



INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

INFORME TÉCNICO

TEMA:

PLAIDML (compilador de tensor portátil)

PROFESOR:

PARRA HERNÁNDEZ ABIEL TOMAS

PRESENTA:

AVILA GOMEZ GIOVANNI ARTURO-181080041 33.33%

GARCIA PACHECO AXEL DAVID-181080026 33.33%

ROJAS PEREZ JOSE RAMON-181080011 33.33%

Total de un 100%



Índice

| | |
|--|----|
| 1. RESUMEN | 1 |
| 2. INTRODUCCIÓN | 2 |
| 3. OBJETIVO GENERAL | 3 |
| 3.1 OBJETIVO ESPECÍFICO | 4 |
| 3.1.1 CPU | 4 |
| 3.1.2 GPU | 4 |
| 3.1.3 ACELERADORES | 4 |
| 4. JUSTIFICACIÓN | 5 |
| 5. METODOLOGÍA | 6 |
| 5.1 DESCRIPCIÓN DE LA METODOLOGÍA | 6 |
| 5.2 TABLA COMPARATIVA METODOLOGÍAS | 6 |
| 6. MARCO TEÓRICO | 7 |
| 6.1 COMPILADOR | 7 |
| 6.2 LLVM | 7 |
| 6.3 MLIR | 8 |
| 6.4 PLAIDML | 9 |
| 6.5 INTELIGENCIA ARTIFICIAL | 9 |
| 6.6 MACHINE LEARNING | 10 |
| 6.6.1 Aprendizaje iterativo | 11 |
| 6.6.2 Enfoques hacia el machine learning | 11 |
| 6.6.3 Aprendizaje supervisado | 11 |
| 6.6.4 Aprendizaje no supervisado | 12 |
| 6.6.5 Aprendizaje de refuerzo | 12 |
| 6.7 DEEP LEARNING | 12 |
| 6.7.1 Importancias del deep learning | 12 |
| 6.7.2 Oportunidades y aplicaciones del deep learning | 13 |
| 6.7.3 Cómo funciona el deep learning | 13 |
| 7. DESARROLLO E IMPLEMENTACION (REPORTE) | 15 |
| 8. RESULTADOS | 25 |
| 9. CONCLUSIONES | 31 |
| 10. FUENTES DE INFORMACION | 32 |

1. RESUMEN

A lo largo del siguiente escrito, conoceremos un poco acerca del PLAIDML desde el momento de su invención, hasta ya el momento de ocuparlo dentro de un ambiente ya funcional. Conoceremos más acerca de este compilador de TensorFlow ya que este fue una adquisición por parte del gigante tecnológico INTEL, teniendo como objetivo que aprendiera muy internamente en todas las plataformas, entonces ahí fue cuando Intel le comparte el público PlaidML.

Retomando un poco su principal función es el aprendizaje profundo, este está dirigido para computadoras portátiles y funciona como backend, funcionando directamente con Python (Siendo este un compilador), funcionando como un lenguaje de alto nivel, convirtiendo sus instrucciones que así mismo se pueden ejecutar al mismo tiempo que el hardware. Las especificaciones que tiene este compilador tenemos que tenerla bien seleccionada ya que por ser un compilador que su objetivo es aprender tiene que tener especificaciones para su correcto funcionamiento.

Ya que se toma el tema podemos observar los diferentes conceptos que se van a explorar para la conceptualización de lo que conforma a PlaidML, siendo los más importantes como lo son:

- CPU
- GPU
- ACELERADOR
- LLVM
- MLIR
- PLAIDML
- IA
- MACHINE LEARNING
- DEEP LEARNING

Teniendo desarrollado más a profundidad todos los conceptos antes mencionados se tendrá una mejor concepción de lo que significa el aprendizaje de una máquina, sus utilidades, su función y el cómo es su desarrollo. Toda esta información él acerca de su desarrollo se explicara más adelante, de acuerdo a la experimentación que nosotros ocupamos para su desarrollo dentro de Python.



2. INTRODUCCIÓN

En agosto de 2018, Intel adquirió Vertex.AI, un startup cuya misión era "aprendizaje profundo para todas las plataformas". Intel lanzó PlaidML como software gratuito bajo los términos de la Licencia Apache (versión 2.0) para mejorar la compatibilidad con nGraph, TensorFlow y otro software del ecosistema.

Es un compilador de tensor portátil y avanzado para permitir el aprendizaje profundo en computadoras portátiles, dispositivos integrados u otros dispositivos donde el hardware informático disponible no es bien compatible o la pila de software disponible contiene restricciones de licencia.

También se encuentra debajo de los marcos comunes de aprendizaje automático, lo que permite a los usuarios acceder a cualquier hardware compatible con PlaidML. PlaidML es compatible con Keras, ONNX y nGraph.

3. OBJETIVO GENERAL

TensorFlow sirve como backend para Keras, interpretando la sintaxis de Python de alto nivel de Keras y convirtiéndola en instrucciones que se pueden ejecutar en paralelo en hardware especializado como una GPU. PlaidML es un backend alternativo para Keras que admite marcos de paralelización distintos del CUDA de Nvidia. En un Mac, puede usar PlaidML para entrenar modelos Keras en su CPU, los gráficos integrados de su CPU, un procesador de gráficos AMD discreto o incluso una GPU AMD externa conectada a través de Thunderbolt 3.

Primero comencé a hurgar con PlaidML porque estaba buscando una forma de entrenar una red neuronal convolucional profunda en un conjunto de datos de imágenes muy grande. Intenté hacer esto en Colab de Google, pero la herramienta en línea resultó ser muy frustrante para trabajos de larga duración. Tenía una eGPU Radeon RX580 acumulando polvo, así que quería una forma de usarla para entrenar modelos localmente con mi MacBook.

Después de unos pocos pasos rápidos, estaba en funcionamiento con PlaidML. A continuación, le indicamos cómo puede usarlo en su sistema. Primero, instale PlaidML a través de pip. Recomiendo encarecidamente utilizar entornos virtuales aquí para aislar su instalación de PlaidML del resto de su sistema.

El poder de PlaidML viene con su simplicidad. Después de la instalación, activar su GPU es tan simple como ejecutar

Configuración de Plaidml

Después de seleccionar si desea habilitar las funciones experimentales, esta herramienta le preguntará qué dispositivo computacional le gustaría usar. Debería ver una lista como la siguiente:

- 1: llvm_cpu.0
- 2: metal_intel(r)_hd_graphics_530.0
- 3: metal_amd_radeon_pro_450.0
- 4: metal_amd_radeon_rx_580.0

La primera opción es mi CPU, la segunda son los gráficos integrados de Intel dentro de mi CPU, la tercera opción es la discreta GPU AMD en mi MacBook Pro de 15" y la cuarta opción es mi eGPU RX 580. Me encanta lo fácil que es para cambiar de procesador; esto me permite entrenar modelos simples sobre la marcha con la discreta GPU de mi computadora portátil y usar mi eGPU para tareas más pesadas.

La única advertencia aquí es que ya no tiene acceso a las funciones de TensorFlow, como los conjuntos de datos de TensorFlow. Todo el código que escriba debe estar en Keras puro. No he encontrado que esto sea una gran limitación y, de todos modos, conduce a un software más portátil. PlaidML también funciona con las GPU de Nvidia, por lo que si trabaja en un equipo que usa diferentes arquitecturas de GPU, PlaidML hace las cosas muy fáciles. Usar PlaidML como backend de Keras es tan simple como lo siguiente:

```
Desde el entorno de importación del sistema operativo
Entorno ["KERAS_BACKEND"] = "plaidml.keras.backend"
Importar keras
```

Y eso es literalmente. A continuación se muestra un ejemplo completo que puede probar en su propio sistema después de instalar PlaidML. Entrena una red neuronal muy simple con una capa oculta que suma los vectores de entrada.

```
importar numpy como np
desde el entorno de importación del sistema operativo
from keras.backend import 'plaidml.keras.backend'
importar keras
de keras . Importación de capas . Demo
de matplotlib importar pyplot como plt

# Parámetros
num_samples = 100000 ; vect_len = 20 ; max_int = 10 ; min_int = 1 ;

# Generar conjunto de datos
X = np . random ( min_int , max_int , ( num_samples , vect_len ) )
Y = np . suma ( X , eje = 1 )

# Obtén el 80% de los datos para entrenar
split_idx = int ( 0.8 * len ( Y ) )
train_X = X [: split_idx , :] ; test_X = X [ split_idx :, :]
train_Y = Y [: split_idx ] ; test_Y = Y [ split_idx : ]

# Haz un modelo
modelo = keras . modelos . Secuencial ( )
modelo . agregar ( keras . capas . Demo ( 32 , activación = 'relu' , input_shape = ( vect_len ,
modelo . añadir ( keras . capas . Demo ( 1 ) )
modelo . compilar ( 'adam' , 'mse' )

historia = modelo . ajustar ( train_X , train_Y , validation_data = ( prueba_X , prueba_Y ) , \
    épocas = 10 , tamaño_lote = 100 )

# resumir la historia
plt . trama ( historia . historia [ 'pérdida' ] )
plt . trama ( historia . historia [ 'val_loss' ] )
plt . título ( 'pérdida de modelo' )
plt . ylabel ( 'pérdida' )
plt . xlabel ( 'época' )
plt . leyenda ( [ 'entrenar' , 'prueba' ] , loc = 'superior izquierda' )
plt . mostrar ( )
```

3.1 OBJETIVO ESPECÍFICO

Funciona en los principales sistemas operativos: Linux, macOS y Windows, y es un compilador de tensor que facilita modelos de aprendizaje automático portátiles reutilizables y de rendimiento en varios objetivos de hardware, incluidas CPU, GPU y aceleradores.

3.1.1 CPU

La CPU se puede considerar el cerebro de un dispositivo. Muchas veces es llamado: procesador, procesador de ordenador o microprocesador. Aunque también se puede insertar en una unidad llamada "CPU Socket": es un tipo de zócalo electrónico instalado en la placa base, que permite conectar el microprocesador sin soldarlo.

3.1.2 GPU

La tecnología GPU aprovecha múltiples unidades de procesamiento de gráficos (GPU) profesionales para proporcionar un escalamiento inteligente de su aplicación y acelerar de forma impresionante el flujo de trabajo. Esto brinda un impacto comercial significativo en sectores como la fabricación, los medios y el entretenimiento y la exploración de energía.

3.1.3 ACELERADORES

Es un dispositivo que utiliza campos electromagnéticos para acelerar partículas cargadas a altas velocidades, y así, hacerlas colisionar con otras partículas. De esta manera, se generan una multitud de nuevas partículas que -generalmente- son muy inestables y duran menos de un segundo, esto permite estudiar más a fondo las partículas que fueron desintegradas por medio de las que fueron generadas. Hay dos tipos básicos de aceleradores de partículas: los lineales y los circulares. El tubo de rayos catódicos de un televisor es una forma simple de acelerador de partículas.

4. JUSTIFICACIÓN.

El desarrollo del proyecto de investigación, se realizará con base al compilador PlaidML por la necesidad de aprender de manera autónoma e introductoria la comprensión de dicho compilador y de la misma manera demostrar el punto de vista de cada uno de los integrantes del equipo con respecto a las necesidades de la información sobre características del tema.

Al momento de PlaidML encontramos pequeños datos de acerca de cómo trabaja, ya que este funciona como un back end (teniendo en cuenta que un software se divide en dos partes técnicas, frontend (lo que ve el usuario) y el back siendo el cómo es que funciona dicho software) y este se implementa en multiplataforma, ya que es compatible con diferentes versiones como lo es Keras, Open CL, Linux, macOS y Windows. Siendo su punto más importante dentro de funcionamiento que puede usar la tarjeta gráfica para tener un aprendizaje profundo.

Con PlaidML se agrupan los temas de compiladores donde se incluyen lenguajes formales y de compiladores. Por lo que se consigue establecer esta comunicación entre seres humanos y computadores estableciendo el PlaidML con un tipo de lenguaje conocido, el cual se puede definir como un conjunto de notaciones usadas para describir procesos computacionales a las personas y las máquinas (programas).

