Facultad de Ingeniería
y Ciencias Hídricas
UNIVERSIDAD NACIONAL
DEL LITORAL

Proyecto Final de Carrera

Ingeniería en Informática

Título: **Implementación de un framework para la construcción de redes neuronales con aprendizaje profundo en forma distribuida. Caso de aplicación: clasificación de señales cerebrales.**

Estudiante: Ferrado, Leandro J.

Director: Dr. Rufiner, Hugo L.

Co-director: Bioing. Gareis, Iván E.

2015

Resumen:

El aprendizaje profundo es una rama de la inteligencia artificial que, debido al éxito de su utilización en problemas de gran complejidad, se ha vuelto popular y ampliamente desarrollado tanto a nivel empresarial como en el campo de la investigación. El mismo se basa en el diseño de redes neuronales capaces de aprender a extraer características sobre los datos presentados, para lograr con mayor precisión tareas de clasificación o regresión. La profundidad mencionada en su denominación se debe a que las arquitecturas implementadas suelen componerse de numerosos niveles, con el fin de lograr representaciones de los datos que sean adecuadas para el objetivo planteado sobre la red.

El problema que presenta implementar esta técnica es el elevado costo computacional, el cual se debe al cálculo comprendido en el entrenamiento de la arquitectura profunda. Esto ocurre especialmente en casos típicos de aplicación cuando se trata con cantidades masivas de datos. Además, el hecho de manejar una numerosa cantidad de hiper-parámetros y sus posibles combinaciones incrementa la dificultad de encontrar de forma rápida una red neuronal adecuada para el problema tratado.

En este proyecto se plantea como objetivo general el desarrollo de un "framework" que ofrezca la posibilidad de entrenar redes neuronales con los algoritmos y funcionalidades más populares de los que dispone el aprendizaje profundo, y validar la efectividad de dichos diseños utilizando herramientas de cómputo en paralelo. Esto último implica poder distribuir el trabajo computacional tanto a nivel local sobre los núcleos de un procesador, como también sobre los nodos que componen un clúster. Además se define como un objetivo específico evaluar la potencialidad del software implementado mediante su aplicación en tareas de alta complejidad, particularmente en problemas de clasificación sobre señales de electroencefalograma.

Palabras claves: ***red neuronal, aprendizaje profundo, autocodificador, clúster, electroencefalografía.***

Justificación

El aprendizaje profundo es una rama de la inteligencia artificial que, debido al éxito de su utilización en problemas de gran complejidad, se ha vuelto popular y ampliamente desarrollado tanto a nivel empresarial como en el campo de la investigación. El mismo se basa en el diseño de redes neuronales capaces de aprender a extraer características sobre los datos presentados, para lograr con mayor precisión tareas de clasificación o regresión. La profundidad mencionada en su denominación se debe a que las arquitecturas implementadas suelen componerse de numerosos niveles, con el fin de lograr representaciones de los datos que sean adecuadas para el objetivo planteado sobre la red.

Estas técnicas se presentaron como una mejora del enfoque tradicional que comprendía el aprendizaje maquinal, y en su surgimiento ha logrado hitos mejorando el estado del arte en tareas particulares: Hinton y su grupo de trabajo mejoraron la tecnología de reconocimiento de la voz en un 30% utilizando aprendizaje profundo[1]; Google utilizó el poder computacional de 16.000 computadoras para obtener un sistema que, mediante algoritmos de aprendizaje profundo, logró reconocer con gran precisión “rostros humanos y también de gatos” en videos de YouTube, sin utilizar información de clases [2](es decir, nunca se le daba información de lo que estaba aprendiendo).

El problema que presenta implementar estos algoritmos es el elevado costo computacional, el cual se debe al cálculo comprendido en el entrenamiento de la arquitectura profunda. Esto ocurre especialmente en casos típicos de aplicación cuando se trata con cantidades masivas de datos. Además, el hecho de manejar una numerosa cantidad de hiper-parámetros y sus posibles combinaciones incrementa la dificultad de encontrar de forma rápida una red neuronal adecuada para el problema tratado.

Dado que actualmente el acceso a clústeres de computadoras se encuentra más facilitado para investigadores y desarrolladores, en este proyecto se propone aprovechar este tipo de herramienta para distribuir el trabajo de los algoritmos profundos. Con ello se reduciría importantemente el tiempo de entrenamiento de las redes neuronales, y la búsqueda de hiper-parámetros se facilitaría al poder probar diversas configuraciones concurrentemente.

En este proyecto se opta por la metodología de construir redes utilizando autocodificadores apilados en la etapa de pre-entrenamiento [3]. Los autocodificadores son redes neuronales de tres capas, donde la capa de entrada y la de salida son la misma, y la oculta se entrena para aprender una representación de la capa de entrada pero en distinta dimensión. Esto es realizado con el objetivo de que dicha representación extraiga características que puedan facilitar la búsqueda de la mejor red neuronal profunda para la tarea asignada.

