Interpretabilidade de modelos de aprendizado de máquina por meio de uma rede de autoencoders

Hélisson Oliveira Magalhães Cerqueira, Leonardo Nogueira Matos

Departamento de Computação – Universidade Federal de Sergipe (UFS)
São Cristóvão – SE – Brasil

{helissonomc,leonardo}@dcomp.ufs.br

Abstract. *In recent years, with the increasing ability of computers to process vector computations, deep learning models have spread with the advent of new machine learning algorithms. These models, which are used in a wide variety of business areas, can be viewed as a black box that can neither be debugged nor audited. In this paper we will introduce a tree-based architecture which, because it is hierarchical, is more transparent and interpretable. The presented work is in its initial stage and the results obtained so far are still preliminary.*

Resumo. *Nos últimos anos com o aumento na capacidade dos computadores para processar cálculos vetoriais, com o aparecimento de novos algoritmos para aprendizagem de máquina, modelos de aprendizado profundo tornaram-se mais comuns. Estes modelos que são usados nas mais diferentes áreas de negócios podem ser considerados como uma caixa preta, impossível de serem depurados ou auditados. Neste trabalho vamos introduzir uma arquitetura baseada em árvore que, por ser hierárquica, possibilita ser mais transparente e interpretável. O trabalho apresentado está em fase inicial e os resultados obtidos até então são ainda preliminares.*

1. Introdução

O avanço na área de aprendizado de máquinas na última década decorre seguramente do aperfeiçoamento de modelos neurais complexos, com milhares de neurônios e parâmetros, denominados máquinas de aprendizado profundo, do inglês deep learning.

Atualmente, empresas da área de negócios, como bancos e seguradoras, e a área médica também empregam algoritmos de aprendizado profundo como ferramenta de auxílio à tomada de decisão [Miotto 2018]. Em alguns domínios, como a área médica e jurídica, os modelos devem ser interpretáveis, isto é, devem fornecer um mecanismo que permita depurar o processo de tomada de decisão [Ahmad 2018]. Na área médica, um especialista não confiaria na decisão proferida por um algoritmo que não pudesse ser explicada. O mesmo ocorre com a área jurídica. A condenação ou absolvição de um suspeito não pode ser feita por um algoritmo sem transparência. A máquina, portanto, não pode ser uma caixa preta. Ocorre que, quanto mais sofisticado é o modelo, mais ele se torna difícil de depurar. Uma árvore de decisão [Witten 2016] é um modelo naturalmente interpretável, mas tem um desempenho medíocre em tarefas onde tipicamente se usa aprendizado profundo, como visão computacional. Florestas aleatórias (random forests) [Witten 2016], por sua vez, que são modelos mais complexos construídos a partir de um conjunto de árvores de decisão, não são interpretáveis, sobretudo se empregarem grande número de árvores de decisão, no entanto possuem desempenho superior ao de muitos

outros em algumas tarefas complexas de classificação [Fernández-Delgado, Cernadas, Barro e Amorim 2016]. Existe, portanto, um antagonismo entre a complexidade do modelo e sua capacidade de ser interpretável, isto é, quanto mais complexo, mais difícil se torna a tarefa de interpretar sua tomada de decisão.

Neste projeto foi investigado a composição de um modelo hierárquico e interpretável, semelhante a uma árvore de decisão, construído como um agrupamento de autoencoders [Subramanian 2018], um modelo particular de rede neural que possui dois estágios denominados codificador e decodificador. A ideia, a grosso modo, é usar o estágio codificador como um extrator de características e desprezar o estágio decodificador. Presumimos que, ao fazer um agrupamento hierárquico destes autoencoders, será possível compor uma máquina inteiramente interpretável e que, neste caso, também seja capaz de ser assistida durante o treinamento.

