Sistema de Identificação de Aeronaves

SISTEMA DE IDENTIFICAÇÃO DE AERONAVES

1 - RESUMO

O notebook realiza o processo completo de treinamento de um modelo de detecção de objetos com YOLOv8, desde a preparação dos dados até a avaliação do modelo em novas imagens. O objetivo é criar um modelo capaz de detectar diferentes tipos de aviões em imagens. O notebook está usando a técnica de **Redes Neurais Convolucionais (CNN)** para detecção de objetos, através da biblioteca Ultralytics YOLOv8.

YOLO (You Only Look Once) é um algoritmo de detecção de objetos em tempo real que utiliza uma única rede neural para prever caixas delimitadoras e probabilidades de classe para os objetos na imagem. As versões mais recentes do YOLO, como o YOLOv8, são baseadas em **arquiteturas CNN**.

CNNs são um tipo de rede neural profunda projetadas para processar dados com uma estrutura de grade, como imagens. Elas utilizam filtros convolucionais para extrair características da imagem em diferentes níveis de abstração, permitindo que o modelo aprenda padrões complexos e realize a detecção de objetos.

O dataset utilizado foi o *https://www.kaggle.com/datasets/a2015003713/militaryaircraftdetectiondataset*. Este conjunto de dados abrange 74 tipos diferentes de aeronaves militares. O conjunto de dados é organizado em três pastas principais: ‘dataset’, ‘crop’, ‘annotated_sampes’. Para este treinamento, usamos apenas a pasta ‘dataset’
A pasta ‘dataset’ contém imagens JPEG (.jpg) e arquivos de notação correspondentes no formato CSV com nomes de arquivos correspondentes.Cada arquivo CSV de anotação detalha os objetos dentro de uma imagem, usando o formatop PASCAL VOC. As colunas nos arquivos CSV de anotação são:

- *filename:* O identificador da imagem e do arquivo de anotação correspondente
- *width e height:* as dimensões da imagem em pixels
- *class:* O tipo de aeronave
- *xmin, ymin, xmax, ymax:* As coordenadas da caixa delimitadora

O dataset completo contém mais de 36.000 fotos, impossibilitando o uso do dataset completo para este trabalho. Por isso, foram selecionados apenas 2 aviões (1.586 fotos):

- A10 – ThunderBolt

- A400M

2 - INTRODUÇÃO

2.1 – Contexto:

O reconhecimento de aeronaves em aplicações militares é de fundamental importância, pois fornece informações cruciais para tomadas de decisões estratégicas e táticas. Com o reconhecimento de aeronaves é possível:

- Identificar ameaças
- Avaliar a situação
- Planejar missões
- Proteger alvos
- Coletar inteligência

Os desafios no reconhecimento visual de aeronaves ainda apresentam diversos desafios devido à complexidade das imagens e as variáveis presentes no ambiente real. Alguns dos principais são:

- Variabilidade de ângulos
- Condições de iluminação
- Camuflagem
- Ruído e Distorção
- Variabilidades intraclasses
- Objetos semelhantes
- Real-Time processing

2.2 – Objetivo:

O objetivo deste trabalho é a criação de uma IA usando deep learning, capaz de identificar e classificar diferentes tipos de aeronaves a partir de imagens. Como mencionado antes, para este trabalho estamos usando apenas 2 aeronaves (A10 e A400M)

3 - CHECKLIST

3.1 Definição do Problema

3.1.1– Qual a descrição do problema?

Desenvolver um modelo de detecção de objetos capaz de identificar e classificar diferentes tipos de aviões em imagens

3.1.2 – Você tem premissas ou hipóteses sobre o problema? Quais?

Existem algumas premissas e hipóteses implícitas no problema de detecção e classificação de aviões em imagens:

- Premissas
  - **Qualidade de dados:** Assume-se que o conjunto de dados usados para treinar o modelo é representativo do problema e contêm imagens de aviões com anotações precisas de caixas delimitadoras e classes.
  - **Diversidade de dados:** O conjunto de dados deve conter uma variedade suficiente de imagens de aviões em diferentes condições de iluminação, ângulos de visão, fundos e tipos de aviões para que o modelo possa generalizar bem as novas imagens.

- **Formato dos dados:** Assume-se que as imagens e as anotações estão no formato correto e compatível com o YOLOv8, incluindo a estrutura de diretórios e o formato das anotações.

- **Capacidade YOLOv8:** Assume-se que a arquitetura do YOLOv8 e suas capacidades de objetos são adequadas para o problema de detecção e classificação de aviões.

- Hipóteses
  - **Aviões Visíveis:** O modelo é treinado para dsetectar aviões que são visíveis nas imagens. Se um avião estiver "escondido" ou muito pequeno, o modelo por ter dificuldade em detectá-lo

  - **Imagens de Qualidade razoável:** O modelo espera receber imagens com qualidade razoável, sem muito ruído ou distorções. Imagens de baixa qualidade podem afetar a precisão da detecção.

