MARCOS WENNETON VIEIRA DE ARAUJO

VERIFICAÇÃO DA AUTENTICIDADE DE ASSINATURAS MANUSCRITAS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Orientador(a): Profa. Dra. Elloá B. Guedes

Manaus – Novembro – 2019

MARCOS WENNETON VIEIRA DE ARAUJO

VERIFICAÇÃO DA AUTENTICIDADE DE ASSINATURAS MANUSCRITAS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Aprovado em: 06/12/2019

BANCA EXAMINADORA

Profa. Dra. Elloá B. Guedes

UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. Dr. Luis Cuevas Rodriguez

UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. M.Sc. Rodrigo Tavares Teixeira

UNIVERSIDADE DO ESTADO DO AMAZONAS

Resumo

Este trabalho apresenta uma proposta para a verificação da autenticidade de assinaturas manuscritas feitas por indivíduos utilizando conceitos de *Deep Learning* por meio da aplicação de redes neurais convolucionais. Neste contexto, o problema da verificação de autenticidade de assinaturas manuscritas é tratado como um problema de classificação binária, em que os exemplos dispostos como entrada para os modelos consistiam em imagens compostas por duas assinaturas, uma de referência e outra para inferência. Para a tarefa proposta foram conduzidas abordagens de treinamento e teste de diversos modelos de redes neurais convolucionais, sujeitas à variações de hiperparâmetros. Os resultados mostraram que os melhores modelos para esta tarefa foram baseados na arquitetura LeNet, para a primeira abordagem, e MobileNet, na segunda abordagem. Estes modelos obtiveram um valor de 88.56% e 98.65% de acurácia, respectivamente, evidenciando o potencial de adequação dos modelos propostos para tarefas dessa natureza.

Palavras-chave: Verificação de autenticidade, Redes Neurais Convolucionais, *Deep Learning*

Agradecimentos

Agradeço primeiramente a Deus, com Ele ao meu lado eu sou forte, confiante e não tenho medo de nada e ninguém.

Agradeço o grande e total apoio de minha mãe, Gessy Vieira dos Santos. Sem seu sustento, conselhos e incentivos eu, definitivamente, não conseguiria alcançar todas as coisas nas quais deposito o meu orgulho hoje.

Agradeço também aos outros membros da minha família, minha irmã Denize, minha sobrinha Emanuella, meu pai Walter, minha avó Antônia, minha tia Marionete e minha prima Jamille, que sempre me fizeram acreditar no meu potencial e me ajudaram a ser uma pessoa melhor.

Agradeço à incrível Profa. Dra. Elloá B. Guedes, por ter confiado na minha capacidade e aceitado me orientar neste trabalho. Esta professora serviu como uma grande base de inspiração e aprendizado. Agradeço pelos conhecimentos aprendidos durante o desenvolvimento deste trabalho e durante todas disciplinas ministradas em que tive o prazer de participar.

Aos meus colegas de curso que estiveram comigo durante toda essa caminhada. Agradeço à Gillane, Karen e Syra que me ajudaram nos primórdios da minha experiência acadêmica. Agradeço de coração aos colegas do DNCG que acabaram se tornando meus grandes amigos. Nicoli, Rafaela, Giovana, Janderson, Luiz, Lucas, Leo, Miranda e Emanuel, sem os nossos momentos de descontração teria sido muito mais difícil aguentar toda a pressão.

Agradeço ao Instituto de Desenvolvimento Tecnológico (INDT) por me proporcionar uma

experiência profissional digna e, principalmente, por permitir a associação de meus estudos às minhas atividades ali desenvolvidas. Agradeço o apoio moral e a compreensão de meus colegas de trabalho. Um agradecimento especial a todos os membros do Lanchus, com eles meus dias se tornam muito mais divertidos.

Agradeço a todos os professores do Núcleo de Computação com os quais tive a oportunidade de aprender, que os ensinamentos absorvidos por mim através deles, possam ser passados para vários outros grandes talentos. Agradeço à Universidade do Estado do Amazonas, seus servidores e alunos. Agradeço ao Governo do Estado do Amazonas por garantir subsídios à esta instituição de ensino e, desta maneira, auxiliar o crescimento da nossa sociedade.

Agradeço à Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) que, por meio dos Projetos PROTI Pesquisa e PPP 04/2017, colaborou para a consolidação da infraestrutura física e tecnológica do Laboratório de Sistemas Inteligentes da Escola Superior de Tecnologia da Universidade do Estado do Amazonas. Este trabalho de conclusão de curso é um dos produtos destes projetos, pois foi desenvolvido no referido laboratório, fez uso dos recursos computacionais ali disponíveis e foi melhorado graças às discussões e interações com o grupo de pesquisa nele sediado.

Erga-se como o sol e batalhe até o trabalho estar pronto.

Brandon Flowers, Daniel Lanois

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A autenticação possui importância fundamental na segurança de sistemas computacionais, pois consiste em permitir ou negar acesso à certas informações ou serviços com base na identidade associada à entidade que solicita acesso ao recurso ou em algum atributo que depende desta identidade (COSTA; OBELHEIRO; FRAGA, 2006).

A Biometria, em particular, tem sido uma das técnicas mais difundidas para autenticação, já sendo utilizada de forma abrangente na rotina diária da população em geral. Ela é definida como a utilização de características fisiológicas ou traços comportamentais para a comprovação da identidade de um indivíduo. A autenticação por biometria ganhou muita popularidade como uma alternativa confiável para sistemas baseados em segurança por chave, devido suas propriedades únicas e a capacidade quase nula de cópia, roubo ou adivinhação (KHOLMATOV, 2003).

Dentre as técnicas de autenticação por Biometria, tem-se aquelas baseadas em características fisiológicas do indivíduo, as quais levam em conta, por exemplo, as impressões digitais, o exame de fundo de retina, a palma da mão e até a arcada dentária. Estas técnicas são muito seguras, mas ainda incorrem em problemas operacionais que dificultam o seu uso em larga escala, como o custo com *hardware* e o grau de intrusão elevado aos usuários na captura destas características (HEINEN, 2002). Existem também técnicas de autenticação biométrica que utilizam-se de traços comportamentais como, por exemplo, expressões faciais, gestos, assinaturas manuscritas, modo de andar, dentre outras, as quais são características dinâmicas e que podem variar

fortemente ao longo do tempo. Porém, apesar destes desafios, estas características podem ter seus padrões capturados mediante experiência (COSTA; OBELHEIRO; FRAGA, 2006).

A assinatura manuscrita, em especial, é particularmente utilizada para autenticação biométrica de identidade desde os tempos primórdios. Ela é caracterizada pela produção, de próprio punho, de uma marca referente ao nome ou rubrica do autor como uma prova de sua identidade (HEINEN, 2002). Nos sistemas biométricos de autenticação, apresenta vantagens em relação às senhas, por exemplo, por possuir reprodução não-trivial. Citam-se como suas vantagens os aspectos não-invasivos para a sua obtenção, diferentemente, por exemplo, da análise de certas características fisiológicas, e também o baixo custo de aquisição, o que colabora para sua ampla difusão (HEINEN; OSóRIO, 2004; SOUZA; PANTOJA; SOUZA, 2009).

O maior desafio das técnicas de autenticação de assinaturas é determinar se uma assinatura em questão é, de fato, escrita por quem afirma ser e se falsificações podem ser identificadas. Apesar de todas as vantagens previamente mencionadas desta informação biométrica, a criação de métodos automáticos para autenticação de assinaturas manuscritas não é uma tarefa trivial, pois, ao contrário das características biométricas fisiológicas, os padrões ali existentes podem apresentar grande variabilidade para um mesmo indivíduo (HEINEN, 2002). Levando em conta estes desafios, a literatura já contempla diversos métodos com vistas a endereçar esta tarefa, nos quais bons resultados foram encontrados principalmente utilizando Máquinas de Vetores de Suporte, Modelos Escondidos de Markov, Análise do Componente Principal, *Dynamic Time Warping*, dentre outros (SOUZA; PANTOJA; SOUZA, 2009).

Porém, com o crescente desenvolvimento de técnicas de *Deep Learning* aplicadas à problemas de Visão Computacional, houve o desejo de investigar o desempenho de tais técnicas na autenticação de assinaturas, problema que está sendo considerado como escopo deste trabalho. Esta subárea do Aprendizado de Máquina, inserida no escopo da Inteligência Artificial, baseia-se na proposição de modelos que aprendem a partir da experiência, sendo muito aplicados em problemas de classificação, detecção, localização e segmentação de objetos em imagens, com muitos resultados expressivos em diversos domínios (KHAN et al., 2018). Assim, almeja-se investigar as capacidades de tais modelos, especialmente baseados no uso de Redes Neurais

Convolucionais, na identificação de assinaturas autênticas e forjadas de diversos indivíduos.

1.1 Objetivos

O objetivo geral deste trabalho consistiu em verificar a autenticidade de assinaturas manuscritas com redes neurais convolucionais. Para alcançar esta meta, alguns objetivos específicos precisaram ser consolidados, a citar:

1. Realizar a fundamentação teórica acerca dos conceitos das redes neurais convolucionais;
2. Consolidar uma base de dados representativa de assinaturas;
3. Descrever o problema considerado como uma tarefa de Aprendizado de Máquina;
4. Propor, treinar e testar redes neurais convolucionais para a tarefa considerada;
5. Analisar os resultados obtidos.

1.2 Justificativa

Apesar da capacidade tecnológica atual e da proposição crescente de diversas características e métodos para autenticação biométrica, as assinaturas manuscritas ainda possuem um papel importante em nossa sociedade, estando presentes na ampla maioria dos documentos oficiais do País e servindo também para comprovação de diversas transações financeiras. Outro aspecto que ressalta a importância deste trabalho consiste na possibilidade da adoção dos modelos elaborados na verificação de autenticidade de documentos históricos ou artísticos, colaborando também na diminuição de fraudes nestes âmbitos. Assim, é importante a contínua proposição, melhoria e avaliação de métodos para este fim, com vistas a aumentar a eficiência e diminuir as eventuais vulnerabilidades.

Considerando que as técnicas de *Deep Learning* ainda são emergentes, é importante propor trabalhos que possam ajudar a verificar a adequação destas soluções, indicando vantagens e limitações, bem como comparações com o estado da arte.

No mais, do ponto de vista do bacharel em Engenharia de Computação em formação, a proposta de trabalho de conclusão de curso corrobora para a prática de conceitos, tecnologias e métodos de uma área emergente do Aprendizado de Máquina que é o *Deep Learning*. Por fim, deve-se mencionar a importância da realização deste trabalho com vistas a colaborar com as atividades desenvolvidas pelo *Laboratório de Sistemas Inteligentes* (LSI), uma iniciativa do *Grupo de Pesquisas em Sistemas Inteligentes* da Escola Superior de Tecnologia (EST) da Universidade do Estado do Amazonas (UEA).

1.3 Metodologia

Para alcançar os objetivos propostos no escopo deste trabalho, a condução das atividades que foram realizadas obedeceram à metodologia descrita a seguir:

1. Estudo dos conceitos relacionados ao Aprendizado de Máquinas, Redes Neurais Convolucionais e *Deep Learning*;
2. Descrição do problema considerado como uma tarefa de Aprendizado de Máquina;
3. Consolidação de uma base de dados representativa de assinaturas originais e forjadas;
4. Levantamento do ferramental tecnológico para implementação das redes neurais convolucionais;
5. Proposição de modelos de redes neurais convolucionais para o problema considerado, contemplando arquitetura, parâmetros e hiperparâmetros;
6. Treino das redes propostas para a tarefa de aprendizado considerada;
7. Teste das redes previamente treinadas com vistas a coleta de métricas de desempenho;
8. Análise dos resultados e identificação dos modelos mais adequados para o problema considerado;
9. Escrita da proposta de Trabalho de Conclusão de Curso;

10. Defesa da proposta de Trabalho de Conclusão de Curso;

11. Escrita do Trabalho de Conclusão de Curso; e

12. Defesa do Trabalho de Conclusão de Curso.

1.4 Cronograma

Considerando as atividades enumeradas na metodologia, a Tabela 1.1 sintetiza o cronograma de execução deste trabalho.

Tabela 1.1: Cronograma de atividades levando em consideração os dez meses (de 02/2019 a 12/2019) para a realização do TCC.

	2019										
	02	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**
Atividade 1	X	X	X								
Atividade 2		X									
Atividade 3		X	X								
Atividade 4			X								
Atividade 5				X	X	X	X				
Atividade 6				X	X	X	X				
Atividade 7							X	X			
Atividade 8									X	X	
Atividade 9	X	X	X	X	X						
Atividade 10					X						
Atividade 11						X	X	X	X	X	X
Atividade 12											X

1.5 Organização do Documento

Para a apresentação deste trabalho de conclusão de curso, este documento encontra-se dividido nas seções a seguir. O Capítulo 2 relaciona os fundamentos teóricos que se fizeram necessários para a resolução do problema apresentado. No Capítulo 3 é descrita uma análise de trabalhos relacionados. O capítulo 4, por sua vez, discorre sobre a solução proposta para a verificação da

autenticidade de assinaturas manuscritas, que é seguida pelos resultados e discussões considerando estas soluções, e estão relatados no Capítulo 5. Por fim, no Capítulo 6, encontram-se as considerações finais sobre a realização deste trabalho.

Capítulo 2

Fundamentação Teórica

A fundamentação teórica para a elaboração deste trabalho consiste em conceitos relativos ao Aprendizado de Máquina. Primeiramente, os conceitos gerais desta área serão apresentados na Seção 2.1, seguidos pelas Redes Neurais Artificais, na Seção 2.2, um dos modelos inferenciais mais representativos. As definições elementares da subárea de Aprendizado de Máquina conhecida como *Deep Learning* são apresentadas na Seção 2.3. A Seção 2.3.1 discorre sobre as características das Redes Neurais Convolucionais que, por fim, é seguida pela Seção 2.3.2 na qual são apresentadas algumas de suas arquiteturas canônicas.

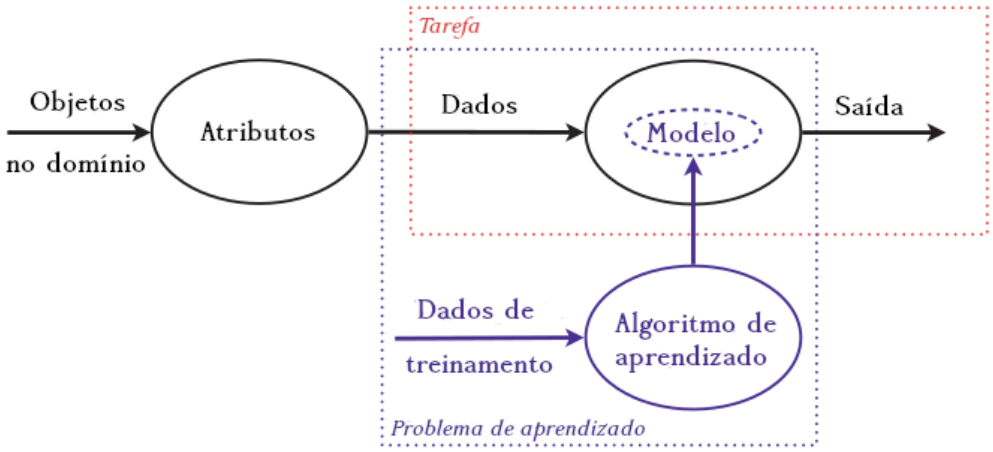
2.1 Aprendizado de Máquina

Aprendizado de Máquina (AM), do inglês *Machine Learning*, é o estudo sistemático de algoritmos e sistemas que melhoram seu conhecimento ou desempenho com o uso da experiência (FLACH, 2012). Em 1959, o pioneiro em jogos de computador Arthur Samuels definiu AM como um "campo de estudos que dá aos computadores a habilidade de aprender sem serem explicitamente programados" (SIMON, 2013). De acordo com Murphy (MURPHY, 2012) , AM pode ainda ser definido como um conjunto de métodos que conseguem detectar automaticamente padrões em dados e, em seguida, utilizar estes padrões para predizer dados não previamente vistos ou para realizar outros tipos de decisão mediante incerteza.

A essência dos métodos de AM consiste em utilizar os atributos corretos para construir os

modelos certos que resolvem determinadas tarefas (FLACH, 2012). Os atributos são dados oriundos dos objetos relevantes no domínio do problema. Com eles, efetua-se o treinamento de um modelo para resolver um problema. Este problema é representado abstratamente por uma tarefa. Ao final do treinamento, então, o modelo é usado para endereçar a tarefa proposta, colaborando na resolução do problema original. Estas ideias são ilustradas na Figura 2.1.

Figura 2.1: Uma visão geral de como o AM é utilizado para endereçar uma tarefa. Adaptado de: (FLACH, 2012).



O AM é comumente dividido em três paradigmas principais de aprendizado, chamados de aprendizado supervisionado, não-supervisionado e semi-supervisionado. No caso dos algoritmos de aprendizado supervisionado, o objetivo é aprender um mapeamento de entradas para saídas, dado um conjunto rotulado de pares de entradas e saídas. No aprendizado não supervisionado, o algoritmo é apresentado somente aos dados de entrada e o seu propósito é encontrar padrões significativos nos mesmos. O aprendizado semi-supervisionado, por sua vez, normalmente combina uma pequena quantidade de dados rotulados com uma grande quantidade de dados não rotulados para criar um classificador próprio a ser aplicado aos dados não rotulados. Em alguns casos, a abordagem de aprendizado semi-supervisionado pode ser de grande valor prático (KHAN et al., 2018).

No caso do paradigma de aprendizado supervisionado, em particular, destacam-se as tarefas de classificação e de regressão. Em uma tarefa de classificação, um algoritmo é selecionado para especificar quais das k categorias possíveis uma entrada pertence. Para resolver essa

tarefa, o algoritmo de aprendizado normalmente produz uma função $f : \mathbb{R}^n \rightarrow \{1, \ldots, k\}$. Quando $y = f(x)$, isto significa que o modelo mapeia uma entrada descrita pelo vetor $x \in \mathbb{R}^n$ para uma categoria identificado por um valor numérico $y \in \{1, \ldots, k\}$. Quanto à tarefa de regressão, é solicitado a um algoritmo de AM a predição de um valor numérico a partir de uma entrada. Desta forma, o algoritmo de aprendizado é proposto a inferir uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (GOODFELLOW; BENGIO; COURVILLE, 2016).

Dentre os diversos modelos de AM existentes, Flach considera a categorização dos mesmos segundo os tipos geométricos, probabilísticos e lógicos (FLACH, 2012). Um modelo geométrico é construído diretamente em função do espaço da solução, utilizando-se de conceitos como linhas, planos, hiperplanos e distâncias. Nesta categoria encontram-se a regressão linear, as redes neurais artificiais e as máquinas de vetores de suporte, por exemplo. Nos modelos do tipo probabilístico, tendo como exemplo o classificador Bayesiano, a questão principal é modelar a relação entre os dados de entrada e de saída assumindo que existe algum processo aleatório implícito que produz os valores para essas variáveis, de acordo com uma distribuição de probabilidade bem definida, porém desconhecida. Um modelo lógico, por sua vez, é o mais naturalmente algorítmico, considerando a capacidade de ser facilmente transformado em regras que podem ser entendidas por seres humanos. Dentre os modelos lógicos estão, por exemplo, as árvores de decisão e as florestas aleatórias.

Existe uma grande quantidade de tarefas que podem ser resolvidas com AM, entre estas podemos citar, por exemplo, o reconhecimento de objetos em uma imagem (PATHAK; PANDEY; RAUTARAY, 2018), a determinação da idade de um indivíduo em uma fotografia (ARAUJO, 2018), a classificação de atividades humanas (LIRA et al., 2017), entre outras. Na próxima seção serão descritas e apresentadas as redes neurais artificias, um dos modelos de AM para o paradigma supervisionado com papel protagonista nas soluções apresentadas.

