# Performance Analysis of Deep Neural Networks in piRNAs Classification

**Alisson Hayasi da Costa**[1], **Renato Augusto Corrêa dos Santos**[2], **Ricardo Cerrid**[1]

[1]Departamento de Computação – Universidade Federal São Carlos
Rod. Washington Luís km 235 – São Carlos – SP – Brasil

[2]Faculdade de Ciências Farmacêuticas de Ribeirão Preto – Universidade de São Paulo
Avenida do Café s/n – Ribeirão Preto – SP - Brasil

{hayasi.ahs,renatoacsantos}@gmail.com, cerri@ufscar.br

***Abstract.*** *Modern machine learning techniques, such as Deep Learning, have been successful in many complex Bioinformatics tasks. The capacity of Deep Neural Networks to handle large volumes of data has made them essential tools for multiple areas of knowledge. However, developing the best model for a given task is a hard work. Deep Neural Networks have a very large number of hyperparameters, making them as powerful as complex to be adjusted. Therefore, in order to better understand the behavior of Deep Neural Networks when applied to biological data, we present in this paper a performance analysis of a Deep Feedforward Network in piRNAs classification. Different configurations of activation functions, initialization of weights, number of layers and learning rate are experienced. The effects of different hyperparameters are discussed and certain organizations are proposed for similar domains of data.*

***Resumo.*** *Técnicas modernas de Aprendizado de Máquina, como Deep Learning, têm sido bem-sucedidas em diversas tarefas complexas da Bioinformática. A grande capacidade das Redes Neurais Profundas de lidar com imensos volumes de dados fez delas ferramentas essenciais para múltiplas áreas do conhecimento. Contudo, desenvolver o melhor modelo para uma tarefa em questão é um trabalho difícil. Redes Neurais Profundas possuem um número muito grande de hiperparâmetros, tornando-as tão poderosas quanto complexas de serem ajustadas. Portanto, a fim de melhor compreender o comportamento das Redes Neurais Profundas quando aplicadas em dados biológicos, apresentamos neste trabalho uma análise do desempenho de uma Deep Feedforward Network na classificação de piRNAs. São experimentadas diferentes configurações de funções de ativação, inicialização de pesos, número de camadas, gradiente e taxa de aprendizagem. São discutidos os efeitos dos diferentes hiperparâmetros e propostas certas organizações para domínios semelhantes de dados.*

## 1. Introdução

Um RNA não codificante (ncRNA) é uma molécula de ácido ribonucleico (RNA) não traduzida em proteínas. Entre as diversas classes de RNAs não codificantes existentes, os PIWI-interacting RNAs (piRNAs) formam o maior grupo de sequências de ncRNAs, cujo comprimento em geral varia de 23 a 36 nucleotídeos [Han and Zamore 2014]. São responsáveis pela regulação gênica, manutenção das células germinativas e expressos

principalmente nas gônadas animais, onde trabalham no silenciamento epigenético de elementos transponíveis (também denominados transposões ou *transposons*), mantendo a integridade do genoma. Por esta razão, os piRNAs desempenham um papel indispensável para a fertilidade das espécies [Hirakata and Siomi 2016, Iwasaki et al. 2015].

Já os elementos transponíveis (TEs) são sequências de ácido desoxirribonucleico (DNA) capazes de se locomover de uma região a outra no genoma. Tal mobilidade pode gerar mutações que provocam instabilidade genética e, consequentemente, levam ao desenvolvimento de doenças cancerígenas, no caso de humanos, e outras complicações biológicas, fazendo dos piRNAs não apenas indispensáveis para a fertilidade, mas também, elementos fundamentais para a ocorrência do câncer e seu tratamento [Moyano and Stefani 2015, Specchia 2017, Assumpcao et al. 2015].

Desenvolver ferramentas para a classificação de piRNAs é um trabalho complexo. Isto porque, os piRNAs aparentam não dispor da conservação de estruturas secundárias, o que torna a utilização de métodos baseados em estruturas duvidosos [Zhang et al. 2011]. Além disso, o intervalo de comprimento das sequências é muito semelhante ao de outros ncRNAs como miRNAs (microRNAs) e siRNAs (small interfering RNA), dificultando a criação de estratégias baseadas em tal aspecto [Zhang et al. 2011, Wang et al. 2014]. Portanto, dadas tais complicações, a predição de piRNAs mostra-se um problema nitidamente adequado para analisar o desempenho dos modelos de *Deep Learning* (DL) em diferentes ajustes de hiperparâmetros, assim como, o impacto de tais variações na capacidade de predição.

