Trabalho Final - Reconhecimento e digitalização de algarismos analógicos de hidrômetros utilizando Redes Neurais Convolucionais (CNN)

Jean Wellington de Souza
Ytalo Ysmaicon Gomes

***Resumo*— Com o poder crescente dos processadores modernos, os sistemas estão se aproximando do usuário final – o que geralmente é chamado de computação de borda, em que o processamento não é mais realizado externamente ou na nuvem por poderosos dispositivos, e sim mais próximo do processo e/ou do usuário por microcontroladores de baixo custo e baixo consumo de energia imersos no paradigma de TinyML. Neste projeto a computação em borda é trazida para um exemplo prático. Com o objetivo de implementar um modelo de reconhecimento de algarismos em hidrômetros, foi desenvolvido um modelo e um protótipo para testes. Ao final, apesar de algumas limitações e obstáculos, os resultados expressaram que a aplicação tem potencial tecnológico para aplicações do cotidiano e para a indústria.**

I. Partes Envolvidas

Projeto desenvolvido pelos alunos Jean Wellington de Souza e Ytalo Ysmaicon Gomes submetido ao prof. Marcelo Rovai, como requisito parcial para aprovação na disciplina de TinyML - Aprendizado de Máquina para Dispositivos IoT Embarcados (IESTI01 2022.1) da Universidade Federal de Itajubá - UNIFEI.

II. Introdução

O termo *Indústria 4.0*, originado de um projeto estratégico de alta tecnologia do governo alemão em 2011, trouxe um novo conceito às indústrias - este conceito se baseia em promover a informatização da manufatura e realizar a integração de dados através de novas tecnologias com emprego de *Internet of Things* (IoT) e *Inteligência Artificial* (IA). [1]

Inerente ao contexto das fábricas, montadoras e produtoras atuais, sempre houve o questionamento se realmente valeria a pena trocar todo seu maquinário e equipamentos para se adequar ao conceito de Indústria 4.0, pois, no final das contas, estamos falando de muitos processos, muitas vezes extremamente caros, que demandariam um tempo considerável e custos para ser substituídos e implementados.

Nesta realidade, surge uma alternativa baseado em realizar alterações e adaptações nas máquinas que já funcionam, incorporando a elas sensores inteligentes que coletem e trate os dados de maneira local de forma menos evasiva possível. Assim, um tipo de adequação possível, por exemplo, é através do emprego de câmeras visando realizar a leitura de algarismos mecânicos de sistemas de controles já empregados às máquinas mas que não possuem integração digital.[2]

III. Fundamentação teórica

A. Sensor Mecânico - Hidrômetro

Um tipo de sensor mecânico muito utilizado é o hidrômetro. De maneira geral, é um equipamento utilizado para medir o consumo de fluidos, possuindo internamente uma turbina que ao ser movimentada pelo fluxo do fluido alterna os valores numéricos contidos no rolete conforme pode-se visualizar na Figura 1.

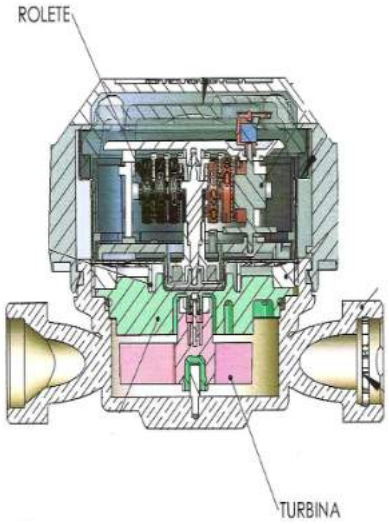

Fig. 1. Hidrômetro

Há motivos específicos para este tipo de sensor ser empregado, principalmente pela confiabilidade e precisão da medição - sendo que existem hidrômetros que possuem precisão de até 6 casas decimais. Esses equipamentos não dependem de nenhum dispositivo elétrico ou eletrônico para o seu funcionamento, podendo atuar em locais remotos com segurança sem depender de uma fonte de alimentação. Além disso, podem operar em locais com risco de explosão imerso a gás, já que a grande maioria constitui de peças de plásticas.