  - **Classes bem definidas:** As classes de aviões devem estar definidas e bem distintas entre si. Se houver muita sobreposição entre as classes, o modelo pode ter dificuldade em classificá-las corretamente.

3.1.3 – Que restrições ou condições foram impostas para selecionar os dados?

Devido a quantidade de informações do dataset escolhido e a limitação física dos equipamentos utilizados, foram selecionados apenas 2 tipos de aviões para o treinamento (A10 e A400M)

3.1.4 – Descreva o dataset (atributos, imagens, anotações, etc)

O dataset é composto por imagens de aviões e suas respectivas anotações. As anotações são fornecidas em arquivos CSV que contém os seguintes atributos:

- *filename:* O identificador da imagem e do arquivo de anotação correspondente
- *width e height:* as dimensões da imagem em pixels
- *class:* O tipo de aeronave
- *xmin, ymin, xmax, ymax:* As coordenadas da caixa delimitadora

As imagens do dataset são fotos dos aviões A10 e A400M em diferentes ângulos e com diferentes fundos. As imagens são em .JPG

3.2 Preparação de dados

3.2.1– Separe o dataset entre treino e teste (e validação, se aplicável)

O dataset é dividido em conjuntos de treinamento (80%) e validação (20%). Depois do treinamento, o modelo treinado é testado com 5 novas figuras não identificadas

3.2.2– Faz sentido utilizar um método de validação cruzada? Justifique se não utilizar

O método de validação cruzada (como o K-Fold) é indicado para datasets médios e pequenos. Neste caso específico, considero que o dataset é médio (1586 fotos) o que poderia ate justificar o uso de validação cruzada.. Só que, neste notebook, estou utilizando o YOLOv8 que já possui diversas métricas de avaliação durante o treinamento. Essas métricas permitem monitorar o desempenho do modelo e evitar o overfitting. O YOLOv8 também suporta o early stopping, que interrompe o treinamento automaticamente quando o desempenho no conjunto de validação para de

melhorar. Isso ajuda a evitar o overfitting e a encontrar o melhor modelo dentro de um número menor de épocas.

Conclusão: Até poderia usar o K-fold, mas isso geraria um custo operacional bem maior e não traria ganhos significativos

3.2.3– Refine a quantidade de atributos disponíveis, realizando o processo de feature selection de forma adequada

Para este caso, não será necessário fazer o feature selection, já que as informações necessárias para o aprendizado já estão definidas nas imagens em .JPG e nos arquivos CSV

3.3 Modelagem e Treinamento

3.3.1– Selecione os algoritmos mais indicados para o problema e dataset escolhidos, justificando suas escolhas

Para o dataset em questão, o algoritmo utilizado é o YOLOv8 que se destaca como a melhor opção para o problema devido à sua combinação de velocidade, precisão, facilidade de uso e recursos.

3.3.2– Há algum ajuste inicial para os hiperparâmetros?

O YOLOv8 possui alguns hiperparâmetros que podem ser ajustados inicialmente para melhorar o desempenho do modelo no seu problema de detecção de objetos em imagens de aviões.

epochs: Número de épocas de treinamento.

Sugestão: Comece com um valor entre 10 e 30 épocas. Se o modelo estiver com overfitting (desempenho bom no treinamento, mas ruim na validação), reduza o número de épocas. Se o modelo ainda não estiver convergindo, aumente o número de épocas.

imgsz: Tamanho da imagem de entrada.

Sugestão: Mantenha o valor padrão de 640x640 pixels, que é um bom equilíbrio entre precisão e velocidade para o YOLOv8. Se você tiver imagens com resoluções muito diferentes, ajuste o imgsz para um valor próximo à resolução média das suas imagens.

batch: Tamanho do batch.

Sugestão: Use o maior tamanho de batch que a sua GPU suportar. Isso acelera o treinamento. Se você tiver problemas de memória, reduza o tamanho do batch.

lr0: Taxa de aprendizado inicial.

Sugestão: Comece com o valor padrão de 0.01. Se o modelo estiver divergindo, reduza a taxa de aprendizado. Se o modelo estiver convergindo lentamente, aumente a taxa de aprendizado.

lrf: Taxa de aprendizado final.

Sugestão: Mantenha o valor padrão, que é uma fração da taxa de aprendizado inicial. O YOLOv8 usa um escalonamento de taxa de aprendizado para reduzir a taxa de aprendizado ao longo do treinamento.

momentum: Momentum do otimizador SGD.

Sugestão: Mantenha o valor padrão de 0.937. O momentum ajuda a acelerar o treinamento e a evitar mínimos locais.

weight_decay: Decaimento de peso (regularização L2).

Sugestão: Mantenha o valor padrão de 0.0005. O decaimento de peso ajuda a prevenir overfitting.

optimizer: Otimizador.

