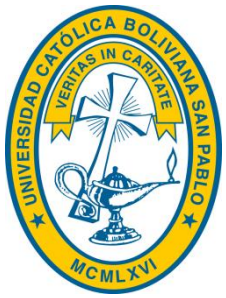
DML – 013/2020



Universidad Católica Boliviana "San Pablo"
Facultad de Ingeniería
DIPLOMADO EN *MACHINE LEARNING*
(1RA VERSIÓN)

ARTÍCULO DE INVESTIGACIÓN EN FORMATO IEEE
PRESENTADO A LA FACULTAD
DE INGENIERÍA PARA LA OBTENCIÓN DEL GRADO ACADÉMICO
DEL DIPLOMADO EN MACHINE LEARNING
(1RA VERSIÓN)

MODELO DE CONDUCCIÓN AUTÓNOMA BASADO EN ENFOQUE END-TO-END

Realizado por:

DAVY ALFONSO ROJAS YANA

2020

Modelo de conducción autónoma basado en enfoque End-to-end

Documento para optar por el Título de Diplomado en *Machine Learning*

Davy Alfonso Rojas Yana

Resumen — **Los sistemas autónomos para la conducción de vehículos han ganado relevancia recientemente debido a varios factores. Uno de los factores más importantes para adoptar el enfoque end-to-end es la basta disponibilidad de información proveniente de horas de conducción realizadas por humanos que son grabadas y paralelamente almacenada la información de parámetros internos del vehículo. Igual de importante es que se requiere menor procesamiento utilizando este enfoque. Ya que dicho enfoque usa generalmente como entrada del sistema imágenes captadas por una o varias cámaras, y como salida el valor del acelerador, freno y ángulo de giro (entre otros parámetros) que deberá tener el vehículo. A diferencia del enfoque tradicional que descompone el problema en componentes más técnicos como son: detección de vía, planeación de ruta, sistema de control para el giro, etc. Adicionalmente se aborda el caso del enfoque end-to-end cuando el vehículo requiere dar un giro en una intersección, dicha tarea no puede ser abordada solamente con el uso de una imagen como entrada, sino que deberá contar con una entrada condicional que indique hacia donde se requiere hacer el giro, para lo cual se propone una estructura ramificada en la arquitectura. En el presente trabajo, se aborda el enfoque end-to-end utilizando el simulador de conducción CARLA, utilizando un dataset provisto por los desarrolladores de dicho simulador que consta de horas de grabación de conducción de un vehículo el cual incluye imágenes captadas por una cámara y un vector con parámetros diferentes de mediciones del vehículo.**

***Palabras clave* —End-to-end, conducción autónoma, CNN, aprendizaje por imitación**

***Línea de investigación* — Automatización y Robótica**

I. Introducción

De De acuerdo a reportes de la Administración Nacional de Seguridad del Tráfico en Carreteras (NHTSA) de Estados Unidos, el 94% de los accidentes en carreteras son causados por errores humanos [1]. Para contrarrestar este tipo de estadísticas los sistemas de conducción autónoma están siendo desarrollados bajo la premisa de en un futuro prevenir accidentes, reducir emisiones, ayuda a gente con discapacidad entre otros.

El enfoque tradicional al abordar este tipo de soluciones divide la tarea en varias partes como son el seguimiento de carril [2], planeación de ruta [3] y lógica de control, y usualmente son tópicos que son estudiados por separado, a pesar de que este enfoque tradicional demuestra ser efectivo en muchos casos, presenta algunos inconvenientes, uno de los principales desde el punto de vista técnico es el error que puede ser acumulado de las etapas previas del procesamiento a la siguiente etapa, llegando a producir algunas veces un resultado impreciso. De aquí el interés por estudiar el desempeño de sistemas basados en el enfoque end-to-end.

El enfoque end-to-end para el desarrollo de proyectos de vehículos autónomos ha ganado interés últimamente. Debido a que se ha generado y existe bastante información que puede ser aprovechada por este enfoque y porque las demostraciones de conducción hechas por humanos son fáciles de recolectar. Dadas estas demostraciones, el aprendizaje por imitación (imitation learning) puede ser utilizado para entrenar un modelo que mapea entradas perceptivas como pueden ser imágenes a comandos de control como son ángulo de giro y aceleración de un vehículo. Lo que también se traduce en un procesamiento reducido comparado con el enfoque tradicional. Adicionalmente si hablamos de imágenes y reconocimiento de características, en este tipo de sistemas no existe la necesidad de reconocer categorías de objetos pre-definidos para etiquetar esos objetos durante el entrenamiento o el desarrollo de la lógica de control basado en el reconocimiento de dichos objetos. Por lo tanto, menos esfuerzo manual es requerido. Un diagrama de un sistema end-to-end genérico para conducción autónoma y su comparación con el enfoque tradicional se ve en la Figura 1.

