Cinemática Inversa com Redes Neurais Aplicadas em Robôs Manipuladores

Gabriel Daltro Duarte[1], Claudio Pereira Mego Quinteros[2], Lincoln Machado de Araújo[3]

[1] gdaltrod@gmail.com. [2] claudioquinteros_ee@hotmail.com. [3]machado.lincoln@gmail.com. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB) / Laboratório de Instrumentação, Sistemas de Controle e Automação (LINSCA).

RESUMO

O mundo passa hoje pelo que alguns meios de comunicação estão denominando de quarta revolução industrial. Os robôs vêm ganhando cada vez mais espaço na indústria, onde o emprego de robôs está associado ao aumento da produção e da qualidade do produto. Para que um robô manipulador possa se movimentar com precisão, é necessário obter seu modelo cinemático inverso, porém a obtenção desse modelo requer a difícil solução de um conjunto de equações não lineares, não existindo um método de solução genérico. Diante desse problema, este trabalho tem como objetivo demonstrar a resolução do problema da cinemática inversa de um robô manipulador utilizando redes neurais artificiais sem a necessidade de modelar a cinemática direta do robô. Foram utilizadas duas metodologias de treinamento da rede neural, denominadas de treinamento *offline* e treinamento *online*. A diferença básica entre essas duas metodologias de treinamento é que, na metodologia *offline*, todos os pontos de treinamento são obtidos antes de ocorrer qualquer treinamento da rede neural, enquanto que, no treinamento online, o uso de um método de treinamento é recorrente. À medida que o robô se movimenta, novos pontos de treinamento são obtidos e os processos de treinamento com os novos pontos adquiridos são executados, permitindo um processo de aprendizagem da cinemática inversa contínuo. Para validar a metodologia proposta, foi desenvolvido um protótipo de um robô manipulado planar, com um grau de liberdade, e foram testadas várias arquiteturas de redes neurais, a fim de encontrar a arquitetura ótima. As redes neurais treinadas em modo *offline*, quando comparadas com as redes treinadas no modo *online*, apresentaram uma maior capacidade de generalização e um menor valor de erro de saída. O treinamento *online* só obteve resultados satisfatórios em arquiteturas de redes neurais com quantidades de neurônios muito maiores do que as quantidades utilizadas nas arquiteturas treinadas no modo *offline*.

Palavras-chave: Cinemática inversa. Robô manipulador. Redes neurais artificiais. *Back Propagation*.

ABSTRACT

Today's world is going through what is known as the Fourth Industrial Revolution. Robots have been gaining more and more space in the industry and going beyond expectations. The use of robots in industry is related to the increasing production and the quality of the electronic products. For an accurate movement of a robot manipulator it is necessary to obtain its inverse kinematic model, however, obtaining this model requires the challenging solution of a set of nonlinear equations. For this system of nonlinear equations there is no generic solution method. In view of this matter, this research aims to solve the problem of the inverse kinematics of a robot manipulator using artificial neural networks without the need to model the robot's direct kinematics. Two neural network training methodologies, called offline training and online training were used. The basic difference between these two is that in the offline methodology all training points are obtained before any training of the neural network occurs, whereas in online training the use of a training method is recurrent. As the robot moves, new training points are obtained and training processes with the new acquired points are executed, allowing a learning process of continuous inverse kinematics. To validate the proposed methodology a prototype of a manipulated planar robot with one degree of freedom was developed and several architectures of neural networks were tested to find the optimal architecture. The offline training methodology obtained very satisfactory results for most of the neural network architectures tested. The online training only achieved satisfactory results in neural network architectures with quantities of neurons much larger than the quantities used in the architectures used in offline training and still obtained inferior results. The neural networks trained in offline mode, when compared to the training networks in the online mode, presented a greater capacity of generalization and a smaller value of output error. The online training has only achieved satisfactory results in neural network architectures with quantities of neurons much larger than the quantities used in the trained architectures in offline mode.

Keywords: *Inverse Kinematics. Manipulator Robot. Artificial Neural Network. Back Propagation*

1 Introdução

O mundo passa atualmente pelo que alguns meios de comunicação estão denominando de quarta revolução industrial (PERASSO, 2016). Entre as principais características dessa revolução, podem ser citadas a grande presença de tecnologias digitais na economia, grande mobilidade e conectividade. Nesse contexto, a automatização de processos se torna algo essencial.

Assumindo de maneira massiva os postos de trabalho de pessoas em atividades tidas como repetitivas e exaustivas, os robôs vêm ganhando cada vez mais espaço na indústria e indo além disso. O emprego de robôs na indústria está associado ao aumento da produção e da qualidade do produto. As atividades de baixa remuneração e de média complexidade que eram consideradas insubstituíveis por robôs vêm sendo, aos poucos, ameaçadas. Uma grande variedade de robôs, cada vez mais complexos e dotados de cada vez mais sensores, vem sendo desenvolvida, a fim de suprir essa demanda da indústria. Para que isso possa ser alcançado, novas tecnologias devem ser desenvolvidas para que robôs com estrutura cada vez mais elaborada possam entrar em operação.

Um robô manipulador industrial pode ser considerado um conjunto de corpos rígidos, denominados elos, conectados em cadeia por juntas rotacionais ou prismáticas (CRAIG, 2012). No último elo da cadeia, é fixada a ferramenta de trabalho, que desempenha o papel para o qual o robô foi projetado. Uma junta rotacional permite que um elo gire em relação ao elo vizinho, e uma junta prismática permite uma movimentação linear entre dois elos. Para cada junta é associada uma variável, denominada variável de junta, que determina qual é a posição – angular ou linear – relativa entre os dois elos conectados a uma mesma junta.

Para que a ferramenta alcance no espaço a posição e a orientação desejadas com precisão, é necessário que todas as variáveis de junta do manipulador robótico assumam valores específicos. Segundo Craig (2012), determinar quais são os valores que as variáveis de junta devem assumir, para que um robô manipulador alcance uma posição e uma orientação específica no espaço, é um problema fundamental do uso prático dos manipuladores. Esse problema é conhecido como cinemática inversa.

As equações que modelam a cinemática inversa de um robô manipulador são não lineares e de difícil solução, não existindo um método único de solução (DAYA *et al.*, 2014). Resolver problemas que envolvem cinemática inversa de robôs manipuladores, segundo Duka (2014), é uma tarefa desafiadora e que tem um nível de dificuldade significativo. Isso se dá pelo fato de o próprio robô possuir equações trigonométricas não lineares responsáveis por descrever o mapeamento entre os pontos no espaço e um conjunto de ângulos de juntas.

As Redes Neurais Artificiais (RNAs) têm inspiração nas redes neurais biológicas presentes no cérebro. Haykin (2001) define que, na sua forma geral, uma rede neural é uma máquina projetada para modelar a maneira como o cérebro realiza uma determinada tarefa particular ou função de interesse.

As redes neurais são ferramentas eficientes na solução do problema da cinemática inversa, devido a sua capacidade de aproximar funções. O teorema da aproximação universal garante que uma rede neural alimentada adiante com, no mínimo, uma camada oculta, é capaz de aproximar qualquer função contínua e uma rede neural alimentada adiante com, no mínimo duas camadas ocultas, é capaz de aproximar qualquer função descontínua (HAYKIN, 2001).

As pesquisas envolvendo Redes Neurais Artificiais têm sido motivadas, de acordo com Haykin (2001), pelo reconhecimento de que o cérebro humano é capaz de realizar certos tipos de processamento de informações – como, por exemplo, o reconhecimento de padrões, percepção, controle motor – muito mais rapidamente que o mais rápido computador digital existente.

