UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE ENGENHARIA QUÍMICA
CURSO DE ENGENHARIA QUÍMICA

PEDRO PINGUELLI BORGES

APLICAÇÃO DE REDES NEURAIS PARA A DETERMINAÇÃO DO COEFICIENTE DE ATRITO DE FILMES PLÁSTICOS FLEXÍVEIS

TRABALHO DE CONCLUSÃO DE CURSO

Londrina
2020

PEDRO PINGUELLI BORGES

APLICAÇÃO DE REDES NEURAIS PARA A DETERMINAÇÃO DO COEFICIENTE DE ATRITO DE FILMES PLÁSTICOS FLEXÍVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Química da Universidade Tecnológica Federal do Paraná, *Campus* Londrina, como requisito parcial à obtenção do título de Engenheiro Químico.

Orientador: Prof. Dr. Lucas Bonfim Rocha

Coorientadora: Profª. Drª. Pricila Marin

Londrina

2020

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO - TCC
APLICAÇÃO DE REDES NEURAIS PARA A DETERMINAÇÃO DO COEFICIENTE DE ATRITO DE FILMES PLÁSTICOS FLEXÍVEIS

Por

Pedro Pinguelli Borges

Monografia defendida em sessão pública às 19 horas 00 min. do dia 02 de dezembro de 2020 como requisito parcial, para conclusão do Curso de Engenharia Química da Universidade Tecnológica Federal do Paraná, *Campus* Londrina. O candidato foi arguido pela Banca Examinadora composta pelos Membros abaixo assinados. Após deliberação e conferidas, bem como achadas conforme, as alterações indicadas pela Banca Examinadora, o Trabalho de Conclusão de Curso, em sua forma final, pela Coordenação do Curso de Engenharia Química foi considerado aprovado.

Banca examinadora:

Prof. Dr. Lucas Bonfim Rocha - UTFPR *Campus* Londrina – Orientador

Profª. Drª. Pricila Marín - UTFPR *Campus* Londrina – Coorientadora

Fernanda Pagani - Amcor – Supervisora de Estágio e Membro

Prof. Dr. Felipi Luiz Assunção Bezerra - UTFPR *Campus* Londrina – Membro

A “Ata de Defesa” e o decorrente “Termo de Aprovação” encontram-se assinados e devidamente depositados no SEI-UTFPR da Coordenação do Curso de Engenharia Química da UTFPR *Campus* Londrina, após a entrega da versão corrigida do trabalho, conforme Norma aprovada pelo Colegiado de Curso.

AGRADECIMENTOS

Primeiramente agradeço a Deus, a minha família, pelo suporte e apoio, aos meus professores, em especial Lucas Bonfim Rocha e Pricila Marin, que me auxiliaram no desenvolvimento deste trabalho, aos meus colegas de classe e amigos.

Agradeço também, a toda a equipe da AMCOR por ter me dado a oportunidade e auxílio em todos os momentos em que precisei.

RESUMO

BORGES, Pedro Pinguelli. Aplicação de Redes Neurais para a Determinação do Coeficiente de Atrito de Filmes Plásticos Flexíveis. 56 f. TCC (Curso de Engenharia Química), Universidade Tecnológica Federal do Paraná (UTFPR). Londrina, 2020.

As embalagens plásticas flexíveis têm seu uso sendo aumentado gradativamente, visto que são as embalagens mais versáteis, com variadas aplicações, e de grande durabilidade. Para a obtenção dessas embalagens, faz-se necessário toda uma estratégia de produção que passa não só pelo seu processo produtivo, como pelo controle de qualidade do produto. Visando a predição de parâmetros, o presente trabalho tem como objetivo a determinação do coeficiente de atrito (COF) em filmes plásticos flexíveis por meio da utilização de redes neurais do tipo *Perceptron*. De acordo com os dados obtidos nos testes experimentais, as redes neurais tanto em *Python* quanto em *Matlab* obtiveram resultados satisfatórios para a determinação de COF dos filmes, chegando a um R^2 = 95% e um erro médio de 0,01 sendo assim, criou-se um aplicativo para ambas as redes neurais para que os engenheiros de materiais pudessem interagir como usuários com estas redes. As redes neurais surgem como um possível auxiliar para os testes de controle de qualidade, visando a melhoria na velocidade de obtenção e análise de informações, mantendo a precisão das respostas.

Palavras-chave: Redes Neurais. *Perceptron*. COF. Plástico. Flexíveis.

ABSTRACT

BORGES, Pedro Pinguelli. Application of Neural Network for Determination of Friction Coefficient in Flexible Plastic Films 56 f. TCC (Course of Chemical Engineering) - Federal University of Technology – Paraná (UTFPR). Londrina, 2020.

Flexible plastic packaging is gradually being used, since it is the most versatile packaging, with various applications, and of great durability. In order to obtain these packages, a whole production strategy is required, which involves not only its production process but also the product quality control. Aiming the parameters prediction, the present work aims to determine the friction coefficient (COF) in flexible plastic films by using Perceptron neural networks. According to data obtained in experimental tests, the neural networks in both software (i.e., *Python* and *Matlab*) obtained satisfactory results for determining the COF of the films with an R^2 = 95% and a mean error of about 0.001, therefore, an application was created for both neural networks so that material engineers could interact as users with these networks. Neural networks appear as a possible aid for quality control tests, aiming at improving the speed of achieving and analyzing, keeping accuracy of responses.

Keywords: Neural Network. Perceptron. COF. Plastic. Flexibles

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na indústria, a embalagem é um recipiente ou uma envoltura em que se armazena determinado produto visando sua proteção ou seu agrupamento de unidades. As primeiras embalagens surgiram a 10.000 anos somente com funções alimentícias, como utensílios para comer, beber e armazenar alimentos, sendo eles recipientes naturais como, casca de coco ou conchas. Com a evolução da humanidade, desenvolveram-se também embalagens ainda simples, como tigelas de madeira e bolsas de pele de animais (SOUSA, 2012).

A descoberta do vidro trouxe uma grande mudança no ramo, pois pela primeira vez na história, houve a produção em larga escala, e pelo fato de vidro ser um material maleável, a aparição de diferentes moldes de embalagens que são utilizadas até hoje, podendo sem encontradas nos ramos de perfumarias, alimentos entre outros como mostra a Figura 1.

Figura 1 - Embalagens de vidro

Fonte**:** Brasil Escola (2020)

Por volta de 1830, iniciou-se a comercialização de enlatados de alimentos utilizando estanho em sua fabricação. Com o começo da primeira guerra mundial, o preço do estanho estava elevado, fazendo com que os produtores de embalagens procurassem por novas alternativas de matéria-prima, iniciando assim o ramo de embalagens a partir de ferro (COSTA, 2005). A Figura 2 mostra uma embalagem de ferro.

Figura 2 - Embalagem de Ferro

Fonte**:** AYALA (2018)

Após a segunda guerra mundial surgiram os supermercados, e com eles a necessidade de produção em larga escala e estocagem. Para contornar este obstáculo, surgiram as embalagens de papel e papelão, como as que podem ser observadas na Figura 3, e também as embalagens plásticas que atenderam a estas necessidades do mundo pós-guerra (COSTA, 2005).

Figura 3 - Embalagem de Papelão

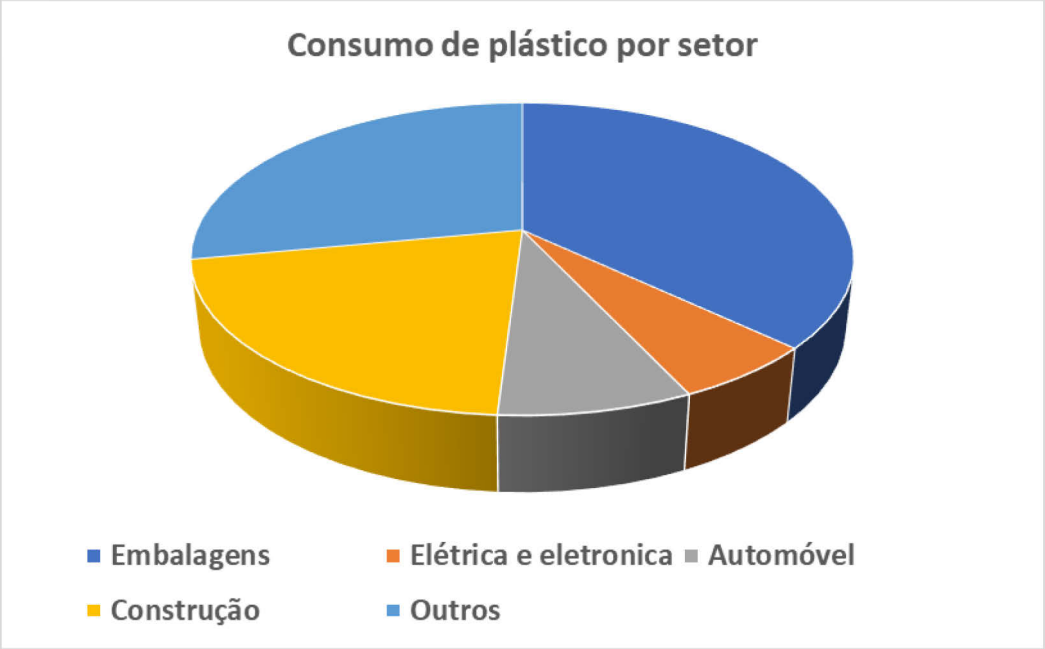
Fonte: KraftBox (2017)

De forma breve, embalagens plásticas são revestimentos feitos para com a finalidade de proteger e aumentar o seu tempo de vida útil. De acordo com Fachin (2006), para determinados produtos: “A embalagem de plástico pode ser adaptada a qualquer necessidade e constantemente melhorada para acompanhar o desenvolvimento dos próprios produtos e também dos hábitos e requisitos dos diversos tipos de consumidor com os seus diversos estilos de vida”.

As embalagens plásticas estão ganhando cada vez mais destaque no cenário brasileiro atual, devido a sua versatilidade e alta gama de propriedades

distintas (RODRIGUES,2018), sendo elas, o maior mercado do plástico atualmente, seguido da utilização para construções e produção de automóveis, respectivamente. Ademais, o setor agroalimentar é o que mais consome embalagens de plástico, sendo responsável por 65% das suas aplicações (PLASTVAL, 2008). A Figura 4 mostra o consumo de plástico por setor no mundo.

Figura 4 - Consumo de plástico por setor



Fonte**:** Adaptada de PlastVal (2020**)**.

Segundo a Associação Brasileira de Embalagens - ABRAE (ABRAE, 2018), no ano de 2017, o valor bruto da produção física de embalagens atingiu o patamar de R$71,5 bilhões, o que corresponde a um aumento de 5,1% em relação ao ano de 2016, sendo que, o setor de embalagens plásticas correspondeu a 38,35% do valor total desta produção. Ainda, segundo a ABRE, o setor de embalagens no brasil apresentou um crescimento de 3,0% no ano de 2019 (ABRE,2019). A Figura 5 apresenta alguns exemplos de embalagens plásticas.

Figura 5 - Embalagens plásticas

Fonte**:** Asterplas (2020).

Outro destaque importante do setor é que, em comparação com outras matérias-primas como vidro, fibras e metal, a resina plástica apresenta melhores características em relação ao processo produtivo, comercial e de reciclagem, com um maior reaproveitamento de produtos já utilizados, revelando uma vantagem do plástico perante aos seus concorrentes em termos de degradação ambiental, um ponto importante no desenvolvimento do setor de embalagens. (RODRIGUES, 2018)

Observando a crescente no mercado de plásticos tanto flexíveis quantos rígidos frente aos demais tipos de embalagens, o presente trabalho procura, por meio da utilização de redes neurais, melhorar a produção de embalagens plásticas flexíveis visando o aumento da velocidade de resposta para teste de qualidade e o aumento da eficácia da resposta obtida, pois, o mesmo, poderá reduzir gastos e possíveis erros de leitura, o que acarreta em desperdício de material já produzido.

Uma vez produzidas, as embalagens plásticas flexíveis são utilizadas em vários ramos e, em todos eles, para que sejam consideradas embalagens de qualidade, é necessário que atendam as *performances* mínimas de qualidade, específicas de cada setor, de propriedades físicas, como resistência a tração, resistência ao rasgo, alta força de laminação, resistência a perfuração e alta força de selagem.

De forma complementar ao estudo dos parâmetros das embalagens e com grande relevância, também se faz necessário o estudo de características

de cada filme que compõe a embalagem em si, para que este percorra com eficiência dentro do maquinário da empresa e, a posteriori, atenda a utilidade do cliente.

O coeficiente de fricção (COF) do material analisado deve atender a especificação dos equipamentos de corte, impressão e envase (maquinário do cliente). Tal parâmetro é avaliado em uma escala de 0 a 1 e, quanto mais próximo de 1 o material alcançar, mais atrito ele possui. Logo, quanto maior seu valor de COF, maior será a força necessária para que haja movimentação e deslize do filme nos maquinários.

Quando o COF está fora do padrão estabelecidos nos filmes de embalagem, problemas graves no processo de envase ocorrem, como por exemplo:

- Baixo COF: O material desliza mais que o necessário e escapa da pinça de coleta, perdendo o envase e podendo ocasionar em grandes perdas de lotes.

- Alto COF: O material trava a máquina, impedindo que o processo ocorra em regime continuo.

