Universidade Católica do Salvador
Bacharelado em Engenharia de Software

Rafael Rembrandt Aquino

Adaptação do WGAN ao processo estocástico

Salvador
2020

Rafael Rembrandt Aquino

Adaptação do WGAN ao processo estocástico

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software.
Orientador: Prof. Me. André Brasil Vieira Wyzykowski

Universidade Católica do Salvador

Salvador
2020

Rafael Rembrandt Aquino

Adaptação do WGAN ao processo estocástico

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como requisito parcial para a obtenção do título de Engenheiro de Software.

Comissão Examinadora

Prof. Me. André Brasil Vieira Wyzykowski
Universidade Católica do Salvador
Orientador

Prof. Mario Jorge Pereira
Universidade Católica do Salvador

Prof. Me. Marcelo Índio
Universidade Católica do Salvador

Salvador, 30 de setembro de 2020

Dedico este trabalho a meus pais, por todos os esforços que fizeram para que eu pudesse chegar até aqui, por tudo que me ensinaram durante toda a minha trajetória e por sempre estarem do meu lado.

Agradecimentos

Agradeço primeiramente a toda a minha família. Meu pai, Igor, minha mãe, Aretusa, não tenho como fazer agradecimentos diferentes para eles, ambos foram importantes da mesma forma, só eles sabem o quanto lutaram para me proporcionar uma educação de qualidade, só tenho a agradecer! Eles são muito mais do que parte disso, eles fizeram isso acontecer, essa é a nossa conquista.

À minha namorada, Zulmerinda, que acompanhou de perto todo o desenvolvimento desse trabalho e que me apoiou em todos os momentos e me ajudou a não desistir, sendo muito importante em todo esse processo.

Não posso em hipótese alguma deixar de agradecer ao professor Mario Jorge que foi muito além do que um professor, foi um amigo e uma das principais pessoas nessa minha trajetória, definitivamente, o responsável por me fazer começar a programar para dispositivos móveis, sendo isso até hoje minha ocupação como trabalho, sou e serei eternamente grato por sua presença na minha graduação, literalmente, mudou a minha vida.

Agradeço também a pessoa que me ajudou a tornar essa conclusão real, o professor André Brasil, meu orientador, que quase me fez enlouquecer com suas doses de cobranças, mas balanceadas com as doses de apoio e motivação, agradeço por sua paciência e dedicação em todo o meu processo de aprendizado. Agradeço as inúmeras horas diárias em que me orientou (de domingo a domingo), têm todo o meu respeito e admiração, que continue sendo esse exemplo de professor, sempre preocupado com o real aprendizado do aluno, eu realmente não sei como agradecê-lo por tudo.

Agradeço também a todos os meus colegas, pelos momentos que compartilhamos, pelos estudos em grupo, por toda a amizade que desenvolvemos em nossa caminhada! Agradeço ao meu amigo Saulo, que esteve lado a lado comigo em todas as etapas da graduação, vencemos isso juntos.

Gratidão, esse é o meu sentimento ao escrever agora, obrigado à todos que fizeram parte desse processo, peço desculpas se não citei o seu nome aqui, mas sou verdadeiramente grato por todos que participaram dessa trajetória, vocês me fizeram crescer, amadurecer e aprender.

"Sometimes life's going to hit you in the head with a brick. Don't lose faith. I'm convinced that the only thing that kept me going was that I loved what I did."

(Steve Jobs)

Resumo

Dentro de diversas áreas do conhecimento, os dados (diversos tipos de informações) são valiosos e a sua análise é mais valiosa ainda. Então, associando a área da inteligência artificial, observa-se uma nova moda, a geração de dados sintéticos para suprir a falta de dados. Sendo assim, analisando contextos atuais, esse trabalho tem como objetivo demonstrar a utilização de técnicas baseadas em eventos aleatórios para a otimização do resultado na execução de algoritmos baseados em *GAN (Generative adversarial networks)* e através de uma validação por meio do cálculo do *FID (Frechet Inception Distance)* foi possível analisar os resultados, determinando a qualidade dos dados gerados pelo algoritmo proposto em comparação a *WGAN*.

Palavras-Chave: 1. *GAN*. 2. Estocástico. 3. Inteligência artificial. 4. Otimização. 5. Computação.

Abstract

Within different areas of knowledge, data (different types of information) are valuable and their analysis is even more valuable. Then, associating the area of artificial intelligence, a new trend is observed, the generation of synthetic data to fill the lack of data. Therefore, analyzing current contexts, this work aims to demonstrate the use of techniques based on random events to optimize the result in the execution of algorithms based on *GAN (Generative adversarial networks)* and through a validation through the calculating the *FID (Frechet Inception Distance)* it was possible to analyze the results, determining the quality of the data generated by the proposed algorithm compared to *WGAN*.

Keywords: 1. *GAN*. 2. Stochastic. 3. Artificial intelligence. 4. Optimization. 5. Computation.

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

GAN	*Generative adversarial networks*
WGAN	*Wasserstein Generative adversarial networks*
CGAN	*Conditional Generative adversarial networks*
WCGAN	*Wasserstein Conditional Generative adversarial networks*
SMOTE	*Synthetic Minority Over-sampling Technique*
FID	*Frechet Inception Distance*
ADASYN	*Adaptive Synthetic Sampling Method for Imbalanced Data*

Sumário

1 Introdução

Considerada a ser uma das ideias mais interessantes dos últimos 10 anos, no âmbito de aprendizado de máquina, as *generative adversarial networks* (*GAN*) e suas variações estão sendo massivamente utilizadas na atualidade. A *GAN* e a *Wasserstein GAN* (algoritmo em foco utilizado no desenvolvimento do trabalho em questão) são executados de forma que um modelo generativo seja confrontado com um modelo discriminativo (adversário) que vai aprendendo a classificar a amostra do conjunto de dados proveniente dos dados reais ou dos dados gerados, forçando também o modelo generativo a melhorar para que seus dados possam ser confundidos como verdadeiros pelo classificador (GOODFELLOW et al., 2014).

Com a evolução da *GAN* diversos outros modelos originados da mesma foram desenvolvidos, cada um com sua peculiaridade, mas basicamente seguindo o mesmo principio de redes adversárias. Hoje, muitas empresas e centros de pesquisa investem nesses algoritmos, ainda mais com a sua importância dentro de negócios que movem o mundo, como, por exemplo, bancos. Esses algoritmos também estão sendo amplamente utilizados na área da saúde e em diversos outros segmentos.

A *GAN* pode, no meio da saúde, ajudar de diversas formas, como por exemplo, na geração de imagens sintéticas de raio-x para facilitar o diagnóstico de novas doenças. Já no meio bancário, a *GAN* pode ajudar na prevenção de fraudes, trabalhando em cima de uma amostra já existente, gerando dados sintéticos suficientes para a classificação da natureza de uma transação, considerando-a fraudulenta ou não.

1.1 Aplicabilidade e Motivação

O trabalho em questão iniciou-se com o objetivo de analisar e classificar um conjunto de dados de fraudes em transações de cartão de crédito disponível no *Kaggle* (Machine Learning Group - ULB, 2018). Analisando que dentro de um conjunto de dados relacionados a fraude a quantidade de não fraudes é majoritária e quase que absoluta considerando o cenário geral, percebe-se um problema para os algoritmos de aprendizado, a falta de dados de amostras fraudulentas para promover, de fato, o aprendizado da rede. Antes de chegar na intenção de desenvolvimento do trabalho em questão, para o mesmo conjunto de dados foram realizados vários testes e um deles foi a utilização do *SMOTE* como técnica de *oversampling* que com base na interpolação de instâncias da classe minoritária (dos dados reais já existentes) irá gerar as amostras *fakes*, fazendo com que não haja grande discrepância entre a quantidade de dados das classes (CHAWLA et al., 2002).

De acordo com a empresa de cibersegurança *Psafe*, entre Janeiro de 2018 e Agosto do

mesmo ano, houveram 3,6 fraudes em cartões de crédito no Brasil por minuto. Visando utilizar o atual estado da arte e produzir um trabalho de maior relevância para o cenário atual, todas as outras questões serviram de aprendizado, mas foram descartadas no desenvolvimento do presente trabalho, identificando que o problema não está na classificação e sim na geração de dados que permitam isso, o trabalho explora as melhores formas de fazê-lo.

1.2 Objetivos

O objetivo do trabalho foi executar e modificar o algoritmo *WGAN*, acrescentando o processo estocástico para gerar uma maior variabilidade nos dados sintéticos.

1.3 Objetivos Específicos

- Executar o algoritmo *WGAN* no conjunto de dados de fraudes em cartões de crédito.
- Adicionar valores estocásticos ao fim da execução do *WGAN*.
- Comparar os resultados obtidos pelo algoritmo *WGAN* com os resultados obtidos pela execução do algoritmo com as alterações de estocasticidade.