Trabalhos recentes vêm demonstrando que as pesquisas na área continuam crescendo e envolvendo novas áreas do conhecimento. Em Zou *et al.* (2017) foi utilizado um modelo de programação não linear, em um algoritmo genético, para um braço robótico capaz de pegar e selecionar melancias em uma linha de produção. Em Shi e Xie (2017), a adoção de algoritmos meta-heurísticos, como Adaboost, em conjunto com redes neurais foram aplicados na obtenção da cinemática inversa de braços manipuladores com até 6 graus de liberdade, com grande precisão e estabilidade. Soluções mais complexas, que exigem maior esforço computacional, também foram empregadas na solução de cinemática inversa, como a utilização de mapas de Kohonen, decomposição espacial, estimação de parâmetros utilizando métodos

de aprendizado de máquina e SVR (*Support Vector Regression*), apresentando relevantes resultados (MUSTAFA *et al.*, 2016).

Diante desse problema, este trabalho tem por objetivo aproximar a função cinemática inversa de um robô manipulador planar, de juntas rotacionais, com um grau de liberdade, utilizando redes neurais artificiais alimentadas adiante (*feedforward*) de múltiplas camadas.

2 Referencial teórico

O problema da cinemática inversa é assim denominado porque ele é o inverso do problema da cinemática direta. Enquanto na cinemática direta os valores das variáveis de junta são conhecidos e deseja-se saber a posição e a orientação do sistema de referência da ferramenta, na cinemática inversa deseja-se encontrar os valores dos ângulos das juntas que permitem que o sistema de referência da ferramenta de trabalho assuma uma posição e uma orientação desejadas (NIKU, 2015).

As diferentes técnicas para a resolução de problemas relacionados à cinemática inversa podem ser classificadas em algébricas, geométricas e iterativas. O método algébrico não garante soluções fechadas, enquanto que, no método geométrico, as soluções fechadas são destinadas a robôs com, no máximo, três juntas. Assim, o método iterativo converge apenas para uma única solução, podendo não ser eficiente em casos particulares (JHA; BISWAL, 2014).

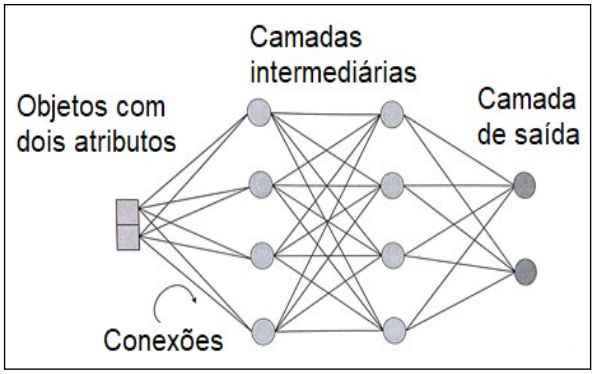
Segundo Duka (2014), um método eficiente para a obtenção da cinemática inversa de um robô manipulador é o de aproximá-la com a utilização de Redes Neurais Artificiais (RNA). As redes neurais conseguem lidar bem com a relação não linear entre os pontos do espaço que se deseja alcançar e o ângulo entre as articulações que permitem alcançar esse ponto no espaço. As RNAs podem, por meio do processo de aprendizagem, convergir para qualquer função matemática, inclusive a cinemática inversa de um robô manipulador.

As RNAs são sistemas computacionais distribuídos e compostos de unidades de processamento simples, densamente interconectadas, cujas unidades conhecidas como neurônios artificiais calculam funções matemáticas. Faceli *et al.* (2011) afirmam que as unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais.

Na maioria das arquiteturas, essas conexões, que simulam as sinapses biológicas, possuem pesos associados, que ponderam a entrada recebida por cada neurônio da rede. Os pesos podem assumir valores positivos ou negativos, dependendo do comportamento da conexão ser excitatório ou inibitório, respectivamente. Os pesos têm seus valores ajustados em um processo de aprendizagem e codificam o conhecimento adquirido pela rede. O processo de aprendizagem consiste em ajustar cada peso sináptico da rede para que ela sempre produza uma saída desejada para uma determinada entrada. Portanto, uma RNA é caracterizada por sua estrutura (quantidade de neurônios e camadas) e pelo seu algoritmo de aprendizagem.

A Figura 1 exemplifica uma RNA com a seguinte arquitetura: dois neurônios na camada de entrada, dois neurônios na camada de saída e duas camadas intermediárias com quatro neurônios cada uma.

Figura 1 – Rede neural artificial com múltiplas camadas.



Fonte: Faceli et al. (2011).

O processo de aprendizagem de uma rede neural pode ser realizado de duas maneiras: o aprendizado supervisionado e o não supervisionado. No primeiro processo, de acordo com Montgomery e Ludwig (2007), a rede é treinada com pares de entrada e de saída desejadas. Já no aprendizado não supervisionado, a rede não recebe informações sobre a saída desejada, sendo treinada apenas com valores de entrada e organizando sua estrutura de modo a fazer classificações desses valores em grupos.

Entre os tipos de redes neurais existentes, o *perceptron* é a arquitetura mais simples, apresentando duas camadas de neurônios: um conjunto de neurônios de entrada e um conjunto de neurônios de saída. Esse tipo de rede neural é simples e não pode ser utilizado

em aplicações avançadas pois, segundo Faceli *et al.* (2011), as redes *perceptrons* só conseguem classificar objetos linearmente separáveis. Para resolver problemas não linearmente separáveis utilizando RNAs, a alternativa é utilizar *perceptron* multicamadas que possui camadas intermediárias entre a camada de neurônios de entrada e a camada de neurônios de saída. A utilização de duas camadas intermediárias permite a aproximação de qualquer função.

De acordo com Faceli *et al.* (2011), o obstáculo que havia para utilizar RNAs multicamadas era a ausência de um algoritmo para treinamento dessas redes. Esse problema foi superado com um algoritmo de treinamento denominado *back-propagation*.

O *back-propagation* é dividido em duas etapas, uma etapa denominada *forward* e uma etapa *backward*. Na fase *forward*, cada objeto de entrada é apresentado à rede. O objeto é primeiramente recebido por cada um dos neurônios da primeira camada intermediária da rede, quando é ponderado pelo peso associado às conexões de entrada da camada. Cada neurônio nessa camada aplica sua função de ativação a sua entrada total e produz um valor de saída que é utilizado como valor de entrada pelos neurônios da camada seguinte. Esse processo continua até que os neurônios da camada de saída produzam um valor de saída que é comparado com o valor desejado.

Ainda na fase *forward*, a diferença entre os valores de saída produzidos e desejados para cada neurônio da camada de saída indica o erro cometido pela rede para o objeto apresentado. O valor do erro de cada neurônio da camada de saída é então utilizado na fase *backward* para reajustar seus pesos de entrada. O ajuste prossegue da camada de saída até a primeira camada intermediária. Esse processo é repetido várias vezes com diferentes objetos de treinamento até que o critério de parada seja atingido. Diferentes critérios de parada podem ser utilizados, como, por exemplo, um número máximo de ciclos de treinamento ou uma taxa máxima de erro.

3 Método da pesquisa

Para a implementação do sistema foram utilizados os seguintes componentes de *hardware* e *software*:

- Software Matlab;
- Placa Arduíno UNO;
- Sensor acelerômetro/giroscópio de três eixos, modelo MPU6050;
- Servomotor, modelo HS-422.

A metodologia utilizada neste trabalho se propõe a simplificar o processo de obtenção da cinemática inversa, utilizando-se as RNAs descritas na seção 2. Ao invés de obter o modelo cinemático direto do robô valendo-se de suas dimensões físicas e a representação de Denavit e Hartenberg (1955), o mapeamento (modelo cinemático direto) foi obtido utilizando-se o sensor acelerômetro-giroscópio MPU6050 da seguinte forma: seja o conjunto das n-uplas que representam todas as combinações que as variáveis de junta de um robô podem assumir. Sorteia-se aleatoriamente elementos de , formando o conjunto ilustrado na Equação 1.