Sugestão: Use o otimizador padrão SGD (Stochastic Gradient Descent), que é eficiente para treinar o YOLOv8. Você pode experimentar outros otimizadores, como Adam, mas o SGD geralmente funciona bem para detecção de objetos.

3.3.3– O modelo foi devidamente treinado ? Foi observado algum problema de underfitting ?

Durante o treinamento, foi identificado que o modelo estava aprendendo de forma consistente, como mostra a figura abaixo:

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances
1/100      2.5G      1.148      4.843      1.237         31
           Class     Images  Instances      Box(P          R

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances
2/100     2.52G      1.136      3.973      1.252         42
           Class     Images  Instances      Box(P          R

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances
3/100     2.71G      1.176        3.4      1.286         38
           Class     Images  Instances      Box(P          R
```

O “box_loss” e “cls_loss” estão diminuindo, sendo que no “Epoch” = 100 os resultados são:

```
  Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances
100/100     2.36G     0.4625      1.029     0.8744         24
           Class     Images  Instances      Box(P          R
```

Perceba que as “perdas” foram ficando menores a cada “Epoch”, as vezes variando para cima, mas na grande maioria, os valores foram decrescendo. Acima de 200 “Epochs” notei que os valores não baixaram tanto, podendo ser um problema de underfitting

3.3.4– É possível otimizar os hiperparâmetros de algum dos modelos? Se sim, faça-o, justificando todas as escolhas

Sim, é possível altera alguns dos parâmetros (exemplo: o *imgsz* de 640 para 416). Infelizmente não consegui fazer o teste usando a mesma quantidade de Epochs do primeiro teste (acabaram as unidades computacionais do Colab)

3.3.5– Há algum método avançado ou mais complexo que possa ser avaliado?

Sim, existem métodos mais avançados e complexos que podem melhorar o desempenho do modelo. Abaixo seguem algumas sugestões:

- Arquiteturas e Modelos personalizados

- Criar um modelo YOLOv8 personalizado
- Explorar outras arquiteturas (Exemplo: EfficientDet)
- Combinar diferentes modelos (Ensemble)

- Técnicas de Treinamento avançadas
  - Aprendizado por transferência
  - Otimizadores avançados
  - Técnicas de Regularização

- Melhoria no dataset
  - Aumento de dados
  - Limpeza e curadoria do dataset
  - Geração de dados sintéticos

3.3.6– Posso criar um comitê de modelos diferentes para o problema (ensembles)?

Sim, é possível criar um comitê de modelos diferentes

3.4 Avaliação de Resultados

3.3.1– Selecione as métricas de avaliação condizentes com o problema, justificando.

Como usei o YOLv8, ao fim do treinamento, vários gráficos são gerados entre eles:

- **RESULTS** – Resume o desempenho do modelo onde apresenta a progressão das métricas ao longo das "epochs"
  - `Box Loss`
  - `Objectness Loss`
  - `Classification Loss`
  - `Precision`
  - `Recall`
  - `mAP50` (mean Average Precision com IoU de 0.5)
  - `mAP50-95` (mean Average Precision com IoU de 0.5 a 0.95)

Nos gráficos apresentados (para EPOCH = 100), nota-se que as perdas vão reduzindo a cada nova aprendizagem

- **MATRIZ DE CONFUSÃO** – Representação visual do desempenho da classificação do modelo, mostrando a contagem de positivos, falso positivos, verdadeiros negativos e falsos negativos

- **PR CURVE** – Uma curva que mostra a relação entre a precisão e revocação para diferentes limiares de confiança. Perceba que no gráfico geral, a curva "all classes" ela vai decrescendo e as curvas referentes ao A10 e ao A400 "seguem" esta tendência tendo uma maior "queda" em relação aos demais aviões. Isso porque o foco do treinamento foi exatamente nestes 2 modelos de aviões. Os demais ocorrem porque em alguma foto ou outra, os mesmos aparecem

- **F1 CURVE** - Uma curva que mostra a relação entre precisão e revocação para diferentes limiares de confiança, mas ponderada pela média harmônica (F1-score).

- **P, R, mAP@.5 & mAP@.5:.95**: Tabela com os valores finais para precisão, revocação, mAP@.5 e mAP@.5:.95.

3.3.2– Treine o modelo escolhido com toda a base de treino, e teste-o com a base de teste

Testes foram realizados

3.3.3– Os resultados fazem sentido?

Sim. Notei em alguns testes que o modelo treinado conseguia identificar os aviões, as vezes com um pouco de imprecisão quando os testes eram realizados com "Epochs" abaixo de 100. Com mais "Epochs" o modelo demonstrou maior precisão

3.3.4– Foi observado algum problema de overfitting ?

Os gráficos apontam para não

3.3.5– Descreva a melhor solução encontrada, justificando

Para o aprendizado desses 2 modelos, o modelo apresentou bons resultados com 100 "Epochs". Acredito que com 200 "epochs" seria o ideal. Infelizmente não foi possível realizar tal teste devido a falta de capacidade operacional