Universidade de Brasília - UnB
Instituto de Ciências Exatas - IE
Departamento de Estatística - EST

Modelos de Aprendizado Profundo para Detecção de Planetas Extrassolares

Ricardo Torres Bispo Reis

Orientador: Prof. Dr. Donald Matthew Pianto

Brasília

2018

Ricardo Torres Bispo Reis

Modelos de Aprendizado Profundo para Detecção de Planetas Extrassolares

Monografia apresentada ao Departamento de Estatística, Instituto de Exatas, Universidade de Brasília, como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Orientador: Prof. Dr. Donald Matthew Pianto

Brasília

2018

Ricardo Torres Bispo Reis
Modelos de Aprendizado Profundo para Detecção de Planetas Extrassolares/ Ricardo Torres Bispo Reis. – Brasília, 2018-
56 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Donald Matthew Pianto

Relatório Final – Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Estatística
Trabalho de Conclusão de Curso de Graduação, 2018.

1. redes neurais convolucionais. 2. aprendizado profundo. 3. séries temporais. 4. detecção de exoplanetas.

Ricardo Torres Bispo Reis

Modelos de Aprendizado Profundo para Detecção de Planetas Extrassolares

Monografia apresentada ao Departamento de Estatística, Instituto de Exatas, Universidade de Brasília, como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Trabalho aprovado. Brasília, junho de 2018:

Prof. Dr. Donald Matthew Pianto
Orientador

Prof. Dr. Jhames Matos Sampaio
Membro da Banca

Prof. Dr. Eduardo Freitas da Silva
Membro da Banca

Brasília
2018

"The cosmos is all that is or ever was or ever will be.
Our feeblest contemplations of the cosmos stir us
- there is a tingling in the spine, a catch in the voice, a faint sensation,
as if a distant memory, of falling from a height.
We know we are approaching the greatest of mysteries."
(Carl Sagan)

Resumo

A busca e descoberta de novos planetas orbitando outras estrelas da Via Láctea vem com o passar dos anos produzindo quantidades de informação cada vez maiores, à medida que as missões espaciais e os observatórios crescem tecnologicamente. Com isso, a utilização de inteligência artificial para a detecção e classificação de exoplanetas tem se tornado crescente. Baseando-se nisto, este trabalho propõe a utilização de uma rede neural convolucional como método de detecção de exoplanetas baseado nas curvas de luz de suas estrelas. Foram ajustados três modelos baseados em formas diferentes de destacar características de interesse associadas aos trânsitos dos exoplanetas. Os testes dos modelos mostraram que, apesar da dos resultados satisfatórios da rede, a aplicação dos processamentos nos dados pode implicar em melhoria significativa de sua performance.

Palavras-chave: redes neurais convolucionais, aprendizado profundo, séries temporais, detecção de exoplanetas.

Abstract

The search for and discovery of new planets orbiting other stars in the Milky Way has produced an enormous amount of information as the space missions and observatories grow technologically. This fact has the use of artificial intelligence for detection and classification of exoplanets more common. This work proposes the use of a convolutional neural network as a means of exoplanet detection based on the light curve of its stars. Three models were fitted based on different ways of highlighting the features of interest associated with the exoplanets' transits. Despite the satisfactory results, the models tested showed that data processing can lead to significant improvements in performance.

Keywords: convolutional neural networks, deep learning, time series, exoplanet detection.

Lista de ilustrações

Figura 1 – Gráficos das funções de ativação sigmoide, tangente hiperbólica e retificadora linear. 27
Figura 2 – Curva de luz da estrela número 3197 sem exoplaneta confirmado do banco de dados de treinamento. 35
Figura 3 – Curva de luz da primeira estrela com exoplaneta confirmado do banco de dados de treinamento. 36
Figura 4 – Curva de luz da trigésima primeira estrela com exoplaneta confirmado do banco de dados de treinamento. 37
Figura 5 – Arquitetura da rede proposta. 40
Figura 6 – Curva de luz da primeira estrela com exoplaneta confirmado do banco de dados de treinamento com spline ajustado. 42
Figura 7 – Resíduos de um ARIMA(1, 0, 1) ajustado à trigésima primeira estrela do banco de treinamento. 43
Figura 8 – Curva PR dos modelos durante a validação. 44
Figura 9 – Acurácia dos modelo durante o treino e a validação. 45
Figura 10 – Perda dos modelos durante o treino e a validação. 46
Figura 11 – Curvas PR dos modelos durante os testes. 48
Figura 12 – Fluxo da quarta estrela com exoplaneta confirmado do banco de teste. 49

Lista de tabelas

Tabela 1 – Comparação do tempo de execução e AUC para redes com diferentes números de camadas. 41
Tabela 2 – Comparação do tempo e AUC de diferentes probabilidades de exclusão para a rede com 4 camadas de 16 filtros. 41
Tabela 3 – Matriz de confundimento do modelo 1 durante a validação. 43
Tabela 4 – Matriz de confundimento do modelo 2 durante a validação. 44
Tabela 5 – Matriz de confundimento do modelo 3 durante a validação. 44
Tabela 6 – Tabela resumo das medidas de desempenho dos três modelos durante os testes. 47
Tabela 7 – Matriz de confundimento do modelo 1 durante o teste. 47
Tabela 8 – Matriz de confundimento do modelo 2 durante o teste. 47
Tabela 9 – Matriz de confundimento do modelo 3 durante o teste. 48
Tabela 10 – Probabilidades estimadas de uma estrela apresentar exoplaneta. 48

Sumário

1 INTRODUÇÃO 17
1.1 Contextualização 17
1.2 Definição do Problema 18
1.3 Objetivos do Projeto 19

2 FUNDAMENTOS TEÓRICOS DE APRENDIZADO DE MÁQUINA 21
2.1 Aprendizado de Máquina 21
2.1.1 Aprendizado Supervisionado 21
2.1.2 Aprendizado Não-Supervisionado 22
2.2 Aprendizado Profundo 22

3 FUNDAMENTOS TEÓRICOS DE REDES NEURAIS ARTIFICIAIS 25
3.1 Redes Neurais Artificiais 25
3.1.1 Arquitetura de Redes Neurais 26
3.2 Funções de Ativação 26
3.3 Treinamentos de Redes Neurais 28
3.3.1 Funções de Custo 28
3.3.2 Método de Descida do Gradiente 29
3.4 Redes Neurais Convolucionais 30
3.4.1 Convolução 31
3.4.2 Arquitetura de Redes Neurais Convolucionais 32

4 METODOLOGIA 35
4.1 Base de Dados 35
4.2 Metodologia de Aplicação 36
4.3 Arquitetura Proposta 39
4.4 Modelos Propostos 41

5 RESULTADOS 47
5.1 Apresentação dos Resultados 47
5.2 Discussão dos Resultados 49

6 CONCLUSÃO 51

1 Introdução

1.1 Contextualização

Ao longo de toda a história humana, o universo exerceu grande fascínio, reflexão e curiosidade. Tamanha grandeza e mistério serviram de inspiração tanto para diversas obras artísticas quanto para importantes descobertas científicas.

Hoje em dia, a exploração do cosmo leva a humanidade a fronteiras outrora próximas apenas na imaginação. A descoberta de planetas orbitando outros sistemas estelares no século passado [24] trouxe novamente à tona a antiga busca por outros planetas similares à Terra e preencheu novamente o imaginário popular com as antigas questões relacionadas à exploração do universo e à busca de vida extraterrestre.

Devido a descobertas recentes [23], estima-se hoje que seja comum a ocorrência de sistemas planetários ao redor de outras estrelas na Via Láctea, especialmente de planetas pequenos. Porém a grande diferença de proporções entre planetas e estrelas torna muito difícil a tarefa de detectar esses objetos; em geral, planetas possuem tamanho e massa muito menor do que os de suas estrelas, além de o brilho da estrela ser muito superior em magnitude ao dos planetas que a circulam.

Existem várias técnicas que tornam possível a detecção de planetas orbitando estrelas distantes. A presença de um planeta causa uma leve oscilação periódica no movimento da estrela. Sendo assim essa técnica é mais precisa na a detecção de objetos massivos, em geral planetas gigantes gasosos proporcionais a Júpiter em ordem de grandeza. Essa diferença de movimento é em geral é muito pequena e de difícil detecção, o que torna essa uma medida de pouca confiabilidade.

Uma forma de detecção mais confiável é através da observação do espectro eletromagnético da estrela. As oscilações causadas pela presença de planetas ao redor, causam um pequeno desvio nas linhas espectrais da estrela, visto como efeito Doppler. Através dessa técnica, o primeiro sistema de exoplanetas com massa proporcional à da Terra foi confirmado em 1992 [24] orbitando uma estrela pulsar e o primeiro planeta orbitando uma estrela de massa próxima à massa do Sol foi descoberto em 1995 [8].

Outra técnica de detecção baseia-se no brilho de uma estrela. Um planeta pode ser visto por um observador externo ao passar na frente de sua estrela, supondo que estes estejam no mesmo plano orbital, em um fenômeno chamado ocultação. Durante esse trânsito, pode-se observar uma pequena queda de brilho na estrela causada pela passagem do planeta. Essa técnica é chamada de trânsito planetário e também permite o estudo de características dos planetas como a composição atmosférica, devido à refração da luz

incidente.

A Missão *Kepler*[1] foi lançada pela NASA juntamente com o telescópio espacial em março de 2009 com o objetivo primário de detectar o trânsito de exoplanetas com o tamanho próximo ao da Terra localizados nas zonas habitáveis de suas estrelas. Ela passou cerca de quatro anos monitorando mais de 150.000 estrelas na região das constelações de Cisne e Lira, excedendo seu tempo de duração nominal de três anos.

A missão K2 foi desenvolvida nos meses subsequentes ao término da missão *Kepler*, após a perda de duas rodas de reação, peças importantes na movimentação do telescópio espacial. A missão tornou-se operacional em maio de 2014 e consiste em uma série de campanhas com regiões-alvo de observação sugeridas pelos membros da comunidade científica. Essas regiões são distribuídas no plano de eclíptica, o plano definido pela trajetória do Sol na esfera celeste, cada uma com duração aproximada de 80 dias e limitada pelo ângulo do telescópio com o Sol. Até o presente momento[2], a missão *Kepler* foi responsável pela descoberta de 5.373 exoplanetas entre candidatos e confirmados, 30 deles proporcionais à Terra e localizados em zonas habitáveis, sendo 772 descobertos pela missão K2.

1.2 Definição do Problema

A grande quantidade de informação sendo gerada trouxe a necessidade de processamento de forma automática e precisa. Nos últimos anos, foram desenvolvidos vários projetos para classificar os dados e criar listas de candidatos a planetas. O projeto *Robovetter* [25], por exemplo, utiliza uma árvore de decisões para rejeitar falsos positivos com precisão que rivaliza a classificação humana. Outros projetos também fizeram uso de aprendizado de máquina para executar a tarefa, como por exemplo o projeto *Autovetter* [26], que faz uso de um modelo de floresta aleatória. Shallue e Vanderburg [10] utilizaram uma rede neural convolucional para classificar sinais em planetas em transito ou falsos positivos e foram capazes de detectar dois novos planetas orbitando as estrelas *Kepler-80* e *Kepler-90*.

Percebe-se então que modelos automatizados possuem grande potencial na classificação de sinais astronômicos. Em especial, modelos de aprendizado profundo, por permitirem à máquina aprender características dos dados por meio de representação, vêm tornando-se ótimos candidatos para tarefas do tipo.

[1] https://keplerscience.arc.nasa.gov/objectives.html
[2] https://www.nasa.gov/kepler/discoveries

1.3 Objetivos do Projeto

Utilizando o contexto da aplicação de aprendizado profundo para a classificação e detecção de exoplanetas como motivação para o estudo de redes neurais profundas, este trabalho possui como objetivo geral a aplicação de uma rede neural convolucional que seja capaz de predizer se uma estrela contém ou não um planeta em sua órbita com base em sua curva de luminosidade.

A partir do objetivo geral, define-se os objetivos específicos como: *i)* detectar pontos de interesse nas curvas de luminosidade que possam representar a passagem de um planeta; *ii)* apresentar uma arquitetura de rede neural convolucional capaz de classificar se uma estrela possui um planeta em sua órbita com base na sua curva de luminosidade; *iii)* treinar modelos com base em filtros aplicados aos dados para evidenciar as características de interesse das curvas; *iv)* avaliar a eficácia da rede considerando os modelos propostos.