A pesar de las ventajas del enfoque end-to-end, estos sistemas presentan limitaciones características, por ejemplo, una es la suposición de que la acción óptima se puede inferir solo de la entrada perceptiva, es decir, que se puede determinar el valor del ángulo de giro solamente basado en la percepción de la imagen tomada por una cámara. Esta suposición no es válida en la práctica, por ejemplo, cuando el vehículo se acerca a una intersección, la entrada de la cámara no es suficiente para inferir si el vehículo debe girar a la derecha, izquierda o debe seguir en dirección recta. Incluso si el sistema basado en redes neuronales de algúna forma pudiera

Este trabajo fue entregado para su revisión en fecha 19 de junio de 2020. El autor declara que el trabajo es de su completa autoría y referencia adecuadamente otros trabajos.

D. A. Rojas, cursa actualmente el Diplomado en Machine Learning en el Departamento de Ingeniería Mecatrónica, en la Universidad Católica Boliviana “San Pablo”, La Paz, Bolivia (e-mail: da.rojas@acad.ucb.edu.bo).

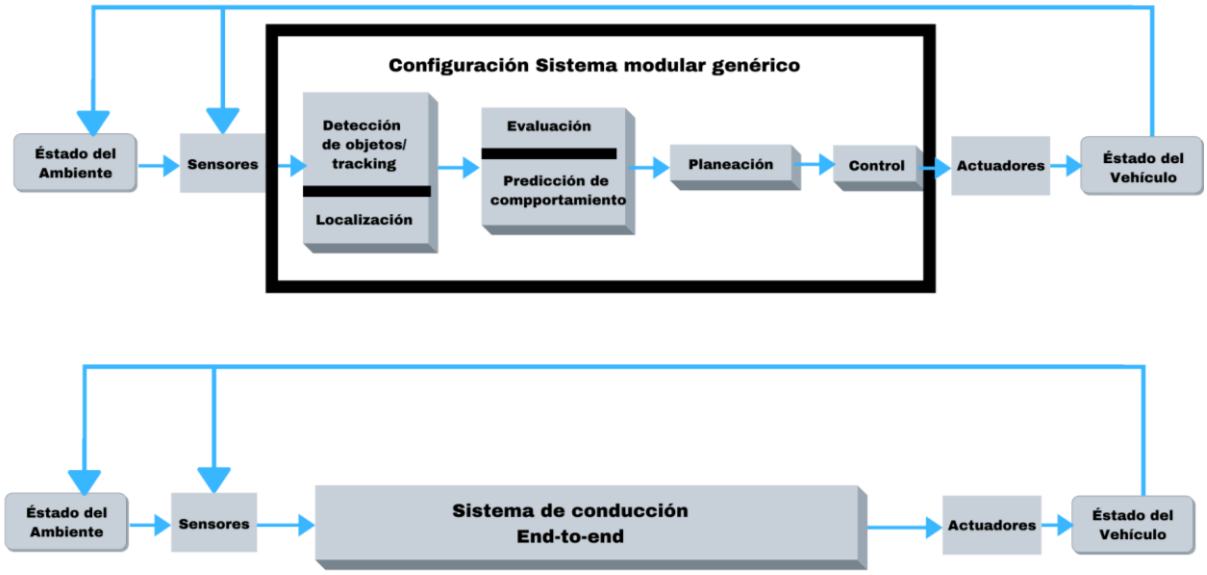


Fig. 1. Diagramas de un sistema modular genérico (arriba) y el modelo End-to-end para conducción autónoma (abajo)

resolver la anterior ambigüedad y tomar alguna dirección para avanzar, esta podría no ser la dirección a la que desea ir el pasajero, quien no cuenta con un canal de comunicación para controlar la red misma.

Es por esto, que en este trabajo se añade como entrada al sistema además a la entrada perceptual (imagen obtenida por cámara) y algunos parámetros de medición interna del vehículo (velocidad), una entrada que representa la intención del experto en dirigir el vehículo. Este comando sirve para resolver la ambigüedad en la toma de decisión del sistema y permite que sea controlado por la intención del pasajero o un planeador topológico, tal como aplicaciones de mapas proveen de indicaciones de dirección a los conductores.

