Propuesta de Trabajo Profesional de Ingeniería en Informática

Oxidized Neural Orchestra

Tutor: Ing. Ricardo Andrés Veiga

Co-Tutor: Dr. Ing. José Ignacio Alvarez Hamelin

Alumnos

Alejo Ordoñez *(Padrón 108397)*
alordonez@fi.uba.ar

Lorenzo Minervino *(Padrón 107863)*
lminervino@fi.uba.ar

Marcos Bianchi Fernández *(Padrón 108921)*
mbianchif@fi.uba.ar

Facultad de Ingeniería, Universidad de Buenos Aires

Índice

1 Acta de Acuerdo

En la Ciudad Autónoma de Buenos Aires, al día (???) se reúnen en el Departamento de Informática de la Facultad de Ingeniería de la Universidad de Buenos Aires el profesor Ing. Ricardo A. Veiga y el profesor Dr. Ing. J. Ignacio Alvarez-Hamelin, con los estudiantes de la carrera de Ingeniería Informática el Sr. Alejo Ordoñez (Padrón: 108397), el Sr. Lorenzo Minervino (Padrón: 107863) y el Sr. Marcos Bianchi Fernández (Padrón: 108921) para tratar la elección y acuerdo del tema del Trabajo Práctico Profesional para el Ciclo Superior de la carrera.

Teniendo en cuenta la propuesta presentada por los alumnos más las observaciones y mejoras propuestas por los profesores, se ha acordado el plan de trabajo para el desarrollo e implementación del trabajo práctico profesional "Oxidized Neural Orchestra" que figura en el documento adjunto.

El acuerdo consiste en las siguientes pautas:

1. Los alumnos Sr. Alejo Ordoñez, Sr. Lorenzo Minervino y Sr. Marcos Bianchi Fernández realizarán todas las etapas para el análisis y realización de pruebas del proyecto.
2. El trabajo a realizar será presentado para cumplir los requisitos de la materia Trabajo Práctico Profesional.
3. El profesor Ing. Ricardo A. Veiga acepta la función de tutor (Facultad de Ingeniería, Universidad de Buenos Aires), y el profesor Dr. Ing. J. Ignacio Alvarez-Hamelin acepta la función de cotutor para dicho trabajo.

Alejo Ordoñez _____

Lorenzo Minervino _____

Marcos Bianchi Fernández _____

Ing. Ricardo A. Veiga _____

Dr. Ing. José Ignacio Alvarez Hamelin _____

2 Resumen

El entrenamiento de modelos de aprendizaje profundo, requiere de recursos computacionales elevados y largos tiempos de ejecución. Junto con la revolución de la Inteligencia Artificial, se volvió un tema candente el minimizar el tiempo de entrenamiento de estos modelos. Para lograr esto último, la estrategia más escalable es su ejecución distribuida.
Algunos de los problemas del entrenamiento distribuido son el manejo de los grandes volúmenes de datos, la sincronización del avance de cada uno de los nodos del sistema y el asegurar la convergencia. En base a esta problemática, en la actualidad están surgiendo muchas modificaciones de los principales algoritmos que implementan este entrenameinto distribuido. En particular, en este proyecto se estudian los dos algoritmos más relevantes: *Parameter Server* y *All-Reduce*, y sus derivados; sobre una implementación de un sistema distribuido que sirve como framework para el análisis.

3 Estado del arte

El principal desafío en la convergencia del entrenamiento distribuido de modelos de aprendizaje profundo radica en que cada paso de la optimización del error depende del anterior.
En un esquema de paralelización del entrenamiento por datos, existe el riesgo de incurrir en *overfitting* sobre las particiones si los pesos no se sincronizan con la frecuencia suficiente.

Por otro lado, si la sincronización se realiza con demasiada frecuencia, el tiempo de comunicación entre los nodos puede volverse dominante. Este costo no es en absoluto despreciable: si la comunicación representa una fracción significativa del tiempo total de entrenamiento, la distribución del cómputo pierde sentido, ya que la ejecución paralela termina siendo más lenta que el entrenamiento secuencial.

Este equilibrio entre **convergencia** y **eficiencia de comunicación** ha motivado una amplia línea de investigación. En la actualidad, existen dos enfoques fundamentales que sirven como base para el desarrollo de nuevas técnicas: **Parameter Server** [1] y **All-Reduce** [2].
Uno de los casos de los claros casos de combinación de los algoritmos mencionados es **Strategy-Switch** [3], que inicia iterando sobre All-Reduce y, guiado por una regla empírica, sigue con Parameter Server asincrónico una vez que el modelo en entrenamiento se estabiliza; logrando así mantener la precisión del entrenamiento de *All-Reduce* y la reducción significativa del tiempo total de entrenamiento de *Parameter Server asincrónico*.

En este trabajo se estudian en profundidad ambos algoritmos base y se investigan los métodos que de ellos derivan. Y se busca a partir de dicho análisis proponer mejoras mediante la combinación de estrategias o la optimización de sus componentes de comunicación y sincronización.

