UNIVERSIDADE FEDERAL DE CAMPINA GRANDE - UFCG

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA - CEEI

DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEE

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

APRENDIZAGEM PROFUNDA APLICADO A PREDIÇÃO DE CIRCUITOS RLC

Professor: Edson Porto da Silva

Alunos: Ygor de Almeida Pereira - Matrícula: 121110166

CAMPINA GRANDE - PARAÍBA

JANEIRO- 2024

LISTA DE FIGURAS

SUMÁRIO

1. OBJETIVO

O presente trabalho tem como objetivo predizer os valores de resistência, indutância e capacitância de um circuito RLC série a partir do sinal de resposta da corrente elétrica em função do tempo. Para tal, serão utilizadas técnicas de treinamento de aprendizado profundo, além é claro do conhecimento da teoria de circuitos elétricos.

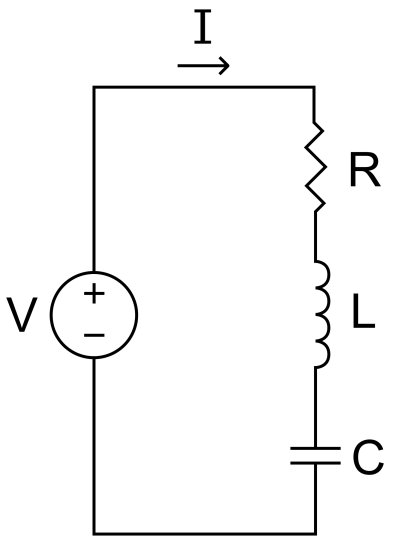

Figura 1 - Circuito RLC-série.

Como consequência do trabalho, comprovamos a possibilidade de criar uma ferramenta que realize esse tipo de previsão, podendo assim ser expandida para casos ainda mais gerais, contendo outros tipos de elementos e que possa ser útil em outros estudos e desenvolvimentos.

2. DESENVOLVIMENTO

2.1 Gerando os dados

Em primeiro lugar para realizar um treinamento em aprendizagem profunda devemos coletar uma quantidade suficiente de recursos e rótulos, ou melhor, entradas e saídas, as entradas serão as formas de onda da corrente em relação ao tempo e as saídas serão os valores de resistência, indutância e capacitância correspondente ao circuito que gerou tal onda de corrente. Esses tipos de recursos são essenciais para que o computador consiga treinar, aprender e diferenciar os variados tipos de entradas para suas correspondentes saídas.

Então, para conseguir esse conjunto de dados, foi utilizado como base um simulador de circuitos-RLC-série feito na linguagem python pelo professor Edson Porto e disponível em seu github. Além disso, o simulador sofreu algumas alterações para cumprir melhor o objetivo dessa pesquisa, as simulações são feitas utilizando uma fonte de tensão de onda quadrada com amplitude de 15 volts e frequência de 1 hertz, o código em python da simulação também foi alterado para gerar apenas o gráfico da corrente elétrica, removendo as legendas dos gráficos, pois essas não são necessárias para o treinamento, e salvando cada gráfico simulado como uma imagem no formato JPG.

No que diz respeito à geração de valores para os componentes dos circuitos simulados, foi utilizado uma distribuição uniforme. Para indutância e capacitância o intervalo de valores gerados foi de 0.01 a 0.1 e para os de resistência o intervalo escolhido foi de 1 a 20. No fim dessa etapa, foram simulados 6000 circuitos, ou seja, temos 6000 imagens de ondas de corrente que são nossas entradas e outras 6000 saídas que correspondem aos valores de RLC que ofertam tal onda. Uma informação importante é que o sinal de corrente elétrica resultante do circuito RLC pode ser classificado em três tipos: Superamortecido, subamortecido e criticamente amortecido. Portanto os circuitos gerados foram divididos para originar igualmente os três tipos de onda.

