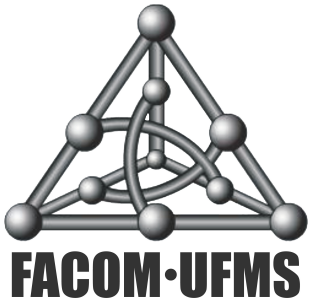
# Inteligência Artificial e Bioinformática

## Relatório de Atividade Orientada de Ensino

Thaís Dos Santos Garcez Maia

**Orientação: Prof. Dr. Carlos Henrique Aguena Higa**

Curso: Sistemas de Informação




Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Campo Grande, julho de 2024

# 1 Inteligência Artificial e Bioinformática

O problema de *inferência de redes de regulação gênica* na Bioinformática é considerado um problema inverso mal posto (*ill-posed inverse problem*), ou seja, existem várias soluções possíveis (redes) que podem explicar um conjunto de dados observado. Sendo assim, não existe um algoritmo que é o melhor em todos os sentidos; cada um tem suas vantagens e desvantagens dependendo do caso estudado. Com isso, podemos observar a necessidade de desenvolver diferentes metodologias para resolver esse tipo de problema.

Os algoritmos para inferência de redes de regulação gênica são basicamente utilizados para entender as interações entre os genes a partir de dados disponibilizados por biólogos, químicos, bioquímicos, médicos, entre outros. Portanto, uma grande motivação para se estudar tal problema é a capacidade do mesmo de gerar hipóteses que possam ser verificadas por esses profissionais. Tais hipóteses podem levar a um melhor entendimento acerca dos próprios genes e do comportamento dos organismos de modo geral.

Nos anos 90, desenvolveu-se uma tecnologia conhecida como *microarrays*, que permitia analisar uma grande quantidade de expressões gênicas simultaneamente. No entanto, os microarrays eram caros e os dados gerados possuíam ruídos, inerentes da tecnologia aplicada. Em meados dos anos 2000, com o desenvolvimento dos sequenciadores de nova geração (NGS), uma nova ferramenta surgiu, possibilitando a análise de transcriptomas; essa técnica é conhecida como RNA-Seq e é utilizada até hoje, servindo de fonte de dados para vários problemas tratados na Bioinformática. O RNA-Seq permitiu a análise de expressões gênicas de maneira mais confiável e mais acessível. Com o aperfeiçoamento dos equipamentos de sequenciamento, desenvolveu-se uma nova técnica conhecida como *single cell RNA-Seq*, ou scRNA-Seq. Os microarrays e RNA-Seq analisam o transcriptoma de uma grande população de células e assim, tal análise, pode não deixar claro as diferenças entre células individuais nesta população. Já o scRNA-Seq provê o perfil de expressão gênica de células individuais e é considerado o estado da arte no que diz respeito à análise de estados celulares e fenótipos. Temos, cada vez mais, dados sendo gerados, não só em quantidade mas também em qualidade. O desenvolvimento das tecnologias citadas neste parágrafo possibilitou o estudo de inferência de redes de regulação gênica a partir de técnicas encontradas na área de Inteligência Artificial, mais especificamente Deep Learning. Dito isso, em nosso trabalho focamos no uso de uma rede neural conhecida como rede neural convolucional (*convolutional neural network*, ou CNN) para a inferência de redes. Apresentamos mais detalhes quanto às redes neurais e seu funcionamento a seguir.

Uma vez que Inteligência Artificial é a teoria e desenvolvimento de sistemas computacionais capazes de realizar tarefas que normalmente precisam de inteligência humana, *Machine Learning* se preocupa em dar aos sistemas a habilidade de "pensar" para resolver tais tarefas. Com esse intuito, algoritmos são usados para desempenhar tarefas específicas sem necessariamente serem programados para tal,

ao invés disso, eles reconhecem padrões nos dados e fazem predições quando novos dados lhes são apresentados. Basicamente, Machine Learning é o termo geral para quando nos referimos a computadores aprendendo através de dados, e tal processo de aprendizado pode ser supervisionado ou não supervisionado. No primeiro caso, os dados de entrada são previamente rotulados para que seja possível medir a precisão do método utilizado; no segundo caso, deseja-se identificar padrões a partir dos dados, sem a busca de um atributo alvo específico.