Normalmente, o estudo e avaliação do COF feito pelas empresas produtoras de filmes, é realizado embalagem por embalagem, utilizando uma máquina de ensaio de atrito, como a ilustrada na Figura 6.

Figura 6 - Equipamento de leitura de COF

Fonte: medicalExpo (2020).

De forma a interferir e corrigir o coeficiente de fricção das embalagens, na estocagem é aplicado na superfície do filme um material denominado de *anti-block*, responsável por fazer com que os filmes, após o corte, não se aglomerem

em blocos (blocagem) enquanto estão armazenados em bobinas. Além disso, em certos casos, para o controle do atrito, utilizam-se aditivos (silicone ou erucamida) na composição dos filmes, sendo uma prática primordial para o bom deslizamento dos filmes nas máquinas.

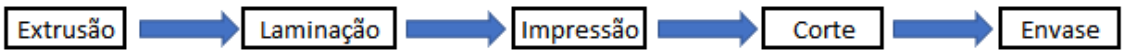
Tendo em vista a quantidade de variáveis que interferem no COF do filme e a necessidade de testar experimentalmente a influência de tais parâmetros, há uma grande demanda de tempo para a realização de testes experimentais para predição de suas características, tanto para a criação de novos filmes quanto para a aprovação do mesmo. Visando a otimização deste processo, este trabalho propõe a implementação de uma rede neural artificial para a determinação do COF de filmes plásticos flexíveis de até três camadas (3 canhões), com a utilização de características como: espessura total do filme, gramatura de cada camada, quantidade de *anti-block*, quantidade de aditivo em cada camada e densidade do filme.

2 REVISÃO BIBLIOGRÁFICA

2.1 Processo Produtivo

Logo, quando se trata de embalagens plásticas, deve-se considerar que elas são constituídas, basicamente, por filmes plásticos que, em sua maioria, são compostos por polímeros formados pela união de substâncias simples chamadas de monômeros. A produção destes filmes, pode ser realizada de diversas formas, distinguindo-se entre si de acordo com a necessidade de cada produto. Neste trabalho, serão estudados filmes do tipo flexíveis e seu processo produtivo inicia-se dentro de uma máquina extrusora com os pellets de resinas que compõem o filme, como descrito adiante com as demais etapas. A Figura 7 apresenta, de forma simplificada, as etapas para a fabricação de embalagens feitas a partir de filmes flexíveis.

Figura 7 - Diagrama de blocos referente às etapas para fabricação de embalagens



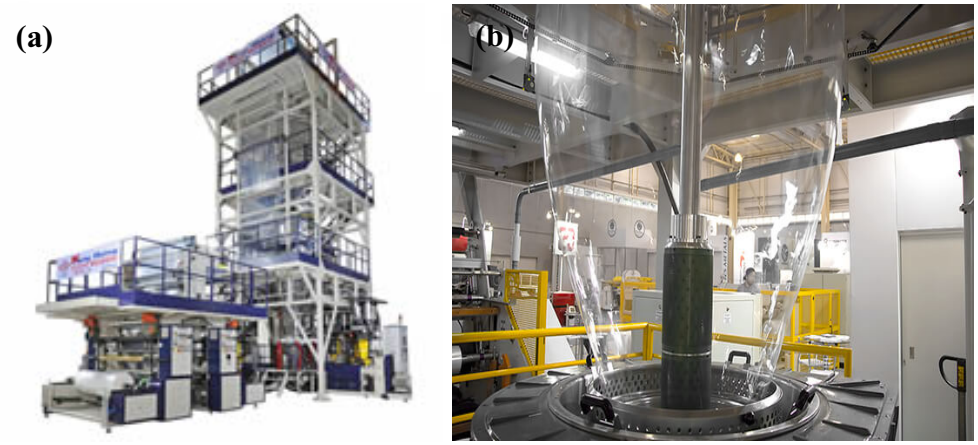
Fonte: O autor (2020)

1ª Etapa de processamento: Extrusão

A extrusão é realizada em um equipamento chamado extrusora (Figura 8(a)), constituído por uma ou mais entradas denominadas canhões, na qual se depositam os pellets de materiais.

Após passar pelos canhões, a resina é passada por um recipiente aquecido contendo um parafuso giratório, que realiza a homogeneização da massa formada pela mescla de resinas pré-selecionadas para que, a posteriori, essa mesma massa seja levada para o balão de extrusão (Figura 8(b)), no intuito de reduzir sua temperatura ao longo de sua passagem pelo balão e, assim, se transformar em um filme plástico.

Figura 8 - (a) Coextrusora parafuso; (b) Balão extrusor



Fonte: (a) Ming Jilee (2014); (b) Megateelmáquinas (2016).

2ª Etapa de processamento: Laminação

Após a produção do filme extrudado, o filme plástico segue para o processo de laminação, no qual ocorre a junção, por meio de adesivos, de dois (ou mais) filmes com características distintas, para alcançar as características requeridas do produto, isto é, da embalagem plástica desejada. Esta etapa pode, em certos casos, ser realizada junta com o processo de impressão, dependendo de qual maquinário estará sendo utilizado. Este processo pode ocorrer por flexografia ou rotografia. A Figura 9 apresenta a uma máquina de laminação.

Figura 9 - Máquina de laminação

Fonte**:** Hcpackingmachine (2020).

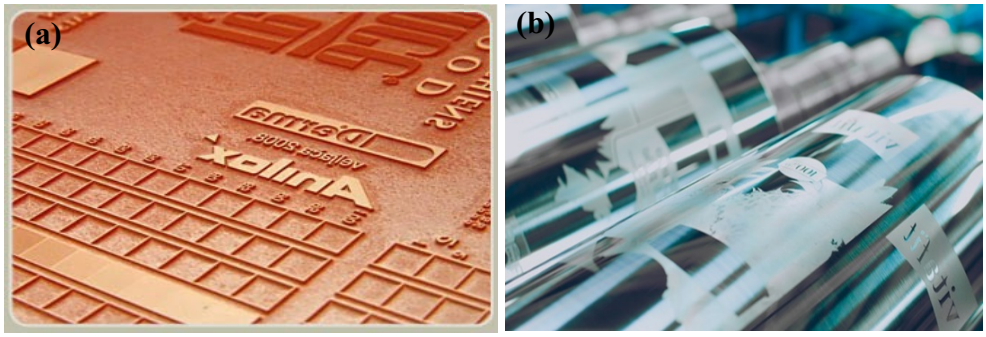
3ª Etapa de processamento: Impressão

A impressão pode ocorrer de duas maneiras, flexografia e rotografia, e a escolha depende da qualidade que é desejada. A rotografia traz maior nitidez para a impressão, porém, demanda mais capital do que a flexografia.

A flexografia ocorre com a utilização de chapas em alto relevo feita de borracha chamada de clichês (Figura 10 (a)), as quais são utilizadas em formato de rolo por onde o filme percorre. Cada clichê é responsável pela pigmentação de uma cor especifica, sendo necessário vários clichês para formar uma só arte (GRAVAPAC, 2020).

A rotografia também é realizada utilizando-se rolos para impressão (Figura 10(b)), porém, ao invés de clichês, utilizam-se cilindros revestidos de cobre e cromo. Por cima deste revestimento são gravadas as figuras utilizadas nas artes, tendo como característica principal o baixo relevo em seus cilindros.

Figura 10 - (a) Flexografia; (b) Rotografia



Fonte**:** (a) Flexografia total (2019); (b) Inova print (2017).

4ª Etapa de processamento: Corte

Sendo a última etapa de processamento antes da embalagem estar pronta, o corte é utilizado para que o filme atenda o tamanho requerido pelo cliente e seja dimensionado para rodar em sua própria máquina, pois todo filme produzido pelas extrusoras possuem a mesma largura, já as máquinas dos clientes são distintas, necessitando o corte adequado para se tornar uma embalagem útil. A Figura 11 apresenta uma máquina de corte.

Figura 11 - Máquina de corte

Fonte**:** Poli Máquinas (2019).

5ª Etapa de processamento: Envase

Realizado na maioria das vezes na empresa que solicita a embalagem (cliente), é o processo em que há alocação do produto dentro de sua embalagem, sendo realizado em distintas temperaturas e de diferentes formas, de acordo com as exigências do filme produzido. A Figura 12 mostra uma máquina de envase de sucos.

Figura 12 - Máquina de envase.

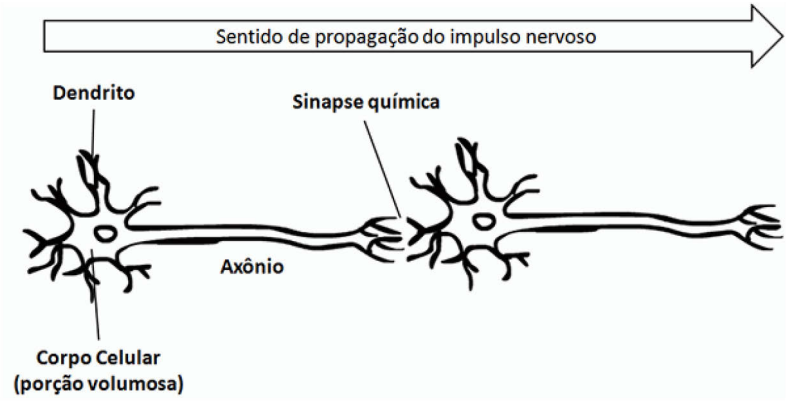
Fonte**:** Sulprint Embalagens (2019).

2.2 Redes Neurais

Redes neurais são formadas por um conjunto de neurônios artificiais que procuram simular a forma, o comportamento e as funções de neurônios biológicos. Ao comparar os dentritos (projeções de um neurônio), representados pela Figura 13, com as funções das redes neurais, observa-se que são semelhantes com as informações de entrada, denominadas de *Inputs*, e a

informação de comando que é gerado para o corpo, pode ser relacionada com uma função de ativação. Ademais, a ponderação para a tomada de decisão pode ser realizada atribuindo-se pesos, de forma a representar as sinapses neurais do corpo humano.

Figura 13 - Representação dos Dentritos



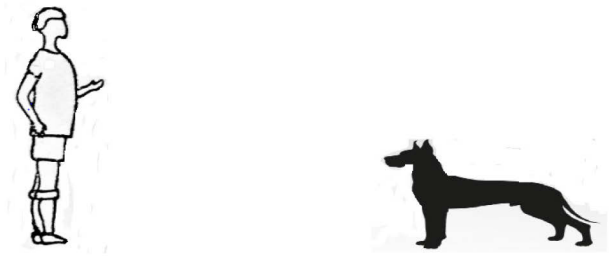
Fonte**:** Resumos de Biologia (2019).

Para que se possa aumentar os graus de liberdade da rede neural costuma-se utilizar uma variável denominada de *bias*, a qual possibilita um intervalo maior de valores de saída. Ainda fazendo a relação com os neurônios biológicos, depois de analisadas as possíveis decisões, o homem chega à uma conclusão e repassa esta informação ao corpo humano, que a executa. De forma semelhante, a rede neural pondera todos os pesos e, após analisá-los, também executa a ação final. Esta ação é denominada *target* da rede neural.

Rede neurais são utilizadas em vários ramos distintos, como em carros autônomos, identificação de imagens, reconhecimento facial, entre outros; e funcionam de acordo com os princípios básicos de sistemas de inteligência artificial.

Para exemplificar este comportamento, consideremos a situação hipotética de um homem avistando um cachorro, como ilustrado na Figura 14.

Figura 14 - Imagem ilustrativa de um homem avistando um cachorro.



Fonte**:** O Autor (2020)

Após avistar o cachorro, o homem averigua alguns detalhes, como, por exemplo, o semblante do animal, a distância entre eles, o tamanho do cachorro e como está o ambiente que os cerca. Esta captação de informações ocorre em milésimos de segundos e, após a coleta, o cérebro dará maior ou menor importância a cada uma destas informações, para que, assim, o homem tome uma decisão qualquer, seja se aproximar ou se distanciar do animal.

O comportamento do homem de analisar todos os detalhes ao seu redor representa o computador simulando as informações de entrada, o pensamento em relação à atitude a ser tomada, representa a multiplicação dessas entradas por cada peso pré-estabelecido e a ação tomada pode ser substituída pelo *output*, ou seja, a resposta que o programa dará ao analisar um determinado problema. No entanto, para que o homem chegasse a esta ação, ele necessitara ter passado pela fase de aprendizado, que quando comparada com as redes neurais, é representada pela etapa de treinamento. Para o ser humano, esta etapa pode acontecer de forma empírica, entrando em contato com cachorros quando criança para aprender o seu comportamento ou aprendizado guiado pelos pais. Já na rede neural, são fornecidas as respostas para que a rede pondere seus pesos e consiga, na sequência, encontrar a resposta sem a necessidade de seu valor tabelado.

Dentre as formas de aprendizado de redes neurais, a utilizada neste trabalho é denominada como Aprendizagem Supervisionada, na qual, o valor a ser buscado como *Target*, é fornecido de várias formas diferentes para a rede, a fim de que haja a estabilização dos neurônios e a obtenção de respostas diferentes das fornecidas pelo banco de dados após o final do treinamento. No entanto, com a Aprendizagem Não Supervisionada, os valores de respostas não

são fornecidos a rede, ocorrendo um aprendizado por reforços positivos ou negativos, dependendo da resposta dada pela rede. Há também outra variação de aprendizado denominado como Aprendizado Misto, no qual ocorre a mescla entre os dois tipos de aprendizado.

