**LEONARDO SANCHEZ RONCANCIO**

**BootCamp de Inteligencia Artificial**

**Qualetum**

**19/05/2024**

**IDENTIFICADOR DE SEÑALES DE TRANSITO POR COMPUTER VISION**

**UTILIZANDO UN MODELO REENTRENADO DE YOLO V5**

Dentro del marco de un proyecto de educación vial, surge la necesidad de innovar en los métodos de enseñanza sobre señalización, dado que resulta evidente que la mayoría de las personas desconocen gran parte de las señales de tráfico presentes en la ciudad y en las vías. El objetivo primordial consiste en idear una aplicación o actividad que capte poderosamente la atención del público, incentivando su participación en la experiencia y, al mismo tiempo, propicie el aprendizaje acerca de la señalización vial.

La propuesta de experiencia, validada por el cliente para alcanzar los objetivos establecidos, radica en el desarrollo de un modelo de visión por computadora que sea capaz de identificar las señales y brindar instrucciones a los individuos en tiempo real. Este modelo de visión por computadora se integrará en una cámara que el usuario llevará consigo, grabando su recorrido y lo que visualiza en su entorno. Simultáneamente, en otra pantalla se mostrará cómo el modelo de visión por computadora procesa el video en tiempo real, identificando las señales de tráfico mediante su capacidad de detección de objetos, lo cual resultará sumamente llamativo para los espectadores de la experiencia.

Además, el desarrollo incluirá una componente de interacción verbal, a través de la cual se proporcionarán indicaciones tanto al participante como a los observadores sobre la señal en cuestión y las acciones pertinentes a realizar. Esta funcionalidad busca reforzar el conocimiento de las señales de tráfico entre los usuarios.

En la fase inicial del proyecto, que se explicará y entregará con este documento, se llevará a cabo el entrenamiento del modelo utilizando un conjunto inicial de señales de tráfico. Posteriormente, se evaluará la capacidad del modelo para identificar y distinguir estas señales en tiempo real, y finalmente se integrará con otros componentes para poder entregar un output con le sonido del nombre de las señales que se están detectando en tiempo real

La segunda fase consistirá en implementar el modelo en producción, de manera que pueda ser utilizado con una cámara portátil y no desde el computador que tiene el código. Se desarrollará un aplicativo para permitir la interacción con la experiencia y facilitar su despliegue en un servidor, eliminando la necesidad de contar con el entorno de desarrollo original, adicionalmente, este aplicativo debe tener una conexión con un gestor de modelos, para poder ir reentrenando y mejorando el modelo de visión por computador que consume el desarrollo, de acuerdo con el feedback loop, que también debe ser definido e implementado

Por último, se evaluará la capacidad de captura de datos y generación de contenido verbal en la experiencia, que consistirá en explicaciones pregrabadas que se reproducirán cuando el modelo identifique una señal específica. Este paso asegurará la efectividad comunicativa de la experiencia educativa.

Para un correcto desarrollo de la experiencia se deben estructurar y cumplir las siguientes actividades:

**Desarrollo de la solución:**

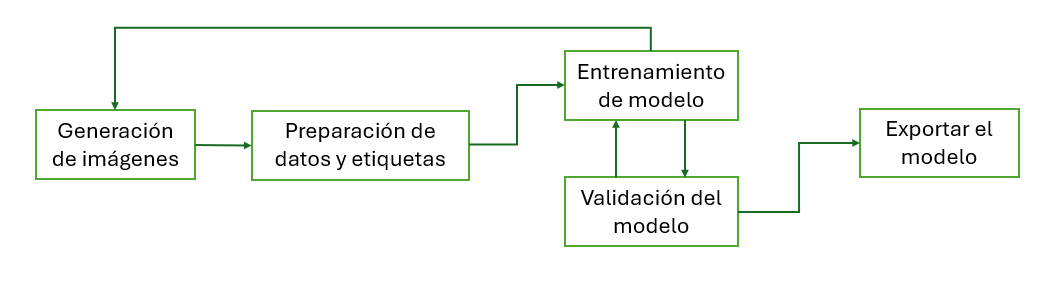
* **Acceso a datos, identificación de variable objetivos y definición de hipótesis de partida:**

Los datos necesarios para realizar el desarrollo se descargarán de internet, que fueran fotos, imágenes y diseños de 4 señales de tránsito, las cuales serán el objetivo de identificación de este modelo. Para poder entrenar modelos de computer visión es necesario generar la etiqueta de estas imágenes, las cuales se generan con una aplicación llamada Make Sens, y separar tanto las imágenes como las etiquetas en train y test, se evidencia que es fundamental entrenar el modelo con la misma cantidad de imágenes para cada tipo de señal, de lo contrario el modelo identificará mejor la señal que tuvo mas datos de entrenamiento, que las señales con menos imágenes, y esto no es deseable en el desarrollo.