2 Fundamentação Teórica

Esta seção visa esclarecer conceitos e até mesmo o funcionamento de aprendizado supervisionado e não supervisionado, modelo generativo, WGAN (*Wasserstein Generative Adversarial Network*) e algoritmos importantes para a realização deste trabalho. Iniciando com a explicação do aprendizado supervisionado e não supervisionado, seguindo com uma seção para explicar os modelos generativos. Será tratado também, nesse capítulo, as principais particularidades do WGAN e alguns otimizadores importantes no contexto do trabalho, sendo eles o *Gradient Descent* e o *ADAM*, além disso o capítulo contém uma seção para explicação do FID (*Frechet Inception Distance*), algoritmo que foi utilizado como forma de validação do trabalho. Finalizando com a seção que aborda de forma geral sobre fraudes em cartões de crédito.

2.1 Aprendizado supervisionado

O aprendizado de máquina de forma indutiva pode ser dividida em duas subseções: O aprendizado não supervisionado e o aprendizado supervisionado, como mostrado na Figura 1. O aprendizado supervisionado é normalmente utilizado quando, como entrada, um determinado conjunto de dados já possui uma correlação com a sua respectiva saída esperada. Ou seja, quando há uma forte relação entre a entrada e saída e os "problemas" são bem definidos. A aprendizagem supervisionada é amplamente utilizada, como exemplo, em: *Neural network*, *support vector machines* e *naïve bayes classifiers*, tais exemplos são bastante explorados em aplicações e na comunidade cientifica e já tratados com profundidade em outros trabalhos. (SCHMIDHUBER, 2014; BEN-HUR; WESTON, 2010; RISH, 2001).

O aprendizado supervisionado faz parte da hierarquia do aprendizado indutivo (olhar a Figura 1), juntamente com o aprendizado não supervisionado que pode ser visto com mais detalhes na seção 2.2.

2.2 Aprendizado não supervisionado

O aprendizado não supervisionado é aplicado com o objetivo de formar agrupamentos, isso é, a partir de um conjunto de dados, os mesmos serão agrupados utilizando alguma estratégia, seja por repetições em padrões ou até mesmo por alguma regra de associação (MONARD; BARANAUSKAS, 2003). Como um dos principais algoritmos de *cluster* ou agrupamento temos o *K-Means* (SHI; LIU; GUAN, 2010), observe o Algoritmo 1, que é frequentemente utilizado no campo de mineração de dados, principalmente por orga-

nizar uma grande quantidade de dados dispersos. O algoritmo por si só, traz algumas deficiências (por exemplo: encontro do valor k ideal e o recálculo da distância entre os objetos e os centros de *cluster*) que podem ser tratadas com a implementação de algumas técnicas e aprimoramentos no próprio algoritmo.

Algoritmo 1: K-Means - Traduzido de (ZHOU; CHAN, 2014).

Entrada: k (o número de *clusters*, D (um conjunto de de proporções de *lift*)
Saída: um conjunto de k *clusters*

1 **início** De forma arbitrária escolha objetos k de D como o cluster central inicial.
2 **repita**
3 1. (re)atribuir cada objeto ao cluster ao qual o objeto é o mais semelhante, com base no valor médio dos objetos no cluster
4 2. Calcular o valor médio dos objetos para cada cluster
5 **até** *não haver mudança*;
6 **fim**

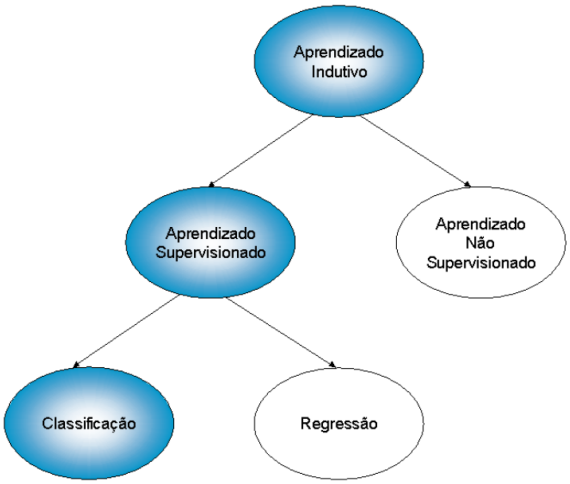


Figura 1 – Hierarquia de aprendizado
Fonte: (MONARD; BARANAUSKAS, 2003).

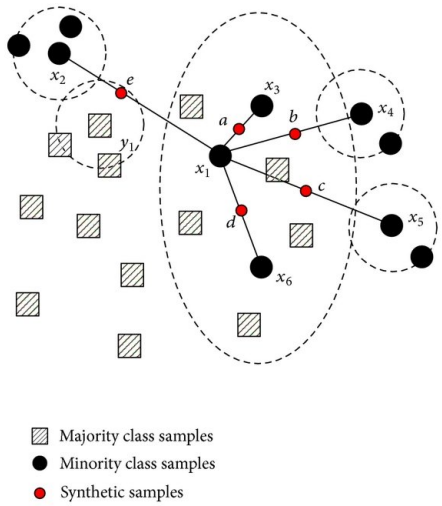
2.3 *SMOTE*

Dentro do âmbito de *machine learning* existem casos em que os dados se apresentam de forma desbalanceada, isso é, quando as classes não estiverem aproximadamente iguais em quantidade. Como técnica de *oversampling* é possível utilizar o SMOTE, que a partir dos dados existentes, irá gerar mais amostras de dados com base na interpolação de instâncias de classes que estejam em menor quantidade (CHAWLA et al., 2002).

Explicando de forma mais simplificada, a ideia do *SMOTE* e de outras técnicas de *oversampling* é utilizar a classe minoritária e a partir dela, criar novas amostras até que

seu tamanho seja correspondente as amostras da classe majoritária, como mostrado na Figura 2.

Figura 2 – *SMOTE*



Fonte: (HU; LI, 2013).

Por exemplo, no *dataset* utilizado no trabalho existem, originalmente, 492 casos de fraude (classe minoritária) e 284.315 casos onde não existem fraude (classe majoritária), então o objetivo maior é aumentar a amostra de classes minoritárias, tornando-a mais próxima da quantidade de classes da amostra de classes majoritárias.

O objetivo é identificar as linhas que unem todos os pontos (dados da classe minoritária) com base na quantidade de vizinhos próximos considerados, gerando novas instâncias nos intervalos dessas linhas, consequentemente diminuindo a diferença entre a classe de amostras majoritárias com as minoritárias, para melhor entendimento, olhar o Algoritmo 2.

Algoritmo 2: SMOTE - (MACHADO, 2007).

Entrada: S : conjunto de dados de entrada; k : número de vizinhos mais próximos considerados; qtd : número de novos casos desejado para cada caso da classe minoritária;

Saída: S balanceado

```
1  início
2      Para cada caso i pertencente à classe minoritária faça {
3       Calcule os k vizinhos mais próximos do caso i;
4       Faça qtd vezes {
5        Para cada atributo contínuo de i faça {
6         Crie um valor entre i e um dos k vizinhos;
7        }
8       }
9      }
10 fim
```

2.4 *Generative adversarial network*

Uma *generative adversarial network* é baseada no treinamento simultâneo de duas redes, a generativa que é responsável por gerar os dados falsos e a discriminativa que categoriza os dados fornecidos do conjunto de dados de treinamento e da rede generativa (GOODFELLOW et al., 2014), uma das finalidades da utilização da *GAN* é a geração de dados sintéticos.

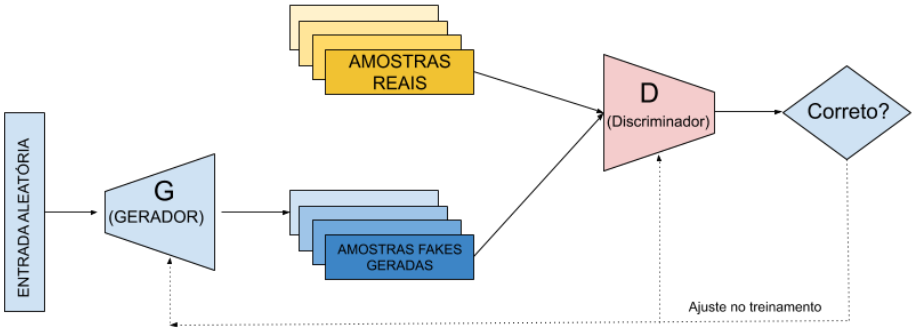


Figura 3 – Arquitetura *GAN*

Através da Figura 3 pode-se observar a arquitetura de funcionamento de uma *GAN*. Como dito anteriormente, tanto o gerador quanto o discriminador representam duas redes neurais distintas.

O processo acontece da seguinte maneira: No inicio, o gerador começa a produzir amostras sintéticas no objetivo de conseguir uma resposta positiva do discriminador, a

partir da classificação, o discriminador fornece informações para que o gerador possa atualizar seus pesos no objetivo de aumentar sua taxa de sucesso, isso, através da retro-propagação.

Diferentemente do *critic* que é explicado na sessão *Wasserstein GAN*, o discriminador é um classificador "simples", podendo utilizar qualquer arquitetura de rede neural que permita a classificação de dados.

No Algoritmo 3 pode-se observar o processo de treinamento de uma *GAN*, realizado por etapas através dos *minibatches* das amostras até a atualização do discriminador e do gerador, já que a *GAN* conta com duas redes distintas, o algoritmo deve se preocupar no treinamento de ambas.