Neste contexto, o conceito de *Deep Learning* se apresenta como uma evolução dessa capacidade dos computadores de aprender e realizar tarefas de forma mais eficiente, trazendo algoritmos que analisam dados com uma estrutura lógica similar ao pensamento humano. Para atingir tal comportamento, as aplicações de Deep Learning utilizam uma estrutura de algoritmos em camadas chamada de rede neural artificial (*artificial neural network*, ou ANN). A ideia por trás dessa rede neural artificial é fortemente inspirada pela rede de neurônios que temos em nosso cérebro, levando a um processo de aprendizado muito mais capaz do que os modelos de Machine Learning.

## 1.1 Redes Neurais

De acordo com [2], uma rede neural é apenas uma coleção de unidades conectadas; as propriedades da rede são determinadas pela sua topologia e pelas propriedades dos "neurônios".

Um neurônio é uma unidade de processamento de informação que é fundamental para a operação de uma rede neural. A Figura 1 mostra o modelo de um neurônio, que forma a base para o projeto de redes neurais (artificiais). Aqui nós identificamos três elementos básicos do modelo neuronal:

1. Um conjunto de *sinapses* ou *elos de conexão*, cada uma caracterizada por um *peso* ou *força* própria. Especificamente, um sinal $x_j$ na entrada da sinapse $j$ conectada ao neurônio $k$ é multiplicado pelo peso sináptico $w_{kj}$. É importante notar a maneira como são escritos os índices do peso sináptico $w_{kj}$. O primeiro índice se refere ao neurônio em questão e o segundo se refere ao terminal de entrada da sinapse à qual o peso se refere. O peso sináptico de um neurônio artificial pode estar em um intervalo que inclui valores negativos bem como positivos.

2. Um *somador* para somar os sinais de entrada, ponderados pelas respectivas sinapses do neurônio; as operações descritas aqui constituem um *combinador linear*.

3. Uma *função de ativação* para restringir a amplitude da saída de um neurônio. A função de ativação é também referida como *função restritiva* já que restringe (limita) o intervalo permissível de amplitude do sinal de saída a um valor finito.