Há algumas diferenças entre as formulações utilizadas dentre as redes neurais existentes para a construção de algoritmos de inteligência artificial. A classe de algoritmos compreendidos na forma de ANN's (*Artificial Neural Networks*) abrange todas as outras categorias, sendo a mais genérica de todas as construções de rede, na qual os comandos da rede acontecem em sentido único, ou seja, a informação passa apenas uma vez por cada neurônio até alcançar o *target* e retornar ao início.

Uma das vertentes de redes neurais são chamadas de CNN (*Connvolutional neural networks*), as quais são as mais utilizadas atualmente. Diferentemente da forma mais generalizada (ANN's), as do tipo CNN possuem neurônios artificiais correlacionados, capazes de criar blocos de informações.

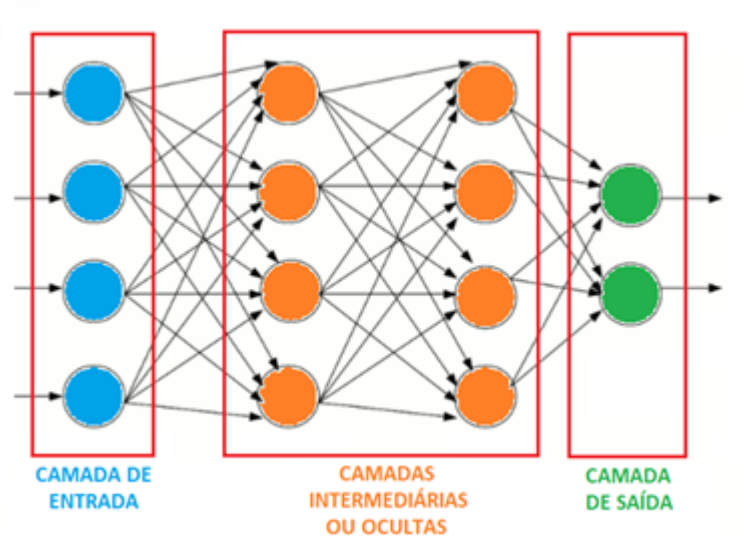
Além das CNN's, existem as redes neurais mais complexas, denominadas RNN's (*Recurrent neural networks*). Ao contrário das anteriores, as informações podem retornar ao neurônio para serem reavaliadas, ou seja, o sistema não avalia apenas o *output* encontrado, mas, sim, avalia as saídas de cada camada desde as camadas internas da rede até o *target*.

Neste trabalho, são utilizadas as redes neurais em sua forma mais simples, pois, ainda assim, apresentam grande eficácia em seus resultados. Dentre os modelos de rede ANN optou-se pela rede neural do tipo *Perceptron.*

2.3 Redes Neurais do tipo Perceptron

A rede *Perceptron* é uma das redes neurais mais objetivas e abrangentes a serem utilizadas em estudos preditivos, porém, de grande eficácia. Como um exemplo das principais aplicações, pode-se citar a sua utilização para a análise da concessão de crédito para consumidores (LIMA, 2009). O fluxograma básico do funcionamento de uma rede do tipo *Perceptron* está ilustrado na Figura 15.

Figura 15 - Estrutura típica de uma rede *Perceptron* multicamada



Fonte**:** Viceri (2020)

Como representado na Figura 15, inicialmente, na avaliação de um sistema, há o *input* das informações oriundas de um banco de dados previamente organizado e estabelecido (indicado pelas cores azuis). A rede neural recebe as informações que seguirão para o processo de análise, dados estes que são as variáveis diretamente relacionadas ao problema. Os *Inputs* variam de acordo com o problema a ser resolvido e cada *input* presente na rede possuirá um peso especifico, ou seja, cada *input* presente no modelo do sistema será multiplicado por um valor, de acordo com a sua relevância. Esta ponderação visa maximizar ou minimizar a influência do *input* na resposta do sistema e, a partir da informação de entrada, o comando é passado para a próxima camada presente na rede, representada pela cor laranja, para que a rede neural chegue em uma resposta que é comumente conhecida com *target* ou *output* (Cor verde).

Inicialmente, os pesos propostos são utilizados de forma aleatória, porém, durante o treinamento da rede neural, estes valores vão sendo poderados para atender as necessidades dos *input* e *output*/*target*, a fim de que seja alcançado o valor ideal para a obtenção da(s) resposta(s). Este método de trabalho é conhecido como *backpropagation*, no qual, a rede neural é treinada por meio de informações de *target* e *input* previamente conhecidas (SOARES, 2018).

Primeiramente, o sistema é equacionado para realizar operações matriciais multiplicando todas as entradas de 1 até *n* por seus respectivos pesos

somando-os ao bias de acordo com as seguintes equações (1-3):

$$uh_1 = \sum_{n=1}^{i} w_i x_i + bias \quad (1)$$

$$uh_2 = \sum_{n=1}^{i} w_i x_i + bias \quad (2)$$

$$\vdots$$

$$uh_n = \sum_{n=1}^{i} w_i x_i + bias \quad (3)$$

nas quais: h representa os neurônios artificiais intermediários; x são as entradas de informações; w os pesos associados a cada entrada e $bias,$ a variável que aumenta o grau de liberdade dentro da equação.

Após a obtenção dos valores internos dos neurônios artificiais, é necessário utilizar uma função de ativação, a qual possui características que facilitam o processo de aprendizagem da rede neural, pois trazem características de não-linearidade ao processo.

Esta função converte valores de grandezas diferentes para um espaçamento amostral entre 0 e 1, ou seja, independentemente do valor encontrado dentro de cada neurônio, este valor, após aplicada a função de ativação, se encontrará no intervalo entre 0 e 1, facilitando a comparação numérica e a ponderação. Utilizou-se a função sigmoide, pois, ela traz a não linearidade ao sistema e, quando utilizada como função de ativação, a saída de resposta não é linear (Deep Learning Book,2018).

Função sigmoide (4-6):

$$g(h1) = \frac{1}{1+e^{uh1}} \quad (4)$$

$$g(h2) = \frac{1}{1+e^{uh2}} \quad (5)$$

$$\vdots$$

$$g(hn) = \frac{1}{1+e^{uhn}} \quad (6)$$

Considerando um sistema de interação simples, pode-se descrever que o neurônio final é representado pelas equações (7-9):

$$uo_1 = \sum_{n=1}^{i} j_i gh_i + bias \quad (7)$$

$$uo_2 = \sum_{n=1}^{i} j_i gh_i + bias \quad (8)$$

$$\vdots$$

$$uo_n = \sum_{n=1}^{i} j_i gh_i + bias \quad (9)$$

nas quais, o neurônio final será representado pela letra u, os pesos presentes na segunda camada pela letra j e gh representa as i funções de ativação;

Encontrados os valores dos neurônios finais, aplica-se novamente a função de ativação, para que novamente os valores sejam normalizados e estejam em grandezas conforme as equações (10-12):

$$g(o1) = \frac{1}{1 + e^{uo1}} \quad (10)$$

$$g(o2) = \frac{1}{1 + e^{uo2}} \quad (11)$$

$$\vdots$$

$$g(on) = \frac{1}{1 + e^{uon}} \quad (12)$$

Devido à aleatoriedade dos pesos, é provável que a resposta da primeira iteração não seja condizente com a resposta esperada. Portanto, será necessária a correção dos valores dos pesos de acordo com a distância entre as respostas.

Para determinação do erro total da rede (E_{total}), será utilizada a expressão do erro quadrático médio, por meio da qual, é feita a comparação entre o valor encontrado pelo sistema e o valor real. Esta equação é muito utilizada em análises de investimentos financeiros e em probabilidade e estatística (Mais Retorno,2020) (Eq. 13):

$$E_{total} = \frac{1}{2}\sum_{k=1}^{n}\big(dk - g(ok)\big)^2 = Eo1 + Eo2 \quad (13)$$

na qual, os erros das respostas são escritos como Eo, a resposta é representada pela letra d e as respostas encontradas no sistema são descritas como $g(ok)$, sendo este erro calculado para todos os *outputs* presentes da rede.

Para atualização e correção dos valores dos pesos, é necessário encontrar os gradientes de erro, que indicam a distância do valor encontrado pela rede do valor *target* utilizado para o treinamento, pois, embora o valor exato não seja conhecido, sabe-se a direção para a qual o gradiente deve ir, seja buscando aumentar o valor do peso ou diminui-lo. Este processo deve ser realizado para cada um dos *outputs* o que torna o processo, muitas vezes, inviável de ser realizado manualmente (Eq. 14).

$$\frac{\partial E_{total}}{\partial w} = \frac{\partial E_{total}}{\partial gon}\frac{\partial gon}{\partial uon}\frac{\partial uon}{\partial w} \tag{14}$$

Após determinação do valor do gradiente, pode-se encontrar os novos valores dos pesos (Eq. 15).

$$w_n(t+1) = w_n(t) - \eta\frac{\partial E_{total}}{\partial w} \tag{15}$$

na qual, $w_n(t+1)$ será o novo valor do peso e $w_n(t)$ o primeiro valor do peso. η representa a taxa de aprendizado (previamente determinada) e o gradiente de direção $\frac{\partial E_{total}}{\partial w}$ que indica se os valores dos pesos deverão ser aumentados ou diminuídos.

Ao encontrar o novo peso, o processo se repete até que o erro seja minimizado, dado um critério de convergência, e as respostas possam ser avaliadas com confiança e representatividade.

Normalmente, para a escolha da rede neural ideal, muitas vezes utiliza-se a regressão linear de dados, porém, quando a mesma varia, pode-se utilizar a resolução dos gráficos *candlestick* com erro quadrático e a sua respectiva incerteza do sistema. Gráficos estes que indicam o ponto inicial e final de uma variável em um determinado tempo junto com sua incerteza. A equação (16) descreve a equação de incerteza utilizada:

$$i = \frac{\sqrt{\frac{\sum(\bar{e} - e)^2}{n - 1}}}{\sqrt{n}} \quad (16)$$

na qual, $\bar{e}$ é o erro médio presente no grupo amostral *e* é o erro amostral e n representa o número de amostras utilizadas.

Dentro da área proposta de pesquisa, há alguns trabalhos semelhantes, porém, não há nenhum que aplique nas mesmas funcionalidades/condições deste trabalho. Yang et al. (2012) no trabalho *An artificial neural network for predicting the friction coefficient of deposited Cr1−xAlxC films* propuseram a utilização de redes neurais para a predição do coeficiente de atrito de filmes com depósito de Cr1−xAlxC, e encontraram aproximadamente apenas 0,97% de erro em sua rede neural, mostrando assim sua validade e viabilidade de aplicação.

Deiab (2009), no artigo *On Prediction of Friction Coefficient Using Artificial Neural Networks,* comparou o COF encontrado experimentalmente e o COF encontrado pela rede neural de peças metalizadas encontrando resultados razoáveis de erro com a diferença entre a resposta e o valor verdadeiro menor que 0,01 em grande parte das análises. Nasir et al. (2009), no trabalho *An artificial neural network for prediction of the friction coefficient of multi-layer polymeric composites in three different orientations,* utilizaram aproximadamente 7389 corpos de prova de polímeros orientados em todos os sentidos para realizar os testes de coeficiente de atrito e a posteriori utilizar seus dados nas redes neurais, e encontraram uma alta acurácia superior a 90%.

Roushangar e Homayounfar et al. (2015) publicaram um trabalho intitulado *Prediction of Flow Friction Coefficient using GEP and ANN Methods* e utilizaram rede neurais para a determinação de coeficiente de atrito em tubulações, notando que a redes neurais juntamente com a programação genica obtiveram boas respostas, em suas melhores condições encontraram um R^2=0,958.

Nematollahi e Khaneghah (2019) no artigo *Neural network prediction of friction coefficients of rosemary leaves* estudaram o coeficiente de atrito de folhas de alecrim com superfícies variadas (aço galvanizado, vidro entre outras). Quando utilizado MLP (*Multi Layer Perceptron*) para a determinação desse atrito

obtiveram um coeficiente de regressão R^2 =0,968, sendo este um valor considerável e eficaz para uma rede neural.

Portanto, a redes neurais surgem com um caráter inovador para a otimização dos testes de qualidade dos filmes plásticos flexíveis apresentando um grande potencial para a predição do coeficiente de atrito, trazendo ao processo maior confiabilidade, praticidade e velocidade na obtenção de respostas.

3 METODOLOGIA

Para a análise e predição do Coeficiente de fricção (COF) dos filmes plásticos, foi utilizada uma rede neural artificial multicamadas do tipo *Perceptron*, utilizando comandos auxiliares dos softwares *Matlab*® e *Python*. Neste processo, foram utilizados 65 padrões previamente analisados pela rede, ou seja, 65 resultados de análises de COF obtidos experimentalmente na produção de filmes plásticos flexíveis e que alimentam o banco de dados da rede.

A Tabela 1 exemplifica o banco de dados utilizado, sendo que, os dados escritos na cor preta são os dados de entrada e o destacado em vermelho seria o valor de *target*, valor este que a rede neural deveria alcançar. Inicialmente, tem-se o peso total do filme e, em seguida, os dados subdivididos em 1, 2, 3, o que representa cada canhão da extrusora, dividindo, assim, cada camada do filme coextrudado. Porém, os dados estão distorcidos por questões de políticas de segurança da empresa.