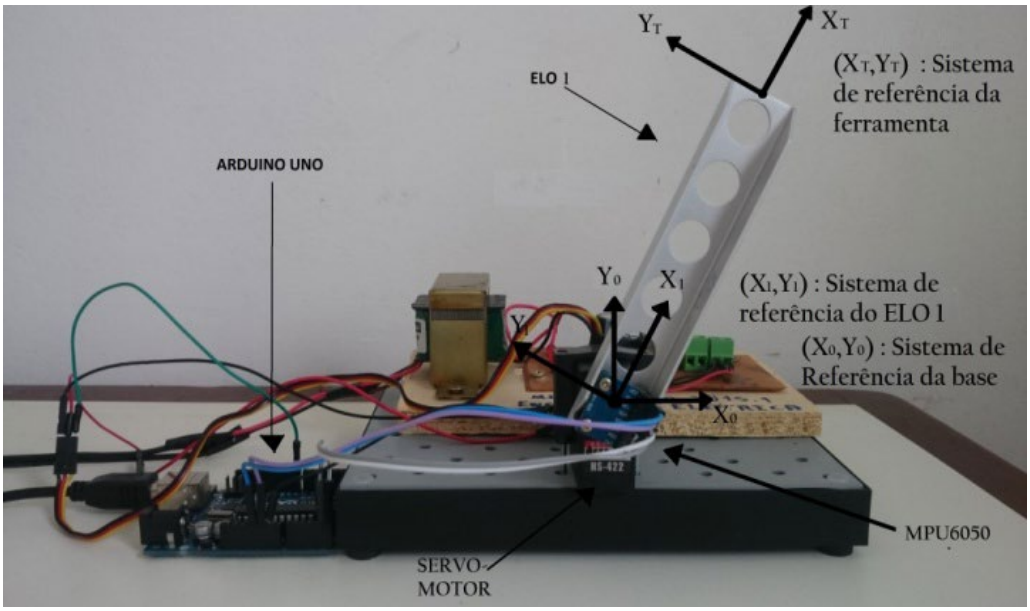
$$\Theta=((\theta_0,\theta_1,...,\theta_n)_1,...,(\theta_0,\theta_1,...,\theta_n)_k) \quad (1)$$

Posteriormente, cada junta do robô manipulador assume os valores de cada um dos elementos do conjunto . Para cada elemento , a posição e a orientação da ferramenta é determinada, não usando a cinemática direta, mas sim o sensor MPU6050 fixado no robô manipulador. Dessa forma, na metodologia empregada, não é necessária a realização de nenhuma modelagem matemática do robô.

O protótipo construído para a avaliação da metodologia proposta é ilustrado na Figura 2. O manipulador construído é constituído por um único elo de alumínio que é movimentado por um servomotor. O controle da posição angular do servomotor é realizado por meio do Arduíno Uno. Fixado ao único elo (ELO 1), há o sensor acelerômetro-giroscópio MPU6050 que, como dito anteriormente, permite a determinação da posição e da orientação do sistema de referência da ferramenta fixado no final do elo.

Como há um único elo, a posição e a orientação do sistema de referência da ferramenta pode ser determinado utilizando-se apenas a leitura do acelerômetro e o comprimento do ELO 1, não sendo necessárias as medidas de aceleração angular do giroscópio do MPU6050. Devido à resolução de 1 (um) grau do servo-motor, a variável de junta do ELO 1 só pode assumir 181 valores. Dessa forma, o ponto final do ELO 1, onde se localiza o sistema de referência da ferramenta, só pode alcançar 181 pontos no espaço.

Figura 2 – Protótipo construído para validação da metodologia proposta.



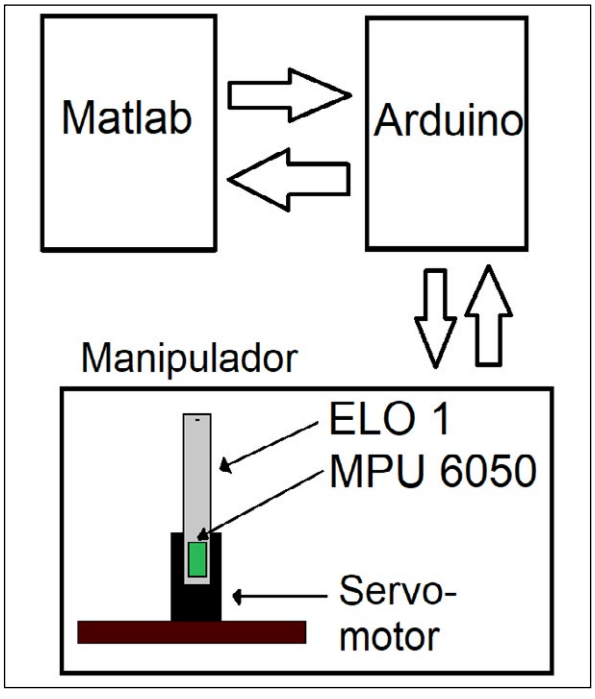
Fonte: Elaboração própria.

Toda a simulação da rede neural e o processamento dos dados foram realizados no Matlab®. O Arduíno apenas constitui uma interface que proporciona a comunicação entre o Matlab® e as outras partes do *hardware* que compõem o sistema, composto por um sensor MPU6050 e o servomotor no manipulador robótico.

Para que o Matlab possa se comunicar com o Arduíno por meio da porta USB, emulando uma comunicação serial COM, foram desenvolvidas bibliotecas para o Arduíno e para o Matlab®. Também foram desenvolvidas bibliotecas para que o Arduíno pudesse se comunicar e obter as leituras do sensor MPU6050 por meio do protocolo I2C. Para que as leituras de ângulo do MPU6050 tivessem maior confiabilidade, rejeitando ruídos e erros de leitura devido a vibrações às perturbações do meio, uma biblioteca para Arduíno foi desenvolvida para tratar estatisticamente os dados lidos do MPU6050. A biblioteca escrita em C++ utiliza o critério de descarte de amostras de Chauvenet (VUOLO, 1996). Para o controle de posição angular do servo-motor, foi utilizada uma biblioteca nativa do próprio Arduíno. A Figura 3, apresentada mais adiante, exibe a arquitetura do protótipo.

A aquisição dos pontos de treinamento e o treinamento da rede neural foram realizados de duas maneiras: treinamento *online* e treinamento *offline*.

Figura 3 – Arquitetura do protótipo.



Fonte: Elaboração própria.

3.1 Treinamento *offline*

No Matlab®, dos 181 valores de ângulo que a variável de junta do ELO 1 () pode assumir, sorteiam-se aleatoriamente 136 valores, formando o conjunto . Posteriormente, a junta do robô manipulador assume cada um dos valores enviando o valor de para o Arduíno, que, por sua vez, envia o sinal de controle para o servomotor assumir a posição angular desejada. Após o ELO 1 assumir a posição angular desejada, o Arduíno realiza a leitura das informações fornecidas pelo acelerômetro do MPU6050 e as repassa para o Matlab, que, baseado nessas informações, estima a posição e a orientação do sistema de referência da ferramenta.

A posição e a orientação calculadas, para cada um dos 136 ângulos, são salvas em uma variável do tipo vetor que representa um conjunto denominado. Posteriormente, redes neurais de diferentes arquiteturas são treinadas tendo o conjunto como conjunto de entradas e tendo o conjunto como conjunto de saídas desejadas. Dessa maneira, as diferentes redes testadas aprendem a realizar o mapeamento que consiste no modelo cinemático inverso manipulador. Após o treinamento, os 45 valores de ângulos não utilizados no treinamento são apresentados às redes neurais treinadas. Para cada rede, a resposta aos 45 ângulos não treinados é comparada com a resposta ideal, a fim de verificar se a rede foi capaz de generalizar a função cinemática inversa do manipulador.

3.2 Treinamento *online*

O treinamento *online* difere do *offline* quanto à sequência em que é realizada a obtenção dos pontos de treinamento e o treinamento em si. Ao contrário do *offline*, em que todos os pontos são obtidos para, posteriormente, iniciar-se o treinamento, no treinamento *online*, para cada ponto obtido, um novo treinamento é realizado com os pontos obtidos anteriormente juntamente com novo ponto. Espera-se que, à medida que a quantidade de pontos de treinamento cresça, o valor desse erro diminua.

