Facultad de Ingeniería
Universidad de Buenos Aires

Propuesta de Trabajo Profesional

Licenciatura en Análisis de Sistemas, Ingeniería en Informática

---

Sistema de detección en tiempo real de publicidad en la vía pública

---

Tutor: **Ing. Martín Buchwald**

del Mazo, Federico
100029

Pastine, Casimiro
100017

Índice

1. Motivación

Dentro del marco de trabajo profesional de las carreras de Licenciatura en Análisis de Sistemas y de Ingeniería en Informática de la Facultad de Ingeniería, Universidad de Buenos Aires, se presenta la siguiente propuesta de proyecto con el objetivo de desarrollar un sistema de detección de publicidades en la vía pública en tiempo real.

El interés detrás de este proyecto radica en la creciente intrusión e invasión de publicidades no deseadas en nuestro entorno cotidiano. Este proyecto busca concientizar y ofrecer una solución práctica a esta problemática, enfocándose principalmente en la detección, y luego permitiendo al usuario aplicar un post-procesamiento configurable para difuminar, resaltar o transformar de otra manera lo detectado.

2. Objetivos

- Aplicar los conocimientos interdisciplinarios de comprensión y evaluación de sistemas aprendidos en nuestra carrera académica en un campo enteramente nuevo para nosotros como lo es *Computer Vision*.
- Realizar un proyecto integral con su debida investigación y comprensión del estado del arte en las disciplinas relevantes, abarcando desde el entrenamiento de un modelo de clasificación de objetos hasta el despliegue del sistema en algún entorno fácilmente accesible por el usuario final.
- Encontrarnos con desafíos prácticos como la falta de los datos necesarios para el modelo a entrenar y la evaluación y comparación de distintos métodos para lograr un rendimiento aceptable y encontrar las soluciones adecuadas para los obstáculos que vayan apareciendo, sopesando las distintas alternativas posibles que surgirán durante el desarrollo del trabajo.
- Lograr hacer un sistema de código abierto, modularizable y extensible tal que en un futuro se pueda cambiar cualquier eslabón de su cadena como los objetos a detectar, el algoritmo de detección y rastreo de objetos o el post procesamiento a aplicar.

3. Descripción del problema

3.1. Computer Vision

El campo de *Computer Vision* se refiere a la disciplina de la inteligencia artificial que se ocupa de permitir a las computadoras "ver" y comprender el mundo visual de la misma manera que lo hacen los seres humanos. Utiliza diversas técnicas y algoritmos para adquirir, procesar, analizar y comprender imágenes y videos digitales con el objetivo de extraer información significativa y tomar decisiones basadas en ella.

En los últimos años este campo ha experimentado un crecimiento y una atención considerable debido al aumento del poder computacional, la disponibilidad de grandes conjuntos de datos y los avances en el aprendizaje automático y las redes neuronales.

Uno de los avances más revolucionarios dentro del campo de *Computer Vision* es el uso de las *Redes Neuronales Convolucionales (CNN)*. Son un tipo de arquitectura de redes neuronales especialmente diseñadas para el procesamiento de datos de imágenes o señales de audio. La característica determinante es la capa convolucional de la red, a través de la cual las CNN aprenden a extraer características y combinan esta información para realizar tareas de clasificación y reconocimiento de patrones.

Uno de los problemas iniciales de *Computer Vision* es la clasificación de imágenes: Partiendo de una imagen, queremos saber si la misma contiene un objeto deseado. Por ejemplo, podríamos tener una foto en la vía pública y querer saber si contiene o no a un cartel publicitario.

El paso siguiente a esto es la detección de objetos. Ya no solo importa si el cartel publicitario está presente, sino exactamente en qué porción de la imagen lo está.

Naturalmente el paso siguiente a detectar objetos en imágenes estáticas (fotografías), es hacerlo en video. Queremos detectar cuándo un objeto entra en escena, cuando sale, y poder seguirle el rastro en todo momento. Esto se denomina *Object Tracking* y es la técnica principal sobre la cual basamos nuestro trabajo.

