para uso.

O módulo *pyTorch* disponibiliza funções para programação de redes neurais e foi selecionado devido à facilidade de uso das suas rotinas e da liberdade de manipulação das camadas e funções, que permite a elaboração de redes com formatos diferentes, como o pretendido.

O arcabouço *Streamlit* é um mecanismo bastante simples e funcional para desenvolvimento de sites. Ele oferece, na forma de um módulo *Python*, diversas funções para montar uma plataforma web simples. Ainda é possível hospedar o site criado na própria plataforma do *Streamlit*.

3.3 Processamento dos dados

Para o treinamento e teste da rede neural desenvolvida, foi necessário que existissem dados disponíveis representando fenômenos de natureza correlata ao do sistema modelado. Além disso, conhecendo-se o tipo de entradas e saídas a serem computadas e previstas, pôde-se desenvolver os formatos de entrada e saída das redes menores de forma a favorecer seu funcionamento.

3.3.1 Conteúdo

Como exposto, a rede proposta é bastante complexa, e, como uma rede simples já necessita de uma quantidade grande de dados para convergir, esperava-se que fosse necessário uma quantidade muito maior de conjuntos de informações para que a rede apresentada possa gerar resultados.

Sobre o formato dos conjuntos, uma vez que este for conhecido, será possível determinar os tamanhos de entrada e saída de cada um dos nós da rede, possibilitando o desenho das arquiteturas de cada uma das camadas, como definidas na seção [3.1](#), para que elas possam interagir da forma correta, gerando as informações desejadas.

Por fim, sobre o conteúdo que se deseja representar, era desejado que os dados de treino e teste da rede representem situações ou experimentos do cenário do desenvolvimento, como determinação de altura de um líquido em vasos comunicantes ou determinação da

flexibilidade de uma barra dadas suas propriedades, experimentos utilizados por Piaget [4]. Estes exemplos podem ser resolvidos de forma analítica, mas como são experimentos que já foram aplicados em crianças, poderiam ser úteis para averiguar a correspondência do modelo com o que é observado no mundo real.

Mesmo depois de extensa busca na Internet e com pesquisadores da área, não foi possível encontrar nenhum conjunto de dados como o descrito acima. Naturalmente, essa situação era esperada, já que os dados desejados, assim como o caso analisado, são extremamente específicos. Neste caso optou-se pela utilização de um dataset padrão de imagens, o EMNIST.

3.3.2 Formato

Um dataset mais comumente utilizado é o MNIST, que contém imagens de dígitos escritos manualmente. O EMNIST é um dataset predecessor ao MNIST, contendo muitos dados de caracteres escritos manualmente, não se limitando à dígitos [28].

O subconjunto do EMNIST apenas com dígitos foi o utilizado no projeto e tem 280k imagens de 28x28 pixels com seu devido identificador numérico [28]. O identificador do conjunto EMNIST é composto por um número inteiro referente ao dígito que está escrito na imagem correspondente [28].

Como exposto na seção 2.1.1, para problemas de classificação é mais indicado o uso da técnica *One-hot encoding* para representação das classes. Assim que esse será o padrão adotado para as classes do problema. Felizmente, o *pyTorch* realiza automaticamente a conversão dos dados para *One-hot encoding*, não sendo necessário nenhum tratamento isolado.

3.3.3 Aumento

Como exposto anteriormente, foi necessário uma quantidade muito grande de dados para treinar e testar o modelo proposto, de forma que não se acreditava serem suficientes as 280k imagens originais do dataset escolhido.

Para alcançar um maior número de imagens para o projeto, foram aplicadas técnicas de aumento de dados sobre o dataset. Para realizar esse aumento, foi utilizado o módulo *Imgaug* do *Python*, específico para fazer aumento de dados com imagens [29].

Neste módulo, estão disponíveis diversas as funções para criar variações sobre ima-

gens[29]. Em especial observa-se métodos para cortar e girar as imagens, inserir ruídos com diferentes formas e distribuições de probabilidade e suprimir pedaços das imagens[29]. Para aumentar o conjunto de dados disponível, foram utilizadas rotações e, caso fosse necessário, seriam utilizadas inserções de ruído gaussiano.

3.4 Métricas

Para finalidade de avaliação da compatibilidade do modelo desenvolvido, estudou-se quais parâmetros poderiam ser utilizados para treinar o modelo e quais podem ser avaliados para refletir as diversas propriedades dos resultados obtidos.

Existem algumas métricas que são classicamente utilizadas em problemas de aprendizado de máquina e outras não tão usuais que podem mostrar aspectos específicos dos dados. A seguir encontra-se uma descrição das métricas que foram utilizadas no trabalho.

Como foi largamente discutido na seção 2.2.3, a função mais apropriada para o caso específico, em que as saídas são funções massa de probabilidades e são comparadas com vetores em *One-hot encoding*, é a métrica de entropia-cruzada.

Já para avaliar o desempenho do modelo depois de treinado, algumas métricas tiveram que ser escolhidas para que fosse analisado o cenário de forma completa. As métricas escolhidas foram *acuidade*, *pontuação f1*, *matriz de confusão* e *pontuação de Brier*, conforme descritas a seguir.

A *acuidade* e a *pontuação f1*¹ são duas das métricas que foram utilizadas e que representam alguns aspectos a serem considerados[30]. Usualmente são consideradas conjuntamente, pois ambas refletem a assertividade do classificador desenvolvido em termos das classificações corretas e erradas que ele faz.

A *acuidade* mede quantas previsões corretas foram feitas pelo modelo em comparação com todas as previsões feitas[30]. Já a *pontuação f1* indica quantos dos dados foram corretamente classificados entre os que deveriam ser e os que realmente são de determinada classe, desconsiderando todos os casos em que não aparece a classe sendo analisada[30].

Essas duas métricas variam entre 0% e 100%, sendo que seu valor será tanto maior quanto mais entradas forem corretamente classificadas. A *f1* é mais sensível que a *acuidade*, uma vez que desconsidera os verdadeiros negativos. Como o caso de estudo é um classificador de 10 classes, a *acuidade* mínima esperada é de 10%.

¹Do inglês f1-score

Outra métrica importante é a *matriz de confusão*, que pormenoriza todas as combinações de resultados retornadas pelo modelo [30]. A *matriz de confusão* é uma matriz quadrada, com lado de tamanho igual ao número de classes do problema.

A *matriz de confusão* apresenta na i -ésima linha e na j -ésima coluna quantos dados da classe i foram classificados como da classe j [30]. Assim que pode indicar quais classes foram mais difíceis de serem separadas pelo classificador desenvolvido.

Os valores nos espaços da matriz podem variar entre 0 e o total de amostras da classe da linha. No melhor caso, a matriz será uma matriz diagonal, indicando que todos os dados foram corretamente classificados.

Vale notar que as métricas de *acuidade* e *pontuação $f1$* podem ser obtidas diretamente da *matriz de confusão*, realizando-se as operações adequadas. No entanto, fazer um estudo separado dessas métricas é proveitoso para que diferentes características possam ser avaliadas individualmente.

Por fim, é interessante que seja possível avaliar o quão próxima de uma distribuição ideal (com probabilidade 1 na classe predita) está a distribuição obtida pelo modelo. Para tal, poderia ter sido utilizada a *DKL* ou a *EC*, mostradas na seção [2.2.3], mas estas métricas não levam em conta as probabilidades das classes que não sejam a desejada. Por esta razão, optou-se por utilizar a *pontuação Brier* [2], que que é um *EQM* aplicado a cada um dos valores das classes ao invés de a cada vetor de saída da rede [31].

$$P = \frac{1}{n} \sum_{j=1}^r \sum_{i=1}^n (f_{ij} - E_{ij})^2 = \frac{1}{C} \sum_{i=1}^M \sum_{j=1}^C (\bar{v}_j^i - v_j^{Li})^2 \quad (3.1)$$

[3]

O vetor \bar{v} , utilizado para fazer a comparação com a saída da rede, não é, necessariamente, o vetor de saídas esperadas, y , mas um outro vetor formatado em *One-hot encoding*, indicando a classe predominante na saída da rede, v^L .

Ponderando-se algumas questões sobre o comportamento geral da *pontuação de Brier* e da sua aplicação no caso específico estudado, introduziu-se uma pequena alteração na fórmula apresentada (para mais detalhes ver o apêndice [A]).

$$P_{modificado} = \frac{1}{0.09 \cdot M \cdot C} \sum_{i=1}^M \sum_{j=1}^C (\bar{v}_j^i - v_j^{Li})^2 \quad (3.2)$$

²Do inglês Brier score

³Fórmula expandida de [31] para se adequar à notação de [16]

Com essa alteração, a métrica será chamada de *pontuação de Brier modificada* e tem seu valor variando entre 0% e 100%, sendo 0% o caso ideal, onde a saída do modelo se apresenta na forma de *One-hot encoding*, e 100% o pior caso, onde a saída é uma distribuição uniforme de probabilidades.

4 DESENVOLVIMENTO

Nesta seção serão expostos os passos e métodos aplicados para o desenvolvimento do trabalho, passando por todas as sub-etapas de cada fase do processo. Para tudo que for exposto, não será mostrada a implementação e a explicação ater-se-á ao nível conceitual.

Na primeira seção estão cobertas as simplificações realizadas na rede, na segunda o que foi implementado em termos de processamento de dados e desenvolvimento das redes, na terceira o processo de treinamento das redes e na quarta os resultados da implantação da plataforma web.

4.1 Simplificação da rede

Como a rede prevista para o processo completo (do terceiro estágio de Piaget-Inhelder) é muito complexa e inclui muitas camadas diferentes e muitas realimentações, a sua implementação poderia ser muito mais complicada que a das outras redes.

Outro fator proibitivo para criação do modelo completo é a falta de dados para tal, como o estágio III se propõe a criar inferências a partir dos dados, um classificador é simples demais para testar seu funcionamento.

Dessa forma, optou-se por limitar o estudo aos dois primeiros estágios, podendo redesenhar as redes neurais para eles de forma a reduzir o número de neurônios e simplificar a interpretação. As figuras 8 e 9 mostram as redes redesenhadas para cada caso, com suas respectivas entradas e saídas.

É importante notar que, embora o ciclo maior presente no modelo para o estágio III tenha sido retirado, ainda há um ciclo presente nas redes, entre os nós BA17 e BA19. Este ciclo confere a propriedade ação-reação do estágio I e, pelo menos conceitualmente, ajudará no reconhecimento dos dígitos no estágio II.

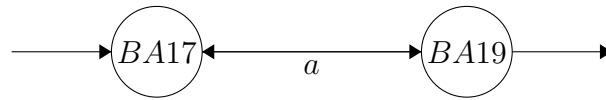


Figura 8: Modelo do estágio I



Figura 9: Modelo do estágio II

4.2 Implementação

Com o desenho dos modelos reestruturado, pôde-se passar para a etapa de programação efetiva do projeto. Como é de se esperar, houveram dificuldades que precisaram ser superadas durante o tempo de projeto. Nesta seção serão detalhados os processos desenvolvidos e os problemas enfrentados.

4.2.1 Tratamento de dados

Como exposto na seção [3.3](#), o conjunto de dados escolhido precisou passar por alguns tratamentos antes de poder ser utilizado para treinamento e teste dos modelos desenvolvidos. As classes, bem como as próprias imagens passaram por processos de reformatação e aumento para atingir os objetivos visados. Nesta seção estão resumidos os processos que nelas foram aplicados.

Como o problema abordado é do tipo classificação, o melhor formato para as classes serem apresentadas para a rede é no modelo de *One-hot encoding*, para que possa ser mais facilmente comparado às saídas das redes, de acordo com o explicado na seção [2.1.1](#).

Porém, como já foi comentado, o *pyTorch* realiza automaticamente a conversão das classes para este formato, assim que não foi necessário nenhum tipo de processamento para adequar os dados ao *One-hot encoding*, reduzindo significativamente o trabalho.

As imagens, por outro lado, precisavam ser aumentadas para serem suficientes para permitir um bom treinamento, como discutido na seção [3.3.3](#). Dentre as várias técnicas disponíveis no módulo *imgaug*, optou-se por realizar apenas rotações sobre as imagens disponibilizadas, para que não fosse comprometida a qualidade das imagens de nenhuma

forma.

De acordo com o que foi visto nos testes com este módulo, os métodos de introdução de ruído e supressão de partes da imagem inserem elementos que podem desfigurar o conteúdo da imagem, de forma que poderiam ocorrer problemas durante a aprendizagem do modelo.

Sobre cada uma das imagens foi aplicado um conjunto de quatro rotações, 5° e 10° no sentido horário e 5° e 10° no sentido anti-horário. Agrupou-se estas novas imagens ao conjunto original, conseguindo quintuplicá-lo, alcançando um total de 1.4M de imagens, o que se considerou um número adequado para o treinamento.

Exemplos de aplicações deste aumento podem ser vistos na figura [10](#), na qual a primeira linha corresponde à imagem original, a duas seguintes às rotações de 5° e as duas últimas às rotações de 10°.

A posteriori, poderiam ter sido identificados problemas de sobreajuste dos modelos, requerendo, assim, mais dados para convergir o processo de treino, como explanado na seção [2.1.2](#). Se este caso fosse detectado, poderia ser acrescentado um processo de introdução de ruído nas imagens, com o devido cuidado em sua intensidade para não descaracterizá-las.

4.2.2 Redes neurais

Com os modelos e dados estruturados, foi possível iniciar a codificação das redes neurais para o seguinte processo de treinamento e validação. O processo de desenvolvimento das redes foi longo e encontrou diversos problemas, nesta seção estão explicados os processos gerais desenvolvidos.

Inicialmente tentou-se fazer uso dos módulos *TensorFlow* e *Keras*, que impõe, junto com a praticidade de uma codificação simples e poderosa, limitações no uso das funções disponíveis. O grande problema enfrentado foi a grande dificuldade em criar ciclos entre as camadas de neurônios, como o que é preciso para retornar do nó BA19 para o BA17.

Para tentar resolver esse problema, implementou-se uma função que altera dinamicamente as entradas das redes. No início de cada *lote*, a função seria invocada automaticamente pelos módulos e ela retornaria uma lista com os próximos conjuntos de entrada e saída.