Para tanto este trabalho apresenta um resumo dos conceitos de aprendizado de máquina e de aprendizado profundo no segundo capítulo. No terceiro capítulo, são introduzidos conceitos de redes neurais artificiais e o seu funcionamento. O quarto capítulo discorre sobre a metodologia e os dados utilizados na aplicação dos modelos. Por fim, o quinto capítulo apresenta os resultados obtidos da aplicação e uma análise.

2 Fundamentos Teóricos de Aprendizado de Máquina

2.1 Aprendizado de Máquina

Aprendizado de máquina é o termo que descreve o conjunto de métodos derivados de aplicações de estatística em análise de dados. Ele se refere à capacidade de sistemas de inteligência artificial de extrair padrões a partir de bancos de dados e aprender suas características. O enfoque recente na capacidade computacional e na manipulação de grandes conjuntos de dados desfavoreceu a necessidade de obter intervalos de confiança sobre as distribuições de probabilidade [2].

A definição dada por Mitchell [4] descreve um algoritmo de aprendizado de máquina como um programa que melhora a performance de uma tarefa com o ganho de experiência, de acordo com uma medida de performance. Em outras palavras, um algoritmo de aprendizado de máquina deve executar uma tarefa com maior desempenho, medido por uma função de custo, por meio das características extraídas do conjunto de dados. Algoritmos de aprendizado de máquina podem executar vários tipos de tarefas, como classificação, regressão e tradução.

Duas abordagens principais em aprendizado de máquina são o aprendizado supervisionado e o aprendizado não-supervisionado.

2.1.1 Aprendizado Supervisionado

O aprendizado supervisionado é utilizado quando o conjunto de dados pode ser dividido entre um conjunto de características X e um conjunto de respostas associadas Y. Baseado nesta divisão, o programa deve retornar uma previsão da resposta condicionada aos valores de entrada. Essa previsão é feita assumindo-se que existe uma associação dos valores de Y em função dos valores de X, $p(y|x)$, que é estimada durante a fase de treinamento do programa. O termo "supervisionado" é baseado no ponto de vista de o algoritmo sendo provido com o conjunto de respostas Y como um guia a execução da tarefa. O tipo da variável resposta pode ser numérico ou categórico, a depender da tarefa a ser executada pelo programa. Alguns exemplos desse tipo de aprendizado são os problemas de Regressão e as Redes Neurais Artificiais [1].

2.1.2 Aprendizado Não-Supervisionado

O aprendizado não-supervisionado é utilizado quando o conjunto de dados é composto apenas do conjunto de características X, não contendo valores ou rótulos Y associados. Dessa forma, o algoritmo tenta extrair alguma estrutura com base apenas nos dados de entrada. Para isso o programa tenta estimar a distribuição de probabilidades do conjunto de características $p(x)$ a partir dos dados de entrada. Algumas das aplicações deste tipo de aprendizado são a Análise de *Clusters* e a Análise de Fatores Latentes [2].

Na prática, problemas de aprendizado supervisionado e não-supervisionado podem ser comumente relacionados. Por exemplo, da Lei de Multiplicação de Probabilidades tem-se que o problema de aprendizado não-supervisionado de estimar a probabilidade conjunta de um vetor de características $\mathbf{x}$ pode ser decomposto na forma

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_1, ..., x_{i-1}) \quad , \tag{2.1}$$

de tal sorte que cada probabilidade condicional $p(x_i|x_1, ..., x_{i-1})$ representa um problema de aprendizado supervisionado de estimar a probabilidade de x_i com base nas características anteriores. De forma similar, pode-se estimar a distribuição conjunta $p(\mathbf{x}, y)$ por meio de técnicas de aprendizado não-supervisionado para então inferir $p(y|\mathbf{x})$ através do Teorema de Bayes,

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_i p(\mathbf{x}, y_i)} \quad . \tag{2.2}$$

Além do aprendizado supervisionado e do aprendizado não-supervisionado, existem outras abordagens em aprendizado de máquina, como o aprendizado por reforço e o aprendizado semi-supervisionado. Na primeira, o programa recebe informação com base nas decisões tomadas ao invés de receber como entrada um conjunto de dados fixo, enquanto que na segunda há uma mistura das abordagens de aprendizado supervisionado e não-supervisionado, como para o caso de o programa receber dados parcialmente rotulados como entrada [1].

Os métodos de aprendizado de máquina também podem ser descritos pelo número de parâmetros a serem estimados na fase de treinamento, sendo ditos métodos paramétricos os que envolvem um número fixo de parâmetros e não-paramétricos os que dependem do tamanho do conjunto de dados fornecido [2].

2.2 Aprendizado Profundo

Aprendizado profundo é uma abordagem de inteligência artificial que parte do princípio de que, para diversos tipos de conceitos, pode-se entende-los desmembrando-os em conceitos mais simples, montando uma estrutura de aprendizado que os relaciona de forma hierárquica [3]. O tipo de abordagem, na qual o modelo é capaz de aprender

sozinho características do conjunto de dados, apresentado nessa estrutura é chamado de Aprendizado por Representação. Dessa forma, um modelo computacional é capaz extrair feições e identificar padrões existentes para resolver problemas mais abstratos de forma mais simples e segmentada.

De acordo com essa estrutura, um modelo de aprendizado profundo é composto de múltiplas camadas. Cada camada é responsável pela extração de características do conjunto de dados, de tal forma que as características mais simples são utilizadas pelas camadas seguintes para a extração de características mais complexas. Por exemplo, uma tarefa onde o programa precisa reconhecer um dígito escrito à mão [12] pode ser simplificada extraindo-se características mais primárias que compõem os dígitos, como bordas específicas e em seguida padrões específicos, que serão utilizadas para classificar a imagem de entrada baseado na probabilidade de ela conter tais características.

3 Fundamentos Teóricos de Redes Neurais Artificiais

3.1 Redes Neurais Artificiais

Para problemas não lineares, é necessário a utilização de modelos que levam em consideração as interações entre características de um conjunto de dados. Em especial, programas que envolvem dados de entrada com muitas dimensões, como classificação de vídeos, tornam-se muito complicados para a aplicação de modelos lineares.

Redes Neurais Artificiais (ANN - *Artificial Neural Network*, em inglês) são métodos de modelagem capazes de estimar relações lineares e não-lineares, inclusive, entre as características de um conjunto de dados, cuja estrutura foi inspirada em estudos de redes neurais biológicas em mamíferos [13].

Em redes neurais biológicas, os neurônios propagam informação entre si através de impulsos elétricos. Esses impulsos são recebidos de outros neurônios através dos dendritos e são propagados para outros neurônios por meio das terminações do axônio. ANNs são compostas em seu nível mais elementar por unidades, também chamadas de neurônios artificiais. De forma similar aos neurônios biológicos, cada neurônio artificial recebe como entrada as saídas de vários outros neurônios artificiais da camada anterior. Através dessa informação, o neurônio calcula um valor de saída que é alimentado como entrada aos neurônios da camada seguinte.

O i-ésimo neurônio artificial da j-ésima camada pode ser representado matematicamente na forma

$$a_i{}^{(j)} = g\left(\sum_k w_k{}^{(j-1)} a_k{}^{(j-1)} + b_i \right) \quad , \tag{3.1}$$

na qual $w_k{}^{(j-1)}$ representa o peso associado à k-ésima entrada do neurônio, $a_k{}^{(j-1)}$ representa a saída do k-ésimo neurônio da camada anterior, utilizada como entrada do neurônio subsequente, e b_i representa o viés, o valor para o qual a saída do i-ésimo neurônio tende na ausência de entradas. A função que recebe os parâmetros, g, é a chamada função de ativação, por meio da qual é introduzida a não-linearidade do modelo. O valor de saída de um neurônio artificial, $a_i{}^{(j)}$, é também referido como o nível de ativação desse neurônio.

Dessa forma, a operação realizada entre as camadas da rede pode ser escrita na forma matricial

$$\mathbf{a}^{(j)} = g\Big(\mathbf{W}\mathbf{a}^{(j-1)} + \mathbf{b} \Big) \quad , \tag{3.2}$$

na qual $\mathbf{a}^{(j)}$ é o vetor de neurônios da j-ésima camada da rede, $\mathbf{W}$ é o vetor de pesos associados com cada entrada da camada, $\mathbf{a}^{(j-1)}$ é o vetor de saídas da camada anteriore $\mathbf{b}$

é o vetor de vieses dos neurônios da camada. O conjunto de parâmetros a serem estimados por uma rede neural é composto pelo conjunto de pesos e vieses associados a cada camada, de modo que o número total de parâmetros a serem estimados pela rede seja geralmente muito alto.

3.1.1 Arquitetura de Redes Neurais

Os neurônios artificiais de uma rede são organizados em camadas, as quais compõem a rede neural. Cada camada da rede é responsável pela extração de um conjunto de característica do conjunto de dados, que são então utilizados na camada seguinte.

A estrutura de uma rede neural contém uma camada de entrada que recebe os dados, as camadas intermediárias responsáveis pela extração das características e uma camada de saída que retorna a resposta da rede. No exemplo de uma tarefa de classificação, a camada de saída retornaria como resposta a categoria mais provável à qual o conjunto de características dados como entrada pertencem. Por não mostrarem a saída específica para um conjunto de características dado como entrada da rede, as camadas intermediárias são também referidas como camadas escondidas.

O número de camadas que compõem uma rede neural define a profundidade da mesma; a redes com esse tipo de estrutura dá-se o nome de Redes Neurais Profundas. A capacidade da rede de representar funções não-lineares mais complexas também está diretamente ligada ao número de camadas e de neurônios. A forma com que as camadas se conectam determina a direção da propagação da informação. A estrutura de redes neurais descrita anteriormente é chamada de *feedforward*, por permitir a propagação da informação em sentido único: da camada de entrada para a camada de saída. Há ainda estruturas que permitem algum tipo de conexão das camadas finais novamente para o início da rede, como as Redes Recorrentes, usadas para processamento de dados sequenciais [3].

3.2 Funções de Ativação

Função de ativação, $g : \mathbb{R} \to \mathbb{R}$, é a função que determina o nível de ativação, ou valor de saída, de um neurônio artificial. Esse tipo de função limita o valor de saída do neurônio dentro de um intervalo específico de interesse. O uso de funções de ativação não-lineares permite que a rede seja capaz de representar modelos não-lineares, aumentando o número de funções que podem ser estimadas. De fato, redes neurais com múltiplas camadas adquirem a capacidade de aproximar qualquer tipo de função utilizando funções de ativação não-polinomiais [15].

Similar à atividade neuronal em sistemas biológicos [13], a função de ativação mais

simples consiste na função unitária

$$u(x) = \begin{cases} 1, & x \geq -b_i \\ 0, & x < -b \end{cases} \quad , \tag{3.3}$$

a qual transmite informação apenas quando essa ultrapassa o limite estabelecido pelo viés do neurônio artificial em questão. As funções de ativação mais comumente utilizadas na literatura, no entanto, são a função sigmoide, a função tangente hiperbólica e a função linear retificada, todas as três apresentando gradiente.

A função sigmoide, ou função logística, dada pela expressão

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad , \tag{3.4}$$

limita os valores de saída no intervalo $[0, 1]$, sendo que valores negativos muito grandes em módulo se tornam valores mais próximos de zero e valores positivos grandes em módulo se tornam valores próximos a um. Para valores intermediários, a função apresenta um comportamento crescente e contínuo, com o gráfico em forma de "S". Essa função é muito usada em problemas de regressão logística como classificação.

A função tangente hiperbólica, relacionada com a função sigmoide através da expressão

$$g(x) = tanh(x) = 2\sigma(x) - 1 \quad , \tag{3.5}$$

apresenta comportamento semelhante à sigmoide, porém no intervalo $[-1, 1]$, assumindo valores próximos a zero quando o valor do argumento da função também é próximo de zero.

A função retificadora linear é definida como a parte positiva de seu argumento e dada pela expressão

$$g(x) = max(0, x) \quad . \tag{3.6}$$

As unidades que utilizam essa função são chamadas de ReLU (*Rectified Linear Units*). A facilidade de computação e sua eficiência em métodos de otimização baseados em gradiente tornaram essa função amplamente utilizada em redes neurais profundas.

