FACULDADE DE TECNOLOGIADE SÃO PAULO

RAPHAEL KENJI MOREIRA YAMANAKA

RA 17101359

SISTEMA DE RECOMENDAÇÃO COM PERSONALIZAÇÃO

São Paulo

2019

FACULDADE DE TECNOLOGIA DE SÂO PAULO

RAPHAEL KENJI MOREIRA YAMANAKA

SISTEMA DE RECOMENDAÇÃO COM PERSONALIZAÇÃO

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Analise e Desenvolvimento de Sistemas.

Orientador: Professor Silvio do Lago Pereira

São Paulo

2019

FACULDADE DE TECNOLOGIA DE SÃO PAULO

RAPHAEL KENJI MOREIRA YAMANAKA

SISTEMA DE RECOMENDAÇÃO COM PERSONALIZAÇÃO

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador

_____

_____

_____

Conceito/Nota Final: _____

Atesto o conteúdo contido na mídia entregue pelo aluno e assinada por mim para avaliação do TCC.
Estou ciente de que se o aluno não tiver entregado a mídia conforme regras do Roteiro ele estará reprovado na disciplina mesmo que esteja aprovado por mim.

Orientador: Professor Silvio do Lago Pereira

SÃO PAULO, ____ de _____ de 2019.

Assinatura do Orientador

Resumo

O presente trabalho tem por objetivo criar um software com Inteligência Artificial capaz de customizar o design de produtos de acordo com o perfil do cliente. O software receberá imagens como base e irá “transformá-las” e retornará uma imagem de como ficaria o produto final, realizando pequenas modificações até o cliente estar satisfeito com a versão final. Esta é uma monografia que tem um objetivo descritivo e explicativo e com um delineamento pesquisa-ação e experimental, pois para o desenvolvimento do protótipo do sistema foram realizados diversos testes com diferentes tipos de imagens durante todo o projeto.

Palavras-chave: Inteligência Artificial. Customização. Personalização. Software.

Abstract

The present work aims to create an Artificial Intelligence software capable of customizing product design according to customer profile. The software will receive images as a base and will "transform" them and return an image of what the final product would look like, making minor modifications until the customer is satisfied with the final version. This is a monograph that has a descriptive and explanatory objective and with an action-research and experimental design, because for the development of the system prototype several tests were performed with different types of images throughout the project.

Keywords: Artificial Intelligence. Customization. Personalization. Software

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

1.1. PROBLEMA

O mercado está mudando nas últimas décadas, bem como o perfil dos consumidores. Eles exigem mais que as marcas personalizem seus produtos para atender a seus gostos [16]. De acordo com a Equipe Reamp (2018, **apud Pure360, 2018**) a personalização básica falha em engajar seus consumidores, apenas 8% se sentem encorajados a engajar com uma marca se forem abordados pelo seu primeiro nome e apenas 7% se receberem um e-mail de aniversário, mesmo assim muitas marcas ainda utilizam essas formas básicas de personalização para envolver seus clientes [5].

A Equipe Reamp (2018, **apud Econsultancy, 2018**) também cita que a personalização está em quinto lugar e prioridade das marcas com apenas 7% das organizações as escolhendo como primeira prioridade. De acordo coma Infosys (2013) 31% dos consumidores querem que a personalização seja maior do que é. [5] [9]

Podemos observar também que os consumidores estão querendo mais personalização, pois segundo a Salesforce (2016) 57% dos consumidores estão dispostos a compartilhar informações por ofertas personalizadas e descontos e que 52% estão dispostos a mudar de marca se ela não fizer algum esforço para personalizar a comuniação entre eles. [23]

1.2. MOTIVAÇÃO

Quando se analisa os sites que têm ganhado destaques atualmente, principalmente com os jovens, encontra-se um ponto em comum, a personalização. Sites como Amazon, Youtube e Netflix analisam o comportamento do usuário, mapeiam o “gosto do cliente” e sugerem produtos/entretenimento semelhante ao adquirido pelo usuário. [27]

Atualmente as pessoas estão ficando mais exigentes quanto aos serviços que adquirem, elas desejam o produto da forma que imaginaram o mais rápido possível, a empresa Adobe, desenvolvedora do software

Photoshop, já informou que 77% dos profissionais de marketing acreditam que personalizar o conteúdo para o cliente em tempo real é algo imprescindível. [20]

A personalização com o perfil do cliente é um ponto muito forte para aumentar as vendas, segundo a Infosys (2013) 86% dos consumidores disseram que a personalização afetou de algum modo na compra, 59% disseram que a personalização afetou notavelmente e 25% disse que a personalização afetou significantemente na compra. Segundo a Equipe Reamp (2018, **apud Delloitte**) 36% dos consumidores expressaram interesse em comprar produtos ou serviços personalizados e 48% disse que estão dispostos a esperar mais por produtos ou serviços personalizados. [9] [5]

Com personalização é possível:

- Cobrar mais pelo produto ou serviço: De acordo com a Salesforce (2016) dois terços dos clientes até pagam premium para as empresas para receber experiencias superiores. [22]
- Lealdade dos consumidores: Segundo a Saleforce (2016) 70% dos clientes dizem que uma empresa entendendo suas necessidades individuais influencia na sua lealdade e 69% diz o mesmo a respeito do atendimento ao cliente.

1.3. OBJETIVO

Diante desses fatos, o objetivo deste trabalho é criar um software com Inteligência Artificial capaz de customizar o design de produtos de acordo com o perfil do cliente. O software irá receber uma imagem e transformá-la em uma foto fazendo pequenos ajustes até o cliente ficar satisfeito com o resultado. Ele terá como objetivo customizar produtos já levemente customizados por um profissional de design. Com essa personalização, a empresa conseguirá aumentar o relacionamento com o cliente.

1.4. ORGANIZAÇÃO

No Primeiro Capítulo: **Fundamentos Teóricos**, iremos discutir alguns conceitos que usaremos neste trabalho, a maioria sendo sobre Inteligência Artificial. O capitulo ira explicar desde conceitos mais gerais como o que é Inteligência Artificial, até conceitos mais específicos que como métodos e algoritmos que utilizaremos no desenvolvimento do projeto. Este capitulo é muito importante para quem não tem um conhecimento sobre Inteligência Artificial, explicando seus detalhes.

O Segundo Capítulo: **Aplicação**, iremos explicar como o software foi feito mostrando detalhes do seu código e utilizando os métodos mencionados no capítulo anterior. Iremos explicar também como a Inteligência Artificial do software funciona.

No Terceiro Capítulo: **Experimentos**, iremos mostrar resultados do software, analisá-los e explicar como chegamos a esses resultados.

2. FUNDAMENTOS TEÓRICOS

Este capítulo irá apresentar alguns conceitos teóricos que iremos utilizar no ao longo deste TCC. Eles serão importantes para um melhor entendimento de todo tema e assumindo que o leitor não tenha um conhecimento profundo dos conceitos. A explicação não será completa devido ao alto grau de complexidade de alguns conceitos, mas será satisfatória para o entendimento.

2.1. INTELIGÊNCIA ARTIFICIAL (IA)

Inteligência Artificial pode parecer um campo novo na tecnologia, mas sua fundação começou séculos atrás, quando Aristóteles inventou o pensamento lógico e o termo décadas atrás por John McCarthy em uma conferência no Dartmouth College, em Hanover, New Hampshire em 1956. [2]

Mesmo com essa área crescendo ultimamente, ainda é difícil definir precisamente o que é IA porque não existe uma definição exata, o que gera muitas dúvidas e confusões. Por exemplo, segundo Ben Coppin (2004, p. 4, tradução nossa) "Inteligência Artificial envolve em utilizar métodos baseados no comportamento inteligente dos humanos e outros animais para solucionar problemas complexos"[1] e segundo Amit Konar (2000, p. 28, tradução nossa) IA é uma "simulação da inteligência humana em uma máquina para torna-la eficiente em identificar e usar o "Conhecimento" certo em um determinado passo da solução de um problema"[2] .

Um teste muito famoso para determinar se uma máquina é capaz de pensar ou não é o Teste de Turing. Quem criou esse teste foi Alan Turing (por isso o nome Teste de Turing) que ajudou a decifrar o código alemão na Segunda Guerra Mundial. Após a guerra, ele escreveu um artigo chamado "Computing Machinery and Intelligence" em que propôs esse teste. O teste tinha base na ideia que se um humano interrogasse

[1] "Artificial Intelligence involves using methods based on the intelligent behavior of humans and other animals to solve complex problems."
[5] "simulation of human intelligence on a machine, so as to make the machine efficient to identify and use the right piece of "Knowledge" at a given step of solving a problem"

uma máquina e não conseguisse distinguir entre um humano ou uma máquina, então ela era inteligente. [2]

O teste consistia em dois humanos e um computador. Um humano interrogaria o outro humano e a máquina ao mesmo tempo sem interagir diretamente com nenhum dos dois e seu trabalho seria identificar qual é a máquina e qual é o humano. O humano interrogado deveria ajudar o interrogador, mas se a máquina fosse inteligente o suficiente, ele conseguiria enganar o interrogador, passando, assim, no teste. [2]

Porém mesmo se uma máquina passar no teste de Turing, para muitos, não significa que ela é inteligente ou pensa. Existem dois tipos de pensamentos sobre esse tema: IA Fraco e IA Forte. O primeiro diz que uma máquina nunca será capaz de pensar e ser inteligente, mas sim de somente imitar pensamento e inteligência. Já o segundo diz que as máquinas são sim capazes de pensar de tiverem dados e poder de processamento suficiente [2]. Mesmo passando no teste de Turing, alguns acreditam que para uma máquina pensar é preciso que ela tenha Consciência, ou seja, que ela saiba a sua existência e das suas ações; Fenomenologia, ou seja, que ela tenha emoções; ou, ainda, que ela tenha Intencionalidade, ou seja, que as crenças e desejos da máquina venha do mundo real. [21]

