**[slide\_01] [long\_pause]**

Bem-vindos à aula Modelos de Deep Learning para Visão Computacional.

Hoje vamos explorar como técnicas de aprendizado profundo podem ser aplicadas a tarefas de visão computacional, entendendo os principais modelos e exemplos práticos.

**[slide\_02] [long\_pause]**

Aqui está o conteúdo do curso de hoje:

Primeiro, uma introdução a conceitos básicos.

Depois, veremos os diferentes tipos de modelos de visão computacional.

Em seguida, as tarefas mais comuns nessa área.

Faremos uma apresentação do fluxo prático para a implementação de uma solução de visão computacional.

E, por fim, apresentarei a tarefa individual que cada um deverá desenvolver.

**[vignette]**

**[slide\_03] [long\_pause]**

Vamos começar com a Introdução a Conceitos Básicos.

**[short\_pause]**

Visão Computacional é um campo da inteligência artificial que ensina computadores a ver e compreender imagens. Com o uso de imagens digitais e modelos de deep learning, as máquinas podem identificar, classificar e responder a objetos com alta precisão.

**[slide\_04] [short\_pause]**

Deep Learning é um tipo de aprendizado de máquina que utiliza redes neurais artificiais com muitas camadas. Em visão computacional, permite que os modelos aprendam padrões diretamente dos pixels da imagem, indo de formas simples a objetos complexos.

**[slide\_05] [short\_pause]**

As Redes Neurais Convolucionais, ou CNNs, são o principal método de deep learning para análise de imagens. Elas aprendem padrões visuais em etapas, partindo de bordas básicas até estruturas mais detalhadas.

**[slide\_06] [long\_pause]**

Agora, vamos conhecer os Tipos de Modelos de Visão Computacional.

ResNet: usa conexões residuais para treinar redes muito profundas, facilitando o fluxo de gradientes.

**[slide\_07] [short\_pause]**

Vision Transformer, ou ViT: processa imagens como sequências de patches usando blocos transformer e autoatenção.

**[slide\_08] [short\_pause]**

Redes Siamesas: pares de redes gêmeas que aprendem métricas de similaridade entre entradas para tarefas de verificação.

**[slide\_09] [short\_pause]**

Autoencoders: arquiteturas encoder-decoder para aprender representações comprimidas e realizar tarefas generativas.

**[slide\_10] [short\_pause]**

U-Net: também baseada em encoder-decoder, com conexões de atalho, ideal para segmentação semântica e imagens médicas.

**[slide\_11] [short\_pause]**

Redes Adversárias Generativas, conhecidas como GANs: combinam um gerador e um discriminador para criar imagens cada vez mais realistas.

**[slide\_12] [short\_pause]**

E, por fim, os Modelos de Difusão, que geram imagens refinando iterativamente um ruído aleatório até formar a imagem desejada.

**[slide\_13] [long\_pause]**

Vamos agora às Tarefas Comuns em Visão Computacional, começando pela Classificação de Imagens.

Aqui, atribuímos um único rótulo para toda a imagem.

A pergunta central é: “O que há nesta imagem?”.

**[slide\_14] [long\_pause]**

A próxima tarefa é a Detecção de Objetos.

Além de identificar quais objetos estão presentes, precisamos localizar cada um, desenhando caixas delimitadoras.

**[short\_pause]**

A pergunta é: “Quais objetos estão nesta imagem e onde eles estão?”.

**[slide\_15] [long\_pause]**

Seguimos com a Segmentação Semântica.

Aqui, classificamos cada pixel da imagem em uma categoria, sem distinguir diferentes instâncias de uma mesma classe.

**[short\_pause]**

A pergunta é: “A qual categoria pertence cada pixel, como carro, estrada ou céu?”.

**[slide\_16] [long\_pause]**

Agora, Rastreamento de Objetos.

O objetivo é acompanhar o movimento de um objeto específico ao longo de vários quadros de um vídeo.

**[short\_pause]**

Pergunta: “Onde está esse objeto no próximo quadro?”.

**[slide\_17] [long\_pause]**

Próxima tarefa: Geração de Legendas para Imagens, ou Image Captioning.

**[short\_pause]**

Criamos descrições em linguagem natural para o conteúdo da imagem, combinando visão computacional com processamento de linguagem natural.

**[short\_pause]**

Pergunta: “O que está acontecendo nesta imagem?”.

**[slide\_18] [long\_pause]**

Temos também a Estimativa de Esqueleto.

**[short\_pause]**

Ela detecta e estima a pose do corpo humano, mapeando as articulações e os pontos anatômicos principais.

**[short\_pause]**

Pergunta: “Onde estão as articulações e como o corpo está posicionado?”.

**[slide\_19] [long\_pause]**

A seguir, a Detecção de Ações.

**[short\_pause]**

Aqui identificamos e localizamos ações humanas em vídeos, combinando informações espaciais e temporais.

**[short\_pause]**

Pergunta: “Qual ação está sendo executada e onde?”.

**[slide\_20] [long\_pause]**

E, finalmente, o Reconhecimento Óptico de Caracteres, ou OCR.

**[short\_pause]**

Essa tarefa converte imagens de texto em uma sequência de bytes que representam um texto legível por máquina, reconhecendo e extraindo o conteúdo de placas, documentos ou fotos.

**[short\_pause]**

Pergunta: “Qual texto está escrito nesta imagem?”.

**[slide\_21] [long\_pause]**

O objetivo da demonstração plrática é usar o YOLO, através da biblioteca Ultralytics, para detectar objetos em uma foto.

Passo 1 – Instalação: Instalar o pacote Ultralytics YOLO com o comando pip install ultralytics.

**[slide\_22] [short\_pause]**

Passo 2 – Carregamento do Modelo: Importar a classe YOLO, carregar um modelo pré-treinado, por exemplo YOLO('yolov8n.pt'), e escolher o tamanho adequado do modelo, equilibrando velocidade e precisão.

**[slide\_23] [short\_pause]**

Passo 3 – Detecção de Objetos: Executar a inferência em um arquivo de imagem usando o método model.predict().

**[slide\_24] [short\_pause]**

Passo 4 – Análise dos Resultados: Visualizar os objetos detectados com caixas delimitadoras, pontuações de confiança e rótulos de classe.

**[slide\_25] [long\_pause]**

Para concluir, a Tarefa Individual:

Implemente uma segmentação semântica utilizando o modelo pré-treinado Segment Anything Model 2.1, ou SAM 2.1. Aplique o modelo para segmentar objetos em imagens ou quadros de vídeo, gerando máscaras em nível de pixel para diferentes objetos.

**[short\_pause]**

Experimente diferentes métodos de prompting, como pontos, caixas delimitadoras ou geração automática de máscaras, e documente a qualidade da segmentação, o tempo de processamento e o uso de memória, comparando com abordagens tradicionais.

**[vignette]**