Sistema Inteligente para o Controle de Dispositivos Elétricos/Eletrônicos e Consumo de Energia Elétrica em Residências de Alto Padrão

Elton de Sousa e Silva * **Mauro Sérgio Silva Pinto** **
José Pinheiro de Moura *** **Paulo Fernandes da Silva Junior** ****

* *Pós-Graduação da Engenharia de Computação e Sistemas*
Universidade Estadual do Maranhão
e-mail: eltonss.eng@gmail.com.
** *Pós-Graduação da Engenharia de Computação e Sistemas*
Departamento da Engenharia de Computação
Universidade Estadual do Maranhão
e-mail: mauropinto@professor.uema.br
*** *Departamento da Engenharia de Computação*
Universidade Estadual do Maranhão,
e-mail: josepinheiro@professor.uema.br
**** *Pós-Graduação da Engenharia de Computação e Sistemas*
Universidade Estadual do Maranhão
e-mail: pfs1224@gmail.com

Abstract: The evolution of the Internet of Things (IoT) is positively impacting people's lives. With the increasing advancement of technology, homes are being automated with electronic devices, which are available to be programmed with intelligent software, to be functional, keeping people comfortable and reducing costs with electricity consumption . This work presents a proposal for a software based on artificial neural networks (ANN) to manage the devices in a home, using an automatic system to minimize the consumption of electrical energy with the least impact on the comfort of its residents.

Resumo: A evolução da *Internet* das Coisas (*Internet of Things* - IoT), está impactando positivamente a vida das pessoas. Com o avanço da tecnologia, as residências estão sendo automatizadas com dispositivos elétricos/eletrônicos, que estão disponíveis para serem programados com *softwares* inteligentes, para serem funcionais, mantendo o conforto das pessoas e reduzindo custos com o consumo de energia elétrica. Neste trabalho, apresenta-se uma proposta de um *software* com base em redes neurais artificiais (RNA) e Aprendizado por Reforço (AR) concebido na plataforma *Google Colaboratory* para gerenciar os dispositivos de uma residência, utilizando um sistema chamado AutoDomo (com os modos de operação automático e manual), que pode minizar o consumo de energia elétrica sem impactar, negativamente, no conforto dos seu moradores.

Keywords: Internet of Things; Advancement of technology; Electronic devices; Electricity; Artificial neural networks.

Palavras-chaves: Internet das Coisas; Avanço da tecnologia; Dispositivos eletrônicos; Energia elétrica; Redes neurais artificiais.

1. INTRODUÇÃO

O sistema AutoDomo foi desenvolvido priorizando o conforto de uma maneira simples e amigável para o usuário. O qual, o monitoramento, as informações de acessos e alarmes são facilmente integrados a outros dispositivos, buscando facilitar as atividades rotineiras de casas que podem não ser tão simples para pessoas com alguma deficiência. O sistema é configurado de tal forma que tem condições de capturar o perfil de consumo da residência. Ainda, pode-se obter informações sobre objetos através de dados coletados por instrumentos e assim demonstrar de forma gráfica e fácil visualização.

O foco deste trabalho é o desenvolvimento de um Sistema de Gerenciamento de Energia Elétrica via a plataforma AutoDomo chamado de SIGEAD com base em uma Rede Neural Artificial (RNA), utilizando a técnica de Aprendizagem por Reforço em Lote (ARL) (Lacerda, 2013), para minimizar os gastos desnecessários com o consumo de energia elétrica com base em dados históricos do sistema AutoDomo e que possa ser acoplada ao mesmo sistema com a finalidade de realizar ações de desligamento de cargas desnecessárias. Ainda neste contexto, preocupa-se com o consumo racional e eficiente de energia elétrica.

O sistema proposto neste artigo, além de permitir o controle manual remotamente, de ligar/desligar os dispositivos elétricos/eletrônicos, reduzindo o consumo de energia elétrica por uso desnecessário ainda, faz o desligando de tomadas, doas equipamentos que não estão sendo usados, garantindo assim, o desligamento da energia de *stand-by*.

1.1 Caracterização e Proposta de Solução do Problema

Uma residência de alto padrão é definido conforme a norma NBR 12721, sendo composta de no mínimo quatro dormitório, sendo uma suíte (com banheiro e *closet*), outro com banheiro, banheiro social, sala de estar, sala de jantar e sala íntima, circulação, cozinha, área de serviço completa e varanda com abrigo para automóvel com area real minima de $224,82m^2$. Devido as características de um imóvel de alto padrão, há dificuldades de se controlar a eficiência energética sem que haja o desconforto para os moradores e não é comum o uso de métodos de controle, utilizando-se de técnicas de Inteligência Computacional (IC) e bio-inspiradas sem modelos matemáticos sofisticados (Moura, 2019). Em residências de alto padrão, normalmente, existem sistemas elétricos automatizados, para o conforto de seus moradores e para evitar o consumo desnecessário de energia elétrica de aparelhos. Propõe-se neste artigo um método baseado em uma RNA, desenvolvido em *Pyton*, na plataforma do *Google colaboratory* para atuar com base no comportamento dos moradores, desligando dispositivos elétricos/eletrônicos que não são utilizados.