Originalmente, IA era utilizada somente na área de jogos e de prova de teoremas, mas com a ajuda e avanço da Filosofia, Psicologia, Ciência Cognitiva, Ciência Computacional, Matemática e Engenharia, a área de atuação da IA ampliou muito, alguns exemplos é processamento de imagens e linguagem natural, navegação e robótica. [13]

2.1.1. APRENDIZADO DE MÁQUINA (MACHINE LEARNING)

É um subcampo da IA que visa fazer com que a máquina aprenda. Ao darmos dados e informações sobre esses dados para um algoritmo, ele é capaz de encontrar padrões nos dados e generalizar esses dados, e com essa generalização o algoritmo cria um modelo que é capaz de determinar aspectos importantes da data e que consegue fazer predições sobre dados novos que nunca viu antes. [24]

Para entender melhor vamos supor que você recebe muitos e-mails que contenha "ganhe dinheiro fácil", como um ser humano, você consegue facilmente entender que qualquer e-mail que contenha "ganhe dinheiro fácil" é spam. Isso é uma generalização, você criou um modelo do que é spam. A mesma coisa acontece com um algoritmo de aprendizado de máquina, com base em dados de o que é spam e o que não é ele é capaz de aprender e criar um modelo para spam.

Machine Learning (ML) pode ser classificado em três categorias:

- Aprendizado Supervisionado: Quando há um treinador humano que já sabe as respostas para os inputs que alimenta a máquina. Conforme a máquina erra as respostas, ela adapta os parâmetros do algoritmo para gerar o output desejado.
- Aprendizado Não Supervisionado: Quando não há um treinador humano nem um dataset com outputs conhecidos. A máquina deve alterar seus parâmetros experimentando e agrupando os dados por um padrão desconhecido.
- Aprendizado por Reforço: Também não há um treinado humano nem outputs conhecidos como no aprendizado não supervisionado, mas a cada output da máquina, ela recebe uma recompensa ou punição e então ajusta seus parâmetros. [13] [26]

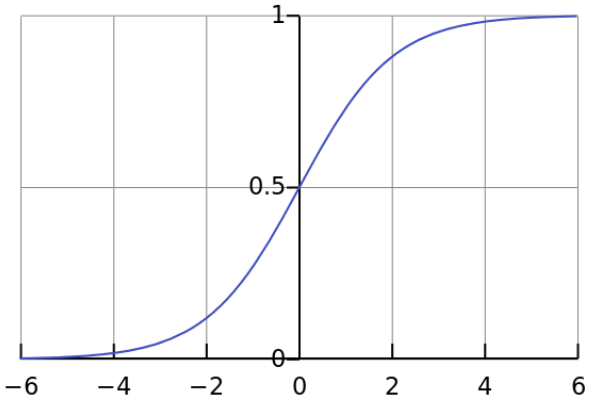
Porém ainda existem desvantagens em se utilizar ML, como a eficiência dos algoritmos em generalizar padrões varia muito e a probabilidade de interpretar erradamente um padrão desconhecido é muito grande. Isso porque ele só tem como base os dados enviados a ele enquanto um ser humano tem muito mais dados e tempo vindos de conhecimento culturais e experiência de vida para analisar novas ocasiões. Outro problema é generalizar muito, seguindo o exemplo de spam, você pode receber um e-mail com a frase "ganhe dinheiro fácil" de algum amigo que pareça um spam mais não é. Você teria que avisar o algoritmo que e-mails vindos do seu amigo não são spam. [24]

2.1.1.1. FUNÇÕES DE ATIVAÇÃO (ACTIVATION FUNCTION)

As funções de ativação, como o próprio nome já diz, são funções utilizadas para decidir se determinado neurônio irá ativar ou não. Elas são usadas por redes neurais para que elas possam aprender sobre algo muito complexo e não-linear [31]. É importante que a função de ativação não seja linear porque se for, quando a rede neural tiver camadas ocultas, o resultado de uma camada continua na outra, então não importa quantas camadas tiverem, ela será equivalente a somente uma camada [25]. Existem vários tipos de funções de ativações como por exemplo Sigmoid, TanH, ReLu e outros.

- Sigmoid: é muito utilizado e ele ajuda a mapear o output entre os valores 0 e 1, sendo a maior diferença com o input entre os números -2 e 2. [25]

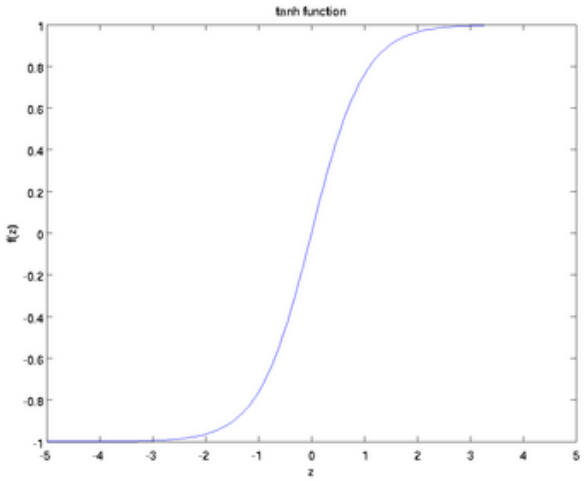
Figura 1 – Valores do output da função de ativação Sigmoid



Fonte: Avinash Sharma (2017) [Medium]

- TanH: é parecido com o Sigmoid, a diferença está que ao invés de mapear o output entre os valores 0 e 1, ele mapeia entre -1 e 1 e seu gradiente é maior. [25]

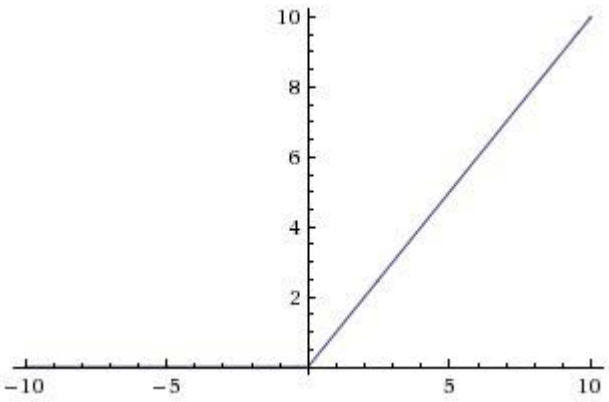
Figura 2 - Valores do output da função de ativação TanH



Fonte: Avinash Sharma (2017) [25]

- ReLu: como visto nas figuras anterios, quando ao valor do input é muito alto ou muito baixo, a valor do output não alera significantemente, para contornar essa situação utlia-se o ReLu. O intervalo de valores dele é de 0 ao infinito e ele exige menos poder de processamento que as outras duas funções. [25]

Figura 3 - Valores do output da função de ativação TanH



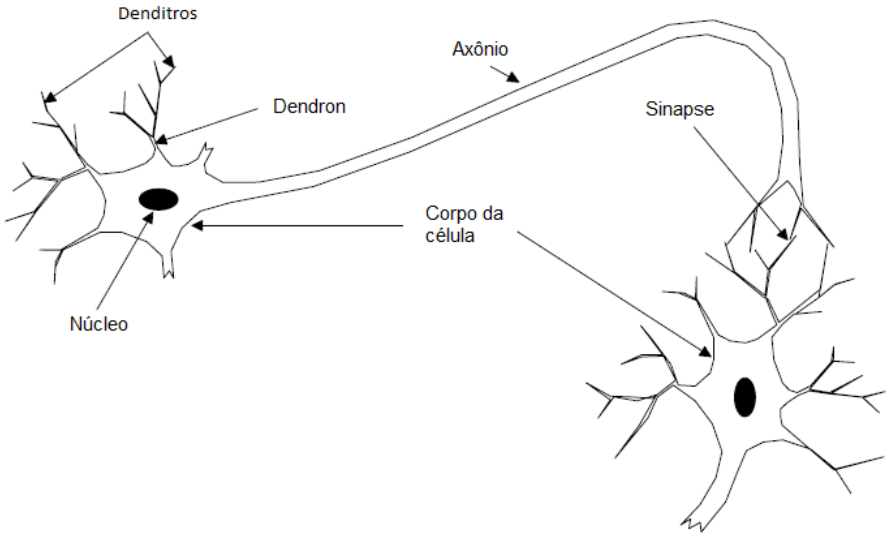
Fonte: Avinash Sharma (2017) [25]

2.1.1.2. PERCEPTRON

O Sistema nervosa humano consiste em pequenas unidades celulares chamadas de neurônios que quando conectados formam uma fibra nervosa. Uma rede neural biológica é formada dessas fibras. [13]

Um neurônio é composto por quatro elementos principais: o dendrito, que é um receptor, ele recebe sinais vindos de outros neurônios; o corpo da célula, que é onde os sinais é processado; o axônio, que é onde se passam os sinais para outro neurônio após ser processado pelo corpo; e pôr fim a sinapse, que é responsável por controlar o fluxo de um neurônio para outro vizinho. [13]

Figura 4 - Representação de um neurônio biológico



Fonte: Adaptado de Amit Konar (2000)

