Verificação de Autenticidade de Assinaturas Manuscritas Utilizando Redes Neurais Convolucionais

Marcos Wenneton Vieira de Araújo
Orientadora: Elloá B. Guedes

[1]Laboratório de Sistemas Inteligentes
Grupo de Pesquisas em Sistemas Inteligentes
Escola Superior de Tecnologia
Universidade do Estado do Amazonas
Av. Darcy Vargas, 1200, Manaus, AM

{mwvda.eng, ebgcosta}@uea.edu.br

1. Introdução

A autenticação possui importância fundamental na segurança de sistemas computacionais, pois consiste em permitir ou negar acesso à certas informações ou servicos 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 Biometrica, em particular, tem sido uma das técnicas mais difundidas para autenticação, já sendo utilizada de 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 e modo de andar, dentre outras, as quais são características dinâmicas e que podem variar fortemente ao longo do tempo. Porém, apesar destes desafios, 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 também os aspectos não-invasivos na 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 no 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 consiste em verificar a autenticidade de assinaturas manuscritas com redes neurais convolucionais. Para alcançar esta meta, alguns objetivos específicos precisam 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 a serem realizadas obedecerá 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 sintetiza o cronograma de execução deste trabalho.

1.5. Organização do Documento

Para apresentar a proposta deste trabalho de conclusão de curso, este documento encontra-se dividido nas seções a seguir. A Seção 2 relaciona os fundamentos teóricos necessários para a resolução do problema apresentado. Na Seção 3 é descrita uma análise de trabalhos relacionados. A Seção 4, por sua vez, discorre sobre a solução proposta para a verificação da autenticidade de assinaturas manuscritas, que é seguida pelos resultados parciais obtidos considerando estas soluções, e estão relatados na Seção 5. Por fim, na Seção 6, encontram-se as considerações parciais sobre a realização deste trabalho.

Tabela 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

2. Fundamentação Teórica

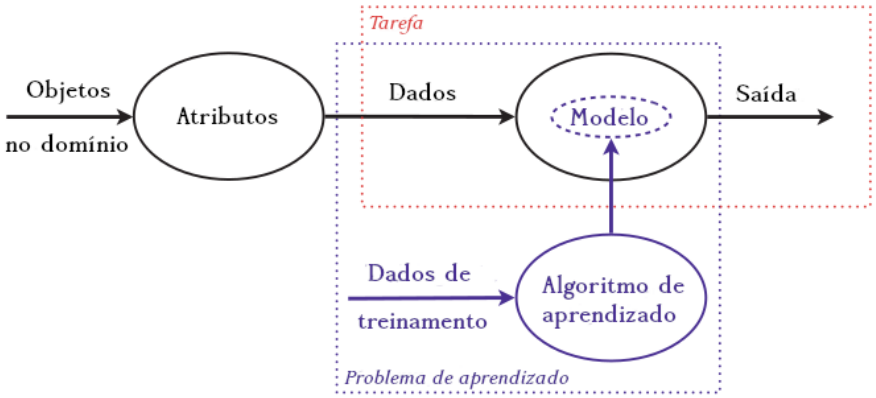
A fundamentação teórica para a elaboração deste trabalho consiste em conceitos relativos à Aprendizado de Máquina. Primeiramente, os conceitos gerais desta área serão apresentados na Seção 2.1, seguidos pelas Redes Neurais Artificais, na Seção 2.2, um dos modelos inferenciais mais representativos. As 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 1.

Figura 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, conside-

rando 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.

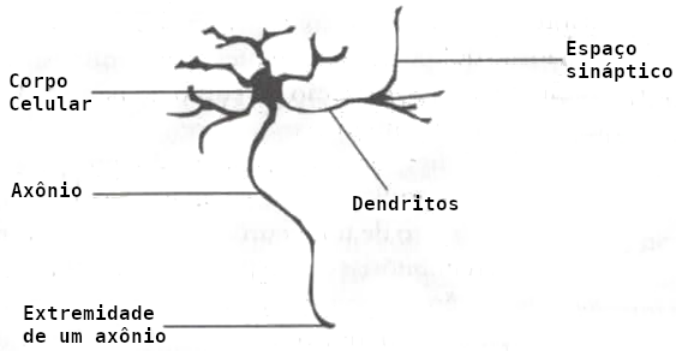
Existe uma grande quantidade de tarefas que podem ser resolvidas com AM, entre estas podemos citar, por exemplo, o reconhecimento de objetos em uma imagem (PATHAK; PANDEY; RAUTARAY, 2018), a determinação da idade de um indivíduo em uma imagem (ARAUJO, 2018), a classificação de atividades humanas (LIRA 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, 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).

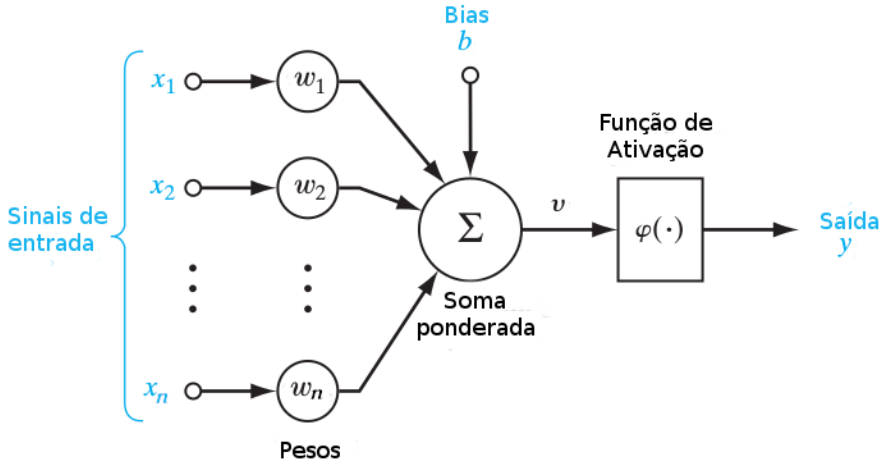
Figura 2: Estrutura de um neurônio biológico. Adaptado de: (BRAGA; CARVALHO; LUDERMIR, 2000)



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 3 (BRAGA; CARVALHO; LUDERMIR, 2000).

Figura 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 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.

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

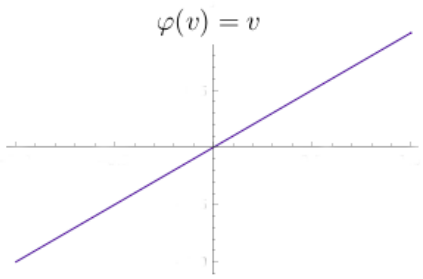
$$v = \sum_{i=1}^{n} x_i w_i + b \tag{2}$$

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 4.

