# 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

### 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

### 1.3. Metodologia

### 1.4. Cronograma

## 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.

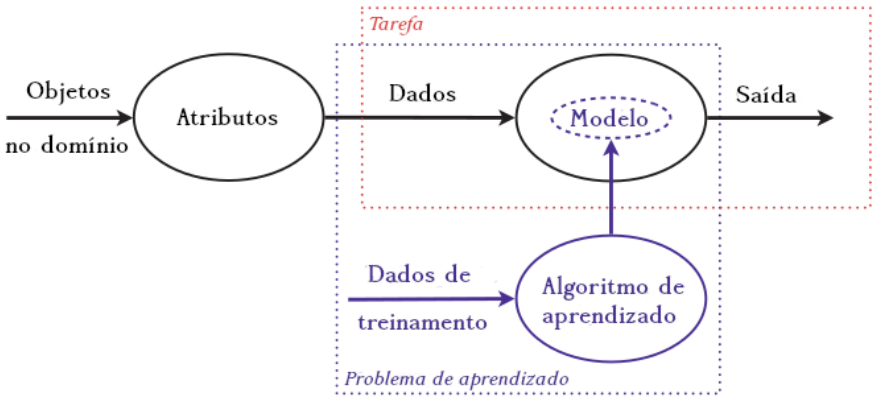
### 2.1. Aprendizado de Máquina

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, 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 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.

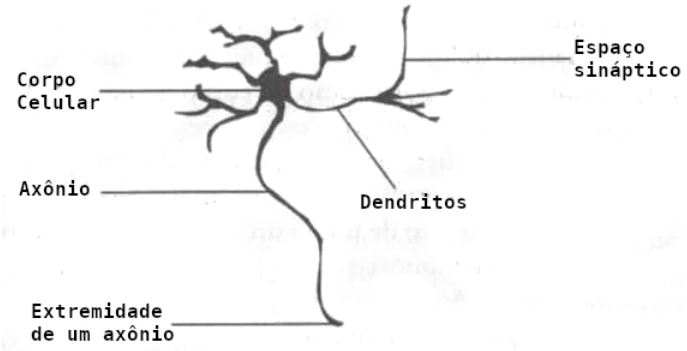
### 2.2. Redes Neurais Artificiais

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).

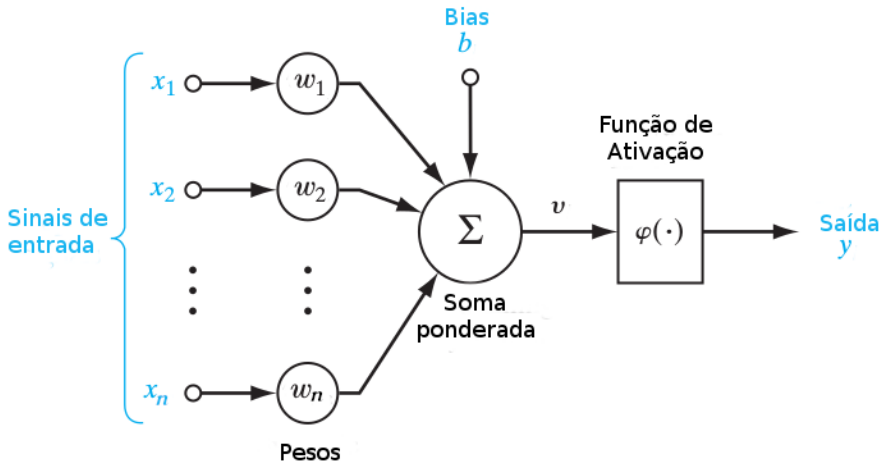
Considerando uma analogia com os neurônios biológicos, modelou-se então a primeira noção de neurônios artificiais. Nestes neurônios, as entradas são valores $x =$

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




$x_1, ..., x_n$ 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* $\theta$ previamente definido e o valor da soma ponderada da entrada, como mostrado na Equação 2.