Técnicas de *Deep Learning* são intensamente utilizadas para a execução de diversos tipos de tarefas complexas. Alguns exemplos são tradução de máquina, transcrição de dados, síntese e reconhecimento de fala, entre outras [Goodfellow et al. 2016]. Modelos de DL como as *Convolutional Neural Networks* (CNNs) alcançaram o estado da arte no campo da Visão Computacional, possibilitando o desenvolvimento de softwares capazes de localizar e classificar objetos em imagens com desempenho surpreendente [van Doorn 2014].

Ao decorrer do sucesso crescente desses novos algoritmos de Aprendizado de Máquina (AM), muitos resultados foram obtidos pelas diversas arquiteturas nas mais variadas classes de problemas, trazendo a tona uma grande quantidade de estudos sobre os melhores modelos e técnicas de treinamento em problemas específicos. Esses estudos foram feitos por meio de uma profunda análise crítica sobre todos os elementos que definem um modelo de *Deep Learning* com bom desempenho para uma dada tarefa (como é o caso da classificação de imagens). No trabalho de [Glorot and Bengio 2010], por exemplo, foram investigados os efeitos das diferentes funções de ativação em conjunto com a propagação e saturação do gradiente. Através dessa investigação, métodos de inicialização dos pesos sinápticos (conexões sinápticas) e formas eficientes de se realizar o treinamento de *Deep Neural Networks* (DNNs) foram sugeridas, sendo empregues e aprimoradas em diversas situações até os dias de hoje.

Poucos são os estudos realizados sobre os modelos de DL na aplicação de problemas biológicos. Os trabalhos relevantes mais próximos são de [Min et al. 2017] e [Angermueller et al. 2016]. Porém, ambos apresentam somente uma compilação, com breves comentários, dos resultados obtidos pelos métodos de DL em diferentes tarefas

da Bioinformática como, localização de proteínas no meio celular, predição de função de proteínas e análise de expressão gênica. Todavia, os resultados são apenas comentados, não havendo um diagnóstico crítico sobre o impacto dos hiperparâmetros de cada modelo no desempenho final. Além disso, não há métodos de classificação de piRNAs utilizando DNNs na literatura até o momento, o que torna imprescindível a necessidade de aplicação de uma abordagem tão bem sucedida, como é o *Deep Learning*, tanto no desenvolvimento de um classificador eficiente quanto no estudo do mesmo.

Neste trabalho, portanto, é apresentada uma análise sobre os impactos das diferentes configurações de hiperparâmetros de uma *Deep Feedforward Network* (DFN) na classificação de piRNAs. São experimentadas múltiplas funções de ativação, técnicas de inicialização de pesos, número de camadas intermediárias e taxas de aprendizado. Os resultados obtidos são avaliados abordando fatores como a capacidade de generalização, *overfitting*, *underfitting* e curva de aprendizado.

## 2. Metodologia

No contexto do Aprendizado de Máquina, um modelo com bom desempenho é aquele cujo o erro de generalização é baixo. Tal objetivo pode ser alcançado ao se reduzir o erro de treinamento – *underfitting* – e a diferença entre o erro de treinamento e o erro de teste – *overfitting* – através da minimização de uma função de custo [Goodfellow et al. 2016, Deng and Yu 2014]. Tanto o *underfitting* quanto o *overfitting* podem ser controlados através da capacidade do modelo de se ajustar a diversas funções. Modelos com baixa capacidade tendem a resultar em *underfitting*, já modelos com alta capacidade podem acabar em *overfitting*.

Visto que os valores definidos para os hiperparâmetros prescrevem uma capacidade ao modelo, é necessário um bom ajuste para se alcançar a minimização do erro de generalização [Goodfellow et al. 2016, Deng and Yu 2014, Schmidhuber 2014]. No entanto, quando as tarefas em questão são muito complexas, o ajuste correto de hiperparâmetros tende a não ser o suficiente. Por esta razão, torna-se necessário o uso de técnicas de regularização, como, por exemplo, o *Dropout* [Srivastava et al. 2014].

### 2.1. Dropout

O *Dropout* é uma das técnicas de regularização mais poderosas para modelos de *Deep Learning*. Seu funcionamento consiste basicamente em remover neurônios aleatoriamente durante a fase de treinamento, evitando uma coadaptação exagerada e forçando-os a aprender representações mais robustas sobre dados. Com isso, a ocorrência de *overfitting* é reduzida significativamente e o desempenho dos modelos em tarefas de aprendizado supervisionado é aprimorado [Srivastava et al. 2014, Goodfellow et al. 2016].

