UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Análise do desempenho da transferência de aprendizagem para classificação de autômatos celulares

Daniel Santos da Silva

Trabalho de Conclusão de Curso
MBA em Inteligência Artificial e Big Data

ICMC USP
SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Análise do desempenho da transferência de aprendizagem para a classificação de autômatos celulares

Daniel Santos da Silva

USP - São Carlos

2022

Daniel Santos da Silva

Análise do desempenho da transferência de aprendizagem para a classificação de autômatos celulares

Trabalho de conclusão de curso apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Odemir Martinez Bruno

USP - São Carlos

2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S586a

Silva, Daniel
Análise do desempenho da transferência de aprendizado para a classificação de autômatos celulares / Daniel Silva; orientador Odemir Bruno. -- São Carlos, 2022.
59 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Autômatos Celulares. 2. Redes Neurais. 3. Transferência de aprendizado. 4. Classificação de imagens. I. Bruno, Odemir, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

AGRADECIMENTOS

Em primeiro lugar, a Deus, por ter permitido que eu tivesse saúde e por me ajudar a ultrapassar todos os obstáculos encontrados ao longo do curso.

A todos da minha família que me apoiaram, incentivaram e sempre se dedicaram para garantir-me a melhor educação possível.

À minha esposa Bárbara, pelo amor, companheirismo, paciência, insistência e compreensão em relação à minha ausência em todos os momentos desta jornada.

Ao meu orientador pela indispensável ajuda e apoio e por despertar meu interesse por esse tema fascinante.

A todos os professores e tutores do curso MBA em Inteligência Artificial e Big Data pelos ensinamentos que permitiram apresentar um melhor desempenho no meu processo de formação profissional.

A todos aqueles que contribuíram de alguma forma, para a realização deste trabalho, com ideias e conselhos.

RESUMO

SILVA, D. S. **Análise do desempenho de Transferência de Aprendizagem para classificação de autômatos celulares**. 2022. 52 f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

Autômatos Celulares (AC) são modelos matemáticos que conseguem simular comportamentos de sistemas complexos dinâmicos com definições simples. Busca-se esse conhecimento, quando se procura entender a complexidade destes em diversas áreas da Ciência. As aplicações atuais se baseiam, principalmente, nos trabalhos de John von Neumann, John Conway e Stephen Wolfram, que foram os principais precursores desse estudo. A evolução da dinâmica desses AC pode ser classificada de diferentes maneiras. Aqui usa-se as classificações de Wolfram e Li-Packard que separam esses autômatos em quatro e seis classes, respectivamente. Com o uso de imagens que ilustram as dinâmicas das evoluções desses modelos, é possível extrair informações e padrões com intuito de classificá-las automaticamente usando redes neurais. A proposta deste trabalho é analisar o desempenho da transferência de aprendizado para a classificação de autômatos celulares e comparar um modelo proposto simplificado de rede neural convolucional com as redes *Inception* e *ResNet50*. Para isso, é conduzida uma avaliação experimentos seguida de uma análise dos resultados de acurácia, área sob a curva ROC, precisão, revocação e pontuação F1. Foram testadas algumas técnicas de transferência de aprendizado nos modelos avaliados em base de dados de autômatos celulares e totalistas.

Palavras-chave: autômato celular; rede neural convolucional; transferência de aprendizagem; classificação.

ABSTRACT

SILVA, D. S. **Analysis of Transfer Learning performance for cellular automata classification.** 2022. 52 f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

Cellular Automata are mathematical models simulating systems that are complex systems with simple definitions. This knowledge is sought when understanding the complexity of behaviors in different areas of Science. Current applications are mainly based on the work of John von Neumann, John Conway and Stephen Wolfram, who were the main precursors of this study. The evolution of the dynamics of these AC can be classified in different ways. Here we use the Wolfram and Li-Packard classifications that separate these automata into four and six classes, respectively. With the use of images that illustrate the dynamics of the evolution of these models, it is possible to obtain information and patterns in order to classify them automatically using neural networks. The purpose of this work is to analyze the performance of Transfer Learning applied for the classification of cellular automata and compare a proposed simplified model of convolutional neural network with the Inception and ResNet50 networks. For this, an experimental evaluation is conducted followed by an analysis of the results of accuracy, area under ROC curve, precision, recall and F1 score. Some Transfer Learning techniques were tested in the models evaluated in elementary and totalistic cellular automata databases.

Keywords: cellular automata; convolutional neural network; transfer learning; classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Muitos sistemas físicos, biológicos, econômicos e sociais interagem entre si de forma não-linear exibindo propriedades coletivas decorrentes da combinação de múltiplos agentes interagentes cujos acoplamentos podem ser diversos (SIMÕES, 2016). Essas propriedades, consideradas emergentes, exibem comportamentos com efeitos de larga escala que não são trivialmente previstos ou auto-organizados a partir das características de seus agentes. Esse tipo de sistema é definido como sistema complexo (MICHELL, 2011).

A tentativa de se construir um esquema teórico geral que capture a evolução do comportamento emergente desses fenômenos baseado em algum conjunto de regras cujas soluções descrevem os seus estados ao longo do tempo deu origem a novos ramos da física, como a teoria do caos e a física dos sistemas complexos.

Uma classe particular de modelos matemáticos e computacionais que vem recebendo cada vez mais atenção nos últimos anos, principalmente por permitir realizar essa investigação, é a dos autômatos celulares (AC) (GLERIA *et al.*, 2014). Esses são modelos matemáticos utilizados para a determinação e/ou formação de padrões dos fenômenos da natureza, bem como comportamentos sociais, ambientais e biológicos, assim como fatores que influenciam em suas regras de interrelação/interação (GREMONINI; VICENTINI, 2008). Consideradas estruturas computacionais de construção extremamente simples, que podem gerar um amplo espectro de padrões complexos permitindo a manipulação direta de seus parâmetros para o estudo de sua dinâmica (WOLFRAM, 2002).

Um AC é definido por seu espaço celular e por sua regra de transição e possui na sua estrutura um conjunto organizado de células que podem assumir um dos estados de um conjunto finito, transitando entre eles ao longo do tempo discreto devido à ação de uma das funções de transição local de um conjunto finito de probabilidades. Dentre as habilidades dos AC, destaca-se a possibilidade de execução de tarefas computacionais complexas, mesmo sendo baseados em iterações simples e locais, o que tem atraído cada vez mais o interesse dos pesquisadores (MITCHELL, 1996). O entendimento desse comportamento pode auxiliar os pesquisadores na elaboração de novos paradigmas computacionais através do uso de ferramentas relacionadas, como, por exemplo, redes neurais artificiais.

Inspiradas no cérebro humano, as redes neurais artificiais são sistemas computacionais aptos a reconhecer padrões complexos, tais como os de um autômato celular, com uma grande capacidade de aprendizado (CAMOLESI, 2020). Essas redes usam diferentes camadas de

processamento para entender e extrair informações relevantes dos dados de entrada, e são frequentemente usadas em aplicações que vão desde auxílio em diagnósticos médicos até reconhecimento de faces em imagens (GOODFELLOW; BENGIO; COURVILLE, 2016).

Atualmente, existem várias arquiteturas de redes neurais artificias implementadas, entre elas, as Redes Neurais Convolucionais (*Convolutional Neural Network*, em inglês). Estudos indicam que essas arquiteturas de redes neurais fornecem altos níveis de desempenho quando comparadas aos modelos estatísticos convencionais porque esses modelos artificiais podem lidar bem com as incertezas de dados espaciais (BEZERRA *et al*, 2011). Além das redes neurais convolucionais utilizarem uma arquitetura bem adaptada, têm demonstrado alto desempenho no reconhecimento de padrões e classificação de imagens, sendo frequentemente utilizada para essa tarefa.

Neste projeto, serão avaliados os desempenhos de algumas redes neurais aplicadas à tarefa de classificação de imagens que representam a dinâmica de autômatos celulares.

1.1 Justificativa e motivação

A escolha pela utilização de autômatos celulares para serem os itens a serem classificados recai na importância da aplicabilidade desses modelos na área de simulação de dinâmicas complexas. Através de seus usos é possível extrair padrões globais através do comportamento local de um número reduzido de elementos, e que proporcionam neste caso, um estudo mais aprofundado e, principalmente, auxilia na compreensão e classificação desses fenômenos (AGUIAR, 2014).

Os modelos são definidos como estruturas simplificadas do funcionamento de um aspecto do mundo real e construídos para retratar fenômenos em escalas específicas, destacando somente as características fundamentais. A utilização dessas representações se mostra útil quando os experimentos não podem ser aplicados na realidade devido às limitações de custos e de regras específicas do contexto analisado (CHORLEY & HAGGETT, 1967).

Desta forma, a representação da evolução temporal de um autômato celular forma uma imagem que revela padrões que inspirou vários pesquisadores a classificar os diferentes tipos de AC conhecidos. A pesquisa sobre autômatos celulares cresceu rapidamente desde que Wolfram (1983; 1984) constatou que esses sistemas aparentemente simples podem gerar

estruturas bastante complexas e além de fornecerem uma técnica útil para explorar uma ampla gama de questões teóricas fundamentais em dinâmica e evolução.

Porém, a principal dificuldade dessa abordagem é entender como a estrutura visual de um AC está relacionada à representação de baixo nível da simulação e a codificação de algum problema. No entanto, classificar autômatos se torna uma maneira útil e sucinta de descrever quando analisado de forma qualitativamente o comportamento representado na imagem produzida (COMELLI *et al*, 2021, SILVERMAN, 2019).

O aprendizado profundo emergiu como um dos métodos que mais cresceram de forma acelerada dentro do paradigma de aprendizado de maquina e no contexto de inteligência artificial. Um dos principais benefícios desse método é a utilização para classificação de imagens (HASSAN, 2019).

Essa conjuntura motivou o desenvolvimento do presente trabalho, visto que oferece uma alternativa simplificada para tratar o problema de classificação dos autômatos celulares.

1.2 Questões de pesquisa e objetivos

Neste trabalho, o objetivo principal é analisar o desempenho da transferência de aprendizado na tarefa de classificação de Autômatos Celulares Elementares e Totalistas baseados nas classificações de Wolfram e Li-Packard. Para tanto, será realizada uma análise quantitativa e comparativa de desempenho com modelos de redes neurais convolucionais considerados estado-da-arte e o modelo que será proposto neste trabalho.

A fim de atingir o objetivo proposto, são definidos os seguintes objetivos específicos:

- Gerar as bases de dados com as imagens dos autômatos celulares
- Definir a arquitetura para modelo de rede neural convolucional que será avaliado, os parâmetros configuráveis e as métricas para avaliação
- Utilizar a técnica de Transferência de Aprendizado para treinar os modelos estado-da-arte e o modelo proposto
- Analisar os resultados das métricas após os testes dos modelos

Diante dos desafios e problemas atualmente enfrentados na classificação de autômatos celulares utilizando-se redes neurais convolucionais, foram elaboradas as seguintes questões de pesquisa que nortearão este projeto:

- O modelo proposto com uma arquitetura de rede neural convolucional mais simplificada pode obter um desempenho igual ou superior aos modelos estado-da-arte quanto à utilização da transferência de aprendizado para a realização da tarefa?
- Como os resultados deste trabalho podem auxiliar na decisão de qual abordagem deve ser utilizada em trabalhos futuros?

1.3 Organização deste trabalho

Este trabalho está organizado da seguinte maneira: o Capítulo 2 aborda a fundamentação teórica, apresentando conceitos relacionados a Autômatos Celulares e às classificações consideradas para o desenvolvimento do projeto; e a Redes Neurais Convolucionais e seus parâmetros; o Capítulo 3 descreve a metodologia executada durante o desenvolvimento do projeto, discutindo as etapas desde a geração das bases de dados até a Transferência de Aprendizado; no Capítulo 4 é apresentada a análise dos resultados obtidos neste trabalho, além de responder às questões de pesquisa levantadas na Seção 1.2; Por fim, é possível encontrar as considerações finais e discussões sobre trabalhos futuros no Capítulo 5

2 REVISÃO BIBLIOGRÁFICA

2.1 Autômatos celulares

O tópico de Autômatos Celulares (AC) tem sido bastante explorado devido às diversas aplicações que emergem de áreas como processamento de imagens, criptografia, redes neurais, desenvolvimento de dispositivos eletrônicos e modelagem de sistemas biológicos; e por isso, merecem destaque pela simplicidade de simulação e por fornecerem resultados bastantes promissores e condizentes com os dos sistemas reais (GHOSH & ADSULE, 2018).

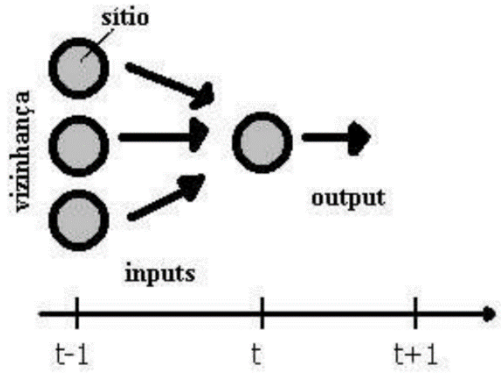
O conceito base para a aplicação dos autômatos celulares preconiza que é possível extrair padrões globais através do comportamento local de um número reduzido de elementos. Da observação desses comportamentos individuais são extraídos grupos que categorizam padrões, formas e intensidades no espaço e no tempo (MASSABAKI, 2017). A característica que torna os AC atrativos deve-se ao fato deles possuírem um atributo de espaço (funciona em um "universo") e, embora os elementos sejam conhecidos (na medida em que são definidos pelo utilizador), o comportamento é independente. Além disso, apesar de sua construção simples, os autômatos celulares são capazes de capturar uma rica variedade de comportamentos complexos (BAR-YAM, 1997).

Os primeiros traços dos AC iniciaram na década de 50 quando Von Neumann baseou-se em estudos de Turing para desenvolver um modelo abstrato de crescimento e autorreprodução em biologia. Alguns anos mais tarde, inventou instruções matemáticas que futuramente serviriam para a modelagem dos fenômenos complexos e que definem formalmente os Autômatos Celulares, cujos conceitos impulsionaram o desenvolvimento da ciência da complexidade (NEUMMANN & BURKS, 1966).

