UNIVERSIDADE FEDERAL DE SÃO CARLOS

Departamento de Física

Carlos Eduardo de Souza

Classificação não supervisionada para Autómato Celular Elementar do Wolfram

Trabalho Final de Curso

São Carlos, SP
29 de setembro de 2022

Carlos Eduardo de Souza

Classificação não supervisionada para Autómato Celular Elementar do Wolfram

Trabalho Final de Curso submetido à Universidade Federal de São Carlos, como requisito necessário para obtenção do grau de Bacharel em Engenharia Física

São Carlos - SP
29 de setembro de 2022

do Souza, Carlos Eduardo
Classificação não supervisionada para Autómato Celular Elementar do Wolfram /
Carlos Eduardo de Souza – 2022.
87f.

TFC (Graduação) - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador: Fábio Aparecido Ferri
Co-orientador: Odemir Martinez Bruno
Banca Examinadora: Pedro Augusto Franco Pinheiro Moreira, Matheus Paes Lima
Bibliografia

1. Autómato Celular. 2. Wolfram. 3. Aprendizado de Máquina. I. de Souza, Carlos Eduardo. II. Título.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CARLOS EDUARDO DE SOUZA

Esta Monografia foi julgada adequada para a obtenção do título de Bacharel em Engenharia Física, sendo aprovada em sua forma final pela banca examinadora:

Orientador(a): Prof. Dr. Fábio Aparecido Ferri
Universidade Federal de São Carlos - UFSCar

Co-orientador(a): Prof. Dr. Odemir Martinez Bruno
Universidade de São Paulo - USP

Prof. Dr. Pedro Augusto Franco Pinheiro Moreira
Universidade Federal de São Carlos - UFSCar

Prof. Dr. Matheus Paes Lima
Universidade Federal de São Carlos - UFSCar

São Carlos - SP
29 de setembro de 2022

Conteúdo

Resumo

Os Autómatos Celulares estão ganhando espaço em muitas áreas de pesquisas como biologia, física e química. Porém conforme são adicionados mais parâmetros, começa a ficar impossível encontrar características que possam ser interessantes para determinada aplicação. Esse trabalho tem como objetivo por métodos de aprendizado de máquina não supervisionado como descritores de textura e autoencoder convolucional determinar quais regras do Autómato Celular Elementar pertencem às classes determinadas por Wolfram. Para fazer o aprendizado não supervisionado, foi gerado histogramas com o Local *Binary Pattern Variance* (LBPV) e o Autoencoder Convolucional, então utilizado os histogramas no modelo *K-Means* analisar se o modelo é capaz de separar os dados em quatro cluster de acordo com a classificação proposta por Woflram. No final observou-se que ambos os métodos não foram capaz de realizar tal classificação, sendo que o LBPV apresentou resultados mais próximos.

Palavras-chave: Autómato Celular ,Wolfram, Aprendizado de Máquina

1 Introdução

Desde que a evolução temporal dos Autómatos Celulares (CAs) podem ser visualizadas facilmente através dos computadores modernos, os intrigantes padrões espaço-temporais que eles evoluem têm estimulado a pesquisa em muitas áreas da ciência, algumas delas apenas pouco relacionadas à ciência da computação. A título de exemplo, biólogos e ecologistas têm utilizado ACs para explicar padrões espaciais interessantes que aparecem em ecossistemas [1], [2], em crescimento de tumores [3], bem como a pigmentação de animais [4]. Com tal característica, sistemas químicos como a reação de Belousov-Zhabotinsky foram modelados usando ACs [5].

Além do uso de ACs como modelos de fenômenos naturais complexos, eles também foram pesquisados de uma perspectiva mais teórico por cientistas da computação e matemáticos. Dado que os ACs constituem as contrapartes discretas de equações diferenciais parciais, eles podem ser investigados em termos de suas características dinâmicas, como equilíbrio, estabilidade e dependência da sensibilidade das condições iniciais [6, 7, 8, 9, 10]. Para caracterizar os ACs é importante enfatizar que existem a principio duas maneiras de analisar esse problema. O primeiro remete a uma conjectura de Wolfram [11], que assegura que todos os ACs podem ser classificados em uma das quatro classes comportamentais com base em um observação dos diagramas de espaço-tempo. A principal desvantagem desta abordagem consiste no fato de que tal exame visual torna-se demorado se pretende-se apurar uma família inteira de ACs e que o resultado do processo de classificação pode variar, pois envolve a possibilidade de subjetividade, colocando em risco sua reprodutibilidade. A segunda abordagem se baseia em medidas que quantificam a dinâmica dos ACs, como os expoentes de Lyapunov [12, 13, 9], entropias [14] e outros [15, 16], que são semelhantes em essência a aqueles que são normalmente usados para estudar outros tipos de sistemas dinâmicos.

No momento em que se trata de uma classificação automatizada de ACs, existem apenas alguns trabalhos na literatura que tratam desse assunto. Kunkle [17] apresenta um algoritmo baseado em redes neurais para realizar esta tarefa no caso de ACs elementares e uma família de ACs totalísticos. Este algoritmo envolve sete parâmetros, sendo atividade, determinismo reverso, sensibilidade, atividade absoluta, dominância de vizinhança, propagação de atividade e incompressibilidade. No trabalho de Wuensche [18], as regras do AC são classificadas por uma medida relacionada à entropia. Em ambos os casos, a complexidade dos padrões gerados é examinada ao longo do tempo, o que contrasta com nossa abordagem que se baseia na observação dos diagramas de espaço-tempo como um todo.

Uma classificação através de observações dos diagramas de espaço-tempo de ACs de acordo com a classificação proposta por Wolfram [11] está relacionado à percepção de textura pelo sistema visual humano. A interação dos componentes de textura, assim como a ordem estatística dos níveis de cinza da imagem de textura, são os principais motivos implícitos a uma distinção visual de texturas [19, 20]. As propriedades dos padrões em estudos de percepção de textura servem de inspiração para

nossa proposta. Seu objetivo é investigar o potencial dos descritores de textura em representar uma inspeção dos diagramas de espaço-tempo de ACs e classificá-las de forma automatizada de acordo com a classificação proposta por Wolfram [11]. Dessa forma, é possível superar as principais desvantagens associadas a uma classificação manual de ACs.

A partir de agora, esses diagramas de espaço-tempo também serão chamadas de imagens, pois serão obtidas como imagens e tratadas por meio de técnicas bem estabelecidas na análise de imagens. Mais propriamente, duas técnicas de extração de características são utilizadas, sendo a Variância de Padrão Binário Local (LBPV) e a codificação de Autoencoder Convolucional [21]. Será realizado um extenso estudo experimental com ACEs, no qual será verificado através de aprendizado não supervisionado para verificar o potencial dos descritores de imagens, para classificação de ACEs.

2 Fundamentos Teórico

2.1 Autómato Celular

Geralmente, um AC pode ser visto como uma matriz n-dimensional de células, cada uma delas portando um número finito de estados discretos, e o estados dessas células são atualizados em intervalos de tempo discretos de acordo com os estados das células em sua vizinhança [22].