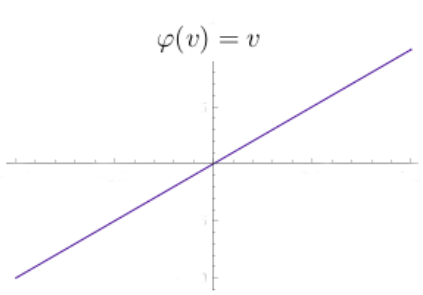
$$\varphi(v) = \begin{cases} 1, & \text{se } v > \theta. \\ 0, & \text{caso contrário.} \end{cases} \tag{1}$$

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

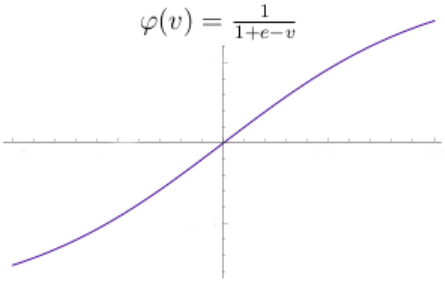
Embora o modelo MCP tenha considerado apenas funções de ativação do tipo degrau deslocada, outras definições também são possíveis. As funções identidade, sigmóide, tangente hiperbólica e retificada linear (ReLU) são comumente utilizadas, definidas tais como mostrado na Figura 4.

**Figura 4: Exemplos de funções de ativação.**

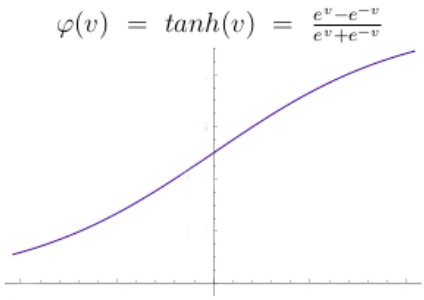
**(a) Identidade ou Linear**



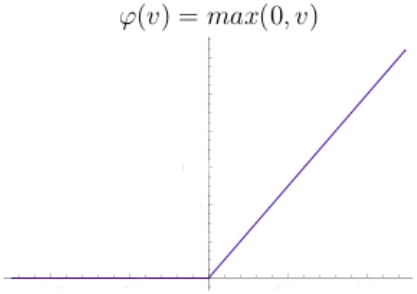

**(b) Tangente Hiperbólica**




**(c) Sigmóide ou Logística**




**(d) Unidade Linear Retificada (ReLu)**

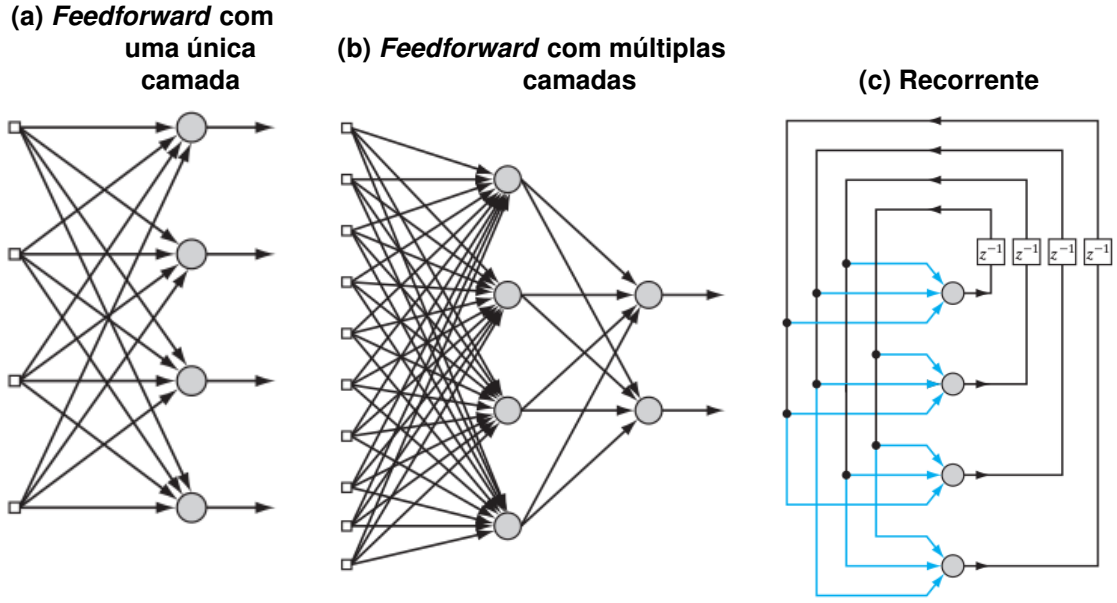



Em 1958, visando a melhoria do neurônio MCP, Frank Rosenblatt desenvolveu o modelo *Perceptron* (ROSENBLATT, 1958). Neste modelo, criou-se o primeiro conceito de aprendizado através de neurônios artificiais, em que foi projetada uma regra de correção de erros para modificar os pesos associados a um neurônio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