Da forma que foi implementada, esta função rodaria a rede com as saídas do *lote*

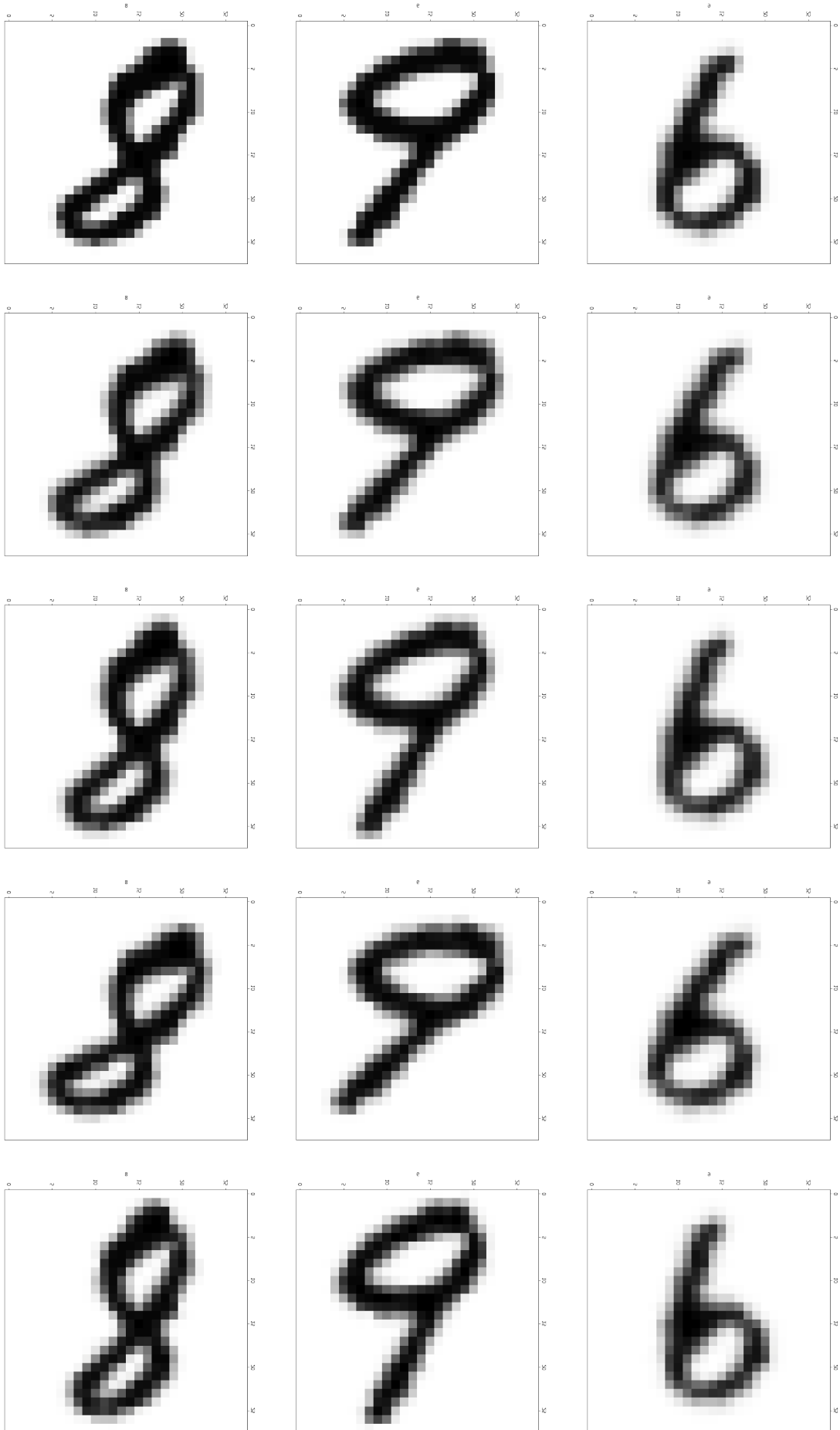


Figura 10: Exemplo de aumento dos dados

anterior, guarda as saídas dos nós que devem criar o ciclo e inserem essas saídas como entradas para o próximo *lote*.

Infelizmente a codificação dessa função se mostrou muito difícil e os resultados não tão promissores, com a ocorrência de diversos erros. Com essa dificuldade e mais tempo de pesquisa e estudo, optou-se pela troca dos módulos *TensorFlow* e *Keras* pelo módulo *pyTorch*, que consegue resolver o problema de forma bastante mais simples. Este novo módulo requer a codificação de estruturas mais complexas, porém imprime maior liberdade ao programador.

Com isso, criou-se, dentro da classe da rede de cada estágio estudado, uma variável, inicializada como um tensor de zeros, que armazena, a cada entrada passada pela rede, o valor da saída de retorno do nó BA19 para o BA17, que é utilizado na próxima iteração.

Alguns hiperparâmetros das camadas já foram inicializados, a fim de representar os modelos e servir para os primeiros testes. Nas figuras [11](#) e [12](#) as estruturas para os estágios I e II, como inicialmente formulados, estão representadas com os formatos das informações entre cada componente.

Em ambas as redes implementadas, a saída do nó BA19 retorna como entrada para o nó BA17. Na rede do estágio I, que pode ser vista na figura [11](#), a saída do nó BA19 que fecha o ciclo é um estado anterior da saída final da rede.

Já na rede do estágio II, que pode ser vista na figura [12](#), a saída do nó BA19 que era a saída final passa a ser entregue ao nó BA39, que fornecerá a saída da rede, depois de realizar ela mesma o seu processamento sobre os dados.

Vale notar que, para retornar para o nó BA17, a saída do nó BA19 tem que passar por um processo de reformatação para poder ser concatenada adequadamente e que na saída da rede é necessário inserir uma camada de 10 neurônios lineares para fornecer os dados no formato correto.

As funções de ativação escolhidas para cada camada estão representadas nos esquemáticos das figuras [11](#) e [12](#). Como as classes dos dados estão representados em *One-hot encoding*, a função de perda mais apropriada para atuar nas redes é a função de *entropia cruzada*, como já foi largamente pontuado em outras seções.

A implementação dessa função de perda no *pyTorch* já engloba o cálculo de uma função do tipo *softmax* sobre a saída da rede, por isso a última camada nos esquemáticos não tem função de ativação. O otimizador escolhido para gerenciar os ciclos da rede é o otimizador *SGD*. Todas estas escolhas foram baseadas no que foi explanado na seção [2.2](#).

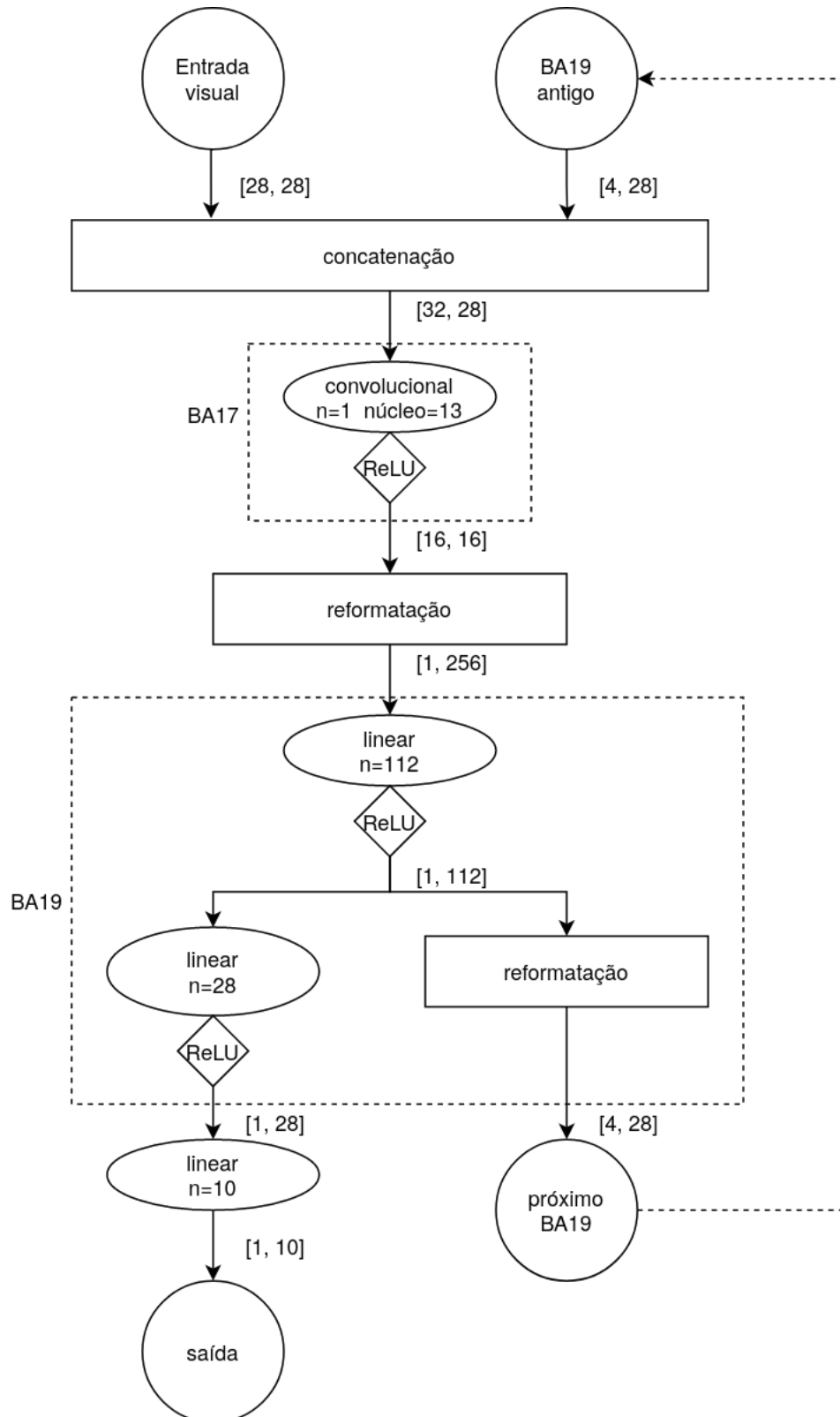


Figura 11: Primeira rede desenvolvida para o estágio I

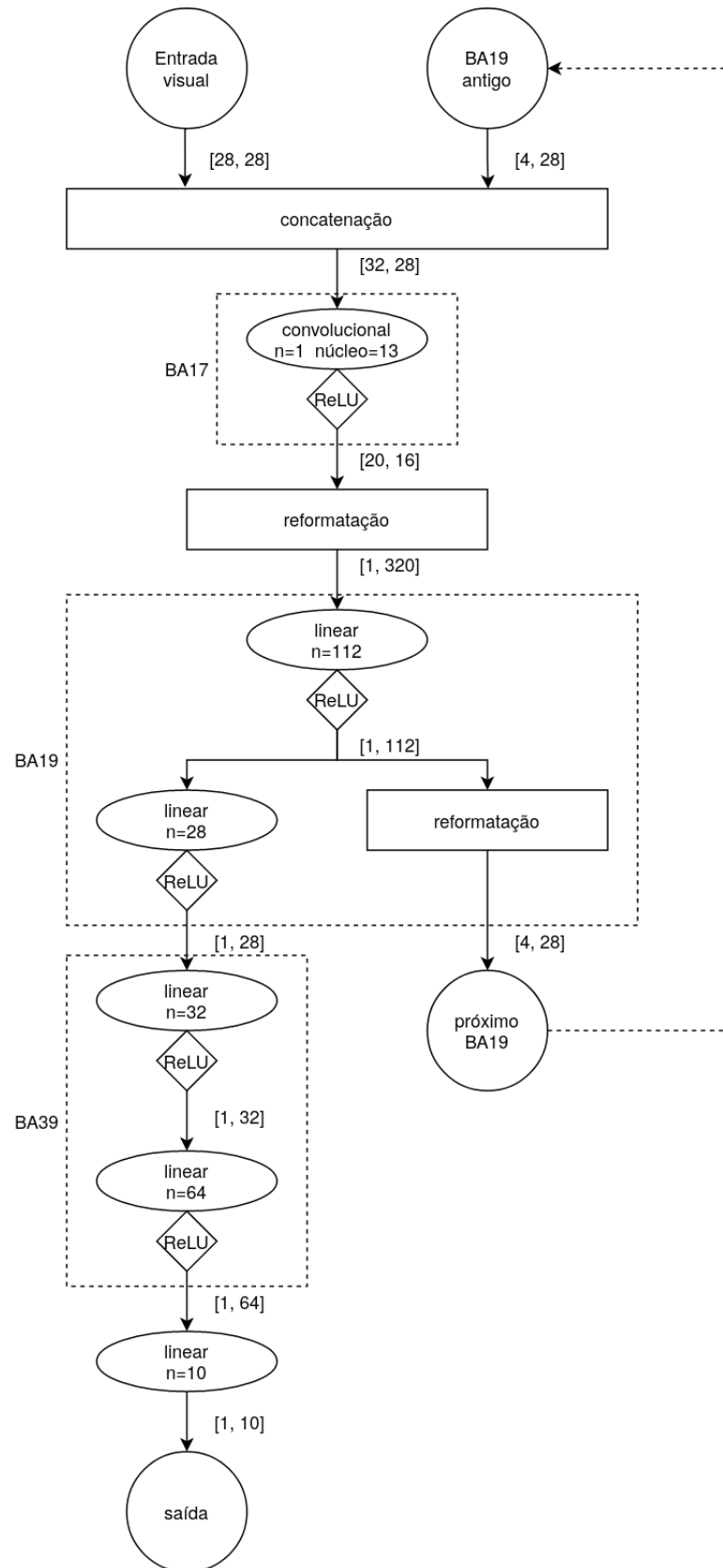


Figura 12: Primeira rede desenvolvida para o estágio II

4.3 Treinamento

Com as redes projetadas e devidamente implementadas no *pyTorch*, iniciou-se a etapa de treinamento das redes. Como foi explicado na seção 2.2, nessa etapa todos os dados são passados pelos modelos, buscando reduzir o valor da função de perda. Esta seção apresentará os detalhes da execução de cada uma das redes.

Antes de efetivamente rodar as redes, foi preciso separar uma parte do dataset aumentado para ser utilizado exclusivamente para testes, sem cálculo e retropropagação da perda, como forma de controle dos resultados. Como há uma quantidade muito grande de dados disponíveis, foram reservados 25% deles para esta finalidade, um total de 350k imagens.

