UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PREDICCIÓN DE FACTORES DE TRANSCRIPCIÓN USANDO DEEP LEARNING

TESIS

QUE PARA OPTAR POR EL GRADO DE:

DOCTOR EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

M. en C. LEONARDO LEDESMA DOMÍNGUEZ

TUTOR:

DR. ERNESTO PÉREZ RUEDA

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y SISTEMAS

COMITÉ TUTORAL:

DRA. KATYA RODRÍGUEZ VÁZQUEZ

DR. PAUL ERICK MÉNDEZ MONROY

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y SISTEMAS

AREA: COMPUTACIÓN CIENTÍFICA E INTELIGENCIA ARTIFICIAL

CIUDAD UNIVERSITARIA, CD.MX. JUNIO 2024

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a ala institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado "Predicción de Factores de Transcripción usando Deep Learning", que presenté para obtener el grado de Doctor en Ciencia e Ingeniería de la Computación, es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Programa de Posgrado, citando las fuentes, ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad e los actos de carácter académico administrativo del proceso de titulación/graduación

Atentamente,

Leonardo Ledesma Domínguez
308314878

וַיְדַבֵּר יהוה אֶל־מֹשֶׁה לֵאמֹר : דַּבֵּר אֶל־אַהֲרֹן וְאֶל־בָּנָיו
לֵאמֹר כֹּה תְבָרֲכוּ אֶת־בְּנֵי יִשְׂרָאֵל אָמוֹר לָהֶם :
יְבָרֶכְךָ יהוה וְיִשְׁמְרֶךָ : יָאֵר יהוה פָּנָיו אֵלֶיךָ וִיחֻנֶּךָּ :
יִשָּׂא יהוה פָּנָיו אֵלֶיךָ וְיָשֵׂם לְךָ שָׁלוֹם

Para mi amada madre, Irene Domínguez Reyes

Descanse en paz

"¿Final? No, el viaje no termina aquí. La muerte solo es otro camino, uno que todos vamos a recorrer. La cortina de lluvia gris del mundo se abre, y se transforma en plata y cristal, después lo ves. Blancas costas, y más allá, un país lejano y verde a la luz de un amanecer."

- El retorno del Rey, J.R.R Tolkien

Agradecimientos

Al motor inmóvil, al sempiterno, al Él Olam, al **אהיה אשר אהיה**, *al Tetragrámaton* יהוה, *a* יְהוָה צִדְקֵנוּ, *la primera causa, el terror de los ateos y el juez moral de los hombres, a Dios.*

A mi padre Rogelio Ledesma González, que con su sudor y fuerzas me ha mostrado el camino de la perseverancia y la constancia. A mi hermana, Abigail Ledesma Domínguez, un milagro de vida, un amor inconmensurable.

A mi Loki, juguetón y fiel amigo. A mi Baxter amoroso y tierno.

A mi mejor amigo Erik Carbajal-Degante que me ha apoyado en mis momentos más oscuros y que me ha apoyado a lo largo de todo este trabajo de investigación.

A mi tutor Ernesto Pérez Rueda, por sus consejos académicos siempre tan atinados y por su tiempo dedicado en mi formación doctoral, a mi jefe y líder Javier Lozano Espinosa por enseñarme a tener un punto crítico y lógico de las cosas, por sus consejos personales y sobre todo por su apoyo.

A la Dirección General de Evaluación Institucional de la UNAM y a su director Imanol Ordorika Sacristán por hacerme crecer profesionalmente.

Al posgrado en Ciencia e Ingeniería de la Computación, la gloriosa Universidad Nacional Autónoma de México y al Conahcyt por darme los recursos científicos y académicos para desarrollar este trabajo.

- V1DE -

A lo largo de estos cuatro años, he vivido, conocido, sufrido, perdido, llorado, disfrutado pero sobre todo he aprendido...

Vivido, muchos cambios en mi modo de pensar y muchas experiencias enriquecedoras...

Conocido, quien es amigo y quien no es amigo... quien es auténtico y quien está solo por interés...

Sufrido, la pérdida de personas muy importantes en mi vida que han dejado este mundo...

Perdido, a mi mismo algunas veces, pero me he vuelto a encontrar...

Llorado, lágrimas de sangre y fuego del corazón...

Disfrutado, de vivir, de lo que tengo, pero no solo lo que tengo, si no lo que puedo alcanzar por y para Dios.

Y aunque varias personas se han ido por voluntad propia, conservo aquellas que me hacen una mejor persona con sus consejos y sus exhortaciones.

Muchos buscan la felicidad, otros buscan el placer, otros el dinero, yo solo busco y vivo por una cosa, por la VERDAD.

Pero sobre todo he aprendido, tantas cosas de mí, de las personas, del mundo.

Solo puedo concluir de mi mismo:

העיניים האלה שאוהבות בצורה ייחודית ואמיתית,
...שמאמינות עד הרגע האחרון באנשים שלא מגיע להם שיאמינו להם

עיניים עצובות שחוו דברים רבים בארבע השנים האחרונות
...שאיבדו הרבה והכל
.אבל למרות זאת עדיין מאמינות בלהתקדם

Resumen

Los factores de transcripción juegan un rol fundamental en la regulación de la expresión génica. Los factores de transcripción son proteínas que se unen a sitios de unión específicos del ADN. Las uniones de factores de transcripción con sus respectivas secuencia ADN dan como inicio el proceso de transcripción, que es el primer paso de la expresión génica que tiene como objetivo producir una copia de ARN (transcrito) de la secuencia de ADN de un gen.

Los factores de transcripción están íntimamente relacionados a las respuestas al medio ambiente, patogénesis y desarrollo de un organismo, es por eso que identificarlos y caracterizarlos es importante para el conocimiento del mundo biológico.

La vía más fiable para identificar un factor de transcripción es mediante la base experimental sin embargo, es costosa y tardada. Actualmente gracias a todos los datos existentes de proteomas reportados y anotados han permitido que las técnicas de identificación ahora vayan orientadas al auge de los algoritmos de Inteligencia Artificial específicamente del área del *Deep Learning*.

Aunque originalmente las bases del *Deep Learning* surgen para resolver problemas de Visión Computacional y del Procesamiento del Lenguaje Natural, en los últimos 10 años modelos basados en *Deep Learning* han funcionado bastante bien para resolver problemas de la Bioinformática.

Teniendo entonces los datos como materia prima y los modelos de *Deep Learning* como la herramienta, se puede asegurar que es posible realizar casi cualquier tarea con resultados confiables y deseables. Este trabajo propone una nueva arquitectura basada en *Deep Learning* con el objetivo de predecir factores de transcripción con alta confiabilidad y mejorando las métricas de desempeño del modelo, eliminando y evitando el sobreajuste.

También se propone un enfoque novedoso basado en la medición del sesgo y la varianza de las predicciones para medir la calidad y la certidumbre de un modelo. En otras palabras, no es lo mismo que un modelo predija un verdadero positivo con una probabilidad cerca a 1 que cercana a 0.5. Cualitativamente el modelo será más certero si su tendencia de las probabilidades en sus predicciones de verdaderos

positivos es cerca a 1.

Finalmente, se caracterizan algunos factores de transcripción de tres organismos fúngicos que no fueron identificados por otro modelo de *Deep Learning* y que carecen de evidencia experimental.

Abstract

Transcription factors play a fundamental role in the regulation of gene expression. Transcription factors are proteins that bind to specific sites on DNA. The relations of transcription factors with their respective DNA sequence start the transcription process, which is the first step in gene expression that aims to produce an RNA copy from the DNA sequence of a gene.

Transcription factors are closely related to responses to the environment, pathogenesis, and growth of an organism, which is why identifying and characterizing them is important for understanding the biological world.

The most reliable way to identify a transcription factor is through the experimental basis; however, it is expensive and time consuming. Currently, thanks to all the existing data of reported and annotated proteomes, they have allowed identification techniques to now be oriented towards the rise of algorithms. Artificial Intelligence specifically from the area of Deep Learning.

Although originally the theory of Deep Learning arose to solve Computer Vision and Natural Language Processing problems, in the last 10 years models based on Deep Learning have worked quite well to solve Bioinformatics problems.

Having then the data as raw material and the Deep Learning models as the tool, it can be ensured that it is possible to carry out almost any task with reliable and desirable results. This work proposes a new architecture based on Deep Learning with the objective of predicting transcription factors with high reliability and improving the performance metrics of the model, eliminating and avoiding overfitting.

A novel approach based on the measurement of the bias and variance of the predictions to measure the quality and certainty of a model is also proposed. In other words, it is not the same for a model to predict a true positive with a probability close to 1 than close to 0.5. Qualitatively, the model will be more accurate if its trend of probabilities in its predictions of true positives is close to 1.

Finally, some transcription factors of three fungal organisms that were not identified by another Deep Learning model and lacking experimental evidence are characterized.

Índice general

Índice de figuras

Índice de tablas

Glosario de Palabras Clave y Abreviaturas

- Inglés
  - TFBS (Transcription Factor Binding Sites): Sitios de unión de factores de transcripción.
  - TF (Transcription Factor): Factor de Transcripción.
  - DL (Deep Learning): Aprendizaje profundo.
  - ML (Machine Learning): Aprendizaje automatizado.
  - PSSM (Position-Specific Scoring Matrix) : Matriz de puntuación de posición específica.
  - PWM (Position Weight Matrix): Matriz de pesos de posición.
  - BLAST (Basic Local Alignment Search Tool): Herramienta básica de búsqueda de alineación local.
  - PFAM (Protein Families): Algoritmo basado en HMMs para asociar funciones y caracterizar familias de proteínas.
  - HMM (Hidden Markov Model): Modelo Oculto de Markov.
  - SMART (Simple modular architecture research tool) : Herramienta de búsqueda de arquitectura modular simple.
  - GO (Gene Ontology): Ontología génica
  - CNN (Convolutional Neuronal Network): Red neuronal convolucional.
  - RNN (Recurrent Neuronal Network): Red neuronal recurrente.
  - TL (Transfer Learning): Transferencia de conocimiento.
  - AM (Attention Mechanism): Mecanismo de atención.
  - LSTM (Long Short-term Memory): Tipo de red RNN llamada Memoria a Corto Plazo.

- BiLSTM (Bidirectional Long Short-term Memory): Tipo de LSTM que lee una entrada de forma reversa y anversa.
- GRU (Gated recurrent unit): Tipo de red RNN llamada Unidad recurrente cerrada.
- Sobrejauste (Overfitting): Problema del entrenamiento de las redes neuronales donde el modelo se sesga a los datos de entrada de entrenamiento, ya sea porque el modelo está incorrectamente parametrizado o bien porque la variedad de los datos es o tiende a ser nula, por consiguiente, predecir datos nuevos falla.
- NLP (Natural Language Processing): Procesamiento del Lenguaje Natural.
- BERT (Bidirectional Encoder Representations from Transformers): Representación de Codificador Bidireccional de Transformadores.
- Encoder: Codificador, parte de una red Transfomer dedicado a codificar una entrada basada en tokens para generar una representación interna.
- Decoder: Decodificador, parte de una red Transformer, dedicado a decodificar un estado interno a un lenguaje entendible. GPT (Generative Pre-trained Transformer)

- Español
  - IA - Inteligencia Artificial
  - ADN - Ácido Desoxirribonucleico
  - ARN - Ácido Ribonucleico

Capítulo 1

Introducción

"Algún matemático dijo que el verdadero placer no reside en el descubrimiento de la verdad, sino en su búsqueda".

- Lev Tolstói

A mediados del 2021, se publicó en la revista Nature un algoritmo que revolucionó el campo de la biología computacional llamado *AlphaFold2* [Jumper et al., 2021]. Dicho algoritmo es capaz de predecir de manera exacta la estructura tridimensional de una proteína, donde casi dos tercios de las predicciones coinciden con los datos tridimensionales que se han obtenido por medio de la experimentación y que están depositados en la base de datos *Protein Data Bank* [Berman, 2000].

Gracias a *AlphaFold2*, las miradas de muchos investigadores han vuelto a reconocer a los modelos de *Deep Learning* [Sapoval et al., 2022, Zhang et al., 2023, He et al., 2020c, Bzdok et al., 2024] como excelentes metodologías para realizar tareas de predicción y clasificación con alta precisión. Sin embargo, uno de los talones de Aquiles de los modelos basados en *Deep Learning* es la cantidad de datos necesaria para alcanzar a resolver el Problema de Generalización [Neyshabur et al., 2017].

El Problema de Generalización se define como: "La capacidad de un algoritmo de ser efectivo a través de un rango nuevo de entradas". Con rango nuevo de entradas se hace referencia a la capacidad de adaptar ese algoritmo a datos nuevos, los cuales no fueron utilizados para el entrenamiento o un proceso previo.

Actualmente, la cantidad de datos en el área de la Biología Computacional no es del todo una limitante, ya que están disponibles 0.1 T $[0.1 \times 10^{12}]$ de pares de bases secuenciadas en la base de datos *GenBank* [Benson et al., 2007] y más de 500 mil secuencias de proteínas bien anotadas en la base de datos *UniProt* [Apweiler, 2004]. En este contexto, en el mundo de la genómica, hay muchos datos, pero

se conoce poco de su relevancia en el mundo biológico. De esta forma, aplicar metodologías y modelos basados en *Deep Learning* se vuelve atractivo, dado que la limitante de datos se solventa. Como otro gran beneficio del uso de este tipo de herramientas, es el ahorro de costos en contraste con la experimentación *in vivo*, esta en todo caso puede ser usada como confirmación de las predicciones de los modelos basados en *Deep Learning* .

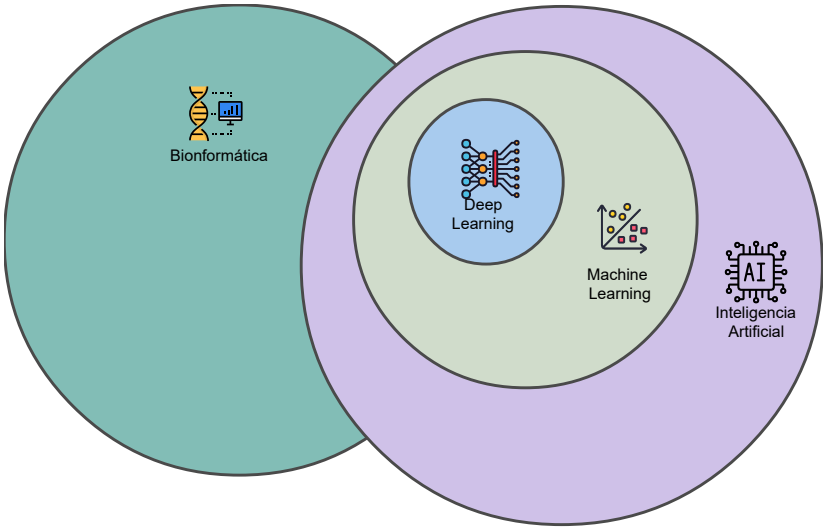


Figura 1.1: La división disciplinaria de los algoritmos basados en Inteligencia Artificial (IA) y su relación con la bioinformática, como se puede observar, la cantidad de algoritmos que se ofrecen disminuye a lo largo de la profundización de la IA

1.1. Las ómicas y la diversidad de los datos bioinformáticos

Gracias a la innovación de la secuenciación masiva de alto rendimiento (NGS, por sus siglas en inglés, *Next-Generation Sequencing*), la disponibilidad de cada vez más genomas y proteomas de diferentes organismos y los avances científicos como la terapia génica, la técnica *CRISPR/Cas9* y *Chromosome Conformation Capture* (Hi-C), el mundo de la bioinformática está preparado para apelar al uso del *Deep Learning* y otros algoritmos matemáticos como nuevos pasos al análisis masivo de datos.

En este sentido, la diversidad de los datos sigue la línea de las ómicas (ver Figura 1.2):

- Genómica (Genomas): para la estructura, función y mapeo del ADN. Son secuencias de un alfabeto de cuatro letras (A,T,C,G), se usa la bioinformática para determinar la expresión génica, el alineamiento (comparación) de secuencias y ensamblaje, entre otras tareas.
- Proteómica (Proteomas): Como parte del proceso de traducción o síntesis de proteínas, conocer

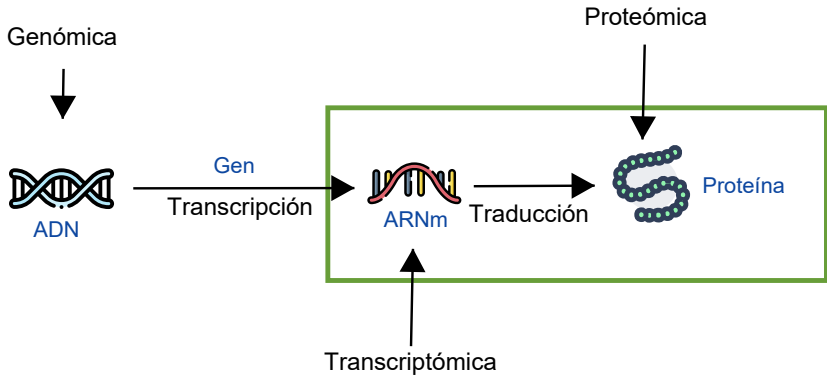


Figura 1.2: El dogma central de la Biología y las áreas de investigación en “omicas”.

todas aquellas proteínas y sus familias es fundamental para comprender la fisiología y la relevancia en el tratamiento de una enfermedad. Son secuencias de un alfabeto de 20 letras y se usa la bioinformática para alineamiento, predicción de estructuras e interacciones de proteína a proteína, entre otras.

- Transcriptómica (Transcriptomas): secuencias de ARN ya sea mensajero, ribosomal, o de transferencia, de un alfabeto de cuatro letras. Se evalúa la expresión de los genes con base en la abundancia de los ARN mensajeros.

Hay otros tipos de datos, como por ejemplo los que pertenecen al campo de la metabolómica, interactómica, epigenómica y la metagenómica.

Los datos bioinformáticos descargados en las diferentes plataformas que ofrecen estos datos son del tipo FASTA [NLM, 2004]. El tipo FASTA es un archivo de texto plano, el cual tiene la siguiente estructura:

- Un encabezado (>), seguido de un identificador de la secuencia, así como una descripción opcional.

- Datos de la secuencia que bien pueden ser nucleótidos (en pares de bases) o péptidos (en aminoácidos) representados por letras únicas.

Los identificadores de secuencia pueden cambiar según el estándar que use la plataforma de descarga.

1.2. Homología por comparación de secuencias

Los algoritmos tradicionales en Bioinformática utilizan un término llamado Homología para "inferir" funciones de una secuencia dada la similitud a partir de otra secuencia caracterizada experimentalmente.

Por ejemplo, sean las secuencias de proteínas no iguales A y B, donde de la secuencia A se conoce vía experimentalmente algunas funciones, se dice que B es una secuencia homóloga de A, si por alineamiento de secuencias de A y B son similares. Biologicamente hablando, se puede decir que al ser B una secuencia homóloga de A, comparte o realiza algunas funciones de A.

Las secuencias homólogas pueden ser ortólogas, parálogas o xenólogas, dependiendo del evento evolutivo que las haya originado. Las ortólogas son producto de separación de un evento de especiación, las secuencias parálogas por un evento de duplicación y las xenólogas por transferencia horizontal.

La similitud de secuencias no implica necesariamente homología, dependerá del porcentaje de identidad para llegar a esa conclusión. Es entonces donde el alineamiento de secuencias toma un rol fundamental para descubrir dicho porcentaje de identidad.

El alineamiento de secuencias se puede realizar de manera local o global, o se pueden usar aproximaciones heurísticas de las mismas como *BLAST* [McGinnis and Madden, 2004] y su familia de alineadores de secuencias. Al final, el objetivo del alineamiento de secuencias es reducir los *mismatches* y *gaps*, y maximizar los *matches* .

