PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Identificación y clasificación automática de repeticiones en estructuras de proteínas repetidas

Tesis para obtener el grado académico de
Magíster en Informática con mención en Ciencias de la Computación que presenta:

Luis Fernando Muroya Tokushima

Asesora:

Dra. Layla Hirsh Martinez

Lima, 2021

Agradecimientos

A mi familia, por la vida, educación y apoyo que me han dado siempre
A mi asesora, Layla Hirsh, por la orientación y amistad brindada en estos años
A mis profesores, por su apoyo y empeño constante dentro y fuera de aulas
A mi amiga, Ana Paula Galarreta, por sus consejos y ánimos en los momentos más difíciles que he tenido que vivir en el trascurso de esta aventura.
A mi amigo, Gerardo Cardoso, por todos los trabajos en grupo y los ánimos brindados cuando la carga se hacía muy pesada
A todos mis amigos, por ser mis soportes durante estos años.
A mis jefes y compañeros de trabajo, por darme las facilidades para poder vivir esta aventura del saber.

Resumen

Las proteínas repetidas son proteínas no globulares caracterizadas por la presencia de repeticiones a nivel de secuencia y estructura. Pueden ser de 5 clases, cada una con un número variable de subclases. Estas proteínas son relevantes porque están relacionadas con una diversidad de enfermedades. Su correcta clasificación es parte fundamental para su estudio; sin embargo, la anotación manual de todas las estructuras de proteínas conocidas es una tarea que es logísticamente imposible completar. Por ello, la automatización de esta tarea es muy importante.

En el presente trabajo, se desarrolló una herramienta para la identificación y clasificación de repeticiones de clase IV. Esta herramienta fue construida por el acoplamiento de dos módulos: uno de filtro y otro de clasificación. El primero fue construido reutilizando una red neuronal convolucional entrenada para la detección de simetrías rotacionales en la estructura de una proteína. Su uso estuvo fundamentado en el hecho que las repeticiones clase IV son de estructura cerrada, por lo que la presencia de simetrías rotacionales era altamente probable. Para el módulo de clasificación se transformó la información estructural en imágenes, por medio del cálculo y superposición de tres matrices. Estas imágenes fueron usadas para aplicar una técnica de transferencia de aprendizaje a una red Densenet, seleccionada luego de un análisis cualitativo y cuantitativo. Como resultado, el clasificador obtenido logró una exactitud de 89.8% sobre una muestra de 658 cadenas de proteínas.

Los anteriores módulos fueron integrados en un servicio web construido sobre Flask. Se construyó una aplicación de una página (SPA) para hacer disponible dicho servicio en una forma amigable con el usuario. Dicha aplicación fue desplegada en la nube para su acceso.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Capítulo 1.

Aspectos generales

En el este capítulo se presentará el proyecto de investigación: su objetivo, alcance, viabilidad y aspectos metodológicos.

1.1. Problemática

Las proteínas son moléculas orgánicas con una gran importancia por la diversidad de funciones que desempeñan. Desde la queratina de las uñas hasta la fibroína de la seda, las proteínas se encuentran en todos los organismos vivos (McMurry, 2010). A pesar de su gran diversidad, pueden ser clasificadas en solo dos grandes grupos: fibrosas y globulares (McMurry, 2010). Las proteínas fibrosas como el colágeno suelen ser rígidas e insolubles en agua (McMurry, 2010). Las proteínas globulares; por el contrario, son móviles y solubles en agua (McMurry, 2010).

Entre las proteínas no globulares (fibrosas) se encuentran las proteínas repetidas, las cuales se caracterizan por presentar repeticiones a nivel de estructura y secuencia (Paladin et al., 2017). Estas biomoléculas tienen funciones biológicas importantes: son dominios de unión macromolecular, funcionan como enzimas y constituyen unidades para la formación de materiales fibrosos (Doyle et al., 2015). Por otro lado, están relacionadas con una variedad de enfermedades como la artritis reumatoide, el cáncer de próstata, la distrofia miotónica, la enfermedad de Huntington, entre otras (Jorda, Xue, Uversky y Kajava, 2010).

Por lo anteriormente mencionado, la detección y anotación (clasificación) de estas proteínas es importante para mejorar la comprensión de sus mecanismos de patogenicidad, el diseño de proteínas para el tratamiento de enfermedades como el cáncer e incrementar nuestro conocimiento de la estructura y funcionamiento de estas biomoléculas (Paladin et al., 2017). En este contexto, el uso de las computadoras ha apoyado la labor de investigación; en particular, para llevar a cabo cálculos y simulaciones complejas (Hirsh, 2018). Este apoyo ha permitido el ahorro de tiempo y dinero en muchos proyectos; sin embargo, no elimina la necesidad de la experimentación para la validación de los resultados obtenidos (Hirsh, 2018).

Dada la importancia que tienen la detección y clasificación de proteínas repetidas y la imposibilidad de anotar manualmente las 167 mil estructuras de proteínas disponibles en el banco de proteínas (cifra obtenida a partir de su API web[1]), la automatización de ambas tareas ha sido estudiada ampliamente en los últimos años. La mayor parte de estas investigaciones toman uno de dos posibles enfoques: basarse en la información proveniente de la secuencia o basarse en la información de la estructura (Pellegrini, 2015).

Los métodos basados en secuencia, como refleja su nombre, utilizan la secuencia de aminoácidos de la proteína para identificar y clasificar sus repeticiones (Pellegrini, 2015). Algunos de los métodos se basan en el alineamiento de cadenas; otros, en aplicar técnicas de aprendizaje no supervisado; e incluso, aplicar técnicas de aprendizaje profundo (Pellegrini, 2015). Contrario a lo que puede parecer, la respuesta a este problema no es trivial con estos algoritmos, ya que una misma configuración y funcionalidad puede estar asociada a dos o más secuencias repetidas distintas (Turjanski, Parra, Espada, Becher y Ferreiro, 2016). Dicho de otra manera, las secuencias pueden presentar divergencias y observarse en todas ellas una misma estructura y función. Esta divergencia se produce por dos factores: los errores en los procesos de duplicación y la tendencia intrínseca a la divergencia por parte de las unidades repetitivas (Paladin et al., 2017).

Los algoritmos basados en estructura no sufren esta dificultad porque la funcionalidad de una proteína está más vinculada con su estructura tridimensional que con su secuencia de aminoácidos (Pellegrini, 2015). Muchos algoritmos dentro de este grupo se basan en la estrategia de alinear estructuras (Pellegrini, 2015). Sin embargo, esta técnica es computacionalmente costosa por tener que considerar las transformaciones (rotaciones y/o traslaciones) y la flexibilidad de la proteína al momento de realizar la alineación (Pellegrini, 2015). ReUPred representa una mejora ante esta situación, ya que realiza una búsqueda iterativa sobre una librería de estructuras (Hirsh, Piovesan, Paladin y Tosatto, 2016). Por su parte, otros investigadores también han abordado el problema buscando formas alternativas de explotar la información de estructura; por ejemplo, mediante el uso de grafos y de programación dinámica (Pellegrini, 2015). Sin embargo, el uso de aprendizaje automático en este campo ha sido limitado, pues solo se tiene registro de Deep Symmetry, una red neuronal entrenada para detectar simetrías

[1] *http://www.rcsb.org/pdb/rest/customReport.csv?pdbids=*&reportName=Sequence&service=wsfile&format=csv*

en proteínas (Pagès y Grudinin, 2018). Es más, se trata de un uso alternativo de la red, pues esta fue diseñada para la detección de simetrías rotacionales en la estructura de una proteína.

En suma, el desarrollo de una herramienta para automatizar la detección y clasificación de proteínas repetidas representa un hito muy importante para la investigación en el dominio. Dicha herramienta debería explotar la información de estructura de la proteína ya que se ha visto que los métodos basados en secuencia tienen limitantes muy grandes. No obstante, ésta deberá usarse de una manera más eficiente que en un alineamiento de estructuras porque consume muchos recursos computacionales. El uso de un clasificador resulta, en este caso, una alternativa con mucho potencial porque el mayor consumo de recursos se da durante su etapa de entrenamiento (se realiza una vez) y no durante la inferencia. Luego de una revisión del estado del arte, no se ha encontrado una herramienta de tales características para proteínas repetidas; y, en particular, para aquellas de estructura cerrada (clase IV). En este proyecto se buscó, pues, cubrir este vacío para facilitar la investigación en el campo.

1.2. Objetivos

A continuación, se enuncia los objetivos del presente proyecto de investigación.

1.2.1. Objetivo general

Implementar una herramienta de identificación y clasificación automática de repeticiones en estructuras de proteínas de clase IV.

1.2.2. Objetivos específicos

1. Recolectar información estructural de unidades y regiones de repetición identificadas y clasificadas previamente en RepeatsDB.
2. Diseñar, implementar y evaluar un modelo de identificación y clasificación de repeticiones en la estructura de una proteína repetida.
3. Implementar un servicio web para la identificación y clasificación automática de repeticiones en estructuras de proteínas cerradas.

1.2.3. Resultados esperados

Los resultados esperados se encuentran enumerados en la tabla 1.1, la cual se muestra a continuación.

Tabla 1.1

Resultados esperados por objetivo específico.

ID Objetivo Especifico	ID Resultado	Resultado esperado (R.E.)
1	1.1	Librería de estructuras de proteínas del PDB.
	1.2	Librería de estructuras de unidades y regiones de repetición.
2	2.1	Análisis comparativo para selección y adaptación del modelo y su arquitectura.
	2.2	Conjunto de datos procesados para la fase de entrenamiento y prueba de clasificadores.
	2.3	Modelo de clasificación entrenado sobre el conjunto de datos de entrenamiento y probado en el conjunto de datos de prueba.
3	3.1	Diseño de un servicio web que muestra si una proteína presenta una estructura cerrada y su clasificación.
	3.2	Servicio web implementado e integrado con el modelo y disponible para uso.

Asimismo, cada resultado esperado fue asociado con un medio de validación. La tabla 1.2 especifica cada uno de ellos.

Tabla 1.2.

Mecanismos de verificación para los resultados esperados.

R.E. asociado	Medio de validación
1.1	Reporte que describe las estadísticas de la librería (número de proteínas extraídas, cantidad de cadenas).
1.2	Reporte que describe estadísticas de la librería (número de regiones recopiladas, número de regiones sin inserciones).
2.1	Reporte del análisis comparativo.
2.2	Descripción del conjunto de datos en general (número de observaciones, proporción entre las clases).
2.3	Documento que describe el entrenamiento del modelo (curva de aprendizaje) y el desempeño del modelo sobre el conjunto de pruebas (exactitud).
3.1	Resultados de pruebas funcionales de la herramienta implementada.

1.3. Metodología y Herramientas

En esta sección se presentarán las herramientas que se utilizaron para completar cada objetivo específico del proyecto. Asimismo, se enumerará y explicará la aplicación de metodologías para la gestión del proyecto y del producto.

1.3.1. Herramientas

Las herramientas necesarias para la ejecución del presente proyecto se enumeran en la tabla 1.3. Cabe mencionar que el siguiente listado tiene solo una naturaleza enumerativa. De hecho, se están nombrando únicamente las librerías principales.

Tabla 1.3.

Herramientas de software necesarias para la ejecución del proyecto.

O.E.	Herramienta (*)	Descripción
1	Python	Lenguaje de programación usado para la programación de scripts para varias actividades.
	Selenium	Librería de Python para la automatización de extracción de información de la web.
	Beautiful Soup	Librería de Python para el procesamiento de información de web.
	Pypdb	Librería de Python que actúa como interfaz con el banco de datos de proteínas. Facilita la extracción de información de dicho portal.
	ProDy	Librería de Python que realiza cálculos de dinámica molecular. También provee herramientas que permiten extraer sub-cadenas a partir de un archivo en formato PDB.
2	ProDy	Librería de Python. Realiza cálculos de distancia interatómica y del modelo anisotrópico.
	Sklearn	Librería de Python para el aprendizaje automático. Provee herramientas útiles para la clasificación.
	Keras	Librería de Python para aprendizaje profundo. Permite implementar la red neuronal.
	SPSS	Paquete estadístico de IBM que provee de utilidades para el análisis de datos de la experimentación.
	MySQL 8.0.1	Motor de base de datos. Le da persistencia a los datos extraídos y procesados.
3	Flask	Framework de Python que facilita la producción de servicios web.
	OS	Librería de Python para la gestión de elementos del sistema operativo.
	Java	Lenguaje de programación usado para la elaboración del *front-end*.

Las herramientas de software anteriormente mencionadas fueron ejecutadas en una PC con procesador i7 (Intel 9700k), 32Gb de memoria RAM y una GPU de NVidia RTX 2060 de 6Gb. Otro entorno de ejecución fue aquel provisto por Google Colaboratory. Las actividades realizadas en dicho ambiente serán indicadas en sus respectivas secciones.

1.3.2. Metodología de gestión del proyecto

Para la gestión del proyecto se tomaron los elementos del PMBOK que se consideraron más relevantes y aplicables al proyecto.

A. Alcance

El proyecto consiste en el diseño e implementación de un servicio web que permita la detección y clasificación de repeticiones cerradas (clase IV) en estructuras de proteínas. Por tanto, el alcance del proyecto cubre las actividades necesarias para el diseño e implementación de los dos frentes de la aplicación: la interfaz gráfica (*frontend)* y la capa lógica (*backend)*. Los entregables incluidos en el alcance del proyecto se encuentran identificados en la estructura de desglose de trabajo (EDT), la cual se ilustra en la figura 1.1.

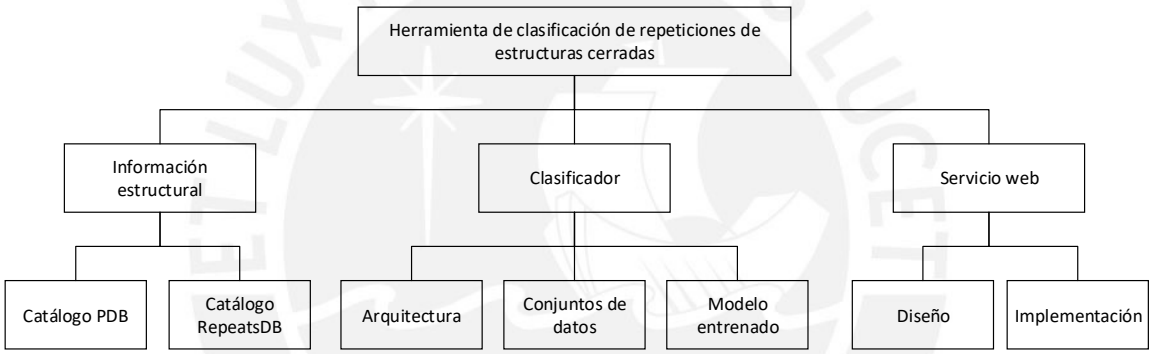


Figura 1.1. Estructura de desglose de trabajo (EDT) del proyecto.

El diccionario asociado al EDT mostrado anteriormente se resume en la tabla 1.4. Puede verse los detalles adicionales del mismo en el anexo 1 del documento.

Tabla 1.4.

Diccionario de la estructura de desglose de trabajo.

Id	Entregable	Descripción	Pred.
1	***Información estructural***	***Recopilación de información de estructuras de proteínas***	***1.2, 1.3***
1.1	Catálogo PDB	Reporte de estructuras extraídas de PDB.	No tiene
1.2	Catálogo RepeatsDB	Reporte de información estructural de regiones anotadas de RepeatsDB.	No tiene
2	***Clasificador***	***Back-end de la herramienta para la detección y clasificación de la región.***	***2.2, 2.3***
2.1	Arquitectura	Definición de la arquitectura del modelo de clasificación.	No tiene

Id	Entregable	Descripción	Pred.
2.2	Conjunto de datos	Procesamiento de información estructural para entrenar y evaluar modelo.	1.2, 1.3
2.3	Modelo entrenado	Entrenamiento del modelo.	2.2
3	***Servicio web***	***Front-end de la herramienta integrado con el clasificador.***	***2, 3.1***
3.1	Diseño	Diseño de interfaz del servicio.	No tiene
3.2	Implementación	Codificación e integración de la herramienta.	3.1

B. Tiempo

Se ha estimado que el proyecto sea realizado entre el 15 de abril del 2020 y el 22 de diciembre del 2020. La ruta crítica del proyecto indica que éste tiene una duración de 325 días. Los tiempos se han calculado mediante estimación análoga; es decir, el tiempo que se ha necesitado para cumplir tareas similares en el pasado. Para este cálculo se consideró una dedicación de 1 hora diaria para realizar las tareas del proyecto. El cronograma se resume en la tabla 1.5.

Tabla 1.5.

Cronograma del proyecto.

ID	Actividad	Dur. (días)	Inicio	Fin	Predecesor
1	**Información estructural de repeticiones**	**52**	**15/04/20**	**05/06/20**	**NA**
1.1.1	Obtener lista de proteínas disponibles en PDB	2	15/04/20	16/04/20	NA
1.1.2	Implementar script para extraer archivos PDB	4	17/04/20	20/04/20	NA
1.1.3	Ejecutar script (1.1.2) para extraer archivos PDB	10	21/04/20	30/04/20	1.1.1, 1.1.2
1.2.1	Implementar webscrapper de RepeatsDB	7	01/05/20	07/05/20	NA
1.2.2	Ejecutar script para extraer información de regiones en RepeatsDB.	13	08/05/20	20/05/20	1.2.1
1.0.1	Implementar script para extraer información estructural de regiones.	5	21/05/20	25/05/20	NA
1.0.2	Ejecutar el script (1.0.1).	11	26/05/20	05/06/20	1.1.2, 1.2.2
2	**Clasificador**	**205**	**06/06/20**	**22/12/20**	**1**
2.1.1	Identificar propuestas de arquitectura	30	06/06/20	06/07/20	NA
2.1.2	Realizar análisis comparativo entre propuestas.	10	07/07/20	16/07/20	2.1.1
2.2.1	Evaluar cadenas por Deep Symmetry (Base PDB).	101	17/07/20	25/10/20	NA
2.2.2	Evaluar cadenas por Deep Symmetry (Base RepeatsDB).	10	08/10/20	17/10/20	NA
2.2.3	Identificar umbral de detección.	7	18/10/20	24/10/20	2.2.2
2.2.4	Implementar script para convertir PDB a imagen.	6	25/10/20	31/10/20	NA

ID	Actividad	Dur. (días)	Inicio	Fin	Predecesor
2.2.5	Aplicar el script sobre cadenas con potencial de tener repetición.	15	01/11/20	15/11/20	2.2.4
2.3.1	Entrenar modelo	13	16/11/20	28/11/20	2.2
2.3.2	Optimizar los hiperparámetros del modelo	2	29/11/20	30/11/20	2.3.1
2.0.1	Integrar la salida de Deep Symmetry con el clasificador.	5	01/12/20	05/12/20	2.2, 2.3
2.0.2	Aplicar solución integrada sobre el conjunto de pruebas.	16	06/12/20	22/12/20	2.0.1
3	**Servicio Web**	**68**	**15/10/20**	**22/12/20**	**2**
3.1.1	Identificar los requisitos funcionales y no funcionales.	10	15/10/20	24/10/20	NA
3.1.2	Elaborar el prototipo del servicio web.	7	25/10/20	31/10/20	3.1.1
3.2.1	Implementar la interfaz gráfica	15	01/11/20	15/11/20	3.1.2
3.2.2	Realizar pruebas unitarias	5	10/11/20	15/11/20	3.1.1, 3.2.1
3.0.1	Integrar el clasificador con la interfaz gráfica.	7	06/12/20	13/12/20	2.0.1, 3.2.2
3.0.2	Realizar pruebas de integración.	3	14/12/20	16/12/20	3.0.1
3.0.3	Implantar solución en ambiente de producción.	5	17/12/20	22/12/20	3.0.2

C. Costo

El costo estimado del proyecto es de 3363.50 soles. El análisis de la estructura de costo consideró tres elementos: los costos directos, los costos indirectos y los costos de amortización. A continuación, se detallará el cálculo de cada uno de dichos elementos.

- **Costos directos:** se incluyó el costo de la mano de obra. Si bien se trató de un proyecto académico, se tuvo en consideración el costo de oportunidad; es decir, el pago no percibido en alguna actividad remunerativa por dedicar el tiempo en la investigación. Para este cálculo se tomó como referencia el sueldo mínimo vital del Perú en 2020: 930 soles mensuales (equivalen a 4.70 soles la hora para 22 días de trabajo con una jornada de 9 horas).
- **Costos indirectos:** se incluyó el costo del consumo de energía eléctrica. Se identificó dos tarifas para este servicio. La primera de ellas se aplicó durante los meses de consumo promedio. En este caso, se calculó el monto de consumo per cápita a partir del recibo del servicio de febrero 2020. Finalmente, se asignó el 50% de este valor al costo unitario por mes. La segunda tarifa correspondió a lo que se facturó por el consumo de energía eléctrica durante los meses en los que se hizo un consumo intensivo de electricidad para realizar las tareas del proyecto (por ejemplo, entrenar y aplicar las redes neuronales). En estos meses,

el ordenador estuvo encendido todo el día, lo cual se tradujo en un mayor consumo energético. Para estimar este costo se calculó la diferencia entre el monto facturado por un usuario promedio y un usuario intensivo (una persona que tiene encendido su ordenador la mayor parte del día).

- **Costos de amortización:** se consideró la depreciación del ordenador usado durante la ejecución del proyecto. Para este cálculo se hizo uso del método de línea recta y una tasa de depreciación del 24% anual (la SUNAT admite una tasa máxima de 50% anual para equipos de cómputo según el Decreto Legislativo 1488). El valor inicial del producto fue de 7200 soles.

La tabla 1.6. muestra el detalle de la estructura de costos del proyecto.

Tabla 1.6.

Estructura de costos del proyecto

Concepto	Unidad	P.Unit (S/.)	Cantidad	Total
Costos Directos				
Mano de obra (*)	Horas hombre	4.7	325	1527.5
Costos Indirectos (Gastos operacionales)				
Luz (no intensivo)	Mes de servicio	30	6	180
Luz (intensivo)	Mes de servicio	120	3	360
Costos de amortización (Gastos no operacionales)				
Desgaste de PC	Mes	144	9	1296
Total				**3363.5**

(*) Mano de obra calculada sobre una remuneración mínima vital del 2020.

D. Riesgos

Los riesgos identificados, así como su evaluación, mitigación y contingencia, se encuentran identificados en la matriz de riesgos en el anexo 2 de este documento. El resumen de dicha matriz se muestra en la tabla 1.7. Para evaluar el riesgo se usó la métrica de severidad, que se calculó como el producto del impacto y la probabilidad de ocurrencia del riesgo. El impacto se midió con una escala que va del 1 (bajo impacto) al 5 (alto impacto). De forma análoga, la probabilidad también se midió con una escala del 1 (baja probabilidad) al 5 (alta probabilidad).

Tabla 1.7.

Identificación y evaluación de riesgos. Mide impacto (I), probabilidad (P) y severidad (S).

Id	Descripción	I	P	S	Estrategia
1	Subestimación del alcance del proyecto.	5	2	10	Mitigar
2	Mala estimación de tiempos de entregables.	5	3	15	Mitigar
3	Problemas personales o de salud del tesista o asesor.	2	3	6	Aceptar
4	Curva de aprendizaje de herramientas muy alta.	2	3	6	Aceptar
5	Indisponibilidad de algunas herramientas	4	2	8	Evitar
6	Pérdida, avería o destrucción de ordenador donde se desarrolla el proyecto.	5	2	10	Mitigar
8	Caída de infraestructura Cloud	3	1	3	Mitigar
9	Desastre natural.	5	2	10	Aceptar

Como se puede observar en la tabla, los riesgos que tienen mayor impacto sobre el proyecto también tienen una probabilidad baja de ocurrencia. Esto indica que el nivel de riesgo del proyecto es bajo. Asimismo, para cada riesgo se definió una estrategia de mitigación (si aplica). Los detalles del plan de mitigación y contingencia se pueden encontrar en el anexo 2.

1.3.3. Metodología de gestión del producto

Como se mencionó en los acápites anteriores, el producto final del presente trabajo tiene dos componentes: uno lógico (*backend*) y uno gráfico (*frontend*).

Para la construcción del primer componente se adaptó un modelo de desarrollo muy usado, conocido como CRISP-DM. De acuerdo con el mismo, el proceso para el desarrollo de una solución debe pasar por una sucesión de etapas: entendimiento del negocio, entendimiento de los datos, preparación de los datos, modelado, evaluación y despliegue (Provost y Fawcett, 2013). A continuación, se explica cómo se ajusta cada etapa del proceso al presente trabajo:

- **Entendimiento del negocio:** en este caso, resulta conveniente renombrar el paso como entendimiento del dominio. Consiste en entender los conceptos del dominio de investigación (Provost y Fawcett, 2013); en este caso, las proteínas repetidas. Estos mismos están explicados a detalle en el marco teórico (capítulo 2).

- **Entendimiento de los datos:** consiste en recopilar y analizar los datos que se tienen disponibles (información estructural de PDB y de RepeatsDB) para luego definir cómo usarlos para resolver el problema que se está abordando.
- **Preparación de los datos:** consiste en transformar los datos disponibles a una forma que puedan ser utilizados por las redes neuronales.
- **Modelado:** consiste en aplicar las técnicas de aprendizaje automático sobre los datos transformados (Provost y Fawcett, 2013).
- **Evaluación:** implica comprobar que los resultados obtenidos con el producto del modelado sean correctos y confiables (Provost y Fawcett, 2013). Con frecuencia, implica también usar los resultados de la evaluación para identificar las debilidades del modelo y mejorarlas.

El modelo evolutivo de desarrollo de software se acomoda perfectamente a las necesidades para gestionar el desarrollo de la interfaz gráfica y su integración con la capa lógica. En este modelo se entrelazan las actividades de especificación, desarrollo y validación (Sommerville, 2002). Consiste en desarrollar rápidamente una primera versión del sistema en función de especificaciones abstractas (Sommerville, 2002). Luego, esta versión es refinada con las sucesivas retroalimentaciones que se reciba de los interesados hasta que se satisfaga la necesidad (Sommerville, 2002). Debido a que el producto software a desarrollar no es muy grande y tampoco hay muchos participantes en el proyecto, este modelo de desarrollo resulta muy conveniente para un trabajo rápido.

1.4. Alcance y Limitaciones

El objetivo del presente proyecto es desarrollar una herramienta que permita detectar y clasificar repeticiones a partir de la información estructural de una proteína de una forma más eficiente que otras ya existentes y reportadas en el estado del arte (capítulo 3). Como se verá en el capítulo 2, las repeticiones pueden ser de 5 clases. En esta investigación se trabajará con repeticiones de estructura cerrada (clase IV) Para ello, se hará uso de técnicas de aprendizaje automático; en particular, redes neuronales.

A partir de lo mencionado en el párrafo anterior, queda claro que la herramienta necesita de dos módulos: uno que funcione como filtro y otro que se encargue de la clasificación. Para el primer módulo, el alcance contempla la adaptación y uso de alguna herramienta existente. Para el segundo módulo, se incluye en el alcance: la extracción y procesamiento de datos, el entrenamiento de una red neuronal y su evaluación. Como

ya se mencionó en el primer párrafo, el proyecto está enfocado en la clasificación de las repeticiones de clase IV. Las otras clases de repeticiones están fuera del alcance de la investigación.

El proyecto también contempla la implementación de un servicio web que permita recibir como entrada el identificador de la proteína y devuelva como salida dos datos: si pertenece o no a la clase IV y la subclase a la que pertenece. Por consiguiente, el proyecto incluye en su alcance las actividades necesarias para el diseño, implementación e integración del servicio web. El alcance no contempla, sin embargo, la implantación de la herramienta en un entorno de producción permanente ni su posterior mantenimiento, pues no se cuenta con una inversión para ello.