Dado que el desarrollo de este tipo de sistemas requiere altos costes para el entrenamiento y la prueba en un ambiente real. La posibilidad de trabajar en un ambiente simulado es una de las mejores alternativas, siendo una herramienta que ha democratizado el desarrollo de sistemas de conducción autónoma. En el trabajo se ha utilizado CARLA Simulator [4] y un dataset provisto por los desarrolladores del mismo para realizar el entrenamiento y las pruebas correspondientes.

II. Antecedentes

Los primeros trabajos relacionados con el enfoque end-to-end para conducción autónoma se remontan a el trabajo presentado por Pomerleau llamado ALVINN [5] donde una fully-connected network (FCN) de 3 capas fue entrenada para predecir la dirección que el vehículo debía seguir. Otros trabajos relacionados como el de LeCun y otros [6], proponen un procesamiento remoto, y se propone una red neuronal convolucional (CNN) de 6 capas que permite la navegación de un robot con la capacidad de eludir obstáculos. Y el trabajo de Bojarski y otros [7], donde se presenta el enfoque que sería base en el desarrollo de sistemas end-to-end para vehículos autónomos moderno, desarrollado por investigadores de NVIDIA, donde se aplica una CNN para mapear píxeles sin procesar de una sola cámara frontal directamente a los comandos de dirección del vehículo.

Adicionalmente al enfoque end-to-end, el trabajo de Codevilla y otros [8] desarrolla una formulación condicional que posibilita la aplicación del aprendizaje por imitación a un entorno de conducción urbano. El presente trabajo adopta este tipo de formulación, donde adicionalmente a una entrada de imagen por parte de una cámara al sistema, se toma en cuenta con una entrada que especifica la intención de direccionamiento por parte del pasajero. Lo cual es de ayuda para resolver la ambigüedad que existiría en el agente al tratar de evaluar la dirección a seguir cuando se encuentra con una intersección como se mencionó anteriormente.

El presente trabajo se ocupa de la conducción basada en visión end-to-end utilizando redes profundas. Los sistemas en este dominio se han limitado a imitar al experto sin la capacidad de aceptar comandos naturalmente después de la implementación. Entonces se trata de introducir dicha capacidad en redes profundas para una conducción basada en la visión end-to-end.

Debido a las dificultades encontradas para realizar el entrenamiento y pruebas en ambientes reales, simuladores de conducción son generalmente utilizados en trabajos relacionados. Algunos simuladores populares utilizados son el juego de carreras de código abierto TORCS [9], y el juego comercial Grand Theft Auto V (GTA V) [10], aunque debido a que TORCS presenta un ambiente demasiado sencillo, y que en GTA V no existe mucha libertad para personalizar y controlar el ambiente de desarrollo, se optan por nuevas opciones mas adecuadas para el desarrollo. Tal es el caso del simulador de código abierto CARLA [4] el cual provee una correspondencia entre realismo y flexibilidad, abordando los problemas descritos por los otros simuladores.

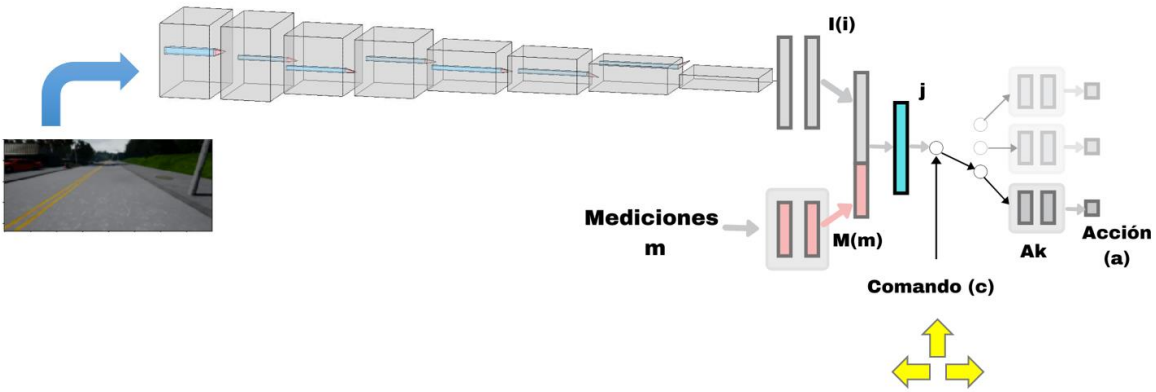


Fig. 2. Arquitectura ramificada propuesta en el trabajo en el que se observan los diferentes módulos