Ambas as redes foram executadas por 10 *épocas*, o que, em primeira análise pode parecer pouco, mas, devido ao número de dados e à complexidade das redes, cada treinamento durou muitas horas. Além disso, foram utilizados *lotes* de tamanho 16, que permitiu o particionamento inteiro dos dados, ou seja, 65625 lotes de 16 dados de treino.

Inicialmente tentou-se utilizar nas redes o otimizador Adam, porém os parâmetros obtidos durante o treinamento não seguiram o comportamento esperado. Com o avançar das épocas o valor da função de perda aumentou e a acuidade do modelo diminuiu, como está mostrado nas figuras 13 e 14.

Ao verificar esse comportamento anômalo, decidiu-se trocar o otimizador Adam pelo SGD, que retornou um comportamento condizente. Acredita-se que o otimizador Adam tenha causado esta variação por se tratar do otimizador mais avançado e com mais passos internos, não se acoplando bem ao modelo aqui proposto, ao passo que o SGD, muito mais simples, obteve bons resultados.

Com os modelos ajustados para utilizar o otimizador SGD, as redes foram treinadas no formato mostrado na seção anterior e presente nas figuras 11 e 12. No entanto, os resultados do treino delas foram muito similares.

Dessa forma, a rede que deveria simular o estágio I estava aprendendo a diferenciar as imagens do dataset aumentado tão bem quanto a que representava o estágio II. Diversas variações nas arquiteturas foram testadas, mas estes resultados anômalos continuaram sendo obtidos.

Observando-se esse comportamento inesperado da primeira rede, mostrado nas figuras 15 e 16, que deveria ser incapaz de aprender e manter taxas de acuidade próximas de 10%,

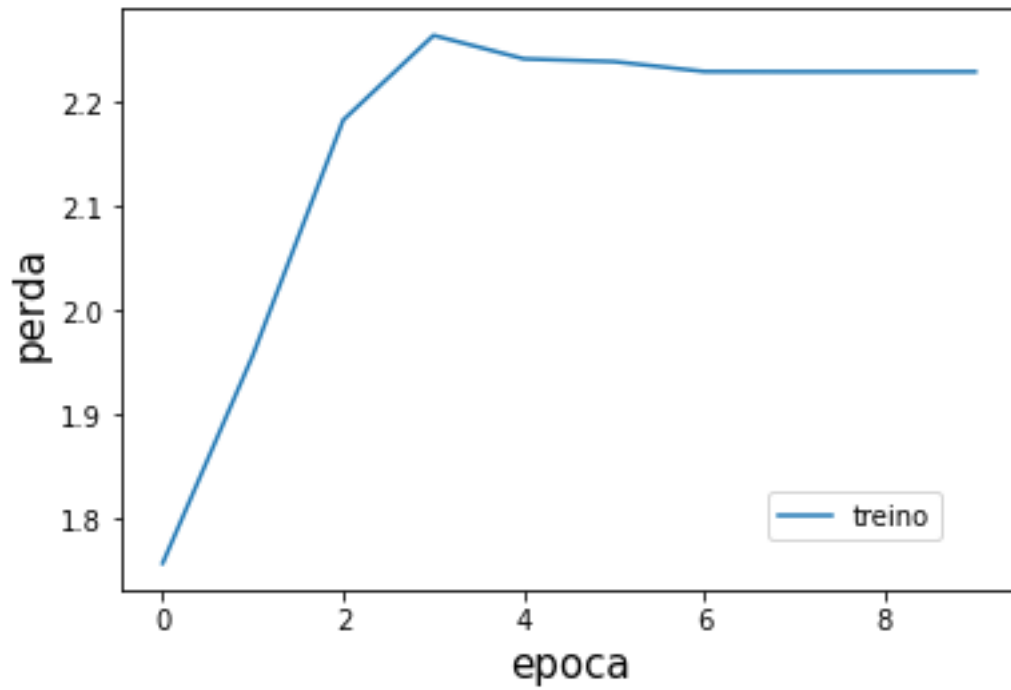


Figura 13: Perdas para o otimizador Adam

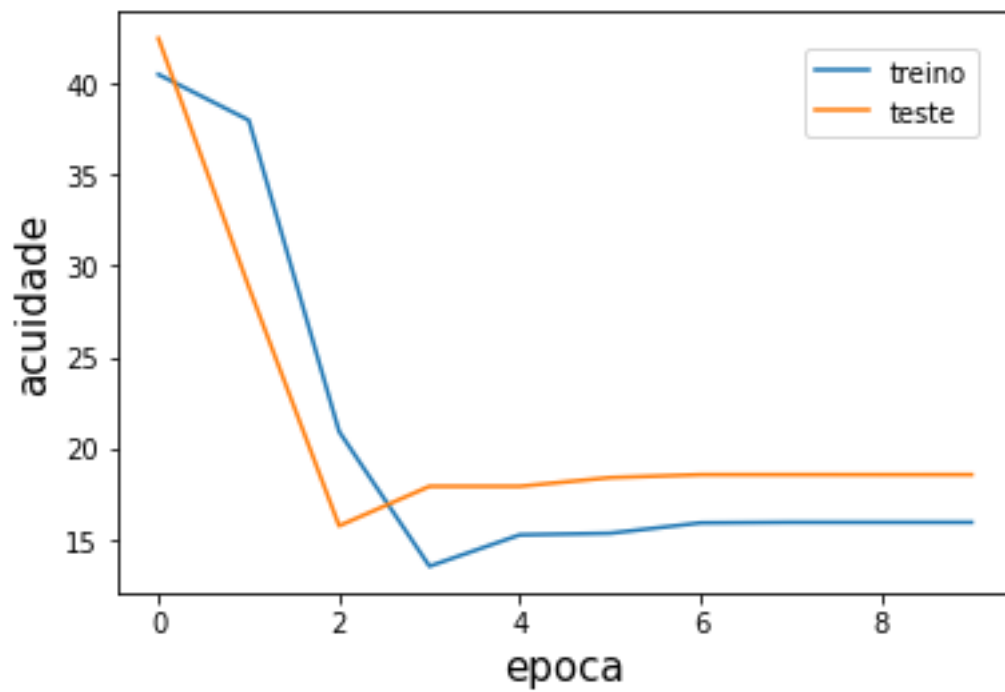


Figura 14: Acuidades para o otimizador Adam

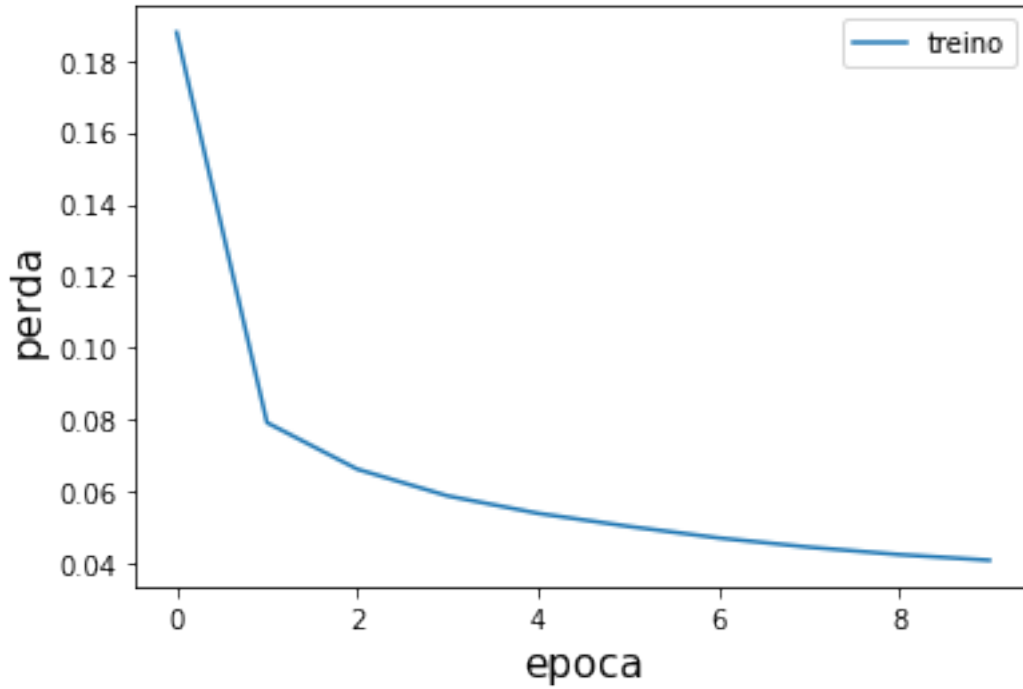


Figura 15: Perdas da primeira rede do estágio I

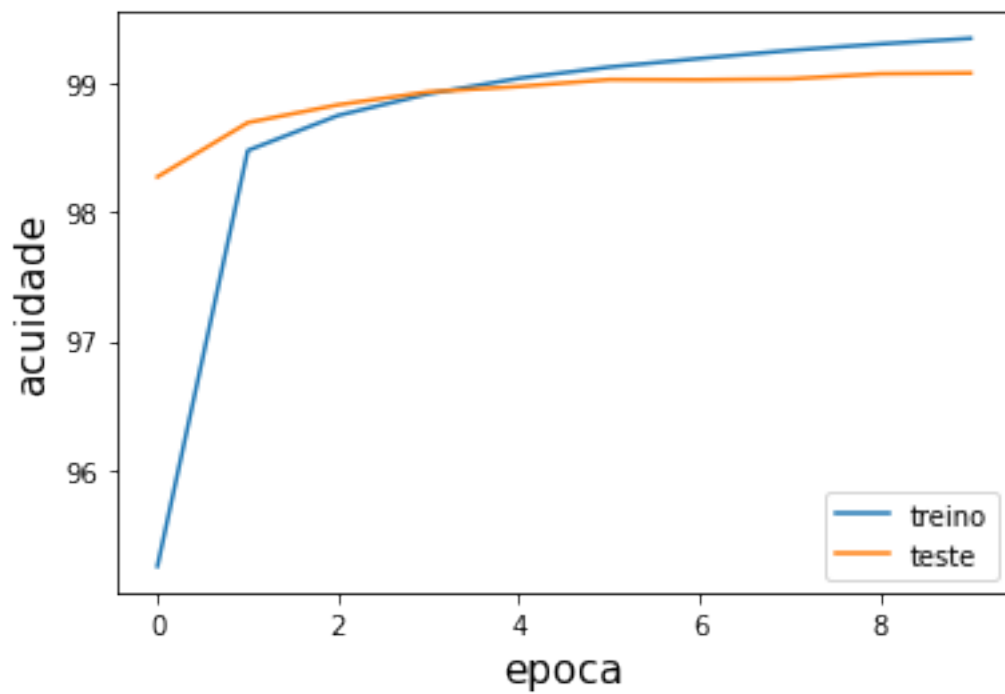


Figura 16: Acuidades da primeira rede do estágio I



Figura 17: Novo modelo do estágio I

concluiu-se que parte da hipótese formulada estava errada.

A associação do estágio I do desenvolvimento às áreas BA17 e BA19 foi incorreta, embora o modelo desenvolvido seja satisfatório. Por isso optou-se por remodelar o estágio I para conter apenas o nó BA17, a primeira camada visual, como pode ser visto na figura [17](#).

Partindo das novas premissas e do modelo reelaborado, uma nova rede para simular o estágio I foi codificada, mantendo o otimizador SGD para evitar o problema mencionado anteriormente. Esta nova rede apresentou o resultado esperado, não sendo capaz de aprender e estagnando sua acuidade em 10%, corroborando com a hipótese.

Diversos hiperparâmetros foram testados para as redes de forma a tentar gerar melhores resultados. No geral as tentativas se basearam em alterar o tamanho do cerne da camada convolucional no nó BA17 e a quantidade de unidades lineares por camada nos nós BA19 e BA39. Entre os testes houve pouca variabilidade, mas os melhores resultados foram mantidos.

No geral, os treinamentos das redes do estágio I depois da revisão duraram pouco mais de uma hora, enquanto antes da revisão tinham duração de aproximadamente 2 horas. Já para as redes do estágio II, os treinamentos duraram quase quatro horas.

As evoluções das perda e acuidades para essas duas redes finais estão nas figuras [18](#) a [21](#) e as redes com os melhores hiperparâmetros encontrados estão em [22](#) e [23](#).

É interessante notar que, no novo modelo do estágio I, independente dos esforços, a rede foi incapaz de reduzir muito a perda, melhorando um pouco seu desempenho na primeira época, mas se mantendo estável durante todo o resto do processo.

Já no treinamento do estágio II, houve melhora contínua durante toda a execução, tanto a perda quanto a acuidade de treino tiveram tendência de melhora ao fim das 10 épocas, mas a acuidade de teste se mostrava praticamente estável, indicando que mais épocas de treino poderiam gerar efeitos de sobreajuste, que são indesejados, como explicado na seção [2.1.2](#).

