**Tech Challenge 5 – Fluxo de Desenvolvimento**

Outputs dos vídeos analisados:  
Video 1 (42 seg) – <https://drive.google.com/file/d/1hE9Z9Eth6dKOkb6ZsppfkeL9-KRtorp5/view?usp=sharing>

Video 2 (3 seg) - <https://drive.google.com/file/d/1aPo-f2PBqLmP1UEtTl3uk1KjUFQkBOxS/view?usp=sharing>

**Passo a Passo:  
1 – Identificação do Problema**

A FIAP VisionGuard deseja criar uma funcionalidade para identificar objetos cortantes (como facas, tesouras e similares) em fluxos de vídeo de câmeras de segurança, visando aumentar a segurança de estabelecimentos. O objetivo é usar Inteligência Artificial para detectar automaticamente esses objetos e emitir alertas preventivos à central de segurança, contribuindo para a redução de incidentes e resposta rápida em situações de risco. Para validar a viabilidade da solução, a empresa pretende construir um MVP que envolva desde a coleta e anotação de um dataset até o desenvolvimento do sistema de alertas.

**2 – Solução Inicial Proposta:**

Porque solução inicial proposta? Pois esse não foi o modelo real utilizado e só foi utilizado para fins de neuroplasticidade no momento que eu utilizasse a solução de negócios ideal para esse tipo de problema em pequena escala, no caso a Roboflow, com ela foi possível facilitar e gerenciar todo o processo de anotação de imagens até o deploy do modelo. Roboflow é uma plataforma online que simplifica a criação de pipelines de visão computacional, oferecendo ferramentas para anotar e gerenciar datasets.  
Também disponibiliza recursos para data augmentation, conversão de formatos e deploy de modelos, facilitando todo o ciclo de vida de projetos de IA.

Para atender ao objetivo de identificar objetos cortantes em vídeos de câmeras de segurança, a solução inicial proposta consistia em:

1. **Dataset Público curado e Anotações com a lib Labellmg**:
   * Coletar ou buscar imagens que contenham facas, tesouras e outros objetos potencialmente cortantes, cobrindo diferentes ângulos, iluminação e cenários. Instalar a lib Labelmg para realizar as anotações dos objetos.
   * Tela de celular com publicação numa rede social

     O conteúdo gerado por IA pode estar incorreto.Anotar as imagens no formato YOLO (ou equivalente), definindo bounding boxes ao redor dos objetos de interesse.

Labellmg

1. **Seleção do Modelo:**
   * Utilizar o **YOLOv8m** (a versão “médio”) da biblioteca “ultralytics” como ponto de partida por equilibrar velocidade e acurácia, também utilizei a versão nano do YOLO que é o **YOLOv8n.pt**, mas não obtive resultados relevantes com o tamanho do dataset de apenas 300 imagens de objetos cortantes diferentes.
   * Esse modelo já vem pré-treinado em um conjunto de dados genérico (COCO), sendo então refinado (fine-tuning) com o dataset específico de objetos cortantes.
2. **Treinamento Local:**
   * Executar o treinamento do modelo YOLOv8m em uma máquina local ou servidor com GPU disponível.
   * Ajustar hiperparâmetros (épocas, taxa de aprendizado, tamanho de batch, etc.) visando maximizar a precisão de detecção e reduzir falsos negativos/positivos.
   * Acompanhar métricas (mAP, precision, recall) para medir a evolução do modelo ao longo das épocas de treinamento.
3. **Validação e Testes:**
   * Separar parte das imagens para validação e teste, garantindo que o modelo não sofra overfitting e apresente bom desempenho em cenários reais.
   * Avaliar resultados em cenários variados (diferentes iluminações e ângulos) para assegurar robustez.
4. **Deploy Simples (MVP):**
   * Integrar o modelo treinado a um script que processa os frames de vídeo em tempo real (ou próximo disso), identificando facas e tesouras e desenhando bounding boxes nas detecções.
   * Em caso de detecção, disparar alertas por e-mail à central de segurança. Neste caso utilizei a API da Open AI para gerar o conteúdo do email e o envio do disparo é realizado com a biblioteca smtplib.