No entanto, devido a não dependência de energia elétrica, restringe-se monitorar seus dados de forma manual e visual e, portanto, para tornar este dispositivo em um equipamento integrado nos termos da Indústria 4.0, é necessário integrar um dispositivo de visão computacional que reconheça os valores numéricos e realize a leitura de forma automatizada.

Sobre essa óptica, não é viável empregar uma câmera sofisticada que exige um processador de alto desempenho para simplesmente realizar a leitura de algorismos, uma vez que seria comprometido a maior vantagem que os hidrômetros possuem de não depender de conexão física conectados para realizar medição e operar em locais remotos. Uma solução adequada e proposta é o emprego de dispositivos microcontrolados de baixo consumo que, por sua vez, são capazes de realizar a leitura e a interpretação dos algorismos através de uma abordagem com TinyML *Tiny Machine Learning* utilizando redes neurais convolucionais (CNN) e operando com bateria e transmitindo os dados a longas distâncias.

B. Redes Neurais Convolucionais

Uma Rede Neural Convolucional (ou *Convolucional Neural Network* - CNN) é uma variação das redes de *Perceptrons* de Múltiplas Camadas, tendo sido inspirada no processo biológicas de processamentos de dados visuais. De maneira semelhante aos processos tradicionais de visão computacional, uma CNN é capaz de aplicar filtros em dados visuais, mantendo a relação de vizinhança entre os pixels da imagem ao longo do processamento da rede. A Figura 2 ilustra uma CNN. Este tipo de rede vem sendo amplamente utilizado, principalmente nas aplicações de classificação, detecção e reconhecimento em imagens e vídeos.

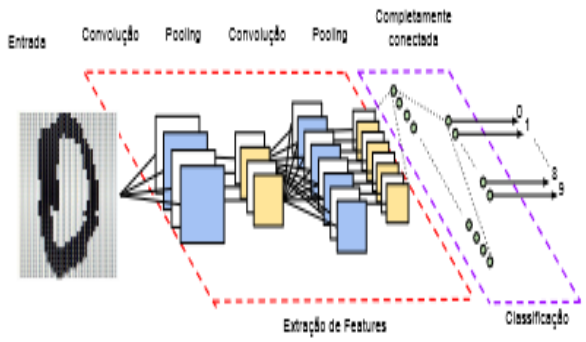

Fig. 2. Exemplo de uma rede Neural convolucional e suas diferentes camadas

Nos modelos desse tipo de rede neural, a ideia principal é filtrar linhas, curvas e bordas de uma imagem e aprender os detalhes desta imagem de forma a classificá-la. Desta forma, as etapas de uma rede neural convolucional, resumidamente, passa pelas etapas de:

1) Entradas: A entrada é a imagem transformada em uma matriz tridimensional com altura, largura e profundidade (dimensões dependendo da imagem) e pela quantidade de canais de cores. Em geral, as imagens utilizam três canais (RGB) com valores para cada pixel.

2) Convolução: As convoluções funcionam como filtros que enxergam pequenos quadrados e vão "escorregando" por toda a imagem captando os traços mais marcantes. Por exemplo, com uma imagem 32x32x3 e um filtro que cobre uma área de 5x5 da imagem com movimento de 2 saltos (chamado de *stride*), o filtro passará pela imagem inteira, por cada um dos canais, formando ao final um *feature map* ou *activation map* de 28x28x1.