Al presente, se encuentran diversos frameworks con aprendizaje profundo utilizando cómputo paralelo. Se mencionan algunas para entender el trabajo actual en el tema.

H2O es una plataforma de código abierto escalable a nivel clúster, con interfaz de desarrollo en varios lenguajes, para análisis y predicciones sobre datos [4]. Ofrece la construcción de redes neuronales profundas pero con la particularidad de sólo incluir

autocodificadores profundos para problemas de regresión, y no de clasificación como lo tratado en el presente proyecto.

Deeplearning4j [5] es otro framework open-source hecho en Java, que también trabaja en forma distribuida y posibilita el uso de autocodificadores en el pre-entrenamiento de las redes profundas. Una posible desventaja es que no provee otra interfaz que la del lenguaje propio, por lo que dificulta su reutilización por investigadores que no manejan bien la programación orientada a objetos.

Como último ejemplo, Theano [6] es una biblioteca de funciones en Python que permite programar simbólicamente y compilar tanto en CPU como en GPU (no es escalable a clusters). Es muy utilizada en investigaciones que involucran aprendizaje profundo, aunque su estructura compleja para lograr la simbología dificulta su adopción por parte de desarrolladores.

Lo que se busca en este proyecto es lograr un producto de software que facilite la inserción del paradigma profundo de la inteligencia artificial en el tratamiento de problemas de alta complejidad relacionados con la Big Data [7]. Dicha facilitación se medirá respecto a dos ejes: la simplicidad en la estructuración del software para su prestación a ser reutilizable y útil para cualquier tipo de desarrollador; la potencia de cómputo adquirida en el trabajo de forma distribuida. Con estas propiedades, el producto obtenido aportaría a la comunidad una plataforma para utilizar las herramientas que ofrece el aprendizaje profundo y que evade el problema de lentitud en la construcción de las redes neuronales mediante el cálculo paralelo involucrado. Esta ventaja computacional a su vez será transparente al usuario para que el software abstraiga lo mayor posible su complejidad, de modo que puedan aplicarse todas las herramientas ofrecidas a cualquier tipo de datos de gran dimensión.

Dado que el software obtenido pretende tener utilidad en resolución de problemáticas, se define un caso de aplicación para analizar su desempeño y potencialidad. Por lo dicho inicialmente respecto al uso de este enfoque de inteligencia artificial, se define en el alcance de este proyecto el tratamiento de problemas de clasificación sobre señales de electroencefalograma (EEG). Dicha aplicación específica es un tema de amplio estudio debido, entre otras cosas, a su utilidad en construcción de sistemas de interfaz cerebro-computadora (ICC). Estos últimos se encargan de traducir la actividad cerebral en comandos para una computadora o dispositivo, lo cual podría implicar una importante utilidad para personas con discapacidades en la comunicación [8]. Un objetivo muy perseguido (y difícil de alcanzar) en ICC es conseguir comunicación confiable utilizando épocas únicas (del inglés single-trial), con el fin de maximizar la tasa de transferencia de información entre el usuario y el dispositivo o computadora a utilizar [9]. Diversos fenómenos fisiológicos pueden ser utilizados para generar la actividad cerebral que la ICC debe reconocer, los cuales determinan distintas problemáticas a tratar. En el proyecto se planea hacer una elección de algunas de las problemáticas disponibles a tratar. A priori, una de las problemáticas más relevantes en la actualidad es la del habla imaginada, en la cual en la señal de EEG se busca detectar los efectos producidos debido a la imaginación de pronunciar palabras o vocales sin realizar movimientos. Es un fenómeno fisiológico estudiado en la actualidad con enfoque de aprendizaje maquinal pero no profundo, por lo cual este proyecto podría establecer una tendencia para el análisis del estado del arte.

Objetivos Generales:

- Desarrollar un framework con algoritmos de aprendizaje profundo para entrenamiento y validación de redes neuronales, con posibilidad de distribuir el trabajo computacional sobre una computadora y/o una red de ellas.
- Aplicar la implementación obtenida en problemas de clasificación sobre datos de señales cerebrales, de manera de analizar la potencialidad de las herramientas desarrolladas.

Objetivos Específicos:

- Definir las funcionalidades y herramientas que ofrecería el framework.
- Investigar e implementar un motor de procesamiento distribuido para lograr el paralelismo computacional escalable a clústeres.
- Lograr la concurrencia para el entrenamiento de distintas redes neuronales a través de los nodos de un clúster.
- Lograr un software con un código suficientemente documentado en todas sus funcionalidades.
- Conseguir una interfaz para posibilitar al usuario la abstracción del cómputo paralelo implicado en el software.
- Desarrollar un protocolo de experimentación sobre el caso de aplicación, definido en base a las funcionalidades implementadas.
- Llevar adelante las pruebas especificadas en el protocolo de experimentación.
- Obtener resultados mejores que el azar sobre los datos tratados.