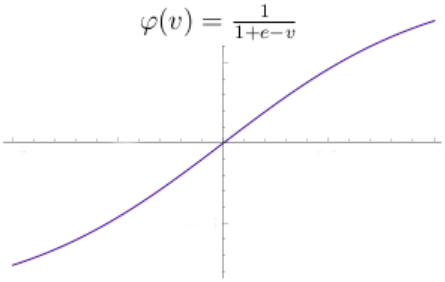
Em 1958, visando a melhoria do neurônio MCP, Frank Rosenblatt desenvolveu o modelo *Perceptron* (ROSENBLATT, 1958). Neste modelo, criou-se o primeiro conceito de aprendizado através de neurônios artificiais, em que foi projetada uma regra de correção de erros para modificar os pesos associados a um neurônio quando suas respostas

Figura 4: Exemplos de funções de ativação.

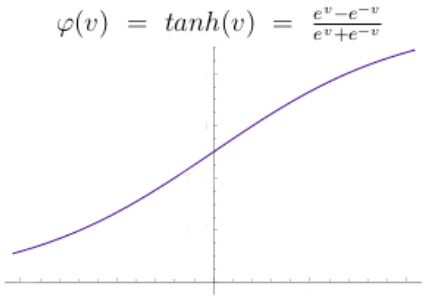
(a) Identidade ou Linear



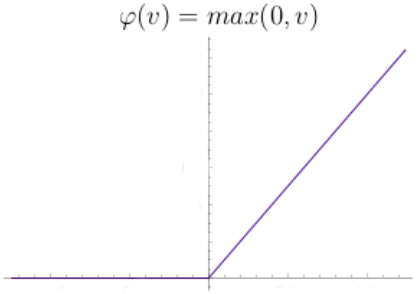
(b) Tangente Hiperbólica



(c) Sigmóide ou Logística



(d) Unidade Linear Retificada (ReLu)



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 3, denominada Regra Delta, cuja obtenção, descrita na Equação 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).

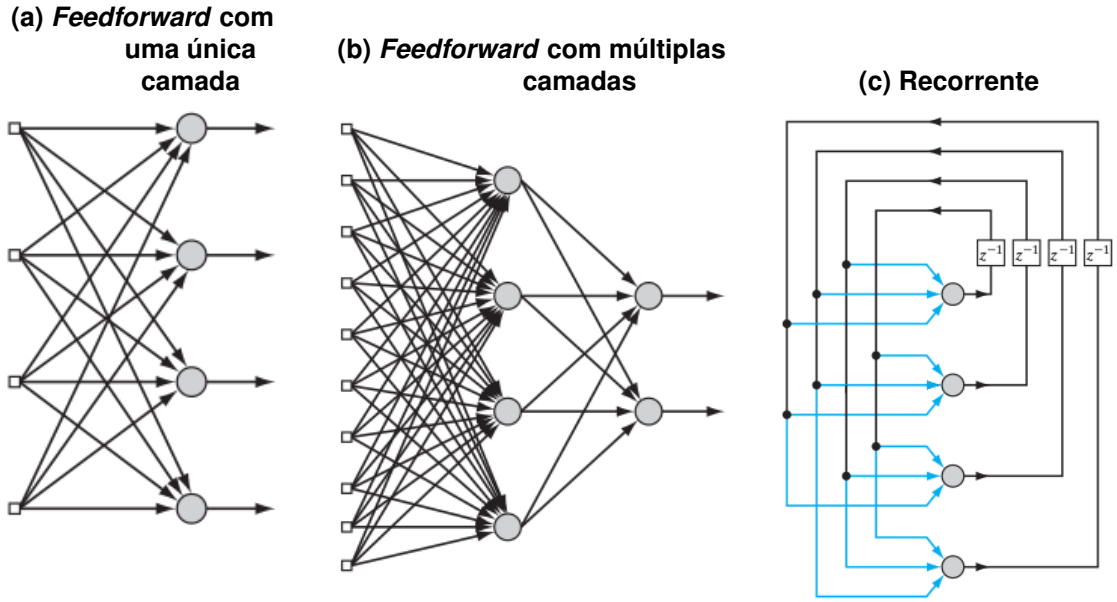
$$w(t+1) = w(t) + \Delta w \quad (3)$$

$$= w(t) + \eta(y - \hat{y})x(t). \quad (4)$$

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 5a e 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

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).

Figura 5: Arquiteturas populares de RNAs. Fonte: (HAYKIN, 2009)



Redes com múltiplas camadas, como na Figura 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).

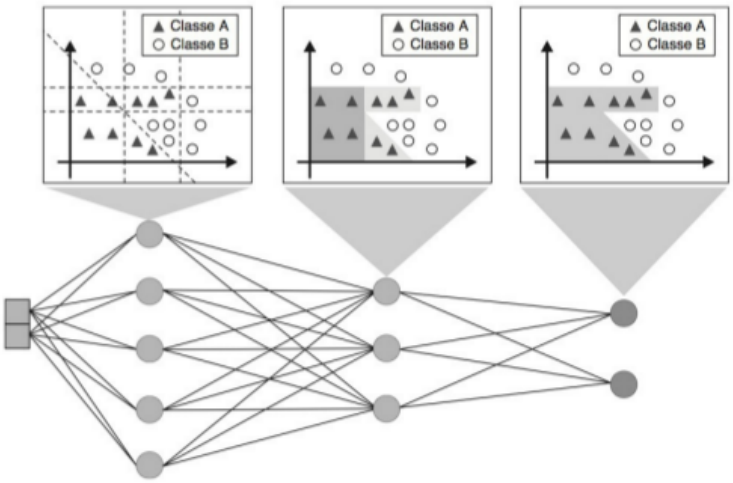
2.2.1. *Multilayer Perceptron*

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 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

Figura 6: Papel exercido pelos neurônios em cada camada de uma rede MLP. Fonte: (FACELI et al., 2011).



função de ativação deve considerar funções contínuas e diferenciáveis (HAYKIN, 2009). Neste algoritmo utiliza-se 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.