Formalmente, um AC $\mathscr{C}$ pode ser expresso como uma quíntupla.

$$\mathscr{C} = \langle \tau, S, s, N, \Phi \rangle$$

Onde temos que:

- $\boldsymbol{\tau}$: Uma tesselação que é composta por um conjunto de células c_i. Em geral ACs são representados em tesselações regulares em espaços euclidianos n-dimensionais $\mathbb{Z}^n$;
- **S**: O espaço de fase formado por um conjunto finito de estados;
- **s**: A função $s(c_i, t)$ mapeia os estados da célula c_i em um determinado tempo;
- **N**: Mapeia o conjunto de vizinhanças de cada célula c_i. Além disso, em ACs bidimensionais podem ser definidos as vizinhas de von Neumann ($|N| = 4$) e Moore ($|N| = 8$), em que o número de vizinhos da célula c_i, é representado por $|N(c_i)|$;
- $\boldsymbol{\Phi}$: $S^{|N(c_i)|} \rightarrow S$ define a dinâmica de evolução do AC. Assim, os estados futuros da célula c_i depende da configuração atual dos estados dela mesma e/ou da sua vizinhança. A evolução dos estados de todas as células ocorre de forma simultânea e sincronizada;

Os ACs ainda podem ser classificados como a maneira que interage com seus vizinhos, reversibilidade, espaço contínuo e o tipo de grade, descrevemos tais classificações como:

- **Totalístico (*totalistic*)**: Corresponde a o AC que sua função de transição depende apenas do estado da células vizinhas e não do arranjo espacial dos vizinhas
- **Exterior totalista (*outer-totalistic*)**: Corresponde a o AC que sua função de transição depende do estado da células vizinhas e do estado da célula central.
- **AC reversível**: Um AC é reversível se, para cada configuração atual AC, houver exatamente uma configuração passada. Se pensarmos em um AC como uma função que mapeia configurações para configurações, a reversibilidade implica que essa função é bijetora. Se um AC é reversível, seu comportamento reverso no tempo também pode ser descrito como um AC.

- **ACs espaciais contínuos**: Os Autómatos espaciais contínuos têm um contínuo de localizações. O estado de uma localização é um número finito de números reais. O tempo também é contínuo e o estado evolui de acordo com as equações diferenciais.
- **Malha / grade não quadrada**: As malhas nas quais um CA é executado não precisam ser quadradas. Maurice Margenstern, por exemplo, introduziu CA nos planos hiperbólicos [23].

2.2 Autómato Celular Elementar

Em 1983, Wolfram [24] propôs o Autómato Celular Elementar (ACE). Este Autómato pode dar origem a comportamentos extraordinariamente complexos, apesar da sua simplicidade em termos de definições.

Então temos que o ACE $\mathscr{C}_{ACE}$ é:

$$\mathscr{C}_{ACE} = \Big\langle \mathbb{Z}^1, S = \{0,1\}, s_0, \mathbb{N} = (c_{i-r}, c_i, c_{i+r}), \Phi : \mathbf{S}^3 \to \mathbf{S}) \Big\rangle$$

Observe que, como o ACE é do tipo totalista e o raio $r = 1$, a vizinhança inclui as células da esquerda, da direita e ela mesma, também conhecido como acoplamento com o vizinho mais próximo.

As Regras de transições do ACE são definidas por uma tabela, na qual são mapeadas cada uma das possíveis arranjos de vizinhança em relação ao possível estado de saída [24]. Portanto, dado que o tamanho da vizinhança é $2r + 1$, existem $2^8 = 256$ regras. Por outro lado, para poder visualizar o diagrama espaço-tempo, Wolfram [24] descreveu uma estrutura bidimensional, isto é, a evolução das células ao longo do tempo t são empilhadas verticalmente, representando uma imagem de tamanho $|\tau| \times t$.

Na Figura 1 é ilustrado o ACE de regra 30, ou seja a função de transição, em que 30 é convertido para 00011110 em binário, a forma de representar a dinâmica do ACEs segundo o modelo de Wolfram [24]. Já que a tesselação contém 3 vizinhos, logo existem $2^3 = 8$ possíveis combinações.

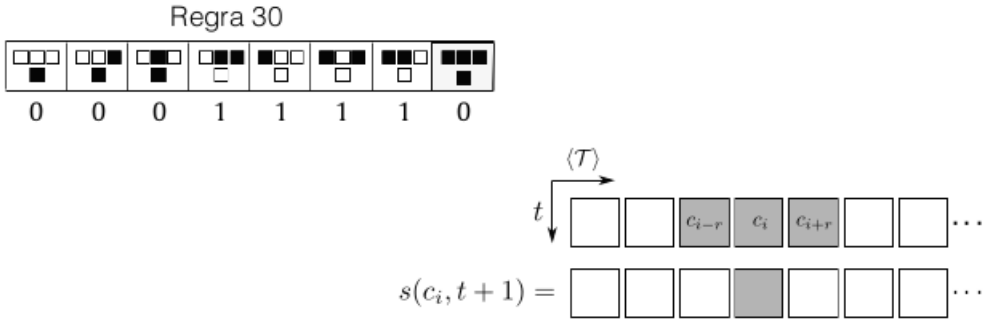
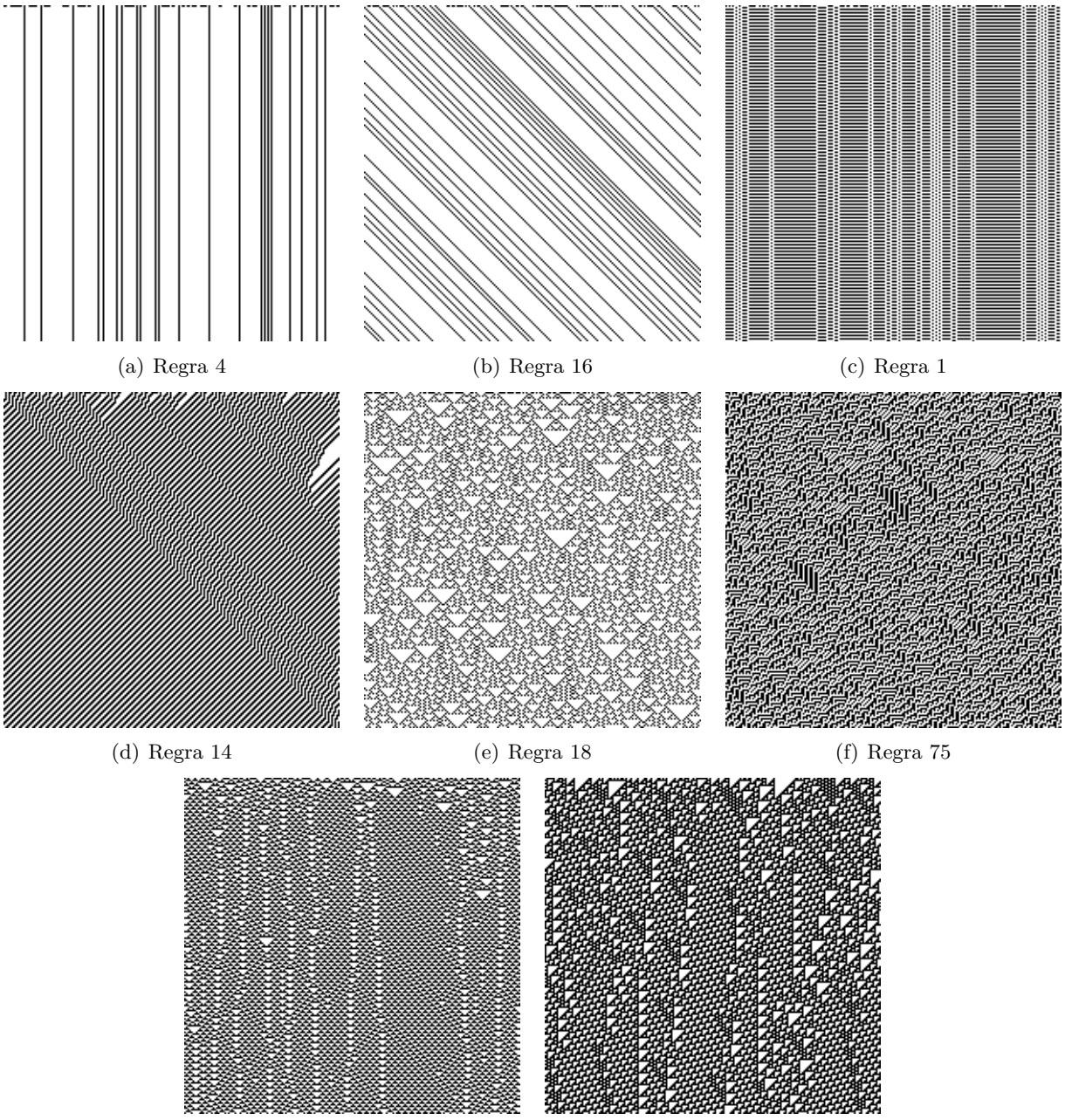


Figura 1: Autómato Celular Elementar com a regra 30 (superior) e como ocorre a evolução do do diagrama espaço-tempo(inferior), no qual 1 corresponde a branco e 0 a preto.