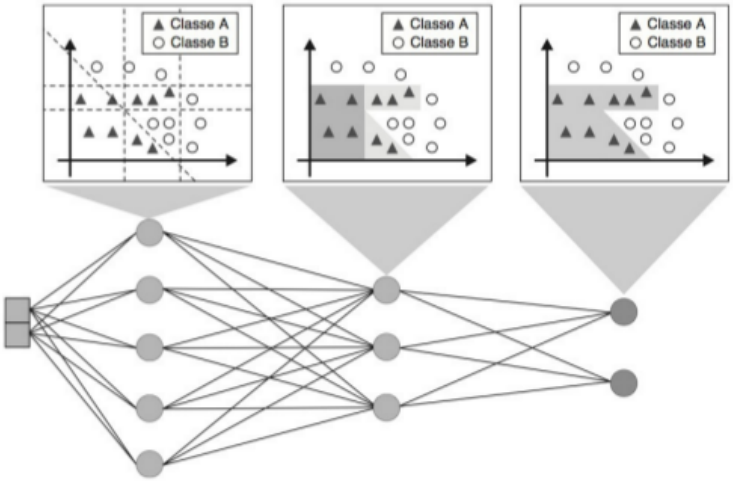

Redes com múltiplas camadas, como na Figura 5b, caracterizam-se pela presença de pelo menos uma camada oculta. Isso acarreta um grande poder às redes deste tipo pois, conforme Cybenko, uma rede com uma camada oculta é capaz de mapear qualquer função contínua, enquanto uma rede com duas camadas ocultas 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.

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




O algoritmo de aprendizado supervisionado mais conhecido e utilizado para treinamento das MLPs é o *backpropagation*. 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).

inserir informação sobre a necessidade de uma função de ativação contínua e diferenciável para o uso de backpropagation

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*, quere-

mos 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 ML que enfatiza o aprendizado através de sucessivas camadas de representações cada vez mais significativas dos dados submetidos. Estas representações são quase sempre obtidas atráves de redes neurais profundas (CHOLLET, 2017). De acordo com Heaton, qualquer RNA com mais de duas camadas ocultas é, em sua essência, considerada profunda (HEATON, 2015). O DL tem obtido um êxito incrível em endereçar problemas de visão computacional e processamento de linguagem natural. Estes algoritmos não só ultrapassaram outras variedades de algoritmos de ML, 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).

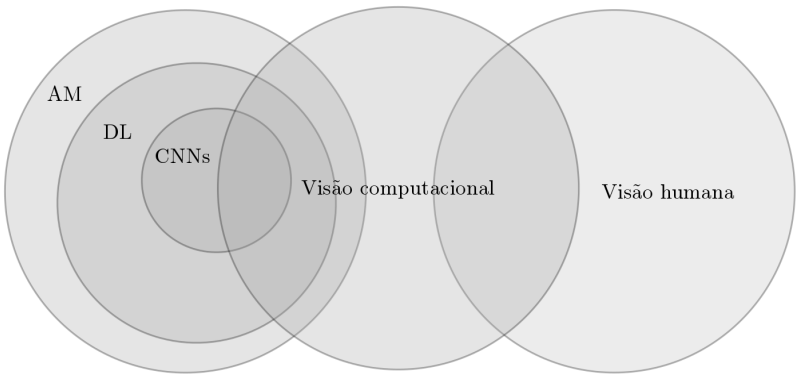
O DL reúne um conjunto de técnicas e modelos que podem ser aplicados a tarefas de aprendizado supervisionado e não-supervisionado, nas quais as redes neurais convolucionais se destacam expressivamente. A próxima seção descreve os pontos principais relacionados a este tipo de RNA.