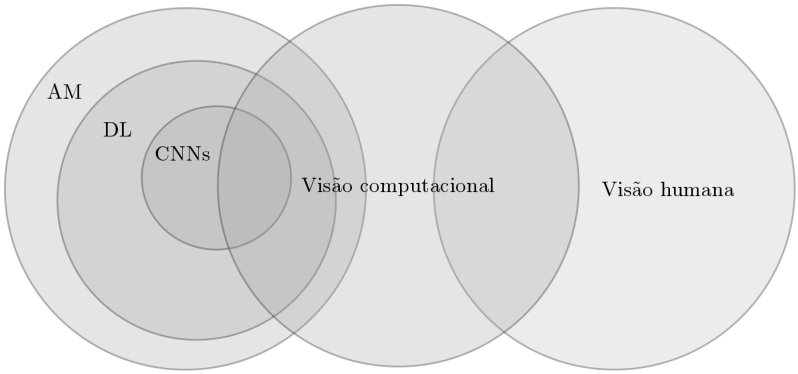
O ferramental de DL compreende um conjunto de técnicas e modelos que podem ser aplicados a tarefas de aprendizado supervisionado e não-supervisionado. Porém, dentre os diferentes modelos existentes, as redes neurais convolucionais se destacam expressivamente, com um bom desempenho em diversos tipos de 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 7 é ilustrada a relação das CNNs com alguns campos de estudos conhecidos.

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

Figura 7: A relação entre a visão humana, visão computacional, AM, DL e CNNs. Adaptado de: (KHAN et al., 2018).



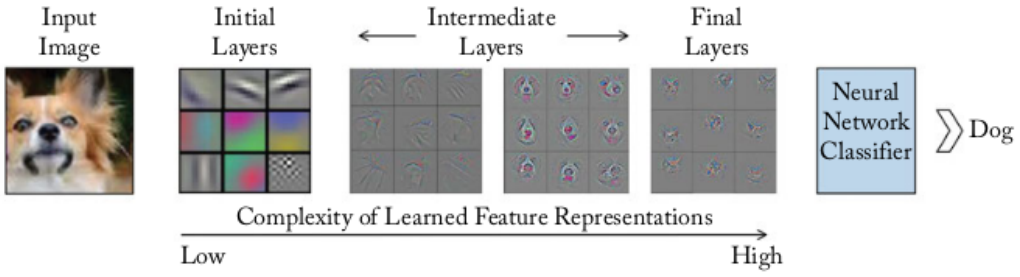
consiste na soma dos produtos de toda a extensão de duas entradas em função de um deslocamento. Sendo assim, a convolução $s(t)$ de duas entradas $x_1(t)$ e $x_2(t)$ é uma função representada simbolicamente por $s(t) = x_1(t) * x_2(t)$ é definida conforme a Equação 5 (LATHI, 2008).

$$s(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(\tau)x_2(t-\tau)d\tau \tag{5}$$

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).

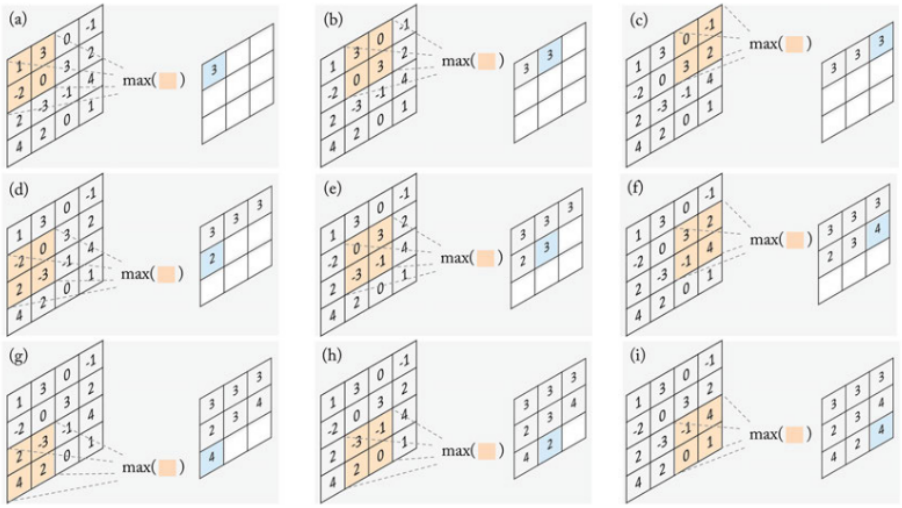
Nas CNNs, as camadas convolucionais são responsáveis por aplicar as operações de convolução. Tomando como exemplo um problema de reconhecimento de padrões em uma imagem, cada camada convolucional é responsável por desenvolver os atributos detectados nas camadas anteriores – de linhas, a contornos, a 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 8.

Figura 8: Exemplo de processo realizado pelas camadas convolucionais de uma CNN, aplicado a um problema de classificação de imagens. Fonte: (KHAN et al., 2018).



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, 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 9.

Figura 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 armazenar 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).

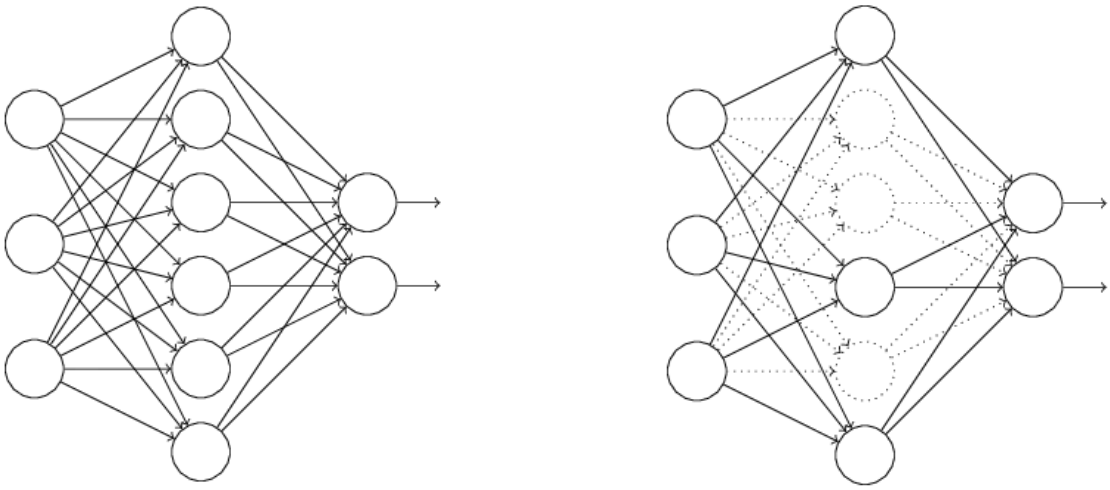
O *dropout* é um tipo de regularização muito efetivo e que é usualmente utilizado em CNNs. Consiste na desativação temporária de alguns neurônios durante a fase de treinamento de uma rede. Nesse processo, os neurônios são desativados mediante uma

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 prevenir o *overfitting* ao possibilitar uma forma de combinar diferentes arquiteturas de redes neurais (BUDUMA, 2017). Esta operação pode ser visualizada na Figura 10, na qual os neurônios desativados são demonstrados pelas circunferências com a borda pontilhada.