Algoritmo 3: Treinamento do *Minibatch* estocástico de gradiente descendente de redes adversárias generativas. O número de etapas a serem aplicadas ao discriminador, k, é um hiper-parâmetro. É utilizado $k = 1$, a opção menos dispendiosa, nos experimentos. TRADUZIDO E ADAPTADO DE: (GOODFELLOW et al., 2014)

início durante N iterações do treinamento **faça:**

  repita

    1. *Minibatch* de amostras de m amostras de ruído $\{z^{(1)}, ..., z^{(m)}\}$ do ruído anterior $p_g(z)$.

    2. *Minibatch* de amostras de m exemplos $\{z^{(1)}, ..., z^{(m)}\}$ da distribuição de geração de dados $p_{data}(x)$.

    3. Atualize o discriminador, aumentando o gradiente estocástico:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]. \tag{2.1}$$

  até *k etapas*;

  1. *Minibatch* de amostras de m amostras de ruído $\{z^{(1)}, ..., z^{(m)}\}$ do ruído anterior $p_g(z)$.

  2. Atualize o gerador por gradiente descendente estocástico:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} + \log(1 - D(G(z^{(i)})))]. \tag{2.2}$$

fim

Durante o processo de treinamento, tanto a rede generativa quanto a rede discriminativa trabalham na tentativa de otimizar a função de perda. Considerando o jogo *minimax* (BLACKWELL et al., 1956) e as duas redes como jogadores, o gerador (jogador um) tenta a todo momento maximizar a probabilidade de que seus dados gerados sejam reconhecidos

como reais e a rede discriminativa (jogador dois) tenta minimizar.

2.5 *Wasserstein GAN*

Mesmo que a GAN tenha tido um grande sucesso na geração de dados sintéticos, o processo ainda é considerado lento e instável (WENG, 2019). A *Wasserstein GAN*, é uma das extensões que existem da *GAN*, ela traz mais estabilidade ao treinamento do modelo (ARJOVSKY; CHINTALA; BOTTOU, 2017). A Distância de *Wasserstein* é uma medida da distância entre duas distribuições de probabilidade e a *Wasserstein loss* encoraja previsões semelhantes à *ground truth*, de maneira sólida à identificação incorreta de classes semelhantes. É calculada a distância euclidiana entre a previsão e o *ground truth vs* o número de classes com média de diferentes níveis de ruído e o nível de ruído com a média do número de classes (FROGNER et al., 2015).

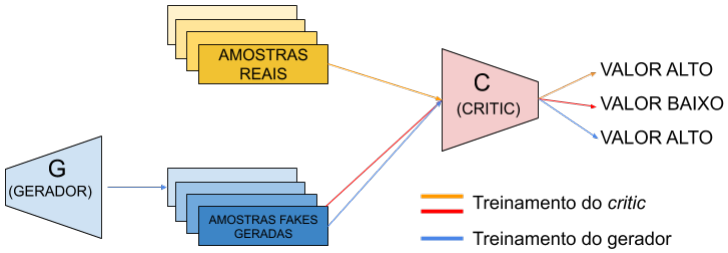


Figura 4 – Arquitetura *WGAN*

Pode-se observar na Figura 4 que O *WGAN* utiliza o modelo *critic* que tem papel semelhante ao discriminador da *GAN*, pode-se dizer que essa é uma das principais diferenças na arquitetura das redes. Diferente do discriminador, que apenas retorna verdadeiro ou falso para o conjunto de dados passado e evolui de acordo com o aumento de épocas, o *critic* do *WGAN* calcula a melhor distancia de *Wasserstein*, simulando a função *Lipschitz* contínua (ZHOU et al., 2019) e a sua atualização só acontece sob restrição que o *critic* satisfaz a condição de continuidade de *Lipschitz*.

Observa-se no Algoritmo 4 que o processo é bem parecido com o que ocorre na *GAN*. Nele a entrada também é dividida em *batches*, mas o treinamento é realizado com a ajuda do otimizador *RMSProp*. O treinamento promove melhorias no *critic* e também nos parâmetros θ_0 iniciais do gerador.

Algoritmo 4: WGAN, o algoritmo proposto. Considerar os valores padrões de $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{critic} = 5$. TRADUZIDO E ADAPTADO DE: (ARJOVSKY; CHINTALA; BOTTOU, 2017)

Entrada: a, a taxa de aprendizado, c, o parâmetro de *clipping*, m, o tamanho do *batch*. n_{critic}, o número de iterações críticas por iteração de gerador.

Entrada: w_0, parâmetros iniciais criticos. θ_0, parâmetros iniciais do gerador

início
  repita
    repita
      Amostra $\{x^i\}_{i=1}^{m} \sim \mathbb{P}_r$ um *batch* dos dados reais.
      Amostra $\{z^i\}_{i=1}^{m} \sim p(z)$ um *batch* de amostras anteriores
      $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
      $w \leftarrow w + \alpha \cdot RMSProp(w, g_w)$
      $w \leftarrow clip(w, -c, c)$
    até $t = 0,\ldots,n_{critic}$;
    Amostra $\{z^i\}_{i=1}^{m} \sim p(z)$ um *batch* de amostras anteriores.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot RMSProp(\theta, g_\theta)$
  até θ *não convergir*;
fim

Existem algumas diferenças notáveis entre a *GAN* e a *Wasserstein GAN*, por exemplo: a troca do *momentum* pelo *RMSProp*. Além disso outras diferenças podem ser evidenciadas, como, a utilização do *critic* no lugar do discriminador e a utilização da perda de *Wasserstein* para treinamento dos modelos *critic* e gerador.

2.6 Otimizador: ADAM

Os otimizadores estão presentes na etapa de atualização de pesos da rede, com o objetivo de minimizar a função de erro. Existem diversos otimizadores, como Adagrad, Adadelta e até mesmo o próprio gradiente descendente com a aplicação pura do algoritmo *backpropagation*. Nessa seção, o otimizador ADAM, Kingma e Ba (2014) será tratado com maior profundidade através do pseudocódigo que representa o funcionamento do algoritmo, pois foi o otimizador utilizado no desenvolvimento do trabalho.

São os otimizadores que definem a forma como as redes neurais aprendem, eles são responsáveis por encontrarem os parâmetros ideais de forma que a função de perda seja a mais baixa possível. Um dos motivos para a escolha do *ADAM* como otimizador desse trabalho é por conta da sua grande popularidade, diversos pesquisadores já o testaram e

ele funciona bem nos problemas aplicados.

Algoritmo 5: Adam - Adaptado de (RODRIGUES et al., 2018).

Entrada:

α : (taxa de aprendizado),

β_1 e β_2 : (taxas de decaimento exponenciais para estimativas de momento),

$f(\theta)$: (função objetivo estocástica com parâmetros θ),

θ_0 : (vetor inicial de parâmetros).

início

  $m_0 \leftarrow 0$ (inicialização do vetor de primeiros momentos)

  $v_0 \leftarrow 0$ (inicialização do vetor de segundos momentos)

  $t \leftarrow 0$ (inicialização do passo de tempo)

  repita

    $t \leftarrow t + 1$

    $g_t \leftarrow \nabla_\theta \mathrm{f}_t(\theta_{t-1})$ (calcular os gradientes da função objetivo no tempo t)

    $m_t \leftarrow \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$ (atualizar a estimativa do primeiro momento)

    $v_t \leftarrow \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$ (atualizar a estimativa do segundo momento)

    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (computar a estimativa do primeiro momento)

    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (computar a estimativa do segundo momento)

    $\theta_t \leftarrow \theta_{t-1} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (atualizar parâmetros)

  até θ_t *não convergir*;

  RETORNE: θ_t (parâmetros resultantes)

fim

No Algoritmo 5 pode-se observar os parâmetros de configuração do Adam, que são: α como taxa de aprendizado, $beta_1$ como taxa de decaimento exponencial para as estimativas do primeiro momento, $beta_2$ como taxa de decaimento exponencial para as estimativas do segundo momento e o ϵ que representa um número pequeno para impedir que ocorra alguma divisão por zero na implementação. Seguindo o artigo original do otimizador Adam, são boas configurações de parâmetro: $\alpha = 0,001, \beta_1 = 0,9, \beta_2 = 0,999$ *e* $\epsilon = 10^{-8}$ (KINGMA; BA, 2014). O algoritmo trabalha a todo momento no objetivo de atualizar e melhorar seus parâmetros iniciais.

2.7 *Frechet Inception Distance*

A distância de *Frechet* faz parte do grupo de medidas de similaridade e dissimilaridade, hoje, muitos algoritmos de comparação de similaridade utilizam, por exemplo, a distância euclidiana, que é uma das mais utilizadas. Também existem outras distâncias consideradas

importantes nesse grupo, tais como: A distância *Manhattan* (*city-block*), *Mahalanobis*, *Minkowski*, *Chessboard*, *Bottleneck*, *Hausdorff* e a própria distância de *Frechet*, a qual foi selecionada para a etapa de validação do trabalho.