Por exemplo, no painel superior da Figura 1, os quadrados superiores representam a vizinhança e o quadrado inferior a saída da função. Da esquerda para a direita, para uma configuração da vizinhança $\mathbf{N} = (c_{i-r,}, c_i, c_{i+r})$ no qual todas as células estão vivas, seria representado por 111 em binário, os estado da célula c_i é $s(c_i, t+1) = 0$ e assim sucessivamente sucessivamente seguindo a regra no quadro superior Wolfram [24].



(a) Regra 4 (b) Regra 16 (c) Regra 1

(d) Regra 14 (e) Regra 18 (f) Regra 75

(g) Regra 54 (h) Regra 110

Figura 2: Diagrama de espaço-tempo dos ACEs do Wolfram pertencendo elas as Classe 1 (a)-(b), Classe 2 (c)-(d), Classe 3 (e)-(f), Classe 4 (g)-(h), evoluídos a partir de uma condição inicial aleatória.

Os ACEs segundo Wolfram [24] podem ser classificados em 4 classes que ditam como é sua dinâmica no diagrama de espaço-tempo.

As 4 classes que dizem sobre sua dinâmica, podem ser descritas como:

- **Classe 1:** A evolução leva a um estado homogêneo.
- **Classe 2:** A evolução leva a um conjunto de estados simples e periódicos.
- **Classe 3:** A evolução leva a um padrão caótico.

- **Classe 4:** A evolução leva para estruturas localizada complexas, algumas vezes por um longo período de tempo.

2.3 Local Binary Pattern Variance

O Local Binary Pattern Variance (LBPV) é uma abordagem efetiva para adicionar a informação local de variância no clássico LBP (Local Binary Pattern). A variância local quantifica a taxa de variação de cinza e, consequentemente, nos permite caracterizar regiões de alta frequência nas imagens. Incluir essa informação contribui para a melhor classificação da textura.

LBP é um método capaz de identificar estruturas espaciais locais na textura da imagem. Essencialmente, ele traduz os padrões locais para códigos binário e utiliza o mesmo para distinguir entre diferentes padrões de textura. Os códigos binários são computados pela comparação de intensidade de cada pixel p_{ij} na imagem com a intensidade dos pixeis vizinhos.

A simples caracterização dos códigos binários através dos seus números decimais correspondentes. O padrão binário de cada pixel p_{ij} é calculado e convertido em um número decimal usando

$$L_{ij} = \sum_{n=0}^{|\mathcal{N}_{ij}|-1} s(g(p_{ij}^n - g(p_{ij}))2^n \tag{1}$$

Onde $\mathcal{N}_{ij}$ contém os pixeis vizinhos p_{ij}, p_{ij}^n é o enésimo (pixel) vizinho de p_{ij}, e temos que $g(p_{ij})$ retorna a intensidade de cinza do pixel p_{ij} e s é o função de Heaviside. O histograma de L_{ij} para todos os pixeis em uma imagem $M \times N$ podem ser utilizados com vetor de característica.

LBP tem duas proprieades: uniformidade em *bitwise* e invariância sobre rotação. A uniformidade do pixel p_{ij} quantifica a relação entre sua intensidade e a intensidade do vizinho como mostra a equação a seguir

$$\begin{aligned} U_{ij} = &\left| s(g(p_{ij}^{|\mathcal{N}_{ij}|-1}) - g(p_{ij})) - s(g(p_{ij}^0) - g(p_{ij})) \right| \\ &+ \sum_{n=1}^{|\mathcal{N}_{ij}|-1} \left| s(g(p_{ij}^n) - g(p_{ij})) - s(g(p_{ij}^{n-1}) - g(p_{ij})) \right| \end{aligned} \tag{2}$$

Por exemplo, o padrão local $256 = (11111111)_2$ tem uniformidade 0, enquanto o padrão $24 = (00011000)_2$ tem uniformidade 2. Se sabe que uniformidade no máximo 2 são padrões fundamentais de textura local em imagens.

A invariância sobre rotação é alcançada apenas considerando o número de bits não nulos em uma sequência binária que se obtém através da função degrau 1. Assim, o identificador único para um dado padrão binário uniforme local invariante de rotação é definido por

$$\widetilde{L_{ij}} = \begin{cases} \sum_{n=0}^{|\mathcal{N}_{ij}|-1} s(g(p_{ij})^n - g(p_{ij})), & \text{se } U_{ij} \leq 2, \\ |\mathcal{N}_{ij}| + 1, & \text{se } U_{ij} > 2 \end{cases} \tag{3}$$

Onde o til faz referência a invariância sobre rotação considerando um padrão uniforme, para este valor de U_{ij} é no máximo 2. Similar a Equação 1, o histograma de $\widetilde{L_{ij}}$ para todos os pixeis pode ser utilizado como vetor de característica.

Finalmente, como oposição ao clássico LBP, LBPV inclui a variância σ_{ij}^2 da intensidade de cinza do p_{ij} vizinhos como uma peso adaptativo quando computado o histograma de $\widetilde{L}_{ij}$. Desta forma, o histograma do LBPV é calculado como