5. METODOLOGÍA.

(Kanban)

La metodología kanban resuelven los problemas surgidos, posteriormente, a la masificación del uso del computador personal, dado que las expectativas y necesidades por parte de los usuarios se hicieron más urgentes y frecuentes. Fue así como a comienzo de los 90 surgieron propuestas metodológicas para lograr resultados más rápidos en el desarrollo de software sin disminuir su calidad.

Además de que es un enfoque en tiempo real a la gestión de proyectos que ayuda a gestionar el proceso de programación que se va desarrollando, de forma progresiva a medida que va evolucionando. Con la metodología ágil creas, incorporas retroalimentación, pruebas y gestionas tus proyectos, todo ello de manera simultánea.

Metodología Kanban. (s. f.). Kanban Tool. Recuperado 22 de junio de 2021, de <https://kanbantool.com/es/metodologia-kanban>

5.1 DESCRIPCIÓN DE LA METODOLOGÍA.

Usamos la metodología ágil por lo compleja que es y por la formulación que se requiere para el desarrollo del proyecto, ya que se necesita de un equipo eficiente para llevarlo a cabo, además de rapidez y flexibilidad por parte de todo.

También, porque esta metodología está enfocada a mejorar el resultado a futuro del proyecto, por ello la implementaremos en nuestro proyecto.

5.2 TABLA COMPARATIVA METODOLOGÍAS

| AGILES | TRADICIONALES |
|--|--|
| EN LAS AGILES UTILIZAMOS POCOS ARTEFACTOS (MAQUINARIA, HERRAMIENTAS) | EN LAS TRADICIONALES SE USAN MUCHOS MAS ARTEFACTOS |
| EL CLIENTE PUEDE INTERACTUAR CUANDO DESEE CON EL EQUIPO DE DESARROLLO | EL CLIENTE SOLO PUEDE INTERACTUAR CON EL EQUIPO DE DESARROLLO CON UNA PREVIA CITACION Y SI EL EQUIPO DE DESARROLLO ESTA DE ACUERDO |
| EL CLIENTE ES PRIMERO QUE EL FACTOR ECONOMICO | EN EL TRADICIONAL SI NO HAY EL FACTOR ECONOMICO. NO SE LABORA |
| HAY MENOS ENFASIS EN LA ARQUITECTURA | EL LA M.T LA ARQUITECTURA ES ESCENCIAL |
| LOS GRUPOS DE TRABAJO SON MAS PEQUEÑOS, REGULARMENTE SON MENOS DE 10 INTEGRANTES | LOS GRUPOS SIEMPRE SON GRANDES |
| HAY ASIGNADOS POCOS ROLES | MUCHOS ROLES |
| HAY MAS FLEXIBILIDAD ANTE LOS CAMBIOS REPENTINOS Y SE EJECUTAN CON RAPIDEZ | LOS CAMBIOS SE EJECUTAN DE UNA MANERA LENTA |

6. MARCO TEÓRICO

6.1 COMPILADOR

Los compiladores son programas de computadora que traducen un programa escrito en un idioma en un programa escrito en otro idioma. Al mismo tiempo, un El compilador es un gran sistema de software, con muchos componentes internos y algoritmos e interacciones complejas entre ellos. Así, el estudio de la construcción de compiladores es una introducción a las técnicas de traducción y perfeccionamiento de programas y ejercicio práctico de ingeniería de software.

Un compilador es una herramienta que traduce software escrito en un idioma a otro idioma. Para traducir texto de un idioma a otro, la herramienta debe comprender tanto la forma o la sintaxis como el contenido o el significado del idioma de entrada. Necesita comprender las reglas que gobiernan la sintaxis y el significado en el lenguaje de salida. Finalmente, necesita un esquema para mapear contenido. Del idioma de origen al idioma de destino.

La estructura de un compilador típico se deriva de estas simples observaciones. El compilador tiene una interfaz para manejar el lenguaje fuente. Tiene espalda terminar para tratar con el idioma de destino. Conectando la parte delantera y trasera Al final, tiene una estructura formal para representar el programa en una forma intermedia cuyo significado es en gran medida independiente de cualquiera de los dos idiomas. A mejorar la traducción, un compilador a menudo incluye un optimizador que analiza y reescribe esa forma intermedia.

6.2 LLVM

El compilador LLVM utiliza un solo IR de bajo nivel; de hecho, el nombre LLVM significa para "máquina virtual de bajo nivel". El IR de LLVM es un código lineal de tres direcciones. La IR está completamente mecanografiado y tiene soporte explícito para direcciones de matriz y estructura. Eso proporciona soporte para operaciones y datos vectoriales o SIMD. Los valores escalares son mantenido en forma SSA en todo el compilador. El entorno LLVM utiliza interfaces de GCC, por lo que LLVM IR se produce mediante una pasada que realiza la traducción de GIMPLE a LLVM.

El Proyecto LLVM es una colección de tecnologías de cadena de herramientas y compiladores modulares y reutilizables. A pesar de su nombre, LLVM tiene poco que ver con las máquinas virtuales tradicionales. El nombre "LLVM" en sí mismo no es un acrónimo; es el nombre completo del proyecto.

LLVM comenzó como un proyecto de investigación en la Universidad de Illinois, con el objetivo de proporcionar una estrategia de compilación moderna basada en SSA capaz de soportar la compilación estática y dinámica de lenguajes de programación arbitrarios. Desde entonces, LLVM se ha convertido en un proyecto general que consta de una serie de subproyectos, muchos de los cuales están siendo utilizados en la producción por una amplia variedad de proyectos comerciales y de código abierto, además de ser ampliamente utilizados en la investigación académica.

6.3 MLIR

Este trabajo presenta MLIR, un enfoque novedoso para la construcción de materiales reutilizables y extensibles. Infraestructura del compilador. MLIR tiene como objetivo abordar la fragmentación del software, mejorar compilación para hardware heterogéneo, reduce significativamente el costo de construcción compiladores específicos de dominio y ayuda a conectar los compiladores existentes. MLIR facilita el diseño e implementación de generadores de código, traductores y optimizadores en diferentes niveles de abstracción y también en dominios de aplicaciones, objetivos de hardware y entornos de ejecución.

La contribución de este trabajo incluye discusión de MLIR como un artefacto de investigación, construido para extensión y evolución, e identificando los desafíos y oportunidades que plantea esta novela punto de diseño en diseño, semántica, especificación de optimización, sistema e ingeniería. Evaluación de MLIR como una infraestructura generalizada que reduce el costo de construcción de compiladores, que describe diversos casos de uso para mostrar la investigación y oportunidades educativas para futuros lenguajes de programación, compiladores, ejecución entornos y arquitectura informática. El documento también presenta la justificación de MLIR, sus principios de diseño originales, estructuras y semántica.

El diseño del compilador es un campo maduro con una amplia gama de algoritmos bien conocidos, con aplicaciones a la generación de código, análisis estático, transformación de programas y más. El campo también ha visto el desarrollo de una serie de plataformas tecnológicas maduras que han permitido una reutilización masiva en la comunidad de compiladores, incluidos sistemas como la infraestructura de compiladores LLVM [25], Java

Virtual Machine (JVM) [26] y muchos otros. Una característica común de estos sistemas populares es su enfoque de "talla única": un único nivel de abstracción para interactuar con el sistema: el LLVM Representación intermedia (IR) es aproximadamente "C con vectores", y JVM proporciona un "orientado a objetos sistema de tipos con un recolector de basura" abstracción. Este enfoque de "talla única" es increíblemente valioso, y en la práctica, el mapeo a estos dominios desde lenguajes fuente ubicuos (C / C ++ y Java respectivamente) es sencillo.

Al mismo tiempo, muchos problemas se modelan mejor en una abstracción de nivel superior o inferior, p. Ej. El análisis a nivel de fuente del código C ++ es muy difícil en LLVM IR. Observamos que muchos idiomas (incluidos, por ejemplo, Swift, Rust, Julia, Fortran) desarrollan su propio IR para resolver dominios específicos problemas, como optimizaciones específicas de lenguaje / biblioteca, verificación de tipo sensible al flujo (por ejemplo, para * Con SiFive en el momento de la publicación.

Si bien el desarrollo de RI específicos de dominio es un arte bien estudiado, su costo de ingeniería e implementación sigue siendo alto. La calidad de la infraestructura no siempre es una primera prioridad (o fácil de justificar) para los implementadores de estos sistemas. En consecuencia, esto puede conducir a un compilador de menor calidad. Sistemas, incluidos problemas visibles para el usuario, como tiempos de compilación lentos, implementaciones con errores, calidad de diagnóstico, mala experiencia de depuración para código optimizado, etc.

El proyecto MLIR tiene como objetivo abordar directamente el diseño e implementación de estos lenguajes de programación. Desafíos, al hacer que sea muy barato definir e introducir nuevos niveles de abstracción, y proporcionar Infraestructura "en la caja" para resolver problemas comunes de ingeniería de compiladores. MLIR hace esto por estandarizar las estructuras de datos IR basadas en la asignación única estática (SSA), proporcionar una sistema para definir dialectos IR, y proporcionar una amplia gama de infraestructura común (incluyendo documentación, lógica de

impresión y análisis, seguimiento de ubicación, soporte de compilación multiproceso, pase gestión, etc.).

Este documento explora varios puntos de diseño del sistema MLIR, relata nuestra experiencia aplicándolo a unos varios problemas diferentes, y analiza las implicaciones que este trabajo puede tener para el diseño del lenguaje y educación.

6.4 PLAIDML

PlaidML puede acelerar las cargas de trabajo de entrenamiento con código Tile personalizado o generado automáticamente. Funciona especialmente bien en GPU y no requiere el uso de CUDA / cuDNN en hardware Nvidia, al tiempo que logra un rendimiento comparable.

También como componente dentro de la pila del compilador nGraph, PlaidML amplía aún más las capacidades del hardware especializado de aprendizaje profundo (especialmente las GPU) y hace que lo que de otro modo estarían limitadas por la computación. Limitaciones del dispositivo. Sea más fácil y rápido acceder o hacer uso de optimizaciones de nivel de subgráfica

6.5 INTELIGENCIA ARTIFICIAL

La inteligencia artificial (IA) es, en informática, la inteligencia expresada por máquinas, sus procesadores y sus *softwares*, que serían los análogos al cuerpo, el cerebro y la mente, respectivamente, a diferencia de la inteligencia natural demostrada por humanos y ciertos animales con cerebros complejos. En ciencias de la computación, una máquina inteligente ideal es un agente flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea. Coloquialmente, el término inteligencia artificial se aplica cuando una máquina imita las funciones «cognitivas» que los humanos asocian con otras mentes humanas, como por ejemplo: «percibir», «razonar», «aprender» y «resolver problemas». Andreas Kaplan y Michael Haenlein definen la inteligencia artificial como «la capacidad de un sistema para interpretar correctamente datos externos, para aprender de dichos datos y emplear esos conocimientos para lograr tareas y metas concretas a través de la adaptación flexible». A medida que las máquinas se vuelven cada vez más capaces, tecnología que alguna vez se pensó que requería de inteligencia se elimina de la definición. Por ejemplo, el reconocimiento óptico de caracteres ya no se percibe como un ejemplo de la «inteligencia artificial» habiéndose convertido en una tecnología común. Avances tecnológicos todavía clasificados como inteligencia artificial son los sistemas de conducción autónomos o capaces de jugar al ajedrez o al Go.





Ilustración 1: Cerebro bajo el control de la inteligencia artificial

La inteligencia artificial es una nueva forma de resolver problemas dentro de los cuales se incluyen los sistemas expertos, el manejo y control de robots y los procesadores, que intenta integrar el conocimiento en tales sistemas, en otras palabras, un sistema inteligente capaz de escribir su propio programa. Un sistema experto definido como una estructura de programación capaz de almacenar y utilizar un conocimiento sobre un área determinada que se traduce en su capacidad de aprendizaje.

Según Takeyas (2007) la IA es una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos con base en dos de sus características primordiales: el razonamiento y la conducta.⁷

En 1956, John McCarthy acuñó la expresión inteligencia artificial, y la definió como «la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes».⁸

También existen distintos tipos de percepciones y acciones, que pueden ser obtenidas y producidas, respectivamente, por sensores físicos y sensores mecánicos en máquinas, pulsos eléctricos u ópticos en computadoras, tanto como por entradas y salidas de bits de un software y su entorno software.