A FID (*Frechet Inception Distance*) é uma métrica utilizada na comparação entre imagens geradas e as imagens reais. A FID entre imagens reais e imagens geradas é calculada com a Equação 2.3.

$$FID(x, g) = ||\mu_x - \mu_g||^2 + Tr(\sum_1 + \sum_2 - 2(\sum_1 \sum_2)^{\frac{1}{2}}) \tag{2.3}$$

Onde μ_x e μ_g representam a média em termos de recursos das imagens reais e geradas, TR é o montante de todos os elementos diagonais, x representa as imagens reais e g as imagens geradas. Já $\sum_1$ e $\sum_2$ representam as matrizes de covariância dos vetores de recursos reais e gerados.

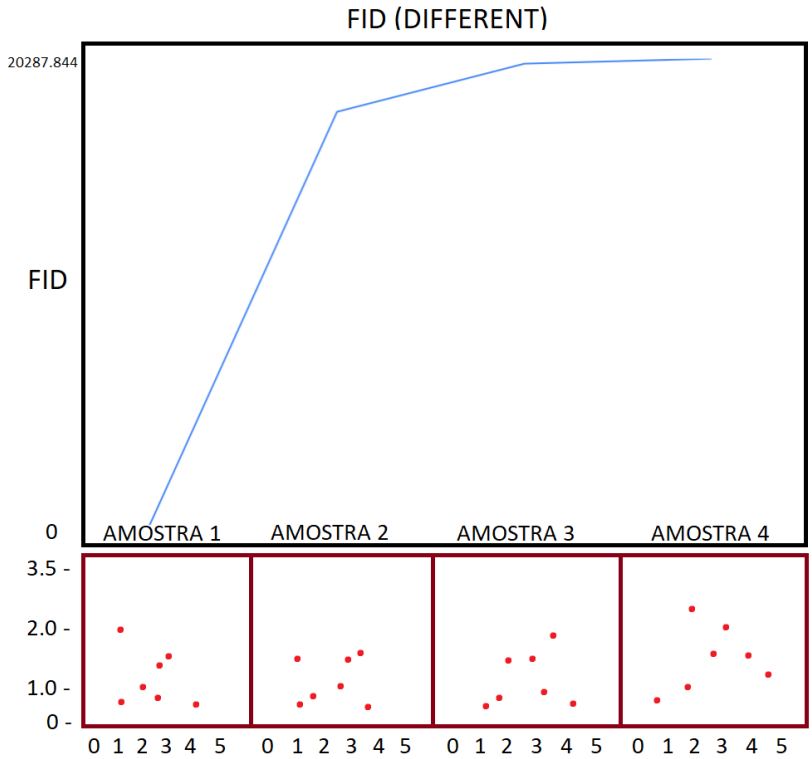


Figura 5 – Amostras de teste para exemplo do FID

As 4 amostras da Figura 5 foram geradas com o objetivo único de explicar de forma simples o funcionamento do algoritmo *FID*. Ao comparar as amostras 2, 3 e 4 com a amostra 1 (real), pode-se observar (ainda sem a utilização do algoritmo) que a amostra 2 da Figura 5 possui um maior grau de similaridade com a amostra real e por consequência dessa similaridade o algoritmo do *FID* obteve uma pontuação de diferença menor que as demais (Amostra 4: 20287.844, Amostra 3: 20285.755, Amostra 2: 20268.125). Por demonstrar efetividade e por estar sendo amplamente utilizado na validação de trabalhos

realizados com *GANs*, o algoritmo prova-se mais uma vez essencial para a validação do presente trabalho.

2.8 Fraudes em cartões de crédito

Segundo a *Encyclopedia Britannica*[1], o uso de cartões de crédito começou nos Estados Unidos durante a década de 1920, porém, apenas um grupo seleto de clientes poderiam tê-los em mãos.

Assim como em qualquer coisa, quanto maior o número de usuários, maiores serão os números de problemas que poderão ser encontrados e é aí que as fraudes em cartão de crédito começam a surgir, os primeiros sinais de fraude em escala apareceram no Reino Unido na década de 90 e o crescimento foi rápido, 1997: 122 milhões; 1998: 135 milhões; 1999: 188 milhões; 2000: 293 milhões (Association for Payment Clearing Services London, último acesso em 06/2020).

As formas como as fraudes acontecem são das mais diversas e a cada ano as formas de fazê-las mudam de forma drástica (DELAMAIRE; ABDOU; POINTON, 2009) o que prova que a melhor forma de identificá-las é fazendo uma boa utilização dos algoritmos de inteligência artificial, os padrões devem ser aprendidos de forma rápida para que não hajam muitos problemas.

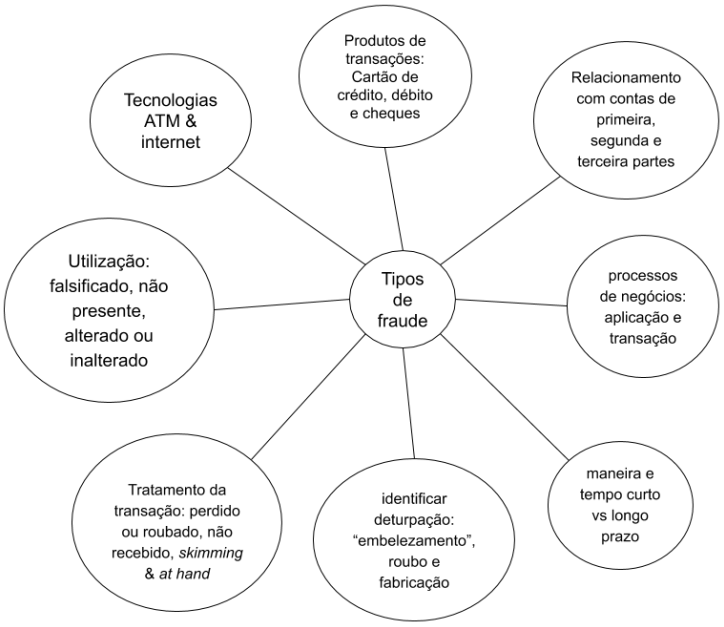


Figura 6 – Tipos de fraudes em cartão de crédito
Fonte: Traduzido de (DELAMAIRE; ABDOU; POINTON, 2009).

[1] https://www.britannica.com/topic/credit-card (último acesso em 06/2020)

3 *WGAN* Estocástico

O *WGAN* estocástico foi implementado visando comparar seus resultados com a implementação base do *WGAN*. Ambos os algoritmos foram aplicados utilizando um *dataset* de fraudes em cartões de crédito, porém, essa comparação pode e deve ser feita em qualquer *dataset* de fraude, onde a diferença entre as classes minoritárias e majoritárias sejam discrepantes, objetivando a criação de dados sintéticos para igualá-las. Nesse capítulo, as etapas de implementação do *WGAN* estocástico serão detalhadas posteriormente à implementação do *WGAN* que conta com a adaptação de um *notebook* disponibilizado no *Jupyter* (CODY NASH)[1]. Vale ressaltar também que todo o desenvolvimento em código do trabalho foi realizado no *Google Colab*[2].

3.1 *Dataset*

O *Dataset* do presente trabalho foi retirado do *Kaggle*[3], nele constam 284,807 transações em cartões de crédito, sendo que todos os recursos de 28 colunas (V1, V2,...V28) que podem indicar fraude ou não, sofreram transformação *PCA*, por conta da política de privacidade dos bancos com seus clientes os dados originais de transações não podem ser disponibilizados.

Das 284,807 transações disponibilizadas, apenas 492 são consideradas fraudes, representando 0.172% do total, tornando o aprendizado difícil nessas condições. Por conta desse problema, os algoritmos aplicados nas seções seguintes trabalham no objetivo de criar dados sintéticos com a finalidade de aproximar o número de fraudes ao de não fraudes.

3.2 *SMOTE* vs *WGAN*

Nessa seção será explicado o porquê da utilização do *WGAN* como parâmetro no desenvolvimento e experimentos do trabalho no lugar do *SMOTE* (um dos mais conhecidos algoritmos de *oversampling*).

Um dos principais motivos para a não utilização do *SMOTE* é que ele não é considerado efetivo quando se trata de grandes dimensões de dados (que acaba sendo o caso atual), além disso ele trabalha de forma "individual", ou seja, não leva em consideração as outras classes vizinhas, podendo ocasionar em sobreposição de classes, enquanto os métodos do

[1] https://nbviewer.jupyter.org/github/codyznash/GANs_for_Credit_Card_Data/
[2] https://colab.research.google.com/
[3] https://www.kaggle.com/mlg-ulb/creditcardfraud

WGAN aprendem com a distribuição geral das classes, o problema de sobreposição de classes que acontece com a utilização do *SMOTE* pode ser observado na Figura 7.