1.2 Organização do Artigo

O artigo está organizado da seguinte forma: Na Seção 2, faz-se uma breve descrição das RNAs, com suas principais características e métodos usados neste trabalho. Na Seção 3 Apresenta-se a plataforma AutoDomo e seu funcionamento. Na Seção 4, é apresentado o SIGEAD, os experimentos e os resultados alcançados e por fim, Na Seção 5, apresenta-se a conclusão do artigo.

2. REDES NEURAIS ARTIFICIAIS - RNA

As RNAs são ferramentas poderosas para modelar processos multivariáveis complexos, está técnica é mais usada no campo computacional (Haykin, 1994). Porém, com a proliferação rápida e bem sucedida de aplicações que incorporam tecnologia de RNA em campos tão diversos como comércio, ciência, indústria e medicina oferece uma garantia e quebra do paradigma das RNAs.

2.1 Características das RNAs

As RNAs têm três características importantes que sustentam esse sucesso: i) a primeira é a maneira comparativamente direta em que as RNAs adquirem informações/conhecimentos sobre um determinado domínio complexo por meio de uma etapa de treinamento (Ding, 2013); ii) a segunda característica é a forma compacta (embora completamente numérica) na qual a informação/conhecimento adquirida é armazenada dentro da RNA treinada e a facilidade de velocidades comparativas com que esse conhecimento pode ser acessado e usado e a iii) terceira característica é a robustez da solução com base em uma RNA na presença de ruído nos dados de entradas da planta. Além dessas características, uma das vantagens mais importantes das RNAs treinadas é o alto grau de precisão relatado quando uma solução RNA é usada (Andrews, 1995).

2.2 Aprendizado por Reforço

O aprendizado automático explora o estudo e construção de algoritmos que podem aprender de seus erros e fazer previsões sobre dados (Kohavi, 1998). Tais algoritmos operam construindo um modelo a partir de entradas amostrais a fim de fazer previsões ou decisões guiadas pelos dados, em vez, de simplesmente, seguir instruções programadas (Qazi, 2021).

O AR pode utilizar duas técnicas diferentes: i) aprendizado supervisionado, que treina um modelo com base em dados conhecidos tanto de entrada quanto de saída e desta forma consegue prever resultados futuros a partir de novos dados de entrada e ii) aprendizado não supervisionado, que encontra padrões escondidos ou estruturas inerentes aos próprios dados. No Algoritmo 1 está ilustrado os passos genéricos do AR.

ALGORITMO 1: ALGORITMO DE AR

Saída: Q
i$\leftarrow 0$
while $i < k$ do
  foreach $t\epsilon[1, \dots |D|]$ do
    $T_i \leftarrow \gamma \; max_a \; Q_{i-1} \; (S_{i+1} \; , a)$
  end
  $Q_i \leftarrow Q_0$;

while $i = k \quad do$
  foreach $t\epsilon[1, \dots |D|] \; do$
    $Q_i(S_t, a_t) \leftarrow Q_i(S_t, a_t) + \alpha_{supervisionado}(T_i - Q_i(S_t, a_t))$
  end
end

2.3 Políticas

Políticas são padrões de ação tomados pelo agente. Em outros termos, são funções que, quando providas do estado atual e uma ação escolhida, retornam a probabilidade de se tomar aquela ação. Podem ser definidas matematicamente como uma distribuição de probabilidades das ações a serem tomadas para todos os estados e é definida por

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]. \tag{1}$$

Na Eq. (1), onde lê-se "probabilidade de se tomar a ação a estando no estado s.