Desse modo, neste estudo foi implementada uma *Deep Feedforward Network* com diferentes ajustes de função de ativação, técnica de inicialização de pesos, número de camadas e taxa de aprendizado. Para todas as variações, foi implementada entre todas as camadas intermediárias uma camada de *Dropout* com $50\%$ de probabilidade para o desligamento de neurônios. Contudo, para verificar e comparar o impacto da técnica na capacidade de generalização e na ocorrência de *overfitting* das variações, também foram realizados testes sem *Dropout*.

## 2.2. Número de Neurônios e Camadas

A quantidade de camadas e neurônios definem, juntos, a complexidade e dimensionalidade da DNN onde, quanto maior o valor de ambos os hiperparâmetros, maior sua complexidade. Entretanto, se a DNN projetada for mais complexa do que o problema em questão, problemas de *overfitting* são cada vez mais prováveis. Em contra partida, se a DNN for projetada com uma dimensão muito pequena, menor sua capacidade e maiores as chances de *underfitting*. Sendo assim, é necessário escolher um número de neurônios e camadas que compreenda a complexidade da tarefa [Gama et al. 2011, Goodfellow et al. 2016].

A fim de analisar tal aspecto, a DFN foi implementada com variações de $3$ e $5$ camadas intermediárias, nas quais foram fixados $340$ neurônios (o mesmo número de neurônios da camada de entrada (Seção 2.6)). Além disso, como a tarefa em questão se trata de uma classificação binária, apenas um neurônio foi definido na camada de saída.

## 2.3. Funções de Ativação

Duas funções de ativação foram adotadas para experimentação, a função logística e a *Rectified Linear Unit* (*ReLU*). A escolha da logística (Equação 1) se deu graças a sua capacidade de lidar com dados esparsos e, quando fixada na camada de saída, realizar predições binárias. Porém, a função logística também possui alguns defeitos que tendem a prejudicar o treinamento de DNNs com muitas camadas, como, a saturação dos neurônios. [Glorot and Bengio 2010, Haykin 2009].

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

Já a escolha da *Rectified Linear Unit* (Equação 2) se deu pelo fato de ser, atualmente, o estado da arte das funções de ativação para DNNs. Seu comportamento permite com que os neurônios aprendam muito mais rapidamente (quando comparado com a logística) sem saturar, evitando o problema de desaparecimento do gradiente e, consequentemente, permitindo redes neurais com um número muito grande de camadas intermediárias [He et al. 2015, Hinton 2010].

$$f(x) = \begin{cases} 0, \text{se } x \leq 0 \\ x, \text{se } x > 0 \end{cases} \tag{2}$$

Assim, dadas as características de ambas ativações, para cada variação da DFN foram aplicadas as funções logística e *ReLU*. Contudo, em todos os casos, no neurônio de saída foi definida a ativação logística exatamente por se tratar de um problema de classificação binária e seu comportamento ser útil para tal.

## 2.4. Estratégias de Inicialização de Pesos

Da mesma maneira que as funções de ativação foram escolhidas, também foram definidas estratégias para determinar o intervalo de inicialização dos pesos sinápticos (conexões sinápticas). Dois métodos foram adotados, a inicialização de Glorot (ou, Xavier) [Glorot and Bengio 2010] e a inicialização de He [He et al. 2015].

A inicialização de Glorot foi desenvolvida para a função de ativação logística sigmoide, e sua variante tangente hiperbólica. Essa estratégia determina um intervalo plausível para que os pesos sejam inicializados por todas as camadas da rede neural, reduzindo os problemas de saturação do neurônio e o desaparecimento do gradiente. Para que isso aconteça, [Glorot and Bengio 2010] estabelece que as conexões devem ser inicializadas em uma distribuição uniforme na forma $[-a, a]$, sendo $a$ dado pela Equação 3, onde $n_{in}$ e $n_{out}$ representam, respectivamente, o número de conexões de entrada e saída do neurônio $n$ [Glorot and Bengio 2010, He et al. 2015].

$$a = \sqrt{\frac{6}{n_{in} + n_{out}}} \tag{3}$$

A estratégia de [He et al. 2015] é uma extensão do método de [Glorot and Bengio 2010], porém, voltada para a aplicação em funções *ReLU*. Neste caso, a inicialização é baseada em uma distribuição uniforme da forma $[-a, a]$, sendo $a$ dada pela Equação 4 [Glorot and Bengio 2010, He et al. 2015, Deng and Yu 2014].

$$a = \sqrt{\frac{6}{n_{in}}} \tag{4}$$

A partir das ênfases de cada estratégia, em todos os neurônios cuja função de ativação era logística, foi adotada a inicialização de Glorot. Da mesma maneira, a inicialização de He foi adotada para as variações que utilizaram a função de ativação *ReLU*.