Os autômatos celulares são descritos como modelos matemáticos para sistemas naturais complexos, discretos no espaço e no tempo, contendo grandes números de componentes idênticos simples, com interações locais, que podem apresentar comportamentos caóticos a partir de regras simples (WOLFRAM, 1984). Os AC consistem em uma grade de componentes individuais (células), onde cada célula pode assumir um conjunto finito de estados que se alteram em passos de tempo discreto (evolução ou iteração) de acordo com um conjunto de regras de transição local e baseando-se no estado anterior da própria célula e /ou sua vizinhança. Por meio desses estados é possível observar desde as mudanças na grade até acompanhar fenômenos geográficos (SMITH, 1994).

Segundo Souza e Silva (2013), a célula a ser atualizada analisa os sinais (*inputs*) enviados pelas células vizinhas a ela conectada e/ou um sinal enviado por ela mesma e de acordo com as regras de evolução, responde alterando-a para um novo estado (*output*), produzindo o sinal de resposta atualizado para as outras células vizinhas no próximo instante de tempo (Figura 1).

Figura 1 – Sítio (célula) sendo influenciado por seus vizinhos e enviando um sinal de resposta.

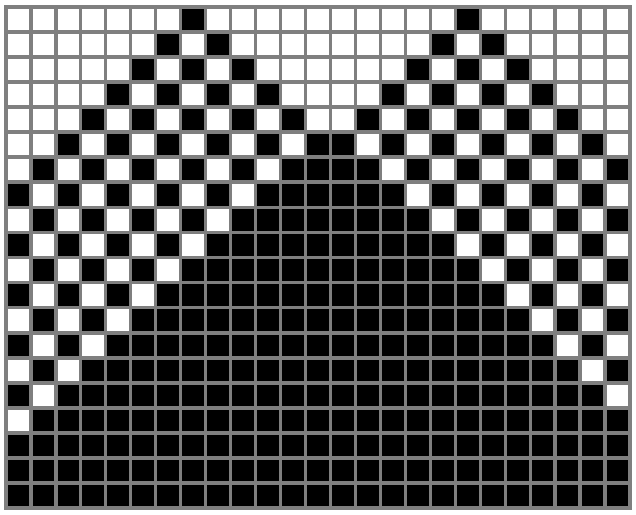


Fonte: SOUZA E SILVA, 2003.

Um autômato celular também pode ser conceituadocomo uma grade de células que possui reticulados (linhas)de tamanho *n*(definido como vizinhança global) que evolui através de*t* passos de tempo discretosde acordo com um conjunto de regras baseadas no seu estado (ou cor) e no das células vizinhas. Cada linha representa umageração ou uma iteração dos autômatos e cada célula pode assumir *k* estados. Além disso, cada célula na grade possui uma vizinhança local *m* que representa o conjunto de células que estão na adjacência a ser considerada e um conjunto de regras de transições definidoras nas quais ele evolui em passos de tempo discretos. Por fim, o padrão espaço-temporal exibe as transições do reticulado de tamanho *n* ao longo do tempo *t*e pode ser de qualquer dimensão (GHOSH & ADSULE, 2018).

É possível definir a quantidade de configurações (estados) possíveis de um AC. Por exemplo, para um AC com $k = 2$ estados e com vizinhança global $n = 10$, existirá um total de $k^n = 2^{10} = 1024$ configurações possíveis para serem visitadas. Logo, à medida que o tempo passa, o AC eventualmente revisitará configurações pelas quais já passou anteriormente. (LEWIN, 1993; SOUZA E SILVA, 2003). A Figura 2 exibe um diagrama espaço-tempo para um AC com uma regra específica.

Figura 2 – Padrão espaço-temporal de um AC unidimensional com $k = 2$ estados (cheio e vazio) e $n =$ 25células e $t = 20$.



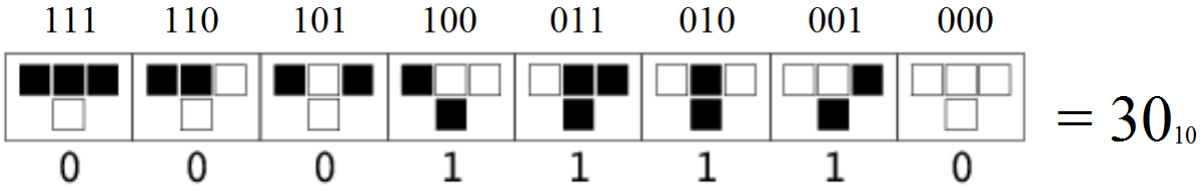
Fonte: KUNKLE, 2003.

2.1.1 Tabela de Regras

O conjunto de regras locais usadas para atualizar cada célula é chamado de tabela de regras. Essa tabela define o valor da célula analisada na próxima geração dado o conjunto possível de estados que a vizinhança local pode assumir. Da mesma forma, ela é utilizada para tentar classificar o comportamento dos autômatos celulares (KUNKLE, 2003).

Neste trabalho são considerados dois conjuntos de regras: (i) o de regras elementares e (ii) o de regras totalistas. Os Autômatos Celulares Elementares (ACE) são construídos através de um reticulado inicial unidimensional considerando $k = 2$ estados e vizinhança local $m = 3$. Isso garante 2^{2^3} tabelas de regras possíveis para este tipo de autômato, ou seja, 256 regras (WOLFRAM, 1994). Essas tabelas são definidas na forma (t_7 t_6t_5 t_4 t_3 t_2 t_1 t_0), onde a vizinhança (111) corresponde a t_7, (110) a t_6, ..., e (000) a t_0. A junção dos valores de t_7 a t_0 cria um número binário, que provê um identificador único ao ACE, que em decimal possui um intervalo de 0 a 255 (LI & PACKARD, 1990; WOLFRAM, 2002), conforme é possível visibilizar na Figura 3.

Figura 3 - Tabela de regras para ACE unidimensional – Regra 30.



Fonte: GHOSH & ADSULE, 2018 (adaptado).

Os conjuntos de regras totalistas contém regras os quais os valores se atualizam de acordo com a soma (ou a média) dos valores na vizinhança local de cada célula. Esses conjuntos podem ser especificados por $\{t_m\ t_{m-1}\ \ldots\ t_1\ t_0\}$, onde m é o tamanho da vizinhança local e cada t_i especifica o valor que a célula assumirá na próxima geração quando a soma dos valores na vizinhança local é i(WOLFRAM, 2002). O número total de regras totalistas com k estados e vizinhança m é igual k^{m+1}. Neste trabalho, adotou-se $k = 2$ estados e vizinhança local $m = 7$, alcançando 2^{7+1}, ou seja, 256 conjuntos de regras possíveis, similar ao que foi proposto por KUNKLE (2003).

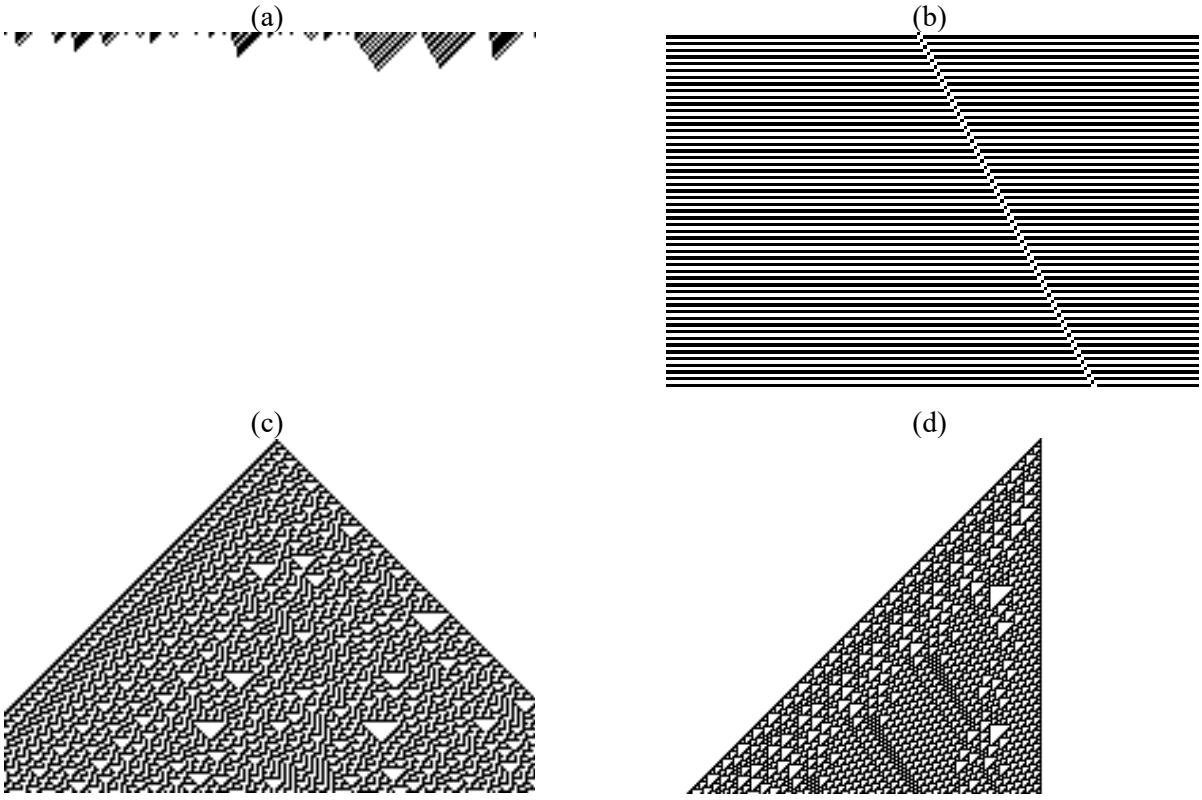
2.1.2 Classificações de Wolfram e Li-Packard

Com base na investigação de uma grande amostra de autômatos celulares, Wolfram (1984) sugere que os autômatos celulares unidimensionais binários estejam enquadrados em quatro classes qualitativas básicas baseadas na observação dos padrões gerados na evolução a partir de uma configuração inicial aleatória. As diferentes classes de comportamento de autômatos celulares permitem diferentes níveis de previsão do resultado da evolução do AC de determinados estados iniciais:

- **Classe I (comportamento fixo):** o resultado da evolução é determinado, independente do estado inicial, levando-o a um estado estável e homogêneo no qual todas as células atingem o mesmo valor;
- **Classe II (comportamento cíclico ou periódico):** o valor de uma célula é determinado pelos valores iniciais das células em uma região limitada criando estruturas periódicas com estados estáveis e espacialmente não homogêneos;
- **Classe III (comportamento caótico):** possui uma dependência cada vez maior das condições iniciais na qual a evolução no tempo leva o AC a um estado caótico, não possuindo padrão reconhecível;

- **Classe IV (comportamento complexo):** os AC formam estruturas localizadas de estados repetitivos ou estáveis capazes de sobreviver durante longos períodos, mas também formam estruturas que interagem (propagar, criar e/ou destruir) uns com os outros de forma complexa com evolução imprevisível.

Figura 4 – Padrões espaço-temporais característicos de regras típicas em cada uma das quatro classes de Wolfram. Classe I (a), classe II (b), classe III (c), classe IV (d).



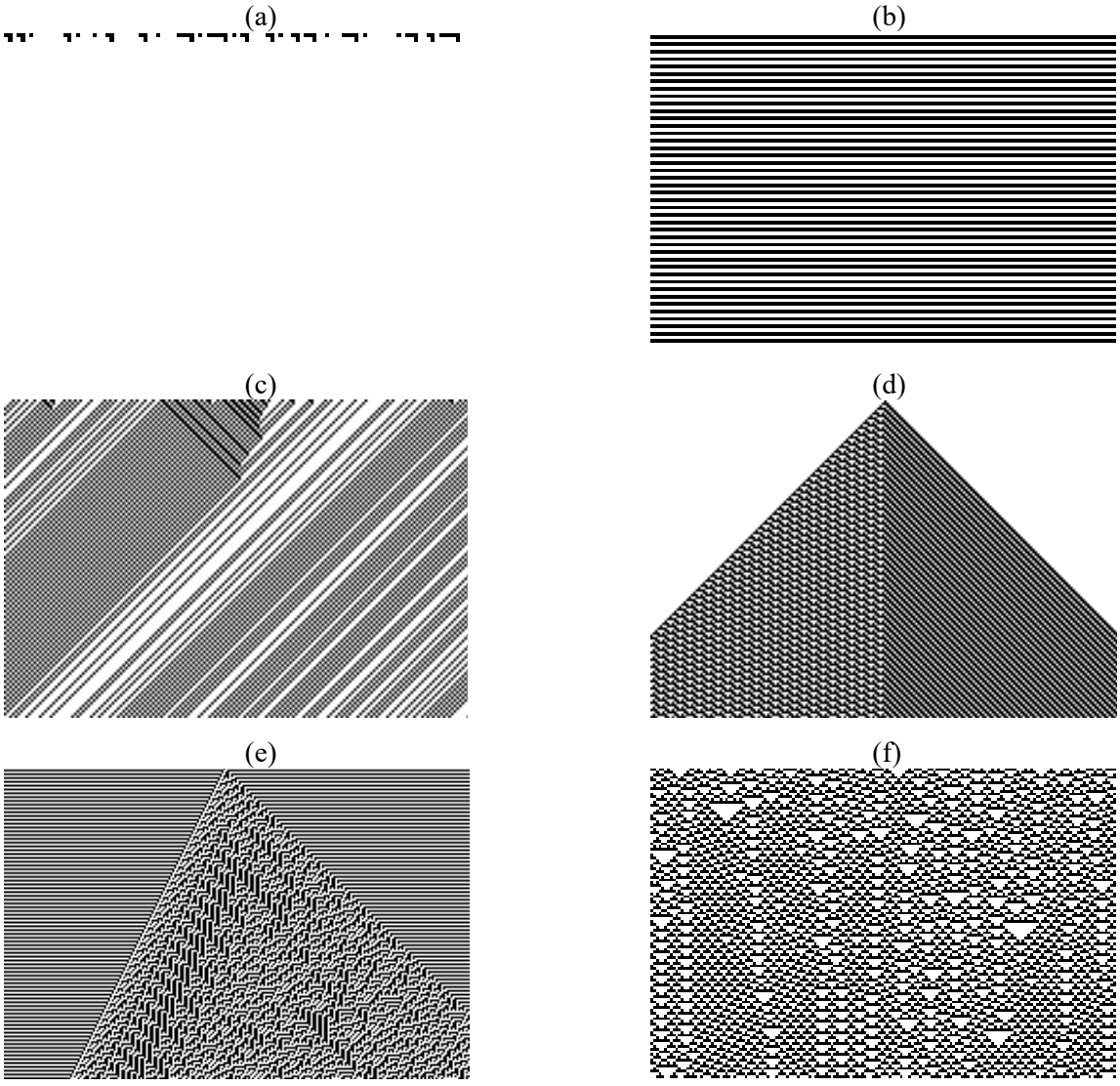
Fonte: WOLFRAM, 1984.