Neste trabalho, as redes neurais foram treinadas para aprender a mapear a variável de junta do ELO 1, em função apenas da posição do sistema de referência da ferramenta, com a orientação desejada do sistema de referência da ferramenta não sendo considerada. Para um manipulador simples de um grau de liberdade, como o usado no experimento, só existe uma orientação possível para cada ponto do espaço de trabalho alcançável, e essa orientação é igual à variável de junta do ELO 1.

Em ambos os tipos de treinamento, todas as redes neurais possuem duas entradas, representando o ponto no plano XY – o qual o sistema de referência da ferramenta deseja alcançar – e uma saída, representando o valor que a variável de junta do ELO 1 deve assumir para alcançar a posição desejada no plano XY. No treinamento *offline*, foram testadas redes com uma camada oculta com cinco, dez ou vinte neurônios. As funções de ativação testadas na camada oculta foram: sigmoide, linear e tangente hiperbólica. Na camada de saída sempre foi utilizada a função de ativação linear.

No treinamento *online*, foram realizados testes com uma e duas camadas ocultas com 20, 30 e 50 neurônios em cada camada. Em todos os testes *online*, a função de ativação usada nas camadas intermediárias foi a tangente hiperbólica e, na camada de saída, foi usada a função de ativação linear. O algoritmo usado em todas as redes foi o *backpropagation*, e todos os pontos usados no treinamento e no teste, após o fim do processo, pertenciam ao espaço de trabalho do manipulador.

Os cálculos das simulações das redes que possuíam apenas uma camada oculta foram realizados utilizando-se apenas um processador Intel Core i5. Porém, os cálculos das simulações das redes que possuíam duas camadas intermediárias foram realizados em uma placa de vídeo NVIDIA GTX960, devido ao fato de as GPUs possuírem grande capacidade de processamento paralelo, diminuindo significativamente o tempo de simulação.

4 Resultados da pesquisa

Nesta seção serão apresentados os resultados obtidos pelas diferentes arquiteturas de rede testadas nos modos de treinamento *offline* e *online*.

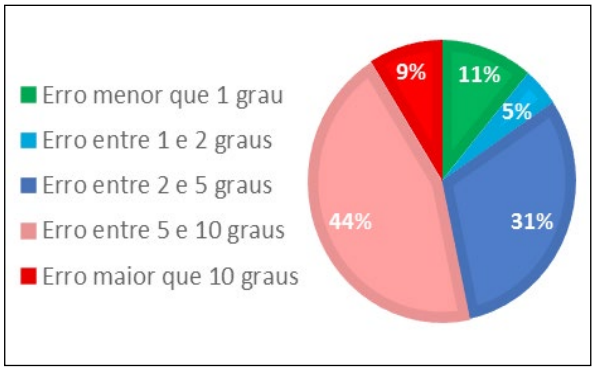
4.1 Treinamento *offline*

Nesta subseção é apresentado o desempenho por cada arquitetura de rede neural testada, após o processo de treinamento. O desempenho de cada arquitetura de rede é ilustrado por meio de dois gráficos, ambos no formato de gráfico de pizza, em que o percentual de pontos de teste está relacionado a cada faixa de erro estabelecida. As faixas de erro são: erro menor que 1 grau; erro entre 1 e 2 graus; erro

entre 2 e 5 graus; erro entre 5 e 10 graus; erro maior que 10 graus. O segundo gráfico compara a resposta ideal e a resposta real apresentadas pela rede aos pontos de treinamento.

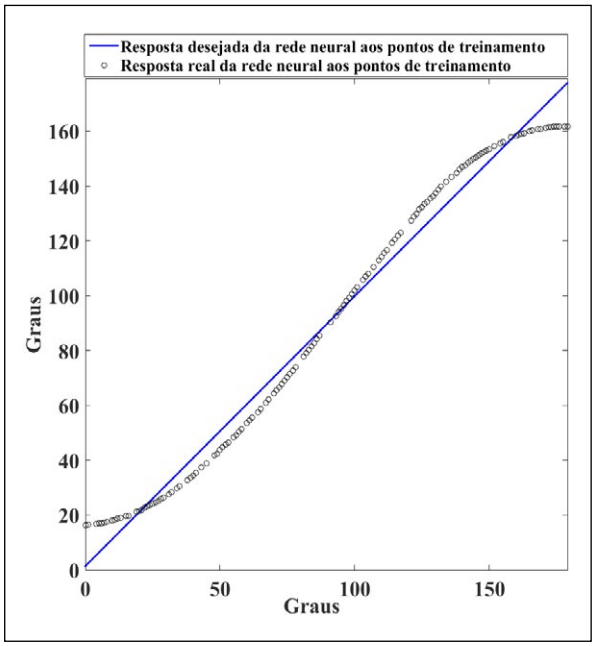
As Figuras 4 e 5 se referem aos resultados obtidos com uma rede neural com uma camada oculta com cinco neurônios e com função de ativação linear.

Figura 4 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com uma camada oculta com cinco neurônios e função de ativação linear em todas as camadas).



Fonte: Elaboração própria.

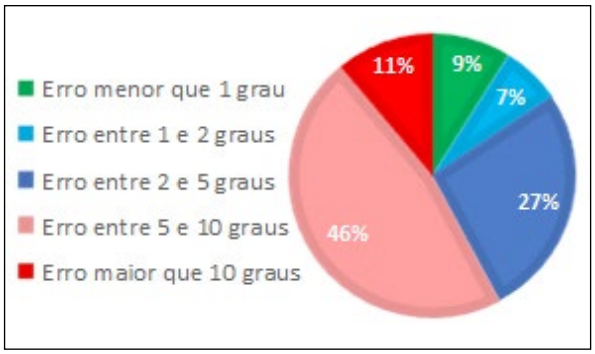
Figura 5 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com cinco neurônios e função de ativação linear em todas as camadas).



Fonte: Elaboração própria.

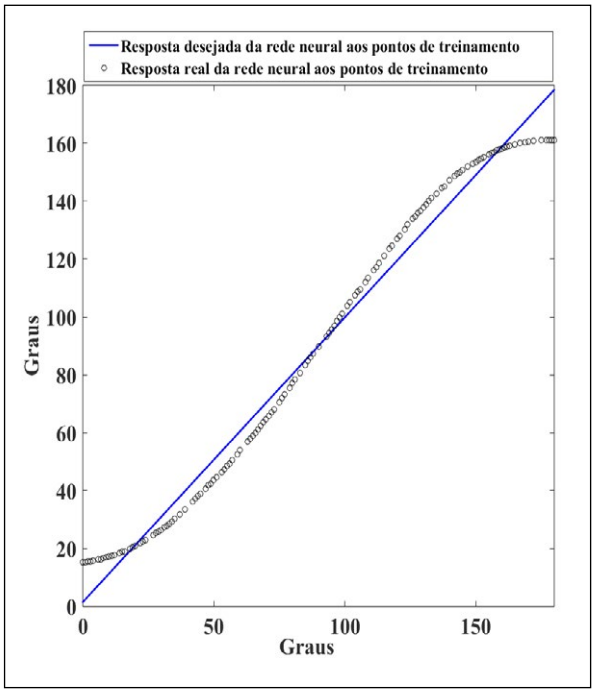
As Figuras 6 e 7 ilustram os resultados obtidos com uma rede neural com uma camada oculta com dez neurônios e com função de ativação linear.

Figura 6 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com uma camada oculta com dez neurônios e função de ativação linear em todas as camadas).



Fonte: Elaboração própria.

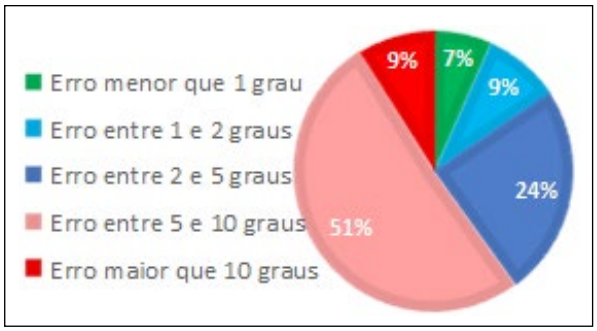
Figura 7 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com dez neurônios e função de ativação linear em todas as camadas).