2. Revisão da Literatura

2.1. Redes Neurais Convolucionais

Uma CNN é um tipo específico de rede neural normalmente utilizada para classificação de imagens [Albawi 2017]. Imagens no computador pode ser vista como uma matriz, onde cada valor da matriz é a intensidade do pixel da imagem, se a imagem for colorida (RGB), então significa que a imagem possui três canais, portanto três valores de intensidade diferente para um só pixel, um valor para a luz vermelha, um para verde e outro para azul, porém se a imagem for tons de cinza ela só possuirá um canal, portanto um só valor para um pixel. Na CNN existe um componente chamado kernel, que consiste em uma matriz NxN de valores chamados pesos, esse kernel faz a convolução na imagem, assim extraindo características, curvas e outros padrões [Albawi 2017]. A Figura 1 mostra como kernel atual na extração de características em uma imagem em tons de cinza.

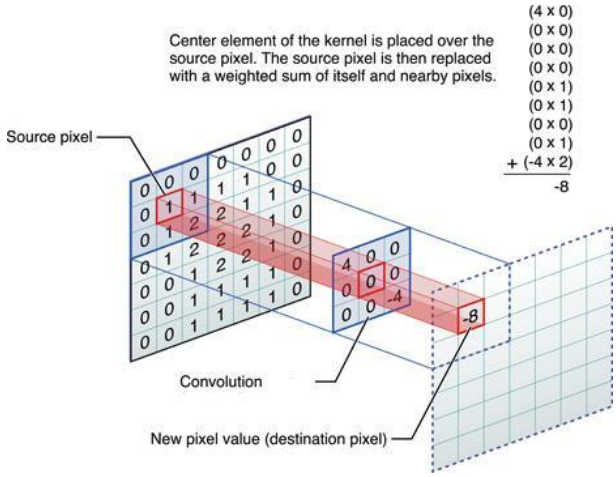

Figura 1 - Convolução usando um kernel 3x3 em uma imagem em tons de cinza.
Fonte:[BASAVARAJAIAH 2019].

A extração de características ajuda na classificação de imagens, pois com essas características o modelo poderá entender diferenças entre diversas classes de imagens, como por exemplo a diferença entre a imagem de um gato e a imagem de um cachorro.

2.2. Autoencoder

Autoencoder é uma rede neural artificial que aprende como comprimir dados de entrada e como reconstruir informações a partir dos dados comprimidos [Liang 2015]. Existem vários componentes em um autoencoder:

- Encoder – Este componente recebe os dados de entrada e é responsável por comprimir estes dados para um espaço menor;
- Bottleneck – É a camada do autoencoder que contém a representação comprimida dos dados de entrada;
- Decoder – Este componente é o responsável por aprender como reconstruir as informações a partir do bottleneck;
- Loss de reconstrução – É onde a reconstrução das informações das imagens de entrada tem a sua performance calculada.

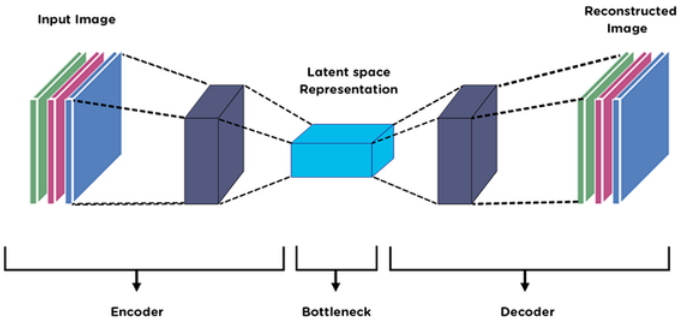

Figura 2 -Arquitetura do Autoencoder. Fonte: [Birla 2019].