1.5. Justificación

Este proyecto está orientado al desarrollo de una herramienta que permita identificar y clasificar proteínas repetidas de clase IV en función de su información estructural. En los últimos años se ha venido desarrollando soluciones que usan la estructura de una proteína para clasificarla. Sin embargo, muchas de ellas se basan en estrategias que son computacionalmente costosas ya que deben tener en cuenta la naturaleza multidimensional de los datos (Pellegrini, 2015) y, por tanto, lentas. Es más, no se ha encontrado herramienta conocida que permita realizar las tareas enunciadas para repeticiones de clase IV. Como consecuencia, la anotación manual sigue siendo una alternativa usada. Sin embargo, esta resulta inviable cuando se lleva a escalas mayores. Por ejemplo, anotar el Banco de Datos de Proteínas (en adelante, PDB) completo de forma manual implicaría un esfuerzo demasiado grande como para ser factible (la base de datos posee información de más 160 mil proteínas distintas, distribuidas en 570 mil cadenas).

Para el desarrollo de esta herramienta se plantea el uso de redes neuronales, las cuales han dado excelentes resultados en varios dominios (Basheer y Hajmeer, 2000). Es debido a este éxito que se considera plausible su uso en este caso. Adicionalmente, a diferencia de otras estrategias, su uso implica un costo computacional alto solo durante su entrenamiento (que se realiza solo una vez). Por tanto, se espera una mejor velocidad de respuesta que herramientas análogas.

Con la disponibilidad de la herramienta se facilitaría la clasificación de estructuras de proteínas repetidas. Con ello, será posible enriquecer más la base de proteínas

repetidas y, a la par, nuestro entendimiento de ellas. Este conocimiento es el punto de partida para el desarrollo de tratamiento de patologías en donde éstas participan (Paladin et al., 2017). Por ejemplo, la enfermedad de Huntington y la artritis reumatoide (Jorda et al., 2010).

1.6. Viabilidad

Luego de llevar a cabo el análisis de alcance, tiempo, costo y riesgos, es posible realizar el análisis de factibilidad del proyecto para determinar si es que es que se debería iniciar su ejecución o no.

El análisis del alcance del proyecto muestra que el trabajo tiene un nivel de complejidad adecuado, ya que implica actividades necesarias para extraer datos de fuentes de información distintas; analizar y transformar los datos; usarlos para entrenar un clasificador; y, finalmente, desarrollar una interfaz e integrarla con el clasificador. Este análisis se ve reflejado en la estructura de desglose de trabajo (EDT).

El análisis de la ruta crítica de las actividades del proyecto muestra que es factible culminarlo en 325 días, un tiempo estándar para una investigación de esta naturaleza. En este tiempo se espera completar las actividades necesarias para cumplir el objetivo del proyecto. El detalle de las actividades puede revisarse en el cronograma del proyecto.

Por otra parte, se estima que el proyecto tiene un costo de 3660 soles. Cabe mencionar que parte importante de este costo está compuesto por la mano de obra, que en realidad no recibe remuneración; es decir, no existe un desembolso de dinero por dicho elemento de la estructura de costos. En general, se trata de un monto bajo, si se compara con los fondos que se otorgan a proyectos similares.

Finalmente, el análisis de riesgos muestra que el riesgo asociado al proyecto es bajo y manejable en cuanto es posible mitigar la mayoría de estos. Adicionalmente la matriz de riesgos muestra que aquellos de mayor impacto tienen una baja probabilidad de ocurrencia. Por tal motivo, es posible afirmar que los riesgos del proyecto son posibles de controlar. Por todo lo anteriormente sustentado, se considera sensato afirmar que el proyecto es viable.

1.7. Conclusión

Las proteínas repetidas son aquellas que presentan repeticiones a nivel de secuencia y estructura. Tienen una arquitectura modular y pueden pertenecer a una de cinco clases. Cumplen una variedad de funciones; sin embargo, también están asociadas a muchas enfermedades. Su estudio es importante para comprender cuál es su mecanismo de participación en estas dolencias. La clasificación es parte trascendental de este estudio.

En el contexto anteriormente descrito, si bien la anotación manual es posible, no es una alternativa factible si se quiere llevar a volúmenes de información como los que maneja el PDB (cientos de miles de estructuras). Por ello, una herramienta que automatice parte de este proceso es necesaria. En particular, es necesaria una que explote la información estructural de la molécula de una forma computacionalmente eficiente. No se ha encontrado en la literatura una herramienta de tales características.

Debido a lo anterior, en este escrito se describirá la implementación de un servicio web que permita la clasificación de repeticiones usando la información de estructura. Debido a que el universo de repeticiones es grande, se hará un enfoque en aquellas de estructura cerrada (clase IV). Para el funcionamiento de esta, se entrenará una arquitectura de clasificadores que permita detectar repeticiones y luego clasificarlas. Luego del análisis del plan de proyecto, se concluyó que este era factible a nivel de alcance, tiempo, costo y riesgos.

Capítulo 2.

Marco teórico

En este capítulo se presentarán los términos y conceptos básicos involucrados en el proyecto. En la primera parte del capítulo se introducirán las proteínas y sus propiedades más importantes. En la segunda parte, se presentarán las proteínas repetidas. En la tercera parte se llevará a cabo una introducción la simetría molecular. La cuarta parte está enfocada en la teoría necesaria para el cálculo de energías no covalentes. La quinta parte introducirá el análisis de modos normales y el modelo anisotrópico. En la sexta y última parte del capítulo se definirán conceptos relacionados con el aprendizaje automático.

2.1. Aminoácidos, péptidos y proteínas

Las proteínas son moléculas orgánicas que desempeñan una variedad de roles biológicos (McMurry, 2010). Estrictamente, son polímeros de α-aminoácidos (aminoácidos en lo sucesivo), caracterizados por la presencia de un grupo ácido (COOH) y un grupo amino (NH2) en su estructura (McMurry, 2010). La secuencia de estas unidades básicas determina las propiedades físicas y químicas de la proteína resultante (Wade, Montaña Pedrero y Batalla García, 2004).

2.1.1. Aminoácidos

Los aminoácidos son los elementos que constituyen las proteínas (McMurry, 2010). Cuentan con un átomo de carbono al que están unidos dos grupos funcionales distintos (un grupo carboxilo -COOH y un grupo amino -NH2), un átomo de hidrógeno y otra cadena que le da identidad a la molécula y que recibe el nombre de cadena lateral (McMurry, 2010). Dicho átomo de carbono recibe el nombre de carbono-α (Wade et al., 2004).

Hay veinte aminoácidos que se encuentran en prácticamente todas las proteínas (Wade et al., 2004). Como ya se mencionó, la identidad de cada aminoácido es determinada por su cadena lateral (Wade et al., 2004). Cada aminoácido puede ser representado -según necesidad- con un símbolo de 1 letra o una abreviatura de 3 letras (Wade et al.,

2004). La tabla 2.1. enumera los 20 aminoácidos esenciales, sus símbolos respectivos y el grupo funcional de sus cadenas laterales.

Tabla 2.1.

Aminoácidos: identidad, símbolos y abreviaturas. Adaptado de (Wade et al., 2004).

Nombre	Simb.	Abrev.	Grupo funcional de cadena lateral
Glicina	G	GLY	Ninguno
Alanina	A	ALA	Alquilo (apolar)
Valina	V	VAL	Alquilo (apolar)
Leucina	L	LEU	Alquilo (apolar)
Isoleucina	I	ILE	Alquilo (apolar)
Fenilalanina	F	PHE	Aromático (apolar)
Prolina	P	PRO	Ciclo rígido (apolar)
Serina	S	SER	Hidroxilo
Treonina	T	THR	Hidroxilo
Tirosina	Y	TYR	Fenol
Cisteína	C	CYS	Tiol
Metionina	M	MET	Sulfuro
Asparagina	N	ASN	Amida
Glutamina	Q	GLN	Amida
Triptófano	W	TRP	Indol
Ác. Aspártico	D	ASP	Carboxilo
Àc. Glutámico	E	GLU	Carboxilo
Lisina	K	LYS	Amino
Arginina	R	ARG	Guanidino
Histidina	H	HIS	Anillo imidazol

2.1.2. Enlaces covalentes entre aminoácidos

La reacción más importante entre dos aminoácidos es la formación de un enlace peptídico (Wade et al., 2004). Aminas y ácidos pueden condensarse en amidas a través de una reacción de deshidratación (Wade et al., 2004). Como se sabe, un aminoácido tiene ambos grupos en su estructura, por lo que puede condensarse con otro para dar lugar a un péptido (Wade et al., 2004). La figura 2.1 muestra que la reacción de condensación forma un equilibrio químico en el cual se favorece el sentido de ruptura de enlace (Berg, Tymoczko y Stryer, 2008). Por este motivo, se requiere de un aporte energético para formar el enlace, el cual se vuelve muy estable cinéticamente una vez

formado (Berg et al., 2008). Esto quiere decir que la velocidad de la reacción de ruptura es muy lenta. En este caso, es cercano a los 1000 años (Berg et al., 2008). Esta estabilidad se debe a que en el enlace ocurre lo que se denomina resonancia o deslocalización de electrones (Wade et al., 2004).

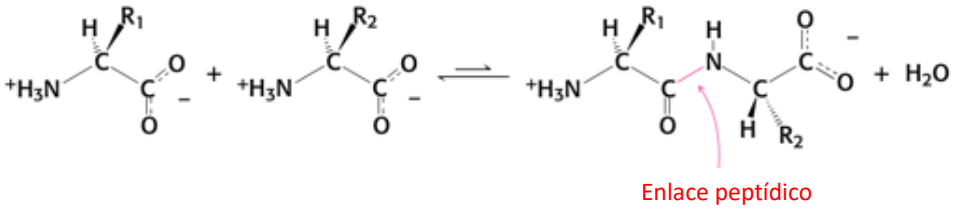


Figura. 2.1. Formación de enlace peptídico. Adaptado de: (Berg et al., 2008).

2.1.3. Polipéptidos y proteínas

Las proteínas son las macromoléculas biológicas con mayor abundancia en la naturaleza (Wade et al., 2004). Adicionalmente, cumplen varias funciones biológicas: estructura, catálisis enzimática, expresión de la información genética, entre otras (Lehninger, Nelson y Cox, 2005). Ellas se forman a partir de la unión de dos a más aminoácidos mediante un enlace peptídico (McMurry, 2010). De forma genérica, la unión de dos a más aminoácidos recibe el nombre péptido o polipéptido (Wade et al., 2004). Particularmente, la unión de 2 aminoácidos es denominada dipéptido (Wade et al., 2004). Solo si el peso molecular del polímero supera las 5000uma se le da la denominación de proteína (Wade et al., 2004).

Toda proteína presenta la secuencia repetitiva -N-CH-CO-, la misma que recibe el nombre de cadena principal o *backbone* (McMurry, 2010). Otros dos elementos importantes de reconocer son los dos extremos de la secuencia: un extremo contiene el grupo amino libre y recibe el nombre *N-terminal*; el otro está formado por el grupo carboxilo libre y se le denomina *C-terminal* (Wade et al., 2004). Es importante reconocer cada extremo ya que por convención la secuencia de aminoácidos se especifica partiendo del extremo *N-terminal* hacia el *C-terminal* (Wade et al., 2004).

2.1.4. Caracterización de proteínas

Para caracterizar completamente una proteína, se suele utilizar 4 niveles de estructura: primaria, secundaria, terciaria y cuaternaria (McMurry, 2010). En general, la estructura primaria expresa la secuencia de aminoácidos que la componen (McMurry, 2010). La estructura secundaria describe cómo segmentos de la cadena se orientan en patrones

conocidos (McMurry, 2010). La estructura terciaria indica el doblamiento tridimensional de la proteína (McMurry, 2010). Finalmente, la estructura cuaternaria expresa la forma en la que distintas proteínas interactúan para formar estructuras complejas (McMurry, 2010).

A. Estructura primaria

Como se mencionó anteriormente, la estructura primaria especifica la secuencia de aminoácidos que componen la proteína (McMurry, 2010). Para representarla, se listan los aminoácidos que la componen de forma ordenada desde el extremo N-terminal hasta el C-terminal (Berg et al., 2008; Wade et al., 2004). Esta secuencia es muy importante en los organismos vivos y está especificada por los genes a través de procesos complejos de transcripción y traducción (Berg et al., 2008). Un error en este proceso puede dar origen a una enfermedad (Berg et al., 2008).

B. Estructura secundaria

La estructura secundaria describe la forma en que partes de la cadena se pliegan en estructuras regulares (Berg et al., 2008). Esto es posible gracias a que si bien el enlace peptídico es rígido, los otros enlaces que componen la cadena principal sí son flexibles (Berg et al., 2008). Si bien en un inicio se reconocía únicamente las hélices-α y las hojas plegadas-β, actualmente se reconoce también el giro-β y el bucle-ω, aunque no son tan frecuentes como las dos primeras (Berg et al., 2008).

La hélice-α es una estructura, como su nombre indica, helicoidal. Esta hélice puede tener un giro en sentido de las manecillas del reloj -dextrógiro- o contrario a ellas -levógiro (Berg et al., 2008). La figura 2.2 muestra un ejemplo de este pliegue. En esta figura también se puede observar que existen interacciones no enlazantes (líneas punteadas) entre residuos alejados en la hélice (Berg et al., 2008). Estas interacciones tienen un efecto de estabilización (Berg et al., 2008).

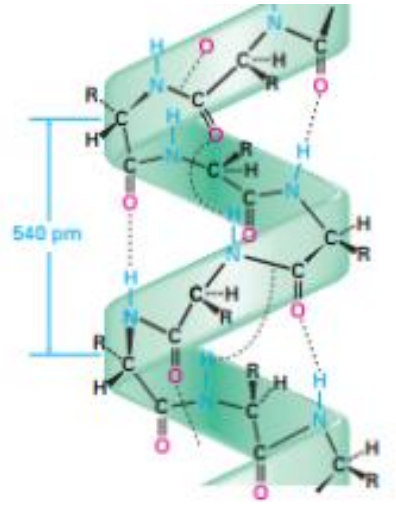


Figura. 2.2. Fragmento de una hélice-α. Tomada de: (McMurry, 2010).

El segundo motivo estructural importante dentro de las estructuras secundarias es la hoja plegada β (McMurry, 2010; Wade et al., 2004). Se trata de una estructura muy diferente a la hélice-α. Están formadas por dos o más cadenas polipeptídicas (hebras-β) que están completamente extendidas y tienen interacciones no enlazantes con las hebras vecinas (Berg et al., 2008). Estas hebras pueden ir en el mismo sentido -paralelas- o en sentido opuesto -antiparalelas- (Berg et al., 2008). La imagen 2.3 muestra un ejemplo de este arreglamiento.

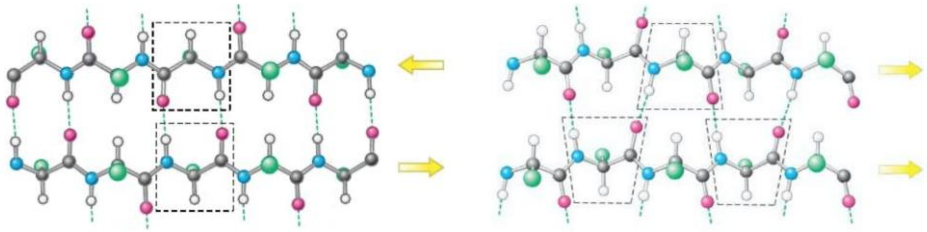


Figura 2.3. Fragmento de dos hebras-β. Las flechas amarillas indican el sentido de la hebra. Las líneas punteadas muestran las interacciones no enlazantes entre ellas. **Izquierda:** Conformación antiparalela. **Derecha:** Conformación paralela. Tomada de: (Berg et al., 2008).

En la mayor parte de proteínas con formas compactas y globulares se requieren cambios en la dirección de las cadenas (Berg et al., 2008). Esto ya ha sido apreciado en el caso de las hojas plegadas β. Los giro-β (o giro inverso) y los bucles-ω son las estructuras que permiten llevar a cabo dicho cambio (Berg et al., 2008; Lehninger et al., 2005).

C. Estructura terciaria

La estructura terciaria examina cómo los aminoácidos se agrupan en una proteína completa (Berg et al., 2008). A este nivel, es de utilidad clasificar las proteínas en dos grupos: globulares y fibrosas (Lehninger et al., 2005; McMurry, 2010). Como indican sus nombres, las proteínas globulares son compactas y esféricas; por el contrario, las fibrosas son largas (Lehninger et al., 2005). Como es de esperarse, las propiedades de ambos grupos son muy diferentes.

Las proteínas fibrosas suelen estar formadas por la repetición de un elemento simple de estructura secundaria (Lehninger et al., 2005). Todas son insolubles en agua debido a la proliferación de residuos hidrofóbicos tanto en su superficie como en su interior (Lehninger et al., 2005). Suelen tener funciones estructurales o de protección (Lehninger et al., 2005).

Las proteínas globulares, a diferencia de las fibrosas son más variadas. Pueden presentar más de un elemento de estructura secundaria (Lehninger et al., 2005). Es más, diferentes segmentos pueden plegarse unos sobre otros y dar origen a estructuras compactas (Lehninger et al., 2005). La figura 2.4 muestra algunas de ellas. Como consecuencia de esta variedad estructural existe también una variedad funcional (Lehninger et al., 2005). Las enzimas y reguladores pertenecen a esta clasificación (Lehninger et al., 2005; McMurry, 2010).

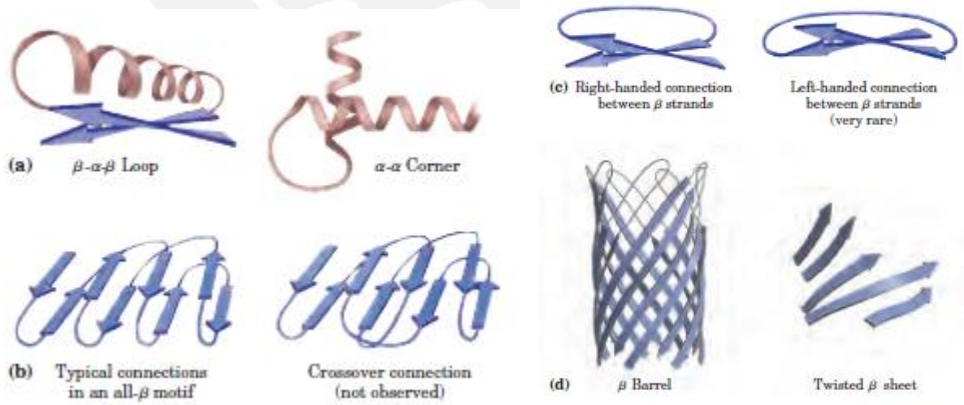


Figura 2.4. Ejemplos de pliegues de estructura terciaria. Tomada de: (Lehninger et al., 2005)

D. Estructura cuaternaria

Existen proteínas que constan de más de una cadena polipeptídica (Berg et al., 2008). En tales casos, cada una recibe el nombre de subunidad (Berg et al., 2008). La

estructura cuaternaria describe la organización de estas subunidades (Berg et al., 2008). La organización más simple es el *dímero* y consta de dos subunidades idénticas (Berg et al., 2008). Adicionalmente, cuando el compuesto está formado por más de dos subunidades, recibe el nombre genérico de *multímero* (Lehninger et al., 2005). Un ejemplo clásico de *multímero* lo constituye la hemoglobina (Lehninger et al., 2005).

2.2. Proteínas repetidas

2.2.1. Arquitectura modular

Las proteínas repetidas son aquellas en las que se puede encontrar una estructura tridimensional repetitiva (Hirsh et al., 2016). Han sido explotadas por la naturaleza en una multitud de procesos celulares (Hirsh et al., 2016). Como corolario, también se encuentran relacionadas con varias enfermedades (Paladin et al., 2017).

Una característica de estas proteínas es su organización modular: se reconoce un conjunto de estructuras (y secuencia) que funcionan como unidades de construcción (Hirsh et al., 2016; Paladin et al., 2017). Estos pequeños módulos reciben el nombre de unidades de repetición (Hirsh et al., 2016; Paladin et al., 2017).

La unión de al menos 3 de dichas unidades conforman lo que se conoce como una región de repetición (Paladin et al., 2017). El valor umbral definido para las regiones de repetición fue escogido para que no se confunda con las repeticiones de dos residuos, frecuentes en las proteínas globulares (Di Domenico et al., 2014). El tamaño de estas regiones es variado: puede ir desde los 5 hasta los 60 (o más) residuos (Turjanski et al., 2016).

Por motivos evolutivos, algunas repeticiones presentan segmentos nos repetidos en sus unidades de repetición o entre dos de ellas (Di Domenico et al., 2014). Dichos segmentos no repetidos reciben el nombre de inserciones y son importantes porque complican las tareas de detección al romper las simetrías de las repeticiones (Di Domenico et al., 2014).

2.2.2. Clasificación de repeticiones

Las repeticiones pueden ser clasificadas en 5 clases en función de la longitud de su secuencia y arreglo estructural (Di Domenico et al., 2014). Cada clase, a su vez, puede

ser subclasificada en función de las estructuras secundarias de la repetición (Paladin et al., 2017). La tabla 2.2 enumera y describe cada una de las 5 clases (y subclases de cada una).

Tabla 2.2.

Clasificación de unidades de repetición en función de su estructura tridimensional. Adaptado de: (Di Domenico et al., 2014; Paladin et al., 2017)

Clase	Descripción(*)	Subclase	Descripción
I	Cristalitas (Corta)	I.1	Agregado insoluble
II	Estructuras fibrosas (Corta)	II.1	Triple hélice de colágeno
		II.2	Bobina helicoidal α
III	Estructuras elongadas (Intermedia)	III.1	Solenoide-β
		III.2.	Solenoide α/β
		III.3	Solenoide-α
		III.4	Trébol-β / Horquilla-β
		III.5	Capa antiparalela β / Horquilla-β
		III.6	Caja
IV	Estructuras cerradas (Intermedia)	IV.1	Barril TIM
		IV.2	Barril-b/Horquilla-β
		IV.3	Trébol-β
		IV.4	Propulsor-β
		IV.5	Prisma α/β
		IV.6	Barril-α
		IV.7	Barril α/β
		IV.8	Propulsor α/β
		IV.9	Trébol α/β
		IV.10	Prisma alineado
V	Estructura de perlas en cadena (Largas)	V.1	Perlas-α
		V.2	Perlas-β
		V.3	Perlas-α/β
		V.4	Perla sándwich β
		V.5	Perla sándwich α/β

(*) La descripción incluye la longitud de la repetición. Corta=5 residuos. Larga=60+ residuos.

2.3. Repositorios de información de proteínas

Con el transcurso de los años, se ha ido implementando varios repositorios de información de proteínas. Sin embargo, para el presente trabajo se utilizarán dos: el banco de datos de proteínas (PDB) y una base de datos de proteínas repetidas, RepeatsDB.

2.3.1. Banco de datos de proteínas (PDB)

El banco de datos de proteínas (PDB) es un repositorio información estructural de proteínas y otras macromoléculas biológicas importantes (Berman et al., 2000). Esta información estructural es obtenida a través de una diversidad de métodos como cristalografía de rayos X, espectroscopía de resonancia magnética nuclear (NMR), etc (Berman et al., 2000). Se trata de una base de datos que se encuentra en constante crecimiento, reflejo de la intensidad de investigación en distintas áreas (Berman et al., 2000).

Si bien fue creado en 1971 por el Laboratorio Nacional de Brookhaven, actualmente se encuentra bajo la gestión del Grupo de Investigación de Bioinformática Estructural (RCSB), cuya visión es convertirlo en una herramienta para el análisis de información estructural (Berman et al., 2000).

La información del banco de datos de proteínas puede ser descargada en forma de archivos de extensión *pdb*. Estos archivos tienen varios tipos de registros; pero los de interés para este trabajo son los registros de coordenadas atómicas (ATOM), cuya estructura se especifica en la tabla 2.3.

Tabla 2.3.

Especificación de los registros de coordenadas atómicas en archivo pdb. Adaptado de: (Berman et al., 2000)

Columnas	Campo	Definición
1-6	“ATOM “	Tipo de registro.
7-11	serial	# serie del átomo.
13-16	name	Nombre del átomo.
17	altLoc	Ind. Localización alternativa.
18-20	resName	Nombre del residuo (código de 3 letras).
22	chainID	Identificador de cadena.
23-26	resSeq	# serie del residuo.
27	iCode	Código para inserción de residuos.
31-38	x	Coordenada X del átomo (en Angstroms).
39-46	y	Coordenada Y del átomo (en Angstroms).
47-54	z	Coordenada Z del átomo (en Angstroms).
55-60	occupancy	Ocupancia.
61-66	tempFactor	Factor de temperatura.
77-78	element	Símbolo del elemento.
79-80	charge	Carga del átomo.

La forma más común para obtener los archivos *pdb* es por medio de la aplicación web (buscar la proteína por código y descargarla) o por descargas directas de los servidores

espejo (Berman et al., 2000). La información contenida en el repositorio puede ser accedida, adicionalmente, de manera programática mediante el uso de la API que se expone al público (Berman et al., 2000).

2.3.2. RepeatsDB

RepeatsDB es una base de datos construida a partir de PDB y contiene información relativa solo a proteínas repetidas (Paladin et al., 2017). En ella puede encontrarse, por ejemplo, la secuencia repetida, su inicio, su fin y su clasificación (Paladin et al., 2017). La data disponible ha sido curada manualmente en algunos casos; y en otros, mediante el uso de métodos computacionales (Paladin et al., 2017).

2.4. Simetría molecular

Algunos objetos son “más simétricos” que otros (Atkins y Cwi, 2008). Por ejemplo, se considera que una esfera es más simétrica que un cubo porque su disposición no varía al ser rotada en cualquier ángulo, a diferencia del cubo que solo permanece invariable al ser rotado en ciertos ángulos alrededor de ciertos ejes (Atkins y Cwi, 2008). Al igual que con objetos geométricos, puede definirse y compararse la simetría a nivel molecular. En este sentido, una molécula de amoniaco es más simétrica que el agua pues tiene más ángulos en los que puede ser rotada sin modificarse su disposición (Atkins y Cwi, 2008).

A partir de lo anterior, puede definirse una operación de simetría como una acción que no modifica la disposición del objeto original (Atkins y Cwi, 2008). Las operaciones de simetría típicas son la rotación, reflexión e inversión (Atkins y Cwi, 2008). Adicionalmente, cada operación de simetría tiene lo que llama elemento de simetría (Atkins y Cwi, 2008). Así, por ejemplo, el elemento de simetría en una operación de rotación es el eje alrededor del cual se lleva a cabo (Atkins y Cwi, 2008).

Dado que en esta investigación solo se trabajará con simetría rotacional, se profundizará un poco más en ella. Así, una rotación de orden n alrededor de un eje de simetría C_n es una rotación de 360°/n (Atkins y Cwi, 2008). Así, por ejemplo, el agua tiene un eje C_2 y un orden n=2 ya que debe girarse 180° para que no se modifique su disposición (Atkins y Cwi, 2008). Una molécula puede tener varios órdenes de simetría rotacional; y por tanto, varios ejes de simetría (Atkins y Cwi, 2008). El eje asociado a la rotación de mayor grado recibe el nombre de eje principal (Atkins y Cwi, 2008).

2.5. Interacciones moleculares

Las interacciones moleculares son las responsables de las propiedades observables de las diferentes sustancias de la naturaleza (Atkins y Cwi, 2008). Propiedades tan elementales como los estados de la materia son consecuencias directas de la fuerza de estas interacciones (Atkins y Cwi, 2008). Entre ellas se encuentran las interacciones iónicas (con cargas), las interacciones que involucran dipolos (dipolo-dipolo, dipolo-dipolo inducido, etc) y las de Van Der Waals, que dependen de las distancias (Atkins y Cwi, 2008).