Varios ejemplos se encuentran en el área de control de sistemas, planificación automática, la capacidad de responder a diagnósticos y a consultas de los consumidores, reconocimiento de escritura, reconocimiento del habla y reconocimiento de patrones. Los sistemas de IA actualmente son parte de la rutina en campos como economía, medicina, ingeniería, el transporte, las comunicaciones y la milicia, y se ha usado en gran variedad de aplicaciones de software, juegos de estrategia, como ajedrez de computador, y otros videojuegos.

6.6 MACHINE LEARNING

Machine learning es una forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita. Sin embargo, machine learning no es un proceso sencillo. Conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en datos. Un modelo de machine learning es la salida de información que se genera cuando entrena su algoritmo de machine learning con datos. Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida. Por ejemplo, un algoritmo predictivo creará un modelo predictivo. A continuación, cuando proporcione el modelo predictivo con datos, recibirá un pronóstico basado en los datos que entrenaron al modelo.



6.6.1 Aprendizaje iterativo

Machine learning permite modelos a entrenar con conjuntos de datos antes de ser implementados. Algunos modelos de machine learning están online y son continuos. Este proceso iterativo de modelos online conduce a una mejora en los tipos de asociaciones hechas entre los elementos de datos. Debido a su complejidad y tamaño, estos patrones y asociaciones podrían haber sido fácilmente pasados por alto por la observación humana. Después de que un modelo ha sido entrenado, se puede utilizar en tiempo real para aprender de los datos. Las mejoras en la precisión son el resultado del proceso de entrenamiento y la automatización que forman parte del machine learning.

6.6.2 Enfoques hacia el machine learning

Las técnicas de machine learning son necesarias para mejorar la precisión de los modelos predictivos. Dependiendo de la naturaleza del problema empresarial que se está atendiendo, existen diferentes enfoques basados en el tipo y volumen de los datos. En esta sección, discutimos las categorías del machine learning.

6.6.3 Aprendizaje supervisado

El aprendizaje supervisado comienza típicamente con un conjunto establecido de datos y una cierta comprensión de cómo se clasifican estos datos. El aprendizaje supervisado tiene la intención de encontrar patrones en datos que se pueden aplicar a un proceso de analítica. Estos datos tienen características etiquetadas que definen el significado de los datos. Por ejemplo, se puede crear una aplicación de machine learning con base en imágenes y descripciones escritas que distinga entre millones de animales.

6.6.4 Aprendizaje no supervisado

El aprendizaje no supervisado se utiliza cuando el problema requiere una cantidad masiva de datos sin etiquetar. Por ejemplo, las aplicaciones de redes sociales, tales como Twitter, Instagram y Snapchat, tienen grandes cantidades de datos sin etiquetar. La comprensión del significado detrás de estos datos requiere algoritmos que clasifican los datos con base en los patrones o clústeres que encuentra. El aprendizaje no supervisado lleva a cabo un proceso iterativo, analizando los datos sin intervención humana. Se utiliza con la tecnología de detección de spam en e-mails. Existen demasiadas variables en los e-mails legítimos y de spam para que un analista etiquete una cantidad masiva de e-mail no solicitado. En su lugar, los clasificadores de machine learning, basados en clustering y asociación, se aplican para identificar e-mail no deseado.

6.6.5 Aprendizaje de refuerzo

El aprendizaje de refuerzo es un modelo de aprendizaje conductual. El algoritmo recibe retroalimentación del análisis de datos, conduciendo al usuario hacia el mejor resultado. El aprendizaje de refuerzo difiere de otros tipos de aprendizaje supervisado, porque el sistema no está entrenado con el conjunto de datos de ejemplo. Más bien, el sistema aprende a través de la prueba y el error. Por lo tanto, una secuencia de decisiones exitosas conduce al fortalecimiento del proceso, porque es el que resuelve el problema de manera más efectiva.

6.7 DEEP LEARNING

El deep learning es un tipo de machine learning que entrena a una computadora para que realice tareas como las hacemos los seres humanos, como el reconocimiento del habla, la identificación de imágenes o hacer predicciones. En lugar de organizar datos para que se ejecuten a través de ecuaciones predefinidas, el deep learning configura parámetros básicos acerca de los datos y entrena a la computadora para que aprenda por cuenta propia reconociendo patrones mediante el uso de muchas capas de procesamiento.

6.7.1 Importancias del deep learning

El deep learning es una de las bases de la inteligencia artificial (AI) y el interés actual en el deep learning se debe en parte al auge que tiene ahora la inteligencia artificial. Las técnicas de deep learning han mejorado la capacidad de clasificar, reconocer, detectar y describir – en una palabra, entender. Por ejemplo, el deep learning se utiliza para clasificar imágenes, reconocer el habla, detectar objetos y describir contenido. Sistemas como Siri y Cortana son potenciados, en parte, por el aprendizaje a fondo.

Varias novedades están integrando avances ahora al aprendizaje a fondo:

- Mejoras algorítmicas han elevado el desempeño de los métodos de aprendizaje a fondo.
- Nuevos métodos de aprendizaje basado en máquina han mejorado la precisión de los modelos.
- Se han desarrollado nuevas clases de redes neurales que encajan bien en aplicaciones como la traducción de texto y la clasificación de imágenes.

- Tenemos muchos más datos disponibles para construir redes neurales con muchas capas profundas, incluyendo datos de streaming de la Internet de las Cosas, datos textuales de medios sociales, notas de médicos y transcripciones de investigaciones.
- Los adelantos computacionales de la computación en la nube distribuida y unidades de procesamiento gráfico han puesto a nuestra disposición una cantidad increíble de poder de cómputo. Este nivel de poder de cómputo es necesario para entrenar a algoritmos profundos.

Al mismo tiempo, las interfaces de humano a máquina han evolucionado considerablemente también. El mouse y el teclado están siendo reemplazados con gesticulaciones, deslizamientos de los dedos, tacto y lenguaje natural, generando un interés renovado en la inteligencia artificial y el deep learning.

6.7.2 Oportunidades y aplicaciones del deep learning

Se necesita mucho poder de cómputo para resolver problemas de deep learning debido a la naturaleza iterativa de los algoritmos de deep learning, su complejidad conforma aumenta el número de capas y los grandes volúmenes de datos que se necesitan para entrenar a las redes.

La naturaleza dinámica de los métodos de deep learning su capacidad de mejorar y adaptarse continuamente a cambios en el patrón de información implícito presenta una gran oportunidad para introducir un comportamiento más dinámico a la analítica.

Una mayor personalización de la analítica de clientes es una posibilidad. Otra gran oportunidad es mejorar la precisión y el desempeño en aplicaciones donde se han utilizado redes neurales por largo tiempo. A través de mejores algoritmos y más poder de cómputo podemos agregar mayor profundidad.

Aunque el enfoque actual del mercado de las técnicas de deep learning está en aplicaciones de cómputo cognitivo, también hay un gran potencial en aplicaciones analíticas más tradicionales; por ejemplo, el análisis de series de tiempo.

Otra oportunidad es simplemente ser más eficiente y simplificado en operaciones analíticas existentes. Recientemente, SAS experimentó con redes neurales profundas en problemas de transcripción de habla a texto. Comparado con las técnicas estándares, el índice de errores en palabras disminuyó más de 10% cuando se aplicaron redes neurales profundas. También eliminaron cerca de 10 pasos del procesamiento de datos, ingeniería de características y modelado. Los impresionantes incrementos de desempeño y los ahorros de tiempo cuando se comparan con la ingeniería de características se traducen en un cambio de paradigma.

6.7.3 Cómo funciona el deep learning

El deep learning cambia su forma de pensar acerca de la representación de los problemas que resuelve con la analítica. Pasa de decir a la computadora cómo resolver un problema a entrenarla para que resuelva el problema mismo.

Un enfoque tradicional de la analítica consiste en utilizar los datos que se tienen a la mano para diseñar características por ingeniería a fin de obtener nuevas variables, luego seleccionar un modelo analítico y finalmente calcular los parámetros (o los valores desconocidos) de ese modelo. Estas técnicas pueden producir sistemas predictivos que no generalizan bien porque la integridad y la corrección dependen de la calidad del modelo y sus características. Por ejemplo, si desarrolla un modelo de fraude con ingeniería de características, comienza con un conjunto de variables y lo más probable es que obtenga un modelo a partir de esas variables utilizando transformaciones de datos. Puede terminar con 30,000 variables de las cuales dependa su modelo, luego tiene que darle forma, averiguar qué variables son significativas, cuáles no lo son, etc. La adición de más datos requiere que haga todo el proceso de nueva cuenta.



El nuevo enfoque con el aprendizaje a fondo consiste en reemplazar la formulación y especificación del modelo con caracterizaciones (o capas) jerárquicas que aprendan a reconocer características latentes de los datos de las regularidades en las capas.

El cambio de paradigma con el deep learning es un cambio de la ingeniería de características a la representación de características.

La promesa del deep learning es que generar sistemas predictivos que generalicen bien, se adapten bien, mejoren continuamente conforme lleguen nuevos datos y sean más dinámicos que los sistemas predictivos basados en reglas de negocios estrictas. Ya no necesita ajustar un modelo. En su lugar, se entrena la tarea.

7. DESARROLLO E IMPLEMENTACION (REPORTE)

Al momento de conocer un poco más acerca de PlaidML y de su documentación como primer punto es que esta documentación no está actualizada a las últimas versiones de Windows puesto que al momento de instalar Python desde el scrip está separado en partes, y la importante que es el pip se debe descargar aparte al igual que las instalaciones del PowerShell (Siendo una interfaz de consola con la posibilidad de escritura y unión de comandos por medio de escritura, está diseñada para que ocupe por administradores). Se explicara más en los siguientes puntos:

1. Ejecutamos CMD desde administrador puesto que si no lo hacemos de esta manera no tendremos los permisos requeridos para poder seguir trabajando:



Ilustración 2: Inicio de CMD desde el modo de administrado.

2. Como siguiente paso iniciaremos el powershell, poniendo el siguiente comando "PowerShell":

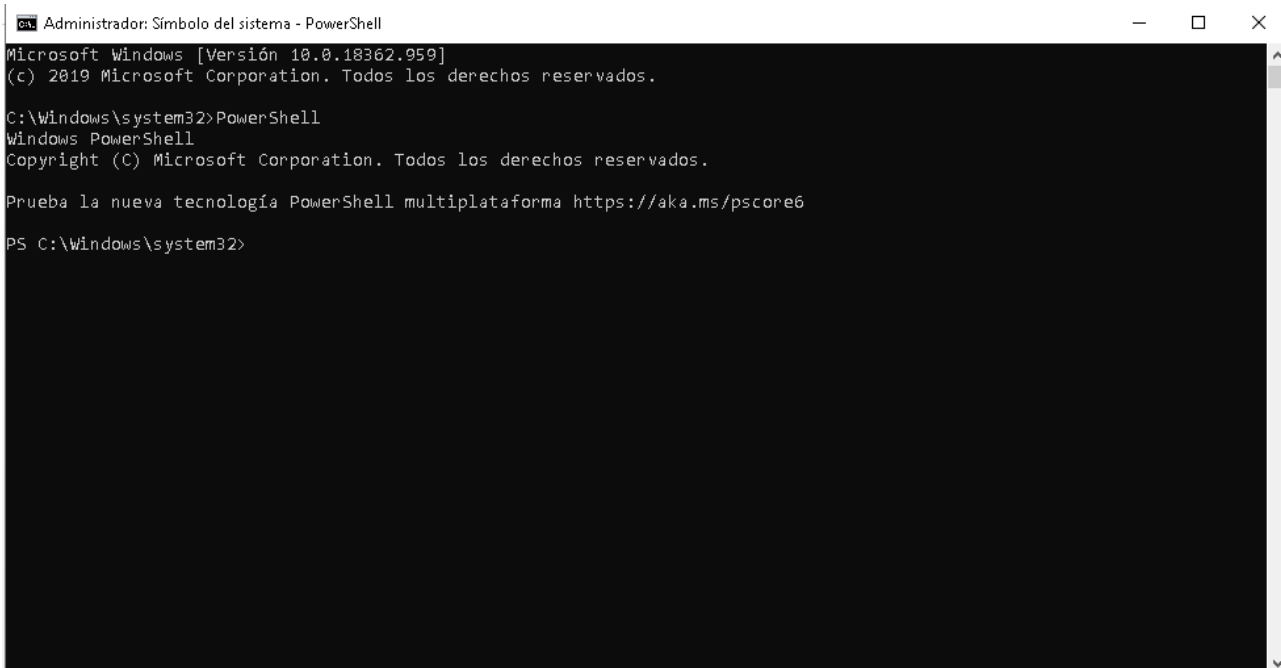


Ilustración 3: PowerShell iniciado.

