

Self-Driving using CNN

Franz Maguiña

*Maestría en Ciencia de la Computación
Facultad de Ciencias - UNI*

Resumen—El presente paper es acerca del análisis, desarrollo, diseño e implementación de modelos de Inteligencia Artificial usando Redes Neuronales Convolucionales (Convolutional Neural Networks) para el tema de Self-driving usando los datos captados por una cámara.

Index Terms—self-driving, tensorflow, artificial intelligence, convolutional neuronal networks

I. INTRODUCCIÓN

Self-driving o automóvil autónomo es un vehículo capaz de detectar su entorno y navegar sin intervención humana. Los automóviles autónomos de hoy en día combinan una variedad de técnicas para percibir su entorno, incluido el radar, la luz láser, el GPS, la odometría y la visión por computadora.

Los sistemas de control avanzados interpretan la información sensorial para identificar las rutas de navegación adecuadas, así como los obstáculos y la señalización relevante. Por ello, la idea de hoy en día es que requieran de un único sensor, la cámara. Esto debido a que la cámara es un sensor cómodo, fácil de implementar y además brinda mayor información que otros sensores como información 3D y temporal alrededor del carro, por consiguiente. También se usarían más modelos de CNN, como por ejemplo, modelos que requieran la identificación de objetos, la proximidad de objetos y la predicción del movimiento de otros objetos alrededor del automóvil.

II. OBJETIVOS

El objetivo principal de este paper es analizar y desarrollar un modelo de self-driving con las metodologías aprendidas en clase de Procesamiento de Imágenes.

Objetivos secundarios:

- Analizar y desarrollar un modelo de self-driving usando CNN.
- Analizar la diferencia del modelo con diversos filtros aplicados a la imagen capturada.
- Analizar otros modelos de self-driving

III. ESTADO DEL ARTE

III-A. End to End Learning for Self-Driving Cars

En este paper se entrenó una CNN para mapear píxeles sin procesar desde una sola cámara frontal directamente a los comandos de dirección. Este enfoque de End to End donde demostró ser sorprendentemente poderoso. Puesto que, con datos de entrenamiento mínimos de humanos, el sistema aprende a conducir en el tráfico en carreteras locales con o sin marcas de carril y en carreteras. También opera en áreas con una guía visual poco clara, como en estacionamientos y en

caminos sin pavimentar. El sistema aprende automáticamente las representaciones internas de los pasos de procesamiento necesarios, como la detección de características útiles de la carretera con solo el ángulo de dirección humano como señal de entrenamiento. El modelo nunca se entrenó explícitamente para detectar, por ejemplo, el contorno de carreteras. En comparación con la descomposición explícita del problema, como la detección de marcas de carril, la planificación y el control de rutas, su sistema de end to end optimiza todos los pasos de procesamiento simultáneamente. Se argumenta que esto eventualmente conducirá a un mejor rendimiento y sistemas más pequeños. Se obtendrá un mejor rendimiento porque los componentes internos se optimizan automáticamente para maximizar el rendimiento general del sistema, en lugar de optimizar los criterios intermedios seleccionados por el ser humano, por ejemplo, la detección de carriles. Es comprensible que dichos criterios se seleccionen para facilitar la interpretación humana, lo que no garantiza automáticamente el máximo rendimiento del sistema. Las redes más pequeñas son posibles porque el sistema aprende a resolver el problema con un número mínimo de pasos de procesamiento. Usaron una NVIDIA DevBox y Torch 7 para el entrenamiento y una computadora de auto con conducción autónoma NVIDIA DRIVE (TM) PX que también ejecuta Torch 7 para determinar dónde conducir. Su sistema funciona a 30 fotogramas por segundo (FPS).