Autoencoder possui várias utilidades [Badr 2019], entre elas estão:

- Detecção de anomalias – Quando um autoencoder é treinado para reconstruir exatamente as imagens de entrada, se o treinamento foi bem feito, então às imagens reconstruídas serão muito semelhante as imagens de entrada, porém essa reconstrução só será bem feita se a entrada for do mesmo domínio das imagens que o autoencoder aprendeu, logo se uma imagem que não faz parte deste domínio ao ser reconstruída será detectado uma anomalia, pois esta reconstrução causará uma valor da função de loss muito grande;
- Remoção de ruído da imagem – O autoencoder tem a capacidade de comprimir uma imagem com ruídos e reconstruí-la sem este ruído;
- Segmentação de imagens – Usando o autoencoder é possível detectar objetos em uma imagem, a imagem reconstruída pelo o decoder é uma máscara onde o valor do pixel representa qual é objeto que está presente no respectivo pixel da imagem.

2.3. Rede de Autoencoders

A rede de autoencoders é uma rede que sua estrutura lembra muito uma árvore. Esta rede pode ser dividida em duas partes. A primeira é onde os autoencoders estão presentes e a segunda é a parte densa. O objetivo da primeira parte é de extração de características da imagem de entrada e da segunda parte é de classificação da imagem.

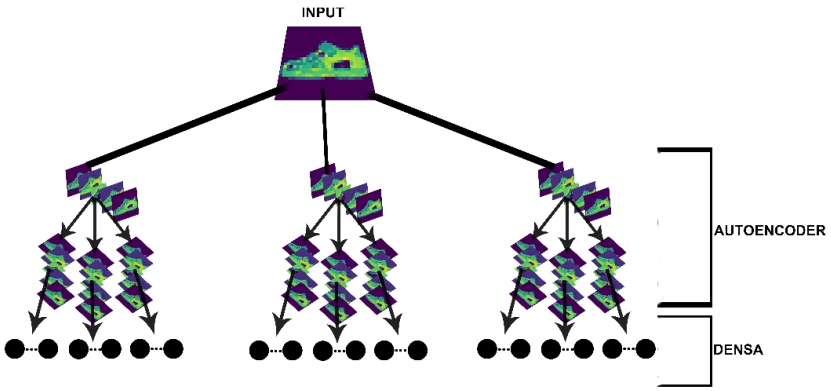

Figura 3 - Arquitetura da Rede de Autoencoder.

Os autoencoders presentes na primeira parte estão divididos em níveis, os de níveis abaixo são chamados de filhos dos autoencoders do nível superior, por exemplo, o autoencoder do nível 2 é filho do autoencoder do nível 1 e o do nível 3 é filho do nível 2. A camada de entrada dos autoencoders filhos é a imagem gerada no bottleneck do autoencoder pai. A imagem de entrada da parte densa é a imagem gerada no bottleneck do autoencoder de último nível. A camada densa possui apenas uma camada de entrada e uma camada saída. O número de neurônios na camada de saída será igual ao número de classes presente no conjunto de dados, então se a rede possuir apenas dois níveis, onde o primeiro nível possui três autoencoders, se cada um desses autoencoders possuir três filhos, logo o segundo nível terá nove autoencoders como é visto na Figura 3, e o número de neurônios na camada densa para um conjunto de dados de 10 classes será de 90 neurônios, 10 para cada autoencoder do último nível. A imagem de entrada é passada para todos os autoencoders do primeiro nível, assim podemos escolher qual caminho na rede a imagem irá percorrer. A escolha do caminho pode ser feita baseada na função de loss da reconstrução da imagem do autoencoder, ou pode ser baseada no valor do gradiente na camada do bottlenek. Após escolher o autoencoder o processo é repetido apenas para os filhos do autoencoder escolhido, usando a imagem gerada no bottleneck do autoencoder escolhido como imagem de entrada. Quando o ultimo autoencoder é escolhido a camada densa usa a imagem do seu bottleneck como entrada para a classificação da imagem.