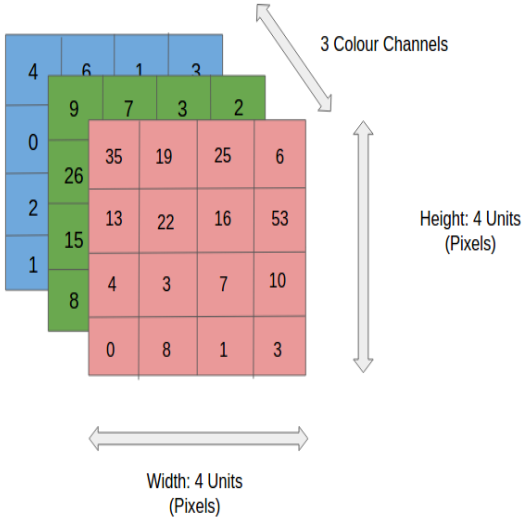

Fig. 3. Matriz de imagem com 3 camadas RGB

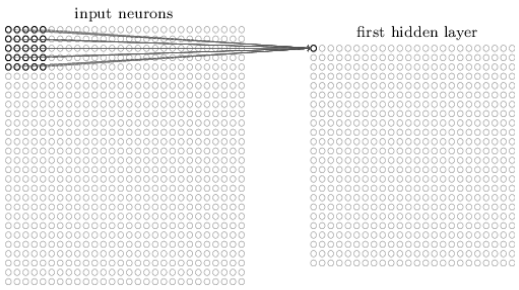

Fig. 4. Filtro de Convolução

3) Função de ativação: As funções de ativação servem para trazer a não-linearidades ao sistema, para que a rede consiga aprender qualquer tipo de funcionalidade. Há muitas funções, como *sigmoid*, *tanh* e *softmax*, porém a mais indicada para redes convolucionais é a *Relu* - por ser mais eficiente computacionalmente sem grandes diferenças de acurácia quando comparada a outras funções.

4) Pooling: Uma camada de *pooling* serve para simplificar a informação da camada anterior. Assim como na convolução, é escolhida uma unidade de área, para transitar por toda a saída da camada anterior. A unidade é responsável por resumir a informação daquela área em um único valor. Se a saída da camada anterior for 24x24, a saída do *pooling* será 12x12. Além disso, é preciso escolher como será feita a sumarização. O método mais utilizado é o *maxpooling*, no qual apenas o maior número da unidade é passado para a saída. Essa sumarização de dados serve para diminuir a quantidade de pesos a serem aprendidos e também para evitar *overfitting*.

5) Fully connected: Ao final da rede é colocada uma camada *Fully connected*, em que sua entrada é a saída

da camada anterior e sua saída são N neurônios, com N sendo a quantidade de classes do seu modelo para finalizar a classificação.

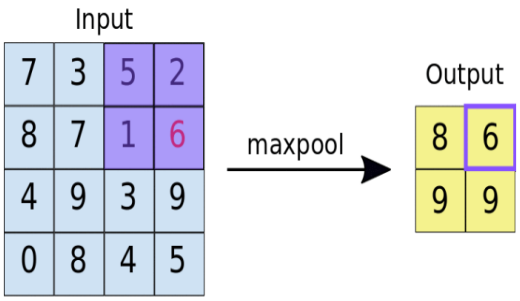

Fig. 5. Maxpooling 2x2

IV. Objetivo do projeto

Realizar a leitura automatizada de um medidor de água analógico, sem qualquer interferência estrutural no equipamento tirando regularmente imagens do medidor e realizar a classificação de leitura dos números usando inteligência artificial na forma de redes neurais convolucionais (CNN).

V. Motivação

Um estímulo para a definição da temática do projeto foi considerando a situação de quê, em grande parte do país, mediante o uso de hidrômetros, a medição volumétrica de água da rede de abastecimento público é realizada de forma manual e esse processo envolve alguns problemas ou, por outra perspectiva, algumas condições que poderiam ser melhoradas. Alguns desses pontos são que para leitura dos hidrômetros é exigido uma mão de obra humana periodicamente, no qual os agentes das empresas de saneamento visualmente anotam os números em um dispositivo ou formulário e, portanto, tal aferimento está sujeito à erros e temos um procedimento não tão eficiente e com credibilidade questionável. Logo, a proposta do trabalho também oferece um valor agregado relevante e possibilita a consolidação do conceito de IoT.

VI. Descrição do projeto

Para a consolidação do projeto, foi utilizado um microcontrolador de baixo consumo *Arduino Nano 33 BLE Sense* disponível no kit de *Arduino Tiny Machine Learning* que possui uma câmera integrada (módulo OV775) e também com o uso da ferramenta *Edge Impulse*, onde é concebível realizar todo o fluxo de trabalho de um projeto de TinyML ao possibilitar a coleta dos dados, o pré-processamento, a definição do modelo, o treinamento e, neste caso, os resultados da classificação, além de permitir a geração de uma biblioteca em linguagem C e, desse modo, exportar o modelo construído para o placa de desenvolvimento, efetivando a etapa de *deployment*.