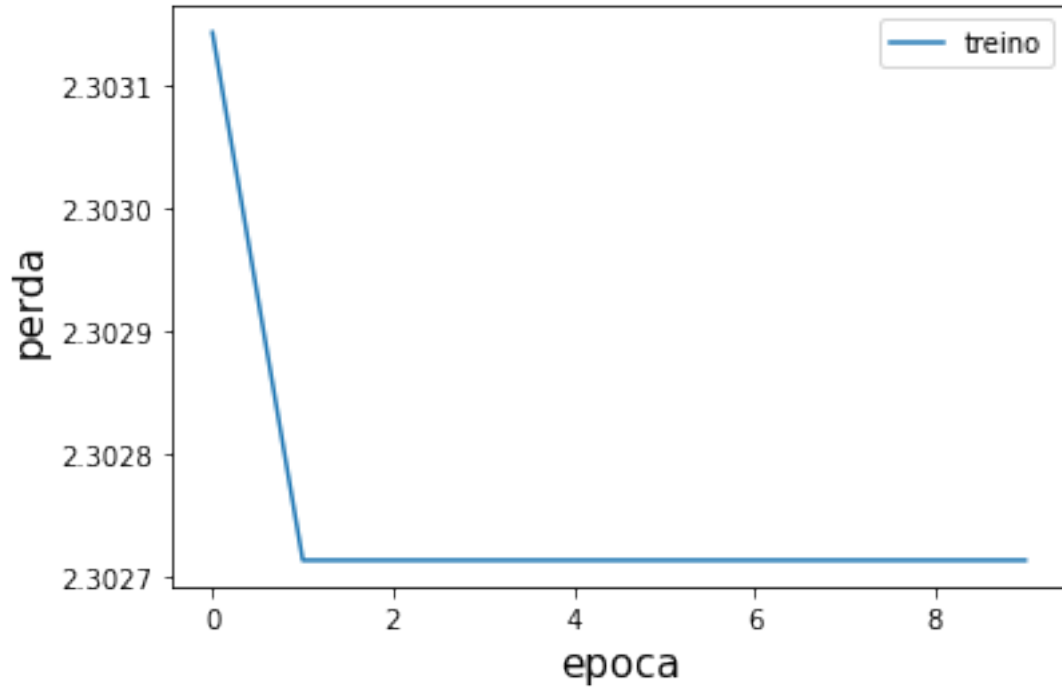


Figura 18: Perda para estágio I reelaborado

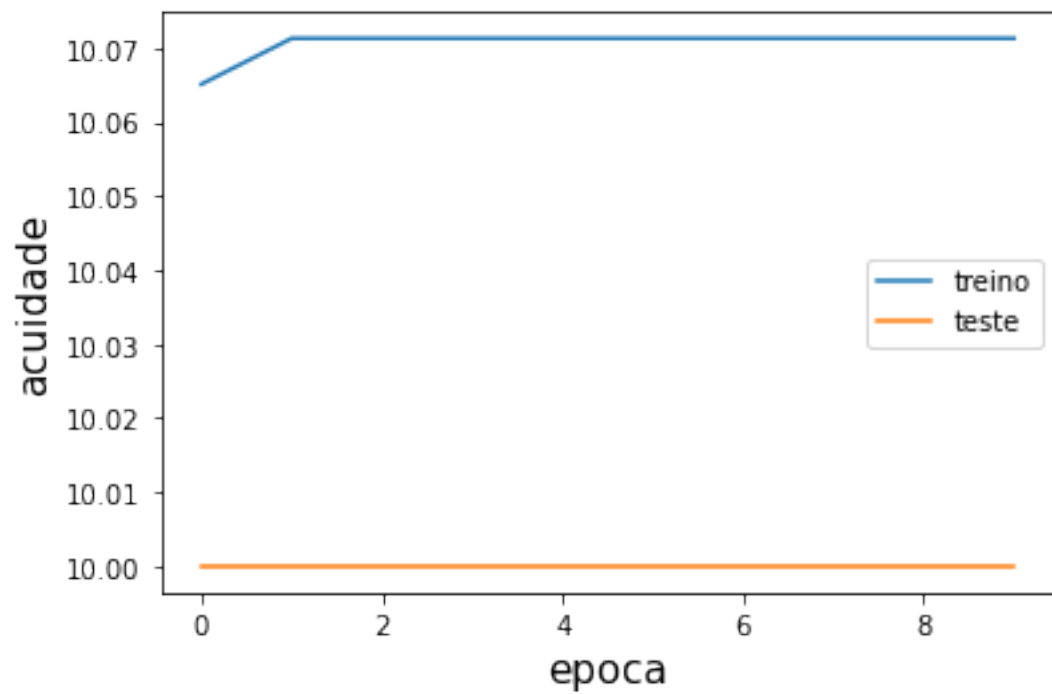


Figura 19: Acuidades para estágio I reelaborado

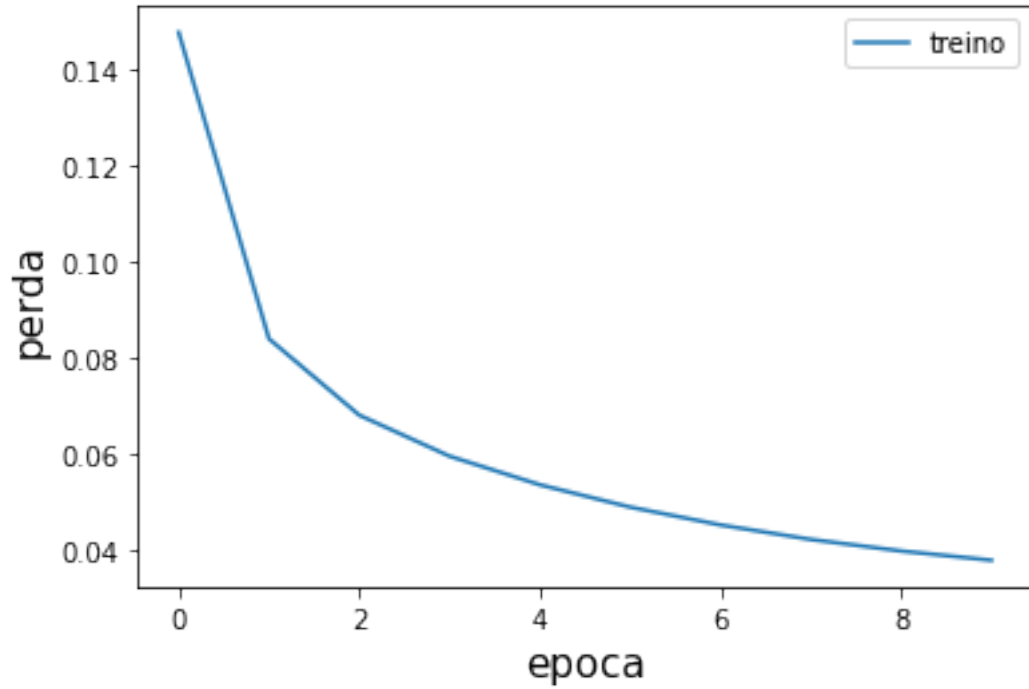


Figura 20: Perda para estágio II

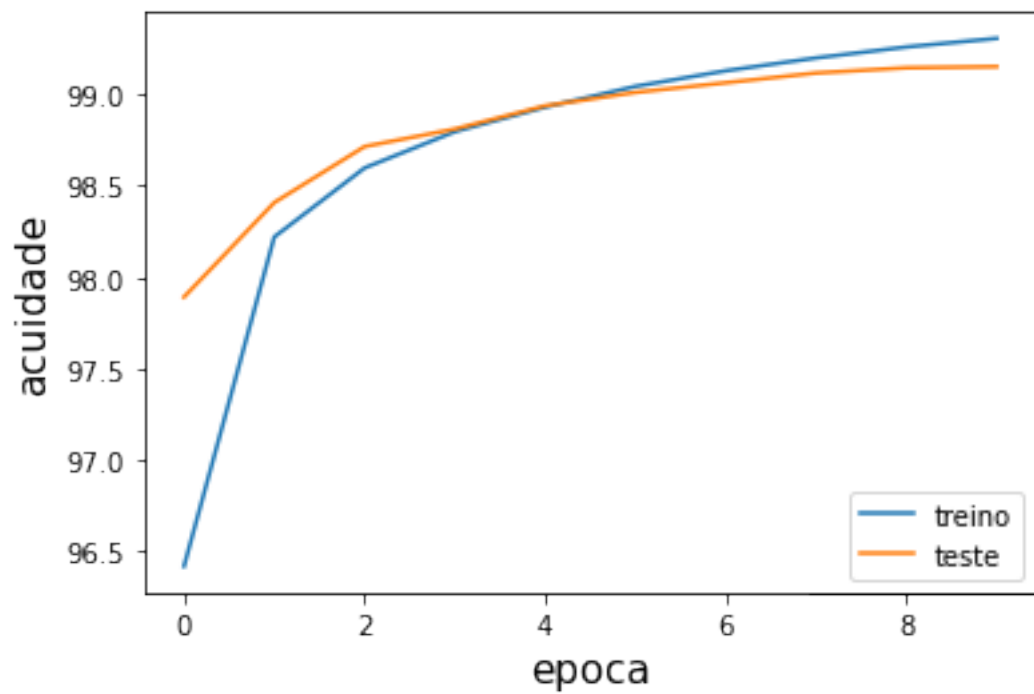


Figura 21: Acuidades para estágio II

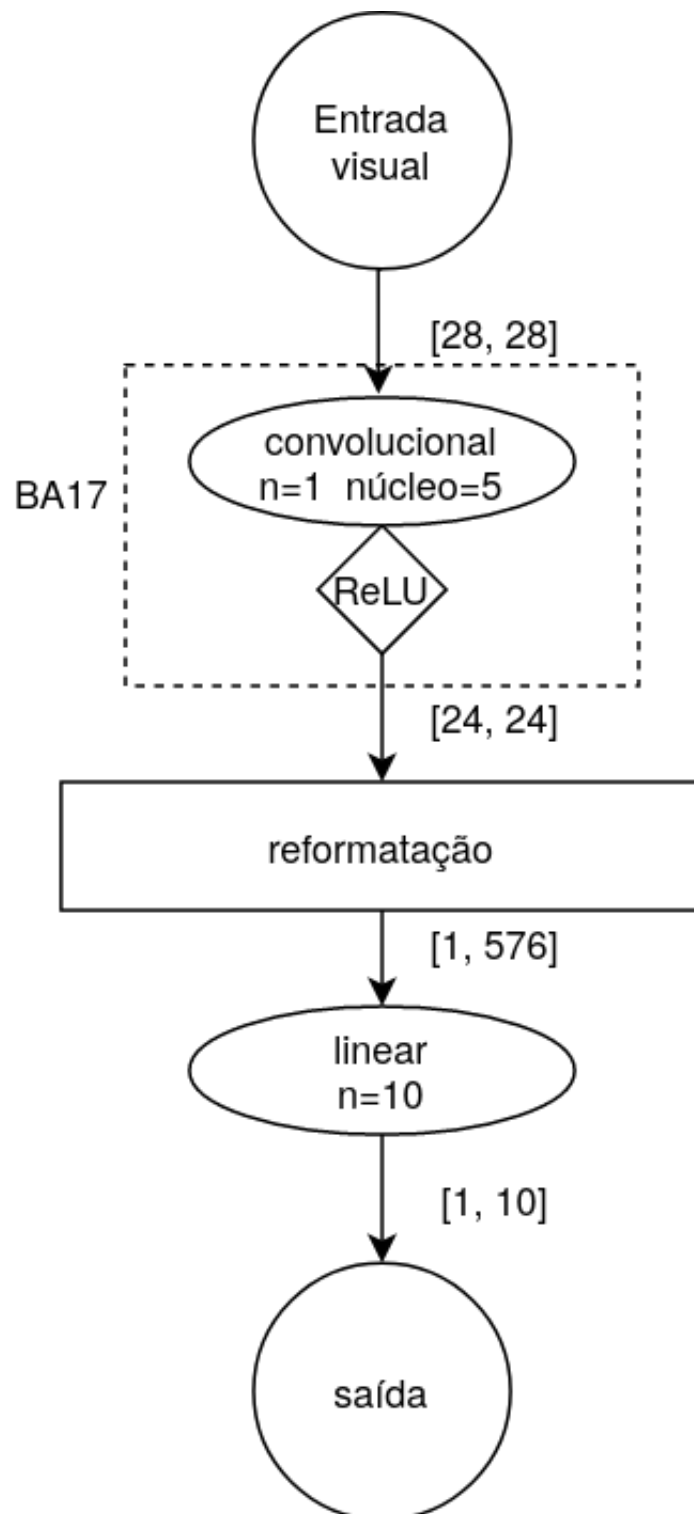


Figura 22: Rede final desenvolvida para o estágio I

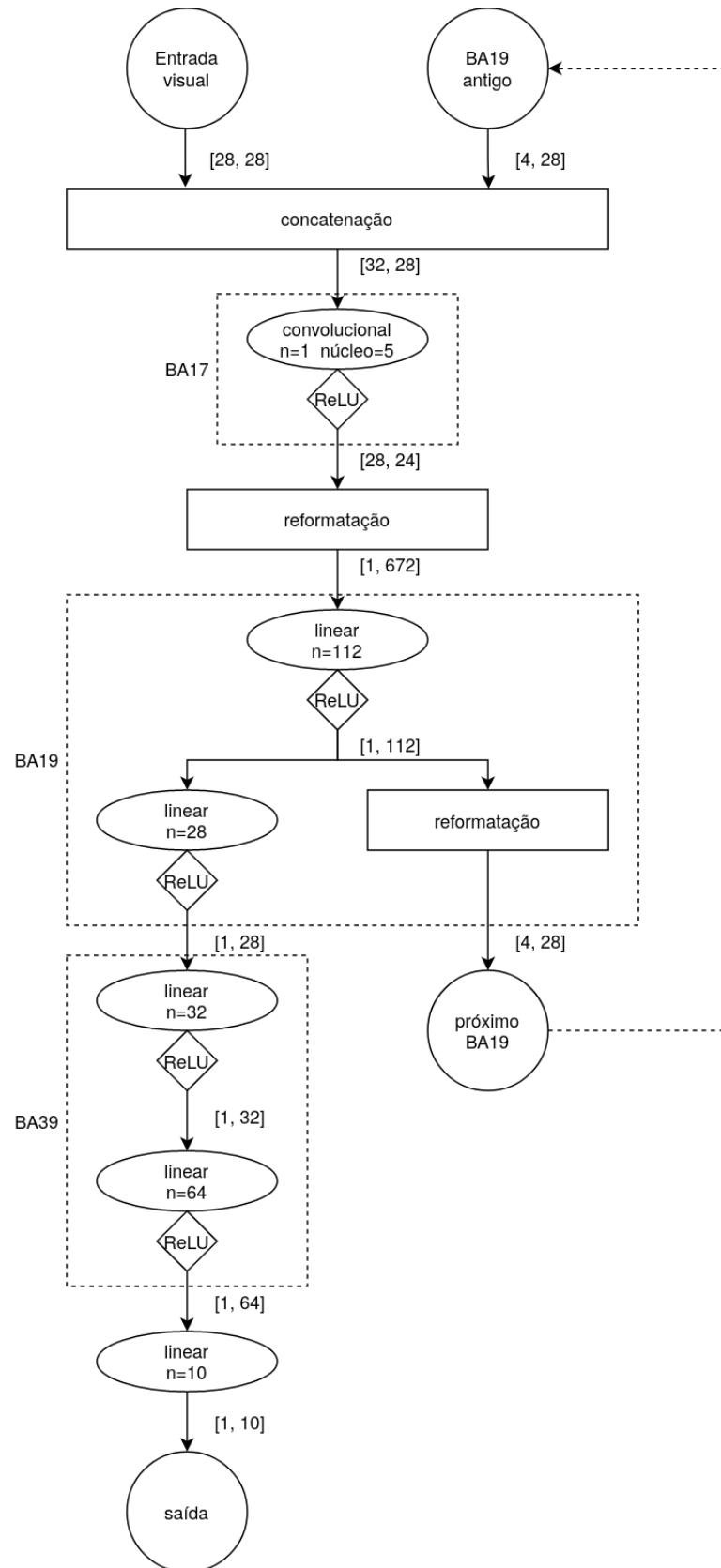


Figura 23: Rede final desenvolvida para o estágio II

4.4 Implantação do website

Com os resultados encontrados na rede do estágio II desenvolvida, que se mostrou extremamente eficaz para classificar os dados do dataset, foi desenvolvido um website para servir como demonstrativo prático dos resultados obtidos e apresentar, para um público não técnico, os processos e resultados do trabalho. Nesta seção serão apresentados os detalhes do desenvolvimento desta plataforma.