Fonte: Elaboração própria.

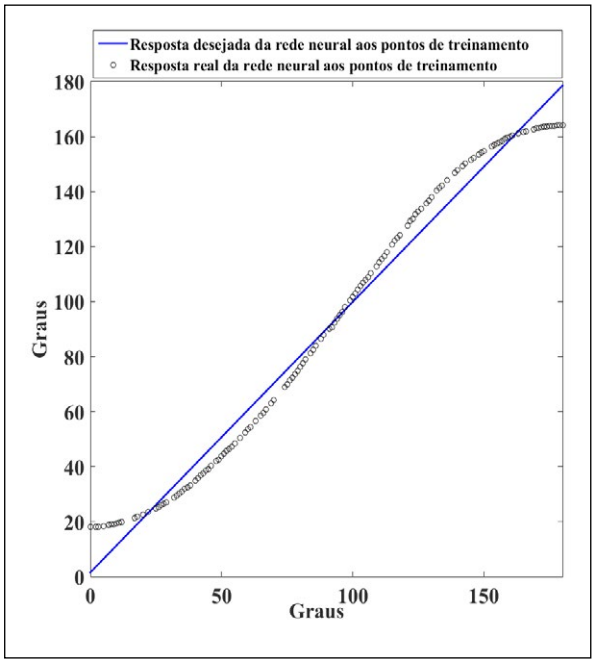
Nas Figuras 8 e 9 são ilustrados resultados obtidos com uma rede neural com uma camada oculta com 20 neurônios e com função de ativação linear.

Figura 8 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com uma camada oculta com vinte neurônios e função de ativação linear em todas as camadas).



Fonte: Elaboração própria.

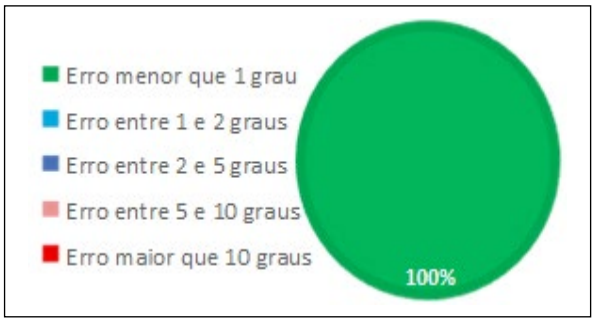
Figura 9– Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com vinte neurônios e função de ativação linear em todas as camadas).



Fonte: Elaboração própria.

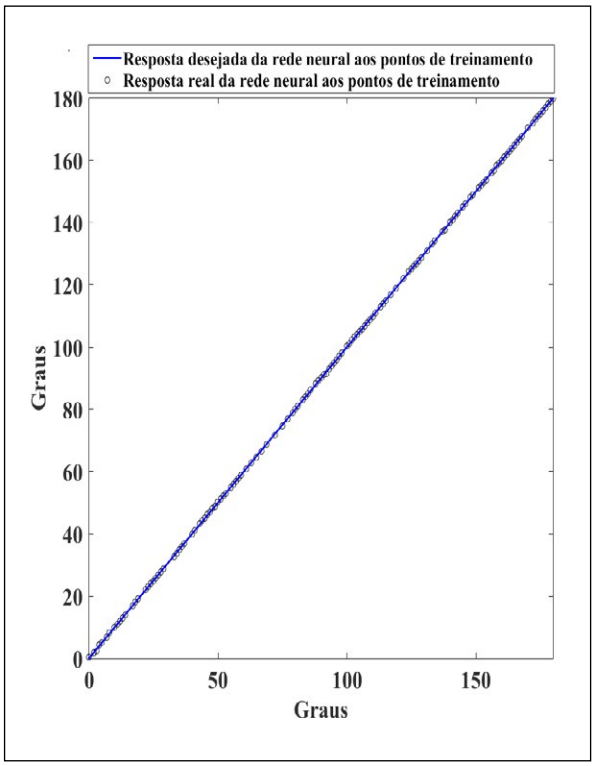
As Figuras 10 e 11 se referem aos resultados obtidos com uma rede neural com uma camada oculta com cinco neurônios e com função de ativação sigmoide.

Figura 10 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com uma camada oculta com cinco neurônios e função de ativação sigmoide na camada oculta).



Fonte: Elaboração própria.

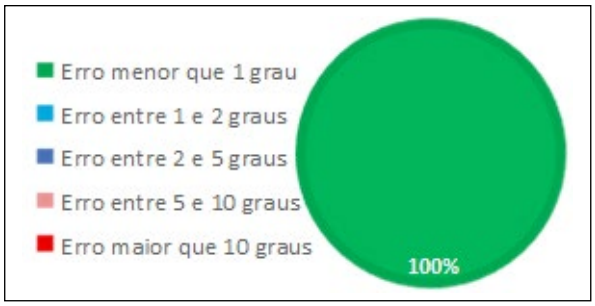
Figura 11 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com cinco neurônios e função de ativação sigmoide na camada oculta).



Fonte: Elaboração própria.

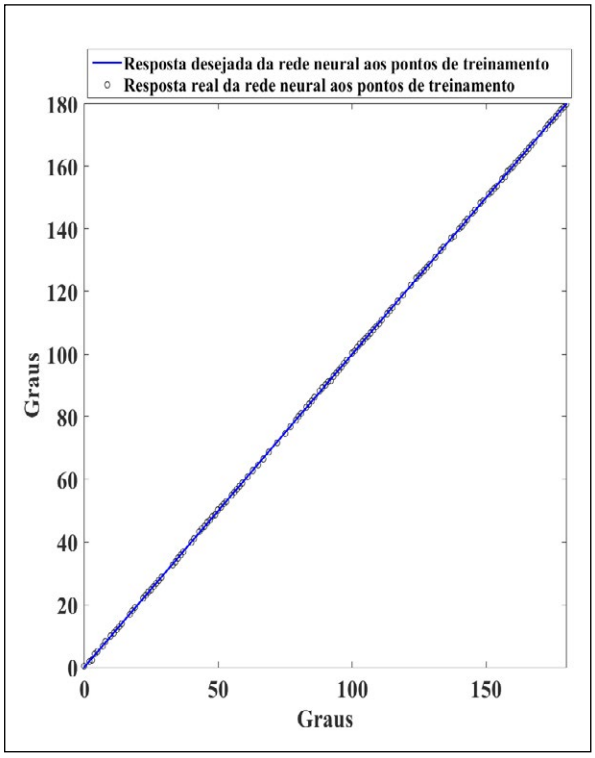
As Figuras 12 e 13 se referem aos resultados obtidos com uma rede neural com uma camada oculta com 10 neurônios com função de ativação sigmoide.

Figura 12 – Percentual de pontos de teste pertencentes a cada faixa de neural (rede com uma camada oculta com dez neurônios e função de ativação sigmoide na camada oculta).



Fonte: Elaboração própria.

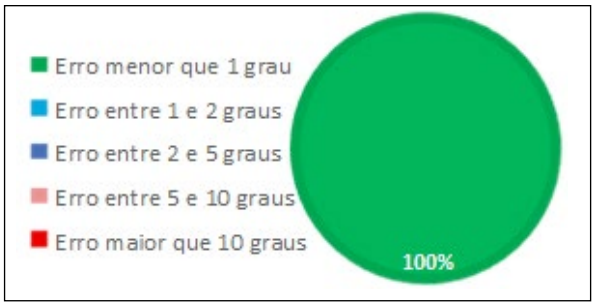
Figura 13 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com dez neurônios e função de ativação sigmoide na camada oculta).



Fonte: Elaboração própria.