Baseado no trabalho de Wolfram, Li & Packard (1990) refinaram a classificação anterior e separaram os ACE em seis classes de acordo com algumas métricas estatísticas, tais como taxa de espalhamento da diferençano padrão, entropia e informação mútua. Assim, os autores puderam distinguir facilmente seis classes com base nas diferenças nas várias medidas aplicadas ao comportamento assintótico dos autômatos celulares unidimensional:

1. **Pontos fixos espacialmente homogêneos (*Null*):** a configuração limitante é homogênea, com todas as células tendo o mesmo valor;
2. **Pontos fixos espacialmente não homogêneo (*Fixed Point*):** a configuração limite é invariável após a aplicação da regra uma vez. Isso inclui regras que simplesmente mudam espacialmente o padrão e exclui regras que levam a estados homogêneos;

3. **Dois Ciclos (*Two-Cycle*):** a configuração limitante é invariável após a aplicação da regra duas vezes, incluindo regras que simplesmente mudam espacialmente o padrão;
4. **Periódico (*Periodic*):** a configuração limitante é invariável aplicando a atualização da regra L vezes, com a duração L do ciclo independente ou fracamente dependente do número de células;
5. **Caótico (*Chaotic*):** dinâmica não periódica caracterizada por uma divergência exponencial do ciclo com tamanho igual ao número de células;
6. **Complexo (*Complex*):** pode ter configurações limitantes periódicas, mas o tempo necessário para atingir essa condição pode ser extremamente longo.

Figura 5 – Padrões espaço-temporais característicos de regras típicas em cada uma das seis classes de Li-Packard. Classe *Null* (a), classe *Fixed Point* (b), classe *Two-Cycle* (c), classe *Periodic* (d), classe *Chaotic* (e), classe *Complex* (f).



Fonte: LI & PACKARD, 1990.

As Tabelas 1 e 2 resumem as distribuições das 256 regras dos autômatos elementares e totalistas (com dois estados e vizinhança igual a sete) nas classificações de Wolfram e Li-Packard.Logo, é possível perceber o desbalanceamento nas distribuições das regras, inclusive nas variações de regras classificadas comparando os totalistas e elementares na classificação de Li-Packard.

Tabela 1 – Distribuição das classes de Wolfram.

Classificação Wolfram - ACE	**Regras**
Classe I	24
Classe II	194
Classe III	32
Classe IV	6

Fonte: próprio autor (2022).

Tabela 2 – Distribuição das classs de Li-Packard.

Classes Li-Packard	**AC Elementares**	**AC Totalistas**
Null	24	44
Fixed Point	97	20
Two-Cycle	79	16
Periodic	18	20
Complex	6	2
Chaotic	32	154

Fonte: próprio autor (2022).

2.2 Redes Neurais Artificiais

Modelos computacionais que utilizam técnicas de aprendizado de máquina (*Machine Learning - ML*, em inglês) têm capacidade para resolver tarefas complexas como reconhecimento de imagens, detecção de anomalia, entre outras. Em geral, essas tarefas são solucionadas por um modelo que processa um conjunto de dados os quais têm suas características reconhecidas através de treinamentos e que são aperfeiçoados para predições utilizando-se técnicas de otimização (GOODFELLOW; BENGIO; COURVILLE, 2016).

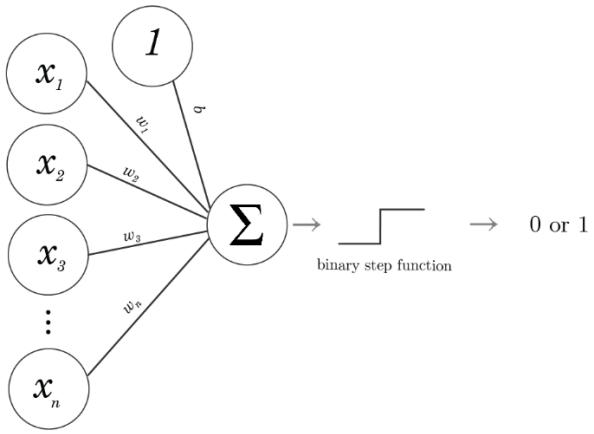
Uma rede neural é um desses modelos computacionais que é caracterizado como estrutura interconectada que se assemelha às conexões de neurônios biológicos, realiza

combinações lineares de suas entradas e emprega funções de ativação, adicionando um comportamento de não-linearidade para modelar funções mais complexas(MESSIAS, 2022).

A arquitetura precursora das Redes Neurais Artificiais (RNA) é a do *Perceptron* de camada única (*Single Layer Perceptron - SLP*), introduzido por Rosenblatt (1958), caracterizada como um classificador binário que faz previsões com base em uma função preditora linear, combinada com um conjunto de pesos e um vetor de características.

De acordo com Soares e Silva (20111), nesse modelo uma unidade de aprendizagem (neurônio) recebe um vetor $\vec{X}$ com múltiplas entradas em que cada entrada está associada a um peso w do vetor $\vec{W}$. Cada valor de entrada x_i é multiplicado ao seu peso associado w_i e o resultado é somado a um valor constante chamado *bias* (b). A unidade de saída usa uma função de sinal como uma função de ativação para processar a soma dos valores calculados para cada entrada e então, irá responder à tarefa de classificação entre duas classes no espaço {-1; 1}. Figura 6 exibe o modelo geral do *Perceptron.*

Figura 6 – Modelo *Perceptron.*



Fonte: MARQUES, 2017 (adaptado).

A exemplo, dado o conjunto de valores de treinamento $\vec{X}$ = {x_1,..., x_n} com as saídas desejadas associadas {t_1,...,t_n} ($t_i \in$ {-1, 1}) onde -1 representa a classe C_1 e 1 representa a classe C_2. Para cada entrada x_i, o *Perceptron* gera uma saída $o_i \in$ {-1, 1}. O treinamento minimiza o critério de *Perceptron* E_p (DUDA & HART, 1974; BISHOP, 1995) definido sobre o conjunto de treinamento como:

$$E_p = - \sum_{n, o_n \neq t_n} t_n \boldsymbol{w}^t \boldsymbol{x}_n$$

onde a soma é realizada sobre os exemplos classificados incorretamente ($o_i \neq t_i$).

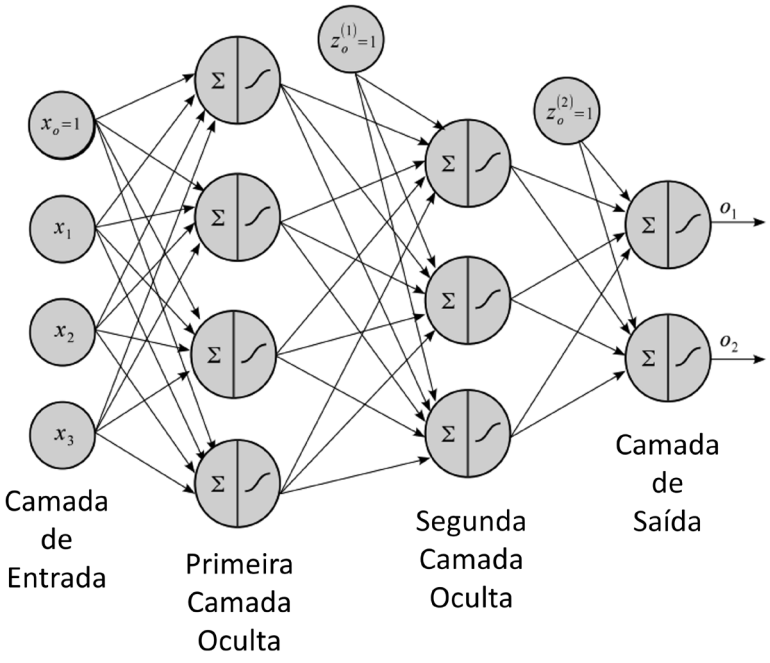
2.2.1 Rede Perceptron Multicamadas

Hopfield (1982) e Werbos (1974) contribuíram ativamente para os recentes avanços na área de redes neurais. O primeiro escreveu sobre a associatividade das redes neurais mais precisamente das redes denominadas redes de Hopfield, enquanto o segundo descreveu o algoritmo *backpropagation* proposto em sua tese de doutorado. Essas duas contribuições introduziram uma abordagem para combinar redes neurais artificiais, possibilitando assim que um modelo de rede neural que fosse capaz de solucionar problemas não linearmente separáveis. Essa combinação de *Perceptrons* em várias camadas foi denominada de *Perceptron* Multicamadas (*Multi Layer Perceptron - MLP*).

Esse modelo implementa algumas características relevantes, tais como, o uso de pelo menos uma camada intermediária entre as camadas de entrada e saída, chamada de oculta, possibilitando a adição de mais recursos; o uso de funções não lineares, como as sigmoides, como funções de ativação nas camadas ocultas e de saída; e treinamento de toda a rede de forma hierárquica, através do algoritmo de *Backpropagation* (MARQUES, 2017).

Na Figura 7 é apresentado um exemplo da arquitetura de um *MLP*. As unidades (neurônios) dispostas em camadas que compreendem uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída.

Figura 7 - Arquitetura *MLP* composta por uma camada de entrada, duas camadas ocultas e uma camada de saída.



Fonte: AGGARWAL, 2014.

As camadas são organizadas de forma sequencial da entrada a saída, passando pelas camadas intermediárias ocultas. Os dados são coletados e previamente processados na camada de entrada e, posteriormente, propagados para os neurônios das camadas ocultas. Cada uma dessas unidades recebe dados das unidades que estão na camada anterior e fornecem os seus resultados para as unidades que estão na próxima camada, até que cheguem à camada de saída que fornece a decisão final da classificação. Nas *MLP* as funções de ativação utilizadas nos neurônios das camadas ocultas e de saída são tipicamente funções sigmoidais (MARQUES, 2017).

Para problemas de classificação binária {C1, C2}, o *MLP* contém uma ou mais camadas ocultas e uma única unidade de saída com função sigmoidal produzindo um único valor de saída *o*. Supondo que o valor alvo seja 0 para a classe C1 e 1 para classe C2, a regra de classificaçãoé descrita como (AGGARWAL, 2014):

$$\begin{cases} x \in C_1 \;\; if \;\; o > 0.5 \\ x \in C_2 \;\; senão \end{cases}$$

com a saída o no intervalo de [0, 1] ou ([-1; 1] caso a função de ativação *tanh* seja usada, onde o limiar de decisão passar a ser zero).

Para problemas envolvendo várias classes ($k > 2$), o *MLP* possui uma ou mais camadas ocultas e k neurônios de saída, cada uma correspondente a uma classe específica do conjunto de treinamento. A função de destino é codificada com um esquema 1*–of–k* com valores entre [0, 1] e a saída o_k é vista como uma função discriminante da classe C_k, permitindo a decisão de classificação como (AGGARWAL, 2014):

$$x \in C_k \quad if \; k = \arg\max_j \; o_j$$

para j $\in$ {1, ..., k} com $o_k \in$ [0, 1] quando a sigmoide é utilizada como função de ativação. Ouso de um esquema de codificação 1*–of–k* para a função de destino é padrão.

2.2.2 Funções de ativação

As funções de ativação são um dos elementos mais importantes das RNA. Elas basicamente decidem se um neurônio deve ser ativado baseando-se no resultado da transformação não linear. Ou seja, verificam se a informação que o neurônio está recebendo

deve ser enviada para a próxima camada ou se deve ser ignorada. Os gradientes e os errospara atualização dos pesos e dos *bias* são (retro)propagados por essas funções (HAYKIN, 2001).

Segundo Hastie (2008), existem vários tipos de funções de ativação, cada qual com características relevantes para serem utilizadas em diferentes cenários. Segue a descrição e os gráficos (Figura 8) de um rol exemplificativo dos tipos mais populares:

- **Limiar (Degrau):** a saída do neurônio é igual a 0 quando o seu valor for menor do que o limite estabelecido e 1 caso contrário.

$$\phi(x) = \begin{cases} 1, \text{if } x \geq 0.5. \\ 0, \text{otherwise.} \end{cases}$$

- **Sigmóide:** comumente utilizada por redes neurais com propagação positiva (*feedforward*) que precisam ter como saída apenas números positivos, em redes neurais multicamadas e em outras redes com sinais contínuos, funcionando melhor em classificadores. É definida como uma função crescente com balanceamento adequado entre o comportamento linear e não linear e assume um intervalo de variação entre 0 e 1.