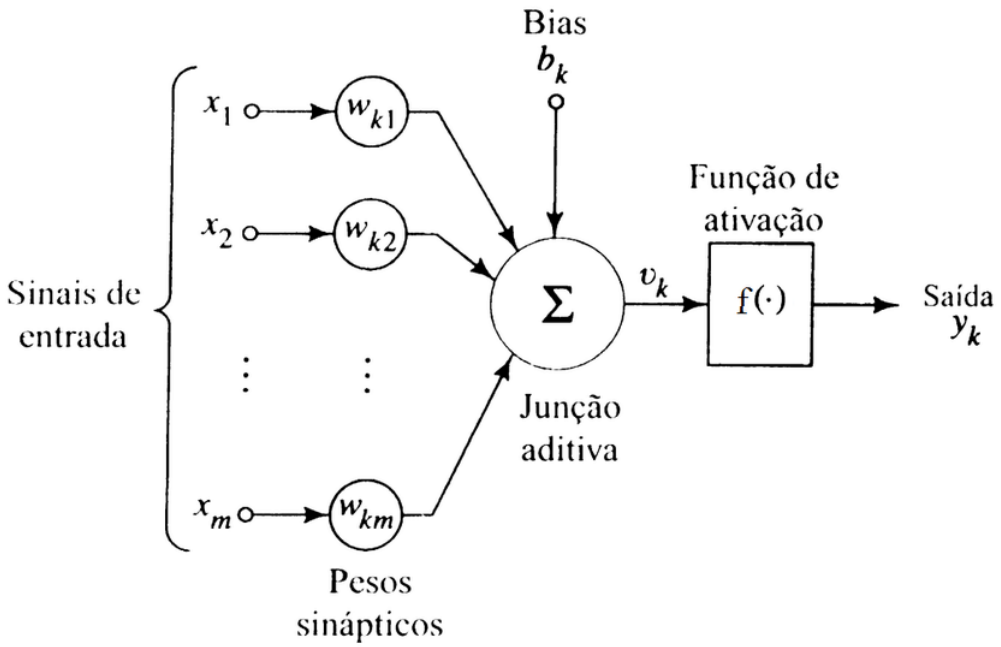



Figura 1: Modelo não-linear de um neurônio (HAYKIN, 2001).

O modelo neuronal da Figura 1 inclui também um *bias* aplicado externamente, representado por $b_k$. O bias $b_k$ tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente [1].

Em termos matemáticos, podemos descrever um neurônio $k$ escrevendo o seguinte par de equações:

$$u_k = \sum_{j=1}^{m} w_{k,j} x_j \tag{1}$$

e

$$y_k = f(u_k + b_k) \tag{2}$$

onde $x_1, x_2, ..., x_m$ são os sinais de entrada; $w_{k1}, w_{k2}, ..., w_{km}$ são os pesos sinápticos do neurônio $k$; $u_k$ é a saída do combinador linear devido aos sinais de entrada; $b_k$ é o bias; $f(.)$ é a função de ativação; e $y_k$ é o sinal de saída do neurônio. O uso do bias $b_k$ tem o efeito de aplicar uma transformação afim à saída $u_k$ do combinador linear no modelo da Figura 1, como mostrado por:

$$v_k = u_k + b_k \ . \tag{3}$$

Em particular, dependendo se o bias $b_k$ é positivo ou negativo, a relação entre o potencial de ativação $v_k$ do neurônio $k$ e a saída do combinador linear $u_k$ é modificada [1].

Com base no modelo de um neurônio, podemos definir uma rede neural como um

grafo orientado constituído de nós com elos de interligação sinápticos e de ativação, sendo caracterizada por quatro propriedades:

1. Cada neurônio é representado por um conjunto de elos sinápticos lineares, um bias aplicado externamente e um elo de ativação possivelmente não-linear.

2. Os elos sinápticos de um neurônio ponderam os seus respectivos sinais de entrada.

3. A soma ponderada dos sinais de entrada define o potencial de ativação do neurônio em questão.

4. O elo de ativação limita o potencial de ativação do neurônio para produzir uma saída [1].

Uma vez que as redes neurais são compostas por vários nós (neurônios), existem duas formas distintas de conectá-los: através de uma rede com alimentação para a frente (*feedforward neural network*) e através de uma rede recorrente (*recurrent neural network*). A primeira forma tem conexões somente em uma direção, isto é, forma um grafo acíclico dirigido. Cada nó recebe a entrada de nós antecessores e libera a saída para nós sucessores; não há laços. Uma rede com alimentação para a frente representa uma função de sua entrada atual; portanto, não tem estado interno que não seja os próprios pesos. A rede recorrente, por outro lado, alimenta suas saídas de volta às suas próprias entradas. Isso significa que os níveis de ativação da rede formam um sistema dinâmico que pode atingir um estado estável ou apresentar oscilações ou até mesmo um comportamento caótico. Além disso, a resposta da rede para determinada entrada depende do seu estado inicial, que pode depender de entradas anteriores. Portanto, as redes recorrentes (ao contrário das redes com alimentação para a frente) podem suportar memória de curto prazo. Isso as torna mais interessantes como modelos de cérebro, mas também mais difícil de serem compreendidas [2].