### 2.5. Taxa de Aprendizado e Gradiente Adaptativo

A taxa de aprendizado (aprendizagem) é o hiperparâmetro responsável por controlar o grau de ajuste das conexões sinápticas ao decorrer das épocas de treinamento. Quanto menor o valor, mais lento é o aprendizado e a rede neural pode acabar não convergindo. Por outro lado, se o valor for muito alto, a velocidade acelerada pode causar uma oscilação na superfície de erro, provocando um aprendizado instável e um modelo sem capacidade de generalização. Portanto, o ideal é utilizar a maior taxa de aprendizado possível que não leve à oscilação [Gama et al. 2011, Goodfellow et al. 2016, Deng and Yu 2014]. Para auxiliar nesse processo, diversos gradientes adaptativos foram desenvolvidos. O *Adaptative Gradient* (*Adagrad*), por exemplo, é um algoritmo de otimização que realiza atualizações menores para padrões frequentes e atualizações maiores para padrões menos frequentes sendo, por esse motivo, excepcional para lidar com dados esparsos.

Com o intuito de analisar o impacto das diferentes taxas de aprendizado junto do *Adaptative Gradient*, dois valores foram escolhidos. Uma taxa de aprendizado igual a $0.01$ cujo valor é o recomendado pelos autores do gradiente [Duchi et al. 2010] e um valor igual $0.02$ a fim de verificar o comportamento das variações para uma taxa de aprendizado maior do que o recomendado. A Equação 5 apresenta como é realizada a atualização dos pesos através do *Adagrad* [Duchi et al. 2010, Goodfellow et al. 2016].

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i} \tag{5}$$

Na Equação 5, $w_i^{(t+1)}$ e $w_i^{(t)}$ são os valores da conexão $i$ nos instantes $(t+1)$ e $t$, respectivamente. A constante $\eta$ representa a taxa de aprendizagem e $g$ o valor do gradiente.

### 2.6. Dados

O conjunto de dados utilizado para os experimentos consiste em um total de $27996$ exemplos de RNAs não codificantes da espécie *Mus musculus* dos quais, $13998$ são piRNAs derivados de elementos transponíveis (exemplos positivos) e $13998$ são pseudo-piRNAs (exemplos negativos). Os dados foram obtidos por meio do material suplementar contido no artigo [Luo et al. 2016], onde é possível encontrar detalhes de sua elaboração. A Tabela 1 apresenta a quantidade de exemplos positivos, negativos e totais.

| | Quantidade |
|---|---|
| Exemplos Positivos | 13998 |
| Exemplos Negativos | 13998 |
| Total de Exemplos | 27996 |

**Tabela 1. Conjunto de dados para os experimentos**

A partir dos dados coletados, atributos foram extraídos utilizando a ferramenta *Pse-in-One-2.0* (em sua versão local) [Liu et al. 2017]. Como atributos, foram utilizados $k$-mers. Dada uma sequência, é calculada a ocorrência de ácidos nucleicos vizinhos de tamanho $k$ que diferem em $m$ posições da sequência original. Foram extraídos 340 atributos considerando as seguintes combinações de $m$ e $k$: $m = 0$ e $k = 1$, $m = 1$ e $k = 2$, $m = 1$ e $k = 3$, $m = 1$ e $k = 4$. A escolha desse atributo se deu devido ao seu habitual uso para diversas tarefas de classificação na Bioinformática e também porque trata-se de um atributo independência de fatores puramente estruturais [Zhang et al. 2011]. Por fim, os atributos passaram por um processo de padronização (*standardization*) com média $0$ e desvio padrão $1$ (processo realizado por meio do software *Weka* [Hall et al. 2009]). Tal filtro reduz a esparsidade dos dados possibilitando com que as DNNs aprendam mais rápido. Para uma descrição detalhada dos atributos, o leitor pode referir-se a [Liu et al. 2017].

### 2.7. Implementações

Todos os experimentos foram realizados utilizando a linguagem *Python 3.6.6* tendo como apoio as bibliotecas *keras* [Chollet et al. 2015], *theano* [Theano Development Team 2016], *scikit-learn* [Pedregosa et al. 2011] e *numpy* [Travis E 2006].