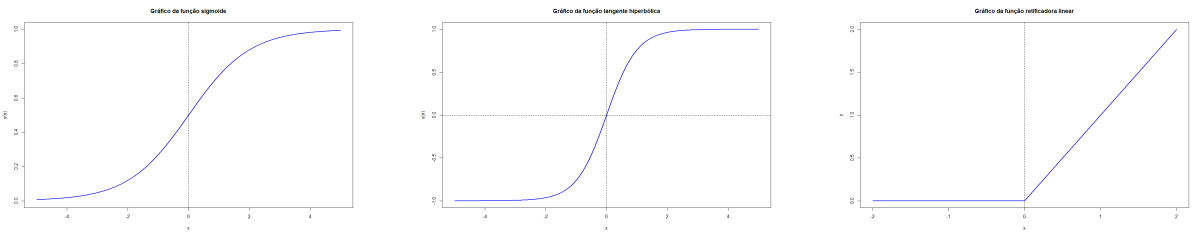

Figura 1 – Gráficos das funções de ativação sigmoide, tangente hiperbólica e retificadora linear.

3.3 Treinamentos de Redes Neurais

Em uma rede de aprendizado supervisionado os parâmetros do modelo são estimados durante a etapa de treinamento. Essa etapa consiste em apresentar à rede um conjunto de dados que serão utilizados no ajuste e comparar o resultado com um conjunto de rótulos associados aos dados de entrada, com o objetivo de que a saída da rede seja a mais próxima possível dos rótulos dos dados de entrada. Outras características associadas à estrutura da rede, como número de camadas e quantidade de unidades por camada, são chamadas hiper-parâmetros e são geralmente definidas antes da etapa de treinamento ou por meio de validação cruzada.

O ajuste do modelo será testado durante a etapa de teste, onde a rede será exposta a um novo conjunto de dados distinto do primeiro e nunca antes apresentado à rede. Sobre os conjuntos de dados de treino e de teste são necessárias certas suposições de que estes sejam independentes e gerados a partir da mesma distribuição de probabilidade, de forma que se possa controlar a performance do modelo sobre os dados de teste havendo-se observado apenas o conjunto de dados de treinamento.

Os nomes dados às medidas de erro associadas às saídas da rede durante as fases de treinamento e de teste são erro de treinamento e erro de teste, respectivamente. O maior interesse ao se ajustar um modelo de aprendizado de máquina é que não somente sua performance nos dados de treinamento seja boa, mas também que o modelo seja capaz de atuar bem quando receber um conjunto de dados nunca visto. Em outras palavras, deseja-se reduzir tanto o erro de treinamento quanto o erro de generalização, a diferença entre o erro de teste e o erro de treinamento. Quando um modelo cumpre de forma satisfatória esses dois requisitos diz-se que tem boa capacidade de generalização. Quando o modelo não apresenta erro de treinamento pequeno o bastante diz-se que houve subajuste (*underfit*). Quando a diferença entre o erro de teste e o erro de treinamento é muito grande houve sobreajuste (*overfit*).

Um método muito utilizado durante o treinamento de uma rede para prevenir sobre-ajuste é o *dropout* [29], que consiste simplesmente em reduzir o número de unidades ativas em cada camada, atribuindo a elas valor zero com uma probabilidade certa p e diminui a dependência entre elas na extração de características.

3.3.1 Funções de Custo

Para avaliar a precisão das estimativas de um modelo em relação aos verdadeiros rótulos é necessária uma medida de performance. A performance de um modelo geralmente é medida utilizando-se as funções de custo. Uma função de custo J, $J : \mathbb{R}^n \to \mathbb{R}$, ou risco, é definida como sendo o valor esperado da função perda entre os valores preditos pela saída da rede e os verdadeiros valores de Y, calculado em relação à distribuição dos dados.

A função custo é descrita pela fórmula

$$J(\theta) = \mathbb{E}\Big[L(\hat{y}, y)\Big] \quad , \tag{3.7}$$

na qual θ representa o vetor de parâmetros da rede, L é a função perda específica para cada exemplo, $\hat{y}$ é o valor predito pela rede e y é o verdadeiro rótulo associado ao conjunto de características de entrada. O valor da função de custo definido com respeito ao conjunto de dados de treinamento fornece o risco empírico e é calculado com o uso da distribuição empírica. Dessa forma, o problema de minimizar o erro médio de generalização pode ser resolvido minimizando a perda esperada nos dados de treinamento. Esse processo é conhecido como minimização do risco empírico.

Um exemplo de função de custo comumente utilizado, por exemplo em problemas de regressão, é a variância, com a função de perda nesse caso sendo a perda quadrática. No caso de problemas de classificação de redes neurais, uma das funções de custo mais utilizadas é a entropia cruzada

$$H(p, q) = -\mathbb{E}_p\Big[log(q)\Big] \quad , \tag{3.8}$$

na qual $\mathbb{E}_p\Big[log(q)\Big]$ é o valor esperado, com respeito à distribuição verdadeira dos dados, do logaritmo natural da distribuição dos valores preditos pelo modelo. Para o caso discreto, a função de entropia cruzada é dada por

$$H(p, q) = -\sum_x p(x) log\Big[q(x)\Big] \quad . \tag{3.9}$$

3.3.2 Método de Descida do Gradiente

Para minimizar o erro médio de generalização da rede neural, deseja-se ajustar os parâmetros de forma que a saída da rede produza valores preditos próximos aos valores reais dos dados. Dessa forma, é possível utilizar a função de custo para estimar os parâmetros da rede durante a etapa de treinamento. O método de descida do gradiente é um método de otimização que realiza essa tarefa em etapas graduais, calculando os parâmetros que resultam na direção na qual o gráfico da função decresce.

A direção de crescimento do gráfico de uma função é dada pelo seu gradiente, definido pelas derivadas parciais da função

$$\nabla g(\theta) = \left(\frac{\partial g}{\partial \theta_1}, \ldots, \frac{\partial g}{\partial \theta_n}\right) \quad . \tag{3.10}$$

O gradiente da função de custo calculado em um ponto qualquer indica a direção de crescimento dos parâmetros para que o valor da função aumente naquele ponto. Portanto, minimizar o valor da função de custo é equivalente a maximizar o negativo do gradiente da função. A magnitude do valor dos parâmetros que se deve alterar na direção de

decréscimo corresponde ao tamanho do "passo" que o algoritmo deve dar e em problemas de aprendizado de máquina é chamada de taxa de aprendizado, ε. A cada nova iteração do algoritmo, o gradiente da função de custo é calculado utilizando os novos valores dos parâmetros

$$\theta = \theta - \varepsilon \nabla J(\theta) \quad . \tag{3.11}$$

Quando a taxa de aprendizado é muito pequena, o número de iterações que o algoritmo deve realizar para chegar ao ponto de mínimo aumenta. Porém, se a taxa de aprendizado for muito grande, o algoritmo pode não convergir.

O método de descida do gradiente pode exigir um número muito grande de iterações se o conjunto de dados de entrada for muito grande. Com base nisso, o método apresenta ainda duas principais variações com respeito à quantidade de exemplos processados a cada iteração. Uma das variações do método mais usadas, especialmente em aprendizado profundo, é a descida do gradiente estocástico. Nela apenas uma pequena amostra de exemplos, chamada de *batch*, independentes e identicamente distribuídos é retirada e utilizada na operação. A partir dessa amostra é possível calcular um estimador não enviesado do gradiente da função de custo calculando-se o gradiente médio dos pontos amostrados. Devido ao ruído introduzido pela amostragem dos exemplos, é necessário diminuir a taxa de aprendizado a cada nova iteração para garantir a convergência do algoritmo.

Durante a fase conhecida como propagação, a rede estima os valores dos parâmetros seguindo o sentido da camada de entrada até a camada de saída e obtendo um erro de estimação associado ao valor calculado pela rede. Na fase seguinte, conhecida como retro propagação (*backpropagation*), a rede calcula a contribuição de cada parâmetro no erro de estimação. Essa contribuição é obtida ao propagar o erro da camada de saída da rede de volta até a camada de entrada, calculando o gradiente da função de custo em relação a cada parâmetro, e utilizada para atualizar o valor do parâmetro.

3.4 Redes Neurais Convolucionais

O aumento de camadas de uma rede neural amplia a sua capacidade de representação, porém traz consigo um aumento considerável da quantidade de parâmetros que devem ser estimados e do número de iterações que a rede deve executar. Por exemplo, uma rede neural com duas camadas escondidas de 16 unidades cada, uma camada de saída com apenas 2 unidades e que receba uma imagem de $32 \times 32 = 1.024$ *pixels* deverá estimar $1.024 \times 16 + 16 \times 16 + 16 \times 2$ pesos mais $16 + 16 + 2$ vieses, totalizando 16.706 parâmetros.

Redes Neurais Convolucionais (CNN - *Convolutional Neural Network*, em inglês) são redes com capacidade de estimar um número muito elevado de parâmetros e de considerar a topologia dos dados de entrada: observações muito próximas tendem a

ser altamente correlacionadas. Um exemplo de uma estrutura de dados bidimensional seria uma imagem, que representa uma grade de *pixels*. No caso unidimensional, pode-se considerar uma série temporal como uma grade de observações amostradas em intervalos de tempo regulares [3]. O grande sucesso das CNNs em classificação de vídeos [16] e imagens [14] vem popularizando seu uso em tempos recentes, porém o método também tem apresentado avanços significativos e destaque no estudo de outros tipos de aplicações como reconhecimento facial [20], reconhecimento de fala [18] e de atividade humana [19] e séries temporais [21] [10].

ANNs, no geral, descrevem a interação entre as entradas e saídas das unidades em suas camadas por meio de multiplicação matricial. CNNs são caracterizadas principalmente por utilizar a operação de convolução no lugar da multiplicação matricial. Uma ANN é dita convolucional quando emprega pelo menos uma camada que faz uso dessa operação.

3.4.1 Convolução

Convolução é um tipo de operador linear especializado que opera em duas funções, $f : \mathbb{R} \to \mathbb{R}$ e $g : \mathbb{R} \to \mathbb{R}$, com argumentos reais. A operação de convolução geralmente é denotada por um asterisco e é definida para funções contínuas como

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \quad , \tag{3.12}$$

na qual τ representa um deslocamento, e para funções discretas é definida como

$$s(t) = (f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau) \quad . \tag{3.13}$$

Em geral, a operação pode ser útil para vários tipos diferentes de aplicações e é definida para qualquer função onde a integral acima é definida [3]. No contexto de aprendizado de máquina, a operação de convolução é a operação de correlação cruzada discreta.

Na prática, usualmente assume-se que essas funções valem zero em todos os pontos a menos de um conjunto específico. Devido a isso, pode-se aplicar uma soma finita sobre esse conjunto específico de pontos. Também nesse contexto, no lugar das funções f e g usa-se as funções I, a entrada, e K, o núcleo (*kernel*, em inglês) ou filtro, respectivamente. A entrada e o filtro podem ser representados por matrizes que têm mais de uma dimensão:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \quad . \tag{3.14}$$

A operação consiste em deslocar o filtro pelos dados de entrada e calcular o produto interno em cada deslocamento. Dessa forma, o resultado da operação é um conjunto de características extraídas dos dados de entrada e é referido como mapa de características.

Por exemplo, aplicada em uma série temporal, a operação pode extrair características que não fazem parte da tendência da série como quedas abruptas. O uso de diferentes filtros resulta em diferentes tipos de características extraídas. Outro exemplo, no caso da aplicação em imagens, filtros específicos podem resultar em contornos específicos. As unidades da camada de convolução então utilizam os mapas de características para calcular a saída, realizando a detecção.

A operação de convolução é comutativa, o que significa que

$$S(i,j) = (I * K)(i,j) = (K * I)(i,j) \quad . \tag{3.15}$$

Em muitos casos implementa-se uma função relacionada, chamada correlação cruzada, definida por

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \quad . \tag{3.16}$$

3.4.2 Arquitetura de Redes Neurais Convolucionais

O desempenho superior das CNNs em relação ao das ANNs tradicionais em alguns tipos de tarefas vem principalmente da implementação de convolução nas camadas da rede. Em ANNs tradicionais a utilização de multiplicação de matrizes convencional faz com que cada saída de uma unidade se conecte com cada entrada das unidades na camada subsequente e, portanto, a rede precisa estimar um número muito grande de parâmetros. As camadas de convolução permitem que cada unidade faça menos conexões diretas e armazene menos parâmetros através de conexões esparsas. Isso se deve ao fato de o tamanho do filtro ser menor do que o tamanho da entrada, o que possibilita que a rede necessite executar menos operações. Interações complicadas entre variáveis podem ser descritas pela rede através de ligações indiretas de unidades entre camadas.