3. Materiais e Métodos

3.1. Materiais

O conjunto de dados usado foi o Fashion-MNIST [Xiao, Rasaul e Vollgraff 2017], que consiste em um conjunto de treinamento de 60.000 exemplos e um conjunto de teste de 10.000 exemplos. Cada exemplo é uma imagem em tons de cinza de 28x28, associada a um rótulo de 10 classes (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot).

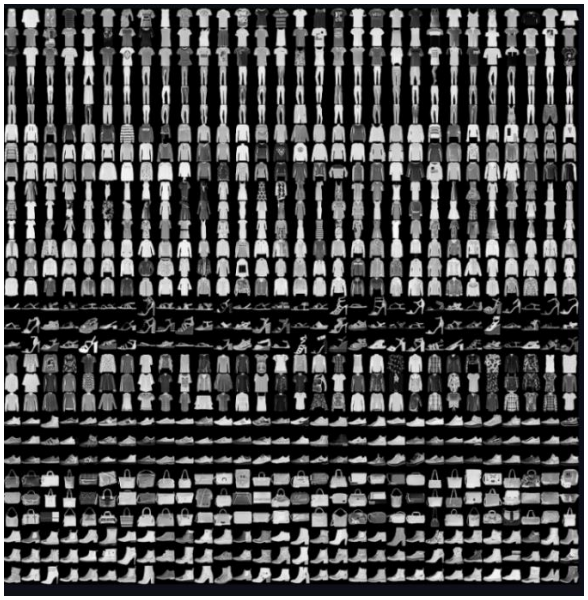

Figura 4- Amostra de imagens do conjunto de dados Fashion-MNIST. Fonte:[Xiao, Rasaul e Vollgraff 2017]

3.2. Métodos

A Metodologia usada possui quatro fases: a primeira é a definição da arquitetura dos autoencoders. A segunda é a definição da arquitetura da rede de autoencoders, isso inclui o número de filhos que cada autoencoders vai possuir, número de níveis da rede, incluindo a parte dos autoencoders e a parte densa. A terceira é a parte do treinamento, onde é definido a forma que vamos escolher o caminho para percorrer a rede. A quarta e última é usar a rede treinada para classificação de imagens.

3.2.1. Primeira etapa

Os autoencoders presentes na rede são autoencoders comuns, porém como o bottleneck é usado como uma imagem de entrada para um nível abaixo na rede de autoencoders, então o bottleneck pode gerar um feature map para caso de imagem em tons de cinza ou três feature maps para imagens RGB. Neste trabalho a rede foi implementada na linguagem Python, utilizando o Pytorch. O Keras também foi testado, porém devido ao seu contínuo aumento do uso de memória RAM durante a execução do processo de treinamento, foi optado o uso do Pytorch. A arquitetura dos autoencoders pode ser vista na Figura 5, duas camadas de convolução e duas camadas de convolução transposta.

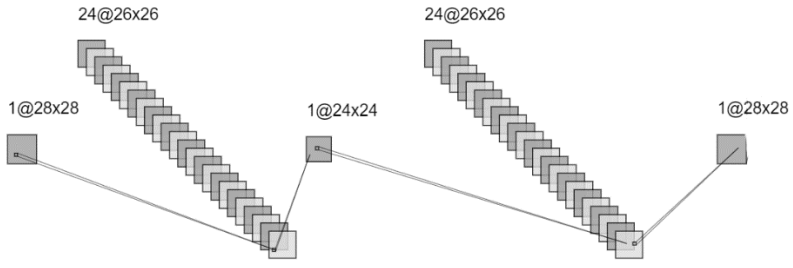

Figura 5-Arquitetura de um autoencoder.

3.2.2. Segunda etapa

A definição da arquitetura da rede é realizada na segunda etapa. Neste trabalho foram utilizados dois níveis de autoencoders seguido de um nível onde a camada densa está. Os autoencoders do primeiro nível possuem três autoencoders filhos. A Figura 6 mostra a arquitetura completa da rede. O conjunto de dados Fashion-MNIST possui 10 classes, logo a camada densa pra cada autoencoder do nível dois possui 10 neurônios de saída.