A. *Aprendizaje por imitación condicional*

La idea básica del aprendizaje por imitación es entrenar un controlador que imite a un experto. Considerando un dataset con pares de acción-observación $D\{\langle \boldsymbol{o}_i, \boldsymbol{a}_i \rangle\}_{i=1}^{N}$, recolectado a partir de demostraciones de expertos. El objetivo en el aprendizaje por imitación es tratar de aprender una función de la forma $F_\theta(\boldsymbol{o}_t) : O \rightarrow A$ que como se ha mencionado mapee las observaciones $\boldsymbol{o}_t$ a acciones $\boldsymbol{a}_t$ en cada paso de tiempo y sea capaz de imitar al experto. Los parámetros θ de la función son optimizados minimizando la distancia L entre la acción predicha y la acción real del experto:

$$\min_{\theta} \sum_{i} L(F_\theta(\boldsymbol{o}_i; \theta), \boldsymbol{a}_i) \tag{1}$$

El aprendizaje por imitación condicional busca adicionalmente condicionar la función anterior con un comando de alto nivel $\boldsymbol{c}$ que pueda transmitir la intención del usuario en el momento de la prueba. Dicho comando puede servir para desambiguar el comportamiento del agente: por ejemplo, cuando el vehículo se acerca a una intersección, la imagen entregada por la cámara no es suficiente para predecir si el vehículo debe girar a la izquierda o a la derecha, o continuar recto. Incluso si se logra entrenar un controlador que sea capaz de dar giros y evitar colisiones, no sería verdaderamente útil, ya que tomaría decisiones arbitrarias al encontrarse con una intersección. Para lidiar con este inconveniente es necesario implementar un canal de comunicación entre el controlador y el usuario, o proveer al controlador comandos respecto a que giro debe tomar. Este comando puede ser entonces una entrada del usuario o proveniente de un módulo de planeación como podría ser un GPS, un comando típico podría ser: "Girar a la izquierda en la siguiente intersección". Entonces, proveyendo a la función con el comando $\boldsymbol{c}$ se ayuda a resolver la ambigüedad. El objetivo entonces en este casi puede ser escrito como:

$$\min_{\theta} \sum_{i} L(F_\theta(\boldsymbol{o}_i, \boldsymbol{c}_i; \theta), \boldsymbol{a}_i) \tag{2}$$

III. Metodología

A continuación se describe la implementación práctica del sistema basado en enfoque end-to-end con aprendizaje por imitación condicional.

A. *Arquitectura del Modelo*

Se plantea que cada observación $\boldsymbol{o} = \langle \mathbf{i}, \mathbf{m} \rangle$ Comprende una imagen $\mathbf{i}$ y un vector de mediciones $\mathbf{m}$. El controlador F se representa por una red neuronal profunda. La red es alimentada con la imagen, las mediciones y el comando $\boldsymbol{c}$, y produce la acción $\boldsymbol{a}$ como salida. En este caso $\boldsymbol{a}$ tiene un dominio continuo y es un vector con dos componentes: ángulo de giro y aceleración. Mientras que $\boldsymbol{c}$ es una variable categórica representada por un vector con valores únicos.

La arquitectura propuesta se ilustra en la Figura 2 y se plantea de la siguiente manera, la red toma como entradas la imagen y las mediciones. Estas dos entradas se trabajan en módulos independientes. Un módulo de imagen $I(\mathbf{i})$, un módulo de mediciones $M(\mathbf{m})$. Donde el módulo de imagen se implementa como una CNN, el módulo de mediciones como una red fully-connected (FCN). Se concatenan las salidas de estos módulos en una representación conjunta $\boldsymbol{j}$. Adicionalmente se asume un conjunto de comandos discretos $C = \{c^0, \dots, c^k\}$. Y se introduce una ramificación especializada $\boldsymbol{A}^i$ para cada uno de los comandos $\boldsymbol{c}^i$. $\boldsymbol{c}$ funciona como un selector que decide cual ramificación será utilizada en cada paso determinado. Entonces, la salida de la red viene a ser:

$$F(\boldsymbol{i}, \boldsymbol{m}, \boldsymbol{c}^i) = \boldsymbol{A}^i\big(J(\boldsymbol{i}, \boldsymbol{m})\big) \tag{3}$$

Se puede decir que en esta arquitectura ramificada, cada ramificación $\boldsymbol{A}^i$ aprende ciertas sub funciones profundas, como podría ser que una ramificación se especialice en el seguimiento del carril, otra en giros a la derecha y la otra en giros a la izquierda y todas estas ramificaciones comparten los módulos previos de imagen y medición.