2.4 Princípio da Otimalidade de Bellman

No princípio da otimalidade de *Bellman*, diz-se:

> "Uma estratégia ótima apresenta a propriedade segundo a qual, a despeito das decisões tomadas para se atingir um estado particular num certo estágio, as decisões restantes a partir deste estado devem constituir uma estratégia ótima".
>
> [(Bellman, 1966)]

A equação de *Bellman*, é uma condição necessária para a otimização associada ao método de otimização matemática conhecido como programação dinâmica. Onde, escreve o "valor"de um problema de decisão em um determinado ponto no tempo em termos do retorno de algumas escolhas iniciais e o "valor"do problema de decisão restante que resulta dessas escolhas iniciais. Conforme mostra-se na equação abaixo.

$$V(s) = max_a(R(s, a) + \gamma V(s')). \tag{2}$$

$V(s)$ é o retorno esperado (valor) no estado atual s, max_a é o valor máximo de qualquer ação a possível, $R(s, a)$ é a

recompensa esperada de retorno para ação a no estado s e o $\gamma V(s')$ é o fator de desconto multiplicado pelo valor do próximo estado. A equação de *Bellman* decompõe a função de valor em duas partes, sendo: i) Parte 1 - $max_a(R(s, a)$ a recompensa imediata e ii) Parte 2 - $\gamma V(s'))$, são os valores futuros com o fator de desconto.

3. PLATAFORMA AUTODOMO

A plataforma AutoDomo é um sistema que integra sensores e acionadores inteligentes em uma plataforma única e plenamente compatível com os assistentes de voz usados em dispositivos eletrônico. Todos os equipamentos conectados em um dispositivo ficam automaticamente disponíveis e acessíveis na *internet* para serem controlados por meios de $Smartphones$ e $Iphones$.

3.1 Funcionamento da Platoforma AutoDomo

A plataforma AutoDomo utiliza o protocolo *Message Queuing Telemetry Transport*(MQTT) (Sanjuan, 2020), baseado na *Internet* das Coisas (Wortmann, 2015), que funciona baseado no protocolo TCP/IP (Forouzan, 2002). Na Figura 1, apresenta-se a topologia do sistema AutoDomo.

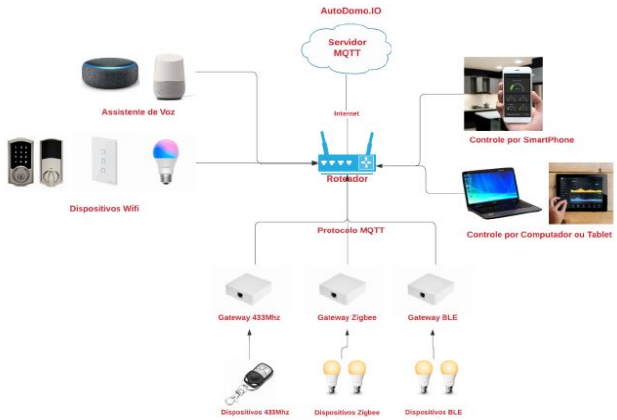

Figura 1. Topologia do sistema AutoDomo.
Fonte: Autor

O Sistema de automação residencial da plataforma AutoDomo, permite controlar e configurar os dispositivos de uma residência de forma rápida e fácil. A configuração do sistema é feito em alguns minutos, onde todos os dispositivos elétricos/eletrônicos do ambientes da residência podem ser conectados em um *Smartphone*. E ainda permite o controle dos objetos da residência de forma remota. A AutoDomo não restringe o número de usuários do sistema por residência. Todos podem utilizar simultaneamente com a devida permissão do proprietário e é compatível com *IOS* e *android*.

3.2 Padrão Message Queuing Telemetry Transport

Nos anos 90 a IBM criou o protocolo MQTT (Chen, 2014). Sua origem se deu à necessidade de um protocolo simples e leve que conseguisse comunicar várias máquinas entre si, uma comunicação que ocorreria utilizando microcontroladores para a obtenção de dados que tivesse uma taxa de transmissão leve para a comunicação entre as máquinas e os sensores. O protocolo MQTT se popularizou pela simplicidade, baixo consumo de dados e pela possibilidade de comunicação bilateral.

4. SIGEAD

O SIGEAD, foi desenvolvidos com dados coletados de uma residência de alto padrão na cidade de São Luís-MA. Este foi concebido na plataforma AutoDomo, com base na RNA da base de dados do *Google Colaboratory* para a tomada de decisão de acordo com o perfil comportamental de seus moradores.

Quando o SIGEAD está ativado, funciona como um usuário do sistema. Assim, pode desligar os dispositivos com a finalidade de economizar energia. Entretanto, para evitar algum tipo de transtorno e para aumentar o controle do usuário, a qualquer momento, o sistema pode ser desativado por meio de um botoeira de comando no próprio aplicativo disponibilizado para o usuário controlar a sua residência no modo manual. Na Figura 2, mostra-se o controle do SIGEAD com o comando de seleção de modo autmático/manual destacado por um circulo em vermelho.

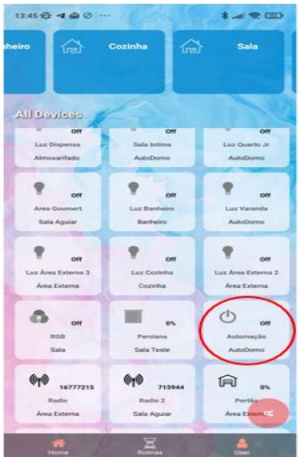

Figura 2. Controle do SIGEAD.