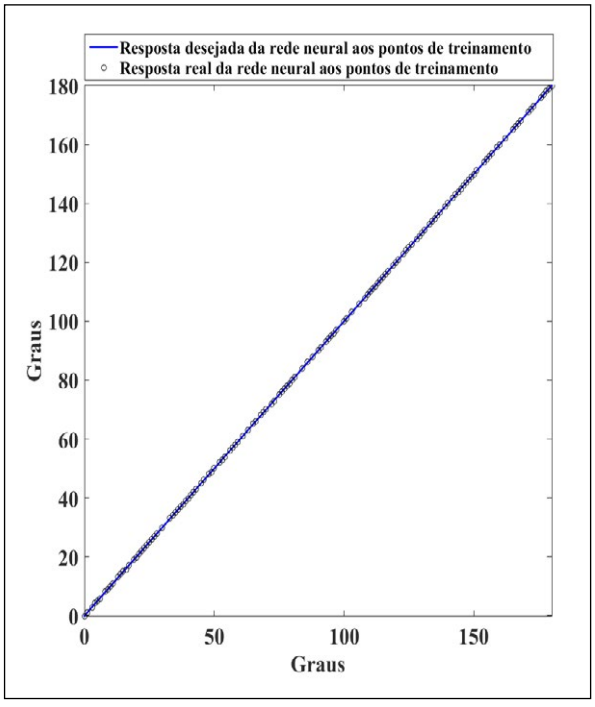
As Figuras 14 e 15 ilustram os resultados obtidos com uma rede neural com uma camada oculta com 20 neurônios com função de ativação sigmoide.

Figura 14 – Percentual de pontos de teste pertencentes a cada faixa de neural (rede com uma camada oculta com vinte neurônios e função de ativação sigmoide na camada oculta).



Fonte: Elaboração própria.

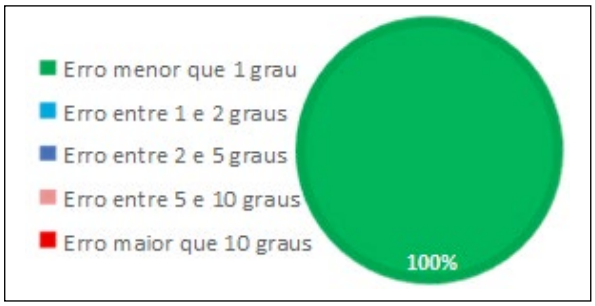
Figura 15 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com vinte neurônios e função de ativação sigmoide na camada oculta).



Fonte: Elaboração própria.

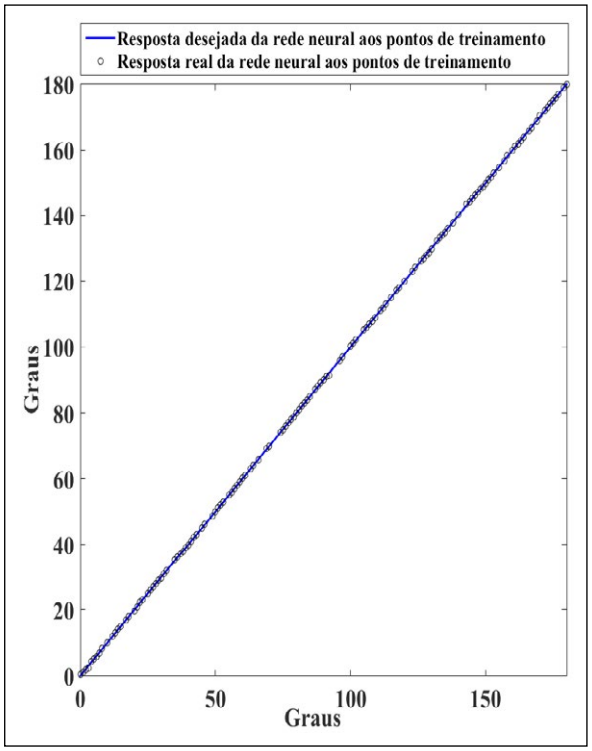
Nas Figuras 16 e 17 são ilustrados os resultados obtidos com uma rede neural com uma camada oculta com cinco neurônios com função de ativação tangente hiperbólica.

Figura 16 – Percentual de pontos de teste pertencentes a cada faixa de neural (rede com uma camada oculta com cinco neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

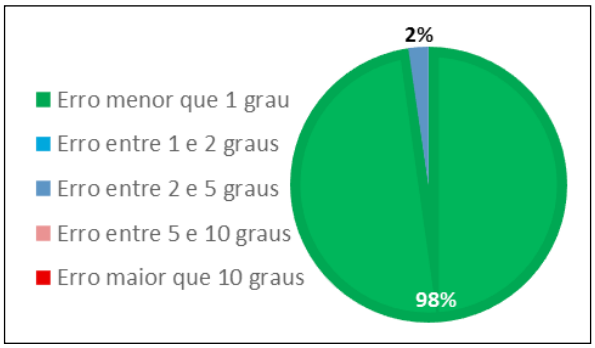
Figura 17 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com cinco neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

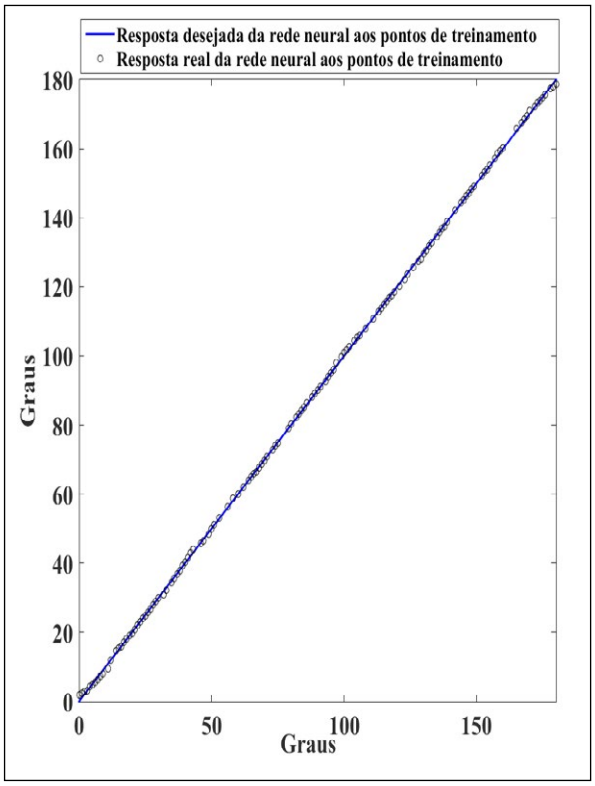
As Figuras 18 e 19 se referem aos resultados obtidos com uma rede neural com uma camada oculta com 10 neurônios com função de ativação tangente hiperbólica.

Figura 18 – Percentual de pontos de teste pertencentes a cada faixa de neural (rede com uma camada oculta com dez neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

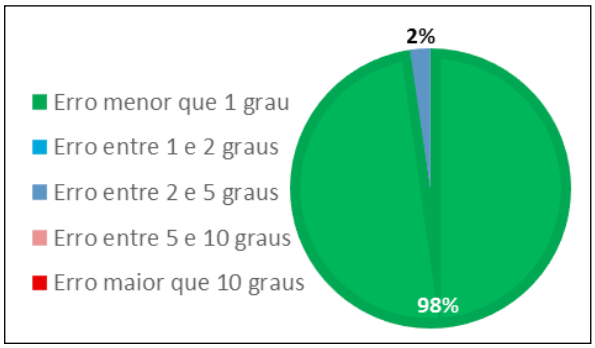
Figura 19 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com dez neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

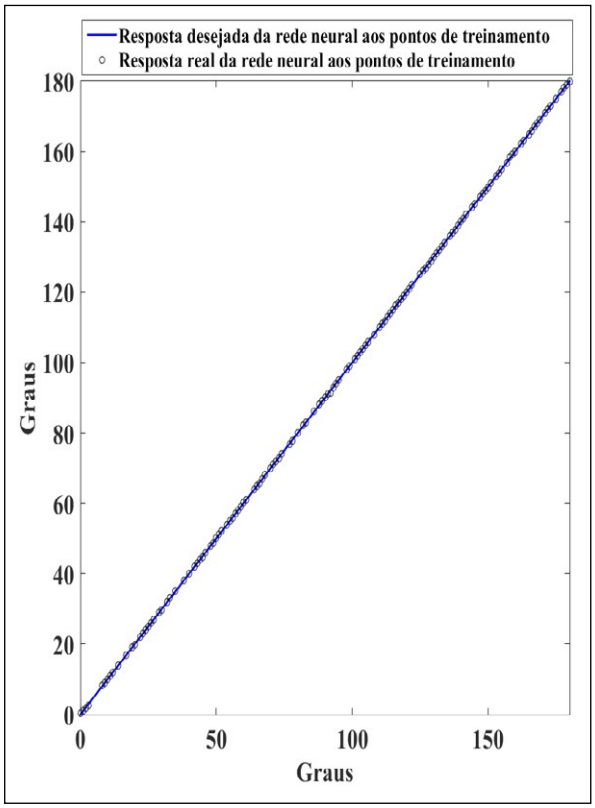
As Figuras 20 e 21 referem-se aos resultados obtidos com uma rede neural com uma camada oculta com 20 neurônios com função de ativação tangente hiperbólica.

