Taller: Haciendo Inteligentes las Cámaras

Enseñar cómo integrar inteligencia artificial en cámaras de seguridad, utilizando Python, OpenCV, YOLOv5 y Telegram para detección de personas y alertas en tiempo real.

Arturo Téllez Cortés

Actuario y maestro en ciencias e ingeniería en computación, ambos títulos por la UNAM.

Cuento con 8 años de experiencia en el área de ciencia de datos.

Actualmente, soy líder técnico de ciencia de datos, en Neoris, con especialidad en visión computacional y grandes modelos de lenguaje (LLM), además de fungir como punto de contacto con 104 consultores asignados al cliente.

He trabajado en los sectores asegurador, retail y manufactura, colaborando con empresas como Zurich, Allianz, Ragasa, Coca-Cola y Cemex.

Objetivo del Taller:

Proveer a los participantes los conocimientos y herramientas prácticas necesarias para integrar **procesamiento de video**, **detección de objetos** y **automatización de alertas** utilizando Python y bibliotecas clave como OpenCV, YOLOv5 y Telegram.

Los asistentes aprenderán a conectar cámaras, aplicar transformaciones básicas a videos, implementar modelos de inteligencia artificial para detección en tiempo real y enviar notificaciones automatizadas, desarrollando una base para construir sistemas de cámaras inteligentes.

Definiciones

Modelos de Cámaras Principales

- **Contenido:**
  - **Cámaras analógicas (CCTV):** Transmiten señales de video en formato analógico y requieren dispositivos de grabación específicos.
  - **Cámaras IP:** Transmiten video digital a través de redes IP, permitiendo acceso remoto y mayor flexibilidad en la instalación.
  - **Cámaras con inteligencia artificial integrada:** Equipadas con algoritmos de IA que permiten análisis en tiempo real sin necesidad de procesamiento externo.

Gongusca

¿Qué es la Inteligencia Artificial (IA)?

La IA es la capacidad de las máquinas para realizar tareas que normalmente requieren inteligencia humana, como el reconocimiento de patrones, la toma de decisiones y el aprendizaje a partir de datos.

Cámaras Inteligentes

Las cámaras inteligentes son dispositivos de videovigilancia que incorporan algoritmos de inteligencia artificial (IA) para analizar imágenes y videos en tiempo real, permitiendo la detección y respuesta automática a eventos específicos.

Importancia en la seguridad moderna: Estas cámaras mejoran la eficacia de los sistemas de seguridad al proporcionar alertas inmediatas y reducir la necesidad de monitoreo humano constante.

Aplicaciones comunes: Se utilizan en entornos como hogares, empresas, espacios públicos y sistemas de transporte para detectar intrusiones, monitorear comportamientos sospechosos y gestionar el flujo de personas.