A propriedade que é de importância primordial para uma rede neural é a sua habilidade de *aprender* a partir de seu ambiente e de *melhorar* o seu desempenho através da aprendizagem. A melhoria do desempenho ocorre com o tempo de acordo com alguma medida preestabelecida. Uma rede neural aprende acerca do seu ambiente através do treinamento, um processo interativo de ajustes aplicados a seus pesos sinápticos e níveis de bias. Idealmente, a rede se torna mais instruída sobre o seu ambiente após cada iteração do processo de aprendizagem. [1].

Considere um neurônio $k$ que constitui o único nó computacional da camada de saída de uma rede neural alimentada adiante (*feedforward*), como ilustrado pela Figura 2. O neurônio $k$ é acionado por um vetor de sinal $x(n)$ produzido por uma ou mais camadas de neurônios ocultos, que são, por sua vez, acionadas por um vetor de entrada (estímulo) aplicado aos nós de fonte (camada de entrada) da rede neural. O argumento $n$ representa o instante de tempo discreto, ou mais precisamente, o passo de tempo de um processo iterativo enolvido no ajuste dos pesos sinápticos do

neurônio $k$. O sinal de saída do neurônio $k$ é representado por $y_k(n)$. Este sinal de saída, representando a única saída da rede neural, é comparado com uma resposta desejada ou saída-alvo, representada por $d_k(n)$. Consequentemente, é produzido um sinal de erro, representado por $e_k(n)$. Por definição, temos

$$e_k(n) = d_k(n) - y_k(n) \ . \tag{4}$$

O sinal de erro $e_k(n)$ aciona um *mecanismo de controle*, cujo propósito é aplicar uma sequência de ajustes corretivos aos pesos sinápticos do neurônio $k$. Os ajustes corretivos são projetados para aproximar passo a passo o sinal de saída $y_k(n)$ da resposta desejada $d_k(n)$, e o método mais utilizado para isso é o algoritmo de *backpropagation*. Este objetivo é alcançado minimizando-se uma *função de custo* ou *índice de desempenho*, $E(n)$, definido em termos do sinal de erro $e_k(n)$ como:

$$E(n) = \frac{1}{2} e_k^2(n) \ . \tag{5}$$

Com isso, $E(n)$ é o valor instantâneo da energia do erro. Os ajustes passo a passo dos pesos sinápticos do neurônio $k$ continuam até o sistema atingir um estado estável (*i.e.*, os pesos sinápticos estão essencialmente estabilizados). Neste ponto, o processo é encerrado [1]. Em particular, a minimização da função de custo $E(n)$ resulta na

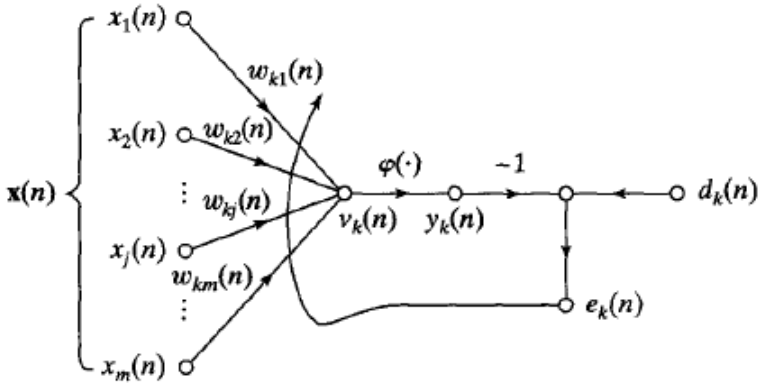



Figura 2: Ilustração da aprendizagem por correção de erro (HAYKIN, 2001).

regra de aprendizagem normalmente referida como *regra delta*. Suponha que $w_{jk}(n)$ represente o valor do peso sináptico $w_{kj}$ do neurônio $k$ excitado por um elemento $x_j(n)$ do vetor de sinal $x(n)$ no passo de tempo $n$. De acordo com a regra delta, o ajuste $\Delta w_{kj}(n)$ aplicado ao peso sináptico $w_{kj}$ no passo de tempo $n$ é definido por

$$\Delta w_{kj}(n) = \alpha e_k(n) x_j(n) \tag{6}$$

onde $\alpha$ é uma constante positiva que determina a taxa de aprendizado quando avançamos em um passo no processo de aprendizagem [1].