## 3. Análises e Resultados

Para os experimentos, o conjunto de dados total foi dividido em dois subconjuntos disjuntos, cada um com 50% dos exemplos, sendo um subconjunto (S1) dedicado ao estudo das variações dos hiperparâmetros e o outro subconjunto (S2) dedicado ao teste e comparação com diferentes modelos de classificação da literatura. A estratégia de validação adotada para a análise do impacto das variações foi a *Holdout*, portanto, o subconjunto S1 foi dividido em 70% para treino e 30% para validação. Já para as comparações e testes sobre

o subconjunto S2, foi utilizada a estratégia *K-Fold Cross Validation*, cujo valor escolhido para $k$ foi $10$. Além disso, a função de custo definida para minimização foi o Erro Quadrático Médio (EQM) (Equação 6). Logo, os valores de erro apresentados correspondem à taxa do EQM.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

### 3.1. Análise dos Dados

O conjunto de dados utilizado possui características que podem dificultar a capacidade de generalização dos modelos. Por exemplo, tanto os exemplos positivos quanto os exemplos negativos são muito parecidos em seu comprimento de nucleotídeos, como mostra a Figura 1. Além disso, mesmo após a padronização aplicada sobre os atributos, ainda há uma esparsidade muito grande nos exemplos, dificultando a convergência da rede neural profunda. A Figura 2 apresenta a variação de alguns atributos do conjunto de dados antes e após a padronização.

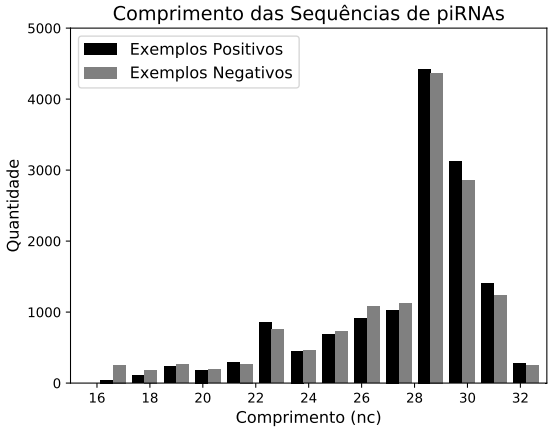
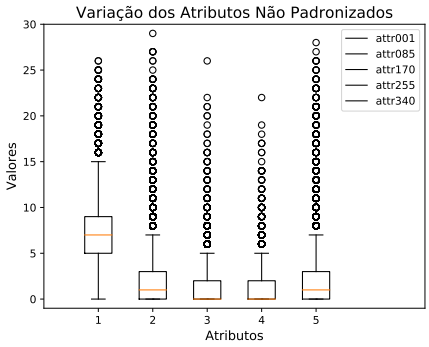
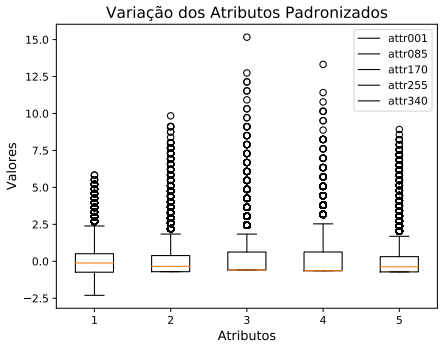



**Figura 1. Comprimento das sequências em quantidade de nucleotídeos**




(a) Variação de valores dos exemplos não padronizados




(b) Variação de valores dos exemplos padronizados

**Figura 2. Variação dos exemplos utilizados no conjunto de dados**

### 3.2. Impacto das Variações de Hiperparâmetros

A Tabela 2 apresenta a média e o desvio padrão da taxa de erro obtida pelas diferentes variações através do *Holdout* no subconjunto S1. A tabela apresenta oito variações de redes neurais, sendo C.I. o número de camadas intermediárias e T.A. a taxa de aprendizado utilizada.

| Modelo | C.I. | Ativação | T.A. | Erro (Treino) | Erro (Validação) |
|---|---|---|---|---|---|
| DFN1 | 3 | ReLU | 0.01 | $0.091 \pm 0.042$ | $0.131 \pm 0.021$ |
| DFN2 | 3 | ReLU | 0.02 | $0.250 \pm 0.051$ | $0.258 \pm 0.044$ |
| DFN3 | 3 | Logística | 0.01 | $0.109 \pm 0.016$ | $0.126 \pm 0.005$ |
| DFN4 | 3 | Logística | 0.02 | $0.083 \pm 0.023$ | $0.130 \pm 0.005$ |
| DFN5 | 5 | ReLU | 0.01 | $0.265 \pm 0.043$ | $0.269 \pm 0.039$ |
| DFN6 | 5 | ReLU | 0.02 | $0.495 \pm 0.010$ | $0.510 \pm 0.000$ |
| DFN7 | 5 | Logística | 0.01 | $0.110 \pm 0.017$ | $0.126 \pm 0.006$ |
| DFN8 | 5 | Logística | 0.02 | $0.084 \pm 0.027$ | $0.131 \pm 0.011$ |