Tabela 1- Tabela reduzida do banco de dados

Parâmetros	Valores
Peso total	355,3438
Gramatura camada1	123,0236
LayerRatio1	256,8804
Densidade1	245,0836
Slip 1	9089,5
AB1	52311
Gramatura Camada 2	116,123
LayerRatio2	242,5598
Densidade2	10,6848
Slip 2	9089,5
AB2	7791
Gramatura Camada 3	116,123
LayerRatio3	242,5598
Densidade3	10,61906
Slip3	9089,5
AB3	7791
COF	1,4098

Fonte**:** O Autor (2020)

Utilizou-se, inicialmente, um banco de dados com as composições de 50 filmes para o primeiro treinamento, no qual analisou-se o comportamento da rede frente aos dados, pois, a partir deste teste iniciaram-se os ajustes até atingir os parâmetros desejáveis. Para as entradas (*Inputs*), serão consideradas as características que influenciam diretamente a magnitude do COF divididas por camadas, camadas estas que correspondem aos canhões de coextrusão. Estes dados são calculados com a ponderação de composição com suas respectivas resinas internas, são elas:

- Peso total do filme: Altera propriedades mecânicas do filme como resistência a tração, resistência a rasgo, resistência a perfuração e regula interferências realizadas por gordura, água e o ar.

- Gramatura de cada camada: determina a quantidade de massa presente em uma área pré-estabelecida. Em filmes, utiliza-se normalmente g/m^2.

- Quantidade de anti-block (AB): Aditivo utilizado para que não haja a aderência das camadas da bobina. Este processo recebe o nome de blocagem.

- Quantidade de aditivo deslizante em cada camada (Slip): responsável pelo deslizamento do filme dentro dos maquinários.

- Densidade do filme: regula o peso do material e auxilia em propriedades mecânicas.

- LayerRatio: Valor total da Proporção de cada resina dentro de cada camada.

Para a criação da rede neural, primeiramente realizou-se um levantamento das propriedades de 50 filmes distintos com 3 camadas de filme (i.e., trilaminados) já produzidos em uma empresa de fabricação de filmes plásticos da região de Londrina, sendo estes dados, relacionados aos valores citados anteriormente. Após compilados estes dados, gerou-se uma planilha para a organização e, posteriormente, transferência de informações para os softwares de avaliação da rede neural. Com estes dados, foi realizado o treinamento da rede neural, que consiste nas etapas presentes no processo de *backpropagation* descrito previamente. Após o treinamento, realizou-se o primeiro teste de resposta da rede, com as propriedades de 5 novos filmes como entradas na validação da rede e verificou-se o erro em relação ao valor verdadeiro, fazendo com que haja ajuste na quantidade de dados até que o erro

esteja dentro de um padrão pré-estabelecido anteriormente. Este erro varia de acordo com o problema a ser estudado.

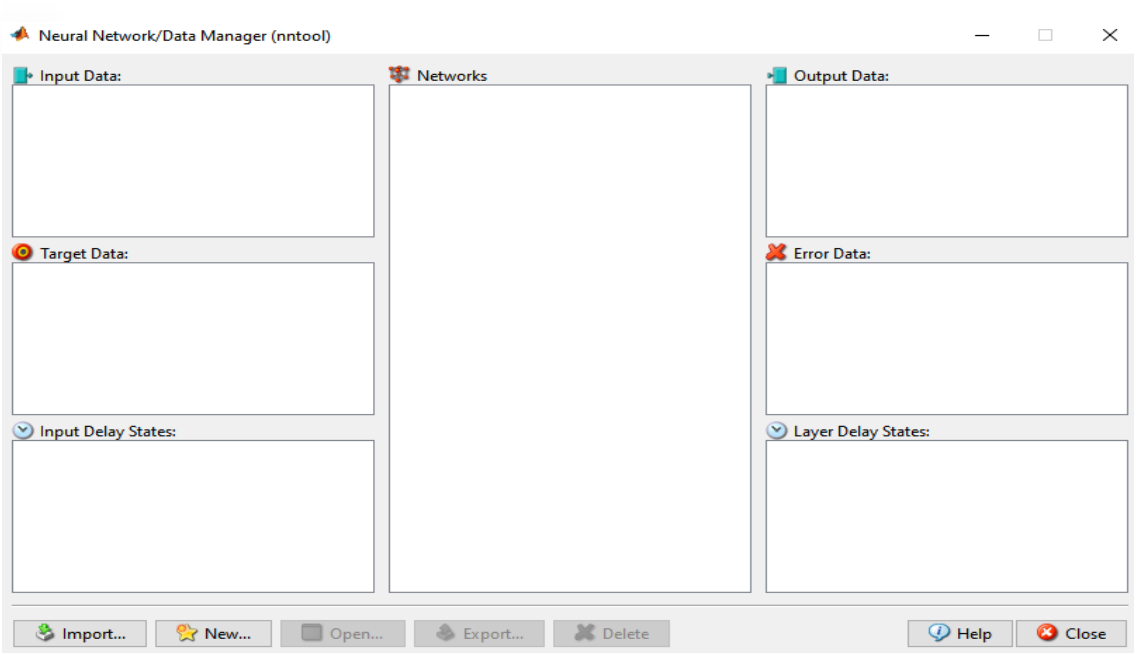
3.1 Configurações em Matlab

Para configuração da rede Perceptron no *Matlab*, utilizou-se a ferramenta suplementar do *Matlab* chamada *nntool*, a qual possui uma interface capaz de avaliar redes neurais internas, na qual o usuário pode definir os parâmetros iniciais de controle.

Primeiramente, o banco de dados, na forma matricial, é importado para o *Matlab*. Após a importação, os dados são divididos entre valores *input* e valores *target.* Tendo em vista que a ferramenta do *Matlab* entende as variáveis em forma de linha, fez-se necessário a utilização da função *transpose* para que os valores que estavam em forma de coluna fossem convertidos para a configuração em linha.

Uma vez carregados os dados na forma de variáveis para serem trabalhadas na compilação do código do *Matlab*, há a inicialização da rede neural, utilizando o comando *nntool*. A Figura 16 mostra a interface de inicialização da rotina de aplicação da rede neural.

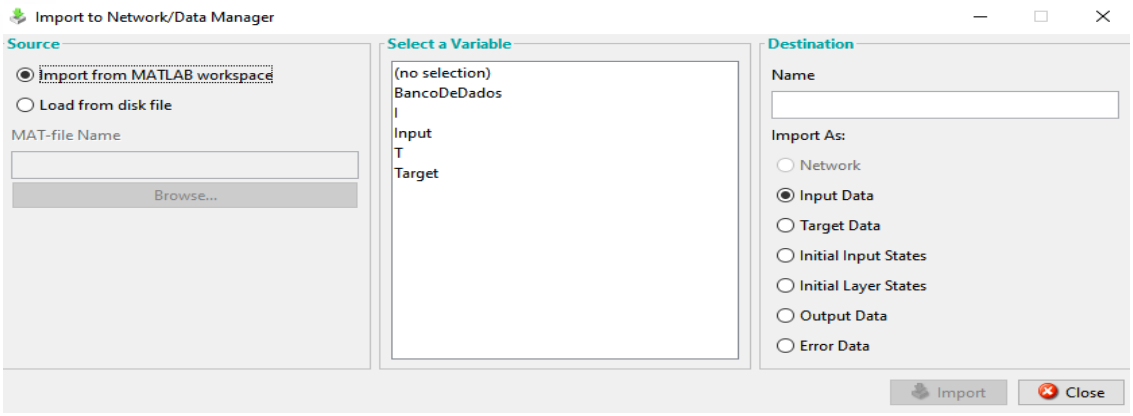
Figura 16 - Tela inicial de ferramenta *nntool*



Fonte: O Autor (2020).

Em seguida, para que os dados importados para o *Matlab* fossem selecionados no suplemento da rede neural, utilizou-se a função *import,* localizada no canto inferior esquerdo da tela, como mostra a figura 17.

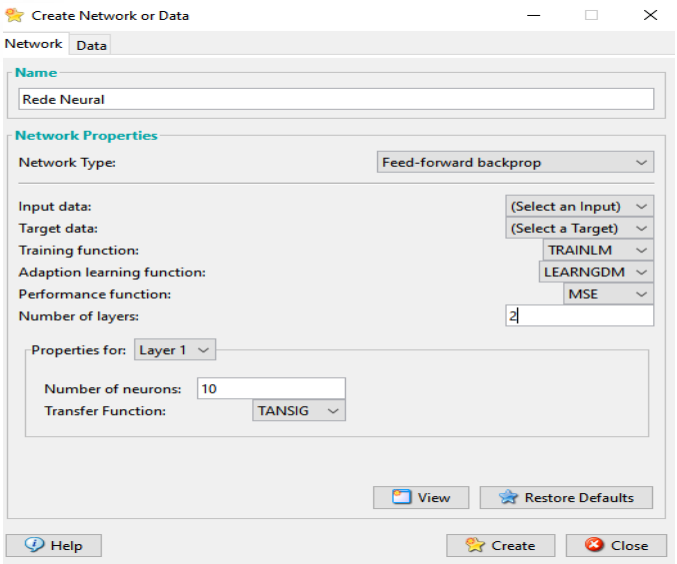
Figura 17 - Tela inicial de importação *nntool*



Fonte: O Autor (2020).

Neste momento, há a seleção dos dados para que os mesmos sejam divididos em *Inputs* e *Targets.* Após a seleção, deve-se configurar as especificações de aplicação da rede e o *nntool* disponibiliza para que os usuários façam modificações internas na mesma. As Figuras 18 e 19 ilustram a inicialização da criação da rede junto para as modificações dos parâmetros.

Figura 18 - Definições dos parâmetros da rede *nntool*



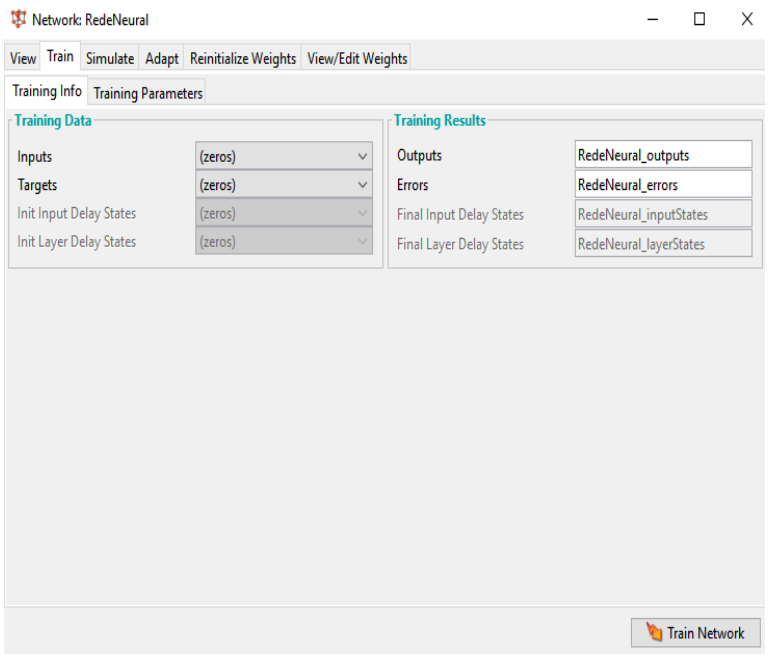
Fonte: O Autor (2020).

Nesta interface (Figura 18), o usuário pode definir o nome e o tipo de rede utilizada (*BackPropagation*), selecionar dentro do banco de dados do *nntool* quais serão os *inputs* e os *Targets* para o treinamento da rede, definir a fórmula utilizada para o cálculo do erro e, consequentemente, o término do treino, além das suas respectivas funções de transferência e o número de camadas internas.

Em ambas as redes, utilizou-se a fórmula de erro quadrático e a função sigmoide, porém, em *Matlab ,* utilizou-se apenas 10 camadas internas na rede, pois, como há poucos dados de entrada, caso o sistema tivesse uma quantidade excessiva de camadas internas, a rede poderia sofrer o processo de *overfitting*, no qual, ao invés de aprender e fornecer respostas, ela inicia uma padronização tão específica para aquele conjunto de dados que, se impostas características levemente diferentes, a rede já não corresponde de forma adequada.

Após configurada a execução da rede neural, utiliza-se o botão *open*, presente na Figura 16, para abrir a tela de treinamento da rede, mostrada na Figura 19. Nesta tela, há os campos de preenchimento para os *target* e inputs, nos quais, pode-se selecionar o que será utilizado como *input* e como *target* para aquela rede neural em especifico. Nestes campos há a importação de dados para dentro do sistema da rede neural.

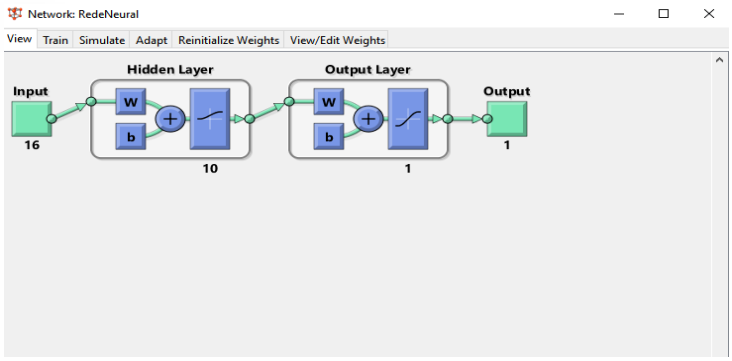
Figura 19 - Tela de treinamento *nntool*



Fonte: O Autor (2020).