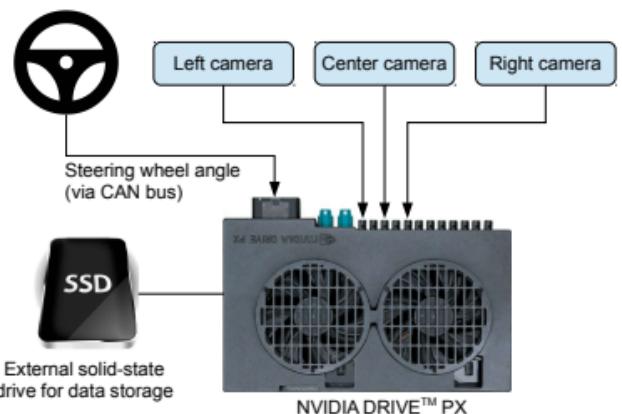


Figura 1. Nvidia DevBox

Arquitectura

La arquitectura de la red se muestra en la Figura 2. La red

usada consta de 9 capas, incluida una capa de normalización, 5 capas convolucionales y 3 capas completamente conectadas. La imagen de entrada se divide en planos YUV y se pasa a la red.

La primera capa de la red realiza la normalización de imágenes. El normalizador está codificado de forma rígida y no se ajusta en el proceso de aprendizaje. Realizar la normalización en la red permite modificar el esquema de normalización con la arquitectura de la red y acelerarlo mediante el procesamiento de la GPU.

Las capas convolucionales se diseñaron para realizar la extracción de características y se eligieron empíricamente a través de una serie de experimentos que variaron las configuraciones de las capas. Se usa convoluciones escalonadas en las primeras tres capas convolucionales con una zancada de 2×2 y un núcleo de 5×5 y una convolución no escalonada con un tamaño de núcleo de 3×3 en las dos últimas capas convolucionales.

Seguidamente, tiene cinco capas convolucionales con tres capas completamente conectadas que conducen a un valor de control de salida que es el radio de giro inverso. Las capas completamente conectadas están diseñadas para funcionar como un controlador para la dirección, pero observamos que al entrenar el sistema de un extremo a otro, no es posible hacer una separación clara entre qué partes de la red funcionan principalmente como extractor de características y cuáles servir como controlador.

Para eliminar el sesgo hacia la conducción en línea recta, los datos de entrenamiento incluyen una mayor proporción de fotogramas que representan curvas de la carretera.

Evaluación

La evaluación de su redes se realizó en dos pasos, primero en simulación y luego en pruebas en carretera.

En la simulación, las redes proporcionan comandos de dirección en un simulador a un conjunto de rutas de prueba pregrabadas que corresponden a aproximadamente un total de tres horas y 100 millas de conducción en el condado de Monmouth, Nueva Jersey. Los datos de la prueba se tomaron en diversas condiciones climáticas y de iluminación e incluyen carreteras, caminos locales y calles residenciales.

También se estima qué porcentaje del tiempo la red podría conducir el automóvil (autonomía). La métrica se determina contando las intervenciones humanas simuladas. Estas intervenciones ocurren cuando el vehículo simulado se aleja de la línea central en más de un metro. Se supone que en la vida real una intervención real requeriría un total de seis segundos: este es el tiempo necesario para que un humano retome el control del vehículo, lo vuelva a centrar y luego reinicie el modo de autodirección. Se calcula el porcentaje de autonomía contando el número de intervenciones, multiplicando por 6 segundos, dividiendo por el tiempo transcurrido de la prueba simulada, y luego restando el resultado de 1:

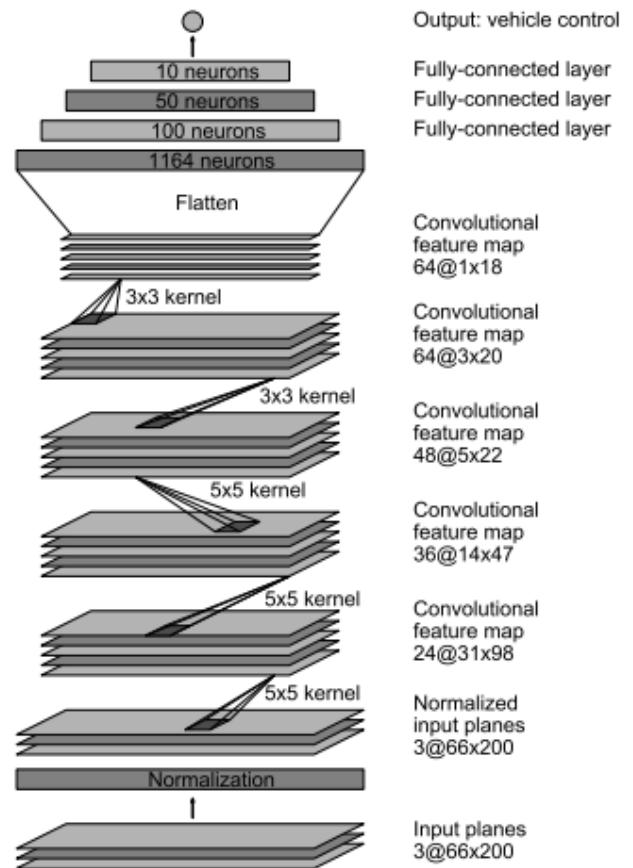


Figura 2. Arquitectura Propuesta

$$\text{autonomy} = \left(1 - \frac{(\#\text{of interventions}) \times 6\text{seconds}}{\text{elapsed time}[seconds]}\right) \cdot 100 \quad (1)$$

Así, si se tuviera 10 intervenciones en 600 segundos, tendríamos un valor de autonomía de

$$\left(1 - \frac{10 \times 6}{600}\right) \cdot 100 = 90\% \quad (2)$$

De la figura 3 el área verde de la izquierda se desconoce debido a la transformación del punto de vista. El rectángulo ancho resaltado debajo del horizonte es el área que se envía a la CNN.

De la figura 4 se tiene que arriba: subconjunto de la imagen de la cámara enviada a la CNN. Abajo a la izquierda: activación de los mapas de características de la primera capa. Abajo a la derecha: activación de los mapas de características de la segunda capa. Esto demuestra que la CNN aprendió a detectar características viales útiles por sí misma, con solo el ángulo de dirección humana como señal de entrenamiento. Nunca lo entramos explícitamente para detectar los contornos de las carreteras.



Figura 3. Captura del sistema desarrollado

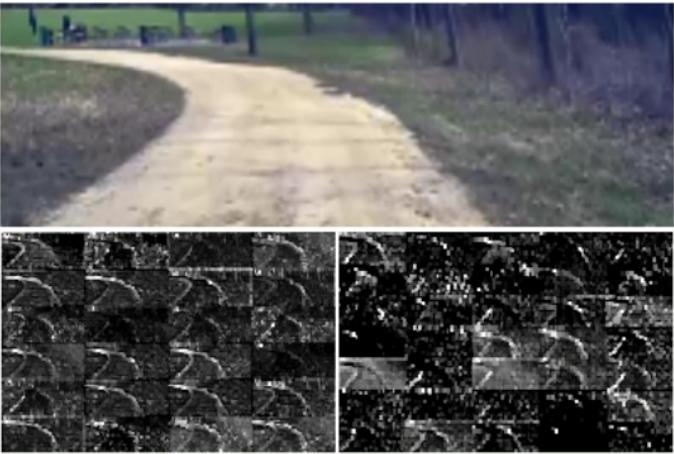


Figura 4. Lo que el modelo ve en imágenes con camino

De la figura 5 es un ejemplo de una imagen sin carretera. Las activaciones de los dos primeros mapas de características parecen contener principalmente ruido, la CNN no reconoce ninguna característica útil en esta imagen.



Figura 5. Lo que el modelo ve en imágenes sin caminos

IV. METODOLOGÍA

IV-A. Preparación de datos

Obtención de Datos

Como data de entrenamiento se usó el siguiente dataset: [Driving Dataset](#). Que contiene aproximadamente 63,000 imágenes con 3.1GB. Este dataset fue grabado por los alrededores del Rancho Palos Verdes y San Pedro de California. También, como información acerca de las imágenes, la fuente nos brinda un archivo con la información de las imágenes contenidas con la nomenclatura de título, ángulo, fecha y hora que capturo la imagen.