$$H(n) = \sum_{i=i}^{N} \sum_{j=1}^{M} w(\widetilde{L_{ij}}, n), \tag{4}$$

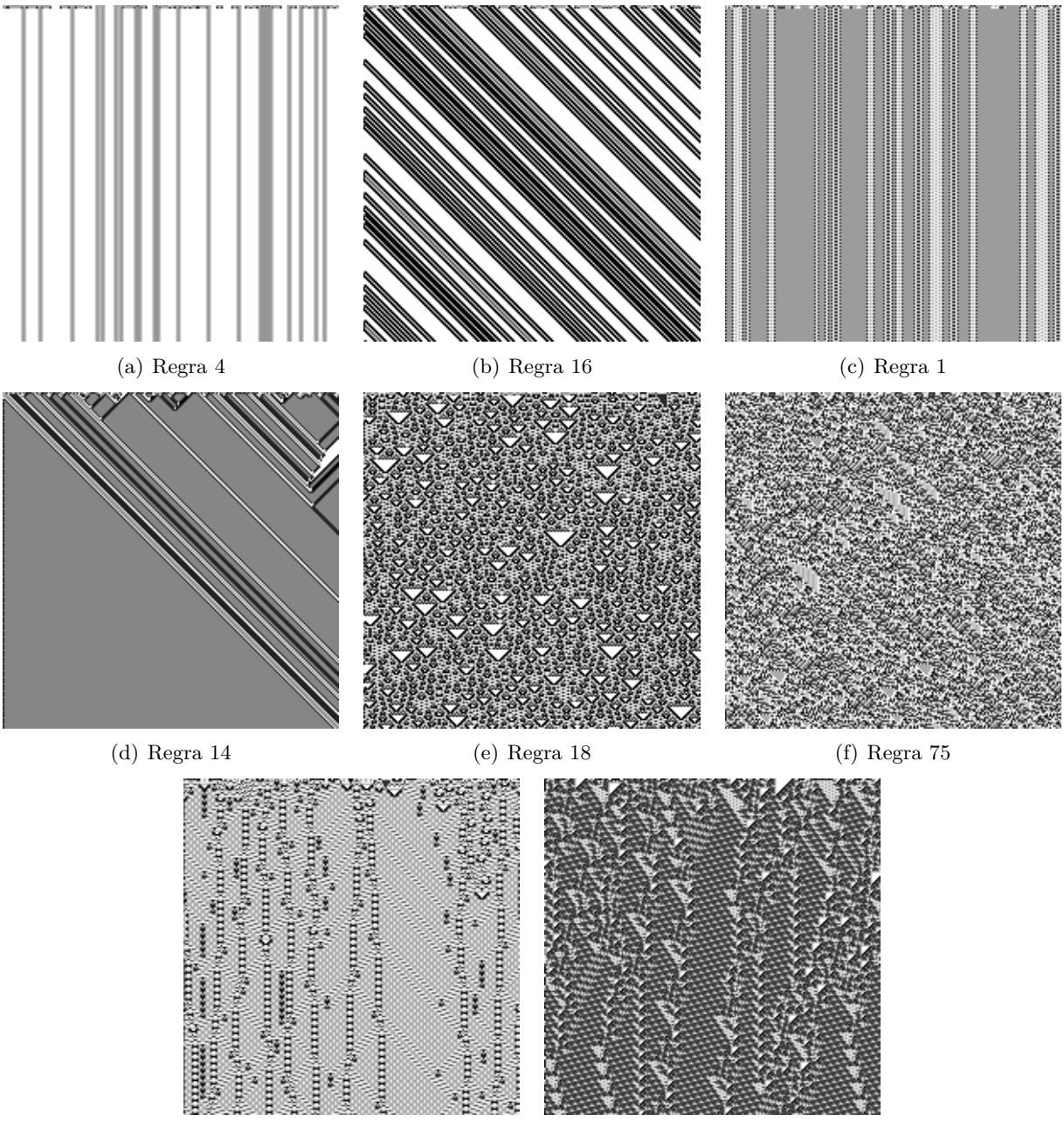
onde $n \in [0, |\mathcal{N}_{ij} + 1|]$ e o peso w é dado por

$$w(\widetilde{L_{ij}}, n) = \begin{cases} \sigma_{ij}^2, \text{ se } \widetilde{L_{ij}} = n, \\ 0, \text{ senão} \end{cases} \tag{5}$$

Neste trabalho, foi utilizado a seguinte vizinhança de Moore

$$\mathcal{N}_{ij} = (p_{i-1,j-1}, p_{i-1,j}, p_{i-1,j+1}, p_{i,j-1}, p_{i,j+1}, p_{i+1,j-1}, p_{i+1,j}, p_{i+1,j+1})$$

Por esse motivo, o histograma do LBPV representa toda a imagem como um vetor de características com $|\mathcal{N}_{ij}| + 2 = 10$ elementos



(a) Regra 4 (b) Regra 16 (c) Regra 1

(d) Regra 14 (e) Regra 18 (f) Regra 75

(g) Regra 54 (h) Regra 110

Figura 3: Diagrama de espaço-tempo dos ACEs do Wolfram após a aplicação do LBPV, tal qual pertence eles as Classe 1 (a)-(b), Classe 2 (c)-(d), Classe 3 (e)-(f), Classe 4 (g)-(h) evoluídos a partir de uma condição inicial aleatória.

2.4 Rede Neural Convolucional

A Rede Neural Convolucional (RNC) teve resultados inovadores nas últimas décadas em diversos campos relacionados ao reconhecimento de padrões. A suposição mais importante sobre problemas que são resolvidos por RNC é que não deve existir características que são espacialmente dependentes. Outro aspecto importante da RNC é que se deve obter recursos abstratos quando a entrada se propaga para as camadas mais profundas [25].

2.4.1 Convolução

Suponhamos que exista uma matriz de entrada $M \times N$, no qual é seguido por uma camada de convolução. Se utilizarmos um filtro w com dimensões $m \times n$, a saída da camada de convolução vai ter tamanho $(N-n+1) \times (N-m+1)$. A convolução para um pixel na próxima camada é calculado de acordo com a Equação 6

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} w_{ab} y_{(i+a)(j+b)}^{l-1} \tag{6}$$

Onde x_{ij} é a saída na próxima camada, y é a imagem de entrada e w é o filtro que será aplicado sobre a imagem. Na Figura 4, é mostrado como ocorre a operação de convolução sobre a imagem. Como pode ser observado, o produto de elemento por elemento da entrada é somado, e então representa o pixel correspondente na próxima camada [25].

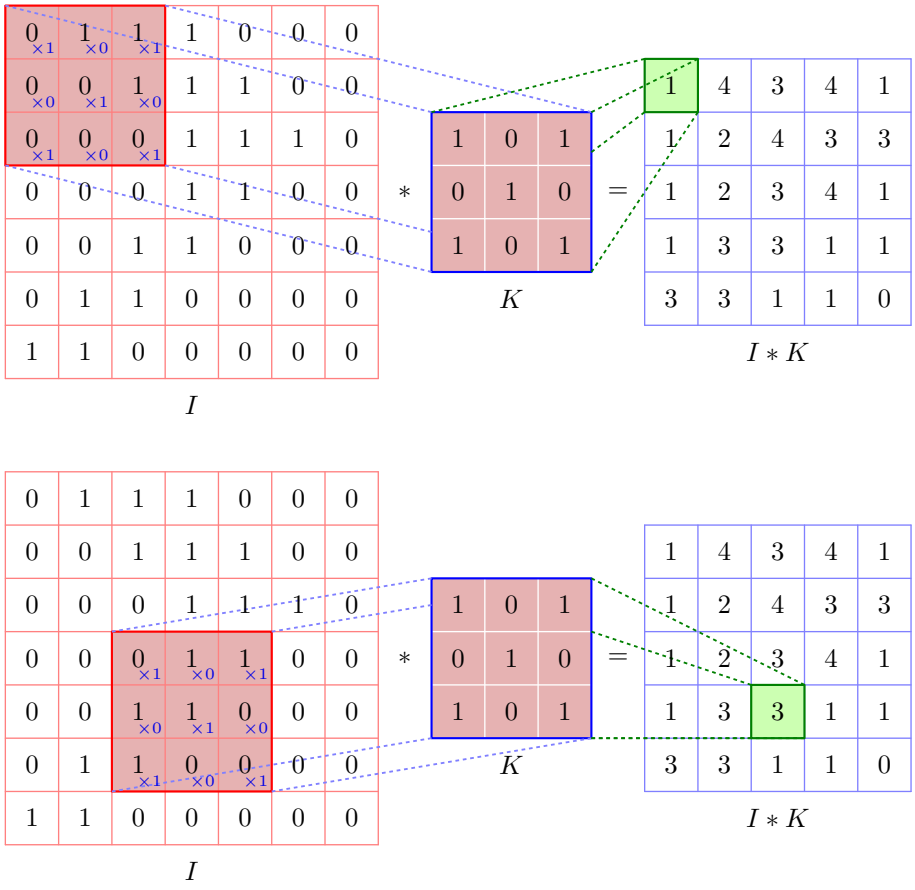


Figura 4: Demonstração visual da operação de convolução.

2.4.2 Função de ativação

A próxima camada depois da convolução é a função de ativação. A função de ativação pode ser utilizada para ajustar a saída gerada. Esta camada é aplicada para saturar a saída ou limitar a saída gerada.

Por muitos anos a função sigmoid e tanh foram as mais utilizadas. Na Figura 5, mostra as quatro funções de ativação não lineares mais comuns. No entanto, a *Rectified linear unit* (ReLU) tem sido utilizada mais frequentemente pelas seguintes razões.