Figura 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*.



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.

2.3.2. Arquiteturas Canônicas de Redes Neurais Convolucionais

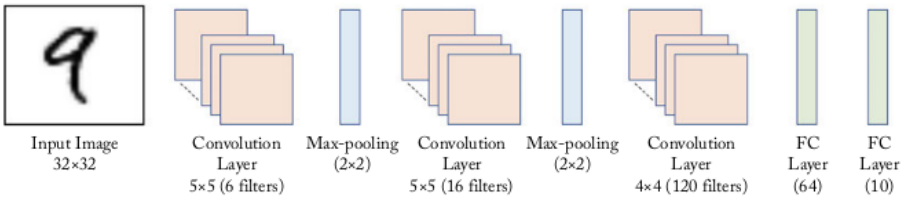
Como mencionado anteriormente, o *ImageNet* é uma base de dados que possui mais de 15 milhões de imagens rotuladas manualmente, de alta resolução, separadas em mais de 22 mil categorias (IMAGENET, 2019). Visando o uso dessa base, tem sido lançando desde 2010 um desafio anual chamado *ImageNet Large Scale 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 11 é possível visualizar a composição da LeNet-5.

Figura 11: A arquitetura LeNet-5. Fonte: (KHAN et al., 2018).



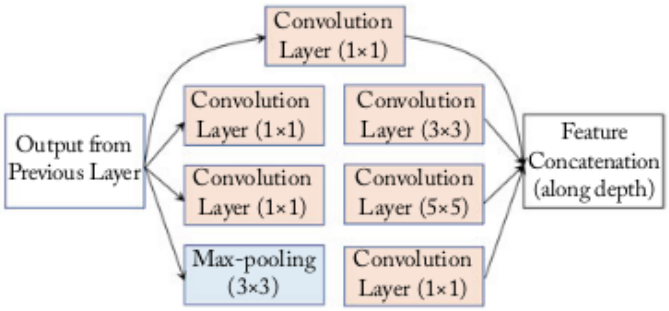
O primeiro modelo em larga escala de CNNs utilizou-se da arquitetura AlexNet, garantindo o primeiro lugar no desafio ILSVRC em 2012 por uma grande margem de diferença dos outros modelos e levando ao ressurgimento da utilização de redes neurais profundas em visão computacional. O uso da função de ativação 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

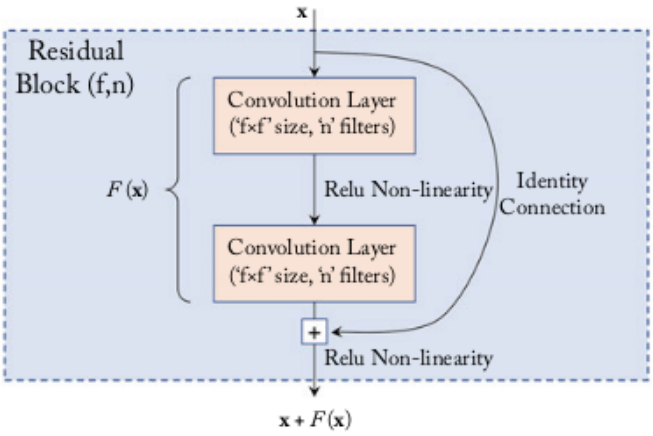
empresa Google e se tornou a vencedora do desafio ILSRVC de 2014. Seu diferencial em relação às outras arquiteturas foi a combinação não sequencial das suas 22 camadas convolucionais. Como mostrado na Figura 12, cada camada é executada de forma paralela com outras camadas formando um módulo chamado Inception, 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 com intuito de diminuir as dimensões do problema antes das dispendiosas convoluções de 3×3 e 5×5 (SEWAK; KARIM; PUJARI, 2018).

Figura 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 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 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 são, em sua maior parte, relacionadas à linguagem de programação *Python*. A escolha dessa linguagem se dá 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 a biblioteca `matplotlib` (NUMFOCUS, 2019a).

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).

3. Trabalhos Relacionados

Começar a escrever qualquer coisa aqui!!!!

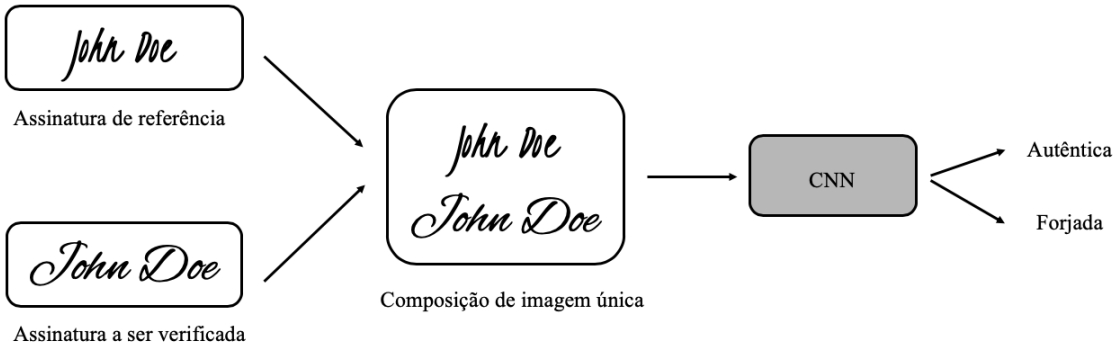
4. Solução Proposta

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 saída de uma CNN para esta tarefa a predição da autenticidade da segunda assinatura que, por ser uma tarefa de

Figura 14: Uma visão geral da tarefa de aprendizado considerada.



classificação binária, poderá assumir somente duas classificações: *autêntica* ou *forjada*. Os passos compreendidos nesta tarefa podem ser visualizados na Figura 14.

Para esta tarefa, diferentes CNNs serão consideradas. O treinamento e teste destes modelos 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* e *F-score*. 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.

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 Docu-*

ment Analysis and Recognition em 2009. 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 15.

Figura 15: Uma amostra das assinaturas *offline* e *online* do 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 2.

Tabela 2: 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 2, 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 14. 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.