Alcance

Se definen las siguientes cuestiones para el alcance del producto logrado con el proyecto:

- Se implementarán algoritmos de aprendizaje profundo con el enfoque de autocodificadores para el pre-entrenamiento de las redes neuronales.
- El framework será pensado para ser reutilizado por desarrolladores, por lo que se prioriza su estructuración simple y fácil legibilidad en el código. A raíz de ello, se decide utilizar Python como lenguaje de programación.
- La plataforma desarrollada se basará en el motor de procesamiento distribuido Apache Spark (de código abierto) [10] que combina un gran poder de cómputo paralelo como su relativa facilidad de uso.

- Se debe tener la posibilidad de realizar el trabajo de cómputo distribuido, tanto a nivel clúster como a nivel local (sobre los hilos de ejecución de una sola computadora).

- Las señales cerebrales utilizadas para los casos de aplicación se consideran ya pre-procesadas de forma adecuada para ser utilizadas en las redes neuronales, de manera que el proyecto no incluye la investigación en las problemáticas involucradas en dichos casos.

- Las bases de datos a utilizar serán obtenidas exclusivamente de forma gratuita.

Metodología:

Dado que este proyecto consiste de un híbrido entre desarrollo de software y aplicación en casos de investigación, la metodología resultante contempla en sus etapas dichos aspectos. La misma se define en base a un ciclo de vida para el software con un modelo en cascada, en el cual las etapas se llevan a cabo de una forma prácticamente lineal sin iterar sobre todo el ciclo. Esto se justifica con el hecho de que se encuentran suficientemente pre-definidos los requisitos en el alcance ya presentado. Además, la arquitectura del software se considera estable desde el principio, ya que está basada en un motor de procesamiento validado y altamente mantenido. Todo esto permite que el desarrollo del framework se logre con mayor rapidez que utilizando modelos iterativos.

Etapas:

1. Estudio estratégico y recolección de requisitos:

Se realiza un estudio de los frameworks existentes que estén relacionados con el del presente proyecto. Se analizan las soluciones implementadas en cuanto al paralelismo de cómputo, los algoritmos de aprendizaje profundo que soportan dicho paralelismo, las tecnologías involucradas en las implementaciones, y demás cuestiones relacionadas a los objetivos y alcances del software a desarrollar. Además, se elige y describe la problemática a tratar para el caso de aplicación en señales de electroencefalograma (EEG), y se investiga acerca de los tratamientos que se han realizado en otros trabajos. Pueden definirse otras problemáticas secundarias a analizar, siempre y cuando estén relacionadas con la clasificación de señales cerebrales.

Con todo el análisis estratégico elaborado, se detallan los requisitos de todo tipo referidos la implementación a obtener, y se plasma en un documento toda especificación relacionada con el caso de aplicación a tratar.

Entregables:

- **Documento de requerimientos**. Criterio de aceptación: El mismo debe presentar todos los requisitos con su clasificación, y tiene que ser congruente con el alcance del proyecto y abarcar todos sus aspectos.
- **Especificación del caso de aplicación.** Criterio de aceptación: Debe detallar al menos una problemática en la cual se trate con la clasificación de señales de EEG (cuyos

datos sean posibles de adquirir de forma gratuita), incluyendo trabajos ya realizados sobre el tema.

Hito 1: Interiorización en los aspectos a trabajar en el proyecto, con información suficiente del estado del arte y de las posibles soluciones a ofrecer.

2. Diseño de propuesta de solución:

En base a lo analizado inicialmente, se utilizan los requisitos recopilados y la declaración del alcance del proyecto para concretar el diseño de la implementación. Para ello se definen las funcionalidades y herramientas a desarrollar, las tecnologías a utilizar y las interfaces que va a proveer la plataforma final. Dicha definición viene ligada con la propuesta de solución para el caso de aplicación. Esta se escoge a partir del estudio anterior realizado, y debe ajustarse a las técnicas de aprendizaje profundo pactadas en el diseño del framework (incluyendo uso de autocodificadores como se especificó en el alcance). Aun así, la propuesta resultante se considera sólo como enfoque inicial para abordar la solución a la problemática correspondiente, y probablemente sufra ajustes luego de adquirirse un mayor marco teórico y principalmente en la experimentación final.

Entregables:

- **Definición de arquitectura del framework:** Criterio de aceptación: El mismo debe ser acorde con el alcance y los objetivos del proyecto planteados, así como abarcar todos los requisitos documentados.
- **Propuesta de solución para caso de aplicación**: Criterio de aceptación: No puede incluir técnicas que no estén contempladas en la arquitectura diseñada para el software. Debe especificar al menos un tratamiento similar a alguno ya realizado en otro trabajo de investigación, con los resultados correspondientes para poder efectuar una comparación.