B. *Detalles del modelo*

Para el controlador la observación $\boldsymbol{o}$ es la imagen de una cámara central ubicada en el vehículo (que se detalla más adelante) de una resolución de 200×88 pixeles RGB (tres canales). Para las mediciones finalmente solo se utiliza la medición de la velocidad actual del vehículo. El módulo de imagen $I(\mathbf{i})$ se compone de 8 capas convolucionales de dimensiones 32, 32, 64, 64, 128, 128, 256, 256, el tamaño del

kernel en la primera capa es de 5 y de 3 en el resto de las capas. La primera, tercera y quinta capa tienen un salto (stride) de 2. Adicionalmente el módulo de imagen cuenta con dos capas fully-connected las cuales contienen 512 neuronas cada una.

El módulo de medición cuenta con dos capas fully-connected de 128 neuronas cada una. Las salidas del módulo de imagen y de medición son concatenadas y se aplica una capa fully-connected de 512. Se ha aplicado ReLU después de todas las capas ocultas, aplicando 50% de dropout después de cada capa fully-connected, y 20% de dropout después de cada capa convolucional. La acción $\boldsymbol{a}$ es un vector de dos dimensiones que cuenta con los valores de ángulo de giro y aceleración: $\boldsymbol{a} = \langle g, a \rangle$. Entonces, dada una acción predicha y una acción perteneciente a la verdad fundamental $\boldsymbol{a_{vf}}$. La función de perdida se define como en el trabajo [8]:

$$l(\boldsymbol{a}, \boldsymbol{a_{vf}}) = l(\langle s, a \rangle, \langle s_{vf}, a_{vf} \rangle) = \|s - s_{vf}\|^2 + \lambda_a \|a - a_{vf}\|^2 \quad (6)$$

C. Dataset

La información para el entrenamiento es provista por una de las secciones de desarrollo del simulador CARLA. En el trabajo de Codevilla y otros [8] se describe a detalle cómo fue obtenido el dataset y bajo que parámetros. Entre los más importantes se resaltan lo siguiente: De que se trata de un dataset que es recolectado a partir de demostraciones de un humano llevando a cabo la tarea de conducir el vehículo en el simulador. Para incrementar la robustez en trabajos posteriores tomaron la decisión de tomar en cuenta el entrenamiento cuando el conductor se encuentra con alguna perturbación, por lo que el dataset de entrenamiento contiene observaciones de recuperaciones después de que se presenta una perturbación. Adicionalmente el dataset es obtenido utilizando 3 cámaras simultáneamente. Una cámara enfocando al frente y las otras dos rotadas a la izquierda y a la derecha, rotadas 30 grados entre sí.

Para incrementar la capacidad de generalización en diferentes controladores a desarrollar, también al dataset se le han aplicado procedimientos de aumento de datos, estas son transformaciones que incluyen cambio en el contraste de las imágenes, brillo, tono, adición de ruido Gaussiano, entre otros. Adicionalmente a las observaciones (imágenes) y acciones, en el dataset también se encuentran registrados los comandos de alto nivel provistos por el conductor, resumidos en un grupo de cuatro comandos: continúe (seguir el camino), “left” (girar a la izquierda en la intersección), “right” (girar a la derecha en la intersección), “straight” (seguir recto en caso de una intersección). Durante la recolección de información para el entrenamiento, al aproximarse a una intersección se añadió una interfaz por la que el conductor indicaba en qué dirección deseaba dirigirse, para indicar el comando correspondiente al curso de acción previsto. El dataset puede ser consultado en el Apéndice 1.

IV. Configuración del sistema

Se ha evaluado el presente enfoque en un ambiente simulado, las observaciones utilizadas solamente son las imágenes que genera la cámara frontal, no tomando en cuenta las dos cámaras laterales mencionadas previamente. La señal de control grabada tiene dos dimensiones, y posee los valores del ángulo de giro y la aceleración, (el parámetro de frenado es calculado simplemente con el valor decreciente del valor de la aceleración). Ambos valores (ángulo de giro y frenado) poseen un dominio continuo de valores entre -1 y 1. Donde en el caso del ángulo de giro -1 representa un giro completamente a la izquierda y un ángulo completamente a la derecha. En el caso de la aceleración 1 representa una aceleración a fondo y -1 una aceleración en reversa a fondo.

Fig. 3 : Demostración de los ambientes simulados, las imágenes de arriba corresponden a los mapas de cada ciudad y las de abajo una vista típica. Las imágenes de la izquierda corresponden a la Ciudad 1 y las de la derecha a la Ciudad 2.