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

- **Tangente hiperbólica:** em muitos casos substitui a função sigmóide, pois preserva a forma sigmoidal, mas as saídas assumem valores entre -1 e 1.

$$\phi(x) = \tanh(x)$$

- **ReLU:** ela não ativa todos os neurônios ao mesmo tempo, ou seja,se a entrada for negativa, ela será convertida em zero e o neurônio não será ativado. Isso significa que, ao mesmo tempo, apenas alguns neurônios são ativados, tornando a rede esparsa e eficiente e fácil para a computação. Deve ser usada apenas nas camadas ocultas

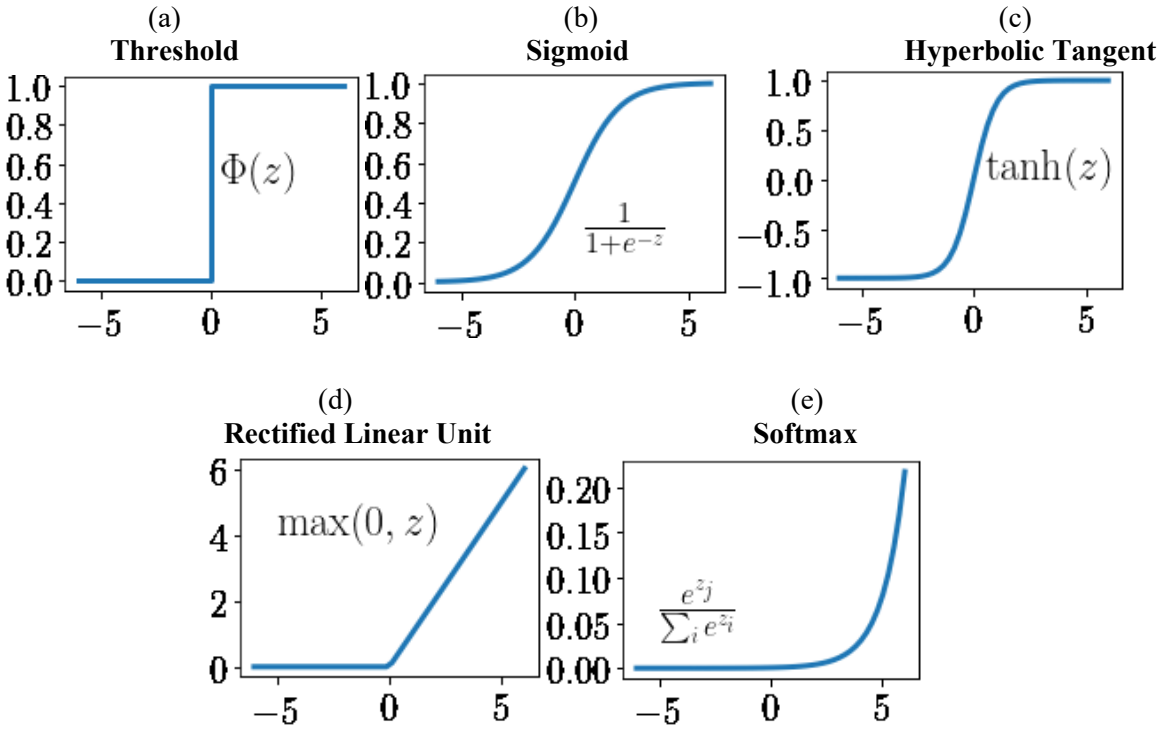
$$\emptyset_i = \max(0, z)$$

- **Softmax:** usadas em redes neurais de classificação, ela força a saída de uma rede a representar a probabilidade dos dados serem de uma das classes definidas, ou seja, a que possuir o maior valor numérico.

$$\phi_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

Na equação anterior, *i* representa o índice do neurônio de saída (*o*) sendo calculado e *j* representa os índices de todos os neurônios de uma camada. A variável *z* designa o vetor de neurônios de saída. Vale notar que a função de ativação *softmax* é calculada de forma diferente das demais apresentadas, uma vez que a saída de um neurônio depende dos outros neurônios de saída.

Figura 8 – Gráficos de diferentes funções de ativação: Limiar (a), Sigmóide (b), Tangente Hiperbólica (c), ReLU (d), Softmax (e).



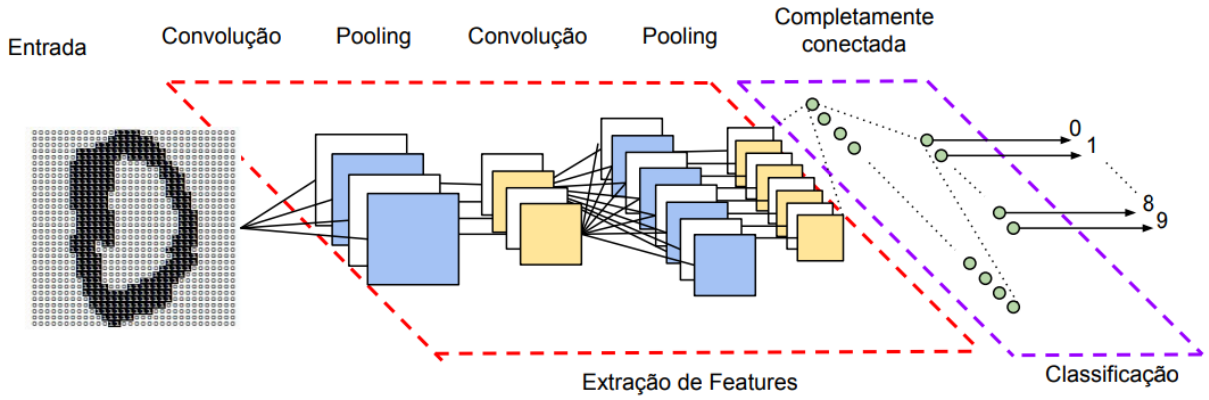
Fonte: Johnson *et al.,* 2020 (adaptado).

2.3 Redes Neurais Convolucionais

As Redes Neurais Vonvolucionais (RNC) (*Convolutional Neural Network - CNN*) são variantes das redes de *MLP* que ganharam notoriedade pelos resultados expressivos quanto à classificação, detecção e reconhecimento em imagens e vídeos, tendo sido inspirada no processo biológico de processamentos de dados visuais (VARGAS *et al.*, 2016). Uma *CNN* é uma topologia ou arquitetura capaz de aplicar filtros em dados visuais, mantendo a relação de vizinhança entre os *pixels* da imagem ao longo do processamento da rede e que se aproveita derelações espaciais para reduzir o número de parâmetros que devem ser aprendidos utilizando os métodos de *feedforward* de *backpropagation* (HINTON *et al.*, 2012). O nome

"rede neural convolucional" indica que a rede emprega uma operação matemática chamada convolução. A Figura 9 ilustra um exemplo de RNC.

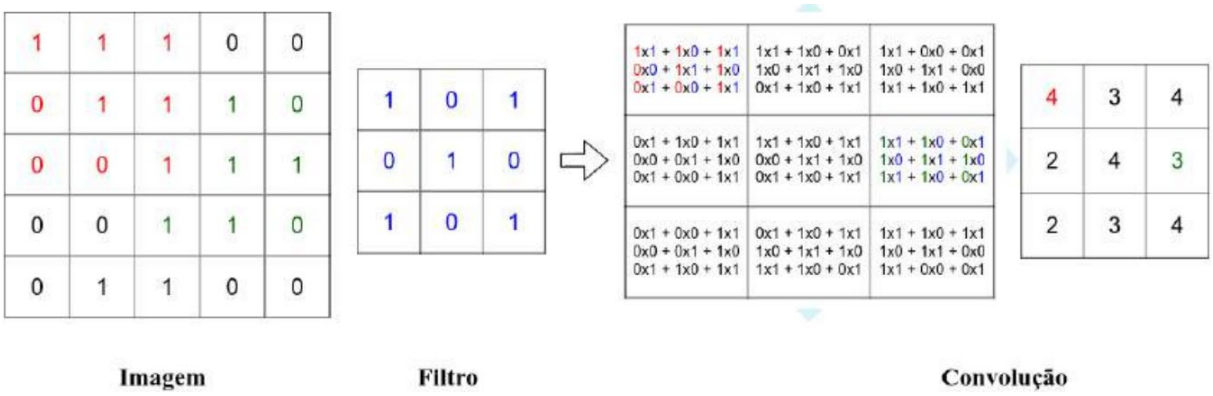
Figura 9 – Exemplo de uma rede neural convolucional e suas diferentes camadas para a tarefa de classificação.



Fonte: VARGAS *et al.*, 2016.

Convolução é uma operação matemática entre duas funções *f* e *g*, produzindo uma terceira função, que pode ser interpretada como uma função modificada de *f*. Ela pode ser interpretada como o somatório dos dados de entrada com os elementos da matriz que representa um filtro de convolução ou um *kernel* para produzir um mapa de funcionalidades (*Feature maps*). A operação é executada movendo horizontalmente e verticalmente o filtro pela matriz de entrada, na qual, para cada movimentação, é executada a multiplicação entre os valores da matriz de filtro pelos valores da entrada que estão na mesma posição e em seguida é feita a soma dos resultados (FERREIRA, 2017; MANDOJU, 2019). Esse cálculo é ilustrado na Figura 10.

Figura 10 – Exemplo de aplicação da convolução na imagem.

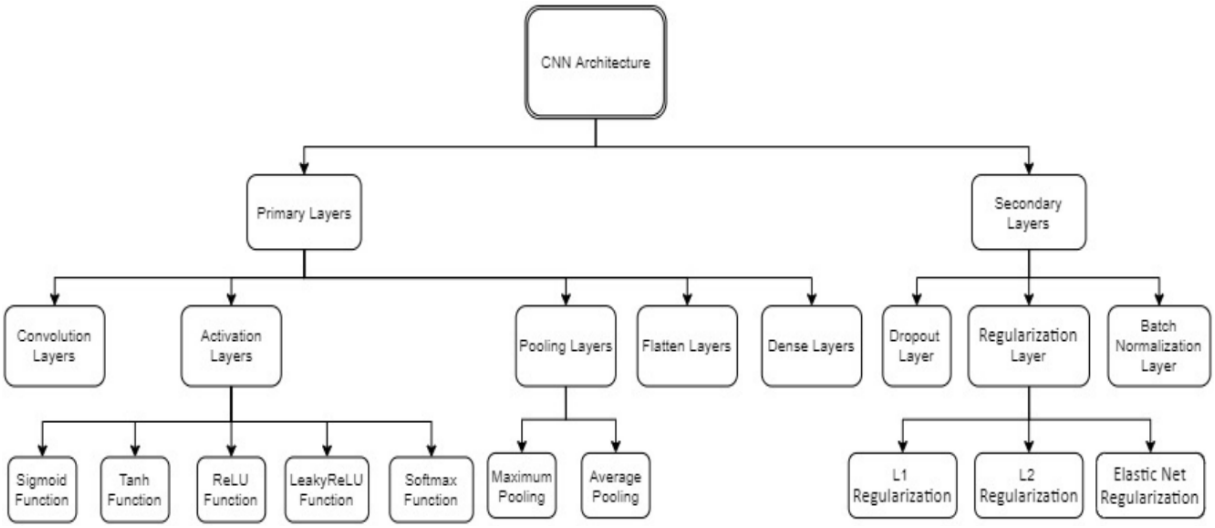


Fonte: FERREIRA, 2017.

No processamento de imagens, onde a imagem é definida como uma função bidimensional (matriz), a convolução e útil para detecção de bordas, suavização, extração de atributos, entre outras aplicações.

As camadas *CNN* podem ser classificadas em duas categorias: camadas primárias e camadas secundárias. As primárias são as camadas principais usadas nessa rede neural e consistem em camadas de convolução (*Convolution*), de ativação (*Activation*), de agrupamento (*pooling)*, de achatamento (*Flatten*) e de camadas densas (*Dense*). Já as secundárias são camadas opcionais que podem ser adicionadas para tornar as CNNs mais robustas contra *overfitting* e aumentar a capacidade de generalização, sendo elas as camadas de *dropout*, de regularização (*regularization*) e de normalização de lotes (*batch normalization*) (KANDEL & CASTELLI, 2020). A Figura 11 mostra a estrutura de uma Rede Neural Convolucional

Figura 11 – Estrutura da Rede Neural Convolucional.



Fonte: KANDEL & CASTELLI, 2020.

Nas próximas quatro seções, serão descritas as características das camadas primárias. Optou-se por não descrever sobre as camadas secundárias, pois elas não serão utilizadas neste trabalho.

2.3.1 Camada convolucional

A primeira e mais importante camada em uma rede neural convolucional é a camada de convolução, que é responsável por extrair atributos da imagem de forma automática. A área de interseção entre a matriz dos dados de entrada e o *kernel* é o valor da convolução. Normalmente, os pesos dentro dos filtros são inicializados aleatoriamente e então esses valores são atualizados durante o processo de treinamento da rede (VARGAS *et al.*, 2016, MONDOJU, 2019).

As primeiras camadas de convolução são responsáveis pela extração de características básicas da imagem, como arestas, contornos, entre outros, enquanto as camadas de convolução mais profundas extraem os recursos abstratos e a camada densa é necessária para a classificação (ZHANG, 2021).

As etapas que são executadas pelo filtro sobre a matriz de entrada, definem o parâmetro passo (*stride*). Ele representa a quantidade de movimentos que faz após executar uma ativação do filtro. Normalmente, após a operação de convolução, o mapa de atributos tem dimensão menor do que a entrada. Para contornar essa situação, pode-se usar a técnica de *padding*, que adiciona zeros ao redor do sinal de entrada para manter o tamanho original na saída (KANDEL & CASTELLI, 2020), conforme é possível visualizar na Figura 12.

Figura 12 – Estrutura da Rede Neural Convolucional.

0	0	0	0	0	0
0					0
0					0
0					0
0					0
0	0	0	0	0	0

Fonte: KANDEL & CASTELLI, 2020.

Assim, a saída da operação de convolução depende do tamanho da entrada, tamanho do filtro, valor *stride*, e uso (ou não) de *padding*. A dimensão do *feature map* é calculada de acordo com:

$$Altura = \frac{H - K + S + P}{S}, Largura = \frac{W - K + S + P}{S}$$

Onde o tamanho do filtro e *K x K*, da entrada é *H x W*, *stride* é *S* e *padding* é *P*.

2.3.2 Camada de agrupamento (*pooling*)

A camada de agrupamento geralmente é executada depois de uma camada de convolução com o objetivo de extrair as características mais significativas do mapa de atributos e fornecê-las para a próxima camada (também conhecida como subamostragem). Assim, essa funcionalidade executa operações matemáticas com intuito de diminuir o número de parâmetros na rede agrupando os valores (RAMOS NETO, 2018).