Como las imágenes de tránsito son las mimas y no se encuentran muchas fotos e imágenes en internet, se crean imágenes para cada señal con diferentes ángulos, para que el modelo sea capaz de generalizar la señal y no solo la interprete desde una sola posición.

* **Diseño del flujo de trabajo de las diferentes fases/etapas:**

**Entrenamiento:**

****

En la parte de entrenamiento, se utilizará un modelo ya desarrollado llamado YoloV5 (https://github.com/ultralytics/yolov5) el cual debe ser reentrenado con las imágenes de tránsito que requieren ser detectadas por el modelo, es importante entrenar el modelo con diferentes datasets para saber cuál tiene mejores resultados, dicho entrenamiento se hace en Google Colab, utilizando la GPU que ofrecen de manera gratuita por periodos de tiempo limitados, pero suficientes para entrenar 2 o 3 modelos, guardarlos, compararlos con tensorboard y descargarlos.

* **Análisis de datos y conclusiones preliminares.**

Los datos para estos proyectos son generados, se toman las imágenes de fotos de tránsito, y se le generan algunos cambios, girando las imágenes , recortándolas, alejándolas y acercándolas. También se toman fotos de las señales impresas, desde diferentes ángulos distancias y ángulos, para que el modelo permita generalizar, esta es la parte inicial del desarrollo, para posteriormente, entrenarlo con fotos de las señales objetivo que se encuentren en la calle o donde se vaya a presentar la experiencia de aprendizaje con señales y modelos de IA

* **Selección de métricas técnicas y de negocio.**

Como es un modelo de clasificación, se tendrá en cuenta la matriz de confusión, y el parámetro de precisión para medir la cantidad de predicciones correctas sobre el total de predicciones, teniendo en cuanta que acá no hay ninguna clase más importante o critica, sino que se quiere acertar lo máximo posible en cada una de ellas.

* **Preprocesamiento de datos.**

No se hace preprocesamiento de los datos, ya que se están generando los datos que se van a cargar, y se evalúa cada imagen antes de agregarla al Dataset.

* **Selección de la arquitectura de modelo y decisión de si es necesario entrenamiento desde cero, fine-tuning o refinamiento (p. ej.: prompts).**

Se elige el modelo YoloV5 el cual permite hacer un fine tuning con nuestro propio dataset para hacer la detección de objetos que se desee, y es un modelo que funciona bastante bien.

* **Entrenamiento/ajuste del modelo seleccionado.**

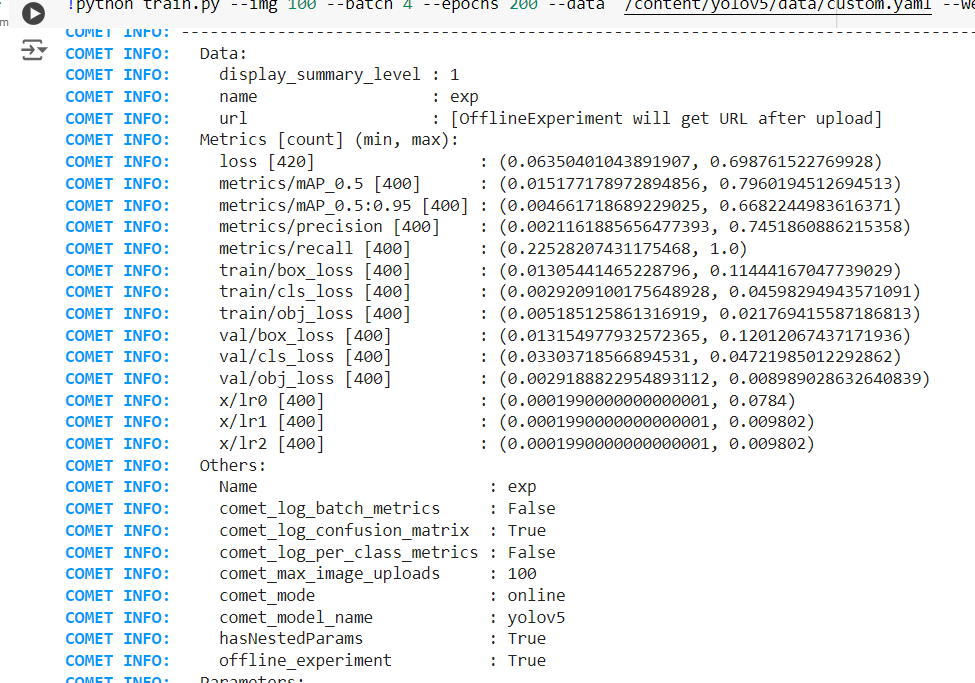
El modelo se va a entrenar en Google Colab para poder utilizar la GPU, y se utilizara la documentación de Yolo para poder obtener buenos resultados, así mismo se realiza la evaluación de los modelos utilizando la librería de tensorboard para poder evaluar los diferentes modelos viendo gráficamente las métricas de evaluación de cada modelo y las matrices de confusión.

