VERTEX

# Aula 3
# Uso de Modelos de Visão Computacional em Pipelines Reais

Softex

# Conteúdo da Aula

1. Introdução a Conceitos Básicos

# Conteúdo da Aula

1. Introdução a Conceitos Básicos
2. Como Usar um Modelo de Visão Computacional

# Conteúdo da Aula

1. Introdução a Conceitos Básicos
2. Como Usar um Modelo de Visão Computacional
3. Considerações-Chave para o Uso de Modelos

# Conteúdo da Aula

1. Introdução a Conceitos Básicos
2. Como Usar um Modelo de Visão Computacional
3. Considerações-Chave para o Uso de Modelos
4. Demonstração Prática

# Conteúdo da Aula

1. Introdução a Conceitos Básicos
2. Como Usar um Modelo de Visão Computacional
3. Considerações-Chave para o Uso de Modelos
4. Demonstração Prática
5. Definição da Tarefa Individual

# Introdução a Conceitos Básicos

- **Modelo de Visão Computacional:** Um modelo computacional, tipicamente uma rede neural profunda, treinado para interpretar e compreender informações visuais do mundo. O objetivo é automatizar tarefas que o sistema visual humano executa, como reconhecimento, identificação e medição.

# Introdução a Conceitos Básicos

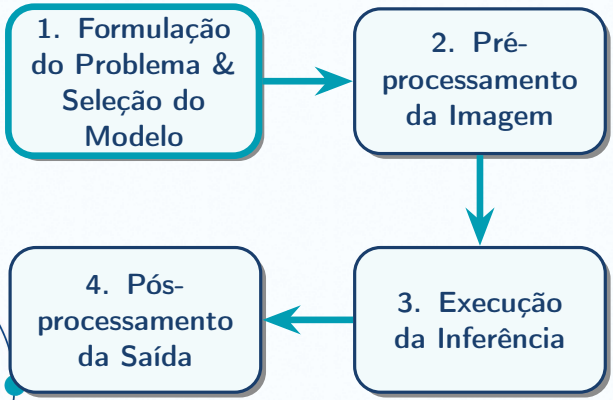
- **Modelo de Visão Computacional:** Um modelo computacional, tipicamente uma rede neural profunda, treinado para interpretar e compreender informações visuais do mundo. O objetivo é automatizar tarefas que o sistema visual humano executa, como reconhecimento, identificação e medição.
- **Inferência:** Processo de usar um modelo treinado para prever dados novos. No pipeline de visão, esta é a etapa central: uma imagem ou quadro de vídeo é passado ao modelo para obter uma saída (ex.: rótulos de classe ou coordenadas de caixas delimitadoras).

# Introdução a Conceitos Básicos

- **Modelo de Visão Computacional:** Um modelo computacional, tipicamente uma rede neural profunda, treinado para interpretar e compreender informações visuais do mundo. O objetivo é automatizar tarefas que o sistema visual humano executa, como reconhecimento, identificação e medição.
- **Inferência:** Processo de usar um modelo treinado para prever dados novos. No pipeline de visão, esta é a etapa central: uma imagem ou quadro de vídeo é passado ao modelo para obter uma saída (ex.: rótulos de classe ou coordenadas de caixas delimitadoras).
- **Pipeline:** Um fluxo de trabalho automatizado de ponta a ponta. Integra etapas como ingestão de dados, pré-processamento, inferência, pós-processamento e, por fim, acionamento de uma lógica de negócio ou ação.