Se utilizó el simulador CARLA [4] un simulador de conducción urbana en su versión 0.8.2, esta versión contiene 2 ciudades diseñadas realísticamente con vegetación, edificios, señales de tráfico y movimiento de peatones. Se pueden editar las escenas de simulación, editando las condiciones climáticas, cantidad de vehículos entre otros. En la Figura 3 se observan los mapas y algunas vistas de estas ciudades, la Ciudad 1 (Town 1) fue utilizada exclusivamente para el entrenamiento y la Ciudad 2 (Town 2) para las pruebas.

Adicionalmente a la información mencionada en la recolección del dataset, el simulador porvee información extra como ser la distancia recorrida, colisiones y la ocurrencia de infracciones como invadir parcialmente un carril o intersectar con la acera. La gran ventaja de utilizar este simulador es que cuenta con una Interfaz de Programación de Aplicaciones (Application Programming Interface) o API para Python. Para este trabajo se ha utilizado Python en su versión 3.5 y Tensorflow 1.14

V. Experimentos y Resultados

CARLA es un simulador bastante completo desarrollado como open-source, justamente para democratizar el desarrollo de sistemas de conducción autónoma, por lo que cuenta con una aplicación en Python para evaluar los diferentes agentes o controladores que se desarrollan. Llamado benchmark, esta aplicación nos brinda una interface para conectar nuestro controlador con el simulador y realizar los experimentos

automáticamente y devolviendo los resultados de los mismos.

Lo que hace el benchmark es básicamente ejecutar episodios de simulación y medir la efectividad del vehículo comandado por nuestro controlador para cumplir ciertas tareas. En cada episodio el vehículo es inicializado en una ubicación en el mapa diferente y debe conducir hasta un punto de destino dado, el vehículo en el trayecto es provisto de comandos de alto nivel para realizar los giros correspondientes, estos comandos los proporciona un planeador topológico incorporado en el simulador. Se cuenta como un episodio exitoso cuando el vehículo culmina el objetivo en un intervalo de tiempo dado. Además de la tasa de éxito, se mide la calidad del manejo, midiendo la distancia media recorrida sin cometer infracciones.

Las dos ciudades mencionadas se utilizan en la experimentación. La Ciudad 1 es usada para el entrenamiento y validación, y solo la Ciudad 2 es usada para la etapa de prueba.

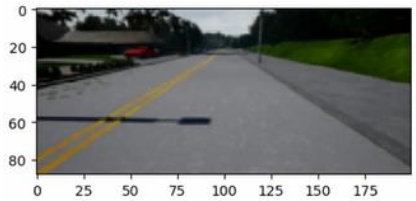


Fig. 4 : Ejemplo de una imagen captada por la cámara localizada en el vehículo

Para la evaluación fueron usados 60 pares de ubicaciones de inicio-fin que son localizadas con una distancia mínima de 1 kilómetro para cada ciudad. Aclarar que el dataset de entrenamiento comprende 2 horas de conducción humana en la Ciudad 1 y solamente 12 minutos contienen demostraciones con perturbaciones como se mencionó previamente. Como parte de los experimentos se ha visto importante mostrar las el tipo de imagen que es captado por la cámara situada en el vehículo, que se puede ver en la Figura 4. Y algunas de las convoluciones obtenidas internamente por la red, estas se pueden evidenciar en la Figura 5.

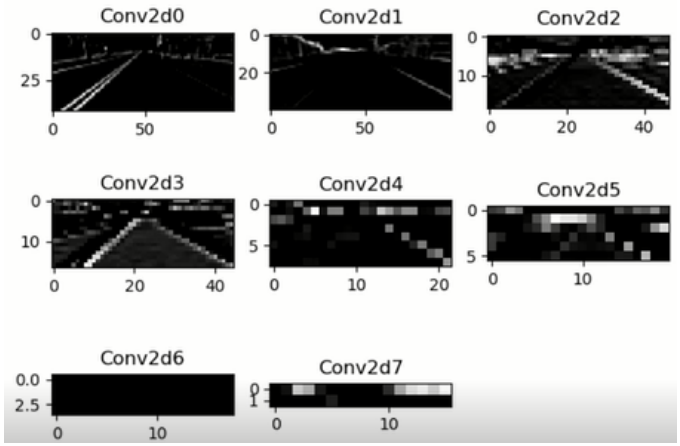


Fig. 5 : Visualización de algunas convoluciones captadas en los ocho diferentes bloques convolucionales