Inventado em 1957 por Frank Rosenblatt no laboratório Aeronáutico Cornell, Shiffman (2012, p.448, tradução nossa) define o perceptron como "um perceptron é a rede neural mais simples possível: um modelo computacional de um único neurônio. O perceptron consiste em um ou mais inputs, um processador e um único output"[3] [26]. O perceptron tenta imitar um neurônio biológico recebendo sinais, processando e enviados a outros neurônios. [13] Cada input tem seu

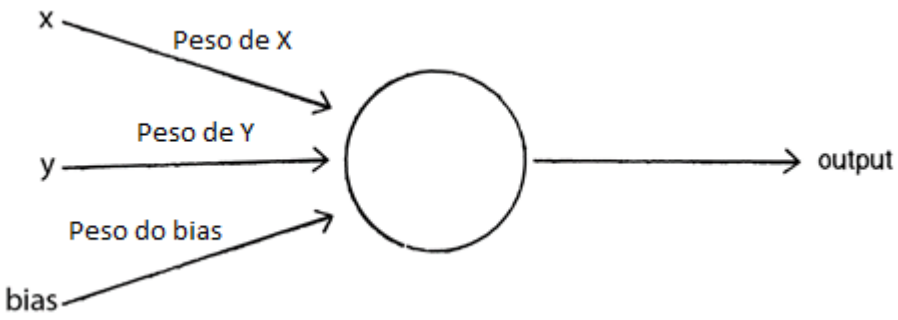
[3] "[…] a perceptron is the simplest neural network possible: a computational model of a single neuron. A perceptron consists of one or more inputs, a processor, and a single output."

peso representando sua importância [17]. Após calcular os inputs com seus respectivos pesos, adiciona-se o bias, eles passam por uma função de ativação e, por fim, gera-se o output.

O bias é um input constante “extra” que serve para ajudar a rede neural a ser mais flexível. Ele ajuda a controlar o valor que a função de ativação irá ativar. [geeksforgeeks] [Quora]

Para entender melhor vamos supor um perceptron simples que recebe dois inputs e retorna um output e sua função de ativação é a de passo binário, ou seja, se o input for negativo retorna 0, e for positivo retorna 1. Nosso perceptron pode ser representado como:

Figura 5 - Ilustração de um perceptron



Fonte: Adaptado de Daniel Shiffman (2012)

Então o cálculo do output seria se x * peso de x + y * peso de y + bias * peso de bias > 0 então 1, senão 0.

Para deixar simples, vamos supor que o peso de x é 3, o peso de y é 2 e o bias é 1 com o peso 2. Assim para x = -3 e y = 5 teremos: -3 * 3 + 5 * 2 + 1 * 2 = 3, então o output seria 1. Nesse caso os valores dos pesos e do bias foi aleatório, mas para o perceptron realmente aprender, esses valores deverão ser ajustados conforme ele for treinando.

2.1.1.3. REDES NEURAIS

As redes neurais artificiais são baseadas nas redes neurais biológicas para resolver certos problemas [26]. O cérebro humano é formado por inúmeros de neurônios, o que o torna capaz de processar

níveis de complexidade muito maiores do que são possíveis hoje com máquinas. [13]

Para esse trabalho iremos chamar as redes neurais artificiais de apenas redes neurais pois já fica claro que se trata da artificial pelo contexto.

Para Amit Konar (2000, p. 284, tradução nossa):

> "Redes neurais consistem em alguns nós, em que cada pode ser pensado como uma representação de um neurônio. Normalmente, esses neurônios são organizados em camadas e os neurônios de uma camada estão conectados nos neurônios das duas camadas de cada lado"[4].

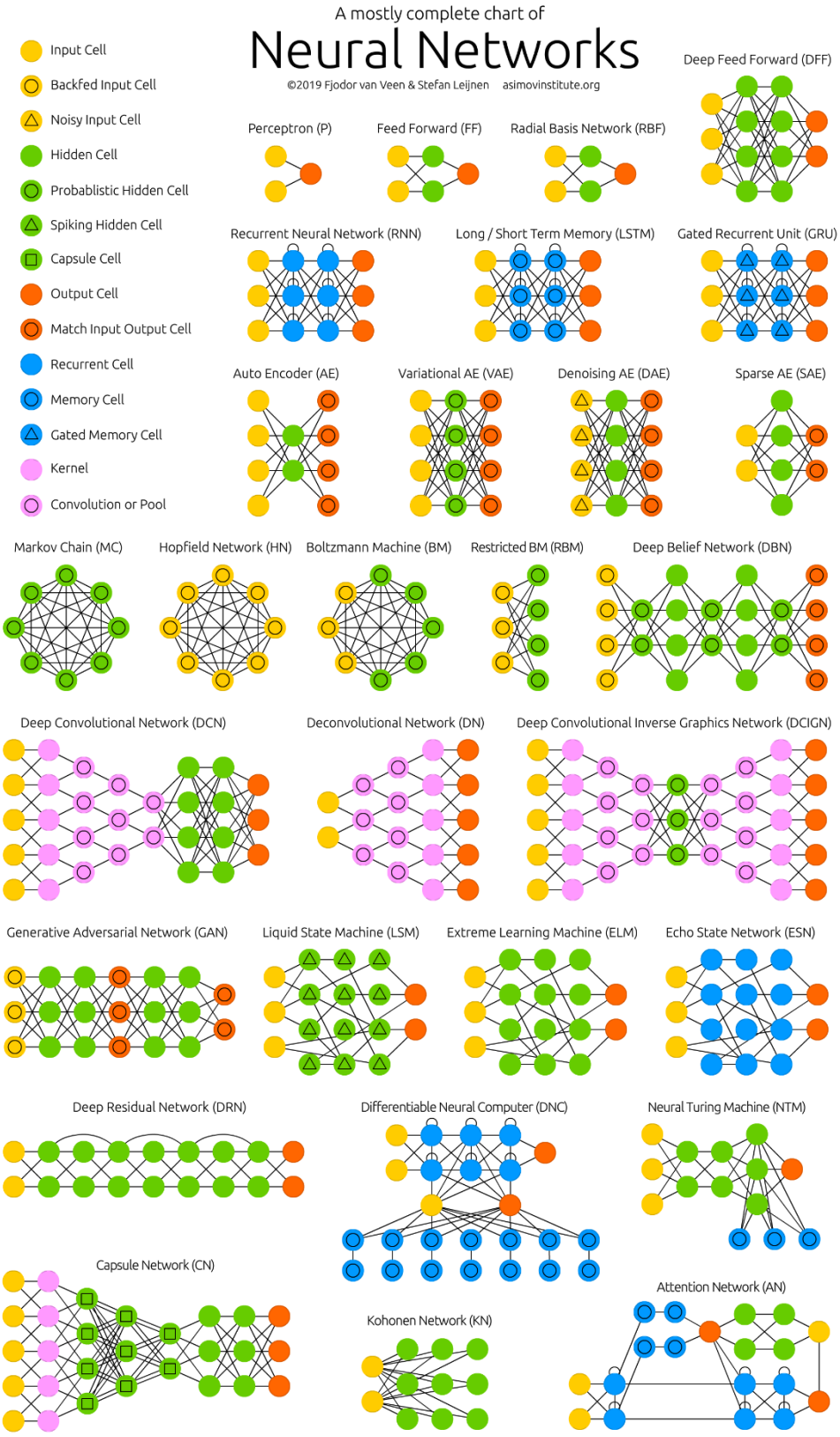
Normalmente o output de um neurônio é o input de outro neurônio [13], exceto na primeira e última camada. Os inputs passam pela primeira camada de neurônios que ativa alguns que passam para a próxima camada, que então ativa alguns e assim por diante. Essa rede complexa de ativação tem como resultado alguns neurônios na camada final de output ativarem. [13]

Com ajustes dos pesos a rede neural consegue realizar tarefas complexas como reconhecer faces e dígitos escritos à mão. [13]

Existem várias arquiteturas já criadas e estudadas, cada uma com seu objetivo e pontos fortes e fracos. Fjodor Van Veen criou um uma ótima ilustração das arquiteturas mais populares:

[4] "Neural networks consist of a number of nodes, each of which can be thought of as representing a neuron. Typically, these neurons are arranged into layers, and the neurons from one layer are connected to the neurons in the two layers on either side of it."

Figura 6 - Exemplos de arquiteturas de Redes Neurais



Fonte: Fjodor Van Veen; Stefan Leijnen (2019)

2.1.1.4. GENERATIVE ADVERSARIAL NETS (GAN)

É um tipo de aprendizado de máquina em que se treina dois algoritmos de perceptons de multidicamada ao mesmo tempo: um generativo (G) e um discriminativo (D). O algoritmo G tenta aumentar a chances de erro no algoritmo D, correspondendo a um minimax[5] de dois jogadores. G tenta imitar os dados recebidos e enganar D, enquanto D tenta descobrir se o dado apresentado vem de G ou se realmente vem dos dados reais. Essa competição melhora os métodos de geração e discriminação até que G consiga construir um dado igual ao real. [6]

O algoritmo G recebe um vetor com valores aleatórios e tenta transformar no dado real, enquanto D recebe o dado gerado por G e determina se é um dado real ou um dado gerado por G. [6]

2.1.1.5. PIX2PIX

É um algoritmo que utiliza GAN para criar uma imagem de outra imagem. O motivo de utilizar GAN é porque com ele não é preciso especificar qual a função para determinar a perda[6], já que ela se adapta com os inputs. O gerador recebe uma imagem e então codifica e decodifica a imagem “diminuindo” e “aumentando” novamente gerando uma nova imagem que em seguida passa pelo discriminador que também recebe a imagem original e determina se a imagem recebida é real ou se foi feita pelo gerador. [10]

2.1.1.6. CYCLEGAN

Se basea na Pix2Pix e no conceito de consistência cíclica, ou seja, se traduzirmos algo, a tradução de volta deve ser a mesma. Por exemplo, se traduzirmos a palavra "cachorro" para inglês obteremos "dog" e se traduzirmos de volta "dog" para português devemos obter

[5] O método Minimax é um método de decisão proposto por John Von Neumann em sua teoria de jogo que tem como objetivo minimizar a perda máxima. Também pode ser chamado de Maximin, com o objetivo de maximizar o ganho mínimo. Normalmente utilizado com dois jogadores onde o objetivo de cada um é aumentar a perda do outro.
[6] Perda é a diferença do dado real com a previsão da inteligência artificial, existem várias funções para determinar a perda, como Mean Squared Error (Erro médio quadrático) que calcula o a perda multiplicando os erros ao quadrado e tirando uma média deles.