Figura 20 – Percentual de pontos de teste pertencentes a cada faixa de neural (rede com uma camada oculta com vinte neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

Figura 21 – Comparação entre a saída desejada e a saída real da rede neural (rede com uma camada oculta com vinte neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

Com base nos resultados mostrados nesta seção, a utilização da função de ativação linear na camada oculta não apresentou resultados satisfatórios.

As redes com função de ativação tangente hiperbólica ou sigmoide apresentaram resultados quase idênticos e bastante satisfatórios, independentemente da quantidade de neurônios na camada oculta. Diante disso, pode-se afirmar que a quantidade ótima de neurônios na camada oculta, para o treinamento *offline*, é cinco neurônios, pois é a arquitetura com menor quantidade de neurônios na camada oculta que foi capaz de generalizar a função cinemática inversa do manipulador.

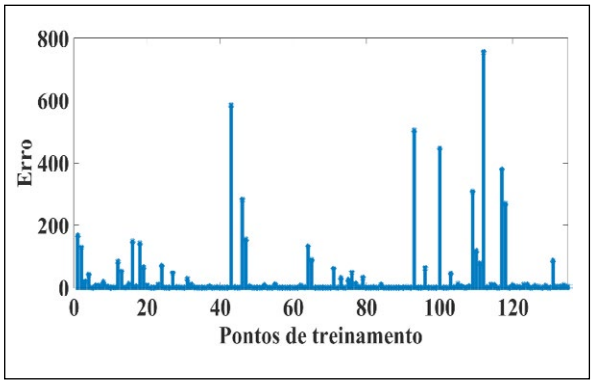
4.2 Treinamento *online*

O desempenho do processo de aprendizagem *online*, para cada arquitetura de rede neural testada, é ilustrado nesta subseção. As Figuras 22, 24, 26 e 28 ilustram o erro absoluto da resposta da rede neural para cada novo ponto de treinamento, obtido por meio do sensor MPU6050, antes de o ponto participar do processo de treinamento da rede. Caso a rede esteja conseguindo generalizar a função cinemática inversa do manipulador, o erro da resposta tenderá a zero, à medida que a quantidade de pontos de treinamento aumentar.

As Figuras 23, 25, 27 e 29 ilustram o percentual de pontos de teste que pertencem a cada faixa de erro estabelecida. As faixas de erro são: erro menor que 1 grau; erro entre 1 e 2 graus; erro entre 2 e 5 graus; erro entre 5 e 10 graus; erro maior que 10 graus.

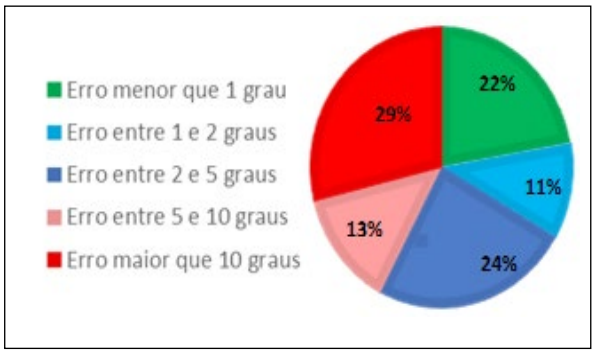
Nas Figuras 22 e 23, são ilustrados os resultados obtidos com uma rede neural com uma camada oculta com 20 neurônios e com função de ativação tangente hiperbólica.

Figura 22 – Erro absoluto associado a cada ponto de treinamento antes de participar do processo de treinamento (rede com uma camada oculta com vinte neurônios e com função de ativação tangente hiperbólica).



Fonte: Elaboração própria.

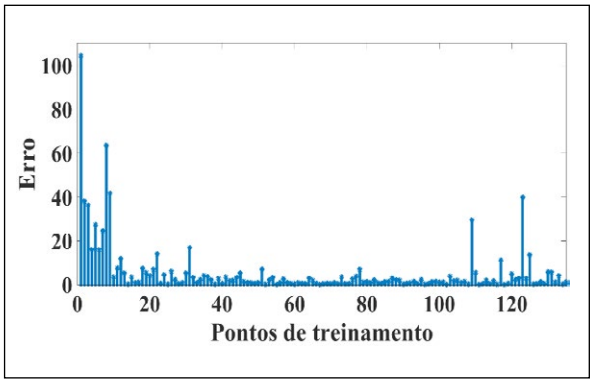
Figura 23 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com uma camada oculta com vinte neurônios e com função de ativação tangente hiperbólica).



Fonte: Elaboração própria.

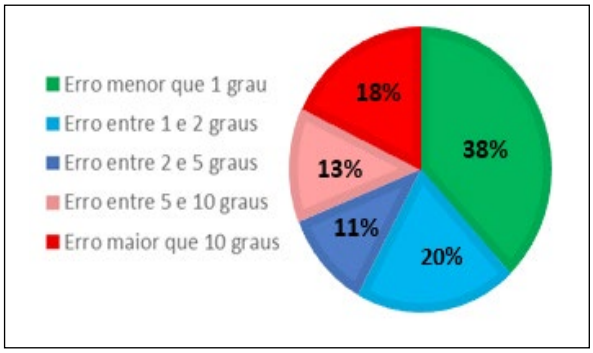
Nas Figuras 24 e 25 são ilustrados os resultados obtidos com uma rede neural com duas camadas ocultas com 20 neurônios em cada camada e função de ativação tangente hiperbólica.

Figura 24 – Erro absoluto associado a cada ponto de treinamento, antes de participar do processo de treinamento (rede com duas camadas ocultas com vinte neurônios em cada camada e função de ativação tangente hiperbólica).



Fonte: Elaboração própria.

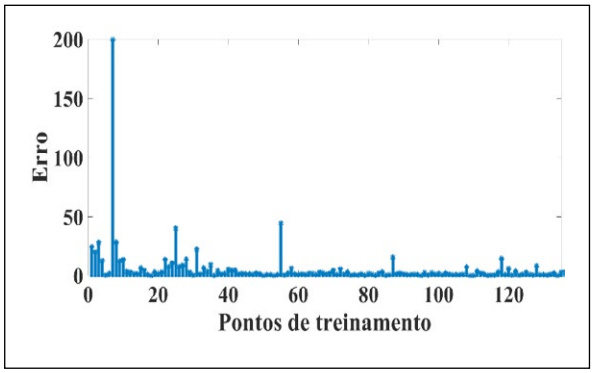
Figura 25 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com duas camadas ocultas com vinte neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

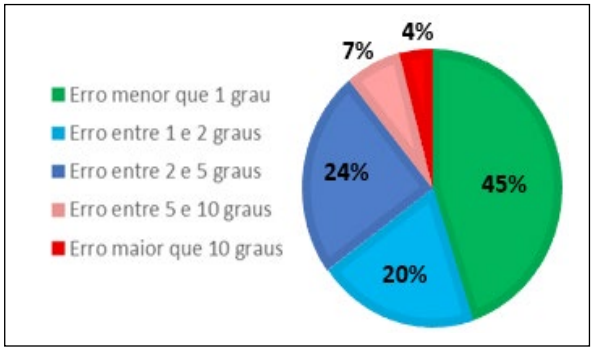
As Figuras 26 e 27 ilustram os resultados obtidos com uma rede neural com duas camadas ocultas com 30 neurônios em cada camada e função de ativação tangente hiperbólica.

