

Verificação de Autenticidade de Assinaturas Manuscritas Utilizando Redes Neurais Convolucionais

Marcos Wenneton Araújo
Orientadora: Elloá B. Guedes

¹Laboratório 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 Artificiais, na Seção 2.2, um dos modelos inferenciais mais representativos. As definições elementares da técnica de *Machine Learning* conhecida como *Deep Learning* são apresentadas na Seção 2.3. A Seção 2.3.1 discorre sobre as características das Redes Neurais Convolucionais, prosseguindo até a Seção 2.3.2 na qual são apresentadas algumas de suas arquiteturas canônicas. Por fim, a Seção ?? contém informações sobre a técnica conhecida como *Transfer Learning*, um conceito emergente utilizado em aplicações de *Deep Learning*.

Deep Learning é
uma técnica?

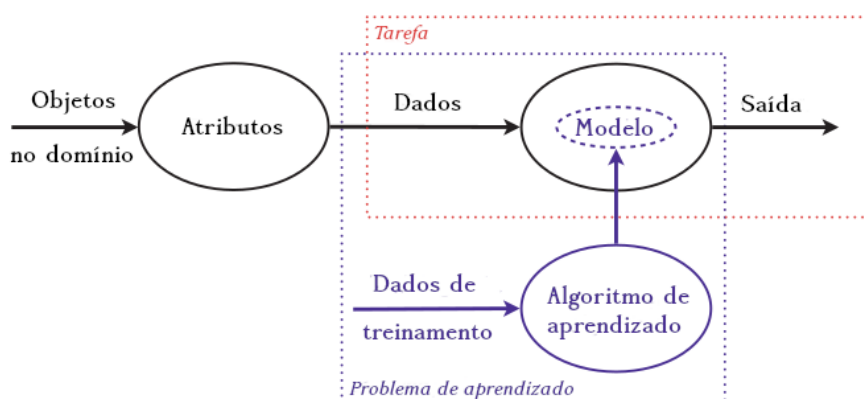
Rever introdução

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 prever 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 $f : \mathbb{R}^n \rightarrow \{1, \dots, k\}$. 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 $y \in \{1, \dots, k\}$. 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 probabilísticos, 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.

Construir transição sutil para próxima seção. Sugestão: citar vários exemplos de problemas resolvidos com AM.

Na próxima seção serão descritas e apresentadas as redes neurais artificiais, um dos modelos de AM para o paradigma supervisionado com papel protagonista nas soluções apresentadas.

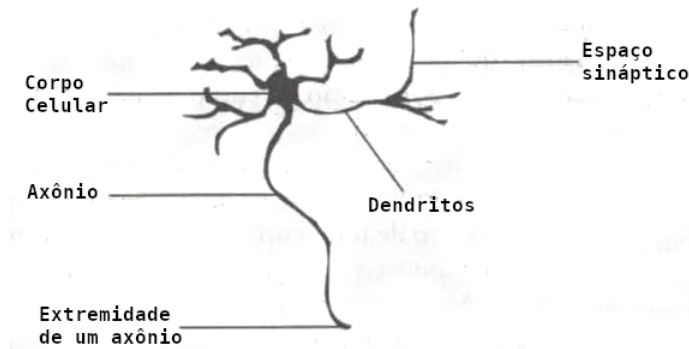
2.2. Redes Neurais Artificiais

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).]

Olhar Facelli

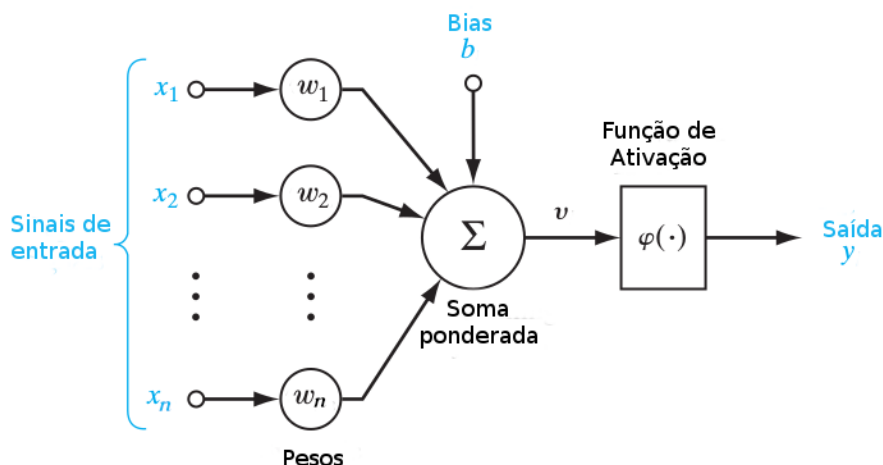
A idealização dos neurônios artificiais foi inspirada nos neurônios biológicos encontrados no cérebro humano. Como mostrado na Figura 2, cada neurônio biológico é composto pelo corpo celular, os dendritos e o axônio. Os dendritos têm como papel a recepção das informações, ou impulsos nervosos, de outros 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 dendritos, 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 $x = x_1, \dots, x_n$ aos quais estão sujeitos um conjunto de pesos $w = w_1, \dots, w_n$. 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