El paso final en cuanto a la detección de objetos en nuestro trabajo incluye la meta de hacerlo en tiempo real. A diferencia de analizar un video pregrabado, donde es prescindible la velocidad de procesamiento, esto plantea el desafío considerable de tener que procesar los cuadros del video más rápido de lo que están siendo capturados para poder mostrar el resultado final inmediatamente.

3.2. Despliegue

En el despliegue del sistema se nos presentan alternativas que van a ser evaluadas en términos del rendimiento y la experiencia de usuario que buscamos alcanzar.

El mínimo entregable que proponemos para nuestro trabajo es una herramienta de procesamiento de video pregrabado que se corra localmente por línea de comando y devuelva un video procesado como salida.

Nuestra meta principal es entregar una experiencia de usuario con una interfaz mejor a la de línea de comando, que pueda ser utilizada desde cualquier parte, sin requerir ningún tipo de instalación previa. Por esto es que consideramos que la interfaz ideal del sistema es en un sitio web. El hecho de que la interfaz sea una página web nos da dos posibles arquitecturas a evaluar. Por un lado contemplamos la arquitectura cliente/servidor, mientras que su contraposición natural es un sistema que sea ejecutado solamente en el cliente local.

En una arquitectura cliente/servidor, la capacidad computacional del cliente no es relevante, pero la complejidad principal del sistema recae en poder minimizar la latencia entre el cliente y el servidor. Por otro lado, en una arquitectura sin servidor donde todo se ejecuta localmente, el principal desafío es poder seguir ofreciendo una buena experiencia de usuario independientemente del poder computacional del dispositivo sobre el cual corre el sistema. Sobre esta última idea, una alternativa disponible es la de utilizar *WebAssembly*, ya que nos permitiría desarrollar el sistema en cualquier lenguaje de programación, donde podamos enfocarnos en la velocidad de procesamiento por sobre lo disponible en el entorno web.

3.3. Tecnologías Relevantes al Trabajo

El lenguaje de programación que vamos a utilizar para realizar el trabajo es *Python*, debido a que es ampliamente utilizado en el campo de *Computer Vision* gracias a su flexibilidad, bibliotecas especializadas y facilidad de uso.

La principal herramienta que utilizaremos es *OpenCV* **[1]**, una biblioteca de código abierto popular que proporciona una amplia gama de funciones y algoritmos para el procesamiento de imágenes y videos.

Para la construcción de la red neuronal convolucional haremos uso de *Keras* **[2]**. Esta es una biblioteca de código abierto con una interfaz de alto nivel para redes neuronales de la biblioteca *TensorFlow* **[3]**, que usaremos para facilitarnos el entrenamiento y la evaluación de los modelos de aprendizaje profundo.

Para el armado del modelo haremos uso del *dataset* de imágenes *Open Images* **[4]**, el cual proporciona una amplia colección de imágenes anotadas en diversas categorías. Este será una de las fuentes principales para entrenar nuestra red neuronal ya que nos permitirá trabajar con un conjunto diverso y representativo de imágenes.

Una tecnología que ya mencionamos y de la cual vamos a hacer uso es *WebAssembly* **[5]**. *WASM* nos va a permitir compilar el código fuente que realicemos en *Python* a un código que pueda correr directamente en un navegador web sin tener que resignar su rendimiento.

Durante el desarrollo inicial utilizaremos *Roboflow* **[6]**, un producto que ofrece acceso a redes neuronales convolucionales de código abierto para clasificar imágenes y videos a través de llamados *HTTP*. Estas redes nos ayudarán en las pruebas de concepto y evaluación de opciones de desarrollo sin necesidad de crear nuestra propia red convolucional. Sin embargo, debido a la dependencia de un servicio externo y la latencia asociada, *Roboflow* no formará parte del trabajo final. De todas formas, los conjuntos de datos utilizados para entrenar las redes de *Roboflow* son públicos y se utilizarán como una de las fuentes para entrenar nuestra propia red.