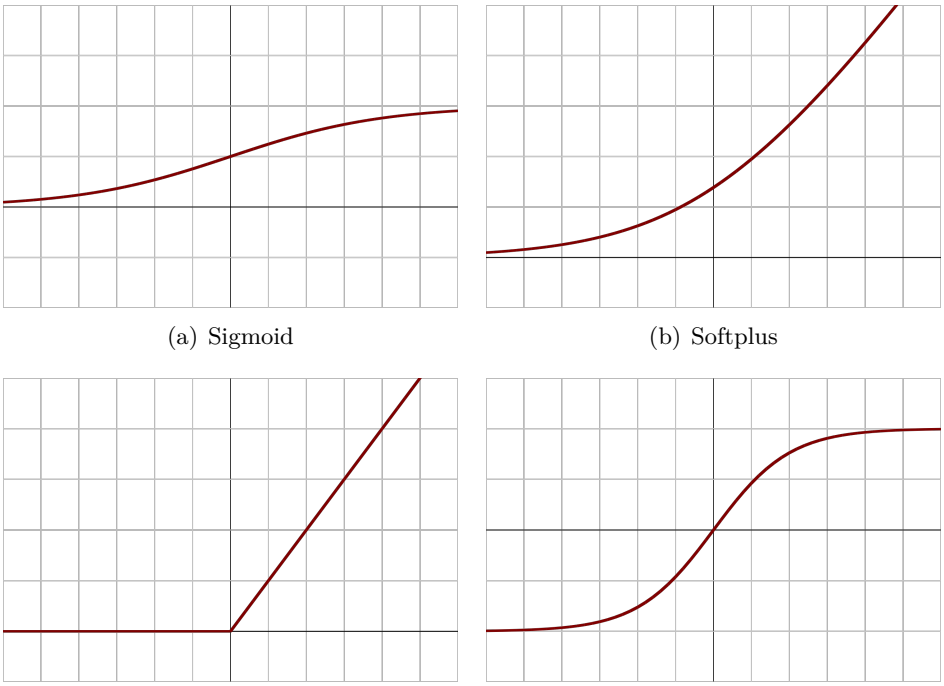
1. ReLU tem a definição mais simples entre as funções e o gradiente.

$$ReLU(x) = max(0, x) \tag{7}$$

$$\frac{d}{dx} ReLU(x) = \begin{cases} 1 & \text{if } x > 0; \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

2. As funções de ativação tal como sigmoid e tanh causam muitos problemas no *back propagation*. Como a rede neural é mais profunda, o sinal do gradiente começa a desaparecer, o qual é chamado de "gradiente de fuga". Isto acontece porque o gradiente destas funções é muito próximo de zero. No entanto, a ReLU tem um gradiente constante para valores positivos.

3. A ReLU cria uma representação esparsa. Porque o zero no gradiente leva a obter zeros completos. Contudo, sigmoid e tanh sempre leva a valores diferentes de zero para o gradiente, o qual pode ser um problema para o treinamento [25].



(a) Sigmoid (b) Softplus

(c) Rectified linear unit (ReLU) (d) Hyperbolic tangent (thanh)

Figura 5: Funções de ativação (a) Sigmoid, (b) Softplus, (c) ReLU e (d) thanh

2.4.3 Pooling

A principal ideia do pooling é reduzir a resolução da entrada ao longo de suas dimensões espaciais a fim de diminuir a complexidade para as camadas seguintes. O Max Pooling é um dos métodos mais comuns de realizar o pooling. Ele particiona a imagem em sub-regiões retangulares, então retorna o valor máximo dentro da sub-região. A dimensão mais comum de se realizar o Max Pooling é a dimensão 2×2. Na Figura 6, quando é realizado a operação de pooling temos que o passo é 1, porém é comum a operação de pooling com 2 passos. Para 1 passo, deve ser levado em consideração quando a redução de resolução não conserva a posição da informação. Portanto , deve ser aplicado apenas quando a presença da informação importa. Além disso, o pooling pode ser usado com filtros e passos diferentes para aumentar a eficiência. Como exemplo, um Max-Pooling com dimensão 3×3 e 2 passos, assim mantendo alguma sobreposição entre as áreas [25].

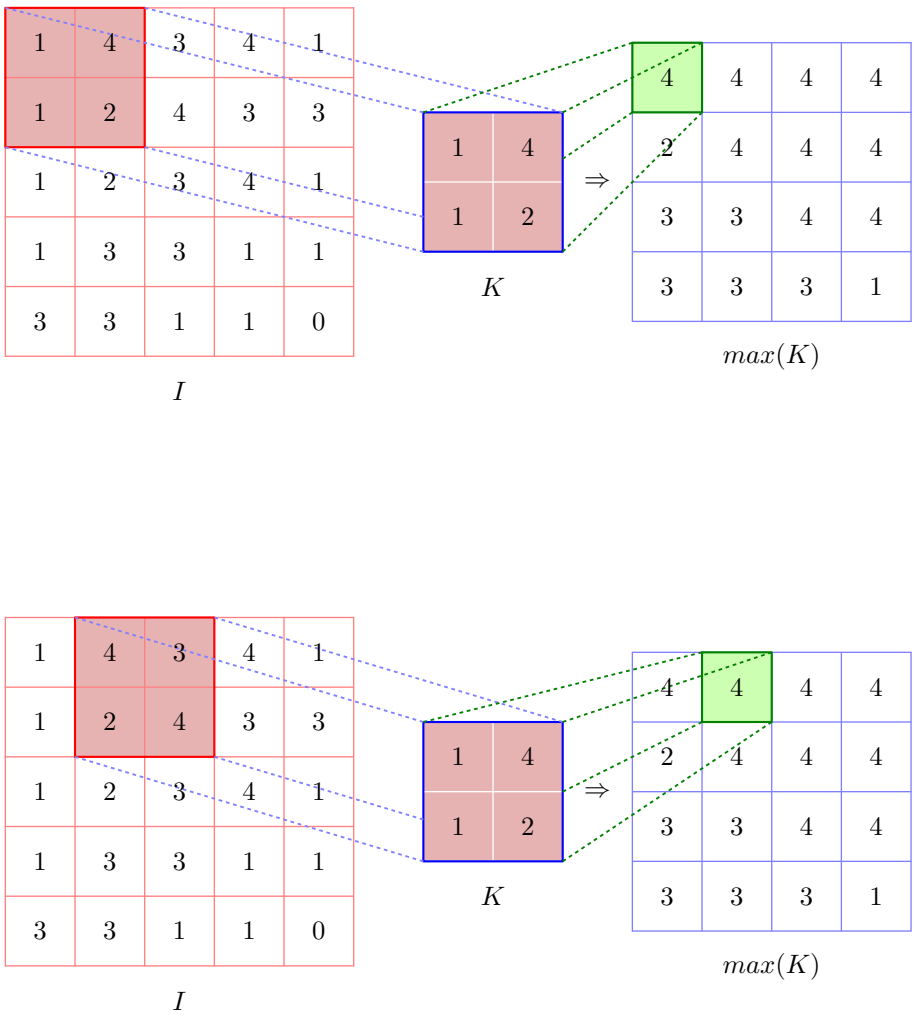


Figura 6: Demonstração visual da operação de Max Pooling.

2.5 Autoencoder

A abordagem supervisionada é útil quando temos dados já classificados, no qual os pesos são atualizados através de algoritmos de *forward* e *backward propagation*. Comparado com abordagem supervisionada, a abordagem não supervisionada pode receber diretamente os dados sem uma classificação prévia, o qual reduz drasticamente o fluxo de trabalho para classificar manualmente os dados [26].

