Projeto YOLOv8 – Detecção de Acessórios Faciais

📝1. Objetivo do projeto

Desenvolver um modelo YOLOv8 para detectar acessórios faciais (barba, oculos, mascara, chapeu) em selfies e integrá-lo a um sistema de validação de identidade (RG x Selfie).

⚙️2. Instalação do ambiente

🔗2.1. Criação do ambiente virtual

```
python -m venv venv
venv\Scripts\activate
```

🔗2.2. Instalação de bibliotecas

```
pip install ultralytics opencv-python torch torchvision torchaudio
```

✔️**Observação**: Torch foi instalado como:

```
pip install torch torchvision torchaudio --index-url https://
download.pytorch.org/whl/cpu
```

Pois a máquina não possui GPU Nvidia.

🗂️3. Organização do dataset

Estrutura de pastas criada:

```
YOLOv8_Custom_Accessories/
 ├── images/
 │    ├── train/
 │    ├── val/
 │    └── test/
 └── labels/
      ├── train/
      ├── val/
      └── test/
```

🔗3.1. Anotação no Roboflow

1. Criado projeto *Acessorios Faciais*.
2. Imagens anotadas com as classes:
3. barba
4. oculos
5. mascara
6. chapeu
7. Dataset exportado em formato **YOLOv8 PyTorch**.
8. Verificado o arquivo `data.yaml`:

```
train: images/train
val: images/val
test: images/test

nc: 4
names: ['barba', 'oculos', 'mascara', 'chapeu']
```

---

4. Treinamento do modelo YOLOv8

🔗**4.1. Script **``

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')

model.train(
    data='data.yaml',
    epochs=50,
    imgsz=640,
    batch=8,
    device='cpu'
)
```

🔗4.2. Resultado final do treinamento

Classe	mAP50	mAP50-95	Precision	Recall
barba	0.995	0.895	0.00847	1.0
chapeu	0.995	0.796	0.0222	1.0
face	0.0	0.0	0.0	0.0
oculos	0.995	0.895	0.0133	1.0

✔️**Modelo salvo em:** `runs/detect/train/weights/best.pt`

💻5. Teste do modelo treinado (inferencia_YOLO.py)

```
import cv2
from ultralytics import YOLO

model = YOLO('runs/detect/train/weights/best.pt')

img_path = "images/test/selfie.jpg"

results = model(img_path, show=True)
results.print()
```

🔗6. Integração ao projeto (attribute_detector.py)

🔗6.1. Código de detecção de atributos

```
import cv2
from ultralytics import YOLO

model = YOLO('runs/detect/train/weights/best.pt')

def detect_attributes(img_path):
    atributos = []
    img = cv2.imread(img_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    results = model(img_rgb)

    for r in results:
        for box in r.boxes:
            cls_id = int(box.cls[0])
            conf = float(box.conf[0])
            label = r.names[cls_id]
            if conf > 0.5:
                atributos.append(label)
    return atributos
```

📌7. Observações finais

✔️Torch foi instalado em versão CPU devido ausência de GPU CUDA.\ ✔️O modelo detectou classes com alta mAP mas baixa precision devido ao dataset pequeno.\ ✔️Recomenda-se **ampliar o dataset (≥50 imagens por classe)** e treinar novamente para melhor performance.

🕗8. Próximos passos sugeridos

- Aumentar dataset com fotos variadas (ângulos, luz, fundo).
- Treinar mais epochs após aumento de dataset.
- Integrar modelo final ao pipeline de validação de RG x Selfie com geração automática de relatórios.

Documento gerado automaticamente para organização e estudo estratégico do projeto YOLOv8 – Discrepância RG x Selfie.

🔽Me avise para gerar slides ou fluxogramas visuais deste pipeline para TCC e apresentações.

🔗3.2. Detalhamento do uso do YOLO no Roboflow

Foi criado um projeto de detecção de objetos no Roboflow, onde foram feitas as seguintes etapas detalhadas:

1. Upload de todas as imagens de selfie contendo acessórios faciais.
2. Anotação manual das caixas delimitadoras (bounding boxes) para cada classe (barba, oculos, mascara, chapeu) em cada imagem.
3. Verificação individual das anotações para garantir precisão de rótulo e posição.
4. Geração do dataset via botão "Generate Dataset", definindo splits de treino, validação e teste automáticos.
5. Exportação do dataset no formato YOLOv8 PyTorch com arquivo data.yaml configurado com classes e caminhos relativos.
6. Download e organização local do dataset para uso no treinamento.