Outra característica da rede convolucional que a permite armazenar menos parâmetros e melhorar sua eficiência estatística, partindo do princípio de que certos padrões podem ser detectados mais de uma vez nos dados de entrada, é o compartilhamento de parâmetros. Como o filtro é deslocado por todos os dados, cada elemento seu é utilizado em cada posição da entrada. Isso permite que a rede aprenda apenas um conjunto de parâmetros e o compartilhe entre as camadas.

A maneira com a qual os parâmetros são compartilhados em uma camada de convolução possibilita que esta seja equivariante a deslocamentos, ou seja, a saída se modifica da mesma forma que a entrada. Neste caso, seja $T(x)$ uma função qualquer que provoca um deslocamento nos dados de entrada, a função de convolução $S(x)$ é dita equivariante a $T(x)$ se

$$S\Big(T(x)\Big) = T\Big(S(x)\Big) \quad . \tag{3.17}$$

Logo, aplicar a convolução à transformação nos dados de entrada é equivalente a aplicar a transformação na saída da convolução. Por exemplo, aplicar uma transformação $T(x)$ que desloca cada *pixel* em uma imagem uma unidade para a direita à saída de uma operação de convolução produz o mesmo resultado de aplicar a convolução à imagem de entrada deslocada e um objeto contido na imagem será detectado da mesma maneira. Em séries temporais, se uma observação for deslocada no tempo, sua representação também será deslocada na saída.

A próxima etapa de uma camada de convolução consiste em realizar uma subamostragem da saída das unidades. Nessa etapa, uma função de subamostragem (por exemplo, a função de máximo) é aplicada na saída das unidades em intervalos igualmente espaçados e retorna uma estatística resumo dos pontos próximos em um dado intervalo. A subamostragem então fornece como entrada para a próxima camada uma representação menor que ajuda a aumentar a eficiência estatística da rede. A saída da função de subamostragem é aproximadamente invariante a pequenos deslocamentos, uma propriedade muito útil na detecção de padrões. Por exemplo, essa propriedade torna mais fácil detectar se há uma queda abrupta em uma série temporal ao invés de detectar o momento exato que ela ocorre.

Ao final de uma rede neural utiliza-se camadas totalmente conectadas (*fully connected*). Esse tipo de camada, assim como as camadas de uma ANN, possui todas as suas unidades conectadas e é utilizada para modelar funções não-lineares a partir das características subamostradas extraídas das camadas anteriores.

4 Metodologia

4.1 Base de Dados

Definidos a motivação e os fundamentos teóricos, este trabalho propõe apresentar uma configuração de rede neural convolucional capaz de classificar se uma estrela contém algum planeta orbitando-a ou não com base na técnica do trânsito planetário, que consiste na observação da queda de luminosidade da estrela causada por um planeta em trânsito.

Os dados utilizados foram obtidos do *site* de competições *Kaggle*[1] e representam os fluxos de luminosidade (curvas de luz) de mais de 5.000 estrelas, contendo 3.197 observações coletadas durante um período de 80 dias. A cada fluxo está atribuído um rótulo codificado como "2" se a estrela possui um planeta confirmado em seu sistema ou "1" se não possui. Os rótulos são utilizados na supervisão do aprendizado da rede.

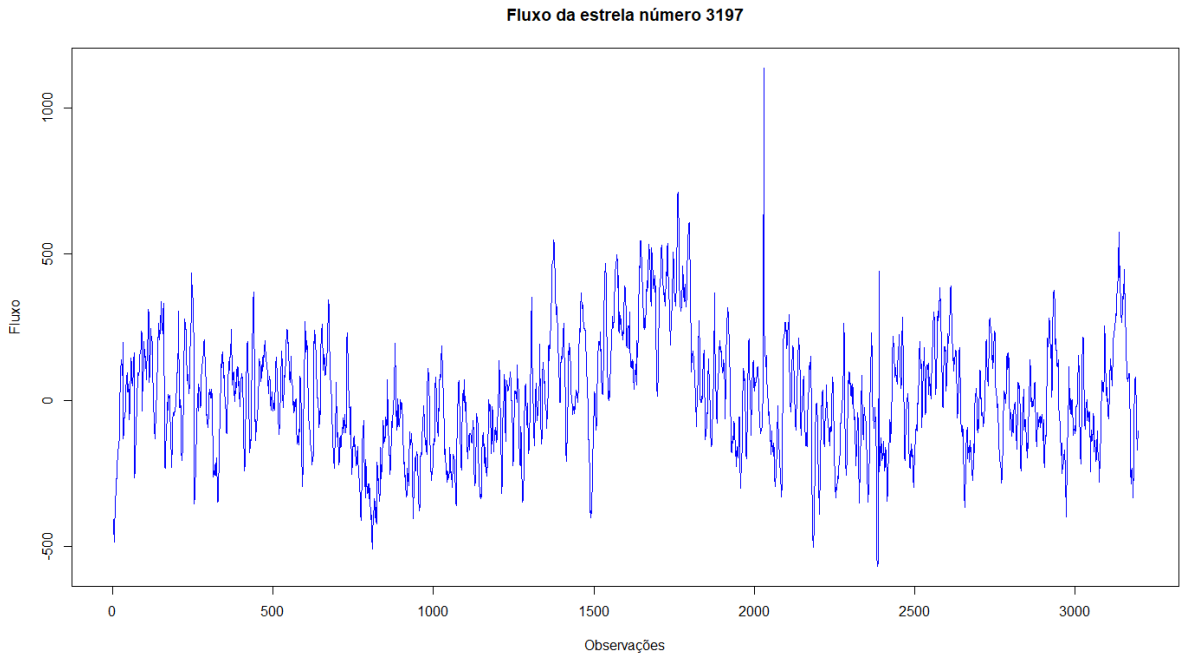

Figura 2 – Curva de luz da estrela número 3197 sem exoplaneta confirmado do banco de dados de treinamento.

Os dados foram obtidos já processados e divididos entre conjunto de treinamento e de teste, contendo os fluxos de 5.087 e 570 estrelas, que representam aproximadamente 90% e 10% do total dos dados, respectivamente. As observações são derivadas majoritariamente da Campanha 3 do telescópio espacial *Kepler* da NASA, com algumas estrelas com

[1] 1. https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data

planetas confirmados incluídas de outras campanhas para aumentar o número de exemplos positivos. Ainda assim os conjuntos de dados são altamente desbalanceados: o conjunto de treinamento possui cerca de $0,07\%$ de estrelas com planetas confirmados enquanto que o conjunto de teste possui cerca de $0,08\%$. Os dados originais da NASA são disponibilizados e armazenados no *Mikulski Archive*[2].

4.2 Metodologia de Aplicação

A técnica de observação que deu origem aos dados se baseia no fato de que um planeta, ao passar entre a sua estrela e um observador, provoca uma pequena alteração na luminosidade do astro. Essa alteração pode ser percebida como uma queda periódica em sua curva de luz, que se destaca da variabilidade natural da estrela. O período e o tamanho da queda geralmente dependem do período orbital e do tamanho do planeta, respectivamente.

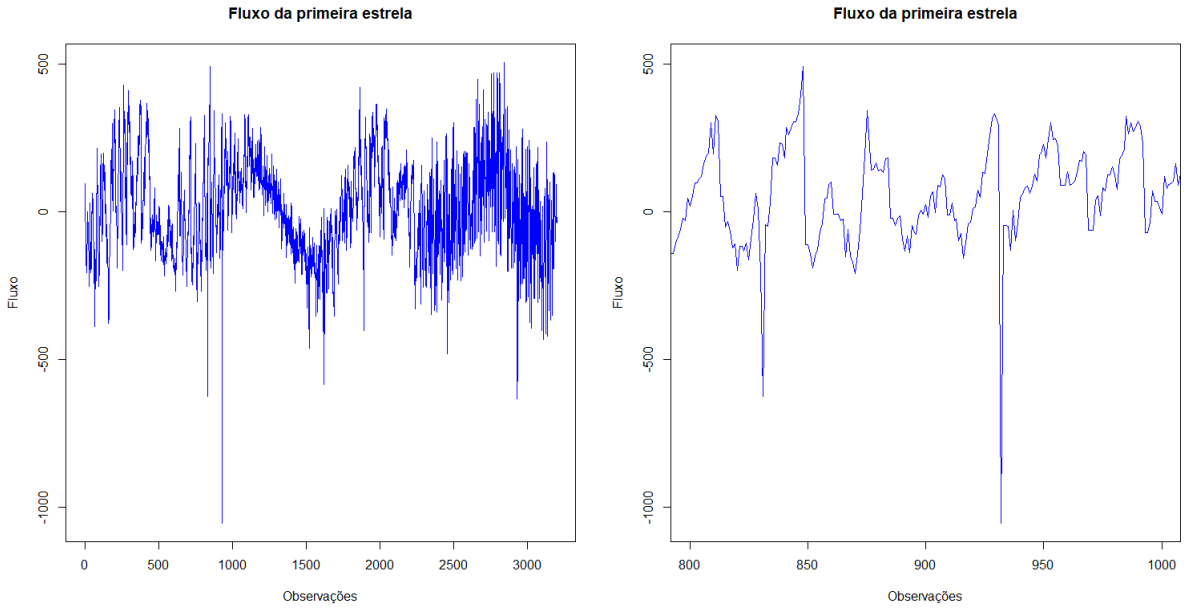

Figura 3 – Curva de luz da primeira estrela com exoplaneta confirmado do banco de dados de treinamento.
À direita, detalhe da queda de luminosidade causada por um possível trânsito planetário.

Há ainda outros tipos de fenômenos que podem causar interferências periódicas na curva de luz de uma estrela, tais como a passagem de outra estrela em um sistema binário ou a própria variação do seu ciclo natural, porém o trânsito planetário possui características muito específicas. Neste trabalho não é levada em consideração a distinção dos diferentes tipos de eventos que causam alteração de luminosidade de uma estrela. A

[2] 8. https://archive.stsci.edu/k2/

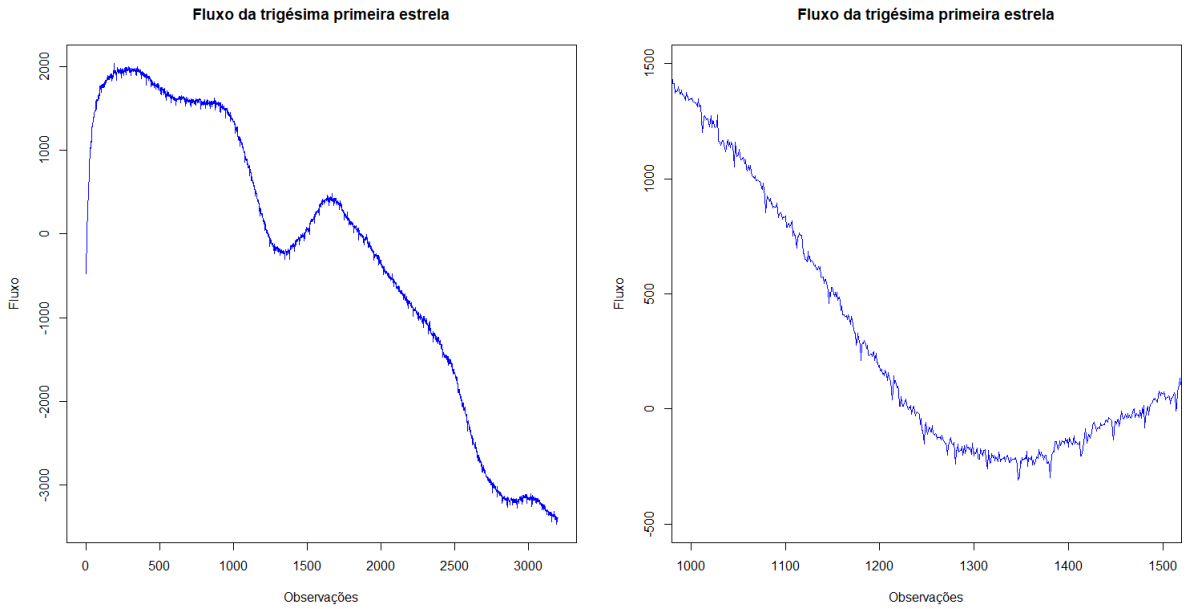

Figura 4 – Curva de luz da trigésima primeira estrela com exoplaneta confirmado do banco de dados de treinamento.
À direita, detalhe da queda de luminosidade causada por trânsito de curto período.

rede neural, portanto, deverá identificar a presença dos sinais que caracterizam a passagem de um planeta em curvas de luz de estrelas já confirmadas e distingui-los dos demais ruídos para que possa classifica-las corretamente.