Hito 2: Determinación de las particularidades del proyecto, con el diseño de soluciones conciso y medible para el control durante la ejecución del trabajo.

3. Obtención de recursos y del marco teórico:

Se procede a adquirir los recursos y herramientas necesarias para el desarrollo del framework y de su aplicación: el motor de procesamiento en forma distribuida, que permita escalar el cómputo realizado por la plataforma hacia clústeres de computadoras; el acceso a clústeres computacionales, en una etapa temprana para configuración y pruebas iniciales de los algoritmos; las bases de datos de señales cerebrales reales, referidas a la problemática a tratar (de dominio público y preferentemente ya pre-procesadas). En conjunto, se obtiene además un marco teórico necesario para el desarrollo de las técnicas pactadas (referido al aprendizaje profundo con autocodificadores, y al paralelismo a conseguir en los algoritmos) y a su implementación en el caso de aplicación.

Hito 3: Disposición de todos los recursos y conocimientos necesarios para efectuar el trabajo del proyecto.

4. Producción y testeo del software:

A partir del diseño definido, se procede a realizar las actividades referidas al desarrollo del framework. Se implementan los algoritmos de aprendizaje profundo acordados (en base

al marco teórico adquirido), con respaldo del motor de procesamiento paralelo ya integrado, y se produce la interfaz necesaria para que el software logre su calidad de plataforma para desarrolladores. A raíz de esto último se hace un especial énfasis en la documentación de todos los módulos, de manera que sirva de guía clara a los desarrolladores con propósitos de reutilización. Cada una de las funcionalidades es testeada tanto a nivel local (una sola computadora) como a nivel clúster (muchas computadoras) para evaluar el desempeño de la implementación, hasta finalmente validar la integración de todos los módulos de la plataforma desarrollada.

Por otro lado, también se construye un protocolo de experimentación sobre los datos relacionados con los casos de aplicación. El mismo se utilizará sobre el software logrado cuando éste se encuentre validado y en correcto funcionamiento.

Entregables:

- **Framework desarrollado y en funcionamiento**. Criterio de aceptación: El mismo debe ser compatible con los datos obtenidos para la problemática a tratar, y disponer de funcionalidades que utilicen autocodificadores en el entrenamiento de redes neuronales profundas. Además se exige que funcione correctamente a nivel local, con un margen de error adecuadamente bajo a nivel clúster. Todos los módulos deben estar documentados en el código fuente, y se requiere como mínimo que ofrezca una interfaz de scripts para uso mediante consola.
- **Protocolo de experimentos.** Criterio de aceptación: Cada experimento debe basarse en la propuesta de solución redactada, y en caso de que el conjunto sea extenso debe abarcar distintas configuraciones.

Hito 4: Software en funcionamiento adecuado para la experimentación determinada para el caso de aplicación.

5. Puesta en marcha y experimentación:

Una vez que la plataforma se encuentra validada en cada una de sus funcionalidades, se procede a utilizarla para experimentar con las propuestas de solución formuladas acerca de los casos de aplicación. Se provee al framework con los datos de señales cerebrales, y en base a configuraciones y selección de parámetros, se realiza el tratamiento mediante inteligencia artificial para la clasificación de estas señales. Los resultados obtenidos son documentados junto a la especificación del experimento correspondiente. Además, la experimentación sobre estos casos de aplicación permite examinar el comportamiento del software producido, para que en caso de encontrar fallas puedan ser reparadas a tiempo.

Entregables:

- **Framework validado.** Criterio de aceptación: Correcto funcionamiento de la plataforma en todos sus módulos, con interfaz definida para acceder a todas las herramientas disponibles.
- **Resultados de experimentos.** Criterio de aceptación: Deben corresponderse a todos los experimentos especificados en el protocolo.

Hito 5: Implementación de framework finalizada y validada mediante su aplicación en la problemática de investigación definida.

6. Análisis de resultados e informe de trabajo:

Con los resultados alcanzados sobre la investigación, se realiza un estudio de los mismos por medio de análisis estadísticos y métodos de validación. A partir de ello se realiza un informe de todo ese análisis, en comparación también con los trabajos ya existentes en cada problemática tratada. Finalmente se procede a la redacción del informe final de proyecto, con la información de todo el trabajo realizado en el framework e incluyendo la investigación efectuada en la clasificación de señales cerebrales.

Entregable:

- **Informe final del proyecto, incluyendo investigación sobre el caso de aplicación**. Criterio de aceptación: Debe contener todo el marco teórico y práctico abarcado, y presentar los resultados obtenidos en la investigación, junto a las conclusiones generadas en base al análisis realizado.

Hito 6: Investigación sobre el caso de aplicación finalizada, y culminación del trabajo de proyecto.