En este trabajo se hará uso de las propiedades de las interacciones de Van Der Waals, por lo que se profundizará en ellas. Como ya se mencionó anteriormente, este tipo de interacciones es dependiente de las distancias entre las moléculas (Atkins y Cwi, 2008). La energía potencial asociada a estas interacciones puede ser calculada usando la ecuación del potencial de Lennard-Jones (ecuación 2.1) (Atkins y Cwi, 2008).

$$V = 4\varepsilon\left[\left(\frac{r_0}{r}\right)^{12} - \left(\frac{r_0}{r}\right)^{6}\right] \dots (ec.\,2.1)$$

En esta ecuación, los parámetros son dos: ε (profundidad del pozo) y r_0 (distancia donde la energía se hace cero) (Atkins y Cwi, 2008). La variable r es la distancia intermolecular (Atkins y Cwi, 2008). Otra forma de expresar el potencial entre dos moléculas heteronucleares es la ecuación 2.2 (Cornell et al., 1995).

$$V = \frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} \dots (ec.\,2.2)$$

En esta forma, R_{ij} se calcula según la ecuación 2.3 (Cornell et al., 1995).

$$R_{ij} = R_i^* + R_j^* \dots (ec.\,2.3)$$

Donde R_i^* y R_j^* son valores que se pueden encontrar en tablas (ver tabla 2.4). Por otro lado, para calcular A y B se emplean las ecuaciones 2.4 y 2.5 (Cornell et al., 1995).

$$A_{ij} = \varepsilon_{ij} * \left(R_{ij}^*\right)^{12} \dots (ec.\,2.4)$$

$$B_{ij} = 2 * \varepsilon_{ij} * \left(R_{ij}^*\right)^{6} \dots (ec.\,2.5)$$

Finalmente, el valor de ε_{ij} puede ser calculado usando la ecuación 2.6 (Cornell et al., 1995).

$$\varepsilon_{ij} = \left(\varepsilon_i * \varepsilon_j\right)^{\frac{1}{2}} .. (ec. 2.6)$$

Donde ε_i y ε_j pueden ser encontrados en tablas (ver tabla 2.4).

Tabla 2.4.

Parámetros de Van der Waals. Valores rescatados de: (Cornell et al., 1995).

Átomo	R* (A)	ε* (kcal/mol)
C	1.9080	0.0860
Hx (unido a N)	0.6000	0.0157
H (unido a C, prom.)	1.3679	0.0154
O	1.6612	0.2100
O-H	1.7210	0.2104
N	1.8240	0.1700
P	2.1000	0.2000
S	2.0000	0.2500
Cs	3.3950	0.0000806
F	1.7500	0.0610
K	2.6580	0.000328
Li	1.1370	0.0183
Rb	2.9560	0.00017

2.6. Análisis de modo normal

El análisis de modos normales (NMA, en adelante) es una técnica de simulación usada para predecir movimientos funcionales en una molécula biológica (Hayward y Groot, 2008). Un movimiento funcional es aquel relacionado con alguna función de la molécula (Hayward y Groot, 2008). Múltiples estudios han demostrado que los modos con más movimiento (menos frecuencia) suelen ser los más relevantes (Hayward y Groot, 2008). NMA involucra tres pasos:

1) **Minimización de energía potencial conformacional**, mediante el uso de la información de coordenadas (Hayward y Groot, 2008).
2) **Cálculo de la matriz Hessian**, que es la matriz de segundas derivadas de la energía potencial con respecto a las coordenadas atómicas ponderadas por la masa (Hayward y Groot, 2008).
3) **Diagonalización de la matriz**, produciendo un conjunto de valores y vectores propios (modos normales) (Hayward y Groot, 2008).

Cada uno de los anteriores pasos puede ser computacionalmente costoso dependiendo del tamaño de la molécula (Hayward y Groot, 2008). Como medio para reducir esta carga, se idearon los modelos de red elástica (Hayward y Groot, 2008). Estos modelos realizan un NMA, sin embargo, simplifican la molécula de forma tal que los cálculos sean menos demandantes (Hayward y Groot, 2008). En ellos se suele trabajar solo con los carbonos alfa de los residuos y se representa la molécula como si fuera un conjunto de resortes (Hayward y Groot, 2008). La figura 2.5 ilustra la simplificación realizada por estos modelos.

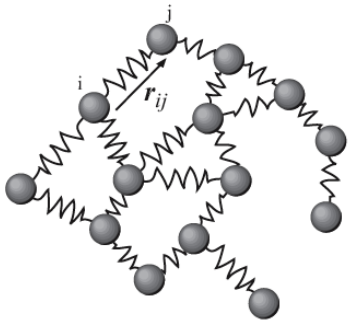


Figura 2.5. Esquema de la simplificación realizada por un modelo elástico. Tomada de: en.wikipedia.org/wiki/Anisotropic_Network_Model

En un modelo anisotrópico (ANM), que es un tipo de modelo elástico, todos los resortes tienen la misma constante de elasticidad, lo cual simplifica los cálculos en gran medida (Eyal, Yang y Bahar, 2006). Las ventajas del ANM sobre un NMA tradicional son dos: ANM es más rápido y permite detectar fluctuaciones a larga escala (Eyal et al., 2006). Sobre este último punto, la información de las fluctuaciones de los residuos y las correlaciones entre ellas está recogida en la llamada matriz de covarianza, la cual es proporcional a la inversa de la matriz Hessian (Eyal et al., 2006).

2.7. Aprendizaje automático

Para la resolución de un problema computacional, lo más común es que se requiera de un algoritmo, una secuencia finita de pasos que le indiquen al computador cómo transformar un conjunto de entradas en la salida deseada (Bishop, 2006). Sin embargo, para algunos problemas, el algoritmo para resolverlos no está disponible o no se puede aplicar (Bishop, 2006). En estos casos, es posible recurrir a datos históricos para que el computador pueda resolver el problema en función de ellos (Bishop, 2006). El aprendizaje automático consiste, pues, en programar al computador para que optimice

un criterio de desempeño en función de datos históricos (Bishop, 2006). Con este fin, se dispone de una función modelo que tiene un conjunto de parámetros que el computador va optimizando en base a ellos (Bishop, 2006).

2.7.1. Aprendizaje supervisado y no supervisado

En el contexto del aprendizaje de máquina se distinguen dos tipos principales de técnicas de aprendizaje: supervisado y no supervisado. Si bien en ambos se proporciona al modelo un conjunto de características de una observación en forma de un vector (Bird, Klein y Loper, 2009), hay una diferencia esencial entre ambos: en el aprendizaje supervisado se proporciona al computador la variable respuesta para cada observación (Bird et al., 2009). Por el contrario, en el aprendizaje no supervisado no se da tal información (Bishop, 2006). Esta diferencia explica el nombre que recibe cada técnica: en el caso del aprendizaje supervisado, existe un supervisor que le da la respuesta correcta al modelo (Bishop, 2006). En el caso del aprendizaje no supervisado, tal supervisor no existe; y, por tanto, no se dispone de la variable respuesta para las observaciones (Bishop, 2006).

Dos problemas clásicos que se resuelven por medio de aprendizaje supervisado son la clasificación y la regresión (Bishop, 2006). La clasificación es la tarea de asignar la etiqueta correcta a una entrada (Bird et al., 2009). Un ejemplo de clasificación es la definición si un correo con ciertas características es spam o no (Bird et al., 2009). La regresión es la tarea de predecir un valor en función de la entrada (Bishop, 2006). Un ejemplo de esta tarea es la predicción de precios de un carro en función de sus características (Bishop, 2006).

El aprendizaje no supervisado, por otro lado, busca estructura y patrones en la entrada del modelo (Bishop, 2006). Un problema típico que se resuelve con esta forma de aprendizaje es el agrupamiento (*clustering*), cuyo objetivo es encontrar agrupamientos entre los elementos de la entrada (Bishop, 2006).

2.7.2. Clasificación supervisada

Como ya se explicó anteriormente, la clasificación es la tarea de escoger la etiqueta correcta para una observación en función de sus características (Bird et al., 2009). En este contexto, cada observación es tratada de manera independiente de las otras y las etiquetas están ya definidas de antemano (Bird et al., 2009). En el caso particular en que el número de clases es dos, la tarea recibe el nombre de clasificación binaria

(Alpaydin, 2014). El esquema de trabajo para un problema de clasificación se encuentra ilustrado en la figura 2.6.

En la mencionada figura puede observarse la existencia de dos etapas: una de entrenamiento y otra de predicción (Bird et al., 2009). En la etapa de entrenamiento, un extractor de características captura las características de cada observación y las vectoriza (Bird et al., 2009). Después, estos vectores acompañados por sus respectivas etiquetas, son usados como entradas por el algoritmo de aprendizaje de máquina para optimizar los parámetros del modelo subyacente (Bird et al., 2009).

En la etapa de predicción, se usa el mismo extractor para capturar las características de la observación nueva (Bird et al., 2009). Luego, éstas son enviadas al clasificador entrenado el cual les asigna la etiqueta correspondiente (Bird et al., 2009).

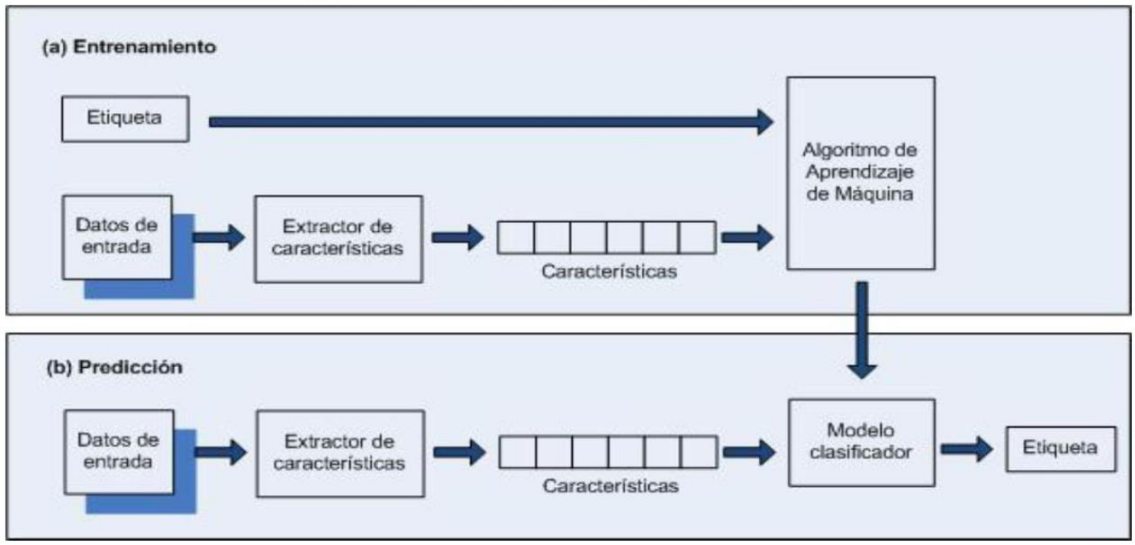


Figura 2.6. Esquema del marco de trabajo para una clasificación supervisada. Adaptado de (Bird et al., 2009) por (Vargas, 2013).

2.7.3. Evaluación del desempeño del aprendizaje

Para poder determinar si un modelo de clasificación ha aprendido correctamente, resulta necesario poder medir su desempeño (Bird et al., 2009). La evaluación es la herramienta que permite determinar la confiabilidad de un modelo e identificar posibles mejoras sobre el mismo (Bird et al., 2009).

2.7.3.1. Métricas de evaluación del aprendizaje de un clasificador

Para la clasificación se han definido una variedad de métricas para medir el desempeño del clasificador (Alpaydin, 2014). Para la clasificación binaria se definen los siguientes conceptos:

1. **Verdadero positivo (TP):** es una observación que el clasificador predice como positiva y realmente es positiva (Alpaydin, 2014).
2. **Falso positivo (FP):** se trata de un registro que el clasificador marca como positivo y realmente pertenece a la clase negativa (Alpaydin, 2014).
3. **Verdadero negativo (TN):** es una observación que el clasificador identifica como miembro de la clase negativa y realmente pertenece a dicha clase (Alpaydin, 2014).
4. **Falso negativo (FN):** es una observación etiquetada como negativa pero que realmente pertenece a la clase positiva (Alpaydin, 2014).

Como puede observarse, estas definiciones fueron hechas en base a un problema de clasificación binaria. Sin embargo, pueden adaptarse fácilmente para un problema de clasificación multiclase. Existe una gran variedad de métricas que pueden ser usadas para medir el desempeño de un clasificador. Algunas de las más usadas son la exactitud, la precisión y la sensibilidad. A continuación, se describirá cada una de ellas.

1. **Exactitud (*accuracy*):** es la métrica más básica que se puede usar (Bird et al., 2009). Indica cuántas observaciones fueron clasificadas correctamente (Bird et al., 2009). Se calcula según la ecuación 2.7.

$$Exactitud\ (\%) = \frac{TP + TN}{TP + TN + FP + FN} \dots (2.7)$$

2. **Precisión (*precisión*):** indica qué fracción de los ítems identificados como interesantes son realmente de interés (Alpaydin, 2014; Bird et al., 2009). Está definida por la ecuación 2.8.

$$Precisión\ (\%) = \frac{TP}{TP + FP} \dots (2.8)$$

3. **Memoria/sensibilidad (*recall*):** se trata de una métrica que indica el porcentaje de ítems de interés que fueron correctamente identificados (Alpaydin, 2014; Bird et al., 2009). Se calcula siguiendo la fórmula 2.9.

$$Memoria\ (\%) = \frac{TP}{TP + FN} \dots (2.9)$$

2.7.3.2. Conjunto de entrenamiento y prueba

La mayoría de métricas de evaluación de desempeño se basan en comparar las etiquetas que el modelo induce sobre un conjunto de prueba contra sus etiquetas reales (Bird et al., 2009). Este conjunto de prueba debe ser separado del conjunto de datos inicial, de manera que se asegure que no sea usado para el entrenamiento (Bird et al., 2009). Esta condición es muy importante ya que de lo contrario se podrían obtener resultados más altos de lo real porque el modelo memorizó las entradas (Bird et al., 2009).

El tamaño del conjunto de prueba también es importante: debe ser lo suficientemente grande como para ser representativo de la población; sin embargo, no puede ser tan grande que se termine con pocos datos para entrenar (Bird et al., 2009). En la práctica, suele reservarse entre el 20% y 30% de los datos para este conjunto.

2.7.3.3. Validación cruzada

Para la evaluación de un modelo es necesario separar una porción del conjunto de datos para las pruebas (Bird et al., 2009). Como ya se explicó en la sección anterior, este conjunto de pruebas no puede ser muy pequeño porque dejaría de ser representativo; sin embargo, tampoco puede ser muy grande ya que el modelo se quedaría con pocos datos para entrenar (Bird et al., 2009). Para superar dicha dicotomía, una alternativa es la validación cruzada (Bird et al., 2009).

Esta técnica consiste en dividir el conjunto de datos en N subconjuntos llamados particiones (*folds*) (Bird et al., 2009). El entrenamiento se realiza N veces (Bird et al., 2009). En cada uno de ellos se selecciona una partición, la cual es usada como conjunto de prueba (Bird et al., 2009). El modelo se entrena con las observaciones de las otras particiones (Bird et al., 2009). Finalmente, se reporta la métrica promedio de todas las iteraciones (Bird et al., 2009).

Esta técnica ofrece una doble ventaja. En primer lugar, ofrece un resultado más confiable en tanto representa el promedio de varias mediciones (Bird et al., 2009). En

segundo lugar, la evaluación se hace menos dependiente del corte particular que se tiene al separar el conjunto de datos de prueba (Bird et al., 2009).

2.7.4. Redes neuronales y aprendizaje profundo

En un proceso de aprendizaje automático, el objetivo general es separar los factores de variación que explican los datos observados (Goodfellow, Begio y Courville, 2017). Sin embargo, en muchas aplicaciones reales, es difícil encontrar estos factores a partir de los datos crudos: se requiere una comprensión casi humana de los datos para ello (Goodfellow et al., 2017). El aprendizaje profundo resuelve este problema central introduciendo el concepto de representación de conocimiento en términos de otras representaciones más sencillas (Goodfellow et al., 2017).

En el contexto anteriormente descrito, las redes neuronales son el ejemplo por excelencia del paradigma del aprendizaje profundo (Goodfellow et al., 2017). Las redes neuronales son herramientas de modelamiento que han encontrado aceptación en una variedad de disciplinas por su capacidad de modelar problemas reales de alta complejidad (Basheer y Hajmeer, 2000). Pueden ser descritas como estructuras compuestas por unidades de procesamiento interconectadas (Basheer y Hajmeer, 2000). Estas estructuras reciben el nombre de neuronas artificiales o nodos (Basheer y Hajmeer, 2000). En ellas se genera una función que mapea una salida a un conjunto de entradas (Goodfellow et al., 2017). Esta función se genera a través de la composición de funciones más sencillas (Goodfellow et al., 2017).

A diferencia de otros algoritmos de aprendizaje, en los que se le da al ordenador un conjunto de características ya procesadas (la representación de la observación), las redes neuronales son capaces de descubrir la mejor representación de los datos (Goodfellow et al., 2017). Estas representaciones con frecuencia son mejores que aquellas diseñadas manualmente (Goodfellow et al., 2017). Por este motivo, estos sistemas suelen tener un mejor desempeño (Goodfellow et al., 2017).

Existen diferentes familias de arquitecturas de redes neuronales. Una de las más conocidas son las redes profundas "*feed-forward*" o perceptrones multicapa (Goodfellow et al., 2017). En estas arquitecturas, la información fluye en un solo sentido: de las entradas a las salidas (Goodfellow et al., 2017). No existen conexiones de retroalimentación (Goodfellow et al., 2017). Cuando la arquitectura permite alimentar a la red con una retroalimentación de las salidas, se habla de una red recurrente o

recursiva (RNN). Estas redes se usan cuando se trabaja con datos secuenciales (Goodfellow et al., 2017). Finalmente, en las redes convolucionales (CNN) se aplica una operación lineal llamada convolución (Goodfellow et al., 2017). Se utilizan para procesar datos que tienen una estructura local, como una imagen (grilla de dos dimensiones) (Goodfellow et al., 2017).

2.8. Conclusión

En este capítulo se presentó el sustento teórico necesario para una mejor comprensión del trabajo realizado en esta investigación. Se inició el capítulo presentando las proteínas, macromoléculas formadas por la unión de varios aminoácidos. Ellas pueden ser descritos en cuatro niveles de estructura: primaria (secuencia), secundaria, terciaria (estructura tridimensional) y cuaternaria. Luego, se introdujo el concepto de proteínas repetidas, que son aquellas que presentan una arquitectura modular repetida. Este tipo de proteína, como ya se mencionó, tiene un rol en varias enfermedades, por lo que su entendimiento es de gran relevancia para el diseño de tratamientos.

Luego de la introducción al mundo de las proteínas, se procedió a presentar los conceptos de simetría molecular. Se vio que existen distintos tipos de operaciones de simetría. Una de ellas es la simetría rotacional, la cual puede tener distintos órdenes. También se revisaron conceptos de interacciones moleculares como preludio a la fórmula del potencial de Lennard-Jones, usada para calcular la fuerza de las interacciones Van Der Waals. Adicionalmente, se hizo una presentación del análisis de modo normal (ANM), que es una técnica de simulación para predecir movimientos funcionales. Dado que algunos cálculos pueden ser muy complejos, aparecieron los modelos elásticos para simplificarlos. El modelo anisotrópico es uno de esos modelos.

En la última parte del capítulo, se introdujo conceptos de aprendizaje de máquina, los tipos de aprendizaje que existen y el flujo que se sigue para abordar un problema de esta naturaleza. Asimismo, se presentó tres métricas frecuentemente usadas para medir el desempeño de un clasificador. Finalmente, se dio una introducción breve a las redes neuronales.

Capítulo 3.

Estado del Arte

En este capítulo se presenta un mapeo sistemático de las técnicas y herramientas que se han desarrollado para la clasificación de proteínas repetidas.

3.1. Objetivo del mapeo

El objetivo del presente mapeo es la identificación de las técnicas desarrolladas paras llevar a cabo la clasificación de regiones de repetición en proteínas repetidas.

3.2. Preguntas de investigación

Para definir las preguntas de investigación, se utilizó el método PICo (población, intervención, contexto). Cada elemento de los anteriores es descrito en la tabla 3.1.

Tabla 3.1.

Conceptos de investigación.

Población	Proteínas repetidas.
Intervención	Clasificación, predicción, identificación, aprendizaje de máquinas, aprendizaje profundo.
Contexto	Bioinformática, bioquímica, ciencias de la computación.

P1: ¿Qué técnicas y herramientas existen para clasificar regiones de repetición en proteínas repetidas?

P2: ¿Qué técnicas de aprendizaje profundo se han usado para clasificar regiones de repetición?

3.3. Búsqueda de artículos

Para llevar a cabo la búsqueda de los artículos se definió la cadena de búsqueda en función de los elementos de investigación definidos en la sección anterior. La búsqueda se llevó a cabo con el uso de la versión anglosajona de dichos elementos. Los términos seleccionados se muestran en la tabla 3.2.

Tabla 3.2.

Palabras claves usadas en la búsqueda

Población	“Repeat protein”, “repeated protein”, “protein repeats”
Intervención	Classification, identification, prediction, database, “machine learning”, “deep learning”
Contexto	Bioinformatics, computer, biochemistry.

A partir de los términos anteriores, la cadena de búsqueda inicial definida fue la siguiente:

(“Repeat protein” OR “repeated protein”) AND (*classification OR prediction OR identification OR database OR “machine learning” OR “deep learning”*) AND (bioinformatics OR computer OR biochemistry).

La cadena anterior fue adaptada según las especificaciones de las dos bases de datos consultadas: *Web of Science* y *Scopus*. La búsqueda se realizó el 16 de abril del 2020. Los parámetros de la búsqueda se resumen en la tabla 3.3. En la misma se incluye, también, la cantidad de resultados iniciales de dicha búsqueda.

Tabla 3.3.

Cadena de búsqueda y resultados por base de datos.

Base	Cadena de búsqueda	Resultados
Web of Science	TI=("repeat protein" OR "repeated protein" OR "protein repeats") AND TS=(("machine learning" OR "deep learning" OR classification OR prediction OR database OR identification)) AND WC=("Biochemistry & Molecular Biology" OR "Computer Science" OR "Technology")	88
Scopus	TITLE-ABS (("repeat protein" OR "repeated protein" OR "protein repeats") AND ("machine learning" OR "deep learning" OR classification OR identification OR database OR prediction)) AND (LIMIT-TO (SUBJAREA , "BIOC") OR LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "BIOI"))	345
Total		433

3.4. Selección de resultados

Como primer paso, se consolidó todos los resultados encontrados en la etapa de búsqueda. Luego, se procedió a identificar y eliminar las publicaciones duplicadas a partir de su DOI. Finalmente, se aplicó los criterios de exclusión e inclusión detallados en la tabla 3.4. Para aplicar los filtros, se revisó el título y el resumen de las publicaciones.

Tabla 3.4.

Criterios de exclusión e inclusión de artículos.

Id	Descripción	Tipo
I1	Fecha de publicación del resultado debe ser del 1990 en adelante.	Inc.
I2	La publicación debe pertenecer al dominio de la bioinformática y, en particular, de las proteínas repetidas.	Inc.
I3	La publicación desarrolla al menos uno de los siguientes temas: menos uno de los siguientes temas dentro del dominio de estudio (detallado en I2): identificación, clasificación, predicción o almacenamiento (base de datos)	Inc.
I4	La publicación debe estar redactada en uno de los siguientes idiomas: español, inglés, italiano.	Inc.
E1	No se cumple uno de los criterios de inclusión.	Exc.
E2	La publicación no está disponible en línea y su resumen no está disponible.	Exc.

Como resultado, se seleccionó 26 artículos, tal como se muestra en la tabla 3.5.

Tabla 3.5.

Proceso de selección de artículos

Base de datos	Inicial	Sin duplicados	Selección
Web of Science	88	88	6
Scopus	345	301	20
Total	**433**	**389**	**26**

3.5. Análisis de resultados

P1. ¿Qué técnicas y herramientas existen para identificar y clasificar regiones de repetición en proteínas repetidas?

Existe una variedad de técnicas para clasificar las regiones de repetición en una proteína repetida. Ellas pueden agruparse en dos grupos: las técnicas basadas en secuencia y aquellas que se basan en estructura (Pellegrini, 2015).

A. Algoritmos basados en secuencia

La tarea de identificar y clasificar una región de repetición en una proteína mediante el uso de información de secuencia es difícil porque esta puede cambiar de diferentes formas y aun así mantener la estructura y/o funcionalidad (Pellegrini, 2015). Esta dificultad, sumada al incremento de poder de cómputo, ha ocasionado que la investigación en estos algoritmos disminuya en favor de aquellos basados en estructura. Esta tendencia se puede comprobar analizando la evolución de la cantidad de

publicaciones de este tipo de técnica a lo largo del tiempo. Dicho evolutivo se muestra en la figura 3.1.

Fig. 3.1. El número de investigaciones encontradas que trabajan con algoritmos basados en secuencia llegan hasta el 2009. En los siguientes años se empiezan a desarrollar investigaciones basadas en estructura. La figura excluye resultados en la categoría de Base de Datos y Otros. ***Fuente***: propia.

Algunos de los métodos iniciales para detectar regiones de repetición estuvieron basados en la búsqueda de alineamientos subóptimos dentro de la misma secuencia (Pellegrini, 2015). En principio, se buscaron similitudes comparando la secuencia contra sí misma. Para ello, se utilizaron métodos de alineamiento de secuencias. Como puede observarse, la principal característica de estos métodos era la falta de necesidad de conocimiento previo de características de las familias de repeticiones. Entre estos métodos estaban los algoritmos que operaban en REPRO (2000), RADAR (2000) y TRUST (2004). Como conjunto, recibieron el nombre de métodos *de novo* (Söding, Remmert y Biegert, 2006). Estos algoritmos, si bien servían para encontrar repeticiones largas, eran muy lentos; y, por tanto, poco escalables (Jorda y Kajava, 2009).

REP (Andrade, Ponting, Gibson y Bork, 2000) se basó en la identificación de homologías entre perfiles de secuencias para la identificación de repeticiones. Dicho de otro modo, identificaba las repeticiones por medio de la comparación entre la secuencia objetivo y un banco de perfiles de secuencia. El proyecto identificó más de 2000 repeticiones nuevas en la base SwissProt. Posteriormente, en (Andrade, Petosa, O'Donoghue, Müller y Bork, 2001) se identificó cuatro perfiles de secuencia adicionales para mejorar la detección automática de estructuras de repetición. Las similitudes encontradas entre dichos perfiles permitieron comprobar la similitud entre dos familias conocidas de repeticiones (ARM y HEAT).

(Kajava, 2001) llevó a cabo la identificación y clasificación de repeticiones tipo solenoide con una longitud de cadena en el rango de los 5 a 40 aminoácidos. En el mismo, señaló la existencia de una correlación entre la composición de aminoácidos (secuencia) con elementos de estructura tridimensional; en particular, la capacidad de realizar pliegues.

Con HHrep, herramienta desarrollada por (Söding et al., 2006), se propuso un nuevo enfoque entre las técnicas *de novo*. A diferencia de sus congéneres, en lugar de comparar secuencias, comparaba perfiles basados en modelos de Markov ocultos. Dos años después, se desplegó una versión mejorada de HHrep, que recibió el nombre de HHrepID. Debido su gran sensibilidad, la nueva herramienta fue capaz de detectar patrones con una divergencia de secuencia muy grande (Biegert y Söding, 2008).

Por otro lado, en (Kelil, Wang y Brzezinski, 2007 - 2007) se propuso usar una métrica de similitud entre secuencia de proteínas (SMS) como medida de distancia para un algoritmo de agrupación que denominaron CLUSS. A través de este algoritmo, pudieron identificar familias y subfamilias de proteínas que compartían estructura y función, entre ellas, proteínas repetidas. Según los autores, se trataba de uno de los primeros algoritmos independientes del alineamiento de secuencias. Por tal motivo, se trataba de una propuesta muy atractiva, en tanto muchas secuencias eran, por su naturaleza, imposibles o muy difíciles de alinear.

TPRpred (Karpenahalli, Lupas y Söding, 2007) fue una herramienta web para identificar y clasificar estructuras tipo solenoides. A diferencia de las herramientas existentes hasta el momento fue capaz de detectar secuencias repetidas con un mayor grado de divergencia. Esto se atribuyó, principalmente, a la construcción de perfiles más variados a través de un proceso iterativo de inclusión de secuencias al perfil y al uso de evaluaciones estadísticas más robustas. Dentro del marco de las estructuras solenoides, REPETITA, del año 2009, utilizaba algunas propiedades químicas (polaridad, volumen molecular, carga, entre otros) y una transformada de Fourier discreta para detectar y clasificar este tipo de estructura (Pellegrini, 2015).

T-Reks (Jorda y Kajava, 2009) fue una herramienta cuyo funcionamiento se dio a través de un algoritmo que se basaba en el análisis de la distribución de secuencias cortas mediante el uso de un algoritmo de K-Medias. Con sus resultados, se construyó una base de datos de proteínas repetidas.