Figura 6. Imagen a usarse en el entrenamiento

Además, como dataset de pruebas se descargo algunos dataset de la fuente [The KITTI Vision Benchmark Suite](#), que es un repositorio con diferentes tipos de dataset, se puede encontrar imágenes grabadas en ciudad, campus, carretera y camino en trocha. Además, brindan adicionalmente los siguientes datos: la presión ejercida en el acelerador, torque de las ruedas, velocidad, latitud, longitud, entre otros.



Figura 7. Imagen a usarse en pruebas de verificación

Transformación de Datos

La lectura de datos se realiza utilizando la librería de OpenCV, aplicando las funciones de filtros como la transformación a Escala a Grises, la redimensión a 100×100 y la reducción de datos que están en un rango de $[0, 255]$ a un

intervalo determinado por la media y la desviación estándar determinada por:

$$X = \frac{X - \text{mean}}{\text{std}} \quad (3)$$

Además, el ángulo de las imágenes, también son transformadas ángulos radianes de la siguiente forma:

$$Y = \frac{Y \cdot \pi}{180} \quad (4)$$

De esta transformación, al no afectarse el signo de rotación, los sentidos seguirían siendo los mismos, es decir, se tendrían los mismos casos de:

- Si el ángulo es negativo, el timón esta rotado para la derecha
- Si el ángulo es cero, el timón esta sin rotar
- Si el ángulo es positivo, el timón esta rotado para la izquierda

IV-B. Arquitectura propuesta

Durante el proceso de entrenamiento y además, con otros modelos encontrados, se decidió usar el modelo de la figura 8

La arquitectura del modelo propuesto consta de 19 capas, incluidas 5 capas de convolucionales, 5 capas de maximización, 5 de normalización y 4 capas completamente conectadas.

El algoritmo de optimización usado es Adam con un learning rate de 0,00001

IV-C. Diversificación y ajustes en diversos ambientes

Para diversificar los datos de entrenamiento, se busco el que mayores lugares pueda contener, como caminos en ciudad, carretera y trochas.

V. EXPERIMENTACIÓN Y RESULTADOS

Una vez terminada el entrenamiento de ambos modelos, bajo el mismo dataset.

La figura 10 muestra que el modelo del paper, es mas eficiente que el nuestro, mostrado en la figura 12, y al tratarse de una ángulos, que son valores numéricos reales, el dato predicho para los datos de validación no necesariamente va a coincidir con el ángulo real, pero se puede tomar como referencia para verificar la diferencia entre ambos modelos.

Para ello, el siguiente paso es visualizar como se comporta ambos modelos con otros datos ajenos a los datos de entrenamiento, puesto que, implementar el paso de correcciones con intervenciones humanas o correcciones provenientes del dataset, vendría para un trabajo a futuro con otros pequeños ajustes.

Para esto, se elaboro un pequeño programa que combina un timón de un auto con el ángulo predicho de la imagen extraída del dataset. Además, estas imágenes se combinan en un archivo de vídeo para ver la simulación de recorrido del auto ent tiempo real con las correcciones de timón.

Además se al igual que el modelo del paper, se realizó la búsqueda de los casos donde el ángulo predicho difiere de gran magnitud con el valor real. Y se presenta lo que el modelo visualiza para ese caso.