Esta solução local com YOLOv8m fornece uma base sólida para testar a viabilidade da detecção de objetos cortantes, permitindo ajustes e melhorias contínuas conforme a empresa FIAP VisionGuard evoluir seu protótipo. Porém uma vez que a viabilidade foi testada, precisamos de uma solução robusta que consiga suportar diferentes modelos e versões em produção e oferecer métricas para monitoramento destes modelos.

**3 – A escolha da Roboflow:**

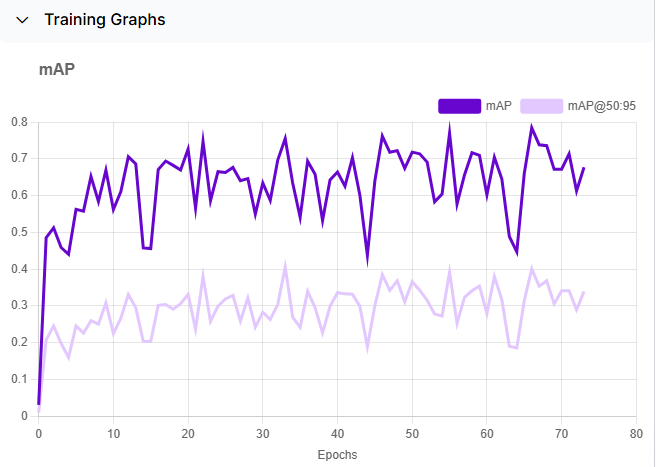
A Roboflow é uma solução integrada que unifica o processo de coleta, anotação e treinamento de modelos de visão computacional, otimizando a criação de sistemas de detecção de objetos cortantes (facas, tesouras etc.) e outros usos. Com recursos de gestão centralizada de dados, anotação simplificada, conversão de formatos, data augmentation e integração com bibliotecas consagradas, a plataforma possibilita um desenvolvimento ágil de modelos robustos e custo para equipes pequenas muito acessível. Além disso, oferece hospedagem e escala para inferência em produção, e pode ser aplicada em diversos cenários, desde segurança até agricultura de precisão. Essa combinação de praticidade, robustez e versatilidade torna a Roboflow uma escolha ideal para projetos como o da FIAP VisionGuard, que buscam criar MVPs confiáveis e de fácil expansão.

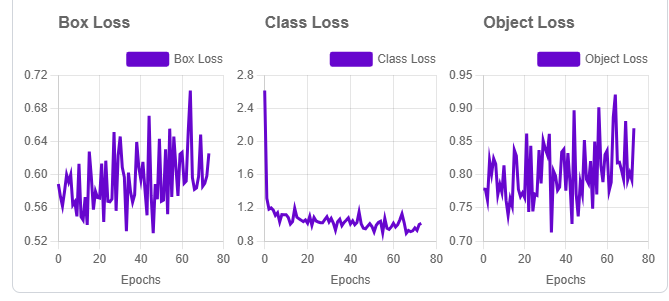
**4 – Passo a Passo no Roboflow:**

**1. Criação ou Importação do Dataset na Roboflow**

1. **Criar um Projeto na Roboflow**
   * Acesse app.roboflow.com e crie um novo projeto (por exemplo, “Deteccao-Objetos-Cortantes”), neste caso o projeto **KnifeDetectionV4**.
   * Selecione a opção de **Detecção de Objetos** (Object Detection).
2. **Importar as 3.000 Imagens em Formato COCO**
   * Ao configurar o upload, escolha o tipo de anotação “COCO” e aponte para a pasta onde estão o arquivo .json (com as anotações) e as imagens.
   * A Roboflow lerá a estrutura e organizará automaticamente as imagens. Caso seja necessário, você pode escolher a proporção de divisão em **treino**, **validação** e **teste**.
3. **Verificar e Revisar Anotações**
   * Após o upload, é recomendável revisar algumas imagens para garantir que as bounding boxes e as classes (“knife”, “scissor”, etc.) estejam corretas.
   * Se precisar fazer correções, utilize o editor interno de anotações da Roboflow.
4. **Configurar Data Augmentation** (Opcional, mas altamente recomendado)
   * A Roboflow oferece opções para rotação, flip, alteração de brilho/contraste, entre outros.
   * Essa etapa ajuda o modelo a lidar melhor com variações de ângulo e iluminação, especialmente relevantes em cenários de segurança.