A imagem a seguir é uma das muitas obtidas, onde no eixo horizontal consta o tempo em segundos e no eixo vertical o valor da corrente elétrica em amperes.

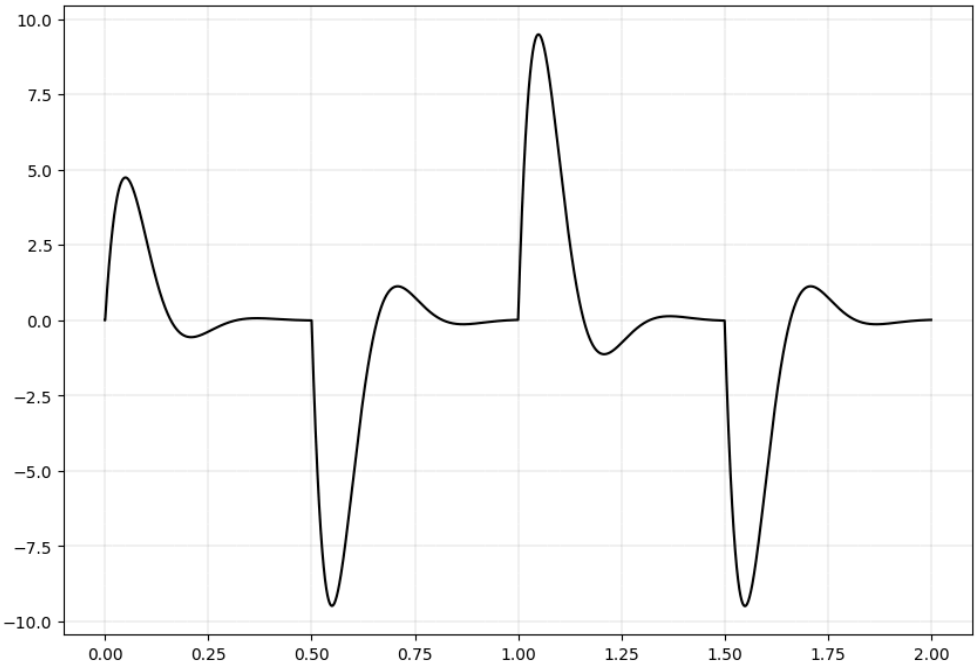

Figura 2 - Um dos sinais de corrente obtidos na simulação.

Podemos observar essa relação de recursos e rótulos por meio de uma matriz, no entanto é importante destacar, que uma imagem não é um recurso de uma única coluna, como está na tabela, uma imagem por si só é uma matriz de pixels.

Recursos	Rótulos		
Imagem	Resistência	Indutância	Capacitância
img_01	valor_resit_01	valor_indut_01	valor_cap_01
img_02	valor_resit_02	valor_indut_02	valor_cap_02
img_03	valor_resit_03	valor_indut_03	valor_cap_03
…	…	…	…
img_6000	valor_resit_6000	valor_indut_6000	valor_cap_6000

2.2 Organizando os dados

É necessário analisar e avaliar os dados obtidos antes de iniciar a etapa do treinamento, para isso observamos os valores máximos, mínimos e médios que obtemos e julgamos se eles estão dentro do enquadramento que prescrevemos. Também embaralhamos o conjunto de dados e separamos 5000 entradas e saídas para treinamento e as outras 1000 para teste, esse ato é significante, pois não faz sentido testar o modelo treinado com os mesmos dados que foram utilizados para sua rotina de aprendizado.

2.3 Preparando os dados para o treinamento

Anteriormente foi citado que as imagens são matrizes de pixels, na verdade as imagens que abordamos aqui não são matrizes comuns, elas são matrizes multidimensionais, é como ter uma matriz atrás da outra, formando o espaço tridimensional.