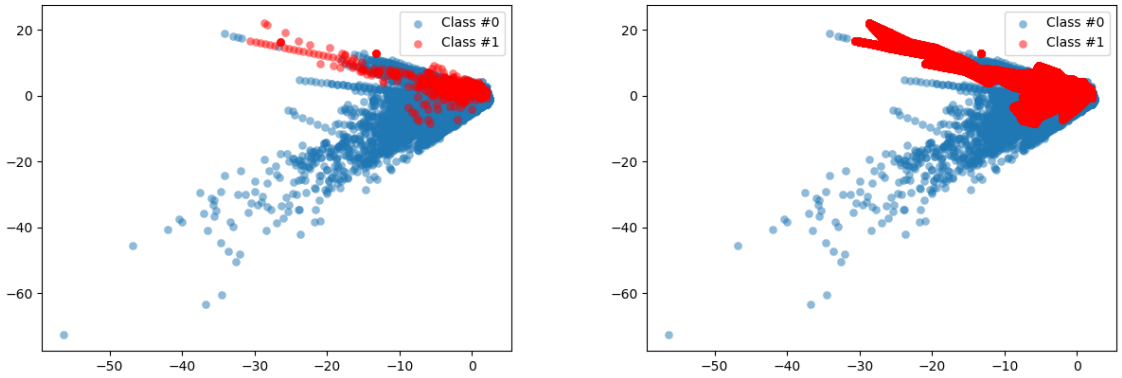


Figura 7 – Sobreposição de classes geradas pelo SMOTE
Fonte: (TOWARDS DATA SCIENCE).

Fora esses pontos, o *WGAN* também possui benefícios que tornam o seu uso atraente, como a utilização do *critic* e a sua otimização (mais detalhes sobre o *WGAN* e o uso do *critic* podem ser vistos na Seção 2.5).

Com essas informações, junto à análise de trabalhos relacionados (Capítulo 5), foi considerado a utilização do *WGAN* para o desenvolvimento desse trabalho.

3.3 Execução da *WGAN*

Para a execução da *WGAN* foi realizada uma adaptação do trabalho retirado do *Jupyter* (Cody Nash, 2018), essa adaptação permitiu uma melhor visualização e entendimento do código, enxugando-o ao mínimo necessário para a sua execução, removendo tudo referente a outros algoritmos que não fossem a *WGAN*.

Após a preparação do ambiente, com a importação das bibliotecas necessárias e importação do arquivo do *dataset* (Machine Learning Group - ULB, 2018). O trabalho seguido, já citado no ínicio dessa seção conta com um arquivo de apoio, o *GAN_171103.py*, nele estão contidos toda a base para execução de algoritmos que surgiram da GAN, como o *CGAN* e *WGAN*.

O objetivo desse trabalho é a comparação da *WGAN* com o algoritmo proposto, os demais foram descartados após a análise da literatura, constatando que o *WGAN* é o melhor dentre os citados. Para facilitar a visualização dos dados, os alvos da execução do algoritmo *WGAN* foram as colunas $V17$ e $V10$ do *dataframe real_samples*, que antes da execução do algoritmo sofreram uma transformação, onde todos os valores negativos se tornariam positivos, calculando a raiz quadrada da segunda potência de todos os valores

de ambas as colunas. No Algoritmo 6 pode-se observar os parâmetros para a execução do *WGAN*.

Algoritmo 6: Parâmetros WGAN

```
1  início
2      import GAN_171103
3      data_cols = [ i for i in train.columns if i not in label_cols ]
4      train_no_label = train[ data_cols ]
5      rand_dim = 32
6      base_n_count = 128
7      nb_steps = 500 + 1
8      batch_size = 128
9      k_d = 1
10     k_g = 1
11     critic_pre_train_steps = 100
12     log_interval = 100
13     learning_rate = 5e-4
14     data_dir = 'cache/'
15     generator_model_path, discriminator_model_path, loss_pickle_path =
         None, None, None arguments = [rand_dim, nb_steps, batch_size, k_d,
         k_g, critic_pre_train_steps, log_interval, learning_rate, base_n_count,
         data_dir, generator_model_path, discriminator_model_path,
         loss_pickle_path, show]
16     adversarial_training_WGAN(arguments, train_no_label,
         data_cols=data_cols )
17 fim
```

3.4 Tratamento dos dados utilizados no *WGAN* para execução do *WGAN* estocástico

Assim como os alvos da execução do *WGAN* foram as colunas $V17$ e $V10$, a adaptação para a execução do *WGAN* estocástico segue em cima delas. A execução do *WGAN* resultou em um segundo *dataframe*, o *gen_sample*, o qual conta com todos os dados gerados pelo algoritmo. Uma variável auxiliar *v_aux* foi criada para receber todos os dados do *gen_sample* que serão manipulados. Foi criada também uma lista *wgan* (olhar Algoritmo 7), que por meio de um *append* recebe os dados de todas as linhas da coluna $V10$ e $V17$, formando pares X e Y que podem ser plotados em um eixo cartesiano bidimensional.

Algoritmo 7: *Append X,Y*

```
1 início
2 |  i = 0
3 |  repita
4 |  |  wgan.juntar((v_aux[col2][i],v_aux[col1][i]))
5 |  |  i++
6 |  até i == v_aux.size();
7 fim
```

Seguida a execução do *append* dos valores de $V17$ e $V10$ foram criadas, então, duas listas *pontos_x* e *pontos_y* para receber os valores da lista *wgan*. Outras duas variáveis *eixo_x* e *eixo_y* também foram criadas para receber o somatório dos valores de *pontos_x* pela quantidade de elementos em *pontos_x* e a mesma coisa para *pontos_y*, esse cálculo tem como objetivo retornar a centroide dos elementos em questão.

3.5 Distância euclidiana e ato estocástico

O cálculo da distância euclidiana foi realizado para todos os valores da lista *wgan* com a centroide, como mostra o Algoritmo 8.

Algoritmo 8: Distância euclidiana

```
1 início
2 |  x = 0
3 |  repita
4 |  |  distancia_euclidiana < - junção da raiz quadrada de
  |  |    ((sqr(wgan[x][0]-centroide[0][0])) + (sqr(wgan[x][1]-centroide[0][1])))
5 |  |  x++
6 |  até x == 492;
7 fim
```

Após o cálculo e armazenamento da distância euclidiana de todos os valores, uma sequência de *if*, *elif's* e *else* dentro de uma estrutura de repetição foi criada para estabelecer a adição de valores aleatórios (ato estocástico) dentro de uma faixa pré-determinada baseada no tamanho da distância entre os valores de *wgan* e *centroide*, o resultado disso trouxe uma alteração nos valores do *dataframe gen_sample*.

3.5.1 Intervalo de valores baseados na distância euclidiana

Sabendo que a distância máxima é de 4,5 e todos os valores sofrem uma transformação positiva, os valores sorteados foram divididos em 9 intervalos entre 0 e 4,5. A escolha dos limites dos valores sorteados foi feita de forma que não causasse grandes alterações nos

valores gerados originalmente pela *WGAN*, por isso os valores não ultrapassam |0,9|. Foram realizados 3 testes diferentes aumentando o valores sorteados, o que causou efeito negativo no resultado, logo, para o trabalho em questão os valores definidos na Tabela 1 se encaixaram de forma mais assertiva.

Distância	Valor sorteado
{0 <= x <0,3}	{-0,1 <= x <= 0,1}
{0,5 <= x <1,0}	{-0,2 <= x <= 0,2}
{1,0 <= x <1,5}	{-0,3 <= x <= 0,3}
{1,5 <= x <2,0}	{-0,4 <= x <= 0,4}
{2,0 <= x <2,5}	{-0,5 <= x <= 0,5}
{2,5 <= x <3,0}	{-0,6 <= x <= 0,6}
{3,0 <= x <3,5}	{-0,7 <= x <= 0,7}
{3,5 <= x <4,0}	{-0,8 <= x <= 0,8}
{4,0 <= x <4,5}	{-0,9 <= x <= 0,9}

Tabela 1 – Distância e Valor sorteado

3.5.2 Ato estocástico

Enfim a execução do ato estocástico, escolhendo aleatoriamente o valor dentro do intervalo pré-definido na Tabela 1. Cada x e cada y recebeu um valor sorteado diferente (como mostra o Algoritmo 9), mesmo representados pela mesma faixa de intervalo de distância, um novo valor é sorteado para cada um.

Algoritmo 9: Ato estocástico

```
1  início
2      se (distancia_euclidiana[x] < 0,3): então
3          rand_number = valor aleatório entre (-0.1,0.1)
4          pontos_x[x] = (pontos_x[x] + rand_number)
5          gen_samples[col2][x] = pontos_x[x]
6          rand_number = valor aleatório entre (-0.1,0.1)
7          pontos_y[x] = (pontos_y[x] + rand_number)
8          gen_samples[col1][x] = pontos_y[x]
9          wgan_estocastico.juntar((pontos_x[x],pontos_y[x]))
10     fim
11 fim
```

4 Experimento e validação

Como forma de validação da escolha das colunas utilizadas como alvo no experimento, a matriz de correlação foi utilizada fornecendo informações sobre a influência dos recursos (dados das colunas) na determinação da fraude.

A Figura 8 representa a matriz de correlação obtida na realização desse trabalho, com ela pode-se observar a influência das colunas em classificar uma transação como fraude.