En el caso de los Factores Transcripcionales (TFs, por sus siglas en inglés), se utiliza la herramienta de homología por similitud de secuencias para inferir que secuencias no caracterizadas experimentalmente puedan tener funciones parecidas a los TF comprobados experimentalmente.

Por ejemplo, el caso de las técnicas basadas en ML para predicción de factores de transcripción, la técnica que se usa en *HMMer* implementada en la base de datos *PFAM* [Sonnhammer et al., 1997].

Partir de la similitud de secuencias puede traer en sí mismo un sesgo, dado que será imposible descubrir nuevas funciones de secuencias si no se tiene validado experimental al menos una secuencia que caracteriza dichas funciones nuevas.

1.3. Algoritmos tradicionales de predicción de TF basados en *Machine Learning*

El primer esfuerzo por aplicar algoritmos tradicionales de *Machine Learning* para predicción de TFs fue en 1998 con *PFAM* [Sonnhammer et al., 1997]. .

PFAM nació como un proyecto para disponer alineamientos de alta calidad para poder identificar dominios de familias de proteínas en diferentes bases de datos usando modelos ocultos de Markov, o por sus siglas en inglés HMM (*Hidden Markov Model*).

Los HMMs son aplicados para la construcción de *PFAM* y la detección de "*matches*" en los alineamientos múltiples de secuencias. Básicamente los perfiles HHMs calculan la probabilidad de que un aminoácido esté en una determinada posición en una secuencia, así como la probabilidad de insertar o saltar a otro residuo.

Inicialmente, *PFAM* trabaja una semilla de alineamientos de familia de dominios usando *ClustalW* [Thompson et al., 1994], para posteriormente construir un HMM por cada alineamiento usando el programa *hmmb*. Una vez obtenida la caracterización de cada familia de dominios por perfiles HMM, se puede saber por homología que funciones se le atribuyen a una secuencia *query*. Para ver un ejemplo de un perfil HMM vea la Figura 1.3.

Más tarde en el año 2012 se presentó *P2TF* [Ortet et al., 2012], un algoritmo que recolecta genomas y metagenomas de NCBI y IMG/M para obtener los respectivos proteomas y ORFeomas, aplicando una serie de familias basado en el *DNA-binding Domain* o DBDs (en inglés, Familia de Dominios a Sitios de Unión al ADN) que tengan funciones de TFs.

Posteriormente, *P2TF* sugiere posibles TFs de ese conjunto de proteomas y ORFeomas, a través de *RPS -BLAST* [Bla, 2018], complementando su información con *PFAM* y *SMART* [Letunic et al., 2009].

En 2013 se publicó otro trabajo de *Machine Learning* llamado *TFPredict* [Eichner et al., 2013]. En este trabajo se fundamentaron las bases para los trabajos posteriores de *Deep Learning*, tanto en la construcción de la base de datos de TFs como en el uso de diversas estrategias que se usaron para mejorar el set de datos de entrenamiento.

TFPredict redujo el set de datos de TFs y no TFs aplicando el programa*CD-HIT* [Li and Godzik, 2006], por similitud de secuencias al 80 % y al 56 % respectivamente, para finalmente tener un conjunto de datos 1:10 en relación de TFs-No TFs.

También, *TFPredict* combinó los beneficios de la homología de secuencias y los algoritmos tradicionales de *Machine Learning* utilizando la extracción de características evolutivas con una representación

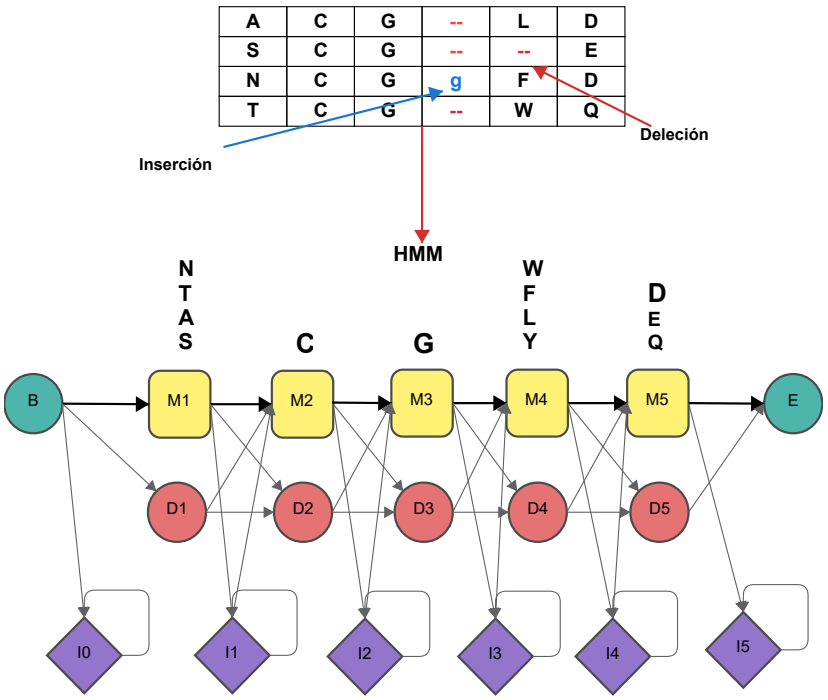


Figura 1.3: Modelo Oculto de Markov para cuatro secuencias de aminoácidos, donde se puede observar las dos operaciones básicas de inserción y deleción (estas dos no se mapean en la HMM). Aquel aminoácido que se encuentre con mayor frecuencia en la secuencias tendrá una mayor probabilidad de presentarse en dicha posición, como es el caso de la posición dos y tres.

basada en matrices PSSM usando PSI-*BLAST* [Altschul, 1997], que posteriormente inspiró al trabajo de *TFNet* [Du et al., 2023] para seguir la misma metodología pero aplicando *Deep Learning*

En la tabla 1.1 se muestran los algoritmos más usados basados en *Machine Learning* para la predicción de factores de transcripción.

1.4. Antecedentes de la resolución de tareas en bioinformática usando algoritmos de *Deep Learning*

En los últimos 10 años el uso de algoritmos basados en *Deep Learning* han sido frecuentemente usados para resolver tareas propias de bioinformática [Min et al., 2016, Li et al., 2019].

Las tareas en bioinformática en esta área se pueden categorizar en las referidas al dominio de las ómicas, la imagenología biomédica y el procesamiento de señales biomédicas.

Respecto a las tareas que se intentan resolver en el dominio de las ómicas, se podían clasificar de la

Algoritmo de *Machine Learning*	**Arquitectura Modelo de predicción**	**Tipo de aprendizaje**	**Uso**
HMM (Modelos Ocultos de Markov)	*PFAM*	Modelo Probabilístico-Estadístico	Extracción de características que definen la distribución de probabilidad de los datos de entrada. Herramienta auxiliar para alineamientos heurísticos tipo *BLAST* y sus variantes.
SVM (Support Vector Machine)	*TFpredict*	No supervisado	Construcción de superclases y dominios proteicos de familias de TFs
KNN (k-nearest neighbors) Clustering	*TFpredict* *P2TF*	No supervisado	Construcción de superclases y dominios proteicos de los TFs
Random Forest	*TFpredict*	No supervisado	Construcción de superclases y dominios proteicos de los TFs

Tabla 1.1: Algoritmos de *Machine Learning* usados para la clasificación y predicción de factores de transcripción

siguiente manera:

- Predicción de estructura de proteínas (PSP):

  Básicamente, consiste en mapear y caracterizar las formaciones biofísicas de las proteínas a través del reconocimiento de patrones de ciertas subsecuencias de aminoácidos. Estas formaciones biofísicas pueden determinar plegamientos en estructuras secundarias (hélices α, hojas β, etc.), terciarias y cuaternarias.

  Conocer dichas estructuras tridimensionales de las proteínas son vitales para conocer la actividad o inactividad de cada proteína en la fisiología de cada organismo o ser vivo.

- Regulación de expresión génica (RGE):

  La regulación es un proceso bastante amplio y abarca muchas ramas de estudio de la bioinformática. Dentro de la RGE, se puede encontrar tareas aún más específicas relacionadas al área de las genómicas como:

  - Predicción de sitios de unión al ADN (*DNA-Binding Sites Prediction*).

  - Predicción de sitios de unión al ARN (*RNA-Binding Sites Prediction*).

  - Predicción de funciones de secuencias de ADN *(DNA Sequence Function Prediction*).

  - Predicción de propiedades biomoleculares (*Biomolecular property prediction*).

- Predicción de función de las proteínas (PFP):

  Definir las funciones que realizan las proteínas al interior y exterior de la célula es fundamental para el avance de las ciencias biológicas y médicas. Sus alcances llegan hasta el estudio de enfermedades, mutaciones virales e incluso en la industria farmacéutica. Como tareas específicas de PFP están:

  - La predicción de la actividad enzimática.

- La predicción de factores de transcripción (tema propuesto en este trabajo).
- La predicción de factores de virulencia (tema nuevo propuesto en este trabajo).
- La predicción de proteínas de transporte.
- La predicción de modificaciones postraduccionales.
- Interacción de proteína a proteína.

Existen otras tareas como la predicción de espectrometría de masas MS^2 y modelos generativos de secuencias de novo [Wen et al., 2020], entre otros.

- Ingeniería genética y aplicativa (GEA):

  En este dominio, las tareas a resolver están asociadas a la aplicación de ciertos fenómenos bioquímicos o médicos como:

  - El descubrimiento de redes metabólicas.
  - El descubrimiento y análisis de fármacos.
  - La clasificación y análisis de enfermedades, tales como tumores y diversos tipos de cáncer.

A continuación se muestra el *timeline* de los algoritmos basados en *Deep Learning* propuestos hasta hoy (fecha de corte: 23 de abril de 2024) más importantes en las tareas de bioinformática descritas anteriormente.

Otros algoritmos propuestos a la fecha, son: *DeepSNR* [Salekin et al., 2017] (2017, 3 CNN), *Dilated Convolutions* [Gupta and Rush, 2017] (2017, dilated CNN), *BiRen* [Yang et al., 2017] (2017, CNN + RNN), *EPI-DLMH* [Min et al., 2020] (2021, CNN + Heurística).

1.5. *Timeline* de los algoritmos para la predicción de TFs

Como se puede ver en la Figura 1.4, los algoritmos basados en *Deep Learning* para la predicción de TFs y que fueron utilizados como antecedentes a este trabajo de investigación son: *DeepTFactor* [Kim et al., 2020] y *TFNet* [Du et al., 2023].

DeepTFactor [Kim et al., 2020] es una triple red neuronal convolucional paralelizada que recibe como datos de entrada las secuencias de proteínas de entrenamiento y tiene como salida una clasificación binaria (para determinar si es o no un TF), para más detalles ver la Figura 1.5.

DeepTFactor usa la teoría de filtros y campos receptivos del área de visión computacional al usar tres diferentes redes neuronales convolucionales con diferentes tamaños de filtros. Utiliza la misma me-

#	Año	Red neuronal/ Trabajo de investigación	Tipo de Arquitectura/ Complejidad Paramétrica	Metodologías adicionales para mejorar el desempeño	Mejores Métricas	Base de datos	Tecnología	Dominio
1	2015	*DeepBind* *Predicting the sequence specificities of DNA- and RNA-binding proteins by Deep Learning .* *[Alipanahi et al., 2015]*	CNN Baja	Validación Cruzada	AUC: 0.72	DREAM5	CUDA Nvidia	RGE
2	2015	*DeepSea* *Predicting effects of noncoding variants with Deep Learning –based sequence model.* *[Zhou and Troyanskaya, 2015]*	CNN Baja	Regularización	AUC: 0.896	Diferentes tipos celulares	Tensorflow Torch	RGE
3	2016	*DanQ* *A hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences.* *[Quang and Xie, 2016]*	CNN + RNN Media	Multi Task Output	AUC: 0.972	GM12878 EFB1 H1-hESC SIX5	Keras Theano	RGE
4	2018	*iDeepS* *Prediction of RNA-protein sequence and structure binding preferences using deep convolutional and recurrent neural networks.* *[Pan et al., 2018]*	CCN RNN paralelizadas + BLSTM RNN Media	Ninguno	AUC: 0.86	31 diferentes experimentos	Keras	RGE
5	2019	*DeePromoter* *Robust Promoter Predictor Using Deep Learning .* *[Oubounyt et al., 2019]*	CNNs paralelizadas + RNN Media	Preprocesamiento	MCC: 0.88 0.87	Human TATA Mouse TATA	PyTorch	RGE
6	2020	*Deep convolutional neural networks for predicting leukemia-related transcription factor binding sites from DNA sequence data.* *[He et al., 2020a]*	CNN + LSTM Media	Ninguna		GTRD		GEA (TFBS relacionados a leucemia)
7	2020	*TBiNet* *Enhancing the interpretability of transcription factor binding site prediction using attention mechanism. [Park et al., 2020]*	CNN + AM + Bi LSTM	Mecanismo de Antención		ENCODE-DREAM		RGE (Interpretabilidad de las predicciones de TFBS)
8	2021	*DeepGRN* *Prediction of transcription factor binding site across cell-types using attention-based deep neural networks.* *[Chen et al., 2021]*	CNN + RNN + AM Media	Mecanismos de Atención	AUC: 0.98	DREAM Challenge	Tensorflow	RGE

Tabla 1.2: Algoritmos antecedentes en la resolución de tareas.

todología que *TFPredict*, que reducen el set de datos aplicando similitud de secuencias, extrayendo los datos de *UniProt KB (Reviewed SwissProt)* aplicando 36 anotaciones (términos) de *Gene Ontology* .

En 2022, bajo la misma línea de *DeepTFactor* y *TFPredict* , se propone una nueva arquitectura llamada *TFNet* [Du et al., 2023]. *TFNet* simplifica el modelo de *DeepTFactor* eliminando las 3 CNNs y sustituyéndola por una sola CNN, dado que no usan la CNN como extractor de características sino como acondicionador de entrada a una red BiLSTM (Figura 1.6). Otra característica de *TFNet* es que emplea PSSMs (matrices de peso por posición) como datos de entrada a su red en lugar de utilizar las secuencias traducidas a one-hot encoding, como lo plantea *TFPredict* . Finalmente, *TFNet* extiende el modelo agregando un AM de tipo *Squeeze & Excitation* con la finalidad de mejorar el rendimiento de la red.

En el capítulo IV de este trabajo se presenta un análisis computacional sobre *DeepTFactor* y *TFPredict*, resaltando ventajas y desventajas contra el nuevo modelo propuesto en este trabajo de investigación, el cual llamaremos *DeepReg* [Ledesma-Dominguez et al., 2024].

También de manera extensiva, se abordarán los trabajos en el procesamiento de lenguaje natural para

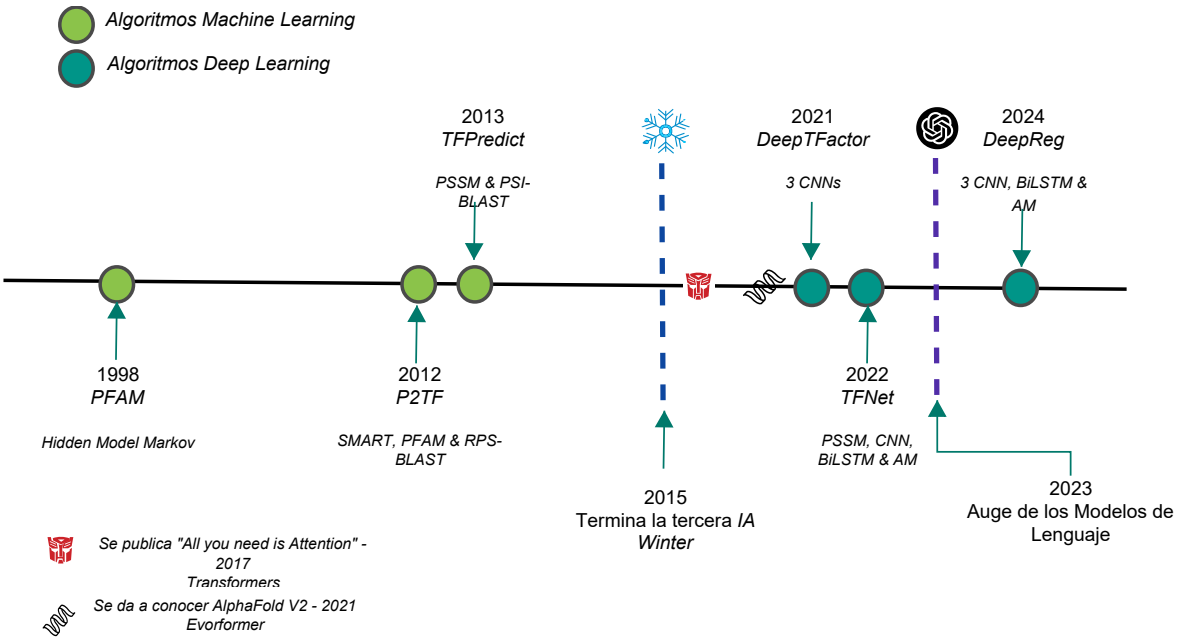


Figura 1.4: Timeline de los algoritmos para la predicción de TFs. En verde claro se indican los algoritmos basados en *Machine Learning* y en turquesa, los basados en *Deep Learning*.

modelos de lenguaje en proteínas y genomas y se mostrarán algunas pruebas de clasificación y predicción de TFs usando la familia *BERT*.

1.6. Thesis outline

En el capítulo 2 se analiza la importancia biológica de los TFs así como los datos que se encuentran a disposición para realizar las tareas predicción y clasificación de TFs. Posteriormente en el capítulo 3 se estudian las diferentes arquitecturas y técnicas que el *Deep Learning* proporciona para resolución de tareas de clasificación y predicción, definiendo su funcionamiento y aplicación sobre los datos de TFs.

En el capítulo 4 se analiza la problemática que existe en la predicción de TF, tanto en metodologías basadas en homología de secuencias, como en los algoritmos tradicionales de ML y DL. El capítulo 4 proporciona un estudio cuantitativo de las predicciones por el método de *PFAM* y *DeepTFactor*, para comprobar hipótesis y resultados en organismos de referencia experimental.

En el capítulo 5 se desarrolla una metodología y se diseña una nueva arquitectura de *Deep Learning* para predicción de TFs nombrada *DeepReg*. Después de realizar un conjunto de experimentos para seleccionar el mejor modelo, se realiza un análisis cualitativo, midiendo la calidad de las predicciones de *DeepReg* versus *DeepTFactor* con los organismos de referencia experimental del capítulo 4.

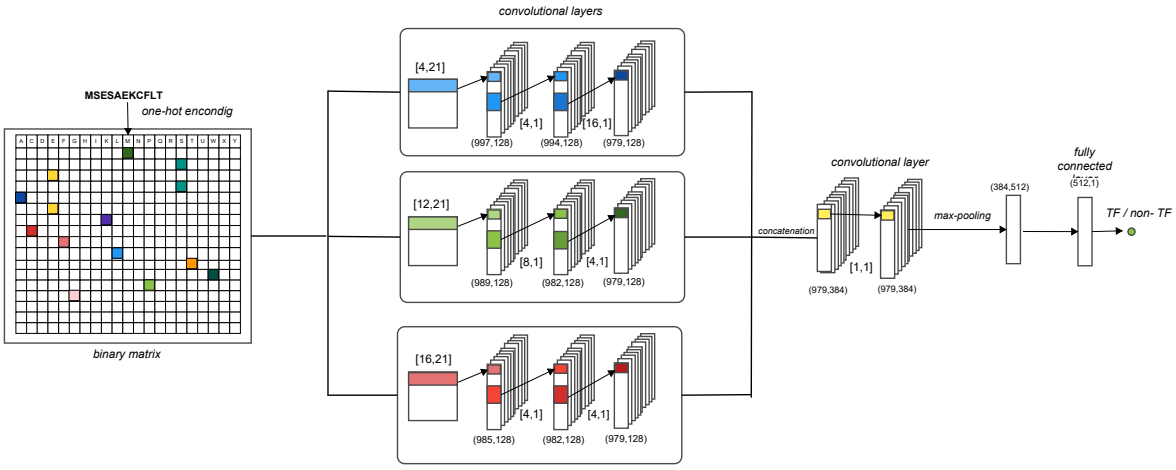


Figura 1.5: Arquitectura de *DeepTFactor*. Se muestran las 3 CNNs con diferentes tamaños de filtro, así como la clasificación binaria al final de la red.