A. Resultados

Comparamos nuestro sistema con cuatro enfoques de referencia presentados en [8], dos de estos enfoques son muy básicos, estos enfoques son: Un enfoque de aprendizaje por imitación estándar y otro de aprendizaje de imitación condicionado a objetivos. En el enfoque estándar la acción $\boldsymbol{a}$ es predicha solamente tomando en cuenta la observación $\boldsymbol{o}$ y las mediciones $\boldsymbol{m}$. En el enfoque condicionado a objetivos el controlador es provisto en la entrada con un vector de direcciones como el visto en nuestro caso, y es agregado paralelamente a las imágenes a la red neuronal.

TABLA 1
DESEMPEÑO DE LOS MÉTODOS DE CONDUCCIÓN AUTÓNOMA END-TO-END

Modelo	**Tasa de Éxito**		**Km por infracción**	
	Ciudad 1	**Ciudad 2**	**Ciudad 1**	**Ciudad 2**
No-condicional (Imitación estandar)	20%	26%	5,76	0,89
Goal- condicional (condicionado a objetivos)	24%	30%	1,87	1,22
Branched (*)	**88%**	**64%**	**2,34**	**1,18**
Cmd. Input (*) (Enfoque con comando de entrada)	78%	52%	3,97	1,30
Trabajo propuesto	**77%**	**58%**	**3,64**	**1,20**

Se muestran los diferentes resultados de los modelos relacionados mencionados para hacer una comparación con el trabajo propuesto. Como se mencionó las columnas de tasa de éxito se refieren al porcentaje de episodios completados exitosamente en cada ciudad. Y las columnas de kilómetros por infracción hacen referencia a la cantidad de kilómetros en promedio transcurridos antes de la primera infracción en un episodio.
*Son las arquitecturas similares al trabajo propuesto, que toman en cuenta el comando $\boldsymbol{c}$.

Los otros dos enfoques son más completos y vienen a ser bastante relacionados al enfoque presentado en este trabajo. El enfoque con comando de entrada es el primero, este incluye un módulo para el tratamiento del vector de direcciones, para luego agregarlo paralelamente al módulo de imagen. El siguiente es un modelo al que se le denomina Branched, que presenta básicamente el mismo tipo de arquitectura al propuesto en el presente trabajo, es decir una estructura ramificada condicionada por el comando $\boldsymbol{c}$.

Dichos resultados se ven en la Tabla 1. El controlador que utiliza aprendizaje por imitación estándar completa exitosamente solo el 20% de los episodios en la ciudad 1 y 24% en la ciudad 2, es comprensible debido a que no cuenta con información acerca de la ruta que debe seguir. El controlador con aprendizaje por imitación condicionado a objetivos se desempeña apenas mejor que el anterior completando el 24% de los episodios en la Ciudad 1 y el 30% en la Ciudad 2.

El enfoque denominado Branched es el que mejor desempeño reporta comparando todos los controladores, completando exitosamente el 88% de los episodios en la Ciudad 1 y el 64% en la Ciudad 2. El enfoque con comando de entrada reporta una tasa de éxito de 78% y 52% en la Ciudad 1 y 2 respectivamente. Con estos resultados queda claro que para el desarrollo de este tipo de sistemas es impresindible

contar con el comando $\boldsymbol{c}$ que guie la dirección del vehículo para completar los episodios exitosamente. Es por eso que el trabajo hace uso de ese recurso, obteniendo como resultados una tasa de éxito de 77% en la Ciudad 1 y de 58% en la Ciudad 2. Obteniendo un mejor desempeño que al menos tres de los anteriores controladores. Sin embargo a pesar de que se plantea una arquitectura similar a la Branched, no se ha podido superar ni igualar los resultados presentados en [8] para dicha arquitectura, esto se puede deber a varios factores que se analizan en las conclusiones.

VI. Conclusiones

Se ha propuesto un modelo basado en end-to-end utilizando el aprendizaje por imitación condicional adicionando un comando $\boldsymbol{c}$, es un enfoque para aprender de demostraciones expertas de controles de bajo nivel y comandos de alto nivel. En el momento del entrenamiento, los comandos resuelven ambigüedades en el mapeo perceptual motor, lo que facilita el aprendizaje. En el momento de la prueba, los comandos sirven como un canal de comunicación que puede usarse para dirigir el controlador, dicho comando en sistemas más integrados puede corresponder a las indicaciones dictadas por un sistema GPS.