A função principal do site é mostrar a rede desenvolvida funcionando e o quão bem ela é capaz de realizar o processo de classificação. Para isso, foi desenvolvido um tipo de Teste de Turing, com as características explicadas na seção 2.3, que exige a participação de dois jogadores, um para competir com a máquina e outro para avaliar os resultados de ambos os competidores.

Inicialmente é apresentada uma breve descrição do funcionamento do experimento (imagem 24), então o primeiro jogador se identificado (imagem 25) e, sem que o segundo veja, é apresentado à 12 imagens do dataset de treino e as classifica entre os dez dígitos possíveis (imagem 26).

Em seguida, o segundo jogador entra no jogo e, sem que o primeiro veja, observa as mesmas 12 imagens, além das classificações dadas pelo primeiro jogador e pela rede treinada, e responde se é possível identificar uma máquina entre os jogadores e qual dos conjuntos de respostas é dela e qual é do humano que jogou o Teste (imagem 27).

Em cada caso os jogadores são redirecionados para uma tela onde se apresenta uma mensagem de fechamento do teste. Existem algumas telas intermediárias pelas quais os jogadores passam para organização da troca entre os jogadores, mas elas não são importantes para o objetivo geral aqui apresentado.

Vale notar que nas imagens 26 e 27 é indicada a existência de um outro jogador realizando o teste além do primeiro jogador. Este outro jogador é, na verdade, um nome fictício dado à máquina, escolhido aleatoriamente em cada execução, e serve apenas como isca para distrair os jogadores, não deixando óbvio que há uma máquina jogando com o primeiro jogador.

Assim como no Jogo da Imitação [2], a máquina do Teste de Turing desenvolvido deve ser capaz de simular o comportamento de um ser humano para que seja aprovada. No caso específico, é dito que a máquina passou no Teste de Turing se o segundo jogador, que avalia as duas respostas, não for capaz de distinguir ou distinguir de forma errônea os conjuntos de respostas, indicando que a rede neural foi capaz de confundí-lo.



Figura 24: Tela inicial do jogo

Teste de Turing

Preencha os campos abaixo com as informações do primeiro jogador

Qual seu nome?

Miguel Sarraf

Qual sua idade?

22

- +

Começar o jogo

Voltar ao começo

Figura 25: Tela de identificação do primeiro jogador

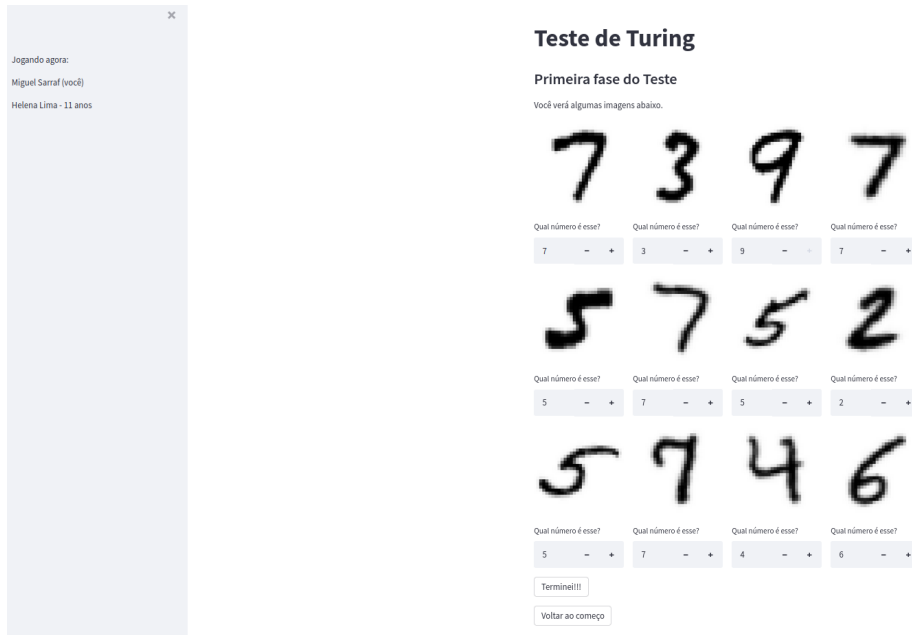


Figura 26: Tela preenchida pelo primeiro jogador

Além disso, como a plataforma tem a função de explicar, em linhas gerais e de forma simples, os conceitos envolvidos no trabalho, foram adicionadas páginas para informar sobre os processos desenvolvidos neste trabalho.

Criaram-se duas outras abas, acessíveis da página principal, que inicia o experimento, onde se encontram informações teóricas. Em uma explica-se o conceito de Teste de Turing e na outra apresenta-se um resumo muito breve do trabalho, seus métodos e resultados. A partir desta última página também é possível baixar esta monografia, o banner e o press-release do projeto.

Por fim, foi desenvolvido um formulário para avaliar os resultados de usuários reais na aplicação do teste. O formulário contém sete perguntas simples sobre o processo e resultados de toda a experiência de uso do website.

A primeira pergunta questiona de que dispositivo o usuário acessou a plataforma, as quatro seguintes identificam os resultados dos dois jogadores nas perguntas do jogo e as duas últimas são perguntas opcionais sobre problemas e sugestões.

Tanto no site como no formulário, por estes serem voltados para o público geral, foi utilizada uma linguagem menos formal, sem nenhuma perda de validade do conteúdo.

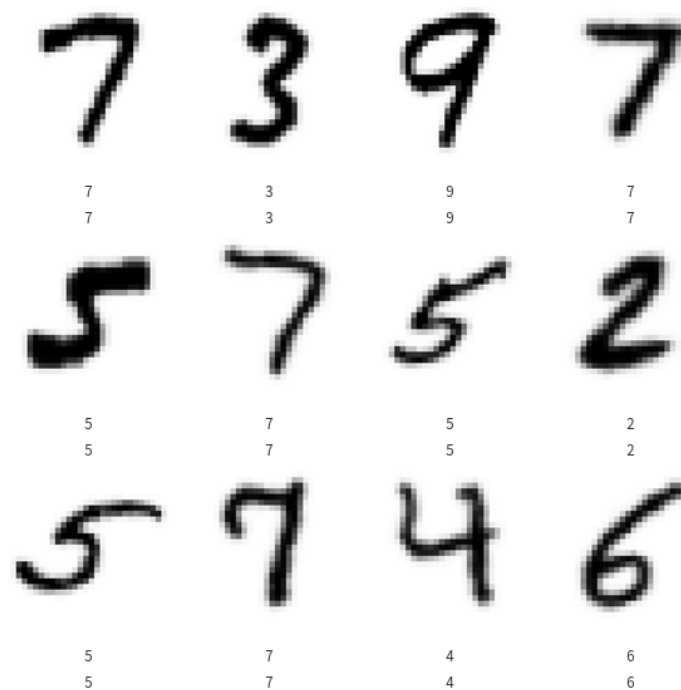
A plataforma web e o formulário podem ser acessados em https://share.streamlit.io/miguelsarraaf/teste_turing/teste_turing.py e <https://forms.gle/SdxrFshTGXbfYZj38>, respectivamente.

Teste de Turing

Segunda fase do Teste

Observe as imagens e as classificações feitas para elas.

Nas duas primeiras linhas abaixo da imagem estão as respostas dos competidores, Miguel Sarraf e Helena Lima. As respostas de cada competidor estão ou na linha de cima ou na de baixo.



Algum dos conjuntos de respostas foi dado por uma máquina?

Sim, a máquina deu as respostas de cima

Sim, a máquina deu as respostas de baixo

Não sei dizer, ambos são indistinguíveis

Não, ambos competidores são humanos

Voltar ao começo

Figura 27: Tela apresentada ao segundo jogador

5 RESULTADOS

Com tudo que foi desenvolvido e de acordo com o explicado anteriormente, foi possível obter as métricas que foram discutidas na seção 3.4 para os modelos finais dos estágios I e II. Nas seguintes seções serão apresentadas estas métricas, as observações feitas pelo formulário da plataforma web, bem como uma discussão sobre as implicações desses resultados.

5.1 Avaliação das métricas

Como foi discutido, diversos fatores devem ser considerados na avaliação dos modelos desenvolvidos e, para cada característica que se deseja observar, há uma métrica correspondente. Nesta seção serão analisadas as métricas apresentadas na seção 3.4 para os dois modelos finais encontrados.

Como dito e explicado, serão utilizadas as métricas *acuidade* e *pontuação f1* para observar o quão corretos estão os valores de saída dos modelos, a *matriz de confusão* para determinar qual o comportamento do modelo e quais classes ele mais confunde e a *pontuação de Brier modificada* para averiguar com quanta certeza o modelo determina suas classes de saída.

5.1.1 Acuidade e pontuação f1

Na tabela 1 estão apresentados os valores de *acuidade* e *pontuação f1* para os modelos treinados.

Como a *acuidade* só pode ser aplicada para todas as classes de uma vez, não se pode inferir nada mais do que o comportamento geral da rede. Já quando analisamos a *pontuação f1*, é possível discriminar os valores para cada uma das classes, possibilitando uma análise mais específica do comportamento das redes.

Como pode ser visto pela *acuidade* de exatamente 10%, o modelo do estágio I fez

Métricas		Estágio I	Estágio II
Acuidade		10.00%	99.31%
pont. f1	Classe 0	0.00%	99.62%
	Classe 1	18.18%	99.64%
	Classe 2	0.00%	99.31%
	Classe 3	0.00%	99.20%
	Classe 4	0.00%	99.25%
	Classe 5	0.00%	99.28%
	Classe 6	0.00%	99.49%
	Classe 7	0.00%	99.29%
	Classe 8	0.00%	99.07%
	Classe 9	0.00%	98.94%

Tabela 1: Valores de *acuidade* e *pontuação f1* para os modelos

exatamente o que se esperava dele, ou seja, apresentou um chute completo das respostas entre as 10 classes existentes.

Já o modelo do estágio II atingiu um valor muito maior, de 99.31%, acertando a classificação em quase todos os casos, indicando que, indiscutivelmente, houve alguma forma de aprendizado na rede desenvolvida.

Ao observar os valores da *pontuação f1* para o modelo do estágio I, percebe-se que ele falha em identificar todas as classes com exceção do dígito 1, para o qual ele tem um valor de quase 20%, o que ainda é bastante baixo.

Já, ao se observar os valores para o modelo do estágio II, os valores são bastante promissores, a pior classe, 9, tem valor de *f1* pouco inferior a 99%, e a melhor, 1, de 99.64%.

Esses números indicam que enquanto a segunda rede classifica corretamente as imagens tanto em relação às classes esperadas quanto em relação às classes obtidas, a primeira falha em pelo menos uma dessas tarefas.

5.1.2 Matriz de confusão

A fim de entender melhor a distribuição desses acertos e falhas das duas redes, pode-se analisar as matrizes de confusão destes modelos, presentes nas imagens [28](#) e [29](#).

Observando os resultados apresentados nessas matrizes de confusão, entende-se, por fim, os comportamentos mostrados pela *acuidade* e pela *pontuação f1*, uma vez que as saídas de cada rede estão discriminadas por classe.