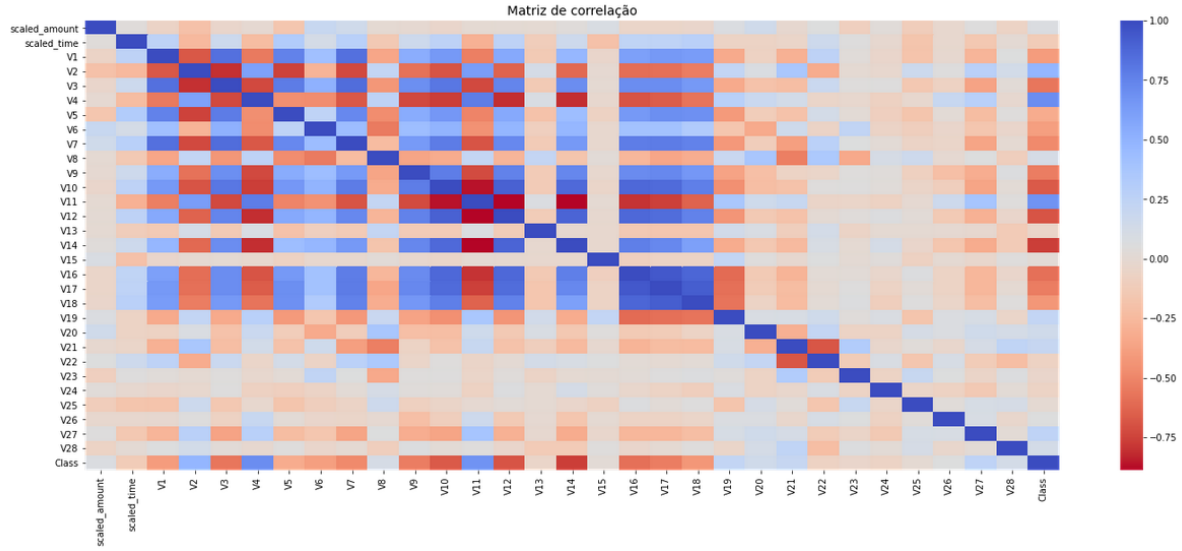


Figura 8 – Matriz de correlação

A matriz de correlação da Figura 8 foi retirada de uma subamostra gerada dos dados reais, com uma distribuição meio a meio dos casos de fraudes e não fraudes para fins únicos de observação.

Ao garantir a equivalência das classes igualando as classes majoritárias às classes minoritárias, algumas informações importantes foram adquiridas com a utilização da matriz de correlação como, por exemplo, as correlações negativas, que são aquelas que influenciam fortemente no resultado ($V10$, $V12$, $V14$ e $V17$), analisando que quanto menor forem os valores, maior é a chance do resultado final ser uma transação fraudulenta.

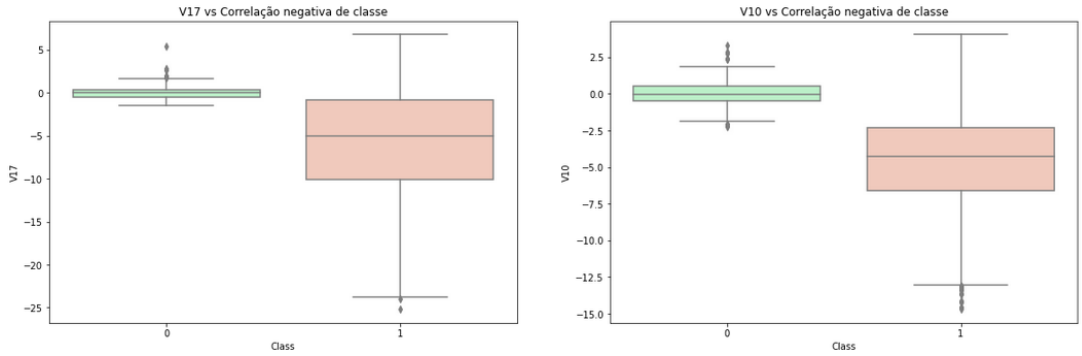


Figura 9 – $V10$ e $V17$ vs Correlação negativa de classe

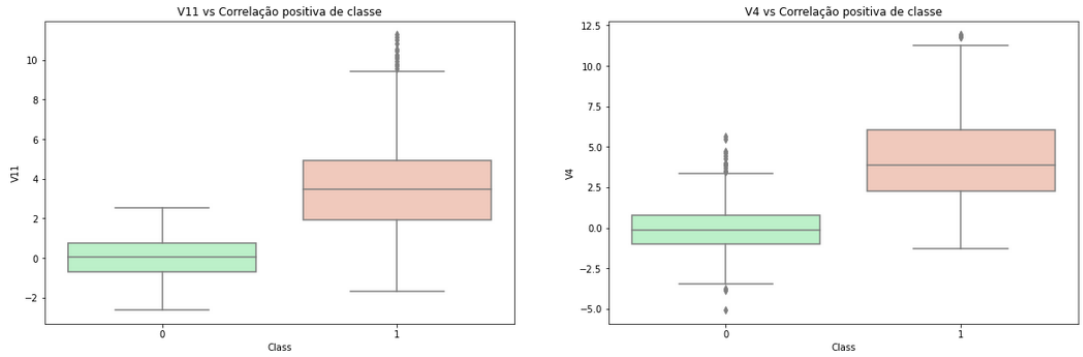


Figura 10 – $V11$ e $V4$ vs Correlação positiva de classe

Analisando a Figura 9 e a Figura 10 observa-se o impacto que as colunas $V10$ e $V17$ tem em relação a uma classe fraudulenta, diferentemente das colunas $V11$ e $V4$. Sendo assim para a realização de parte desse trabalho as colunas $V10$ e $V17$ foram selecionadas.

Esse trabalho visa garantir que através da estocasticidade, dados sintéticos do *WGAN* sejam esparsados com a finalidade de dar uma maior variabilidade. Para validar o experimento, foi implementado um algoritmo utilizando *FID* (*Frechet Inception Distance*, ver Seção 2.7).

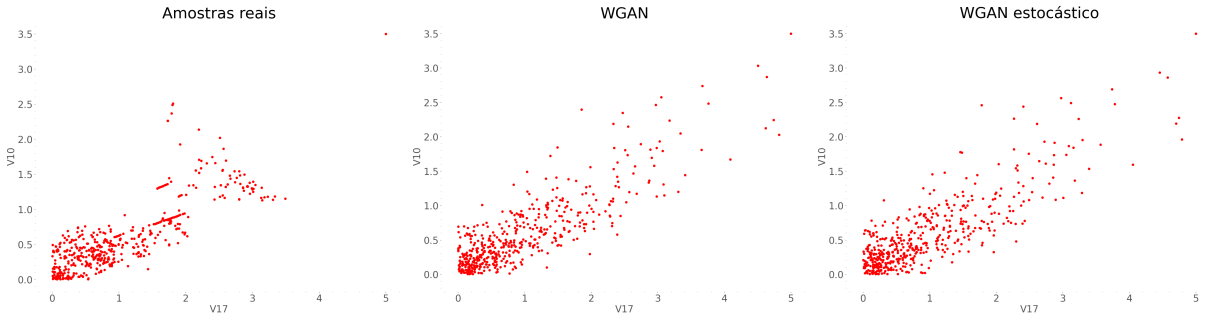


Figura 11 – Amostras dos dados do experimento

O *FID (different)* entre os dados reais e os dados gerados com o *wgan* foi de 478652.708. Já o *FID (different)* entre os dados reais e os dados gerados com o *wgan* em conjunto com a adição da estocasticidade foi de 478403.036. Quanto menor a pontuação de diferença,

melhor, isso significa que com a utilização da estocasticidade a diferença entre os dados reais e os dados gerados diminuíram, aumentando a assertividade do algoritmo *wgan*.

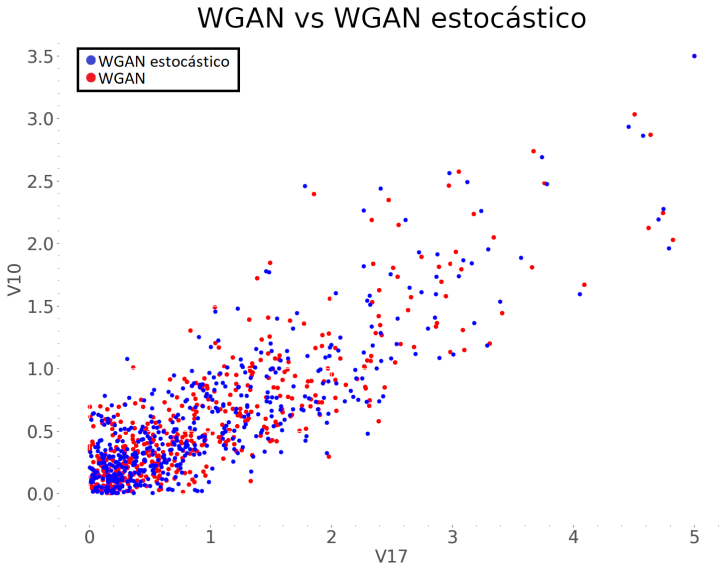


Figura 12 – Comparativo do *WGAN* com o *WGAN* estocástico

A Figura 12 é o resultado da junção do *WGAN* com o *WGAN* estocástico da Figura 11, nela pode-se observar o grau de similaridade e dissimilaridade entre os valores obtidos de ambos os algoritmos, tal figura também permite observar que os pontos, dos dois algoritmos, foram dispersos, alcançando o objetivo. Para constatar a diferença de similaridade entre as saídas dos algoritmos, foi realizado o cálculo de distância euclidiana entre todos os pontos sintéticos e o somatório dos valores obtidos foi igual a 37.72, indicando diferença entre ambas as saídas, isso comprova que o *wgan* estocástico possui em suas saídas um acréscimo de valor aleatório.