Além de treinar a rede, este campo também permite ao usuário definir vários outros parâmetros para o seu melhor desempenho, como, visualizar a rede. Na Figura 20, pode-se observar estruturalmente como é montada a rede.

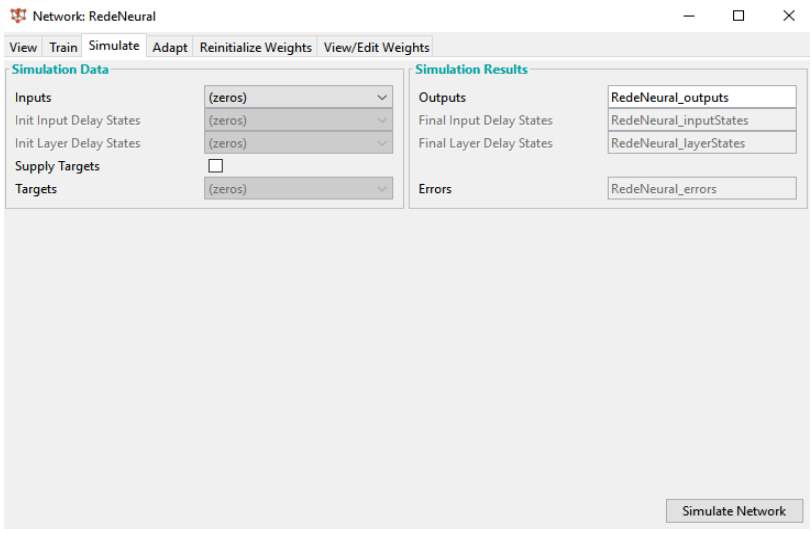
Figura 20 - Visualização de rede *nntool*



Fonte: O Autor (2020).

A Figura 21 mostra a interface de simulação da rede. Com esta ferramenta, é possível simular e salvar as respostas para serem analisadas posteriormente, sem interferir no treinamento da rede.

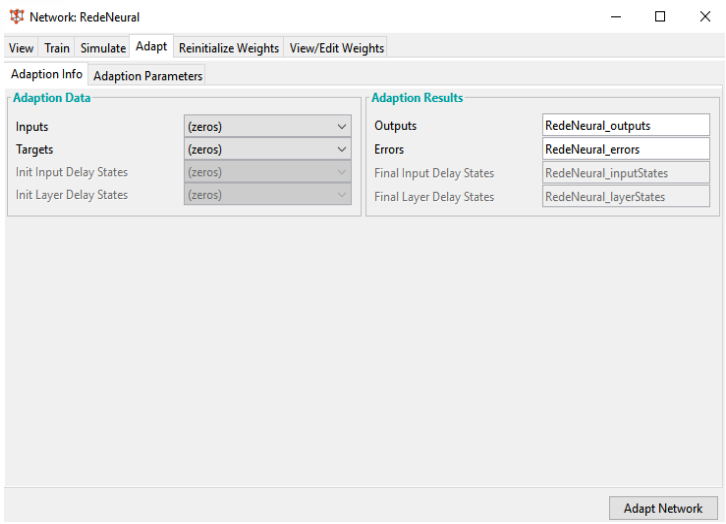
Figura 21 - Simular a rede *nntool*



Fonte: O Autor (2020).

Os parâmetros de adaptação (Figura 22), são utilizados para quando a rede neural já estiver treinada com os pesos estabelecidos, sendo adaptada para ser executada com um novo bando de dados.

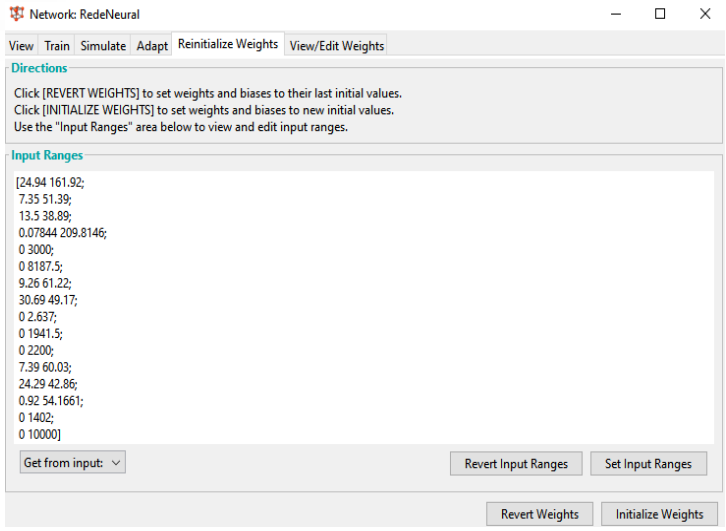
Figura 22 - Adaptar valores *nntool*.



Fonte: O Autor (2020).

Uma possibilidade de renovar as repostas dadas pela rede, é faze-la partir de locais diferentes, o comando de reinicializar os pesos da rede de forma randômica (Figura 23), faz exatamente esse processo, pois reinicializa os pesos para que a rede possa novamente calibrá-los.

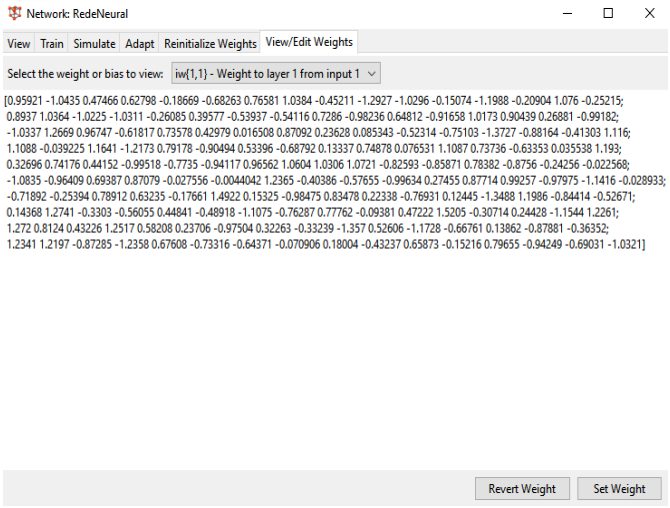
Figura 23 - Redefinição dos pesos *nntool*.



Fonte: O Autor (2020).

Quando há o conhecimento de um bom ponto de partida da rede neural a aba de ver/editar os pesos da rede pode facilitar o processo de aprendizado podendo editar os pesos para um melhor desempenho (Figura 24).

Figura 24 - Tela de edição dos pesos *nntool*.

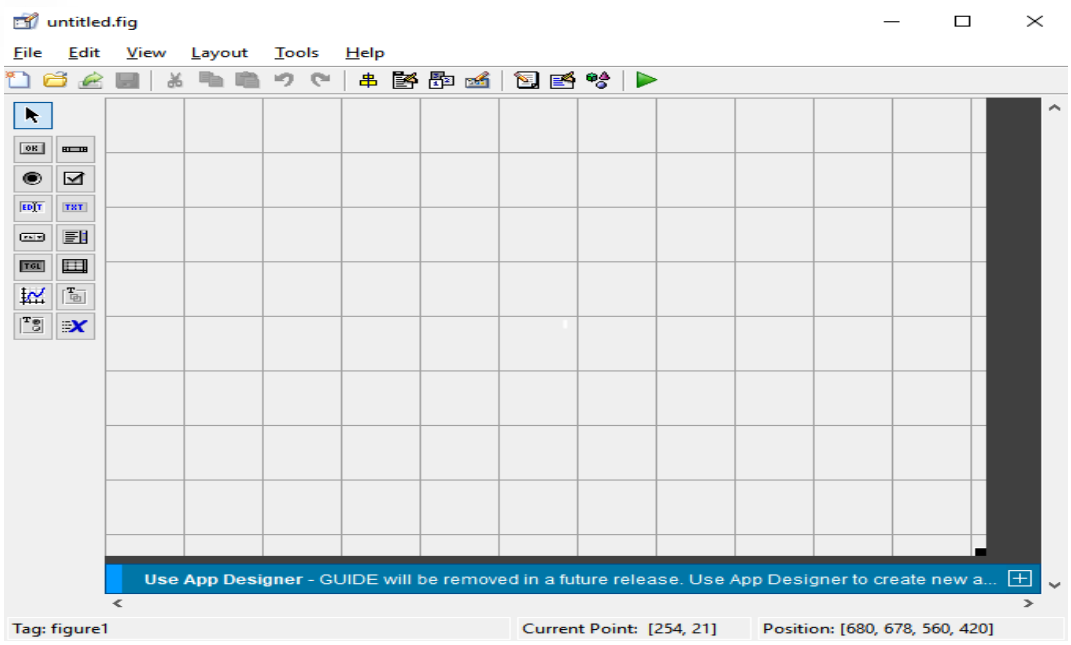


Fonte: O Autor (2020).

Após o treinamento da rede, a mesma estará apta para ser utilizada e realizar a predição do coeficiente de atrito dos filmes de prova.

Após a implementação da rede neural, propôs-se a criação de uma interface interativa para os usuários que venham a utilizar e aprimorar a rede posteriormente a este trabalho. Para isso, utilizou-se uma ferramenta própria do *Matlab* chamada GUIDE. A Figura 25 mostra a tela inicial da interface.

Figura 25 - Interface GUIDE



Fonte: O Autor (2020).

Dentro do GUIDE, o layout funciona de forma interativa, no qual é possível selecionar os botões, as barras e as informações que serão inseridas e posicioná-los de forma manual apenas com o auxílio do mouse.

Após a montagem do layout, faz-se necessário a criação das funcionalidades do sistema. Para isso, utilizou-se a ferramenta GUIDE do *Matlab*, que proporciona uma criação de interface interativa, de "arrasta e solta", podendo-se inserir botões, área de texto e caixa em branco para obtenção de dados. A partir disso, o código é gerado automaticamente, sendo necessário, somente atribuir um nome a cada caixa utilizada e atribuir funções internas ao botão. Dentro dos comandos do botão é coletado todos os dados inseridos pelos os usuários, e são transformados em números a partir do comando "*str2double*", e armazenados dentro de uma única variável chamada "*inp"*. Após coletados são transpostos a partir do comando "*transpose"* e lidos pela rede como *input* para gerar a resposta de COF. A Figura 26 mostra a captação das informações de entrada da rede neural e a posteriori apresentação da resposta do sistema.

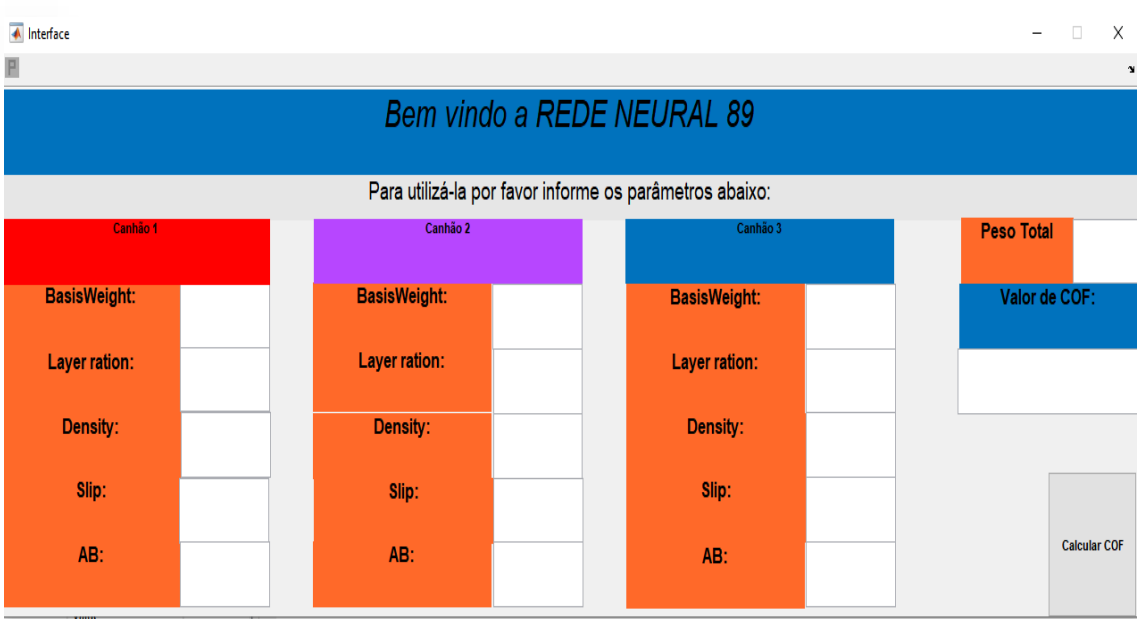
Figura 26 - Código da funcionalidade do botão no GUIDE.

```
function Calculo_Callback(hObject, eventdata, handles)
% hObject    handle to Calculo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
P = str2double  (get(findobj(gcf,"Tag","P"),'String'));
BW1 =str2double (get(findobj(gcf,"Tag","BW1"),'String'));
BW2 =str2double (get(findobj(gcf,"Tag","BW2"),'String'));
BW3 =str2double (get(findobj(gcf,"Tag","BW3"),'String'));
L1 = str2double (get(findobj(gcf,"Tag","L1"),'String'));
L2 = str2double (get(findobj(gcf,"Tag","L2"),'String'));
L3 = str2double (get(findobj(gcf,"Tag","L3"),'String'));
D1 = str2double (get(findobj(gcf,"Tag","D1"),'String'));
D2 = str2double (get(findobj(gcf,"Tag","D2"),'String'));
D3 = str2double (get(findobj(gcf,"Tag","D3"),'String'));
S1 = str2double (get(findobj(gcf,"Tag","S1"),'String'));
S2 = str2double (get(findobj(gcf,"Tag","S2"),'String'));
S3 = str2double (get(findobj(gcf,"Tag","S3"),'String'));
A1 = str2double (get(findobj(gcf,"Tag","A1"),'String'));
A2 = str2double (get(findobj(gcf,"Tag","A2"),'String'));
A3 = str2double (get(findobj(gcf,"Tag","A3"),'String'));
inp = [P; BW1; BW2; BW3; L1; L2; L3; D1; D2; D3; S1; S2; S3; A1; A2; A3]
input= transpose[inp]
RedeNeural[input]
set(findobj(gcf,'tag','COF'),'String',RedeNeural);
```

Fonte: O Autor (2020).