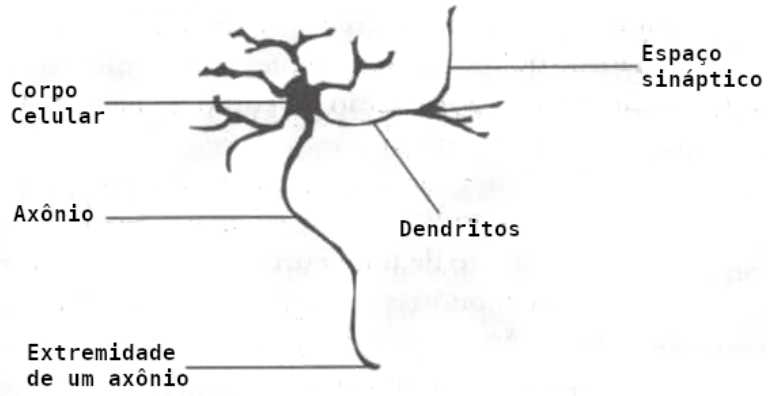
2.2 Redes Neurais Artificiais

As *Redes Neurais Artificiais* (RNAs) são uma tentativa computacional de modelar a capacidade de processamento de informação do sistema nervoso humano (ROJAS, 1996). Para alcançarem

um bom desempenho, as RNAs empregam uma interligação de estruturas bases chamadas de neurônios artificiais que, por sua vez, possuem pesos com valores numéricos positivos ou negativos associados entre si. Uma vantagem das RNAs é a grande capacidade de generalização, ou seja, a habilidade de produzir saídas adequadas para entradas que não estavam presente anteriormente durante seu aprendizado (HAYKIN, 2009). As RNAs têm sido frequentemente aplicadas nas áreas de medicina e negócios, além de um frequente desenvolvimento nos campos de processamento de sinais, reconhecimento de padrões em imagens e reconhecimento e produção de fala (FAUSETT, 1993).

A idealização dos neurônios artificiais foi inspirada nos neurônios biológicos encontrados no cérebro humano. Como mostrado na Figura 2.2, cada neurônio biológico é composto pelo corpo celular, os dendritos e o axônio. Os dendritos têm como papel a recepção das informações, ou impulsos nervosos, de outros neurônios e a submissão destas informações ao corpo celular, onde são processadas e novos impulsos são gerados. Estes impulsos são enviados aos dendritos de outros neurônios através do axônio. O ponto de contato entre os neurônios através do axônio e os dentritos, denominado sinapse, é onde ocorre toda a troca de informação necessária para conceber uma rede neural (BRAGA; CARVALHO; LUDERMIR, 2000).

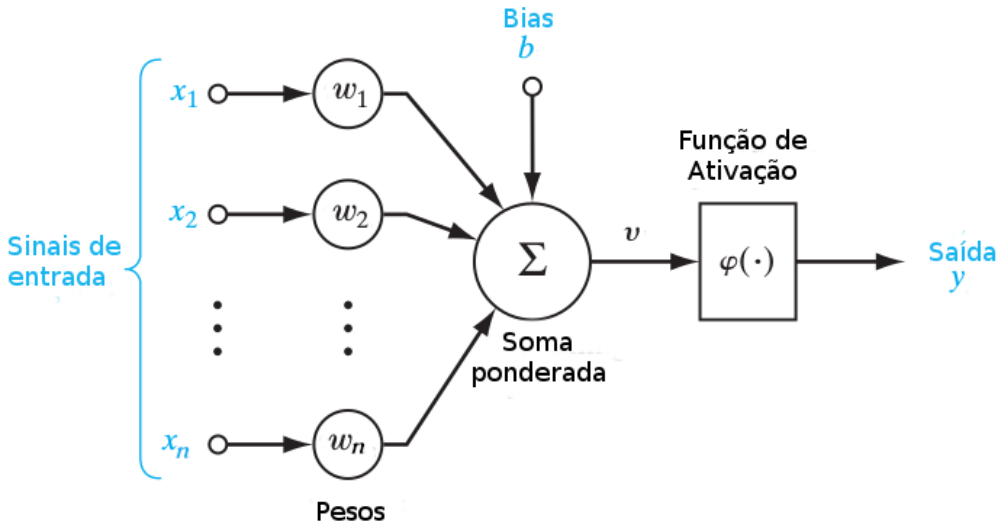
Figura 2.2: Estrutura de um neurônio biológico. Adaptado de: (BRAGA; CARVALHO; LUDERMIR, 2000)



Considerando uma analogia com os neurônios biológicos, modelou-se então a primeira noção de neurônios artificiais. Nestes neurônios, as entradas são valores $x = x_1, ..., x_n$ aos quais estão sujeitos um conjunto de pesos $w = w_1, ..., w_n$. Este modelo de neurônio utiliza ainda um

bias externo, denotado por b. Este bias é utilizado para o aumentar ou diminuir os valores de entrada da função de ativação, dependendo se o seu valor é positivo ou negativo, respectivamente (HAYKIN, 2009). Um neurônio artificial dispara quando a soma ponderada da entrada e do *bias* sujeita aos pesos ultrapassa um certo limiar de excitação, denominado *threshold*. No modelo de neurônio artificial apresentado, proposto por McCulloch e Pitts (MCP) (MCCULLOCH; PITTS, 1943), a ativação (disparo) do neurônio é obtida através da aplicação de uma *função de ativação*, como mostrado na Figura 2.3 (BRAGA; CARVALHO; LUDERMIR, 2000).

Figura 2.3: Representação de um neurônio artificial. Adaptado de: (HAYKIN, 2009).



No caso do neurônio MCP, a função de ativação é do tipo degrau deslocada, conforme Equação 2.1, e o seu valor de saída é obtido como resultado da comparação entre o *threshold* θ previamente definido e o valor da soma ponderada da entrada, como mostrado na Equação 2.2.

$$\varphi(v) = \begin{cases} 1, & \text{se } v > \theta. \\ 0, & \text{caso contrário.} \end{cases} \tag{2.1}$$

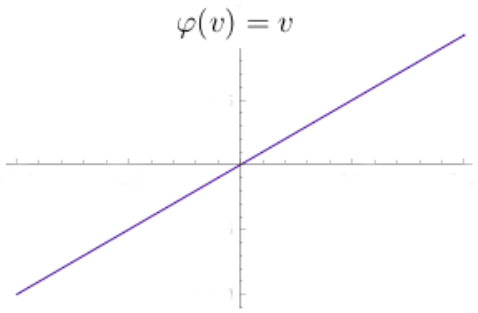
$$v = \sum_{i=1}^{n} x_i w_i + b \tag{2.2}$$

Embora o modelo MCP tenha considerado apenas funções de ativação do tipo degrau deslocada, outras definições também são possíveis. As funções identidade, sigmóide, tangente

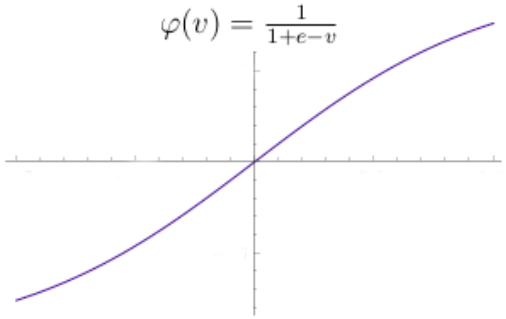
hiperbólica e retificada linear (ReLU) são comumente utilizadas, definidas tais como mostrado na Figura 2.4.

Figura 2.4: Exemplos de funções de ativação.

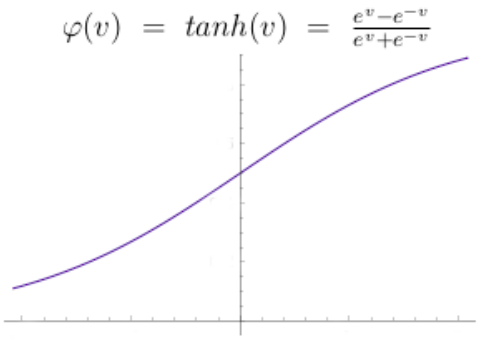
(a) Identidade ou Linear



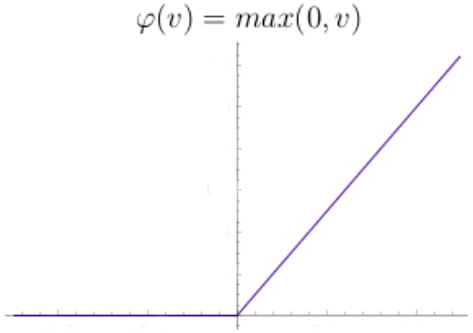
(b) Tangente Hiperbólica



(c) Sigmóide ou Logística



(d) Unidade Linear Retificada (ReLu)



Em 1958, visando a melhoria do neurônio MCP, Frank Rosenblatt desenvolveu o modelo *Perceptron* (ROSENBLATT, 1958). Neste modelo, criou-se o primeiro conceito de aprendizado através de neurônios artificiais, em que foi projetada uma regra de correção de erros para modificar os pesos associados a um neurônio quando suas respostas aos estímulos apresentados ao modelo forem erradas (ARBIB, 2003). Durante o processo de adaptação à resposta real, deseja-se identificar um valor Δw a ser aplicado ao vetor de pesos atual $w(t)$, para que seu valor atualizado $w(t+1)$ esteja mais próximo da solução desejada do que o valor atual $w(t)$. Para isso, definiu-se a Equação 2.3, denominada Regra Delta, cuja obtenção, descrita na Equação 2.4 estabelece o modo detalhado como esse ajuste de pesos é efetuado. Nesta segunda equação, η indica uma *taxa de aprendizado*, isto é, a velocidade em que o vetor de pesos será atualizado, e $\hat{y}(t)$ significa o valor previsto pelo modelo naquela iteração para a entrada $x(t)$, enquanto

$y(t)$ refere-se à saída real para esta entrada. Desta forma, o neurônio Perceptron adquiriu a capacidade de resolver problemas linearmente separáveis (BRAGA; CARVALHO; LUDERMIR, 2000).

$$w(t+1) = w(t) + \Delta w \tag{2.3}$$

$$= w(t) + \eta(y - \hat{y})x(t). \tag{2.4}$$

Neurônios artificiais possuem uma capacidade de generalização limitada, independente da função de ativação escolhida, devido a sua habilidade de resolver apenas problemas linearmente separáveis. Entretanto, a combinação desses neurônios para a formação de uma rede é capaz de resolver problemas de elevada complexidade (BRAGA; CARVALHO; LUDERMIR, 2000). Geralmente, identificam-se três classes fundamentais de RNAs, as *feedforward* com uma única camada, as *feedforward* com múltiplas camadas e as recorrentes. Numa rede do tipo *feedforward*, como as mostradas nas Figuras 2.5a e 2.5b, existe uma camada de entrada que é projetada diretamente para uma camada de saída constituída de neurônios, e nunca ao contrário. Uma rede recorrente, como a na Figura 2.5c, por sua vez, possui conexões ponderadas dentro de uma camada e diferencia-se pela presença de pelo menos um loop de retorno a camadas anteriores. Esses loops de retorno possuem ainda um retardo de uma unidade de tempo aplicado ao vetor de saída, denotado por z^{-1} (HAYKIN, 2009).

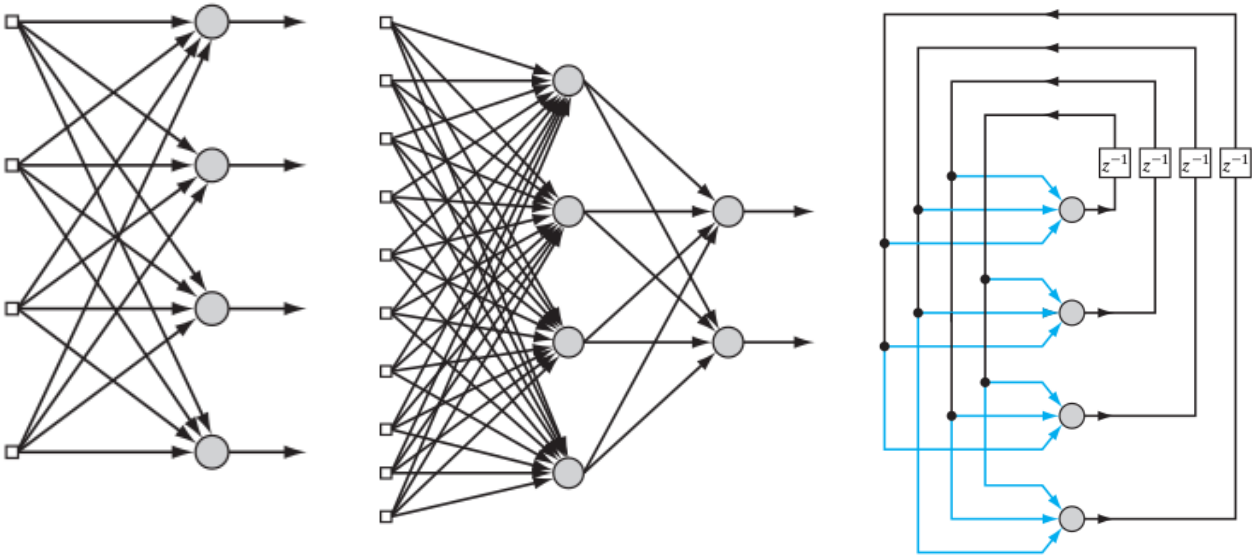
Redes com múltiplas camadas, como na Figura 2.5b, caracterizam-se pela presença de pelo menos uma camada oculta. Isso acarreta um grande poder às redes deste tipo pois, conforme Cybenko, uma rede com uma camada oculta é capaz de mapear qualquer função contínua, enquanto uma rede com duas camadas ocultas é suficiente para mapear qualquer função (CYBENKO, 1989).

Figura 2.5: Arquiteturas populares de RNAs. Fonte: (HAYKIN, 2009)

(a) *Feedforward* com uma única camada

(b) *Feedforward* com múltiplas camadas

(c) Recorrente



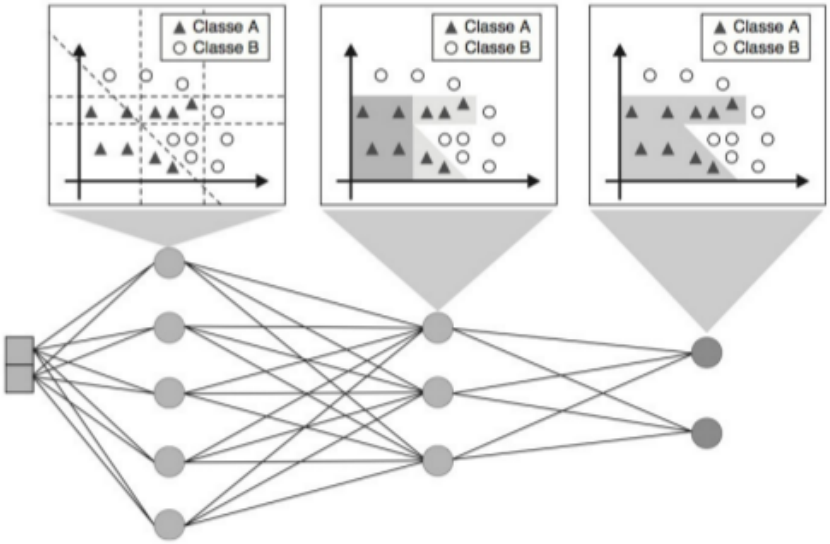
2.2.1 *Multilayer Perceptron*

As RNAs do tipo *Multilayer Perceptron* (MLP), são redes constituídas do neurônio Perceptron, *feedforward* e com múltiplas camadas, sendo estas uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. A arquitetura mais comum para uma rede MLP é a completamente conectada, de forma que os neurônios de uma camada estão conectados a todos os neurônios da próxima camada (FACELI et al., 2011).

Em uma rede MLP, a função implementada por um neurônio de certa camada é uma combinação das funções realizadas pelos neurônios da camada anterior que estão conectados a ele. Na primeira camada, cada neurônio aprende uma função que define um hiperplano. Na camada seguinte, os neurônios combinam um grupo de hiperplanos, formando regiões convexas. Os neurônios da camada seguinte combinam então um subconjunto das regiões convexas em regiões de formato arbitrário (FACELI et al., 2011). Na Figura 2.6, tem-se uma visualização do processo ocorrido.

O algoritmo de aprendizado supervisionado mais conhecido e utilizado para treinamento das MLPs é o *backpropagation* e para o seu correto funcionamento, a escolha da função de ativação deve considerar funções contínuas e diferenciáveis (HAYKIN, 2009). Neste algoritmo utiliza-se

Figura 2.6: Papel exercido pelos neurônios em cada camada de uma rede MLP. Fonte: (FACELI et al., 2011).



as entradas e as saídas desejadas para o ajuste dos erros da rede. O treinamento ocorre em duas fases, a fase *forward* e a fase *backward*, em que cada fase percorre a rede em um sentido. Na fase *forward*, a saída da rede é definida considerando certo padrão de entrada. A fase *backward* utiliza a saída desejada e a saída fornecida pela rede para atualizar os pesos nas suas conexões (BRAGA; CARVALHO; LUDERMIR, 2000). O *backpropagation* é simplesmente um método que utiliza o gradiente descendente para minimizar o erro total da saída calculada pela rede, na qual a derivada parcial define o ajuste dos pesos. Essa derivada mede a contribuição de cada peso no erro da rede para a classificação de dado objeto (FAUSETT, 1993; FACELI et al., 2011).

No âmbito do cálculo, o gradiente indica o sentido e a direção para os quais devem-se mover os valores dos pesos e do bias nas camadas, de forma a garantir o maior incremento possível de perda. Ou seja, nas técnicas de *backpropagation*, queremos mudanças de peso que trarão a inclinação mais íngreme ao longo da função de erro, com o intuito de encontrar o mínimo global desta função (GOODFELLOW; BENGIO; COURVILLE, 2016; KUBAT, 2015).

Um grande crescimento do poder computacional em termos de velocidade e memória tem acontecido nos últimos tempos. Dado isto, houve a viabilidade de treinamento das chamadas

redes neurais profundas, MLPs que possuem mais camadas escondidas do que o usual. Devido a ampla popularidade dessas redes e a capacidade computacional para a utilização de grande quantidade de dados de treinamento, foram desenvolvidas técnicas de *deep learning* em pleno estado da arte para detecção, segmentação, classificação e reconhecimento de objetos em imagens (KHAN et al., 2018). Utilizando-se redes neurais convolucionais, podemos ainda elencar aplicações como o reconhecimento de padrões em imagens para uso na medicina (CHA et al., 2016), a modelagem de frases por computadores (KALCHBRENNER; GREFENSTETTE; BLUNSOM, 2014) e o reconhecimento de caracteres e dígitos (LECUN et al., 1998). Essas e outras técnicas serão apresentadas mais profundamente nas seções a seguir.

2.3 *Deep Learning*

Deep Learning (DL), também conhecido como Aprendizado Profundo, é uma subárea específica de AM que enfatiza o aprendizado através de sucessivas camadas de representações cada vez mais significativas dos dados submetidos. No caso das redes neurais, tais representações são obtidas pelas camadas profundas (CHOLLET, 2017), isto é, pelas camadas ocultas que ocorrem em maior número do que nas redes neurais rasas, como as *multilayer perceptron*, comumente contendo mais que duas camadas ocultas (HEATON, 2015). A utilização de DL tem obtido êxito em endereçar problemas de Visão Computacional e Processamento de Linguagem Natural. Estes algoritmos não só ultrapassaram o desempenho de outras variedades de algoritmos de AM, como também pleiteam a eficácia na classificação alcançada por seres humanos (BUDUMA, 2017).

Os motivos para o corrente sucesso do DL podem ser exemplificados pela grande quantidade de dados disponíveis – como a base de dados *ImageNet*, organizada conforme a hierarquia *WordNet* e que disponibiliza imagens para pesquisadores ao redor do mundo (IMAGENET, 2019) – e o custo relativamente baixo de Unidades de Processamento Gráfico (GPUs), que são utilizadas para uma computação numérica muito mais eficiente. Grandes companhias do ramo tecnológico utilizam técnicas de DL diariamente para a análise de enormes quantidades de dados. Entretanto, esta especiliadade não é mais limitada somente ao domínio acadêmico e

industrial, ela tornou-se parte integrante da produção de softwares modernos disponibilizados aos consumidores (GULLI; PAL, 2017).