### 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 extrema-

mente 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.

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




A *convolução* 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 $s(t) = x_1(t) * x_2(t)$ e definidade conforme a Equação 5 (LATHI, 2008).

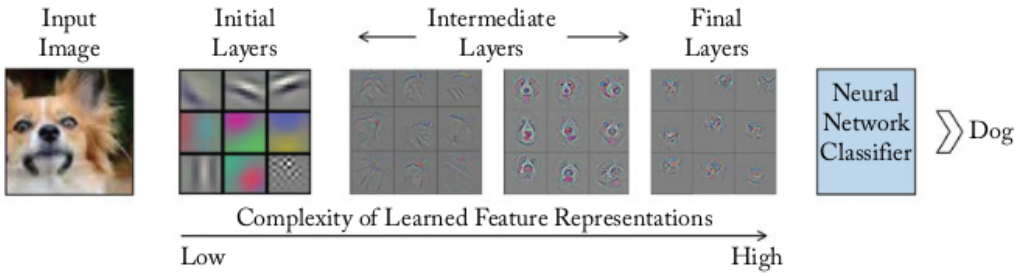
$$s(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(\tau)x_2(t-\tau)d\tau \tag{5}$$

Nas aplicações de ML, 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 ML é comumente referenciada como *tensor* (GOODFELLOW; BENGIO; COURVILLE, 2016).

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 é responsável por desenvolver os atributos detectados nas camadas anteriores - de linhas, a contornos, a formatos, até construir um objeto por completo. Esse processo pode ser visualizado na Figura 8. Nestas camadas os mapas de características são capturados, e nestes consistem os pesos da rede que são responsáveis por identificar onde são encontrados os atributos na imagem original (BUDUMA, 2017).

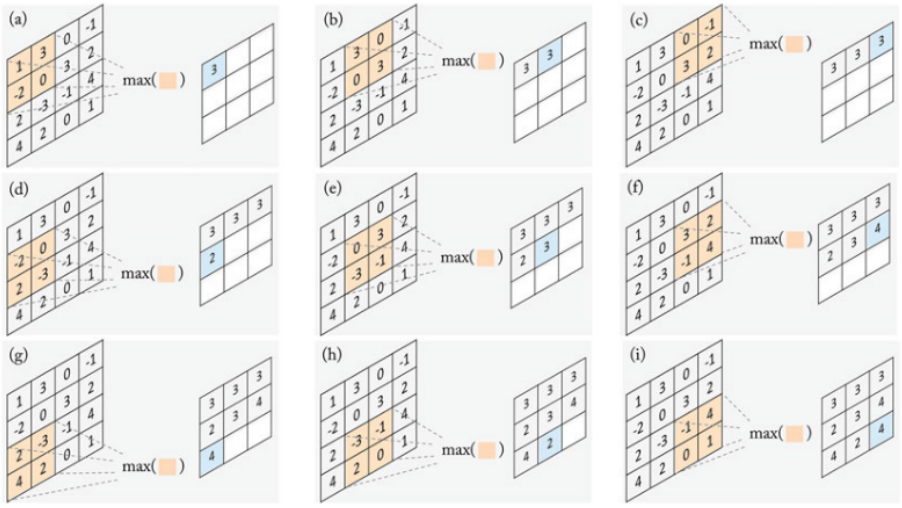
Uma camada de *pooling* em uma CNN opera em blocos do mapa de características e combina seus atributos através da operação de *max pooling* ou *average pooling*. Esse bloco é deslizado através do mapa de características com um passo definido como *stride*. A operação de *max pooling* retorna o valor 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

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




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).

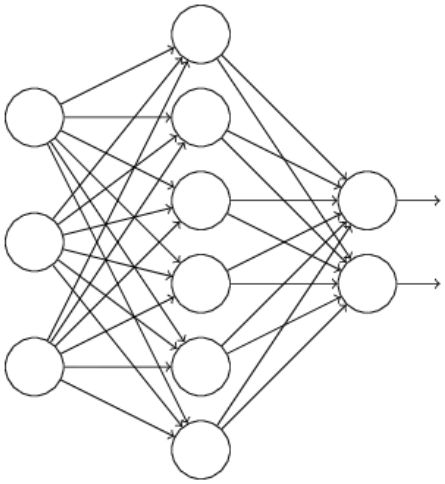
Para prevenir o aprendizado de padrões irrelevantes por um modelo de ML, pode-se articular a quantidade de informações que esse modelo pode armazenar ou adicionar restrições sobre as informações que podem ser armazenadas. Se uma CNN tende a memorizar um pequeno número de padrões, o processo de otimização 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).