Tendo calculado o ajuste sinaptico $\Delta w_{kj}(n)$, o valor atualizado do peso sináptico

$w_{kj}$ é determinado por

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{7}$$

onde $w_{kj}(n)$ e $w_{kj}(n+1)$ podem ser vistos como os valores *antigo* e *novo* do peso sináptico $w_{kj}$, respectivamente [1].

Existem diversos tipos de redes neurais, cada qual com uma diferente topologia voltada a um tipo de problema específico. Entretanto, para fins práticos, nesse trabalho nos ateremos a apenas um tipo de rede, a rede neural convolucional. Tal rede é bastante eficiente quando se busca analisar imagens, processo usado pelo algoritmo estudado.

## 1.2 Redes Neurais Convolucionais

O reconhecimento de imagem é um clássico problema de classificação, e as redes neurais convolucionais (*convolutional neural networks* - CNN) possuem um histórico de alta acurácia para esse problema, tendo atualmente diversas aplicações práticas em nosso cotidiano. Essas redes identificam certos padrões nas imagens e as classificam sem precisar de tanto pré-processamento quanto outros algoritmos de classificação. Enquanto que em métodos mais primitivos os filtros são feitos à mão, com treinamento suficiente, as redes neurais convolucionais possuem a habilidade de aprender esses filtros por conta própria.

Quando falamos em reconhecimento ou classificação de imagens, as entradas são usualmente matrizes tridimensionais com altura e largura (de acordo com as dimensões da imagem) e profundidade, determinada pela quantidade de canais de cores. Em geral as imagens utilizam três canais, RGB, com os valores de cada pixel. Uma vez que as imagens podem atingir grandes dimensões, exigindo assim muito poder computacional, a rede convolucional tenta reduzir as imagens em um formato mais fácil de se processar, sem perder características que são importantes para conseguir uma boa predição.

### 1.2.1 Camadas convolucionais

Para realizar esse processo de filtragem, a rede possui as chamadas convoluções, que funcionam como filtros que enxergam pequenos quadrados e vão "passando" por toda a imagem captando os traços mais marcantes. Por exemplo, vamos considerar uma imagem de $32 \times 32 \times 3$, onde a altura e a largura têm 32 pixels, com 3 canais de cores. Suponha um filtro (também chamado de *kernel*) de $5 \times 5 \times 3$, onde a altura e a largura têm 5 pixels, com obrigatoriamente a mesma profundidade da imagem (nesse caso 3). Deslizando-se o filtro sobre toda a área da imagem, com movimento de 1 salto (chamado de *stride*), teremos uma matriz convoluída de tamanho $28 \times 28 \times 1$, pois a cada passo, é calculado o somatório do produto de todos os valores sobrepostos resultando em um único valor. Esse processo pode ser visto na imagem 3.

A matriz resultante representa um *feature map* ou *activation map*, e dependendo de sua dimensão e dos valores presentes é possível se obter características diferentes nas imagens como: suavização, deslocamento, contraste, bordas, texturas, dentre outras. A profundidade da saída de uma convolução é igual a quantidade de filtros aplicados. Quanto mais profundas são as camadas das convoluções, mais detalhados são os traços identificados com o activation map. O filtro, ou kernel, é formado por pesos inicializados aleatoriamente, atualizando-os a cada nova entrada durante o processo de backpropagation. A pequena região da entrada onde o filtro é aplicado é chamada de *receptive field*.