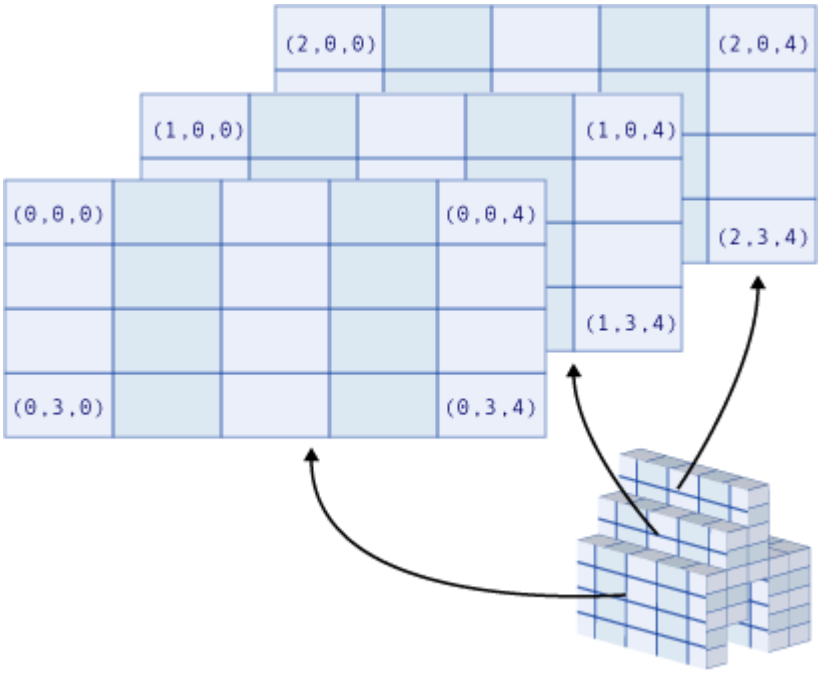

Figura 3 - Exemplo de uma matriz multidimensional.

Nesse caso, nossas imagens no formato JPG têm 3 dimensões referentes aos canais de cor *RGB (red, green, blue)*, podemos ver isso como três matrizes comuns empilhadas uma atrás da outra. No entanto, nossas imagens não são ricas em cores,

por isso simplificamos e ficamos apenas com um dos canais de cor, assim voltando para o caso de uma simples matriz unidimensional.

Matrizes grandes tornam o treinamento mais custoso, para evitar problemas redimensionamos as cada imagem para um tamanho 200x200 pixels, que pode ser vista como uma matriz de 200 linhas e 200 colunas. Após as simplificações realizadas nossas imagens agora se apresentam da seguinte forma:

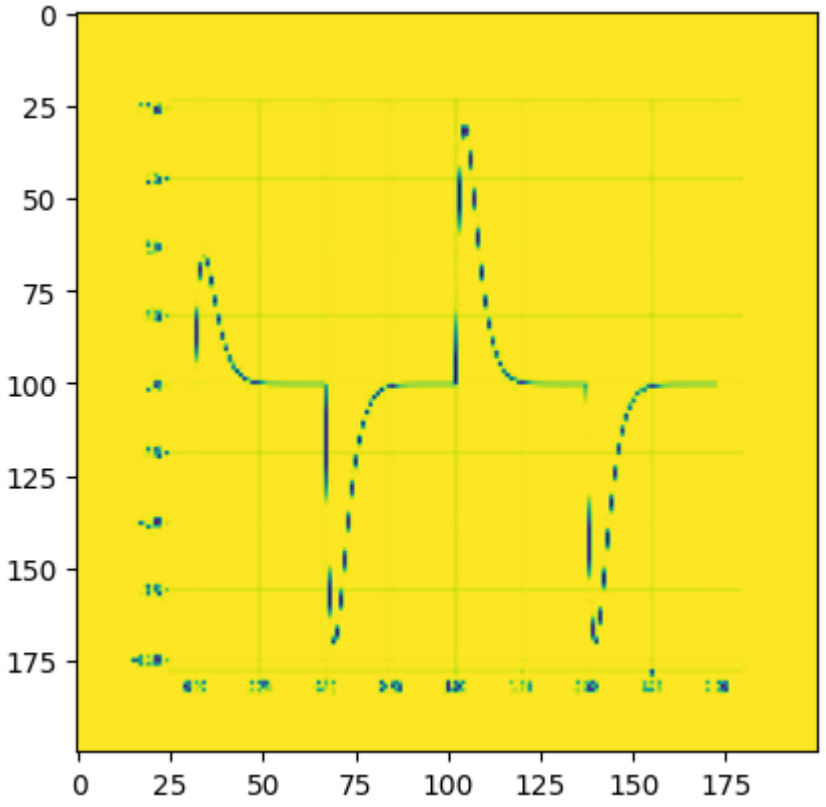

Figura 4 - Imagem após as simplificações realizadas.

2.4 Realizando o treinamento da rede neural

A partir desse ponto nos focamos na biblioteca *Tensorflow* e como realizamos o treinamento da rede neural, para isso criamos uma arquitetura de aprendizado, apresentamos aqui aquela que se saiu melhor perante os treinamentos e testes realizados. Como estamos tratando de entradas que correspondem a imagens se torna muito interessante usarmos Redes Neurais Convolucionais RNC, a ideia por trás desse tipo de rede neural é que as imagens sejam filtradas antes do modelo ser treinado,

assim os recursos, isto é, alguns conjuntos de pixels, mais importantes seriam escolhidos a partir de filtros.

Após a realização do treinamento pode-se evidenciar sua eficiência a partir do teste do conjunto de dados separado para tal e de parâmetros como *loss*, o parâmetro *loss* é uma métrica utilizada para medir o quão bem um modelo está performando em relação aos dados de treinamento.