3. Después ocuparemos la herramienta de chocolatey así que descargaremos esa herramienta desde el siguiente código "**Set-ExecutionPolicy Bypass -Scope Process -Force; iwr https://community.chocolatey.org/install.ps1 -UseBasicParsing | iex**":

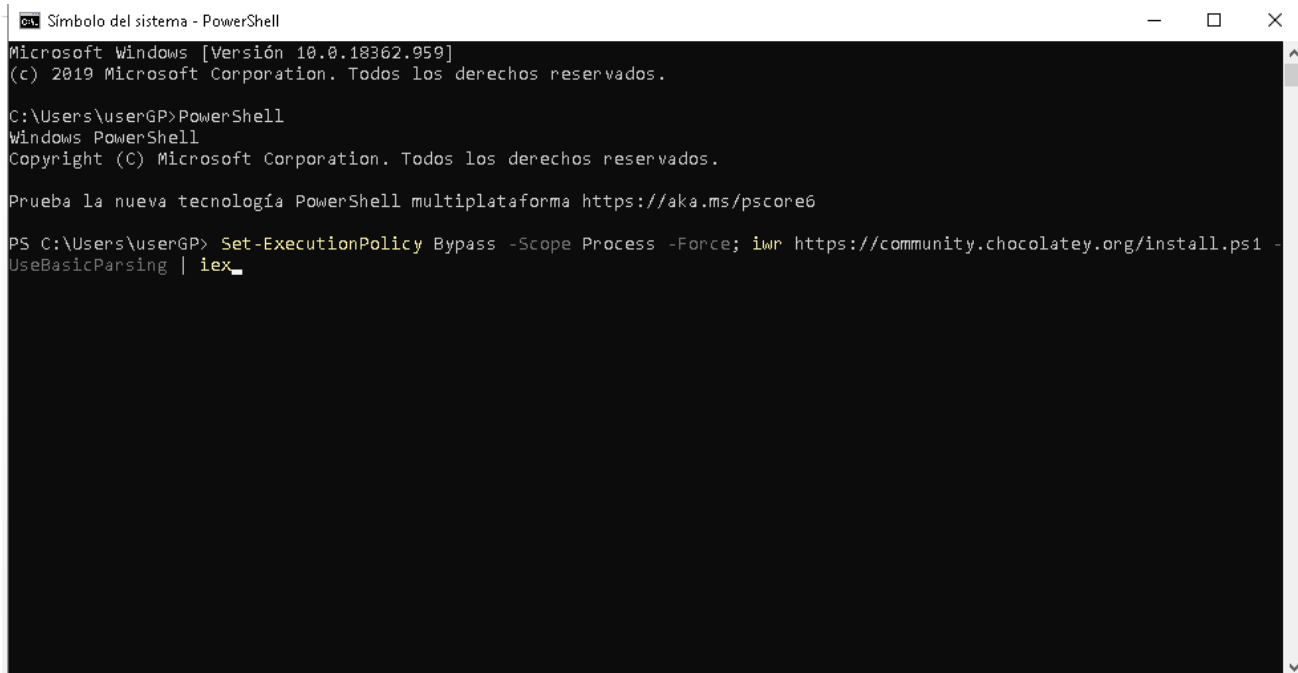
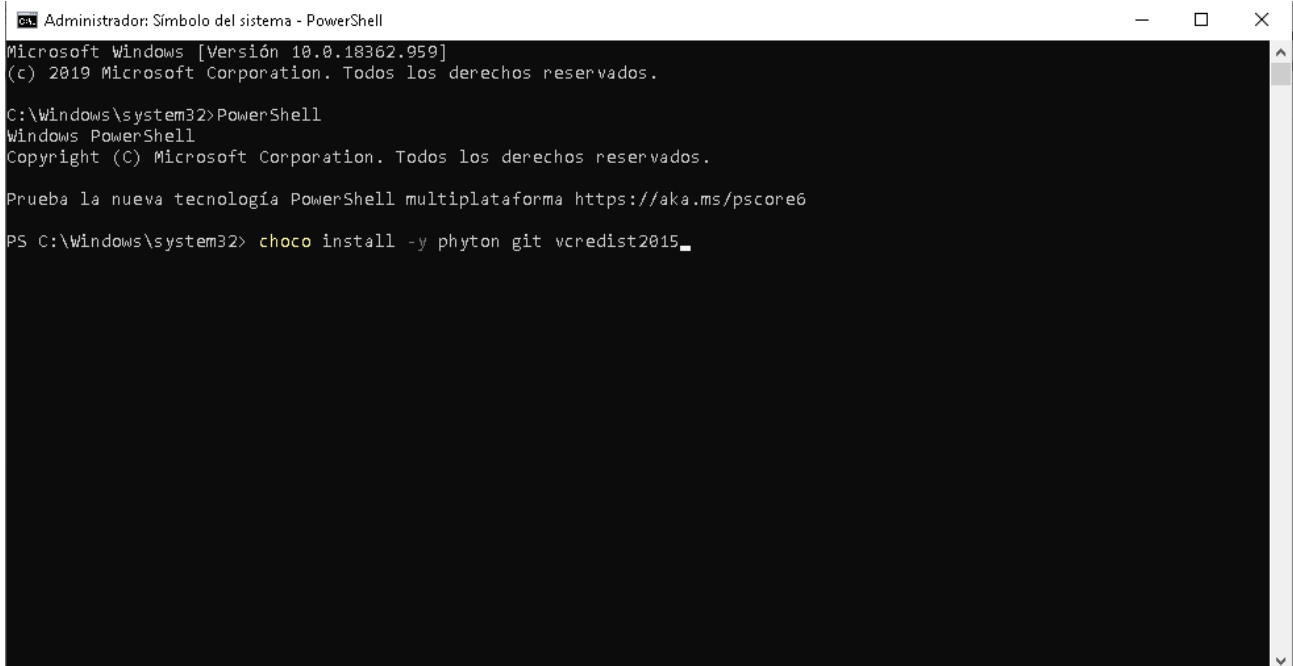


Ilustración 4: Ya puesta la dirección de archivo que necesitamos

POR MOTIVOS DE LA INSTALACION, SOLO SE PONDRAS LAS NORMATIVAS QUE SE OCUPARON, PUESTO QUE YA AL MOMENTO DE EJECUTARLAS MANDA ERRORES DE QUE SE ESTA DUPLICANDO.

4. Consecutivamente se instala Python 3.9 y Virtual studio 2015 con la siguiente normativa ***"choco install -y phyton git vcredist2015"***:



```
Administrador: Símbolo del sistema - PowerShell
Microsoft Windows [Versión 10.0.18362.959]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

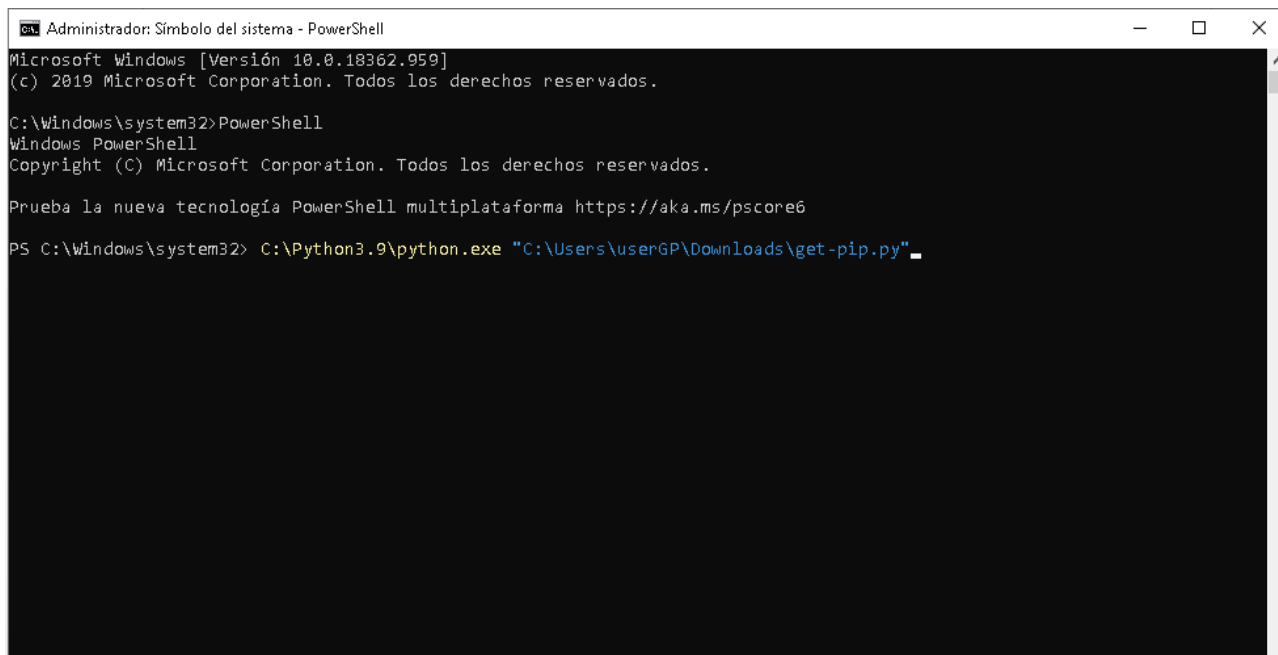
C:\Windows\system32>PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Windows\system32> choco install -y phyton git vcredist2015_
```

Ilustración 5: Instalación de Python y virtual studio.

5. Ahora se procede a instalar “pip” puesto que ya no se descarga esta parte, se procede a buscar se descarga y lo ejecutamos en cmd con la siguiente normativa **“C:\Python3.9\python.exe “C:\Users\userGP\Downloads\get-pip.py”**:



```
Administrador: Símbolo del sistema - PowerShell
Microsoft Windows [Versión 10.0.18362.959]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

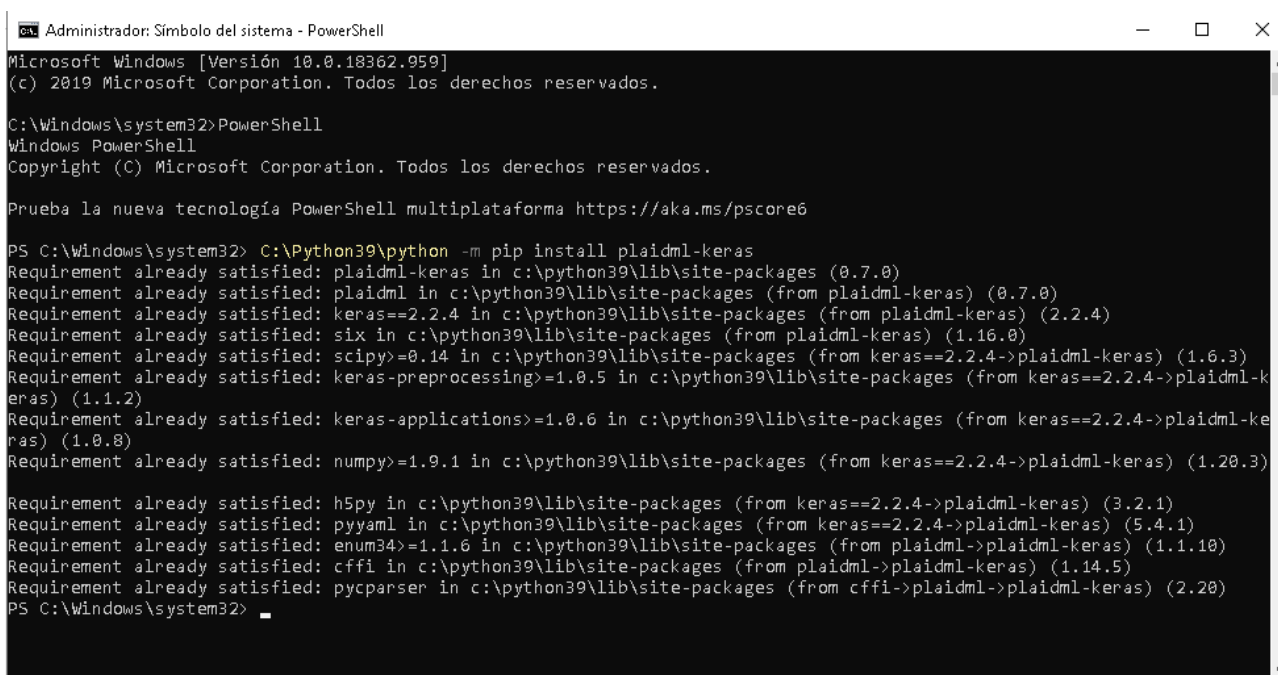
C:\Windows\system32>PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Windows\system32> C:\Python3.9\python.exe "C:\Users\userGP\Downloads\get-pip.py"
```