B. Algoritmos basados en estructura

Con el aumento del poder de cómputo, en los últimos años, los algoritmos de identificación y clasificación basados en estructura han recibido una mayor atención. Un motivo adicional para este fenómeno es el hecho que las propiedades funcionales de las proteínas están más vinculadas con su estructura tridimensional que con su secuencia de aminoácidos. Este enfoque es computacionalmente complejo debido a que requiere realizar operaciones de transformación (traslaciones, rotaciones) sobre la estructura, teniendo en cuenta que éstas presentan un distinto grado de flexibilidad. En este paradigma, DAVROS (2004) fue un sistema de predicción y clasificación de elementos de repetición construido sobre un programa de alineamiento de estructuras que evalúa simetrías estructurales internas. Por otro lado, Swelfe (2008) y Prostrip (2010) fueron creados usando técnicas de programación dinámica para encontrar repeticiones estadísticamente significativas en la estructura tridimensional de una proteína (Pellegrini, 2015).

RAPHAEL (Walsh et al., 2012) fue una herramienta desarrollada para la identificación y clasificación de repeticiones en proteínas, específicamente, las de tipo solenoide. Para ello, entrenaron un clasificador tipo SVM con características usadas por curadores manuales para la misma tarea; por ejemplo, la frecuencia y la distancia. Estas variables fueron calculadas usando la información de coordenadas que extrajeron de algunas bases de datos de estructuras, como PDB. Al utilizar la herramienta sobre un conjunto de prueba, se obtuvo una exactitud de 89.5% con regiones solenoides. Esta herramienta fue utilizada posteriormente en (Di Domenico et al., 2014) para crear RepeatsDB, una base de datos de estructuras de proteínas repetidas.

ReUPred (Hirsh et al., 2016) fue una herramienta desarrollada para la predicción de unidades de repetición y clasificación de repeticiones. Su mecanismo para encontrar estructuras de repetición se basaba en una búsqueda iterativa sobre una base de estructuras de unidades de repetición (SRUL). Esta librería fue obtenida a partir de la información estructural de RepeatsDB. Probado sobre un conjunto de proteínas repetidas (solenoides), ReUPred detectó la repetición en un 92% de los casos; es más, fue capaz de clasificar correctamente el 89% de ellas. Este algoritmo fue mejorado y utilizado en RepeatsDB-Lite (Hirsh, Paladin, Piovesan y Tosatto, 2018), un servicio web para la predicción de elementos y unidades de repetición en proteínas repetidas. A diferencia de su antecesor, esta versión mejorada fue capaz de hacer predicciones sobre todas las clases identificadas en RepeatsDB. Adicionalmente, tuvo menos errores y una velocidad de procesamiento mayor.

Por otro lado, poniendo en secuencia a RAPHAEL con ReUPred, en (Paladin et al., 2017) implementó una versión mejorada de RepatsDB. Fue bautizada como RepeatsDB 2.0. El conjunto inicial de datos provino del Banco de Datos de Proteínas (PDB). Se extrajeron las repeticiones candidatas con RAPHAEL, las cuales fueron procesadas con RepUPred para identificar las posiciones de los fragmentos repetidos, así como su clase y subclase.

En (Jain, Yalamanchili y Parekh, 2009 - 2009) se propuso un enfoque alternativo. Este consistió en aplicar conceptos conocidos en teoría de grafos, las medidas de centralidad, para clasificar proteínas repetidas tipo HEAT y ARM. Para ello, se conceptualizó la biomolécula como un grafo. En este grafo, cada carbono-α correspondía a un vértice. Las aristas, por otro lado, a enlaces peptídicos. Esta técnica se basó en la intuición de que un nodo con alto grado de conectividad debería ser atravesado más veces por la ruta más corta entre otros dos nodos del grafo. Este nodo tendría un valor alto para una medida de centralidad llamada centralidad de intermediación (*Betwenness centrality*). En las proteínas, estos nodos tienen una participación importante en el doblamiento y funcionalidad de la proteína. Es así que se esperaba que los nodos involucrados en una hélice, estructura típica en proteínas ARM y HEAT, tuvieran un alto valor para esta métrica.

En un enfoque similar, (Chakrabarty y Parekh, 2014) propusieron PRIGSA, un algoritmo de detección de estructuras repetidas en proteínas. La herramienta representaba la proteína por medio de un grafo en el que cada vértice correspondía a un átomo. Adicionalmente, se consideró la existencia de una arista entre dos vértices cuando la distancia euclideana entre ambos era menor a los 7Å. El algoritmo que usaron se basó en el principio de que las interacciones entre y dentro de unidades de repetición podían ser detectadas por medio del vector propio de la matriz de adyacencia con el que representaron al grafo. De acuerdo con los autores, estos vectores propios eran únicos para cada uno de los distintos tipos de repeticiones, consecuencia de los diferentes arreglos que podían existir entre los elementos de estructura secundaria. Como consecuencia, pudieron formar perfiles propios por tipo de unidad de repetición y usarlos para la detección y clasificación de los mismo. Así, la herramienta recibía como entrada un archivo de coordinadas en formato PDB y entregaba como respuesta tres resultados: periodicidad, clasificación y límites de las unidades de repetición.

De forma similar, ConSole intentó identificar dominios tipo solenoide utilizando información estructural representada en una matriz de contactos. Su funcionamiento se basó en la búsqueda de patrones repetidos en dicha matriz. Adicionalmente, se aplicó reglas ad-hoc para manejar las inserciones (Pellegrini, 2015).

C. Bases de datos de proteínas repetidas

Los datos son elementos necesarios para el estudio de las proteínas repetidas. Por ello, algunos investigadores han puesto a disposición de la comunidad científica algunos repositorios con estructuras anotadas de proteínas repetidas.

ProtRepeatsDB (Kalita, Ramasamy, Duraisamy, Chauhan y Gupta, 2006) es una base de datos de repeticiones encontradas en secuencias de proteínas de 141 especies. Esta base de datos contiene información de distintos tipos de repeticiones: perfectas (homopeptídicas y heteropeptídicas) e imperfectas.

PRDB (Jorda, Baudrand y Kajava, 2012) es una base de datos curada que incluye las repeticiones encontradas por el programa T-REKS. Fue creada con el objetivo de realizar un análisis comparativo completo de este tipo de proteínas. Cabe recordar que el programa T-REKS utilizaba la secuencia de la proteína para identificar y clasificar las repeticiones.

ProRepeat es una base de datos creada para la investigación de las propiedades biológicas de repeticiones en proteínas. Esta base de datos fue construida a partir de la curación de secuencias disponibles en UniProt y RefSeq (Pellegrini, 2015).

RepeatsDB (Di Domenico et al., 2014) es una base de datos de estructuras repetidas. Fue construida con la finalidad de que se convirtiera en un recurso centralizado para la anotación y clasificación de repeticiones, repositorio inexistente en ese momento. Para ello, se utilizó RAPHAEL sobre información de estructura recuperada del PDB (*Protein Data Bank*). Como resultado, se identificó 10745 estructuras repetidas. Esta base fue actualizada posteriormente con RepeatsDB 2.0 (Paladin et al., 2017). Para esta actualización, se utilizó RAPHAEL en combinación con ReUPred para identificar y anotar las repeticiones en las estructuras candidatas.

P2: ¿Qué técnicas de aprendizaje profundo se han usado para clasificar regiones de repetición?

Las redes neuronales han sido usadas por años para estudiar distintos aspectos de las proteínas; por ejemplo, para predecir su estructura secundaria (Wafaa Wardah, M.G.M. Khan, Alok Sharma y Mahmood A. Rashid, 2019). Sin embargo, su uso específico en el dominio de las proteínas repetidas; y más aún, en su identificación y clasificación, es bastante limitado.

(Palidwor et al., 2009) entrenó y utilizó una red neuronal para identificar y clasificar repeticiones tipo vara-α. Para ello, se recopiló manualmente información de secuencia de repeticiones de este tipo. Como resultado, obtuvo un clasificador que era capaz de hacer la tarea con mayor efectividad que otros métodos basados en secuencia existentes en la época.

Para construir DeepSymmetry (Pagès y Grudinin, 2018) se entrenó una red neuronal convolucional 3D. El objetivo general de esta red fue detectar simetrías cíclicas en mapas volumétricos. Esta red recibía, en su capa de entrada, información del mapa de densidad electrónica y daba, como salida, información del orden y eje de simetría. A partir de esta información, la herramienta era capaz de detectar dos cosas: repeticiones en la estructura tridimensional y grupos simétricos de proteínas. DeepSymmetry se aplicó sobre datos del PDB e identificó cerca de 10000 repeticiones candidatas que no se encontraban en RepeatsDB.

3.6. Conclusión

En este capítulo se presentó los resultados de un mapeo sistemático realizado para conocer el estado del arte sobre el tema de clasificación de proteínas repetidas. En particular, se intentó responder dos preguntas: ¿qué técnicas y herramientas existen para identificar y clasificar regiones de repetición en estructuras proteínas? ¿Se ha usado técnicas de aprendizaje profundo para identificar y clasificar repeticiones?

Para cumplir el objetivo, se identificó un conjunto de palabras clave que fueron usadas para la búsqueda, realizada el 16 de abril del 2020, en los portales Scopus y Web of Science. Como resultado, se obtuvo 443 artículos; sin embargo, luego de aplicar un conjunto de criterios de exclusión e inclusión, la lista se redujo a 26.

Luego de la lectura de estos artículos, se procedió a responder cada pregunta de investigación. Con respecto a la primera pregunta, se encontró que existen dos grandes grupos de investigaciones y herramientas: aquellas que usan la información de secuencia y las que usan información de estructura. Se identificó un conjunto de herramientas en cada grupo y se describió brevemente su funcionamiento. Asimismo, se explicó las ventajas que tiene el enfoque basado en estructura en comparación con el enfoque basado en secuencia. También se mencionó un conjunto de bases de datos de repeticiones disponibles, entre las cuales destacó RepeatsDB como repositorio centralizado de este tipo de proteínas. Finalmente, con respecto a la segunda pregunta, se encontró que a la fecha no existe mucho desarrollo en la tarea de clasificación de repeticiones usando redes neuronales.

Capítulo 4.

Recolección de información estructural

En el presente capítulo se detallará el procedimiento realizado para cumplir con el primer objetivo específico. También se realizará el análisis de los resultados obtenidos.

4.1. Generalidades

El objetivo específico que se desarrollará en el presente capítulo es el primero: recolectar información estructural de unidades y regiones de repetición identificadas y clasificadas en RepeatsDB.

El alcance del proyecto en este punto incluye dos entregables: una librería de estructuras extraídas del banco de datos de proteínas (PDB en lo que resta del capítulo) y un catálogo de información estructural de regiones y unidades de repetición de clase IV encontradas en RepeatsDB.

4.2. Librería de estructuras de proteínas del PDB

El primer paso para la extracción de datos desde el PDB fue obtener un listado de todas las proteínas almacenadas en la base de datos. Para ello, se aprovechó el API web que expone la página oficial del PDB para descargar dicho listado. La URL usada para obtener dicho listado fue:

http://www.rcsb.org/pdb/rest/customReport.csv?pdbids=&reportName=Sequence&service=wsfile&format=csv*

El archivo descargado constaba de 606’068 registros y 9 columnas. Se aplicó un filtro sobre el campo *entityMacromoleculeType* para que solo se incluya las proteínas (*Polypeptide*). Una vez aplicado el filtro, la cantidad de registros se redujo a 570’177. En estos registros se identificaron 167’945 proteínas únicas, puesto que cada registro correspondía a una cadena (varias cadenas pueden pertenecer a una misma proteína). La tabla 4.1. resume las métricas básicas que describen la librería.

Tabla 4.1.

Reporte de métricas básicas de la librería de estructuras de proteínas.

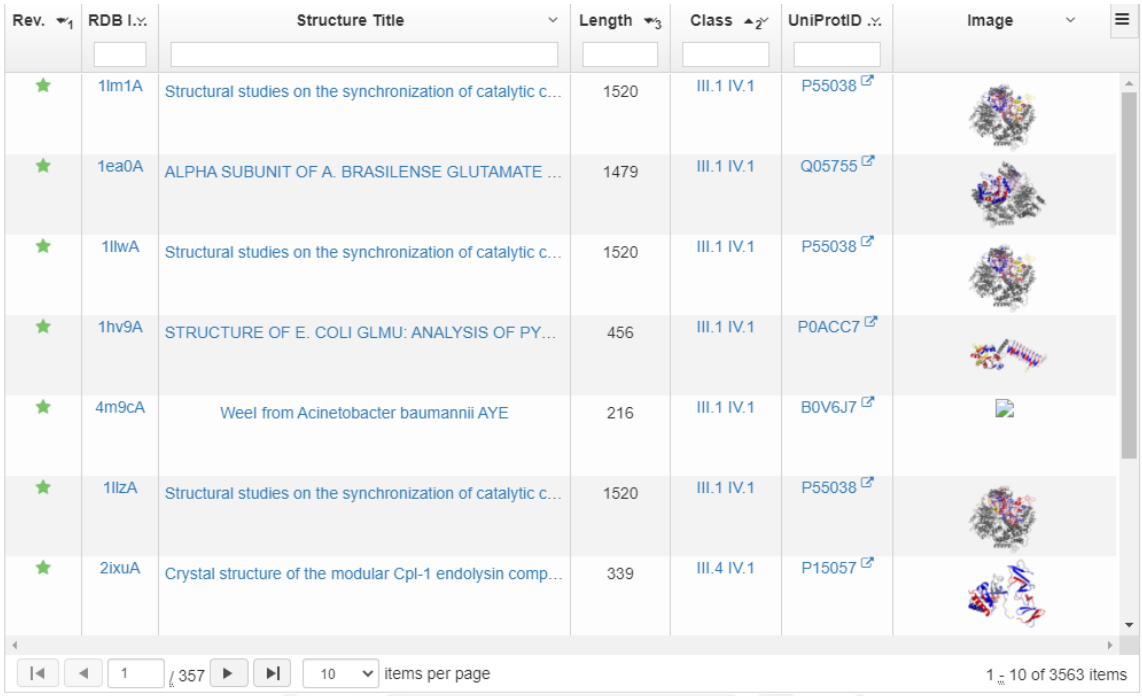
Métrica	Valor
Número de registros	606’068
Número de cadenas únicas	570’177
Número de proteínas únicas	167’945
Número de proteínas extraídas	167’130

Luego de obtener la lista de proteínas disponibles en el PDB, se procedió a extraer los archivos con la información estructural de cada una. Estos archivos tenían un formato especial llamado PDB cuya estructura ya se revisó en el capítulo 2. Para descargarlos, se utilizaron dos librerías de Python: PyPdb y ProDy. Cabe mencionar que durante la extracción no se pudo encontrar el archivo de 815 proteínas, por lo que el número se redujo a 167’130 archivos descargados. Finalmente, también es relevante indicar que, filtrando archivos corruptos, duplicados y cadenas con longitudes inferiores a 5 aminoácidos, el tamaño de la librería se redujo a 423’341 cadenas.

4.3. Librería de estructuras de unidades y regiones de repetición de clase IV

4.3.1. Extracción de información de unidades y regiones de repetición

De manera similar al procedimiento seguido para extraer la librería de PDB, el primer paso para la generación de librería de estructuras fue descargar un listado de cadenas de proteínas de clase IV que se encontraban en RepeatsDB. Para ello, se implementó un navegador (*web scapper*) que recorriera la grilla de resultados del buscador de RepeatsDB (configurado para buscar todas las cadenas de proteínas con repeticiones de clase IV). Esta grilla se encontraba paginada y mostraba 10 resultados por vez, tal como muestra la figura 4.1.



Rev.	RDB I.	Structure Title	Length	Class	UniProtID	Image
★	1lm1A	Structural studies on the synchronization of catalytic c…	1520	III.1 IV.1	P55038	
★	1ea0A	ALPHA SUBUNIT OF A. BRASILENSE GLUTAMATE …	1479	III.1 IV.1	Q05755	
★	1llwA	Structural studies on the synchronization of catalytic c…	1520	III.1 IV.1	P55038	
★	1hv9A	STRUCTURE OF E. COLI GLMU: ANALYSIS OF PY…	456	III.1 IV.1	P0ACC7	
★	4m9cA	Weel from Acinetobacter baumannii AYE	216	III.1 IV.1	B0V6J7	
★	1llzA	Structural studies on the synchronization of catalytic c…	1520	III.1 IV.1	P55038	
★	2ixuA	Crystal structure of the modular Cpl-1 endolysin comp…	339	III.4 IV.1	P15057	

1 / 357 10 items per page 1 - 10 of 3563 items

Figura 4.1. Grilla de resultados de una búsqueda en RepeatsDB.

Para implementar el navegador se utilizaron dos librerías de Python: *Selenium* para la navegación entre páginas y *Beautiful Soup* para extraer la información de cada página de resultados. Con la combinación de ambas librerías fue posible extraer una lista de hipervínculos a la página de cada una de las cadenas de proteínas de la página. Se observó que todos los enlaces tenían la forma siguiente: http://repeatsdb.bio.unipd.it/protein/<id_proteina><id_cadena>.

Una vez obtenido el listado de cadenas, se volvió a emplear *Beautiful Soup* para ingresar a cada enlace y extraer el archivo de datos de cada uno. El archivo en mención tenía extensión DB y contenía la información de las regiones de repeticiones contenidas en la cadena. A saber, indicaba el tipo de repetición, el residuo de inicio y fin de cada región y las unidades que componente. La figura 4.2. muestra un ejemplo de este tipo de archivo.

```
SOURCE  Reviewed
PDB     1lm1
CHAIN   A
RAPHAEL 33.7149
REG     871 1123 IV 1
REG     1300 1447 III 1
UNIT    871 897
UNIT    898 940
UNIT    941 961
UNIT    962 1029
UNIT    1030 1052
UNIT    1053 1100
UNIT    1101 1123
UNIT    1300 1318
UNIT    1319 1338
UNIT    1339 1369
UNIT    1370 1388
UNIT    1389 1406
UNIT    1407 1425
UNIT    1426 1447
INS     913 936
INS     968 1005
INS     1061 1077
```

Figura 4.2. Ejemplo de archivo DB extraído de RepeatsDB.

Como puede observarse en la figura 4.2, cada archivo DB contiene información valiosa de la repetición. Tiene 5 campos relevantes:

- **PDB:** identificador de la proteína en PDB.
- **CHAIN:** identificador de la cadena.
- **REG:** indica información de la región. El primer número corresponde al índice del primer aminoácido de la región. El segundo número es el del último aminoácido de la región
- **UNIT:** indica que lo que sigue son registros de información de la unidad de repetición. Los números que le siguen marcan el inicio y fin de la región de repetición.
- **INS:** la marca no se encuentra presente en todos los archivos, sino solo cuando la región presenta una o más inserciones. Al igual que en los puntos anteriores, los dos números que le siguen son el número de aminoácido de inicio y fin.

Cada uno de los archivos DB extraídos fue guardado en Google Drive con el nombre *<id_proteina>_<id_cadena>.db*.

4.3.2. Extracción de la información estructural de unidades y regiones de repetición

Como se mencionó en la sección anterior, el archivo de formato DB que se extrajo contenía cierta información de las unidades y regiones de repetición; sin embargo, carecía de la información de estructura y coordenadas de átomos. Para ello, fue

necesario recurrir nuevamente a los archivos PDB y extraer de ellos las coordenadas de los átomos que formaban parte de la región de repetición. Para facilitar esta tarea, se creó una jerarquía de clases como la que se ilustra en la figura 4.3.

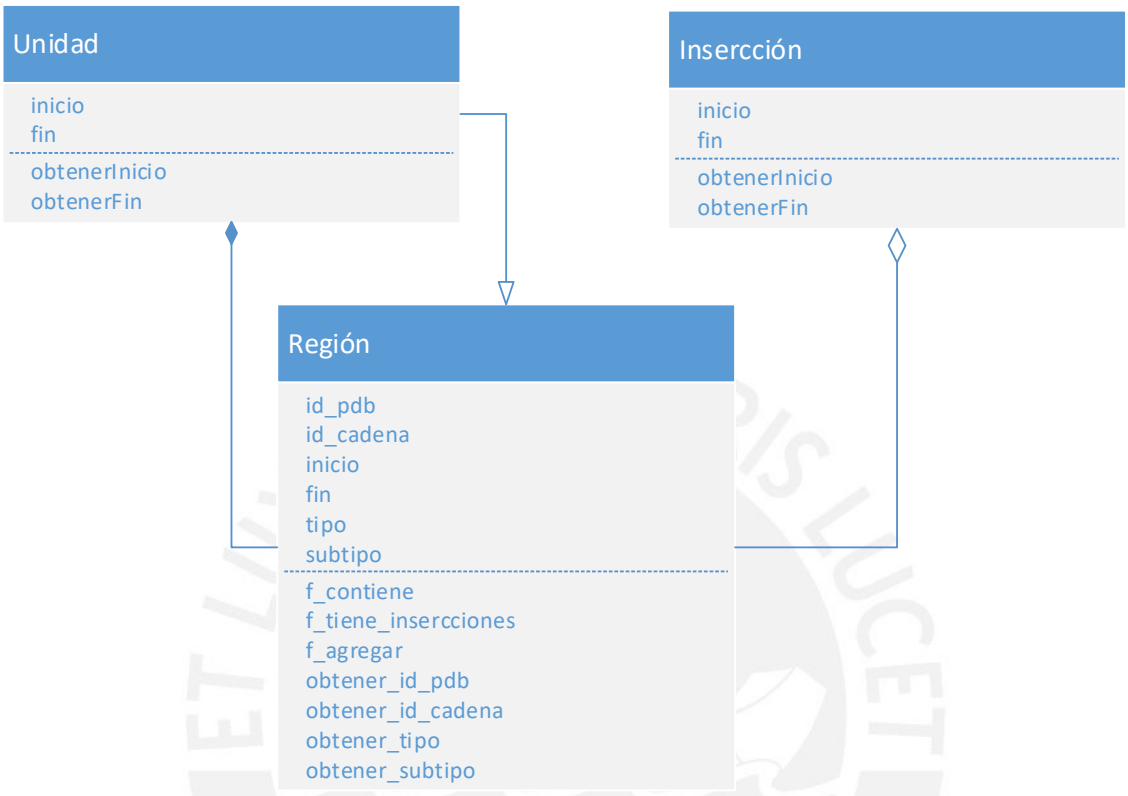


Figura 4.3. Diagrama de clases usadas para la extracción de información estructural.

Como primer paso, se completó la información de todas las regiones extraídas en RepeatsDB (archivos DB). Luego, por cada región, se extrajo la información contenida en el archivo PDB que correspondía a los residuos localizados entre el inicio y fin de la región de repetición. Esto se llevó a cabo empleando la librería de Python ProDy, la cual provee una función que permite seleccionar subconjuntos de átomos en un archivo PDB (ver la documentación de la herramienta, función *select*). Finalmente, se grabó la información recolectada en un nuevo archivo de formato PDB en Google Drive.

4.3.3. Resultados de la extracción

Posterior a la recopilación de los datos, se procedió a analizar los volúmenes descargados. A nivel de regiones, se recolectó información de 3’684 regiones, de las cuales 1’045 tenían inserciones. Se discriminó las regiones con inserciones debido a que el alcance inicial del proyecto no las toma en consideración, pues tienen el potencial

de introducir ruido a los clasificadores. La tabla 4.2 muestra cómo se distribuyen las regiones de repetición extraídas según la subclase a la que pertenecen.

Tabla 4.2.

Regiones de repetición de clase IV por subclase recolectadas de RepeatsDB. La base potencial se obtiene de remover las regiones con inserción de los datos.

Subclase	Regiones		
	Total	C/Inserción	Potencia
1	1’095	341	754
2	853	106	747
3	24	22	2
4	1’199	385	814
5	191	90	101
6	27	16	11
7	5	1	4
8	175	72	103
9	70	7	63
10	45	5	40
Total	**3’684**	**1’045**	**2’639**

La figura 4.4. muestra claramente que las subclases IV.1, IV.2 y IV.4. concentran más del 80% de los datos disponibles.

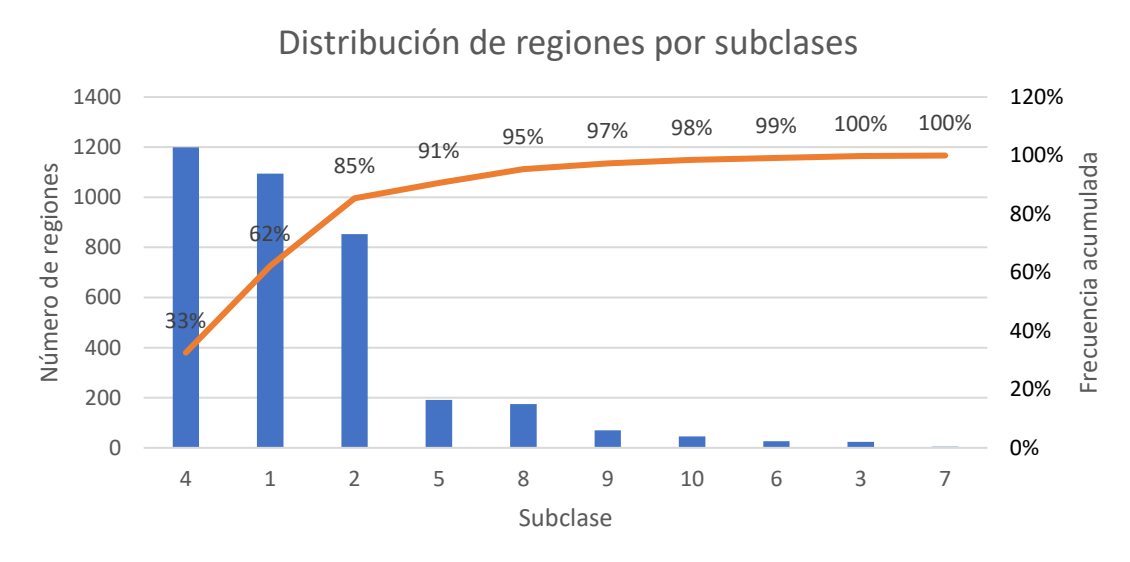


Figura 4.4. Distribución de regiones por subclases. El 85% de los datos recolectados se concentra en las subclases IV.1, IV.2 y IV.4.

Asimismo, como puede apreciarse en la figura 4.5, estas tres subclases tienen un porcentaje bajo de inserciones (máximo de 30% para las subclases 1 y 4) en relación con las otras subclases. Esto indica que, de excluirse las inserciones del análisis, se perderían menos casos para estas tres subclases.

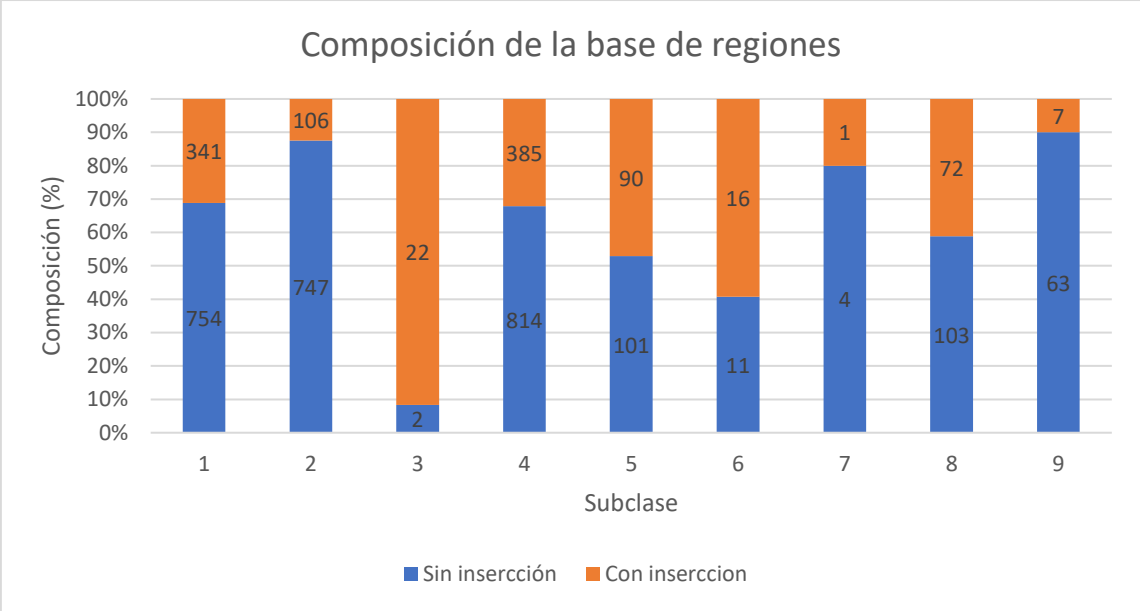


Figura 4.5. Proporción de inserciones por subclase (vista por regiones de repetición). Las subclases IV.1, IV.2 y IV.4 tienen un máximo de 30% de regiones con inserción, que es un porcentaje menor que el resto de subclases (excepto IV.7 y IV.9).