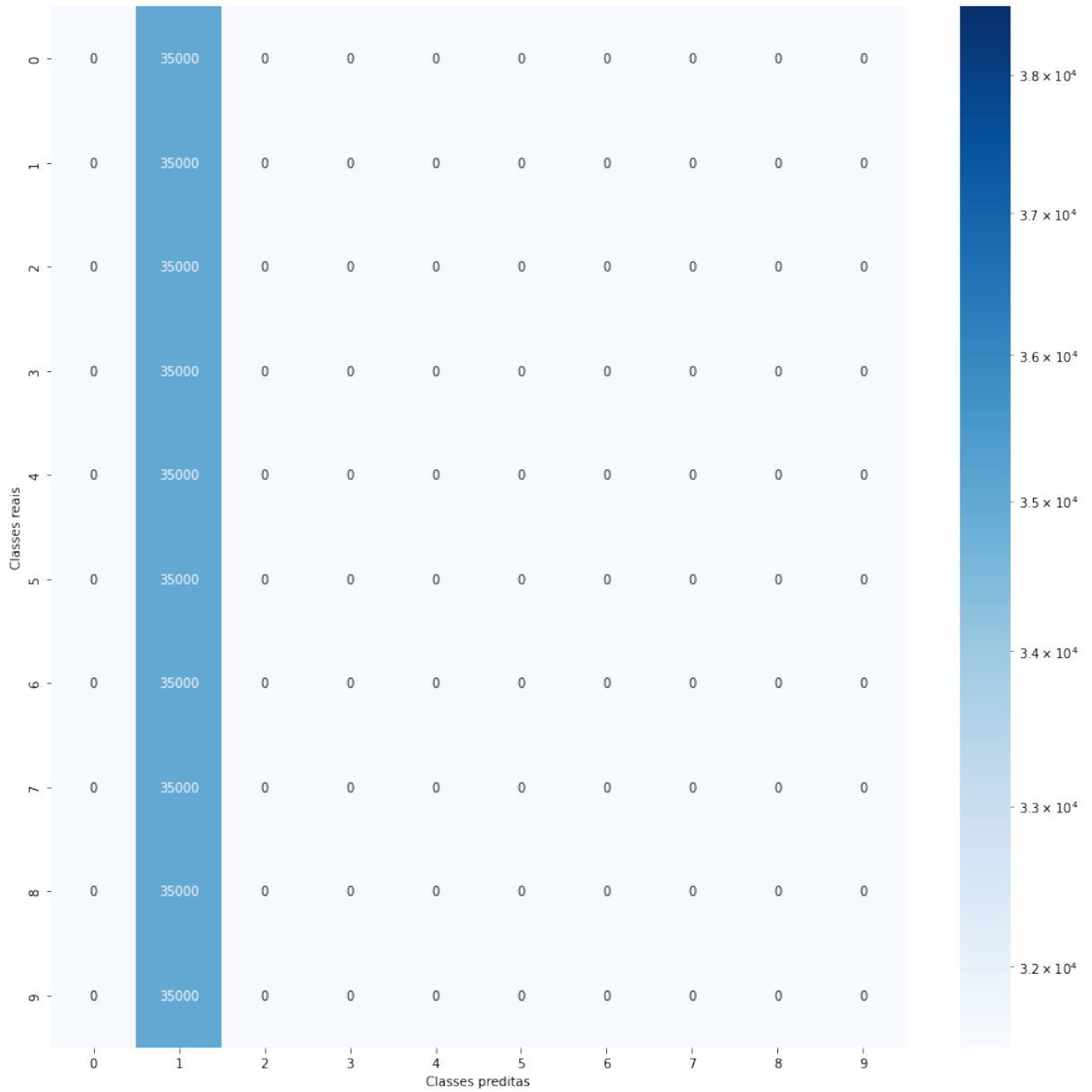


Figura 28: Matriz de confusão para o modelo do estágio I

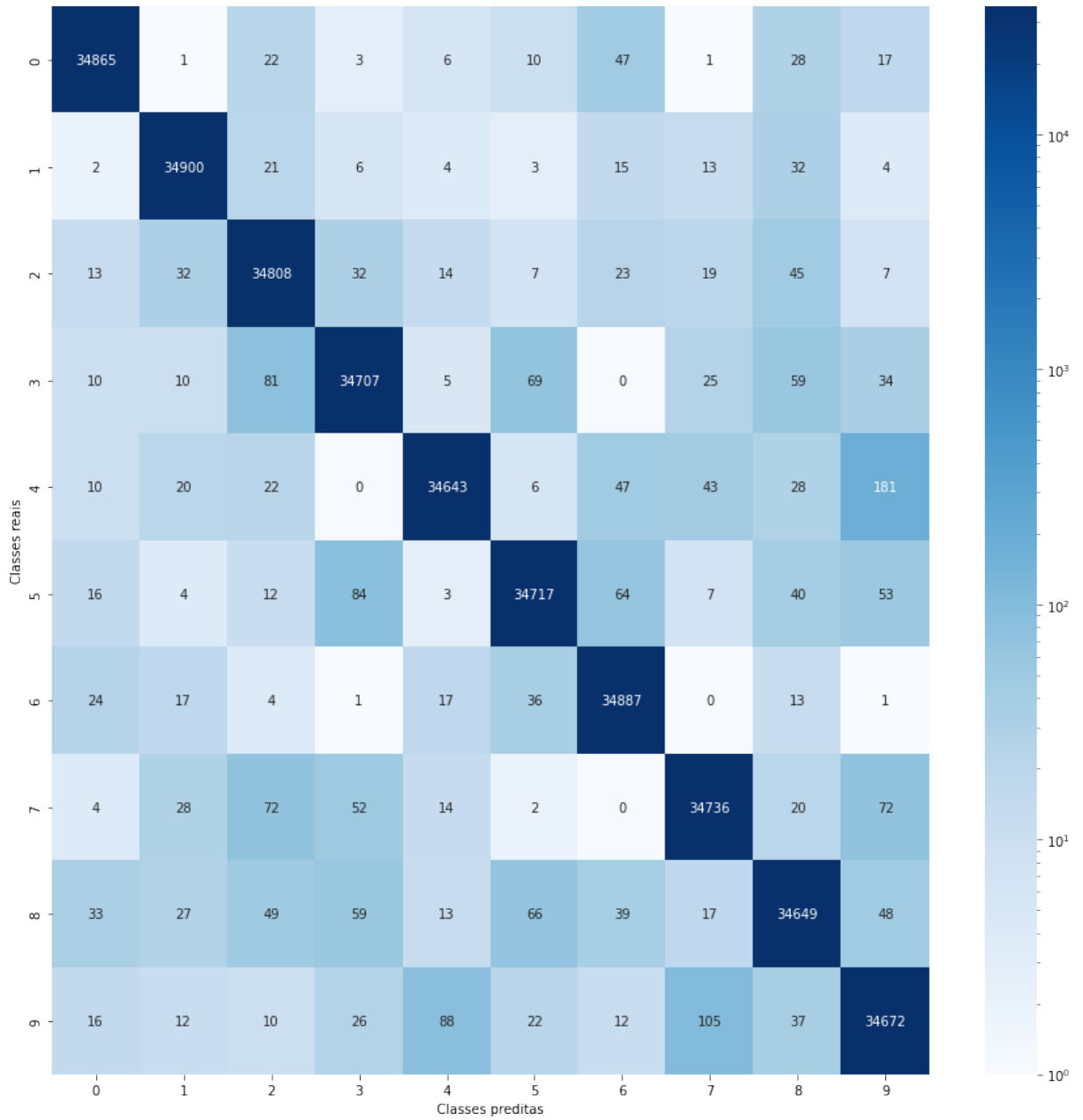


Figura 29: Matriz de confusão para o modelo do estágio II

Pela matriz de confusão do estágio I, na imagem [28](#), observa-se que a rede aprendeu a sempre identificar o dígito recebido como 1, isso explica o fato da *acuidade* ter sido exatamente 10%, já que os dados do dataset são balanceados entre as classes. Isto corrobora com a hipótese inicial, de que no estágio I a classificação dos dados seria uma escolha aleatória, todas as classes sendo interpretadas como um mesmo tipo de objeto.

Já quando se observa a matriz de confusão do estágio II, na imagem [29](#), observa-se, como entendido pelas métricas anteriores, uma alta concentração das combinações de classes reais e preditas na diagonal principal. A maior parte das outras posições da matriz está abaixo de 50 ocorrências, com destaque para os vários valores abaixo de 10, que mostram distribuição pouco concentrada dos erros e valores muito baixos desses.

É interessante perceber que os dígitos mais confundidos pelo modelo são o 4 e 9, seguidos pelo 7 e o 0, o que faz sentido devido à anatomia destes dígitos. Para cada par de dígitos esperado e predito diferentes e que tiveram mais de 50 ocorrências no dataset de treino para o modelo do estágio II, estão representadas duas imagens que causaram essa confusão nos anexos [A](#) a [N](#). Mais uma vez, esse resultado corrobora com a tese proposta, de que o estágio II deveria conseguir classificar bem os dados.

5.1.3 Pontuação de Brier modificada

Embora as métricas analisadas já tenham levado a conclusões bastante fortes e interessantes, elas deixaram de lado um aspecto fundamental a ser analisado, o quanto de certeza o modelo apresenta na saída escolhida. Para isso, foi analisada a *pontuação de Brier modificada* para cada classe e no caso geral, como mostrado na tabela [2](#).

Para efeito de comparação dos resultados obtidos, alguns dos valores analisados nessa seção serão traduzidos na distribuição de probabilidades equivalentes. Essas distribuições serão tais que terão uma classe com probabilidade máxima e todas as outras nove com distribuição uniforme (ver apêndice [B](#)).

Observando-se apenas os valores gerais para ambos os modelos já é muito relevante a diferença entre eles. Para o estágio I, o valor da *pontuação de Brier modificada* é extremamente alto, equivalendo ao valor que seria obtido com uma distribuição com probabilidade máxima de 10.32% e 9.96% nas outras, ou seja, muito próxima de uma distribuição uniforme.

Por outro lado, o valor obtido para o estágio II é extremamente baixo, equivalendo a uma distribuição com probabilidade máxima de 96.75% e todas as outras de 0.36%, ou

Métricas		Estágio I	Estágio II
pont. Brier	Geral	99.29%	0.13%
	Classe 0	100.00%	0.09%
	Classe 1	99.29%	0.08%
	Classe 2	100.00%	0.14%
	Classe 3	100.00%	0.14%
	Classe 4	100.00%	0.14%
	Classe 5	100.00%	0.15%
	Classe 6	100.00%	0.09%
	Classe 7	100.00%	0.14%
	Classe 8	100.00%	0.18%
	Classe 9	100.00%	0.20%

Tabela 2: Valores da *pontuação de Brier* para os modelos

seja, próximo a uma curva em *One-hot encoding*.

Ao avaliar os valores para cada classe desta métrica, apenas se reafirma o comportamento já encontrado. Para o estágio I, como a única classe que tem valores preditos é a do dígito 1, como visto na seção [5.1.2](#), a distribuição das outras classes é necessariamente uniforme, com valor 100%¹ e, para este dígito, o valor coincide com o global.

Já para o estágio II, observa-se que há pequena variação entre as classes, com 0.08% no melhor caso e 0.2% no pior, correspondendo a distribuições com máxima probabilidade de 97.45% e 95.97%, respectivamente.

Esses resultados indicam, ainda mais, que os modelos correspondem ao que era esperado pelas hipóteses levantadas. No estágio I, não apenas ocorre um chute aleatório entre as classes, como a certeza do modelo quanto a essa classificação é extremamente baixa, enquanto no estágio II o modelo faz a grande maioria das classificações corretamente e apresenta, para elas, uma enorme certeza.

5.2 Plataforma web

Terminado o desenvolvimento do website e aplicado o questionário, também foi possível obter alguns dados sobre o comportamento do modelo desenvolvido, em especial na sua relação com usuários reais, caso que era impossível de ser testado de outra forma.

O formulário desenvolvido teve um total de 35 respostas, sem que nenhum participante tenha se recusado a continuar a pesquisa depois de ler o curto termo de consentimento

¹Como para estes casos não há nenhum dado, o código retornou valor ∞ para a *pontuação de Brier modificada*, mas eles foram traduzidos para 100% por se tratar de uma distribuição uniforme

apresentado na primeira página.

Entre os participantes, aproximadamente metade utilizou um computador para responder e a outra metade utilizou um telefone celular. O uso praticamente igual das plataformas é um resultado interessante para averiguar a performance da estruturação do site nas diversas plataformas.

Numa análise geral das respostas às outras questões do formulário, foi possível perceber que algumas pessoas se confundiram com o significado de algumas perguntas e não perceberam detalhes sobre o uso da plataforma. Acredita-se que os poucos casos em que esses problemas ocorreram não tenham prejudicado o resultado geral da pesquisa, que será detalhado a seguir.

Esse efeito ficou muito evidente nos comentários finais, onde houve muitas respostas que diziam não ter conseguido identificar informações que estavam explicadas no texto das páginas ou que conseguiram identificar qual era a máquina pois tinham jogados sozinhos e sabiam quais respostas tinham dado, entre outros casos parecidos.

Apenas 6 entre os participantes não perceberam que havia outro jogador classificando as imagens supostamente ao mesmo tempo que eles, informação mostrada na aba lateral. A maioria destes participantes, porém, utilizou o celular para acessar o site e, nesta plataforma, a barra lateral não fica ativa por padrão, de forma que uma distração foi perdida nesses casos.

Apenas em 3 das respostas recebidas o primeiro jogador não foi capaz de classificar corretamente todas as imagens. Nesses casos os usuários devem ter encontrado algumas das imagens de difícil classificação, como explicado na seção [5.1.2](#).

Entre essas ocasiões de erro do primeiro jogador, apenas em uma o segundo jogador conseguiu identificar a máquina, mas isto ocorreu em um dos casos em que o jogador jogou sozinho e identificou a máquina por saber a resposta dada na primeira fase.

Na grande maioria dos casos, correspondendo a um total de 77.1% das respostas, o segundo jogador não foi capaz de identificar uma máquina entre as classificações que lhe foram apresentadas, como era desejado para o trabalho. Conjectura-se que este número seja ainda maior devido aos erros de interpretação mencionados anteriormente.

Numa observação geral, acredita-se que a pesquisa e o website possam ser considerados um sucesso em relação ao esperado e que serviram ao propósito de mostrar que o modelo desenvolvido é capaz de simular de forma convincente o comportamento de um ser humano, na maioria dos casos.

5.3 Implicações dos resultados

Todos os aspectos anteriores considerados, tanto as métricas de distribuição dos resultados quanto a relação do modelo com pessoas reais, é possível vislumbrar resultados mais profundos que apenas um bom modelo para classificação de imagens. Segue uma pequena discussão sobre esses resultados.

Primeiramente, deve-se ressaltar o grande sucesso do modelo proposto em classificar as imagens, tarefa que, embora pareça muito simples, na verdade é um grande desafio computacionalmente. Esse resultado, isoladamente, seria suficiente para considerar o estudo um sucesso.

Não apenas a rede do estágio II foi muito capaz de realizar a classificação como a do estágio I foi completamente incapaz de se sair melhor que uma escolha aleatória simples. Como já foi explicado, esse comportamento condiz totalmente com a teoria Piagetiana do desenvolvimento, que é uma das bases fundamentais da psicologia moderna. Essa concordância de resultados mostra que estes dois universos abordados podem coexistir e produzir resultados conjuntamente e de forma muito mais estruturada.

Além disso, é preciso considerar a fonte original da arquitetura do modelo elaborado. A inspiração para o modelo de processamento de dados visuais foi obtida do neurocientista Karl Friston, que trabalha em ainda outra área distinta do conhecimento.

Dessa forma, também é conclusão do trabalho a afirmação da hipótese sobre a possibilidade de relação entre as áreas funcionais do cérebro e os tipos de camadas de redes neurais. E mais do que isso, como esse modelo neurológico foi utilizado para representar as etapas do desenvolvimento, a correspondência feita entre as partes da rede e os estágios do desenvolvimento se provou verdadeira.

Por uma visão macroscópica desta ligação de abordagens, obtida pelos testes feitos na plataforma web, pôde-se observar um comportamento quase humano, na tarefa de classificação, da rede desenvolvida, sendo na maioria das vezes confundido com um jogador real.

Esse resultado potencialmente coloca em xeque os limites da superioridade humana sobre uma máquina, no sentido em que mostra um indistinguível do outro. Embora a aplicação ainda seja simples, essa característica do resultado continua válida e abre espaço para o desenvolvimento de casos mais avançados.