4.1 Experimentos

Nesta Seção apresenta-se os experimentos do sistema SIGEAD desenvolvidos no *Google Colaboratory* e os resultados alcançados.

4.2 Google Colaboratory

O *Google Colaboratory*, chamado de *Colab*, é um serviço de nuvem gratuito hospedado pelo próprio *Google* para incentivar a pesquisa de Aprendizado de Máquina e IA.

O *Google Colaboratory* é uma ferramenta que permite a fusão de código fonte (geralmente em *python*) e texto rico (geralmente em *markdown*) com imagens e resultados. É uma técnica conhecida como *notebook* (“caderno”) disponível em um ambiente colaborativo, que pode ser compartilhado, permitindo que outros executem o código e até modifiquem criando novas versões.

As principais características do *Google Colaboratory* são: i) como é executada em uma máquina do *google*, não precisa realizar qualquer configuração; ii) o *google* disponibiliza gratuitamente acesso a *Graphics Processing Units* (GPU) e iii) é fácil de ser compartilhada.

4.3 Redes Neurais com TensorFlow

As RNAs são modelos computacionais inspirados pelo sistema nervoso central (em particular o cérebro) que são capazes de realizar o aprendizado de máquina bem como o reconhecimento de padrões. RNAs, geralmente, são apresentadas como sistemas de "neurônios interconectados, que podem computar valores de entradas", simulando o comportamento de redes neurais biológicas.

Em 2015, a *Google* criou a biblioteca *TensorFlow* para criação e treinamento de RNAs. A *Application Programming Interface* (API), é escrita em *Python*. Porém, é executada em C++ na *Central Processing Units* (CPU) ou em *Compute Unified Device Architecture* (CUDA) no GPU. O Keras é uma biblioteca de código aberto para fácil criação e treinamento de RNAs, projetado para permitir experimentação rápida com redes neurais profundas (RNP), ele se concentra em ser fácil de usar, modular e extensível. Em 2017, a equipe do *TensorFlow* decidiu apoiar o Keras como sua biblioteca principal.

4.4 Função de Ativação

As funções de ativação são um elemento extremamente importante das RNAs. Elas basicamente decidem se um neurônio deve ser ativado ou não. Isto é, se a informação que o neurônio está recebendo é relevante para a informação fornecida ou deve ser ignorada. A função de ativação é mais uma camada matemática no processamento e é dada por

$$Y = A\left(\sum (W \times E) + bias\right). \tag{3}$$

A é ativação, W o peso (varia de 0 a 1) e E a entrada. A função de ativação é uma camada matemática no processamento da RNA e $bias$ são pequenas alterações sofridas pelos pesos W.

Na equação de *Bellman*, Eq. (4), o agente atualiza o valor atual percebido com a recompensa futura ótima estimada, assumindo que o agente executa a melhor ação conhecida no momento. Em uma implementação, o agente pesquisará todas as ações de um determinado estado e escolherá o par estado-ação com o valor da função de valor correspondente mais alto. Na Figura 3, apresenta-se a estrutura esquemática do agente de ARP.

$$Q(S_t, A_t) = (1-\alpha)Q(S_t, A_t) + \alpha \times (R_t + \lambda \times max_a Q(S_{t+1}, a)). \tag{4}$$

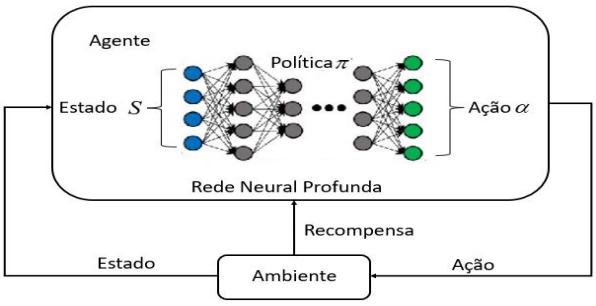

Figura 3. Estrutura esquemática do agente de ARP.
Fonte: Autor

A função de ativação é a transformação não linear que se faz ao longo do sinal de entrada. Esta saída transformada é enviada para a próxima camada de neurônios como entrada. Quando não se tem a função de ativação, os pesos e bias fazem uma transformação linear.