"cachorro". Esse método consegue aprender a traduzir imagem para imagem sem que elas estejam pareadas (diferente o Pix2Pix). Ele utiliza dois algoritmos generativos (G e F) e dois discriminativos (X e Y). Se alimentarmos G com uma imagem I, ela retorna uma imagem J, e se alimentarmos F com a imagem J, ela deve retornar a imagem I. Então F(G(I)) deve ser I e G(F(J)) deve ser J. X tem como objetivo identificar de dado uma imagem, ela é real ou se ela veio de G, e Y identificar se dado uma imagem ela é real ou veio de F. [32]

2.1.1.7. ALGORITIMO GENÉTICO

Foi inventado por John Holland em 1975 e tenta imitar o processo evolutivo biológico da natureza. Ele consiste em 4 partes:

1. Criação de uma população
2. Avaliação de cada indivíduo.
3. Seleção dos melhores cromossomos.
4. Manipulação genética e criação de uma nova população. [13]

Cada indivíduo na população é representado por cromossomos, que são constituídos por genes. Esses genes variam dependendo da sua aplicação, mas podemos representar como uma string de bits como apresentado por John Holland. [2]

A avaliação dos indivíduos é responsável pela “sobrevivência dos mais fortes” [13], ela avalia a aptidão de cada indivíduo. Essa aptidão é calculada por uma função que deve retornar um valor maior para os melhores cromossomos. A seleção dos melhores é feita através de uma *mating pool*, que contém os melhores cromossomos. A população dela pode ser feita de várias maneiras, uma delas é popular proporcionalmente à aptidão. Por exemplo, se indivíduo A tiver a aptidão 2 e B tiver 6, A aparece 2 vezes e A 5, fazendo B ter uma chance 3 vezes maior de ser escolhido. [21]

A manipulação genética pode ser feita através de 2 métodos:

- Crossover: escolhe-se 2 cromossomos e um ponto de crossover, depois quebra-se os cromossomos nesse ponto e então recombina-se o cromossomo da frente com

o do trás, produzindo 2 novos cromossomos. Por exemplo, podemos ter um cromossomo 0000 e um 1111, ao fazer o crossover no index 1, teremos 0011 e 1100. [2]

- Mutação: Normalmente aplica-se uma probabilidade muito pequena de ocorrer. Nela muda-se o valor de um gene. Por exempli se tivermos um cromossomo 00000 e aplicarmos mutação nele (nesse exemplo a probabilidade é grande apenas para demonstração) podemos ter como resultado o cromossomo 00101.

E por fim, os cromossomos resultados dessas manipulações serão a próxima população a ser avaliada, formando um ciclo. [13]

2.2. SISTEMA DE RECOMENDAÇÃO

Como mostrado no nome, é um sistema feito para recomendar produtos aos clientes. É muito comum hoje e diversas empresas já estão utilizando. Empresas como Amazon e Netflix utilizam para recomendar novos produtos com base na sua atividade anterior. [24]

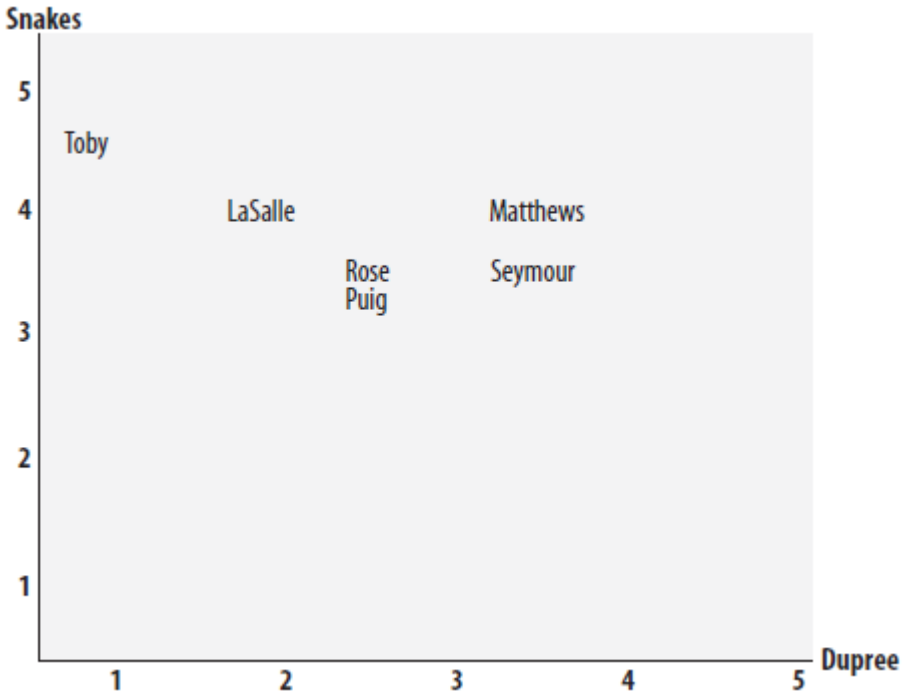
Esse sistema funciona calculando uma pontuação de similaridade de uma pessoa com todas as outras e então recomendar um item. Existem vários sistemas para calcular a pontuação de similaridade, alguns deles são a Distância Euclidiana e a correlação de Pearson. [24]

Após sabermos o quanto alguém é parecido com o outro, pegamos os itens já avaliados por essa outra pessoa (B) e não avaliados pela primeira pessoa (A) e multiplicamos a avaliação com os pontos de similaridade. E então a soma desse produto dividido pela soma dos pontos de similaridade de cada pessoa que avaliaram mostra a possível avaliação do item para a pessoa A. Após todo esse processo obtemos um rank dos itens as suas possíveis avaliações. [24]

2.2.1. DISTÂNCIA EUCLIDIANA

É um jeito bem simples de calcular a pontuação de similaridade. Nela escolhe-se dois itens que as pessoas tenham avaliado e os usa como eixos de um gráfico: [24]

Figura 7 - Gráfico Distância Euclidiana



Fonte: Toby Segaran (2007)

Visualmente é fácil de identificar quem é similar com quem, quanto mais perto duas pessoas estão, mais similares são. Como este gráfico é 2D vemos somente dois itens por vez, mas o princípio é o mesmo para dados maior aumento uma dimensão por item. [24]

Segaran (2007) explica que para obter a distância de duas pessoas no gráfico, calcula-se a diferença de cada eixo e eleva ao quadrado, e tira a raiz quadrada do resultado, ou seja, $\sqrt{\sum_{i=1}^{n}(x_i - x_i)^2}$. Essa formula devolve um resultado menor para quanto mais próximo uma pessoa está da outra, mas precisamos de um ponto maior para pessoas similares, por isso invertemos a função e adicionamos 1 para não dividir por 0: $\frac{1}{1+\sqrt{\sum_{i=1}^{n}(x_i - x_i)^2}}$.

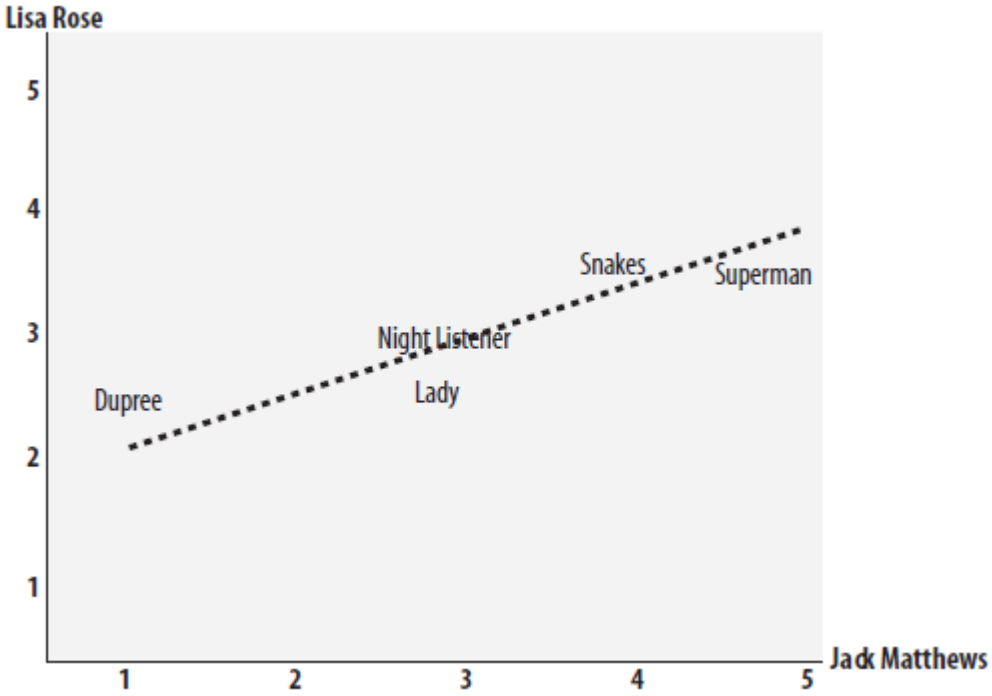
Essa função retorna um número de 0 a 1 sendo 1 gostos totalmente iguais. [24]