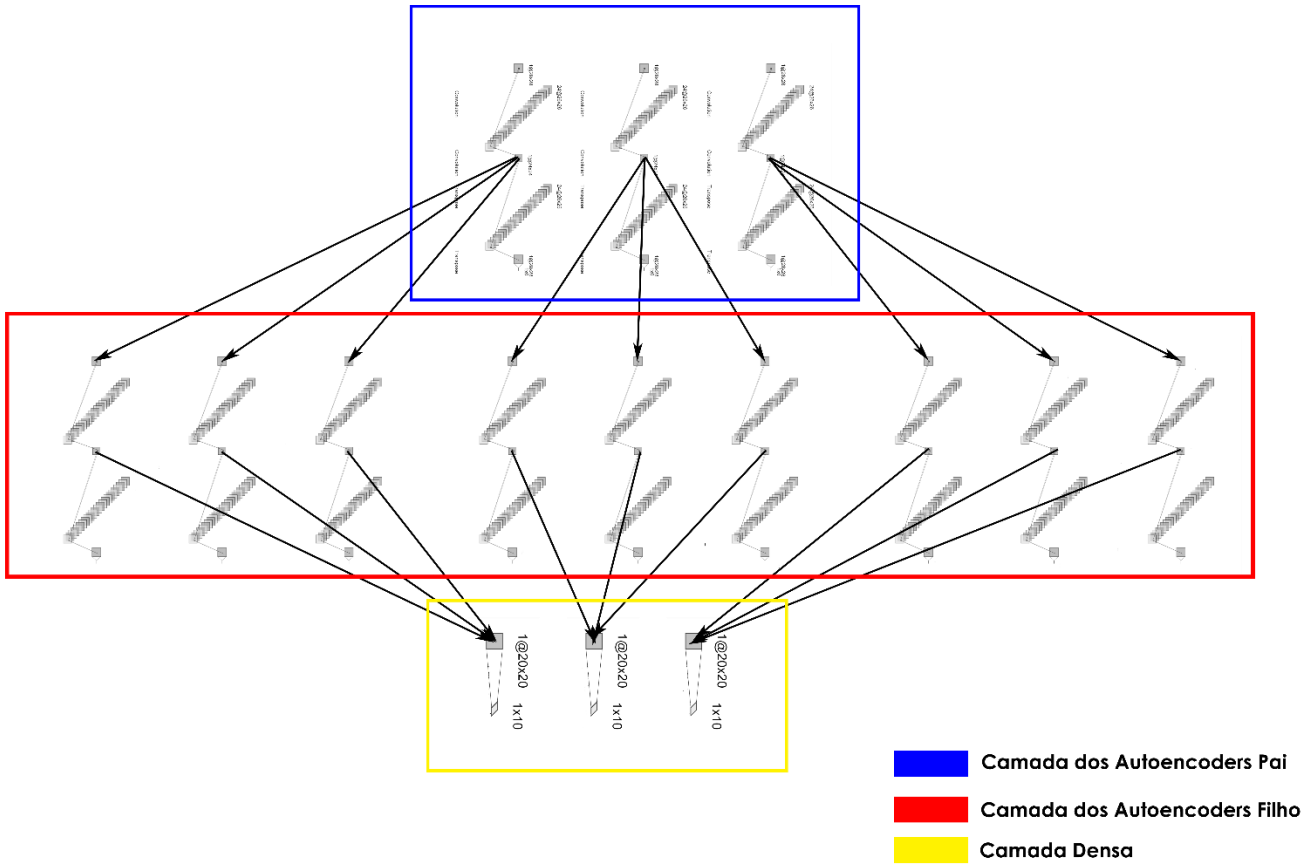

Figura 6 - Rede de autoencoders implementada neste trabalho.

