Procesamiento de imágenes utilizando Deep learning y Carla Simulator

***Resumen*—En este informe se presenta la implementación de una red Pix2Pix capaz de pasar de una imagen RGB a una imagen segmentada donde cada color indica un objeto diferente. Se comenzara desde cero obteniendo el dataset mediante el simulador CARLA para luego poder entrenar la red con una gran variedad de imagenes con diferente clima y vegetación. Finalmente se probara con imagenes reales para analizar la viavilidad de entrenar una red con un simulador y luego ingresarla en el mundo real.**

I. Introducción

En los ultimos años, debido a la necesidad de resolver cada vez mas complejos y la posibilidad de utilizar GPU's las redes neuronales han cobrado gran importancia en nuetras vidas, y ya son parte del dia a dia, cosas tan simples como los filtros de Instagram basan su reconocimiento facial en redes neuronales.

Las redes permiten resolver problemas que van desde la clasificación de imagenes hasta la conducción autonoma, claro esta que para esto existen diferentes arquitecturas las cuales son expecificas para determinadas tareas, como el procesamiento de imagenes (CNN, GAN, YOLO), procesamiento de texto (RNN), procesamiento de datos (NN).

La desventaja principal de estos modelos supervisados es la necesidad de una dataset muy bien estructurado y para el cual es necesario el trabajo humano, esta es la tarea mas importante y que si esta mal realizada el red no tendra un desempeño correcto.

II. Marco Teoríco

II-A. Deep Learning

Las redes neuronal son muy utilizadas en el campo de la minería de datos, debido a su gran versatilidad. Dentro de este mundo existen 2 grandes tipos de entrenamientos para los supervisados que dada una entrada X se conoce la salida deseada Y y otros casos los algoritmos no supervisados como los modelos de Q-learning o los mapas autoorganizados (SOM). En este proyecto se opto por un entrenamiento supervisado.

Para el modelado de la red neuronal se utilizó un modelo llamado Pix2Pix [4]. El cual pertime dada una imagen de entrada X obtener una imagen de salida Y haciendo uso de una arquitecrua U-Net [3].

Este modelo cuenta con 2 redes neuronales una denominada el generador en cual recive la imagen X y debe ser capaz de aprender a transformala en la imagen $\hat{Y}$, la cual es una aproximación de la imagen Y. Y por otro lado el discriminador el cual debe tomar la imagen $\hat{Y}$ para lograr reconocer las partes de esta que sean errorneas. La salida del discriminador debe ser una imagen resucida en tamaño que cuente con un unico canal de color, siendo esta imagen una cuantificacion del error de la red generativa.

Este problema desemboca en una pelea entre en generador ya que debe aprender a engañar al discriminador y el discriminador que debe lograr siempre reconocer que la imagen $\hat{Y}$ es totalmente falsa, ya que es una creación de la red generativa.

II-B. Preporcesado de los datos

Un paso fundamental para trabajar con algoritmos de imagenes es pasar del domininio $[0; 255]$ a un dominio mas acotado, para este trabajo se utilizo el domino de $[-1; 1]$. De esta forma se logra una mayor eficiencia computacional.

Debido a la gran variedad de dimensiones de las imagenes es recomendable hacer un resize a dimensiones mas pequeñas para disminuir el uso de RAM cuando son cargadas en memoria y el uso de GPU al realizar las convoluciones. En este caso las imagenes de entrada y de salida son redimensionadas a $256x256$.

III. Desarrollo

III-A. Obtención del dataset

Para obtener el dataset se colocaron una camara RGB y una camara de segmentación semántica [2], ubicadas en la misma posición. Se recolectaron los datos cada 5 segundos. Las condiciones climaticas utilizadas fueron aletorias, lo que permite obtener un dataset mucho mas variado.

Para lograr un manejo autónomo del vehiculo se utilizó el autopilot [1]. Con la finalidad de aumentar la variedad de imagenes se utilizaron todos los mapas posibles y se adicionaron otros vehiculos autónomos.

III-B. Entrenamiento de la red

Como fue mencinado en el marco teoríco el modelo implementado se denomina Pix2Pix [4]. Se utilizó como entrada la imagen de la camara RGB y la salida será una imagen segmentada, para ello se utilizó el dataset que contava con casi 3000 imagenes.

Todas las imagenes fueron llevadas al domiminio $[-1; 1]$ como es nombrado en la

El dataset fue dividido en 2 partes una utilizada para el entrenamiento, aproximadamente el $80\,\%$ de las imagenes y otra parte para el testing de la red utilizando el $20\,\%$ restante.

Luego de **N** iteraciones se presentan los resultados en la fig.1

Figura 1. Imagenes obtenidas luego del entrenamiento

Las perdidas de la red generativa y del discriminador fueron recolectados para lograr mostrar con mayor facilidad los avances del modelo (ver fig.2-3)

Figura 2. Error del discriminador en función de la iteración con el dataset de testing.

Figura 3. Error del generador en función de la iteración con el dataset de testing.

IV. Conclusiones

Referencias

[1] "Carla api documentation," https://carla.readthedocs.io/en/latest/python_api/.

[2] "Carla sensors reference," https://carla.readthedocs.io/en/latest/ref_sensors/.

[3] "U-net architecture," https://en.wikipedia.org/wiki/U-Net.

[4] Phillip.I, J.Zhu, T.Zhou, and A.A.Efros, "Image-to-image with conditional adversarial networks," 2018.