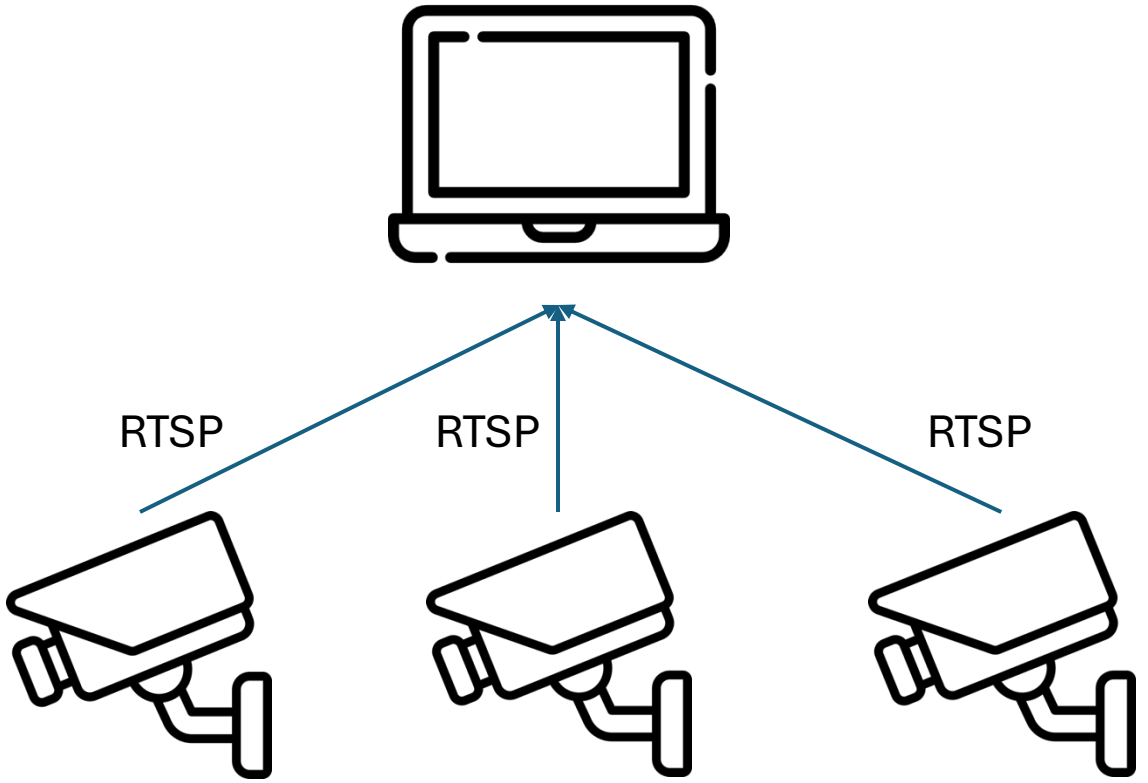
Gongusca

Protocolo de Comunicación RTSP

El Protocolo de Transmisión en Tiempo Real (RTSP) es un protocolo de control para sistemas de transmisión de **medios en tiempo real.**

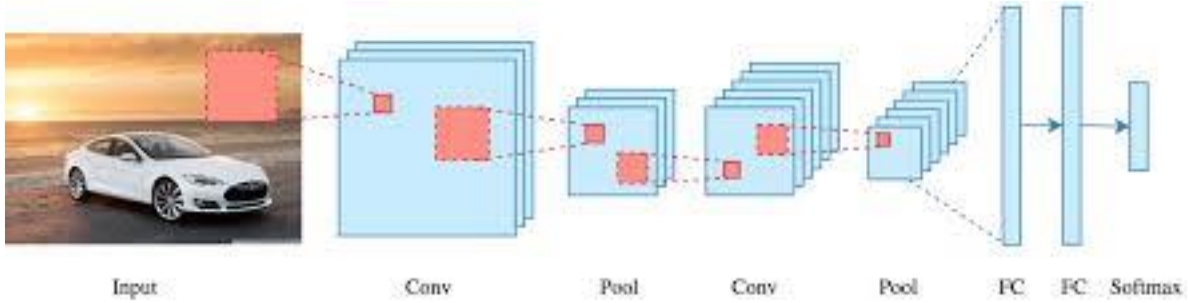
Permite la transmisión de video en vivo desde cámaras IP a sistemas de gestión de video y otros dispositivos.

Facilita la integración de múltiples cámaras y la transmisión eficiente de video en redes IP.



Introducción a Redes Neuronales Convolucionales (CNN)

Las CNN son un tipo de red neuronal diseñada para procesar datos con una estructura de cuadrícula, como imágenes, y son especialmente efectivas en tareas de reconocimiento visual.