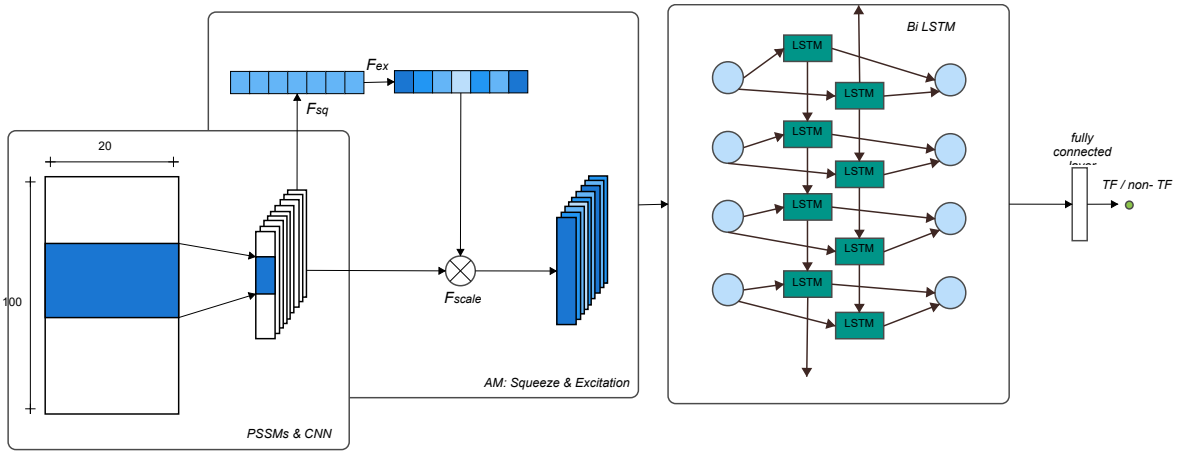


Figura 1.6: Arquitectura de TFNet. Hay tres módulos en la red TFNet, en el primero se calculan las PSSMs para cada secuencia de proteína y se acondiciona la entrada a una red CNN. En el segundo módulo se aplica un AM *Squeeze & Excitation* para finalmente aplicar una red BiLSTM y una clasificación binaria.

En el capítulo 6 *DeepReg* extiende su aplicabilidad a otras áreas de estudio de la Biología Molecular como los factores de virulencia, se demuestra su escalabilidad para introducir datos de diferente dominio y flexibilidad para la inferencia.

Para cumplir con uno de los objetivos de este trabajo de investigación, en el capítulo 6 se aborda la exploración de otro tipo de arquitecturas masivas utilizadas en procesamiento de lenguaje natural, tales como las redes tipo transformer de la familia *BERT* para la predicción de TFs.

Finalmente, en el capítulo 7 se muestran una serie de conclusiones que este trabajo de investigación

nos arroja en el uso de modelos de *Deep Learning* para la solución de tareas de predicción y clasificación.

Capítulo 2

Factores de transcripción

"Nada en la vida es para ser temido, es sólo para ser comprendido. Ahora es el momento de entender más, de modo que podamos temer menos."

- Marie Curie.

2.1. Dogma central de la biología molecular. Proceso de la transcripción génica

El Dogma Central de la biología molecular describe al conjunto de procesos bioquímicos que propician la transferencia de la información genética de todos los seres vivos, y que consiste en tres procesos: replicación del ADN, transcripción del ARNm y la traducción a proteínas (ver Figura 2.1, tomada de [Lodish, 2008] y reeditada en [Ledesma-Dominguez, 2015].

Si bien la replicación de ADN es considerada como parte del Dogma Central de la biología molecular, este es paralelo al de la transcripción y traducción. Para encontrar más del dogma central, referido a la replicación y traducción vaya al apéndice A de este trabajo donde se proporciona información obtenida de [Ledesma-Dominguez, 2015].

Para los fines de este trabajo, se describirá el proceso de la transcripción. Este proceso se puede dividir en tres etapas bien definidas: la iniciación, la elongación y la terminación, descritos en [Ledesma-Dominguez, 2015].

1. Etapa de Iniciación: La ARN polimerasa reconoce un sitio específico del ADN, llamado promotor. En este paso de reconocimiento participan un conjunto de proteínas conocidas como factores de

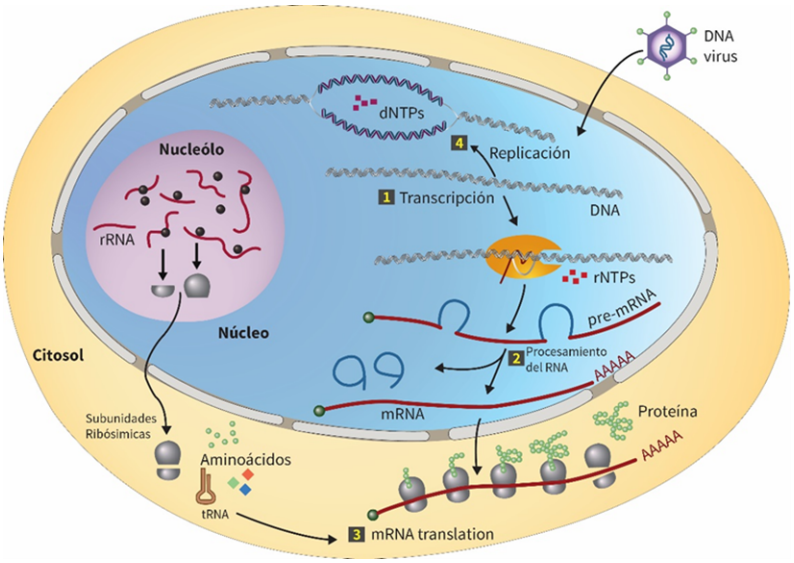


Figura 2.1: En este esquema se muestran los procesos involucrados en la transferencia de la información genética. 4. La replicación del ADN es llevada a cabo por aquellas células preparadas para su división, los monómeros dNTP (desoxirribonucleósido trifosfato) son polimerizados para la producción de dos copias exactas del ADN. 1. Se transcribe a ARN un gen, que codifica una proteína, la maquinaria de la ARN polimerasa participa en este proceso. 2. Se construye un precursor del ARNm mediante la proliferación de monómeros de rNTP (ribonucleósido trifosfato) y ocurre la eliminación de secuencias extrañas del pre-ARNm llamado el procesamiento del ARN. 3. El ARNm funcional producido en la etapa anterior se transporta al citoplasma para su traducción, en esta etapa el código de 4 bases que conforman los ácidos nucleicos es traducido al lenguaje de los 20 aminoácidos. Los ribosomas compuestos por una subunidad mayor y otra menor y con la ayuda de otras proteínas llevan a cabo la síntesis proteica.

transcripción o factores transcripcionales.

Cuando se haya establecido la unión de la ARN polimerasa con el promotor, se disocian las hebras del ADN para que las bases de los ribonucleótidos trifosfatados se apareen y se lleve a cabo el proceso de polimerización. La etapa de iniciación termina cuando se establece la unión de las dos primeras bases de ribonucleótidos de una cadena de ARN, a través de un enlace fosfodiéster.

2. Etapa de elongación: Después de la polimerización de varios ribonucleótidos, la ARN polimerasa se separa del promotor y de los factores de transcripción para llevar a cabo una etapa de elongación. En esta etapa la ARN polimerasa se mueve a lo largo del ADN molde de base en base, abriendo el ADN e hibridando las hebras detrás de sí. Durante esta etapa, más ribonucleótidos se van adicionando uno por uno al extremo 3' de la cadena de ARN creciente. El papel de la ARN polimerasa y del complejo de elongación es mantener una velocidad constante de pares de bases por minuto, aunque la molécula de ARN naciente es sumamente estable, se requiere mantener este proceso intacto, para asegurar la síntesis interrumpida del ARN mensajero.

3. Etapa de terminación: Durante la etapa final de la síntesis del ARN mensajero, el transcrito primario, se libera de la ARN polimerasa y se separa del ADN molde. La culminación se realiza mediante secuencias específicas de nucleótidos en el promotor que le señalan a la ARN polimerasa cuando parar. Cuando se libera la ARN polimerasa, está lista para volver a transcribir el mismo gen u otro distinto.

   Procariontes. En los organismos unicelulares, tales como las bacterias y las arqueas, los procesos de transcripción, traducción y degradación del ARN mensajero pueden ocurrir simultáneamente debido a que sus núcleos celulares no están encerrados.

   La ARN polimerasa que participa en el proceso de transcripción sólo es de un tipo y su ARN mensajero es policistrónico, es decir, contiene varias regiones codificantes que producirán diversas cadenas polipeptídicas.

   El ARN transcrito primario en la etapa de terminación ya es funcional, no requiere estrictamente de maduración postranscripcional, además de que las secuencias de terminación son palindrómicas.

   Eucariotes. En los organismos eucariotes, el ARN transcrito primario sufre en el núcleo un proceso de maduración o proceso postranscripcional.

   La ARN polimerasa que interviene en el proceso de transcripción en eucariontes es de tres tipos (I, II y III).

   A diferencia de los organismos procariotes, es necesario transportar el ARN mensajero al citoplasma para ser traducido, y además, es monocistrónico (codifica para sólo una cadena polipeptídica).

2.2. Factores de transcripción y TFBS en la regulación transcripcional

Las estructuras de regulación en la transcripción son los TFBS (Sitios de unión a factores de transcripción) y los TFs (ver Figura 2.2).

Los TFBS son motivos dedicados a la expresión génica y a menudo se consideran como regiones conservadas entre especies.

Las regiones del ADN que presentan TFBS se clasifican en:

- Promotores: delimitan y marcan el inicio de la secuencia de ADN a ser transcrita.

  El promotor típico en bacterias contiene dos subsecuencias llamadas elementos -10 y -35, que la ARN polimerasa reconoce. La gran presencia de Adeninas y Timinas en el elemento -10, lo hace susceptible al fácil rompimiento.

En eucariotes, es necesario utilizar proteínas auxiliares conocidas como factores basales de la transcripción, que ayudan a la ARN polimerasa a unirse al ADN.

- Potenciadores (*Enhancers*): actúan en cis (regiones no codificantes del ADN que regulan la transcripción de los genes cercanos) y su función es amplificar y aumentar la probabilidad de que ocurra la transcripción de un gen.
- Silenciadores (*Silencers*): evitan que los genes se expresen.
- Aisladores (*Insulators*): elementos del tipo cis regulador que marcan los límites para la expresión génica, están situados entre un promotor y un potenciador.

La regulación transcripcional es el proceso por el cual la célula regula la transcripción o síntesis del ARNm. La regulación se basa en el control de la frecuencia y presencia de expresión de genes. Este control permite a la célula responder de manera correcta a señales intra y extracelulares.

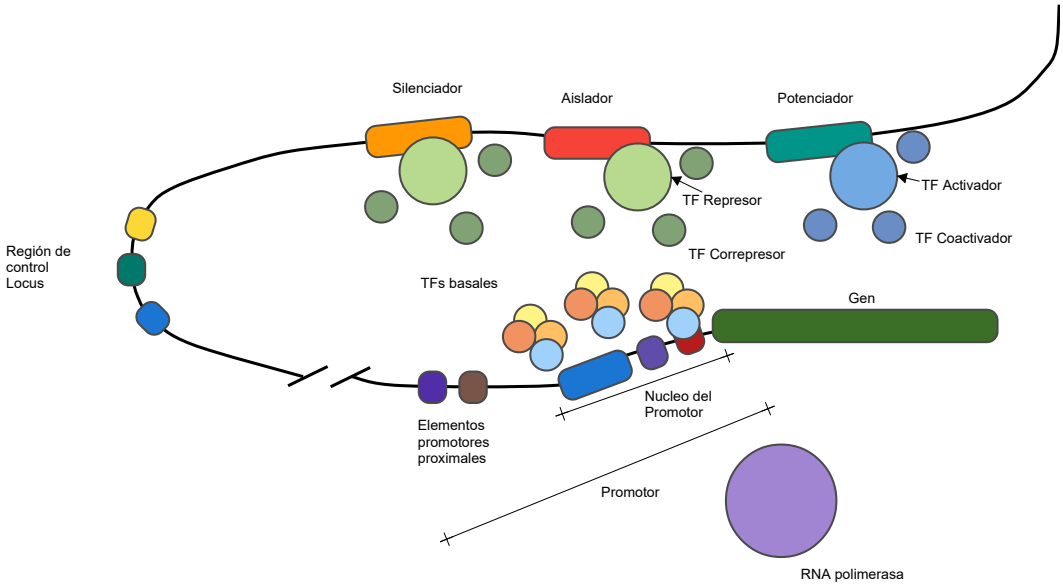


Figura 2.2: Sitios de Unión a Factores de Transcripción (TFBS) y los diferentes tipos de Factores de Transcripción (TF) . En el esquema se representa la interacción de los TFs con los TFBS. Cabe mencionar que las dos rayas simbolizan una región más alejada y no está en escala necesariamente.

Toda la información relacionada a los TFs sintetizada a continuación se puede consultar en [Gonzalez, 2016, Minchin and Busby, 2013, Suzuki and Yagi, 1994].

Los factores de transcripción son proteínas que se unen a secuencias específicas de ADN a través del (o los) dominio(s) proteico(s) y se pueden clasificar en:

- Factores basales o generales de transcripción (*GTF*): tipo de TF, cuyo complejo proteico interacciona con el promotor en la primera etapa de la transcripción.

- Factores específicos de la transcripción: complejo proteico que pueden ser activadores o represores.

  Activadores (*ATF*): se unen a los elementos distales y potenciadores para activar la transcripción mediante un efecto cooperativo con la maquinaria basal de transcripción.

  Represores (*RTF*) : interaccionan el ADN en la región upstream del gen y bloquean el acceso de la maquinaria basal de transcripción, silenciando la expresión genética.

- Coactivadores y correpresores (*CTF*): incentivan a la acción correspondiente de las proteínas asociadas respectivamente. Hay otra clasificación de los TFs que establece dos tipos: TFs constitutivos y TFs inducibles. Los TFs constitutivos (*Housekeeping TFs*) son los que participan en la expresión basal de los genes codificantes de proteínas estructurales, de mantenimiento y fundamentales para la supervivencia de la célula. Por su lado, los TFs inducibles regulan la expresión de genes asociados a estímulos específicos de la célula, tales como respuesta a estrés por calor.

Otras regiones de control importantes para el proceso de transcripción a nivel de secuencia son: el operador, así como la caja TATA, que es una secuencia de ADN encontrada en el promotor y es el sitio de unión con la ARN polimerasa.

Los TFs tienen las siguientes funciones:

1. Potencializan o reprimen la expresión de los genes en el momento indicado y cantidad necesaria según la célula lo requieran para:

   La diferenciación celular, cambios en la morfología y desarrollo celular.

   La señalización celular intercelular y del ambiente, así como la adaptabilidad a cambios de temperaturas.

   La apoptosis y crecimiento celular.

   Autorregulación y regulación de otros TFs.

2. Catalizan la acetilación y desacetilación de las histonas para cambiar la accesibilidad del ADN.

3. Impiden o estabilizan la unión ARN polimerasa con el ADN.

2.3. Estructura y familias de los TFs

Los TFs son modulares y están constituidos por cuatro dominios (ver Figura 2.3):

- Dominio de unión al ADN (*DBD*): la función principal es reconocer los elementos reguladores adyacentes a los genes regulados.
- Dominio de transactivación (*TAD*): posee sitios de unión llamadas funciones activadoras (AFs), para proteínas correguladoras de transcripción o bien a componentes del complejo de iniciación de la transcripción.
- Dominio regulatorio (*RD*): detecta señales exteARNs que afectan directamente a una regulación positiva o negativa. También se le conoce como Dominio Efector o Dominio de Detector de Señales.
- Dominio de dimerización (*DD*): regula la unión específica al ADN.

Todos los TFs tienen los dominios I y II, otras más pueden que tengan el III y/o el IV.

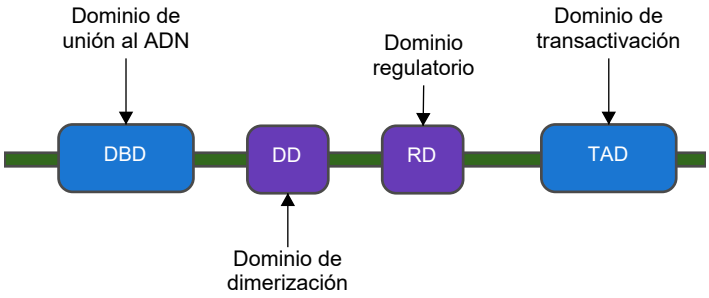


Figura 2.3: Estructura general de los TFs. Los dominios marcados en azul son los dominios comunes para todos los TFs, los dominios de color morado están presentes en algunos TFs.

Las familias de TFs son determinadas por la caracterización del dominio a unión al ADN. Dicha propuesta es realizada en [Wingender, 2013, Latchman, 2004] como:

1. Superclase: Basada en la topología del dominio a unión al ADN.
2. Clase: Basada en la estructura *blueprint* del dominio a unión al ADN.
3. Familia: Basado en la similitud de secuencias y funcional.
4. Subfamilia: Basado en subgrupos de secuencias
5. Género: Basado en el gen asociado al TF.

6. Especie: Basado en la cadena polipeptídica del TF

En la tabla 2.1, se muestran cinco superclases y las clases en los que se clasifican los TFs.

Alrededor del 95 % de los TFs conocidos se encuentran ubicados en las primeras 3 superclases.

2.4. Relevancia clínica en medicamentos y enfermedades.

El estudio amplio de los TFs puede ser aplicado directamente a la medicina para el tratamiento y explicabilidad de enfermedades. Las mutaciones de los TFs pueden desencadenar fallas en el proceso de la transcripción relacionados a velocidad de elongación de la transcripción e incluso la accesibilidad al ADN.

Un caso muy conocido es el TF asociado al gen *BRCA1*, este gen supresor de tumores en el humano, regula el ciclo celular y evita la proliferación incontrolada de fallas en la transcripción. El TF BRCA1 se encarga de la reparación y detección de daños en el ADN.

A las mutaciones del gen *BRCA1* y la falla de su respectivo TF se les asocian algunos tipos de cáncer, como por ejemplo cáncer de mama, de ovario, de próstata y de páncreas.

Recientemente, se identificó la relación de TFs asociados a los genes *APP* y *UBB*, encargados de la formación de proteínas amiloides presentes en el desarrollo del Alzheimer [Chung et al., 2023]. En dicho estudio se demostró que los TFs asociados a esos genes, no logran plegarse de manera adecuada.

Otra enfermedad asociada a los TFs Hnf1a y Hnf4a es la diabetes. En [Boj et al., 2010] se muestra que los TFs Hnf1a y Hnf4a mutados conducen a la secreción anormal de insulina.

Así como estos ejemplos, tenemos los genes *HOX* que codifican a una amplia familia de TFs de la clase homeodominio, en patologías vasculares y renales. También mutaciones en el TF FOXP3 causan enfermedades autoinmunes como *lupus* y artritis reumatoide.

Y finalmente, el caso más conocido es el de la familia de TFs, NF-kB ,que está relacionada a la respuesta celular por estrés, a la radiación ultravioleta y a la oxidación. Mutaciones en el complejo proteico de NF-kb está relacionado con el desarrollo de varios cánceres y enfermedades autoinmunes, entre otras.