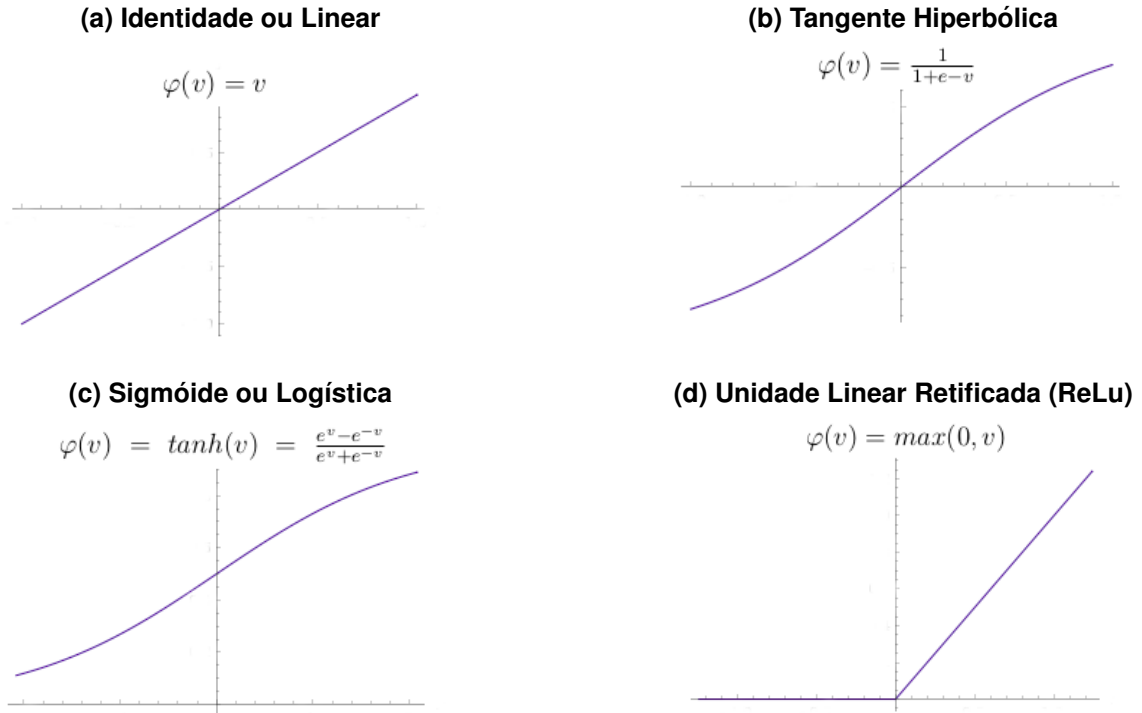
na Equação 2.

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

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

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.



Em 1958, visando a melhoria do neurônio MCP, Frank Rosenblatt desenvolveu o modelo *Perceptron* (ROSENBLATT, 1958). Neste modelo, criou-se o primeiro conceito de aprendizado através de neurônios artificiais, em que foi projetada uma regra de correção de erros para modificar os pesos associados a um neurônio quando suas respostas aos estímulos apresentados ao modelo forem erradas (ARBIB, 2003). Durante o processo de adaptação à resposta real, deseja-se identificar um valor Δw a ser aplicado ao vetor de pesos atual $w(t)$, para que seu valor atualizado $w(t + 1)$ esteja mais próximo da solução desejada do que o valor atual $w(t)$. Para isso, definiu-se a Equação 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