A partir de lo observado, se decidió trabajar solamente con las subclases IV.1, IV.2 y IV.4 porque eran las que tenían una mayor cantidad de datos disponibles. Trabajar con otras clases podría introducir ruido debido al escaso volumen de datos disponibles para ellas. Por ello, se creó una clase rotulada “Otros” que conceptualmente abarcara todo aquello que no perteneciera a las tres subclases anteriormente mencionadas.

4.4. Librería de estructura de repeticiones de clase III

Una vez finalizada la creación de la librería de estructuras de repeticiones de clase IV, se replicó el mismo procedimiento para crear una librería análoga para repeticiones de clase III. Como resultado de esta extracción, se construyó una librería con la información estructural de 2’560 regiones de esta clase. A diferencia de la clase IV, en la clase III se reconocen 6 subclases. Los resultados se resumen en la tabla 4.3. Es importante mencionar que esta librería se extrajo únicamente para efectos de evaluar el comportamiento de la herramienta con repeticiones que no eran de clase IV.

Tabla 4.3.

Regiones y unidades de repetición de clase III por subclase. La base potencial corresponde a la base total luego de remover las regiones con inserción.

Subtipo	Regiones		
	Total	C/Inserción	Potencia
1	515	325	190
2	487	134	353
3	1’309	760	549
4	131	53	78
5	75	27	48
6	43	43	0
Total	**2’560**	**1’342**	**1’218**

La distribución de éstas se puede apreciar en la figura 4.6. En ella se puede observar que cerca del 90% de todos los datos se concentran en las subclases III.1, III.2 y III.3. Finalmente, la figura 4.7. muestra que cerca del 50% de las regiones de clase III contienen al menos una inserción.

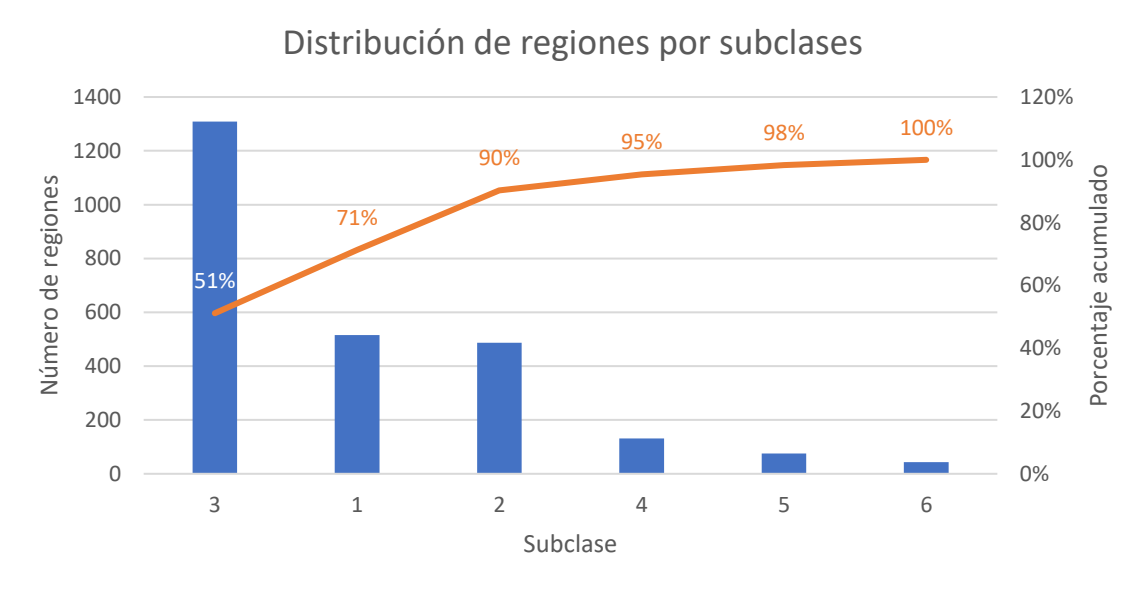


Figura 4.6. Distribución de regiones de clase III a nivel de subclases. El 90% de los datos están concentrados en las subclases III.1, III.2 y III.3.

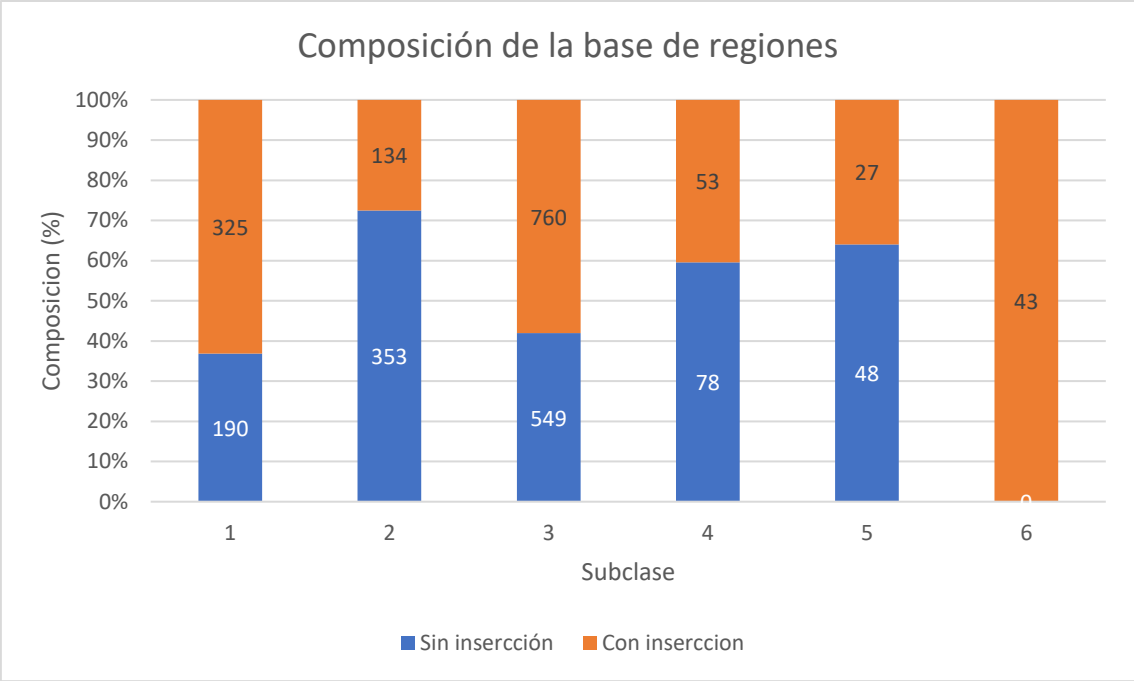


Figura 4.7. Proporción de inserciones en cada subclase (vista a nivel de regiones de repetición). Todas las subclases, a excepción de la III.2, tienen un porcentaje elevado de inserciones (más de 40%).

4.5. Conclusión

En este capítulo se detallaron los procedimientos usados para extraer los datos con los que se va a trabajar posteriormente en el proyecto. En primer lugar, se extrajo la información estructural de todas las proteínas disponibles en el Banco de Datos de Proteínas. Luego, se recolectó la información de las regiones de repetición y las unidades que las conforman en RepeatsDB. Finalmente, se combinó lo recolectado anteriormente para obtener la información estructural de las regiones de repetición de las clases III y IV.

Como resultado, se obtuvo dos librerías: una formada todas las proteínas disponibles en el PDB y otra con la información estructural de las regiones de repetición identificadas en RepeatsDB. De esta última se recolectó 3’684 regiones. A partir del análisis de los datos, se justificó la selección de las subclases IV.1, IV.2 y IV.4 para el resto del trabajo. Asimismo, se diseñó una clase “Otros” que estuviera formado por todo aquello que no fuera de las tres anteriores subclases. Finalmente, se recolectó una librería compuesta por 2’560 regiones de clase III.

Capítulo 5.

Identificación y clasificación de repeticiones en estructuras de proteínas

En el presente capítulo se describirá los procedimientos llevados a cabo para la obtención de la capa lógica de la herramienta; es decir, los módulos de detección y clasificación de repeticiones.

5.1. Generalidades

El objetivo que se detallará en este capítulo es el segundo: diseñar, implementar y evaluar un modelo de identificación y clasificación de repeticiones en la estructura de una proteína repetida.

Para este objetivo se ha identificado 3 entregables y resultados esperados: un análisis comparativo entre arquitecturas para reproducir en la herramienta; el conjunto de datos de entrenamiento y pruebas; y, finalmente, el clasificador entrenado. En los siguientes puntos se describirá con mayor detalle los procedimientos realizados para completar cada entregable.

5.2. Identificación de arquitecturas alternativas

Para abordar el problema de clasificación de repeticiones en proteínas desde un enfoque de aprendizaje automático, se realizó una búsqueda de literatura sobre un problema relacionado: la clasificación de estructuras de proteínas. Dicha búsqueda dio como resultado 5 enfoques alternativos:

1. Uso de una red recurrente bidireccional para la predicción de estructura a partir de secuencia.
2. Conversión de la información estructural en una imagen 2D y aplicar arquitecturas de clasificación de imágenes sobre ellas.
3. Convertir la información estructural en un mapa de volumen electrónico y procesarlo con una red convolucional 3D.
4. Transformar la información estructural en un grafo (*Protein Structure Network*, PSN) y obtener características de este para alimentar una red convolucional.

5. Similar a (4), sin embargo, se propone usar una red convolucional acoplada con una red recurrente LSTM.

A continuación, se presentará un breve resumen de cada propuesta.

5.2.1. Uso de una red recurrente bidireccional para la predicción de estructura

En este trabajo, presentado por (AlQuraishi, 2019), se utilizó una red neuronal recurrente bidireccional con una arquitectura “muchos a muchos” para la predicción de la estructura resultante de una proteína a partir de su secuencia. La información que usó cada celda de la red correspondía a un aminoácido de la estructura primaria, a partir del cual se predecía 3 elementos de la geometría resultante: la rotación del enlace, la longitud del enlace y la rotación torsional. De acuerdo con los autores, con esta arquitectura se consiguieron resultados de estado del arte sin tener que incorporar información evolutiva.

5.2.2. Transformación de información estructural en una imagen de dos dimensiones

En el trabajo descrito por (Sikosek, 2019) se convirtió la información de coordenadas atómicas de proteínas (información almacenada en el PDB) en tres diferentes matrices:

1. Las distancias interatómicas entre los átomos más pesados de la estructura.
2. La información de interacciones no covalentes (interacciones electrostáticas y de Van Der Vaals).
3. La matriz de correlación cruzada de un modelo anisotrópico.

Las tres matrices se utilizaron para crear una imagen 2D a colores. Es decir, cada matriz se convirtió en uno de los canales RGB de la imagen. Posteriormente, las imágenes obtenidas fueron procesadas por arquitecturas especializadas en procesamiento de imágenes (como VGG, Resnet, etc). La figura 5.1 ilustra el procesamiento propuesto por los autores.

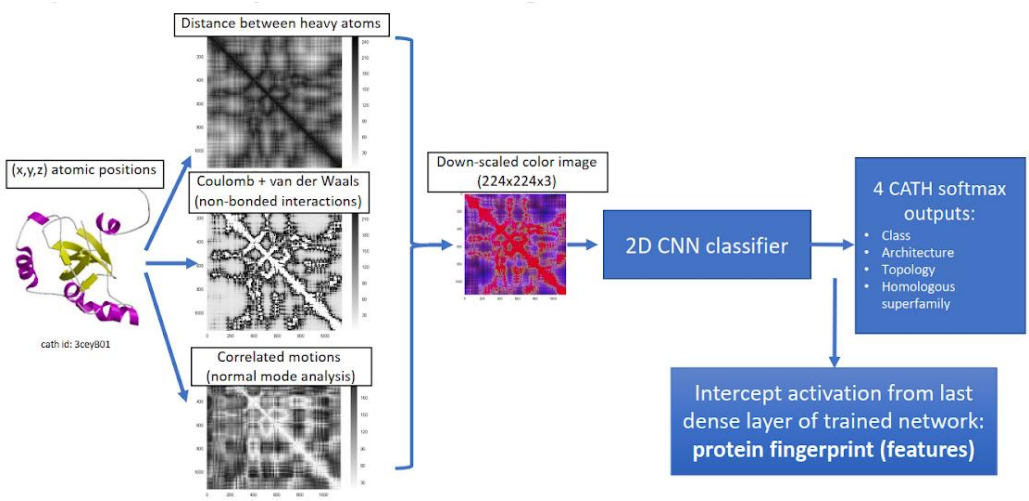


Figura 5.1. Ilustración del procesamiento de información estructural propuesto. Tomado de: (Sikosek, 2019)

5.2.3. Conversión de información estructural en un mapa de volumen electrónico

(Pagès y Grudinin, 2018) presentaron una red neuronal (CNN) que nombraron Deep Symmetry, entrenada para la predicción de simetrías rotacionales en la estructura de la proteína. Para ello, la información estructural fue transformada en mapas volumétricos. La CNN fue capaz de predecir dos elementos: el orden de simetría (del 1 al 10) y su eje. Los autores reportaron un área bajo la curva ROC de 0.7, por lo que -según ellos- representaba una alternativa a otros métodos que requieren un mayor procesamiento.

5.2.4. Transformación de la información estructural en un grafo (*Protein Structure Network*, PSN)

Algunos trabajos encontrados se basaron en la construcción de un grafo donde los átomos pesados de las proteínas fueron representados por los vértices. Las aristas quedaron definidas en función de una distancia umbral entre ellos. El grafo resultante recibió el nombre de red de estructura de proteína (PSN).

Estos trabajos convertían la información estructural en un grafo y hacían la predicción en función de sus propiedades. Así, en (Newaz, Ghalehnovi, Rahnama, Antsaklis y Milenković, 2020) se diseñó y entrenó una red neuronal con las características extraídas de la PSN. Por otro lado, (Guo, Newaz, Emrich, Milenkovic y Li Jun, 2019) utilizó información de una clase especial de subgrafo, los grafitos (graphlets) como

características para una red convolucional acoplada con una recurrente. En este esquema, la CNN se utilizó como un extractor de características capaz de detectar elementos de estructura secundaria y la red LSTM se usó para la predicción de la estructura tridimensional.

5.3. Selección de arquitectura

Para la selección de la alternativa a usar como arquitectura en el presente trabajo, se realizó un análisis comparativo entre las alternativas descritas. Este análisis se llevó a cabo en dos dimensiones:

a. **Complejidad para implementación de herramienta:** indica qué tan compleja es la implementación de la alternativa en el contexto del desarrollo de la herramienta de clasificación
b. **Costo computacional:** indica qué tan costoso en recursos computacionales resulta la ejecución de la alternativa.

En esta sección, se denotará cada una de las alternativas descritas en la sección anterior con los siguientes códigos:
ALT1: uso de red recurrente bidireccional
ALT2: transformación de estructuras en imágenes
ALT3: uso de CNN tridimensional
ALT4: uso de grafos (PSN)

5.3.1. Complejidad para la implementación

La alternativa 1 (ALT1) hace uso de una red recurrente bidireccional para hacer la clasificación. En ella, la red recibe la información de secuencia de la proteína y predice secuencialmente un conjunto de elementos de estructura (como ángulos torsionales). Con esta alternativa no es necesario recurrir a herramientas externas para realizar el preprocesamiento de datos, hecho que facilita su integración. Sin embargo, la complejidad de emplear un enfoque similar a este radica en la necesidad de entrenar un segundo clasificador capaz de explotar la información estructural predicha. Ello, adicionalmente, contribuye a la propagación del error de clasificación.

La alternativa 2 (ALT2) hace uso de CNN con arquitecturas conocidas para clasificación de imágenes para llevar a cabo la tarea de clasificar una proteína en función de su estructura. Esto hace sencillo su uso e implementación, ya que permite aplicar técnicas de transferencia de aprendizaje para realizar la tarea de clasificación. Sin embargo, se

requiere un preprocesamiento pesado de la información estructural para convertirla en imagen. Para conseguirlo, es necesario el uso de una diversidad de herramientas externas que potencialmente dificultarían su integración en una sola aplicación. No obstante, esta desventaja puede mitigarse mediante el uso de librerías en un entorno de desarrollo unificado (Python, por ejemplo).

La alternativa 3 (ALT3), de forma similar a ALT2, requiere transformar los datos de estructura a otra forma (mapa electrónico) para que puedan ser usados por una red neuronal. Además, su uso es más complejo que la ALT2 porque el clasificador predice la existencia de órdenes de simetría, no de regiones de repetición. Se plantea la posibilidad de usarla debido a que el alcance de la aplicación se limita a las regiones de estructuras cerradas, las cuales tienen una alta probabilidad de tener, por lo menos, un orden de simetría rotacional.

Para aplicar la alternativa 4 (ALT4) es necesario transformar la información de estructura en un grafo (PSN). Si bien los grafos son estructuras con propiedades muy conocidas, su representación puede ocupar una gran cantidad de espacio si el número de vértices y aristas es grande. En este caso hipotético, también generaría problemas de espacio en la memoria.

5.3.2. Costo computacional

Una ventaja que trae el uso de clasificadores estadísticos en comparación con otros métodos (como alineamientos) radica en el hecho que el costo computacional para la ejecución es menor. De hecho, el mayor consumo de recursos ocurre durante su etapa de entrenamiento, la cual se ejecuta una vez. El costo para hacer la predicción -una vez entrenado- es bajo. A pesar de lo mencionado anteriormente, cargar los parámetros de redes ya entrenadas sí puede requerir una cantidad importante de memoria.

Entre las cuatro alternativas, la ALT4, que propone acoplar una red convolucional con una red recurrente sería la más costosa porque habría que considerar, adicionalmente, la complejidad que implica transformar la información estructural en un grafo. De forma similar, la ALT1 también tiene un coste alto ya que en ella se trabaja con dos redes recurrentes bidireccionales acopladas. Tanto ALT2 como ALT3 proponen uso de redes convolucionales con distintos grados de complejidad. ALT3 implica el uso de una red convolucional 3D, que sin duda es más costosa que una red convolucional 2D (como las usadas en ALT2) con igual número de capas.

Otro factor que fue tomado en cuenta para evaluar el costo computacional de cada alternativa fue la necesidad de usar herramientas externas para convertir la información estructural a una forma que pudiera ser explotada por las respectivas redes neuronales. Dado que ALT1 y ALT4 no requieren de una transformación avanzada de los datos de entrada, quedaron fuera de este análisis. Por el contrario, tanto en ALT2 como en ALT3 se requiere convertir la información estructural en una imagen 2D y en un mapa de densidad electrónica respectivamente. Para ellos, se valen de herramientas externas. ALT3 usa una aplicación propia en C/C++ para hacer el cambio mencionado. ALT2, por otro lado, usa diferentes herramientas externas para generar la imagen 2D requerida por la red.

5.3.3. Arquitectura propuesta

Tras realizar la comparación entre las alternativas, se decidió usar las alternativas 2 y 3, ya que implicaban una complejidad y costo menor que las otras dos. La tabla 5.1. resume la evaluación realizada y descrita en la sección anterior.

Tabla 5.1.

Cuadro resumen de las características de las 4 alternativas de arquitectura analizadas. Considera: tipo de red, si usa herramientas externas para el preprocesamiento, si requiere un preprocesamiento particular, la complejidad de ejecución/integración y el costo computacional para su ejecución.

Criterio/Alt.	**ALT1**	**ALT2**	**ALT3**	**ALT4**
Tipo de red	RNN	CNN2D	CNN3D	CNN+RNN
Usa herramienta.	No	Sí	Sí	Sí
Transformación	No aplica	Imagen	Mapa densidad	Grafo (PSN)
Complejidad	Alta	Alta, pero mitigable	Media	Alta
Costo computacional	Muy alto	Alto	Alto	Muy alto

La arquitectura propuesta consiste en acoplar las alternativas 2 y 3. En este esquema, ALT3 actuará como filtro de repeticiones y ALT2 clasificará las estructuras que pasen dicho filtro.

5.4. Identificación del umbral de detección

Para identificar el umbral de detección, se trabajó con las dos librerías recopiladas en el capítulo 4. En este sentido, se tomó toda la base de la librería PDB y se separó en dos grupos:

- **Grupo A:** formado por las estructuras de las cadenas que pertenecen a la librería de repeticiones. Como se mencionó en el respectivo capítulo, la librería de estructuras de clase IV tenía 3’684 entradas. De ellas, 1’045 registraban la presencia de al menos una inserción, por lo que fueron descartadas. Asimismo, se filtraron 160 regiones por pertenecer a cadenas con más de 1 región. Por tanto, quedó un total de 2’471 regiones por analizar. El proceso de filtrado se resume en la tabla 5.2.
- **Grupo B:** formado por las cadenas que no pertenecen al grupo A. Adicionalmente, se filtraron aquellas que tenían una longitud menor a 80 aminoácidos, ya que se consideraron muy pequeñas para contener una repetición de clase IV. Este grupo estuvo formado por 225’585 cadenas.

Tabla 5.2.

Filtros aplicados sobre la librería de regiones para el grupo A.

Cantidad inicial	**3’684 regiones**
Regiones con inserción	-1’045 regiones
Regiones que pertenecen a cadenas con más de 1 región	-160 regiones
Cantidad final	**2’471 regiones**

Como se mencionó en la sección anterior, se propone el uso de Deep Symmetry como filtro inicial. La mencionada red está entrenada para identificar la presencia de simetrías rotacionales en la estructura de la proteína (o cadena) ingresada. Específicamente, es capaz de identificar los órdenes de simetría rotacional presentes en la estructura. Dado que las estructuras de clase IV se caracterizan por tener una estructura cíclica cerrada, era de esperar que muchas de ellas tengan ciertos órdenes de simetría rotacional, lo que justificó, en su momento, el uso de esta red con el fin anteriormente descrito.

Para adaptar el uso de esta red a los objetivos del proyecto, fue necesario encontrar un umbral que sirviera para definir cuándo una estructura debía pasar el filtro (positivo) y cuándo no (negativo). Este umbral fue calculado por prueba y error usando los dos grupos descritos al inicio de esta sección. Para ello, se tuvo que realizar un balance entre dos objetivos: maximizar el número de elementos del grupo A que sean marcados como positivos y, a su vez, minimizar el número total de positivos en el grupo B.

Para el cálculo del umbral, se tuvo en consideración el resultado devuelto por Deep Symmetry: un vector que tenía los logits para cada orden de simetría. La figura 5.2. muestra un ejemplo de dicho vector.

Figura 5.2. Ejemplo de una salida de Deep Symmetry

Como puede observarse en la figura, parte de la salida de la red consta de 10 valores, cada uno asociado a un orden de simetría (el orden 1 es ausencia de simetría). Sin embargo, dichos valores tal como aparecían no podían ser interpretados como probabilidades. En consecuencia, fue necesario aplicar una función sobre cada elemento para transformarlo en un número que pueda ser leído como la probabilidad que la cadena presente un determinado orden de simetría. Dos funciones que se usan comúnmente para este fin en tareas de clasificación son la sigmoide y softmax. Para este caso, se optó por aplicar la primera de ellas debido a que las etiquetas no eran excluyentes entre sí: una cadena puede presentar varios órdenes de simetría simultáneamente. La cadena puede, por ejemplo, tener una simetría de orden 4 y 2 simultáneamente. De aplicarse la función softmax, solo se hubiese podido predecir uno de estos órdenes. Como consecuencia, se llevó a cabo el cálculo de 9 umbrales (para los órdenes 2 al 10). Como punto inicial para hallar dichos umbrales, se usó la información descriptiva resumida en la tabla 5.3. Luego de hacer varías pruebas, se encontraron los siguientes umbrales:

1) Un umbral de 0.98 para N=2 (distribución hacia mayores valores)
2) Un umbral de 0.80 para N=(3,4), con valores altos, pero menores que (1).
3) Un umbral de 0.50 para el resto (distribución hacia bajos valores)

Tabla 5.3.

Estadísticos descriptivos de la probabilidad de cada orden de simetría para la clase IV.

Variable	sig_1	sig_2	sig_3	sig_4	sig_5	sig_6	sig_7	sig_8	sig_9	sig_10
Media	0.680	0.971	0.886	0.736	0.433	0.360	0.142	0.102	0.029	0.041
Des.Est.	0.163	0.057	0.147	0.298	0.263	0.235	0.218	0.108	0.027	0.058
Mínimo	0.017	0.021	0.000	0.006	0.001	0.048	0.001	0.001	0.004	0.002
Cuartil 1	0.600	0.972	0.843	0.426	0.187	0.204	0.044	0.043	0.016	0.011
Cuartil 2	0.715	0.988	0.956	0.925	0.387	0.261	0.060	0.069	0.028	0.025
Cuartil 3	0.796	0.996	0.983	0.981	0.658	0.403	0.114	0.107	0.036	0.057
Máximo	0.968	0.999	0.999	0.999	1.000	1.000	1.000	0.966	1.000	1.000

Luego de la aplicación de los umbrales (1) al (3), se realizó un conteo de cuántos órdenes de simetría estaban presentes en la cadena. Si dicho número era mayor a 1, se consideró automáticamente como potencial a tener una región clase IV. Con los umbrales definidos, fue posible identificar correctamente el 78% de las regiones del grupo A. Asimismo, del grupo B pasaron el filtro 134’407 cadenas. Esto representa el 32% de la librería PDB (con sus 423’341 cadenas).

Tabla 5.4.

Distribución de órdenes de simetría por clase de región

Simetrías	Clase IV (Q. regiones)
0	8
1	527
2	483
3	797
4	452
5	191
6	13
Total	**2471**

5.5. Conjuntos de datos de desarrollo y prueba

5.5.1. Variable objetivo

La variable objetivo para los distintos clasificadores entrenados fue la subclase de la cadena ingresada. Se asumió que dicha cadena contenía alguna repetición de clase IV ya que, como se describió en la sección 5.3.3., la herramienta solo clasifica las cadenas que pasan el filtro del detector.

Como se indicó en el capítulo 4 al analizar la distribución de los datos de subclases dentro de la clase IV, se decidió trabajar con 4 etiquetas: IV1, IV2, IV4 y OTROS. La última de ellas estuvo formada por instancias que no pertenecían a ninguna de las 3 subclases anteriores.

5.5.2. Conjunto de desarrollo

Para construir el conjunto de desarrollo, se tomaron las 3’684 regiones de clase IV y se etiquetaron según el esquema descrito en la sección 5.5.1. Posteriormente, se filtraron las regiones que tenían inserciones y las que pertenecían a cadenas con más de una

región. Finalmente, se analizó la distribución de clases en el conjunto de datos resultantes. Esta se puede apreciar en la tabla 5.5.

Tabla 5.5.

Distribución de regiones por tipo en el conjunto de desarrollo.

Etiqueta	Cantidad de regiones
IV1	744
IV2	716
IV4	722
OTROS	457
OTROS (final)	657
Total	**2’839**

Como puede observarse, la distribución de clases está bien balanceada. La única excepción es la clase OTROS. Debido a que en ella se agrupan todas aquellas regiones que no son de subclase IV1, IV2 y IV4, se enriqueció dicha clase con una muestra aleatoria de 200 regiones de clase III.