Ainda, esta camada pode ser dividida em três tipos: *MinPooling*, *AveragePooling*, e *MaxPooling*, sendo a última atécnica mais utilizada,a qual obtém os maiores valores de regiões menores delimitadas pelo *kernel*, dada a saída da camada convolucional. De forma análoga, *MinPooling* retorna os menores valores e *AveragePooling* retorna as médias dos valores (NOGUEIRA, 2020).

2.3.3 Camada de achatamento (*flatten*)

Segundo Kandel & Castelli (2020), é nesta camada que é realizado o achatamento dos dados, ou seja, a matriz de dados é convertida para um vetor unidimensional, pois as camadas posteriores só aceitam vetores 1D como entradas. A dimensionalidade do vetor resultante é dada por:

$$Dim_{flat} = Dim_{img} * Dim_{img} * num_{channels}$$

2.3.4 Camada totalmente conectada (*dense* ou *fully connected*)

A partir desta camada, tem-se a estrutura de uma rede *MLP*, pois, diferentemente das anteriores, cada neurônio dessa camada será conectado com todos os neurônios da camada posterior dentro da rede neural desta camada (por isso, totalmente conectada ou densa). O objetivo principal é considerar todos os recursos que foram extraídos das camadas anteriores e usá-los para classificar a imagem. Geralmente este é o tipo da última camada executada pela rede neural convolucional, oferecendo o resultado da execução da rede por completo. Normalmente utiliza-se a função de ativação *softmax* ou *sigmoid* (MONDOJU, 2019).

2.4 Transferência de aprendizado

A RNC é uma das arquiteturas de rede de aprendizado profundo mais conhecida e utilizada atualmente. Porém, o seu uso apresenta um grande desafio: a necessidade de grande poder computacional para treinar os milhares de parâmetros pertencente a rede. Segundo Silva (2018), a capacidade de convergência (aprendizagem) de uma rede neural convolucional é diretamente influenciada pela inicialização de seus pesos. Por isso, a etapa de definir os valores iniciais é muito importante para obtenção de bons resultados. Porém, essa acaba sendo bastante subjetiva e depende da experiência e conhecimento técnico do criador do modelo (SANTOS *et al.*, 2019).

Apesar disso, esse desafio pode ser superado importando os pesos com valores obtidos por meio de um treinamento prévio realizado em outro banco de dados para inicializar um novo modelo. Essa técnica é conhecida como Transferência de Aprendizado (*Transfer Learning*, em inglês). Dessa maneira, a inicialização é bem mais efetiva e o modelo pode convergir mais rápido, além de poder ser utilizado para problemas que compartilham similaridades (SANTOS *et al*., 2019; VOGADO *et al*., 2019). Utilizando essa técnica é possível diminuir a necessidade de retreinar todos os parâmetros, bem como utilizar as configurações já existentes na rede previamente treinada (YOSINSKI *et al.* 2014).

Várias arquiteturas *CNN* foram treinadas no banco de imagens *ImageNet* (DENG *et al.*, 2009), um banco de imagens com mais de 15 milhões de imagens separadas em mais de 22 mil classes,e conseguiram altos valores de acurácia. Esses pesos podem ser reutilizados em outros modelos para classificar outros bancos de dados completamente diferentes, ao invés de inicializar os pesos com valores gerados de forma aleatória (KANDELL & CASTELLI, 2020).

Existem duas formas principais de se utilizar a técnica da transferência de aprendizado. A primeira compreende na extração das camadas densas originais e no congelamento do restante das camadas com seus respectivos pesos e então adicionar as novas camadas *fullyconnected*ou um novo classificador, como *Support Vector Machine* (SVM). A segunda forma de transferência de aprendizado é o ajuste fino ou *fine-tuning*, que exige grande poder computacional quando comparado com a técnica anterior (VOGADO *et al.*, 2019; YOSINSKI *et al.*, 2014).

De acordo com Tajbakhsh*et al.*(2016) e Izadyyazdanabadi *et al*. (2018), existem dois tipos de ajuste fino, o primeiro é denominado *Shallow Fine Tuning* (*SFT*), que consiste no

congelamento dos parâmetros das camadas do topo da *CNN*, geralmente os conjuntos de camadas convolucionais, treinando somente as camadas totalmente conectadas. O segundo ajuste fino é denominado *Deeply Fine Tuning* (*DFT*), no qual todas as camadas são treinadas fazendo o ajuste fino em todos os pesos da rede neural usando valores baixos de *learning rate* (*LR*). Essa exige mais processamento e uma grande quantidade de dados, porém, a técnica consegue ser mais eficaz, pois garante um refinamento mais profundo com a base utilizada.

Também é possível aplicar a técnica usando alguma arquitetura estado-da-arte e começar a treiná-la do zero, usando apenas a arquitetura que provou funcionar em diferentes conjuntos de dados anteriores (KANDELL & CASTELLI, 2020).

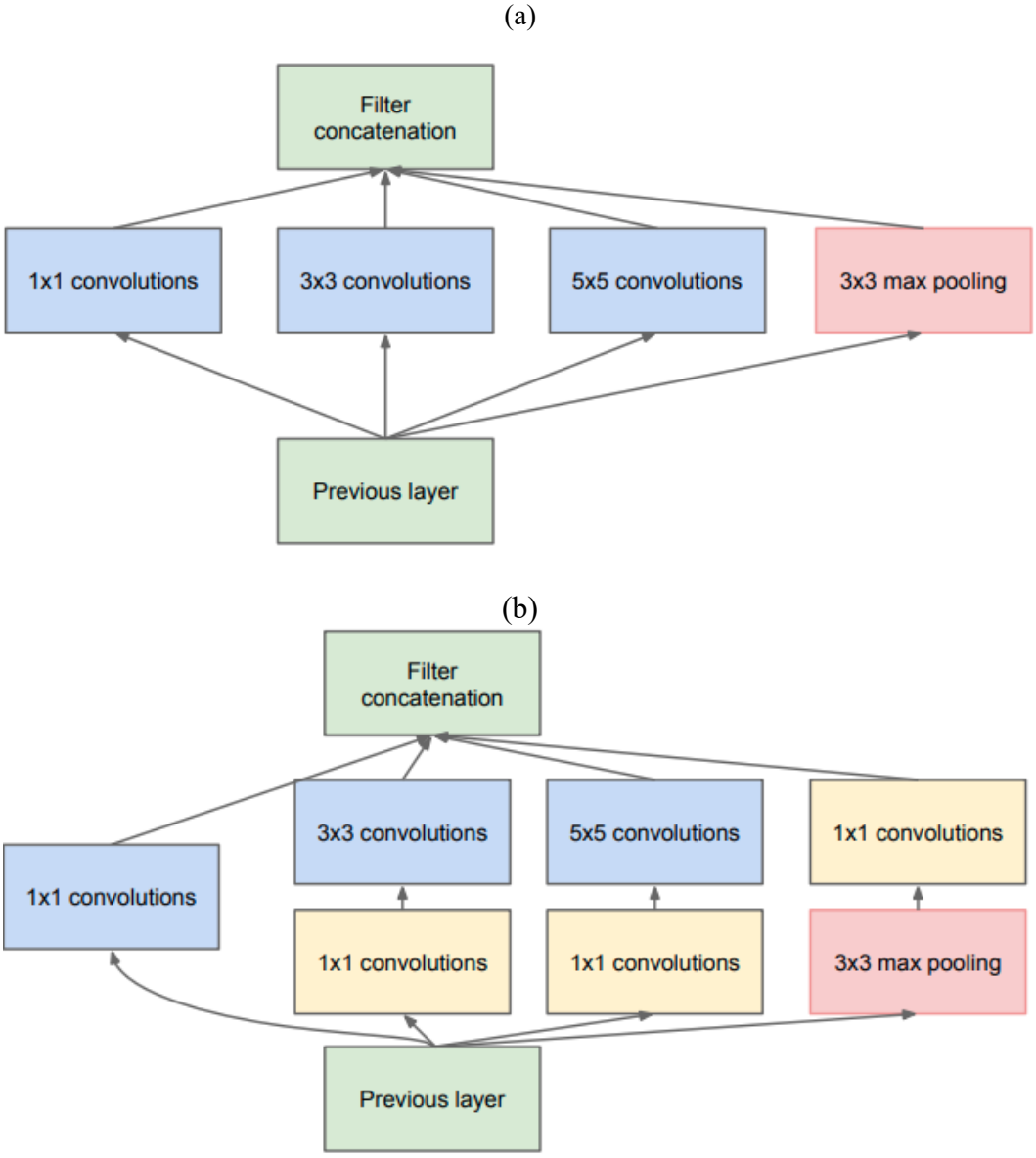
2.5 Arquiteturas *CNN* estado-da-arte

De acordo com Alom *et al.* (2018), a ascensão em pesquisas na área de aprendizado profundo para classificação de imagens começou em 2012 com a introdução da *AlexNet*, a qual introduziu a camada de ativação ReLU (KRZHEVSKY, *et al*., 2012). A exploração desse tipo de rede neural alcançou resultados cada vez mais precisos e eliminou a análise de cada imagem baseada em engenharia de recursos. Logo, várias outras arquiteturas apareceram, tais como, *Inception*, *ResNet*, e outras, as quais introduziram características mais eficientes para classificação de imagens.

2.5.1 GoogleNet Network Architecture - Inception V3

A primeira versão desta arquitetura, chamada de *GoogleNet* (*Inception* V1), foi proposta por pesquisadores do Google e se tornou consagrada ao vencer a competição da *ImageNet* em 2014 com uma taxa de acurácia Top 5 de 92,2% (SZEGEDY, *et al.*, 2015). Após o grande sucesso, os autores introduziram outras versões como *Inception* V2 e *Inception* V3. O principal benefício dessa arquitetura é a presença de módulos chamados *inception* que contêm múltiplas camadas de convolução em paralelo no mesmo bloco. Na Figura 13, é possível visualizar dois modelos desta arquitetura. Mesmo possuindo poucos parâmetros, essa rede consegue extrair várias características diferentes das imagens (HUANG, G. ET AL, 2017).

Figura 13 – Módulo *Inception*: *naive version* (a), com redução de dimensionalidade (b).



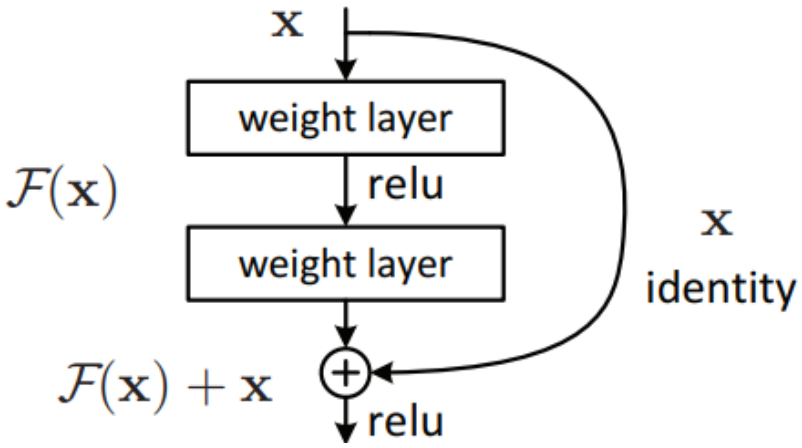
Fonte: SZEGEDY *et al* , 2015.

2.5.2 *ResNet*

ResNet, também conhecida como rede residual, foi apresentada for He *et al.* (2015) quando vendo a competição *ImageNet* daquele ano com acurácia Top 5 de 94.29% utilizado aproximadamente um total de 25 milhões de parâmetros. Comparada a outras arquiteturas, como a *Inception*, ela acaba sendo muito mais "profunda", pois ela pode atingir até 152 camadas. A *ResNet* possui uma conexão única chamada de conexão residual, que é uma conexão aplicada entre as camadas convolucionais que garante que durante a execução do

backpropagation, os pesos aprendidos das camadas anteriores não anteriores, diminuído o decaimento do gradiente. A Figura 14 exibe o modelo da conexão residual. O principal benefício desta rede é o uso destas conexões, pois possibilita a utilização de muitas camadas, além de diminuir a quantidade de parâmetros extras. As principais desvantagens são a utilização de tamanhos únicos para filtros para que possa ser efetuada a soma em cada bloco residual e a necessidade de treinamento com grandes conjuntos de dados. Três versões desta rede foram apresentadas e elas diferem no número de camadas: *ResNet50*, *ResNet101* e *ResNet152* (KANDEL & CASTELLI, 2020).

Figura 14 – Modelo do aprendizado através de conexões residuais.



Fonte: HE *et al* , 2015.

2.6 Trabalhos Relacionados

Kunkle (2015) propõe um método avançado para classificar automaticamente autômatos celulares elementares e totalistas nas seis classes de Li-Packard. Sete parâmetros foram usados para classificação, seis da literatura existente e um recém-apresentado pelo próprio autor. Esses sete parâmetros foram usados em conjunto com uma rede neural *MLP* para classificar as imagens e atingiu uma média de acurácia de 98,3% em autômatos celulares elementares e 93,9% de totalistas (considerando 2 estados e vizinhança igual 7). Além disso, os sete parâmetros foram classificados com base em sua eficácia na classificação de autômatos celulares nas classes de Li-Packard.