3.2.3. Terceira etapa

Neste trabalho foi abordado duas formas de escolher o caminho a qual a imagem irá percorrer e realizar o treinamento/predição. A primeira forma foi baseada na comparação da soma dos valores absolutos do gradiente.

O procedimento usado foi o seguinte:

1. Usa imagem de entrada nos três autoencoders do primeiro nível para encontrar o gradiente no bottleneck de cada autoencoder;
2. Obtém e compara as somas dos valores absolutos dos gradientes;
3. Quem possuir o menor valor é o autoencoder escolhido;
4. Realiza o treinamento no autoencoder escolhido para reconstruir a imagem de entrada;

5. Repete os passos 1, 2, 3 e 4 para os autoencoders filhos do autoencoder escolhido, porém dessa vez a imagem de entrada será a imagem gerada no bottleneck do autoencoder escolhido;
6. Na camada densa a imagem de entrada será a imagem gerada no bottleneck do autoencoder escolhido no segundo nível. Com essa imagem gerada e a classe que ela possui, esta última camada densa é treinada.

Outra forma de escolha que foi realizada neste trabalho foi utilizando o valor gerado na função de loss na reconstrução da imagem. A função de loss usada foi a entropia cruzada binária. O processo é o mesmo do listado anteriormente, porém usando a loss e não o gradiente. Todos estes processos são realizados usando uma imagem por vez.

3.2.4. Quarta etapa

Após o fim da terceira etapa, já é possível realizar predições com a nossa rede de autoencoders. A predição é realizada de imagem por imagem, logo o tamanho do nosso batch é de apenas um. O processo é bem parecido com o de treinamento, a forma que percorremos a rede é idêntica, porem ao escolhermos os caminhos nenhum peso é atualizado, apenas é usado os valores que foi obtido na etapa anterior. No nível da camada densa, a predição é realizada de forma tradicional, gerando uma probabilidade para cada classe do conjunto de dados, assim escolhemos a classe que possui a probabilidade maior.

4. Resultados

O problema abordado neste projeto não é trivial, por isso os resultados alcançados não chegaram perto dos resultados obtidos no estado da arte, porém é um ponta pé inicial para a utilização de arranjo de autoencoders como forma de extrator de características e que pode ser interpretável. O conjunto de dados Fashion-MNIST possui 10000 imagens de teste, essas imagens foram utilizadas neste trabalho para avaliação na nossa rede de autoencoders. A acurácia do estado da arte para este conjunto de dados é de 96,91. A tabela abaixo mostra os valores obtidos usando a loss como métrica de escolha dos caminhos.

Tabela 1 - Acurácias obtidas com dados de teste do FashionMNIST.

Acurácia	Nº Epochs
23,25	10
51,06	1
15,02	10
60,41	1

Foi notado que com o aumento do número de épocas o processo de extração de características das imagens por parte do autoencoder pode gerar imagens com os pixels zerados no bottleneck, os motivos disso acontecer ainda estão sendo estudados. A figura 7 mostra a imagem de entrada e a imagem gerada no bottleneck que teve seus pixels zerados no processo de extração de característica.

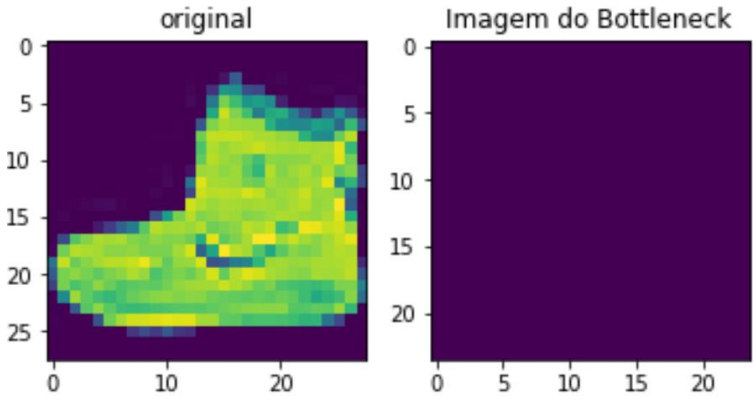

Figura 7 - Imagem gerada no bottleneck com pixels zerados.