Para la gestión de modelos se crea un documento Excel donde se guardan las métricas de cada modelo entrenado, guardando la precisión, el racall, mAP50 como métricas de medición del modelo, incluyendo imágenes de la matriz de confusión y las métricas de entrenamiento a través de las épocas, además se guardan los parámetros utilizados en cada entrenamiento que son las épocas, el batch y los datos de entrenamiento:

Interfaz de usuario gráfica

Descripción generada automáticamente

Como ejemplo se toma el modelo “Modelo\_1-2\_4\_200” en el cual se utilizaron los datos de la carpeta “data1+data2” un batch-size de 4 y 200 epocas, y se obtuvieron como resultados una precisión de 0.738, un Recall de 0.75, con las siguientes gráficas y matriz de confusión:



*Resultados del entrenamiento modelo “Modelo\_1-2\_4\_200”*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

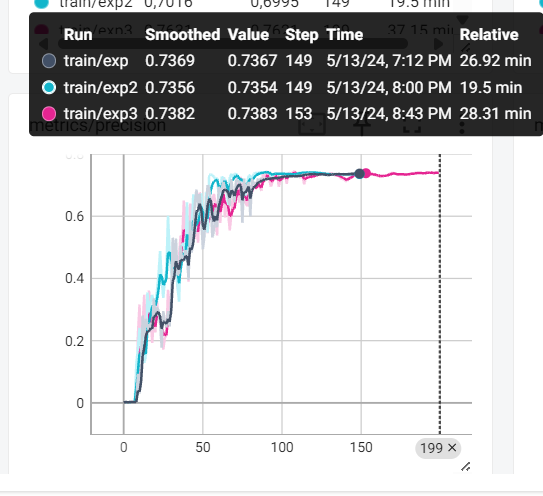
*Métricas del entrenamiento modelo “Modelo\_1-2\_4\_200”*

Gráfico, Gráfico en cascada

Descripción generada automáticamente

*Matriz de confusión del entrenamiento modelo “Modelo\_1-2\_4\_200”*

Los diferentes modelos se pueden comparar utilizando Tensorboard:



*Comparación de la métrica de precisión entre diferentes modelos entrenados*

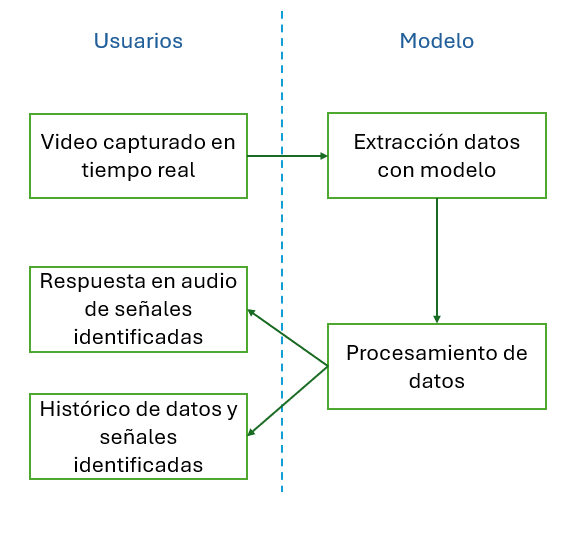
* **Evaluación del modelo**

Para poder evaluar la efectividad del modelo, se realizará una prueba desde la cámara, donde se le podrán en tiempo real imágenes de transito y se evaluara su capacidad para reconocerlas, diferenciarlas y pasar por voz el resultado