Em outro trabalho, Nogueira (2019) implementou dois classificadores da dinâmica apresentada pelas evoluções temporais dos AC, segundo o esquema de classificação de Wolfram, tendo como base o espaço elementar, mas tendo também por objetivo aplicá-lo em um espaço maior com quatro células na vizinhança e dois estados possíveis, cuja classificação

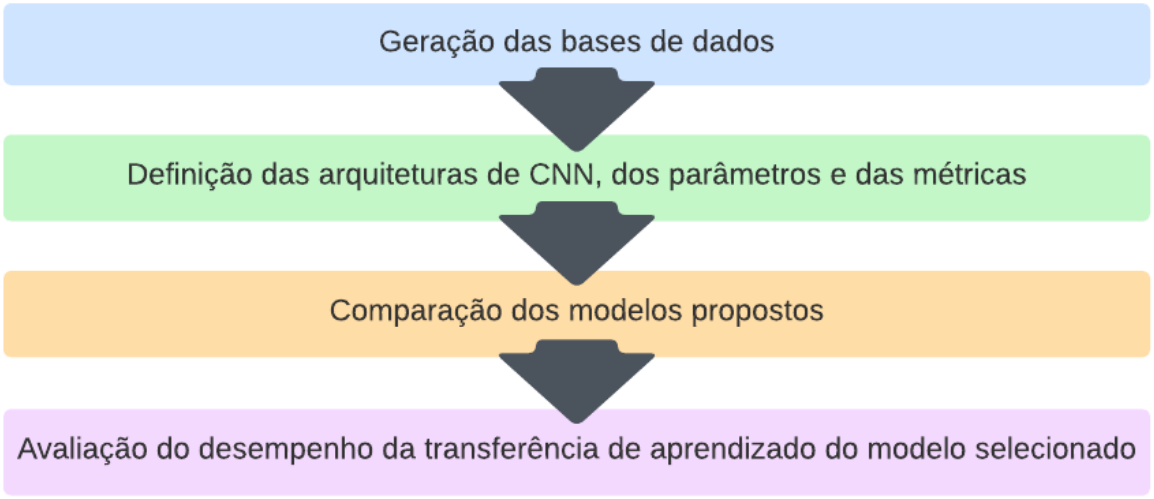
é desconhecida. Dos dois classificadores, um usou rede neural convolucional, treinada para predizer a classe da regra associada a uma evolução temporal; e o segundo usou análise de textura para extrair informações quanto às configurações de vizinhanças das células, de forma a se poder construir um espectro de frequência destas configurações.

Maji *et al.* (2003) apresenta a aplicação de um classificador de padrões de alta velocidade e baixo custo desenvolvido em torno de uma classe especial denominada *Multiple Attractor Cellular Automata* (MACA), através da formulação de um Algoritmo Genético (AG). A versatilidade do esquema de classificação é ilustrada através de sua aplicação em três campos diversos: mineração de dados, compressão de imagens e diagnóstico de falhas. Extensos resultados experimentais demonstram melhor desempenho do esquema proposto sobre algoritmos de classificação populares em relação à sobrecarga de memória e tempo de recuperação com precisão de classificação comparável.

3 METODOLOGIA

Neste capítulo será descrito a abordagem proposta para avaliar o desempenho da transferência de aprendizado de uma rede neural convolucional para classificação de autômatos celulares elementares e totalistas. A Figura 15 apresenta o fluxograma que resume a metodologia aplicada neste trabalho.

Figura 13 – Resumo da abordagem proposta.



Fonte: próprio autor (2022).

3.1 Organização e análise preliminar dos dados

Neste projeto foram utilizadas três bases de imagens referentes aos AC Elementares (ACE) e AC Totalistas (ACT). Todas as imagens foram geradas e classificadas automaticamente através de algoritmos utilizando-se funções da biblioteca *NumPy* e seguindo as funções de transição das regras dos respectivos autômatos. É possível ver na Tabela 3 a distribuição das imagens entre treinamento, validação e teste.

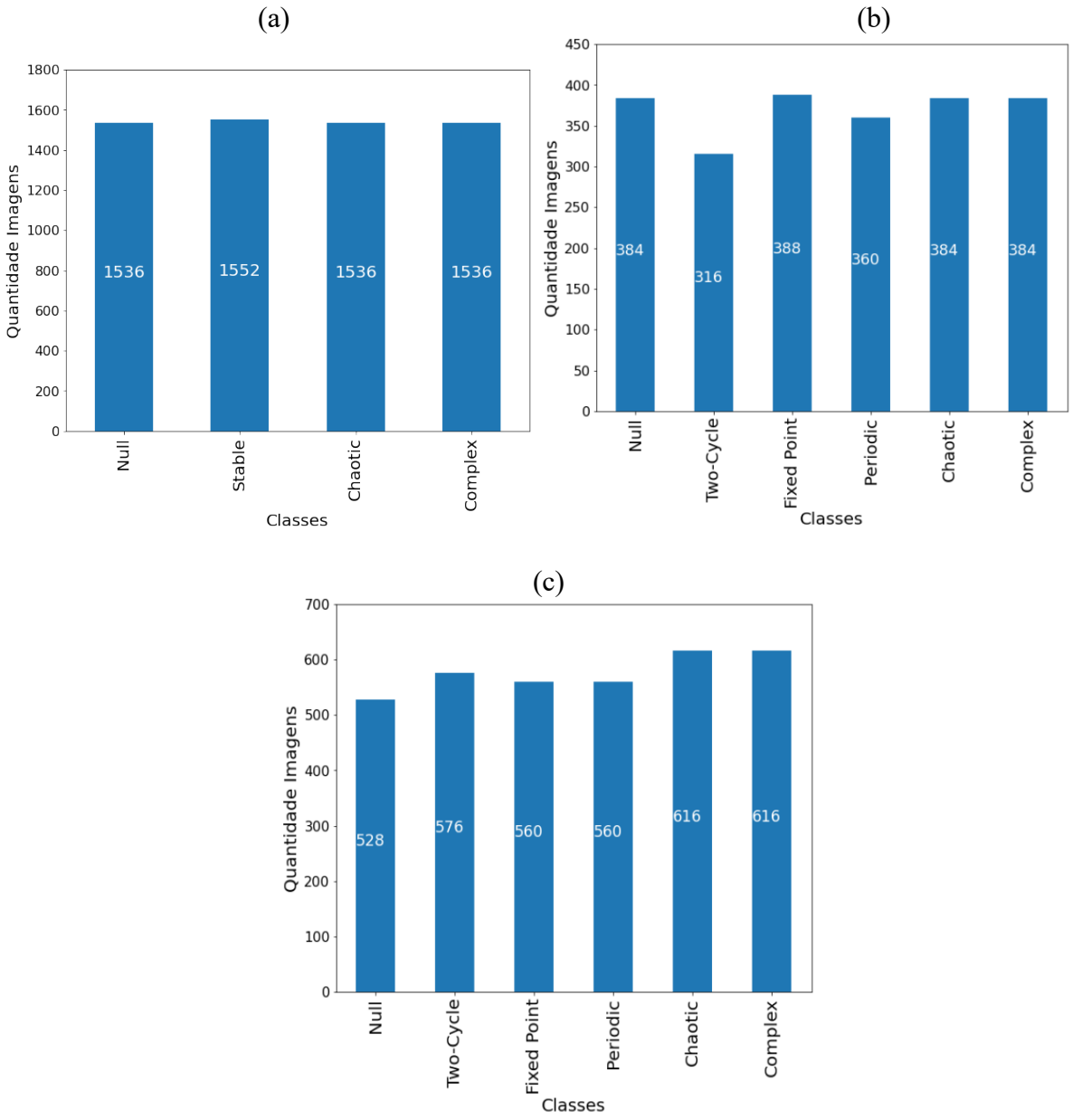
Tabela 3 – Divisão das bases de dados (Treinamento, Validação e Teste).

Base de dados	**Treinamento**	**Validação**	**Teste**	**Total**
ACE Wolfram (4 classes)	4436	492	1232	6160
ACE Li-Packard (4 classes)	797	89	3546	4432
ACT Li-Packard (6 classes)	2488	276	692	3456

Fonte: próprio autor (2022).

Embora a relação quantidade de regras por classe não seja balanceada segundo as classificações de Wolfram e Li-Packard, conforme visto na Seção 2.3, gerou-se imagens com condições iniciais aleatórias suficientes para garantir uma base de dados mais balanceada, conforme é possível visualizar na Figura 16. Cada imagem monocromática possui dimensão de 128x128 *pixels* correspondentes ao total de transições das evoluções temporais e à vizinhança global do reticulado do autômato, respectivamente.

Figura 16 - Distribuição da base de imagens: a) classificação de ACE – Wolfram; b) classificação de ACE – Li-Packard; c) classificação de ACT – Li-Packard.

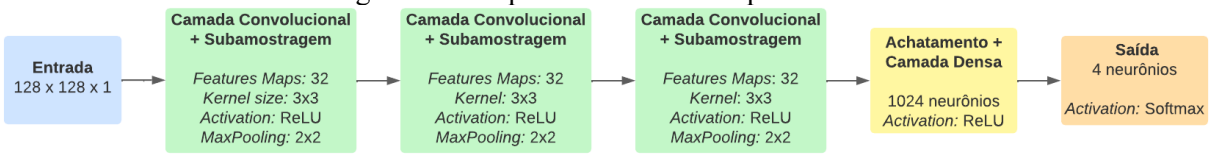


Fonte: próprio autor (2022).

3.2 Arquitetura da rede neural convolucional proposta

Analisando o estado da arte, optou-se por adaptar a arquitetura simplificada utilizada para Reconhecimento da COVID-19 em Imagens de Raios-X apresentado por Barbosa, Coelho e Baffa (2020) para realizar a atividade proposta neste trabalho. A rede é composta porseis camadas, sendo uma de entrada, três camadas ocultas, uma camada densa e uma camada de saída. A camada de entrada é responsável por receber o tensor de entrada e transmitir os dados para as camadas ocultas. Nas três camadas convolucionais seguintes, foram extraídos 32 *Feature Maps* (Filtros) com *Kernel* de tamanho 3x3 e foi utilizada a função de ativação *Recifier Linear Unit* (ReLU). Após cada uma dessas camadas, realizou-se a operação de *Max Pooling* com tamanho 2x2 a fim de diminuir a dimensionalidade dos filtros. A seguir, foi aplicada a operação de achatamento (*Flatten*), que tem como finalidade alterar o formato dos filtros para um vetor unidimensional e, na sequência, uma camada *Fully Connected* (Densa) com 1024 neurônios com função de ativação ReLU, com intuito de detectar padrões no vetor descritor construído pela convolução. Por fim, a camada de saída foi composta por quatro neurônios, responsável pela classificação do autômato celular elementar, utilizando a função de ativação *softmax*.

Figura 17 - Arquitetura de CNN implementada



Fonte: Próprio autor (2022).

3.3 Parâmetros

A seguir são descritos os parâmetros fixos, ou seja, que foram definidos para a implementação de todos os modelos da *CNN* proposta e a Tabela 4 exibe os parâmetros configuráveis utilizados em cada modelo.

Parâmetros fixos:

- Entrada de imagens com resolução de 128x128 pixels e 3 canais, ou seja, *input_shape* igual a (128, 128, 1)
- Saída com a classificação dentre as 4 ou 6 classes equivalentes, dependendo da etapa

- Utilização de 80% dos dados para treinamento e validação da rede neural e 20% para o teste do modelo criado
- Utilização da função de ativação ReLU, nas camadas convolucionais, uma vez que ela não ativa todos os neurônios das camadas ao mesmo tempo
- Utilização de 32 filtros com tamanho 3x3 nas camadas ocultas
    - Compilação dos modelos com a função de custo "*categorical_crossentropy*"
    - Treinamento e validação por 20 épocas usando *batch* igual a 32

Tabela 4 – Parâmetros configuráveis de cada modelo.

Parâmetro	**Modelo Alpha**	**Modelo Beta**	**Modelo Gamma**	**Modelo Yotta**
Learning Rate	0.1	0.01	0.1	0.01
Optimizer	SGD	SGD	ADAM	ADAM
Momentum	padrão	0.9	padrão	padrão

Fonte: Próprio autor (2022).

Além dos parâmetros fixos mencionados anteriormente, foram definidos dois *callbacks*: *ModelCheckpoint* e *EarlyStopping*. O primeiro *callback* salva a última época em que o modelo obteve o menor valor da taxa de errodurante a etapa de validação, enquanto o segundo para antecipadamente o treinamento quando a métrica monitorada (*monitor*) não obteve melhores resultados após *n* épocas (parâmetro*patience*), visando evitar iterações desnecessárias e *overfitting*. Nesse último, foram adotados os parâmetros*monitor* igual a "*val_loss*" e *patience* igual a 5.

3.4 Métricas para avaliação de desempenho

Com o objetivo de gerar maior segurança nos resultados obtidos para avaliação dos modelos de aprendizagem, foi realizada a média aritmética simples das métricas abaixo:

- Acurácia (*Accuracy*), identificando a média global de acertos do modelo;
- Precisão (*Precision*), para obter a informação da exatidão da classificação;
- Revocação (*Recall*), contendo a razão entre o total de predições corretas e o total de exemplos que são de fato da classe em análise;
- Área sob a curva (AUC);
- Pontuação F1 (*Score F1*) que a média harmônica entre a precisão e recall.

3.5 Tecnologias utilizadas

Neste projeto foram utilizadas a linguagem de programação Python v3.8; as bibliotecas *NumPy*, *Pandas*, *MatplotLib* e *Scikit-Learn* para auxiliarem na execução dos algoritmos que geraram as bases de dados, construção dos gráficos e divisão da base de dados com validação cruzada; *TensorFlow* e *Keras* como as principais bibliotecas que fornecem blocos de construção para o desenvolvimento das redes neurais e funções dedicadas ao processamento de imagens. O treinamento das arquiteturas propostas foi executado em uma máquina local DELL Quad-core 2.8 GHz, com 16 GB de RAM, placa de vídeo Nvidia MX150 2GB, em plataforma operacional Windows 10 e ambiente de Anaconda com.

Após a definição do modelo, foi utilizado a técnica de Transferência de Aprendizado, que transfere os parâmetros de uma rede neural treinada com um conjunto de dados para ser aplicada em outro contexto utilizando-se novo conjunto de dados. Desta forma, pode-se treinar somente as últimas camadas para se adaptar ao problema proposto. Isto não só aumenta a velocidade de treino consideravelmente, como também requer muito menos dados de treinamento (GÉRON, 2017).

3.6 Configurações e treinamentos

Para se obter resultados melhores e avaliar a capacidade de generalização dos modelos, foi utilizado o método de Validação Cruzada, especificamente o *K-Fold Cross Validation*, aplicando-se a biblioteca *Scikit-Learn* com *k* (número de *folds*) igual a 5.