Então o Autoencoder retorna uma informação a fim de reconstruir a informação fornecida na entrada. Depois de várias interações, o valor da função de custo atinge o valor ótimo, o que significa que a reconstrução dos dados é aproximadamente igual à dos dados originais.

Seja I a entrada representada por um vetor de dimensão M tal que $I \in \mathbb{R}^n$. Seja E o dado codificado um vetor de dimensão N tal que $E \in \mathbb{R}^m$. Um Autoencoder padrão inclui duas principais etapas:

1. **Codificação:** Converte o dado de entrada I em E na camada escondida por,

$$E = f(I) = \sigma(w \cdot I + b) \tag{9}$$

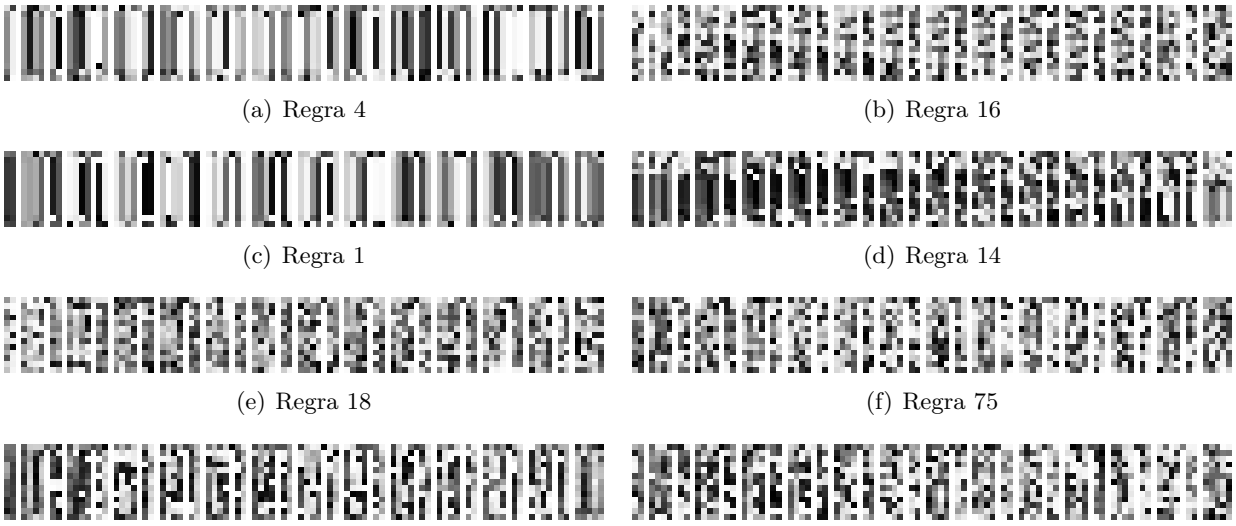
   Onde $w \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^n$. No qual σ é a função de ativação.

2. **Decodificação:** A partir de E a camada codificada, é reconstruindo a valor de entrada O' por,

$$O' = f'(E) = \phi(\hat{w} \cdot E + \hat{b}) \tag{10}$$

Onde $\hat{w} \in \mathbb{R}^{n \times m}$ e $\hat{b} \in \mathbb{R}^m$. No qual ϕ é a mesma função de ativação.

Na Figura 7, temos a ultima camada de codificação do Autoencoder Convolucional para as regras já apresentadas.



(a) Regra 4 (b) Regra 16

(c) Regra 1 (d) Regra 14

(e) Regra 18 (f) Regra 75

(g) Regra 54 (h) Regra 110

Figura 7: Codificação dos ACEs do Wolfram pertencendo elas as Classe 1 (a)-(b), Classe 2 (c)-(d), Classe 3 (e)-(f), Classe 4 (g)-(h), evoluídos a partir de uma condição inicial aleatória.

2.5.1 Upsampling

Existe diversas formas de se realizar o upsampling, uma delas é por interpolação, porém, o processo mais comum de se fazer em um Autoencoder Convolucional, é pegar o pixel da camada anterior e expandir ele na atual como dimensão 2×2. Essa expansão tem como objetivo deixar as dimensões das camadas de codificação, exatamente igual a camada de descodificação.

3 Experimento e Resultados

O LBPV é um método que já foi utilizado em outro trabalho [27], e apresentou excelentes resultados, porém foi utilizado técnicas supervisionadas de Machine Learning, porém o objetivo nesse trabalho é encontrar um modelo não supervisionado, que possa realizar a classificação de forma automática independente do sistema que estamos estudando e sem a necessidade de esforço humano para determinar qual classe do Wolfram pertence.

3.1 Conjunto de dados

Para realizar o experimento foram utilizados dois conjuntos de dados. Ambos gerados com condições inicias aleatórias e 100 amostras para cada regra. Para o LBPV foi utilizado um conjunto de dados com imagens de dimensão de 350×700 porque segundo da Silva et al [27] o tamanho da imagem é diretamente proporcional a taxa de acerto no modelo supervisionado. No entanto, para o *Autoencoder* foi utilizado imagens com dimensão de 100×100, uma vez que o modelo utilizado exige uma grande quantidade de memória para treinamento.

3.2 Arquitetura do Autoencoder

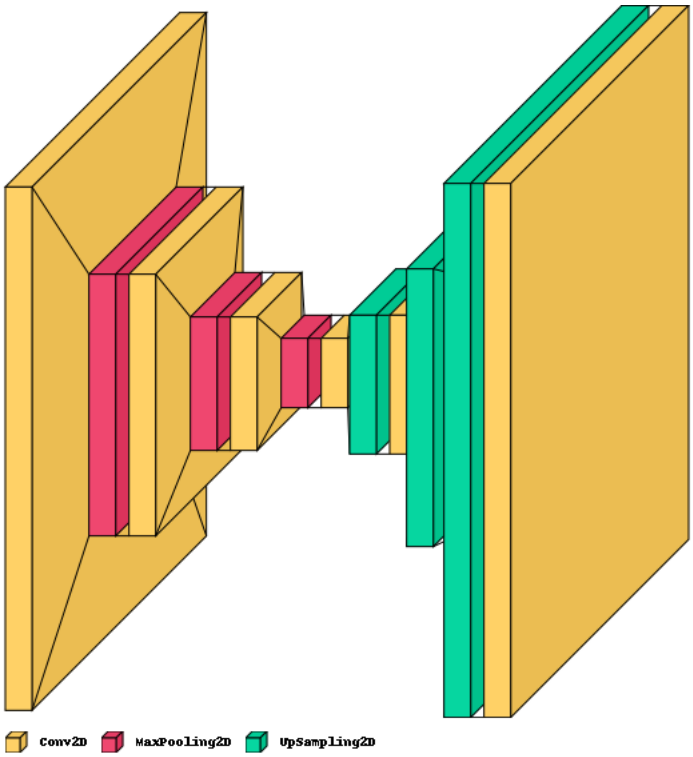
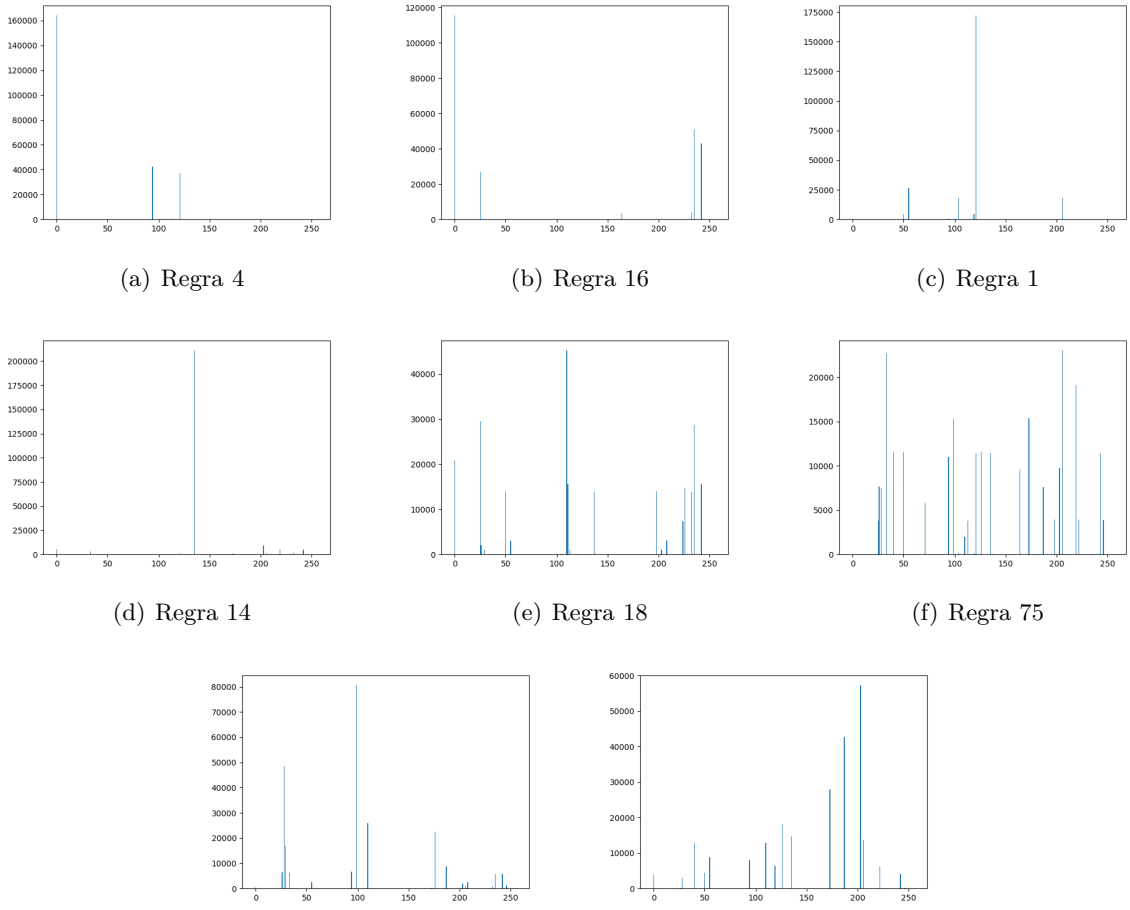