# Como Usar um Modelo de Visão Computacional

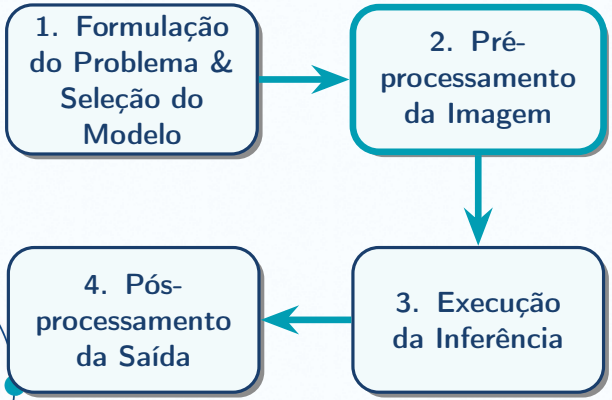
Um fluxo típico de implantação envolve várias etapas sequenciais:




- Definir a tarefa específica (ex.: classificação, detecção, segmentação). Selecionar um modelo pré-treinado adequado (YOLOv8, ResNet, UNet) ou planejar um treinamento customizado.

# Como Usar um Modelo de Visão Computacional

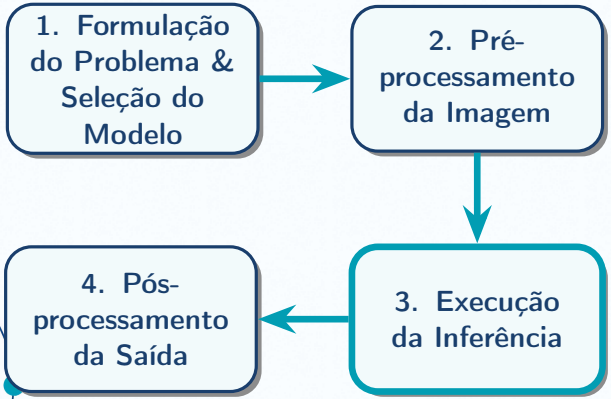
Um fluxo típico de implantação envolve várias etapas sequenciais:




- Imagens brutas precisam ser convertidas no formato esperado pelo modelo. Passos comuns: redimensionamento, normalização dos pixels (ex.: $[0, 1]$), conversão para tensor.

# Como Usar um Modelo de Visão Computacional

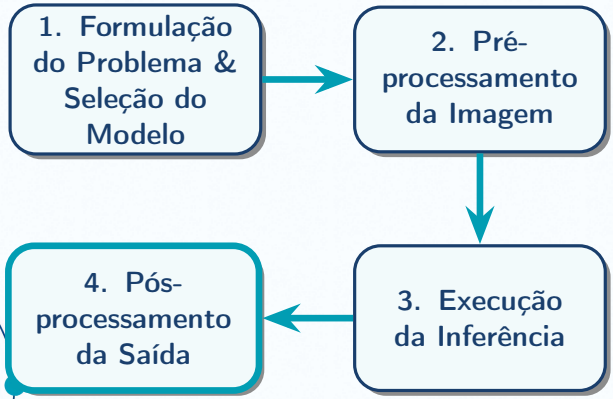
Um fluxo típico de implantação envolve várias etapas sequenciais:




- Passar o tensor pré-processado ao modelo. O modelo faz o forward pass e retorna a saída bruta, como vetores de scores ou caixas previstas.

# Como Usar um Modelo de Visão Computacional

Um fluxo típico de implantação envolve várias etapas sequenciais:




- Converter a saída bruta em um formato utilizável. Exemplo: na detecção, aplicar limiar de confiança e supressão não-máxima (NMS) para filtrar detecções fracas ou sobrepostas.

# ⚖ Requisitos Importantes para a Implantação da Solução

1. **Desempenho (Latência vs. Precisão):** Trade-off entre complexidade do modelo e velocidade. Aplicações em tempo real exigem baixa latência.

# ⚖ Requisitos Importantes para a Implantação da Solução

1. **Desempenho (Latência vs. Precisão):** Trade-off entre complexidade do modelo e velocidade. Aplicações em tempo real exigem baixa latência.
2. **Deriva de Dados e Conceitos:** O desempenho cai quando os dados do mundo real diferem dos dados de treino (iluminação, ângulos, novos objetos). Necessário monitoramento e re-treinamento contínuo.

