🚀 Guia de Execução – Sistema de Detecção de Quedas

## 🔑 Sequência Recomendada de Execução

### 1. Preparar ambiente

make install  
source .venv/bin/activate

* Cria e ativa o ambiente virtual .venv
* Instala dependências do requirements.txt

### 2. Preparar dados (vídeos e anotações)

1. Coloque os vídeos brutos em ./data/
2. Anote dados no **CVAT**:
   * Criar projeto → Labels: Pessoa1, Queda
   * Criar Tasks → importar vídeos
   * Anotar usando caixas (Bounding Box)
   * Exportar dataset no formato **YOLO 1.1**
   * Colocar export em: datasets/queda\_casoX\_yolo/
3. Converter dataset exportado:

make dataset-cvat ZIP=./datasets/queda\_casoX.zip OUT=./datasets/queda\_casoX\_yolo

1. (Opcional) Dividir dataset em treino/validação/teste:

make dataset-split

### 3. Treinamento do modelo

make yolo-train EPOCHS=50 IMG=640

* Modelo é salvo em runs/detect/train/weights/
* Métricas e gráficos ficam em runs/detect/train/

Retomar treino do último checkpoint:

make yolo-resume

### 4. Testes rápidos (inferência)

make yolo-predict SRC=./data/Queda\_banheiro.mp4 CONF=0.5

* Saídas ficam em runs/detect/predict/
* Útil para validar se o modelo reconhece quedas corretamente

### 5. Pós-processamento temporal (suavização de resultados)

make postprocess

* Gera vídeos filtrados \*\_filtrado.mp4
* Log detalhado em: runs/detect/falls.log

### 6. Rodar aplicação principal (produção)

make local

**RSTP**

make rtsp

* Processa todos os vídeos em ./data/
* Salva vídeos anotados em runs/detect/annotated/
* Gera snapshots em runs/detect/snapshots/
* Loga alertas JSON no terminal e em falls.log

Parar execução:

make stop

Ver status da porta:

make status

### 7. Exportar modelo para ONNX (integração com outros sistemas)

make yolo-export-onnx

* Modelo exportado em: models/falls\_yolov8.onnx

### 8. Manutenção e Utilitários

* Limpar runs antigos:

make clean-runs

* Limpar snapshots antigos:

make clean-snapshots

* Formatador automático (Black + Ruff):

make fmt

* Checagem de estilo e lint:

make lint

## 📌 Estrutura do Projeto

* app.py → aplicação principal
* config.yaml → configuração global
* scripts/ → scripts auxiliares (dataset, pós-processamento, etc.)
* runs/ → saídas de treino, predição e snapshots
* datasets/ → datasets convertidos (YOLOv8)
* utils/ → métricas e logs
* signals/ → alertas e eventos
* privacy/ → funções de anonimização (ex: blur)

## ✅ Fluxo resumido (para rodar rápido)

1. make install && source .venv/bin/activate
2. make dataset-cvat ZIP=./export.zip OUT=./datasets/saida
3. make yolo-train
4. make yolo-predict SRC=./data/video.mp4
5. make run

## 📷 Boas práticas de coleta de dados

Para melhorar a qualidade do modelo:

* **Iluminação variada:** grave em ambientes claros e escuros.
* **Ângulos de câmera diferentes:** de cima, lateral e na altura da cintura.
* **Diversidade de cenários:** sala, banheiro, quarto, corredores.
* **Perfis de pessoas distintos:** diferentes alturas, pesos, roupas.
* **Casos positivos e negativos:** incluir quedas reais e atividades normais (sentar, deitar, pegar objeto).
* **Movimentos variados:** quedas para frente, para trás, de lado.
* **Múltiplas pessoas em cena:** simula situações reais com cuidadores ou visitas.

Esses fatores aumentam a robustez do modelo e reduzem falsos positivos.