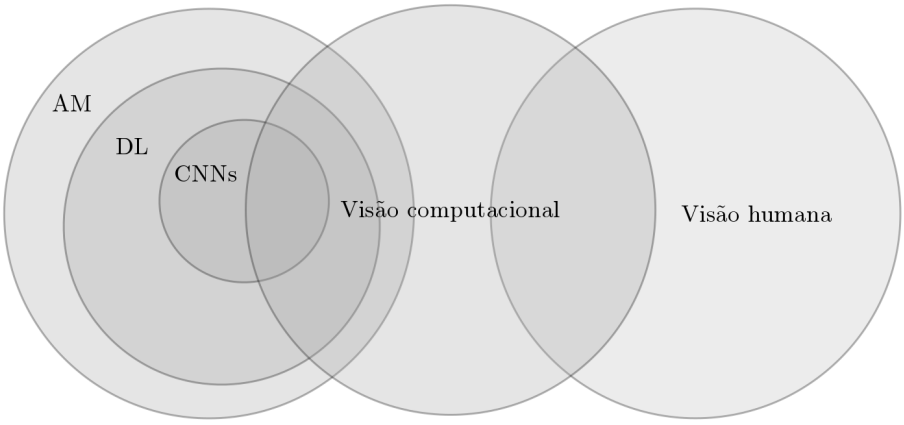
Dentre os diversos problemas que podem ser resolvidos com DL, pode-se citar alguns exemplos como, o reconhecimento automático de *captchas* (PINHEIRO, 2018), a identificação de armas de fogo inseridas em contextos (LIRA, 2018), a estimação da idade de um indivíduo considerando apenas uma imagem do mesmo (ARAUJO, 2018), entre outros.

O ferramental de DL compreende um conjunto de técnicas e modelos que podem ser aplicados a tarefas de aprendizado supervisionado e não-supervisionado. Porém, dentre os diferentes modelos existentes, as redes neurais convolucionais se destacam expressivamente, com um bom desempenho em diversos tipos de tarefas. A próxima seção descreve os pontos principais relacionados a este tipo de modelo.

2.3.1 Redes Neurais Convolucionais

As *Redes Neurais Convolucionais* (CNNs, do inglês *Convolutional Neural Networks*) são uma categoria de redes neurais profundas, *feedforward*, que comprovaram ser extremamente bem-sucedidas no ramo de visão computacional (KHAN et al., 2018). O termo denominado a estas redes, vem do seu aproveitamento da operação matemática chamada convolução, um tipo especializado de operação linear (GOODFELLOW; BENGIO; COURVILLE, 2016). Na Figura 2.7 é ilustrada a relação das CNNs com alguns campos de estudos conhecidos.

Figura 2.7: A relação entre a visão humana, visão computacional, AM, DL e CNNs. Adaptado de: (KHAN et al., 2018).



Para caracterizar as CNNs, é necessário conceituar as suas partes integrantes, em especial, as noções de convolução, *pooling*, as camadas completamente conectadas, operações de *droupout*, dentre outras. A *convolução*, em especial, é uma operação que consiste na soma dos produtos de toda a extensão de duas entradas em função de um deslocamento. Sendo assim, a convolução $s(t)$ de duas entradas $x_1(t)$ e $x_2(t)$ é uma função representada simbolicamente por $s(t) = x_1(t) * x_2(t)$ é definida conforme a Equação 2.5 (LATHI, 2008).

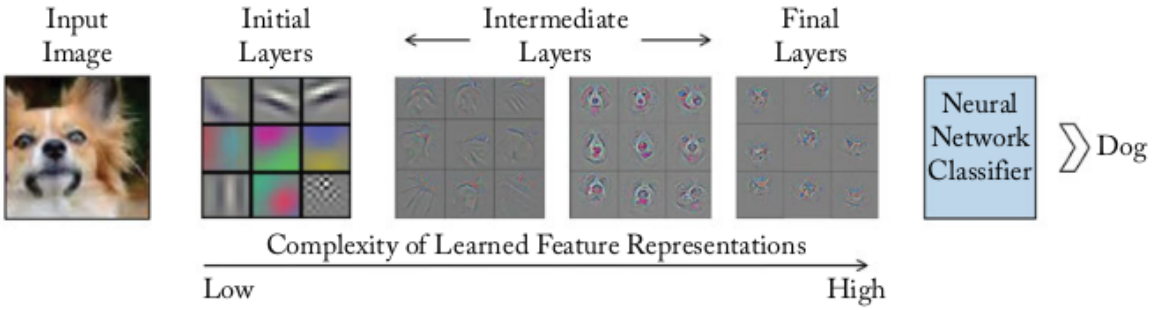
$$s(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(\tau)x_2(t-\tau)d\tau \tag{2.5}$$

Nas aplicações de AM, a função $x_1(t)$ é chamada de *input*, a função $x_2(t)$ é o *filtro*, também conhecido como *kernel*, e a saída $s(t)$ consiste no *mapa de características*, gerado pela convolução. O *input* é geralmente uma matriz multidimensional de dados de entrada e o filtro é uma matriz multidimensional de de parâmetros que são ajustados pelo algoritmo de aprendizado. Uma matriz multidimensional no contexto de AM é comumente referenciada como *tensor* (GOODFELLOW; BENGIO; COURVILLE, 2016).

Nas CNNs, as camadas convolucionais são responsáveis por aplicar as operações de convolução. Tomando como exemplo um problema de reconhecimento de padrões em uma imagem, cada camada convolucional é responsável por desenvolver os atributos detectados nas camadas anteriores – de linhas, a contornos, a formatos, até construir um objeto por completo. Nestas camadas, os mapas de características guardam as localizações desse atributos na imagem original, os quais são capturados através da aplicação de vários filtros, que diferem de acordo com o atributo que se deseja encontrar (BUDUMA, 2017). Esse processo pode ser visualizado na Figura 2.8.

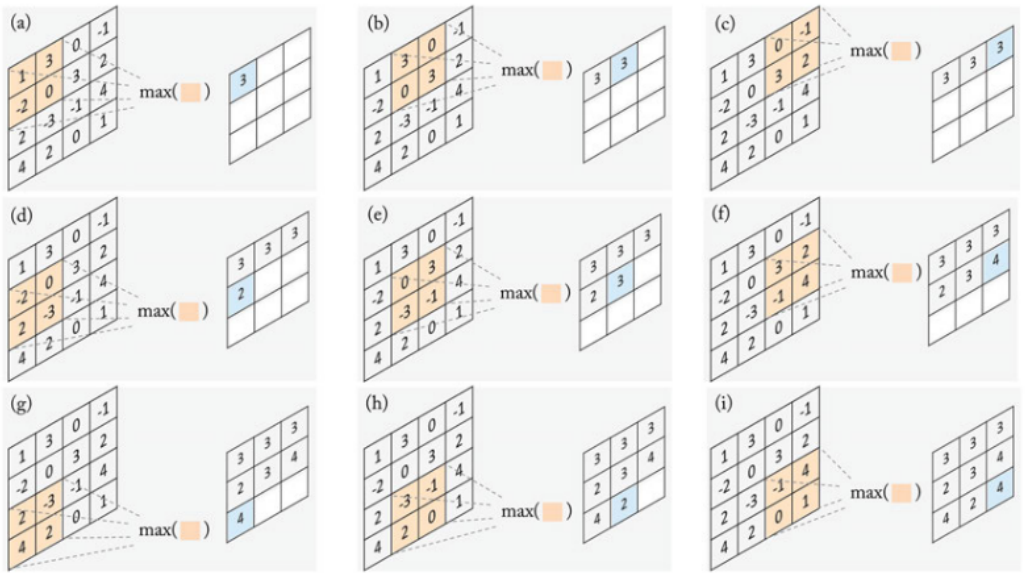
Uma camada de *pooling* em uma CNN opera em blocos do mapa de características e combina seus atributos através da operação de *max pooling* ou *average pooling*. Esse bloco é deslizado através do mapa de características com um passo definido como *stride*. A operação de *max pooling* retorna o valor máximo dos dados em uma área retangular. Enquanto a operação de *average pooling*, realiza o mesmo processo, porém utiliza a média desses valores. O propósito da camada de *pooling* é, além de diminuir a quantidade de amostras no mapa de características,

Figura 2.8: Exemplo de processo realizado pelas camadas convolucionais de uma CNN, aplicado a um problema de classificação de imagens. Fonte: (KHAN et al., 2018).



ajudar a sua representação a se tornar invariante a pequenas mudanças nos dados de entrada (KHAN et al., 2018; GOODFELLOW; BENGIO; COURVILLE, 2016). Uma visualização detalhada dessa operação é demonstrada na Figura 2.9.

Figura 2.9: Visualização da operação de *max pooling* considerando uma região de 2 x 2 com um *stride* igual a 1. Fonte: (KHAN et al., 2018).



As *Camadas Completamente Conectadas* (FCL, do inglês *Fully Connected Layers*) consideram um conjunto de neurônios completamente conectados aos neurônios da camada anterior, sendo usualmente encontradas no final de uma CNN. Possui a capacidade de separar as variações de classificação que serão retornadas na saída, resumindo os resultados dos vários mapas de características produzidos pela rede (KHAN et al., 2018). Na última camada de uma CNN, adota-se geralmente a função de ativação *softmax*, a qual atua escalando as saídas da rede em

um vetor de probabilidades, esse processo pode ser muito útil para problemas de classificação (GULLI; PAL, 2017).

Para prevenir o aprendizado de padrões irrelevantes por um modelo de AM, pode-se articular a quantidade de informações que esse modelo pode arquivar ou adicionar restrições sobre as informações que podem ser armazenadas. Se uma CNN tende a memorizar um pequeno número de padrões, o processo de otimização forçará o foco nos padrões mais proeminentes, que possuem uma chance melhor de gerar uma boa generalização. O processo de evitar o *overfitting* dessa maneira é chamado de *regularização* (CHOLLET, 2017).

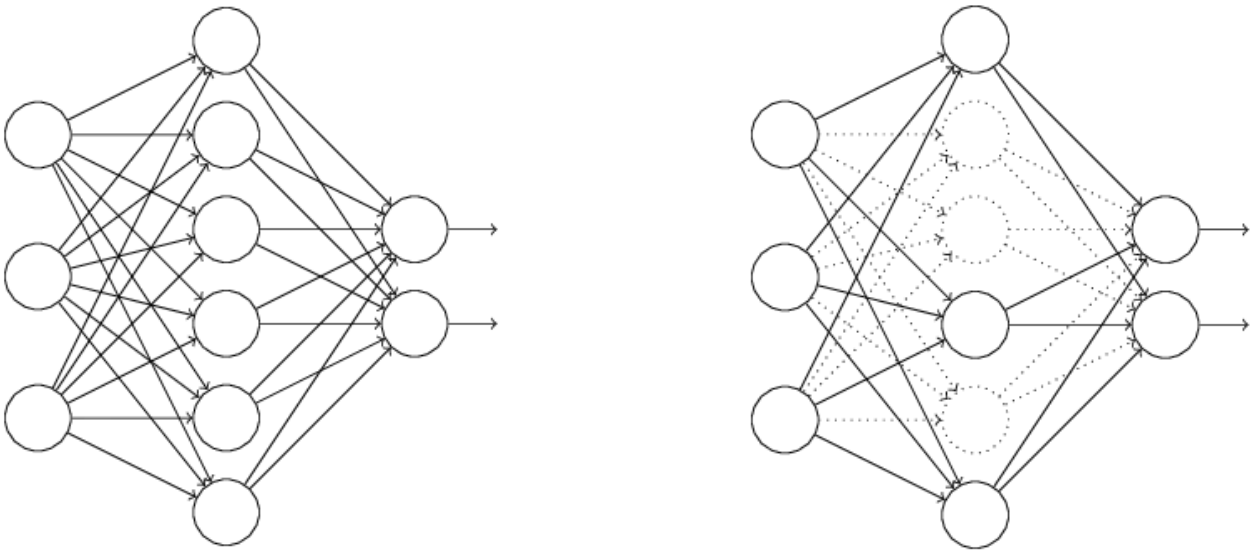
O *dropout* é um tipo de regularização muito efetivo e que é usualmente utilizado em CNNs. Consiste na desativação temporária de alguns neurônios durante a fase de treinamento de uma rede. Nesse processo, os neurônios são desativados mediante uma probabilidade p, conhecida como *dropout rate*, retornando um valor de saída igual a 0. Na fase de teste da rede, nenhum neurônio é desativado, ao passo que, como forma de balanceamento devido a quantidade maior de neurônios presentes em comparação à fase de treinamento, os valores de saída das camadas são reduzidos por um fator igual ao *dropout rate* (CHOLLET, 2017). Dessa maneira, pode-se afirmar que o *dropout* ajuda a previnir o *overfitting* ao possibilitar uma forma de combinar diferentes arquiteturas de redes neurais (BUDUMA, 2017). Esta operação pode ser visualizada na Figura 4.4, na qual os neurônios desativados são demonstrados pelas circunferências com a borda pontilhada.

Uma vez definidos os elementos integrantes das CNNs, estes podem ser compostos de diferentes maneiras para caracterizar a arquitetura de tais redes. Embora a sua utilização possa ser feita de maneira sistemática, algumas organizações de tais camadas já apresentadas na literatura caracterizam arquiteturas canônicas, as quais foram utilizadas em diferentes problemas com um desempenho significativamente positivo. Desta feita, a seção a seguir contempla algumas destas arquiteturas.

Figura 2.10: Processo de aplicação da operação de *dropout*. Os neurônios e ligações desativados estão denotados de forma pontilhada. Fonte: (ACADEMY, 2019)

(a) Arquitetura de uma rede antes da aplicação do *dropout*.

(b) Arquitetura da rede após a aplicação do *dropout*.



2.3.2 Arquiteturas Canônicas de Redes Neurais Convolucionais

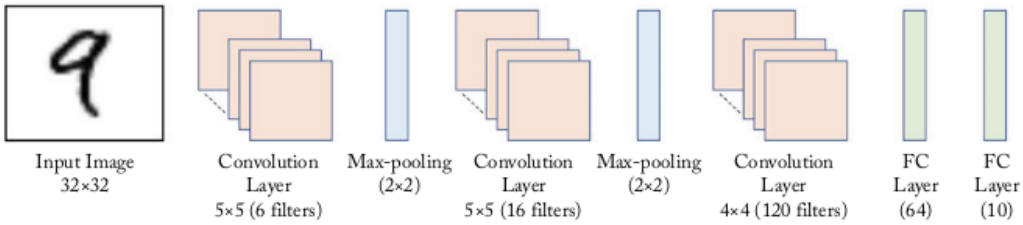
Como mencionado anteriormente, o *ImageNet* é uma base de dados que possui mais de 15 milhões de imagens rotuladas manualmente, de alta resolução, separadas em mais de 22 mil categorias (IMAGENET, 2019). Visando o uso dessa base, tem sido lançando desde 2010 um desafio anual chamado *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), no qual possui o intuito de aumentar o desempenho das tecnologias em estado da arte para classificação de imagens e detecção e localização de objetos em imagens (SEWAK; KARIM; PUJARI, 2018).

Apesar dos conceitos das camadas que compõem as CNNs serem bastante conhecidos e utilizados, ainda é uma atividade de grande dificuldade e responsabilidade propor arquiteturas de redes neurais que executem determinadas tarefas. Portanto, existem diversas arquiteturas canônicas que apresentam um grande desempenho em treinar e executar tarefas de visão computacional, nas quais a grande maioria foi desenvolvida através do desafio ILSVRC. Devido a grande frequência da utilização dessas arquiteturas, a seguir serão apresentados alguns dos seus aspectos mais relevantes.

A primeira das arquiteturas a ser desenvolvida utilizando camadas convolucionais em vez de

camadas completamente conectadas convencionais foi a LeNet, proposta por LeCun em 1998 (LECUN et al., 1998). Uma variante dessa arquitetura é a LeNet-5, composta por sete camadas, nas quais cinco dessas possuem pesos ajustáveis e outras duas são compostas por operações de *max pooling*. Esta arquitetura foi aplicada na identificação de dígitos manuscritos, utilizando o conjunto de dados *Modified National Institute of Standards and Technology* (MNIST) como treinamento. (KHAN et al., 2018). Na Figura 2.11 é possível visualizar a composição da LeNet-5.

Figura 2.11: A arquitetura LeNet-5. Fonte: (KHAN et al., 2018).



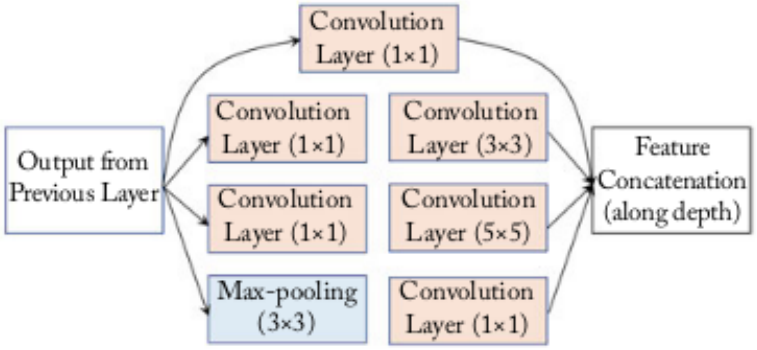
O primeiro modelo em larga escala de CNNs utilizou-se da arquitetura AlexNet, garantindo o primeiro lugar no desafio ILSVRC em 2012 por uma grande margem de diferença dos outros modelos e levando ao ressurgimento da utilização de redes neurais profundas em visão computacional. O uso da função de ativação ReLU após suas oito camadas paramétricas, nas quais as cinco primeiras são convolucionais e as últimas três são completamente conectadas, é um diferencial dessa arquitetura. Operações de *max pooling* são aplicadas após as duas primeiras e a última camada convolucional, ao passo que operações de *dropout* são executadas após as duas primeiras camadas completamente conectadas, acarretando na diminuição do *overfitting* e garantindo uma boa generalização para exemplos não vistos anteriormente (KHAN et al., 2018). Apesar de já existirem arquiteturas de CNNs mais eficientes disponíveis, a AlexNet ainda é muito utilizada atualmente, devido a sua estrutura simples e profundidade relativamente menor (SEWAK; KARIM; PUJARI, 2018).

A VGGNet é uma das mais populares arquiteturas de CNN desde a sua introdução em 2014 e, mesmo não ganhando o desafio ILSRVC, conseguiu uma taxa de erro de apenas 7.3%. Concebida na Universidade de Oxford, a VGGNet é composta por uma combinação de camadas convolucionais, FCLs, camadas de *pooling* e *dropout*. Esta arquitetura existe em duas versões,

a VGG16 e a VGG19, nas quais os números associados aos seus nomes correspondem a sua quantidade de camadas, sem considerar as camadas de *pooling* e *dropout* (KHAN et al., 2018; SEWAK; KARIM; PUJARI, 2018). A VGGNet usa apenas filtros de convolução com uma dimensão de 3×3 e as operações de *max pooling* são realizadas através de uma janela de 2×2 pixeis com um *stride* igual a 2 (SIMONYAN; ZISSERMAN, 2015a).

A arquitetura GoogLeNet, também chamada de Inception, foi desenvolvida pela empresa Google e se tornou a vencedora do desafio ILSRVC de 2014. Seu diferencial em relação às outras arquiteturas foi a combinação não sequencial das suas 22 camadas convolucionais. Como mostrado na Figura 2.12, cada camada é executada de forma paralela com outras camadas formando um módulo chamado Inception, que condensa os mapas de características obtido por cada camada e passa como entrada para o próximo bloco Inception encontrado na rede (KHAN et al., 2018). Na GoogLeNet, geralmente ocorre o uso de convoluções 1×1 com a função de ativação ReLU, objetivando a diminuição das dimensões do problema antes das dispendiosas convoluções de 3×3 e 5×5 (SEWAK; KARIM; PUJARI, 2018).