A tarefa de classificação da rede consiste em aprender, através de representações, padrões e características específicas dos fluxos de luminosidade que tornem possível distinguir se são originados ou não de uma estrela com um sistema planetário. Dessa forma, uma análise exploratória dos dados foi conduzida a fim de identificar visualmente pontos de interesse que pudessem representar passagens de exoplanetas. Visando tornar mais evidentes as características associadas às passagens, foram ajustados três modelos com base nas curvas de luz originais e em dois processamentos diferentes dos dados, e seus desempenhos comparados. O primeiro modelo (modelo 1) foi obtido realizando o treinamento da rede utilizando como entrada as curvas de luz das estrelas. Para a obtenção do segundo modelo (modelo 2), um ajuste polinomial (*spline*) foi realizado e extraído de cada série temporal, resultando nas séries estacionárias utilizadas como entrada da rede. O terceiro modelo (modelo 3) foi ajustado com base nas funções de auto correlação (FACs) dos resíduos de processos ARIMA(1, 0, 1) (processos auto regressivos integrados de média móvel).

A arquitetura da rede foi obtida realizando-se seleções manuais comparando os resultados de diferentes configurações. Para treinar a rede de forma eficiente, foi necessário um pré-processamento simples dos dados. Os rótulos dos exemplos negativos e os positivos

foram renomeados para "0" e "1", respectivamente, e após isso os dados foram normalizados. Além disso, 20% das observações dos conjuntos de treinamento foram amostradas de forma aleatória e utilizadas para a validação dos modelos.

O grande desbalanceamento das classes provou ser um desafio no treinamento da Rede Neural. O número muito pequeno de exoplanetas confirmados em relação às estrelas sem exoplanetas em geral torna difícil a uma rede adquirir a capacidade de generalização. Para contornar esse problema, os exemplos positivos – rotulados como "1" – foram replicados. No treinamento e na validação da rede foram utilizadas 1.000 observações, contendo todas as replicações, juntamente com pesos atribuídos a cada classe correspondentes às proporções obtidas na amostra.

Dado o contexto de classificação dos fluxos das estrelas, foram consideradas as seguintes medidas para avaliar a performance do modelo:

- Acurácia: Taxa de estrelas com e sem planetas classificadas corretamente;
- Perda: Valor da função de perda utilizada no treinamento. Medida de variabilidade;
- Precisão: Fração das estrelas classificadas como confirmadas que realmente contém exoplanetas;
- Sensibilidade (*Recall*): Fração das estrelas que contém exoplanetas que são classificadas corretamente;
- AUC (*Area Under Curve*): Área abaixo da curva de Precisão *vs.* Sensibilidade (*Precision-Recall Curve*). Uma medida de significância dos valores preditos positivos pelo modelo.

Enquanto a acurácia e a perda se mostraram úteis para avaliar a existência de sobreajuste e a performance dos modelos no decorrer do treinamento, o real interesse foi obter valores relevantes de precisão, sensibilidade e AUC. Para dados altamente desbalanceados, essas medidas demonstram trazer maior informação sobre a real capacidade de generalização dos modelos [3]. Por exemplo, uma rede neural que prediz que todos os fluxos representam estrelas sem planetas pode facilmente obter acurácia de $99,12\%$ nos dados de teste. Quanto à precisão e sensibilidade, uma rede que prediz que nenhuma estrela possui planetas alcançaria precisão perfeita, mas sensibilidade zero. No caso de a rede predizer que todas as estrelas analisadas possuem planetas, alcançaria sensibilidade perfeita, mas precisão de $0,08\%$ nos dados de teste. Para classes desbalanceadas, a área abaixo da curva de precisão-sensibilidade demonstra ser mais informativa do que a área abaixo da curva ROC (*Receiver Operator Characteristic*), outra métrica comumente utilizada. A curva PR (*Precision-Recall Curve*) apresenta uma medida da capacidade do modelo de classificar corretamente os

positivos verdadeiros, relacionando a fração de valores positivos corretamente classificados e a fração de exemplos classificados como positivos que são realmente positivos.

Precisão é definida como o número de verdadeiros positivos dividido pelo número total de classificações positivas (verdadeiros positivos mais falsos positivos)

$$P = \frac{V_p}{V_p + F_p} \quad , \tag{4.1}$$

equanto que sensibilidade é definida como o número de verdadeiros positivos dividido pelo total de exemplos positivos (verdadeiros positivos mais falsos negativos)

$$S = \frac{V_p}{V_p + F_n} \quad . \tag{4.2}$$

Da definição de precisão, nota-se que o ponto de corte da classificação pode influenciar no total de classificações positivas, aumentando o denominador. A precisão aumenta se o ponto de corte for muito alto e as novas classificações forem verdadeiras positivas e diminui se o ponto de corte for muito baixo e as novas classificações forem falsas positivas. Portanto, a precisão pode não diminuir com a sensibilidade. Na definição de sensibilidade, por outro lado, o denominador não depende do ponto de corte. Sendo assim, um ponto de corte baixo pode implicar tanto em um maior número de verdadeiros positivos e, por consequência, no aumento da sensibilidade, quanto pode implicar em mudança apenas da precisão, com o aumento dos falsos positivos.

4.3 Arquitetura Proposta

Com base no trabalho desenvolvido por Shallue e Vanderburg [10], foi proposta uma arquitetura simples de Rede Neural Convolucional para a tarefa de classificar os dados. No total, a arquitetura possui cinco camadas, quatro delas de convolução, e 7.121 parâmetros. A rede proposta segue a seguinte estrutura:

- Quatro camadas de convolução, cada uma composta por:
    1. Uma etapa de convolução com 16 filtros de tamanho 5 e função de ativação retificadora linear;
    2. Uma etapa de subamostragem de tamanho 2, utilizando a função de máximo (*maxpooling*);
    3. Uma etapa de *dropout* [29] com probabilidade de inclusão $p = 0,25$;
- Uma camada de saída completamente conectada com uma unidade e função de ativação sigmoide.

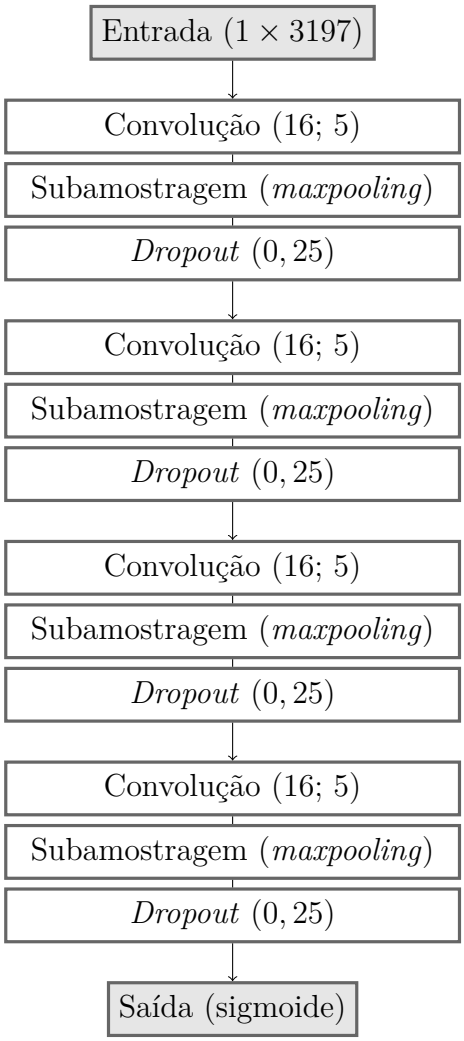

Figura 5 – Arquitetura da rede proposta.

O ajuste dos hiper-parâmetros foi feito com base no trabalho de referência, considerando a diferença entre os dados utilizados. A escolha das funções retificadora linear e sigmoide como funções de ativação, assim como o otimizador Adam [30] para o treinamento da rede e a função de perda de entropia cruzada foram mantidas, porém a taxa de aprendizado da rede foi mantida em 10^{-3}. A adição das etapas de *dropout* se mostrou importante para impedir sobre-ajuste. A rede foi treinada durante 100 épocas (número de iterações), número grande o suficiente para se observar convergência, utilizando uma amostra (*batch*) de 100 observações. Ambas escolhas se mostraram eficientes com relação à velocidade de processamento, porém com pouco impacto no resultado final. Os demais hiper-parâmetros da rede foram mantidos como na referência.

A rede proposta foi obtida variando-se manualmente alguns dos hiper-parâmetros e selecionando as configurações que apresentaram os melhores resultados a partir de exemplos anteriores enviados pelos usuários do *Kaggle*, com base nos mesmos dados obtidos do *site*. Partindo de uma rede simples com uma camada de convolução e 64 filtros, foram realizados testes de configurações com maior número de camadas e total de filtros, levando

em consideração a AUC.

Nos testes realizados com as curvas de luz, o aumento do número de filtros não se mostrou significativo para o aumento da AUC. Por outro lado, o aumento do número de camadas, mantendo-se constante o número de filtros, mostrou-se significativo para a melhora da performance da rede. Por exemplo, a rede inicial com uma camada de convolução de 64 filtros obteve AUC de $0,5897$, enquanto que a rede treinada com duas camadas de convolução de 64 filtros cada obteve AUC de $0,3856$. Com as variações da rede inicial com três e quatro camadas e total de 64 filtros foi possível obter AUC de $0,4262$ e $0,5691$, respectivamente.

Tabela 1 – Comparação do tempo de execução e AUC para redes com diferentes números de camadas.

Número de camadas	Tempo de execução	AUC
1 camada, 64 filtros	13 min 53 s	0,5897
2 camadas, 64 filtros cada	27 min 28 s	0,3856
3 camadas, 64 filtros cada	34 min 36 s	0,3528

Da mesma forma, após ser definida a estrutura básica da rede (4 camadas com 16 filtros cada), testes com diferentes probabilidades de *dropout* foram realizados. Mantendo-se os outros hiper-parâmetros constantes e variando-se a probabilidade de exclusão em cada uma das quatro camadas entre $p = 0,2$, $p = 0,25$ e $p = 0,5$, observou-se que o valor intermediário proporcionou melhores resultados.

Tabela 2 – Comparação do tempo e AUC de diferentes probabilidades de exclusão para a rede com 4 camadas de 16 filtros.

Probabilidade de exclusão	Tempo de execução	AUC
0,20	7 min 46 s	0,4718
0,25	8 min 12 s	0,6518
0,50	8 min 23 s	0,4625

4.4 Modelos Propostos

As passagens dos exoplanetas podem ser de difícil detecção, dependendo das características dele e de sua estrela. Uma das maneiras de deixar esses sinais mais evidentes é removendo a variabilidade da estrela por meio do ajuste de um *spline*, técnica também explorada por Shallue e Vanderburg [10]. Um *spline* consiste na interpolação de uma curva através de polinômios em intervalos espaçados, resultando em uma estimação do valor médio da função que os dados representam naquele intervalo. Sendo assim, *splines* foram ajustados e subtraídos. Com isso foi obtida uma série estacionária para cada curva de luz, nas quais percebe-se que os sinais das passagens dos exoplanetas foram mantidos enquanto as variabilidades das curvas foram removidas.