4. Esquema de Trabajo

4.1. Metas Parciales

Considerando la naturaleza progresiva del problema, el trabajo se descompone en una serie de metas que se pueden agrupar de acuerdo a su dependencia y modularidad. Estos grupos de metas están planificados para que se puedan trabajar de manera independiente.

- **CV**: Partiendo de una red convolucional no propia, y tomándola como ya hecha, el trabajo propio de*Computer Vision* se descompone en las siguientes metas

CV1	Clasificar una imagen de acuerdo a si contiene o no contiene un cartel publicitario con una CNN de base.
CV2	Evaluar los frameworks de detección de objetos como SSD, YOLO y R-CNN. Detectar la porción de la imagen donde se contiene el cartel.
CV3	Descomponer un video en sus imágenes por segundo y procesar cada una de manera separada.
CV4	Evaluar las implementaciones de rastreo de objetos como CSRT, KCF y MOSSE. Detectar los objetos en algunas imágenes del video, y rastrear su movimiento, en vez de procesar cada imagen por separado.
CV5	Introducir modificaciones en las imágenes del video como difuminado y resaltado.
CV6	Optimizar la velocidad de procesamiento de video para poder procesar los videos en tiempo real y en dispositivos de bajo poder computacional.

- **CNN**: El modelo de red convolucional se va a desarrollar por separado para luego poder modificarlo para detectar cualquier objeto deseado en vez de restringirse a un solo dominio.

CNN1	Obtención de datos de carteles publicitarios.
CNN2	Evaluar las arquitecturas de redes convolucionales como VGGNet o MobileNet. Armar una red convolucional que detecta carteles publicitarios.
CNN3	Armar una red convolucional que detecta publicidades en general (por ejemplo, logos en ropa) en vez de solamente carteles.

- **WEB**: El sistema será desplegado en la web y accede a la cámara en tiempo real del cliente.

WEB1	Evaluar ventajas y desventajas de WASM vs Cliente/Servidor y confirmar la viabilidad de la elección.
WEB2	Armar un sitio web local que acceda a la cámara del usuario y utilice el modelo.
WEB3	Desplegar el servicio web a un servidor externo y evaluar la latencia y disponibilidad.

4.2. Cronograma Final

En base a las metas planteadas se arma un cronograma final de las distintas etapas que se van construyendo para llegar al trabajo deseado.

- **Prueba de Concepto y Viabilidad**
  Abarca las metas: ***CV1, CV2, CNN1***

  La idea de esta etapa es confirmar que nuestra propuesta es viable y se va a poder hacer. Una vez confirmado que es posible hacer lo que buscamos, procederemos a la siguiente etapa del trabajo.

- **Arquitectura del Trabajo**
  Abarca las metas: ***WEB1, WEB2***

  Buscamos evaluar las diferentes alternativas de despliegue y decidir la arquitectura final de nuestra solución. Seleccionaremos una, confirmando que es factible y que va a tener un rendimiento adecuado.

- **Trabajo final**
  Abarca las metas: ***WEB3, CNN3, CV5, CV6***

  El paso final del trabajo es integrar las distintas partes que se fueron desarrollando y lograr hacer procesamiento de videos en tiempo real, desde un sitio web ya desplegado.

Referencias

[1] OpenCV: https://opencv.org/

[2] Keras: https://keras.io/

[3] TensorFlow: https://www.tensorflow.org/

[4] Open Images Dataset: https://storage.googleapis.com/openimages/web/index.html/

[5] WebAssembly: https://webassembly.org/

[6] Roboflow: https://roboflow.com/

Bibliografía

Rosebrock, Adrian. (2017). *Deep learning for Computer Vision with Python.*

Ansari, Shamshad. (2020). *Building Computer Vision Applications Using Artificial Neural Networks*, Apress.

Moroney, Laurence. (2020). *AI and Machine Learning for Coders*, O'Reilly.