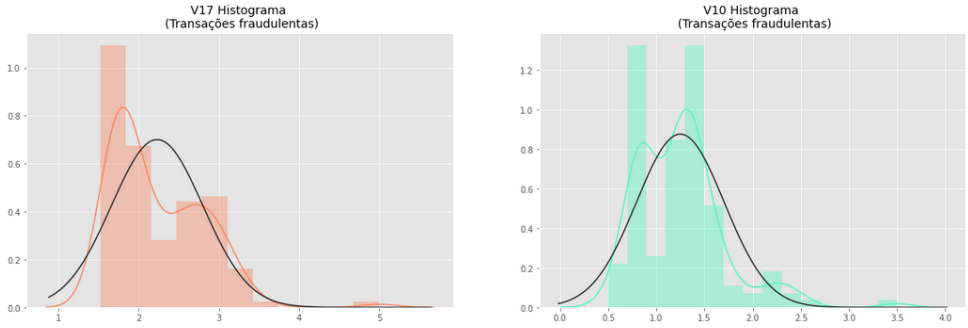


Figura 13 – Histograma dos dados reais

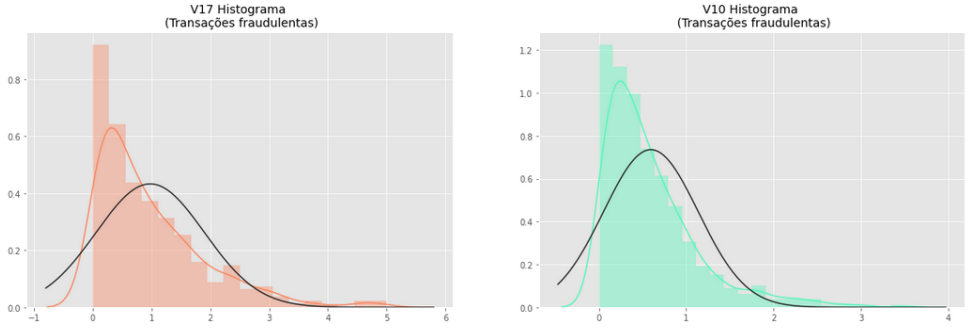


Figura 14 – Histograma dos dados gerados (*wgan*)

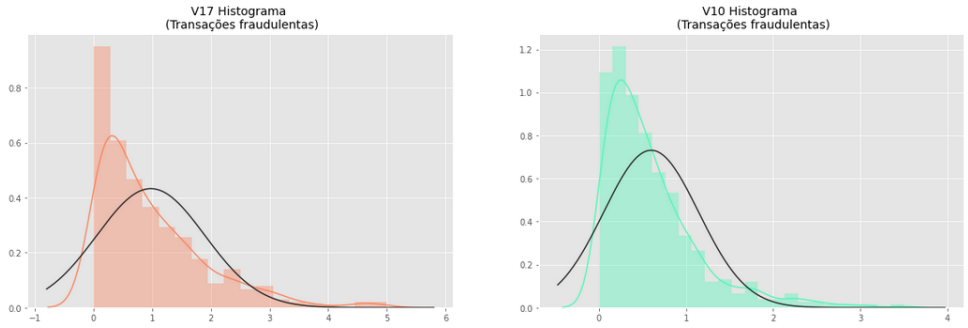


Figura 15 – Histograma dos dados gerados (estocasticidade)

Os histogramas das figuras 13, 14 e 15 foram gerados com finalidade de simplificar a visualização cartesiana da Figura 11. Tais gráficos permitem observar a discrepância da distribuição dos dados, principalmente nos dados reais, diminuindo essa diferença no histograma dos dados gerados com a utilização do *wgan* e mais ainda no histograma dos dados gerados com o *wgan* somados com um valor estocástico.

4.1 Síntese dos experimentos

Com a utilização da matriz de correlação, obtiveram-se informações cruciais que permitiram a escolha das colunas "chaves" para o prosseguimento do trabalho, tais colunas, através de uma análise cartesiana permitiram uma melhor observação da distribuição das amostras reais, que posteriormente foram comparadas com a execução do algoritmo *WGAN* e do algoritmo proposto pelo trabalho em questão. Além da execução do algoritmo *FID* para comprovar a melhora e a diferença entre o *WGAN* estocástico e o *WGAN*, uma nova projeção cartesiana foi gerada, entre ambos, permitindo uma comparação visual dos pontos. De ambas as formas pode-se comprovar diferença, mas a melhora mesmo que pequena foi comprovada com o resultado do *FID (different)* sendo menor entre o *WGAN* estocástico e as amostras reais, com a pontuação de 478403.036.

5 Trabalhos relacionados

Nesse capítulo serão abordados alguns trabalhos relacionados diretamente com o tema do trabalho proposto, envolvendo *datasets* de fraudes em cartão de crédito e a utilização de redes adversárias generativas como solução para tratar seu desbalanceamento das classes.

5.1 *Improving Detection of Credit Card Fraudulent Transactions using GAN*

Esse trabalho é focado na comparação do resultado da *generative adversarial network* (*GAN*) e de suas variações, como a *conditional generative adversarial network* (*CGAN*), *Wasserstein GAN* (*WGAN*) e *Wasserstein conditional GAN* (*WCGAN*). Realizado em cima do mesmo *dataset* de 31 colunas com de informações codificadas, incluindo a classe, 0 ou 1 que indica se há ou não fraude na transação, sendo indentificados 492 fraudes em um total de 284.807 transações de cartões de crédito. Esse *dataset* mostrou-se altamente desbalanceado, com apenas 0,172% de representação fraudulenta.

Nesse trabalho também foram utilizadas outras técnicas conhecidas de *oversampling* como o *ROS*, *SMOTE* e *ADASYN*. Comparando essas técnicas de *oversampling* com as *GANs*, observa-se que as *GANs* produziram melhores valores de balanceamento do *Recall* e *Precision*, resultando em um melhor *F1-Score*, tais dados podem ser observados na Tabela 2. Como resultado conclusivo desse trabalho, a *Wasserstein-GAN* provou-se mais uma vez ser mais estável, além de produzir amostras mais "reais" das classes minoritárias (fraudes).

	AUC	AUPRC	Recall	Precison	F1-Score	Rank
None	0,933	0,745	0,581	0,908	0,680	3,8
ROS	0,949	0,750	0,882	0,067	0,123	3,2
SMOTE	0,944	0,750	0,876	0,062	0,113	4,4
ADASYN	0,941	0,730	0,901	0,018	0,035	5,2
GAN	0,940	0,637	0,502	0,777	0,501	5,6
CGAN	0,901	0,631	0,564	0,643	0,444	6,4
WGAN	0,942	0,723	0,803	0,500	0,583	4,2
WCGAN	0,948	0,717	0,642	0,852	0,710	3,2

Tabela 2 – Desempenho dos algoritmos
Fonte: (BA, 2019)

5.2 *Using generative adversarial networks for improving classification effectiveness in credit card fraud detection*

Nessa seção, o trabalho proposto também é realizado em cima do mesmo *dataset*, com informações obtidas no decorrer de dois dias em setembro de 2013 por titulares de cartões europeus. Nesse trabalho foi realizado um redimensionamento e remoção de dados duplicados, resultando em um *dataset* com 446 transações fraudulentas de um total de 283726. O *dataset* ainda foi particionado em um conjunto de treinamento, sendo representado por dois terços dos dados, tendo, 315 fraudes de um total de 170236 transações representando 0.185%, restando 131 para o conjunto de testes, representando 0.115% do total.

No trabalho em questão, o objetivo é a comparação dos resultados da *GAN* com o *SMOTE* (os resultados dessa comparação podem ser vistos na Tabela 3), diferente do trabalho discutido na Seção 5.1, não houveram tratativas com outras derivações da *GAN*. Seguindo os resultados obtidos através do comparativo da *GAN* e do *SMOTE* observa-se uma diferença mínima, que aumenta (ainda de forma não muito significativa) de acordo com o número de amostras geradas. É possível que algumas modificações na rede tragam diferenças mais significativas, porém nada além foi realizado, modificações foram deixadas para serem implementadas em trabalhos futuros.