# ⚖ Requisitos Importantes para a Implantação da Solução

1. **Desempenho (Latência vs. Precisão):** Trade-off entre complexidade do modelo e velocidade. Aplicações em tempo real exigem baixa latência.
2. **Deriva de Dados e Conceitos:** O desempenho cai quando os dados do mundo real diferem dos dados de treino (iluminação, ângulos, novos objetos). Necessário monitoramento e re-treinamento contínuo.
3. **Aceleração por Hardware:** CPU muitas vezes insuficiente. GPUs, TPUs ou hardwares especializados (Intel Movidius, NVIDIA Jetson) são essenciais para alcançar a velocidade necessária.

# ⚖ Requisitos Importantes para a Implantação da Solução

1. **Desempenho (Latência vs. Precisão):** Trade-off entre complexidade do modelo e velocidade. Aplicações em tempo real exigem baixa latência.
2. **Deriva de Dados e Conceitos:** O desempenho cai quando os dados do mundo real diferem dos dados de treino (iluminação, ângulos, novos objetos). Necessário monitoramento e re-treinamento contínuo.
3. **Aceleração por Hardware:** CPU muitas vezes insuficiente. GPUs, TPUs ou hardwares especializados (Intel Movidius, NVIDIA Jetson) são essenciais para alcançar a velocidade necessária.
4. **Robustez a Casos Extremos:** O modelo deve lidar com baixa iluminação, desfoque, oclusão ou clima adverso.

# Demonstração Prática

**Objetivo:** Desenvolver um script em Python que implemente um pipeline completo de visão computacional para contagem e classificação automática de veículos em monitoramento de tráfego.

1. Uso da biblioteca **OpenCV** para carregar e processar quadros de vídeo.

# Demonstração Prática

**Objetivo:** Desenvolver um script em Python que implemente um pipeline completo de visão computacional para contagem e classificação automática de veículos em monitoramento de tráfego.

1. Uso da biblioteca **OpenCV** para carregar e processar quadros de vídeo.
2. Um modelo pré-treinado **YOLOv8** será carregado via `ultralytics` para detecção de veículos, obtendo caixas delimitadoras, IDs de classe (carro, caminhão, ônibus, moto) e scores de confiança.

# Demonstração Prática

**Objetivo:** Desenvolver um script em Python que implemente um pipeline completo de visão computacional para contagem e classificação automática de veículos em monitoramento de tráfego.

1. Uso da biblioteca **OpenCV** para carregar e processar quadros de vídeo.
2. Um modelo pré-treinado **YOLOv8** será carregado via `ultralytics` para detecção de veículos, obtendo caixas delimitadoras, IDs de classe (carro, caminhão, ônibus, moto) e scores de confiança.
3. Integração de um **algoritmo de rastreamento** (DeepSORT ou ByteTrack) para manter identidades consistentes ao longo dos quadros, garantindo contagem única.

# Demonstração Prática

**Objetivo:** Desenvolver um script em Python que implemente um pipeline completo de visão computacional para contagem e classificação automática de veículos em monitoramento de tráfego.

1. Uso da biblioteca **OpenCV** para carregar e processar quadros de vídeo.
2. Um modelo pré-treinado **YOLOv8** será carregado via `ultralytics` para detecção de veículos, obtendo caixas delimitadoras, IDs de classe (carro, caminhão, ônibus, moto) e scores de confiança.
3. Integração de um **algoritmo de rastreamento** (DeepSORT ou ByteTrack) para manter identidades consistentes ao longo dos quadros, garantindo contagem única.
4. O sistema exibirá estatísticas em tempo real: contagem por tipo de veículo, velocidade média estimada, saída de vídeo anotada com caixas, IDs e rótulos de classificação.

# Definição da Tarefa Individual

- **Contador de Pessoas em Tempo Real:** Desenvolver um pipeline com um modelo de detecção de objetos para contar pessoas em um vídeo ao vivo, contabilizando caixas delimitadoras da classe “person”. O sistema deve exibir o número atual de pessoas detectadas em cada quadro em tempo real.