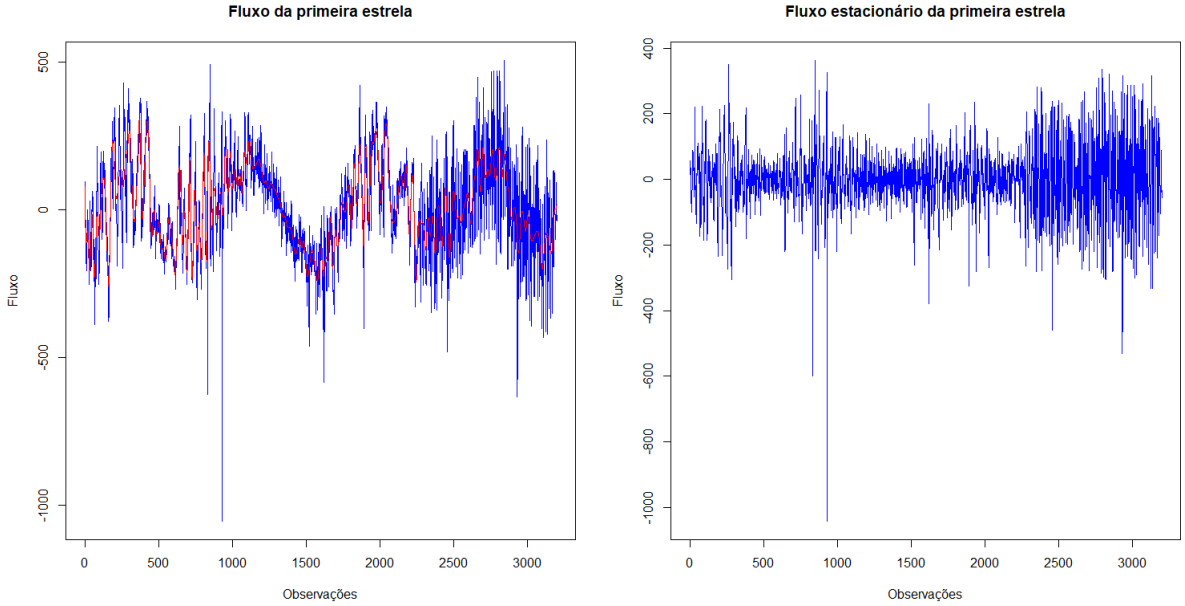

Figura 6 – Curva de luz da primeira estrela com exoplaneta confirmado do banco de dados de treinamento com spline ajustado.
À esquerda, *spline* em vermelho ajustado à curva de luz. À direita, série estacionária.

Outra maneira de se obter padrões sazonais de séries temporais é através da função de auto correlação (FAC). A FAC de uma série temporal, definida por

$$\gamma(s,t) = cov(x_s, x_t) \quad , \tag{4.3}$$

mede a dependência linear de dois pontos em diferentes momentos s e t [9]. Portanto, espera-se que pontos que representem sazonalidade nas séries apresentem correlação positiva com seus pontos consecutivos. Foi feito o ajuste de um modelo ARIMA(1, 0, 1) e obtida a FAC dos resíduos para cada curva de luz estacionária. Pôde-se observar nos gráficos das FAC pontos de correlação positiva periódicos, representando os trânsitos dos exoplanetas.

Para avaliar a capacidade da rede de extrair as características dos trânsitos, foram criados três modelos diferentes baseados nas filtragens dos dados propostas. Os modelos obtidos foram os seguintes:

1. Modelo 1: recebe como entrada as curvas de luz originais;
2. Modelo 2: recebe como entrada as curvas estacionárias;
3. Modelo 3: recebe como entrada as FAC dos resíduos de processos ARIMA(1, 0, 1).

Durante a etapa de validação, a configuração final da rede obteve AUC de $0,6518$ para o modelo 1, $0,8322$ para o modelo 2 e $0,3548$ para o modelo 3. A acurácia ao longo das

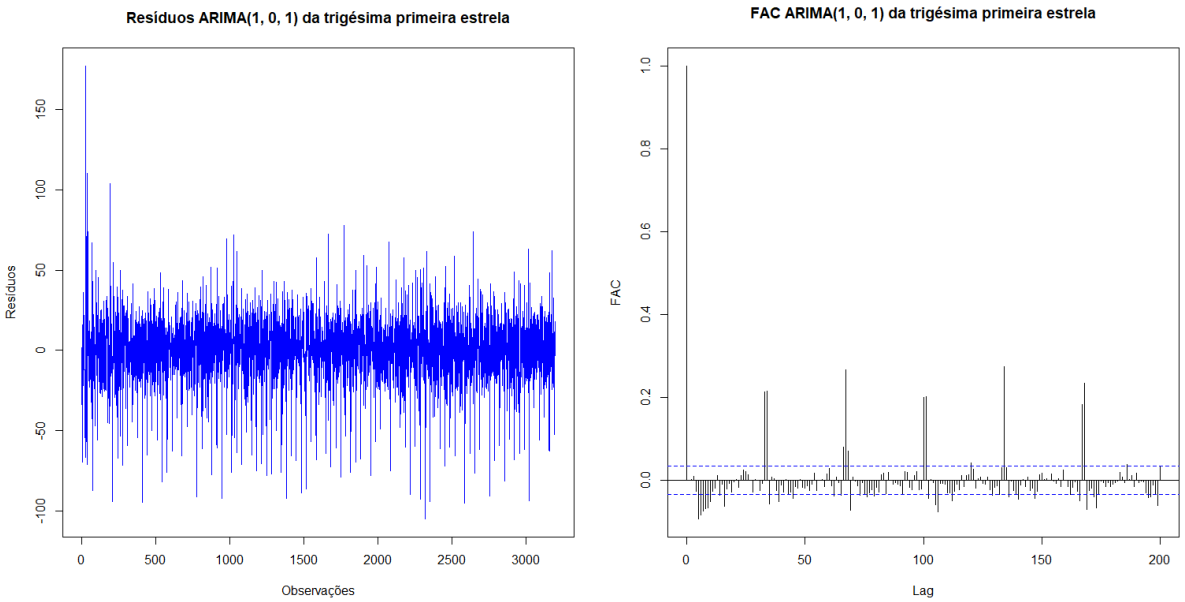

Figura 7 – Resíduos de um ARIMA(1, 0, 1) ajustado à trigésima primeira estrela do banco de treinamento.
À direita, gráfico da função de auto correlação. Passagens são representadas por picos positivos periódicos.

fases de treino e de validação permaneceu muito próxima nos modelos, estabilizando-se rapidamente com valores próximos de um nos modelos 2 e 3 após 40 épocas e mais tardiamente no modelo 1 após 60 épocas. De forma similar, os valores da perda permaneceram muito próximos ao longo das fases de treino e de validação em todos os modelos, estabilizando-se após 30 épocas no modelo 2, após 40 épocas no modelo 3 e após 50 épocas no modelo 1. Na avaliação feita nos modelos utilizando os dados de validação, o modelo 1 obteve perda de $0,0597$ e acurácia de $0,9843$, o modelo 2 obteve perda de $0,0097$ e acurácia de $0,9961$ e o modelo 3 obteve perda de $0,0612$ e acurácia de $0,9862$. O modelo 1 obteve um total de 1.002 classificações corretas contra 16 incorretas, o modelo 2 obteve 1.014 classificações corretas contra 4 incorretas e o modelo 3 obteve 1.004 classificações corretas contra 14 incorretas nos dados de validação, aplicando a regra de decisão

$$\hat{y} = \begin{cases} 1, & \hat{\pi} \geq 0,5 \\ 0, & \hat{\pi} < 0,5 \end{cases} \tag{4.4}$$

na qual $\hat{\pi}$ é a probabilidade estimada de a estrela apresentar um planeta.

Tabela 3 – Matriz de confundimento do modelo 1 durante a validação.

	Exemplos Positivos	Exemplos Negativos
Preditos Positivos	5	14
Preditos Negativos	2	997

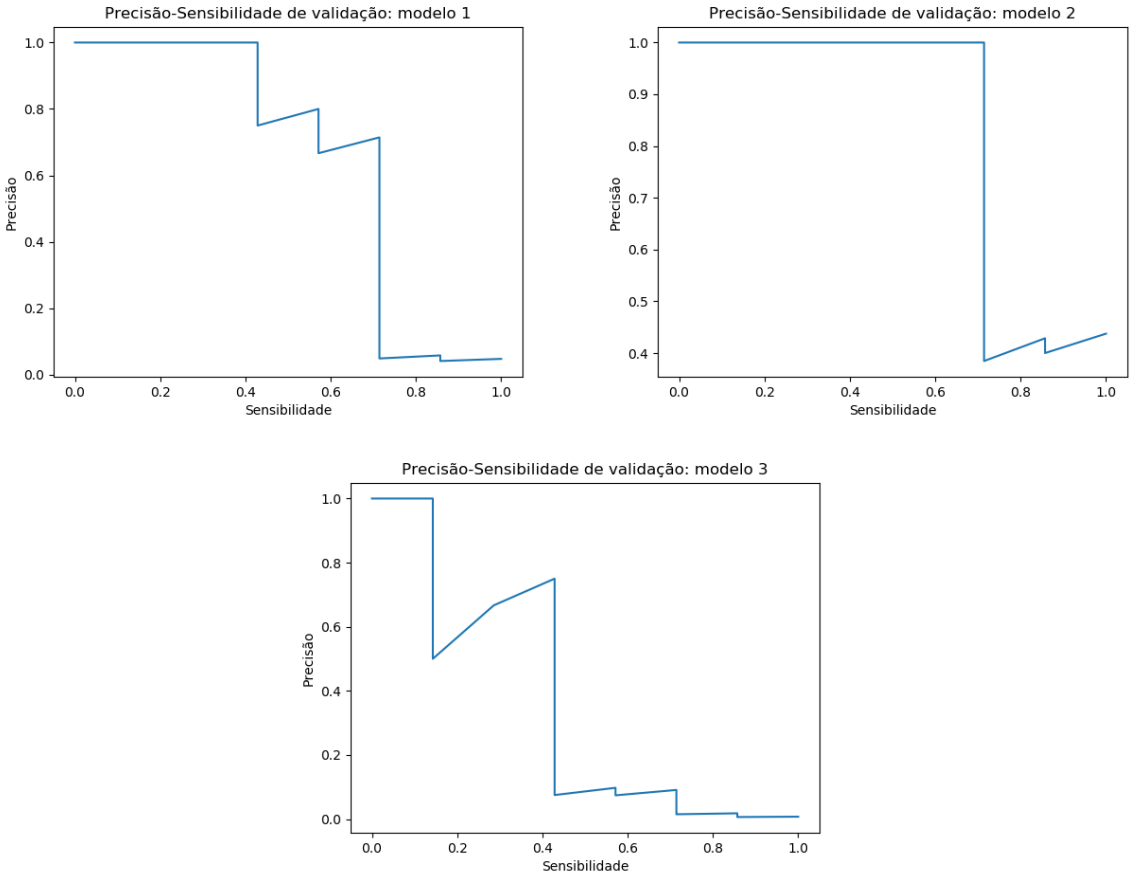

Figura 8 – Curva PR dos modelos durante a validação.
As curvas PR apresentaram $AUC_1 = 0,6518$, $AUC_2 = 0,8322$ e $AUC_3 = 0,3548$, respectivamente, durante a validação.

Tabela 4 – Matriz de confundimento do modelo 2 durante a validação.

	Exemplos Positivos	Exemplos Negativos
Preditos Positivos	5	2
Preditos Negativos	2	1009

Tabela 5 – Matriz de confundimento do modelo 3 durante a validação.

	Exemplos Positivos	Exemplos Negativos
Preditos Positivos	3	10
Preditos Negativos	4	1001

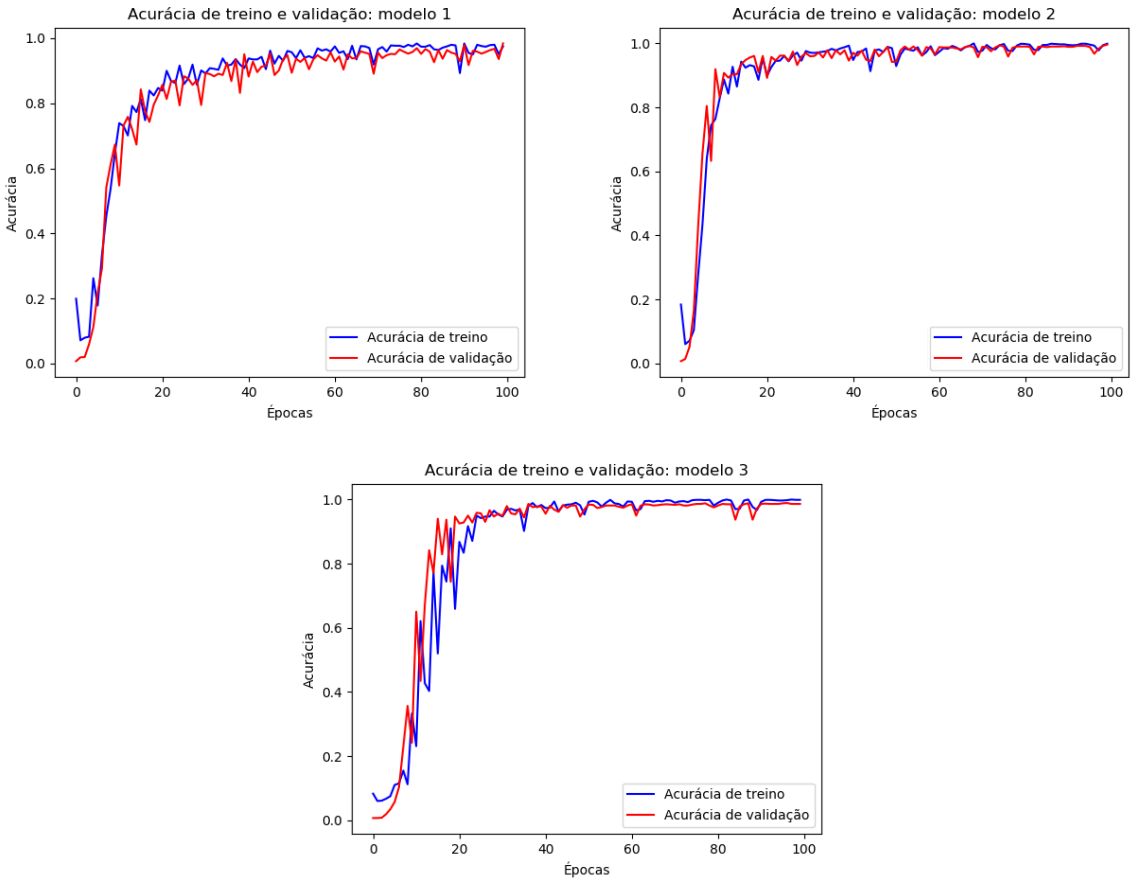

Figura 9 – Acurácia dos modelo durante o treino e a validação.