A figura 27 mostra a Interface criada em *Matlab* onde as camadas de filme são representadas pelos canhões e a gramatura é representada na interface pela nomenclatura de *BasisWeight* já as outras entradas estão com a mesma nomenclatura do que na tabela de entrada.

Figura 27 - Interface GUIDE pronta



Fonte: O Autor (2020).

3.2 Configurações em Python

Mesmo possuindo melhores funcionalidades e uma interface já preparada, o *Matlab* é um software licenciado e, por esse motivo, para comparação e validação dos dados, também houve a aplicação e treinamento da rede neural em programação *Python*, a qual é uma ferramenta gratuita e pode ser utilizada e melhor adaptada para o ambiente empresarial.

David Cornapeau desenvolveu o *sklearn* em 2007 em um evento internacional da google chamado "*Google Summer of Code*", o pacote veio sendo desenvolvido com mais outros programadores durante os anos para que ferramenta tenha como finalidade ter em sua biblioteca uma ampla variedade de algoritmos de aprendizado de máquina, com vários tipos de aprendizados. (scikit-learn,2020)

A Figura 28 apresenta o código desenvolvido em linguagem *Python* para aplicação da rede neural e, também, cada linha de programação necessária para sua execução.

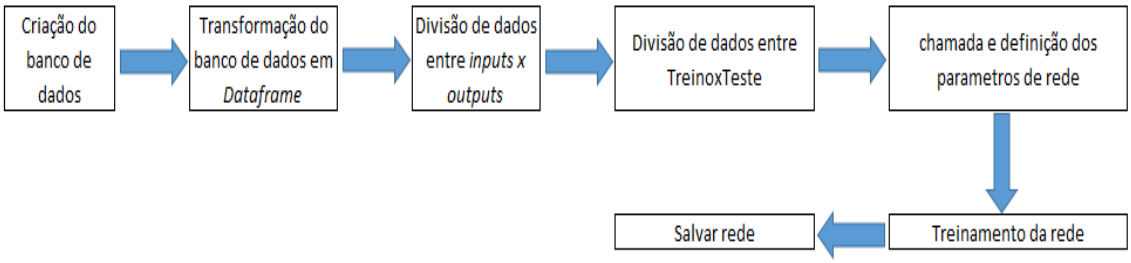
Figura 28 - Código *Python*.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
import random
from sklearn.metrics import mean_squared_error, r2_score
import pickle
from sklearn import metrics
from openpyxl import workbook
dados = pd.read_excel('C:\\Users\\Pedro\\Desktop\\BancoDeDados.xlsx')
df=pd.DataFrame(dados)

"entradas"
atributos =['BasisWeight1',    'LayerRatio',  'Density1',    'Slip1',
'AB1', 'BasisWeight2',    'LayerRatio',  'Density2',    'Slip2','AB2',
'BasisWeight3',    'LayerRatio',  'Density3',    'Slip3',   'AB3']

"target"
atrib_prev= ['COF']
"transformando os dados de entrada/Alvo do banco  de dados em data frame"
X = df[atributos].values
Y = df[atrib_prev].values

"dividindo os dados de treino e de teste"
split_test_size = 0.07
random_state =  500*random.randint(1,100)
X_treino, X_teste, Y_treino, Y_teste  = train_test_split(X,Y,
test_size=split_test_size, random_state= random_state)


"dados de iteração"
ppn=MLPRegressor(max_iter=50000000000,random_state=1*random.randint(1,100),
solver='lbfgs',activation="logistic", hidden_layer_sizes=80)
ppn.fit(X_treino,Y_treino.ravel())
Y_pred= ppn.predict(X_teste)
print("valores corretos: \n ", Y_teste)
print("valores respondidos pelo sistema: \n",Y_pred)

"valores de acompanhamento"
r = r2_score(Y_teste,Y_pred)
erro = mean_squared_error(Y_teste, Y_pred)
r2 = ppn.score(X_teste,Y_pred)
"prints"
print(r2)
print("O erro quadrado é de: \n", erro)
print("R2 é de \n",r)
"Graficos"
plt.scatter(Y_teste, Y_pred, color='Red')
plt.plot(Y_pred, Y_pred, color='Blue')
plt.title('Regressão \n')
plt.xlabel('Banco de Dados')
plt.ylabel('Dados Respondidos pela Rede')
plt.show()
rede= open('rede.py','wb')
pickle.dump(ppn,rede)
print(pickle)
```

Fonte: O Autor (2020).

Inicialmente, neste ambiente de programação, converteu-se o banco de dados que estava em formato do Excel (.xlsx) para *data frame* utilizando as ferramentas do Pandas, para que, assim, possam ser reconhecidos como variáveis de trabalho no *Python*. Com os dados carregados, separou-se os dados de *Input* que foram nomeados como *atributos*, e os dados *Targets* nomeados como *atrib_prev.* Após separados os dados, em *Inputs* e *Targets*, é necessária uma nova separação dos dados entre dados de teste e dados de treino. Esta divisão foi feita randomicamente, de forma que 2% do banco dados sejam sempre destinados para o teste. Após esta etapa, há a inicialização da rede. A rede neural utilizada no *Python* foi a rede *MLPRegressor*, da ferramenta *sklearn*, sendo esta, a rede neural utilizada para a resolução de problemas com variáveis discretas e contínuas. Dentro da rede neural chamada de "ppn" há a possibilidade de escolher alguns parâmetros, foi-se utilizado a função sigmoide como função de ativação além de usar o método de *backpropagation,* com uma camada interna de 80 neurônios. Após o treino, há a utilização da ferramenta *matplotlib e a sklearn.metrics* para a apresentação dos resultados em forma de gráficos e texto e a utilização da ferramenta *pickle* que salva a rede já treinada para ser utilizada na interface posteriormente. A Figura 29 mostra o Pseudocódigo do algoritmo desenvolvido em *Python* para a rede neural.

Figura 29 - Pseudocódigo em Python



Fonte: O Autor (2020).

De forma similar ao *Matlab* para a construção de uma interface, para o *Python*, utilizou-se a plataforma denominada *PythonSimpleGUI*, a qual diferentemente da ferramenta do *Matlab* não é interativa. Portanto, para criar a interface que interage com usuário, ou seja, o layout de entrada de informações pelo usuário (*front-end),* houve a necessidade de linhas adicionais de

programação no código. A Figura 30 apresenta o código utilizado na construção da interface de uso da rede neural desenvolvida em *Python.*

Figura 30 - Código Front-end da Interface desenvolvida em *Python.*

```
import PySimpleGUI as sg
import pandas as pd
import pickle
from sklearn import metrics

class Tela_Rede:
    def __init__(self):
        layout =[
            [sg.Text('PassoTotal' ),sg.Input(key= 'p')],
            [sg.Text('Canhão 1 (Vermelho)')],
            [sg.Text('BasisWeight'),sg.Input(key= 'bw1')],
            [sg.Text('LayerRatio'),sg.Input(key ='l1')],
            [sg.Text('Density'), sg.Input(key='d1')],
            [sg.Text('SLIP'), sg.Input(key='s1')],
            [sg.Text('AB'), sg.Input(key='a1')],
            [sg.Text('Canhão 2 (Purple)')],
            [sg.Text('BasisWeight'), sg.Input(key='bw2')],
            [sg.Text('LayerRatio'), sg.Input(key='l2')],
            [sg.Text('Density'), sg.Input(key='d2')],
            [sg.Text('SLIP'), sg.Input(key='s2')],
            [sg.Text('AB'), sg.Input(key='a2')],
            [sg.Text('Canhão 3 (Azul)')],
            [sg.Text('BasisWeight'),sg.Input(key='bw3')],
            [sg.Text('LayerRatio'), sg.Input(key='l3')],
            [sg.Text('Density'),    sg.Input(key='d3')],
            [sg.Text('SLIP'),       sg.Input(key='s3')],
            [sg.Text('AB'),         sg.Input(key='a3')],
            [sg.Button('Calcular COF')],
            [sg.Output(size=(50,20))]
        self.janela = sg.Window("Calculo do COF",size=(800,600)).Layout(layout)
    def Iniciar(self):
        while True:
            self.button, self.values = self.janela.Read()
            Bw1=self.values['bw1']
            L1=self.values['l1']
            D1=self.values['d1']
            S1=self.values['s1']
            A1=self.values['a1']
            Bw2 = self.values['bw2']
            L2 = self.values['l2']
            D2 = self.values['d2']
            S2 = self.values['s2']
            A2 = self.values['a2']
            Bw3 = self.values['bw3']
            L3 = self.values['l3']
            D3 = self.values['d3']
            S3 = self.values['s1']
            A3 = self.values['a3']
        Valor_Teste = [Bw1,L1,D1,S1,A1,Bw2,L2,D2,S2,A2,Bw3,L3,D3,S3,A3,]
            valor=pd.DataFrame(Valor_Teste).values
            v=valor.transpose()
            arq =open('rede.py','rb')
            redeneural = pickle.load(arq)
            y_resposta =redeneural.predict(v)
            print(y_resposta)
tela = Tela_Rede()
tela.Iniciar()
```

Fonte: O Autor (2020).

Primeiramente é importada a ferramenta *PySimpleGUI* e cria-se uma classe chamada de Tela de Rede. Dentro dessa classe utiliza-se o comando "*__init__"* que realiza a inicialização de um objeto dentro de uma definição para se iniciar o layout, dentro do layout cria-se as caixas de informação e a caixa em branco para a entrada de dados do usuário, logo após há a chamada da tela da interface definindo o nome e o tamanho da mesma. Coloca-se um comando *While* para que os próximos comandos funcionem enquanto uma linha de código for verdadeira. Dentro desse comando, os dados inseridos pelo usuário são guardados, transformados em um vetor de entrada no formato de *DataFrame*, após coletados os dados, é chamada a rede neural criada para que ela execute em cima dos valores coletados e retorne o valor de COF ao usuário. A Figura 31 mostra a interface criada em *Python*.

Figura 31 - Interface *Python* pronta para uso da rede neural.

Fonte: O Autor (2020).

Vale ressaltar que, previamente aos testes da rede, realizou-se uma reunião com os engenheiros responsáveis do setor de pesquisa e desenvolvimento da empresa e estipulou-se que, para que a rede seja eficaz, é necessário atingir uma regressão com eficácia superior a 70.

4 RESULTADOS E DISCUSSÕES

4.1 Resultados Matlab

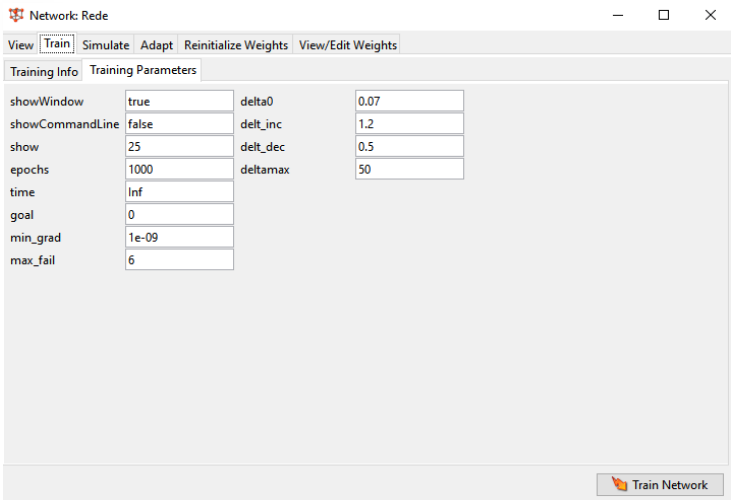
Para obtenção dos resultados dos testes realizados com o software *Matlab*, utilizou-se o código apresentado na figura 32 para, primeiramente, nomear uma nova matriz com dados de dentro da matriz base de banco de dados, essa nova matriz contém todos os dados de entrada e por esse motivo é nomeada como *Input*. A posteriori, selecionar os dados restantes que contenha os dados de COF do banco de dados e nomeá-lo como *Target*, após divididos, houve a transposição com a utilização do comando *transpose* dos dados para que estes possam ser lidos pela ferramenta *nntool.*

Figura 32 - Código *Matlab*