2.2.2. CORRELAÇÃO DE PEARSON

Um outro jeito de calcular a pontuação de similaridade um pouco mais sofisticado. Enquanto a Distância Euclidiana calcula a distância de duas pessoas tendo como eixo os itens, a correlação de Pearson calcula o quão bem os dados cabem em uma linha reta. [24]

Nessa fórmula, os eixos serão as pessoas e no gráfico ficará os itens:

Figura 8 - Gráfico Correlação de Pearson



Fonte: Toby Segaran (2007)

A linha vista no gráfico acima se chama linha de melhor ajuste porque ela chega mais perto de todos os itens da tabela, se as duas pessoas dessem os mesmos pontos por item, todos os itens tocariam a linha. [24]

Um grande aspecto dessa fórmula é que ela compensa a inflação dos dados. Como mostrado na Figura 8, Jack tende a dar notas mais altas que Lisa e mesmo assim podemos ver uma correlação entre eles. Se analisarmos esses dois casos com a Distância Euclidiana, iriamos obter uma pontuação de similaridade menor, pois um é mais severo que o outro. [24]

O cálculo dessa função é feito com a soma das avaliações de cada pessoa, a soma das avaliações de cada pessoa ao quadrado e a soma do produto das avaliações das pessoas: $\frac{\sum_{i=1}^{n}(x_i * y_i) - \frac{\sum_{i=1}^{n} x_i * \sum_{i=1}^{n} y_i}{n}}{\sqrt{\frac{(\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2)}{n} * \frac{(\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2)}{n}}}$. Essa função retorno um valor de -1 a 1 onde 1 é quando as pessoas têm as avaliações iguais. [24]

2.3. GOOGLE COLABORATORY

O Google Colaboratory (Colab) é uma plataforma em nuvem grátis criada pela Google para encorajar a pesquisa de Inteligência Artificial e Aprendizado de Máquina. Ele foi baseado no projeto Jupyter e como ele roda inteiramente em nuvem, não é preciso nenhuma instalação a não ser por um navegador de internet no computador. [7] [15]

A maior barreira para o estudo de IA e ML é o poder de processamento, que é necessário um poder enorme para a máquina aprender e ter sucesso. Por isso o Colab permite a utilização de GPU gratuitamente na plataforma, além disso, ele também disponibiliza a criação de *notebooks* em python 2 e 3, bibliotecas já instaladas, compartilhamento do código, integração como Google Drive, execução de comandos bash e comandos *magic*[7] do IPython [18]. [7] [15]

2.4. TENSORFLOW (TF)

TF é uma biblioteca em Python de código aberto criada em 2015 e recebeu sua primeira versão em 2017. Atualmente, ele é amplamente utilizado em pesquisas de ML e IA. Seu predecessor foi o DistBelief, outra biblioteca criada pela equipe do Google Brain, porém com o passar do tempo o DistBelief começou a apresentar erros limitando sua usabilidade e flexibilidade. TF foi criado com as lições aprendidas dos erros do DistBelief e foi construído para ser flexível, eficiente, extensível e portável. Com ele é possível rodar projetos de IA em qualquer computador e em até celulares. [4] [28]

TF foi construído em C++ e pode ser desenvolvido em Python, Java Script, Swift, Android, IOS e Raspberry Pi. Ele possui sua própria API que é bastante poderosa, mas complexa e com uma dificuldade relativamente alta, por isso, ele também disponibiliza a possibilidade de criação de modelos de ML em Keras, que é uma API para redes neurais com linguagem de alto nível. Keras é mais simples e rápido de aprender, muito utilizado por iniciantes. [4] [8] [12] [28]

[7] Sistema de comandos do interpretador IPython, que fornece efetivamente uma linguagem de mini comando ortogonal à sintaxe do Python e extensível ao usuário com novos comandos. [JUPYTER]

3. APLICAÇÃO

3.1. CRIAÇÃO

O projeto foi feito em linguagem Python com as principais bibliotecas sendo: TensorFlow, para a criação da Inteligência Artificial, e Tkinter (nativa do Python) para a criação da interface. Outras bibliotecas também foram utilizadas, mas somente como suporte para funções auxiliares, como OpenCV para a manipulação de arquivos, como importar o dataset para o programa.

Para a facilitação da criação e treinamento da IA foi utilizado o Google Colaboratory, uma plataforma criada com o objetivo de facilitar e incentivar o estudo em IA, pela sua facilidade de uso, disponibilização de códigos e a permissão de rodar em GPU em nuvem.

O objetivo do programa é ajudar os criadores de conteúdo a criar conteúdo personalizados e customizados para cada cliente, aproximando a relação cliente e empresa.

3.2. INTERFACE

Para a criação da interface foi utilizado uma biblioteca do Python chamada Tkinter, essa biblioteca permite criar janelas facilmente em Python. Para criarmos a primeira janela, adicionaremos um menu, alguns botões, um slide e um canvas para mostrar o desenho e desenhar.

Para começar um projeto com o Tkinter, primeiro precisamos criar o objeto e então atribuir os *widgets*[8]. Para criar o objeto basta chamar Tk():

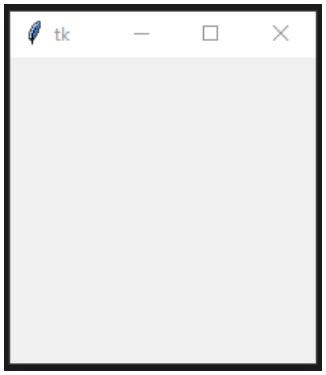
```
import tkinter as tk

if __name__ == "__main__":
        root = tk.Tk()
        root.mainloop()
```

Ao executar esse simples código obtemos uma simples janela cinza:

[8] Componentes na tela, neste caso vamos utilizar botões, canvas, sliders, labels, etc.

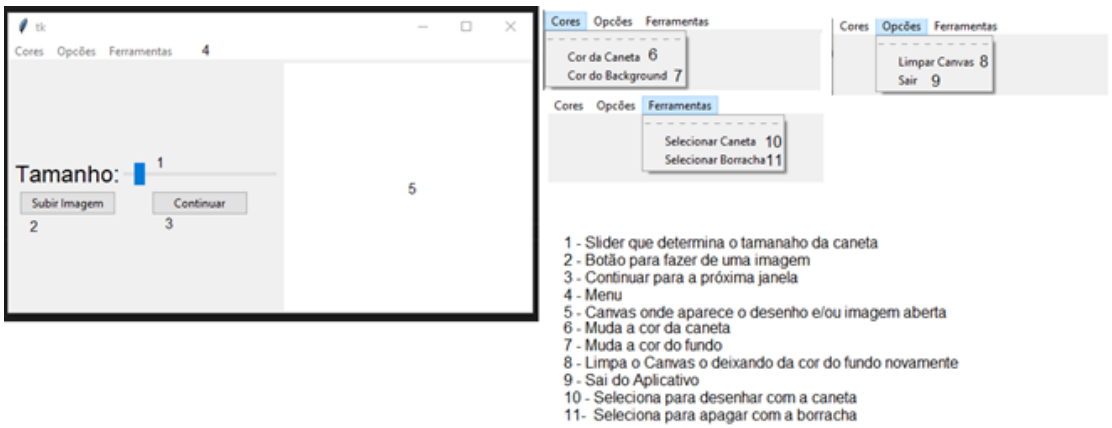
Figura 9 - Janela simples Tkinter



Fonte: Própria (2019)

A seguir, criamos uma classe para atribuir os widgets e então adicionamos eles na janela. O resultado da primeira interface será:

Figura 10 - Segunda Interface do Software



Fonte: Própria (2019)

Após iniciar o software, na primeira página, temos a opção de ajustar o tamanho do pincel que podemos utilizar no canvas para desenhar com o slide, uma opção de fazer upload em uma imagem que está no seu computador, e no menu, localizado na parte superior do software, temos a opção de mudar a cor do pincel, plano de fundo e mudar de pincel para borracha e vice-versa.

Um ponto importante é quando se fa um upload de uma imagem, ela deve estar 256x256 porque a IA foi criada e treinada com imagens nessas dimensões (256x256x3 se contarmos as cores). Por isso, ao fazer o upload, o software ativa uma função automaticamente para redimensionar a imagem. A função recebe como parâmetro a imagem, o tamanho da desejado da imagem e como opcional, a cor que será preenchida as bordas da imagem se ela não for quadrada:

```
def make_square(im, sqr_size, fill_color=[0, 0, 0]):
    y, x = im.shape[0:2]
    if y > sqr_size or x > sqr_size:
        if y > x:
            div = sqr_size / y
            y *= div
            x *= div
        else:
            div = sqr_size / x
            y *= div
            x *= div
    size = max(sqr_size, x, y)
    add_x = int((size - x)/2)
    add_y = int((size - y)/2)
    im = cv2.resize(im, (int(x), int(y)))
    new_im = cv2.copyMakeBorder(
        im, add_y, add_y, add_x, add_x, cv2.BORDER_CONSTANT, value=fill_color)
    return new_im
```

Após termos nossa imagem no canvas, clicamos no botão iniciar que nos leva a uma segunda página que ativa nossa IA e transforma o desenho em uma foto. Ela irá criar 10 fotos ligeiramente diferentes uma da outra. Quando colocamos o mouse em cima de uma imagem, ela recebe pontos que irão ser utilizadas do algoritmo genético como a aptidão de cada indivíduo.