Neste trabalho, usa-se a função de ativação do tipo sigmóide para ativação da RNA , que é dada por

$$F(x) = 1/(1 + e^{-x}). \tag{5}$$

O código em *Python* da função sigmoid apresentada na Eq. (5) é dado por

```
import math

def sigmoid(x):
    sig = 1 / (1 + math.exp(-x))
    return sig
```

Para a implementação numericamente estável da função sigmóide, primeiro precisa-se verificar cada valor do *array* de entrada e em seguida, passar o valor do sigmóide. Para isso, pode-se usar o método *np.where*( ), mostrado abaixo.

```
import numpy as np

def stable_sigmoid(x):

    sig = np.where(x < 0, np.exp(x)/(1 + np.exp(x)), 1/(1 + np.exp(-x)))
    return sig
```

Para ativação das camadas ocultas, usou-se a função de ativação *Rectified Linear Unit* (ReLU), que é dada por

$$ReLU(x) = max\{0,\, x\} \tag{6}$$

$$ReLRU' = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{caso contrário.} \end{cases} \tag{7}$$

A ReLU é conhecida por ser uma função de ativação de baixo custo computacional, que apesar da simplicidade das operações realizadas obtém bons resultados e é amplamente utilizada (Nwankpa, 2018).

4.5 Estrutura do SIGEAD

O SIGEAD é dividido, basicamente, em três etapas, que são: i) Aquisição de dados (*Firebase*); ii) Treinamento da RNA (Colab) e iii) Operação do sistema autodomo. Na Figura 4, apresenta-se o diagrama de blocos do SIGEAD.

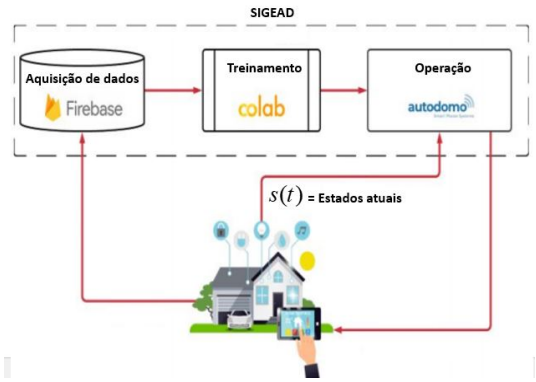

Figura 4. Diagrama de blocos do SIGEAD.
Fonte: Autor

Na etapa **aquisição de dados**, todos os eventos (liga dispositivo, desliga dispositivo) que usuário realiza no SIGEAD é armazenado em um banco de dados em nuvem, utilizando-se o banco de dados *Firebase Real-Time Database* da *Google*. Os dados ficam armazenados em forma de um arquivo *JavaScript Object Notation* (JSON), que é necessário para o correto treinamento do SIGEAD. isto é, ajuda a reduzir os principais problemas encontrados nos dados brutos do sistema, tais como: i) Grande quantidade de valores desconhecidos; ii) Ruídos (atributos com valores incorretos); iii) Atributo de baixo valor preditivo e iv) Desproporção entre número de exemplo de cada classe.

O pré-processamento dos dados desenvolvido na linguagem *Python* é executada no *Google Colaboratory*, com manipulação dos dados via biblioteca "Pandas".

Na etapa de **treinamento** o sistema utiliza os dados para simular ou aprender as rotinas dos moradores da residência por meio de AR. Vale apenas ressaltar, que como o agente se utiliza de dados históricos não se pode esperar que se consiga encontrar uma política ótima. Dessa forma, o objetivo não é mais aprender a política ótima, mas sim a melhor política possível com os dados fornecidos (Lacerda, 2013). Todavia, a aplicação desse método em problemas reais pode, algumas vezes, ser inviável.

Na etapa **operação** o agente é interligado ao SIGEAD com autonomia para enviar comandos (ações) ao sistema, mantendo somente, os dispositivos necessários ligados, na busca da redução no consumo de energia elétrica. Os comandos são enviados utilizando o protocolo MQTT o mesmo protocolo utilizado pelo usuário da residencia, isto é, o agente treinamento passa a ter a mesma autonomia de um usuário comum.

4.6 Arquiteutura da RNA do SIGEAD

A arquitetura da RNA do SIGEAD é apresentada na Figura 5. Onde, pode-se observar que é constituída da seguinte forma: i) A camada de entrada com 10 neurônios; ii) A segunda camada com 42 neurônios; iii) A terceira camada com 64 neurônios; iv) A quarta camada com 128 neurônios e v) a camada de saída com 2 neurônios. Essa arquitetura é gerada pelo algoritmo de ARL, que é similar ao AR não supervisionado, depois de 1000 episódios (iterações), com base na equação de *Belman* associada com a Eq. (2).