Ilustración 6: Instalamos el "pip" para proseguir a la siguiente parte

6. Después que tenemos el “pip” instalado ahora se prosigue a la instalación de plaidML-keras, se ocupara la siguiente normativa **“C:\Python39\python -m pip install plaidml-keras”**



```
Administrador: Símbolo del sistema - PowerShell
Microsoft Windows [Versión 10.0.18362.959]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Windows\system32> C:\Python39\python -m pip install plaidml-keras
Requirement already satisfied: plaidml-keras in c:\python39\lib\site-packages (0.7.0)
Requirement already satisfied: plaidml in c:\python39\lib\site-packages (from plaidml-keras) (0.7.0)
Requirement already satisfied: keras==2.2.4 in c:\python39\lib\site-packages (from plaidml-keras) (2.2.4)
Requirement already satisfied: six in c:\python39\lib\site-packages (from plaidml-keras) (1.16.0)
Requirement already satisfied: scipy>=0.14 in c:\python39\lib\site-packages (from keras==2.2.4->plaidml-keras) (1.6.3)
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:\python39\lib\site-packages (from keras==2.2.4->plaidml-keras) (1.1.2)
Requirement already satisfied: keras-applications>=1.0.6 in c:\python39\lib\site-packages (from keras==2.2.4->plaidml-keras) (1.0.8)
Requirement already satisfied: numpy>=1.9.1 in c:\python39\lib\site-packages (from keras==2.2.4->plaidml-keras) (1.20.3)
Requirement already satisfied: h5py in c:\python39\lib\site-packages (from keras==2.2.4->plaidml-keras) (3.2.1)
Requirement already satisfied: pyyaml in c:\python39\lib\site-packages (from keras==2.2.4->plaidml-keras) (5.4.1)
Requirement already satisfied: enum34>=1.1.6 in c:\python39\lib\site-packages (from plaidml->plaidml-keras) (1.1.10)
Requirement already satisfied: cffi in c:\python39\lib\site-packages (from plaidml->plaidml-keras) (1.14.5)
Requirement already satisfied: pycparser in c:\python39\lib\site-packages (from cffi->plaidml->plaidml-keras) (2.20)
PS C:\Windows\system32>
```

Ilustración 7: Ya instalado plaidML-keras con sus requerimientos ya aprobados.

7. Ya que plaidML está instalado ahora procedemos a su configuración, para esto nos dirigimos hacia la carpeta de disco “C:” → “Archivos de programa” → “Python 39” → “Scripts” → “**plaidml-Setup**”. Posteriormente ejecutamos la aplicación de “**plaidml-setup**” para configurar los parámetros, las tres preguntas se responden de la siguiente manera: “y”, “1” y por ultimo “y”.

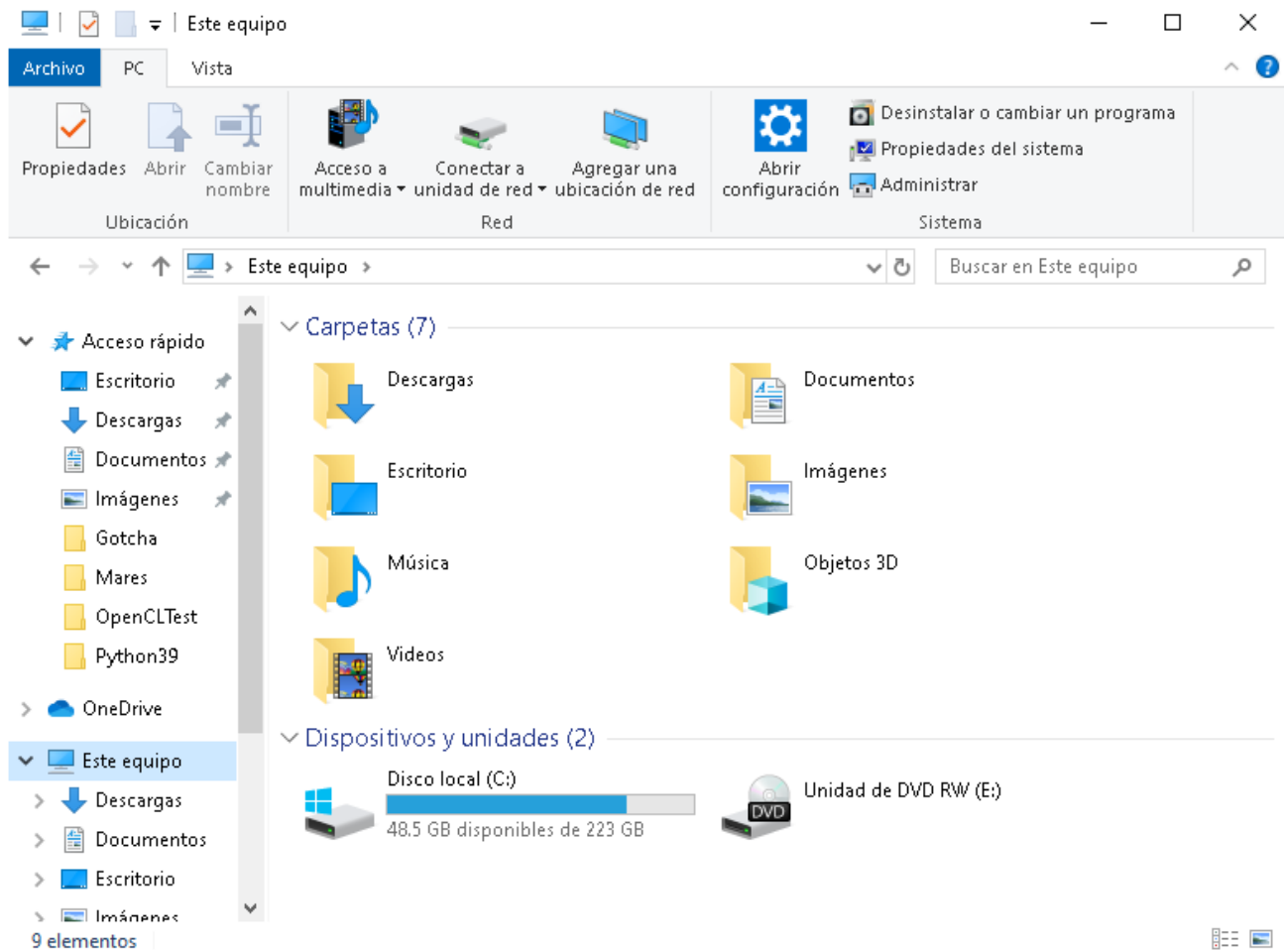


Ilustración 8: Localización del disco “(C:)”