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 a etapa anterior, realizou-se então nos exemplos autênticos, a partição *holdout* previamente especificada. Para o exemplos forjados, percebeu-se que a existência de assinaturas realizadas por um mesmo autor forjador nos conjuntos da partição *holdout*, levaria os modelos a prever, na etapa de teste e validação, dados muito parecidos com os encontrados no conjunto de treinamento. Sendo assim, para contornar este problema, a separação dos exemplos forjados foi baseada na quantidade de autores forjadores que cada assinatura autêntica possui e foi obtida segundo as seguintes regras:

- Se uma assinatura autêntica possui apenas um autor forjador, todos os exemplos forjados referentes a esta assinatura vão para o conjunto de treinamento;
- Se uma assinatura autêntica possui quatro autores forjadores, as assinaturas de três desses autores irão para o conjunto de treinamento e as remanescentes, pertencentes a apenas um autor, irão para o conjunto de teste;
- Se uma assinatura autêntica possui cinco autores forjadores, as assinaturas de quatro desses autores irão para a etapa de treinamento e as remanescentes, pertencentes a apenas um autor, estarão presentes na etapa de teste;
- Se uma assinatura autêntica possui seis autores forjadores, as assinaturas de quatro desses autores irão para a etapa de treinamento e as assinaturas dos outros dois irão para a etapa de teste;
- Se uma assinatura possui trinta ou mais autores forjadores, as assinaturas de dez desses autores irão para o conjunto de teste, três desses autores serão utilizados para o conjunto de validação e as assinaturas restantes serão remanejadas para o conjunto de treinamento.

Os autores forjadores selecionados para estarem presentes em cada uma das partições foram selecionados de forma pseudoaleatória e, em todo o *dataset*, não foram encontradas assinaturas que tenham sido forjadas por dois, três ou mais de seis e menos que trinta autores. Os dados de treinamento e validação, apesar da pequena desproporção encontrada entre as suas classes, foram divididos de modo a garantir o bom aprendizado das características dessas classes pelos modelos propostos. O conjunto de testes, por sua vez, apresenta uma característica desbalanceada. A Tabela 3 apresenta a descrição destes dados, suas quantidades e divisões e a Figura 16 auxilia na visualização da proporcionalidade nos diferentes conjuntos considerados.

Ao serem fornecidas para treinamento pelas CNNs, todos os *pixels* das imagens foram normalizados por meio de uma divisão por 255, passando a residirem no intervalo $[0, 1]$. Esta normalização é realizada em virtude das redes neurais que, em geral, aprendem mais eficientemente nestas condições (CHOLLET, 2017).

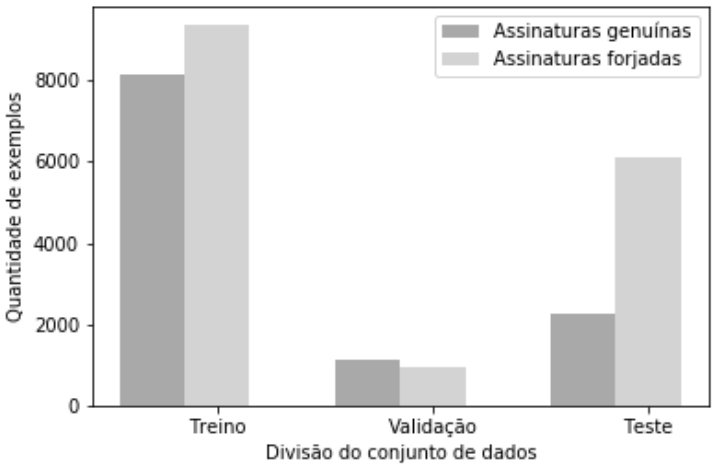
4.4. Modelos, Parâmetros e Hiperparâmetros de CNNs Considerados

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

Tabela 3: Quantitativo de exemplos por finalidade na tarefa de aprendizado considerada e classe.

Conjunto	Tipo de Exemplo	Quantidade de Dados	Proporção
Treinamento	Autêntico	9.374	54%
	Forjado	8.131	46%
Validação	Autêntico	947	46%
	Forjado	1.134	54%
Teste	Autêntico	6.119	73%
	Forjado	2.257	27%

Figura 16: Representação gráfica da proporção dos exemplos por classe e finalidade na tarefa de aprendizado considerada.



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). Possui um total de 13 camadas e aproximadamente 30 milhões de parâmetros treináveis;
- **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);

- **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);
- **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 no `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, considera-se uma modificação na arquitetura geral apenas para compatibilizá-lo com o 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, os quais 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 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.

Tabela 4: 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*, passa-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 melhorar durante o treinamento, decrementa-se uma unidade. O treinamento é finalizado, então, quando este valor torna-se nulo (CHOLLET, 2017). Os valores adotados para *patience*, no escopo deste trabalho, foram obtidos de maneira empírica em testes preliminares.