Tudo isso posto, o resultado final apresentado do projeto é um modelo que compila

três abordagens do mesmo problema de forma coesa e coerente com o contexto em que está inserida. Foram associadas, na mesma solução, as visões computacional, psicológica e neurológica do processo de aprendizagem e do processamento de informações, chegando a um modelo capaz de se passar por um ser humano real.

6 CONCLUSÃO

Utilizando-se dos métodos apresentados, acredita-se ter obtido uma modelagem computacional suficientemente simples, que é capaz de receber um conjunto de dados de entrada e de saída, “aprender” como converter um no outro e, posteriormente, replicar o processo sobre outro conjunto de dados.

Foi perceptível, durante todo o desenvolvimento do projeto, a enorme dificuldade que se enfrenta ao tentar misturar diversas disciplinas distintas. Além da evidente complicação em aprender conceitos e processos relativamente avançados de áreas que não se domina, as diferenças de vocabulário empregadas em cada área são muito evidentes, travando ainda mais o desenvolvimento.

Também foi evidente, por todo o decorrer do estudo, a resistência acadêmica à aceitação de projetos de natureza não convencional e com inclinações para fora das diretrizes usuais de suas áreas. Foi extremamente difícil estabelecer contato e troca de ideias com alguns docentes, que não concebiam a proposta multidisciplinar abordada.

Os resultados obtidos, tanto na execução das redes, como nas aplicações do Teste de Turing, foram muito surpreendentes, excedendo em muito as expectativas. O projeto iniciou com a proposta de desenvolvimento de uma prova de conceito, não sendo esperados acertos maiores que 60% para o modelo, e terminou com uma representação muito forte e eficaz na tarefa de classificação. Além disso, a alta taxa de confusão entre máquina e humano no Teste de Turing mostra que o processo desenvolvido tem capacidade de gerar resultados ainda mais importantes.

Entende-se que o trabalho pode ser considerado um sucesso na medida em que mostrou a validade da abordagem interdisciplinar do processo de aprendizagem, superando as expectativas de uma simples prova de conceito, como foi proposta. Abre-se, portanto, caminho para mais parcerias entre as áreas do conhecimento, em busca de modelos mais potentes, que sejam capazes de modelar aspectos ainda mais complexos do mundo.

No sentido de continuar o trabalho iniciado neste TCC, alguns caminhos e questões

devem ser consideradas. O desenvolvimento, treino e teste do estágio III de Piaget-Inhelder é uma frente que deve ser explorada, podendo possibilitar que o modelo consiga aprender a gerar inferências. Outra possibilidade é aumentar o tamanho do escopo das redes já desenvolvidas, englobando mais classes (e.g. as letras do alfabeto) nos dados disponibilizados nas entradas e saídas, criando uma rede mais generalista. Também é possível aprofundar mais o estudo das disciplinas de psicologia e neurociência, feitos muito superficialmente neste projeto, para aperfeiçoar os modelos, tornando-os mais condizentes com as teorias já desenvolvidas nas outras áreas.

REFERÊNCIAS

- 1 LUCAS, G. *Star Wars - episode II: Attack of the Clones*. [S.l.]: LucasFilm, 2002.
- 2 TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX, n. 236, p. 433–460, 10 1950. ISSN 0026-4423. Disponível em: <https://doi.org/10.1093/mind/LIX.236.433>.
- 3 SINGH, S. *O Último Teorema de Fermat: A história do enigma que confundiu as mais brilhantes mentes do mundo durante 358 anos*. [S.l.]: BestBolso, 2016. 270 p. ISBN 978-85-7799-428-1.
- 4 INHELDER, B.; PIAGET, J. *The growth of logical thinking from childhood to adolescence*. New York: Basic Books, 1958.
- 5 FRISTON, K. J.; HARRISON, L.; PENNY, W. Dynamic causal modelling. *NeuroImage*, v. 19, n. 4, p. 1273–1302, 2003.
- 6 MINSKY, M.; PAPERT, S. *Perceptrons*. [S.l.]: Cambridge, MA: MIT Press, 1969.
- 7 LUXTON, D. Artificial intelligence in psychological practice: Current and future applications and implications. *Professional Psychology: Research and Practice*, v. 45, 11 2013.
- 8 REV, P. et al. Artificial cognition: How experimental psychology can help generate explainable artificial intelligence. *Psychonomic Bulletin & Review*, v. 28, 11 2020.
- 9 TORTORA, G. J.; DERRICKSON, B. Principles of anatomy and physiology. In: _____. 11th ed.. ed. [S.l.]: Wiley & Sons,, 2006. cap. 14.
- 10 PORTNOI, M. *O Que É Engenharia?* 1999. Disponível em: <https://www.eecis.udel.edu/~portnoi/academic/academic-files/eng-whatisit.html>.
- 11 MCCORDUCK, P. *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. [S.l.]: American Psychological Association, 2004. ISBN 1568812051.
- 12 HOGAN, J. D. Encyclopedia of psychology. In: KAZDIN, A. E. (Ed.). *Developmental psychology: History of the field*. [S.l.]: AK Peters Ltd, 2000. v. 3, p. 9–13.
- 13 ZHENG, A.; CASARI, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. 1st. ed. [S.l.]: O’Reilly Media, Inc., 2018. ISBN 1491953241.
- 14 SHORTEN, C.; KHOSHGOFTAAR, T. A survey on image data augmentation for deep learning. *Journal of Big Data*, v. 6, p. 1–48, 2019.
- 15 AGGARWAL, C. C. *Neural Networks and Deep Learning: A textbook*. Cham: Springer, 2018. 497 p. ISBN 978-3-319-94462-3.

- 16 YU, D.; DENG, L. *Automatic Speech Recognition: A Deep Learning Approach*. [S.l.]: Springer Publishing Company, Incorporated, 2014. ISBN 1447157788.
- 17 BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.
- 18 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.
- 19 AUER, P.; HERBSTER, M.; WARMUTH, M. K. K. Exponentially many local minima for single neurons. In: TOURETZKY, D.; MOZER, M. C.; HASSELMO, M. (Ed.). *Advances in Neural Information Processing Systems*. MIT Press, 1996. v. 8. Disponível em: <https://proceedings.neurips.cc/paper/1995/file/3806734b256c27e41ec2c6bffa26d9e7-Paper.pdf>.
- 20 RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. [S.l.]: Prentice Hall, 2010.
- 21 SEARLE, J. R. *Minds, Brains and Science*. [S.l.]: Harvard University Press, 1984.
- 22 BARAD, J. Blade runner and sartre: The boundaries of humanity. In: _____. [S.l.: s.n.], 2007. p. 21–34.
- 23 PIAGET, J. *Seis estudos de Psicologia 24ª edição*. Rua do Rosário, 100, Rio de Janeiro, RJ, Brasil: Editora Forense Universitária, 1999. ISBN 8521802463.
- 24 BRODMANN, D. K. *Brodmann's Localisation in the Cerebral Cortex*. 233 Spring Street, New York, NY 10013, USA: Springer Science+Business Media, Inc., 2006. ISBN 78-0387-26917-7.
- 25 DRESCHER, G. L. Genetic ai: Translating piaget into lisp. *Instructional Science*, Springer, v. 14, n. 3/4, p. 357–380, 1986. ISSN 00204277, 15731952. Disponível em: <http://www.jstor.org/stable/23369064>.
- 26 ZHAOPING, L. *Understanding Vision: Theory, Models, and Data*. 198 Madison Avenue, New York, NY 10016, United States of America: Oxford University Press, 2014. ISBN 9780199564668.
- 27 JUNG-BEEMAN, M. et al. Neural activity when people solve verbal problems with insight. *PLoS Biology*, v. 2, n. 4, p. 500–510, 2004. Disponível em: <http://biology.plosjournals.org/perlserv/?request=get-document&doi=10.1371/journal.pbio.0020097>.
- 28 COHEN, G. et al. *EMNIST: an extension of MNIST to handwritten letters*. 2017.
- 29 JUNG, A. B. et al. *imgaug*. 2020. <https://github.com/aleju/imgaug>. Online; accessed 01-Feb-2020.
- 30 SILVA, R. M. O. Rodrigo Felix Lima Dias da. *Processamento e linguagem natural com análise de sentimentos e modelagem de tópicos para dados de redes sociais*. São Paulo: [s.n.], 2019.
- 31 Brier, G. W. Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, v. 78, n. 1, p. 1, jan. 1950.

APÊNDICE A – LIMITES DA PONTUAÇÃO DE BRIER

A *pontuação de Brier*, como exposto no texto principal, apresenta a similaridade entre duas funções discretas que, no caso específico, são duas funções massa de probabilidade.

Da forma como foi originalmente definido em [31], essa métrica realiza uma soma do erro quadrático médio entre as várias amostras e, para evitar o surgimento de números muito grandes, argumenta-se ser benéfica a substituição dessa soma por uma média.

$$P' = \frac{1}{M \cdot C} \sum_{i=1}^M \sum_{j=1}^C (\bar{v}_j^i - v_j^{Li})^2 \quad (\text{A.1})$$

Nessa fórmula, \bar{v}^i é um vetor em *One-hot encoding* representando a distribuição ideal do vetor com a distribuição de probabilidades predita, v_j^{Li} . Aqui serão analisados os extremos dessa métrica.

Por simplicidade, v_j^{Li} será chamado de p e \bar{v}^i de a . Entende-se que o valor mínimo ocorrerá quando $p = a$, que tem pontuação 0, já que, em todos os outros casos, a pontuação será positiva. Para calcular p que atinge pontuação máxima, será aplicada a técnica dos multiplicadores de Lagrange.

Admitindo os vetores $a = (0, 0, \dots, 1, \dots, 0) \in \mathbb{R}^C$, onde o valor 1 está na posição k , e $p = (p_1, p_2, \dots, p_C) \in \mathbb{R}^C \mid \sum_{i=1}^C p_i = 1 \wedge p_k \geq p_j \forall j \neq k$, a função a ser maximizada é a somatória interna da *pontuação de Brier*, que é o quadrado da diferença entre p e a , e a restrição é p ser um vetor de probabilidades.

$$f(p_1, \dots, p_C) = (a - p)^2 = (p_k - 1)^2 + \sum_{i=1, i \neq k}^C p_i^2 = \sum_{i=1}^C p_i^2 - 2p_k + 1 \quad (\text{A.2a})$$

$$\nabla f(p_1, \dots, p_C) = (2p_1, 2p_2, \dots, 2p_k - 2, \dots) \quad (\text{A.2b})$$

$$g_1(p_1, \dots, p_C) = \sum_{i=1}^C p_i - 1 = 0 \quad (\text{A.3a})$$

$$\nabla g_1(p_1, \dots, p_C) = (1, 1, \dots) \quad (\text{A.3b})$$

Resolvendo o lagrangiano:

$$\begin{aligned} \nabla \mathcal{L}(p_1, \dots, p_C, \lambda) &= \nabla f(p_1, \dots, p_C) - \lambda \nabla g_1(p_1, \dots, p_C) = 0 \\ (2p_1 - \lambda, 2p_2 - \lambda, \dots, 2p_k - 2 - \lambda, \dots) &= 0 \\ p_i &= \frac{\lambda}{2} \forall i \neq k \quad p_k = 1 + \frac{\lambda}{2} \end{aligned} \quad (\text{A.4})$$

Aplicando a restrição.

$$\sum_{i=1}^C p_i - 1 = 0 \rightarrow 1 + \frac{\lambda}{2} + (C-1)\frac{\lambda}{2} - 1 = 0 \rightarrow \frac{C\lambda}{2} = 0 \rightarrow \lambda = 0 \quad (\text{A.5})$$

Da onde se conclui que $p_i = 0 \forall i \neq k$ e que $p_k = 1$ é a única solução que extremiza a função e sabemos que é um mínimo. Portanto, o máximo da função está em algum ponto da borda do domínio, não considerado nas restrições. Para determiná-lo, deve-se aplicar outra restrição ao problema, isto será feito assumindo que uma das dimensões de p está no seu limite, ou seja, assumindo $p_l = p_k$. Com isso, obtém-se uma nova restrição.

$$g_2(p_1, \dots, p_C) = p_l - p_k = 0 \quad (\text{A.6a})$$

$$\nabla g_2(p_1, \dots, p_C) = (0, \dots, 1, \dots, -1, \dots) \quad (\text{A.6b})$$

Resolvendo o novo lagrangiano:

$$\begin{aligned} \nabla \mathcal{L}(p_1, \dots, p_C, \lambda) &= \nabla f(p_1, \dots, p_C) - \lambda_1 \nabla g_1(p_1, \dots, p_C) - \lambda_2 \nabla g_2(p_1, \dots, p_C) = 0 \\ (2p_1 - \lambda_1, 2p_2 - \lambda_1, \dots, 2p_l - \lambda_1 - \lambda_2, \dots, 2p_k - 2 - \lambda_1 + \lambda_2, \dots) &= 0 \\ p_i &= \frac{\lambda_1}{2} \forall i \neq l, k \quad p_l = \frac{\lambda_1 + \lambda_2}{2} \quad p_k = 1 + \frac{\lambda_1 - \lambda_2}{2} \end{aligned} \quad (\text{A.7})$$

Aplicando a restrição g_1 .

$$\sum_{i=1}^C p_i - 1 = 0 \rightarrow \frac{\lambda_1 + \lambda_2}{2} + 1 + \frac{\lambda_1 - \lambda_2}{2} + (C-2)\frac{\lambda_1}{2} - 1 = 0 \rightarrow \frac{C\lambda_1}{2} = 0 \rightarrow \lambda_1 = 0 \quad (\text{A.8})$$

Aplicando a restrição g_2 .

$$p_l - p_k = 0 \rightarrow \frac{\lambda_1 + \lambda_2}{2} - 1 - \frac{\lambda_1 - \lambda_2}{2} = 0 \xrightarrow{\lambda_1=0} \frac{\lambda_2}{2} - 1 + \frac{\lambda_2}{2} = 0 \rightarrow \lambda_2 = 1 \quad (\text{A.9})$$

Da onde se conclui que $p_i = 0 \forall i \neq l, k$ e que $p_l = p_k = 0.5$. Vê-se, portanto, que o ponto de mínimo com essa restrição se dá quando p está o mais distante possível de uma distribuição uniforme e o ponto de máximo estará em algum ponto do resto da borda não considerada do domínio.