Una vez identificadas las regiones que serían usadas, se procedió a transformar su información estructural en imágenes. Para ello, fue necesario implementar un script en Python. En dicho script se usó la información de coordenadas atómicas para construir 3 matrices cuadradas de dimensión N (N es el número de átomos identificados en el archivo PDB) como fueron descritas por (Sikosek, 2019). Las matrices fueron las siguientes:

a) **Matriz de distancias:** se calculó la distancia euclidiana entre cada par de átomos posible. Cabe mencionar que se empleó un tope de distancia de 51.2A. Por encima de dicha distancia, se asignó el valor de cero (0) a la celda. Este cálculo se llevó a cabo empleando la librería ProDy.

b) **Matriz de correlación cruzada:** se trató de una matriz obtenida como resultado de la ejecución de un análisis de modos normales con un modelo anisotrópico. Este cálculo se llevó a cabo empleando la librería ProDy.

c) **Matriz de energía de enlace no covalente:** se calculó la energía de enlace Van Der Waals (en Kcal/mol) entre dos átomos. Para ello, se empleó la ecuación del potencial de Lennard-Jones con los parámetros reportados en el capítulo 2. Para esta matriz, se tomaron en consideración los valores entre -0.2 y 0.2 Kcal/mol. Los valores fuera de este rango fueron reemplazados por cero (0) ya que son valores muy altos para una fuerza Van Der Waals. De la misma forma, se asignó el valor de 0 a las interacciones entre átomos de un mismo residuo. A diferencia de lo propuesto en la investigación de (Sikosek, 2019), no se incluyó el término

de fuerzas iónicas porque el cálculo de cargas parciales requerido iba a hacer demasiado lento el proceso de transformación a imagen. Por el mismo motivo, se usó un umbral de energía distinto al reportado en dicho trabajo.

Posteriormente, se escaló cada matriz a valores entre 0 y 255, ya que es el rango de valores válidos para un canal RGB. Para ello, se empleó la fórmula 5.1.

$$\frac{X - X_{min}}{X_{max} - X_{min}} \times (255 - 0) + 0 = 255 \times \frac{X - X_{min}}{X_{max} - X_{min}} \dots (5.1)$$

Una vez realizado el escalamiento, se utilizó cada matriz como canal RGB para construir la imagen. Finalmente, la imagen resultante se escaló a una dimensión de 224x224. Para ambas acciones, se hizo uso de la librería PIL de Python. Cada imagen fue guardada en Google Drive con el nombre *<id_proteina>_<id_cadena>_<inicio>_<fin>_<clase><subclase>.png*.

Una vez procesadas las imágenes, se procedió a dividir el conjunto de datos de desarrollo en tres: un conjunto de entrenamiento, uno de validación y otro de pruebas. La figura 5.3 esquematiza la división realizada y el número de regiones en cada subconjunto.

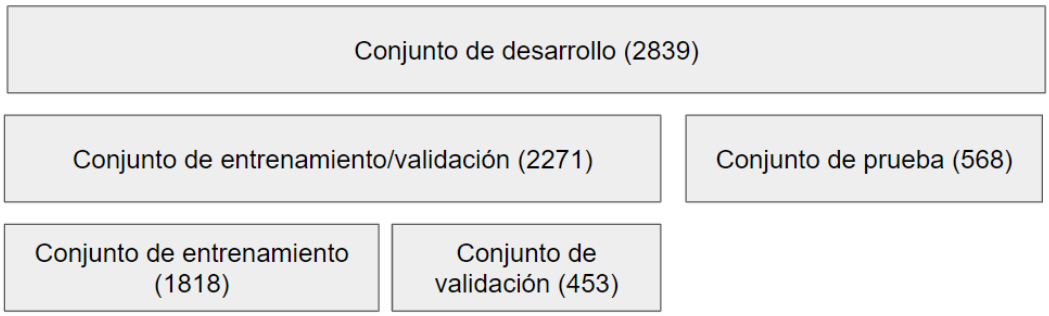


Figura 5.3. División del conjunto de desarrollo. El conjunto total fue dividido inicialmente en 2: entrenamiento (80%) y prueba (20%). El conjunto de entrenamiento fue nuevamente dividido en 2: entrenamiento (80%) y validación (20%).

5.6. Selección de modelo

Para la selección del modelo se empleó un diseño factorial completo 2^2. En este diseño, los factores a analizar fueron 2: la arquitectura de red convolucional (factor A) y el grado de regularización en la capa completamente conectada posterior a la capa convolucional (factor B). Los niveles usados fueron:

- **Factor A:** Densenet en el nivel bajo (-1) y Resnet en el nivel alto (+1).
- **Factor B:** Baja regularización en el nivel bajo (-1) y alta regularización en el nivel alto (+1). Específicamente, la arquitectura de la red completamente conectada para cada caso se detalla en la figura 5.4.

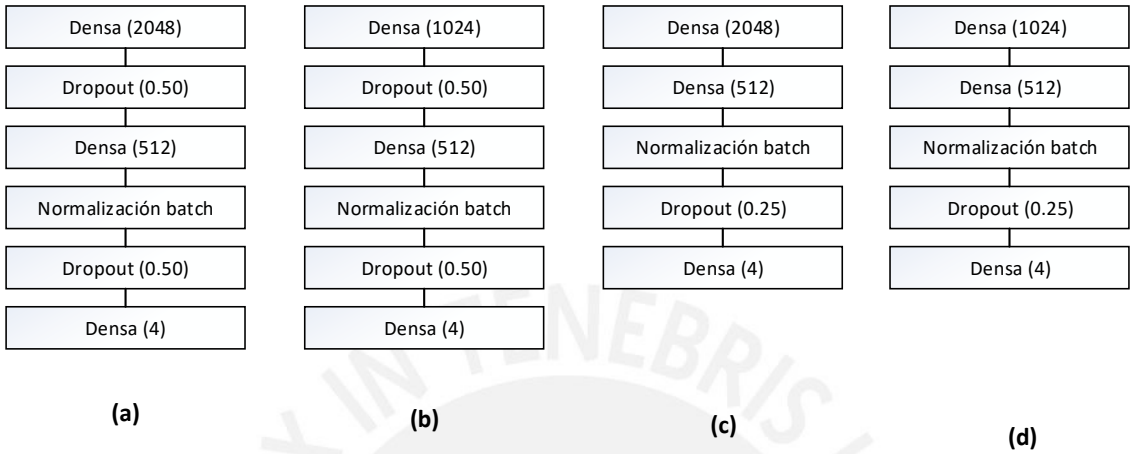


Figura 5.4. Arquitecturas de red usadas para la capa completamente conectada. (a) Resnet, nivel alto de regularización. (b) Densenet, nivel alto de regularización. (c) Resnet, nivel bajo de regularización. (d) Densenet, nivel bajo de regularización.

Debido al balance entre clases en el conjunto de entrenamiento y validación (se hizo la separación con estratificación), fue posible definir la exactitud como métrica a comparar para realizar la selección del modelo. La medición de esta métrica para cada combinación de niveles se realizó 5 veces. Dado que existían 4 posibles combinaciones de los niveles de los factores (alto+alto, alto+bajo, bajo+alto, bajo+bajo), se tomó un total de 20 mediciones en orden aleatorio. El detalle de los resultados obtenidos puede apreciarse en la tabla 5.6.

En todos los experimentos se entrenó el modelo empleando el conjunto de entrenamiento y se evaluó su desempeño sobre el conjunto de validación. El entrenamiento se llevó a cabo en 20 épocas, en lotes de 32 muestras por vez. La función de costo seleccionada fue la entropía categórica cruzada (*categorical cross-entropy*) y usando Adam como optimizador. La tasa de aprendizaje inicial fue de 0.01. Esta tasa se redujo a la 0.002 en la sexta época; a 0.0002 en la undécima; y, finalmente, a 0.00002 en la decimosexta.

Tabla 5.6.

Resultados de las mediciones bajo el diseño factorial completo.

Orden	A	B	Exactitud		Orden	A	B	Exactitud
1	+1	+1	0.8918		11	-1	+1	0.9161
2	+1	+1	0.8896		12	+1	-1	0.9117
3	-1	-1	0.9205		13	+1	-1	0.8918
4	+1	+1	0.8962		14	+1	+1	0.8985
5	-1	+1	0.9183		15	+1	-1	0.9051
6	-1	-1	0.9183		16	-1	+1	0.9205
7	-1	-1	0.9139		17	-1	+1	0.9205
8	+1	-1	0.9073		18	-1	-1	0.9117
9	+1	+1	0.8918		19	+1	-1	0.9073
10	-1	-1	0.9272		20	-1	+1	0.9227

Con los resultados obtenidos, se llevó a cabo un análisis de varianza para determinar qué combinación de arquitectura CNN y regularización de la capa densa escoger a fin de maximizar la exactitud medida. El primer paso fue reorganizar los resultados de manera que se facilitara el cálculo de los efectos y contrastes de cada factor. Dicho reordenamiento adoptó la forma de la tabla 5.7.

Tabla 5.7.

Resultados de las mediciones acompañados de los elementos de notación Yates

A: CNN	B:Reg.	A	B	Exactitud					Total	Yates
Densenet	Baja	-	-	0.9205	0.9183	0.9139	0.9272	0.9117	4.5916	(1)
Resnet	Baja	+	-	0.9073	0.9117	0.8918	0.9051	0.9073	4.5232	a
Densenet	Alta	-	+	0.9183	0.9161	0.9205	0.9205	0.9227	4.5981	b
Resnet	Alta	+	+	0.8918	0.8896	0.8962	0.8918	0.8985	4.4679	ab

Con los valores calculados en la tabla fue posible calcular los estimadores puntuales de los efectos. Así:

$$Efecto\ A = \frac{1}{2n}[a + ab - b - (1)] = \frac{1}{2 * 5}[4.5232 + 4.4679 - 4.5981 - 4.5916] = -0.01986$$

$$Efecto\ B = \frac{1}{2n}[b + ab - b - (1)] = \frac{1}{2 * 5}[4.5981 + 4.4679 - 4.5232 - 4.5916] = -0.00488$$

$$Efecto\ A * B = \frac{1}{2n}[(1) + ab - a - b] = -0.00618$$

A partir de los resultados obtenidos, se pudo observar que el efecto del factor A era más fuerte que el del factor B y el de la interacción AB. Sin embargo, para demostrar si los efectos eran realmente significativos se llevó a cabo un ANOVA de dos vías con estos resultados. Este análisis se llevó a cabo empleando el programa SPSS de IBM. El resumen del análisis se muestra en la tabla 5.8. Cabe mencionar que el nivel de significancia de la prueba fue de 0.05.

Tabla 5.8.

*Resultados ANOVA de dos vías (sig.=0.05). A partir de los p-valores, destaca la significancia del factor de interacción A*B sobre los resultados.*

Pruebas de los efectos inter-sujetos

Variable dependiente: acc

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.	Eta al cuadrado parcial
Modelo corregido	.002[a]	3	.001	26.920	.000	.835
Intersección	16.527	1	16.527	584755.854	.000	1.000
A	.002	1	.002	69.828	.000	.814
B	.000	1	.000	4.172	.058	.207
A * B	.000	1	.000	6.759	.019	.297
Error	.000	16	2.826E-005			
Total	16.530	20				
Total corregida	.003	19				

a. R cuadrado = .835 (R cuadrado corregida = .804)

Como puede observarse en la figura 5.8., el valor p para el factor de interacción (AB) es menor que 0.05, el nivel de significancia de la prueba [23]. Por tanto, se pudo concluir el efecto de la interacción entre A y B sí es significativo para el valor observado. Por tanto, el resultado de la métrica medida sí depende de la interacción entre ambos factores (capa convolucional y nivel de regularización de capa conectada). Como consecuencia adicional de este resultado, el análisis de los efectos individuales de A y B se dejó de lado ya que la interacción de ambos la que influye en la respuesta.

Las figuras 5.5 y 5.6 muestran los gráficos de los efectos de la interacción entre A y B (A*B y B*A, respectivamente). En ellas pudo observarse que la red Densenet tuvo mejor desempeño que Resnet en cualquiera de sus niveles de regularización. Esta observación pudo comprobarse analíticamente mediante prueba de hipótesis con los

[2] Cuando el valor p (p-valor) es inferior al nivel de significancia de la prueba, se rechaza la hipótesis nula.
[3] Para ANOVA, la hipótesis nula (H0) es que la media es igual en todos los niveles. En este caso, que el efecto no es significativo pues es el mismo en cada nivel.

contrastes [45].Como puede observarse en la tabla 5.9, los p-valores fueron inferiores al nivel de significancia de la prueba en ambos niveles del factor B. Por tanto, se aceptó la hipótesis alternativa de que existe una diferencia en las medias observadas. Esto quiso decir que la exactitud media observada con Densenet era mayor a la de Resnet.

De manera similar, se hizo otra prueba de contrastes; esta vez, a través de los niveles del factor A (tabla 5.10). Al igual de lo que ocurrió con la prueba anterior, se observó una diferencia entre los resultados de Resnet con alto y bajo grado de regularización. Sin embargo, esto no se observó para Densenet: de acuerdo con los resultados, no existía una diferencia entre los resultados obtenidos aplicando Densenet con alto y bajo grado de regularización. Esto también se pudo apreciar gráficamente en la figura 5.5.

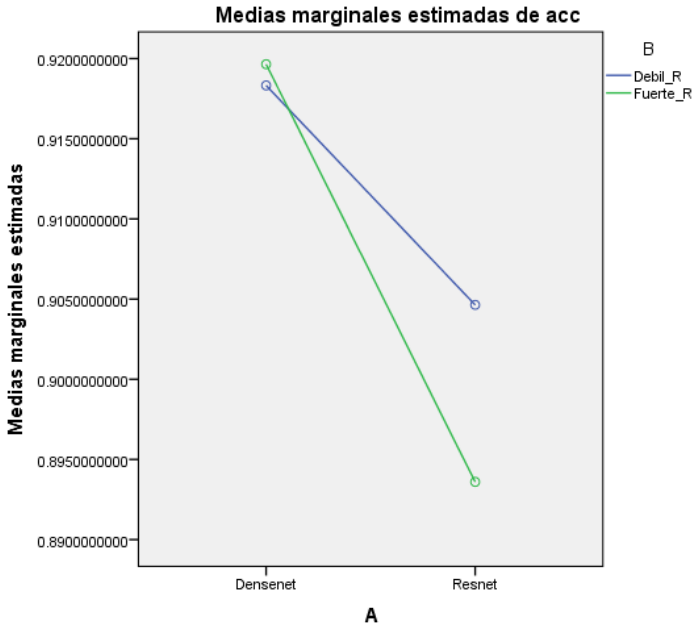


Figura 5.5. Diagrama de efectos de interacción A*B. Se observa que los resultados obtenidos con Densenet con los dos niveles de regularización trabajados son muy casi iguales.

[4] Un contraste es una combinación lineal donde la suma de coeficientes es cero. Así, X1-X2 es un contraste pues 1-1=0.

[5] La prueba de hipótesis tiene como hipótesis nula que existe un contraste con las medias observadas. Como en este caso hay dos niveles: u1-u2=0.

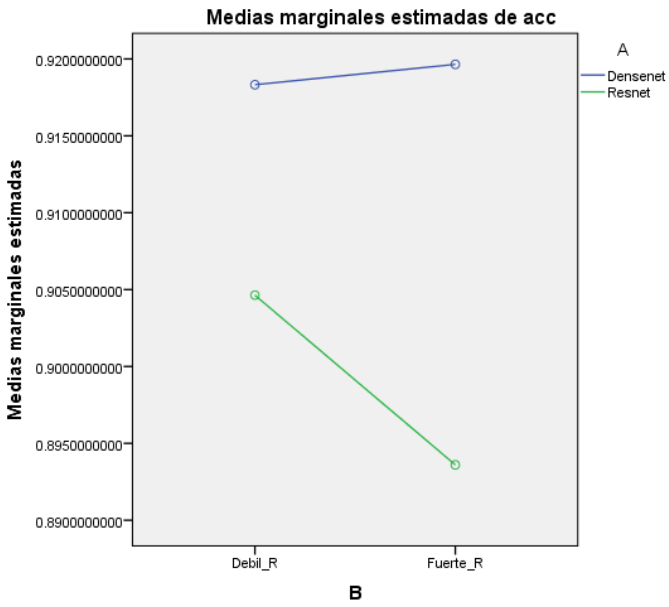


Figura 5.6. Diagrama de efectos de interacción B*A. Se observa que Densenet presenta mejores resultados, independientemente del grado de regularización.

Tabla 5.9.

Resultado de prueba de contrastes en cada nivel del factor B. El p-valor permite comprobar que, independientemente del grado de regularización, sí existe una diferencia significativa entre los resultados obtenidos por Densenet y Resnet.

Contrastes univariados

Variable dependiente: acc

B		Suma de cuadrados	gl	Media cuadrática	F	Sig.	Eta al cuadrado parcial
Debil_R	Contraste	.000	1	.000	24.402	.000	.604
	Error	.000	16	1.474E-005			
Fuerte_R	Contraste	.002	1	.002	115.116	.000	.878
	Error	.000	16	1.474E-005			

Cada prueba F contrasta el efecto de A. Estos contrastes se basan en las comparaciones por pares, linealmente independientes, entre las medias marginales estimadas. Estas pruebas se basan en las comparaciones por pares linealmente independientes entre las medias marginales estimadas.

Tabla 5.10.

Resultado de prueba de contrastes en cada nivel del factor A (CNN). El p-valor permite comprobar que no existe diferencia significativa entre los resultados obtenidos con Densenet para cualquiera de los dos niveles de regularización.

Contrastes univariados

Variable dependiente: acc

A		Suma de cuadrados	gl	Media cuadrática	F	Sig.	Eta al cuadrado parcial
Densenet	Contraste	4.386E-006	1	4.386E-006	.298	.593	.018
	Error	.000	16	1.474E-005			
Resnet	Contraste	.000	1	.000	27.498	.000	.632
	Error	.000	16	1.474E-005			

Cada prueba F contrasta el efecto de B. Estos contrastes se basan en las comparaciones por pares, linealmente independientes, entre las medias marginales estimadas. Estas pruebas se basan en las comparaciones por pares linealmente independientes entre las medias marginales estimadas.

Para finalizar el análisis, se llevó a cabo la verificación de supuestos del modelo ANOVA. Como suele ser una práctica habitual, dicha comprobación se realizó sobre los residuales del modelo. En particular, se probó:

1) **Independencia de mediciones:** para demostrar este supuesto, se graficó el diagrama de dispersión del valor del residual vs el tiempo (figura 5.7). En dicho diagrama pudo observarse que no existía ningún patrón reconocible en el orden de la toma de mediciones, por lo que es una evidencia de su independencia.
2) **Homogeneidad de varianzas:** para probar este supuesto se empleó la prueba de Levene [6]. Como puede observarse en la tabla 5.11, el p-valor calculado permitió aceptar la hipótesis de la homogeneidad de varianzas.
3) **Normalidad:** se empleó la prueba de Shapiro-Wilk [7] para demostrar la normalidad en cada nivel de los factores. Como se observa en la tabla 5.12, a partir de los p-valores pudo demostrarse que las muestras se distribuían según la normal.

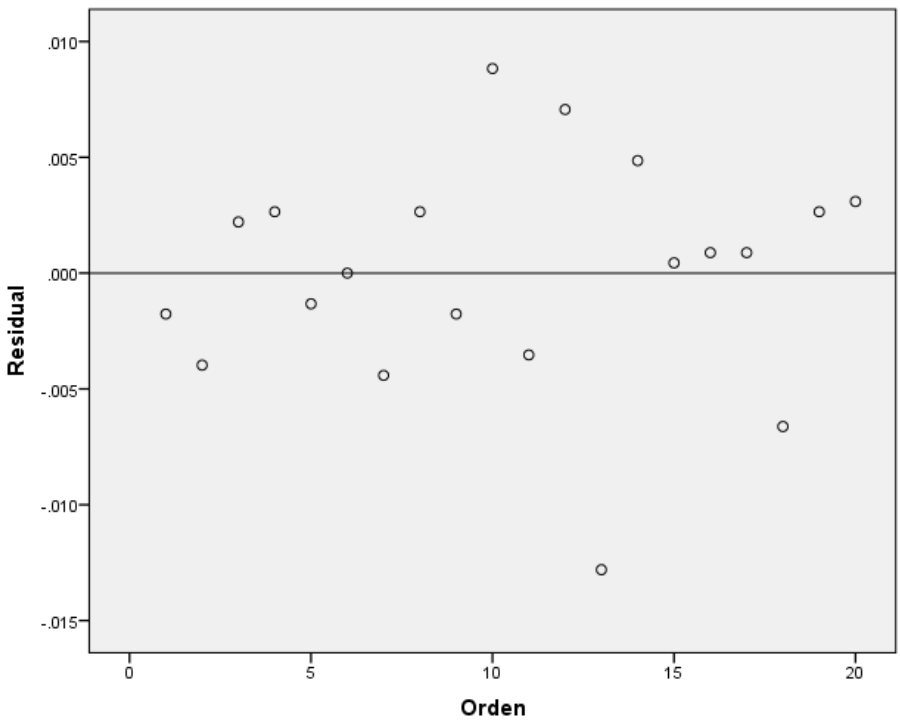


Figura 5.7. Diagrama de dispersión de residuales vs. Orden de recolección.

[6] La hipótesis nula de la prueba de Levene es que la varianza es la misma en todos los niveles.
[7] La hipótesis nula de la prueba de Shapiro-Wilk es que la muestra presenta una distribución normal.

Tabla 5.11.

Resultados prueba de Levene de homogeneidad de varianzas.

Contraste de Levene sobre la igualdad de las varianzas error[a]

Variable dependiente: acc

F	gl1	gl2	Sig.
1.016	3	16	.412

Contrasta la hipótesis nula de que la varianza error de la variable dependiente es igual a lo largo de todos los grupos.

a. Diseño: Intersección + A + B + A * B

Tabla 5.12.

Resultados de la prueba de Shapiro-Wilk de bondad de ajuste a la normal.

Pruebas de normalidad

			Kolmogorov-Smirnov[a]			Shapiro-Wilk		
A	B		Estadístico	gl	Sig.	Estadístico	gl	Sig.
Densenet	Debil_R	Residuo para acc	.167	5	.200*	.964	5	.833
	Fuerte_R	Residuo para acc	.237	5	.200*	.961	5	.814
Resnet	Debil_R	Residuo para acc	.323	5	.095	.828	5	.133
	Fuerte_R	Residuo para acc	.287	5	.200*	.914	5	.490

*. Este es un límite inferior de la significación verdadera.

a. Corrección de la significación de Lilliefors

A través de la experimentación se pudo definir el uso de la arquitectura Densenet. Sin embargo, no fue concluyente en cuanto al nivel de regularización aplicado a la capa densa. Para seleccionar cuál de los niveles de este factor debía ser empleado, se recurrió a las curvas de entrenamiento. La curva para un grado de regularización alto se muestra en la figura 5.8. La figura 5.9 hace lo propio para el nivel bajo. Como puede observarse en la figura 5.8, en el nivel alto del factor ocurre un exceso de regularización, a tal punto que el error del conjunto de entrenamiento termina siendo ligeramente mayor al de validación. Por el contrario, al usar una regularización débil, la curva se comporta más como es esperado: con un error de entrenamiento menor que el de validación, pero cercano a éste. Por este motivo, se decidió usar una regularización débil para el entrenamiento del clasificador final[8].

[8] Si bien era posible usar la arquitectura con regularización fuerte, se optó por la otra dado que no exhibía el comportamiento extraño observado con la primera (error de validación menor al de entrenamiento). El uso del otro esquema puede ser trabajado más adelante.

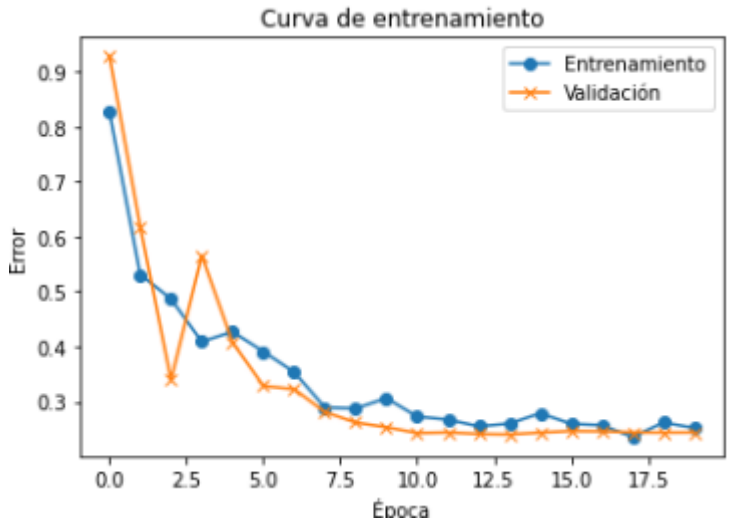


Figura 5.8. Curva de entrenamiento para Densenet con regularización fuerte en capa densa.

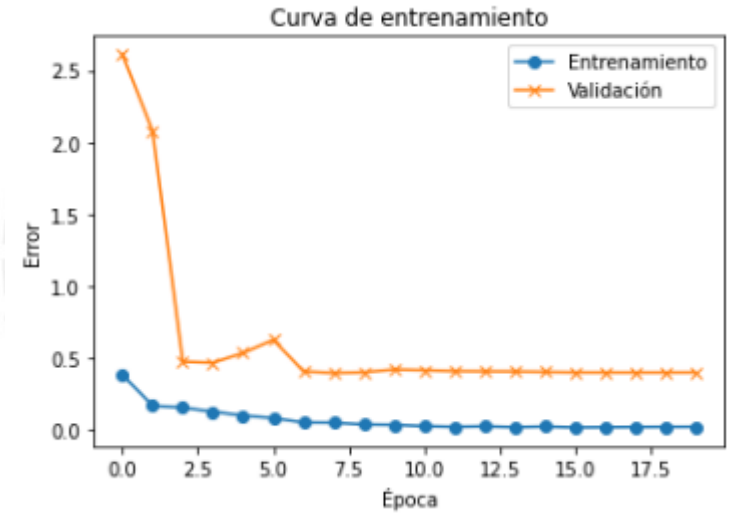


Figura 5.9. Curva de entrenamiento para Densenet con regularización débil en capa densa.

5.7. Entrenamiento del clasificador

Para tener una medida del grado de generalización del modelo, previo a su entrenamiento final, se procedió a entrenarlo sobre todo el conjunto de entrenamiento y evaluarlo sobre el de pruebas. Al igual que durante la experimentación, se entrenó el modelo en 20 épocas usando lotes de tamaño 32. Se usó un optimizador Adam con entropía cruzada como función de costo. Cabe mencionar que luego de cada época se hizo el guardado del modelo. Como puede verse en la figura 5.10, el modelo deja de aprender alrededor de la época 8, con un error mínimo de validación en la quinta época. Por tal motivo, se seleccionó el modelo guardado en la época 5 como clasificador final. El resultado obtenido con este sobre el conjunto de pruebas fue una exactitud del 95%.

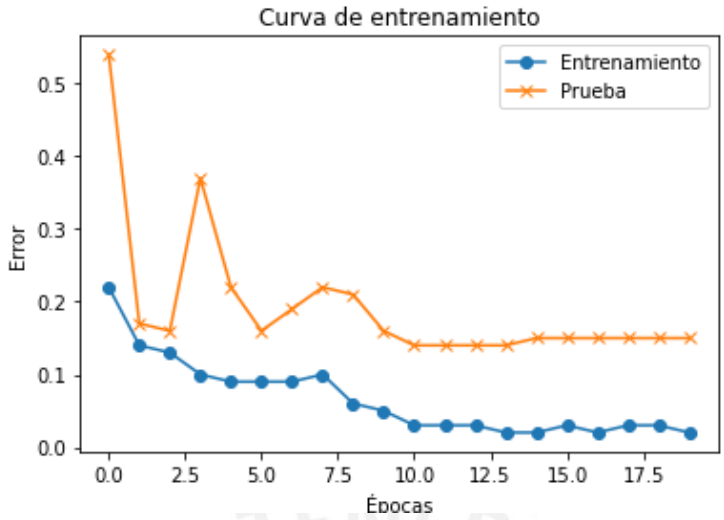


Figura 5.10. Curva de entrenamiento.