## 📋 Checklist para gravação de vídeos de treino

Antes de gravar, confirme os pontos abaixo:

Essa checklist ajuda a manter consistência e qualidade no dataset de treinamento.

## 📊 Métricas ideais para dataset

Para alcançar boa performance do modelo:

* **Quantidade mínima de vídeos:** pelo menos 100 vídeos anotados.
* **Proporção quedas x normais:** ~40% quedas, 60% atividades normais.
* **Duração média por vídeo:** 10 a 30 segundos.
* **Frames por segundo (FPS):** 5 a 12 FPS (evita redundância e acelera treino).
* **Número de imagens totais:** > 10.000 frames anotados.
* **Balanceamento:** garantir que todos os tipos de quedas estejam representados.
* **Validação:** ao menos 20% dos dados devem ser reservados para validação/teste.

Essas métricas servem como referência inicial — quanto maior e mais diverso o dataset, mais robusto será o modelo.

## 🧭 Plano de Expansão do Dataset (12 semanas)

### Objetivo

Aumentar robustez e reduzir falsos positivos/negativos por meio de coletas iterativas, *active learning* e validações mensais, até alcançar métricas-alvo: **mAP@50 ≥ 0.85**, **Recall ≥ 0.90** para classe *Queda*, **FPR ≤ 3%** em cenários reais.

### Visão geral de fases

1. **S0 – Linha de base (semana 0)**
   * Treino com dataset atual.
   * Avaliar em *hold-out* doméstico (vídeos não vistos) + 3 vídeos reais.
   * Registrar métricas no runs/ e planilha.
2. **S1–S3 – Coleta controlada + Negativos difíceis**
   * 50–80 novos vídeos/semana (40% quedas simuladas, 60% ADLs).
   * Foco em **hard negatives** (ex.: sentar brusco, ajoelhar, pegar objeto, deitar).
   * Anotar no CVAT → make dataset-cvat → make yolo-train → make yolo-predict → make postprocess.
   * Medir evolução (mAP/Recall/FPR) e registrar deltas.
3. **S4–S6 – Ambientes novos + Perfis variados**
   * Novas residências; variação de iluminação (noite/dia), roupas, acessórios.
   * Multipessoa: cuidador + idoso, visitas.
   * 100–150 vídeos/semana; manter 20–25% para *val/test* não vistos.
4. \*\*S7–S9 – \*\****Active Learning***
   * Rodar modelo atual em **vídeos não anotados** (make run).
   * Selecionar **frames de alta incerteza** (0.4–0.6) e **falsos alertas** do mundo real.
   * Priorizar anotação desses casos (reduz label effort e maximiza ganho).
   * Re-treinar quinzenalmente.
5. **S10–S12 – Consolidação + Stress tests**
   * Cenas extremas: queda parcial fora de quadro, oclusão por móvel, câmera vibrando.
   * Avaliar em **cenários de implantação** (câmera fixa do cliente piloto).
   * *Freeze* do conjunto de validação final para relatório.

### Critérios de entrada/saída por iteração

* **Entrada:** ≥ 50 novos vídeos brutos; diversidade ≥ 3 ambientes; ≥ 10 *hard negatives*.
* **Saída:** mAP@50, Recall, Precision, FPR e confusão; *changelog* do dataset e do modelo.

### Métricas-alvo por marcos

* **M1 (semana 4):** mAP@50 ≥ 0.78, Recall ≥ 0.85, FPR ≤ 6%
* **M2 (semana 8):** mAP@50 ≥ 0.82, Recall ≥ 0.88, FPR ≤ 4%
* **M3 (semana 12):** mAP@50 ≥ 0.85, Recall ≥ 0.90, FPR ≤ 3%

### Estratégias técnicas

* **Ouro de validação (golden set):** 100–200 clipes imutáveis para acompanhar evolução.
* **Mining de negativos difíceis:** coletar ativamente cenários que geraram falso alerta e **rotular como não-queda**.
* **Aumento de dados (augmentation):** hflip, brightness/contrast, gaussian noise, motion blur, affine leve; evitar distorção que mude a semântica de queda.
* **Balanceamento:** *oversampling* de quedas raras (ex.: escorregar no banheiro), *undersampling* de padrões redundantes.
* **Janela temporal:** se usar classificador temporal, garantir clipes com 2–3 s antes/depois do pico do evento.