**Tabela 2. Desempenho das diferentes variações durante o processo de treinamento e validação**

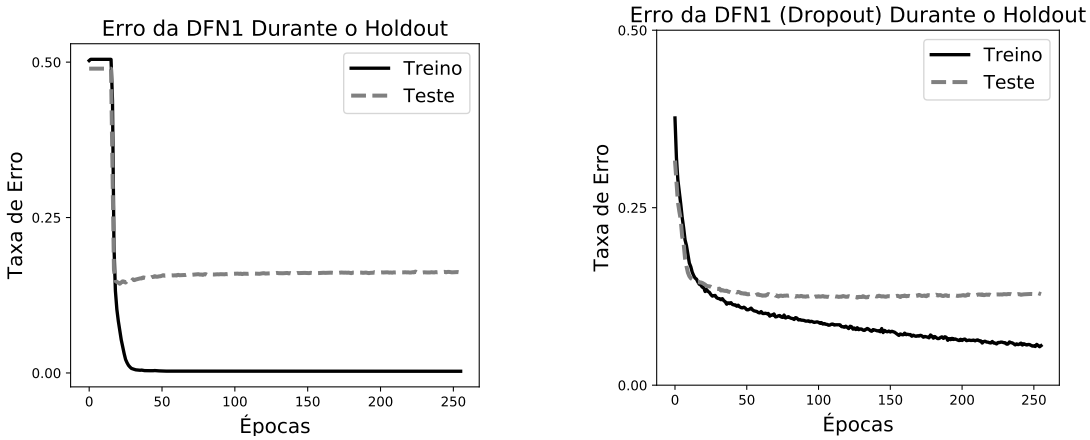
Como dito na Seção 2.3, a função de ativação *ReLU* permite uma velocidade de aprendizado muito superior a logística, sem saturar os neurônios. Tal característica faz com que o ajuste incorreto da taxa de aprendizado e/ou número de camadas prejudique a capacidade de generalização da *DNN* até uma configuração cujo aprendizado é totalmente instável (DFN6). Além disso, apesar de seus defeitos, a função de ativação logística lida muito melhor com a esparsidade dos dados em questão do que a função *ReLU*, visto que, em todas as variações apresentadas, as DFNs com ativação logística se mantiveram com uma boa métrica de erro e não sofreram instabilidade.

Os impactos de um modelo excessivamente complexo também podem ser observados ao se remover as camadas de *Dropout* das DFNs. Como apresentado na Tabela 3, a remoção da técnica de regularização provoca um *gap* entre o erro de treinamento e validação significativamente mais alto se comparado com a Tabela 2, e um grande aumento no desvio padrão para alguns casos. Além disso, salvo a DFN1, todos os outros modelos com ativação *ReLU* tornam-se completamente instáveis e incapazes de aprender. Isso demonstra como todas as implementações são altamente suscetíveis ao *overfitting* e, graças a isso, a utilização do *Dropout* torna-se imprescindível.

Outro aspecto muito importante é a curva de aprendizado. A partir dela é possível detectar a velocidade de aprendizado e convergência da rede, assim como, a ocorrência tanto de *underfitting* quanto de *overfitting*. Na Figura 3(a), o modelo apresenta a ocorrência de *underfitting* nas primeiras épocas de treinamento e, logo após, uma curva de aprendizado extremamente acentuada onde a curva de treinamento alcança uma taxa de erro excepcionalmente baixa. Entretanto, o mesmo não ocorre com a curva de validação que em certo ponto sofre um pequeno aumento na taxa de erro e se mantém. Ao final das épocas de treinamento, a diferença entre o erro de treinamento e o erro de validação é muito alta, caracterizando, portanto, a ocorrência de *overfitting*. Já a Figura 3(b) apresenta o comportamento da curva de aprendizado para o modelo quando este possui camadas de *Dropout*. Neste, nota-se que não há um superajustamento ao dados como ocorre com a Figura 3(a).