Plan de tareas:

Se define el plan de tareas a seguir, con las actividades y sus correspondientes duraciones. El proyecto se realizará en un plazo de 136 días hábiles, donde se define un esfuerzo de 3hs diarias llevadas a cabo por una persona como recurso humano único. Las actividades comienzan el día 01/07/15 y se estima que finalizan el día 07/12/15. El diagrama de Gantt para el cronograma se presenta en la Fig. 1, siendo el camino crítico único por ser de carácter secuencial todas las

Actividad	Duración
1. Estudio estratégico y recolección de requisitos.	**54hs**
1.1. Revisión de implementaciones de aprendizaje profundo con autocodificadores	9hs
1.2. Revisión de plataformas de desarrollo de software utilizando cómputo distribuido.	9hs
1.3. Elección y descripción de problemática/s a tratar como caso de aplicación.	12hs
1.4. Documentación de requerimientos del proyecto.	24hs
2. Diseño de propuesta de solución	**75hs**
2.1. Definición de tecnologías y herramientas a utilizar.	9hs
2.2. Definición de funcionalidades a implementar.	24hs
2.3. Descripción de la arquitectura del software.	30hs
2.4. Descripción de tratamiento a realizar sobre el caso de aplicación.	12hs
3. Obtención de recursos y del marco teórico	**36hs**
3.1. Adquisición de recursos y herramientas.	9hs
3.2. Adquisición de conocimientos teóricos referidos a la implementación.	27hs
4. Producción y testeo del software.	**135hs**
4.1. Instalación de herramientas de software utilizadas.	6hs
4.2. Implementación de algoritmos de aprendizaje profundo.	42hs
4.3. Estructuración de módulos del framework.	24hs

4.4. Depuración y rectificación de módulos del framework.	36hs
4.5. Documentación de módulos y definición de interfaz del software	18hs
4.6. Definición de protocolo de experimentación sobre el caso de aplicación.	9hs
5. Puesta en marcha y experimentación	**60hs**
5.1. Experimentación con datos del caso de aplicación.	36hs
5.2. Validación final del framework implementado.	24hs
6. Análisis de resultados e informe de trabajo.	**48hs**
6.1. Análisis y discusión sobre los resultados de la experimentación.	18hs
6.2. Redacción de informe final del proyecto	30hs
Total:	**408hs**

Los entregables antes descriptos son destinados a los directores del proyecto, mientras que se define como entregables hacia la cátedra de Proyecto Final de Carrera los siguientes:

- **Informe de avance 1**.

Fecha de entrega: **03/09/15** (*Hito 3*).

Contenido: Se hace referencia al trabajo realizado en las etapas 1, 2 y 3. Contiene toda la información de planificación y diseño del framework a realizar y su aplicación.

- **Informe de avance 2**.

Fecha de entrega: **18/11/15** (*Hito 5*).

Contenido: Se incluye información referida a las etapas 4 y 5. En el mismo se comunican las actividades desempeñadas en la ejecución del proyecto, tanto de producción del software como su prueba con el protocolo de experimentos diseñado.

- **Informe de avance 3**.

Fecha de entrega: **03/12/15**.

Contenido: Se informan las actividades finales del proyecto, dadas hasta la tarea 6.1, y se realiza un cierre del trabajo de proyecto informando los resultados finales del mismo.