Na segunda janela temos 10 imagens geradas pela IA em 5 colunas e 2 linhas, números indicando os pontos de cada imagem logo abaixo de cada uma e um botão para gerar a nova geração:

Figura 11 - Segunda Interface do Software

Fonte: Própria (2019)

3.3. IA E TREINAMENTO

A criação e os primeiros testes da IA foi feita em uma plataforma criada pelo Google chamada Google Colabarotary. Essa plataforma foi criada para incentivar as pessoas a estudarem Inteligencia Artificial e Aprendizado de máquina. Ela disponibiliza vários códigos de arquitetura de redes neurais gratuitamente. Além disso, ela também permite rodar os códigos em GPU, acelerando significantemente a execução.

Foi nessa plataforma que o nosso algoritmo de IA, o CycleGAN, foi criado e testado. Foi utilizado como base um algoritmo de exemplo já criado e disponibilizado por eles. O treino do algoritmo foi muito acelerado devido a possbilidade de treinar utilizando a GPU.

Esse algoritmo, o CycleGAN, é o responsável por transformar uma imagem em uma foto e vice-versa. Ele consiste em dois modelos generativos, um responsável em gerar um foto através de um desenho (GF) e outro em gerar um desenho através de uma foto (GD), e dois discriminativos, um responsável por determinar se a foto recebida vem do GF ou da base (DF) e outro em determinar se o desenho recebido vem de GD ou da base (DD).

Escolhemos esse algoritmo pois ele permite um treinamento mesmo com imagens não pareadas, ou seja, não é necessário que haja pareamento entre as imagens, uma imagem tem uma resposta definitiva, mas sim apenas imagens de fotos e desenhos já são o suficiente. Facilitando, assim, a obtenção das datasets necessárias para o treinamento.

3.4. ALGORITMO GENÉTICO

Para a criação do algoritmo genético, precisávamos de 3 coisas: uma população, algum jeito de determinar a aptidão e como fazer o crossover e a mutação. Como o CycleaGAN utiliza redes neurais, elas têm pesos entre as ligações dos neurônios, e quanto maior for esse peso, mais importante é essa ligação. Tendo isso em mente, escolhemos os pesos como o DNA de cada indivíduo e ao gerar cada um, alteramos levemente o peso maior, alterando levemente a imagem também.

Para determinar a aptidão, criamos um sistema de pontos para cada indivíduo em que um ganha quando o mouse do usuário estiver em cima. A função de aptidão é escolher quem tem mais pontos e então fazer o crossover e mutação para criar a nova população.

Antes de fazer o crossover, nós populamos uma mating pool com os indivíduos proporcionalmente pelos seus pontos, assim quem tem mais pontos aparece mais, tendo assim mais chance de ser escolhido. O crossover é feito entre o indivíduo com mais pontuação e um outro escolhido aleatoriamente na mating pool. Após escolher o segundo indivíduo, escolhemos também aleatoriamente o ponto do crossover e qual é a frente e qual é atrás. No final obteremos um ponto de crossover entre 1 e 44, e 50% de chance de ter o DNA do melhor indivíduo primeiro seguido do segundo ou vice e versa.

Para a mutação, pegamos o resultado do crossover e aplicamos uma chance de 10% de ocorrer mutação para cada gene do DNA (sequência de pesos). Se ocorrer mutação, escolhemos o maior peso e multiplicamos ele um número aleatório de 0.01 a 1.01.

Depois de aplicar o crossover e a mutação, atualizamos os pesos da IA e geramos um novo indivíduo. Isso se repete mais 9 vezes totalizando 10 indivíduos gerados.

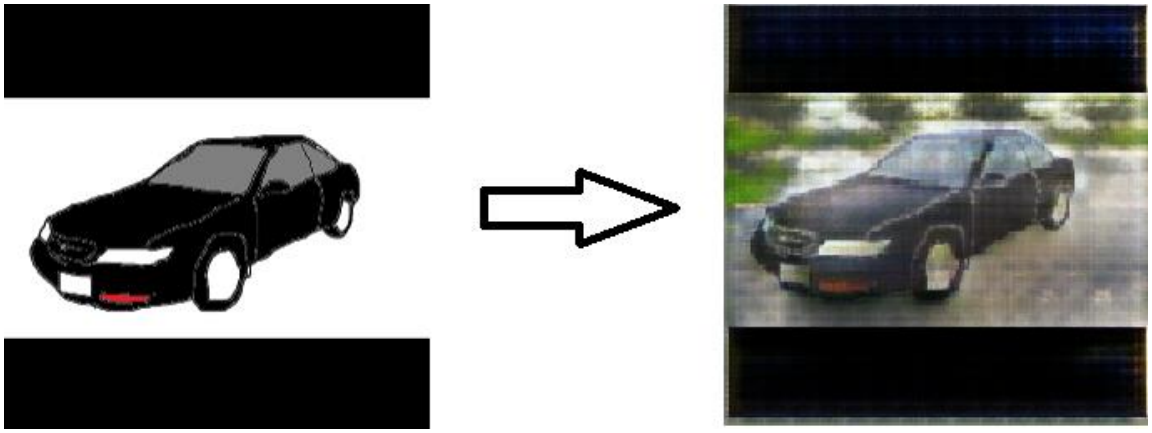
4. EXPERIMENTOS

4.1. CYCLEGAN

Obtivemos o resultado esperado com os algoritmos generativos:

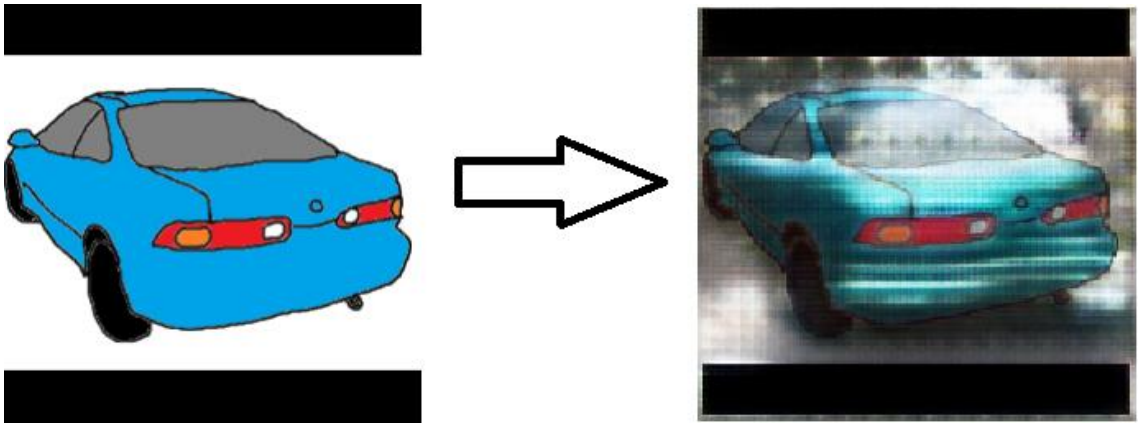
4.1.1. GERANDO DESENHO EM FOTO:

Figura 12 - Exemplo 1 de desenho para foto



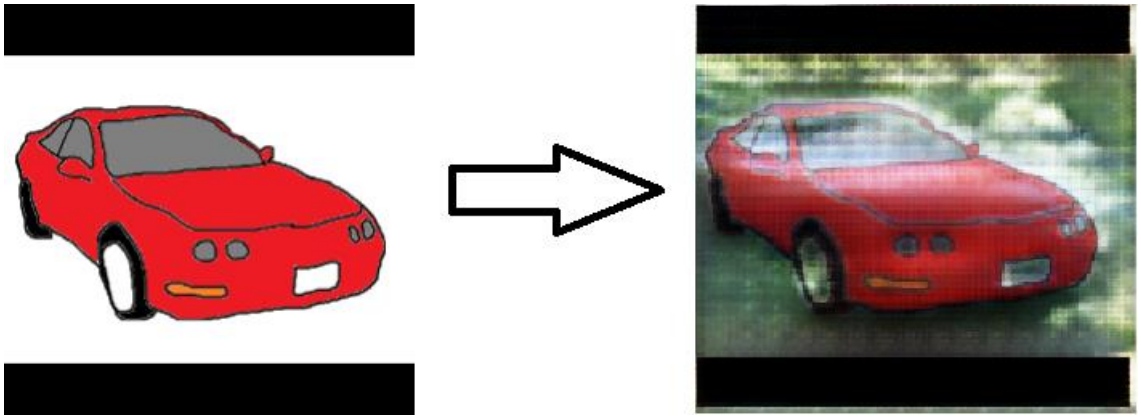
Fonte: Própria (2019)

Figura 13 - Exemplo 2 de desenho para foto



Fonte: Própria (2019)