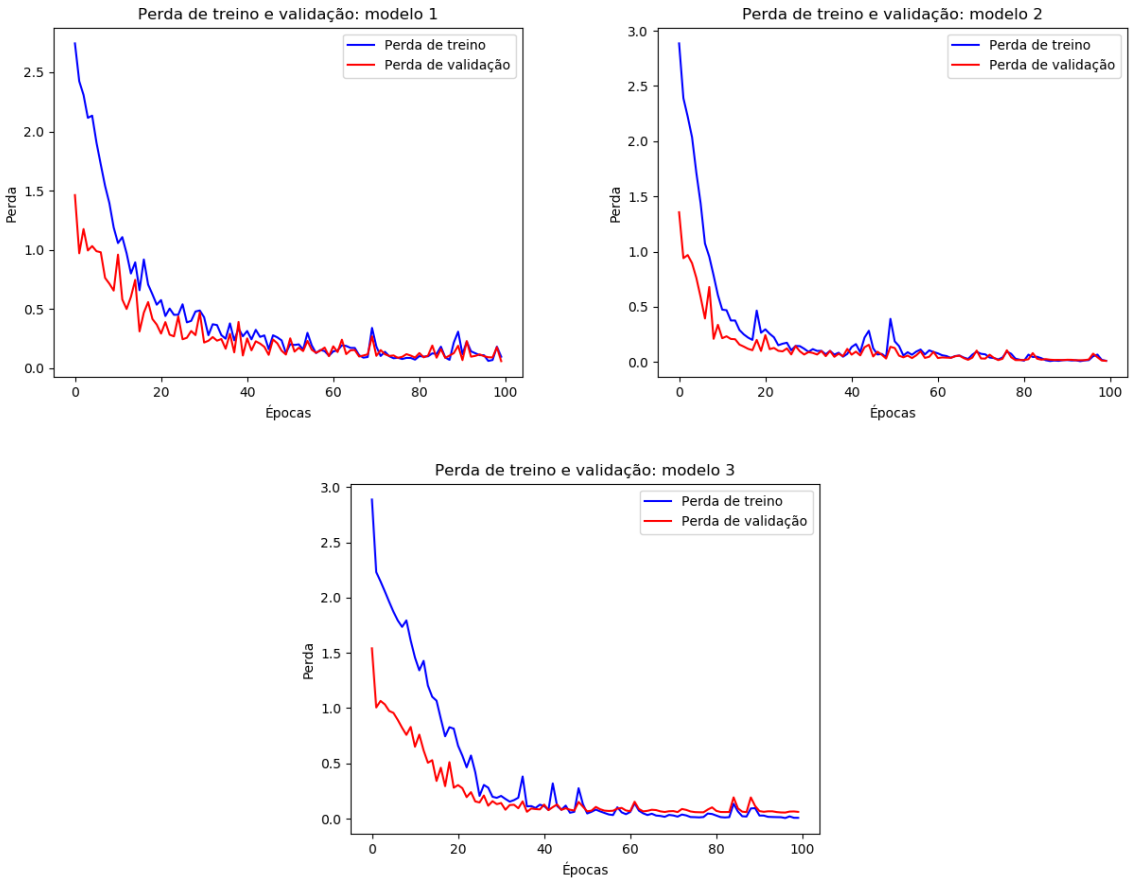

Figura 10 – Perda dos modelos durante o treino e a validação.

5 Resultados

5.1 Apresentação dos Resultados

Toda a metodologia foi aplicada em um *laptop* Vaio utilizando sistema operacional Microsoft Windows 10 *Home Single Language* de 64 *bits*, processador Intel® Core™ i7-4710MQ 2.50 GHz e memória RAM de 8,00 GB. As ferramentas de *software* utilizadas para a obtenção da rede neural e durante todo o treinamento e teste dos modelos foram a linguagem *Python* 3.6.0 com *framework* de aprendizado de máquina *Keras* e *back-end* sobre *TensorFlow*. Para a obtenção dos gráficos das curvas de luz, análise exploratória e aplicação dos filtros sobre os bancos de dados foi utilizado o R 3.4.1 juntamente com o RStudio 1.0.143.

Os modelos foram testados com base no banco de dados de teste também disponibilizado no *site Kaggle*, contendo cinco exemplos positivos (estrelas confirmadas), e nos filtros propostos utilizando *batches* de tamanho 100. Ao final dos testes foram obtidas as AUC, curvas PR, acurácia, perda e probabilidades estimadas dos exemplos positivos dos três modelos. As classes preditas pelos modelos foram obtidas aplicando-se a regra de decisão descrita anteriormente, também utilizada na fase de validação.

Tabela 6 – Tabela resumo das medidas de desempenho dos três modelos durante os testes.

	AUC	Acurácia	Perda
Modelo 1	0,7230	0,9894	0,0294
Modelo 2	0,8382	0,9929	0,0167
Modelo 3	0,7090	0,9877	0,0698

Tabela 7 – Matriz de confundimento do modelo 1 durante o teste.

	Exemplos Positivos	Exemplos Negativos
Preditos Positivos	5	6
Preditos Negativos	0	559

Tabela 8 – Matriz de confundimento do modelo 2 durante o teste.

	Exemplos Positivos	Exemplos Negativos
Preditos Positivos	4	3
Preditos Negativos	1	562

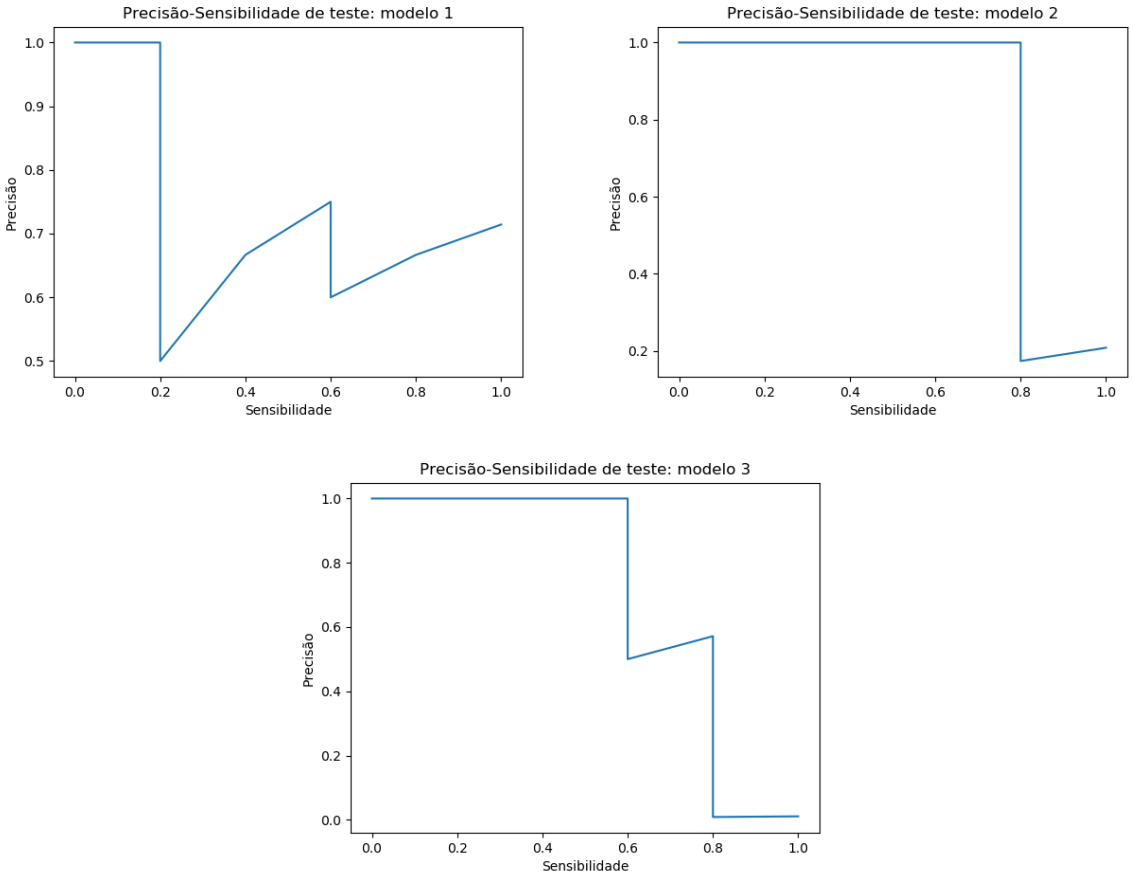

Figura 11 – Curvas PR dos modelos durante os testes.
As curvas PR apresentaram $AUC_1 = 0,7231$, $AUC_2 = 0,8382$ e $AUC_3 = 0,7091$, respectivamente, durante o treino.

Tabela 9 – Matriz de confundimento do modelo 3 durante o teste.

	Exemplos Positivos	Exemplos Negativos
Preditos Positivos	4	6
Preditos Negativos	1	559

Tabela 10 – Probabilidades estimadas de uma estrela apresentar exoplaneta.

	Modelo 1	Modelo 2	Modelo 3
Estrela 1	0,9721	0,9999	1,0000
Estrela 2	0,9337	0,9386	1,0000
Estrela 3	0,9997	0,9999	1,0000
Estrela 4	0,8803	0,0078	$8,3049 \times 10^{-11}$
Estrela 5	0,9943	0,9992	0,9194

5.2 Discussão dos Resultados

Através dos resultados obtidos, observa-se que o modelo 2 obteve a maior AUC, de $0,8382$, e os modelos 1 e 3 obtiveram AUC menores, $0,7231$ e $0,7091$, respectivamente. A avaliação dos três modelos mostra que o modelo 2 também obteve maior acurácia e menor perda em comparação com os outros.

A comparação dos valores preditos, revela que o modelo 1 classificou corretamente todos os exemplos positivos dos dados de teste, enquanto que os outros dois modelos classificaram corretamente quatro deles. No entanto, o modelo 2 classificou apenas 3 exemplos negativos de forma incorreta, contra 6 falsos negativos dos outros dois modelos.

Com relação às probabilidades estimadas pelos modelos, nota-se que o modelo 3 classificou todos os verdadeiros positivos com probabilidade muito alta, em particular os três primeiros exemplos positivos foram classificados com probabilidade $\hat{\pi} = 1$. Todos os modelos, no entanto, estimaram menor probabilidade para o quarto exemplo positivo de teste.