El estudio de los TFs también se ha llevado al campo de la farmacéutica, en el diseño de medicamentos que a través de cascadas de señalización celular buscan modular indirectamente a los TFs.

En los últimos diez años, se han propuesto nuevas alternativas de fármacos de respuesta nuclear

asociados a la farmacogenética, farmacogenómica y la medicina personalizada e incluso se ha llegado a mencionar áreas como la nutrigenómica interesada en el estudio de nutrientes para mejorar, tratar y prevenir cualquier falla funcional o estructural de los TFs.

2.5. Data de factores de transcripción.

Se cuenta con las siguientes bases de datos a disposición de TFs:

- PlantTFBD [Jin et al., 2016]: proporciona un listado de TF de plantas por especie, tiene alrededor 320 mil TFs.
- TF2ADN [Pujato et al., 2014]: proporciona información experimental de seis organismos: *Escherichia coli, Sacharomyces cerevisiae, Caenorhabditis elegans, Drosophila melanogaster, Mus musculus, Homo sapiens*. Presentan una lista de 2,987 TFs con sus respectivos genes a los que regulan.
- TRANSFAC [Matys, 2003]: es una base de datos que reporta 48,000 TFs y TFBS de eucariotes. Utiliza PWMS para representar la relación de los TFBS y presentan una clasificación para los TFs.
- humanTFs [Lambert et al., 2018]: presenta una colección de 1639 TFs en el humano.
- AnimalTFDB [Shen et al., 2022]: proporciona información de TFs en 183 especies de animales y un total de 270 mil TFs.

Otra importante fuente de información es UniProt [Apweiler, 2004] que concentra diferentes bases de datos de dominio público desarrollada por EBI-SIB (*European Bioinformatics Institute - the Swiss Institute of Bioinformatics*) las cuales contienen:

- UniProtKB (*UniProt Knowledgebase*): es una base de datos parcialmente anotada por expertos y se divide en dos, *SwissProt* (entradas manuales y revisadas) y TrEMBL (entradas no revisadas y automáticas). Actualmente (mayo de 2024) *SwissProt* tiene 571,282 registros y TrEMBL tiene 248,234,451.
- *SwissProt* agrega anotaciones basadas en evidencia experimental y/o predicha, revisada manualmente, para caracterizar lo mejor posible cada proteína. Por otro lado, TrEMBL proporciona registros anotados computacionalmente, sin curación.

- UniParc: es una base de datos concentradora que proporciona todas las secuencias de proteínas conocidas y únicas.
- UniRef (*UniProt Reference Clusters*): proporciona tres bases de datos agrupadas UniRef100 (combina secuencias idénticas como una entrada de UniProtKB y UniParc como un solo registro), se utiliza el algoritmo *CD-HIT* [Li and Godzik, 2006] para construir UniRef90 y UniRef50, que son agrupaciones de secuencias al 90 % y 50 % de identidad, respectivamente.

UniProt proporciona *Gene Ontology Annotations* y *Gene Ontology Terms* que se basa en la ontología de la relación de los entes y sus participantes, la ontología génica busca asociar un gen y sus productos génicos.

En la ontología génica (GO) se definen tres ontologías (términos): proceso biológico, funciones moleculares y componentes celulares. Las anotaciones representan las relaciones entre los productos génicos y los genes.

Un producto génico puede ser una proteína, de manera tal que una proteína puede tener o no tener una *Gene Ontology Annotation (GO annotation)*. Para cada GO annotation hay un código evidencial y puede ser cambiante si existen nuevos descubrimientos que lo soporten.

Existe toda una teoría de grafos de GO que puede ser estudiada de manera amplia. En este trabajo sólo se hará referencia a los GO IDs para identificar dentro de *SwissProt* aquellas proteínas que sean candidatas a ser un TF.

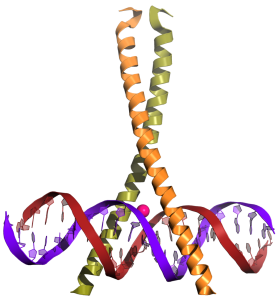
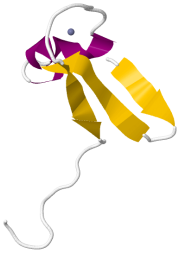
Superclase: Dominios Básicos Clases asociadas: bZIP (Leucine Zipper Factors) bHLH (Helix-Loop-Helix Factors) bHLH-ZIP (Helix-Loop-Helix/Leucine Zipper Factors) NF-1 (Nuclear Factor I) RF-X (Regulatory Factor X) bHSH	Superclase: Dominios de unión al ADN que coordinan el Zinc Clases asociadas: Cys4 zinc finger of nuclear receptor type Diverse Cys4 zinc fingers Cys2His2 zinc domain Cys6 cysteine-zinc cluster Zinc fingers of alternating composition	Superclase: Hélice-giro-hélice Clases asociadas: Homeodomain Paired box Fork head / winged helix Heat shock factors Tryptophan clusters TEA domain
 CREB (1DH3)	 GR (1R4R)	 Pax-6 (6PAX)
Superclase: Beta-Scaffold factors (with minor groove contacts) Clases asociadas: RHR (Rel Homology region) STAT p53 Zipper Factors) MADS box β-Barrel α-helix transcription factors TATA-Binding proteins HMG Heteromeric CCAAT factors Grainyhead Cold-shock domain factors Runt	Superclase: Otros factores de transcripción Clases asociadas: Copper fist proteins HMGI(Y) Pocket domain E1A-like factors AP2/EREBP- related proteins	
 YB-1 (1H95)	 1CO4	

Tabla 2.1: Clasificación TFS por Super Clases y Clases.

Capítulo 3

Modelos de *Deep Learning*

"Aquel que quiebra algo para descubrir lo que es, ha dejado el camino de la sabiduría".
- J.R.R. Tolkien.

3.1. Conceptos fundamentales del *Deep Learning*

Una arquitectura en *Deep Learning*, es un algoritmo formalizado que tiene una entrada, una salida y una función objetivo. Dicha formalización puede estar basada en otras áreas de las ciencias de la computación, la probabilidad, la estadística y las matemáticas avanzadas.

Un modelo en *Deep Learning*, es una arquitectura que ya fue optimizada para la resolución de una tarea específica. Una arquitectura puede tener muchos modelos como producto de la realización de varios entrenamientos.

Una metodología basada en *Deep Learning*, es un conjunto de técnicas y arquitecturas para encontrar el mejor modelo que resuelva una tarea.

Adicionalmente, es necesario definir los siguientes conceptos:

- Tarea: Sea t la abstracción matemática y computacional de un problema a ser resuelto. *e.g.* Tareas de clasificación, detección, reconocimiento y predicción.
- Dominio: Sea D un conjunto de tareas que tienen en común un conjunto de datos de entrada, que son específicos y bien definidos. *e.g.* Predicción de cáncer de mamá, utilizando imágenes de tomografía computarizada, detección de Covid-19 en pacientes sintomáticos usando radiografía de tórax o detección automática en tiempo real en automóviles autónomos.

- Paradigma de aprendizaje: Forma de realizar el proceso de aprendizaje en los diferentes algoritmos basados en IA. Dicha forma está dada por una entrada de datos X, un conjunto de salidas Y esperadas, y un conjunto de salidas obtenidas Y', de esta manera, se tiene que para:

  Supervisado: Conociendo Y, se desea conocer f , tal que: $Y' \approx Y$ donde $Y' = f(X)$.

  No supervisado: Sin conocer Y, se busca f, tal que: $Y' \approx Y$ donde $Y' = f(X)$.

  Semi-supervisado: Se conoce un subconjunto de Y, se busca f, tal que tal que: $Y' \approx Y$ donde $Y' = f(X_c) \bigcup f(X)$.

- Campo: es el conjunto de dominios asociados a una disciplina de las ciencias de la computación. *e.g.* procesamiento de lenguaje natural, visión computacional, procesamiento de voz.

- Training (Entrenamiento): Proceso mediante el cual una arquitectura realiza el proceso de aprendizaje. El entrenamiento tiene dos métricas temporales, por número de épocas de entrenamiento o por número de iteraciones. Una época de entrenamiento se mide a partir del subproceso *forward* hasta finalizar el subproceso *backward*.

- Validation (Validación): Proceso mediante el cual, el modelo obtenido en el entrenamiento es validado con datos "nuevos" (del conjunto de validación) para obtener métricas del desempeño del modelo.

- Testing (Prueba): Proceso que tiene como objetivo medir el desempeño real de un modelo con datos totalmente puros y nuevos a los datos de entrenamiento y validación.

- Inferencia: Proceso donde se evalúa el modelo ante datos conocidos o no conocidos.

- Regresión: Una tarea de regresión, es aquella donde se espera que la salida sea un valor continuo. *e.g.* Predicción del costo de viviendas en una cierta zona de la ciudad, el tiempo de traslado de un alumno a su escuela, predicción de clima y temperatura en la zona sur del país, etc.

- Clasificación: A diferencia de la regresión, la clasificación maneja salidas discretas.

- Predicción: Es una consecuencia directa de haber obtenido un modelo óptimo que realiza una tarea. En clasificación, una predicción es un proceso a posteriori una vez que se tenga un modelo óptimo. Se pueden predecir datos no conocidos o relativamente conocidos para determinar su clasificación.

- Clasificación binaria: Clasificación que tiene como salida dos valores, un verdadero (positivo) o uno falso (negativo).

- Clasificación multiclase: Clasificación que tiene como salidas diferentes valores (clases) mutuamente excluyentes.
- Clasificación multi-etiqueta: Clasificación que tiene como salidas diferentes valores (clases), una misma salida puede apuntar a varias clases.
- Overfitting (Sobreajuste): Estado de un modelo cuando este se sesga a los datos de entrada. Se diagnostica con el análisis del comportamiento de la función de pérdida del modelo en validación, en comparación con la función de pérdida en entrenamiento, o bien con el desempeño en las métricas del modelo en validación y/o prueba.
- Underfitting (Subajuste): Estado de un modelo cuando no tiene buen desempeño en las métricas, ni de entrenamiento, ni de validación. El modelo carece de buena parametrización o diversificación de los datos de entrada.
- Función de pérdida: En el caso de aprendizaje supervisado determina el error entre las salidas esperadas y las salidas obtenidas de un modelo. En el caso de aprendizaje no supervisado, según sea el algoritmo, se evalúa la distancia entre los datos de entrada y los patrones que se desean encontrar. Para la clasificación binaria, se usa la función de pérdida "Entropía Cruzada Binaria" y para la clasificación multiclase, la "Entropía Cruzada Categórica".
- Optimizador: es el corazón de las arquitecturas en *Deep Learning*. Se utiliza para obtener el conjunto de parámetros óptimos y encontrar el mejor modelo posible. El objetivo es minimizar la función de pérdida. *e.g.* Entre los optimizadores más comunes están el *SGD*, *Adam* y *Adagram*.

3.2. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNNs) son las que dan fin a la tercera *IA Winter*, que duró alrededor de 20 años. El concepto de redes neuronales convolucionales se había planteado desde 1980 por Kunihiko Fukushima con su arquitectura *Neocognitron* [Fukushima, 1980] bioinspirada desde los trabajos de investigación de la corteza visual cerebral y de las células encargadas de la visión humana de Hubel y Wiesel, en 1959.

Fue gracias al desarrollo masivo de alta tecnología para el procesamiento de datos en la década de los 2000, que las redes neuronales convolucionales fueron viables para su creación.

Las CNNs tienen como fundamento el proceso de convolución y la retro programación hacia atrás (*backpropagation*).

La convolución es un operador matemático del análisis funcional que permite combinar dos señales para medir la respuesta de una señal $f(x)$ con respecto a otra señal $g(x)$. A la señal $f(x)$ se le llama señal de entrada y a $g(x)$ se le llama filtro o *kernel*.

Al realizar una convolución, lo que se estará obteniendo son un conjunto de características que generalizan a $f(x)$ en respuesta a $g(x)$.

La convolución puede ser llevada a $\Re^2$ hasta $\Re^n$, y puede ser discreta o continua. Para el caso de las CNNs se consideran las datos de entrada como señales discretas y en el caso de que sean imágenes o matrices en $\Re^2$.

Las primeras capas de una arquitectura CNN tienen como función la extracción de características y la capas intermedias se usan para funciones de muestreo; de manera tal, que una CNN es una Red Neuronal Artificial multicapa donde las neuronas son reemplazadas por neuronas convolucionales.

Al resultado de una capa de convolución se llama mapa de activación o mapa de características. Algunas características son obtenidas en diferentes escalas o resoluciones por lo que se usan neuronas de muestreo para obtenerlas.

En la Figura 3.1 se pueden observar el modelo general y tipos de las CNNs.

Cabe mencionar que la operación de convolución es sustituida por una correlación en las CNN, dado que algorítmicamente es más eficiente y más rápida.

A continuación se muestra el desarrollo formal del proceso de convolución y correlación de las CNNs en la ecuaciones 3.1-3.4.

$$G = I * K \tag{3.1}$$

$$G[i,j] = \sum_{u=-k} \sum_{v=-k} I[u,v]K[i-u, j-v] \tag{3.2}$$

$$G = I \oslash K \tag{3.3}$$

$$G[i,j] = \sum_{u=-k} \sum_{v=-k} I[u,v]K[i+u, j+v] \tag{3.4}$$

$$Z^L = W^L \cdot A^{L-1} + b^L \qquad ; \qquad A^L = g^L(Z^L) \tag{3.5}$$

Donde I es la matriz de entrada, K es el filtro, G es el resultado de la convolución o correlación. Z es el equivalente a G y el operador $(\cdot)$ puede ser una convolución o correlación, W es equivalente a K y ese conjunto de pesos es ajustado en el algoritmo de retropropagación. b es el *bias* y g es la función de activación y A es el resultado de aplicar la función de activación a Z. L indica la capa actual y $L-1$ la capa anterior.

A la ecuación 3.5 le llamaremos *forward pass* y a la ecuación 3.6 le llamaremos *backward pass*.

El algoritmo de retropropagación obtiene dW y db en la capa actual L. Por lo que primero se obtiene dZ utilizando la ecuación 3.7 y dA utilizando la ecuación 3.8.

$$dA^L = \frac{\delta L}{\delta A^L}; \quad dZ^L = \frac{\delta L}{\delta Z^L}; \quad dW^L = \frac{\delta L}{\delta W^L}; \quad db^L = \frac{\delta L}{\delta b^L} \tag{3.6}$$

$$dZ^L = dA^L \cdot g'(Z^L) \tag{3.7}$$

$$dA^L = \sum_i \sum_j W \cdot dZ[i,j] \tag{3.8}$$

Las CNNs son las redes neuronales más usadas en el campo de la Visión Computacional, en tareas de clasificación, segmentación, localización y detección de imágenes.

Otros tipo de arquitecturas derivadas de las CNN son: DNN (*Deconvolutional Neuronal Network*), *Depthwise Convolutional*, *Pointwise Convolutional* y *Residual NN*.

3.3. Redes neuronales recurrentes tipo LSTM

Las CNNs presentan el problema del olvido (*catastrophic forgetting*). Básicamente, es una habilidad deficiente para retener las características más antiguas sobre las más recientes y esto sucede debido al uso del algoritmo de retropropagación. Los cambios en las derivadas parciales en las capas más superficiales suelen ser más significativos sobre las primeras capas.

Dado lo anterior, usar CNN para datos secuenciales no espaciales supone un problema, es entonces que en 1986, fue propuesta una nueva arquitectura llamada redes neuronales recurrentes (RNN). Este tipo de redes neuronales fueron ampliamente utilizadas para datos de entrada basados en series de tiempo y procesamiento de lenguaje natural, en tareas como traducción y clasificación de textos.

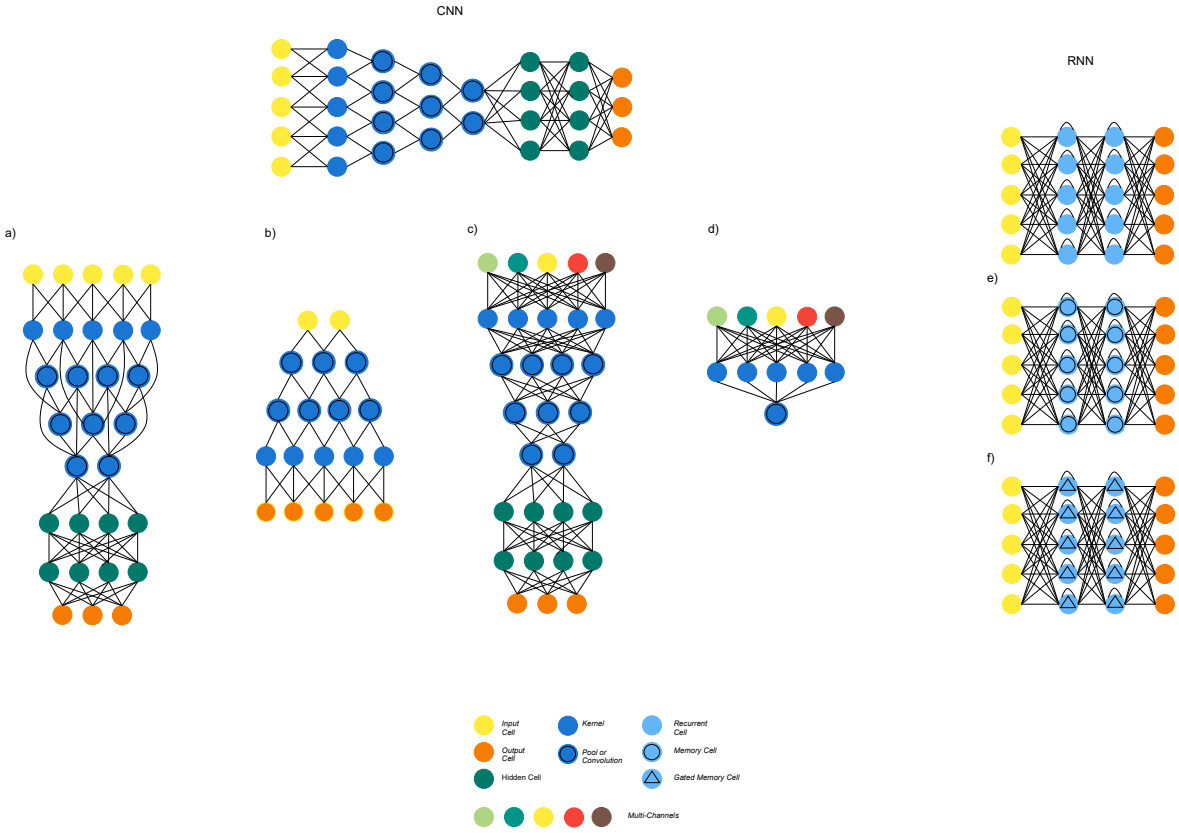


Figura 3.1: Arquitecturas CNNSs y RNNs. En la arquitecturas CNNs: a) *Residual CNN* b) *Deconvolutional Neuronal Network* c) *Depthwise Convolutional* y d) *Pointwise Convolutional*. En las arquitecturas RNNs: e) LSTM y f) GRU.

Las RNN tradicionales presentaron otro inconveniente: El desvanecimiento del gradiente. En 1997 se proponen las redes LSTM y en el 2007 fueron implementadas en las primeras tareas en el procesamiento de lenguaje natural.

Las RNNs tipo *Long short-term memory* (LSTM) resuelven el problema del olvido y el problema del desvanecimiento del gradiente, planteando en sus celdas de memoria la capacidad de tener una memoria a corto plazo y otra a largo plazo.

Una LSTM está construida por celdas de memoria que poseen una entrada, una salida y un estado de la celda. Es en el estado de la celda donde se añaden una compuerta de olvido y una compuerta de actualización de estado, ver Figura 3.2 para más información. Hay otra versión de RNN sin compuerta de olvido llamada GRU.