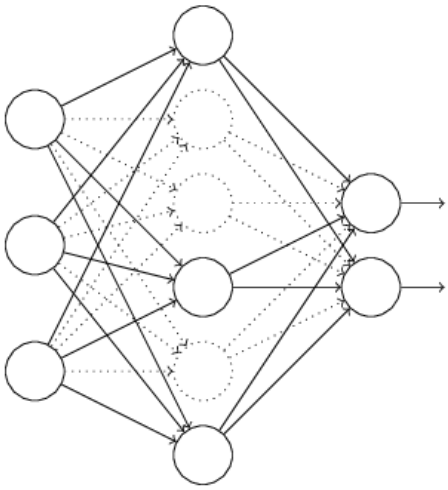
O *dropout* é um tipo de regularização muito efetivo e que é usualmente utilizado em CNNs. Consiste na desativação temporária de alguns neurônios durante a fase de treinamento de uma rede. Nesse processo, os neurônios são desativados mediante uma probabilidade $p$, conhecida como *dropout rate*, retornando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



### 2.3.2. Arquiteturas Canônicas de Redes Neurais Convolucionais

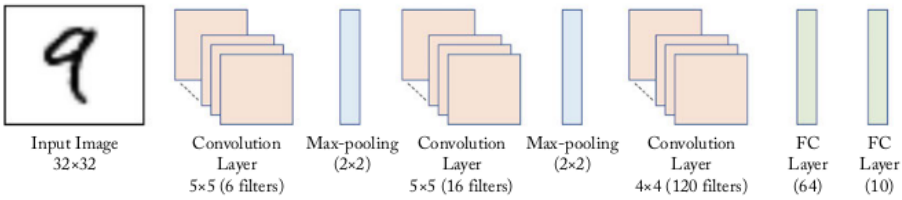
Como mencionado anteriormente, o *ImageNet* é uma base de dados que possui mais de 15 milhões de imagens rotuladas manualmente, de alta resolução, separadas em mais de 22 mil categorias (IMAGENET, 2019). Visando o uso dessa base, tem sido lançando desde 2010 um desafio anual chamado *ImageNet Large Scale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



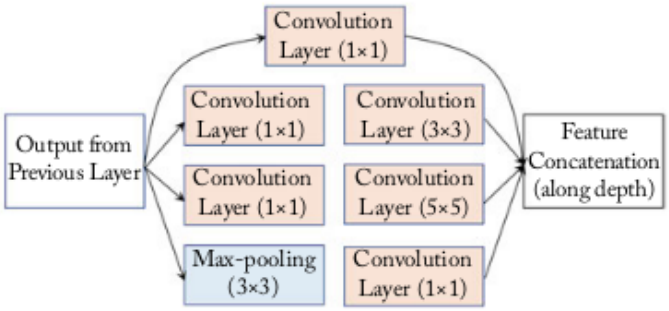

O primeiro modelo em larga escala de CNNs utilizou-se da arquitetura AlexNet, garantindo o primeiro lugar no desafio ILSVRC em 2012 por uma grande margem de diferença dos outros modelos e levando ao ressurgimento da utilização de redes neurais profundas em visão computacional. O uso da função de ativação 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 $3 \times 3$ e as operações de *max pooling* são realizadas através de uma janela de $2 \times 2$ pixeis com um *stride* igual a $2$ (SIMONYAN; ZISSERMAN, 2015).