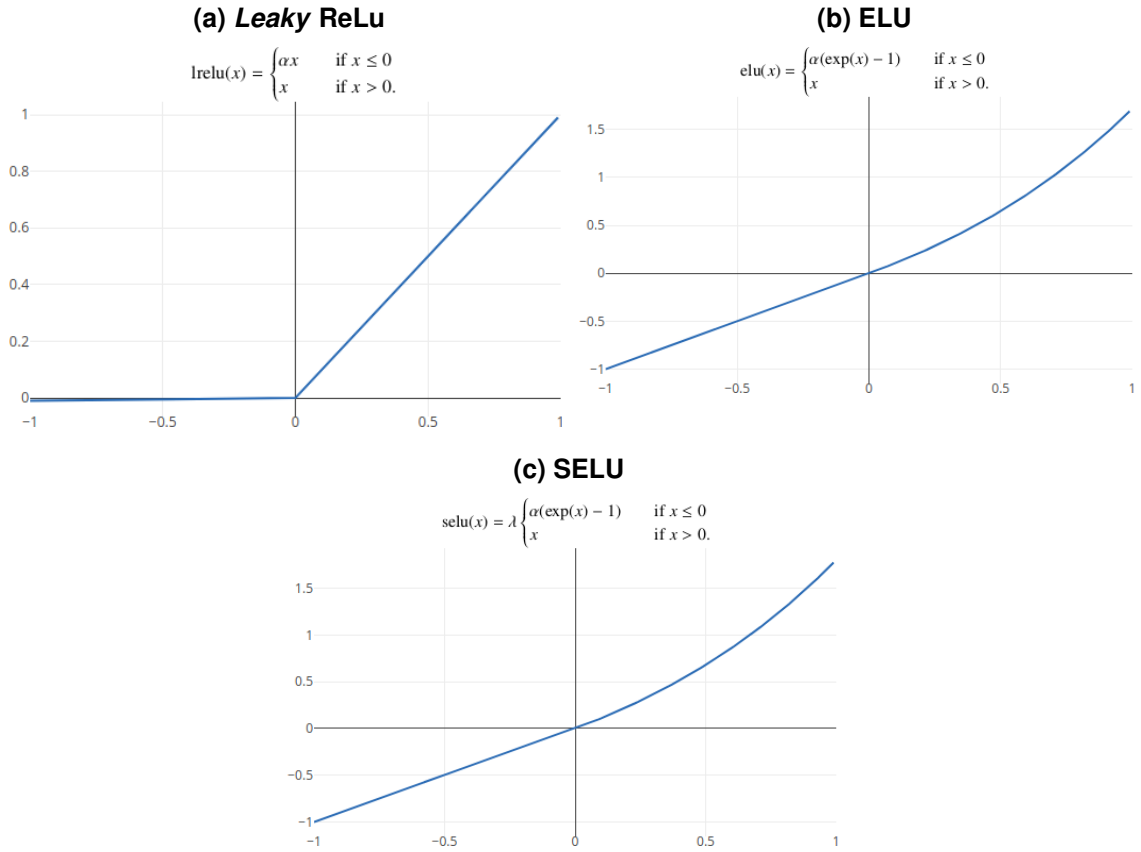
Um otimizador tem como objetivo aumentar o desempenho de um modelo de AM, ajustando os seus parâmetros com vistas a diminuir o erro encontrado na etapa de treinamento de tal modelo. Para o escopo deste trabalho, serão utilizados três otimizadores diferentes, sendo eles, o SGD (*stochastic gradient descent*), 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 função 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 relação não-lineares. Embora a função ReLU seja amplamente adotada, optou-se também por utilizar suas variações, pois esta pode incorrer no 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 variante *Leaky ReLU* possui um parâmetro adicional α, chamado de vazamento, que faz o gradiente seja pequeno, mas nunca nulo. A função ELU 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 17 encontram-se os gráficos das variações da função ReLU.

Todas as três abordagens consideradas, conforme apresentado anteriormente na Seção 4.3, estarão sujeitas ao treinamento de todos os modelos apresentados anteriormente. Nos casos em que o tempo e os recursos computacionais para o treinamento das redes viabilizarem repetições, serão consideradas as variações possíveis dos hiperparâmetros descritas na Tabela 4. Nas demais situações serão consideradas escolhas de hiperparâmetros com melhor desempenho nos cenários que já tiverem sido executados.

Figura 17: Funções de ativação variantes da função ReLU.



5. Resultados Parciais

Nesta seção serão apresentados os resultados parciais obtidos no teste de modelos baseados em algumas das arquiteturas selecionadas para o escopo deste trabalho. As Seções 5.1 e 5.2 demonstram os resultados obtidos pelos melhores modelos encontrados com as arquiteturas LeNet e AlexNet, respectivamente. 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, ajudando no processamento dos algoritmos de aprendizado.

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 a 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 o aprendizado do 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 de-

finidos, conforme Seção 4.4, gerando um total de 36 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. Em relação aos três otimizadores considerados, os modelos dispostos na Tabela 5 foram identificados como tendo melhor desempenho para a tarefa em questão.

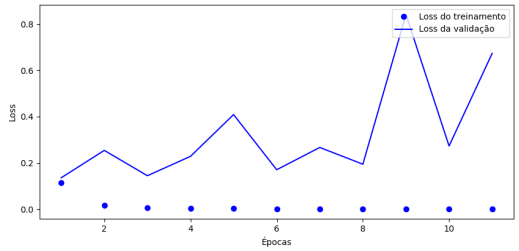
Tabela 5: Detalhamento dos melhores modelos obtidos com a arquitetura LeNet, organizados de forma decrescente considerando o valor de acurácia.

Otimizador	***Patience***	**Função de Ativação**	**Acurácia**	**F-Score**
RMSprop	5	*Leaky* ReLU	0.9865	0.9755
SGD	5	ELU	0.9787	0.9619
Adam	10	ReLU	0.9366	0.8974

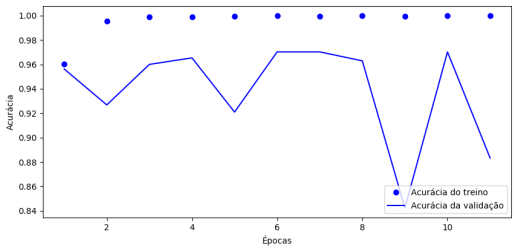
Os gráficos da Figura 18 denotam o histórico da perda (*loss*) e acurácia para o conjunto de treinamento e validação destas redes. Nota-se que nenhuma delas chegou ao limite máximo de épocas possíveis, interrompendo o aprendizado por meio de *early stopping*.

Figura 18: Histórico de *loss* e acurácia durante o treinamento dos melhores modelos obtidos com a arquitetura LeNet.

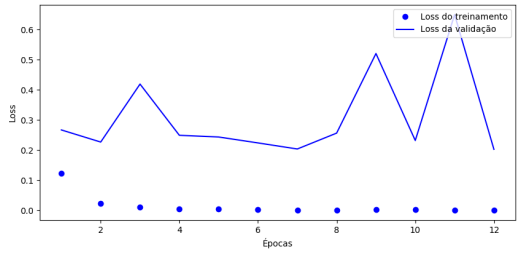
(a) *Loss* durante treinamento do melhor modelo com RMSprop.



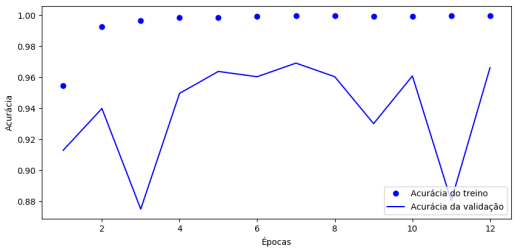
(b) Acurácia durante treinamento do melhor modelo com RMSprop.



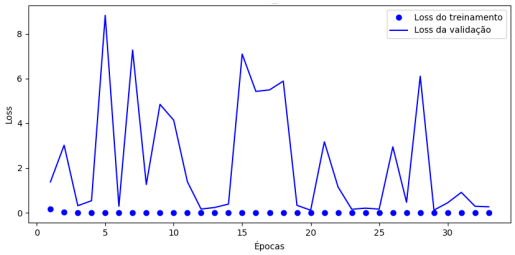
(c) *Loss* durante treinamento do melhor modelo com SGD.