```
%valores para a rede neural com 55 dados
Input = BancoDeDados(1:65,1:16);
Target= BancoDedados(1:65,17);
I = transpose(Input);
T = transpose (Target);
%valores para a rede neural com 65 dados
I5 = transpose(BancoDeDados(1:55,1:16));
T5 = transpose(BancoDeDados(1:55,17));

nntool
```

Fonte: O Autor (2020)

Figura 33 - Interface de ajuste dos parâmetros *nntool*

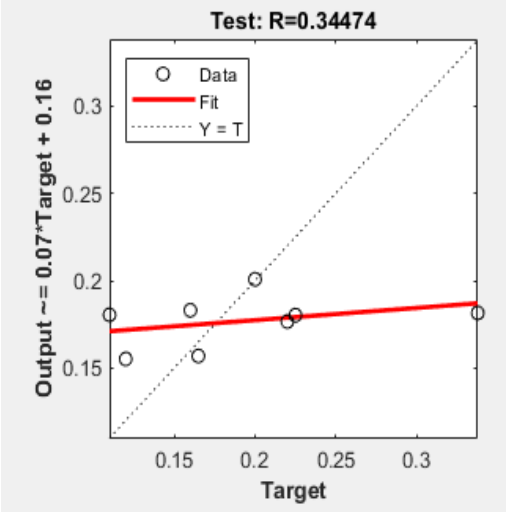


Fonte: O Autor (2020).

Dentro dos parâmetros propostos foi-se estipulado que a rede teria um número máximo de épocas de treinamento (*epochs*) igual a 1.000 tendo o tempo máximo para treinamento (*time)* infinito, ou seja, o tempo não será um fator definitivo para a rede. A classe *Show* está definida como true para que apareça o status do treinamento durante a execução. Já a classe Show*CommandLine* = false faz com que os comando de linhas implícitos a ferramenta não sejam exibidas na tela de comando. Goal é o objetivo de parada, e se definido com valores baixos e se a função cair abaixo desse valor, o sistema para de executar a rede, junto com todos os outros deltas proporcionalmente baixos que também estipulam a hora de parada do sistema, caso o gradiente do sistema esteja inferior ao valor de *mingrad* a rede também é finalizada.

Inicialmente, a rede neural foi avaliada com 55 dados coletados de COF (*Target)* e suas respectivas características (*Inputs)*, pois era necessário um número considerável de amostras para que houvesse uma grande variedade de valores de *output* que buscassem representar o *range* de valores reais para filmes coextrudados de três camadas de filme. Previamente. A Figura 34 ilustra graficamente a regressão realizada pela rede neural.

Figura 34 - Gráfico de regressão da rede neural com 55 dados *nntool*.



Fonte: O Autor (2020).

Como visto o gráfico de regressão dos dados encontrados pela rede *Matlab* na figura 32, o eixo y representa o eixo de resposta da rede e o eixo x os dados de COF presente no banco de dados. A linha tracejada representa onde

os valores do *target* seriam exatamente iguais aos valores respondidos, a linha vermelha representa a resposta encontrada pela rede, para os valores de teste escolhidos randomicamente pelo programa, já os pontos nomeados como *Data* representam os dados do banco de dados escolhidos randomicamente para serem utilizados como teste. Quando analisados, nota-se que ambas estão próximas, portanto, se obtém resultados medianos a partir do gráfico.

Nota-se que a resposta dada pela rede neural obteve um baixo desempenho, pois, como comentado anteriormente, para que houvesse uma resposta confiável, a rede neural necessitava de um desempenho superior a 70%, ou seja, ao menos a rede necessitava predizer ao menos 70% dos resultados (R^2>=70). Portanto como obteve um R^2<70 a rede neural treinada com 55 dados não obteve sucesso na predição COF dos filmes testados.

Tabela 2 - Respostas da rede com R^2=0,34

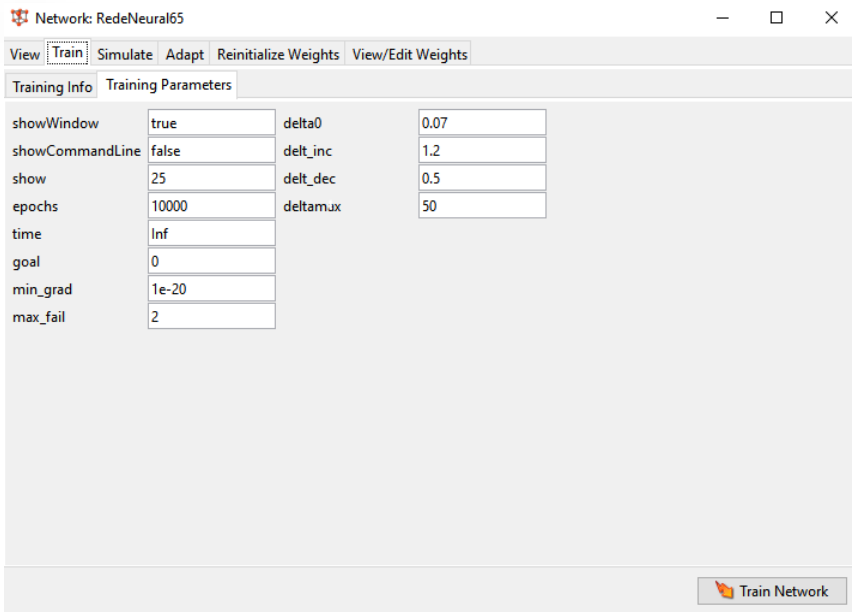
Target	**R^2 =0.34**
0,13	0,1691
0,16	0,1712
0,165	0,17155
0,23	0,1761
0,235	0,17645
0,2	0,174
0,1	0,167
0,34	0,1838

Fonte: O autor (2020)

Analisando os dados da tabela 2 nota-se que a rede neural 1 manteve os valores de respostas basicamente constante, com seu *range* visivelmente pequeno não conseguindo atingir valores mais altos, como o exemplo do *target*=0,34 ou valores menores com o *target*=0,13, sendo assim esta não é uma rede que corresponde aos critérios exigidos.

Em seguida, em busca de melhorar o desempenho da rede, ampliou-se o número de filmes do banco de dados de 55 para 65, mantendo todos os outros parâmetros de treino iguais, porém, reduzindo o valor de erro final de $1e^{-06}$ para $1e^{-20}$, e diminui-se a quantidade de erros máximos de 6 para o valor de apenas 2, a Figura 35 mostra as alterações realizadas.

Figura 35 - Tabela de edição dos valores de treino *nntool*

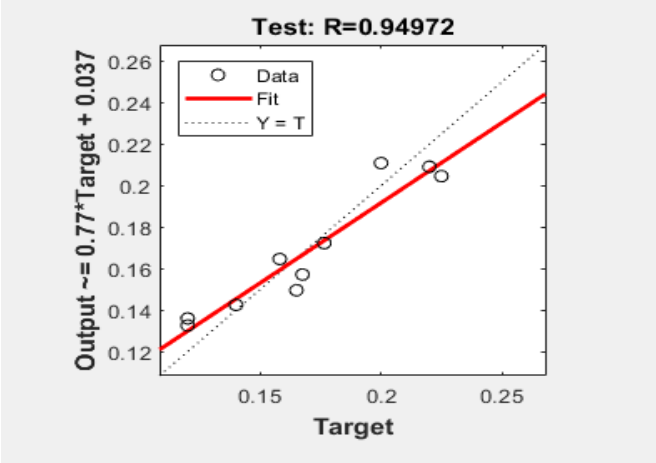


Fonte: O Autor (2020).

Tal mudança acarretou em uma diferença menor entre o valor encontrado e o valor *Target*, porém, aumenta-se o tempo de execução do algoritmo teoricamente. Contudo, pelo fato de que o aumento corresponde a um banco de dados inicial, esse aumento no tempo de processamento foi imperceptível.

Consequentemente, para a nova rede, a Figura 36 apresenta os resultados encontrados para uma rede treinada com 65 dados.

Figura 36 - Gráfico de regressão da rede neural com 65 dados *nntool*



Fonte: O Autor (2020).

Em oposição aos dados encontrados com 55 dados, a rede neural com 65 dados obteve grande sucesso em relação a sua regressão de treino, obtendo um aproveitamento de aproximadamente 95% em relação as respostas obtidas.

Tabela 3 - Respostas R^2=0,95

Target	**R^2 =0.95**
0,12	0,1294
0,125	0,13325
0,14	0,1448
0,16	0,1602
0,17	0,1679
0,175	0,17175
0,18	0,1756
0,19	0,1833
0,21	0,1987
0,225	0,21025

Fonte: O Autor (2020)

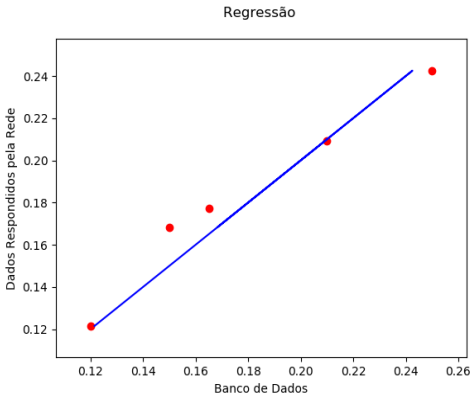
Analisando os dados da Tabela 3 nota-se que, diferentemente das respostas encontradas pelo primeiro treinamento da rede, o segundo treinamento trouxe mais assertividade em suas respostas, sendo que, mesmo com valores baixos com o de 0,12 a rede neural atendia o valor, além de também atender valores maiores como o de 0,225, superando o limite exigido para o seu desenvolvimento e validação.

4.2 Resultados em Python

Devido aos resultados encontrados em *Matlab*, nos quais foi possível verificar que a rede neural performa melhor com 65 dados do que com 55 dados, criou-se e testou-se a rede neural em *Python* já com 65 dados.

A Figura 37 mostra a curva de regressão criada pela rede neural programada em *Python*.

Figura 37 - Gráfico de regressão da rede neural com 65 dados *Python*



Fonte: O Autor (2020).

A Figura 37 mostra o gráfico de regressão dos dados encontrados pela rede *Python*, com o eixo y sendo o eixo de resposta da rede e o eixo x os dados de COF presente no banco de dados. A linha azul representa a resposta encontrada pela rede, para os valores de teste escolhidos randomicamente pelo programa, já os pontos vermelhos representam os dados do banco de dados escolhidos randomicamente para serem utilizados como teste. Quando analisados, nota-se que ambas estão próximas, portanto, se obtém um bom resultado a partir do gráfico.

A Figura 38 mostra a resposta do sistema tendo como dados os valores de COF destinados como *Target,* seguido dos resultados da rede neural e, por fim, os valores do erro quadrado e coeficiente de correlação (R^2) da curva de regressão (Figura 37).

Figura 38 - Tela de dados de respostas *Python*