Na primeira etapa, foi realizada a avaliação baseada na análise quantitativa dos quatro modelos propostos (*Alpha, Beta, Gamma, Yotta*) treinados e avaliados com o primeiro conjunto de autômatos celulares elementares com a classificação de Wolfram, considerando a média dos resultados de acurácia, taxa "positiva", sensibilidade (revocação), precisão, pontuação F1 e a curva ROC, obtidos em todos os *Folds*. A proporção da separação das bases de dados utilizada foi de 70% para treinamento, 10% para validação e 20% para teste.

Posteriormente com a definição do melhor modelo, utilizou-o para avaliar o desempenho da transferência de aprendizado na tarefa de classificação nas bases de autômatos celulares elementares e totalistas, ambas classificadas a Li-Packard, considerando as mesmas métricas anteriormente utilizadas.

4 ANÁLISE DOS RESULTADOS

Nesta seção são apresentados os resultados obtidos na avaliação dos modelos propostos e do desempenho da transferência de aprendizado.

Ao analisar os resultados obtidos, alguns aspectos foram observados: um deles foi a influência do ajuste fino de alguns hiperparâmetros nos resultados obtidos pelos classificadores, o que enfatiza a importância de se conhecer os parâmetros que devem ser configurados em detrimento ao uso dos otimizadores com valores padrão. Outro ponto evidenciado nestes resultados, é que não foi identificado um modelo unânime, ou seja, que obteve os melhores resultados em todas as métricas.

O comparativo dos resultados das métricas obtidas durante a primeira etapa do projeto pode ser observado na Tabela 5. Para cada um dos modelos propostos utilizados nos testes sobre a base de dados com autômatos celulares elementares de Wolfram, os resultados exibidos são as médias aritméticas da validação cruzada após cinco execuções (*Fold* = 5). Todos os valores foram truncados na segunda casa decimal.

Tabela 5 – Comparação dos resultados dos modelos.

Modelo	**Acurácia (%)**	**Área sob a Curva (AUC) (%)**	**Precisão (%)**	**Revocação (%)**	**Pontuação F1 (%)**	**Média (%)**
Alpha	96,69	99,55	96,77	96,65	96,71	97,27
Beta	**97,00**	**99,61**	97,04	**96,96**	**97,00**	**97,52**
Gamma	96,65	99,54	**97,16**	96,47	96,81	97,33
Delta	96,69	99,46	96,96	96,45	96,70	97,25

Fonte: Próprio autor (2022).

Os classificadores desempenham uma tarefa de classificação categórica baseado no vetor de probabilidades dado pela função de ativação *Softmax* presente na última camada. Assim sendo, é possível observar que a acurácia média entre os modelos se define em torno 97%. Com isso, é possível observar que o modelo **Beta** teve o melhor desempenho médio das métricas avaliadas, mesmo não sendo o modelo com a melhor precisão, sendo este o modelo *Gamma*, que obteve o segundo melhor desempenho. Considerando este contexto aplicado, o otimizador SGD com *momentum* de 0,9 e *learning rate* de 0,01 superou o otimizador ADAM.

Na segunda etapa, foi avaliado o desempenho da Transferência de Aprendizado comparando o modelo *Beta* e os dois modelos estado-da-arte, *Inception* e *Resnet*. Os três modelos foram treinados e testados utilizando-se a segunda a terceira base de imagens, que contêm imagens de autômatos celulares elementares e totalistas classificados a Li-Packard, respectivamente.

Foram criados três cenários para execução do treinamento e teste para cada modelo:

1. modelo com os valores dos parâmetros padrão
2. STF *(Shallow Fine Tuning)* com *learning rate* de 0,001 e *momentum* 0,9
3. DFT *(Deeply Fine Tuning)* com *learning rate* de 0,00001 e *momentum* de 0,99

As Tabelas 6, 7 e 8 exibem os resultados das métricas obtidas após os testes nas bases de imagens ACE Li-Packard para os modelos *Inception*, *ResNet* e *Beta*. Todos os valores foram truncados na segunda casa decimal.

Tabela 6 – Resultados *Transfer Learning – Inception* – Base ACE Li-Packard

Modelo	**Acurácia (%)**	**Área sob a Curva (AUC) (%)**	**Precisão (%)**	**Revocação (%)**	**Pontuação F1 (%)**	**Média (%)**
Padrão	93,47	99,19	93,86	93,02	93,44	94,60
SFT	**94,59**	**99,40**	**95,21**	**93,92**	**94,56**	**95,54**
DFT	91,22	99,18	93,29	90,77	92,01	93,29

Fonte: Próprio autor (2022)

Tabela 7 – Resultados *Transfer Learning – ResNet* – Base ACE Li-Packard

Modelo	**Acurácia (%)**	**Área sob a Curva (AUC) (%)**	**Precisão (%)**	**Revocação (%)**	**Pontuação F1 (%)**	**Média (%)**
Padrão	95,50	**99,54**	95,68	94,82	94,82	96,07
SFT	**95,72**	99,34	**96,15**	**95,50**	**95,82**	**96,51**
DFT	92,12	99,00	93,58	91,89	92,73	93,86

Fonte: Próprio autor (2022)

Tabela 8 – Resultados *Transfer Learning – Beta* – Base ACE Li-Packard

Modelo	**Acurácia (%)**	**Área sob a Curva (AUC) (%)**	**Precisão (%)**	**Revocação (%)**	**Pontuação F1 (%)**	**Média (%)**
Padrão	91,89	99,40	93,52	90,99	92,24	93,61
SFT	94,59	**99,62**	96,05	93,02	94,51	95,56
DFT	**95,05**	99,59	**96,50**	**93,24**	**94,85**	**95,82**

Fonte: Próprio autor (2022)

Assim, é possível perceber a influência do ajuste fino de forma positiva e negativa: o uso da técnica *STF* melhorou os resultados dos três modelos comparados à técnica padrão de *transfer learning*. Porém, ao utilizar o *DFT*, os modelos *Inception* e *ResNet50* tiveram um desempenho médio menor do que o padrão, enquanto o modelo *Beta* atingiu seu melhor desempenho médio. Ao final dessa parcial, o modelo *ResNet* obteve o melhor desempenho médio com a técnica de *Shallow Fine Tuning* e ainda obteve os melhores resultados de acurácia.

Por fim, é possível visibilizar os resultados das métricas obtidas após os testes nas bases de imagens ACT Li-Packard nas Tabelas 9, 10 e 11. Todos os valores foram truncados na segunda casa decimal.

Tabela 9 – Resultados *Transfer Learning – Inception* – Base ACT Li-Packard

Modelo	**Acurácia (%)**	**Área sob a Curva (AUC) (%)**	**Precisão (%)**	**Revocação (%)**	**Pontuação F1 (%)**	**Média (%)**
Padrão	89,45	98,10	91,47	88,29	89,95	91,43
SFT	**91,91**	**98,96**	**93,03**	**90,61**	**91,80**	**93,26**
DFT	80,35	97,04	88,05	69,22	77,51	82,43

Fonte: Próprio autor (2022)

Tabela 10 – Resultados *Transfer Learning – ResNet* – Base ACT Li-Packard

Modelo	**Acurácia (%)**	**Área sob a Curva (AUC) (%)**	**Precisão (%)**	**Revocação (%)**	**Pontuação F1 (%)**	**Média (%)**
Padrão	85,12	97,01	85,95	83,96	84,94	82,64
SFT	**92,05**	98,64	**93,14**	**90,32**	**91,71**	**93,17**
DFT	91,04	**99,16**	92,58	88,29	90,38	92,29

Fonte: Próprio autor (2022)

Tabela 11 – Resultados *Transfer Learning* – Beta – Base ACT Li-Packard

Modelo	**Acurácia (%)**	**Área sob a Curva (AUC) (%)**	**Precisão (%)**	**Revocação (%)**	**Pontuação F1 (%)**	**Média (%)**
Padrão	69,22	90,31	69,92	67,20	68,53	73,04
SFT	**87,72**	**98,76**	**87,79**	**87,28**	**87,54**	**89,82**
DFT	74,13	94,23	81,26	55,78	66,15	74,31

Fonte: Próprio autor (2022)

Na última fase dos testes, é possível notar também a influência do ajuste fino como nos resultados anteriores. Porém, desta vez, ao utilizar o DFT, todos os modelos tiveram um desempenho médio menor do que o padrão. Ao comparar os resultados, o modelo *Inception* obteve o melhor desempenho médio com a técnica de *Shallow Fine Tuning* enquanto *ResNet50* obteve os melhores resultados de acurácia.

Foi possível identificar que o modelo *Beta* obteve melhor desempenho na base de dados com ACEs a Li-Packard enquanto, quando treinada na última base de ACT, teve o menor desempenho dentre os três. Assim, observa-se a diferença de desempenho ao utilizar redes que já foram muito bem treinadas anteriormente, pois essas redes atingiram os melhores resultados, mesmo sendo treinadas em uma base de dados com menos imagens.

Sendo assim, pode-se afirmar que modelos especializados e simplificados podem alcançar ótimos resultados quando utilizados na tarefa de classificação de autômatos celulares.

5 CONSIDERAÇÕES GINAIS E TRABALHOS FUTUROS

Neste trabalho, foi proposta a análise do desempenho da transferência de aprendizado para a classificação de autômatos celulares elementares e totalistas. Comparou-se a utilização de técnicas de ajuste fino em um modelo proposto (*Beta*) e nas redes *Inception* e *ResNet50*. Inicialmente treinou-se o modelo *Beta* com um banco de imagens de 6160 imagens de ACE classificados por Wolfram, para então usar a transferência de aprendizado para execução dos testes nas duas bases seguintes, uma contendo imagens de ACE e a outra com imagens de ACT divididas entre as quatro e as seis classes de Li-Packard, respectivamente. Ao final dos testes, percebeu-se que o modelo proposto teve o melhor desempenho médio em na base de dados com 4432 imagens de ACEs, porém o menor dentre os três na base com 3456 imagens de ACT. Isso evidenciou a necessidade da maior quantidade de treinamentos em bases diferentes para que esse desempenho melhore.

O modelo *Beta* foi capaz de classificar os autômatos celulares elementares corretamente com uma média de 94,59%, e os totalistas com 87,72%, garantindo-lhe bons resultados para essa tarefa, mesmo sendo um modelo simplificado.

Baseado nisso, sugere-se que esse experimento possa ser feito alterando parâmetros na estrutura da rede proposta, incluir outras redes estado-da-arte, inclusive de outras arquiteturas.

REFERÊNCIAS

ADHIANTO, L., et al. FPGA-Accelerated Deep Convolutional Neural Networks for High Throughput and Energy Efficiency. **Concurrency Computation Practice and Experience**, v. 22, n. 6, p. 685–701, 2016.

AGGARWAL, C. C. **Data Classification: Algorithms and Applications**. 1a. ed., Chapman & Hall, 2020.

AGUIAR, I. M. M. **Sobre a classifcação dos sutômatos celulares elementares finitos**. 2014, 208f. Dissertação (Mestrado) - Universidade do Minho, 2014.

BAR-YAM, Y. **Dynamics of Complex Systems**. 1a ed., Addison-Wesley, 1997

BEZERRA, E.C. et al. Comparação entre modelos estatísticos e redes neurais usando persistência como referência para a previsão da velocidade do vento. **Anais X Simpósio Brasileiro de Automação Inteligente**, v.10, p.369-374 Minas Gerais, 2011.

BISHOP, C. M. **Neural Networks for Pattern Recognition**. New York, NY, USA: Oxford, 1996.

CAMOLESI, M.A. **Treinamento de autômatos celulares para reconhecimento de padões em imagens de textura.** 2020, 19f. Monografia (Graduação) – Universidade Estadual de Campinas, Campinas 2020.

CHORLEY, R. J.; HAGGETT, P. (Ed.). **Models in Geography**. Londres: Methuen e Co., 1967. p. 19-41.

COMELLI, T., PINEL, F. e BOUVRY, P. Comparing Elementary Cellular Automata Classifications with a Convolutional Neural Network. **Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021)**, v.2, p. 467-474, 2021

DUDA, R. O.; HART, P. E. **Pattern classification and scene analysis**. 1a ed, Wiley-Blackwell, 1973.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**. 1a ed., O'ReillyAddison-Wesley, 1994.

GHOSH, J. B.; ADSULE, R. One dimensional cellular automata - The totalistic approach. **Azim Premji University At Right Angles**, v.7, n.2, p.105-116, julho, 2018

GLERIA, I. et al. Sistemas complexos, criticalidade e leis de potência. **Revista Brasileira de Ensino de Física**, v. 26, n. 2, p. 99-108, outubro, 2004

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning.**1a ed. MIT Press, 2016.

GREMONINI, L.; VICENTINI, E. Autômatos celulares: revisão bibliográfica e exemplos de implementações**. Revista eletrônica lato sensu** – UNICENTRO, 6 ed., 2008.

HASSAN, Y. F. Classification Based on Deep Neural Cellular Automata Model. **World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering**, v.13, n.7, 2019.

HASTIE, T. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**, 2aed., Springer, 2009

HAYKIN, S. **Redes Neurais- Princípios e Práticas.**2ª ed. Bookman, São Paulo, 2001.

HE K. et al. Deep Residual Learning for Image Recognition. **Computer Vision and Pattern Recognition**, 2015.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: The Shared Views of Four Research Groups. **IEEE Signal Processing Magazine**, v. 29, n. 6, p. 82-97, Novembro, 2012

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the national academy of sciences,**v.79, n.8, abril, 1982.

HUANG, G. et al. Densely Connected Convolutional Networks**. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR**), p.2261–2269, 2017

IZADYYAZDANABADI, M. et al. Convolutionalneural networks: Ensemble modeling, fine-tuning and unsupervised semantic localization for neurosurgical CLE images.**Journal of Visual Communication and ImageRepresentation,**v.54, p.10–20, julho, 2018.

JOHNSON, N. et al.Invited review: Machine Learning for Materials Developments in Metals Additive Manufacturing.**Additive Manufacturing**, v.36, dezembro, 2020

KRIZHEVSKY, A.; SUTSKEVER, I.; HITON, G. E. ImageNet Classification with Deep Convolution Neural Networks. **Proceedings of Neural Information Processing Systems.** p.1097-1105, 2012.