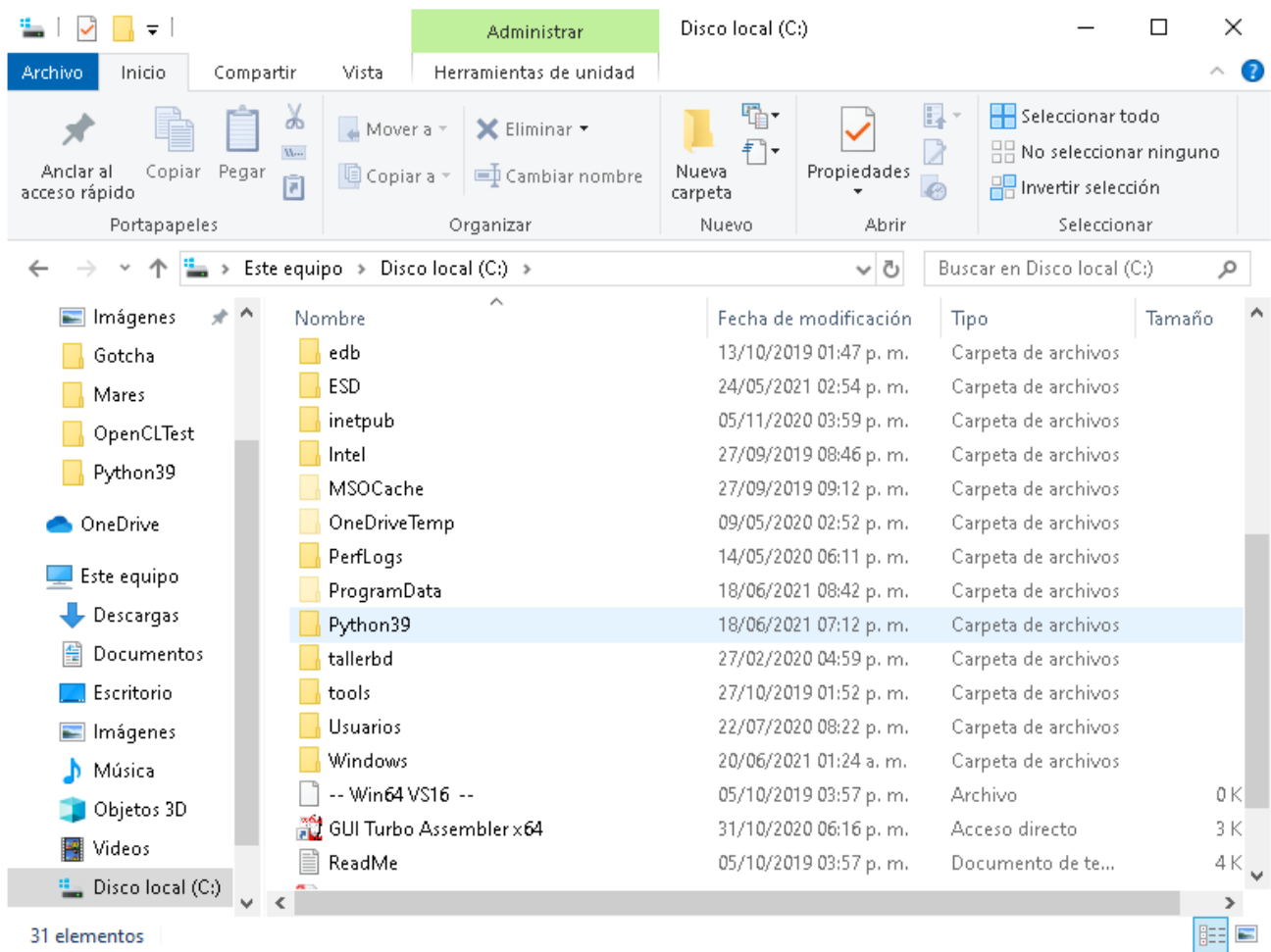


Ilustración 9: Localización de la carpeta "Python 39" que es la versión 3.9

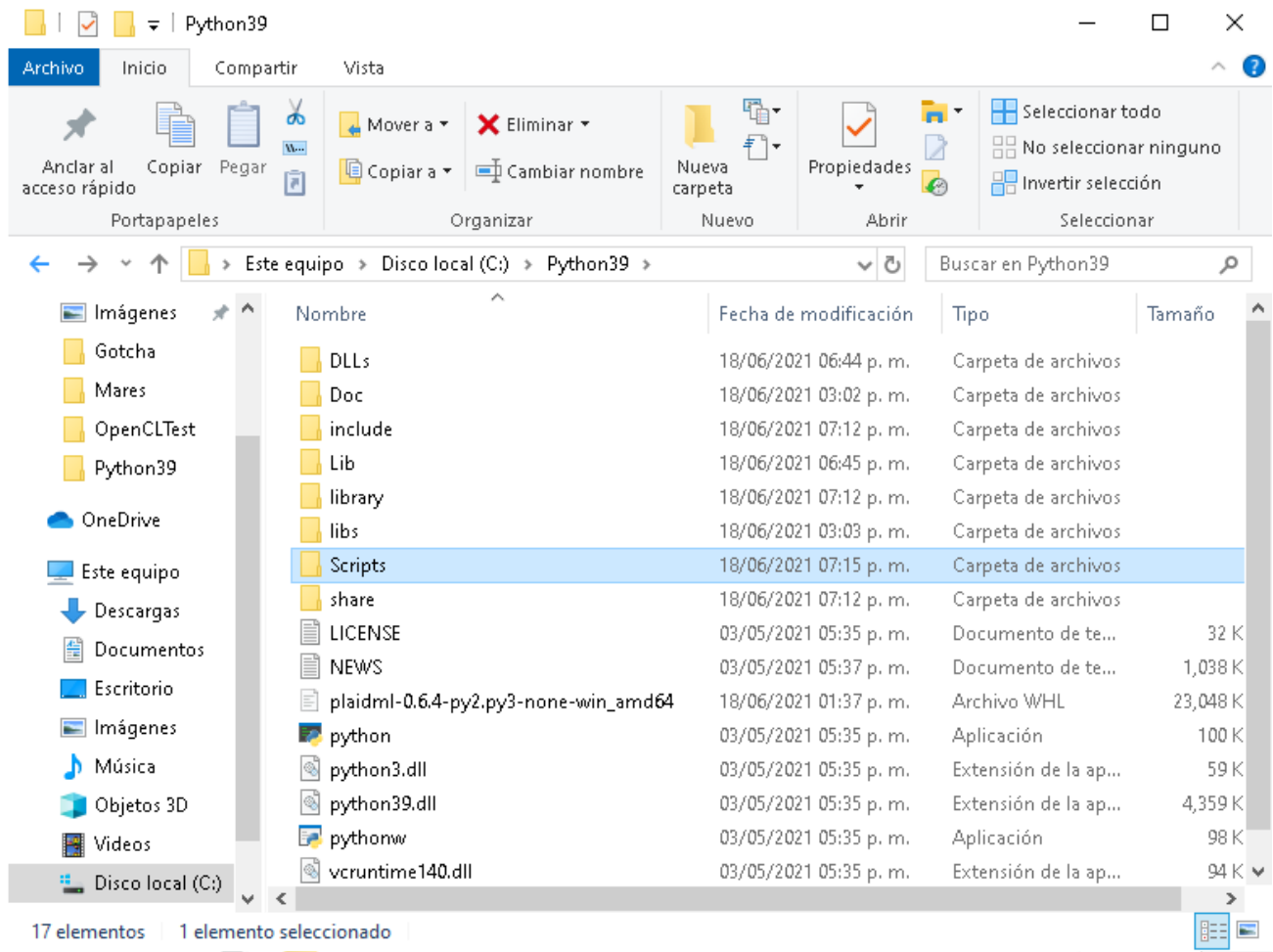


Ilustración 10: Carpeta de “**Scripts**” localizada.

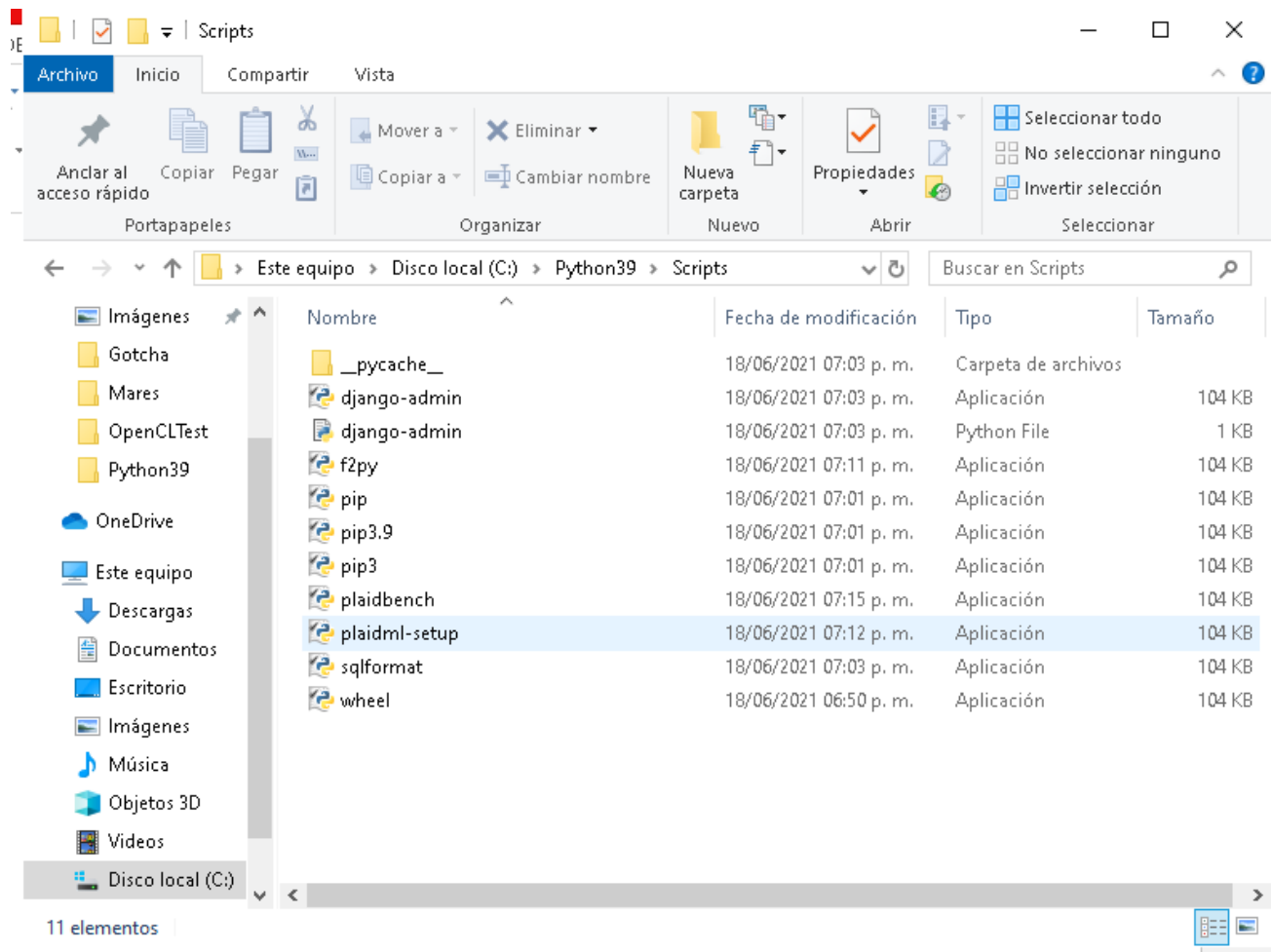


Ilustración 11: Localizamos la aplicación de "**plaidml-setup**"

C:\Python39\Scripts\plaidml-setup.exe

```
PlaidML Setup (0.7.0)

Thanks for using PlaidML!

The feedback we have received from our users indicates an ever-increasing need
for performance, programmability, and portability. During the past few months,
we have been restructuring PlaidML to address those needs. To make all the
changes we need to make while supporting our current user base, all development
of PlaidML has moved to a branch - plaidml-v1. We will continue to maintain and
support the master branch of PlaidML and the stable 0.7.0 release.

Read more here: https://github.com/plaidml/plaidml

Some Notes:
* Bugs and other issues: https://github.com/plaidml/plaidml/issues
* Questions: https://stackoverflow.com/questions/tagged/plaidml
* Say hello: https://groups.google.com/forum/#!forum/plaidml-dev
* PlaidML is licensed under the Apache License 2.0

Default Config Devices:
  llvm_cpu.0 : CPU (via LLVM)

Experimental Config Devices:
  llvm_cpu.0 : CPU (via LLVM)

Using experimental devices can cause poor performance, crashes, and other nastiness.

Enable experimental device support? (y,n)[n]:
```

Ilustración 12: Ejecutamos "plaidml-setup" para que lo configuremos. OJO: POR MOTIVOS ANTERIORMENTE COMENTADOS SE ESTA PONIENDO SOLO DE EJEMPLO LA IMAGEN YA QUE ESTA CONFIGURADO Y CUANDO CONTESTO LAS 3 PREGUNTAS SE CIERRA AUTOMATICAMENTE.

Listo, ahora que todo está instalado se procede a la experimentación del código en “Visual studio code” el código a ejemplificar será el siguiente:

```
import numpy as np
import os
import time

os.environ["KERAS_BACKEND"] = "plaidml.keras.backend"

import keras
import keras.applications as kapp
from keras.datasets import cifar10

(x_train, y_train_cats), (x_test, y_test_cats) = cifar10.load_data() #
batch_size = 8
x_train = x_train[:batch_size]
x_train = np.repeat(np.repeat(x_train, 7, axis=1), 7, axis=2)
model = kapp.VGG19()
model.compile(optimizer='sgd', loss='categorical_crossentropy',
              metrics=['accuracy'])
```



```
print("Ejecutando el lote inicial ")
y = model.predict(x=x_train, batch_size=batch_size)

print("Interferencia de tiempo...")
start = time.time()
for i in range(10):
    y = model.predict(x=x_train, batch_size=batch_size)
print("Se ejecuto en {} segundos".format(time.time() - start))
```

8. RESULTADOS

Este ejemplo se realizó en 2 computadoras con diferentes especificaciones de hardware. Antes de iniciar el código haremos una comparación sin el código ejecutando y con el código en ejecución. Implementamos el código dentro de “**Visual Studio Code**” y abrimos el administrador de tareas para obtener los siguientes resultados:

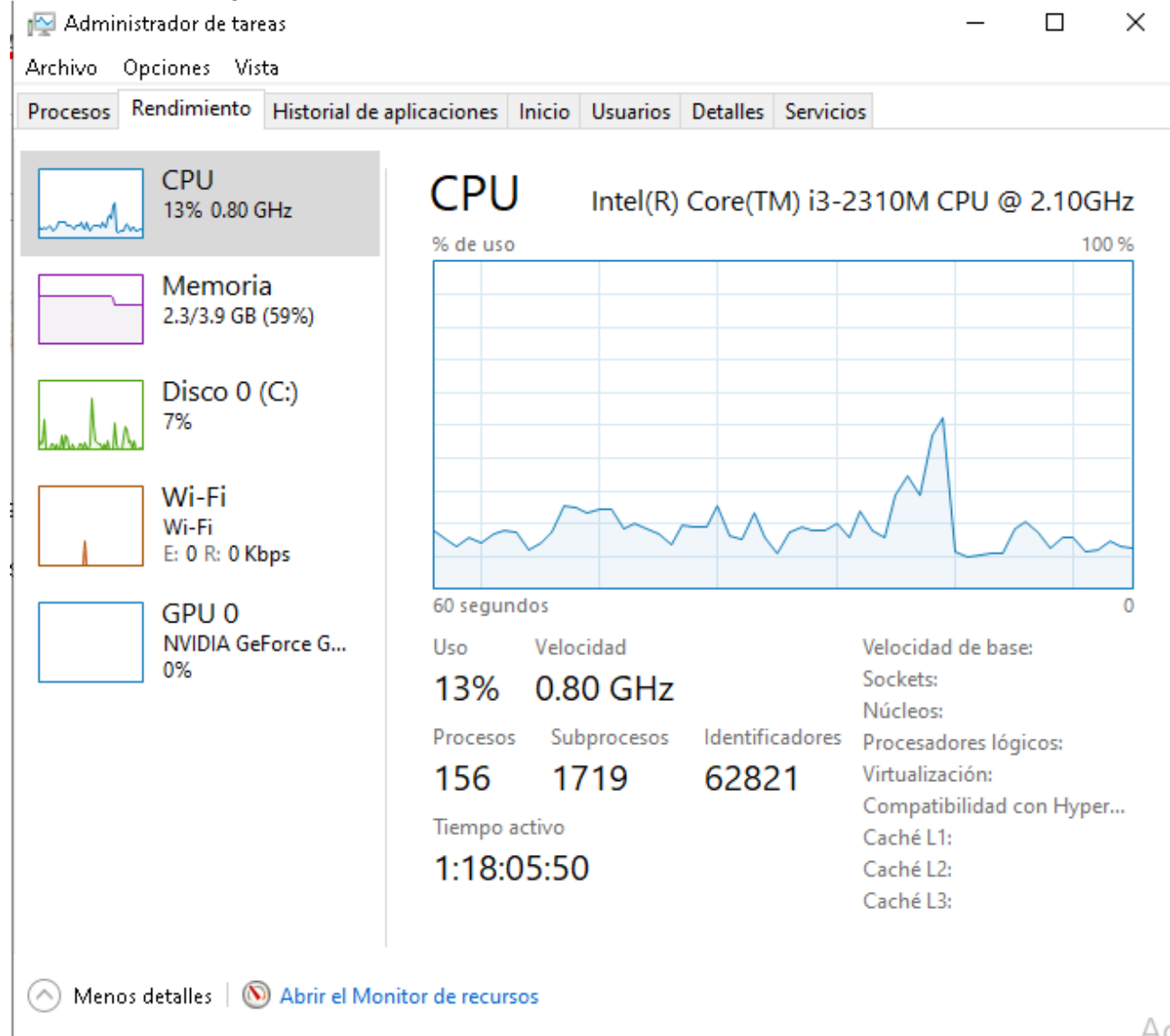
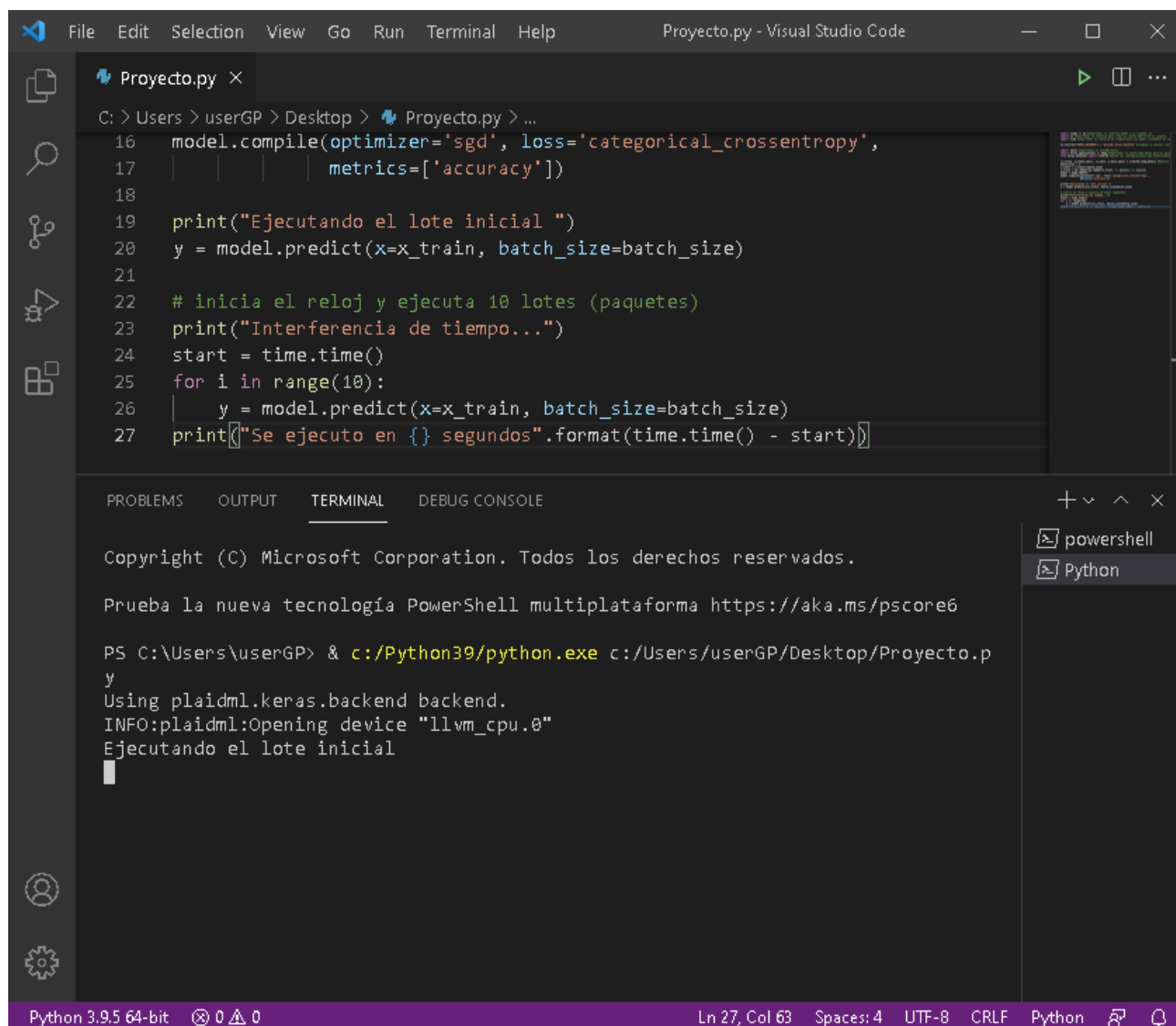


Ilustración 13: Computadora "1" sin el código en ejecución.



```
File Edit Selection View Go Run Terminal Help Proyecto.py - Visual Studio Code

Proyecto.py x
C: > Users > userGP > Desktop > Proyecto.py > ...
16 model.compile(optimizer='sgd', loss='categorical_crossentropy',
17 | | | metrics=['accuracy'])
18
19 print("Ejecutando el lote inicial ")
20 y = model.predict(x=x_train, batch_size=batch_size)
21
22 # inicia el reloj y ejecuta 10 lotes (paquetes)
23 print("Interferencia de tiempo...")
24 start = time.time()
25 for i in range(10):
26 | y = model.predict(x=x_train, batch_size=batch_size)
27 print("Se ejecuto en {} segundos".format(time.time() - start))

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

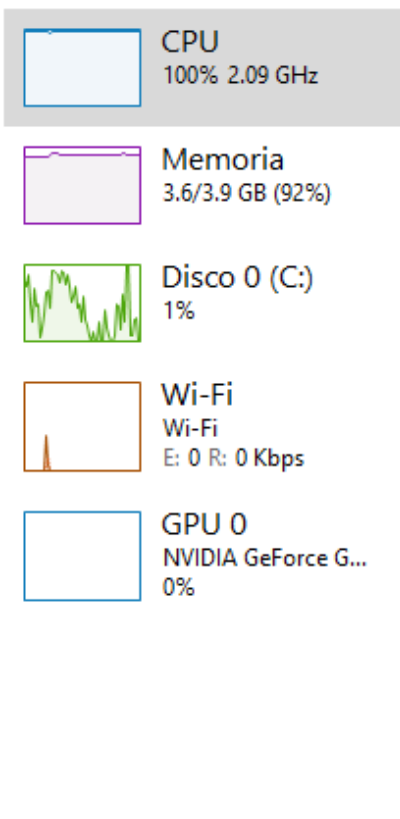
PS C:\Users\userGP> & c:/Python39/python.exe c:/Users/userGP/Desktop/Proyecto.p
y
Using plaidml.keras.backend backend.
INFO:plaidml:Opening device "llvm_cpu.0"
Ejecutando el lote inicial
```

Ilustración 14: El código en función, dentro del entorno de visual studio code.

Administrador de tareas

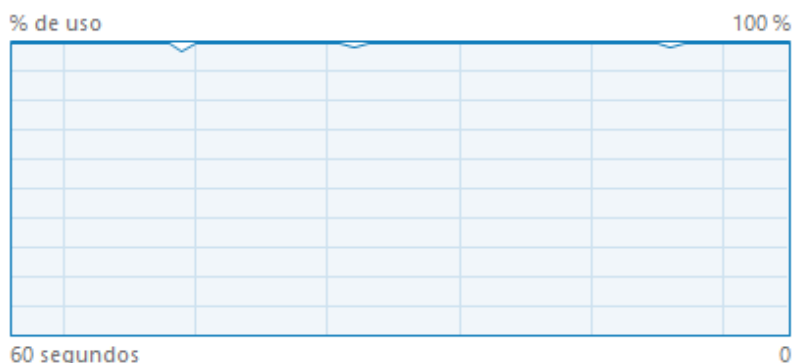
Archivo Opciones Vista

Procesos Rendimiento Historial de aplicaciones Inicio Usuarios Detalles Servicios



CPU

Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz



| Uso | Velocidad | Velocidad de base: |
|-----------------------------|-------------|--------------------|
| 100% | 2.09 GHz | Sockets: |
| Procesos | Subprocesos | Identificadores |
| 174 | 1840 | 69471 |
| Procesadores lógicos: | | |
| Virtualización: | | |
| Compatibilidad con Hyper... | | |
| Caché L1: | | |
| Caché L2: | | |
| Caché L3: | | |

Menos detalles | [Abrir el Monitor de recursos](#)

Ilustración 15: El Procesador de la computadora "1" con la ejecución del plaidML en "Visual studio Code"

Como siguiente ejemplo que la segunda computadora, siendo en su totalidad más potente que la anterior.