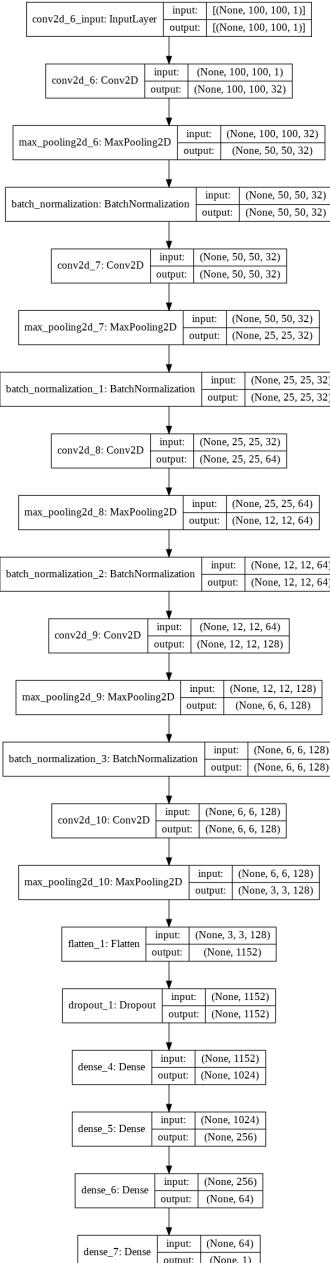


Figura 8. Arquitectura del modelo propuesto

Se presentan los siguientes casos encontrados.

Caso 1: Peor Caso

Para la imagen 13, se observa un árbol al medio pero lo que el dataset indica es que tiene un ángulo de rotación de $-5,7622046$ pero el valor predicho para esta imagen es de $0,124085106$, es decir, en realidad el sentido de giro del auto es hacia la izquierda, quizás se trata de una vuelta en U, pero el modelo se aferra a un ligero ajuste hacia la derecha. Datos como estos, requieren del uso de los anteriores datos y relacionarlo de manera correcta, los cuales eran la presión sobre el acelerador, el torque en las llantas, la velocidad del



Figura 9. Ejemplo de data en ciudad

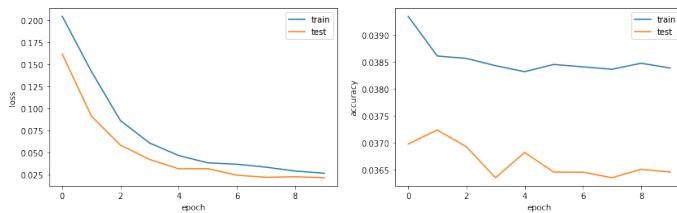


Figura 10. Loss y Accuracy del modelo usado en el modelo del paper

vehículo y la ubicación GPS.

Además, lo que el modelo extrae del la figura 13 es la figura 14, donde se puede apreciar una ligera extracción del borde de la cera del frente.

Caso 2: Mejor Caso

Para la imagen 15, se observa que se esta yendo al medio de la carretera y el dataset indica es que tiene un ángulo de rotación de 0,31328663 pero el valor predicho para esta imagen es de 0,313254, es decir, el modelo predijo con un error muy pequeño.

Además, lo que el modelo extrae del la figura 15 es la figura 16, donde se puede apreciar indicios de extracción de bordes y contornos del camino.

VI. CONCLUSIÓN Y TRABAJOS FUTUROS

Como trabajo a futuro sobre este proyecto, es principalmente el entrenamiento en conjunto con los datos de presión sobre el acelerador, velocidad, torque sobre las ruedas, velocidad y la ubicación por GPS. Así como, probar este modelo sobre las calles del Perú y verificar si se necesita algún ajuste sobre el tratamiento de datos y que otros datos más se podría usar.