**2. Selecionando o Modelo YOLO NAS**

1. **Motivos para Escolher YOLO NAS**
   * **Desempenho Superior**: YOLO NAS (desenvolvido pela Deci) utiliza *Neural Architecture Search*, o que pode levar a detecções mais precisas e rápidas em comparação com versões tradicionais do YOLO.
   * **Redução de Tempo**: Observa-se que a Roboflow cuidando do treinamento em nuvem é capaz de processar 3.000 imagens em cerca de 9 horas, enquanto em um setup local, um modelo YOLO com apenas 300 imagens demorou 13 horas (diferença significativa que comprova a vantagem de usar a infraestrutura otimizada da Roboflow).
   * **Compatibilidade**: A Roboflow já fornece integrações para YOLO NAS, facilitando o deploy posterior.
2. **Configuração na Plataforma**
   * Na interface do seu projeto, acesse a aba de treino (Train New Model).
   * Selecione **YOLO NAS** como a arquitetura desejada. A Roboflow pode oferecer diferentes variações (ex.: YOLO NAS “S”, “M” etc.). Escolha a que melhor se adequa ao tamanho do dataset e aos recursos disponíveis.
3. **Definição de Parâmetros**
   * **Epochs**: De acordo com o tamanho do dataset, é comum usar entre 50 e 100 épocas. Neste caso utilizei 73 epochs de treinamento, como pode ver nos gráficos abaixo.



Com base nesses gráficos de loss (Box Loss, Class Loss e Object Loss) observados durante o treinamento:

* **Box Loss**: Embora apresente oscilações entre aproximadamente 0,52 e 0,72, mantém-se numa faixa relativamente estável ao longo das épocas. Isso indica que o modelo está ajustando a posição e o tamanho das bounding boxes de forma consistente, mas com variações possivelmente relacionadas à diversidade do dataset ou às variações de mini-batches.
* **Class Loss**: Cai drasticamente nas primeiras épocas (de algo em torno de 2,8 para menos de 1,2), estabilizando-se em valores próximos a 1,0 ou abaixo disso. Essa redução rápida mostra que o modelo rapidamente aprendeu a classificar (identificar a classe do objeto), sugerindo um bom aprendizado inicial da tarefa de classificar objetos cortantes.
* **Object Loss**: Oscila entre 0,70 e 0,95, indicando que o modelo ainda varia na confiança sobre a presença de objetos em determinadas regiões. Contudo, a permanência nessa faixa sugere que o modelo não está entrando em overfitting extremo, mas sim refinando consistentemente a detecção (onde e se um objeto está presente).

Em suma, apesar de haver flutuações em todas as métricas, o comportamento geral de queda ou estabilização dos valores indica que o treinamento está evoluindo de forma razoavelmente adequada, sem sinais claros de estagnação ou sobreajuste.

* + **Image Size**: Pode variar (640, 720 ou 1024), dependendo do poder de processamento e da diversidade dos objetos.
  + **Batch Size**: Defina de acordo com a GPU utilizada; a Roboflow automaticamente adapta se for o caso.

**3. Processo de Treinamento na Roboflow**

1. **Iniciar o Treinamento**
   * Clique em **Train** e aguarde a Roboflow provisionar sua infraestrutura.
   * O sistema exibirá estatísticas e logs ao longo do treinamento.
2. **Acompanhar Métricas**
   * A plataforma apresenta em tempo real ou ao final das épocas métricas como mAP@50, precision, recall e gráficos de loss.
   * Verifique se o modelo está evoluindo bem na detecção dos objetos cortantes. Caso haja baixo desempenho, considere rever anotações ou intensificar augmentations.
3. **Tempo Estimado**
   * Em geral, com 3.000 imagens, a Roboflow conclui o treinamento em cerca de **9 horas**, dependendo do plano e dos recursos de GPU usados. Esse resultado demonstra clara vantagem em relação ao **setup local** (que levou 13 horas para apenas 300 imagens, evidenciando gargalos de hardware ou de configuração local).
4. **Resultado e Export**
   * Ao final, você terá acesso aos **checkpoints** e métricas finais. A Roboflow destaca a versão de “best model” obtida no processo.

**5. Deploy e Inferência Local com predict()**

Após o treinamento, você pode tanto:

* **Manter o modelo hospedado na Roboflow**, fazendo inferências via API.
* **Baixar** o modelo para rodar localmente, integrando com sua aplicação de segurança.