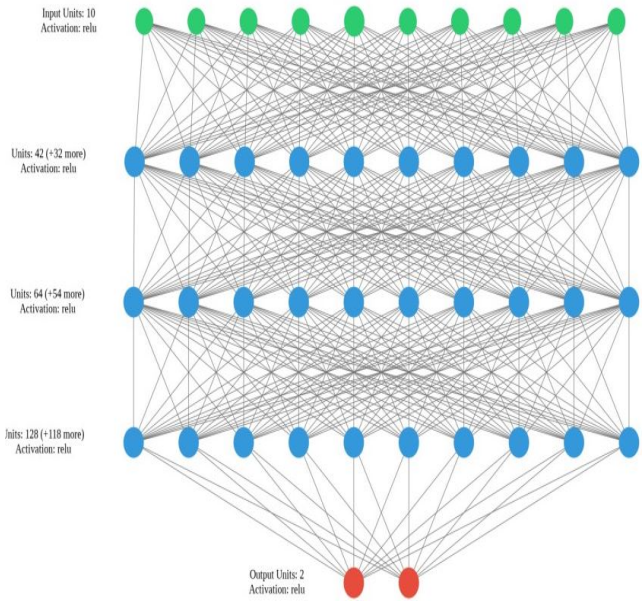

Figura 5. Arquitetura da RNA do SIGEAD.
Fonte: Autor

No Algoritmo 2, mostra-se o passo a passo do Algoritmo de ARL.

ALGORITOMO 2: ALGORITMO DE ARL

Saída: Q
$Q_i \leftarrow Q_0$;
while Q não convergiu *do*
  $D \leftarrow$ *do*
  $i \leftarrow 0$
while experiência $< m$ *do*
  $i \leftarrow i + 1$;
  $S_i \leftarrow$ EstadoDoAmbiente( );
  experiência $\leftarrow$ experiência+1;
  $a_i \leftarrow$ selecionaoAao(Q, S_i);
  executarAcao $\leftarrow (a_i)$;
  $r_i \leftarrow$ recompensadoAmbiente ( );
  $S_{i+1} \leftarrow$ estadodoAmbiente ( );
  $d_i \leftarrow (S_i, a_i, r_i, S_{i+1})$;
  $D \leftarrow D \bigcup d_i$;
end
  $D \leftarrow D$
end

4.7 Algoritmo do SIGEAD Desenvolvido em Pyton

O Algoritmo do SIGEAD, foi dividido em quatro partes para melhor entendimento do seu funcionamento na plataforma *Google Colaboratory*.

Algoritmo SIGEAD - Parte 1: Na Figura 6, apresenta-se a parte 1 do Algoritmo SIGEAD. Nesta, define-se os parâmetros iniciais da classe construtora da RNA associados com equação de Bellman dada Na Eq. 2. Na Linha 1 *state_size* = Quantidade de estados vindo do ambiente (residência a ser controlada). Na linha 2 a*ction_space* = a quantidade de ações (saida da RNA) Ligar/Desligar dispositivos. Na Linha 4 *gamma* = equivale a o parâmetro γ da equação de Bellman (Eq. 2 que indica o fator de desconto e nas linhas 6,7 e 8 epsilon = os parâmetros epsilon serve para definir a probabilidade de uma ação ser aleatória ou executada pela RNA.

```
1     self.state_size = state_size
2     self.action_space = action_space
3     self.model_name = model_name
4     self.gamma = 0.96
5     self.epsilon = 1.0
6     self.epsilon_final = 0.01
7     self.epsilon_decay = 0.995
8     self.model = self.model_builder()
```

Figura 6. Algoritmo SIGEAD - Parte 1.
Fonte: Autor

No inicio do treinamento, tem-se o valor de epsilon = 1 (100%). Pois, a ação executada pelo agente é aleatória, tendo em vista, que os pesos ainda não foram treinados, suficiente para tomar as devidas ações. Porém, com o avanço do treinamento o epsilon pode decair até 0,01 (1%) dessa forma existe uma probabilidade de 99% da ação ser executada com a saída da RNA. Pode-se observar, que mesmo após uma RNA treinada, existe uma possibilidade mínima de executar uma ação aleatória não prevista na RNA treinada, isto tem a importância de evitar que o agente fique preso em um mínimo local.