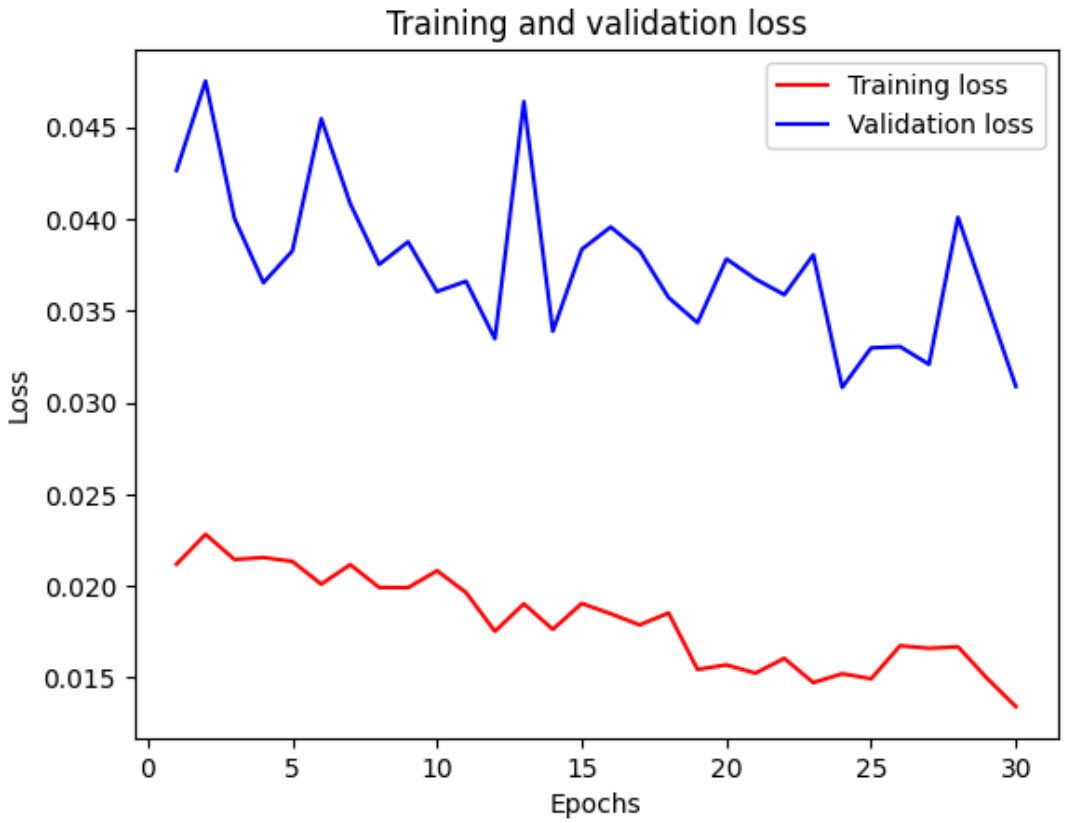

Figura 5 - Gráfico Loss x Epochs.

A partir da figura, observamos que conforme o número de *Epochs* aumenta, ou seja quanto mais execuções de treinamento são feitas, menor é o se torna o parâmetro *loss,* isso é um bom indicativo a respeito do treinamento e da validação no conjunto de teste, representado pela cor azul. Nesse caso, não continuamos com mais execuções do treinamento pois foi observado que continuar treinando não agregaria mais resultados significados.

3. CONCLUSÃO

Após a finalização do treinamento e quando utilizado sob o conjunto de teste obtemos o valor de *loss* de: 0.03087. Além disso ao comparar no conjunto de teste os valores de RLC que o modelo prevê para as ondas de correntes e os reais, podemos observar que estão consideravelmente próximos.

Valores reais

```
[[1.90032660e+01 7.12613641e-02 7.53084094e-02]
 [4.61209190e+00 1.95299650e-02 3.14193635e-02]
 [1.43450865e+00 2.49705049e-02 4.85363564e-02]
 [1.11398215e+00 6.38473009e-02 5.56996300e-02]
 [2.29133298e+00 7.82588425e-02 5.04336568e-02]
 [2.26566240e+00 7.15026229e-02 8.98421522e-02]
 [1.29015306e+00 8.09218333e-02 6.89083076e-02]
 [3.24606439e+00 9.96215205e-02 3.78190176e-02]
 [2.47735425e+00 7.08226381e-02 4.61587123e-02]
 [1.73208276e+00 6.87077311e-02 9.15950200e-02]
 [1.64637648e+01 7.67574642e-02 2.10150081e-02]
 [1.66268958e+00 4.96959667e-02 4.25027183e-02]
 [1.56004891e+01 8.70636084e-02 2.95158249e-02]
 [1.04352837e+01 9.98571140e-02 6.43045946e-02]
 [1.43613473e+00 5.35986178e-02 7.98517435e-02]
 [2.00026750e+00 8.94037861e-02 2.96758213e-02]
 [1.24967293e+01 9.37402714e-02 2.13394533e-02]
 [2.25571551e+00 5.90974272e-02 4.64608555e-02]
 [2.38572915e+00 5.84947586e-02 2.68217508e-02]]
```

Valores previstos

```
[[ 1.89570503e+01  6.84422180e-02  5.60546219e-02]
 [ 5.56042624e+00  3.49041149e-02  2.30156276e-02]
 [ 1.39723647e+00  3.56826112e-02  4.77709882e-02]
 [ 1.08322239e+00  8.18660781e-02  5.25643341e-02]
 [ 2.24022889e+00  8.95325169e-02  4.91834991e-02]
 [ 2.42230320e+00  9.16162059e-02  7.20890239e-02]
 [ 1.26926887e+00  9.27657411e-02  6.98827580e-02]
 [ 3.29081464e+00  1.07645117e-01  2.80398224e-02]
 [ 2.54884911e+00  8.06013271e-02  4.22958843e-02]
 [ 1.73899651e+00  8.55294690e-02  8.44867900e-02]
 [ 1.64881802e+01  8.00074935e-02 -2.73086643e-03]
 [ 1.60714817e+00  6.46444187e-02  3.65315191e-02]
 [ 1.55619116e+01  6.35327846e-02  1.29295681e-02]
 [ 1.06325617e+01  9.46064442e-02  6.11912124e-02]
 [ 1.43131113e+00  6.79458901e-02  7.95450285e-02]
 [ 2.01242995e+00  1.04648881e-01  3.08750961e-02]
 [ 9.31951618e+00  7.84537494e-02  1.36093590e-02]
 [ 2.28498030e+00  6.99520931e-02  4.50691096e-02]
 [ 2.37991595e+00  6.96660802e-02  2.14777086e-02]]
```