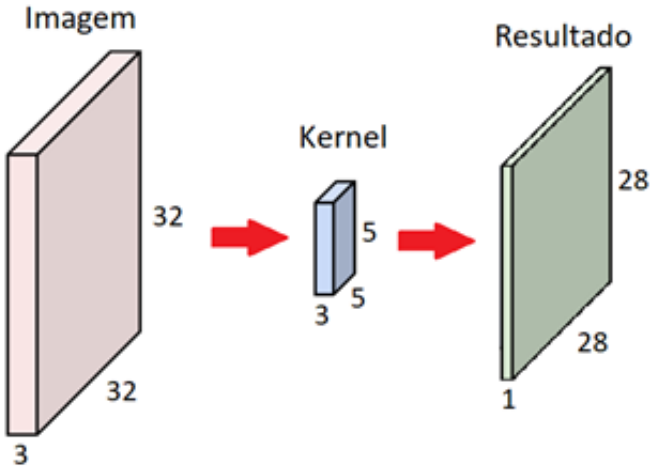



Figura 3: Entrada de dimensão $32 \times 32 \times 3$ com um filtro de $5 \times 5 \times 3$.

A convolução tem o papel de fazer uma filtragem para extração de informações de interesse na imagem. Geralmente a primeira camada convolucional é responsável por capturar os atributos de baixo nível como arestas, cores, etc. Com camadas consecutivas, a arquitetura se adapta também aos atributos de alto nível, em outras palavras, a rede começa a entender melhor a sofisticação das imagens do conjunto.

Além do tamanho do filtro e o stride da convolução como hiperparâmetro, ao modelar uma CNN deve-se também escolher como será o *padding*. O padding é a inclusão de mais pixels nas bordas da imagem original e serve para que as camadas não diminuam muito mais rápido do que é necessário para o aprendizado. O padding pode ser nenhum, no qual a saída da convolução ficará no seu tamanho original, ou *zero pad*, onde as bordas são adicionadas e preenchidas com 0's.

### 1.2.2 Função de ativação

As funções de ativação servem para trazer a não-linearidade ao sistema, para que a rede consiga aprender qualquer tipo de funcionalidade. Há muitas funções para isso, como *Sigmoid*, *Tanh* e *Softmax*, porém a mais indicada para as redes convolucionais é a função de ativação *ReLU* (*rectified linear unit*), por ser mais eficiente computacionalmente sem grandes diferenças de acurácia quando comparada

a outras funções. Essa função zera todos os valores negativos da saída da camada anterior:

$$ReLU(x) = \begin{cases} x & \text{se} \quad x \geq 0 \\ 0 & \text{se} \quad x < 0 \end{cases} . \tag{8}$$

### 1.2.3 Camadas de pooling

Uma camada de *pooling* serve para simplificar a informação da camada anterior. Assim como na convolução, é escolhida uma unidade de área, por exemplo $2 \times 2$, para transitar por toda a saída da camada anterior. A unidade é responsável por resumir a informação daquela área em um único valor. Se a saída da camada anterior for $4 \times 4$, a saída do pooling será $2 \times 2$. Além disso, é preciso escolher como será feita a sumarização. Semelhante a camada convolucional, a camada de pooling é responsável por reduzir o tamanho espacial do atributo convolucionado. Isto serve para diminuir o poder computacional necessário para processar os dados através de redução da dimensionalidade. Além disso, esse processo é útil para extrair características dominantes que são invariantes rotacionais e posicionais, mantendo assim o processo de treinamento efetivo do modelo.

O método mais utilizado é o *maxpooling*, no qual apenas o maior número da unidade é passado para a saída, como mostrado na Figura 4. Essa sumarização de dados serve para diminuir a quantidade de pesos a serem aprendidos e também para evitar *overfitting* (quando o modelo se adapta muito a um conjunto específico de treino). Maxpooling também funciona como um supressor de ruído, uma vez que descarta completamente as ativações ruidosas e também realiza diminuição de ruído juntamente a redução de dimensionalidade.