A arquitetura GoogLeNet, também chamada de Inception, foi desenvolvida pela

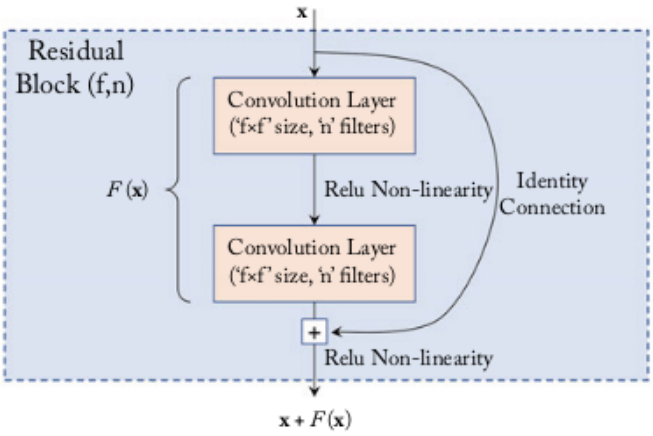
empresa Google e se tornou a vencedora do desafio ILSRVC de 2014. Seu diferencial em relação às outras arquiteturas foi a combinação não sequencial das suas 22 camadas convolucionais. Como mostrado na Figura 12, cada camada é executada de forma paralela com outras camadas formando um módulo chamado Inception, 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 $1 \times 1$ com a função de ativação ReLu com intuito de diminuir as dimensões do problema antes das dispendiosas convoluções de $3 \times 3$ e $5 \times 5$ (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).**

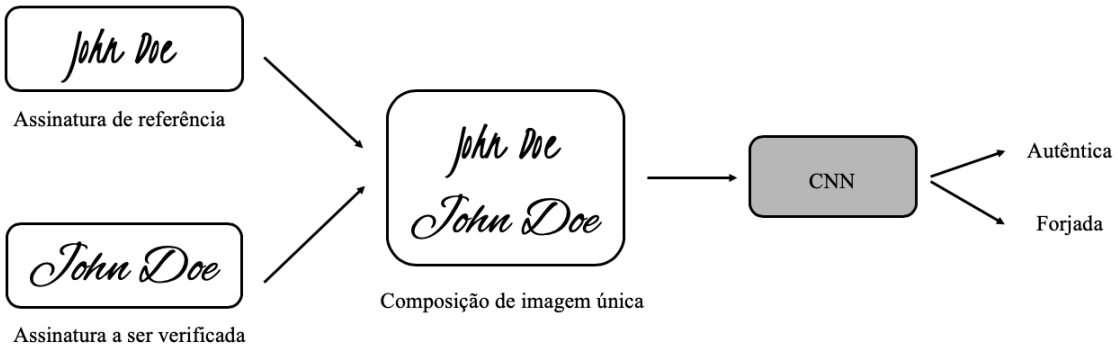


# 3. Trabalhos Relacionados

# 4. Solução Proposta

## 4.1. Tarefa de Aprendizado

**Figura 14: Uma visão geral do processo de aprendizado.**




O treinamento e testes das CNNs seguirão o comportamento *holdout* de validação cruzada, em que $70\%$ dos dados serão utilizados no treino e ajuste de parâmetros e o restante para avaliação, com vista a capturar a qualidade da generalização proposta pelos modelos considerados.

## 4.2. Visão Geral do Conjunto de Dados

Para a criação de um tarefa de aprendizado que seja capaz de verificar a autenticidade de assinaturas de indivíduos, é necessário primeiramente a existência de um conjunto de dados de imagens que possuam exemplos de assinaturas forjadas e genuínas, com o intuito de ser utilizado para o treinamento das CNNs. Para isto, utilizou-se os conjuntos de dados disponibilizados pela Competição de Verificação de Assinaturas de 2009 (SigComp2009, do inglês *Signature Verification Competition*) realizado na Conferência Internacional em Análise e Reconhecimento de Documentos (ICDAR, do inglês *International Conference on Document Analysis and Recognition*).

A SigComp2009 utilizou-se de conjuntos de dados diferentes em cada uma das duas etapa da competição. Para a 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, enquanto para a etapa de validação dos modelos submetidos na competição foi aplicado o conjunto coletado pelo *Netherlands Forensic Institute* (NFI), composto por 1.953 novas assinaturas genuínas e forjadas (BLANKERS et al., 2009).