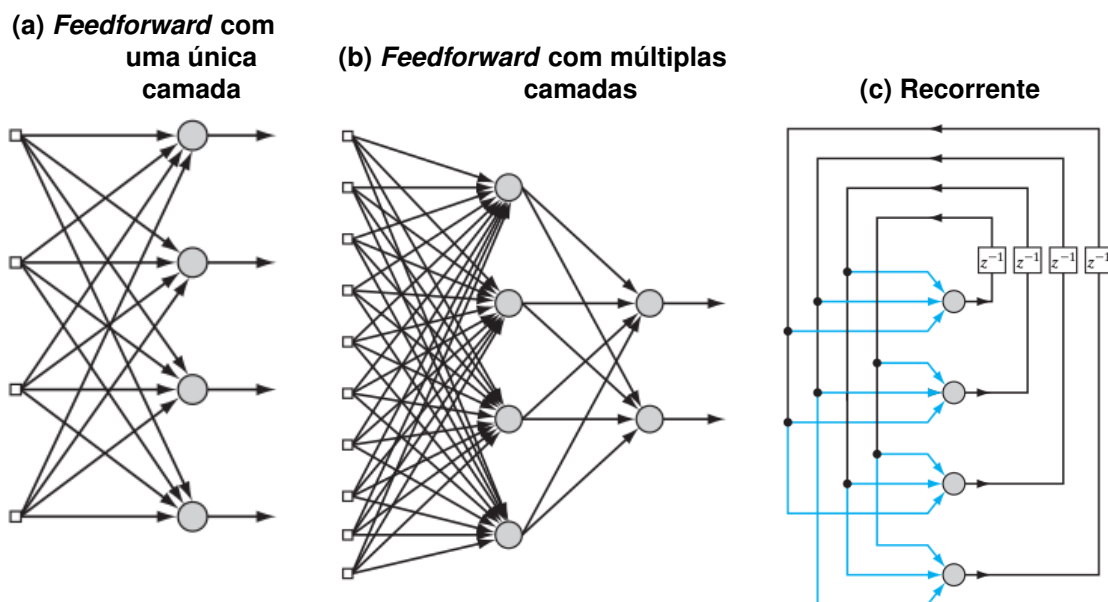
modelo naquela iteração para a entrada $x(t)$, enquanto $y(t)$ refere-se à saída real para esta entrada. Desta forma, o neurônio Perceptron adquiriu a capacidade de resolver problemas linearmente separáveis (BRAGA; CARVALHO; LUDERMIR, 2000).

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

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

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 mostrado 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 (HAYKIN, 2009).

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



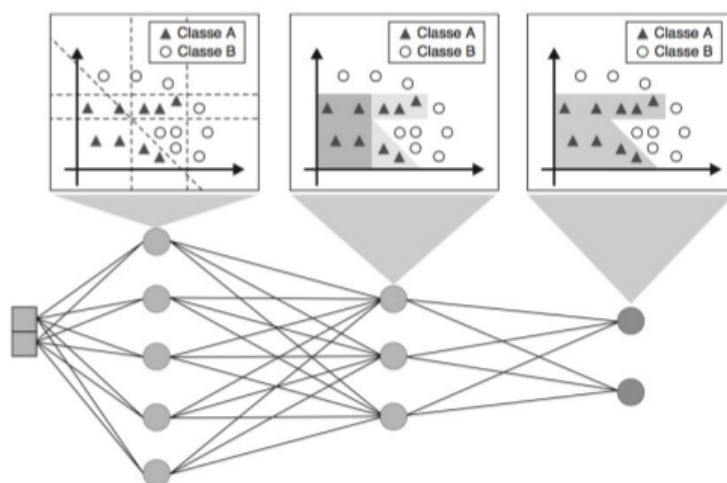
Redes com múltiplas camadas, como na Figura 5b, caracterizam-se pela presença de pelo menos uma camada oculta. Isso acarreta um grande poder às redes deste tipo pois, conforme Cybenko, uma rede com uma camada oculta é capaz de mapear qualquer função contínua, enquanto uma rede com duas camadas ocultas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

tremamente 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).

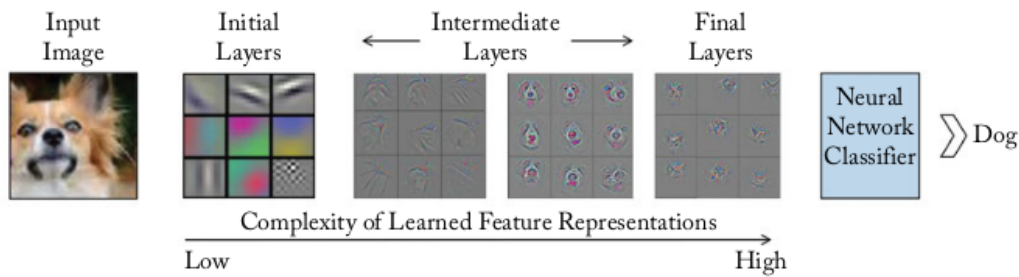
A *convolução* é uma operação que consiste na soma dos produtos de toda a extensão de duas entradas em função de um deslocamento. Sendo assim, a convolução $s(t)$ de duas entradas $x_1(t)$ e $x_2(t)$ é uma função representada simbolicamente por $s(t) = x_1(t) * x_2(t)$ e definida conforme a Equação 5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

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

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 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 7. 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).