N_g	Precision	F-measure	Accuracy
	GAN x SMOTE	GAN x SMOTE	GAN x SMOTE
0	0,97872 0,97872	0,81778 0,81778	0,99964 0,99964
79	0,96842 0,96552	0,81416 0,81991	0,99963 0,99964
158	0,93069 0,97872	0,81034 0,81778	0,99961 0,99964
315	0,91346 0,93137	0,80851 0,81545	0,99960 0,99962
630	0,93204 0,96809	0,82051 0,80889	0,99963 0,99962
945	0,93137 0,95833	0,81545 0,81057	0,99962 0,99962
1260	0,93137 0,97872	0,81545 0,81778	0,99962 0,99964
2520	0,93137 0,97872	0,81545 0,81778	0,99962 0,99964
3150	0,93182 0,94949	0,81883 0,81739	0,99963 0,99963
6300	0,94059 0,97872	0,81897 0,81778	0,99963 0,99964
31500	0,95833 0,97872	0,81057 0,81778	0,99962 0,99964

Tabela 3 – *Precision, F-measure and Accuracy*
Fonte: (FIORE et al., 2019)

5.3 *Performance evaluation of class balancing techniques for credit card fraud detection*

Diferente dos trabalhos da Seção 5.1 e da Seção 5.2, esse, não utiliza a *GAN* ou suas derivações em seu desenvolvimento. Porém, utiliza também as técnicas de *oversampling*, como: *SMOTE*, *SMOTE ENN*, *SAFE SMOTE*, *ROS*, *SMOTE TL*. Nos trabalhos anteriores pode-se observar que o *SMOTE* por conta própria não obteve bons resultados se comparado com a *GAN* e suas derivações, porém, nesse trabalho, mesmo que não comparado com outros algoritmos fora do escopo, pode-se comparar o desempenho do *SMOTE* com seus derivados que tiveram um melhor resultado.

Concluiu-se então, nesse trabalho, que a partir dos experimentos realizados, como método de *oversampling* dentre os utilizados o *SMOTE ENN* obteve melhor desempenho se comparado a todos os outros.

5.4 *Adversarial learning in credit card fraud detection*

Assim como todos os outros trabalhos relacionados, esse também carrega a problemática das fraudes em cartões de crédito, utilizando abordagens que já foram tratadas em outras seções e outras que ainda não foram. Nesse trabalho foi desenvolvido uma forma de aprendizado adversário utilizando a teoria dos jogos modelando estratégias de adaptação, nesse caso o algoritmo adversário usa dessas estratégias como técnica aprimoramento para o classificador e não como método de *oversampling*, diferente das redes adversárias generativas tratadas ao longo do trabalho, com o objetivo de gerar amostras *fakes* para igualar as classes e assim fazer a classificação.

Como método de *oversampling*, nesse trabalho também foi utilizado o *SMOTE* para trazer equilíbrio entre as classes majoritárias (não fraudes) e minoritárias (fraudes).

5.5 *WiP: Generative Adversarial Network for Oversampling Data in Credit Card Fraud Detection*

Mais um trabalho que utilizou a *GAN* e o *WGAN* na geração de dados sintéticos de transações fraudulentas em cartões de crédito. Nesse trabalho também comprovou-se que a utilização da *GAN* e sua variação obteve além de um bom *F1-Score*, obteve também uma redução significativa na contagem de falsos positivos, isso comparado com outros métodos de *oversampling* também já visto em outras seções, o *SMOTE*, *ADASYN* e o *random oversampling*.

O trabalho apresenta, no geral, uma abordagem extremamente interessante e apresenta ideias interessantes que devem ser postas em práticas em trabalhos futuros, propondo utilizar uma arquitetura diferente para o gerador na *GAN*.

Method	TP	FP	Specificity	Precision	Recall	F1-score	AUX
Plain SVM	145	2951	0,97	0,05	0,94	0,09	0,95
Random oversampling + SVM	139	1046	0,99	0,12	0,9	0,21	0,94
SMOTE + SVM	145	1595	0,98	0,08	0,94	0,15	0,96
ADASYN + SVM	145	4874	0,94	0,03	0,94	0,06	0,94
GAN + SVM	135	170	0,99	0,58	0,85	0,69	0,93
WGAN + SVM	132	95	0,99	0,58	0,85	0,69	0,92
SMOTE + GAN + SVM	142	512	0,99	0,22	0,92	0,35	0,96
SMOTE + WGAN + SVM	141	422	0,91	0,25	0,91	0,39	0,95

Tabela 4 – Comparação de performance usando *SVM*
Fonte: (GANGWAR; RAVI, 2019)

5.6 Síntese dos trabalhos relacionados

Observa-se, com clareza, através dos trabalhos relacionados, de relevância e recentes, que algoritmos como as *generative adversarial networks* e seus derivados (*CGAN*, *WGAN* e outros) vem fortemente substituindo outras técnicas de *oversampling* conhecidas no âmbito de *machine learning*, como o *ADASYN*, *random oversampling*, *SMOTE* e seus derivados (*SMOTE ENN, SAFE SMOTE*, *SMOTE TL* e outros).

Todos os trabalhos desse capítulo utilizaram o mesmo *dataset* em seus desenvolvimentos e aqueles que utilizaram dos mesmos algoritmos para a mesma finalidade, alcançaram resultados parecidos, em maioria, todos os que utilizaram o *WGAN*, consideraram-o, dentre os comparativos feitos, como o melhor algoritmo a ser utilizado.

De forma geral o objetivo dos trabalhos relacionados citados é fazer a comparação entre algoritmos de geração de dados sintéticos já existentes. Nesses trabalhos não há uma tentativa de otimização nos algoritmos ou uma tentativa de superar os resultados dos algoritmos já existentes com possíveis novos algoritmos implementados.

6 Conclusões

O objetivo do trabalho foi de comparar o resultado da execução do algoritmo *WGAN* com o resultado da execução do mesmo com a adição de valores estocásticos. Através do algoritmo que calcula o *FID (Frechet Inception Distance)* percebe-se que mesmo que pequena, houve uma melhora no resultado da execução do algoritmo com a utilização da soma de valores estocásticos(*WGAN* vs real: 478652.708, *WGAN* estocástico vs real: 478403.036), gerando uma maior variabilidade.

O experimento não trouxe uma grande discrepância nos resultados, mas uma boa variabilidade, com bons resultado, levando em consideração o pouco custo que a sua implementação e execução obteve.

Esse experimento abre portas para explorar o processo de estocasticidade na otimização de algoritmos. A alteração dos parâmetros utilizados na Tabela 1 permite tentativas de melhora nos resultados atuais.

O mais importante é mostrar que é possível acrescentar elementos aleatórios nos dados sintéticos, mantendo ainda assim uma organização dos dados que se assemelham aos dados reais, mostrando a eficácia da estocasticidade na otimização da *WGAN*. O trabalho também abre portas para investigar e realizar novos experimentos em cima de outros algoritmos derivados da *GAN*.

Referências Bibliográficas

ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. *Wasserstein GAN*. 2017. 23, 24

BA, H. *Improving Detection of Credit Card Fraudulent Transactions using Generative Adversarial Networks*. 2019. 37

BEN-HUR, A.; WESTON, J. A user's guide to support vector machines. *Methods in molecular biology (Clifton, N.J.)*, v. 609, p. 223–39, 01 2010. 18

BLACKWELL, D. et al. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, Pacific Journal of Mathematics, v. 6, n. 1, p. 1–8, 1956. 22

CHAWLA, N. et al. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, v. 16, p. 321–357, 01 2002. 16, 19

Cody Nash. *GAN comparison on Kaggle Credit Card Fraud Data*. 2018. <https://nbviewer.jupyter.org/github/codyznash/GANs_for_Credit_Card_Data/tree/master/>, Último acesso em 29-04-2020. 29

DELAMAIRE, L.; ABDOU, H.; POINTON, J. Credit card fraud and detection techniques: A review. *Banks and Bank Systems*, v. 4, 01 2009. 27

FIORE, U. et al. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, Elsevier, v. 479, p. 448–455, 2019. 38

FROGNER, C. et al. Learning with a wasserstein loss. In: CORTES, C. et al. (Ed.). *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015. p. 2053–2061. Disponível em: <http://papers.nips.cc/paper/5679-learning-with-a-wasserstein-loss.pdf>. 23

GANGWAR, A. K.; RAVI, V. Wip: Generative adversarial network for oversampling data in credit card fraud detection. In: SPRINGER. *International Conference on Information Systems Security*. [S.l.], 2019. p. 123–134. 40

GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2672–2680. 16, 21, 22

HU, F.; LI, H. A novel boundary oversampling algorithm based on neighborhood rough set model: Nrsboundary-smote. *Mathematical Problems in Engineering*, v. 2013, 11 2013. 20

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2014. 24, 25

MACHADO, E. L. Um estudo de limpeza em base de dados desbalanceada e com sobreposição de classes. 2007. 21

Machine Learning Group - ULB. *Credit Card Fraud Detection*. 2018. <https://www.kaggle.com/mlg-ulb/creditcardfraud>, Último acesso em 29-04-2020. 16, 29

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, v. 1, n. 1, p. 32, 2003. 18, 19

RISH, I. An empirical study of the naïve bayes classifier. *IJCAI 2001 Work Empir Methods Artif Intell*, v. 3, 01 2001. 18

RODRIGUES, C. A. d. S. P. et al. Implementacao de redes convolucionais para a segmentacao de imagens em tempo real com vistas a aplicacao em robos autonomos com dispositivos de visao de baixo custo. Universidade Federal de Goias, 2018. 25

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, 04 2014. 18

SHI, N.; LIU, X.; GUAN, Y. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In: . [S.l.: s.n.], 2010. p. 63–67. 18

WENG, L. *From GAN to WGAN*. 2019. 23

ZHOU, P.-Y.; CHAN, K. A model-based multivariate time series clustering algorithm. In: . [S.l.: s.n.], 2014. 19

ZHOU, Z. et al. Lipschitz generative adversarial nets. *arXiv preprint arXiv:1902.05687*, 2019. 23