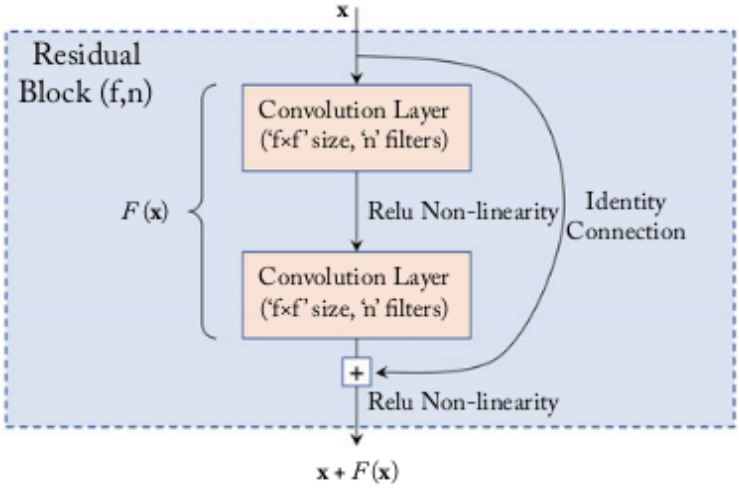
Figura 2.12: Exemplo de um módulo Inception da GoogLeNet. Fonte: (KHAN et al., 2018).



A Microsoft *Residual Network* (ResNet) foi a CNN vencedora do desafio ILSVRC 2015 com um grande ganho em desempenho, diminuindo a taxa de erro top-5 para apenas 3.6% em comparação à taxa de 6.7% da vencedora do ano anterior, a GoogLeNet. Com o total de 152 camadas, a ResNet deve seu sucesso aos chamados blocos residuais, representado na Figura 2.13, no qual as entradas originais são submetidas a uma função de transformação que é conectada diretamente à entrada, chamada de *skip identity connection*. Segundo Khan, uma rede muito profunda sem nenhuma conexão residual obtem uma taxa maior de erro no treinamento e no

teste, portanto, as conexões residuais são consideradas fatores importantes para uma melhor classificação de redes neurais profundas (KHAN et al., 2018).

Figura 2.13: Estrutura de um bloco residual da ResNet. Fonte: (KHAN et al., 2018).



2.4 Tecnologias Utilizadas

As tecnologias utilizadas para a realização desse trabalho foram, em sua maior parte, relacionadas à linguagem de programação *Python*. A escolha dessa linguagem se deu pela sua popularidade para fins de criação de modelos de ML, assim como pela grande quantidade de bibliotecas que facilitam o desenvolvimento desses modelos (BRINK; RICHARDS; FETHEROLF, 2016).

Para o pré-processamento das imagens em termos de redimensionamento e elaboração dos exemplos, utilizou-se a biblioteca `PIL` (Pillow) (PIL, 2019). Quanto à manipulação e organização dos arquivos, foram utilizadas as bibliotecas `os` e `glob` (OS, 2019; GLOB, 2019). A aplicação do treinamento e teste dos modelos propostos ficou por conta das bibliotecas `keras` e `tensorflow`(KERAS, 2019; TENSORFLOW, 2019). No que diz respeito ao cálculo de métricas de desempenho, as bibliotecas que tiveram um papel protagonista foram a `scikit-learn` e a `numpy`, na qual a última também foi utilizada para manipular o conjunto de imagens e as suas representações matriciais (LEARN, 2019; NUMFOCUS, 2019b). Por fim, a visualização dos dados de treinamento e de alguns resultados foi realizada com as bibliotecas `matplotlib` e `seaborn` (NUMFOCUS, 2019a; WASKOM, 2019).

Sabe-se que o treinamento de uma CNN requer um custo computacional muitas vezes não alcançado por unidades de processamento comuns. Portanto, com a finalidade de treinar os modelos propostos, foram utilizadas as GPUs disponíveis no Laboratório de Sistemas Inteligentes da UEA. Porém, durante a fase inicial do treinamento das redes foi utilizada também a plataforma *Kaggle*. Os *kernels* disponíveis nesta plataforma são recursos em nuvem com configurações pré-determinadas e customizadas, facilitando a reprodução de exemplos e preparação do ambiente de treino e teste das CNNs (KAGGLE, 2019).

Capítulo 3

Trabalhos Relacionados

Na literatura, existem uma grande quantidade de soluções elaboradas para resolver o problema de autenticação de assinaturas manuscritas, porém os resultados mais comparáveis aos encontrados no desenvolvimento deste trabalho podem ser verificados a partir da *Signature Verification Competition* ocorrida em 2009 (SigComp2009), na qual a tarefa de aprendizado abordada se assemelha à tarefa aqui considerada. Na referida competição, os participantes foram instruídos a submeterem um sistema que, ao receber uma assinatura genuína de um indivíduo como referência e uma outra assinatura para comparação, deveria retornar um grau de similaridade entre as assinaturas e um valor binário de decisão que definia a autenticidade da assinatura em questão (BLANKERS et al., 2009). Com o intuito de analisar o desempenho dos sistemas submetidos, a organização da competição decidiu adotar a métrica de *equal error rate* (EER), frequentemente utilizada na avaliação de sistemas biométricos. Esta métrica identifica um ponto de equilíbrio entre as taxas de falsa aceitação e falsa rejeição e, deste modo, quanto mais baixo for o seu valor, melhor é a qualidade do sistema biométrico analisado (MAGALHãES; SANTOS, 2003).

Na guia da SigComp2009 que buscava analisar sistemas que verifivam assinaturas *offline*, houve a participação de oito competidores. Dentre estes, o melhor sistema verificador obteve um EER de 9.15% e foi construído através de uma única abordagem que se baseava na informação de cores das imagens. O segundo melhor modelo, com um EER de 15.5%, foi obtido através de redes neurais artificiais que visavam encontrar o conjunto de características ideal das imagens para a classificação das assinaturas, porém os seus autores decidiram manter anônimas quaisquer

outras informações a respeito da construção deste sistema (BLANKERS et al., 2009; VOLKER; UMAPADA; APOSTOLOS, 2018).

Antes disso, em 2008, Impedovo e Pirlo reuniu um conjunto de trabalhos que ditaram o estado da arte em verificação automática de assinaturas (IMPEDOVO; PIRLO, 2008). Considerando uma abordagem de verificação *offline*, dentre estes trabalhos, destaca-se um modelo criado utilizando redes neurais, o qual utilizou como atributos preditores a projeção e o contorno retirado das assinaturas. Este modelo obteve uma taxa de falsa aceitação de 3% e uma taxa de falsa rejeição de apenas 1% (BAJAJ; CHAUDHURY, 1997).

O trabalho de Ribeiro et al. emprega técnicas de DL para a identificação *offline* de assinaturas manuscritas em um *dataset* disponibilizado pelo *Grupo de Procesado Digital de Senales* (GPDS). Este trabalho consiste, primeiramente, no uso de *K-means* e índices de frequência obtidos através das transformadas discretas de Fourier, de cosseno e de wavelet para a extração de características das assinaturas que, em um segundo passo, foram fornecidas à Maquinas de Vetores de Suporte (SVMs) com o intuito de coletar métricas para análise posterior dos modelos obtidos. A abordagem deste trabalho considerou a criação de um modelo híbrido que constituise de uma dupla validação, nas quais na primeira destas, o modelo deve identificar o proprietário da assinatura em questão e, subsequentemente, determinar a sua autenticidade. Dentre as métricas coletadas, as principais foram as taxas de falsa aceitação e falsa rejeição, a acurácia e o *F-score*, na qual a última destas obteve um valor de 0.8615 no melhor modelo produzido. Em um terceiro passo do trabalho, uma pequena parte dos dados utilizados anteriormente foi disponibilizada à uma *Restricted Boltzmann Machine*, visando apenas a demonstração visual dos pesos obtidos por este de tipo de rede profunda, não havendo a existência de testes dessas características quanto à classificação da autenticidade (RIBEIRO et al., 2011).

Mais recentemente, Hafemann et al. propuseram o aprendizado de características de assinaturas manuscritas *offline* utilizando redes neurais convolucionais em conjunto com SVMs. O conjunto de dados utilizado para o treinamento dos modelos foi também disponibilizado pelo GPDS. As abordagens para a solução do problema foram diversas, porém, a que mais se destacou foi aquela na qual os modelos gerados classicavam a autenticidade da assinatura de forma

independente dos autores das mesmas. O melhor dentre estes modelos obteve um EER de 1.72%, conseguindo superar o estado da arte até aquele momento (HAFEMANN; SABOURIN; OLIVEIRA, 2017).

Levando em conta o estado da arte, as diferentes estratégias para abordar o problema e ainda os poucos trabalhos envolvendo redes neurais convolucionais, verifica-se a importância de perseguir esta perspectiva e colaborar com resultados que visem avaliar o potencial de tais modelos no cenário em questão.

Capítulo 4

Solução Proposta

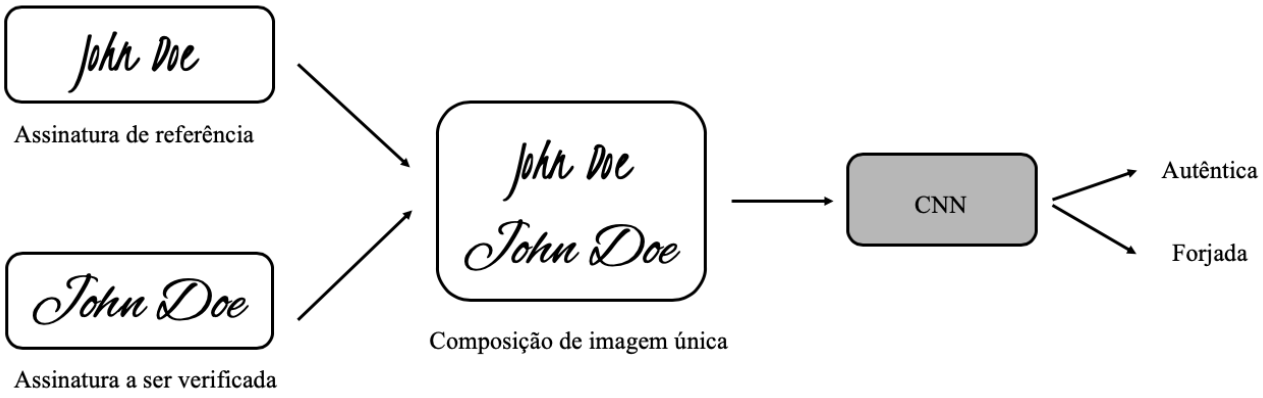
Nesta seção serão apresentados os elementos necessários para o entendimento da solução proposta para este trabalho. A Seção 4.1 demonstra os detalhes da tarefa de aprendizado utilizada para endereçar a verificação de assinaturas manuscritas. Posteriormente, a Seção 4.2 mostra uma visão geral do conjunto de dados original. Este conjunto foi submetido a uma preparação, descrita na Seção 4.3, com o objetivo de produzir modelos de CNNs referentes à tarefa apresentada. Estes modelos e os parâmetros e hiperparâmetros utilizados para criá-los estão dispostos, por fim, na Seção 4.4.

4.1 Tarefa de Aprendizado

Com o intuito de checar a autenticidade de assinaturas utilizando CNNs, concebeu-se uma tarefa de classificação binária para alcançar este objetivo. Nela, uma imagem de 256×256 *pixels* composta por duas assinaturas manuscritas, na qual a primeira delas representa uma assinatura de referência genuína e a segunda compreende uma assinatura a ser verificada. A partir desta entrada, deseja-se obter como a saída de uma CNN para esta tarefa, a predição da autenticidade da segunda assinatura que, por ser uma tarefa de classificação binária, poderá assumir somente duas classificações: *autêntica* ou *forjada*. Os passos compreendidos nesta tarefa podem ser visualizados na Figura 4.1.

Para esta tarefa, diferentes CNNs serão consideradas. O treinamento e teste destes modelos

Figura 4.1: Uma visão geral da tarefa de aprendizado considerada.



serão feitos com o método *holdout* de validação cruzada, em que 70% dos dados serão utilizados no treino e ajuste de parâmetros, enquanto 20% dos dados serão aproveitados para o processo de teste das redes, com vista a capturar o poder de generalização dos modelos considerados. Os 10% de dados remanescentes, serão utilizados para a validação dos modelos durante o processo de treinamento (BRINK; RICHARDS; FETHEROLF, 2016).

Os modelos propostos serão avaliados perante as métricas de desempenho de *Acurácia*, *F-score* e EER (*equal error rate*). A acurácia indica a proporção de predições corretas inferidas pelos modelos. O *F-score*, por sua vez, será calculado pela média harmônica da precisão e da revocação considerando uma tarefa de classificação binária (MARSLAND, 2015), da seguinte forma:

$$\text{Precisão} = \frac{TP}{TP + FP}, \qquad \text{Revocação} = \frac{TP}{TP + FN},$$

$$\text{F-Score} \quad = \quad 2 \cdot \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}},$$

em que TP indica o número de verdadeiros positivos, FP indica o número de falsos positivos e FN indica o quantitativo de falsos negativos nas previsões obtidas.

A métrica EER é calculada baseando-se nas taxas de falsa aceitação (FAR, do inglês *false acceptance rate*) e falsa rejeição (FRR, do inglês *false rejection rate*). Estas taxas referem-se à

quantidade de assinaturas forjadas consideradas genuínas e assinaturas genuínas consideradas forjadas, respectivamente. Dessa maneira, o EER retrata o resultado encontrado quando os valores de FAR e FRR são iguais. A verificação desta igualdade é dada através da variação do limiar definido para a decisão da classificação dos exemplos dispostos ao modelo (WIRTZ, 1997). Uma visão gráfica do cálculo desta métrica pode ser visualizado na Figura 4.2.

Figura 4.2: As curvas típicas da FAR e FRR, plotadas uma ao lado da outra em relação ao limiar de decisão configurado para o modelo. As curvas se cruzam no ponto que equivale ao valor de EER. Adaptado de: (SIMãO, 2008)

4.2 Visão Geral do Conjunto de Dados

Para a obtenção de um modelo inteligente capaz de verificar a autenticidade de assinaturas de indivíduos é necessário, primeiramente, treiná-lo. Para tanto, é necessário um conjunto de exemplos, isto é, uma base de dados que possua exemplos de assinaturas e seus respectivos rótulos, isto é, quando são forjadas ou genuínas, com vistas a prover características relevantes para o aprendizado.

Para este fim, utilizou-se dois conjuntos de dados originalmente disponibilizados pela *Signature Verification Competition* realizada na *International Conference on Document Analysis and Recognition* em 2009 (LIWICKI, 2012). Na ocasião da competição, cada um destes conjuntos de dados foi utilizado em uma etapa. Na etapa de treinamento, foi utilizado o conjunto de assinaturas do *Norwegian Information Security laboratory and Donders Centre for Cognition* (NISDCC), composto originalmente de 1.920 assinaturas genuínas e forjadas. Para a etapa seguinte, de validação dos modelos submetidos, foi utilizado o conjunto coletado pelo

Netherlands Forensic Institute (NFI), composto por 1.953 novas assinaturas genuínas e forjadas (BLANKERS et al., 2009). Considerando que a competição ocorreu há uma década, as bases de dados utilizadas na época são atualmente mantidas pela *International Association for Pattern Recognition* (IAPR) e contam com um número de 1.898 assinaturas do conjunto NISDCC e 1.564 assinaturas do conjunto NFI (LIWICKI, 2012). Esta versão mais recente é amplamente disponibilizada e, por essa razão, está sendo utilizada no escopo deste trabalho.

Os conjuntos disponibilizados pelo IAPR são compostos por dois tipos de assinaturas, as assinaturas *offline* e as assinaturas *online*. Nas assinaturas *offline*, é considerado apenas o aspecto estático da mesma, ou seja, uma imagem obtida após o processo da assinatura ter sido concluído. Estes dados foram segmentados, inspecionados visualmente e, em seguida, pré-processados para fornecer imagens formatadas em cores, em escala de cinza e binárias com a resolução de 600 dpi. Os dados das assinaturas *online*, por sua vez, continham informações dinâmicas, que consistiam em arquivos de texto que descreviam os detalhes capturados em vários pontos durante o processo da assinatura, sendo estes as coordenadas x e y da ponta da caneta, a pressão exercida sobre a caneta, o ângulo azimutal e o ângulo de elevação (BLANKERS et al., 2009). Um exemplo de uma assinatura *offline* e a sua respectiva representação *online* com os pontos plotados pode ser encontrada na Figura 4.3.

Figura 4.3: Uma amostra das assinaturas *offline* e *online* da SigComp2009. Fonte: (BLANKERS et al., 2009).

Nestas bases de dados um certo autor produz várias versões de sua própria assinatura, compondo os exemplos de assinaturas genuínas. Várias pessoas foram convocadas a falsificar esta assinatura, produzindo os exemplos forjados das mesmas. Nestas falsificações utilizou-se a

técnica *over-the-shoulder*, na qual o autor forjador tem a oportunidade de visualizar a assinatura genuína antes da falsificação, podendo, inclusive, ter praticado anteriormente diversas vezes. Segundo Blankers et al., este tipo de falsificação costuma ser de difícil detecção (BLANKERS et al., 2009).

Considerando a demanda por equipamentos específicos para obtenção das assinaturas *online* e da pouca existência dos mesmos em cenários práticos, optou-se apenas pela utilização das assinaturas *offline* para a elaboração deste trabalho, com vistas a concentrar os esforços em uma solução que incorpore aspectos de Visão Computacional. Após a exclusão destes exemplos, o quantitativo remanescente de assinaturas e seus tipos (genuína ou forjada) encontram-se disponíveis na Tabela 4.1.

Tabela 4.1: Quantitativo de indivíduos e assinaturas *offline* por conjunto de dados.

Conjunto	**Autores originais**	**Autores forjadores**	**Autores originais com assinaturas forjadas**	**Assinaturas genuínas**	**Assinaturas forjadas**	**Total de assinaturas**
NISDCC	12	31	12	60	1.838	1.898
NFI	79	33	19	940	624	1.564

Conforme pode ser observado, um mesmo autor produziu diferentes versões de sua assinatura. A coluna "Autores originais" indica o quantitativo destes indivíduos e a coluna "Assinaturas genuínas" indica o total de assinaturas feitas pelos mesmos. No caso do *dataset* NISDCC, em especial, cada autor reproduziu sua própria assinatura 5 vezes. No NFI não houve uma consistência quantitativa, mas, em média, existem 11 reproduções da assinatura original pelo autor verdadeiro.

Ainda conforme a Tabela 4.1, o NISDCC conta com 31 autores forjadores, os quais produziram versões forjadas de todas as assinaturas originais, mas com um quantitativo de falsificações distintos para cada original, totalizando 1.838 assinaturas forjadas com a técnica *over-the-shoulder*. No caso do NFI, isto não ocorreu de mesma forma, pois apenas um subconjunto das originais foi alvo de falsificação.

Considerando o total exposto de assinaturas originais e forjadas, tendo sido compreendida a estrutura, organização e exemplos dos *datasets*, partiu-se então para sua preparação com vistas

a adequar seu uso para a solução proposta.

4.3 Preparação do Conjunto de Dados

Sabe-se que algoritmos de AM necessitam de quantidade significantiva de dados, preferencialmente sem muitos ruídos, para serem utilizados de forma a obter um modelo que possua bom desempenho (MARSLAND, 2015). Levando isto em conta e com vistas a adequar os dados disponíveis com a tarefa de aprendizado considerada, uma etapa de pré-processamento fez-se necessária, cujos passos são descritos a seguir.

Primeiramente foi necessário realizar a adaptação das imagens individuais para as imagens compostas, conforme apresentado anteriormente no esquema da Figura 4.1. Para isto, foi feita a combinação de cada assinatura genuína de um autor com suas diferentes versões originais, produzindo uma nova imagem para cada caso, a qual associou-se o rótulo de autêntica. Após esta etapa, também foram combinados os exemplos genuínos com suas respectivas versões forjadas, aos quais foi associado o rótulo de forjado. Todas as imagens obtidas dessas combinações serão utilizadas como exemplos para o processo de treinamento, validação e teste do modelo proposto.