El modelo matemático para la LSTM, está dado por el cálculo de las puertas i_t (entrada), f_t (de olvido) y o_t (puerta de salida) a través de las siguientes ecuaciones:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i), \tag{3.9}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f), \tag{3.10}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{3.11}$$

Las funciones de activación son sigmoides y h_{t-1} representa la salida de la celda LSTM anterior en el tiempo $t-1$. Finalmente x_t, b_o son las entradas y *bias* en el tiempo actual.

El estado de la celda está dado por el cálculo de c_t (estado actual de la celda) y $\bar{c}_t$ (cálculo inicial del estado actual de la celda).

$$\bar{c}_t = tanh(w_c[h_{t-1}, x_t] + b_c), \tag{3.12}$$

$$c_t = f_t * c_{t-1} + i_t * \bar{c}_t, \tag{3.13}$$

$$h_t = o_t * tanh(c_t) \tag{3.14}$$

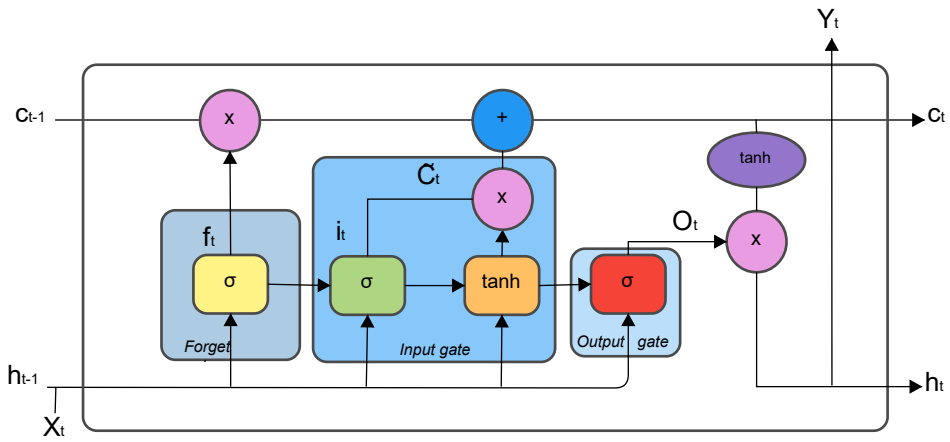


Figura 3.2: Celda de Memoria de una red LSTM. De acuerdo a las ecuaciones 3.9 a 3.11, la celda LSTM está conformada por una puerta de olvido, una de entrada y una de salida. Dada una entrada x_t se tiene una salida y_t, donde se calculan los pares de valores auxiliares, el estado de memoria por las variables c_{t-1} y c_t.

Una BiLSTM [Schuster and Paliwal, 1997] es una versión extendida de una LSTM, donde se tiene una red LSTM para procesar las entradas en dirección *forward* y otra red LSTM que procesa las entradas en sentido contrario o *backward*.

La finalidad principal de BiLSTM es extender el contexto en ambas direcciones y generalizar de manera más amplia los patrones secuenciales. En la Figura 3.3 se muestra una esquematización en el

tiempo y en ambas direcciones de una BiLSTM.

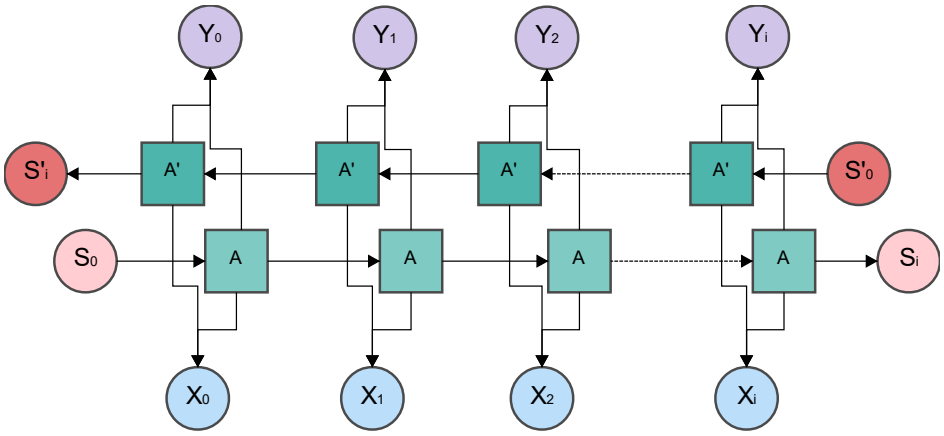


Figura 3.3: Arquitectura BiLSTM. X_i representa el token de entrada y Y_i el token de salida. Es una representación a lo largo del tiempo en donde A y A' son celdas de memoria tipo LSTM.

El resultado final de la BiLSTM está dado por:

$$h^f = LSTM(h_{t-1}, x_{t-1}, c_{t-1}), \tag{3.15}$$

$$h^b = LSTM(h_{t+1}, x_{t+1}, c_{t+1}), \tag{3.16}$$

$$h_i = [h^f, h^b] \tag{3.17}$$

3.4. Mecanismos de atención

La atención es un proceso cognitivo que inspiró el desarrollo de los mecanismos de atención en las redes neuronales y en el *Deep Learning* y son el fundamento de las redes de multi atención como las Transformers.

A pesar de que hubo trabajos pioneros en esta área, fue hasta 2014 que se publicó y formalizó el primer algoritmo de atención en [Graves et al., 2014] aplicado a las Máquinas de Turing Neuronales (NTMs) usando Atención basada en contenido (*Content-based attention*).

Los mecanismos de atención (AMs) surgieron primordialmente para tareas de traducción pero más tarde se emplearon en tareas de visión computacional. Los AMs tiene como objetivo:

1. Mejorar el desempeño del modelo.
2. Trabajar con tamaños variables de entrada.

3. Ayudar a la explicabilidad e interpretabilidad del modelo

El boom de los mecanismos de atención ocurrió con las arquitecturas de tipo autoencoder, donde se usa un codificador (*encoder*) y un decodificador (*decoder*). El *encoder* es una red RNN que construye un vector memoria, que después es tomado por el *decoder* para formar una palabra en el lenguaje destino.

En [Luong et al., 2015] se fundamentan los conceptos de atención global y local. Inicialmente se calcula un vector de alineamiento α_t con la ecuación 3.18 para posteriormente calcular un vector de contexto c_t.

La atención global ocurre cuando todos los vectores de memoria contribuyen en el proceso de formación del vector de contexto c_t (ver ecuación 3.21). En cambio, en la atención global no todos los vectores memoria lo hacen, sino solo aquellos que están a cierta distancia de un momento p_t que está dado por la ecuación 3.19, donde hay otro vector de componentes ν que también es ajustado por la red.

$$a_t = softmax(W \cdot h_t) \tag{3.18}$$

$$p_t = \sigma(\nu W \cdot h_t) \tag{3.19}$$

Considerando el estado oculto del decodificador se tiene que:

$$s_t = f(s_{t-1}, y_{t-1}, c_t), \tag{3.20}$$

$$c_t = \sum_{i=1}^{n} a_{t,i} h_i. \tag{3.21}$$

En la tabla 3.1 se muestran los diferentes tipos de AM reportadas en [Brauwers and Frasincar, 2023].

El AM aditivo [Bahdanau et al., 2014] aplicado a una BiLSTM sigue el modelo matemático:

$$\begin{aligned} h_i &= [h^f, h^b] \\ e_{t,i} = \alpha(s_{t-1}, h_i) &= \nu^T \cdot tanh(W[s_{t-1}; h_i]) \\ \alpha_{t,i} &= softmax(e_{t,i}) \\ c_t &= \sum_{i=1}^{n} a_{t,i} h_i \end{aligned}$$

Tipo de Mecanismo de Atención	***Score function* *score*** (s_t, h_i)	**Taxonomía del Mecanismo de Atención**	**Año de Publicación**
Content-based attention / Similitud *[Graves et al., 2014]*	$cos[s_t, h_i]$	Atención Global y Representación Sencilla	2014
Aditivo / Concatenación [Bahdanau et al., 2014]	$v_a^T \cdot tanh(W_a[s_{t-1}; h_i])$	Atención Global y Representación sencilla	2014
Basado en Localización [Luong et al., 2015]	$\alpha_{t,i} = softmax(W_a \cdot s_t)$	Atención global/local y Representación sencilla	2015
General [Luong et al., 2015]	$s_t^T \cdot W_a \cdot h_i$	Atención global y Representación sencilla	2015
Producto punto [Luong et al., 2015]	$s_t^T \cdot h_i$	Atención global/local y Representación sencilla	2015
Producto punto escalado [Vaswani et al., 2017]	$\frac{s_t^T \cdot h_i}{\sqrt{n}}$	Atención Global, Representación sencilla y self-attentive	2017

Tabla 3.1: Clasificación taxonómica y operativa de los AMs.

Una vez obtenida h_i mediante la BiLSTM, se alimenta el vector de alineamiento a para finalmente calcular $e_{t,i}$, se aplica una función softmax para normalizar los datos entre 0 y 1, que será $\alpha_{t,i}$. Posteriormente, el vector de contexto c_t se obtiene con una suma pesada del producto h_i $\alpha_{t,i}$.

Con este AM lo que se busca es encontrar aquellas características más significativas o que representan patrones que puedan ayudar a la red a mejorar su desempeño.

3.5. Técnicas de regularización para evitar el sobreajuste

El sobreajuste es más común de lo que se cree en los modelos de *Deep Learning*. En particular, en aquellas tareas cuyos datos de entrada presentan las siguientes características:

(a) Los datos son insuficientes para generalizar el modelo.

(b) La distribución de los datos utilizados es altamente variable, por lo que su diversificación presenta un obstáculo. En otras palabras, no existe una manera proba-estadística para diferenciar clases o es demasiado complicado para que el modelo pueda separarlas.

(c) La fuente de los datos no es la apropiada, o no se cuenta con suficiente cantidad de datos reales (validados experimentalmente).

En el caso de los datos de TFs, el gran obstáculo que se presenta es el caso (c) e hipotéticamente hablando también el (b) que posteriormente se hablará en el siguiente capítulo.

La regularización es una estrategia que se usa para eliminar el sobreajuste. Las técnicas de regularización se pueden clasificar según el tipo de incertidumbre [Gal and Ghahramani, 2015, Abdar et al., 2021] en los modelos de *Deep Learning*.

- Regularización Aleatoria: Inherente a los datos, no se puede reducir la incertidumbre aumentando los datos. Relacionada a la incertidumbre aleatoria (propiedad intensiva de los modelos).
- Regularización Epistémica: Relacionado a los parámetros y los hiper-parámetros del modelo, se podría reducir este tipo de incertidumbre aumentando los datos, pero no del todo.

A continuación se muestra en la tabla 3.2 de técnicas de regularización y su clasificación:

Técnica de Regularización	**Tipo de Regularización**	**Objetivo**
Regularización de pesos (L1, L2 y *Elastic Net*)	Epistémica	Pesos del modelo
Dropout	Aleatoria	Neuronas de salida del modelo
Parado Temprano (*Early Stopping*)	Aleatoria	Monitoreo de la función de pérdida.
Batch Normalization	Aleatoria	Salidas en capas intermedias o finales del modelo
Data Augmentation	Aleatoria	Datos de entrenamiento
Elastic Net Convergencia	Epistémica	Función de pérdida, optimizador y tasa de aprendizaje

Tabla 3.2: Técnicas de regularización. La regularización también se puede clasificar por el tipo incertidumbre asociada a la técnica de regularización.

1. L1, L2 y *Elastic Net*.

   La regularización *Ridge* (L1) conjetura que los datos de entrada no poseen una correlación estadística, por lo que introduce un término de penalización a la función de pérdida, con el objetivo de restringir la magnitud de los pesos de la red, y provocar que aquellos pesos menos significativos tiendan a 0 y reducir la cantidad de *features* utilizada por el sistema (ver ecuación 3.26).

   A diferencia de *Ridge*, la regularización *Lasso* (L2) parte de la conjetura que los datos de entrada poseen una correlación estadística, por lo que agregan el cuadrado de los pesos del modelo a la función de pérdida. L2 incentiva la aparición de pesos pequeños y provoca la suavidad de la magnitud de los pesos (ver ecuación 3.23).

   La regularización L2 también se le conoce como decaimiento de pesos (*Weight decay*).

   Para mediar entre los dos enfoques, surge una tercera regularización llamada *Elastic Net* (ver ecuación 3.24), considerando las dos normas de regularización.

$$Loss = Error(y, \hat{y}) + \lambda_1 \sum_{i=1}^{N} |w_i| \tag{3.22}$$

$$Loss = Error(y, \hat{y}) + \lambda_2 \sum_{i=1}^{N} w_j^2 \tag{3.23}$$

$$\begin{aligned} Loss = Error(y, \hat{y}) + \lambda_1 \sum_{i=1}^{N} |w_i| + \lambda_2 \sum_{i=1}^{N} w_j^2 = \dots \\ \dots = Error(y, \hat{y}) + \lambda(\alpha \sum_{i=1}^{N} |w_i| + \frac{1-\alpha}{2} \sum_{i=1}^{N} w_j^2) \end{aligned} \tag{3.24}$$

Se introducen dos nuevos hiperpámetros λ y α.

2. *Dropout*

   En [Gal and Ghahramani, 2015] se demuestra que la técnica de abandono o *dropout* funciona como un estimador de incertidumbre del modelo.

   El *dropout* consiste en apagar aleatoriamente durante el entrenamiento ciertas neuronas con intención de incrementar la generalización del modelo y evitar el sobreajuste.

   Teóricos de la computación señalan que el *dropout* funciona de forma equivalente a una coadaptación. Al aplicar *dropout*, básicamente se está forzando a que la red no dependa de ciertas conexiones.

   El *dropout* sigue la siguiente ecuación:

$$w_{ij} = w_{ij}(1-p) \tag{3.25}$$

   Donde p se vuelve un hiperparámetro de diseño y w_{ij} son los pesos de conexión entre dos neuronas.

3. Parado temprano y monitoreo de la función de pérdida.

   El parado temprano o *Early Stopping* es una técnica de regularización que se basa en el monitoreo de la función de pérdida.

   Normalmente, la función de pérdida de entrenamiento debe decrecer más en comparación con la función de pérdida de validación; sin embargo, cuando esto no sucede así se presenta un sobreajuste en el modelo.

   El parado temprano funciona como una regularización implícita, dado que no es una regularización que afecte a los pesos o conexiones como las técnicas explicadas anteriormente, sino que

simplemente el modelo deja de entrenarse, si la función de pérdida en validación no ha mejorado en una cierta cantidad de épocas, ver Figura 3.4.

El hiperparámetro de diseño será la latencia, que es el número de épocas donde la función de pérdida en validación deberá haber bajado, de lo contrario el modelo dejará de ser entrenado.

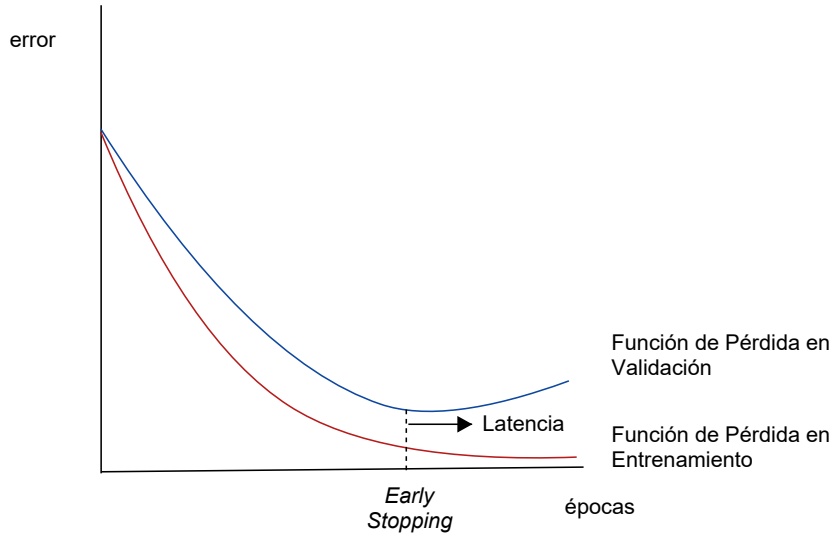


Figura 3.4: Técnica de regularización implícita, *Early Stopping*. Con el hiperparámetro de la tenencia se monitorea la función de pérdida en validación, si esta, no disminuye se deja de entrenar el modelo.

4. *Learning rate*

   Otra técnica para mejorar el desempeño del modelo es encontrar una tasa de aprendizaje (*learning rate*) ideal para que el optimizador encuentre el máximo global donde el modelo obtenga el mejor desempeño.

   Learning Rate Schedules busca variar ese *learning rate* durante el proceso de entrenamiento, reduciendo paulatinamente hasta un mínimo. Los tres principales tipos de *Learning Rate Schedules* son: decaimiento basado en el tiempo, decaimiento basado en pasos y decaimiento exponencial. Para ver el comportamiento de cada una de ellas, vea la Figura 3.5.

   El decaimiento basado en pasos se define por:

$$lr_i = lr_{i-1} * drop^{(\frac{1}{epochs_{drop}})} \tag{3.26}$$

   Donde i es la época actual y $i-1$ es la época anterior o valor inicial. *Drop* es el factor que se va a reducir el *learning rate* y $epochs_{drop}$ es la época donde ocurrirá el cambio del *Learning Rate.*

5. Procesamiento de Lenguaje Natural en la bioinformática.

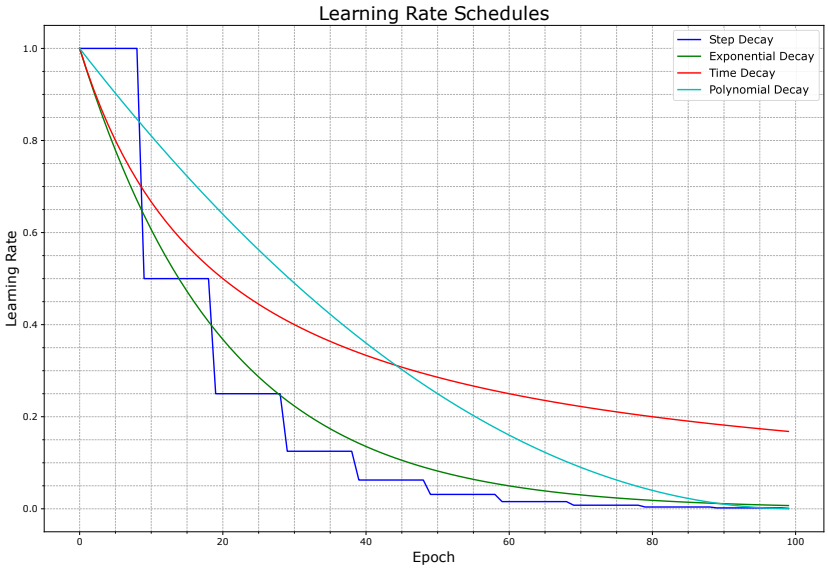


Figura 3.5: Comportamiento de los diferentes *Learning Rate Schedulers*. Se muestra la variación de la tasa de aprendizaje a lo largo de las épocas de entrenamiento.

En 2017 se publicó "*All you need is attention*"[Vaswani et al., 2017] introduciendo una nueva arquitectura basada en mecanismos de atención múltiple llamada *Transformer* y un nuevo paradigma de aprendizaje llamado *self-attention*.

La arquitectura *Transformer* provocó un cisma en la construcción de arquitecturas orientadas al NLP. Al conjunto de modelos basados o inspirados en los mecanismos de atención múltiple aplicados a NLP se les conoce como "Modelos de Lenguaje".

Cuando el modelo de lenguaje se extiende a grandes volúmenes de datos, se les conoce en *Large Language Models* (LLMs).