Figura 8: Representação visual do Autoencoder Convolucional.

Na Figura 8, temos arquitetura da do Autoencoder que utilizamos para realizar a experiência. Nas camadas de convolução foi utilizado filtros com dimensão de 3×3 em todas as camadas, 16 filtros na primeira e penúltima camada e 8 para as demais, no qual todos as camadas utilizaram a função de ativação ReLU e na última camada o sigmoid. Para as camadas de Max Pooling e Upsampling foi utilizado uma área de dimensão 2×2.

3.3 Histograma

Na Figura 9 temos os histogramas com a distribuição da intensidade de cinza para as regras apresentadas anteriormente após a aplicação do LBPV. Podemos observar raias que ganham destaque em cada uma das classes como por exemplo na classe 4, no qual se destaca uma raia próximo a 100 na escala de cinza, que apenas é possível observar no mesmo. Logo o LBPV apresenta um alto grau de organização, porém ainda não é possível chegar a alguma conclusão, já que é necessário fazer uma análise minuciosa com todas as regras, assim garantido que tal padrão se repita para os demais.



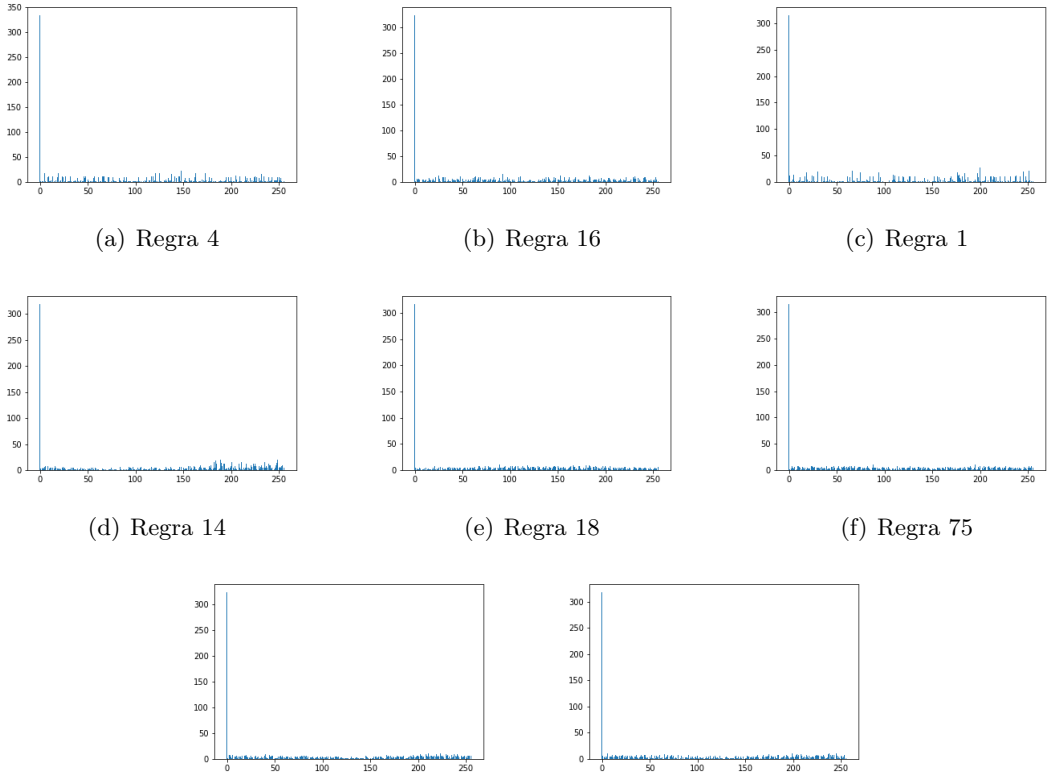
(a) Regra 4 (b) Regra 16 (c) Regra 1

(d) Regra 14 (e) Regra 18 (f) Regra 75

(g) Regra 54 (h) Regra 110

Figura 9: Histograma da imagem após a aplicação do LBPV para Classe 2 (a)-(b), Classe 3 (c)-(d), Classe 4 (e)-(f), evoluídos a partir de uma condição inicial aleatória.

Porém para o histograma do Autoencoder na Figura 10, é possível notar que não existe nenhum padrão quanto a distribuição da intensidade de cinza, o que torna o modelo inviável, de modo que é possível encontrar muito ruído no histograma do mesmo.



(a) Regra 4 (b) Regra 16 (c) Regra 1

(d) Regra 14 (e) Regra 18 (f) Regra 75

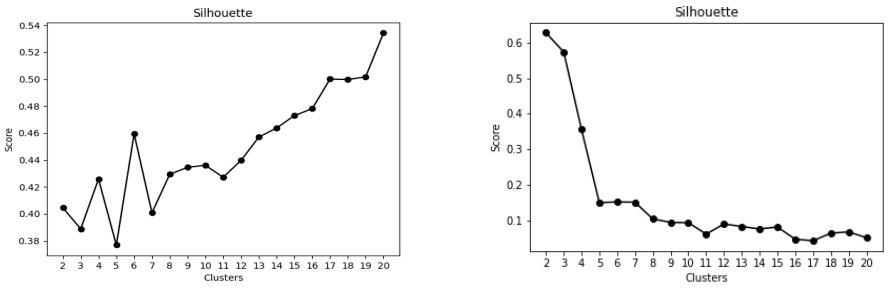
(g) Regra 54 (h) Regra 110

Figura 10: Histograma da camada codificada do AutoEncoder para Classe 2 (a)-(b), Classe 3 (c)-(d), Classe 4 (e)-(f), evoluídos a partir de uma condição inicial aleatória.