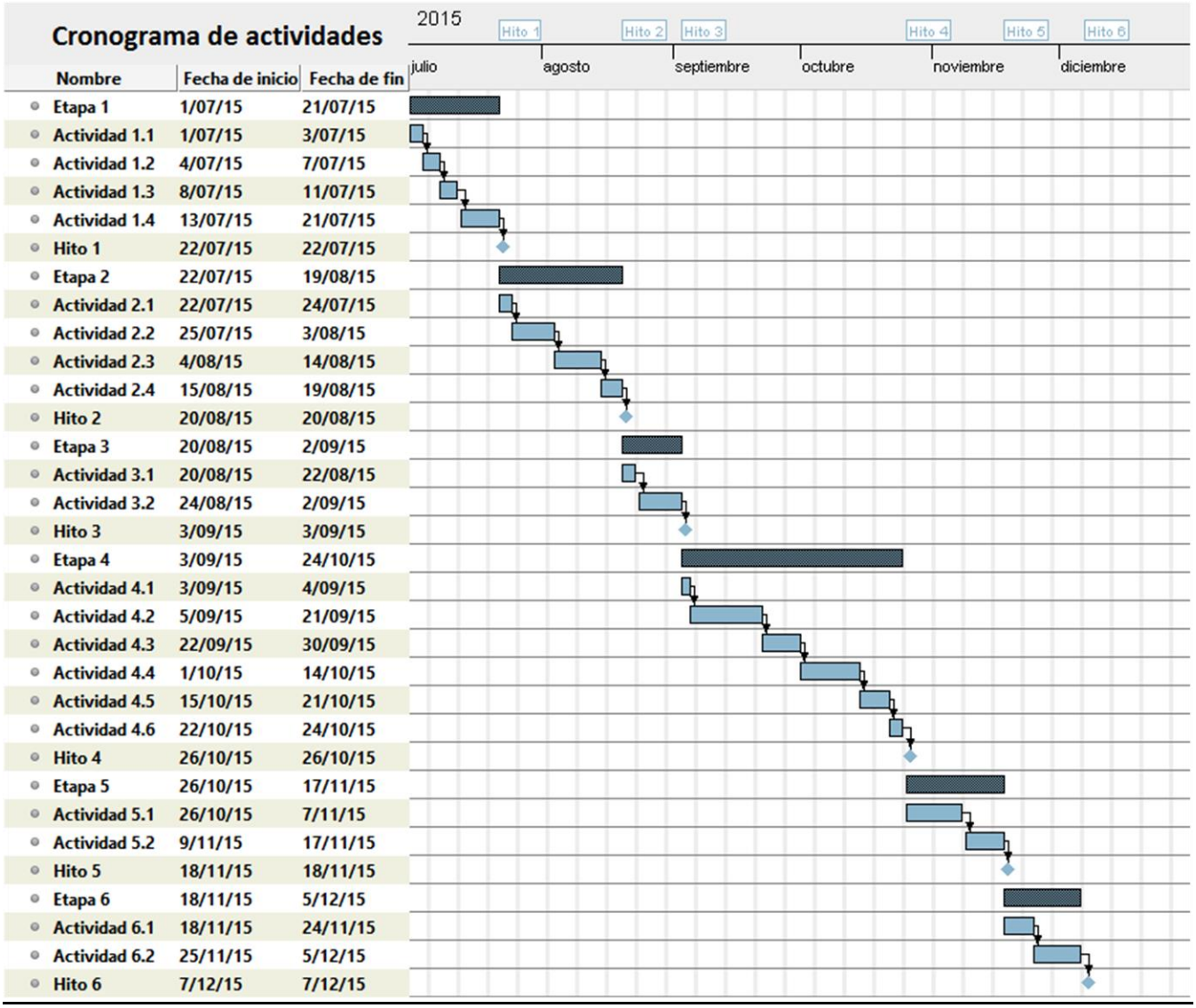

Fig. 1: Diagrama de Gantt del proyecto (realizado con GanttProject).

Riesgos:

En el siguiente apartado se muestran detallados los riesgos identificados para el proyecto en cuestión. En la Fig.2 se muestra la matriz de prioridad utilizada para asignar la estrategia de respuesta apropiada a cada riesgo en base a su probabilidad de ocurrencia y el impacto que puede generar sobre el proyecto. En la Tabla 1 se mencionan cada uno de los riesgos identificados, y se especifican las actividades a realizar como respuesta a los mismos.

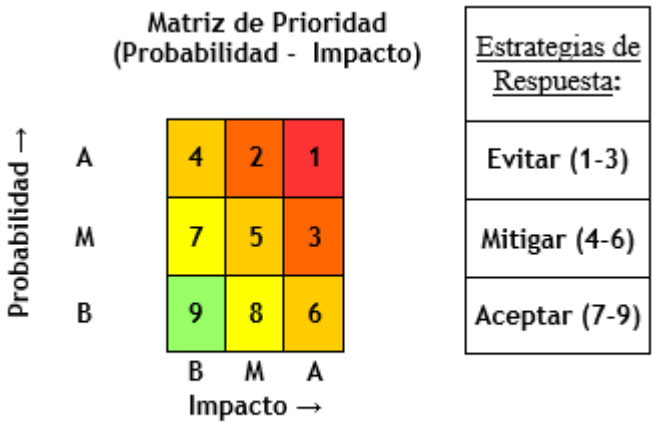

Fig. 2: Detalles de prioridad y respuesta a riesgos

ID	Riesgo	Consecuencia	Síntoma	Probabilidad	Impacto	Prioridad	Respuesta
R001	Incapacidad de integración del motor de procesamiento distribuido en las actividades de producción del software.	Retraso en las actividades de integración y testeo del software.	Dificultad de uso del motor y funcionamiento incorrecto durante las pruebas de integración	Baja	Alto	6	Mitigar probabilidad: Capacitación mediante cursos online y lectura de libros acerca del motor en cuestión y su integración en producción de software.
R002	Incompatibilida-des al migrar el trabajo del framework de la PC a los nodos del clúster.	Incumplimiento en el alcance del proyecto, referido a la paralelización del trabajo en clústeres.	Incorrecto funcionamiento de la implementación al ser ejecutada en los nodos de un clúster.	Media	Alto	3	Evitar: Revisión de arquitecturas y paradigmas de otras implementaciones que utilicen el motor de procesamiento elegido a nivel clúster, y adopción de una estructura similar para asegurar la compatibilidad en el paralelizado.