| Modelo | Erro (Treino) | Erro (Validação) |
|---|---|---|
| DFN1 | $0.038 \pm 0.124$ | $0.180 \pm 0.080$ |
| DFN2 | $0.504 \pm 0.001$ | $0.489 \pm 0.004$ |
| DFN3 | $0.026 \pm 0.031$ | $0.159 \pm 0.011$ |
| DFN4 | $0.023 \pm 0.062$ | $0.163 \pm 0.036$ |
| DFN5 | $0.504 \pm 0.002$ | $0.490 \pm 0.004$ |
| DFN6 | $0.495 \pm 0.010$ | $0.510 \pm 0.000$ |
| DFN7 | $0.035 \pm 0.038$ | $0.166 \pm 0.013$ |
| DFN8 | $0.174 \pm 0.216$ | $0.277 \pm 0.155$ |

**Tabela 3. Desempenho das diferentes variações sem *Dropout***




(a) Desempenho da DFN1 sem Dropout

(b) Desempenho da DFN1 com Dropout

**Figura 3. Diferenças de desempenho entre a DFN1 com Dropout e sem Dropout, evidenciando a importância da técnica de regularização contra o overfitting**

### 3.3. Desempenho Através da K-Fold Cross Validation

O modelo com menor erro de treinamento e menor diferença entre erro de treinamento e teste foi a DFN3. Dessa forma, submetendo tal modelo ao processo de 10-Fold Cross Validation no subconjunto de teste (S2), podemos analisar a qualidade de sua predição para os seus hiperparâmetros. A Tabela 4 apresenta a média aritmética e o desvio padrão obtido pela DFN3 através das métricas acurácia, sensibilidade e especificidade. A técnica de validação cruzada *K-Fold* é muito eficaz para avaliar a capacidade de generalização de um modelo pois o submete a diversos cenários de treinamento e teste exclusivos. Logo, resultados com um baixo desvio padrão como é caso da DFN3 na Tabela 4, mostram que a DFN é capaz de criar uma representação robusta sobre os dados de treinamento e obter um bom desempenho em diversos casos de testes diferentes.

Através da curva *Receiver Operating Characteristic* (ROC), podemos analisar graficamente a qualidade da predição do modelo. Em um gráfico ROC, quanto mais próxima a curva está da borda superior esquerda, mais precisa é a classificação. Assim, partir do gráfico da Figura 4, é possível observar o baixo desvio padrão e a acentuada curva obtida para todos os 10 casos de teste, comprovando uma boa capacidade de generalização da DNN.

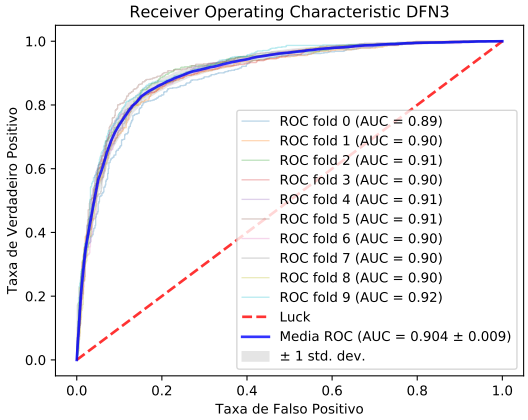



**Figura 4. Curva ROC da DFN3 Durante a 10-Fold Cross Validation**

### 3.4. Comparação com Outros Métodos da Literatura

Estratégias relevantes da literatura utilizaram técnicas de Aprendizado de Máquina para a predição de piRNAs. O método apresentado por [Luo et al. 2016], por exemplo, implementa uma estratégia de *Ensemble Learning* utilizando *Random Forests* e combinação de *features* cujo desempenho é satisfatório. Outro método recentemente desenvolvido com bom desempenho é o *GA-WE* de [Li et al. 2016] onde, é implementada uma estratégia utilizando algoritmos genéticos para a tarefa de classificação.

A Tabela 4 apresenta o desempenho das diferentes estratégias de classificação sobre o conjunto de teste (S2) incluindo, a DFN3. Os resultados apresentados são as médias aritméticas e o desvio padrão obtido pelos três métodos através da *10-Fold Cross Validation*. As métricas de avaliação utilizadas foram acurácia, sensibilidade e especificidade.

| Modelo | Acurácia | Sensibilidade | Especificidade |
|---|---|---|---|
| Ensemble Learning | $0.813 \pm 0.002$ | $0.854 \pm 0.002$ | $0.781 \pm 0.005$ |
| GA-WE | $0.823 \pm 0.002$ | $0.837 \pm 0.003$ | $0.810 \pm 0.004$ |
| DFN3 | $0.835 \pm 0.009$ | $0.853 \pm 0.017$ | $0.815 \pm 0.016$ |

**Tabela 4. Desempenho de diferentes métodos na classificação de piRNAs**

A partir dos resultados, é possível concluir que a utilização de uma estratégia bem sucedida, como é o caso do *Deep Learning*, na classificação de piRNAs, mostra-se não apenas eficiente, como também confiável quando os hiperparâmetros do modelo adotado estão bem ajustados, sendo capaz de se equiparar ou ultrapassar outros algoritmos de Aprendizado de Máquina.