Para preservar a referência aos autores e ids de suas assinaturas, os nomes dos arquivos passaram a conter tais informações. Entretanto, ressalta-se que os modelos de CNNs não terão acesso a este dado (nome do arquivo). Ele serve de referência apenas para verificar a procedência do exemplo.

O processo de combinação das imagens de cada um dos exemplos foi realizado em três etapas. Na primeira etapa, ambas as imagens foram redimensionadas para um tamanho de 256×256 *pixels*. Em seguida, as imagens foram concatenadas verticalmente com a intenção de formar uma única imagem de 256×512 *pixels*. Por fim, a imagem resultante foi redimensionada novamente em um tamanho de 256×256 *pixels* e transformada para um espaço de cores em escala de cinza, com a intenção de padronizar todos os exemplos.

Ao concluir o processo anterior, necessitou-se então a realização da partição dos exemplos conforme o método *holdout* e, com vistas a evitar que fossem apresentadas as assinaturas de um

mesmo indivíduo a um modelo durante as etapas de treinamento e teste, foram adotadas duas abordagens diferentes. Na primeira destas, chamada de abordagem A, decidiu-se evitar o aparecimento apenas dos exemplos forjados de um mesmo autor em duas partições diferentes, sendo assim, os exemplos autênticos foram distribuídos conforme o método previamente especificado, enquanto que os exemplos forjados foram separados seguindo os seguintes critérios:

- Se uma assinatura autêntica possuía apenas um autor forjador, todos os exemplos forjados foram incluídos no conjunto de treinamento;
- Se uma assinatura autêntica possuía quatro autores forjadores, as assinaturas de três desses autores foram para o conjunto de treinamento e as remanescentes, pertencentes a apenas um autor, foram para o conjunto de teste;
- Se uma assinatura autêntica possuía cinco autores forjadores, as assinaturas de quatro desses autores foram para a etapa de treinamento e as remanescentes, pertencentes a apenas um autor, permaneceram na etapa de teste;
- Se uma assinatura autêntica possuía seis autores forjadores, as assinaturas de quatro desses autores foram para a etapa de treinamento e as assinaturas dos outros dois foram para a etapa de teste;
- Se uma assinatura possuía trinta ou mais autores forjadores, as assinaturas de dez desses autores foram para o conjunto de teste, três desses autores foram utilizados para o conjunto de validação e as assinaturas restantes foram remanejadas para o conjunto de treinamento.

Para a segunda abordagem, chamada de abordagem B, decidiu-se que não deveriam existir exemplos de um mesmo autor em duas partições diferentes, sendo estes exemplos forjados ou autênticos. Em um cenário prático de eventual utilização desta abordagem, o modelo construído já terá sido treinado e será requisitado a avaliar uma assinatura, tomando como referência uma assinatura autêntica de um autor nunca antes visto pelo modelo. Levando isto em consideração, a etapa de testes incluírá apenas exemplos de autores inéditos para o modelo. Para que isso fosse

possível, a escolha da quantidade de autores para cada partição foi utilizada conforme o método *holdout* especifidado anteriormente e, todos exemplos de dado autor, foram utilizados para a partição que lhe foi designada. Esta abordagem equivale ao método de avaliação considerado para os modelos submetidos à SigComp2009, sendo assim, seus resultados podem servir como forma de comparação às métricas coletadas durante esta competição (BLANKERS et al., 2009).

Os autores forjadores da abordagem A e todos os autores da abordagem B selecionados para estarem presentes em cada uma das partições foram escolhidos de forma pseudoaleatória. Dessa maneira, ressalta-se que os conjuntos de treino, teste e validação de ambas as abordagens são disjuntos no tocante aos autores forjadores.

Após o particionamento dos dados conforme especificado, tem-se o quantitativo dos dados de treino, validação e teste para cada uma das abordagens dispostos conforme Tabela 4.2 e com proporcionalidades apresentadas nas Figuras 4.4a e 4.4b.

Tabela 4.2: Quantitativo de exemplos por finalidade na tarefa de aprendizado considerada e classe para cada abordagem.

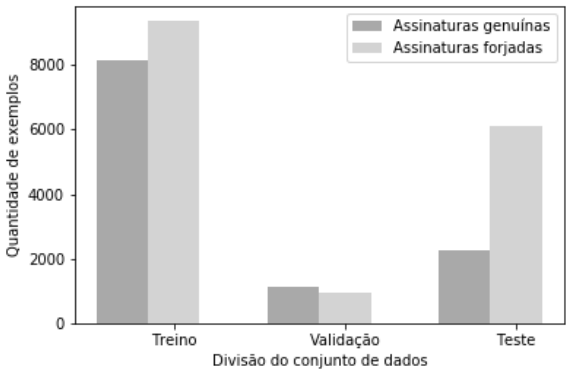
Conjunto	**Tipo de Exemplo**	**Abordagem A**		**Abordagem B**	
		Nº de Exemplos	**Proporção**	**Nº de Exemplos**	**Proporção**
Treinamento	Autêntico	9.374	54%	8.072	43%
	Forjado	8.131	46%	10.887	57%
Validação	Autêntico	947	46%	1.179	37%
	Forjado	1.134	54%	1.976	63%
Teste	Autêntico	2.257	27%	2.271	39%
	Forjado	6.119	73%	3.577	61%

Para a abordagem A, percebe-se que há uma certa desproporção entre as classes nas etapas de treino e validação. Na etapa de testes, esta diferença torna-se mais evidente e será refletida nas métricas de desempenho dos modelos. Quanto à abordagem B, tem-se uma notória desproporção nos conjuntos de validação e teste, isto devido à providência de considerar apenas a quantidade de autores e não a quantidade de exemplos durante a partição dos dados. Deste modo, esta diferença pode refletir tanto na parada precoce do treinamento quanto nas métricas de desempenho dos modelos.

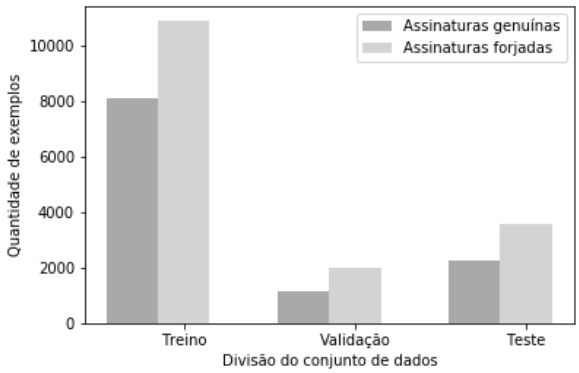
Ao serem fornecidas para treinamento pelas CNNs em etapa posterior, todos os *pixels* das imagens serão normalizados por meio de uma divisão por 255, passando a residirem no intervalo

Figura 4.4: Representação gráfica da proporção dos exemplos por classe e finalidade para as abordagens na tarefa de aprendizado considerada.

(a) Abordagem A.



(b) Abordagem B.



$[0, 1]$. Esta normalização é realizada em virtude das redes neurais que, em geral, aprendem mais eficientemente nestas condições (CHOLLET, 2017).

4.4 Modelos, Parâmetros e Hiperparâmetros de CNNs Considerados

Como visto anteriormente, propor arquiteturas eficientes de CNNs que obtenham bom desempenho em determinada tarefa de aprendizado é considerada uma atividade difícil. Para contornar esta dificuldade, para o problema considerado neste trabalho, escolheu-se então utilizar topologias canônicas de CNNs, que obtiveram bons desempenhos reportados pela literatura, mas promovendo ajustes em seus parâmetros e hiperparâmetros com a finalidade de buscar melhorias no desempenho para a tarefa de aprendizado aqui definida. Dentre estas arquiteturas canônicas, as selecionadas para o escopo deste trabalho encontram-se descritas a seguir:

- **LeNet**. Desenvolvida por LeCun em 1989, a arquitetura LeNet foi uns dos primeiros exemplos da aplicação de CNNs, tendo sido utilizada para a detecção de dígitos manuscritos utilizando o dataset MNIST (LECUN et al., 1998). Possui um total de 7 camadas e aproximadamente 7 milhões de parâmetros treináveis;

- **AlexNet**. Em 2012, a arquitetura AlexNet foi a primeira CNN a vencer o desafio ILSVRC da ImageNet, com uma boa margem de diferença dos outros modelos submetidos à competição. Para o seu treinamento com o conjunto de dados ImageNet, foram utilizadas duas GPUs de 3 GB de memória cada, que foram capazes de armazenar o processamento de aproximadamente 62 milhões de parâmetros (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; KHAN et al., 2018);

- **MobileNet**. Constituída por um conjunto de dois hiperparâmetros, esta arquitetura possui menor latência e um tamanho menor comparada às outras arquiteturas existentes, possuindo os requisitos que facilitam a sua implementação em aplicações para dispositivos móveis e embarcados (HOWARD et al., 2017). No *framework* `keras`, utilizado nas atividades realizadas neste trabalho, esta arquitetura possui uma profundidade de 88 camadas, 4.253.864 parâmetros e um tamanho de 17 MB (KERAS, 2019);

- **ShuffleNet**. Esta arquitetura se destaca pela sua composição de convoluções em grupo, na qual diversas convoluções são efetuadas paralelamente tomando porções dos canais de entrada, diminuindo de maneira eficiente o custo computacional. Posteriormente, os canais de saída da convolução em grupo são mesclados aleatoriamente através do processo chamado de *channel shuffle* (ZHANG et al., 2017);

- **SqueezeNet**. Foi desenvolvida em 2016 através de uma parceria entre os cientistas da DeepScale, University of California, Berkeley e Stanford University. A idéia foi criar uma arquitetura com o nível de acurácia da AlexNet com 50 vezes menos parâmetros e com um tamanho 0.5 MB menor, permitindo uma maior eficiência no treinamento em sistemas distribuídos, menor sobrecarga na exportação de modelos através da rede e sua capacidade de ajustar-se a sistemas com pouca memória (IANDOLA et al., 2016);

- **VGG-16**. Esta CNN, que consiste em 16 camadas convolucionais, possui arquitetura uniforme e é comumente muito utilizada na extração de características em imagens. Possuindo mais de 138 milhões de parâmetros e uma profundidade de 23 camadas, esta arquitetura possui um tamanho total de 528 MB no módulo `applications` do `keras`

(SIMONYAN; ZISSERMAN, 2015b; KERAS, 2019);

- **Inception**. Também chamada de GoogleNet, esta arquitetura é conhecida por ser a primeira a se desviar da forma padrão de simplesmente sequenciar camadas convolucionais e de *pooling*, criando os chamados blocos *Inception*. A sua versão InceptionV3 presente na biblioteca `keras` possui um total de 23.851.784 parâmetros e um tamanho de 92 MB (SZEGEDY et al., 2014; KERAS, 2019).

Em relação aos modelos adotados, considerou-se uma modificação na arquitetura geral apenas para compatibilizá-los ao problema considerado, que consiste em uma tarefa de aprendizado binária. Esta alteração diz respeito à camada de saída que consiste em apenas um neurônio com função de ativação sigmoidal, a qual retornará a probabilidade de pertencimento à cada classe na tarefa considerada.

Uma vez definidas as arquiteturas que serão utilizadas, define-se, em consequência, os parâmetros a serem adotados, que estão relacionados aos pesos destas redes, os quais serão obtidos via treinamento segundo *backpropagation*. Os hiperparâmetros, por sua vez, dizem respeito ao ajuste em nível de arquitetura das CNNs (CHOLLET, 2017). No escopo deste trabalho, considerou-se variações nos valores dos seguintes hiperparâmetros: otimizador para o cálculo do gradiente descendente, função de ativação das camadas intermediárias e *patience*, em que este último corresponde a um valor para interromper o treinamento da rede mediante *early stopping* a fim de evitar *overfitting*. Os valores adotados encontram-se dispostos na Tabela 4.3.

Tabela 4.3: Valores dos hiperparâmetros selecionados para a elaboração dos modelos.

Épocas	***Patience***	**Otimizador**	**Função de ativação**
200	5, 10 e 15	SGD, Adam e RMSprop	ReLU, ELU, SELU e Leaky ReLU

De maneira mais detalhada, com a adoção de *early stopping*, passou-se a monitorar uma métrica de desempenho durante o treinamento da rede, podendo esta ser a perda no conjunto de treinamento ou a acurácia no conjunto de validação, por exemplo. Com o uso de um valor de

patience, sempre que a métrica monitorada não melhorava durante o treinamento, decrementou-se uma unidade. O treinamento foi finalizado, então, quando este valor tornou-se igual a zero (CHOLLET, 2017). Os valores adotados para *patience*, no escopo deste trabalho, foram obtidos de maneira empírica em testes preliminares.

Um otimizador tem como objetivo aumentar o desempenho de um modelo de AM, ajustando os seus parâmetros com vistas a diminuir o erro encontrado na etapa de treinamento de tal modelo. Para o escopo deste trabalho, foram utilizados três otimizadores diferentes, sendo eles, o SGD (*stochastic gradient descent*), RMSprop e o Adam (do inglês *adaptive moment estimation*).

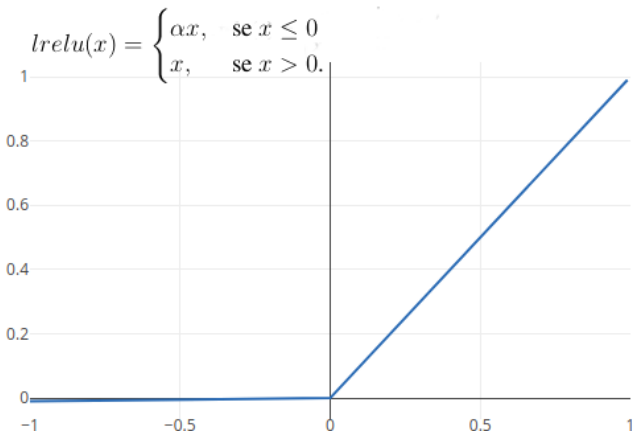
No SGD, a superfície de erro é estimada apenas com respeito a um único exemplo, tornando essa superfície dinâmica. Como resultado, descer nessa superfície melhora significativamente a habilidadede de navegar por regiões planas. O RMSprop, por sua vez, utiliza-se do valor do gradiente da função de custo e da sua raiz média quadrática para atualizar os pesos da CNN. O RMSProp tem mostrado ser um eficiente otimizador para redes neurais profundas e é a escolha principal para muitos praticantes experientes em criação de modelos de DL. E por fim, o otimizador Adam, é um algoritmo que pode ser considerado como a combinação do RMSprop com o *momentum*, que possui base em estimativas adaptativas de momentos de menor ordem. É computacionalmente eficiente e demanda poucos requisitos de memória, sendo adequado para para problemas que possuem grande quantidade dados ou parâmetros (BUDUMA, 2017; HINTON; SRIVASTAVA; SWERSKY, 2012; KINGMA; BA, 2014).

As funções de ativação ReLU e as suas variações *Leaky* ReLU, ELU (*Exponential Linear Unit*) e SELU (*Scaled Exponential Linear Unit*) foram escolhidas para estarem presentes nos neurônios das camadas internas das CNNs por auxiliarem na captura de relações não-lineares. Embora a função ReLU seja amplamente adotada, optou-se também por utilizar suas variações, pois para esta pode incorrer o chamado "*dying ReLU problem*", que acontece quando os neurônios com esta função de ativação tornam-se inativos e produzem apenas a saída zero para toda entrada (LU et al., 2019). Desta maneira, a sua variante *Leaky ReLU* mostrou ser uma boa escolha, pois possui um parâmetro adicional α, chamado de vazamento, que faz com que

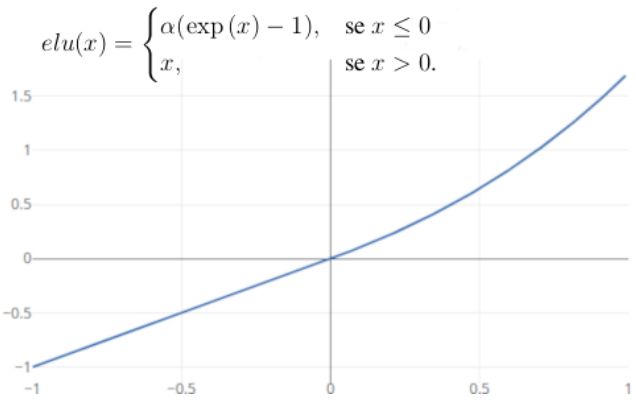
o gradiente seja pequeno, mas nunca nulo. A função ELU também é uma boa alternativa à ReLU pois diminui a mudança do *bias* ao pressionar a ativação média para zero. A SELU, por sua vez, possui uma auto-normalização, fazendo com que a aprendizagem seja altamente robusta e permitindo treinar redes com muitas camadas (PEDAMONTI, 2018). Na Figura 4.5 encontram-se os gráficos das variações da função ReLU.

Figura 4.5: Funções de ativação variantes da função ReLU.

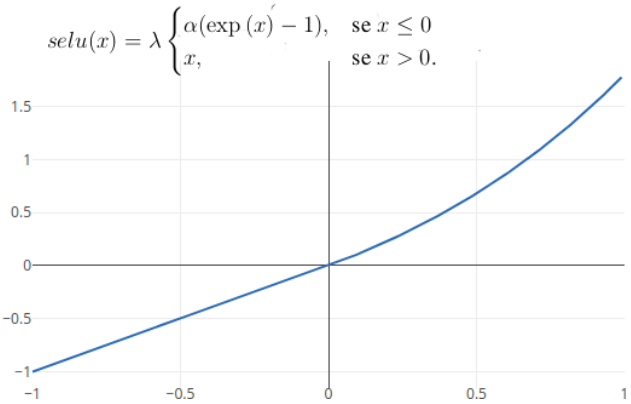
(a) *Leaky* ReLu



(b) ELU



(c) SELU



Sempre que os recursos computacionais e de tempo para o treinamento viabilizaram repetições, foram consideradas todas as variações possíveis dos hiperparâmetros descritos na Tabela 4.3. Nas demais situações foram consideradas escolhas de hiperparâmetros com melhor desempenho nos cenários que já tiverem sido executados.

Capítulo 5

Resultados e Discussão

Nesta seção serão apresentados os resultados obtidos com o treinamento e teste das arquiteturas canônicas consideradas para este trabalho. As Seções 5.1 e 5.2 contemplam os resultados obtidos com as arquitetura LeNet e AlexNet, respectivamente, as quais são arquiteturas bem consolidadas no âmbito de *Deep Learning*. Nas Seções 5.3, 5.4 e 5.5 estão presentes alguns resultados obtidos pelas arquiteturas MobileNet, ShuffleNet e SqueezeNet. Mais adiante, nas Seções 5.6 e 5.7, estão expostos os resultados alcançados pelas CNNs VGG-16 e Inception-V3, as arquiteturas mais profundas escolhidas para este trabalho. O treinamento destas CNNs foi realizado utilizando os recursos computacionais de um servidor, disponível no LSI, dedicado especialmente para tarefas de DL, o qual possui um processador Intel Core i7 com 16 GB de RAM e duas placas gráficas com 11 GB de memória cada, das quais apenas uma foi utilizada.

Após a etapa de treino, foram realizados os testes para aferir os modelos no tocante às métricas de desempenho para o conjunto de testes. Nesta etapa, percebeu-se que alguns modelos tornaram-se degenerados e acabaram prevendo apenas uma das classes. Duas hipóteses podem justificar a ocorrência desse problema: o ReLU *dying problem*, quando a função de ativação ReLU foi utilizada; ou a tendência à permanência em mínimos locais durante o treinamento do modelo. Todas as CNNs que manifestaram este comportamento no conjunto de testes tiveram seus resultados descartados, pois as métricas obtidas não refletiam aprendizado no problema considerado.

5.1 Resultados Obtidos com a CNN LeNet

A primeira fase do treinamento dos modelos foi conduzida utilizando a arquitetura LeNet. Nesta fase, foi realizada uma busca em *grid* por todos os hiperparâmetros previamente definidos, conforme Seção 4.4, e considerando as duas abordagens definidas conforme a Seção 4.3, gerando um total de 72 modelos a serem treinados e testados. Para estes modelos, excluindo aqueles que se tornaram degenerados, utilizou-se a métrica *F-Score* como referência para um melhor desempenho.