VII. Desenvolvimento

A. Coleta de dados

Efetou-se a coleta dos dados por meio da gravação de um vídeo, no qual foi capturado o funcionamento de um hidrômetro a fim de obtermos os algorismos variando do número 0 até o número 9. Com auxílio do software *VLC* foram extraídas os *frames* do vídeo a uma taxa de uma foto por segundo, e , com isso, no total foram obtidas e selecionadas 764 imagens contendo as transições de 0 até 9 da seção do último algarismo do hidrômetro. Fazendo o *upload* do *dataset* criado ao *Edge Impulse*, foram criadas 10 classes (0 a 9), sendo que cada classe portava aproximadamente 80 imagens selecionadas de cada valor. A Figura 6 apresenta 4 imagens contidas, respectivamente, nas classes 0, 1, 4 e 8.

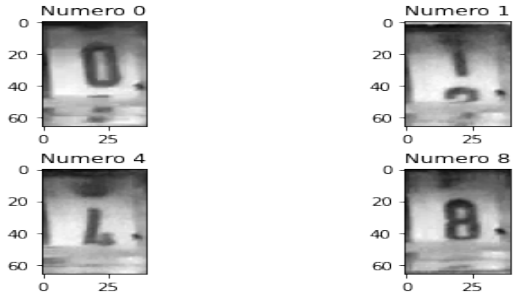

Fig. 6. Algarismos coletados

Observe que as imagens representam a transição dos valores, este é um ponto importante para a classificação, pois visa que o modelo também reconheça a transição e, logo, ele tem mais probabilidade de não deixar de sempre corresponder ao valor exibido.

B. A distribuição de dados e a criação do modelo

O balanceamento de dados foi feito de modo a representar 81% das imagens aleatoriamente para o treinamento e 19% para testes.

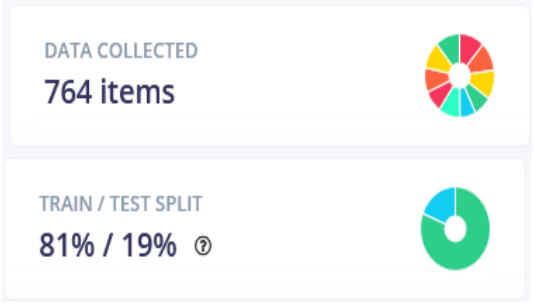

Fig. 7. Separação de dados de treinamento e testes

Essa divisão é importante, pois é possível validar o treinamento inferindo dados nunca vistos pela rede neural, aproximando-se do que será submetido na prática.

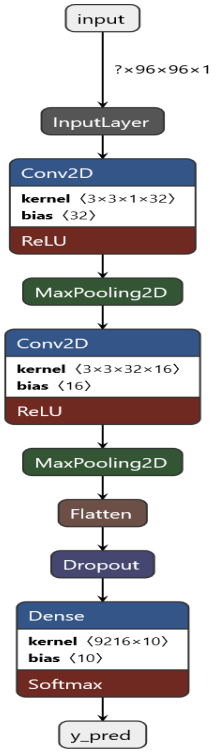

Fig. 8. Modelo da Rede Neural

C. Modelo e Treinamento

1) Pré-processamento: Conforme mencionado na descrição do projeto, para uma imagem ser classificada é necessário que seja feita o pré-processamento. Assim, na criação do Impulso, foi possível determinar o pré-processamento das imagens, no qual todas as imagens de treinamento e teste foram redimensionadas para o tamanho padrão de 96x96 pixel e alternou-se o padrão *RGB* para *Grayscale*, já que nesse cenário não é necessário e também ao se objetivar economia de processamento.