## 4. Conclusões

Dada as características dos dados e os resultados obtidos, nota-se que os hiperparâmetros causam um enorme impacto na capacidade de generalização de uma DNN, sendo assim, necessário conhecer os dados e os impactos causados por cada configuração de hiperparâmetro, a fim de desenvolver uma boa solução para a tarefa proposta. Também é possível inferir que dependendo da função de ativação utilizada, a quantidade de camadas

e neurônios deve ser propriamente ajustada. Além disso, a função de ativação logística é muito menos suscetível a alterações graças ao seu aprendizado lento quando comparada com a *ReLU*, contudo, lidando muito melhor com dados esparsos. Por outro lado, a função *ReLU*, apesar do problema com a esparsidade dos dados, converge muito mais rapidamente do que a logística, requisitando assim muito menos custo computacional para alcançar um bom aprendizado. Também é ressaltado que a técnica de regularização *Dropout* se faz imprescindivelmente necessária contra a ocorrência de *overfitting*. Por fim, DFNs mostram-se extremamente capazes de realizar tarefas complexas não apenas em campos como Visão Computacional e Robótica, mas também, Bioinformática.

## Agradecimentos

Os autores gostariam de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), em especial o processo número 400549/2016-6.

## Referências

Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. *Molecular Systems Biology*, 12(7).

Assumpcao, C. B., Calcagno, D. Q., Araujo, T. M., Santos, S. E., Santos, A. K., Riggins, G. J., Burbano, R. R., and Assumpcao, P. P. (2015). The role of piRNA and its potential clinical implications in cancer. *Epigenomics*, 7(6):975–984.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Deng, L. and Yu, D. (2014). *Deep Learning: Methods and Applications*, volume 7.

Duchi, J., Hazan, E., and Singer, Y. (2010). Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley.

Gama, J., Faceli, K., Lorena, A., and De Carvalho, A. (2011). *Inteligência artificial: uma abordagem de aprendizado de máquina*. Grupo Gen - LTC.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. 9:249–256.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Han, B. W. and Zamore, P. D. (2014). pirnas. *Current Biology*, 24(16):R730 – R733.

Haykin, S. S. (2009). *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.

Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines vinod nair.

Hirakata, S. and Siomi, M. C. (2016). piRNA biogenesis in the germline: From transcription of piRNA genomic sources to piRNA maturation. *Biochim. Biophys. Acta*, 1859(1):82–92.

Iwasaki, Y. W., Siomi, M. C., and Siomi, H. (2015). Piwi-interacting rna: Its biogenesis and functions. *Annual Review of Biochemistry*, 84(1):405–433. PMID: 25747396.

Li, D., Luo, L., Zhang, W., Liu, F., and Luo, F. (2016). A genetic algorithm-based weighted ensemble method for predicting transposon-derived pirnas. *BMC Bioinformatics*, 17(1):329.

Liu, B., Wu, H., and Chou, K.-C. (2017). Pse-in-one 2.0: An improved package of web servers for generating various modes of pseudo components of dna, rna, and protein sequences. 09:67–91.

Luo, L., Li, D., Zhang, W., Tu, S., Zhu, X., and Tian, G. (2016). Accurate prediction of transposon-derived pirnas by integrating various sequential and physicochemical features. *PloS one*, 11(4):e0153268.

Min, S., Lee, B., and Yoon, S. (2017). Deep learning in bioinformatics. *Briefings in Bioinformatics*, 18(5):851–869.

Moyano, M. and Stefani, G. (2015). piRNA involvement in genome stability and human cancer. *J Hematol Oncol*, 8:38.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828.

Specchia, V. (2017). pirnas: the bodyguards of fertility. *Journal of RNA and Genomics*, 13(1).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Travis E, O. (2006). A guide to numpy. [Online; accessed ¡today¿].

van Doorn, J. (2014). Analysis of deep convolutional neural network architectures.

Wang, K., Liang, C., Liu, J., Xiao, H., Huang, S., Xu, J., and Li, F. (2014). Prediction of pirnas using transposon interaction and a support vector machine. *BMC Bioinformatics*, 15(1):419.

Zhang, Y., Wang, X., and Kang, L. (2011). A k-mer scheme to predict pirnas and characterize locust pirnas. *Bioinformatics*, 27(6):771–776.