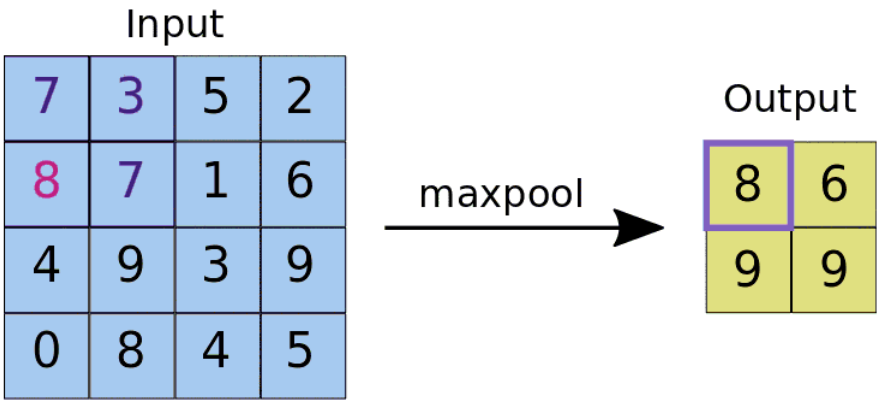



Figura 4: Exemplo da função maxpooling com área $2 \times 2$.

### 1.2.4 Camadas totalmente conectadas

Ao final da rede é colocada uma camada totalmente conectada (*fully connected*), onde sua entrada é a saída da camada anterior e sua saída são $N$ neurônios, com $N$

sendo a quantidade de classes do seu modelo para finalizar a classificação. Adicionar uma camada totalmente conectada é geralmente uma maneira fácil de aprender combinações não lineares dos atributos de alto nível, como os representados pela saída da camada convolucional.

Após termos convertido nossa imagem de entrada em uma forma mais adequada para nossa camada de classificação, devemos "achatar" a imagem através de uma camada *flatten*, transformando-a em um vetor de uma dimensão. A saída achatada é alimentada a uma rede neural com alimentação para a frente e é aplicado back-propagation em cada iteração do treinamento. Ao longo de uma série de épocas (*epochs*) o modelo é capaz de distinguir entre características dominantes e certas características de baixo nível em imagens e classificá-las usando alguma técnica de classificação. Geralmente, a técnica mais utilizada é a função *softmax*, que transforma as saídas para cada classe para valores entre 0 e 1 e também divide pela soma das saídas. Isso essencialmente dá a probabilidade de a entrada estar em uma determinada classe.

# 2 Conclusão

No artigo *Deep learning for inferring gene relationships from single-cell expression data* [3] temos um estudo em que a Inteligência Artificial, mais especificamente Deep Learning, foi utilizada para tratar o problema de inferência de redes gênicas.

Usando técnicas de Deep Learning para tratar o problema, o autor apresenta como podemos analisar dados em nível celular com adaptações das redes neurais convolucionais (CNN).

Na pesquisa é aplicado o sequenciamento de RNA de células únicas (scRNA-Seq) juntamente com a capacidade de aprendizado e generalização das redes CNN, obtendo novas percepções sobre a estrutura e a dinâmica de redes de regulação gênica (GRN).

Como resultado, é obtido uma maior qualidade das inferências e uma base para pesquisas e aplicações na biologia computacional.

Com a combinação apresentada obtemos uma evolução na Bioinformática tornando possível uma análise com maiores detalhes dos dados de expressão gênica. Essa visualização abre portas para maiores explorações de redes gênicas. Podendo obter maior entendimento dos mecanismos moleculares que regulam os organismos.

# Referências

[1] Simon Haykin. *Redes neurais: princípios e prática*. Bookman Editora, 2001.

[2] Stuart J Russell and Peter Norvig. *Inteligência artificial*. Elsevier, 2004.

[3] Ye Yuan and Ziv Bar-Joseph. Deep learning for inferring gene relationships from single-cell expression data. *Proceedings of the National Academy of Sciences*, 116(52):27151–27158, 2019.