2) Configuração da Rede Neural: Os parâmetros exigidos para a criação da rede neural são:

a) Número de ciclos para o treinamento;
b) Taxa de aprendizagem;
c) Separação dos dados de validação do conjunto;
d) Quantidade de neurônios na camada de entrada;
e) Camadas de Convolução (para imagem);
f) *Flatten* 'Achatamento';
g) *Dropout*;
h) Camadas de saída.

Neste trabalho, foram utilizados 35 ciclos de treinamento para a rede neural realizar a aprendizagem, uma taxa de aprendizagem de 0.0005 e a definição de que 20% dos dados fossem destinados à validação do modelo.

A arquitetura da rede neural foi constituída de:

- 9.216 Neurônios de Entrada;
- Uma camada de Convolução 2D com 32 Filtros 3 *Kernels* e 1 camada;
- Uma camada de Convolução 2D com 16 Filtros 3 *Kernels* e 1 camada;
- Uma camada de *Flatten*;
- Camadas de Convolução (para imagem);
- *Dropout* de 25%;
- Camadas de saída com 10 classes (0 a 9).

Com o modelo criado, foi realizado o treinamento e, ao final, avaliou-se a precisão do modelo e o seu erro e, através de testes, repetiu-se os procedimentos e foram definidos novos parâmetros até o atendimento das expectativas.

A Figura 8 apresenta um modelo visual da configuração inicial da rede neural.

VIII. Resultados

Por meio do treinamento da rede neural, foi possível alcançar uma precisão de 92.2% e um erro de 0.26. As Figuras 9 e 10 apresentam, respectivamente, o agrupamento do modelo, que é um parâmetro importante que denota de forma visual como os valores foram agregados para serem classificados e a matriz de confusão que é útil para a indicação da qualidade do modelo.

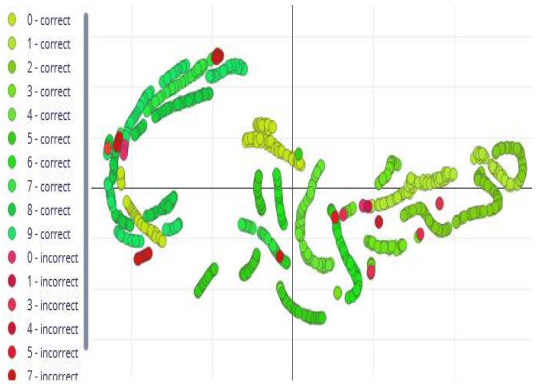

Fig. 9. Agrupamento do Modelo

	0	1	2	3	4	5	6	7	8	9
0	85.7%	0%	0%	4.8%	0%	0%	0%	0%	0%	9.5%
1	0%	94.4%	0%	5.6%	0%	0%	0%	0%	0%	0%
2	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
3	0%	0%	8.3%	75%	0%	8.3%	8.3%	0%	0%	0%
4	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
5	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
6	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
7	0%	0%	0%	0%	0%	0%	0%	66.7%	6.7%	26.7%
8	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
9	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%
F1 SC	0.92	0.97	0.97	0.78	1.00	0.97	0.95	0.80	0.97	0.81

Fig. 10. Matriz de confusão

É válido destacar que, com a análise da matriz de confusão, constata-se que a maior parte dos erros encontrados na validação aconteceram em números próximos (por exemplo, classificar o número 7 como 8 ou classificar o número 5 com o 3), sendo que o único caso crítico exposto ocorreu na classificação do número 0 como número 8, pois, na situação dado na motivação, isso seria um grande problema e uma preocupação já que temos fatores financeiros envolvidos na aferição.

Apesar do modelo ter apresentado um resultado satisfatório, ele apontou uma performance de alto consumo de memória RAM (363,9kB) e FLASH (128kB) - o que tornou sua implementação inadequada para o microcontrolador utilizado. Desta forma, quando exportado o modelo e importado a biblioteca advinda do *Edge Impulse* para a IDE Arduino, encontrou-se um impasse na classificação: ao executar-se o código, houve uma única classificação e, em seguida, a informação de quê não era possível alocar recursos de memória era exibida via serial e, com isso, não foi possível realizar o *deployment*. Ainda que houve tentativas de contornar a situação, refatorando os parâmetros do modelo, diminuindo a quantidade de amostras do *dataset* ou até mesmo diminuindo o número de classes, a ordem de grandeza dos recursos de memória se mantiveram.