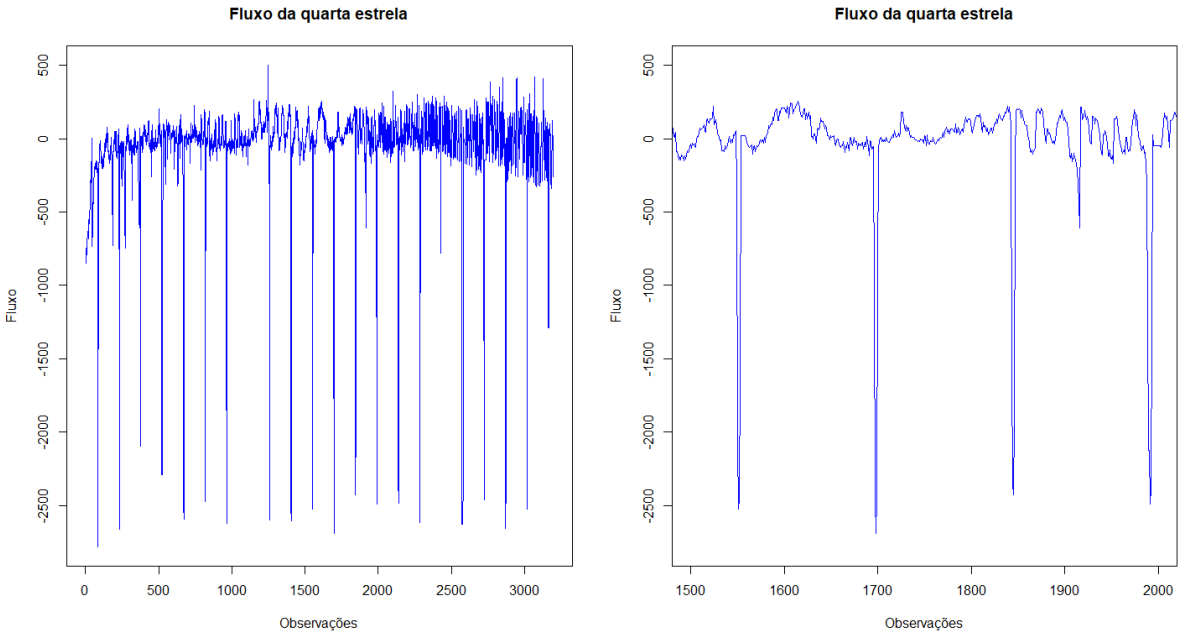

Figura 12 – Fluxo da quarta estrela com exoplaneta confirmado do banco de teste. À direita, detalhe do trânsito periódico.

Destaca-se a partir dos resultados dos testes e validações que o modelo 2 apresentou melhor desempenho geral, com menor perda, maior precisão e maior AUC, medida usada para avaliar a significância das classificações positivas. O modelo 1 foi capaz de classificar corretamente todos os exemplos positivos do conjunto de teste e apresentou performance mediana nas demais métricas. O modelo 3 não foi capaz de apresentar resultados satisfatórios como os demais nas medidas avaliadas, mas atribuiu maior probabilidade e

classificou corretamente os três primeiros exemplos positivos, indicando que a rede pode ter aprendido as características das passagens com sucesso para essas curvas de luz. Na fase de validação, os modelos 1 e 2 obtiveram a mesma quantidade de falsos positivos e verdadeiros positivos, porém o modelo 2 demonstrou mais eficiência para classificar os exemplos negativos corretamente.

6 Conclusão

Este trabalho propõe a aplicação de uma rede neural convolucional como forma de classificar estrelas com relação à presença de exoplanetas, baseando-se em suas curvas de luz. A partir da arquitetura de rede proposta, foram desenvolvidos três modelos que recebem como entrada diferentes filtragens dos dados. Os resultados obtidos sugerem que o modelo 2, que utiliza como entrada as curvas de luz estacionárias, apresentou melhor resultado geral em relação aos demais.

Cada modelo treinado, no entanto, se baseia em um método diferente de extração das características de interesse e os três obtiveram resultados interessantes na fase de testes. O modelo 1, baseado nas curvas de luz originais, classificou corretamente todas as estrelas confirmadas dos dados de teste, porém teve alta taxa de falsos negativos em comparação com o modelo 2. O modelo 3 classificou três das estrelas confirmadas com probabilidade estimada $\hat{\pi} = 1$, um indício de que o modelo pode ter extraído os sinais periódicos das passagem com sucesso nesses exemplos Ambos os modelos 1 e 3 apresentaram AUC muito próximas na fase de teste.

Este trabalho confirma a eficiência do uso de redes neurais convolucionais, por meio da extração de características dos conjuntos de dados, como poderosas ferramentas em tarefas de classificação, sendo possível obter bons resultados mesmo com arquiteturas simples e com custo computacional baixo. Apesar disso, o processamento prévio dos dados pode trazer melhoras ainda mais significativas para esse tipo de modelo, se tornando necessário para obter resultados ótimos na aplicação proposta.

Neste trabalho são aplicados filtros que reduzem a variabilidade e evidenciam padrões sazonais das curvas de luz. Assim como no trabalho de Shallue e Vanderburg [10], a aplicação de filtros que removam pontos discrepantes e isolem os eventos de interesse tem o potencial de melhorar ainda mais os resultados da rede. Além disso, a utilização de mais de uma entrada pode fornecer à rede informação essencial para que sejam detectadas as características de interesse, o que reflete em uma performance superior. Portanto, é importante que trabalhos futuros levem em consideração a utilização de diversos tipos de filtros como entradas simultâneas, afim de potencializar a extração de informação dos dados.

Referências

1. FERREIRA, A. E. T. **Estimação do Ângulo de Direção por Vídeo para Veículos Autônomos Utilizando Redes Neurais Convolucionais Multicanais**. 2017. Monografia apresentada como requisito parcial para conclusão do Bacharelado em Ciência da Computação, Departamento de Ciência da Computação, Instituto de Ciências Exatas, Universidade de Brasília, Brasília, 2017.

2. RANGEL, P. A. **Estudo Sobre Aprendizado Profundo**. 2017. Trabalho de conclusão de curso apresentado como requisito parcial para conclusão do Bacharelado em Estatística, Departamento de Estatística, Instituto de Ciências Exatas, Universidade de Brasília, Brasília, 2017.

3. GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>. Acesso em: 15 de ago. 2017, 23:06:00.

4. MITCHELL, T. M. **Machine Learning**. 1. ed. Maidenhead, UK: McGraw-Hill. 1997.

5. AGRESTI, A. **Categorical Data Analysis**, 2. ed. New York, NY: Wiley. 2003.

6. ROSS, S. M. **A First Course in Probability**, 5. ed. New Jersey: Prentice Hall. 1997.

7. BOX, G. E. P.; TIAO, G. C. **Bayesian Inference in Statistical Analysis**. 1. ed. Massachusetts: Addison-Wesley. 1973.

8. KARTTUNEN, H.; KRÖGER, P.; OJA, H.; POUTANEN, M.; DONNER, K. **Fundamental Astronomy**. 5. ed. New York: Springer-Verlag. 2007.

9. SHUMWAY, R. H.; STOFFER, D. S. **Time Series Analysis and Its Applications with R Examples**. 3. ed. Springer. 2011.

10. SHALLUE, C. S.; VANDERBURG, A. Identifying exoplanets with deep learning: a five planet resonant chain around Kepler-80 and an eighth planet around Kepler-90. **The Astronomical Journal**, The American Astronomical Society, n. 2, v. 155, jan. 2018. Disponível em: < http://iopscience.iop.org/ >. Acesso em: 2 abr. 2018.

11. LECUN, Y.; BENGIO, Y.; HINTON, G. Deep Learning. **Nature**, n. 436-444, v. 521, mai. 2015. Disponível em: < https://www.nature.com/ >. Acessado em: 5 mai. 2018.

12. LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P.; Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, n. 11, v. 86, nov. 1998. Disponível em: < https://ieeexplore.ieee.org/ >. Acessado em: 5 mai. 2018.

13. MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, n. 4, v. 5, dez. 1943. Disponível em: < https://link.springer.com/ >. Acessado em: 5 mai. 2018.

14. LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural Computation**, n. 4, v. 1, inverno 1989. Disponível em < https://www.mitpressjournals.org/ >. Acessado em: 6 mai. 2018.

15. LESHNO, M.; LIN, V. YA.; PINKUS, A.; SCHOCKEN, S. Multilayer feedforward networks with non-polynomial activation function can approximate any function. **Neural Networks**, n. 6, v. 6, 1993. Disponível em: < https://www.sciencedirect.com/ >. Acessado em: 6 mai. 2018.

16. KARPATHY, A. et al. Large-scale Video Classification with Convolutional Neural Networks. **Proceedings of the IEEE**, p. 1725-1732, 2014. Disponível em: < cv-foundation.org >. Acessado em: 8 mai. 2018.

17. LECUN, Y. Generalization and network design strategies. **Connectionism in perspective**, p. 143-155, jun. 1989. Disponível em: < psu.edu >. Acessado em: 6 mai. 2018.

18. LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech and time-series. **The Handbook of Brain Theory and Neural Networks**, 3361, n. 10, abr. 1995. Disponível em: < researchgate.net >. Acessado em: 8 mai. 2018.

19. YANG, J. B. et al. Deep convolutional neural networks on multichannel time series for human activity recognition. **IJCAI**, p. 3995-4001, 2015. Disponível em: < aaai.org >. Acessado em: 8 mai. 2018.

20. LAWRENCE, S. et. al. Face recognition: a convolutional neural-network approach. **IEEE Transactions on Neural Networks**, n. 1, v. 8, jan. 1997. Disponível em: < https://ieeexplore.ieee.org/ >. Acessado em: 8 mai. 2018.

21. ZHENG, Y. et al. Time series classification using multi-channels deep convolutional neural networks. **International Conference on Web-Age Information Management**, p. 298-310, jun. 2014. Disponível em: < https://link.springer.com/ >. Acessado em: 8 mai. 2018.

22. HOWELL, S. B. et al. The K2 mission: characterization and early results. **Publications of the Astronomical Society of the Pacific**, n. 938, v. 126, 2014. Disponível em: < http://iopscience.iop.org/ >. Acessado em: 13 mai. 2018.

23. HOWARD, A. W. et al. Planet occurrence within 0.25 AU of solar-type stars from Kepler. **The Astrophysical Journal**, n. 2, v. 201, jun. 2012. Disponível em: < http://iopscience.iop.org/ >. Acessado em: 14 mai. 2018.

24. WOLSZCZAN, A.; FRAIL, D. A. A planetary system around the millisecond pulsar PSR1257 + 12. 1992. **Nature**, 145-147, 355, jan. 1992. Disponível em: < https://www.nature.com/ >. Acessado em: 13 mai. 2018.

25. COUGHLIN, J. L. et al. Planetary candidates observed by Kepler. VII. The first fully uniform catalog based on the entire 48 month dataset (Q1-Q17 DR24). **The Astrophysical Journal**, n. 1, v. 224, mai. 2016. Disponível em: < http://iopscience.iop.org/ >. Acessado em: 14 mai. 2018.

26. MCCAULIFF, S. D. et al. Automatic classification of Kepler planetary transit candidates. **The Astrophysical Journal**, n. 1, v. 806, jun. 2015. Disponível em: < http://iopscience.iop.org/ >. Acessado em: 14 mai. 2018.

27. RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, 533-536, 323, out. 1986. Disponível em: < https://www.nature.com/ >. Acessado em: 27 mai. 2018.

28. GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. **Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics**, p. 315-323, 2011. Disponível em: <jmlr.org>. Acessado em: 27 mai. 2018.

29. SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **The Journal of Machine Learning Research**, 15, n. 1, p. 1929-1958, 2014. Disponível em: < jmlr.org >. Acessado em: 27 mai. 2018.

30. KINGMA, D. P.; LEI BA, J. Adam: a method for stochastic optimization. **arXiv preprint arXiv: 1412.6980**, dez. 2014. Disponível em: < https://arxiv.org/ >. Acessado em: 27 mai. 2018.

31. DAVIS, J.; GOADRICH, M. The relationship between precision-recall and ROC curves. **Proceedings of the 23rd international conference on Machine learning**, ACM, p. 233-240, 2006. Disponível em: < https://dl.acm.org/ >. Acessado em: 28 mai. 2018.

32. BUCKLAND, M.; GEY, F. The relationship between recall and precision. **Journal of the American Society for Information Science**, n. 1, v. 45, jan. 1994. Disponível em: < https://search.proquest.com/ >. Acessado em: 28 mai. 2018.

33. Python Development Team. **The Python Language Reference, version 3.6**. Amsterdam, The Netherlands, 2017. Disponível em: < http://www.python.org/ >.

34. R Core Team. **R: A Language and Environment for Statistical Computing**. Vienna, Austria, 2017. Disponível em: < https://www.R-project.org/ >.