Figura 26 – Erro absoluto associado a cada ponto de treinamento antes de participar do processo de treinamento (rede com duas camadas ocultas com trinta neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

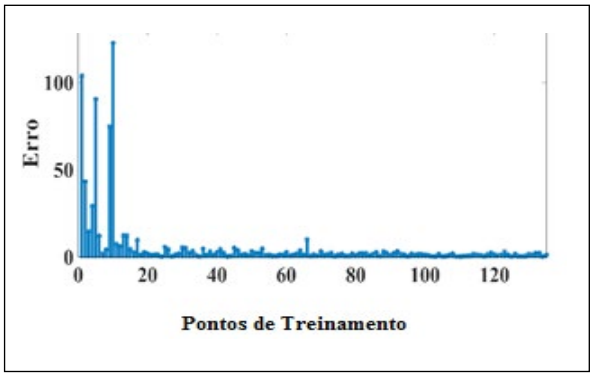
Figura 27 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com duas camadas ocultas com trinta neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

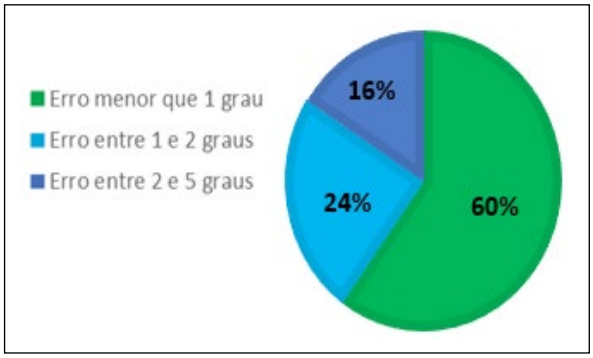
As Figuras 28 e 29 se referem aos resultados obtidos com uma rede neural com duas camadas ocultas com 50 neurônios em cada camada e função de ativação tangente hiperbólica.

Figura 28 – Erro absoluto associado a cada ponto de treinamento antes de participar do processo de treinamento (rede com duas camadas ocultas com cinquenta neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

Figura 29 – Percentual de pontos de teste pertencentes a cada faixa de erro (rede com duas camadas ocultas com cinquenta neurônios e função de ativação tangente hiperbólica na camada oculta).



Fonte: Elaboração própria.

No processo de treinamento *online*, a cada novo ponto de treinamento obtido, ocorre um novo processo de treinamento com o novo ponto e com os pontos antigos. Isso faz com que a quantidade de dados que a rede neural deve generalizar se torne bem maior do que no treinamento *offline*, sendo necessário redes neurais com mais de uma camada intermediária e uma grande quantidade de neurônios.

No processo de treinamento *online*, as redes com apenas uma camada oculta e com poucos neurônios perdem a capacidade de generalização da função e passam apenas a memorizar os pontos de treinamento, não conseguindo alcançar um bom desempenho para pontos que não participaram do processo de treinamento. O aumento da quantidade de neurônios nas camadas ocultas proporcionou uma melhora de desempenho no processo de aprendizagem da rede.

Porém, mesmo a rede com maior quantidade de neurônios testada (cinquenta neurônios em cada uma das duas camadas ocultas), não obteve resultados tão satisfatórios quando comparados com os resultados obtidos pelas redes com apenas uma camada oculta com cinco neurônios, treinadas pela metodologia *offline*. Apesar disso, o treinamento *online* possui a vantagem de poder ser realizado à medida que o robô manipulador realiza alguma tarefa designada para ele, como, por exemplo, uma operação de soldagem ou pintura. Em trabalhos futuros, pode-se empregar a metodologia *online* em um robô manipulador com mais do que um grau de liberdade.

5 Conclusões

A partir de todo o exposto, é conclui-se que este trabalho, na medida do possível e do que foi proposto, conseguiu atingir seu objetivo principal: de resolver o problema da cinemática inversa de um robô manipulador utilizando redes neurais artificiais. O processo de treinamento *online* mostrou que é necessária uma rede neural maior do que as utilizadas neste trabalho para obter resultados próximos dos resultados obtidos pelo treinamento *offline*, que obteve resultados muito satisfatórios.

Em trabalhos futuros, é necessário investigar a utilização de outras arquiteturas de rede para o processo de treinamento *online*, que, apesar dos elevados custos computacionais, permite que o robô seja treinado enquanto realiza tarefas designadas para ele, permitindo um processo de aprendizagem da cinemática inversa contínuo que faz com que a movimentação do robô seja continuamente aprimorada.

REFERÊNCIAS

CRAIG, J. J. **Robótica**. 3. ed. São Paulo: Pearson, 2012.

DAYA, B.; KHAWANDI, S.; CHAUVET, P. Neural network system for inverse kinematics problem in 3 DOF robotics. In: IEEE INTERNATIONAL CONFERENCE ON BIO-INSPIRED COMPUTING: THEORIES AND APPLICATIONS (BIC-TA), 5th., 2010, Changsha (China). **Proceedings...** Beijing: Institute of Electrical and Electronics Engineers, 2010. p. 1550-1557.

DENAVIT, J.; HARTENBERG, R. S. A kinematic notation for lower-pair mechanisms based on matrices. **Journal of Applied Mechanics**, v. 22, p. 215-221, 1955.

DUKA, A. V. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. **Procedia Technology**, v. 12, p. 20-27, 2014.

FACELI, K. et al. **Inteligência artificial**: uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2011.

HAYKIN, S. **Redes neurais**. 2. ed. Porto Alegre: Bookman, 2001.

JHA, P.; BISWAL, B. B. A neural network approach for inverse kinematic of a SCARA manipulator. **International Journal of Robotics and Automation**, v. 3, n. 1, p. 52-61, mar. 2014.

MONTGOMERY, E.; LUDWIG JR., O. **Redes neurais**: fundamentos e aplicações com programas em C. Rio de Janeiro: Ciência Moderna, 2007.

MUSTAFA, A.; TYAGI, C.; VERMA, N. K. Inverse kinematics evaluation for robotic manipulator using support vector regression and Kohonen self organizing map. In: INTERNATIONAL CONFERENCE INDUSTRIAL AND INFORMATION SYSTEMS (ICIIS), 11th, 2016, Roorkee (India). **Proceedings...** Piscataway, NJ: Institute of Electrical and Electronics Engineers, 2016. p. 375-380.

NIKU, S. B. **Introdução à robótica:** análise, controle, aplicações. 2. ed. Rio de Janeiro: LTC, 2014.

PERASSO, V. O que é a 4ª Revolução Industrial - e como ela deve afetar nossas vidas. **BBC Brasil**, 22 out. 2016. Disponível em: <https://www.bbc.com/portuguese/geral-37658309>. Acesso em: ago. 2017.

SHI, Q.; XIE J. A research on inverse kinematics solution of 6-DOF robot with offset-wrist based on Adaboost neural network. In: IEEE INTERNATIONAL CONFERENCE ON CYBERNETICS AND INTELLIGENT

SYSTEMS (CIS) and IEEE CONFERENCE ON ROBOTICS, AUTOMATION AND MECHATRONICS (RAM), 2017, Ningbo (China). **Proceedings....** Piscataway, NJ: Institute of Electrical and Electronics Engineers, 2017. p. 370-375.

VUOLO, J. H. **Fundamentos da teoria dos erros**. 2. ed. São Paulo: Edgar Blücher, 1996.

ZOU, Z.; HAN, J.; ZHOU, M. Research on the inverse kinematics solution of robot arm for watermelon picking. In: IEEE INFORMATION TECHNOLOGY, NETWORKING, ELECTRONIC AND AUTOMATION CONTROL CONFERENCE (ITNEC), 2nd, 2017, Chengdu (China). **Proceedings...** Piscataway, NJ: Institute of Electrical and Electronics Engineers, 2017. p. 1399-1402.