A figura 8 mostra um exemplo de autoencoder que obteve êxito na extração de características da imagem original.

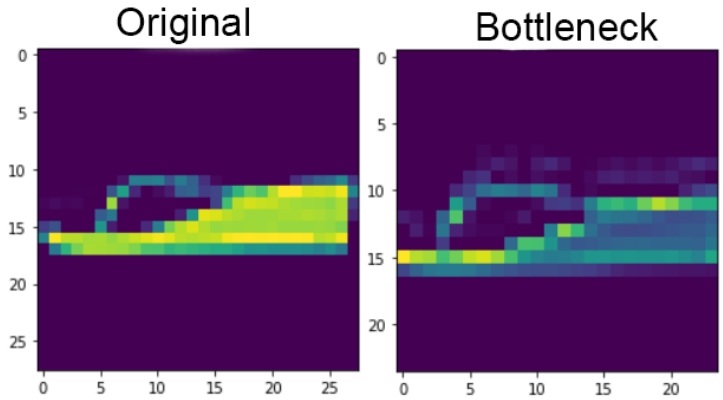

Figura 8 - Imagem gerada no bottleneck.

5. Considerações Finais

O estudo da utilização de uma rede de autoencoder com o objetivo de interpretar um processo de aprendizagem profunda foi investigado neste trabalho. Trata-se de um estudo inicial que foi realizado no âmbito do projeto PIBIC iniciado em 2020 e finalizado em 2021. Os resultados obtidos não igualam o estado da arte, mas como mencionado anteriormente, trata-se de um estudo em fase inicial.

Trabalhos futuros devem descobrir os motivos de alguns autoencoders falharem no momento da extração de características, tentar melhorar o desempenho das classificações das imagens, testar mudanças tanto na arquitetura dos autoencoders, quanto na estrutura da rede de autoencoders, incluindo mudanças na camada densa da rede de autoencoder. Outra coisa que pode ser implementada em trabalhos futuros é utilizar a predição feita na camada densa para atualizar os valores dos pesos dos autoencoders. Além disso é necessário estudar formas de otimizar todo o processo, desde a geração da rede de autoencoders até o processo de treinamento e classificação de imagens. A utilização de uma nova tecnologia para realizar os mesmos procedimentos também pode ser feita em trabalhos futuros.

Referências

Ian H Witten. Data mining. 2016.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? The journal of machine learning research, 15(1):3133–3181, 2014.

Vishnu Subramanian. Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch. Packt Publishing Ltd, 2018.

Basavarajauah, M.6 basic things to know about Convolution. 2019. Disponível em: <https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411>.

Birla, D.Basics of Autoencoders. 2019. Disponível em: <https://medium.com/@birla.deepak26/autoencoders-76bb49ae6a8f>.

XIAO, Han; RASUL, Kashif; VOLLGRAF, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.

Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, Joel T Dudley, Deep learning for healthcare: review, opportunities and challenges, Briefings in Bioinformatics, Volume 19, Issue 6, November 2018, Pages 1236–1246, https://doi.org/10.1093/bib/bbx044

Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. 2018. Interpretable Machine Learning in Healthcare. In Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB '18). Association for Computing Machinery, New York, NY, USA, 559–560. DOI:https://doi.org/10.1145/3233547.3233667

S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

J. Liang and R. Liu, "Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network," 2015 8th International Congress on Image and Signal Processing (CISP), 2015, pp. 697-701, doi: 10.1109/CISP.2015.7407967.

Badr, Will. Auto-Encoder: What Is It? And What Is It Used For? (Part 1), 2019. Disponível em: <https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726 >