KUNKLE, D. R. **Automatic Classification of One-Dimensional Cellular Automata,** 2003, 187f, Dissertação (Mestrado) - Rochester Institute of Technology, 2003

LEWIN, R. **Complexity, Life at the Edger of Chaos.**1a ed., Chicago University Press, 1993.

LORENZ, E. Deterministic nonperiodic flow. **Journal of Atmospheric Science**, v. 20, p.130-141, 1963

MAJI, P. et al. Theory and Application of Cellular Automata For Pattern Classification. **Fundamenta Informaticae,** v. 58, n. 3-4, p. 321-354, 2003

MARQUES, V. G. O. **Avaliação do desempenho das redes neurais convolucionais na detecção de ovos de esquistossomose**, 2017, 49f. Trabalho de Conclusão de Curso - Universidade Federal de Pernambuco, Recife, 2017.

MASSABKI, J. A. R. et al. Modelagem dos padrões da expansão urbana da Região Metropolitana de São Paulo baseada em Autômatos Celulares. **Revista Brasileira deGestão Urbana,** v.9, outubro, 2017.

MESSIAS, B. C.**Rede neural convolucional com dois canais para classificação automática de arritmias em sinais de ECG**, 2022, 67f, Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Florianópolis, 2022.

MICHELL, M. **Complexity: A guided tour**, Oxford University Press, 2011

MITCHELL, M. Computation cellular automata: a selected review. **T. Gramss (ed.),** Nonstandard Computation: Weinheim. VCH Verlagsgesellschaft, 1996

NEUMANN, J. V.; BURCKS, A. W. **Theory of self-reproducing automata**. 1a ed., Champaign, USA: University of Illinois Press, 1966.

NOGUEIRA, M. A. **Classificação automática do comportamento dinâmico de autômatos celulares binários unidimensionais,** 2019, 106f. Tese (Doutorado) – Universidade Presbiteriana Mackenzie, São Paulo, 2019

NOGUEIRA, T. C. **Modelo baseado em redes neurais profundas com unidades recorrentes bloqueadas para legendagem de imagens por referência.** 2020, 122f, Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Goiás, Escola de Engenharia elétrica, mecânica e de computação, Goiânia, 2020.

OLIVEIRA, P. M. C. Sistemas complexos. **Ciência hoje**, v.16, n.92, p.15-22, junho, 1993.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review***,* v.65, n.6, p.386–408, 1958.

SANTOS, C.; et al. Uso de Transfer Learning para o reconhecimento de gestos dinâmicos. **Anais do Congresso Brasileiro de Automática**. v.1, n,1, 2018.

SILVA, R. E. V. **Um estudo comparativo entre redes neurais convolucionais para a classificação de imagens.** 2018, 52f, Trabalho de Conclusão de Curso (graduação) - Universidade Federal do Ceará, Quixadá, 2018.

SILVERMAN, E. Convolutional Neural Networks for Cellular Automata Classification. **The 2019 Conference on Artificial Life,** p. 280-281, janeiro, 2019.

SIMÕES, R. E. O.; **Autômatos celulares e o problema da classificação de densidade: o modelo Gács-Kurdyumov-Levin de quatro estados**. 80f, Dissertaçao (Mestrado) - Universidade São Paulo, São Paulo, 2016.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **Computer Vision and Pattern Recognition**. v.1, Setembro, 2014.

SOARES, P. L. B.; SILVA, J. P. Aplicação de Redes Neurais Artificiais em Conjunto com o Método Vetorial da Propagação de Feixes na Análise de um Acoplador Direcional Baseado em Fibra Ótica. **Revista Brasileira de Computação Aplicada,** v.3, p.58–72, 2011

SZEGEDY, C. et al. A. Going deeper with convolutions. **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p.1–9, 2015

TAJBAKHSH, N. et al. Convolutional neural networks for medicalimage analysis: Full training or fine tuning? **IEEE Transactions on Medical Imaging**, v.35, n.5, p.1299–1312, 2016.

VOGADO, L. H. S.et al. Rede Neural Convolucional para o Diagnóstico de Leucemia. **Anais Simpósio brasileiro de computação aplicada à saúde (SBCAS),**v.19, p. 46-57, 2019.

WERBOS, P. **Beyond regression:new tools for prediction and analysis in the behavioral Science.** 1974, 453f, Tese (Doutorado). Harvard University**,**Massachusetts, 1974.

WOLFRAM, S., **A new kind of science.**1a ed.,Wolfram Media, Champaign, IL, USA, 2002.

WOLFRAM, S., **Cellular Automata and Complexity: Collected Papers.**

WOLFRAM, S., Statistical mechanics of cellular automata. **Review of Modern Physics,** v.55, p. 601-644, julho, 1983.

WOLFRAM, S., Universality and complexity in cellular automata, **Physica 10D**, p.1-35, 1984.

YOSINSKI, J. et al. How transferable are features in deep neural networks? **Advances in neural information processing systems.** p.3320–3328, 2014.

ZHANG, J et al. Classification of Microscopic Laser Engraving Surface Defect Images Based on Transfer Learning Method. **Electronics**. v.10, n.16, 1993.

ANEXO A – Classificação Wolfram de Autômatos Celulares Elementares equivalentes

Grupo de regras	Classe
0, 255	I
1, 127	II
2, 16, 191, 247	II
3, 17, 63, 119	II
4, 223	II
5, 95	II
6, 20, 159, 215	II
7, 21, 31, 87	II
8, 64, 239, 253	I
9, 65, 111, 125	II
10, 80, 175, 245	II
11, 47, 81 ,117	II
12, 68, 207, 221	II
13, 69, 79, 93	II
14, 84, 143, 213	II
15, 85	II
18, 183	III
19, 55	II
22, 151	III
23	II
24, 66, 189, 231	II
25, 61, 67, 103	II
26, 82, 167, 181	II
27, 39, 53, 83	II
28, 70, 157, 199	II
29, 71	II
30, 86, 135, 149	III
32, 251	I
33, 123	II
34, 48,187,243	II
35, 49, 59, 115	II
36, 219	II
37, 91	II
38, 52, 155, 211	II
40, 96, 235, 249	I
41, 97, 107, 121	II
42, 112, 171, 241	II
43, 113	II
44, 100, 203, 217	II
45, 75, 89, 101	III
46, 116, 139, 209	II
50, 179	II
51	II
54, 147	IV
56, 98, 185, 227	II
57, 99	II
58, 114, 163, 177	II
60, 102, 153, 195	III
62, 118, 131, 145	II
72, 237	II
73, 109	III
74, 88, 173, 229	II
76, 205	II
77	II
78, 92, 141, 197	II
90, 165	III
94, 133	II
104, 233	II
105	III
106, 120,169,225	III
108, 201	II
110, 124,137,193	IV
122, 161	III
126, 129	III
128, 254	I
130, 144,190,246	II
132, 222	II
134, 148,158,214	II
136, 192,238,252	I
138, 174,208,244	II
140, 196,206,220	II
142, 212	II
146, 182	III
150	III
152, 188,194,230	II
154, 166,180,210	II
156, 198	II
160, 250	I
162, 176,186,242	II
164, 218	II
168, 224,234,248	I
170, 240	II
172, 202,216,228	II
178	II
184, 226	II
200, 236	II
204	II
232	II

ANEXO B – Classificação Li-Packard de Autômatos Celulares Elementares equivalentes

Grupo de regras	Classe
0, 255	Null
1, 127	Two-Cycle
2, 16, 191, 247	FixedPoint
3, 17, 63, 119	Two-Cycle
4, 223	FixedPoint
5, 95	Two-Cycle
6, 20, 159, 215	Two-Cycle
7, 21, 31, 87	Two-Cycle
8, 64, 239, 253	Null
9, 65, 111, 125	Two-Cycle
10, 80, 175, 245	FixedPoint
11, 47, 81 ,117	Two-Cycle
12, 68, 207, 221	FixedPoint
13, 69, 79, 93	FixedPoint
14, 84, 143, 213	Two-Cycle
15, 85	Two-Cycle
18, 183	Chaotic
19, 55	Two-Cycle
22, 151	Chaotic
23	Two-Cycle
24, 66, 189, 231	FixedPoint
25, 61, 67, 103	Two-Cycle
26, 82, 167, 181	Periodic
27, 39, 53, 83	Two-Cycle
28, 70, 157, 199	Two-Cycle
29, 71	Two-Cycle
30, 86, 135, 149	Chaotic
32, 251	Null
33, 123	Two-Cycle
34, 48, 187, 243	FixedPoint

Grupo de regras	Classe
35, 49, 59, 115	Two-Cycle
36, 219	FixedPoint
37, 91	Two-Cycle
38, 52, 155, 211	Two-Cycle
40, 96, 235, 249	Null
41, 97, 107, 121	Periodic
42, 112, 171, 241	FixedPoint
43, 113	Two-Cycle
44, 100, 203, 217	FixedPoint
45, 75, 89, 101	Chaotic
46, 116, 139, 209	FixedPoint
50, 179	Two-Cycle
51	Two-Cycle
54, 147	Complex
56, 98, 185, 227	FixedPoint
57, 99	FixedPoint
58, 114, 163, 177	FixedPoint
60, 102, 153, 195	Chaotic
62, 118, 131, 145	Periodic
72, 237	FixedPoint
73, 109	Chaotic
74, 88, 173, 229	Two-Cycle
76, 205	FixedPoint
77	FixedPoint
78, 92, 141, 197	FixedPoint
90, 165	Chaotic
94, 133	Periodic
104, 233	FixedPoint
105	Chaotic
106, 120,169,225	Chaotic

Grupo de regras	Classe
108, 201	Two-Cycle
110, 124, 137, 193	Complex
122, 161	Chaotic
126, 129	Chaotic
128, 254	Null
130, 144, 190, 246	FixedPoint
132, 222	FixedPoint
134, 148, 158, 214	Two-Cycle
136, 192, 238, 252	Null
138, 174, 208, 244	FixedPoint
140, 196, 206, 220	FixedPoint
142, 212	Two-Cycle
146, 182	Chaotic
150	Chaotic
152, 188, 194, 230	FixedPoint
154, 166, 180, 210	Periodic
156, 198	Two-Cycle
160, 250	Null
162, 176, 186, 242	FixedPoint
164, 218	FixedPoint
168, 224, 234, 248	Null
170, 240	FixedPoint
172, 202, 216, 228	FixedPoint
178	Two-Cycle
184, 226	FixedPoint
200, 236	FixedPoint
204	FixedPoint
232	FixedPoint

ANEXO C – Classificação Li-Packard de Autômatos Celulares

Totalistas equivalentes

Grupo de regras	Classe
0, 255	Null
1, 127	Two-Cycle
2, 191	Periodic
3, 63	Two-Cycle
4, 223	Null
5, 95	Periodic
6, 159	Periodic
7, 31	Two-Cycle
8, 239	Fixed Point
9, 111	Chaotic
10, 175	Chaotic
11, 47	Two-Cycle
12, 207	Chaotic
13, 79	Periodic
14, 143	Chaotic
15	Two-Cycle
16, 247	Fixed Point
17, 119	Chaotic
18, 183	Chaotic
19, 55	Periodic
20, 215	Chaotic
21, 87	Chaotic
22, 151	Chaotic
23	Two-Cycle
24, 231	Periodic
25, 103	Chaotic
26, 167	Chaotic
27, 39	Two-Cycle
28, 199	Chaotic
29, 71	Periodic
30, 135	Chaotic
32, 251	Null
33, 123	Chaotic
34, 187	Chaotic
35, 59	Two-Cycle
36, 219	Null
37, 91	Chaotic
38, 155	Chaotic
40, 235	Fixed Point
41, 107	Chaotic
42, 171	Chaotic
43	Chaotic
44, 203	Chaotic
45, 75	Chaotic
46, 139	Chaotic
48, 243	Fixed Point

Grupo de regras	Classe
49, 115	Chaotic
50, 179	Chaotic
51	Chaotic
52, 211	Chaotic
53, 83	Chaotic
54, 147	Chaotic
56, 227	Chaotic
57, 99	Chaotic
58, 163	Chaotic
60, 195	Chaotic
61, 67	Two-Cycle
62, 131	Periodic
64, 253	Null
65, 125	Chaotic
66, 189	Chaotic
68, 221	Null
69, 93	Chaotic
70, 157	Chaotic
72, 237	Null
73, 109	Chaotic
74, 173	Chaotic
76, 205	Chaotic
77	Chaotic
78, 141	Chaotic
80, 245	Null
81, 117	Chaotic
82, 181	Chaotic
84, 213	Chaotic
85	Chaotic
86, 149	Chaotic
88, 229	Complex
89, 101	Chaotic
90,165	Chaotic
92, 197	Chaotic
94, 133	Chaotic
96, 249	Null
97, 121	Chaotic
98, 185	Chaotic
100, 217	Null
102, 153	Chaotic
104, 233	Fixed Point
105	Chaotic
106, 169	Chaotic
108, 201	Chaotic
110, 137	Chaotic
112, 241	Fixed Point

Grupo de regras	Classe
113	Chaotic
114, 177	Chaotic
116, 209	Chaotic
118, 145	Chaotic
120, 225	Chaotic
122, 161	Chaotic
124, 193	Null
126, 129	Periodic
128, 254	Null
130, 190	Chaotic
132, 222	Null
134, 158	Chaotic
136, 238	Null
138, 174	Chaotic
140, 206	Chaotic
142	Chaotic
144, 246	Null
146, 182	Chaotic
148, 214	Chaotic
150	Chaotic
152, 230	Periodic
154, 166	Chaotic
156, 198	Chaotic
160, 250	Null
162, 186	Chaotic
164, 218	Null
168, 234	Fixed
170	Chaotic
172, 202	Chaotic
176, 242	Fixed
178	Chaotic
180, 210	Chaotic
184, 226	Chaotic
188, 194	Chaotic
192, 252	Null
196, 220	Null
200, 236	Null
204	Chaotic
208, 244	Null
212	Chaotic
216, 228	Fixed
224, 248	Null
232	Fixed
240	Fixed