Classificação Engrenagem

1st Fábio Augusto Pereira
Programa de Mestrado Inatel
Santa Rita do Sapucaí-MG, Brasil
fabio.pereira@mtel.inatel.br

***Abstract*—Este artigo tem por finalidade apresentar um modelo de Machine Learning desenvolvido para identificação de peças íntegras ou com defeito em uma linha de produção destinada ao abastecimento de montadoras de equipamentos que utilizam as engrenagens em seus produtos. A principal motivação do desenvolvimento desse modelo é utilizar técnicas de visão computacional embarcada em dispositivos IoT para agilizar e melhorar os resultados na linha de produção e mitigar a distribuição de peças com defeito no mercado, tornando o processo mais eficiente e econômico. (*Abstract*)**

Keywords—dataset, Edge Impulse, Machine Learning, Yolov5

I. INTRODUÇÃO

A indústria em geral tem como objetivo fornecer e garantir a qualidade do produto que produzem. Esse processo envolve várias etapas desde a concepção até o produto final, como o desenvolvimento do modelo, liga do metal utilizada e precisão na padronização das peças fabricadas. A identificação de possíveis falhas durante o processo de produção ajuda a minimizar riscos de instalação de peças defeituosas nos produtos e com isso contribui com a credibilidade da marca protegendo a reputação da empresa buscando a satisfação do consumidor final.

Segundo Nick Bild em seu artigo (2) estudos estatísticos de orgãos que medem a qualidade e satisfação de clientes, apontam que investimentos na inspeção efetiva e mais robusta durante o processo de produção em busca de possíveis falhas para garantir a qualidade do produto ofertado, pode resultar em uma significativa margem de economia dos custos de produção, comprovando que o investimento no controle de qualidade do produto traz relevante retorno financeiro ao fabricante.

A implementação de processos efetivos de classificação da qualidade de produtos enfrenta alguns obstáculos para sua implementação, pois envolve várias etapas e exige uma integração muito ajustada de todas as equipes técnicas envolvidas no processo sem interromper a produção, com o objetivo da melhor qualidade do produto final.

Utilizando técnicas de Visão Computacional e Machine Learning, é possível desenvolver uma solução para detecção de peças defeituosas de maneira rápida e efetiva.

II. MATERIAIS E MÉTODOS

A. Equipamentos Utilizados

Para a aquisição das imagens necessárias para o treinamento do modelo, foi utilizado um aparelho Smartphone modelo Poco X3 Pro da marca Xiaomi, equipado com chipset Snapdragon 860 e câmera traseira de 48 Megapixels de resolução.

B. Dataset

Para formação do *dataset* foram geradas 130 imagens separadas em duas classes, *approved* e *defected*. As imagens foram divididas em 80% para o conjunto de treinamento e 20% para o conjunto de testes.

C. Plataforma

Para a criação e treinamento do modelo foi utilizado a plataforma *Edge Impulse*, que é uma plataforma *web* destinada a facilitar a criação e implementação de inteligência artificial destinados a dispositivos IoT e com recursos limitados como por exemplo microcontroladores e sistemas embarcados. Ela é frequentemente utilizada para a criação de modelos de *Machine Learning*, integração de sensores como acelerômetro, microfone, câmeras entre outros, para a adquirir os dados e realizar o treinamento *online* dos modelos.

D. Pré - Processamento

Na estapa de pré-processamento das imagens para o formato quadrado, que é o formato esperado pela arquitetura utilizada para a detecção de objetos, com resolução de 96X96.

Foram geradas as características necessárias para o treinamento das imagens baseada nas definições de cores RGB além da rotulagem para separação dos conjuntos em *approved* e *defected*.

E. Arquitetura da Rede Neural

A arquitetura da rede neural utilizada é uma Rede Neural Convulacional (CNN).

A CNN é muito poderosa para a realização da tarefa de classificação de imagens pois é especialmente projetada para a utilização de tarefas de visão computacional, ela reconhece padrões em imagens através da exploração de hierarquia das caracteríStcas contidas na imagem, permitindo que a rede aprenda as representações complexas dessas características à medida que processa suas camadas composta principalmente pelos componentes:

- Camadas Convulacionais

  A convolução é o processo de utilização de filtros (também podem ser definidos como *kernels*) que deslizam sobre a imagem para a extração de suas características locais, onde durante o treinamento cada filtro captura padrões específicos (bordas, textura, ou formas) com seus pesos sendo ajustados automaticamente para aprender padrões importantes.

- Camadas de Pooling

  Essas camadas são importantes para se evitar a ocorrência de *overfitting* durante o treinamento, pois elas reduzem a dimensionalidade espacial da

representação da imagem. A mais comum dessas operações é a "max pooling", que para ajudar a preservar as características importantes contidas na imagem preserva apenas o valor máximo da região.

- Camadas Densamente Conectadas

Geralmente usadas para combinar informações globais ou específicas extraídas da imagens nas camadas convolucionais e de pooling, alimentando camadas densamente conectadas para realizar classificações ou alguma outra tarefa específica.