4 Objetivos

El trabajo tiene como objetivo principal la implementación de un sistema distribuido de entrenamiento de modelos de aprendizaje profundo, que sirva como base para la investigación de los trabajos previos, actuales y que surjan sobre esta temática. En particular, el desarrollo del trabajo conlleva:

1. Desarrollar un sistema distribuido de entrenamiento de modelos de aprendizaje profundo en Rust.
2. Implementar y comparar los distintos algoritmos que se utilicen para la ejecución del entrenamiento distribuido, sobre el sistema implementado.
3. Implementar una interfaz funcional externa para poder usar el sistema a desarrollar en Python.
4. Simular la ejecución sobre distintas configuraciones del sistema distribuido, para así obtener datos que se puedan analizar, y obtener comparaciones de los distintos algoritmos que se implementen, usando Python.
5. Confeccionar un informe detallado de la evolución del trabajo y los resultados obtenidos.

En lo académico, se busca utilizar y aplicar conocimientos adquiridos principalmente de las materias Aprendizaje Profundo, Redes y Sistemas Distribuidos; y de Fundamentos de Programación, Algoritmos y Estructuras de Datos, Paradigmas de Programación, Análisis Matemático II, Álgebra Lineal, Probabilidad y Estadística, Taller de Programación, Ciencia de Datos, Programación Concurrente, Simulación y Base de Datos. Además, existe un desafío adicional relacionado a la gestión del proyecto, la división de tareas y la estimación de tiempos, para lo cual se aplican contenidos relacionados a las materias Gestión del Desarrollo de Sistemas Informáticos e Ingeniería de Software (I y II).

5 Tecnologías

Las tecnologías que van a ser utilizadas para el desarrollo de este proyecto son:

- **Rust**: Se opta por el lenguaje de programación Rust para la implementación del sistema principal, porque ofrece: en primer lugar: la capacidad de editar código a bajo nivel, por la robustez del lenguaje, siendo que los requerimientos mínimos de compilación son más estrictos que la mayoría del resto de lenguajes, y que ofrece *fearless-concurrency*, haciendo checkeos estáticos de posibles problemas con la concurrencia de los programas, y por la relevancia que está cobrando en estos últimos tiempos.
- **Python**: Se va a usar Python para el análisis de datos obtenidos a partir de las comparaciones de los distintos algoritmos de machine-learning distribuido que serán llevados a cabo en el desarrollo del trabajo, y, por ser uno de los lenguajes más utilizados en la industria del aprendizaje profundo, para implementar una Interfaz de Funciones Externas (en inglés Foreign Function Interface, FFI) del resultado del sistema principal.
- **Docker**: Se hará uso de Docker para simular la ejecución del sistema en distintos entornos, y para agilizar la automatización de esto último.

6 Planificación

6.1 Gestión

Se establece un compromiso por parte de cada estudiante para dedicar un total de 500 horas al desarrollo del trabajo profesional. Esto representa, en promedio 15 horas semanales por persona a la ejecución de las tareas asignadas. Este compromiso se mantendrá a lo largo de 32 semanas (dos cuatrimestres). Además, se tiene previsto llevar a cabo encuentros periódicos en formato virtual entre los miembros del equipo y los tutores, cada semana. El propósito de estas reuniones es informar el avance y desarrollo del proyecto en curso. Asimismo, se abordarán aspectos como la definición de prioridades en las labores a realizar y la planificación requerida para la próxima etapa del proceso.

6.2 Tareas

Las principales tareas para llevar a cabo el desarrollo de este trabajo son:

1. Leer y analizar los trabajos previos, actuales y que surjan sobre el entrenamiento distribuido de modelos de aprendizaje profundo.
2. Investigar implementaciones existentes.
3. Desarrollar un sistema distribuido de entrenamiento de modelos de aprendizaje profundo en Rust, que permita la ejecución parametrizada por algoritmo.
4. Implementar y comparar los distintos algoritmos que se utilicen para la ejecución del entrenamiento distribuido, sobre el sistema implementado en (3).
5. Implementar una interfaz funcional externa para poder usar el sistema a desarrollar en Python.
6. Simular la ejecución sobre distintas configuraciones del sistema distribuido.
7. Analizar los datos obtenidos de la comparación de los algoritmos en , usando Python. nientes encontrados, etc).
8. Realizar un informe detallado de la evolución del trabajo y los resultados obtenidos.

6.3 Carga horaria

Tarea	Duración (hs)	Responsable
1. Revisión bibliográfica sobre entrenamiento distribuido en Deep Learning	90	A, L y M
2. Estudio de implementaciones existentes (TensorFlow, PyTorch, Horovod, etc.)	90	A, L y M
3. Desarrollo del sistema distribuido en Rust	510	A, L y M
4. Implementación y comparación de algoritmos de entrenamiento distribuido	300	A, L y M
5. Creación de la interfaz externa en Python	120	A, L y M
6. Simulación sobre distintas configuraciones del sistema distribuido	120	A, L y M
7. Análisis de datos comparativos usando Python	150	A, L y M
8. Informe detallado de evolución y resultados	120	A, L y M
Total	**1500**	A, L y M

7 Referencias

[1] M. Li *et al.*, «Scaling distributed machine learning with the parameter server», en *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, en OSDI'14. USA: USENIX Association, 2014, pp. 583-598.

[2] Z. Li, J. Davis, y S. Jarvis, «An Efficient Task-based All-Reduce for Machine Learning Applications», en *Proceedings of the Machine Learning on HPC Environments*, en MLHPC'17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3146347.3146350.

[3] N. Provatas, I. Chalas, I. Konstantinou, y N. Koziris, «Strategy-Switch: From All-Reduce to Parameter Server for Faster Efficient Training», *IEEE Access*, vol. PP, pp. 1-1, ene. 2025, doi: 10.1109/ACCESS.2025.3528248.