Una red de tipo *Transformer* está compuesta por un *encoder* y *decoder*, los datos introducidos son codificados posicionalmente y entran a las unidades de multi-atención creando un mapa algebraico de los tokens de entrada y la relación entre ellos, ver Figura 3.6

Cuando se concentran los esfuerzos para mejorar el desempeño de *encoder*, la arquitectura se vuelve ideal para tareas de clasificación, y de forma contraria, cuando se da prioridad al desempeño del *decoder*, la arquitectura es ideal para tareas de generación.

Existen dos grandes familias pioneras de modelos de lenguaje, la familia *BERT* [Devlin et al., 2018] y la familia *GPT* [Radford and Narasimhan, 2018], una de Google y la otra de OpenAI correspondientemente. *BERT* muestra excelente desempeño en tareas de clasificación y *GPT* en tareas de generación.

La arquitectura *BERT* realiza una lectura hacia atrás y hacia adelante, esto ayuda a capturar aún

más relaciones entre palabras y ser una buena opción para tareas de clasificación.

Google usó modelos basados en *BERT* en sus buscadores y servicios principales, y posteriormente *DeepMind* propuso un modelo para predecir estructuras de proteínas en 3D basado en una *Transformer ad-hoc* llamada *Evorformer*, *AlphaFold2*. [Jumper et al., 2021]

La arquitectura *GPT* [Radford and Narasimhan, 2018, OpenAI et al., 2023] fue desarrollada por OpenAI en sus 4 versiones y es altamente implementada en tareas de generación.

En *Transforming the Language of Life* [Nambiar et al., 2020] se propone una metodología para considerar el lenguaje de aminoácidos como un modelo de lenguaje de longitud 20. De manera tal que cada letra de aminoácidos se vuelve un *token*, una palabra. Ahí mismo se proponen dos tareas, una de clasificación por familias de proteínas y una segunda tarea de predicción de proteína a proteína (PPI) usando una red *roBERTa* [Liu et al., 2019].

Actualmente, ya hay muchas aplicaciones de los modelos de lenguaje en la bioinformática, principalmente en tareas de generación como: síntesis de nuevas proteínas, diseño de nuevos fármacos, etc.

"*Así como los modelos de idiomas de IA pueden aprender las relaciones entre las palabras en una oración, nuestro objetivo es que las redes neuronales entrenadas con datos de estructura molecular puedan aprender las relaciones entre los átomos en las moléculas del mundo real*".

- Ola Engkvis.

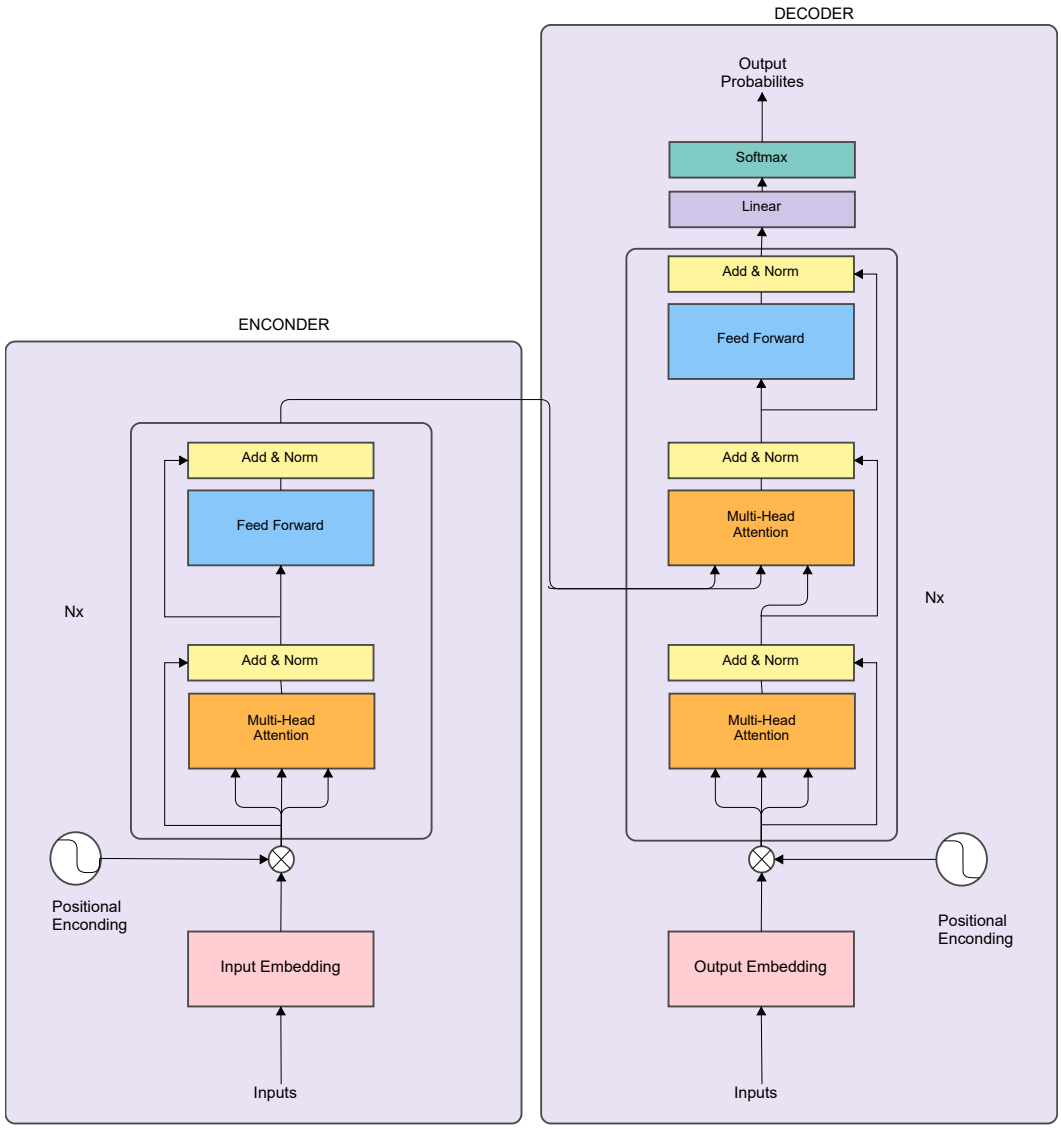


Figura 3.6: Arquitectura *Transformer*. El encoder recibe la entrada que es codificada posicionalmente para posteriormente entrar a los mecanismos de atención. Esta unidad de codificación puede ser repetida n veces para posteriormente mandar la última salida a una celda de mecanismos de multi atención del decoder.

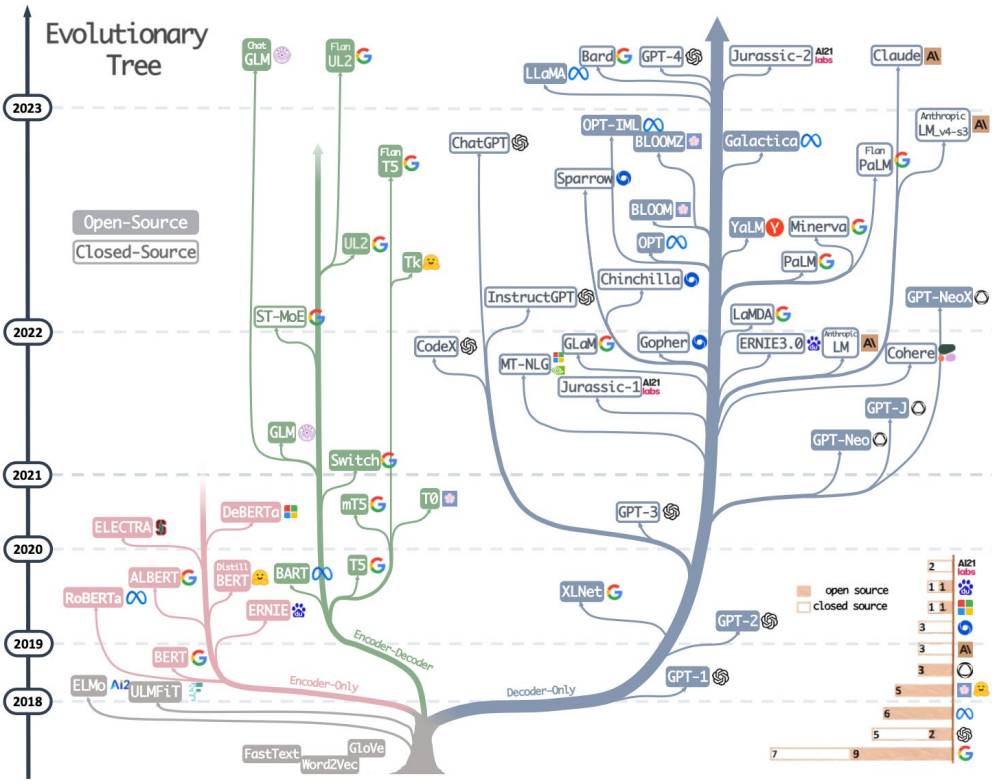


Figura 3.7: Evolución y *timeline* de los LLMs y sus diferentes familias. Imagen obtenida de [Yang et al., 2023].

Capítulo 4

Hipótesis y problemática

Áreas de oportunidad para modelos basados en Deep Learning en la Bioinformática.

"Si buscas la verdad, podrás encontrar confort al final; si buscas confort, no encontrarás ni verdad ni confort."
- C.S. Lewis

4.1. Problemas comunes en modelos para la predicción/clasificación en la Bioinformática

Los problemas comunes se pueden dividir en dos tipos: los relacionados a la propiedad cualitativa de los datos y la propiedad cuantitativa. Entiéndase por cualitativa aquella asociada al origen de los datos y sesgos que podrían existir dada su naturaleza y entiéndase por cuantitativa a la cantidad y su división dentro de cada clase.

1. Sesgo por homología de secuencias.

    - **H1**: Hipotéticamente hablando, hay una posibilidad de que exista un sesgo por homología de secuencias en las tareas de clasificación de los TFs o de cualquier otra clasificación funcional y/o estructural de proteínas.

Esto se debe a que gran cantidad de los datos utilizados para los algoritmos de clasificación tanto en ML como en DL son inferidos virtualmente, es decir no existen suficientes evidencias experimentales que comprueben todas y cada una de las proteínas a las que se les atribuye ciertas características funcionales de regulación o incluso estructurales.

A este problema se le puede llamar "sesgo por homología de secuencias", dado que es mediante la homología que se le atribuyen todas esas características y entonces al trabajar datos que tienen como fuente una base teórica y no experimental se puedan cometer errores de solo caracterizar dichos patrones ya conocidos y que se fueron transfiriendo por homología.

En [Martinez-Liu et al., 2021] se mapean la cantidad TFs versus el tamaño del proteoma de cientos de organismos de arqueas y bacterias. En contrasentido, uno debería de esperar que entre más grande el tamaño del proteoma, la cantidad de TFs debería ser mayor, sin embargo, esto no sucede así.

La distribución de TFs se concentra entre 1,000 a 4,000 ORFs de tamaño del proteoma y a pesar de que se llega a duplicar el tamaño del proteoma, la cantidad de TFs no presenta una correspondencia similar a esos tamaños.

Dicha evidencia nos hace pensar en la existencia de este sesgo por homología, probablemente porque no se están identificando otros tipos o familias de TFs.

Esta incertidumbre inherente de la naturaleza de los datos se puede validar primeramente por consenso algorítmico o mediante comparación de predicciones tomando como *groundtruth* datos experimentales.

Por lo anterior, se pueden establecer las siguientes hipótesis:

- **H2**: Si se comparan dos algoritmos cuya metodología es totalmente diferente (eg. uno basado en ML y otro basado en DL) y la distribución de las predicciones de TFs sigue un mismo patrón, se establece un consenso algorítmico de que no existe sesgo por homología de secuencias o es tan pequeño que se puede considerar despreciable.
- **H3**: Si el rango de cobertura de las predicciones por un algoritmo cubren un alto porcentaje de TFs validados experimentalmente, se puede decir que dicho algoritmo realiza eficientemente dicha tarea y el sesgo por homología es despreciable o es casi nulo.

2. Cantidad de datos y clases desbalanceadas.

   Dos problemas en tareas de clasificación son la cantidad de datos disponibles y la cantidad de datos que se tienen por clase.

   En las tareas de clasificación y predicción de TFs estos dos problemas se vuelven relevantes, en primer lugar porque la cantidad de datos que se tienen de TFs y No TFs es totalmente desbalanceada, la relación es de 1:10.

   Para sosegar dicho problema se puede:

   - Obtener una muestra aleatoria de la clase predominante del mismo tamaño de la otra clase.
   - Crear datos virtuales de la clase minoritaria para tener más ejemplos y permitir la generalización usando *data augmentation*.
   - Utilizar métricas para medir el desempeño real de un modelo de clasificación con clases desbalanceadas como la precisión, sensibilidad, especificidad, F1-Score, y AUC, MCC (*Matthews correlation coefficient*). Ver en el apéndice el cálculo de cada métrica.

   Se puede combinar cualquiera de las estrategias ya mencionadas, e incluso se puede variar la relación de cantidad de datos entre No TFs y TFs.

   - **H4**: Explorar cualquiera de las estrategias mencionadas en este apartado puede generar un modelo no susceptible a depender de la cantidad de los datos.

3. Interpretabilidad y confiabilidad de modelos.

   Otros problemas grandes de los modelos en *Deep Learning* son la falta de interpretabilidad y la confiabilidad de las predicciones.

   La interpretabilidad es un problema abierto en cual se han realizado intentos los últimos 10 años, sin embargo la dimensionalidad y la complejidad algorítmica que conllevan los modelos basados en DL es tan grande que no existe un método que se pueda aplicar para darle una interpretación en todas sus partes.

   Es por eso que los trabajos de investigación de interpretabilidad van más asociados a mapear los procesos de atención que tiene un modelo en distintas capas o bien tratar de ponderar y estudiar las características que obtiene el modelo para realizar una tarea de clasificación.

   Aunado a la interpretabilidad, se tiene la confiabilidad del modelo. La confiabilidad se define como la certidumbre de que un modelo realice una tarea de manera eficiente y efectiva.

La confiabilidad se mide por el grado de incertidumbre de un modelo. En el caso de las tareas de clasificación - predicción, existen diversas formas de hacerlo, pero básicamente se pueden clasificar en *aposteriori* o *apriori*.

- **H5**: Aplicar un análisis de confiabilidad e interpretabilidad a un modelo de clasificación de TFs demostraría su capacidad predictiva y robustez.

4.2. Rendimiento de algoritmos de *Machine Learning* en predicción de TFs

El primer análisis de este trabajo tiene que ver con la comparación de los modelos basados en ML y la medición de su desempeño usando métricas para clases desbalanceadas. Dicho análisis está reportado en [Ledesma-Dominguez et al., 2024] y en el apéndice Tabla A.3 y A.4 de este trabajo, donde se describe con más detalle.

Inicialmente, se acondicionan las entradas y se procesan en una representación basada en bigramas, para generar un modelo probabilístico que nos indica la estimación de dos elementos sucesivos en una secuencia dada.

Los algoritmos basados en ML empleados son: (1) Clasificador bayesiano ingenuo Gaussiano, (2) Regresión logística, (3) Máquina de soporte vectorial, (4) *Random Forest* y (5) Árboles de decisión.

En todos los casos se probó con o sin reducción de dimensionalidad mediante *Principal component analysis* (PCA), tomando 40,000 secuencias y dividendo aleatoriamente, 80 % para entrenamiento y 20 % para validación.

La máquina de soporte vectorial fue la que obtuvo el mejor desempeño, alcanzando un máximo de 88.79 % en F1-Score y 88.62 %, en AUC. Ambas métricas se obtienen sin considerar PCA.

En este análisis se pueden concluir los siguientes puntos:

- A pesar de que el PCA está hecho para reducir la dimensionalidad de los datos y ayudar al modelo a generalizar, el desempeño del modelo se ve realmente comprometido si se usa, por lo que seguir la filosofía del *Deep Learning* parece dar mejores resultados.

- El estado del arte de los modelos basados en DL presentan un mejor desempeño en la clasificación de TFs, por lo que explorar, investigar y aplicar modelos basados en DL, es una ruta viable.

4.3. Crítica a *DeepTFactor* y *TFNet* sobre las predicciones de TFs

Las arquitecturas actuales publicadas en la literatura para la clasificación de TFs son *DeepTFactor* [Kim et al., 2020] y *TFNet* [Du et al., 2023] descritas en la introducción de este trabajo.

En este sentido, *DeepTFactor* es un modelo que presenta *overfitting*, dado que la función de pérdida en validación presenta saltos abruptos a lo largo de las épocas de entrenamiento.

La inestabilidad de las funciones de pérdida en validación, como se menciona en el capítulo anterior, es un fuerte indicativo de presencia de sobreajuste en el modelo, por lo que es conveniente aplicar estrategias de regularización para eliminar este problema.

También en *DeepTFactor* se utiliza *CD-HIT* para obtener modelos en diferentes rangos de similitud de secuencia; sin embargo, en contrasentido al uso del *Deep Learning* y su filosofía, el eliminar datos redundantes de una misma secuencia que son altamente similares se elimina el sentido de la generalización a través de analizar y tener varios ejemplos de una misma secuencia, que es justamente lo que hace que *Deep Learning* tenga un buen rendimiento.

Por otro lado, *TFNet* presenta mejores métricas de desempeño que *DeepTFactor*, aumentando casi en unidad de cada una de ellas. Sin embargo, se tiene que:

1. El costo computacional y costo temporal del modelo es muy alto:

   Se requiere un poder de cómputo adicional para construir las matrices PSSM de las secuencias de entrada y además se requiere de más tiempo porque las matrices PSSM necesitan de MSAs para encontrar sus valores.

   Por lo que, tanto el entrenamiento y la inferencia del modelo se vuelve inflexible y poco portable.

2. Uso comprometedor de las PSSM y mezcla de clases:

   Si se usan como entrada matrices PSSM en lugar de secuencias, se pueden mezclar clases que en la clasificación binaria son mutuamente excluyentes, es decir, existe una probabilidad de que una matriz PSSM de una secuencia no TF sea parecida o igual a una matriz de PSSM de una secuencia TF.

3. Pocos datos de prueba:

   Las métricas de desempeño mostradas de *TFNet* son obtenidas en pocos datos de prueba solamente en organismos modelo para procariotes, tales como E. coli y para eucariotes, H. sapiens. Por qué evaluar el modelo en dos organismos y en todo el conjunto extendido de datos no es significativo.

4. La CNN no funciona como extractor de características.

   En *TFNet* se usa una red CNN para acondicionar las entradas basadas en PSSMs y ser ingresadas a una red BiLSTM. La CNN ocupada no funciona como un extractor de características por lo que no es una red híbrida, sino una red BiLSTM con mecanismos de atención.

4.4. Objetivos

Analizadas las problemáticas planteadas al inicio de este capítulo y sus diferentes hipótesis, vale la pena plantear los siguientes objetivos de este trabajo de investigación:

O1 Comprobar la H1 estudiando los modelos del DL conocidos y los algoritmos tradicionales en ML para generar un consenso algorítmico planteado en H2.

O2 Proponer un modelo de clasificación de TFs basado en DL que evite la presencia de *overfitting*, generando predicciones más certeras aplicando estrategias de regularización (planteadas en H4) sin reducir las excelentes métricas de desempeño del modelo.

O3 Realizar un análisis cualitativo del modelo generado en O2 y planteado en H5 para dar respuesta a H3, para de esta manera medir la incertidumbre y la confiabilidad del modelo.

De manera adicional:

O4 Extender el modelo generado en O2 a otras tareas relacionadas con la regulación celular como factores de virulencia, para de esta manera comprobar la flexibilidad del modelo para el reentrenamiento.

O5 Generar un nuevo modelo de lenguaje para clasificación de TFs basado en alguna familia Transformer y compararlo con el generado en O2.

4.5. Selección de organismos de referencia

Para dar cumplimiento al O3 fueron seleccionados tres organismos de referencia del reino Fungi, dado que son los más estudiados en el área y se conocen las redes de regulación de sus TFs y sus genes diana.