Figura 6 - Comparação dos valores reais e previstos.

Ademais, também podemos plotar os gráficos das ondas armazenadas no conjunto de teste e as geradas pelos valores de RLC que o modelo previu.

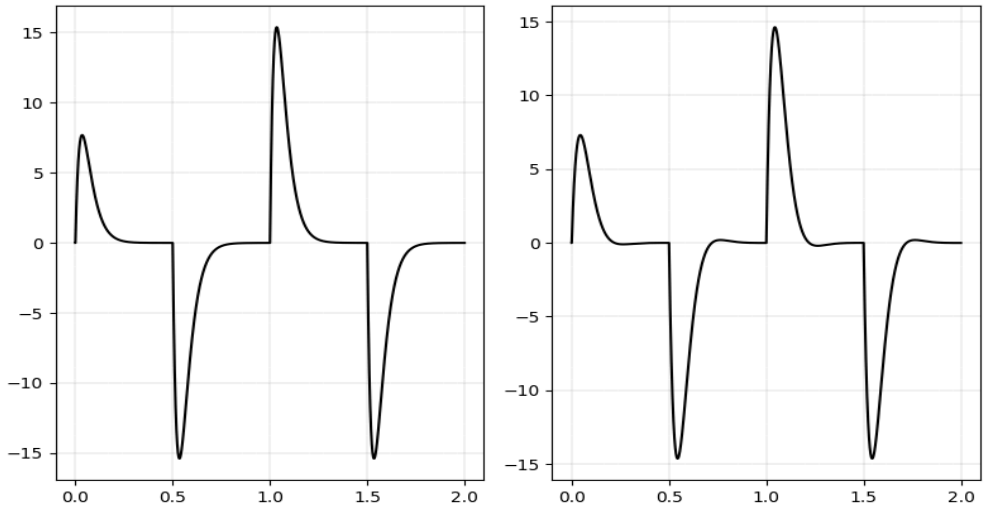

Figura 7 - Comparação dos gráficos gerados pelos valores reais e previstos.

Em alguns detalhes é perceptível que o modelo possui variações nas previsões e para alguns determinados valores ele pode ter maiores enganos, contudo existem manobras para contornar tais problemas e torná-lo mais eficiente uma delas e talvez a principal é realizar o treinamento com um conjunto maior de dados, como dez ou cem vezes maior que o utilizado neste trabalho. E depois foi utilizado um intervalo curto na geração de valores dos parâmetros dos circuitos, aumentar esses intervalos e alcançar um treinamento satisfatório também exige maior conjunto de dados. Vale ressaltar que essas melhorias exigem mais tempo de treinamento e poder computacional.

Por fim, podemos avaliar nosso objetivo principal e concluir que é possível criar um modelo de aprendizado profundo que preveja valores de resistência, indutância e capacitância de um circuito RLC série a partir do sinal de resposta da corrente elétrica em função do tempo.

5. REFERÊNCIAS

Código em python do modelo usado no trabalho. Ygor de Almeida Pereira. Disponível em https://github.com/ygordealmeida/Modelo_simulador. Acesso em 8 de janeiro de 2024.

Simulador de um circuito RLC-série. Edson Porto da Silva. Disponível em https://github.com/edsonportosilva/ElectricCircuits/tree/master/RLC%20circuits. Acesso em 8 de janeiro de 2024.