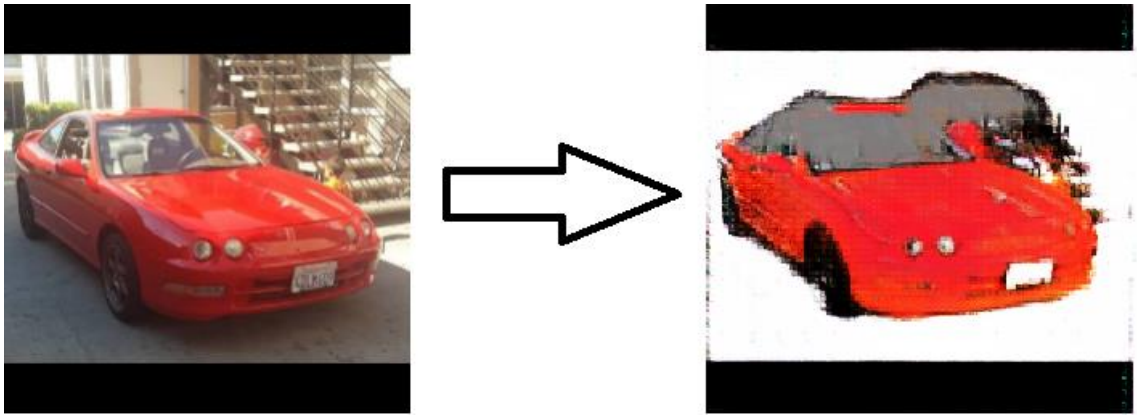
Figura 14 - Exemplo 3 de desenho para foto



Fonte: Própria (2019)

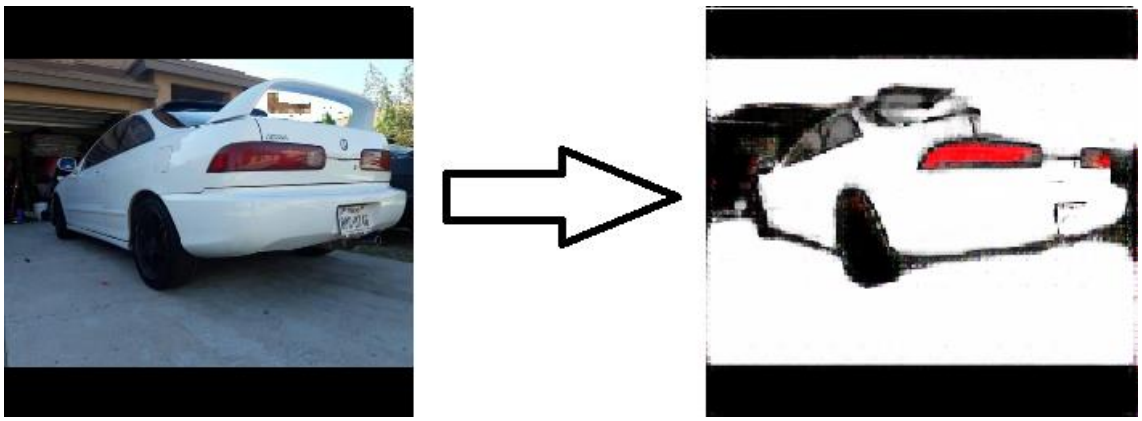
4.1.2. GERANDO FOTO EM DESNHO:

Figura 15 - Exemplo 1 de foto para desenho



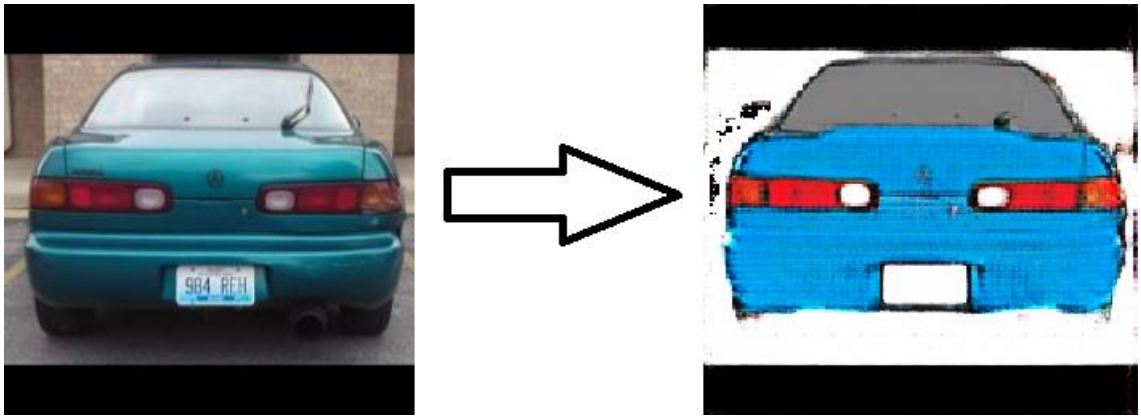
Fonte: Própria (2019)

Figura 16 - Exemplo 2 de foto para desenho



Fonte: Própria (2019)

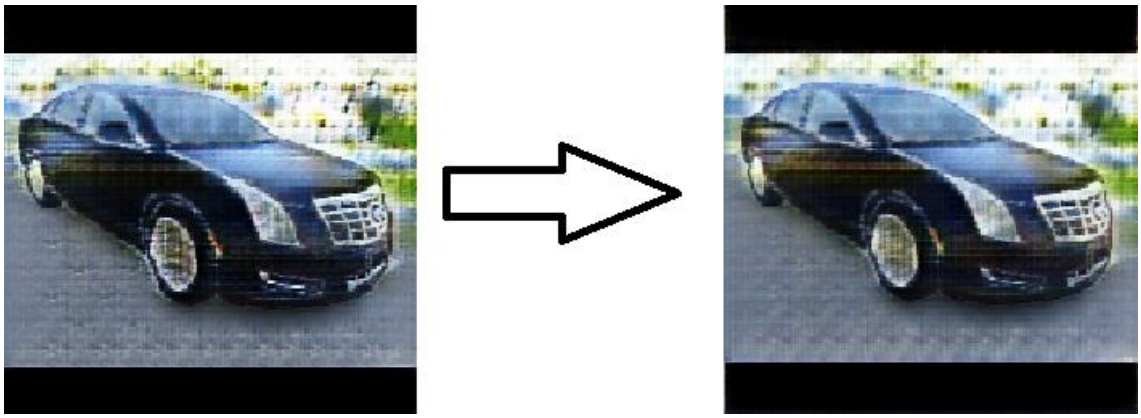
Figura 17 - Exemplo 3 de foto para desenho



Fonte: Própria (2019)

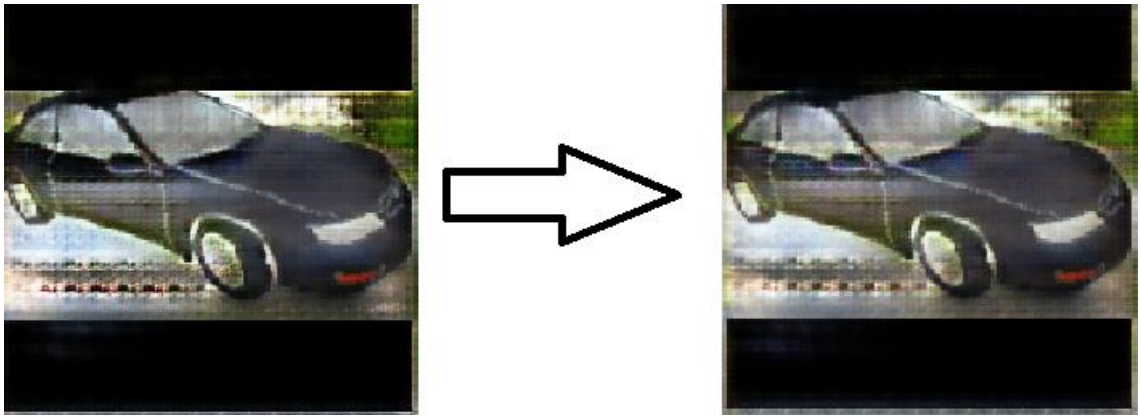
4.1.3. DIFERENTES GERAÇÕES

Figura 18 - Exemplo 1 Geração



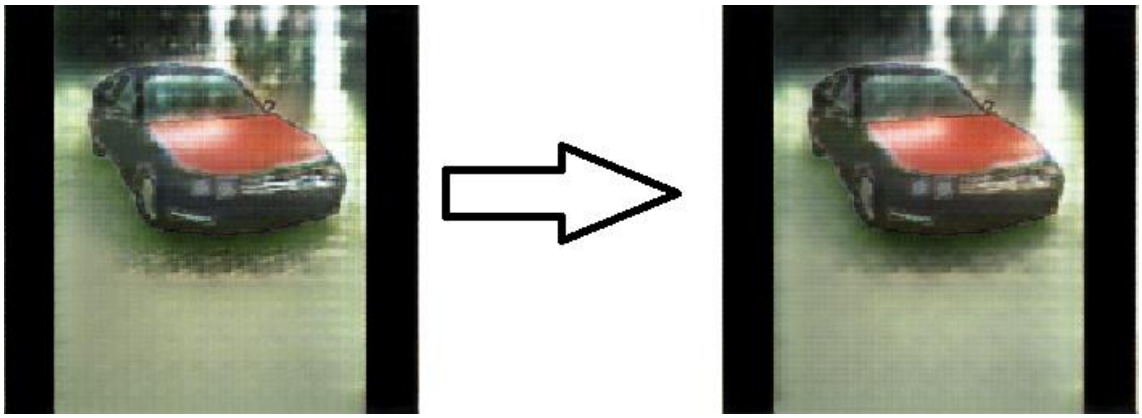
Fonte: Própria (2019)

Figura 19 - Exemplo 2 Geração



Fonte: Própria (2019)

Figura 20 - Exemplo 3 Geração



Fonte: Própria (2019)

Podemos observar que a segunda foto é um pouco mais “suave”, com as linhas que contornam os objetos mais fosca.

5. CONCLUSÃO

Este trabalho teve como objetivo criar um software com Inteligência Artificial para criar produtos personalizados para cada cliente. Ele visa ajudar o mercado de marketing a personalizar os produtos visto o constante crescimento das exigências dos consumidores em algo mais personalizado, sendo vistos como pessoas e não só como número.