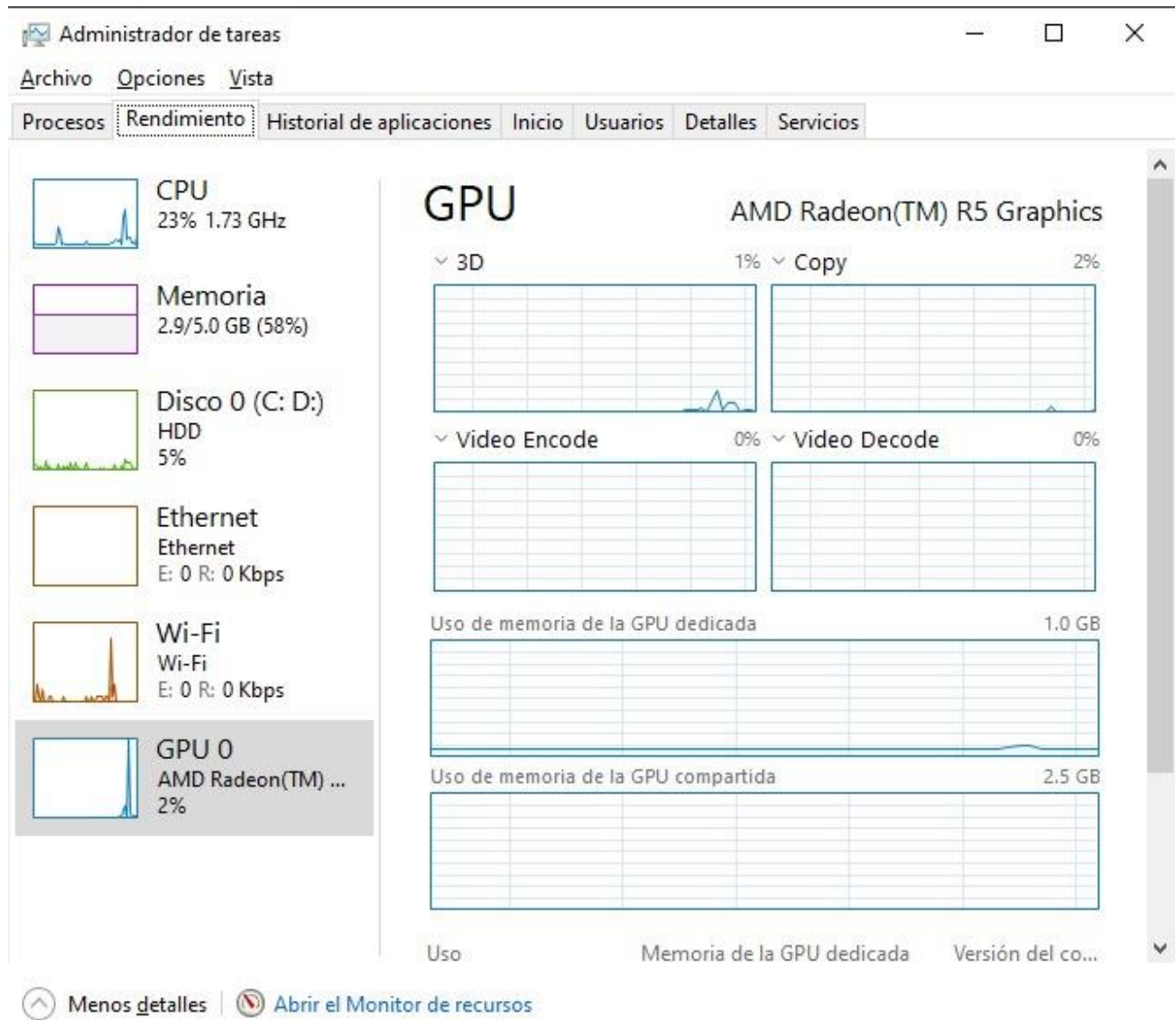


Ilustración 16: Computadora "2" sin ejecutar el código

```
C:\Windows\py.exe
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bi
Type "help", "copyright", "credits" or "license" for more information.
>>> #!/usr/bin/env python
>>>
>>> import numpy as np
>>> import os
>>> import time
>>>
>>> os.environ["KERAS_BACKEND"] = "plaidml.keras.backend"
>>>
>>> import keras
Using plaidml.keras.backend backend.
>>> import keras.applications as kapp
>>> from keras.datasets import cifar10
>>>
>>> (x_train, y_train_cats), (x_test, y_test_cats) = cifar10.load_data()
>>> batch_size = 8
>>> x_train = x_train[:batch_size]
>>> x_train = np.repeat(np.repeat(x_train, 7, axis=1), 7, axis=2)
>>> model = kapp.VGG19()
INFO:plaidml:Opening device "llvm_cpu.0"
>>> model.compile(optimizer='sgd', loss='categorical_crossentropy',
...               metrics=['accuracy'])
>>>
>>> print("Running initial batch (compiling tile program)")
Running initial batch (compiling tile program)
>>> y = model.predict(x=x_train, batch_size=batch_size)
```

Ilustración 17: Ejecutándose el código dentro del entorno de "Py"

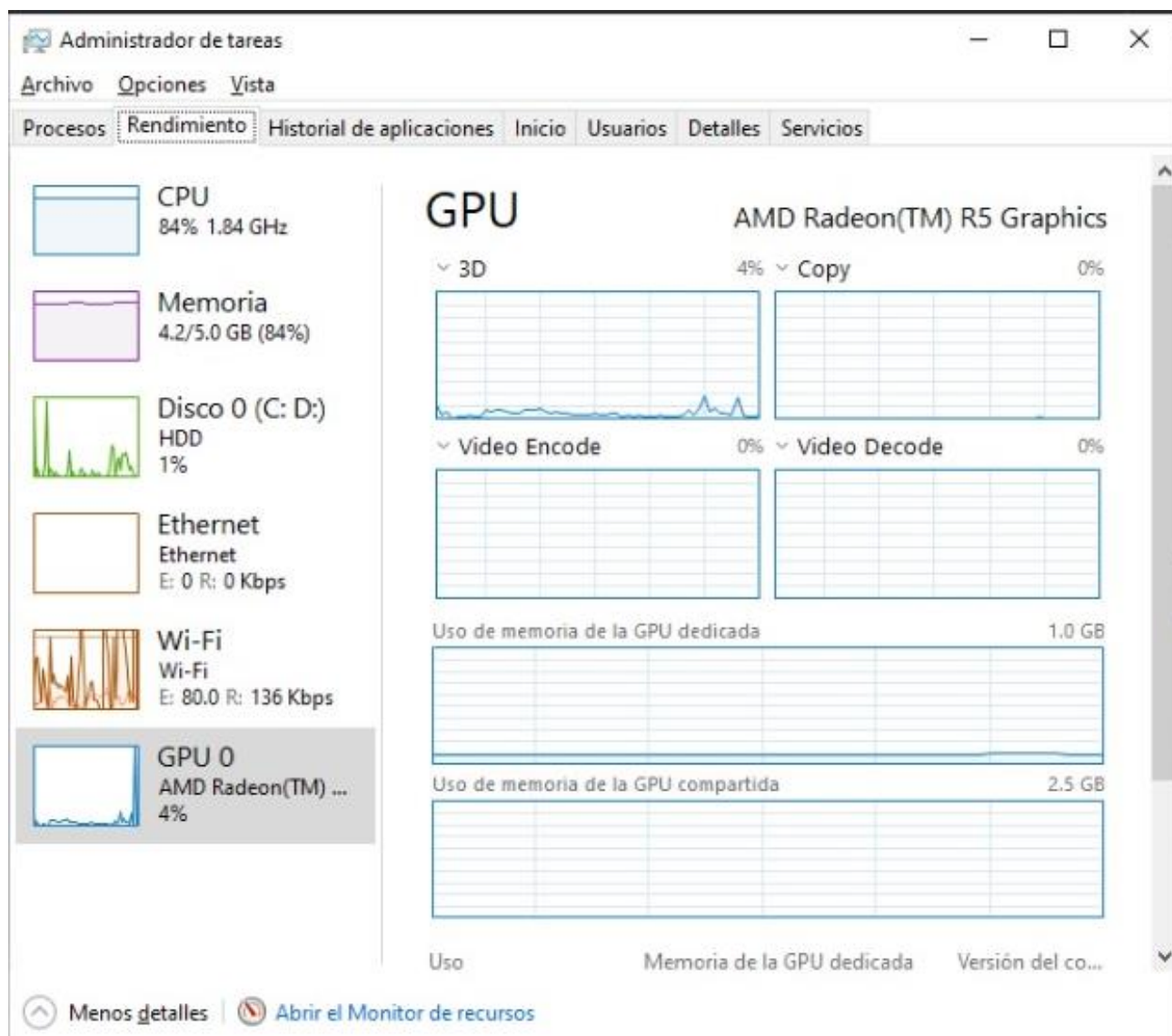


Ilustración 18: El Procesador, memoria y GPU de la computadora "2" con la ejecución del plaidML en "PY"

9. CONCLUSION GENERAL

Con todo lo anterior ya ejecutado se llegó a la siguiente conclusión:

PlaidML funciona en su totalidad dentro de las 2 computadoras ya que si no estuviera bien, nos mandaría varios errores apenas empezar, las observaciones principales se vieron dentro del código ya que nosotros al correr el código debería de mandar todo el trabajo a la tarjeta de video dedicada, pero gracias a que la separación de especificaciones no es correcta, o mejor dicho la más adecuada, su funcionamiento llega a ser bastante adecuado puesto que hace trabajar la tarjeta gráfica integrada pero por no ser la específica entra en un bucle el código y no encuentra la manera de salir para arrojar el resultado del tiempo de comparación entre los paquetes. Pero si se puede notar la forma en la que se ejecuta con el administrador de tareas, pues manda a trabajar al 99%-100% la gráfica integrada en este caso una Intel Graphics 3000. Se trató de hacer el cambio a la tarjeta dedicada Nvidia GeForce 520M que tiene la computadora "1" pero por ser ya una computadora bastante antigua no cumplía con los requerimientos necesarios para poder hacer la ejecución del PlaidML, por ende se calentaba demasiado la computadora "1" y se llegó a apagar en 4 ocasiones, pues el código se dejó ejecutando demasiado tiempo que sobrecalentó la computadora y esto no sabemos si a la larga afecte a esta computadora.

Por otra parte la computadora "2" tuvo resultados bastantes similares, solo que con diferentes variaciones:

En la computadora "2" se ejecutó de la misma manera las configuraciones que en la computadora "1" pero la computadora "2" al ser más actual sus procesos fueron un 40% más rápidos pero tuvo el mismo problema con la gráfica ya que esta no era la adecuada para el funcionamiento del PlaidML. Por otra parte gracias a las mejoras considerables que tiene la computadora "2" pues todos los tiempos que se encontraban en el administrador de tareas eran mucho más bajos que en la computadora "1" llegando la computadoras "2" a un máximo de 75% de trabajo del procesador, teniendo un AMD como gráfica se pudo experimentar un poco más pero llegando al mismo resultado de la computadora "1" que fue no lograr salir del bucle para así arrojar el tiempo de los paquetes. Las especificaciones principales de la computadora "2" son: Tarjeta de video integrada siendo una AMD Radeon R5 Graphics. Por el contrario de la computadora "1" la computadora "2" en ningún momento se apagó por tener unas condiciones más aptas, pero si se calentaba bastante, pues el ventilador se forzaba bastante.

Como conclusión final podemos decir que gracias a no cumplir cada computadora con los requerimientos específicos que se necesitan para realizar una buena ejecución plaidML, pues podemos observar que trabaja en el administrador de tareas, forzando las computadoras en un 100% y 75% respectivamente para realizar el proceso final de arrojar el resultado dentro del **"Visual Studio code"** y **"Py"** así que esta investigación fue un gran aprendizaje para todos ya que podemos comprobar que ya algunos software si necesitan tener sus especificaciones específicas para funcionar adecuadamente y no tener las trabas que se tuvieron en esta experimentación.

10. FUENTES DE INFORMACION

IBM. (n.d.). ¿Qué es Machine Learning? IBM. https://www.ibm.com/mx-es/analytics/machine-learning?p1=Search&p4=43700052827913158&p5=b&gclid=CjwKCAjw_JuGBhBkEiwA1xmbRWsCICUk0WFURvzS9kVIEtUNjaYPe1vOH4gRPUhBz3XF93wgfKqFsBoCdLMQAvD_BwE&gclidsrc=aw.ds

Metodología Kanban. (s. f.). Kanban Tool. Recuperado 22 de junio de 2021, de <https://kanbantool.com/es/metodologia-kanban>

WIKIPEDIA. (05/09/18). INTELIGENCIA ARTIFICIAL. WIKIPEDIA. https://es.wikipedia.org/wiki/Inteligencia_artificial

DEEP LEARNING. (05/03/2021). https://www.sas.com/es_mx/insights/analytics/deep-learning.html

PLAIDML (WIKIPEDIA ed.). (n.d.). <https://en.wikipedia.org/wiki/PlaidML>

Aceleradores (wikipedia ed.). (07/junio/21). https://es.wikipedia.org/wiki/Acelerador_de_part%C3%ADculas

GPU (4th ed.). (07/08/20). invidia. [nvidia.com/es-la/drivers/technologies/](https://www.nvidia.com/es-la/drivers/technologies/)

Tecnologías CPU (Vol. 1). (25/11/2017). Fernando Coelho. <https://computerhoy.com/noticias/hardware/que-es-como-funciona-cpu-64368#:~:text=La%20CPU%20se%20puede%20considerar,procesador%20de%20ordenador%20o%20microprocesador.&text=Aunque%20tambi%C3%A9n%20se%20puede%20insertar,conectar%20el%20microprocesador%20sin%7D>

Cris, Mehdi, Uday, Albert, Andy, Jacques...Oleksandr. (2002). MLIR: A Compiler Infrastructure for the End of Moore's Law. New York, USA

Copper, K. D. & Torczon, L. (2012). Engineering a compiler (2da ed.). Houston, Texas.

LLVM (2nd ed.). (08/octubre/20). llvm-admin. <https://llvm.org/>