O melhor dos modelos baseados na arquitetura LeNet para cada uma das abordagens consideradas encontram-se dispostos na Tabela 5.1, juntamente com os hiperparâmetros utilizados pelos mesmos.

Tabela 5.1: Detalhamento dos melhores resultados obtidos com a arquitetura LeNet.

Abordagem	**Otimizador**	***Patience***	**Função de Ativação**	**Acurácia**	***F-Score***	**EER**
Abordagem A	RMSprop	5	ReLU	0.9865	0.9755	1.1679
Abordagem B	Adam	10	ELU	0.8361	0.8159	12.5245

Os gráficos da Figura 5.1 denotam o histórico da perda (*loss*) e acurácia para o conjunto de treinamento e validação destas redes. Nota-se que nenhuma delas chegou ao limite máximo de épocas possíveis, interrompendo o aprendizado por meio de *early stopping*, comportamento este que também fez-se presente em todas as outras redes treinadas com esta arquitetura.

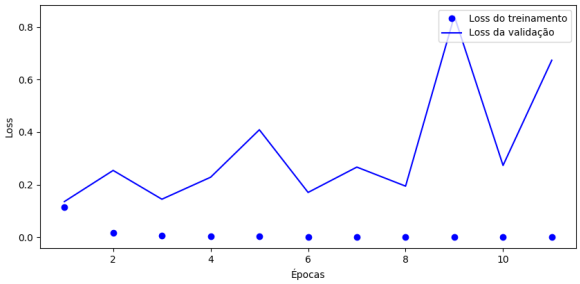
Examinando mais atentamente o desempenho destas redes no conjunto de testes, tem-se, então, as matrizes de confusão mostradas na Figura 5.2. Nestas matrizes, a soma das linhas representam a quantidade de assinaturas previstas para cada classe pelo modelo em questão, enquanto a soma das colunas denotam a quantidade de assinaturas existentes em cada classe.

Para esta arquitetura, é possível visualizar que, dentre os dois modelos tidos como melhores, não há qualquer semelhança entre os hiperparâmetros encontrados. O número de épocas para aprendizado de características foi baixo para abordagem A, enquanto que, para a abordagem B, houve a necessidade de mais épocas de treinamento. Isso aconteceu, possivelmente, pela aparição de um mesmo autor em partições de dados diferentes na abordagem A.

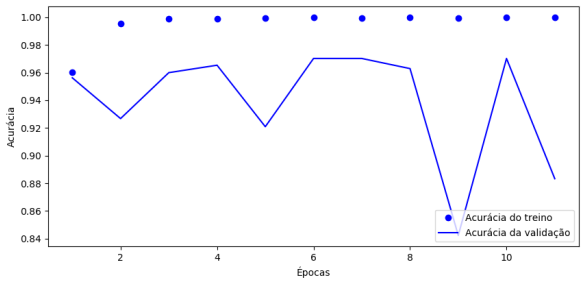
A partir das matrizes de confusão, percebeu-se que o melhor modelo da abordagem A

Figura 5.1: Histórico de *loss* e acurácia durante o treinamento dos melhores modelos obtidos com a arquitetura LeNet.

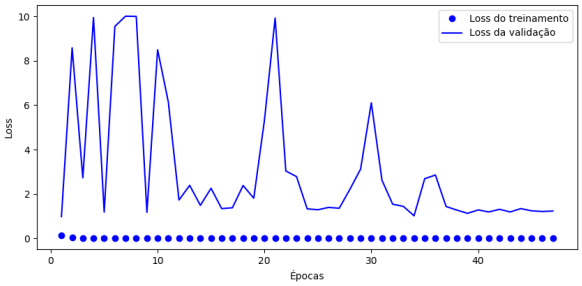
(a) *Loss* durante treinamento da melhor rede LeNet com a abordagem A.



(b) Acurácia durante treinamento da melhor rede LeNet com a abordagem A.



(c) *Loss* durante treinamento da rede LeNet B.



(d) Acurácia durante treinamento da rede LeNet B.

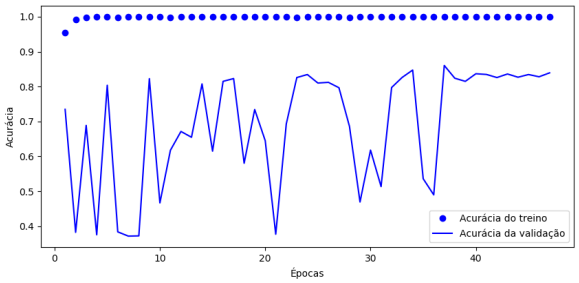
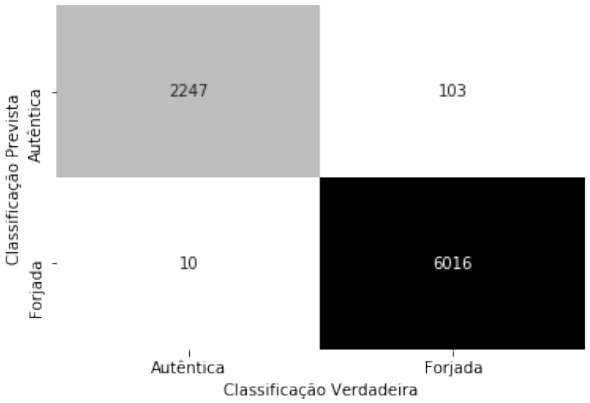
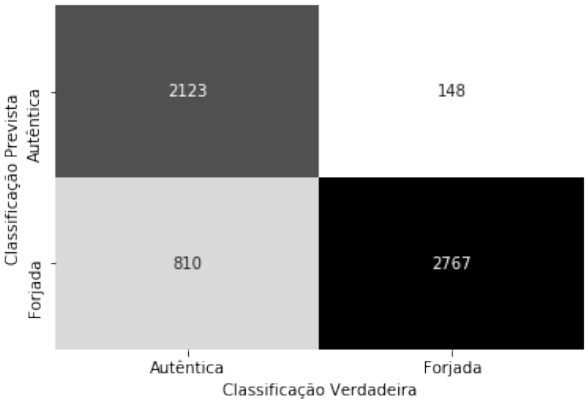


Figura 5.2: Matrizes de confusão dos melhores modelos obtidos com a arquitetura LeNet.

(a) Melhor LeNet com a abordagem A



(b) Melhor LeNet com a abordagem B



tendeu a verificar um baixo número de falsos negativos, ou seja, os exemplos autênticos foram suficientes para identificar assinaturas originais, indepentemente das variações cometidas por este autor. Por outro lado, para o modelo da abordagem B, verificou-se um número menor de falsos positivos, validando que, apesar da boa capacidade de certos forjadores *over-the-shoulder* em realizar reproduções verossímeis, este modelo mostrou uma alta competência na detecção

deste tipo de assinaturas. Não obstante, nota-se a diagonal principal de ambos os modelos bastante densa, sugerindo uma boa adequação para a tarefa considerada.

5.2 Resultados Obtidos com a CNN AlexNet

Para a AlexNet, assim como para a CNN anterior, foi realizada uma busca em *grid* para ambas as abordagens, utilizando os hiperparâmetros selecionados anteriormente, gerando assim, mais 72 modelos a serem avaliados quanto às suas métricas de desempenho.

Considerando a métrica de *F-Score*, foram selecionados os melhores modelos desta arquitetura e estes encontram-se listados na Tabela 5.2. Na Figura 5.3 pode-se observar os gráficos com os comportamentos dos valores de *loss* e acurácia encontrados nos conjuntos de treinamento e validação durante o estágio de treino destes modelos.

Tabela 5.2: Detalhamento dos melhores modelos obtidos com a arquitetura AlexNet para cada uma das abordagens consideradas neste trabalho.

Abordagem	**Otimizador**	***Patience***	**Função de Ativação**	**Acurácia**	**F-Score**	**EER**
Abordagem A	Adam	15	ELU	0.9654	0.9393	1.5401
Abordagem B	RMSprop	5	ELU	0.8593	0.7993	13.8265

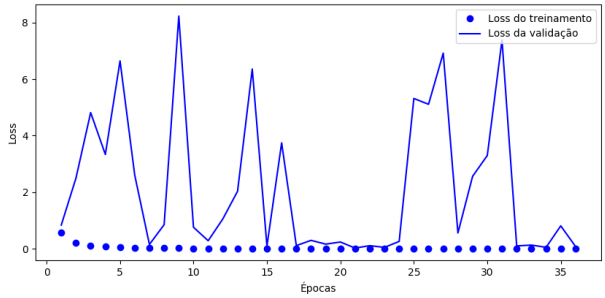
Considerando que estas redes possuem mais parâmetros treináveis, este possivelmente foi um fator responsável pelo maior números de épocas no treinamento na abordagem A. Nota-se ainda que houve oscilações nos treinamentos, resultando em parada precoce. Para esta arquitetura, as redes treinadas com a função de ativação ELU mostraram métricas melhores na etapa de avaliação.

Observando as métricas de acurácia e *F-Score* obtidas, percebe-se que estas foram inferiores às observadas para as redes LeNet, mas ainda assim alcançando valores superiores a 90% na abordagem A e valores próximos a 80% na abordagem B.

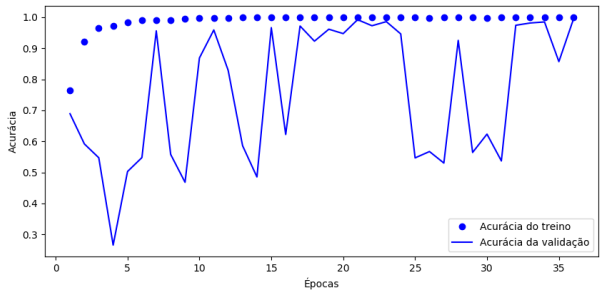
Examinando mais atentamente o desempenho destas redes no conjunto de testes, tem-se as matrizes de confusão mostradas na Figura 5.4.No melhor modelo da abordagem B, para esta arquitetura, a disposição dos valores na matriz de confusão mostra uma reflexão diferente da encontrada no cenário LeNet. Neste caso, houve uma quantidade maior de falsos positivos,

Figura 5.3: Histórico de *loss* e acurácia durante o treinamento dos melhores modelos obtidos com a arquitetura AlexNet.

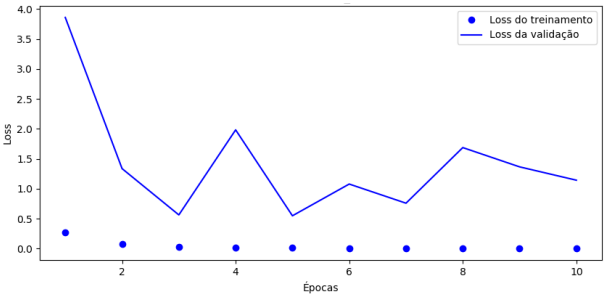
(a) *Loss* durante treinamento da melhor rede AlexNet para a abordagem A.



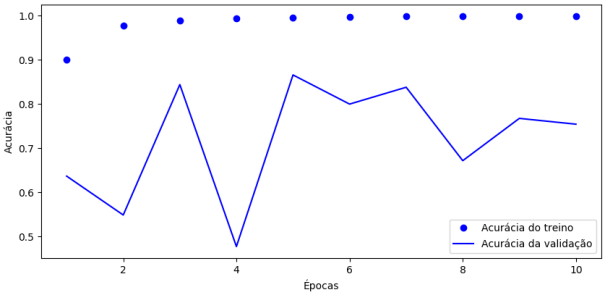
(b) Acurácia durante treinamento da melhor rede AlexNet para a abordagem A.



(c) *Loss* durante treinamento da melhor rede AlexNet para a abordagem B.



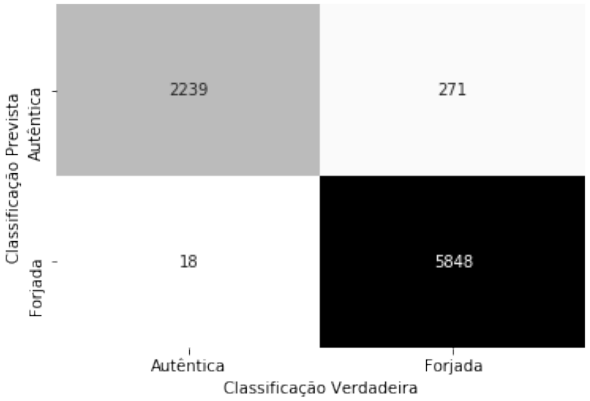
(d) Acurácia durante treinamento da melhor rede AlexNet para a abordagem B.



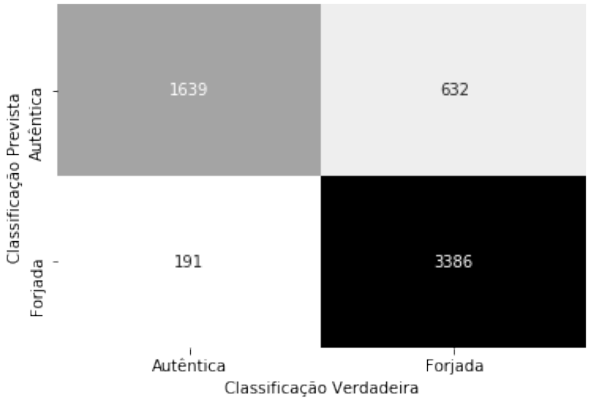
mostrando a dificuldade na detecção das assinaturas *over-the-shoulder* realizadas pelos autores forjadores.

Figura 5.4: Matrizes de confusão dos melhores modelos obtidos com a arquitetura AlexNet.

(a) Melhor AlexNet para a abordagem A



(b) Melhor AlexNet para abordagem B



De modo geral, apesar de possuir boas métricas, o melhor modelo encontrado pela arquitetura AlexNet, com um *F-score* de 0.9393, não foi suficiente para superar o melhor modelo

obtido com a arquitetura LeNet (0.9755). Uma vez que a arquitetura LeNet possui menos parâmetros que a AlexNet e melhor desempenho observado, ressalta-se a sua maior adequação para a tarefa considerada, acrescido ao fato de demandar menos recursos de tempo de treinamento e de memória para seu armazenamento.

5.3 Resultados Obtidos com a CNN MobileNet

Seguindo para os resutlados das arquiteturas com poucos parâmetros, a primeira destas a ser treinada e testada foi a MobileNet. Para esta arquitetura, considerando ambas as abordagens, foi realizada mais uma vez uma busca em *grid* de modelos utilizando todos os hiperparâmetros especificados anteriormente, gerando um total de 72 modelos diferentes.

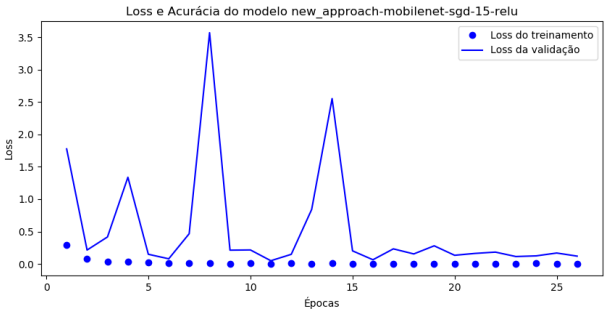
Nesta arquitetura em particular, foi considerada a métrica de EER para a escolha dos melhores modelos e estes encontram-se presentes na Tabela 5.3. Na Figura 5.5 pode-se observar os gráficos com o comportamento da *loss* e acurácia dos conjuntos de treino e validação durante a etapa de ajustamento dos modelos.

Tabela 5.3: Detalhamento dos melhores modelos obtidos com a arquitetura MobileNet para cada uma das abordagens consideradas neste trabalho.

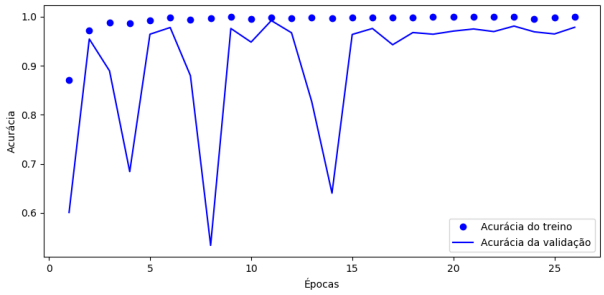
Abordagem	**Otimizador**	***Patience***	**Função de Ativação**	**Acurácia**	**F-Score**	**EER**
Abordagem A	SGD	15	ReLU	0.9606	0.9318	0.9304
Abordagem B	Adam	15	ReLU	0.8856	0.8658	9.9475

Figura 5.5: Histórico de *loss* e acurácia durante o treinamento dos melhores modelos obtidos com a arquitetura MobileNet.

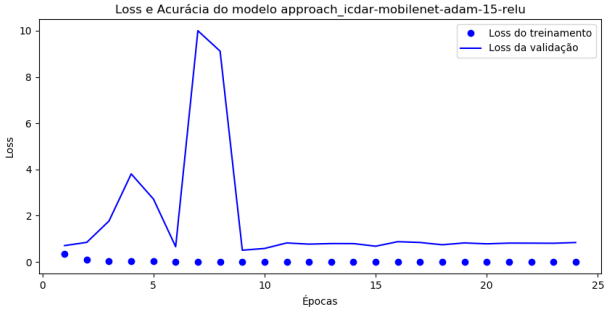
(a) *Loss* durante treinamento da melhor rede MobileNet para a abordagem A.



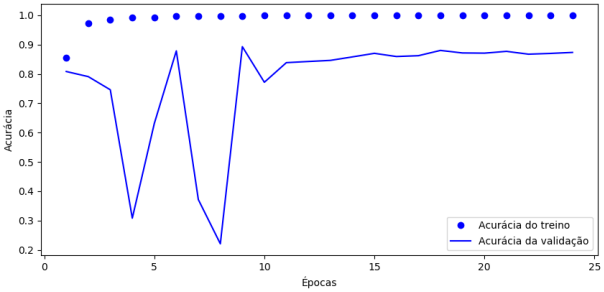
(b) Acurácia durante treinamento da melhor rede MobileNet para a abordagem A.



(c) *Loss* durante treinamento da melhor rede MobileNet para a abordagem B.



(d) Acurácia durante treinamento da melhor rede MobileNet para a abordagem B.

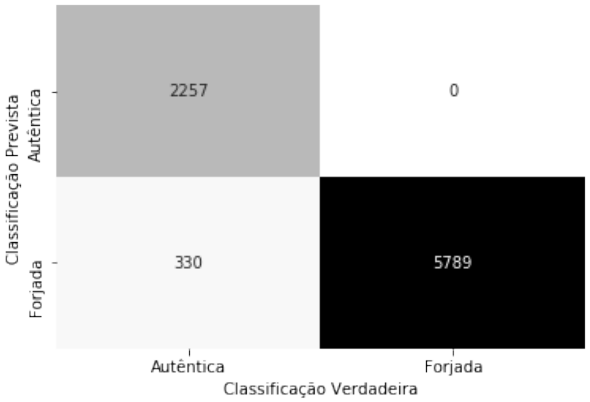


Dentre as CNNs observadas até o momento, a MobileNet foi a que obteve um melhor desempenho. Isso se deve, possivelmente, aos poucos padrões encontrados nas imagens de treinamento, por se tratar de imagens em escala de cinza, e à existência de poucos parâmetros treináveis para esta arquitetura. Essas duas características se fizeram essenciais para a descoberta de modelos superiores aos demais, quando analisando apenas a métrica de EER.

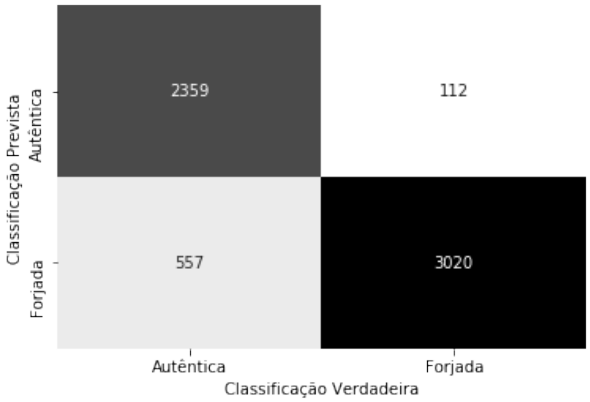
Na Figura 5.6 pode-se visualizar as matrizes de confusão obtidas pelos melhores modelos. Percebe-se que, para abordagem A, apesar da diagonal principal densa, a quantidade de falsos negativos foi maior do que o encontrado nas arquiteturas anteriores para a mesma abordagem. Quanto à matriz encontrada para a abordagem B, podemos considerar as mesmas reflexões concebidas à matriz da arquitetura LeNet.