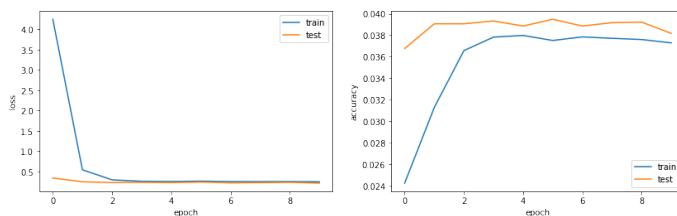


Figura 11. Loss y Accuracy del modelo usado en el modelo propuesto

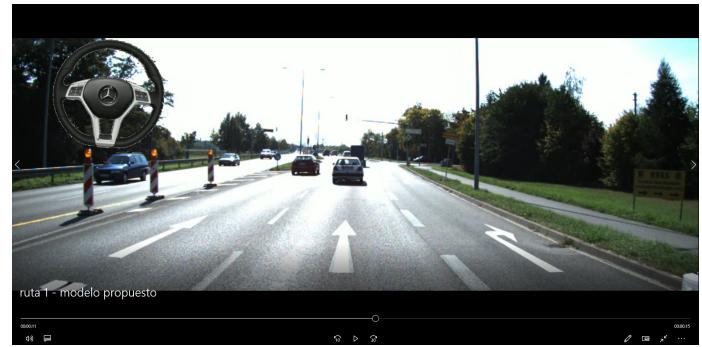


Figura 12. Captura de pantalla del video con el ángulo del timón predicho

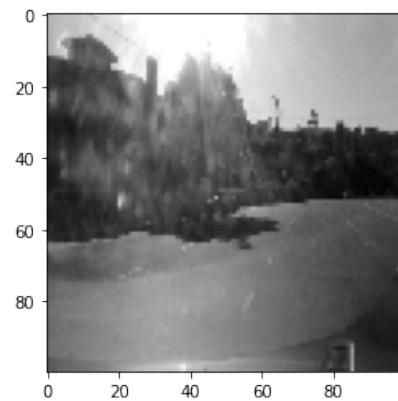


Figura 13. Caso 1: Imagen de test con un árbol al frente

También, las conclusiones de este proyecto, son se puede adaptar un modelo de self-driving a partir solo de las imágenes del recorrido de un auto y el ángulo de rotación del timón. También, se pudo comprobar que usar otro tipo de capas en el modelo, donde no necesariamente sea óptimo, el resultado puede ser mucho mas satisfactorio en los datos de prueba ajenos al dataset de entrenamiento.

VII. ANEXOS

Código Github: [@fmaguinaa](#)

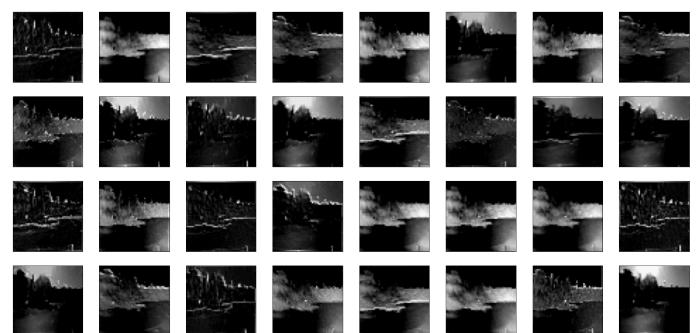


Figura 14. Extracción de características dentro del modelo, especialmente en la primera capa

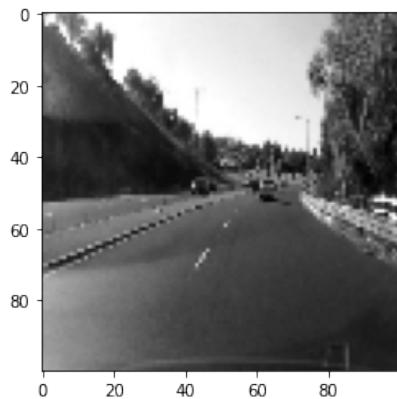


Figura 15. Caso 1: Imagen de test sobre la carretera

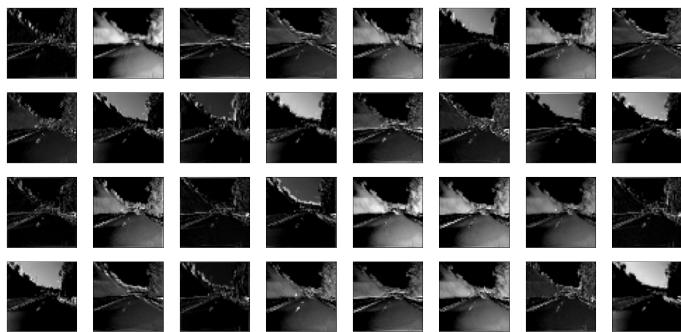


Figura 16. Extracción de características dentro del modelo, especialmente en la primera capa

REFERENCIAS

- [1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowsk, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, Karol Zieba (2016) End to End Learning for Self-Driving Cars. [Paper de Nvidia](#) .