3.4 Método de Silhueta

Para analisar se é possível segmentar as imagens em quatro categorias iremos utilizar o Método de Silhueta. Ele nos permite determinar quantos clusters é possível utilizar, além de determinar o quanto os dados se ajusta nos clusters.

Temos que o resultado do LBPV apresenta uma pontuação razoável para o caso em que queremos apenas 4 clusters, porém a pontuação aumenta quase que linearmente para número clusters maiores, talvez possa ser um indicativo que exista outras características que o LBPV pode segmentar melhor. Enquanto o Autoencoder apresenta um valor ligeiramente inferior para 4 clusters em relação ao LBPV, além de a pontuação decair rapidamente para quantidades maiores de cluster como mostra a Figura 11.



(a) LBPV (b) Autoencoder

Figura 11: Gráfico do Método de Silhuetas para o LBPV e Autoencoder Convolucional

Olhando apenas para o gráfico na Figura 11, podemos observar que os dados do Autoencoder não parecem ajudar a extrair nenhuma segmentação interessante da imagem, independe se os quatro clusters representam a classificação do Wolfram de fato. No entanto, o LBPV se mostrou bem mais promissor para extrair alguma característica do sistema, dando indícios que talvez tenha algo de interessante com número maior de clusters.

Porém apenas essa análise não é suficiente para determinar se ambos os métodos são capazes de fazer a classificação. Então para ter uma análise melhor do problema, então em seguida temos a segmentação do K-means para cada método.

3.5 Clusters

Para montar os seguintes clusters, cada regra foi separada pelo cluster que esteva mais frequente, porque em alguns casos a regra estava contida e mais de um cluster, porém poucos casos. Logo abaixo temos como o K-means fez a segmentação do LBPV, a primeira vista podemos observar que parece existir uma certa organização dos dados, porém para a regra 20 e 52 que estão contidos na Classe 4 e as regras 2, 6, 10, 18, 22, 26, 30, 34, 38, 42, 46 e 50 que estão contidos na Classe 3 estão juntas no Cluster 1.

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

Logo abaixo temos o Autoencoder, no qual obtivemos um resultado muito pior, não é necessário muita análise para notar que praticamente todas as Classes estão no Cluster 1. O que era esperado dado o resultado do Método de Silhueta.

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

- **Cluster 3:** 202, 216, 218, 219, 222, 223, 233, 237, 247
- **Cluster 4:** 0, 8, 32, 40, 64, 96, 128, 136, 160, 168, 192, 224

A partir dos resultados obtidos podemos concluir que o LBPV apresentou um certo grau de organização, então é um método que ainda pode ser explorado com a mudança dos seus parâmetros, como o raio e o número de vizinhos. Já para o Autoencoder, mesmo tendo resultados ruins, ainda é possível explorar diversas arquiteturas de rede neural, como a VGG16 e a Transformer, além de outras.

4 Conclusão

Através de dois modelos de aprendizado não supervisionado, foi observado que o LPBV, um modelo já demonstrado em outras trabalhos com aprendizado supervisionado mostrou ótimos resultados, no qual o mesmo apresentou se promissor para futuros trabalhos com aprendizado não supervisionado, mesmo não atingindo o resultado esperado, sendo que o mesmo mostrou uma distribuição interessante no histograma e quando aplicado o K-Means apresentou classificação com alguma coerência. Por outro lado, o Autoencoder Convolucional apresentou péssimos resultados, o histograma estava com muito ruído, a classificação do modelo não apresentou nenhum resultado interessante, porém todos esses fatores pode estar associado a diversos parâmetros que compõem uma rede neural, provavelmente a arquitetura utilizada não foi apropriada para o problema, já que foi empregado uma arquitetura bem simples. Em um próximo trabalho é interessante se utilizar arquiteturas já conceituadas como VGG16, Transformer e entre outras.

Referências

[1] Emily Burkhead and Jane Hawkins. A cellular automata model of ebola virus dynamics. *Physica A: Statistical Mechanics and its Applications*, 438:424–435, 2015.

[2] Johan van de Koppel, Joanna C Gascoigne, Guy Theraulaz, Max Rietkerk, Wolf M Mooij, and Peter MJ Herman. Experimental evidence for spatial self-organization and its emergent effects in mussel bed ecosystems. *science*, 322(5902):739–742, 2008.

[3] Zhihui Wang and Thomas S Deisboeck. Computational modeling of brain tumors: discrete, continuum or hybrid? In *Scientific modeling and simulations*, pages 381–393. Springer, 2008.

[4] Andreas Deutsch, Sabine Dormann, et al. *Cellular automaton modeling of biological pattern formation*. Springer, 2005.

[5] Mario Markus and Benno Hess. Isotropic cellular automaton for modelling excitable media. *Nature*, 347(6288):56–58, 1990.

[6] Jarkko Kari. Theory of cellular automata: A survey. *Theoretical computer science*, 334(1-3):3–33, 2005.

[7] Olivier Martin, Andrew M Odlyzko, and Stephen Wolfram. Algebraic properties of cellular automata. *Communications in mathematical physics*, 93(2):219–258, 1984.

[8] Genaro J Martinez. A note on elementary cellular automata classification. *arXiv preprint arXiv:1306.5577*, 2013.

[9] Mark A Shereshevsky. Lyapunov exponents for one-dimensional cellular automata. *Journal of Nonlinear Science*, 2(1):1–8, 1992.

[10] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601, 1983.

[11] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[12] Jan M Baetens and Janko Gravner. Stability of cellular automata trajectories revisited: branching walks and lyapunov profiles. *Journal of nonlinear science*, 26(5):1329–1367, 2016.

[13] F Bagnoli, R Rechtman, and Stefano Ruffo. Damage spreading and lyapunov exponents in cellular automata. *Physics Letters A*, 172(1-2):34–38, 1992.

[14] Andrew Ilachinski. *Cellular automata: a discrete universe*. World Scientific Publishing Company, 2001.

[15] Chris G Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1-3):12–37, 1990.

[16] Andrew Wuensche, Mike Lesser, and Michael J Lesser. *Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, volume 1. Andrew Wuensche, 1992.

[17] Daniel R Kunkle. *Automatic Classification of One-Dimensional Cellular Automata*. PhD thesis, Citeseer, 2003.

[18] Andrew Wuensche. Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the z parameter. *Complexity*, 4(3):47–66, 1999.

[19] Benjamin J Balas. Texture synthesis and perception: Using computational models to study texture representations in the human visual system. *Vision research*, 46(3):299–309, 2006.

[20] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[21] Zhenhua Guo, Lei Zhang, and David Zhang. Rotation invariant texture classification using lbp variance (lbpv) with global matching. *Pattern recognition*, 43(3):706–719, 2010.

[22] J von Neumann. The general and logical theory of automata, cerebral mechanism in behavior, edited by w. jeffress, 1951.

[23] Maurice Margenstern. New tools for cellular automata in the hyperbolic plane. *Journal of Universal Computer Science*, 6(12):1226–1252, 2000.

[24] Stephen Wolfram. Cellular automata. *Los Alamos Science*, 9(2-21):42, 1983.

[25] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[26] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 241–246. IEEE, 2016.

[27] Núbia Rosa da Silva, Jan M Baetens, Marcos William da Silva Oliveira, Bernard De Baets, and Odemir Martinez Bruno. Classification of cellular automata through texture analysis. *Information Sciences*, 370:33–49, 2016.