Figura 5.6: Matrizes de confusão dos melhores modelos obtidos com a arquitetura MobileNet.

(a) Melhor MobileNet com a abordagem A



(b) Melhor MobileNet com a abordagem B



5.4 Resultados Obtidos com a CNN ShuffleNet

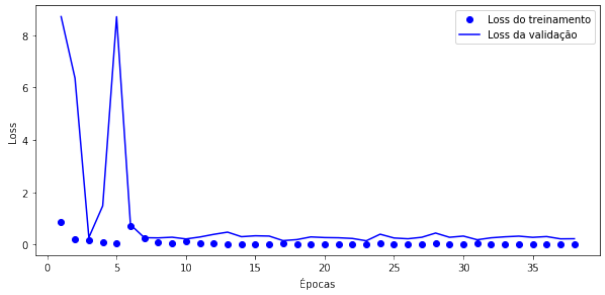
Para a ShuffleNet, não verificou-se a necessidade de uma busca em *grid* entre os hiperparâmetros, considerando os bons resultados apresentados pela MobileNet e pela semelhança existente entre estas CNNs. Entretanto, para demonstrar a eficiência da ShuffleNet, foram escolhidos hiperparâmetros de maneira *ad hoc*, baseando-se apenas naqueles hiperparâmetros que demonstraram um bom desempenho nas arquiteturas anteriores. Por conseguinte, foram treinados dois modelos com os mesmos hiperparâmetros, um para cada abordagem, os quais obtiveram as métricas dispostas na Tabela 5.4. O histórico de *loss* e acurácia durante o ajustamento dos modelos estão retratados na Figura 5.7.

Tabela 5.4: Detalhamento dos modelos obtidos com a arquitetura ShuffleNet para cada uma das abordagens consideradas neste trabalho.

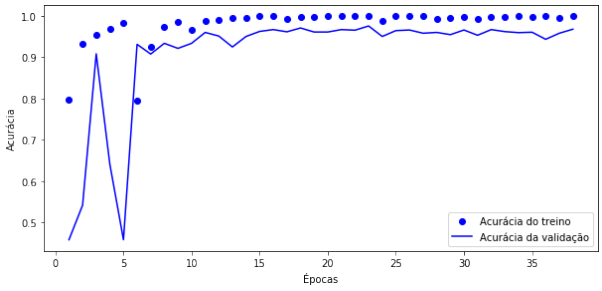
Abordagem	**Otimizador**	***Patience***	**Função de Ativação**	**Acurácia**	**F-Score**	**EER**
Abordagem A	RMSprop	15	ReLU	0.9404	0.9004	7.5400
Abordagem B	RMSprop	15	ReLU	0.8345	0.7705	23.8151

Figura 5.7: Histórico de *loss* e acurácia durante o treinamento dos modelos obtidos com a arquitetura ShuffleNet.

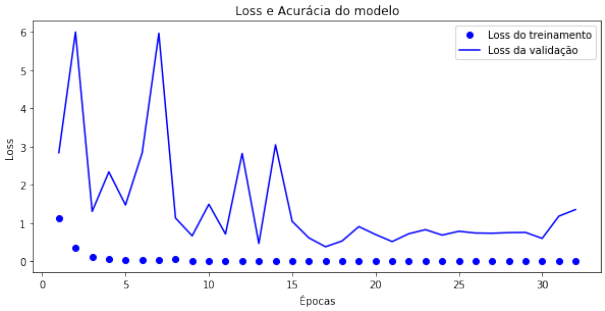
(a) *Loss* durante treinamento da rede ShuffleNet para a abordagem A.



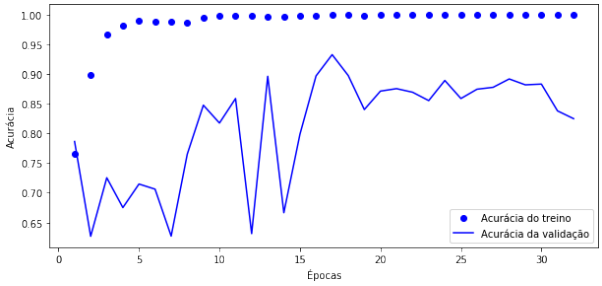
(b) Acurácia durante treinamento da rede ShuffleNet para a abordagem A.



(c) *Loss* durante treinamento da rede ShuffleNet para a abordagem B.



(d) Acurácia durante treinamento da rede ShuffleNet para a abordagem B.



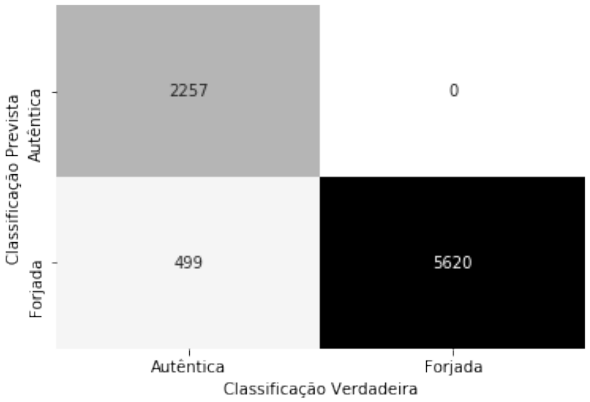
Ao analisar as métricas obtidas por esta arquitetura, observa-se um desempenho inferior ao observado pelas arquiteturas anteriores, isto se deve, presumivelmente, à subexploração dos hiperparâmetros possíveis, principalmente considerando que modelos da arquitetura MobileNet não expostos neste trabalho, por exemplo, obtiveram métricas similares ou inferiores.

As matrizes de confusão destes modelos, dispostas na Figura 5.8, mostram uma grande quantidade de falsos negativos e nenhum falso positivo para a abordagem A. Este cenário mostra uma grande eficiência do modelo em detectar assinaturas falsificadas. Na matriz da abordagem B, por outro lado, o quantitativo de falsos positivos foi por volta de 40% maior do que a quantidade de falsos negativos. Esta quantidade de erros cometido pelo classificador é refletida diretamente na métrica EER, a qual possui o maior valor encontrado entre os modelos expostos até então.

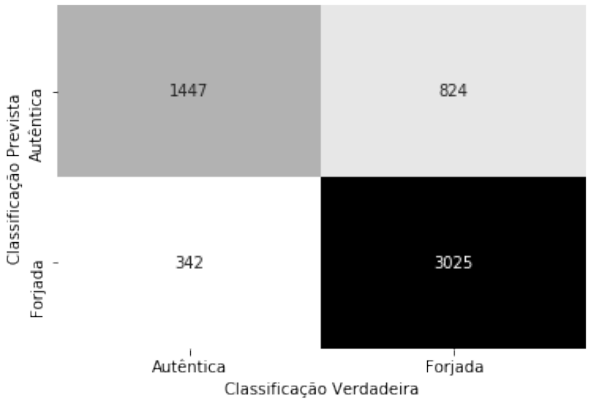
Ao que tudo indica, a utilização da arquitetura ShuffleNet, associada a uma busca de bons hiperparâmetros, pode ser bem aproveitada para a tarefa de aprendizado apresentada neste

Figura 5.8: Matrizes de confusão dos modelos obtidos com a arquitetura ShuffleNet.

(a) ShuffleNet com a abordagem A



(b) ShuffleNet com a abordagem B



trabalho, observando principalmente as necessidades e especificações de um sistema em um cenário de aplicação real.

5.5 Resultados Obtidos com a CNN SqueezeNet

Seguindo então para a última das arquiteturas com poucos parâmetros designadas para este trabalho, tem-se os resultados da SqueezeNet dispostos na Tabela 5.5. Mais uma vez a busca em *grid* foi descartada para esta CNN, pelos mesmos motivos considerados para a arquitetura anterior, realizando as mesmas escolhas de hiperparâmetros.

Tabela 5.5: Detalhamento dos modelos obtidos com a arquitetura SqueezeNet para cada uma das abordagens consideradas neste trabalho.

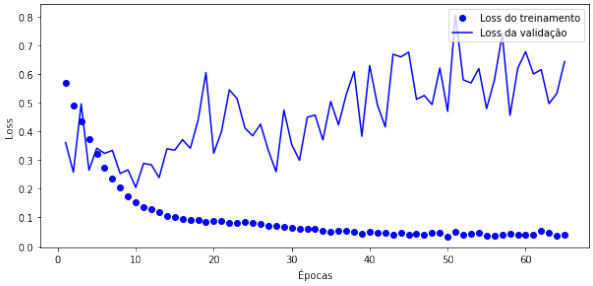
Abordagem	**Otimizador**	***Patience***	**Função de Ativação**	**Acurácia**	**F-Score**	**EER**
Abordagem A	RMSprop	15	ReLU	0.9048	0.8948	11.5074
Abordagem B	RMSprop	15	ReLU	0.8210	0.7709	20.1673

O histórico de *loss* e acurácia durante o estágio de treinamento destes modelos, disponível na Figura 5.9, mostra que, para a abordagem A, a quantidade de épocas atingidas pelo modelo, antes de ocorrer o *early stopping* definido, foi a maior entre todos os modelos aqui representados. Não obstante, é possível notar, também para o modelo da abordagem A, que a sua convergência ocorreu de uma forma mais padronizada, sem muitos altos e baixos na acurácida do conjunto

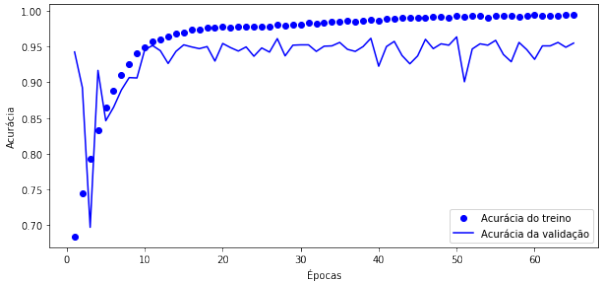
de validação durante o treinamento.

Figura 5.9: Histórico de *loss* e acurácia durante o treinamento dos modelos obtidos com a arquitetura SqueezeNet.

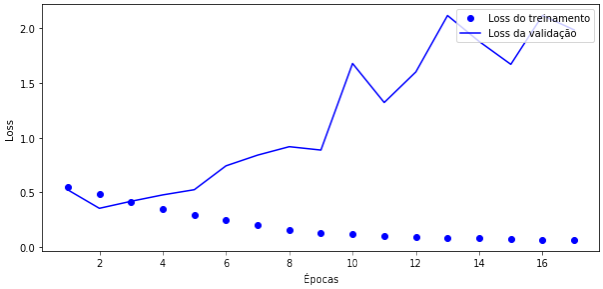
(a) *Loss* durante treinamento da rede SqueezeNet para a abordagem A.



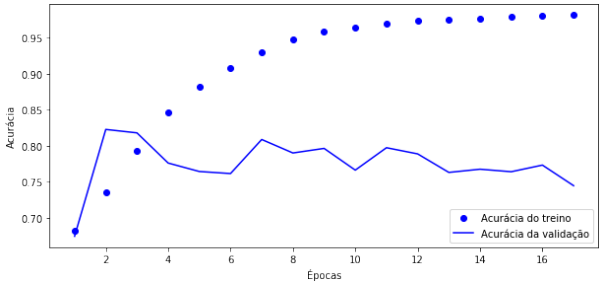
(b) Acurácia durante treinamento da rede SqueezeNet para a abordagem A.



(c) *Loss* durante treinamento da rede SqueezeNet para a abordagem B.



(d) Acurácia durante treinamento da rede SqueezeNet para a abordagem B.

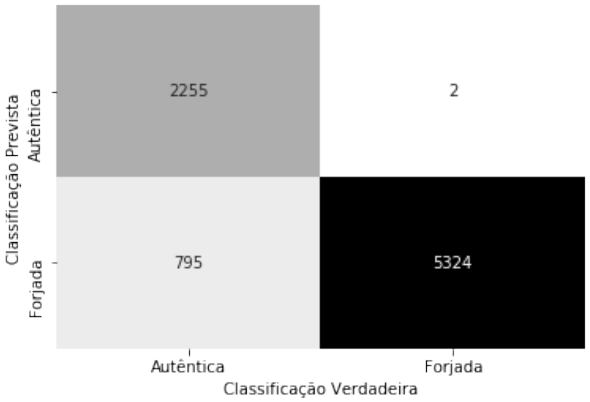


Para uma análise mais intensa destes modelos, foram geradas suas matrizes de confusão, demonstradas na Figura 5.10. No modelo da abordagem A, é possível utilizar as mesmas reflexões quanto às matrizes obtidas pelas arquiteturas MobileNet e ShuffleNet. Quanto a abordagem B, pode-se verificar a existência de valores parecidos na diagonal secundária da matriz, indicando a presença quase similar de falsos positivos e falsos negativos.

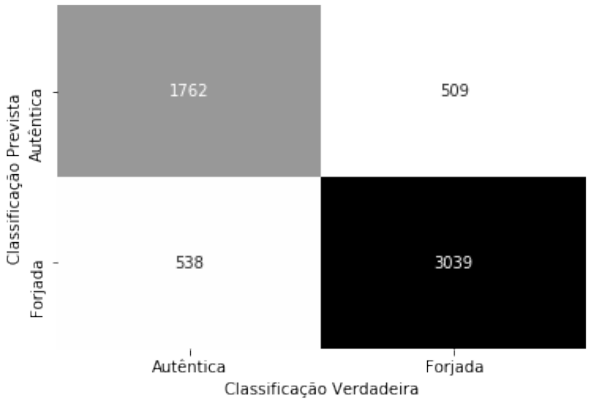
De maneira geral, apesar do desempenho dos modelos aqui demonstrados terem sido insatisfatórios, é possível perceber que existe ainda a possibilidade de uma busca em *grid* de hiperparâmetros, buscando um modelo com um desempenho similar ou superior aos encontrados até então.

Figura 5.10: Matrizes de confusão dos modelos obtidos com a arquitetura SqueezeNet.

(a) SqueezeNet com a abordagem A



(b) SqueezeNet com a abordagem B

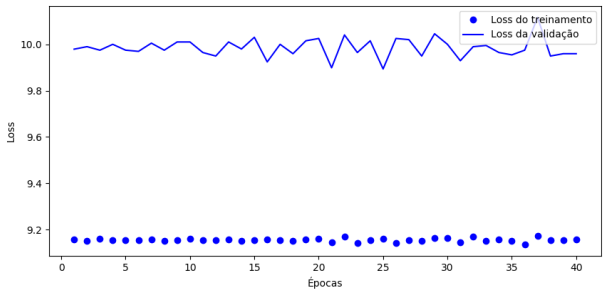


5.6 Resultados Obtidos com a CNN VGG-16

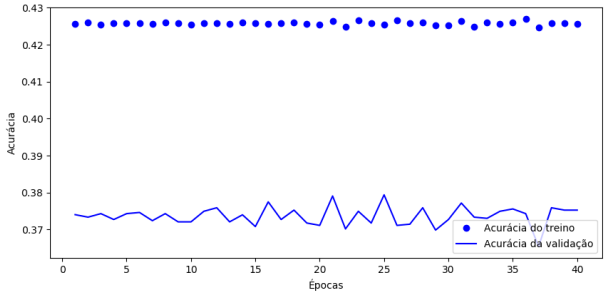
Dentre as CNNs mais profundas, a primeira a ser considerada para estar presente neste trabalho foi a VGG-16. Para esta arquitetura, foi feita a escolha de hiperparâmetros de maneira arbitrária. Primeiramente, preferiu-se trabalhar com a função de ativação ReLU, porém, após a percepção da não convergência dos modelos ajustados com esta função, devido ao *Dying ReLU problem*. A demonstração de *loss* e acurácia durante o treinamento deste primeiro modelo adotado pode ser visualizada na Figura 5.11.

Figura 5.11: Histórico de *loss* e acurácia durante o treinamento de um modelo degenerado obtido com a arquitetura VGG-16.

(a) *Loss* durante treinamento da rede VGG-16 degenerada.



(b) Acurácia durante treinamento da rede VGG-16 degenerada.



Após estes resultados, consequentemente, optou-se por utilizar a função de ativação ELU para a obtenção de apenas um modelo para a abordagem B. A escolha do desenvolvimento de

uma única rede VGG se deu devido ao longo tempo de treinamento de modelos para este tipo de arquitetura.

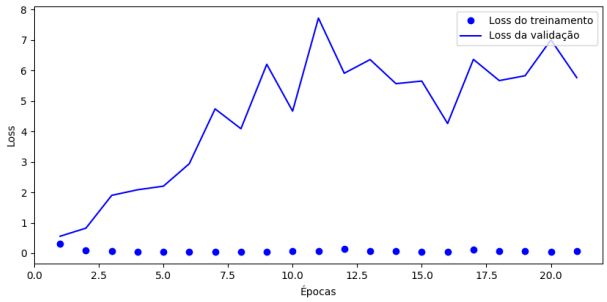
Com a intenção de obter bons resultados, durante o treinamento do modelo, foi utilizado o método *model checkpoint* do Keras, o qual salva os pesos do modelo quando a acurácia do conjunto de validação, neste caso, é a melhor encontrada durante todo o processo de treinamento. Utilizando o último modelo salvo na etapa de testes, tem-se as métricas e os hiperparâmetros expostos na Tabela 5.12. A Figura 5.12 mostra o histórico de *loss* e acurácia deste único modelo.

Tabela 5.6: Detalhamento do modelo obtido com a arquitetura VGG-16 para a abordagem B.

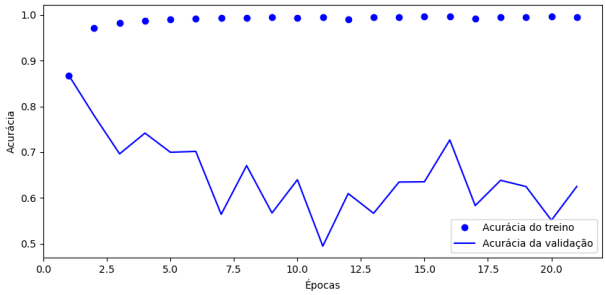
Otimizador	***Patience***	**Função de Ativação**	**Acurácia**	**F-Score**	**EER**
RMSprop	10	ELU	0.8391	0.8019	16.1096

Figura 5.12: Histórico de *loss* e acurácia durante o treinamento do modelo obtido com a arquitetura VGG-16.

(a) *Loss* durante treinamento da rede VGG-16 para a abordagem B.



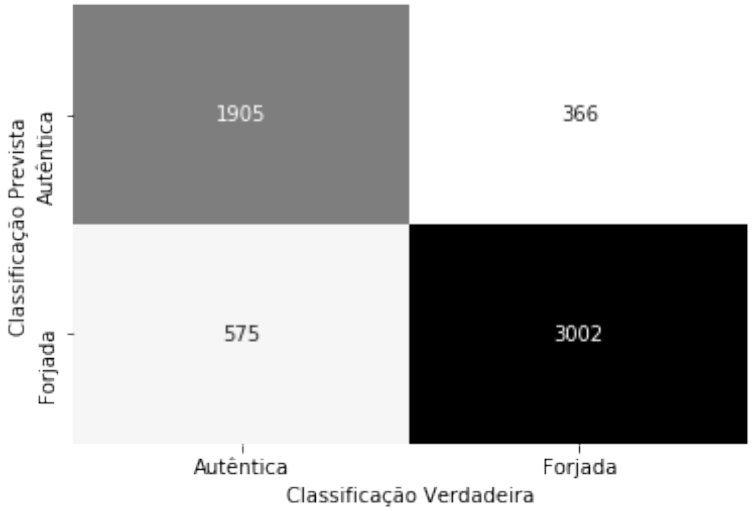
(b) Acurácia durante treinamento da rede VGG-16 para a abordagem B.