Luego del entrenamiento del clasificador se encontró que había ocurrido una actualización de RepeatsDB. Esta actualización ocurrió a dos niveles: interfaz y contenido. Como consecuencia, la cantidad de regiones tipo IV se incrementó. Se tomó una muestra aleatoria de 658 regiones nuevas encontradas y se procedió a clasificarlas. Como resultado, el conjunto de pruebas aumentó de tamaño y la exactitud se redujo a 89.83%. La matriz de confusión respectiva se muestra en la tabla 5.13. Como puede observarse, se clasifica muy bien las subclases IV1 y IV2; sin embargo, el desempeño con las subclases IV4 y OTROS es menor. Probablemente sea posible mejorar dicho desempeño entrenando el clasificador con todas las regiones incorporadas en la actualización de RepeatsDB; sin embargo, realizar tal labor atrasaría el proyecto, por lo que quedó pendiente como trabajo a futuro.

Tabla 5.13.

Matriz de confusión (conjunto de pruebas + enriquecimiento)

		Predicción			
		IV1	**IV2**	**IV4**	**OTROS**
Real	**IV1**	441	0	0	18
	IV2	0	208	1	4
	IV4	0	2	266	71
	OTROS	13	3	10	163

5.8. Análisis de contribución de los canales

Con la finalidad de validar la contribución de cada canal para la predicción, se tomo una muestra aleatoria de 1250 cadenas. Para cada una de las entradas seleccionadas, se construyeron 4 imágenes: la imagen RGB construida como se describió anteriormente y 3 imágenes en escala de grises, cada una de las cuales correspondía a uno de los canales de la imagen. De esta manera, se crearon 4 conjuntos de datos:

- **Conjunto 1 (C1):** solo matriz A
- **Conjunto 2 (C2):** solo matriz B
- **Conjunto 3 (C3):** solo matriz C
- **Conjunto 4 (C4):** combinación ABD

Cada uno de ellos fue posteriormente dividido en 3 subconjuntos (entrenamiento, validación y prueba) usando la misma proporción que se usó para el entrenamiento del clasificador (64/16/20). Es importante destacar que la división fue la misma para cada uno de los 4 conjuntos; es decir, por ejemplo, el subconjunto de entrenamiento de C1 tenía exactamente las mismas cadenas que C2. Se llevó a cabo la división de esta manera con la finalidad de asegurar que los resultados sean comparables.

Posteriormente, se entrenaron 4 clasificadores con la arquitectura seleccionada (Densenet, baja regularización). Cada uno se entrenó con un subconjunto de entrenamiento diferente. Luego, mediante el uso de las curvas de entrenamiento, se identificó la mejor instancia de cada uno. Estas últimas fueron usadas, finalmente, para hacer predicción sobre el conjunto de prueba y así calcular la exactitud en cada caso. Los resultados se resumen en la tabla 5.14., donde se puede ver claramente que si bien se puede obtener buenos resultados con cada canal independiente, la mejor métrica se obtiene cuando éstos se combinan en una sola imagen.

Tabla 5.14.

Exactitud medida para cada conjunto (evaluada sobre grupo de prueba).

Conjunto	Matriz	Exactitud
C1	Distancias	89%
C2	Correlación cruzada	91%
C3	Energía VdW	93%
C4	Combinación	97%

5.9. Conclusión

En el presente capítulo se describió el proceso seguido para la creación de los elementos necesarios para construir la capa lógica de la herramienta. Específicamente, se llevó a cabo un análisis comparativo de potenciales arquitecturas que culminó en el diseño de un flujo entre dos redes neuronales. En este flujo se usó una red convolucional que procesa mapas electrónicos tridimensionales como filtro capaz de detectar de forma indirecta la presencia de repeticiones clase IV en la estructura. Luego, una segunda red neuronal fue usada para llevar a cabo la clasificación de la estructura en una de 4 etiquetas (subclases): IV1, IV2, IV4 y OTROS (para todas las otras clases y subclases).

Se evaluó todas las estructuras disponibles en el banco de datos de proteínas (PDB) y a todas las estructuras de regiones clase IV encontradas en RepeatsDB para establecer un umbral de detección que funcione con la primera red neuronal. Con el umbral encontrado se obtuvo una sensibilidad de 80% y un universo potencial de aproximadamente 130 mil repeticiones.

Por otro lado, para el entrenamiento del clasificador, se construyó un conjunto de desarrollo por medio de la transformación de los archivos PDB recolectados en la librería de repeticiones en imágenes. Para realizar la transformación, se calcularon tres matrices: distancias interatómicas, fuerzas de Van Der Waals y de correlación cruzada entre modos normales de un modelo anisotrópico. Este conjunto de desarrollo se separó en tres subconjuntos: entrenamiento, validación y pruebas.

Para la selección de modelo, se realizó una experimentación numérica con un diseño factorial completo 2^2. En el diseño, se evaluaron dos factores: la arquitectura de la capa convolucional (CNN) y el grado de regularización que se introdujo en la capa final (densa). Las dos arquitecturas convolucionales comparadas fueron Resnet y Densenet. Los resultados demostraron que la red DenseNet con un grado de regularización bajo era la alternativa más adecuada para completar la tarea. Luego de entrenar el clasificador con el conjunto de entrenamiento, se obtuvo una exactitud de 95% sobre el conjunto de pruebas. Finalmente, se entrenó el clasificador sobre todo el conjunto de desarrollo, por lo que se espera un desempeño igual o mejor al medido.

Por último, se comprobó que la aplicación del clasificador sobre una imagen con tres canales tiene mejor resultado que su aplicación sobre imágenes en escala de grises.

Capítulo 6

Servicio Web

En el presente capítulo se describirá el proceso de análisis, diseño, implementación e implantación de la herramienta, en forma de servicio web.

6.1. Generalidades

En este capítulo se describirá en detalle los pasos seguidos para cumplir el tercer objetivo específico: implementar un servicio web para la automatización de la identificación y clasificación de regiones de repetición de clase IV a partir de la estructura. Como se recordará del capítulo 1, este objetivo específico tiene 2 resultados esperados: el diseño del servicio y la implementación de este. A continuación, se revisará a profundidad cada aspecto relevante para completarlo.

6.2. Análisis

El primer paso en el análisis fue identificar los requerimientos de la herramienta. Estos fueron resumidos en el catálogo de requisitos, el cual se puede apreciar en la tabla 6.1.

Tabla 6.1.

Catálogo de requisitos. (F) requisito funcional. (NF) requisito no funcional.

#	Requerimiento	Tipo	Pr.
1	La aplicación deberá recibir un código identificador de PDB (4 letras) y un identificador de cadena (1 letra) y retornar si la proteína en cuestión es de subclase IV1, IV2, IV4 u otra clase.	F	1
2	La aplicación deberá validar que el usuario ingrese el identificador PDB y el identificador de cadena.	F	1
3	La aplicación deberá validar que el identificador PDB conste de 4 letras.	F	2
4	La aplicación deberá validar que el identificador de cadena corresponda a 1 letra del alfabeto (no numérico).	F	1
5	La aplicación deberá visitar una página con la información de contacto de los participantes del proyecto.	F	2
6	La aplicación deberá permitir visitar la página con información del proyecto.	F	3
7	La aplicación debe ser ejecutada en una sola página.	F	3

#	Requerimiento	Tipo	Pr.
8	La aplicación deberá retornar la clasificación según RepeatsDB, si es que existía la entrada al momento de la implementación de la herramienta.	F	1
9	El front-end de la aplicación debe estar implementada en Java.	NF	1
10	El back-end de la aplicación debe estar desarrollada en Python.	NF	1
11	El front-end debe seguir un diseño minimalista.	NF	1
12	El front-end debe incluir indicaciones de cómo usar la herramienta.	NF	2
13	El texto de la página debe ser leíble a simple vista.	NF	2

Posteriormente, se identificaron los casos de uso. Para este caso, se identificaron 2 casos de uso: uno para clasificar las estructuras y otro para acceder a información de contacto y/o del proyecto.

Finalmente, se realizó la especificación de cada caso de uso. El primero de ellos es el de clasificación de estructura. El segundo de ellos fue la visualización de información de contacto y del proyecto. Estas especificaciones se resumen en las tablas 6.2 y 6.3, respectivamente.

Tabla 6.2.

Especificación de caso de uso 1: clasificar estructura.

Nombre	Clasificar estructura
Descripción	
Detección y clasificación de región clase IV en la estructura de una proteína	
Actores	
Usuario	
Precondiciones	
Información de proteína disponible en el banco de datos de proteínas (PDB).	
Flujo normal	
1. El usuario ingresa a la página de la aplicación. 2. El usuario introduce el código de identificación de la proteína según PDB (4 letras). 3. El usuario introduce el identificador de la cadena que se va a evaluar. 4. El usuario pulsa el botón Clasificar para enviar el formulario. 5. La aplicación devuelve la respuesta en la misma página del formulario.	
Flujo alternativo 1	
1. El usuario ingresa a la página de la aplicación. 2. El usuario pulsa el botón clasificar sin especificar el identificador de proteína o cadena. 3. La aplicación muestra un mensaje de error encima de cada campo faltante.	
Flujo alternativo 2	
1. El usuario ingresa a la página de la aplicación. 2. El usuario ingresa un identificador de proteína o cadena que no existe.	

3. La aplicación muestra un mensaje de error en el campo donde debería mostrar la salida.
Postcondición
La aplicación muestra la clase y subclase de la región identificada en la cadena.

Tabla 6.3.

Especificación de caso de uso: ver información adicional.

Nombre	Ver información
Descripción	
Visualizar información del proyecto/contactos.	
Actores	
Usuario	
Precondiciones	
Ninguna.	
Flujo normal	
1. El usuario ingresa a la página de la aplicación. 2. El usuario selecciona la opción "Contacto" y la subcategoría "Contacto" o "Proyecto". 3. La aplicación redirige a la página con la información de interés.	
Postcondición	
Ninguna	

6.3. Diseño

6.3.1. Diseño de la interfaz gráfica

Al ser una aplicación de una página, el diseño de la interfaz consistió en la construcción de un prototipo no funcional de la misma. Este fue construido tomando en consideración los requerimientos identificados en la etapa de análisis. La imagen 6.1 muestra dicho prototipo.

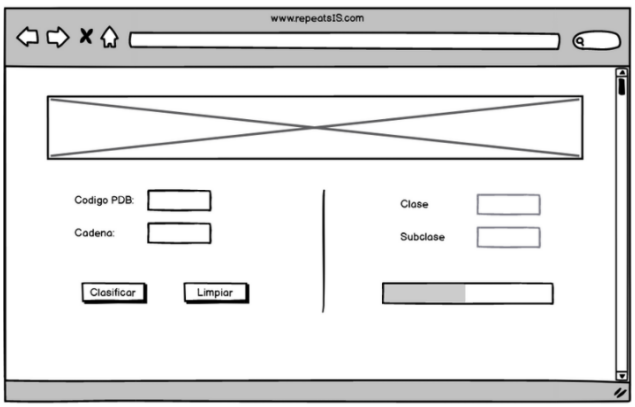


Figura 6.1. Prototipo inicial de la página web

6.3.2. Diseño de la capa lógica

Debido a la simplicidad de la funcionalidad requerida, se decidió trabajar la capa lógica con un micro marco de trabajo de Python llamado Flask. En vista que las dos redes neuronales requeridas para la detección y clasificación fueron entrenadas en versiones distintas de Tensorflow, fue necesario trabajar con dos servicios web: uno para el preprocesamiento y detección (DeepSym fue entrenada en Tensorflow 1.x); y otro para la clasificación (entrenada en Google Collab, que trae Tensorflow 2.x). Para usar dos versiones distintas del mencionado marco de trabajo en un mismo sistema fue necesario encapsular las librerías necesarias para cada servicio en dos entornos virtuales. La figura 6.2 describe el funcionamiento de la capa lógica a través de su diagrama de componentes.

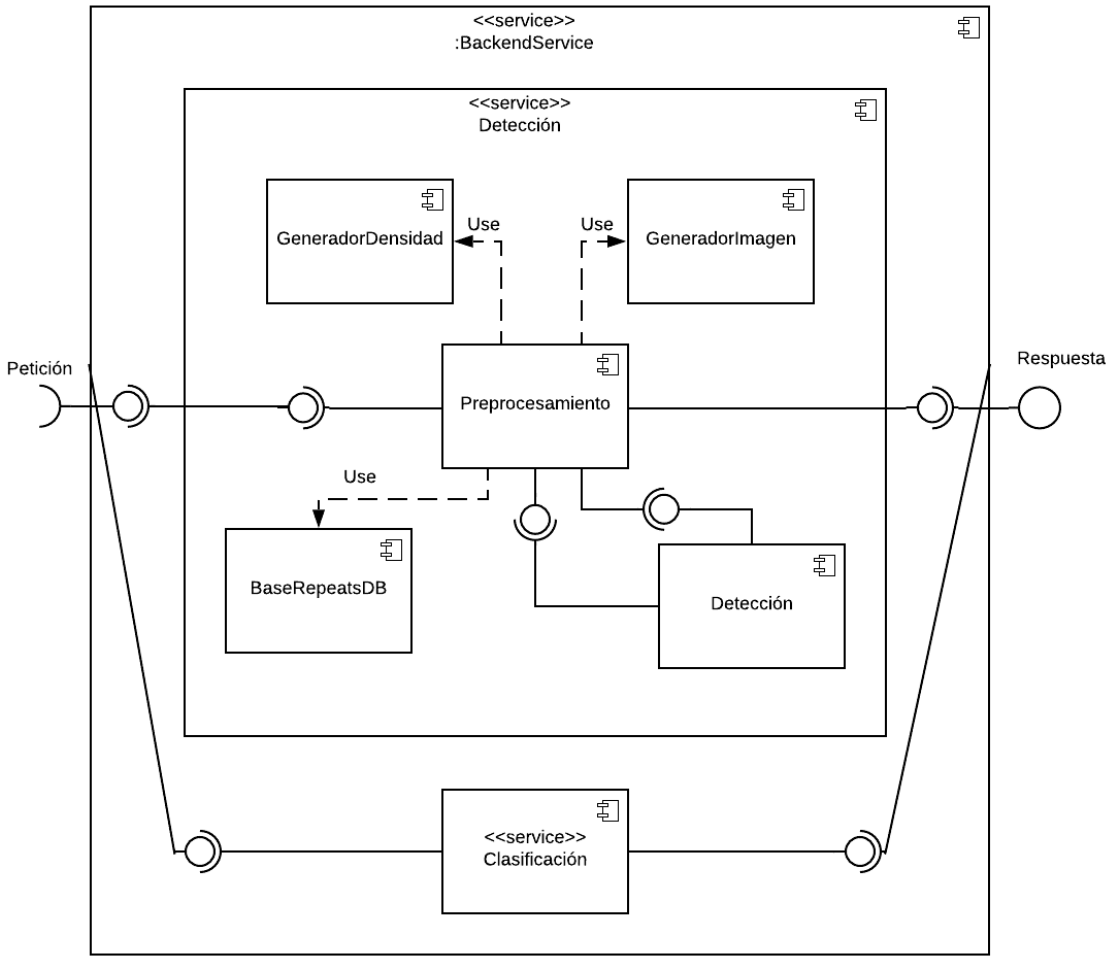


Figura 6.2. Diagrama de componentes de la capa lógica.

Como puede verse en el diagrama de componentes, el primero de los dos servicios es el más complejo, ya que encapsula la lógica de preprocesamiento y detección. En líneas generales, el módulo de preprocesamiento busca el identificador que recibe de la

petición en una base de regiones conocidas en RepeatsDB. De existir el registro ahí, se devuelve directamente la clasificación. Caso contrario, se ejecuta el generador de densidad y luego el detector. A continuación, se revisa la respuesta del detector. Si este indica que no hay órdenes de simetría detectados (es decir, no hay región de repetición clase IV), se devuelve dicha respuesta. Caso contrario, se procede a transformar la estructura (archivo PDB) en imagen y se devuelve una señal de pendiente como respuesta.

El segundo servicio (el de clasificación) es mucho más sencillo: le corresponde únicamente ejecutar el clasificador y devolver la etiqueta en la respuesta.

6.3.3. Integración

Para entender mejor la interacción entre la interfaz y la capa lógica, se incluye el diagrama de secuencia en la figura 6.3.

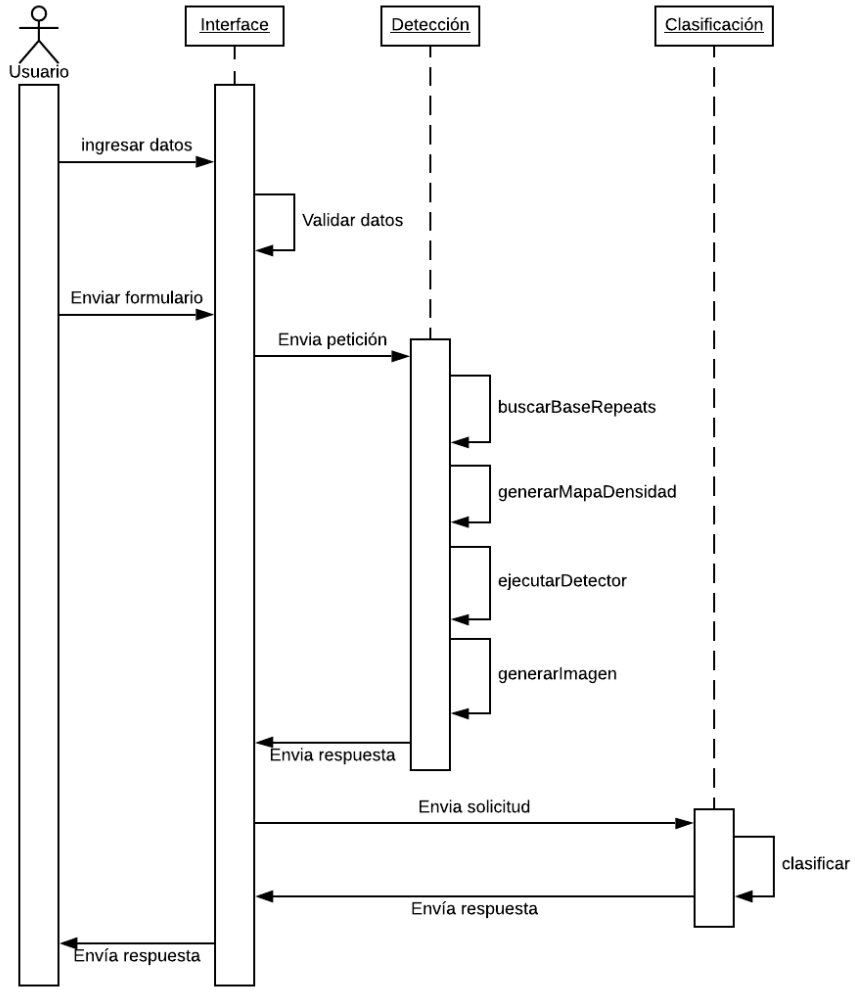


Figura 6.3. Diagrama de secuencia del servicio web.

Como puede observarse en el diagrama de secuencia, el flujo comienza cuando el usuario ingresa los datos de la cadena de proteína (identificador PDB y cadena) en el formulario. Este formulario incluye un botón con la etiqueta “Clasificar”. Al hacer clic sobre dicho elemento, se gatilla la validación de los datos y el envío del formulario. Una vez enviado el formulario, se hace un llamado al servicio web de detección (y preprocesamiento). Una vez ejecutado, si no devuelve una señal de pendiente, se retorna a la interfaz la respuesta de este módulo. Caso contrario, se utiliza el servicio del clasificador para obtener la subclase de la cadena. Finalmente, se devuelve a la interfaz este resultado.

6.4. Implementación

Para la implementación de la interfaz gráfica se empleó el lenguaje de programación Java con tecnología de servlets y siguiendo un modelo MVC (modelo, vista y controlador). La figura 6.4 muestra una captura de la pantalla principal del servicio web.

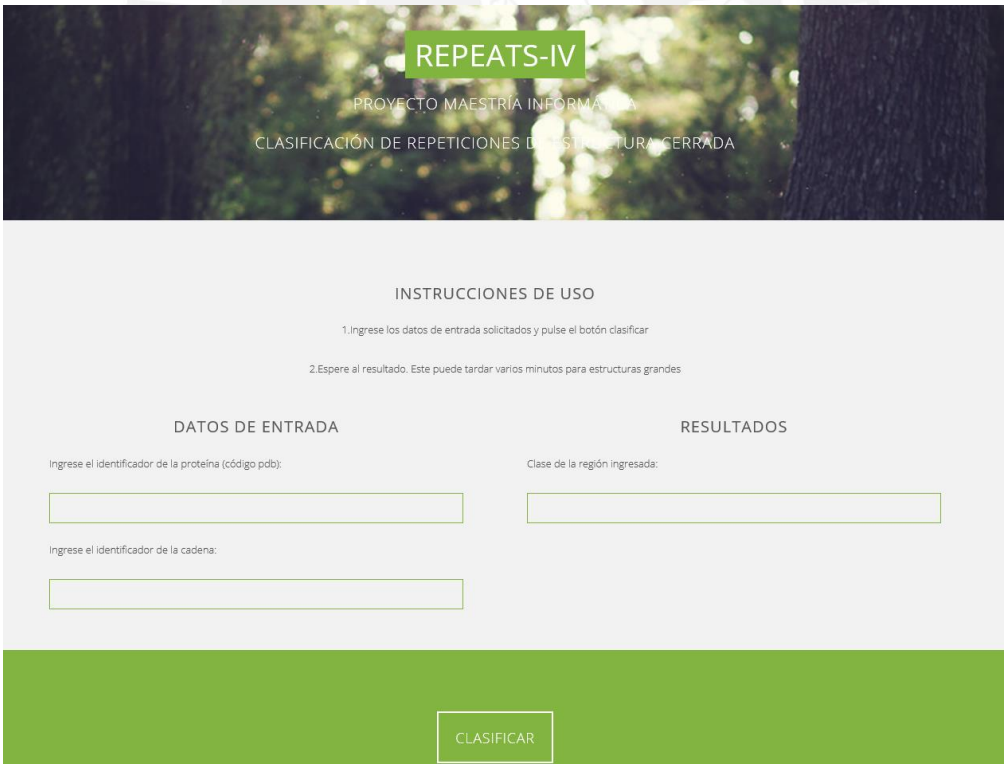


Figura 6.4. Captura de pantalla de la interfaz del servicio web.

La capa lógica fue implementada con el lenguaje Python y la librería Flask. Como se mencionó, se trabajó con entornos virtuales.

6.5. Pruebas

Una vez implementada la herramienta, se llevó a cabo un conjunto de pruebas unitarias y de integración. El detalle de este plan de pruebas, así como los resultados obtenidos en cada una de ellas, puede ser analizado minuciosamente en el anexo 3 del documento. A modo de resumen, se definieron 7 pruebas y la herramienta superó cada una de ellas. La tabla 6.4 muestra un ejemplo del formato seguido en el anexo 3 para especificar cada caso de prueba.

Tabla 6.4.

Ejemplo de documentación de caso de prueba.

ID Prueba	1
Caso uso	Clasificación
Flujo	Normal
Pasos	
1. Ingresar a la página de la aplicación.	
2. Introducir "1LM1" en el campo del identificador de proteína.	
3. Introducir "A" en el campo del identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
IV.1	
Resultado Obtenido	
IV.1	
Pasa la prueba	Sí

Adicionalmente, se llevó a cabo una evaluación de la herramienta aplicándola sobre una muestra aleatoria de 900 cadenas que se encontraron en la actualización de RepatsDB. De ellas, el 26% no pasó el detector y, sobre el universo restante se obtuvo una exactitud de 80%. La matriz de confusión sobre este universo se incluye en la tabla 6.5.

Tabla 6.5.

Matriz de confusión (muestreo que pasó el detector)

		Predicción			
		IV1	**IV2**	**IV4**	**OTROS**
Real	**IV1**	292	0	0	18
	IV2	0	66	1	3
	IV4	0	2	128	64
	OTROS	13	2	4	39

6.6. Implantación

Para poner en producción la herramienta, se creó una máquina virtual en la plataforma Cloud Engine de Google. Específicamente, se configuró una máquina con GPU. En ella, se reprodujo toda la configuración que había en la máquina local y se copiaron todos los archivos necesarios para la ejecución de la herramienta. Esta tarea fue facilitada con los entornos virtuales, ya que en ellos se encontraban empaquetados todas las librerías requeridas para la capa lógica. En cuestión de servidores, la web se montó sobre un servidor Apache Tomcat. Por el contrario, Flask provee de un servidor de desarrollo propio para desplegar sus servicios. Si bien es cierto que el mencionado servidor no tiene mucha capacidad de escalar, para el público objetivo de la aplicación -altamente especializado- resulta suficiente. Dicho servidor fue configurado para escuchar en los puertos 9090 y 9095 (para la detección y clasificación, respectivamente). El despliegue se ilustra en la figura 6.5.

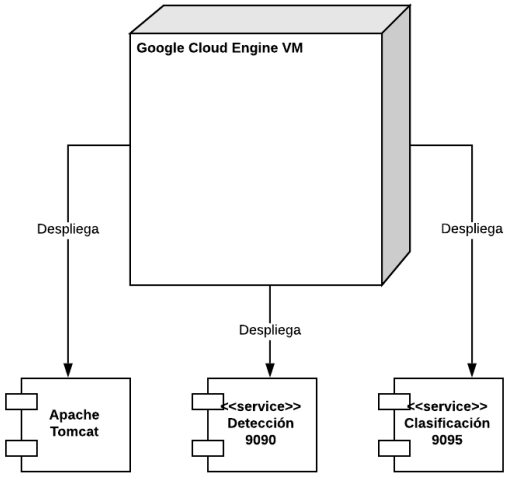


Figura 6.5. Diagrama de despliegue de la aplicación

Cabe mencionar que, de momento, la máquina virtual permanece apagada debido a que su uso tiene un costo que no estaba incluido en el presupuesto del proyecto porque su alcance contemplaba únicamente tener la herramienta en un ambiente previo al de producción.

6.7. Conclusión

En el presente capítulo se describió el proceso seguido para integrar todos los subproductos generados en los capítulos anteriores en una sola herramienta que permita realizar la clasificación de repeticiones en estructuras de proteínas cerradas.

Para ello, se identificaron los requerimientos y se realizó el diseño en base a ellos. Se implementó la interfaz gráfica empleando Java con un modelo básico MVC. Asimismo, se implementó la capa lógica usando el micro marco de trabajo Flask, de Python. Esta capa estuvo formada por dos servicios: uno que realizaba las tareas de preprocesamiento y detección; y otro que realizaba la tarea de clasificación. Una vez integrada la interfaz gráfica con los servicios web, se llevó a cabo un conjunto de pruebas, las cuales fueron superadas al 100% por la herramienta. Finalmente, se desplegó el servicio en un ambiente previo al de producción en la nube de Google.

Capítulo 7

Aspectos finales

7.1. Resumen

Las proteínas repetidas son proteínas globulares que presentan repeticiones a nivel de secuencia y estructura. Participan en una diversidad de funciones biológicas, aunque también juegan un rol en el desarrollo de enfermedad. Por consiguiente, su estudio cobra relevancia para la formulación de posibles tratamientos. Parte importante de dicho estudio es la posibilidad de identificarlas y clasificarlas. La realización de esta tarea es inviable para el volumen de información que se maneja en el banco de datos de proteínas, por lo que su automatización es casi imprescindible.

La búsqueda en la literatura mostró que existen dos tendencias para esta automatización: usar la información de secuencia y de estructura. Con frecuencia, la primera no es suficiente debido al alto grado de degeneración que puede encontrarse en las proteínas. El uso de información estructural es mejor debido a que existe una relación directa entre estructura y funcionalidad. Sin embargo, los cálculos necesarios para realizar una clasificación empleando esta información son muy costosos computacionalmente.

A la fecha de realización de la búsqueda sistemática no se encontró ninguna herramienta que realizara la identificación y clasificación de estructuras de proteínas repetidas de clase IV empleando redes neuronales. Por consiguiente, en el presente trabajo se investigó su adaptación para la clasificación de este tipo de proteínas.

Para llevar a cabo los distintos análisis que se tenía previsto realizar, fue necesario recopilar la información estructural de proteínas repetidas de RepeatsDB. Esto condujo a la creación de una librería de estructuras de regiones de repetición en formato PDB. Dicha librería estuvo formada por 3684 regiones de repetición de clase IV y 2560 regiones de clase III.