Aplicamos el enfoque presentado a la conducción basada en visión en simulaciones realistas de entornos urbanos dinámicos. Nuestros resultados muestran que la formulación condicional con el comando $\boldsymbol{c}$ mejora significativamente el rendimiento comparado con los enfoques básicos con los que se ha comparado en la Tabla 1.

Si bien el trabajo propuesto sigue una línea bastante al controlador denominado Branched en [8], los resultados obtenidos son un poco inferiores, esto se puede deber a muchos factores, entre ellos podríamos destacar el ajuste de los hiperparámetros, donde nuestro trabajo ha tomado una taza de aprendizaje mayor al del otro trabajo, debido a que los tiempos de entrenamiento se prolongaban excesivamente. Dado esto, también es importante señalar la capacidad de procesamiento, ya que su trabajo utiliza una GPU Nvidia TITAN Xp [11], mientras que nosotros trabajamos con una GPU Geforce GTX 1060 [12]

Apéndices

Apendice 1: Dataset de entrenamiento

El dataset con el que se trabajó en el desarrollo del proyecto puede ser accedido en el siguiente enlace:

- https://drive.google.com/file/d/1hloAeyamYn-H6MfV1dRtY1gJPhkR55sY/view

Apendice 2: Repositorio de archivos en Python

Es el repositorio en Github de los archivos con los que fue desarrollado el proyecto.

- https://github.com/Davydero/End-to-end

Referencias

[1] S. Singh, “Critical reasons for crashes investigated in the national motor vehicle crash causation survey,” Tech. Rep., 2015.

[2] A. J. Humaidi and M. A. Fadhel, “Performance comparison for lane detection and tracking with two different techniques,” in 2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA), May 2016, pp. 1–6

[3] C. Li, J. Wang, X. Wang, and Y. Zhang, “A model based path planning algorithm for self-driving cars in dynamic environment,” in 2015 Chinese Automation Congress (CAC), Nov 2015, pp. 1123–1128.

[4] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. arXiv preprint arXiv:1711.03938..

[5] D. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In NIPS, 1988.

[6] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp. Off-road obstacle avoidance through end-to-end learning. In NIPS, 2005

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. arXiv:1604.07316, 2016

[8] Codevilla, F., Miiller, M., López, A., Koltun, V., & Dosovitskiy, A. (2018, May). End-to-end driving via conditional imitation learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1-9). IEEE.

[9] J. Huang, I. Tanev, and K. Shimohara. Evolving a general electronic stability program for car simulated in TORCS. In CIG, 2015.

[10] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In ECCV, 2016.

[11] *Titan Xp User Guide*, NVIDIA Corporation, Santa Clara, CA, USA, 2017. [Online]. Disponible : https://www.nvidia.com/content/geforce-gtx/NVIDIA_TITAN_Xp_User_Guide.pdf

[12] *GEFORCE GTX 1060*, NVIDIA Corporation, Santa Clara, CA, USA, 2016. [Online]. Disponible : https://www.nvidia.com/content/geforce-gtx/GTX_1060_User_Guide.pdf

[13] Hawke, J., Shen, R., Gurau, C., Sharma, S., Reda, D., Nikolov, N., ... & Kendall, A. (2019). “Urban driving with conditional imitation learning”. arXiv preprint arXiv:1912.00177.

Davy A. Rojas nació en La Paz, Bolivia en 1992. Recibió el título de licenciatura en ingeniería Mecatrónica en la Universidad Católica Boliviana “San Pablo”, en La Paz en 2015. Actualmente se encuentra concluyendo sus estudios de especialidad en Aplicaciones de la Tecnología Nuclear en las instituciones de: Universidad Politécnica de Tomsk (Tomsk-Rusia) y la Agencia Boliviana de Energía Nuclear en La Paz. A su vez se encuentra cursando el diplomado en Machine Learning como parte del programa de Maestría en Sistemas Mecatrónicos en la Universidad Católica Boliviana “San Pablo”.

De 2015 a 2018 fue Ingeniero de Investigación y Desarrollo en Automatización en la empresa SIMA S.R.L. En 2019 llega a ser parte de la Agencia Boliviana de Energía Nuclear en calidad de becario para posteriormente incorporarse como

Ingeniero de Automatización en el Centro Multipropósito de Irradiación.

Algunos de los reconocimientos del Ing. Rojas incluyen haber obtenido el primer premio a nivel nacional en el SpaceApps Challenge de la NASA en 2018 y obtener la beca de estudios en Aplicaciones de la Tecnología Nuclear en la Universidad Politécnica de Tomsk en 2019.