Roboflow

Limitaciones de CNN

Roboflow

Desarrollo de proyectos

Pasos de implementación

- Levantamiento de necesidades
- Propuesta de arquitectura de solución
- Entrenamiento de modelo
- Deploy de modelo
- Monitoreo del modelo

Recomendaciones

Conocimiento previo

Antes de desarrollar un proyecto, es fundamental tener claridad sobre los alcances de los modelos que se van a implementar. Esto implica, desde un inicio, conocer una buena variedad de modelos y comprender sus principales limitantes. Entre las consideraciones más importantes se encuentran:

1. **Limitaciones en el desempeño durante la inferencia:** Evaluar cómo los modelos responden a distintos escenarios o casos de uso.
2. **Tiempos de procesamiento:** Determinar la relación entre el tiempo de procesamiento y el hardware disponible, ya sea CPU, GPU o TPU.
3. **Entorno de procesamiento:** Decidir si el procesamiento se realizará de manera local o en la nube, considerando los costos, la escalabilidad y la disponibilidad del hardware.

Conocimiento previo

Si existe una necesidad que se busca cubrir mediante **visión computacional**, es crucial garantizar que la implementación del proyecto se complete **en tiempo y forma**. Para ello, una regla empírica que puede aplicarse a la mayoría de los proyectos es:

> **“Si el ojo humano puede detectarlo, la computadora también podrá detectarlo”.**

Diversidad de modelos

Conocer desde técnicas de **clasificación** hasta **segmentación semántica** nos permite desarrollar aplicaciones más robustas y adaptables. Por ejemplo, la **Pose Estimation** nos proporciona una representación en forma de "esqueleto" de las personas, permitiéndonos analizar movimientos y posturas.

Familiarizarse con diferentes tipos de modelos amplía nuestras capacidades para implementar una mayor diversidad de soluciones.

Caso de Uso: Detección de un Asalto

Para resolver un problema como la detección de una persona cometiendo un asalto, es posible abordar la solución desde diferentes enfoques:

1. **Modelo de Clasificación:**
   Este enfoque clasifica la acción como un asalto basándose en patrones visuales detectados en el video.
   Limitación: El modelo requiere que las situaciones de asalto en tiempo real tengan patrones visuales similares a los utilizados durante su entrenamiento.
2. **Modelo de Pose Estimation:**
   Este enfoque permite centrarse exclusivamente en la estructura esquelética de la persona. Analizando únicamente el "esqueleto", se puede clasificar la acción sin depender de los detalles visuales del entorno.

Manos a la obra

Diagrama de entregable del taller

Conexión a cámara (Usando RTSP)

Lectura de Frames (Opencv)

Creación de buffer

Este buffer permite almacenar los últimos fotogramas

Aplicar algún filtro, transformación (Opencv)

Inferencia usando (YOLO v5)

Proceso de entrenamiento

Para poder entrenar un modelo es necesario recolectar imágenes y etiquetarlas

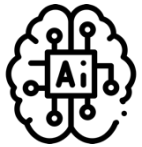
Recolección de imagenes

Herramientas para etiquetado:

Labelme

Cvat

Dividir el dataset en train, validate, test, es recomendable que exista variedad en el dataset, y no incluir imágenes “muy parecidas” en el dataset de train/validate y en el de test



Entrenar usando el dataset de train y validate.

Medir el desempeño del modelo usando el dataset de Test

Despliegue del modelo

Docker será la herramienta para desplegar tu modelo

Es importante hacer el fordward de los puertos necesarios

Puedes activar una base de datos para almacenar eventos

Se puede incorporar el envio de alertas

Crear un Bot de Telegram y Obtener el Token

Abre Telegram y busca el usuario **@BotFather**.

Inicia una conversación con **@BotFather** y envía el comando /start.

Envía el comando /newbot para crear un nuevo bot.

Sigue las instrucciones que te dará **BotFather**:

- Asigna un **nombre** a tu bot (por ejemplo, EchoBot).
- Asigna un **nombre de usuario** único que termine en bot (por ejemplo, EchoBot123_bot).

Al finalizar, **BotFather** te proporcionará un **Token de acceso**. **Guárdalo**; lo necesitaremos más adelante.

Obtener el ChatID

- https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getUpdates