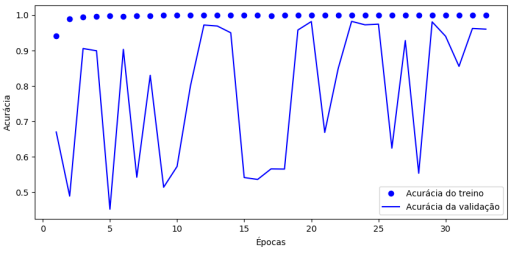
(d) Acurácia durante treinamento do melhor modelo com SGD.



(e) *Loss* durante treinamento do melhor modelo com Adam.



(f) Acurácia durante treinamento do melhor modelo com Adam.



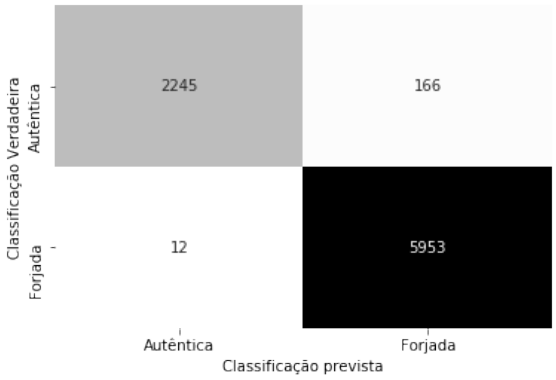
Examinando mais atentamente o desempenho destas redes no conjunto de testes, tem-se, então, as matrizes de confusão mostradas na Figura 19. Nestas matrizes, a soma das linhas representam a quantidade de assinaturas previstas para cada classe pelo modelo em questão, enquanto a soma das colunas denotam a quantidade de assinaturas existentes em cada classe.

Inverter xlabel e ylabel das matrizes

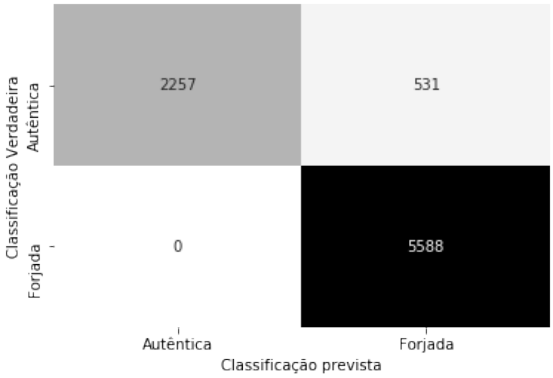
Figura 19: Matrizes de confusão dos melhores modelos obtidos com a arquitetura LeNet.

(a) Modelo com RMSprop

(b) Modelo com SGD



(c) Modelo com Adam



Para esta arquitetura, é possível visualizar que os melhores modelos foram aqueles que possuíram o menor valor de *patience*. Isso revela que houve uma oscilação no treinamento, de modo que o aprendizado de característica sobre o problema foi instável, resultando em uma parada precoce. Para contornar este efeito, pode ser necessário buscar ajustes de parâmetros (*fine-tuning*) ou também avaliar outras arquiteturas nesta tarefa, nas quais este efeito é minimizado.

Percebeu-se que a maioria dos modelos tendeu a acertar mais os exemplos autênticos encontrados no conjunto de teste, no qual o melhor modelo com o otimizador Adam chegou a obter 100% de aproveitamento para esta classe, conforme mostrado na matriz da Figura 19c. Isso pode ser explicado pela presença de assinaturas genuínas do mesmo indivíduo no conjunto de treinamento, porém, esta característica encontrada na partição dos dados, está de acordo com a tarefa definida para o escopo deste trabalho. Apesar disso, quando se observa também a inferência dessas CNNs para exemplos forjados, que são parte majoritária do conjunto de teste, o desempenho obtido com as redes LeNet foi significativamente bom 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* com os hiperparâmetros selecionados anteriormente, com vistas a obter os melhores modelos

para cada abordagem de separação de dados, gerando assim, mais 36 modelos a serem avaliados quanto as suas métricas de desempenho.

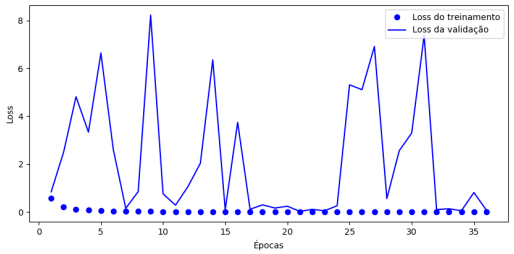
Considerando a métrica de *F-score*, foram selecionados os melhores modelos e estes encontram-se listados na Tabela 6. Enquanto na Figura 20 pode-se observar os gráficos com os comportamentos dos valores de *loss* e acurácia encontrado nos conjuntos de treinamento e validação durante o estágio de treino destes modelos.

Tabela 6: Detalhamento dos melhores modelos obtidos com a arquitetura AlexNet, organizados de forma decrescente considerando o valor de Acurácia.

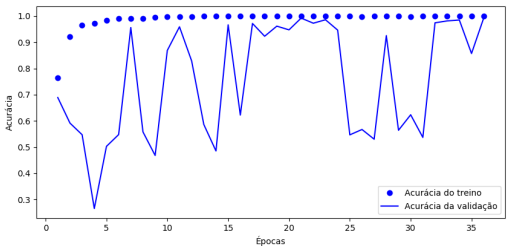
Otimizador	*Patience*	Função de Ativação	Acurácia	F-Score
Adam	15	ELU	0.9654	0.9393
SGD	10	*Leaky* ReLU	0.9601	0.9311
RMSprop	15	*Leaky* ReLU	0.9397	0.8975

Figura 20: Histórico de *loss* e acurácia durante o treinamento dos melhores modelos obtidos com a arquitetura AlexNet.

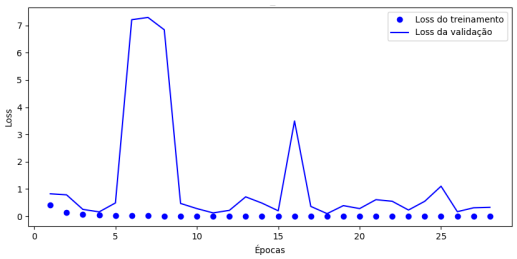
(a) *Loss* durante treinamento do melhor modelo com Adam.