R003	Carencia/deficiencia de la base de datos correspondiente al caso de aplicación elegido.	Imposibilidad de experimentación en el software sobre el caso de aplicación.	Dificultad para conseguir datos referidos a la problemática escogida/ Desperfectos e inconsistencias en la base de datos a utilizar.	Baja	Alto	6	Mitigar probabilidad: Elegir problemáticas secundarias en la etapa de Diseño de propuesta de solución, con la seguridad de que existan bases de datos disponibles y requiriendo un menor estudio de las mismas en la etapa de Obtención de recursos y de marco teórico.
R004	Inconsistencias que surjan durante la experimentación, en el comportamiento de las funcionalidades elegidas.	Incumplimiento con el protocolo de experimentación definido.	Incorrecto funcionamiento durante las pruebas con resultados sin sentido.	Baja	Alto	6	Mitigar impacto: Definir en el protocolo de experimentación un apartado de pruebas más reducido utilizando las funcionalidades más estables que se determinaron durante el testeo de los módulos codificados.
R005	Nodos de clústeres no disponibles por el tiempo necesario para completar la experimentación.	Retraso del inicio de actividades de cierre del proyecto.	Limitación o saturación de uso de los nodos del clúster en cuestión, debido a estar compartido el uso con otros usuarios.	Media	Medio	5	Mitigar impacto: Realizar un grupo reducido de experimentos a nivel clúster y dejar el resto para que se lleven a cabo a nivel local sobre los núcleos de proceso en la computadora disponible para el proyecto.
R006	No disponibilidad del responsable de la ejecución del proyecto.	Retraso de todas las actividades a cargo del responsable del proyecto.	Demora en la ejecución de las actividades por inconvenientes de carácter personal.	Baja	Alto	6	Mitigar impacto: Estipular un margen de tiempo razonable para cada etapa del proyecto destinado a la contingencia de dichos inconvenientes.
R007	No disponibilidad de los directores del proyecto.	Retraso en las actividades de control y de la presentación de los entregables del proyecto.	Demora o falta de respuesta ante la solicitud de revisión del proyecto.	Baja	Medio	8	Aceptar: Informar frecuentemente a los directores sobre los avances del proyecto para evitar la necesidad de reuniones extensas difíciles de coordinar.

R008	Experimentación lenta e insuficiente sobre el caso de aplicación.	Retraso del inicio de actividades de cierre del proyecto.	Demoras en las pruebas y obtención de resultados no satisfactorios.	Medio	Medio	5	Mitigar impacto: Replanificar el protocolo reduciendo la cantidad de experimentos, y la combinación de parámetros especificada, para realizar por completo sólo algunas de las pruebas diseñadas.

Tabla 1: Riesgos identificados con sus especificaciones determinadas.

Recursos necesarios y disponibles:

Se cuenta con los siguientes recursos disponibles para el desarrollo del proyecto:

- Computadora personal: se dispone de una notebook Asus N56VB de propiedad personal. Cumple con el requisito funcional de poseer un procesador multi-núcleo, fundamental para el esquema distribuido a tratar en el trabajo de proyecto.
- Software: todos los programas necesarios para el proyecto a ejecutar serán de carácter gratuito al pertenecer a tecnologías libres.
- Bases de datos para caso de aplicación: no se encuentra disponible al inicio del proyecto, pero se encuentran bajo la restricción de ser adquiridas únicamente de forma gratuita.
- Bibliografía: se dispone de todo el material provisto por la biblioteca centralizada "Dr Ezio Emiliani" ubicada en la FICH. El acceso a las publicaciones científicas utilizadas es concedido gratuitamente a través de la red local instalada en el edificio de la FICH, específicamente mediante la biblioteca electrónica del Ministerio de Ciencia, Tecnología e Innovación Productiva.
- Servicio de clúster computacional: el mismo es necesario para cumplir los objetivos del proyecto, y se consigue su acceso en la Etapa 3.
- Servicios del proyecto: disponibles en el área de trabajo (Internet, electricidad, etc.).

Presupuesto:

El presupuesto estimado para la realización del proyecto será de $126.069. Se detallan a continuación los costos referidos a los recursos definidos.

- Bienes de capital:
  - Notebook Asus N56VB S3065H DF Intel Core i5 (amortización);
    - Valor a nuevo (VN): $22000
    - Valor residual (VR): $1500
    - Vida útil (VU): 12000 horas
    - Monto de amortización: $\frac{VN-VR}{VU} * horas\ de\ uso \approx \$697$