```
valores corretos:
  [[0.21 ]
 [0.15 ]
 [0.25 ]
 [0.165]
 [0.12 ]]
valores respondidos pelo sistema:
 [0.20932554 0.16828947 0.24253495 0.17718714 0.12152021]
1.0
O erro quadrado é de:
 0.00010830484122062
R2 é de
 0.9485243150092111
```

Fonte: O Autor (2020).

A partir desta avaliação, nota-se que a rede neural testada em linguagem *Python* também atingiu valores superiores ao estabelecidos inicialmente e foram similares aos resultados obtidos com o software *Matlab*, com um R^2 de 95%.

4.3 Python x Matlab

A Tabela 4 traz uma comparação entre os R^2 dos melhores treinamentos das redes neurais, tamanho de código, Interatividade da interface em ambas as linguagens de programação consideradas.

***T*abela 4** - Dados comparativos *Python* x *Matlab*

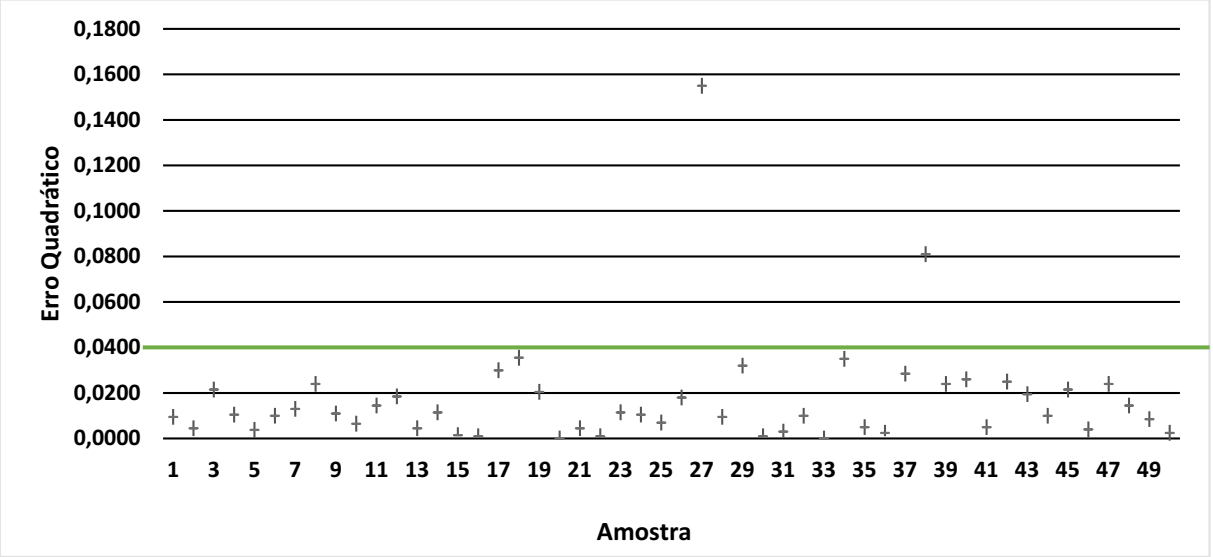
Linguagens	R^2	Erro Quadrado Médio	Tamanho do código	Interatividade da interface
Python	94,85%	0,000108	médio	Alta
MatLab	94,97%	0,000500	pequeno	Baixa

Fonte: O Autor (2020)

A Tabela 4 mostra que ambas as linguagens apresentaram ótimos resultados para a resolução do problema, ou seja, ambas as redes neurais descrevem com boa exatidão aproximadamente 95% das análises a serem feitas, com os erros médios baixos, sendo que em *Python* exige mais linhas de programação, porém há mais funcionalidades, e apesar de quase não houver linhas de código em *Matlab*, o mesmo tem poucas funcionalidades.

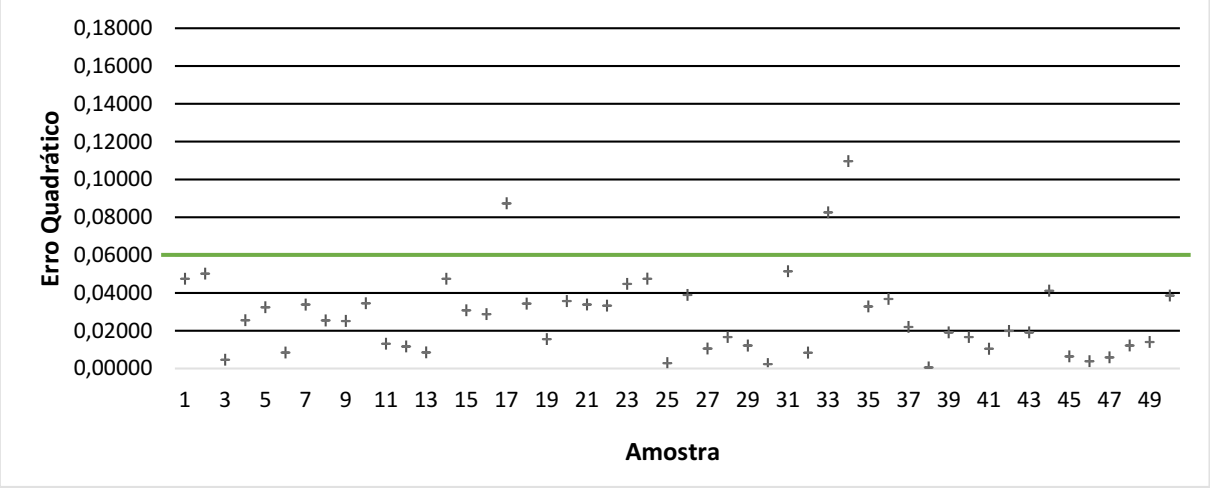
Mesmo com a eficácia alta em relação a regressão de dados, quando treinadas as redes por várias vezes consecutivas com diferentes bancos de dados, esta regressão pode variar. Logo, para tentar encontrar uma distinção entre as redes propostas, foi-se utilizada a rede para obter 50 respostas em cada interface e calculou-se o erro quadrado médio (Eq.13), determinou-se o erro médio para cada rede. A partir disso, calculou-se a incerteza de cada análise para que, com a compilação destes dados, seja possível a criação do gráfico de dispersão do erro junto da incerteza do sistema. A seguir, as Figuras 39 e 40 mostram graficamente suas respectivas dispersões em relação ao erro, sendo elas cabíveis de comparação.

Figura 39 - Gráfico de dispersão do erro *Python*



Fonte: O Autor (2020).

Figura 40 - Gráfico de dispersão do erro *Matlab*



Fonte: O Autor (2020).

Apesar das duas redes neurais testadas corresponderem as exigências impostas ao sistema, com base na análise dos gráficos, a rede da dispersão média dos erros em *Python* possui uma tendência melhor em relação a rede testada com o software *Matlab*, pois grande parte dos valores estão abaixo de 0,04, enquanto no *Matlab*, alguns valores se encontram próximos a 0,06 de erro. As incertezas para cada sistema obtiveram valores pequenos quando comparados com os valores encontrado dos erros, valores de aproximadamente 0,003, o que auxilia no bom desempenho da rede.

5 CONCLUSÕES

O trabalho realizado obteve sucesso em sua proposta, uma vez que, ambos os sistemas (*Python*/*Matlab*) atenderam a qualidade requerida obtendo em seus melhores treinamentos um R^2=0,9497 para a rede em *Matlab* e R^2=0,9485 para a rede em *Python*, com um erro médio muito inferior a 0,01. A partir do estudo de comparações dos dois sistemas notou-se que o sistema em *Python*, além de possuir maior interatividade, também obteve melhor desempenho em relação ao erro médio. As interfaces feitas para os usuários obtiveram um layout limpo, com a utilização de forma intuitiva e sem dificuldades. As redes neurais, se mostraram um caminho interessante para a melhoria do processo, pois, além de possuir uma resposta concisa, entrega-as com velocidade e de forma clara, podendo assim ser aplicadas futuramente a mais parâmetros e em maiores escalas além de utilizar as interfaces convertidas em aplicativos *mobile* para que seja utilizada de forma mais prática, ou serem utilizadas no auxílio de resolução de problemas.

6 REFERÊNCIAS

AGUIAR, F. G**. Utilização De Rede Neurais Artificiais Para a Detecção De Padrões De Vazamento Em Dutos.** 95f. Dissertação (Mestrado) - Pós Graduação em Engenharia Mecânica, Universidade de São Paulo, 2010.

ANALYTICS VIDHYA. CNN vs. RNN vs. ANN – **Analyzing 3 Types of Neural Networks in Deep Learning**. Disponível em: <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/ >. Acesso em 10 nov.2020

ASTERPLAS. Embalagem plástica personalizada. Disponível em: <https://asterplas.com.br/produtos/embalagem-plastica-personalizada/>. Acesso em: 20 de agosto de 2020.

AYALA, Germán V. **Embalagens metálicas: Folhas metálicas e de alumínio.** Acondicionamento e Embalagem para Alimentos. Universidade Federal de Santa Catarina. Florianópolis, 2018

BRASIL ESCOLA. Embalagem de vidro como solução para o ambiente. Disponível em: < https://brasilescola.uol.com.br/quimica/embalagem-vidro-como-solucao-para-ambiente.htm>. Acesso em: 17 nov.2020

COSTA, Marco A.C**. Gestão De Produção Em Uma Industria De Embalagens De Papelão Ondulado:** Caso Korruga Embalagens – Estudo de Caso**.** 96f. Dissertação (Estudo de Caso) - Graduação em Ciências econômicas, Universidade Federal de Santa Catarina, 2005.

DEIAB, Ibrahim M. On Prediction of Friction Coefficient Using Artificial Neural Networks**. International Symposium on Mechatronics and its Applications,** American university of Sharjah, março, 2009.

DEEP LEARNING BOOK. Capítulo 8: Função de Ativação**.** Disponível em: <http://deeplearningbook.com.br/funcao-de-ativacao/> Acesso em: 26 nov.2020.

FACHIN, A. L**. Indústria De Embalagens Plásticas Da Grande Florianópolis: Estudos sobre as condições competitivas.** Monografia - Graduação em Ciências econômicas, Universidade Federal de Santa Catarina, 2006.

FLEXO TOTAL. Clichês a variável de custos Flexografia. Disponível em: <http://flexografiatotal.blogspot.com/2015/04/cliches-variavel-de-custos-flexografia.html>. Acesso em 15 nov.2020.

GEEKSFORGEEKS. Difference Between ANN, CNN and RNN. Disponível em <https://www.geeksforgeeks.org/difference-between-ann-cnn-and-rnn/>. Acesso em 15 nov.2020.

GRAVAPAC. O que é Flexografia? Disponível em: <https://www.gravapac.com.br/o-que-e-flexografia/>. Acesso em: 24 abr. 2020.

INOVA PRINT. O processo de impressão offset. Disponível em:< https://medium.com/@inovaprintconteudo >. Acesso em: 17 nov.2020.

JÚNIOR, Jordão N.O. **Uso De Rede Neural Perceptron Multicamadas Como Estimador De tempos De Flicker.** 48f. Trabalho de conclusão de curso – Departamento de Engenharia Elétrica, Universidade de São Paulo, 2017.

KRAFTBOX.Disponível em: <http://www.kraftbox.com.br/produto/caixas-de-papelao/>. Acesso em: 20 de agost. 2020.

LIMA, F. G; PERERA, Luiz C J; KIMURA, H.; SILVA FILHO, A. C. Aplicação de redes neurais na análise e na concessão de crédito ao consumidor**. Revista de Administração.** Universidade de São Paulo. v.44,n.1, p.34-45, jan/fev/mar, 2009.

MAIS RETORNO. EQM (Erro Quadrático Médio). Disponível em: <https://maisretorno.com/blog/termos/e/eqm-erro-quadratico-medio#:~:text=O%20Erro%20Quadr%C3%A1tico%20M%C3%A9dio%2C%20abreviado,veremos%20ao%20longo%20do%20texto.&text=Vamos%20ent%C3%A3o%20ver%20algo%20mais,ser%20usado%20no%20mercado%20financeiro.

>Acesso em: 27 nov.2020**.**

MEGATEELMAQUINAS. Extrusora de stretch. Descrição. Disponível em:< http://megasteelmaquinas.com.br/produtos/extrusoras/extrusora-de-stretch/>. 15 nov. 2020.

MING JILEE. Coextrusora de cinco capas. Disponível em < https://www.plastico.com/temas/Coextrusora-de-cinco-capas+3090101>. 15 nov. 2020.

NASIR, T; YOUSIF, B.F; MCWILLIAM, S; SALIH, N. D; HUI, L. T. An artificial neural network for prediction of the friction coefficient of multi-layer polymeric composites in three different orientations**. J. Mechanical Engineering Science.** University of Nottingham, vol.224, julho, 2009.

NEMATOLLAHI, Mohammad. A; KHANEGHAH, Amin M. Neural network prediction of friction coefficients of rosemary leaves**. Food Processes Engineering.** Shiraz University, pag.1-12, julho, 2019.

OTEXT. 11.3 Neural network models. Neural network architecture. Disponível em: < https://otexts.com/fpp2/nnetar.html >. Acesso em: 15 nov.2020.

OVIDIO, R.O **ESTUDO COMPARATIVO ENTRE OS TOOLBOXES DE REDES NEURAIS *NNTOOL* E NFTOOL DO *MATLAB*** Trabalho de conclusão de curso – Curso de Engenharia Elétrica, Universidade Federal Do Pará, 2014.

PLASTVAL. Consumo**.** Disponível em: < http://www.plastval.pt/index.asp?info=plastico/consumo>. Acesso em: 20 agost. de 2020.

POLIMAQUINAS. **Polipouch 750**. Disponível em <https://polimaquinas.com.br/maquina-de-corte-e-solda-de-embalagem-pouch. Acesso em: 17 nov.2020.

RODRIGUES, Taynara T**. Polímeros Nas Indústrias De Embalagens.** 59.

Trabalho de conclusão de curso- Graduação em Engenharia Química, Universidade Federal de Uberlândia, 2018.

Resumos de Biologia. Tecido Nervoso (e mais!). Disponível em: < http://maxaug.blogspot.com/2013/11/tecido-nervoso_14.html>. Acesso em: 06 dez.2020

ROUSHANGAR, Kiyoumars; HOMAYOUNFAR, Farzin; Prediction of Flow Friction Coefficient using GEP and ANN Methods**. International Journal of Artificial Intelligence and Mechatronics.** Vol.4, pag.65-68, 2015.

SCKIKIT-LEARN. About us**.** Disponível em: < https://scikit-learn.org/stable/about.html#people>. Acesso em: 27 nov.2020

SOARES, Anderson S. “Redes neurais profundas – Deep Learning**”.** Disponível em < https://www.youtube.com/watch?v=DGNbd2FGw2s&t=8777s>. Acesso em: 10 jan. 2020.

SOUSA, Luci C F S; SOUSA, José S; BORGES, Maria G B; MACHADO, Antonio V; SILVA, Maria J S, FERREIRA, Reginaldo T F V; SALGADO, Alberto B. Tecnologia de embalagens e conservação de alimentos quanto aos aspectos físico, químico e microbiológico**. Agropecuária Cientifica No Semiárido.** Universidade Federal de Campina Grande, Vol.8,1ed, p. (19-27), janeiro-março, 2012.

SOUZA, Líria Alves de. "Embalagem de vidro como solução para o ambiente"; Brasil Escola. Disponível em: <https://brasilescola.uol.com.br/quimica/embalagem-vidro-como-solucao-para-ambiente.htm.> Acesso em 20 de agost. 2020.

SULPRINT. PROCESSO DE ENVASE: COMO OBTER O MELHOR DESEMPENHO OPERACIONAL. Disponível em:< https://blog.sulprint.com.br/processo-de-envase-como-obter-o-melhor-desempenho-operacional/ >. Acesso em 17 nov.2020

VICERI. Arquitetura de Rede Neurais Convolucionais para Reconhecimento de Imagens. Disponível em: < https://www.viceri.com.br/insights/arquiteturas-de-redes-neurais-convolucionais-para-reconhecimento-de-imagens>. Acesso em: 26 Nov.2020

YANG, Yu-Sen; CHOU, Jyh-Horng; HUANG, Wesley; FU, Tsow-Chang; LI, Guo-Wei etial. An artificial neural network for predicting the friction coefficient of deposited Cr1−xAlxC films**. Applied Soft Computing.** University Road, agosto, 2012.