1. **Download do Modelo**
   * Acesse a **Model Version** gerada (ex.: “Version 1 - YOLO NAS”) e clique em **Download**.
   * Selecione **“Inference Script”** ou as instruções para rodar via script Python.
   * A Roboflow fornecerá arquivos e dependências (por exemplo, super-gradients se for YOLO NAS) e um snippet para uso local.
2. **Instalar Dependências**
   * Caso esteja rodando localmente, instale as bibliotecas necessárias (p. ex. pip install super-gradients opencv-python-headless ou conforme indicado).
3. **Utilizando o Método predict()**
   * Tipicamente, um **exemplo de script** para inferência local (usando API da Roboflow) seria:

python

CopyEdit

import roboflow

rf = roboflow.Roboflow(api\_key="SUA\_API\_KEY")

project = rf.workspace().project("PROJETO\_ID")

model = project.version("VERSAO").model

# Inferir sobre uma imagem local

prediction = model.predict("caminho/para/imagem\_com\_faca.jpg", hosted=False)

resultado = prediction.json()

print(resultado)

* + Cada “prediction” traz as coordenadas (x, y, width, height) e a confiança de detecção.

1. **Aplicando em Vídeos**
   * Se você deseja rodar em fluxo de vídeo localmente, basta:
     1. Abrir cada frame via OpenCV (cv2.VideoCapture).
     2. Salvar o frame em um arquivo temporário ou buffer.
     3. Chamar model.predict(frame\_temp.jpg).
     4. Desenhar as bounding boxes no frame e salvar em um VideoWriter.
   * Assim, você pode **detectar objetos cortantes em tempo real** ou próximo disso, dependendo da latência de inferência/transferência de dados.

**5. Vantagens Adicionais**

1. **Escalabilidade**
   * Ao lidar com 3.000 imagens, a Roboflow lida melhor com aumentos de dataset, comparado à limitação de hardware local.
2. **Redução de Tempo**
   * Conforme relatado, o treinamento completo em 9 horas para 3.000 imagens é um ganho expressivo, sobretudo diante das 13 horas de treino local para apenas 300 imagens.
3. **Infraestrutura Gerenciada**
   * A Roboflow facilita a criação de **novas versões** do modelo, permitindo retrain rápido quando houver mais imagens ou novos tipos de objetos de interesse (por exemplo, garrafas quebradas, estiletes, etc.).
4. **Precisão e Desempenho**
   * O YOLO NAS se beneficia da tecnologia NAS para encontrar configurações de rede otimizadas. Em tarefas com objetos pequenos e variados (como facas em diferentes posições), pode alcançar taxas de detecção superiores a outras arquiteturas YOLO tradicionais.

**Conclusão**

Ao adotar o **YOLO NAS** na Roboflow para um dataset de **3.000 imagens** em formato COCO, a FIAP VisionGuard consegue:

* **Economizar tempo** de treinamento (cerca de 9 horas na nuvem vs. 13 horas para apenas 300 imagens localmente).
* **Obter alta acurácia** na detecção de objetos cortantes.
* **Ter flexibilidade** para exportar e rodar inferência localmente com o método predict(), integrado a sistemas de vigilância que precisam detectar situações de risco em tempo real.

Esse conjunto de vantagens faz da Roboflow e do YOLO NAS uma combinação particularmente atrativa para projetos de segurança que demandam escalabilidade, precisão e rapidez na detecção de itens potencialmente perigosos.

**5 – Tipos de deploy:**

A Roboflow disponibiliza **várias opções de deploy** para o modelo treinado, permitindo que você escolha a solução que melhor se adapta ao seu cenário, seja em nuvem, localmente ou em dispositivos de borda (edge). Abaixo estão as principais:

1. **Hosted Image Inference**
   * Permite rodar o modelo em um endpoint **hosteado** pela Roboflow, executando previsões em imagens de forma simples e rápida.
   * Não exige configuração de servidores próprios, mas requer **internet** para enviar as imagens e receber os resultados.
2. **Hosted Video Inference**
   * Indicado para processar **arquivos de vídeo completos** usando a infraestrutura da Roboflow na nuvem.
   * Ideal para cenários em que **não** se necessita de resposta em tempo real, mas sim de um processamento em lote (por exemplo, revisar gravações de segurança após o fato).
3. **Self-Hosted Inference**
   * Permite rodar o modelo em sua **própria máquina** ou em provedores de nuvem como AWS, GCP ou Azure.
   * Oferece maior **flexibilidade** e possibilidade de **processar em tempo real**, já que você controla o ambiente (GPUs locais ou VMs de nuvem).
4. **Edge Device**
   * Voltado para execução em dispositivos de **borda**, como **NVIDIA Jetson** ou outros embarcados.
   * Escalonável via **Docker**, facilitando a implantação em múltiplas unidades, caso você precise de um sistema de segurança distribuído em várias câmeras.