Ao longo do desenvolvimento do trabalho foi feito estudos e pesquisas para entender como criar o software e foram feitos muitos testes até que o resultado tenha saído como esperado. As imagens obtidas através do software foram satisfatórias exigindo apenas mais poder de processamento e tempo para o treinamento da Inteligência Artificial.

Com o algoritmo genético foi possível criar imagens ligeiramente diferente uma das outras e ainda salvar os pesos que produziam tais imagens. O software conta com um espaço para desenhar nele, mas ainda está em estágio inicial sem muitas funcionalidades, tendo, assim, no momento inicial, o foco em fazer uploads e imagens já feitos em softwares mais avançados e profissionais. Não há limite no algoritmo genético e gerações, o usuário pode criar gerações quantas vezes for necessário.

5.1. TRABALHOS FUTUROS

Mesmo que o software tenha alcançado as expectativas e mostrado seu poder e maneira satisfatória, ainda é preciso alguns ajustes:

- A Inteligência Artificial foi treinada com uma base de dados muito limitada devido a falta de exemplos disponíveis em meio eletrônico. É necessário descobrir um meio de obtenção de grande volume de dados mesmo que não estejam tratados. Os dados de desenho utilizados nesse trabalho foram de autoria própria e devido ao limite de tempo não houve possibilidade de criação de muitos desenhos.
- O software precisa de mais tempo de treinamento, pois devido ao tempo limite não foi possível um treinamento muito profundo.
- Ajuste nas interfaces tanto a primeira quanto a segunda. Em ambas as janelas, os widgets utilizados foram o padrão, sem nenhuma personalização. Na primeira janela o canvas que permite desenhar diretamente nele está muito simples, é necessário dar mais

atenção a ele e melhorá-lo. Na segunda janela é necessário ajustar o layout em que as imagens estão sendo mostradas para ficar mais agradável e amigável.

- Há o limite da imagem recebida e gerada se de apenas 256x256. Pode-se ainda aumentar esse limite, mas o treinamento e geração da imagem iria demorar mais.

6. REFERÊNCIAS

[1] ANKIT, Yaduvanshi. **What does "loss" mean in deep neural networks?** 2018. Disponível em: <https://www.quora.com/What-does-loss-mean-in-deep-neural-networks>. Acesso em: 23 nov. 2019.

[2] COPPIN, Ben. **Artificial Intelligence Illuminated.** Sudbury: Jones And Bartlett Publishers, 2004. 739 p.

[3] DATA SCIENCE ACADEMY. **Deep Learning Book**, 2019. Disponível em: <http://deeplearningbook.com.br/>. Acesso em: 19 nov. 2019.

[4] DATA SCIENCE ACADEMY. **O QUE É O TENSORFLOW MACHINE INTELLIGENCE PLATFORM?** 2018. Disponível em: <http://datascienceacademy.com.br/blog/o-que-e-o-tensorflow-machine-intelligence-platform/>. Acesso em: 25 nov. 2019.

[5] EQUIPE REAMP. **12 estatísticas que provam porque a personalização é tão importante.** 2018. Disponível em: <https://www.reamp.com.br/blog/2018/07/12-estatisticas-que-provam-porque-a-personalizacao-e-tao-importante/>. Acesso em: 25 nov. 2019.

[6] GOODFELLOW, Ian J. et al. **Generative Adversarial Nets.** Montreal: Departement D'informatique Et de Recherche Op ´ Erationnelle, 2014. 9 p. Disponível em: <https://arxiv.org/pdf/1406.2661.pdf>. Acesso em: 1 out. 2019.

[7] GOOGLE. **Google Colaboratory.** Disponível em: <https://colab.research.google.com/notebooks/welcome.ipynb#>. Acesso em: 11 out. 2019.

[8] GUALBERTO, Arnaldo. **TensorFlow 2.0: melhores práticas e o que mudou:** Confira as principais novidades na famosa biblioteca de Machine Learning da Google. 2019. Disponível em: <https://medium.com/data-hackers/tensorflow-2-0-melhores-pr%C3%A1ticas-e-o-que-mudou-ec56ba95b6a>. Acesso em: 25 nov. 2019.

[9] INFOSYS. **Rethinking Retail:** Insights from consumers and retailers into an omni-channel shopping experience. 2013. Disponível em: <https://www.infosys.com/newsroom/press-releases/Documents/genome-research-report.pdf>. Acesso em: 25 nov. 2019.

[10] ISOLA, Phillip et al. **Image-to-Image Translation with Conditional Adversarial Networks.** Berkeley: Berkeley Ai Research (bair) Laboratory, 2018. 17 p. Disponível em: <https://arxiv.org/pdf/1611.07004v3.pdf>. Acesso em: 1 out. 2019.

[11] KAPUR, Rohan. **What is bias in artificial neural network?** 2016. Disponível em: <https://www.quora.com/What-is-bias-in-artificial-neural-network>. Acesso em: 19 nov. 2019.

[12] KERAS. **Keras: The Python Deep Learning library.** Disponível em: <https://keras.io/>. Acesso em: 25 nov. 2019.

[13] KONAR, Amit. **Artificial Intelligence and Soft Computing:** Behavioral and Cognitive Modeling of the Human Brain. Boca Raton: Crc Press Llc, 2000. 788 p.

[14] KRIVOKUćA, Mina. **Minimax with Alpha-Beta Pruning in Python.** 2019. Disponível em: <https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/>. Acesso em: 17 nov. 2019.

[15] LALL, Vishakha. **Google Colab — The Beginner's Guide.** 2018. Disponível em: <https://medium.com/lean-in-women-in-tech-india/google-colab-the-beginners-guide-5ad3b417dfa>. Acesso em: 11 out. 2019.

[16] LOTUFO, Larissa. **Você é único, por que seus produtos não seriam?** 2018. Disponível em: <https://www.ecommercebrasil.com.br/artigos/personalizacao-de-produtos-e-commerce/>. Acesso em: 25 nov. 2019.

[17] NIELSEN, Michael. **Neural Networks and Deep Learning**, 2019. Disponível em: <http://neuralnetworksanddeeplearning.com/index.html>. Acesso em: 19 nov. 2019.

[18] PROJECT JUPYTER. **The cell magics in IPython.** Disponível em: <https://nbviewer.jupyter.org/github/ipython/ipython/blob/1.x/examples/notebooks/Cell%20Magics.ipynb>. Acesso em: 11 out. 2019.

[19] RAKESH, Hemant. **Minimax or Maximin?** 2019. Disponível em: <https://becominghuman.ai/minimax-or-maximin-8772fbd6d0c2>. Acesso em: 17 nov. 2019.

[20] REAL-TIME Marketing Insights Study. Disponível em: <https://offers.adobe.com/en/na/marketing/landings/_46316_real_time_marketing_insights_study.html>. Acesso em: 2 nov. 2019.

[21] RUSSELL, Stuart J. et al. **Artificial Intelligence:** A Modern Approach. 3. ed. Upper Saddle River: Pearson Education, 2010. 1132 p.

[22] SALESFORCE. **Customer Expectations Hit All-Time Highs.** 2016. Disponível em: <https://www.salesforce.com/research/customer-expectations/>. Acesso em: 25 nov. 2019.

[23] SALESFORCE. **Customers Are Willing to Swap More Data for Personalized Marketing.** 2016. Disponível em: <https://www.salesforce.com/blog/2016/11/swap-data-for-personalized-marketing.html>. Acesso em: 25 nov. 2019.

[24] SEGARAN, Toby. **Programming Collective Intelligence:** Building Smart Web 2.0 Applications. Sebastopol: O'reilly Media, Inc., 2007. 334 p.

[25] SHARMA, Avinash. **Understanding Activation Functions in Neural Networks.** 2017. Disponível em: <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>. Acesso em: 19 nov. 2019.

[26] SHIFFMAN, Daniel. **The Nature of Code.** 5. ed. Mountain View: Creative Commons, 2012. 498 p. Disponível em: <http://wtf.tw/ref/shiffman.pdf>. Acesso em: 19 nov. 2019.

[27] SMALE, Thomas. **Why a 'Personal' Customer Experience Is Critical to Your Business' Success.** 2018. Disponível em: <https://www.entrepreneur.com/article/315356>. Acesso em: 2 nov. 2019.

[28] TENSORFLOW. **TensorFlow.** Disponível em: <https://www.tensorflow.org/>. Acesso em: 25 nov. 2019.

[29] VAN VEEN, Fjodor. **THE NEURAL NETWORK ZOO.** 2016. Disponível em: <https://www.asimovinstitute.org/neural-network-zoo/>. Acesso em: 21 nov. 2019.

[30] VERMA, Shiva. **Understanding different Loss Functions for Neural Networks.** 2019. Disponível em: <https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718>. Acesso em: 23 nov. 2019.

[31] WALIA, Anish Singh. **Activation functions and it's types-Which is better?** 2017. Disponível em: <https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f>. Acesso em: 19 nov. 2019.

[32] ZHU, Jun-yan et al. **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.** 6. ed. Berkeley: Berkeley Ai Research (bair) Laboratory, 2018. 18 p. Disponível em: <https://arxiv.org/pdf/1703.10593v6.pdf>. Acesso em: 1 out. 2019.

7. APÊNDICES

7.1. APÊNDICE A – CÓDIGOS UTILIZADOS

O código fonte utilizado nesse documento pode ser estudado no repositório do Github:
<https://github.com/RaphaelYamanaka/TCC_Final.git>