- Camadas de Ativação

É comum aplicar funções de ativação como a ReLU (Rectfied Linear Unit), que é uma função não linear amplamente utilizada em camadas ocultas de redes neurais, definida como da seguinte maneira:

$$f(x) = \max(0,x)$$

Por definição, para qualquer valor de entrada x, a saída da função ReLU será 0 se x for menor que 0, e será x se x for maior ou igual a 0.

Ao se utilizar a função para introduzir a não-linearidade na rede neural é necessária para a capacidade da rede de aprender comportamentos complexos e representações não lineares dos dados. A função é computacionalmente eficiente pois ativa muitas unidades diretamente sem exigir cálculos exponenciais ou trigonométricos.

- Camadas de Normalização

Batch Normalization ou Normalização em Lote é uma técnica frequentemente utilizada em redes neurais convulacionais (CNNs) e redes profundas para obter melhor desempenho e estabilidade do treinamento, pois tem por proposta abordar problemas na instabilidade do treinamento e a dependência da inicialização dos pesos, é uma técnica utilizada em redes neurais para normalizar as entradas de cada camada para garantir que a ativação não fique muito alta ou muito baixa durante o treinamento. Podemos analisar seu funcionamento da seguinte maneira:

a) Normalização

Normaliza as entradas de cada mini-lote dos dados subtraindo a média do lote e dividindo pelo desvio padrão do lote, aplicando a cada canal de entrada para ajudar a manter uma escala relativa consistente nas ativações.

b) Transformação afinada

Após a normalização introduz dois parâmetros treináveis, chamados de fator de escala (γ) e fator de deslocamento (β). A saída normalizada é multiplicada pelo fator de escala (γ) e em seguida se adiciona o fator de deslocamento (β), permitindo que a rede neural aprenda a escala e o deslocamento ideais para cada ativação.

A Batch Normalization para cada camada é expressa pela equação:

$$BN(\chi) = \gamma (\chi - \mu / \sqrt{\sigma^2 + \varepsilon}) + \beta$$

onde:

- χ = é a entrada para a camada
- γ = é o fator de escala
- σ = é o desvio padrão do lote
- μ = é o média do lote
- β = é o fator de deslocamento
- ε = é um pequeno valor adicionado para evitar a divisão por 0

A utilização da Batch Normalization nos apresenta benefícios como a aceleração do treinamento, pois permite o uso de taxas de aprendizado mais altas pela estabilização que proporciona, também reduz a dependência de escolha da inicialização dos pesos além de ajudar a mitigar os problemas de desvanecimento ou explosão do gradiente, permitindo assim o treinamento mais profundo.

III. MODELO PARA CLASSIFICAÇÃO

Para a realizar a classificação e treinamento dos dados, foi utilizado a arquitetura YOLOv5 e FOMO para comparação de resultados.

YOLO (You Only Look Once) é uma arquitetura construída em PyTorch de detecção de objetos em tempo real que propõem uma abordagem diferente das redes neurais convulacionais tradicionas para o problema, neste projeto utilizamos a versão 5 da arquitetura.

Em posse do conjunto de dados separados entre treino e teste, no pré processamento foi realizado a rotulagem através de caixa delimitadora (*bounding box*) como *approved* ou *defected*.

Após geradas as características das imagens necessárias para o treinamento, foi iniciado o treinamento em diferentes arquiteturas para comparação, FOMO (Fast Objects, More Objects) que é uma arquitetura baseada no MobileNetV2 e também treinamentos utilizando a arquiterura YOLOv5 em 2 configurações a Community blocks e a Renesas, com variações de quantidade de épocas, com um conjunto de validação de 20% do conjunto de treinamento.

Modelos Treinados			
Rede Neural	***Épocas***	***Precisão***	***Acurácia***
YOLOv5 - Community Blocks	60	46.60%	79.17%
YOLOv5 - Renesas	60	52.80%	25.50%
YOLOv5 - Renesas	100	46.90%	79.17%
YOLOv5 - Renesas	110	49.20%	75.00%
FOMO	60	73.70%	58.33%

Tabela I – Resultados de treinamento

Realizado os treinamentos diferentes os melhores resultados foram obtidos com a arquitetura YOLOv5 – renesas após 100 épocas de treinamento com uma precisão de 46,90% e uma acurácia de 79.17%

IV. Resultados

Feitos os testes no modo Live Classification da plataforma Edge Impulse, os resultados foram satisfatórios em relação ao reconhecimento das peças apresentadas para reconhecimento, mas o modelo ainda apresenta algumas inconsistências no reconhecimento das peças apresentadas.

Espera-se que após adionar mais amostras de imagens ao dataset utilizado nos conjuntos de testes e validação e proporcionar mais treinamento ao modelo, resultados melhores de acurácia podem ser apresentados.

Os arquivos desse projeto estão no repositório do github:

HTTPS://GITHUB.COM/FABIOAUGUSTOPEREIRA/CLASSIFICACAO_ENGRENAGEM

References

Para este artigo foi utilizado como fontes de referência documentação oficial obtida online e disponibilizada pelos idealizadores das arquiteturas utilizadas para o treinamento além de artigos publicados na plataforma Edge Impulse utilizada na criação e treinamento do modelo.

[1] Ultralytics, docs.ultralytics.com/yolov5.

[2] Build Nick, “Elevation quality standarts with AI”, edgeimpulse.com, 2023.