Algoritmo SIGEAD - Parte 2: Na Figura 7, apresenta-se a parte 2 do Algoritmo SIGEAD. Nesta, é definida a função *model_builder* na linha 1. Na linha 2 o *tf,keras.models.Sequential* que cria a estrutura da RNA utilizando a biblioteca Keras no *TensonFlow* representada pela Figura 5. Na linha 3 o *TensorFlow* fornece a implementação do modelo sequencial com tk.keras. Na linha 4, é criada a primeira camada oculta com 42 neurônios. Na linha 5 a segunda camada oculta com 64 neurônios e na linha 6, a terceira camada oculta com 128 neurônios. Nas camadas ocultas, foi atribuído o atributo relu nos parâmetros de ativação. Pois, a função *Rectified Linear Unit* (ReLU) como função de ativação das camadas ocultas. Na linha 7, a camada de saída com 2 neurônios. Na linha 8, calcula a quantidade que o modelo procura para minimizar durante o treinamento e na linha 9, retorna para a saída.

```
1  def model_builder(self):
2    model = tf.keras.models.Sequential()
3    model.add(tf.keras.Input(shape=(self.state_size,)))
4    model.add(tf.keras.layers.Dense(units = 42, activation = "relu"))
5    model.add(tf.keras.layers.Dense(units = 64, activation = "relu"))
6    model.add(tf.keras.layers.Dense(units = 128, activation = "relu"))
7    model.add(tf.keras.layers.Dense(units = self.action_space, activation = "linear"))
8    model.compile(loss = "mse", optimizer = tf.keras.optimizers.Adam(lr = 0.001))
9    return model
```

Figura 7. Algoritmo SIGEAD - Parte 2.
Fonte: Autor

4.8 Algoritmo SIGEAD - Parte 3:

Na Figura 8 é definido a função *action* na linha 1, essa função recebe o parametro *state* (estados) vindo do ambiente e executa as (ações) das decisões a serem tomadas. Na linha 2, inicia com uma ação. Na linha 3, executa a saída (*return*) que pode ser aleatoria quando *randon.random*() for menor que epsilon. Isso, geralmente, acontece no inicio do treinamento da RNA. Na linha 5, mostra as ações (*actions*) da RNA executada pelo comando "*actions=self.model.predict(state)*"depois do treinamento da RNA e na linha 6, esecuta o resultado (a saída) da RNA.

```
1  def action(self, state):
2    if random.random() <= self.epsilon:
3      return random.randrange(self.action_space)
4
5    actions = self.model.predict(state)
6    return np.argmax(actions[0])
```

Figura 8. Algoritmo SIGEAD - Parte 3.
Fonte: Autor

4.9 Algoritmo SIGEAD - Parte 4:

Por fim, na Figura 9, mostra-se a execução da função treinamentoLote, que recebe um lote de dados vindo do banco de dados *Firebase* para realizar o treinamento da RNA. Essa função percorre todo o banco de dados buscando todos os estados da RNA.

```
1  def treinamentoLote(self, tamanhoLote):
2      lote = []
3      for i in range(len(self.memory) - tamanhoLote + 1, len(self.memory)):
4        lote.append(self.memory[i])
5
6      for state, action, reward, next_state, done in lote:
7        if not done:
8          reward = reward + self.gamma * np.amax(self.model.predict(next_state)[0])
9
10       target = self.model.predict(state)
11       target[0][action] = reward
12
13       self.model.fit(state, target, epochs=1, verbose=0)
14
15     if self.epsilon > self.epsilon_final:
16       self.epsilon *= self.epsilon_decay
```

Figura 9. Algoritmo SIGEAD - Parte 4.
Fonte: Autor

4.10 Resultados Alcançados

O SIGEAD foi implantado em outubro de 2021 sem habilitação para tomar decisões, somente, para estimar os dados, caso os dispositivos fossem desligados com base na RNP projetada e apresentada neste trabalho. Na Figura 10, apresenta-se o resultado dos dados estimados pelo SIGEAD em comparação com os dados reais no mês de outubro de 2021. Como pode se observar, que o desempenho estimado foi melhor do que o desempenho real em termo de consumo de energia elétrica, obtendo uma redução estimada de 3,19%. Com base nos dados estimados do SIGEAD, o proprietário da residência em estudo autorizou a habilitação do SIGEAD a partir do mês de novembro de 2021.

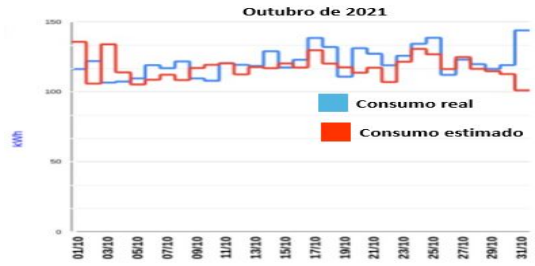

Figura 10. Consumo estimado x consumo real.
Fonte: Autor

Avaliação de Desempenho do SIGEAD: Para avaliar o desempenho do SIGEAD, fez-se uma comparação do sistema, nos meses de novembro e dezembro de 2020, período em que o sistema, ainda não estava implantado, comparado com os meses

de novembro e dezembro de 2021, depois da implantação do mesmo.