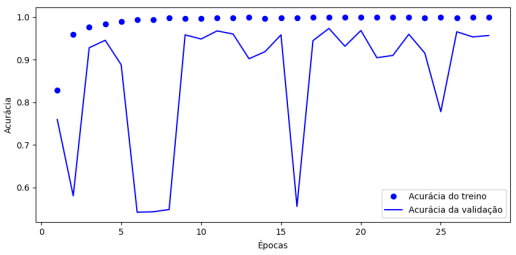
(b) Acurácia durante treinamento do melhor modelo com Adam.



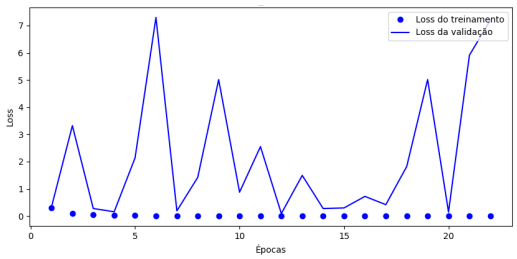
(c) *Loss* durante treinamento do melhor modelo com SGD.



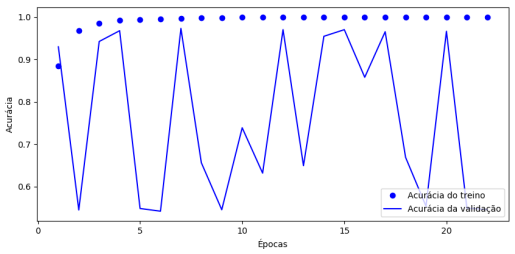
(d) Acurácia durante treinamento do melhor modelo com SGD.



(e) *Loss* durante treinamento do melhor modelo com RMSprop.



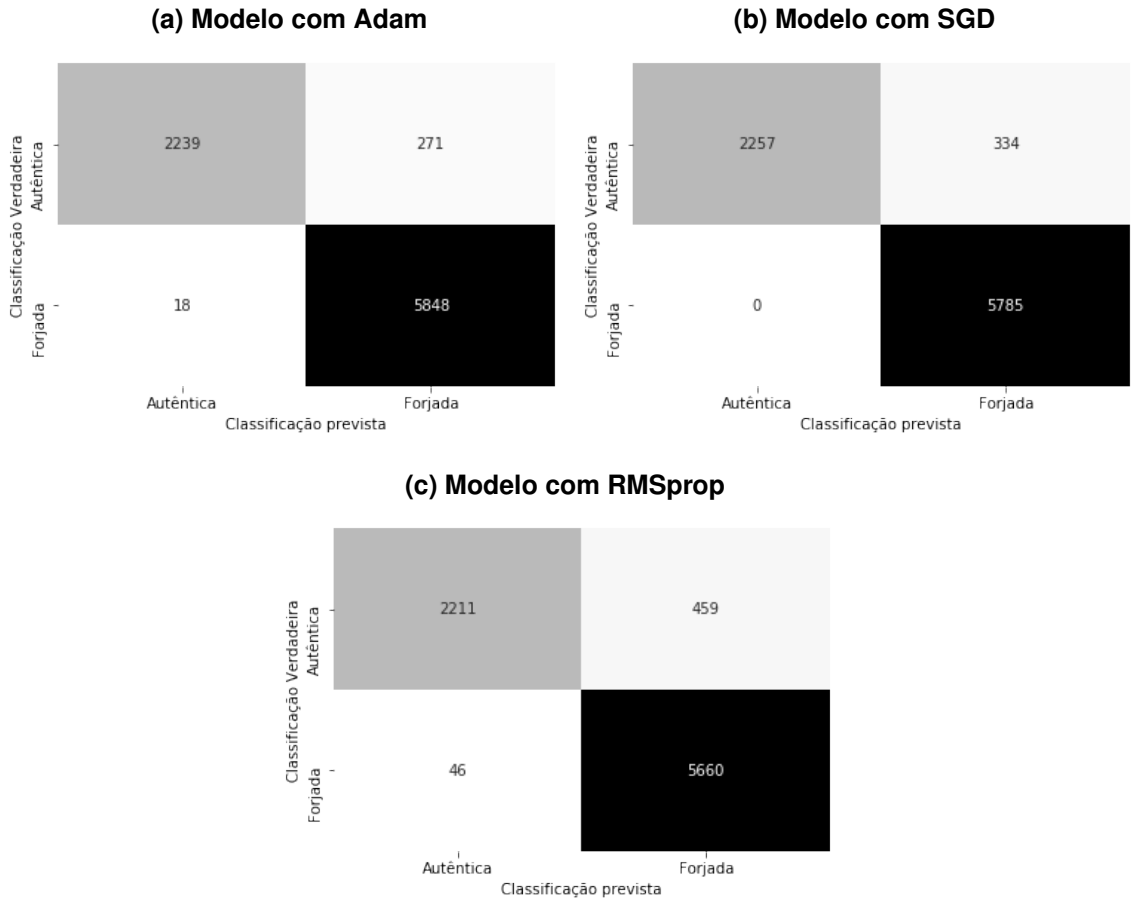
(f) Acurácia durante treinamento do melhor modelo com RMSprop.



Mais uma vez, nota-se que houve uma oscilação nos treinamentos, de modo que o aprendizado de características sobre o problema foi instável, resultando em uma parada precoce, mesmo quando o valor de patience adotado é grande. Porém, desta vez, a tendência de previsão de falsos positivos aumentou quando comparada aos modelos obtidos pela LeNet, embora a quantidade de falsos negativos continue relativamente parecida. Estas previsões podem ser analisadas mais profundamente na Figura 21, que demonstra as matrizes de confusão obtida em cada uma das CNNs apresentadas na Tabela 6.

Inverter xlabel e ylabel das matrizes

Figura 21: Matrizes de confusão dos melhores modelos obtidos com a arquitetura AlexNet.



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 impugnar o melhor modelo obtido com a arquitetura LeNet (0.9755). Como demonstrado anteriormente, a quantidade camadas e parâmetros encontrados na AlexNet supera a quantidade encontrada na LeNet, desta forma os modelos obtidos por essa segunda arquitetura possuem uma eficiência melhor, tanto na questão de previsão de entradas quanto para o tempo de treinamento e tamanho de memória necessário para o seu armazenamento. Sendo assim, os resultados obtidos com a AlexNet foram insuficientes para a tarefa observada.

6. Considerações Parciais

Agradecimentos

Os autores agradecem os recursos financeiros e materiais providos pela Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) por meio do PPP 04/2017.

Referências

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.

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.

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.

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.

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.

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.

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.