A matriz de confusão deste modelo, disposta na Figura 5.13, mostra mais uma vez uma diagonal principal bastante densa, enquanto que a diagonal secundária está bem distribuída entre previsões forjadas e autênticas. Estes valores estão refletidos nas métricas de desempenho apresentadas na Tabela 5.6.

Em suma, este modelo pôde mostrar que esta arquitetura pode ser útil para lidar com a tarefa de aprendizado proposta. Contudo, cabe observar que a utilização desta CNN pode ser demais para trabalhar com imagens com poucos padrões, como é o caso.

Figura 5.13: Matriz de confusão do modelo obtido com a arquitetura VGG-16.



5.7 Resultados Obtidos com a CNN Inception-V3

Seguindo para a última das arquiteturas de CNNs exploradas neste trabalho, temos a Inception-V3. Como na CNN anterior, foi treinado apenas um modelo com a abordagem B, a qual se aplicam as mesmas justificativas para tal.

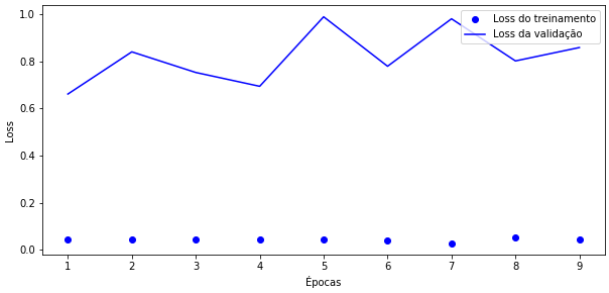
Os hiperparâmetros, definidos de forma arbitrária, e as métricas obtidas para este modelo encontram-se na Tabela 5.7. Uma visualização da *loss* e acurácia durante o treinamento estão caracterizados na Figura 5.14.

Tabela 5.7: Detalhamento do modelo obtido com a arquitetura Inception-V3 para a abordagem B.

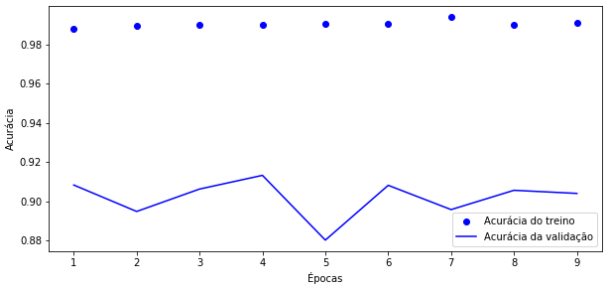
Otimizador	***Patience***	**Função de Ativação**	**Acurácia**	**F-Score**	**EER**
RMSprop	5	ELU	0.8394	0.8070	16.9493

Figura 5.14: Histórico de *loss* e acurácia durante o treinamento do modelo obtido com a arquitetura Inception-V3.

(a) *Loss* durante treinamento da rede Inception-V3 para a abordagem B.



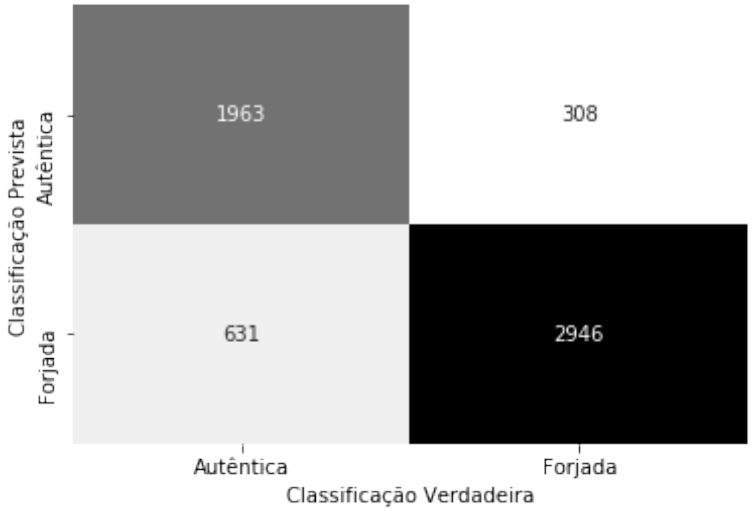
(b) Acurácia durante treinamento da rede Inception-V3 para a abordagem B.



É possível examinar a parada precoce do treinamento, levando apenas 9 épocas antes de chegar ao final. Isto se deve, provavelmente, à capacidade da Inception de detectar os padrões das imagens, o que pode levar a um *overfitting* ao conjunto de treinamento que, consequentemente, diminui o valor da acurácia no conjunto de validação.

A matriz de confusão exibida na Figura 5.15 diz respeito ao modelo treinado com esta arquitetura. Tem-se um resultado similar ao observado na arquitetura VGG-16, portanto, cabem aqui as mesmas ponderações demonstradas anteriormente.

Figura 5.15: Matriz de confusão do modelo obtido com a arquitetura Inception-V3.



Por fim, foi possível comprovar que esta arquitetura consegue prestigiar nosso problema com resultados satisfatórios. Porém, novamente, fica aberta a reflexão sobre a necessidade da

utilização de uma arquitetura tão profunda para tratar um problema que foi adequadamente ajustado por CNNs mais rasas. Não obstante, pode ser realizada uma busca exploratória sobre os hiperparâmetros, com vistas à descoberta de resultados superiores aos observados nestas arquiteturas rasas.

Capítulo 6

Considerações Finais

O objetivo desta proposta de trabalho de conclusão de curso consistiu em endereçar o problema de autenticação de assinaturas manuscritas considerando a perspectiva de Aprendizado de Máquina utilizando Redes Neurais Convolucionais. Para isto, foi selecionado um conjunto de dados contendo assinaturas forjadas e genuínas, o qual foi preparado para a tarefa de interesse, contendo 27.962 exemplos. Estes dados foram particionados em conjuntos de treino, teste e validação considerando duas abordagens distintas. Na primeira delas, chamada de abordagem A, houve a ressalva de que apenas falsificações inéditas compuseram o conjunto de teste. Na abordagem B, por outro lado, definiu-se que apenas assinaturas de autores inéditos estariam presentes no conjunto de teste. Ambas as abordagens foram concepcionadas visando aproximar a avaliação de cenários realísticos. Estas bases de dados foram então utilizadas para treinamento e teste de arquiteturas de redes neurais convolucionais bem estabelecidas na literatura.

Dentre as arquiteturas escolhidas para fazer parte deste trabalho estão a LeNet, AlexNet, MobileNet, ShuffleNet, SqueezeNet, VGG-16 e Inception-V3, as quais algumas destas passaram por uma busca em *grid* que combinou vários valores de hiperparâmetros e as diferentes abordagens, culminando no treinamento e teste de um total de 222 modelos. Dentre estes, aquele que resultou em um melhor desempenho, considerando apenas a abordagem A, é caracterizado pela arquitetura LeNet, utilizando o otimizador RMSProp, possui *patience* igual a 5 e utiliza função de ativação *Leaky* ReLU. Este modelo obteve uma acurácia de 98.65% e um valor de *F-score* igual a 0.9755. Se considerada apenas a habilidade deste modelo em identificar falsifi-

cações, ignorando-se os resultados obtidos para assinaturas autênticas, tem-se um *F-Score* igual a 0.9915 para esta habilidade[1]. Enquanto que para a abordagem B, o melhor modelo obtido, é identificado pela arquitetura MobileNet, utiliza-se do otimizador Adam, possui um valor de *patience* igual a 15 e a função de ativação ReLU. Obteve uma acurácia de 88.56%, *F-Score* de 86.58% e um EER de 9.94%.

Além do bom desempenho obtido, ressalta-se que as arquiteturas associadas a estes modelos são CNNs que possuem poucos parâmetros quando comparadas a outras avalidadas neste trabalho, o que agrega um valor ainda maior aos resultados obtidos em virtude do menor esforço computacional para realização de previsões e menor espaço em disco para armazenamento.

Considerando o bom desempenho conquistado nesta tarefa e demonstrando a adequação dos modelos para o que foi proposto, sugere-se a remoção seletiva das camadas das CNNs pequenas e com bom desempenho nesta tarefa com vistas a obter modelos ainda mais compactos, mas que ainda possuam bom desempenho, o que pode vir a facilitar o encapsulamento da solução proposta em dispositivos móveis com requisitos mais restringentes de processamento e memória. Ademais, investigar a obtenção de mapas de calor a partir das camadas convolucionais pode colaborar com revisores humanos no processo de checagem de assinaturas, aumentando ainda mais a confiabilidade de soluçõesn nesta natureza.

O problema em questão é significativo do ponto de vista prático pois pode colaborar, por exemplo, para a autenticação de documentos de maneira automática e confiável diminuindo os recursos humanos especializados para este fim. Do ponto de vista do bacharel em Engenharia de Computação que desenvolveu este trabalho, construir uma solução para este problema foi a oportunidade de pôr em prática diversos conceitos aprendidos ao longo do curso, principalmente aqueles presentes nas disciplinas de Inteligência Artificial, *Machine Learning*, Redes Neurais Artificiais, Linguagem de Programação, Sinais e Sistemas e Processamento Digital de Imagens.

[1]Calculou-se este *F-Score* tomando o valor 6016 como sendo de verdadeiros positivos e o valor 103 como sendo de falsos negativos.

Referências Bibliográficas

ACADEMY, D. S. *Deep Learning Book*. 2019. Disponível em <http://deeplearningbook.com.br/>. Acesso em 8 de março de 2019.

ARAUJO, N. P. de. Estimação inteligente de idade utilizando deep learning. Trabalho de Conclusão de Curso da Universidade do Estado do Amazonas, Universidade do Estado do Amazonas, Manaus, BR, 2018.

ARBIB, M. A. (Ed.). *The Handbook of Brain Theory and Neural Networks*. Cambridge, Massachussets: The MIT Press, 2003.

BAJAJ, R.; CHAUDHURY, S. Signature verification using multiple neural classifiers. *Pattern Recognition*, v. 30, n. 1, p. 1 – 7, 1997. ISSN 0031-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0031320396000593>.

BLANKERS, V. L. et al. The icdar 2009 signature verification competition. In: *10th International Conference on Document Analysis and Recognition*. Barcelona, Catalonia, Spain: IEEE, 2009. p. 1403–1407.

BRAGA, A. de P.; CARVALHO, A. P. de Leon F. de; LUDERMIR, T. B. *Redes Neurais Artificiais: Teorias e Aplicações*. Rio de Janeiro, RJ: Livros Técnicos e Científicos Editora S.A., 2000.

BRINK, H.; RICHARDS, J.; FETHEROLF, M. *Real-World Machine Learning*. New York, US: Manning Publications, 2016.

BUDUMA, N. *Fundamentals of Deep Learning*. Estados Unidos: O'Reilly Media, Inc., 2017.

CHA, K. H. et al. Urinary bladder segmentation in ct urography using deep-learning convolutional neural network and level sets. Medical Physics, 2016.

CHOLLET, F. *Deep Learning with Python*. Shelter Island, NY: Manning Publications Co., 2017.

COSTA, L. R.; OBELHEIRO, R. R.; FRAGA, J. S. Introdução á Biometria. In: *Minicursos do VI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg2006)*. Porto Alegre: SBC, 2006. v. 1, p. 103–151.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. Math. Control Signals Systems, v. 2, p. 303–314, 1989.

FACELI, K. et al. *Inteligência Artificial: Uma abordagem de Aprendizado de Máquina*. Rio de Janeiro, RJ: Livros Técnicos e Científicos Editora S.A., 2011.

FAUSETT, L. *Fundamentals of Neural Networks: Architectures, algorithms and applications*. [S.l.]: Pearson, 1993.

FLACH, P. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. The Edinburgh Building, Cambridge, UK: Cambridge University Press, 2012.

GLOB. *glob*. 2019. Disponível em <https://docs.python.org/3/library/glob.html>. Acesso em 3 de maio de 2019.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT press, 2016.

GULLI, A.; PAL, S. *Deep Learning with Keras*. Birmingham, UK: Packt Publishing, 2017.

HAFEMANN, L. G.; SABOURIN, R.; OLIVEIRA, L. S. Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition*, p. 163–176, 2017. Acesso em 10 de junho de 2019. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0031320317302017>.

HAYKIN, S. *Neural Networks and Learning Machines*. Hamilton, Ontario, Canada: Pearson, 2009.

HEATON, J. *Artificial Intelligence for Humans: Deep Learning and Neural Networks*. Chesterfield, MO, USA: CreateSpace Independent Publishing Platform, 2015. v. 3.

HEINEN, M. R. *Autenticação On-line de assinaturas utilizando Redes Neurais*. 92 f. Monografia (Bacharel em Informática) — Centro de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2002.

HEINEN, M. R.; OSóRIO, F. S. Biometria comportamental: Pesquisa e desenvolvimento de um sistema de autenticação de usuários utilizando assinaturas manuscritas. Infocomp Revista de Ciência da Computação, Lavras, MG, Brasil, v. 3, p. 31–37, 2004.

HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*. [S.l.]: COURSERA: Neural Networks for Machine Learning, 2012. Disponível em <https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>. Acesso em 23 de maio de 2019.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, 2017.

IANDOLA, F. N. et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <50mb model size. *CoRR*, 2016.

IMAGENET. 2019. Disponível em: <http://www.image-net.org/>. Acesso em 19 de março de 2019.

IMPEDOVO, D.; PIRLO, G. Automatic signature verification: The state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 38, p. 609–635, 2008.

KAGGLE. *Kaggle Kernels*. 2019. Disponível em <https://www.kaggle.com/docs/kernels>. Acesso em 3 de maio de 2019.

KALCHBRENNER, N.; GREFENSTETTE, E.; BLUNSOM, P. A convolutional neural network for modelling sentences. Association for Computational Linguistics, p. 655–665, 2014.

KERAS. *Keras*. 2019. Disponível em <https://keras.io/>. Acesso em 3 de maio de 2019.

KHAN, S. et al. *A Guide to Convolutional Neural Networks for Computer Vision*. Austrália: Morgan & Claypool, 2018.

KHOLMATOV, A. A. *Biometric Identity Verification Using On-Line & Off-Line Signature Verification*. Dissertação (Mestrado) — Graduate School of Engineering and Natural Sciences, Sabanci University, Istambul, Turquia, 2003.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2014. Disponível em <https://arxiv.org/abs/1412.6980>. Acesso em 23 de maio de 2019.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, Toronto, Ontario, Canada, 2012.

KUBAT, M. *An Introduction to Machine Learning*. Coral Gables, FL, USA: Springer International Publishing, 2015.

LATHI, B. P. *Sinais e Sistemas Lineares*. 2. ed.. ed. [S.l.]: Bookman, 2008.

LEARN scikit. *scikit-learn*. 2019. Disponível em <https://scikit-learn.org/stable/>. Acesso em 3 de maio de 2019.

LECUN, Y. et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998.

LIRA, J. et al. Classificação de atividades humanas com redes neurais artificiais com processamento temporal. IV Escola Regional de Informática, Universidade do Estado do Amazonas, Manaus, BR, 2017.

LIRA, J. do N. Detecção de armas de fogo imersas em contextos utilizando redes neurais convolucionais. Trabalho de Conclusão de Curso da Universidade do Estado do Amazonas, Universidade do Estado do Amazonas, Manaus, BR, 2018.

LIWICKI, M. *IAPR TC11 - ICDAR 2009 Signature Verification Competition (SigComp2009)*. 2012. Disponível em: <http://www.iapr-tc11.org/mediawiki/index.php?title=IAPR-TC11:Reading_Systems>. Acesso em 5 de março de 2019.

LU, L. et al. *Dying ReLU and Initialization: Theory and Numerical Examples*. 2019. Disponível em <https://arxiv.org/abs/1903.06733>. Acesso em 17 de maio de 2019.

MAGALHãES, P. S.; SANTOS, H. D. Biometria e autenticação. In: *4a Conferência da Associação Portuguesa de Sistemas de Informação*. Universidade do Minho, Guimarães, Portugal: Associação Portuguesa de Sistemas de Informação, 2003. ISBN 97 2-9354-42-1.

MARSLAND, S. *Machine Learning: An Algorithmic Perspective*. Boca Raton, FL, US: CRC Press, 2015.

MCCULLOCH, W. S.; PITTS, W. H. A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, v. 5, p. 115–133, 1943.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. Cambridge, Massachussets: The MIT Press, 2012.

NUMFOCUS. *matplotlib*. 2019. Disponível em <https://matplotlib.org/>. Acesso em 3 de maio de 2019.

NUMFOCUS. *numpy*. 2019. Disponível em <https://www.numpy.org/>. Acesso em 3 de maio de 2019.

OS. *os*. 2019. Disponível em <https://docs.python.org/3/library/os.html>. Acesso em 3 de maio de 2019.

PATHAK, A. R.; PANDEY, M.; RAUTARAY, S. Application of deep learning for object detection. Procedia Computer Science, School of Computer Engineering, Kalinga Institute of Industrial Technology (KIIT) University, Bhubaneswar, India, v. 132, p. 1706–1717, 2018.

PEDAMONTI, D. Comparison of non-linear activation functions for deep neural networks on MNIST classification task. *CoRR*, 2018.

PIL. *Pillow*. 2019. Disponível em <https://pillow.readthedocs.io/en/stable/>. Acesso em 3 de maio de 2019.

PINHEIRO, S. A. A. Redes neurais convolucionais aplicadas ao reconhecimento automático de *captchas*. Trabalho de Conclusão de Curso da Universidade do Estado do Amazonas, Universidade do Estado do Amazonas, Manaus, BR, 2018.

RIBEIRO, B. et al. Deep learning networks for off-line handwritten signature recognition. In: *Proceedings of the 16th Iberoamerican Congress on Pattern Recognition (CIARP)*. Berlin, Germany: Springer, 2011. p. 523–532.

ROJAS, R. *Neural Networks: A Systematic Introduction*. Berlin: Springer, 1996.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, v. 6, n. 6, p. 386, 1958.

SEWAK, M.; KARIM, M. R.; PUJARI, P. *Practical Convolutional Neural Networks*. Birmingham, UK: Packt Publishing, 2018.

SIMON, P. *Too Big to ignore*. Hoboken, New Jersey: John Wiley and Sons, Inc., 2013.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for larg-scale image recognition. ICLR 2015, Visual Geometry Group, Department of Engineering Science, University of Oxford, 2015.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Disponível em <https://arxiv.org/abs/1409.1556>. Acesso em 21 de maio de 2019.

SIMãO, P. A. R. *Controlo de assiduidade com multiposto e comunicações wireless*. Dissertação (Mestrado) — Universidade de Aveiro, Portugal, 2008.

SOUZA, C. F. S.; PANTOJA, C. E. P.; SOUZA, F. C. M. Verificação de assinaturas offline utilizando *Dynamic Time Wrapping*. In: *Anais do IX Congresso Brasileiro de Redes Neurais (IX CBRN)*. Ouro Preto, MG, Brasil: Sociedade Brasileira de Redes Neurais, 2009.

SZEGEDY, C. et al. *Going deeper with convolutions*. 2014. Disponível em <http://arxiv.org/abs/1409.4842>. Acesso em 23 de maio de 2019.

TENSORFLOW. *Tensorflow*. 2019. Disponível em <https://www.tensorflow.org/>. Acesso em 3 de maio de 2019.

VOLKER, M.; UMAPADA, P.; APOSTOLOS, A. (Ed.). *Document Analysis And Text Recognition: Benchmarking State-of-the-art Systems*. Estados Unidos: World Scientific, 2018.

WASKOM, M. *seaborn: statistical data visualization*. 2019. Disponível em <https://seaborn.pydata.org/>. Acesso em 11 de junho de 2019.

WIRTZ, B. Technical evaluation of biometric systems. In: CHIN, R.; PONG, T.-C. (Ed.). *Computer Vision — ACCV'98*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. p. 499–506. ISBN 978-3-540-69669-8.

ZHANG, X. et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017. Disponível em: <http://arxiv.org/abs/1707.01083>.