O melhor tipo de falsificação de assinatura é a chamada *over-the-shoulder*, na qual o autor forjador tem a oportunidade de visualizar a assinatura genuína antes da falsificação. As assinaturas forjadas encontradas nos conjuntos NISDCC e NFI pertencem a esse tipo de falsificação, ou seja, os seus autores tiveram a oportunidade de praticá-las anteriormente a fim de aumentar a sua semelhança com a assinatura original. De acordo com Blankers, falsificações desse tipo normalmente são muito difíceis de detectar (BLANKERS et al., 2009).

Os conjuntos disponibilizados pela SigComp2009 aos participantes eram 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 as resoluções de 300 e 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 de azimute e o ângulo de elevação (BLANKERS et al., 2009). Um exemplo de uma assinatura *offline* e a sua representação *online* com os pontos plotados pode ser encontrada na Figura 15.

**Figura 15: Uma amostra das assinaturas *offline* e *online*. Fonte: (BLANKERS et al., 2009)**

Desses dados disponibilizados, apenas as assinaturas *offline* de cada conjunto foram utilizadas para a construção dos modelos propostos para a resolução do problema descrito neste trabalho, devido a preferência por desenvolver uma solução em visão computacional. Apesar da quantidade de assinaturas *offline* originalmente utilizadas na competição, foram disponibilizadas atráves da *International Association for Pattern Recognition* (IAPR) apenas 1.898 assinaturas do conjunto NISDCC e 1.564 assinaturas do conjunto NFI (LIWICKI, 2012). Uma demonstração dos dados disponibilizados pela IAPR em relação a quantidade de autores e assinaturas pode ser visualizado na Tabela 1.

**Tabela 1: Quantitativo de indivíduos e assinaturas 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 |

As imagens correspondentes às assinaturas *offline* presentes nos conjuntos estão em formato *png*, com resolução de 600 dpi e com diferentes dimensões, de forma que todo o conteúdo da assinatura em questão seja visualizado. Contudo, a fim de padronizar esta base de dados para a criação de exemplos compatíveis com o modelo que deseja-se desenvolver, foi necessário um pré-processamento dessas imagens, conforme descrito na seção a seguir.

### 4.3. Preparação do Conjunto de Dados

Sabe-se que algoritmos de AM necessitam de quantidade significante de dados, preferencialmente sem muitos ruídos, para serem executados de forma a obter um bom desempenho (MARSLAND, 2015). Portanto, visando a consolidação de uma base de dados própria que se adeque a solução proposta e que diminua a sobrecarga de processamento necessária para o treinamento do modelo, fez se necessário um pré-processamento das imagens contidas no conjunto de dados disponibilizado pelo IAPR.

Na fase de pré-processamento da base de dados, foi necessária a adaptação das imagens de forma a seguir o processo de aprendizado ilustrado na Figura 14. Para isto, foi feita a combinação de cada assinatura genuína de um autor com ela mesma e suas outras assinaturas genuínas, a fim de criar exemplos genuínos, e também a combinação dessa assinatura com cada assinatura forjada do mesmo autor, criando assim os exemplos rotulados como forjados. Todas as imagens obtidas dessas combinações foram utilizadas como exemplos para o processo de treinamento, validação e teste do modelo proposto.

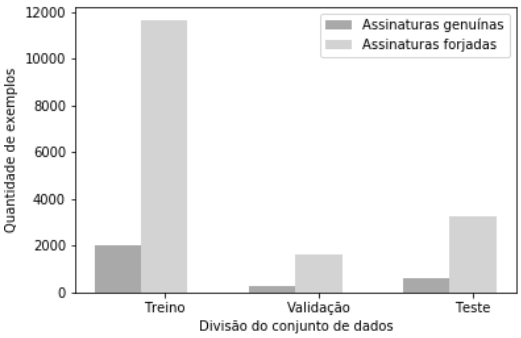
O processo de combinação das imagens de cada um dos exemplos foi concluído em três etapas. Na primeira etapa, ambas as imagens foram redimensionadas para um tamanho de $256 \times 256$ *pixels*. Em seguida, as imagens foram concatenadas verticalmente com a intenção de formar uma única imagem de $256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

Devido a desproporção no número de exemplos genuínos e forjados e considerando que para alguns indivíduos o conjunto de dados original não possuía assinaturas forjadas, apenas genuínas, foi necessário consolidar bases de dados diferentes considerando três abordagens distintas, visando um melhor aproveitamento dos exemplos disponíveis e a consideração de diferentes percentuais de exemplos, como pode ser visualizado na Figura 16. Em todas as abordagens a proporção de dados separados para as etapas treinamento, teste e validação foi a mesma, aproximadamente $70\%$ para treinamento, aproximadamente $20\%$ para teste e aproximadamente $10\%$ para validação.