Em novembro de 2020, o consumo de energia elétrica foi de 3761 kWh e em novembro de 2021, o consumo foi de 3592 kWh, obtendo-se um redução de 4,52%. conforme Figura 11.

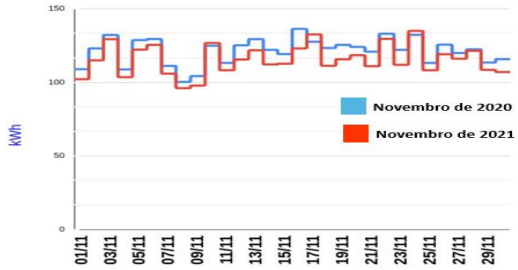

Figura 11. Consumo de novembro 2021 x novembro 2022. Fonte: Autor

Na Figura 12, apresenta-se o consumo de energia elétrica em dezembro de 2020, que foi de 3694 kWh e em dezembro de 2021, o consumo foi de 3502 kWh, reduzindo-se assim o consumo em 5,19% com o SIGEAD.

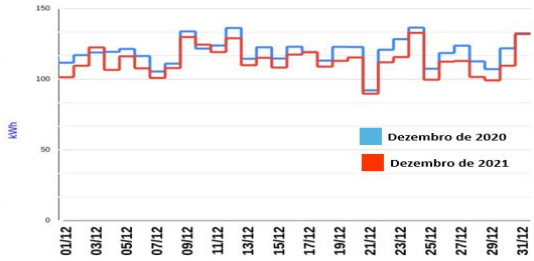

Figura 12. Consumo de dezembro 2021 x dezembro 2022. Fonte: Autor

5. CONCLUSÃO

O SIGEAD, obteve bom desempenho nos meses avaliados, conforme os resultados alcançados. Portanto, o SIGEAD é uma solução viável, que além de fazer o controle de ligar e desligar dispositivos elétricos/eletrônicos de uma residência, ainda atua, na redução de consumo de energia elétrica, desligando a energia de *stand-by* por meio do desligamento das tomadas dos respectivos dispositivos elétricos/eletrôniocos desligados, assegurando maior proteção por queima em função de oscilação da tensão na rede elétrica.

AGRADECIMENTOS

Agradecemos ao Programa de Pós-Graduação em Engenharia de Computação e Sistemas (PECS) e ao Departamento de Engenharia de Computação (DECOMP) da Universidade Estadual do Maranhão (UEMA) por tornar essa pesquisa possível e reconhecemos à CAPES e ao CNPq por promoverem e apoiarem pesquisas e estudos avançados que contribuíram para a realização deste trabalho. Também, agradecemos em especial, à FAPEMA por incentivar e apoiar à pesquisa de alto nível no Estado do Maranhão.

REFERÊNCIAS

Andrews, Robert. Diederich, J..T.A.B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6), 373–389.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731), 34–37.

Chen, Whei-Jen. Gupta, R.L.V.R.D.M.S.N..o. (2014). *Responsive mobile user experience using MQTT and IBM MessageSight*. IBM Redbooks.

Ding, Shifei. Li, H.S.C.Y.J..J.F. (2013). Evolutionary artificial neural networks: a review. *Artificial Intelligence Review*, 39(3), 251–260.

Forouzan, B.A. (2002). *TCP/IP protocol suite*. McGraw-Hill Higher Education.

Haykin, S. (1994). Intelligent signal processing. 1–12.

Kohavi, R. (1998). Glossary of terms. *Special issue on applications of machine learning and the knowledge discovery process*, 30(271), 127–132.

Lacerda, D.A. (2013). *Aprendizado por reforço em lote: um estudo de caso para o problema de tomada de decisão em processos de venda*. Ph.D. thesis, Universidade de São Paulo.

Moura, José Pinheiro. da Fonseca Neto, J.V..R.P.H.M. (2019). A neuro-fuzzy model for online optimal tuning of pid controllers in industrial system applications to the mining sector. *IEEE Transactions on Fuzzy Systems*, 28(8), 1864–1877.

Nwankpa, Chigozie. Ijomah, W.G.A..M.S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

Qazi, Atika. Hardaker, G.A.I.S.D.M.M.J.Z..D.A. (2021). The role of information & communication technology in elearning environments: A systematic review. *IEEE Access*, 9, 45539–45551.

Sanjuan, Eduardo Buetas. Cardiel, I.A.C.J.A..C.C. (2020). Message queuing telemetry transport (mqtt) security: A cryptographic smart card approach. *IEEE Access*, 8, 115051–115062.

Wortmann, Felix & Flüchter, K. (2015). Internet of things. *Business & Information Systems Engineering*, 57(3), 221–224.