- *Saccharomyces cerevisiae* (SC): Es un hongo unicelular comúnmente llamado levadura de cerveza utilizado en la industria de alimentos. Fue el primer organismo eucariota con un genoma completamente secuenciado.

  Se utiliza comúnmente como organismo de referencia por su manejo versátil en experimentación. Tiene 16 cromosomas y alrededor de 6,300 genes. A través de la fisiología comparada podemos trasladar diferentes características de su maquinaria molecular a organismos más complejos. Tiene una similitud del 23 % con el genoma humano.

- *Neurospora crassa* (NC): Es un hongo moho que al igual que SC es un organismo modelo con alrededor de 10,000 genes y 6 cromosomas. Se producen cepas con diferentes mutaciones de sus genes para medir el impacto fisiológico y molecular. Su hábitat natural es la superficie de los panes.

- Aspergillus nidulans (AN): Es un hongo filamentoso y homotálico con alrededor de 9,500 genes y 8 cromosomas. Su forma de reproducción sexual es óptima para cruzar diferentes hongos, es objeto de estudio para reparaciones del ADN, mutaciones y patogénesis.

Para el análisis de TFs se tomaron para SC 304 TFs de la base de datos YEASTRACT [Teixeira et al., 2022], para AN 62 TFs y NC 75 TFs tomados de la colección en [Hu et al., 2018].

4.6. Distribución y relevancia de predicciones.

Para dar cumplimiento al O1 y buscar el consenso algorítmico, se realizó una comparación exhaustiva entre tres modelos: (1) usando *PFAM* como algoritmo basado en ML, (2) usando un método estructural del ADN (Hélice-Giro-Hélice) y (3) usando el modelo de inferencia de *DeepTFactor* reportados en [Ledesma et al., 2022].

Dicho análisis demostró que los métodos se pueden complementar para recuperar más TFs y por consenso algorítmico se puede afirmar que dicho TF predicho está de cierta manera validado por diferentes vías algorítmicas.

Con respecto a la verificación de la H1 y H2, se puede afirmar que el sesgo por homología de secuencias es despreciable para tareas de clasificación de TFs, y esto se demuestra comparando la distribución de TFs predichos por diferentes vías algorítmicas, ver la gráfica de la Figura 4.1.

En la Figura 4.2 se muestra un diagrama de Venn de los dos métodos de predicción en 355 proteomas de hongos, en donde alrededor del 75 % se comparten las predicciones de TFs con respecto a *PFAM*.

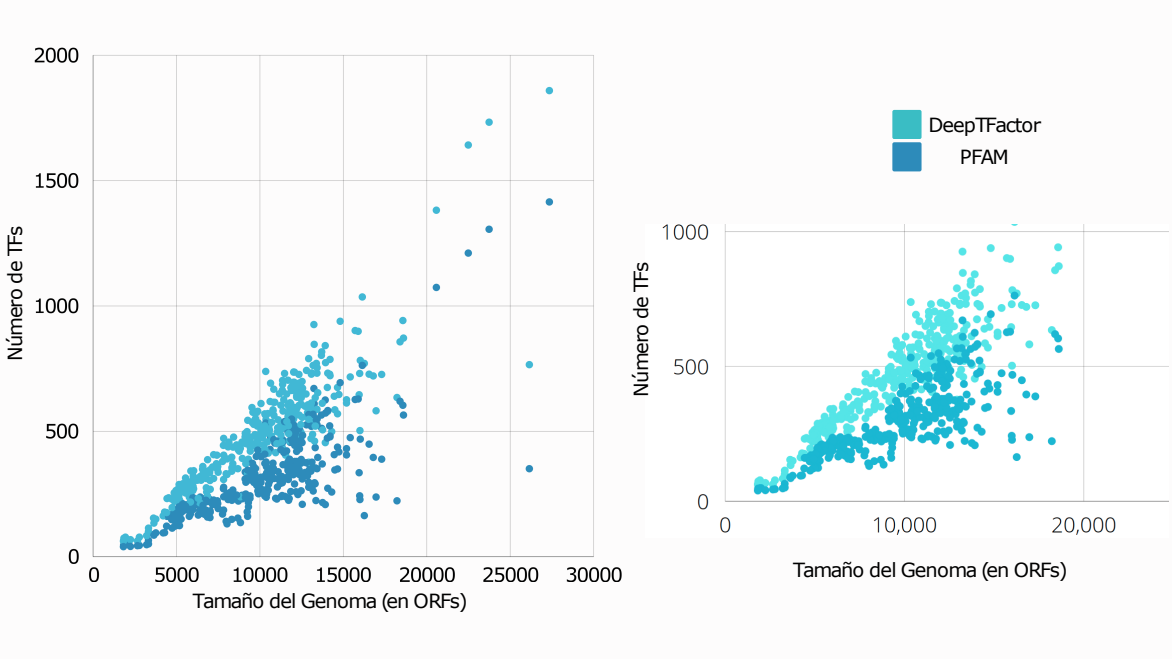


Figura 4.1: Distribución de predicciones de TFs de PFAM y *DeepTFactor* en 355 proteomas de hongos. Se puede observar que por diferentes vías algorítmicas se establece un consenso de que la distribución de predicciones es similar. La única diferencia notable es que la cantidad de predicciones por *DeepTFactor* tiende estar un poco más arriba sobre PFAM, es decir, encuentra más posibles TFs.

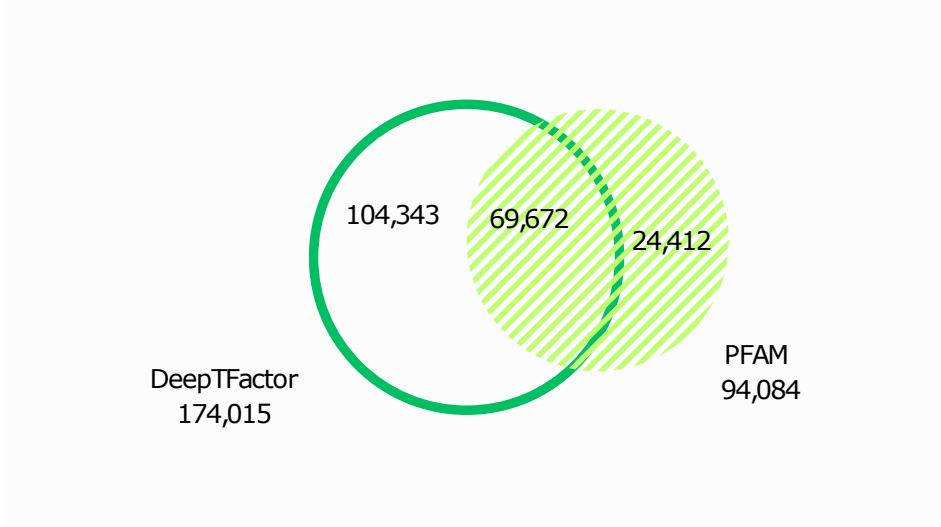


Figura 4.2: Diagrama de Venn de las predicciones de TFs por *DeepTFactor* y PFAM.

Capítulo 5

Metodología y Resultados

"Un hombre provisto de papel, lápiz y goma, y con sujeción a una disciplina estricta, es en efecto una máquina de Turing universal."
- Alan Turing

5.1. Flujo de trabajo para la predicción de los factores de transcripción.

En el capítulo anterior se cumplió el objetivo O1 y en este capítulo se dará cumplimiento a O2 y O3 que dan respuesta a las hipótesis H4 y H3, respectivamente.

La metodología empleada sigue el flujo de trabajo o *workflow* publicado en [Ledesma-Dominguez et al., 2024] (ver apéndice Figura A.1). Donde se propone una nueva arquitectura híbrida tomando las ventajas de las CNN y las BiLSTM, que se nombró *DeepReg* (*Deep Regulation*).

DeepReg es el producto de aplicar diversas técnicas de regularización anteriormente descritas para evitar el sobreajuste y aumentar la confiabilidad de las predicciones. Adicionalmente, también se agregó un mecanismo de atención aditivo descrito en el capítulo 3.

Las primeras etapas del *workflow* son para obtener una base de datos robusta aplicando los criterios de *Gene Ontology Annotations* descritos en [Ledesma-Dominguez et al., 2024]. Una vez obtenido los dos conjuntos de datos, tanto de TFs y No TFs, se aplican las validaciones en tamaño y exclusión de secuencias con aminoácidos no convencionales.

Posteriormente, se realiza un proceso de acondicionamiento de las secuencias de entrada, basado en la tokenización y el padding, la primera se lleva a cabo para construir matrices binarias que puedan ser interpretadas por la Cuádruple CNN y la segunda técnica, para estandarizar a un mismo tamaño cada

secuencia.

Se separan los datos en tres conjuntos para entrenamiento, validación y prueba, siguiendo una proporción de $9 : 0.9 : 0.1$. Durante el proceso de entrenamiento de la arquitectura propuesta, se evalúa heurísticamente el mejor conjunto de hiperparámetros para obtener el modelo con mejor desempeño.

Una vez obtenido el mejor modelo seleccionado por sus mejores métricas, se guardan los pesos obtenidos y la arquitectura final con sus hiperparámetros, para validar el modelo contra datos experimentales.

Finalmente, se propone un análisis cualitativo de predicciones tipo *Bias-Variance Tradeoff* entre las predicciones de los organismos de referencia de *DeepTFactor* y *DeepReg* y un análisis de cobertura.

5.2. Arquitectura propuesta: *DeepReg*

DeepReg consta de 4 módulos basados en arquitecturas DL (ver apéndice Figura A.2). Cada módulo aporta ciertas funciones necesarias para la tarea de clasificación de TFs, el análisis cuantitativo de aportación al rendimiento del modelo se llama *Ablation Analysis* (ver apéndice Tabla A.5).

(a) Módulo 4 CNN (Extracción de características)

Para extraer características se utilizó una red CNN cuádruple paralelizada. Cada CNN tiene un tamaño de filtros diferentes (dos de sobremuestreo y dos de submuestreo).

Adicionalmente, cada CNN tiene una profundidad de tres capas convolucionales 2D y al final se concatenan las salidas de las cuatro CNN. Dicho resultado se puede representar de la siguiente manera:

$$O_{CNN_n} = Conv2D(Conv2D(Conv2D(I))) \tag{5.1}$$

$$O_{4CNN} = Concat(O_{CNN_n}1, O_{CNN_2}, O_{CNN_3}, O_{CNN_4}) \tag{5.2}$$

Tal y como se expresa en la ecuación 5.2, a la salida de la concatenación se aplica de nueva cuenta una capa de convolución 1D para reducir la dimensionalidad y un proceso de muestreo tipo *max pooling* para obtener las características más significativas.

Cabe mencionar que se hicieron otros experimentos integrando más redes CNN y más capas convolucionales, sin embargo, el desempeño no cambió significativamente.

(b) Módulo BiLSTM (Gramática regulatoria)

A la salida del módulo convolucional se conectó una red BiLSTM con 128 unidades o celdas de memoria. La red BiLSTM tiene como objetivo predecir un aminoácido en el tiempo $t+1$ teniendo como entrada un aminoácido en el tiempo t; por ejemplo, qué tan probable es que el aminoácido K aparezca después de un aminoácido S o del conjunto de todos los aminoácidos incluyendo K mismo.

En particular, en *DeepReg*, la red BiLSTM recibe un vector de características producto de aplicar el módulo convolucional. Usando la misma analogía, la red BiLSTM predice una característica con base en otra característica anterior. Por lo que se puede denotar el proceso de la BiLSTM de la siguiente manera:

$$LSTM_{forward} = \sum_{i=1}^{512} P(f_i|f_j) \tag{5.3}$$

$$LSTM_{backward} = \sum_{j=512}^{1} P(f_j|f_i) \tag{5.4}$$

Donde f_i es la característica analizada en el tiempo $t+1$ y f_j es la característica o conjunto de características en tiempo anterior t, y se obtienen todas las probabilidades condicionales de las 512 características.

También se exploraron otras opciones, tales como cambiar el número de celdas de memoria a 64 o 512.

(c) Módulo de Mecanismo de Atención

Se utilizó el mecanismo de atención "aditivo" o también conocido como de Bahdanau, que toma como entrada las características obtenidas en *forward* y en *backward* de la red BiLSTM.

La salida del módulo BiLSTM con el mecanismo de atención incrustado se puede describir con la siguiente ecuación:

$$O_{LSTM} = AM(LSTM_{forward}(O_{4CNN}), LSTM_{backward}(O_{4CNN})) \tag{5.5}$$

El funcionamiento de AM radica en darle mayor peso a aquellas características que destacan más en la caracterización de una secuencia de aminoácidos de entrada.

(d) Etapa de clasificación

Finalmente, a la salida de los módulos anteriores se conecta una red MLP o también llamada *Fully Connected* (FC) para clasificación binaria.

Es en este módulo donde se aplica la regularización *Elastic Net* y se mide el nuevo *learning rate schedules* con base en los hiperparámetros y a la monitorización definidos.

La salida de *DeepReg* quedaría definida como:

$$O_{DeepReg} = LR(R_{ElasticNet}(FC(O_{BiLSTM}))) \tag{5.6}$$

5.3. Construcción y análisis de *DeepReg*.

Durante el entrenamiento se fueron explorando diversas técnicas para mejorar aún más el modelo entre las que destacan:

a) Super convergencia: Para la exploración de las posibles mejores tazas de aprendizaje que el modelo puede utilizar a través del análisis de *Cyclical learning rate* con *One-Cycle Policy*, descrito en [Smith, 2017].

b) *Data Augmentation*: Se exploró la posibilidad de crear secuencias artificiales de la clase menos representada (TFs), creando variaciones de traslación, inversión, inserción y eliminación. Sin embargo, el modelo no obtuvo mejoras significativas y además se corre un alto riesgo de introducir patrones no deseables que biológicamente inhiban la función de la proteína como TF.

c) Eliminar redundancia de datos: Siguiendo la misma lógica de trabajos previos, se intentó entrenar el modelo utilizando la reducción de datos por similitud de secuencias usando el programa *CD-HIT*; sin embargo, como se comentó en el capítulo anterior, el uso de *deep learning* establece que la redundancia de datos no representa una limitante, justo al contrario.

La estructura del modelo se puede ver en la Tabla 5.1:

La función de activación usada fue ReLU y los hiperparámetros usados fueron Tabla 5.2:

Las métricas del desempeño del modelo se pueden analizar en el apéndice, la Tabla A.2 y la Figura A.3

Es importante mencionar que se logra estabilizar la función de pérdida en la validación y que *DeepReg* mejora en tres de las cinco métricas de clases desbalanceadas que *DeepTFactor*.

Num	Capa (Tipo)	Forma del tensor	Parametros	Conectado a	Modulo
1	I1 (Input Layer)	[None, 1000,21,1,0]	0	Sin conexión	
2	I2 (Input Layer)	[None, 1000,21,1,0]	0	Sin conexión	
3	I3 (Input Layer)	[None, 1000,21,1,0]	0	Sin conexión	
4	I4 (Input Layer)	[None, 1000,21,1,0]	0	Sin conexión	
5	CNN1_1 (Conv2D)	[None, 997,1,128]	10880	1	4CNN Layer 1
6	CNN2_1 (Conv2D)	[None, 989,1,128]	32384	2	4CNN Layer 1
7	CNN3_1 (Conv2D)	[None, 985,1,128]	43136	3	4CNN Layer 1
8	CNN4_1 (Conv2D)	[None, 995,1,128]	16256	4	4CNN Layer 1
9	Batch Normalization1_1	[None, 997,1,128]	512	5	4CNN Layer 1
10	Batch Normalization2_1	[None, 989,1,128]	512	6	4CNN Layer 1
11	Batch Normalization3_1	[None, 985,1,128]	512	7	4CNN Layer 1
12	Batch Normalization4_1	[None, 995,1,128]	512	8	4CNN Layer 1
13	Activación RELU1_1	[None, 997,1,128]	0	9	4CNN Layer 1
14	Activación RELU2_1	[None, 989,1,128]	0	10	4CNN Layer 1
15	Activación RELU3_1	[None, 985,1,128]	0	11	4CNN Layer 1
16	Activación RELU4_1	[None, 995,1,128]	0	12	4CNN Layer 1
17	Dropout1_1	[None, 997,1,128]	0	13	4CNN Layer 1
18	Dropout2_1	[None, 989,1,128]	0	14	4CNN Layer 1
19	Dropout3_1	[None, 985,1,128]	0	15	4CNN Layer 1
20	Dropout4_1	[None, 995,1,128]	0	16	4CNN Layer 1
21	CNN1_2 (Conv2D)	[None, 994,1,128]	65664	17	4CNN Layer 2
22	CNN2_2 (Conv2D)	[None, 982,1,128]	131200	18	4CNN Layer 2
23	CNN3_2 (Conv2D)	[None, 982,1,128]	65664	19	4CNN Layer 2
24	CNN4_2 (Conv2D)	[None, 990,1,128]	98432	20	4CNN Layer 2
25	Batch Normalization1_2	[None, 994,1,128]	512	21	4CNN Layer 2
26	Batch Normalization2_2	[None, 982,1,128]	512	22	4CNN Layer 2
27	Batch Normalization3_2	[None, 982,1,128]	512	23	4CNN Layer 2
28	Batch Normalization4_2	[None, 990,1,128]	512	24	4CNN Layer 2
29	Activación RELU1_2	[None, 994,1,128]	0	25	4CNN Layer 2
30	Activación RELU2_2	[None, 982,1,128]	0	26	4CNN Layer 2
31	Activación RELU3_2	[None, 982,1,128]	0	27	4CNN Layer 2
32	Activación RELU4_2	[None, 990,1,128]	0	28	4CNN Layer 2
33	Dropout1_2	[None, 994,1,128]	0	29	4CNN Layer 2
34	Dropout2_2	[None, 982,1,128]	0	30	4CNN Layer 2
35	Dropout3_2	[None, 982,1,128]	0	31	4CNN Layer 2
36	Dropout4_2	[None, 990,1,128]	0	32	4CNN Layer 2
37	CNN1_2 (Conv2D)	[None, 979,1,128]	262272	33	4CNN Layer 3
38	CNN2_2 (Conv2D)	[None, 979,1,128]	65664	34	4CNN Layer 3
39	CNN3_2 (Conv2D)	[None, 979,1,128]	65664	35	4CNN Layer 3
40	CNN4_2 (Conv2D)	[None, 979,1,128]	196736	36	4CNN Layer 3
41	Batch Normalization1_3	[None, 979,1,128]	512	37	4CNN Layer 3
42	Batch Normalization2_3	[None, 979,1,128]	512	38	4CNN Layer 3
43	Batch Normalization3_3	[None, 979,1,128]	512	39	4CNN Layer 3
44	Batch Normalization4_3	[None, 979,1,128]	512	40	4CNN Layer 3
45	Activación RELU1_3	[None, 979,1,128]	0	41	4CNN Layer 3
46	Activación RELU2_3	[None, 979,1,128]	0	42	4CNN Layer 3
47	Activación RELU3_3	[None, 979,1,128]	0	43	4CNN Layer 3
48	Activación RELU4_3	[None, 979,1,128]	0	44	4CNN Layer 3
49	Dropout1_3	[None, 979,1,128]	0	45	4CNN Layer 3
50	Dropout2_3	[None, 979,1,128]	0	46	4CNN Layer 3
51	Dropout3_3	[None, 979,1,128]	0	47	4CNN Layer 3
52	Dropout4_3	[None, 979,1,128]	0	48	4CNN Layer 3
53	Concatenación	[None, 979,1,512]	0	45,46,47,48	
54	CNN5 (Conv1D)	[None, 979,1,512]	262656	53	
55	Batch Normalization5	[None, 979,1,512]	2048	54	
56	Activación RELU5	[None, 979,1,512]	0	55	
57	*max pooling* (*max pooling* 2D)	[None, 489,1,512]	0	56	
58	Series de Tiempo	[None, 489,1,512]	0	57	BiLSTM
59	BiLSTM	[None, 256]	656384	58	BiLSTM
60	AM	[None, 256]	0	59	AM
61	Fully Connected	[None, 1]	257	60	FC

Tabla 5.1: Estructura de *DeepReg*. Son 1,981,441 parámetros de los cuales 1,997,345 son parámetros entrenables y 4,096 paramateros no entrenables.