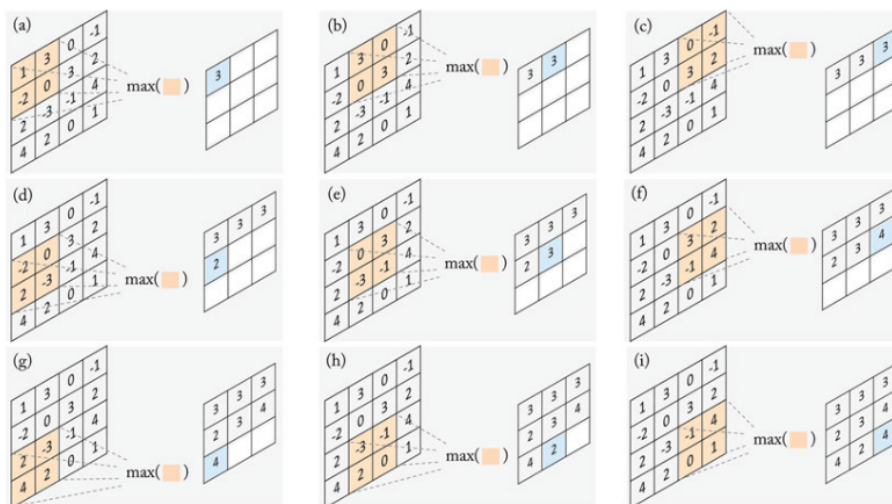
Figura 7: Processo reproduzido pelas camadas convolucionais de uma CNN, aplicado a um problema de classificação de imagens. Fonte: (KHAN et al., 2018).



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 8.

Figura 8: 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* 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 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). Na Figura 9 é possível visualizar este processo.

Dropout (ACADEMY, 2019)

Inserir essa imagem no contexto.

Figura 9: MLP após a aplicação do *dropout*. Fonte: (ACADEMY, 2019)

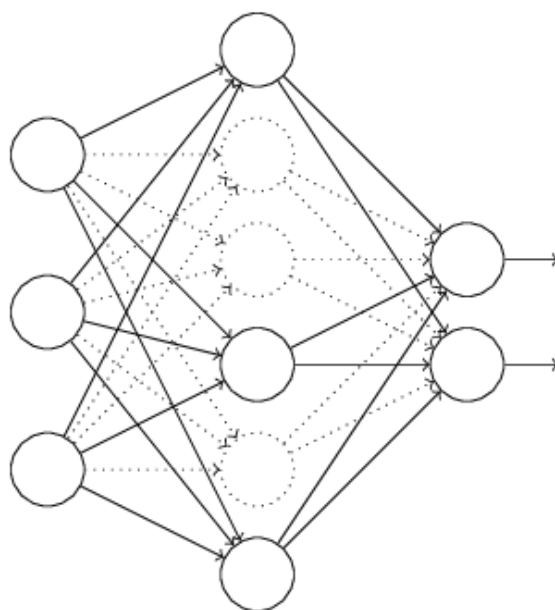
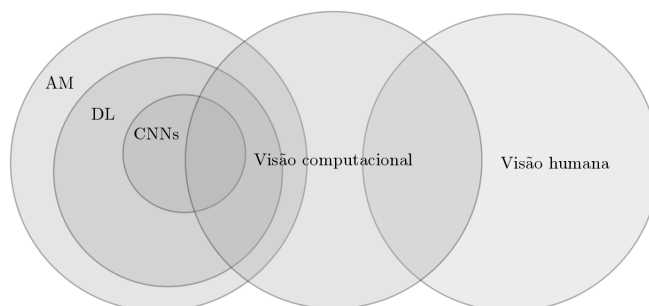


Figura 10: A relação entre a visão humana, visão computacional, ML, DL e CNNs. Adaptado de: (KHAN et al., 2018).



2.3.2. Arquiteturas Canônicas de Redes Neurais Convolucionais

3. Trabalhos Relacionados

4. Solução Proposta

5. Visão Geral do Conjunto de Dados

6. Preparação do Conjunto de Dados

7. Resultados Parciais

8. Considerações Parciais

Referências

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

- ARBIB, M. A. (Ed.). *The Handbook of Brain Theory and Neural Networks*. Cambridge, Massachussets: The MIT Press, 2003.
- 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. <http://www.image-net.org/>. Acesso em 28 de fevereiro de 2019.
- KALCHBRENNER, N.; GREFFENSTETTE, 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.
- 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.
- 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.
- SIMON, P. *Too Big to ignore*. Hoboken, New Jersey: John Wiley and Sons, Inc., 2013.