**Desarrollo de los componentes de inferencia:**

* **Diseño del flujo de trabajo de las diferentes fases/etapas.**

**Inferencia:**



En la parte de inferencia, se ejecuta el desarrollo que consume la información de video de la cámara en tiempo real, estas imágenes capturadas pasan por el modelo Yolo V5 con el fine-tuning de las señales de tránsito, para poder reconocerlas en tiempo real, dicha información es guardada en un Dataframe. Durante la ejecución se guarda la información en un dataframe:

Interfaz de usuario gráfica, Tabla, Excel

Descripción generada automáticamente

El cual se utilizará para hacer una selección de los últimos 6 registros, si dentro de esta selección encuentra 3 nombres de señales iguales, va a reproducir el nombre de la señal en sonido, esto sirve para incrementar la cantidad de aciertos de sonidos generados por el modelo a partir de las señales. Adicionalmente, el dataframe se guardará como un documento Excel por si se desea hacer una consulta de la información capturada durante la ejecución de la implementación.

* **Diseño y definición de contratos de datos entre componentes.**

Para esta fase inicial del proyecto, se integra todo el desarrollo en un código en Python, que se conecta la cámara del computador para consumir el video en tiempo real, este código en Python va a procesar el video, hacer la detección de los objetos, generara el dataframe con toda la información y entregara como output el video haciendo la detección y el audio con el nombre de la señal que esta identificando.

Para las siguientes fases de desarrollo se requiere implementar una aplicación web, donde el usuario pueda ejecutar la aplicación y empezar a hacer la detección de objetos, donde se requiere conectar la cámara con la aplicación, y también el audio donde se va a trasmitir el sonido del output con el nombre de la señal.

Adicionalmente, se requiere implementar un sistema de gestión para los modelos, donde se pueda seleccionar el mejor modelo y que permita ir entrenando nuevos modelos, para poder incluir nuevas señales, nuevos datos de entrenamiento, parámetros, hiperparametros y realizar los cambios dependiendo del feedback loop.

* **Implementación de funciones/componentes para la preparación de datos**

No se requiere hacer una preparación de datos, ya que, al trabajar con detección de objetos, lo mejor es entrenar el modelo con las imágenes que se captarían en el funcionamiento real.

* **Implementación de funciones/componentes para la inferencia de los modelos.**

Se realiza un código en Python, el cual consume el modelo de Yolo con fine - tuning seleccionado, este código tiene funciones que procesan la información y reproduce audios con el nombre de las señales identificadas.

* **Implementación de funciones/componentes para monitorizar el funcionamiento del modelo a partir de las métricas seleccionadas.**

Para monitorizar se debe utilizar el modelo, se hace utilizando el modelo, ya que el desarrollo debe consumir el video al cual le va a realizar la detección de objetos, en primera instancia se revisa en tiempo real el nivel de confidencia de las detecciones, las cuales se ven en la pantalla y adicionalmente quedan guardadas en un dataframe, eso va a permitir evidenciar en cada escenario y condiciones donde se use el modelo, que señales tienen un mayor porcentaje de predicción y cuales tienen valores no aceptables que refleja que el modelo no está haciendo una correcta identificación, adicionalmente, se puede ver tanto en tiempo real como en el histórico guardado, que señales se están confundiendo, ya que van a salir al tiempo dos nombres al mismo tiempo, cuando en realidad debería aparecer solo una (cuando se encuentre solo una señal en el video)

* **Diseño e implementación (si aplica) de feedback loop o capacidad de interacción con el usuario.**

El feddback loop es una etapa fundamental para este proyecto, ya que va a permitir mejorar el modelo constantemente a partir de los video e imágenes que el mismo modelo vaya consumiendo y que no tenga una correcta identificación. Para esta fase la etapa de feedbackloop se hace por revisión manual de un humano, al poner señales en la cámara y evaluar el resultado que esta entregando el modelo.

Durante el desarrollo de esa aplicación, se entrena inicialmente el modelo con un dataset de imágenes extraídas de internet, las cuales dieron malos resultados, pero ende se creó un nuevo dataset con imágenes generadas desde la posición de la cámara, lo que significa que se entrena con las imágenes desde el escenario y condiciones que serán consumidas, lo cual es exactamente lo mismo que se realizara cuando se pruebe con video en la calle, las señales que no se detecten, serán guardadas como imágenes y se le entregaran al modelo para un reentreno y así ir mejorando su capacidad para poder generalizar