- Consultorías y Servicios:
  - Servicio de clúster computacional (Costos directos):
    - Estimación aproximada proporcional a siguientes requisitos:
      - 4 nodos por clúster
      - 4 núcleos por nodo
      - 7 GB de RAM por nodo.
      - 100 GB de almacenamiento
    - Costo: $3,60/h
    - Total $864.
- Materiales e Insumos (Costos directos):
  - Librería. Impresiones de informes y material bibliográfico, encuadernado y fotocopias. Total: $620.
- Viajes y Viáticos (Costos directos):
  - Transporte urbano: $9 por día de trabajo. Total: $1224.
  - Merienda: $50 por mes de trabajo. Total: $250.
- Recursos humanos (Costos directos):
  - Remuneración propia: se considera el 70% de la remuneración definida por el Colegio de Ingenieros Especialistas de Santa Fe, al no estar titulado como ingeniero.
    - Rol de analista funcional (Etapas 1, 2, 3 y 6). Monto: $160/h. Total: $34080.
    - Rol de analista programador y tester (Etapas 4 y 5). Monto: $130/h. Total: $25350.
    - Total: $59430.
  - Remuneración del Director de proyecto:
    - Rol de Líder de Proyecto. 120hs.
    - Monto por hora de trabajo: $300.
    - Total: $36000.
  - Remuneración del Co-director de proyecto:
    - Rol de Líder de Proyecto. 88hs.
    - Monto por hora de trabajo: $300.
    - Total: $26400.
- Otros costos (Costos indirectos):
  - Electricidad:
    - Potencia usada por PC: 440W.
    - Energía consumida durante el proyecto: $440W \times 408hs \approx 180kWh$
    - Costo del kWh : $1,3.
    - Costo total: $234.

- Conexión a Internet: El servicio es abonado por la FICH y brindado a todos los miembros de dicha facultad. Costo estimado: $70 por mes de trabajo. Total: $350.

Referencias:

[1] Hinton, G., Deng, L. et. al. (2012): “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups” en *Signal Processing Magazine, IEEE, 29(6), 82-97*.

[2] Le, Q. V. (2013): “Building high-level features using large scale unsupervised learning” en *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on (pp. 8595-8598)*. IEEE.

[3] Hinton, G. E., Salakhutdinov, R. R. (2006): “Reducing the dimensionality of data with neural networks”. Science, 313(5786), 504-507.

[4] Fu, A., Aiello, S., et. al. (2015): Package ‘h2o’.

[5] http://deeplearning4j.org/compare-dl4j-torch7-pylearn.html, DL4J vs. Torch vs. Theano vs. Caffe. Consultado el 20/05/15.

[6] Bergstra, J., et. al. (2010): “Theano: a CPU and GPU math expression compiler” en *Proceedings of the Python for scientific computing conference (SciPy)* (Vol. 4, p. 3).

[7] Najafabadi, M et. al. (2015): “Deep learning applications and challenges in big data analytics” en *Journal of Big Data, 2(1),* 1-21.

[8] Hoffmann, U. et. al. (2008): “An eficient P300-based brain-computer interface for disabled subjects” en *Journal of Neuroscience methods*.

[9] Blankertz, B., Curio, G., Muller, K. R. (2002): “Classifying single trial EEG: Towards brain computer interfacing” en *Advances in neural information processing systems, 1, 157-164*.

[10] Spark, A. (2014): “Apache spark–lightning-fast cluster computing”.

Bibliografía:

Karau, H. et. al. (2015): “Learning Spark. Lightning-Fast Big Data Analysis”. O'Reilly Media.

Pentreath, N. (2015): “Machine Learning with Spark”. Packt Publishing.

Langtangen, H. (2008): “Python Scripting for Computational Science”. Springer. Tercera edición.

González-Castañeda, E. F., Torres-García, A. A. et al (2014): “Sonificación de EEG para la clasificación de palabras no pronunciadas” en *Research in Computing Science 74*.

DaSalla, C. et al (2009): “Single-trial classification of vowel speech imagery using common spatial patterns” en *Neural Networks 22 (9) (pp. 1334-1339)*.

I. E. Gareis, G. Gentiletti, R. C. Acevedo, H. L. Rufiner (2011): “Feature extraction on Brain Computer Interfaces using Discrete Dyadic Wavelet Transform: Preliminary results” en Journal of Physics: Conference Series (IOP), Volume 313, Number 12011 (pp. 1-7).

Bengio, Y (2009): “Learning deep architectures for AI” en *Foundations and Trends® in Machine Learning archive.*

Haykin, S. (2009): “Neural Networks and Learning Machines”. Prentice Hall. Tercera edición.

Erhan, D. et al (2010): “Why does unsupervised pre-training help deep learning?” en *J. Mach. Learn. Res., 11:625-660*.

Bishop, C.M. (1995): “Neural Networks for Pattern Recognition”. Oxford: Oxford University Press.

Project management institute, inc (2013): “Guía de los Fundamentos para la Dirección de Proyectos”. PMI Book. Cuarta edición.

Emiliani, F. (1995): “Proyectos de investigación científica”. UNL-CONICET-ACNL.

Ander Egg, E, Aguilar Idañez, M. (1998): “Cómo elaborar un proyecto”. Editorial Lumen.