A. *Inferência em tempo real*

Uma maneira de apresentar os resultados encontrada foi através da classificação em tempo real *Live Classification* do *Edge Impulse*. Por intermédio desse recurso foi possível conectar o microcontrolador via serial e fazer a inferência no próprio sistema.

Para isto, foi montado um *setup* exposto nas Figuras 11 e 12 que, além do hidrômetro e da placa de desenvolvimento acoplada, possui um mecanismo de fluxo de água ao se utilizar de uma bomba improvisada e possibilitar o funcionamento do hidrômetro. Com a construção do protótipo, foi possível capturar algumas amostras com o dispositivo e visualizar as classificações.

B. *Resultados Obtidos*

Os resultados da classificação foram satisfatórios, pois foram realizadas 40 inferências e dentre elas, 33 foram corretas e 7 incorretas (implicando um acurácia de cerca de 82,5%). No link da apresentação disponibilizado ao fim deste documento, são apresentadas algumas das classificações feitas. Dentre as inferências realizadas, verificou-se também que, apesar de objetivarmos tratar as transições durante a elaboração do *dataset*, alguns resultados adversos e não ideais foram encontrados.

IX. Conclusão

Alguns pontos de reflexão ao concluir nosso trabalho foram que:

a) É necessário ter um tratamento e cuidado com as transições, possuindo uma estratégia de contorno para isso;

Fig. 11. Placa acoplada ao hidrômetro

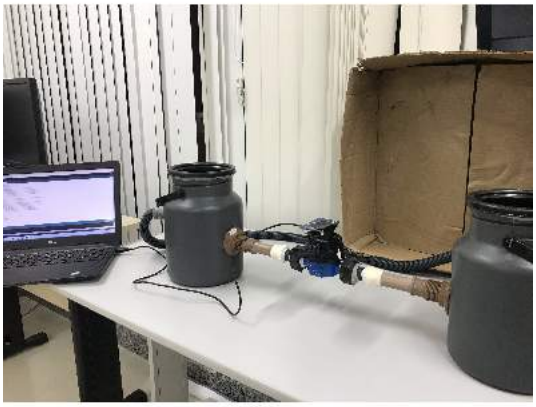

Fig. 12. Setup de testes

b) É ideal refatorar o modelo e possivelmente readequá-lo com um modelo de placa com mais recursos (como uma ESP32, por exemplo).

Embora o objetivo final do trabalho não tenha sido alcançado (tratando-se de embarcar o projeto), foi possível implementar parcialmente o modelo proposto, obtendo-se um bom resultado com a classificação dos algorismos. Ademais, algumas considerações e sugestões para trabalhos futuros são: a mudança do modelo, ao invés de trabalhar com classificação, por se tratar de muitos dados, uma regressão pode atender melhor o objetivo e produzir um modelo menor que pode ser empregado em microcontroladores; enxerga-se também possibilidade de expansão do projeto para leitura de todos algarismos do hidrômetro, em que, nesse cenário, haveria uma captura geral do sensor e o código realizaria um *crop* das seções de cada algarismo, submeteria cada uma das imagens ao modelo e combinaria os resultados individuais.

X. Anexos

Link do projeto no *Edge Impulse*: https://studio.edgeimpulse.com/public/117136/latest

Link do vídeo da apresentação: https://youtu.be/rV6lmdJqsvc

XI. Referências

[1] - Entendendo Redes Convolucionais (CNNs), 2021. Disponível em: https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184

[2] - Welcome to the AI-on-the-edge-device! , 2020. Disponível em: https://github.com/jomjol/AI-on-the-edge-device/wiki. Acesso em: 01 jun. 2020.

[3] - O que fazer com máquinas antigas na transição para a Indústria 4.0?, 2018. Disponível em: https:https://avozdaindustria.com.br/industria-40-totvs/o-que-fazer-com-maquinas-antigas-na-transicao-para-industria-40. Acesso em: 01 jun. 2020.