Posteriormente, se llevó a cabo un análisis comparativo para seleccionar la arquitectura a usar. Como resultado, se optó por utilizar un esquema de dos partes: un detector que actuara como filtro y un clasificador. Para el primer elemento se vio conveniente reutilizar

una red que detectaba simetrías rotacionales en la estructura de una cadena de proteína. Esta decisión se sustentó en el hecho que como las regiones de clase IV son de estructura cerrada, la probabilidad que presenten varios órdenes de simetría rotacional era alta. Para adaptar esta red al presente caso de estudio fue necesario identificar un umbral de detección. Para ello, fue necesario evaluar todas las proteínas disponibles en el banco de datos de proteínas y en RepeatsDB. El umbral se terminó por definir en función de maximizar la sensibilidad sobre las cadenas de RepeatsDB y minimizar el universo potencial encontrado al aplicar dicho umbral sobre el resto de las estructuras del banco de datos de proteínas.

El segundo elemento del esquema anteriormente mencionado fue un clasificador que recibe como entrada una imagen y devuelve la clase. Como resulta evidente, para entrenar dicho clasificador, fue necesario transformar toda la información estructural contenida en la librería de clase IV en imágenes. Para realizar esta transformación, se emplearon tres matrices que actuaron como canales RGB de una imagen: la matriz de distancias interatómicas, la matriz de energías de enlace no covalente (Van Der Waals) y la matriz de covariación resultante de un análisis de modos normales de un modelo anisotrópico.

Para la selección de la arquitectura de clasificación de imágenes a usar, se llevó a cabo una experimentación numérica con los conjuntos de entrenamiento y validación. Como resultado, se obtuvo que una red Densenet era la mejor de las alternativas. En consecuencia, se entrenó un clasificador que tuviera una capa convolucional Densenet seguida por una capa densa regularizada por dropout y regularización por lotes. La capa convolucional fue usada como extractor de características; es decir, se bloqueó su entrenamiento. Solo se entrenó la capa densa. El resultado final fue una red con una exactitud del 95% sobre el conjunto de prueba (se aisló un 20% previo a la etapa de selección de modelo).

Finalmente, se construyó un servicio web cuya función era recibir un identificador PDB y un identificador de cadena y devolver como respuesta si existe la presencia de una repetición de clase IV y su subclase. Para la construcción de la interfaz gráfica se hizo uso del lenguaje de programación Java con una arquitectura MVC. Para la capa lógica, se empleó el lenguaje Python y el micro marco de trabajo Flask. Esta aplicación fue puesta en un entorno de pre-producción disponible en la nube de Google.

7.2. Trabajos futuros

A partir de los resultados del trabajo, es posible encontrar dos puntos de mejora que podrían ser abordados en trabajos futuros. El primero de ellos está relacionado directamente con el elemento detector. La red neuronal usada detectaba, según sus autores, simetrías de tipo rotacional. Existen otros tipos de simetría que podrían ser detectadas usando otras redes neuronales. Asimismo, el establecimiento del umbral de detección fue realizado de manera manual, a prueba y error. Un trabajo futuro podría abordar dicho problema empleando un enfoque automático. Una posibilidad es aplicar métodos de detección de anomalías.

El segundo punto de mejora es el mismo clasificador, pues se realizó una actualización de RepeatsDB posterior a la fecha en que se terminó de entrenar el clasificador. En dicha actualización se aumentó aproximadamente 4 mil regiones de clase IV adicionales que podrían ser usadas para mejorar el desempeño del clasificador. Un trabajo futuro puede repetir los pasos descritos en el capítulo 5 para identificar la mejor arquitectura y reentrenar el clasificador.

Referencias

Alpaydin, E. (2014). *Introduction to machine learning* (Third edition). *Adaptive computation and machine learning*. Cambridge, Massachusetts: The MIT Press.

AlQuraishi, M. (2019). End-to-End Differentiable Learning of Protein Structure. *Cell systems*, *8*(4), 292-301.e3. https://doi.org/10.1016/j.cels.2019.03.006

Andrade, M. A., Petosa, C., O'Donoghue, S. I., Müller, C. W. y Bork, P. (2001). Comparison of ARM and HEAT protein repeats. *Journal of Molecular Biology*, *309*(1), 1–18. https://doi.org/10.1006/JMBI.2001.4624

Andrade, M. A., Ponting, C. P., Gibson, T. J. y Bork, P. (2000). Homology-based method for identification of protein repeats using statistical significance estimates. *Journal of Molecular Biology*, *298*(3), 521–537. https://doi.org/10.1006/JMBI.2000.3684

Atkins, P. W. y Cwi, S. (2008). *Química física* (8a ed.). Buenos Aires, Madrid: Editorial Médica Panamericana.

Basheer, I. y Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, *43*(1), 3–31. https://doi.org/10.1016/S0167-7012(00)00201-3

Berg, J. M., Tymoczko, J. L. y Stryer, L. (2008). *Bioquímica* (6ª ed.). Barcelona: Reverté.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., … Bourne, P. E. (2000). The Protein Data Bank. *Nucleic acids research*, *28*(1), 235–242. https://doi.org/10.1093/nar/28.1.235

Biegert, A. y Söding, J. (2008). De novo identification of highly diverged protein repeats by probabilistic consistency. *Bioinformatics (Oxford, England)*, *24*(6), 807–814. https://doi.org/10.1093/bioinformatics/btn039

Bird, S., Klein, E. y Loper, E. (2009). *Natural language processing with Python* (1. edition). Beijing, Köln: O'Reilly.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.

Chakrabarty, B. y Parekh, N. (2014). PRIGSA: protein repeat identification by graph spectral analysis. *Journal of bioinformatics and computational biology*, *12*(6), 1442009. https://doi.org/10.1142/S0219720014420098

Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M., Ferguson, D. M., … Kollman, P. A. (1995). A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *Journal of the American Chemical Society*, *117*(19), 5179–5197. https://doi.org/10.1021/ja00124a002

Di Domenico, T., Potenza, E., Walsh, I., Parra, R. G., Giollo, M., Minervini, G., … Tosatto, S. C. E. (2014). RepeatsDB: a database of tandem repeat protein structures. *Nucleic acids research*, *42*(Database issue), D352-7. https://doi.org/10.1093/nar/gkt1175

Doyle, L., Hallinan, J., Bolduc, J., Parmeggiani, F., Baker, D., Stoddard, B. L. y Bradley, P. (2015). Rational design of \alpha-helical tandem repeat proteins with closed architectures. *Nature*, *528*(7583), 585–588. https://doi.org/10.1038/nature16191

Eyal, E., Yang, L.-W. y Bahar, I. (2006). Anisotropic network model: systematic evaluation and a new web interface. *Bioinformatics (Oxford, England)*, *22*(21), 2619–2627. https://doi.org/10.1093/bioinformatics/btl448

Goodfellow, I., Begio, Y. y Courville, A. (2017). *Deep Learning*. *Adaptive Computation and Machine Learning Series*. Cambridge, Mass.: MIT Press Ltd.

Guo, H., Newaz, K., Emrich, S., Milenkovic, T. y Li Jun. (2019). *Weighted graphlets and deep neural networks for protein structure classification*. Recuperado de http://arxiv.org/pdf/1910.02594v1

Hayward, S. y Groot, B. L. de. (2008). Normal modes and essential dynamics. *Methods in molecular biology (Clifton, N.J.)*, *443*, 89–106. https://doi.org/10.1007/978-1-59745-177-2_5

Hirsh, L. (2018). Investigación de Layla Hirsh: El mundo de las proteínas repetidas. *Boletin Ingenieria Biomedica*, (5), 3–4.

Hirsh, L., Paladin, L., Piovesan, D. y Tosatto, S. C. E. (2018). RepeatsDB-lite: a web server for unit annotation of tandem repeat proteins. *Nucleic acids research*, *46*(W1), W402-W407. https://doi.org/10.1093/nar/gky360

Hirsh, L., Piovesan, D., Paladin, L. y Tosatto, S. C. E. (2016). Identification of repetitive units in protein structures with ReUPred. *Amino acids*, *48*(6), 1391–1400. https://doi.org/10.1007/s00726-016-2187-2

Jain, R., Yalamanchili, H. K. y Parekh, N. (2009, diciembre - 2009, diciembre). Identifying structural repeats in proteins using graph centrality measures. En *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)* (pp. 110–115). IEEE. https://doi.org/10.1109/NABIC.2009.5393609

Jorda, J., Baudrand, T. y Kajava, A. V. (2012). PRDB: Protein Repeat DataBase. *PROTEOMICS*, *12*(9), 1333–1336. https://doi.org/10.1002/pmic.201100534

Jorda, J. y Kajava, A. V. (2009). T-REKS: identification of Tandem REpeats in sequences with a K-meanS based algorithm. *Bioinformatics (Oxford, England)*, *25*(20), 2632–2638. https://doi.org/10.1093/bioinformatics/btp482

Jorda, J., Xue, B., Uversky, V. N. y Kajava, A. V. (2010). Protein tandem repeats - the more perfect, the less structured. *The FEBS journal*, *277*(12), 2673–2682. https://doi.org/10.1111/j.1742-464X.2010.07684.x

Kajava, A. V. (2001). Review: Proteins with Repeated Sequence—Structural Prediction and Modeling. *Journal of Structural Biology*, *134*(2-3), 132–144. https://doi.org/10.1006/JSBI.2000.4328

Kalita, M. K., Ramasamy, G., Duraisamy, S., Chauhan, V. S. y Gupta, D. (2006). ProtRepeatsDB: a database of amino acid repeats in genomes. *BMC bioinformatics*, *7*, 336. https://doi.org/10.1186/1471-2105-7-336

Karpenahalli, M. R., Lupas, A. N. y Söding, J. (2007). TPRpred: a tool for prediction of TPR-, PPR- and SEL1-like repeats from protein sequences. *BMC bioinformatics*, *8*, 2. https://doi.org/10.1186/1471-2105-8-2

Kelil, A., Wang, S. y Brzezinski, R. (2007, octubre - 2007, octubre). A New Alignment-Independent Algorithm for Clustering Protein Sequences. En *2007 IEEE 7th International Symposium on BioInformatics and BioEngineering* (pp. 27–34). IEEE. https://doi.org/10.1109/BIBE.2007.4375541

Lehninger, A. L., Nelson, D. L. y Cox, M. M. (2005). *Principios de bioquímica* (4ª ed.). Barcelona: Omega.

McMurry, J. (2010). *Organic chemistry* (7th enhanced ed.). Belmont CA: Cengage Brooks/Cole.

Newaz, K., Ghalehnovi, M., Rahnama, A., Antsaklis, P. J. y Milenković, T. (2020). Network-based protein structural classification. *Royal Society open science*, *7*(6), 191461. https://doi.org/10.1098/rsos.191461

Pagès, G. y Grudinin, S. (2018). *DeepSymmetry : Using 3D convolutional networks for identification of tandem repeats and internal symmetries in protein structures*. Recuperado de http://arxiv.org/pdf/1810.12026v1

Paladin, L., Hirsh, L., Piovesan, D., Andrade-Navarro, M. A., Kajava, A. V. y Tosatto, S. C. E. (2017). RepeatsDB 2.0: improved annotation, classification, search and visualization of repeat protein structures. *Nucleic acids research*, *45*(D1), D308-D312. https://doi.org/10.1093/nar/gkw1136

Palidwor, G. A., Shcherbinin, S., Huska, M. R., Rasko, T., Stelzl, U., Arumughan, A., … Andrade-Navarro, M. A. (2009). Detection of alpha-rod protein repeats using a neural network and application to huntingtin. *PLoS computational biology*, *5*(3), e1000304. https://doi.org/10.1371/journal.pcbi.1000304

Pellegrini, M. (2015). Tandem Repeats in Proteins: Prediction Algorithms and Biological Role. *Frontiers in bioengineering and biotechnology*, *3*, 143. https://doi.org/10.3389/fbioe.2015.00143

Provost, F. y Fawcett, T. (2013). *Data science for business*. Sebastopol, CA: O'Reilly.

Sikosek, T. (2019). *Protein structure featurization via standard image classification neural networks.*

Söding, J., Remmert, M. y Biegert, A. (2006). HHrep: de novo protein repeat detection and the origin of TIM barrels. *Nucleic acids research*, *34*(Web Server issue), W137-42. https://doi.org/10.1093/nar/gkl130

Sommerville, I. (2002). *Ingeniería del software* (6a. ed.). México: Addison-Wesley.

Turjanski, P., Parra, R. G., Espada, R., Becher, V. y Ferreiro, D. U. (2016). Protein Repeats from First Principles. *Scientific reports*, *6*, 23959. https://doi.org/10.1038/srep23959

Vargas, I. (2013). Herramienta informática de apoyo a la escritura de resúmenes de textos científicos en español. (Tesis). Pontificia Universidad Católica del Perú, Peru.

Wade, L. G., Montaña Pedrero, Á. y Batalla García, C. (2004). *Química orgánica* (5ª ed.). Madrid: Pearson Prentice Hall.

Wafaa Wardah, M.G.M. Khan, Alok Sharma y Mahmood A. Rashid. (2019). Protein secondary structure prediction using neural networks and deep learning: A review. *Computational Biology and Chemistry*, *81*, 1–8. https://doi.org/10.1016/j.compbiolchem.2019.107093

Walsh, I., Sirocco, F. G., Minervini, G., Di Domenico, T., Ferrari, C. y Tosatto, S. C. E. (2012). RAPHAEL: recognition, periodicity and insertion assignment of solenoid protein structures. *Bioinformatics (Oxford, England)*, *28*(24), 3257–3264. https://doi.org/10.1093/bioinformatics/bts550

Anexo 1.

Diccionario del EDT

Componente	Descripción				
Código del paquete de trabajo	1.1				
Descripción del paquete de trabajo	Obtención de la información estructural de las proteínas en el banco de datos de proteínas (PDB)				
Entregable(s)	Catálogo PDB				
Criterios de aceptación del entregable(s)	- Debe estar compuesto por todas las proteínas del PDB disponibles a la fecha de la extracción. - La información extraída debe cumplir la estructura estándar de un archivo PDB. - El nombre de los archivos debe seguir el siguiente formato: <id_proteina_pdb>.pdb. - El reporte debe indicar cuántas estructuras se llegó a extraer.				
Recursos	Tesista.				
Duración estimada	16 días	Fecha inicio:	15/04/2020	Fecha Fin:	30/04/2020
Hito asociado	Ninguno.				
Dependencias	Ninguna.				
Observaciones	Se utilizará la librería pypdb como apoyo.				
Tareas	-Obtener una lista de todas las proteínas disponibles en el banco de datos de proteínas por medio de su API Rest. -Implementar un script para descargar todas las proteínas del listado mediante el uso de la librería pypdb. -Ejecutar el script para descargar y guardar las estructuras extraídas.				

Componente	Descripción				
Código del paquete de trabajo	1.2				
Descripción del paquete de trabajo	Obtención de la información anotada de regiones y unidades de repetición almacenadas en RepeatsDB.				
Entregable(s)	Catálogo RepeatsDB				
Criterios de aceptación del entregable(s)	- La información extraída de RepeatsDB debe ser guardada en formato DB, - El reporte debe indicar cuántas estructuras llegó a extraerse a nivel de regiones y unidades.				
Recursos	Tesista				
Duración estimada	20 días	Fecha inicio:	01/05/2020	Fecha Fin:	20/05/2020
Hito asociado	Ninguno.				
Dependencias	Ninguna.				
Observaciones	-Para el desarrollo del webscrapper se usará como apoyo las librerías Selenium y Beautiful Soup de Python.				

	-Se estima un tiempo prolongado ya que es necesario pausar el *webscrapper* luego de cada iteración para evitar que la página bloquee la dirección IP por exceder el número de peticiones.
Tareas	-Implementar un *webscrapper* que recorra el buscador de RepeatsDB, entre a cada una de las cadenas y extraiga la información de las regiones y unidades de repetición en un archivo de extensión DB. -Ejecutar el *webscrapper* para extraer y guardar la información de las regiones de repetición anotadas en RepeatsDB.

Componente	Descripción				
Código del paquete de trabajo	1				
Descripción del paquete de trabajo	Extracción de información estructural de las regiones y unidades de repetición extraídas de RepeatsDB.				
Entregable(s)	Información estructural de regiones de repetición.				
Criterios de aceptación del entregable(s)	- La información de coordenadas de las regiones y unidades debe ser guardada en formato PDB. - El nombre de los archivos PDB debe seguir el siguiente patrón: <id_proteina>_<id_cadena>_<inicio>_<fin>_<S/N>.pdb. Nota: S=Sí tiene inserción, N= No tiene inserción.				
Recursos	Tesista.				
Duración estimada	51 días	Fecha inicio:	15/04/2020	Fecha Fin:	05/06/2020
Hito asociado	Información estructural de regiones y unidades de repetición disponible para su uso.				
Dependencias	1.1, 1.2				
Observaciones	Ninguna.				
Tareas	-Implementar script que lea información de la región de repetición y extraiga la información estructural de PDB. -Ejecutar el script.				

Componente	Descripción				
Código del paquete de trabajo	2.1				
Descripción del paquete de trabajo	Selección de la arquitectura del clasificador.				
Entregable(s)	Arquitectura				
Criterios de aceptación del entregable(s)	La arquitectura seleccionada debe estar sustentada por un análisis de las ventajas y desventajas que tiene.				
Recursos	Tesista				
Duración estimada	40 días	Fecha inicio:	06/06/2020	Fecha Fin:	16/07/2020
Hito asociado	Arquitectura propuesta seleccionada.				
Dependencias	Ninguna.				
Observaciones	Ninguna.				
Tareas	-Identificar propuestas de arquitecturas en literatura. -Realizar análisis comparativo entre arquitecturas.				

Componente	Descripción				
Código del paquete de trabajo	2.2				
Descripción del paquete de trabajo	Generación del conjunto de datos para entrenamiento y pruebas.				
Entregable(s)	Conjunto de datos				
Criterios de aceptación del entregable(s)	- La métrica de memoria (*recall*) del detector debe ser por lo menos de un 75%. - Las imágenes deben ser guardadas en formato jpg. - Los nombres de archivo de las imágenes del conjunto de entrenamiento deben tener el formato <id_proteina>_<id_cadena>_<inicio_region>_<fin_region>_<clase>_<subclase>.jpg				
Recursos	Tesista				
Duración estimada	129 días	Fecha inicio:	17/07/2020	Fecha Fin:	15/11/2020
Hito asociado	Conjunto de datos disponible.				
Dependencias	1.2, 1.3				
Observaciones	Ninguna.				
Tareas	-Evaluar todas las cadenas presentes en PDB que no están en RepeatsDB a través de la red Deep Symmetry. -Evaluar todas las regiones de repetición de RepeatsDB usando Deep Symmetry con la información estructural de las regiones extraída en el entregable 1. -Identificar un umbral de orden de simetría usando los resultados de Deep Symmetry sobre la información de RepeatsDB. -Implementar script para convertir la información de un archivo PDB a imagen 2D. -Aplicar script sobre cadenas que tengan potencialmente una repetición.				

Componente	Descripción				
Código del paquete de trabajo	2.3				
Descripción del paquete de trabajo	Entrenamiento del modelo de clasificación.				
Entregable(s)	Modelo de clasificación entrenado.				
Criterios de aceptación del entregable(s)	El modelo entrenado deberá ser entregado en un archivo como objeto serializado mediante la librería pickle de Python.				
Recursos	Tesista				
Duración estimada	15 días	Fecha inicio:	16/11/2020	Fecha Fin:	30/11/2020
Hito asociado	Ninguno.				
Dependencias	2.2				
Observaciones	Ninguna.				
Tareas	-Entrenar el modelo con el conjunto de datos de entrenamiento generado en 2.2. -Optimizar hiperparámetros empleando búsqueda en grilla.				

Componente	Descripción				
Código del paquete de trabajo	2				
Descripción del paquete de trabajo	Integración del detector (Deep Symmetry) y el clasificador.				
Entregable(s)	Clasificador				
Criterios de aceptación del entregable(s)	- Detector y clasificador deben estar integrados, de manera que el bloque funcione automáticamente con solo recibir los datos de entrada.				
Recursos	Tesista.				
Duración estimada	205 días	Fecha inicio:	06/06/2020	Fecha Fin:	22/12/2020
Hito asociado	Clasificadores entrenados con características escogidas.				
Dependencias	2.2, 2.3				
Observaciones	Ninguna.				
Tareas	-Integrar la salida de Deep Symmetry con el clasificador. -Aplicar solución integrada sobre el conjunto de pruebas.				

Componente	Descripción				
Código del paquete de trabajo	3.1				
Descripción del paquete de trabajo	Prototipado de la interfaz de usuario del servicio web.				
Entregable(s)	Diseño del servicio web.				
Criterios de aceptación del entregable(s)	-El prototipo debe cumplir con los requisitos de usabilidad identificados.				
Recursos	Tesista				
Duración estimada	17 días	Fecha inicio:	15/10/2020	Fecha Fin:	31/10/2020
Hito asociado	Ninguno.				
Dependencias	Ninguna.				
Observaciones	Ninguna.				
Tareas	-Identificar los requisitos funcionales y no funcionales del servicio. -Realizar el prototipado del servicio.				

Componente	Descripción				
Código del paquete de trabajo	3.2				
Descripción del paquete de trabajo	Implementación de la interfaz de usuario del servicio web.				
Entregable(s)	Implementación del servicio web.				
Criterios de aceptación del entregable(s)	-La implementación debe respetar el prototipo (3.1). -La implementación debe satisfacer todos los requisitos identificados (3.1).				
Recursos	Tesista.				
Duración estimada	15 días	Fecha inicio:	01/11/2020	Fecha Fin:	15/11/2020
Hito asociado	Interfaz de herramienta implementada.				

Dependencias	3.1
Observaciones	Ninguna.
Tareas	-Implementar la interfaz gráfica. -Hacer pruebas unitarias de funcionalidades.

Componente	Descripción				
Código del paquete de trabajo	3				
Descripción del paquete de trabajo	Integración del servicio web.				
Entregable(s)	Servicio web				
Criterios de aceptación del entregable(s)	-Servicio web debe satisfacer todos los requisitos identificados. -Servicio web debe estar preparado para ser desplegado en un servidor				
Recursos	Tesista.				
Duración estimada	37 días	Fecha inicio:	15/10/2020	Fecha Fin:	22/12/2020
Hito asociado	Ninguno.				
Dependencias	2, 3.2				
Observaciones	Ninguna.				
Tareas	-Integrar el clasificador con la interfaz de usuario. -Realizar pruebas de integración. -Implantar solución en ambiente de producción.				

Anexo 2

Matriz de Riesgos

Id	Descripción	Declaración	I	P	S	Estrategia	Mitigación	Disparador	Contingencia
1	Subestimación del alcance del proyecto.	Si se subestima el alcance, se pueden presentar tareas no contempladas en la planificación, resultando en un incremento del costo del proyecto.	5	2	10	Mitigar	Descomponer al máximo las actividades necesarias para la ejecución del proyecto.	Aparición de tareas que no estaban en el plan de proyecto.	Evaluar factibilidad de cambios.
2	Mala estimación de tiempos de entregables.	Si se da una mala estimación de los tiempos, puede ocurrir un retraso en alguna de las actividades de la ruta crítica, lo cual atrasaría el proyecto.	5	3	15	Mitigar	Usar estimación de peor escenario.	Atraso en un entregable.	Reducir tiempo de actividades siguientes. Aumentar esfuerzo en actividades siguientes.
3	Problemas personales o de salud del tesista o asesor.	Si el asesor o el tesista presenta algún problema personal, su atención se desviará a solucionarlo, resultando en un atraso del proyecto.	2	3	6	Aceptar	No aplica.	Comunicación del problema.	No aplica.
4	Curva de aprendizaje de herramientas muy alta.	Si la curva de aprendizaje para el uso de una herramienta es muy alta, el tiempo de finalización de una actividad puede volverse mayor al estimado, produciendo un atraso del proyecto.	2	3	6	Aceptar	No aplica.	Atraso en entregable por problemas con el uso de herramientas.	No aplica.
5	Indisponibilidad de algunas herramientas	Si alguna herramienta usada en la literatura ya no está disponible, es posible que se deba incluir tareas de búsqueda de una herramienta alternativa. Esto produciría un atraso en el cronograma.	4	2	8	Evitar	Revisar que la herramienta sea mantenida constantemente antes de seleccionarla.	Imposibilidad de descargar o instalar la herramienta.	Buscar herramienta alternativa.
6	Pérdida, avería o destrucción de ordenador donde se desarrolla el proyecto.	Si el ordenador de trabajo se malogra o se pierde, se perderá la totalidad o parte del trabajo realizado. Esto se traduciría en un atraso en el proyecto porque sería necesario rehacer parte o todo el trabajo.	5	2	10	Mitigar	Realizar constantemente backups en la nube y/o en un disco duro externo.	Pérdida/avería del equipo.	Continuar el trabajo a partir de la copia de respaldo.
8	Caída de infraestructura Cloud	Si ocurre una caída de la infraestructura Cloud es posible que no se pueda trabajar con la información en la nube	3	1	3	Mitigar	Mantener una copia local y/o en un disco duro externo del trabajo.	Corte de servicio.	Trabajar con copia local.
9	Desastre natural.	Si ocurre un desastre natural es posible que no se cuente con las herramientas necesarias para continuar con la ejecución del proyecto.	5	2	10	Aceptar	No aplica.	Ocurrencia desastre	No aplica.

Anexo 3

Plan de Pruebas

ID Prueba	1
Caso uso	Clasificación
Flujo	Normal
Pasos	
1. Ingresar a la página de la aplicación.	
2. Introducir "1LM1" en el campo del identificador de proteína.	
3. Introducir "A" en el campo del identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
IV.1	
Resultado Obtenido	
IV.1	
Pasa la prueba	Sí

ID Prueba	2
Caso uso	Clasificación
Flujo	Alternativo 1A
Pasos	
1. Ingresar a la página de la aplicación.	
2. Dejar vacío el campo del identificador de proteína.	
3. Dejar vacío el campo del identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
Mensaje "Completa este campo" en identificador de proteína.	
Resultado Obtenido	
Completa este campo (identificador de proteína).	
Pasa la prueba	Sí

ID Prueba	3
Caso uso	Clasificación
Flujo	Alternativo 1B
Pasos	
1. Ingresar a la página de la aplicación.	
2. Introducir "1LM1" en el campo del identificador de proteína.	
3. Dejar vacío el campo del identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
Mensaje "Completa este campo" en el campo correspondiente.	
Resultado Obtenido	
Completa este campo (identificador de cadena)	
Pasa la prueba	Sí

ID Prueba	4
Caso uso	Clasificación
Flujo	Alternativo 2A
Pasos	
1. Ingresar a la página de la aplicación.	
2. Introducir "XYZ" en el campo del identificador de proteína.	
3. Introducir "A" en el campo identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
Respuesta: "Error: no se ha encontrado el archivo PDB para el id={id}".	
Resultado Obtenido	
Error: no se ha encontrado el archivo PDB para el id=XYZ	
Pasa la prueba	Sí

ID Prueba	5
Caso uso	Clasificación
Flujo	Alternativo 2A
Pasos	
1. Ingresar a la página de la aplicación.	
2. Introducir "1LM1" en el campo del identificador de proteína.	
3. Introducir "1" en el campo identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
Mensaje indicando que el identificador de cadena es una letra entre A y Z.	
Resultado Obtenido	
Mensaje indicando que el identificador de cadena es una letra entre A y Z.	
Pasa la prueba	Sí

ID Prueba	6
Caso uso	Clasificación
Flujo	Normal
Pasos	
1. Ingresar a la página de la aplicación.	
2. Introducir "XZZZ" en el campo del identificador de proteína.	
3. Introducir "A" en el campo identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
Respuesta: "Error: no se ha encontrado el archivo PDB para el id={id}".	
Resultado Obtenido	
Error: no se ha encontrado el archivo PDB para el id=XZZZ	
Pasa la prueba	Sí

ID Prueba	7
Caso uso	Clasificación
Flujo	Normal
Pasos	
1. Ingresar a la página de la aplicación.	
2. Introducir "1ZZZ" en el campo del identificador de proteína.	
3. Introducir "Z" en el campo identificador de cadena.	
4. Pulsar "Clasificar"	
Resultado Esperado	
Respuesta: "Error: no se ha encontrado la cadena asociada a la proteina ingresada".	
Resultado Obtenido	
Error: no se ha encontrado la cadena asociada a la proteina ingresada	
Pasa la prueba	Sí