Se forem adicionadas mais restrições, até que sobre uma única variável, o resultado se repetirá com a probabilidade total dividida entre $p - 1$ dimensões. Neste caso, o valor mínimo será quando está última variável tiver valor 0 e o máximo será no extremo oposto, ou seja, quando p for uma distribuição uniforme.

No melhor caso, $p = a$ e não há erros no modelo de predição.

$$P'_{min} = \frac{1}{M \cdot C} \sum_{i=1}^M \sum_{j=1}^C (a - p)^2 = 0 \quad (\text{A.10})$$

No pior caso, $p = (\frac{1}{C}, \frac{1}{C}, \dots, \frac{1}{C})$ e o modelo se comporta como um chute aleatório.

$$\begin{aligned} P'_{max} &= \frac{1}{M \cdot C} \sum_{i=1}^M \sum_{j=1}^C (a - p)^2 = \frac{1}{M \cdot C} \sum_{i=1}^M \left(\left(1 - \frac{1}{C}\right)^2 + (C-1)\left(\frac{1}{C}\right)^2 \right) = \\ &= \frac{M}{M \cdot C} \left(1 - \frac{2}{C} + \frac{1}{C^2} + \frac{1}{C} - \frac{1}{C^2}\right) = \frac{1}{C} \left(1 - \frac{1}{C}\right) = \frac{1}{C} - \frac{1}{C^2} \end{aligned} \quad (\text{A.11})$$

E como, no caso específico, $C = 10$, o valor de da *pontuação de Brier* para o pior cenário é $P = 0.09$. Assim, propõe-se acrescentar o fator $\frac{1}{0.09}$ na fórmula a ser aplicada para que os valores variem entre 0 e 1.

$$P_{modificado} = \frac{1}{0.09 \cdot M \cdot C} \sum_{i=1}^M \sum_{j=1}^C (a - p)^2 = \frac{1}{0.09 \cdot M \cdot C} \sum_{i=1}^M \sum_{j=1}^C (\bar{v}_j^i - v_j^{Li})^2 \quad (\text{A.12})$$

APÊNDICE B – DISTRIBUIÇÃO DE PROBABILIDADES EQUIVALENTE

Dado um valor de *pontuação de Brier modificada*, deseja-se determinar uma distribuição de probabilidades equivalente que gera aquele valor da *pontuação*. Esse distribuição, por definição, terá um valor máximo em uma das classes, a que seria predita, e em todas as outras o mesmo valor, numa distribuição uniforme.

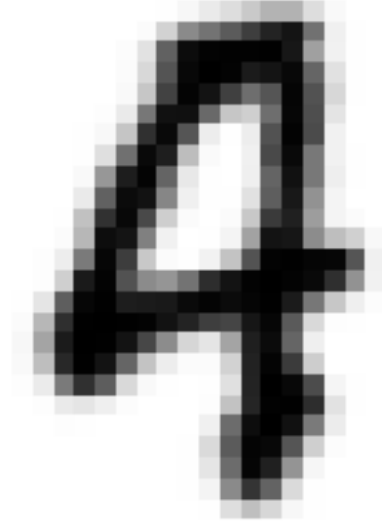
Assim, se a probabilidade máxima dessa distribuição for p' ($0 \leq p' \leq 1$), as outras nove classes têm probabilidade $\bar{p}' = \frac{1-p'}{9}$. Aplicando esses valores na fórmula da *pontuação de Brier modificada*, encontra-se o valor de p' em função de $P_{modificada}$.

$$\begin{aligned}
 P_{modificado} &= \frac{1}{0.09 \cdot M \cdot C} \sum_{i=1}^M \sum_{j=1}^C (a - p)^2 = \frac{1}{0.09 \cdot 1 \cdot 10} \sum_{j=1}^{10} (a - p)^2 = \\
 &= \frac{1}{0.09 \cdot 10} \left((1 - p')^2 + 9 \cdot \left(\frac{1 - p'}{9} \right)^2 \right) = \frac{1}{0.09 \cdot 10} \left((1 - p')^2 + \frac{(1 - p')^2}{9} \right) = \\
 &= \frac{1}{0.09 \cdot 10} \frac{10}{9} (1 - p')^2 = \frac{1}{0.81} (1 - p')^2 \rightarrow \frac{1 - p'}{0.9} = \sqrt{P_{modificado}}
 \end{aligned} \tag{B.1}$$

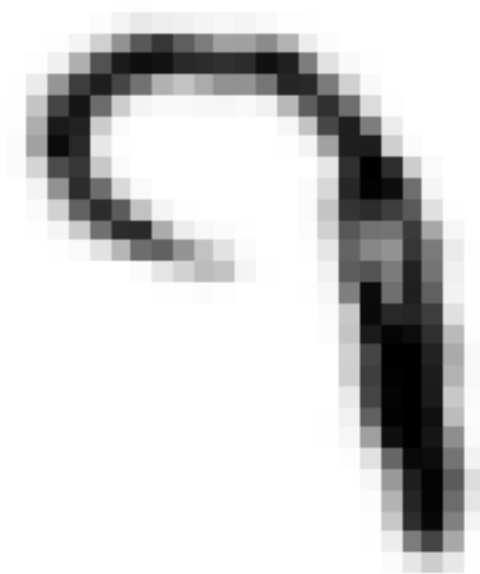
E, portanto:

$$p' = 1 - 0.9 \cdot \sqrt{P_{modificado}} \qquad \bar{p}' = 0.1 \cdot \sqrt{P_{modificado}} \tag{B.2}$$

ANEXO A – DÍGITOS 4 CLASSIFICADOS COMO 9



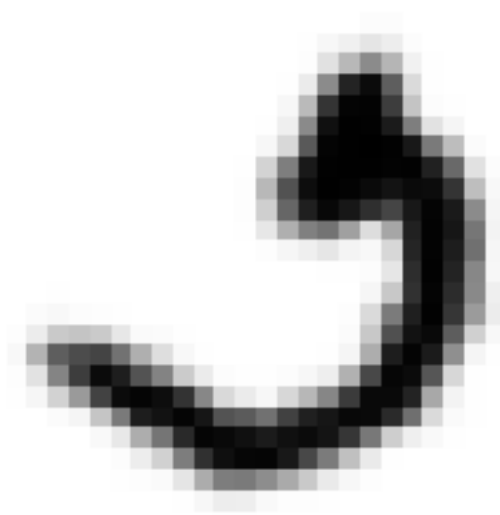
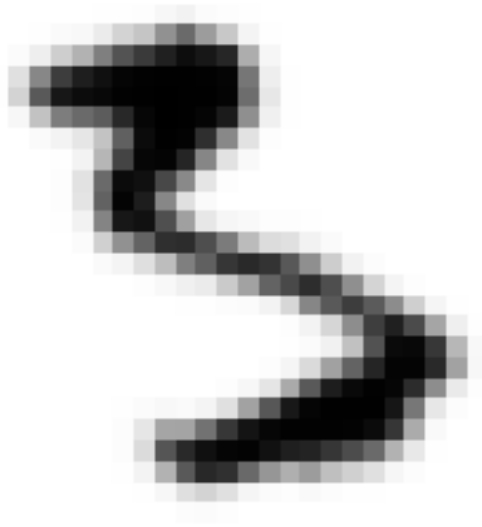
ANEXO B – DÍGITOS 9 CLASSIFICADOS COMO 7



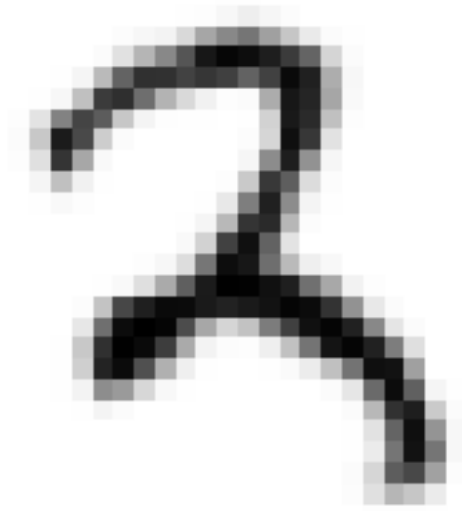
ANEXO C – DÍGITOS 9 CLASSIFICADOS COMO 4



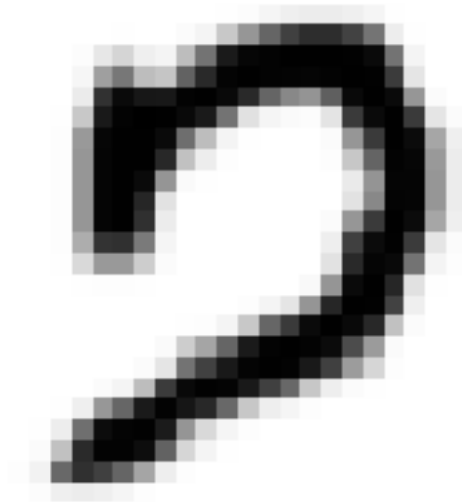
ANEXO D – DÍGITOS 5 CLASSIFICADOS COMO 3



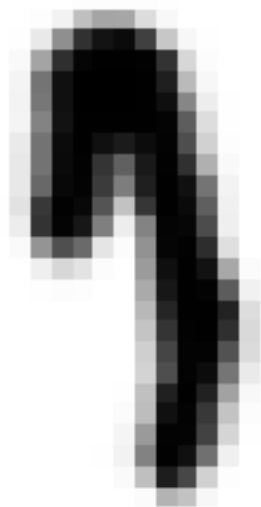
ANEXO E – DÍGITOS 3 CLASSIFICADOS COMO 2



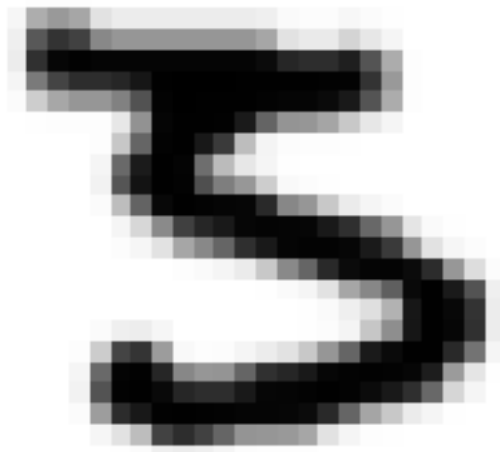
ANEXO F – DÍGITOS 7 CLASSIFICADOS COMO 2



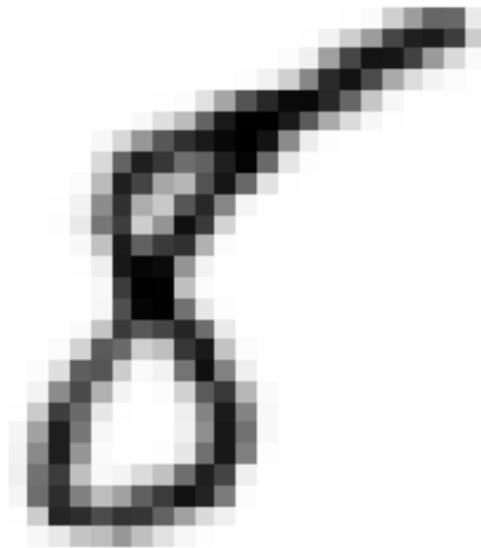
ANEXO G – DÍGITOS 7 CLASSIFICADOS COMO 9



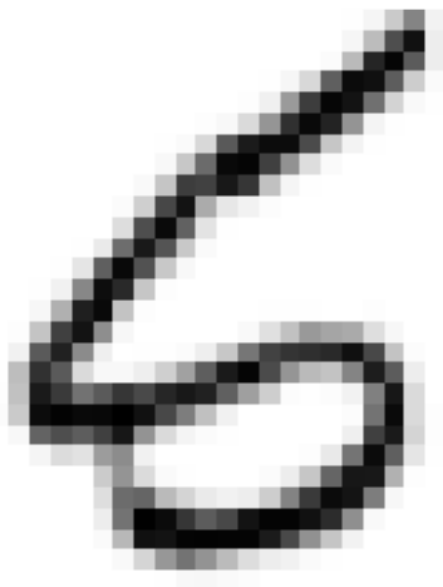
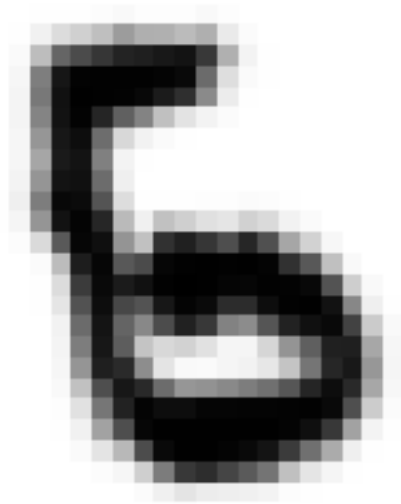
ANEXO H – DÍGITOS 3 CLASSIFICADOS COMO 5



ANEXO I – DÍGITOS 8 CLASSIFICADOS COMO 5



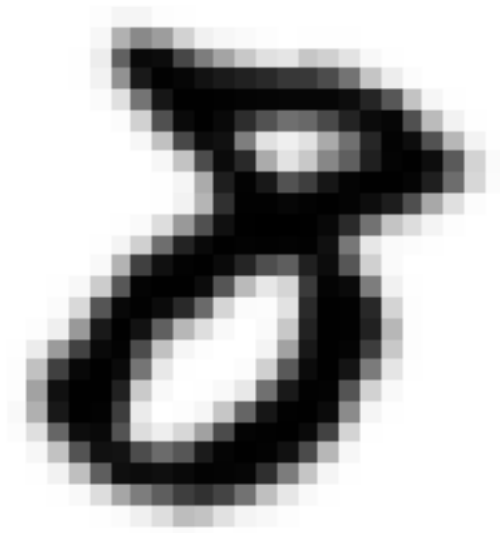
ANEXO J – DÍGITOS 5 CLASSIFICADOS COMO 6



ANEXO K – DÍGITOS 3 CLASSIFICADOS COMO 8



ANEXO L – DÍGITOS 8 CLASSIFICADOS COMO 3



ANEXO M – DÍGITOS 5 CLASSIFICADOS COMO 9



ANEXO N – DÍGITOS 7 CLASSIFICADOS COMO 3