### Operação no CVAT

* **Templates de rótulo:** Pessoa1, Queda (padronizar atributos se necessário).
* **Revisão em 2 passos:** (1) anotador, (2) revisor aprova/rejeita.
* **Diretrizes de caixa:** incluir corpo inteiro; se parcialmente fora, seguir pessoa dominante; padronizar tolerâncias.
* **Qualidade:** amostrar 10% das tarefas para *spot-check* semanal (aceite ≥ 95%).

### Versionamento e governança

* **SemVer do dataset:** datasets/queda\_caso-<major>.<minor>; log em DATASET\_CHANGELOG.md.
* **Snapshot imutável:** export ZIP do CVAT + data.yaml versionado por tag Git.
* **Privacidade:** faces desfocadas com privacy/blur.py quando dados vierem de lares reais; documentação de consentimento.

### Avaliação e relatórios

* **Relatório quinzenal:** incluir curvas PR, matriz de confusão, top 10 erros, exemplos visuais (snapshots).
* **Benchmark cruzado:** medir no *golden set* + 2 casas nunca vistas.
* **Alarmes operacionais:** acompanhar falls.log e taxa de alertas/hora em cenários reais.

### Recursos e logística

* **Equipe:** 1 coordenador de dados, 2 anotadores, 1 revisor técnico.
* **Armazenamento:** 1–2 TB (vídeos brutos + export YOLO + runs).
* **Automação:** make dataset-cvat, make yolo-train, make yolo-predict, make postprocess.
* **Segurança:** criptografar *at rest* (ex.: FileVault) e acesso por grupo restrito.

### Checklist por ciclo

* Coleta planejada (ambientes, perfis, hard negatives)
* Anotação e revisão no CVAT
* Conversão e split do dataset
* Treino + validação
* Seleção de frames incertos (active learning)
* Pós-processamento e avaliação
* Atualização do DATASET\_CHANGELOG.md
* Registro das métricas em relatório quinzenal

## 🔎 10. Active Learning – Mineração de Incertezas

### 10.1. O que faz

* Varre vídeos/imagens em ./data
* Usa o modelo best.pt
* Salva snapshots com confiança entre **LOW** e **HIGH**
* Gera meta.json com metadados

Saída: active\_learning/uncertain/<nome\_do\_video>/\*.jpg

### **10.2. Como rodar**

make active-learning

**Com parâmetros customizados:**

make active-learning \

  ACTIVE\_OUT=active\_learning/uncertain\_v2 \

  ACTIVE\_LOW=0.35 \

  ACTIVE\_HIGH=0.65 \

  ACTIVE\_STRIDE=2 \

  ACTIVE\_MAX=75

### 10.3. Fluxo com CVAT

1. Rodar make active-learning
2. Importar snapshots no CVAT
3. Anotar apenas esses frames
4. Exportar YOLO e mesclar ao dataset
5. Re-treinar make yolo-train
6. Validar make yolo-predict
7. Atualizar DATASET\_CHANGELOG.md

### 10.4. Dicas

* Ajuste LOW/HIGH conforme volume de frames
* Use stride maior para vídeos longos
* Limite snapshots redundantes com ACTIVE\_MAX

### 10.5. Requisitos

* Modelo treinado em runs/detect/train/weights/best.pt
* Dependências: ultralytics, opencv-python, numpy

### 10.6. Troubleshooting

* **Nenhum snapshot:** confira se a classe Queda existe no modelo
* **Classe não encontrada:** verifique o data.yaml
* **Poucos frames:** aumente faixa de incerteza
* **Saída vazia:** verifique vídeos em ./data