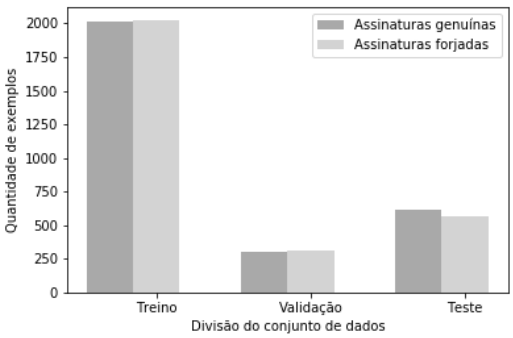
Na primeira abordagem, foram considerados apenas os exemplos nos quais os indivíduos possuíam assinaturas genuínas e forjadas, sem se importar com a quantidade de exemplos para cada classe, criando assim uma base de dados com uma característica desbalanceada. Entretanto, para a segunda abordagem, foram retirados alguns exemplos de assinaturas forjadas de forma pseudoaleatória, a fim de criar uma base de dados balanceada. Por fim, na terceira abordagem, além de se utilizar todos os dados da primeira abordagem, utilizou-se também os exemplos genuínos de indivíduos que não possuíam assinaturas forjadas, criando desta forma uma base de dados com uma característica *quasi*-balanceada. Uma descrição mais detalhada da composição dos dados em cada uma das abordagens pode ser visualizado na Tabela 2.

**Figura 16: Detalhamento da quantidade de exemplos por finalidade e por abordagem.**

**(a) Abordagem 1**




**(b) Abordagem 2**




**(c) Abordagem 3**

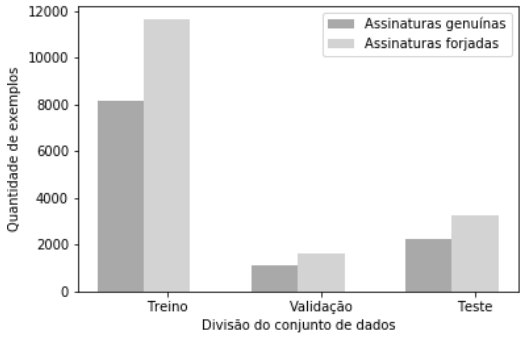



**Tabela 2: Quantitativo de exemplos de cada classe por finalidade dos dados na tarefa de AM.**

| Abordagem | Característica | Tipo de Exemplo | Treino | Validação | Teste | Total | Proporção |
|---|---|---|---|---|---|---|---|
| 1 | Dados desbalanceados | Genuíno | 2.011 | 299 | 618 | 2.928 | 15% |
| | | Forjado | 11.649 | 1.648 | 3.237 | 16.534 | 85% |
| 2 | Dados balanceados | Genuíno | 2.011 | 299 | 618 | 2.928 | 50% |
| | | Forjado | 2.024 | 308 | 569 | 2.901 | 50% |
| 3 | Dados *quasi*-balanceados | Genuíno | 8.131 | 1.134 | 2.257 | 11.522 | 41% |
| | | Forjado | 11.649 | 1.648 | 3.237 | 16.534 | 59% |

## 4.4. Modelos de CNN Considerados

# 5. Resultados Parciais

# 6. Considerações Parciais

# Referências

ACADEMY, D. S. *Deep Learning Book*. 2019. Disponível em ⟨http://deeplearningbook.com.br/⟩. Accesso 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. 10th International Conference on Document Analysis and Recognition, Barcelona, Catalonia, Spain, 2009.

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.

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.

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.

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.

IMAGENET. 2019. Disponível em: ⟨http://www.image-net.org/⟩. Accesso em 19 de março de 2019.

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

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

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.

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.

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⟩. Accesso em 5 de março 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.

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.

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.