**6. Roadmap de Produto e Próximas Etapas**

Para evoluir o MVP de detecção de objetos cortantes em um sistema de vigilância até uma solução mais abrangente e integrada, sugerimos o seguinte roadmap de produto, contemplando tecnologias e casos de uso diversos:

1. **Armazenamento das Detecções em MongoDB**
   * **Objetivo**: Persistir os registros de detecção (câmera, tipo de objeto, horário, confiança, etc.) em uma base de dados NoSQL para consulta e análise futura.
   * **Caso de Uso**:
     + **Histórico de Eventos**: Filtrar e auditar eventos de segurança por período, câmera ou tipo de objeto.
     + **Análises Estatísticas**: Identificar padrões de detecção (horários de maior incidência, regiões mais críticas, etc.).
   * **Tecnologias**: MongoDB para armazenamento e consultas rápidas de dados semiestruturados.
2. **Exposição dos Dados em uma Aplicação**
   * **Objetivo**: Criar uma camada de apresentação (web ou desktop) que permita a equipe de segurança visualizar em tempo real as detecções e relatórios históricos.
   * **Caso de Uso**:
     + **Dashboard de Monitoramento**: Mostrar um mapa ou lista de câmeras em tempo real, indicando alertas recentes.
     + **Gerenciamento de Alertas**: Operadores podem marcar alertas como “verificados”, “em investigação”, etc.
   * **Tecnologias**: Frameworks web (React ou Angular) ou soluções desktop (Electron) para interfaces mais robustas.
3. **RPA para Inserir Informações em Sistemas Legados**
   * **Objetivo**: Automatizar o envio das detecções e alertas para plataformas de terceiros ou sistemas legados que não possuem API aberta.
   * **Caso de Uso**:
     + **Atualização de Logs em Sistemas Antigos**: Inserir eventos de segurança em softwares antigos de monitoramento ou bancos de dados acessados apenas via interface gráfica.
     + **Geração de Tickets**: Criar chamados automaticamente em sistemas de ITSM (ex.: ServiceNow, GLPI) sem depender de um operador manual.
   * **Tecnologias**: Plataformas de RPA (UiPath, Power Automate ou até mesmo Python) para simular ações humanas em telas/sistemas que não tenham integração moderna.
4. **Evolução para um LLM (Large Language Model)**
   * **Objetivo**: Utilizar modelos de linguagem natural para enriquecer a análise de dados ou fornecer suporte em processos decisórios.
   * **Caso de Uso**:
     + **Resumo e Contextualização de Alertas**: O LLM pode gerar relatórios completos, explicando o risco potencial de cada detecção e sugerindo ações de segurança.
     + **Automação de Atendimento Interno**: Chatbot interno para a equipe de segurança, consultando registros de detecções e histórico do MongoDB para responder perguntas (ex.: “Quantas detecções de facas ocorreram nas últimas 48 horas?”).
   * **Tecnologias**: Modelos open-source (Llama 3, Granite, etc.) ou APIs de provedores (OpenAI, Azure, etc.), integrados ao sistema de vigilância.

**Resumo dos Benefícios**

* **MongoDB**: Persistência e consulta ágil de registros de detecções e alertas.
* **Aplicação**: Maior usabilidade e monitoramento centralizado em tempo real.
* **RPA**: Automação de procedimentos manuais em sistemas legados sem necessidade de desenvolvimento de integrações complexas.
* **LLM**: Inteligência adicional para resumir informações, gerar relatórios ou oferecer respostas contextuais sobre dados de segurança.

Esse roadmap possibilita uma **arquitetura escalável**, na qual cada componente (deteção de objetos, armazenamento, integração, insights via LLM) se complementa, proporcionando uma solução mais robusta e eficiente para a FIAP VisionGuard (ou qualquer empresa) que deseje aprimorar sua estratégia de segurança e prevenção de incidentes.