Hyper Parameter	*Values*
Learning rate Schedule	patience = 10
	$min_{lr} = 0.00001$
	factor = 0.1
	learning rate = 0.001
Early stopping	monitor = loss
	validation patience = 15
Batch size	128
Epochs	80
Dropout	0.3, 0.5 y 0.7 por cada layer en modulo 4CNN

Tabla 5.2: Hiperparámetros de *DeepReg*

Dicho análisis comparativo no se pudo realizar con *TFNet*, porque los autores de ese modelo no proporcionaron el código de inferencia.

5.4. Calidad y análisis de predicciones

Para dar respuesta al objetivo O3 relacionado con la hipótesis H3 se proponen dos análisis cualitativos, uno basado en la certeza de las predicciones y otro en la cobertura de las mismas, tomando como *groundtruth* los TFs con evidencia experimental de los tres organismos de referencia.

I) Análisis de Cobertura: Predicciones versus evidencia experimental de factores de transcripción en organismos de referencia.

   DeepReg identificó más TFs con evidencia experimental que *DeepTFactor*.

   Para *S. cerevisiae* se identificaron 185 Y 163 para *DeepReg* y *DeepTFactor* respectivamente, de un total de 304 TFs (33.33 % no fueron detectados por ningún modelo). Para *N. crassa*, se identificaron 44 y 45 de un total de 62 TFs, bajo el mismo orden (22.5 % no fueron detectados). Finalmente 55 y 44 de 66 TFs fueron identificados en *A. nidulas* (13.8 % no fueron detectados).

   Las posibles razones de no haber identificado el resto de los TFs por ambos modelos recaen en que varios de esos TFs experimentales poseen estructuras no canónicas, tales como dominios de Bromo asociados con la histona acetiltransferasa y a la remodelación de ADN o bien al dominio de unión al ADN Mlu1-box en *A. nidulas*.

   Para ver a más detalle el análisis de cobertura, vea el apéndice A.5 parte de *DeepReg*.

II) Enfoque *Bias-Variance Tradeoff* para la medición de calidad de predicciones en una clasificación

Binaria.

En [Geman et al., 1992, Doroudi, 2020] se plantean la bases del análisis *Bias-Variance Tradeoff* como un análisis útil para medir la certeza de las predicciones de un modelo de clasificación - predicción.

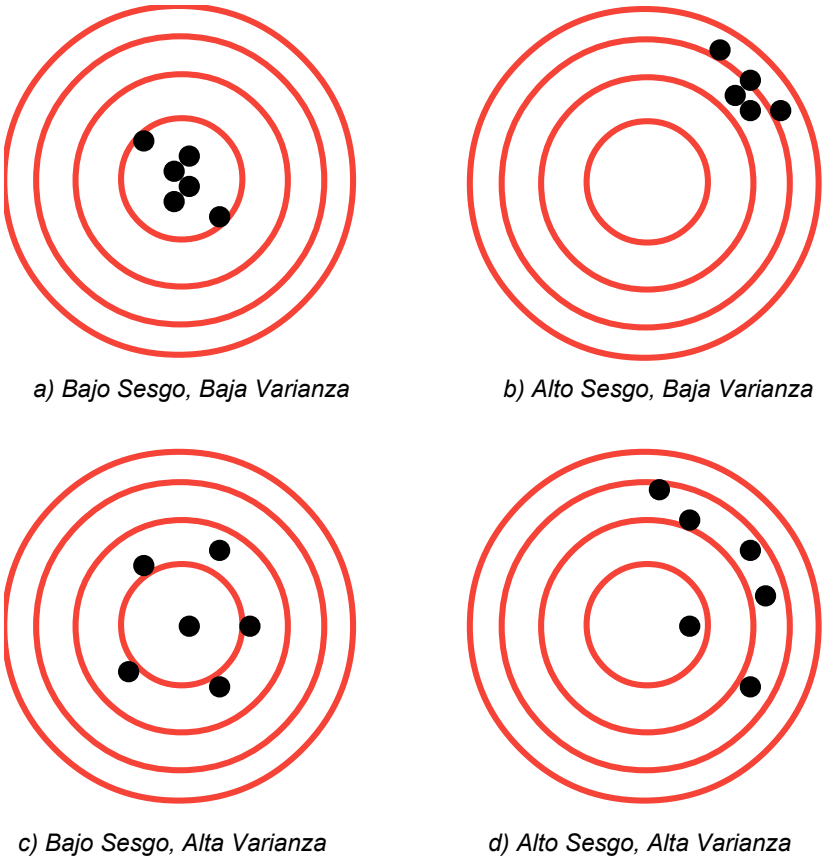


Figura 5.1: Escenarios de *Bias-Variance Tradeoff* sobre las predicciones de un modelo. Cada punto representa una predicción, la varianza se mide como la dispersión entre los puntos y el sesgo que tan lejos se encuentran las predicciones del centro.

En *Bias-Variance Tradeoff* presentan cuatro escenarios posibles (ver Figura 5.1):

a) En un escenario ideal, las predicciones caen cerca del centro y concentradas, el modelo es óptimo para resolver la tarea.

b) Si las predicciones están lejos del centro y concentradas, el modelo presenta *underfitting*, es decir el modelo tiene baja varianza, pero algo de sesgo.

c) Si las predicciones están cerca del centro y dispersas, el modelo presenta sobreajuste, es decir, el modelo tiene bajo sesgo, pero alta varianza.

d) Si las predicciones están lejos del centro y dispersas, el modelo es inútil para resolver la tarea, es decir, el modelo tiene alto sesgo y alta varianza.

Tomando el score de predicción se crea un diagrama del tipo *bull's-eye* de radio 0.5 donde el centro es el 1 y el límite es 0.51, cada punto es una predicción positiva hecha por el modelo y su coordenada polar (r, θ) está definida por:

$$p_i(r, \theta) \quad , \quad i \in [1, t] \tag{5.7}$$

$$t = total\ de\ predicciones\ positivas$$
$$r = 1 - (score\ de\ predicciones) : n, n \in [0.51, 1]$$
$$\theta_i = \frac{t}{360} * i$$

Para las predicciones de *DeepReg* realizadas en los organismos fúngicos de referencia, se pueden ver los diagramas *bull's-eye* en la Figura A.4 del apéndice, en la parte del modelo.

En general, las predicciones realizadas por *DeepReg* son más fiables dado que tienen menos varianza y bajo sesgo versus las predicciones que *DeepTFactor* y además las predicciones de *DeepReg* se concentran en el centro y no están dispersas.

Capítulo 6

Transformando el Lenguaje de la Vida

"El ADN es el lenguaje en el que Dios escribió la vida."
- Francis Collins.

6.1. Estudio de los modelos de lenguaje de la familia BERT para predicción de factores de transcripción.

Para dar cumpliento al O5 descrito en el capítulo 4, se exploraron e investigaron los modelos de lenguaje aplicados a la Bioinformática descritos en el capítulo 3. Se seleccionaron aquellas modelos de la familia BERT (ver Tabla 6.1), dado que se busca trabajar con el codificador de tipo BERT y aplicar mecanismos de multi-atención para generar un modelo predictivo.

6.2. *Pipeline* para usar BERT *Language Model Family* para tareas de clasificación y predicción.

Se usa el *pipeline* descrito en la Figura 6.1, donde se utiliza la parte del *enconder* y se desecha la parte del *decoder* de una red tipo *transformer*.

Cada secuencia de aminoácidos de entrada es tokenizada en el alfabeto $\Sigma = \{A, R, N, D, \ldots$ $\ldots, B, C, E, Q, Z, G, H, I, L, K, M, F, P, O, S, U, T, W, Y, V, X\}$ más los *tokens* especiales:

- $< CLS >$, marca una tarea de clasificación,

Modelo	**Año / Data**	**Descripción**	**Técnicas de predicción / de mejora**	**Tipos**
BERT BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [Devlin et al., 2018]	2018 3.3 Billones de palabras	Modelo de lenguaje pre-entrenado con un CORPUS amplio en lengua inglesa de manera no supervisada. Posteriormente se aplica un fine-tunning para realizar tareas específicas.	NSP (Next sentence prediction) MLM (Masking Language Model)	Bert Base (110 m.p) - 12 T.L. Bert Large (340 m.p)- 24 T.L.
RoBERTa RoBERTa: A Robustly Optimized BERT Pretraining Approach. [Liu et al., 2019]	2019 33 Billones de palabras	Entrenado con un CORPUS, 10 veces más grande que BERT.	Dynamic MLM BPE (byte-pair encoding)	RoBERTa Base (110 m.p) - 12 T.L. RoBERTa Large (340 m.p)- 24 T.L.
AlBERT ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. [Lan et al., 2019]	2019	Una versión ligera de BERT que reduce el consumo de memoria	*Factorized embedding parameterization y Cross-layer parameter sharing*	AlBERT Base (12 m.p) - 12 T.L. AlBERT Large (18 m.p)- 24 T.L. AlBERT XLarge (60 m.p)- 24 T.L. AlBERT XXLarge (235 m.p)- 12 T.L.
Squeeze-BERT SqueezeBERT: What can computer vision teach NLP about efficient neural networks? [Sanh et al., 2019]	2020	Una versión rápida de 4.3 veces más rápida que BERT. Introducen el uso de CNN en tareas de NLP basadas en BERT	*Grouped convolutions*	Base (51 m.p)
DistilBERT DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. [Iandola et al., 2020]	2019 3.3 Billones de palabras	Es una versión optimizada de BERT con 40 % menos de parámetros y 60 % más rápido, preservando el 95 % del desempeño de BERT en tareas de clasificación	*Distillation*	Base (60 m.p) - 6 T.L.
DeBERTa DeBERTa: Decoding-enhanced BERT with Disentangled Attention [He et al., 2020b]	2020	Usa dos técnicas de representación de palabras considerando contenido, posición y pesos de atención.	*Disentangled attention mechanism*	DeBERTa Base (86 m.p) - 12 T.L. DeBERTa Large (304 m.p)- 24 T.L.

Tabla 6.1: Modelos de la familia BERT

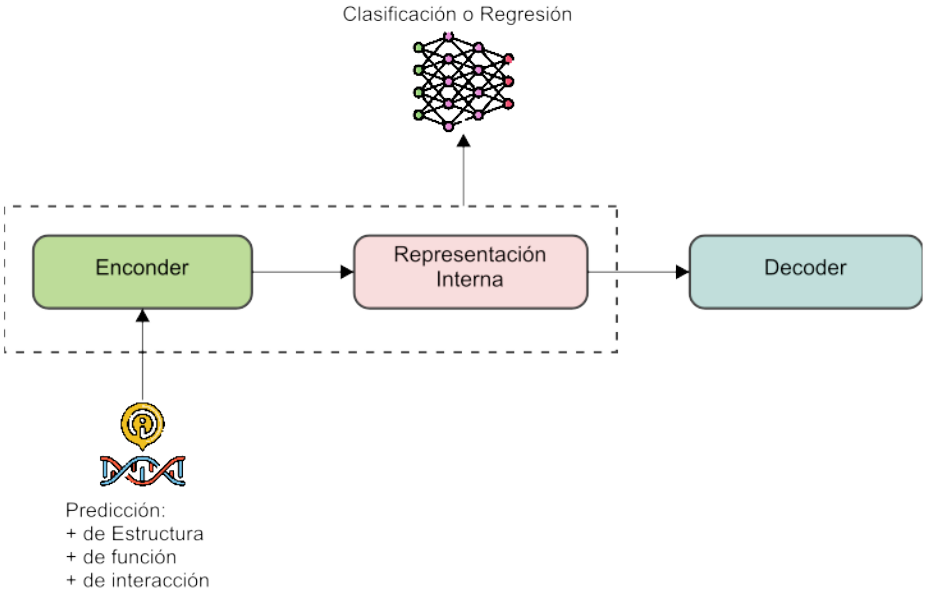


Figura 6.1: *Pipeline* para usar BERT *Language Model Family* para tareas de clasificación y predicción en la Bioinformática. Se utiliza un *encoder* tipo *transformer* para generar una representación interna que después es utilizado para conectarlo a una red *fully connected*.

- $< SEP >$, marca el carácter espacio,
- $< MASK >$, token especial para ocultar *tokens*,
- $< pad >$, token para señalar una operación *padding*.

Se construyen las clases para clasificación binaria y clasificación multiclase, las cuales se consideran:

0. Eucariote no TF
1. Bacteria no TF
2. Archaea no TF
3. Virus no TF
4. Otros no TF
5. Eucariote TF
6. Bacteria TF
7. Archaea TF
8. Virus TF
9. Otros T

Posteriormente se construye el *tokenizer* de un modelo pre-entrenado de la familia BERT y se expande con el alfabeto descrito anteriormente y de la misma forma se guarda el modelo en memoria. Dado que el tokenizer se extendió también es necesario expandir el tamaño de los *embeddings*.

Metrics	BERT	RoBERTa	AlBERT	Squeeze-BERT	DistilBERT
Acuracy	0.9668	0.9678	0.9691	0.9719	**0.9783**
Precision	0.9323	0.9268	0.9446	0.9283	**0.9473**
Recall	0.9386	0.917	0.8986	**0.9341**	0.9251
Specificity	0.9764	0.981	0.9867	0.9809	**0.989**

Tabla 6.2: Métricas de desempeño en clasificación de TFs usando modelos de lenguaje de la familia BERT para clasificación.

Se cargan el modelo, indicando el número de clases para realizar la clasificación multiclase o binaria, también se dividen los conjuntos de datos en validación y entrenamiento en tensores.

Finalmente, una vez que se hayan definido los hiperparámetros y las técnicas de regularización que se deseen utilizar, se obtiene la salida del modelo entrenado para calcular las métricas del desempeño del modelo.

6.3. Métricas de desempeño

En la Tabla 6.2 se muestra cada uno de los modelos de la BERT utilizados, señalando aquellos que tuvieron mejor desempeño.

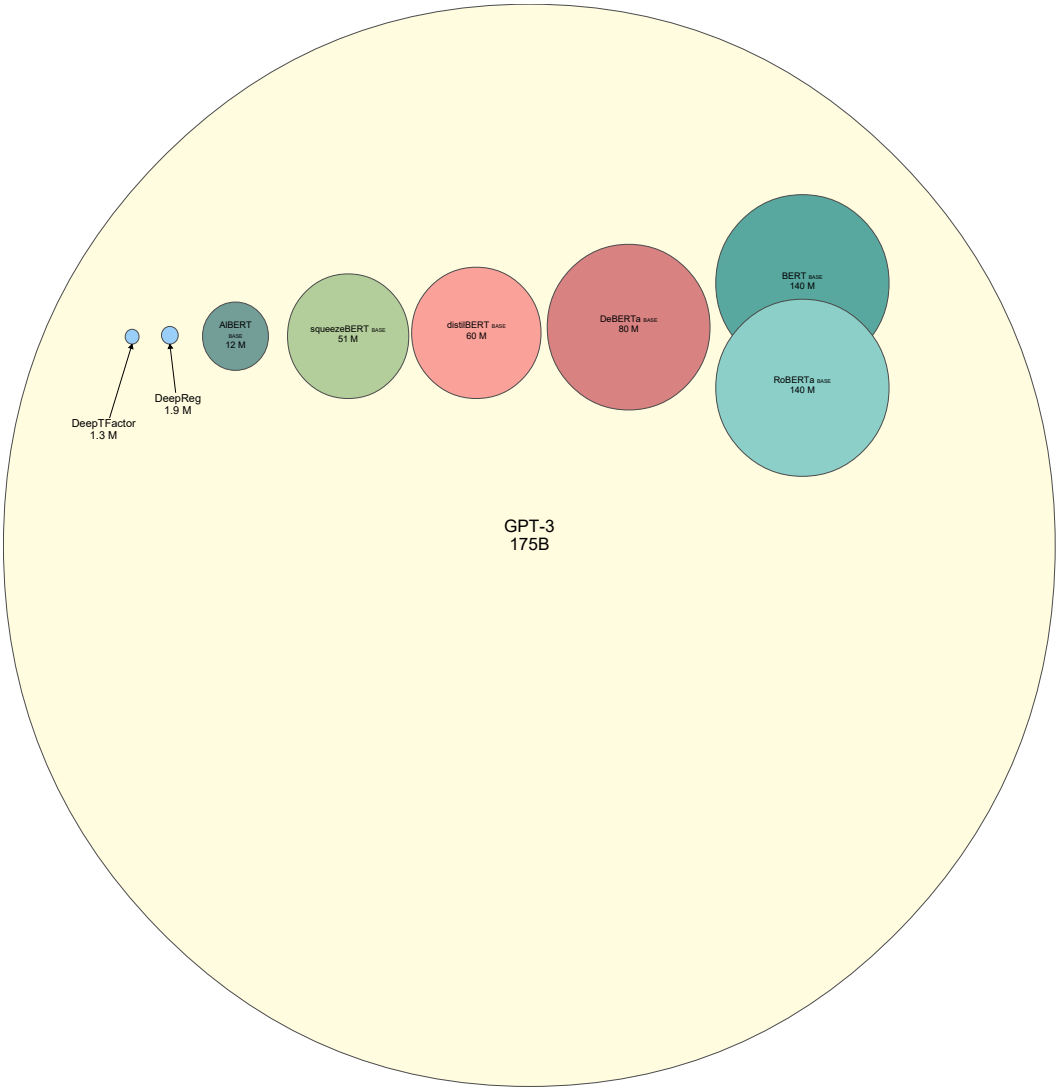


Figura 6.2: Representación de los tamaños modelos de DL. En azul se marcan los modelos *DeepReg* y *DeepTFactor* con alrededor de entre 1-2 millones de parámetros, mientras que los modelos de lenguaje pequeños basados en BERT tienen alrededor de 50 a 140 millones de parámetros. Se comparan contra un *Large Language Model (LLM)* GPT-3 que contiene 175 billones de parámetros. Para mayo de 2024 el modelo más grande es de 2 Teras de parámetros, llamado Olympus.

Capítulo 7

Conclusiones

"La mente humana tiene un primitivo mecanismo de defensa que niega cualquier realidad que provoque un estrés excesivo al cerebro. Se llama negación."

- Dan Brown.

Tomando como referencia las hipótesis y los objetivos planteados en el capítulo 4 y considerando el trabajo desarollado a lo largo de este trabajo se puede llegar a las siguientes conclusiones:

1. La hipótesis H1 y H2 establecen que:

   - H1: Hipotéticamente hablando hay una posibilidad de que exista un sesgo por homología de secuencias en las tareas de clasificación de los TFs o de cualquier otra clasificación funcional y/o estructural de proteínas.
   - H2: Si se comparan dos algoritmos cuya metodología es totalmente diferente (eg. uno basado en ML y otro basado en DL) y la distribución de las predicciones de TFs sigue un mismo patrón se establece un consenso algorítmico de que no existe sesgo por homología de secuencias o es tan pequeño que se puede considerar despreciable.

   Por lo que mediante un consenso algorítmico planteado en el objetivo O1 donde dos metodologías de naturaleza distinta, en este caso un algoritmo tradicional de ML como *PFAM* y una red neuronal *DeepTFactor*, llegan a resultados similares, donde la distribución de predicciones se concentran en ciertos tamaños de proteomas, es decir no existe una correlación entre el tamaño del proteoma y el número de TFs , lo que demuestra que el sesgo por homología de secuencias se puede considerar despreciable.

2. Como segunda conclusión, se comprobó mediante el objetivo O2, la hipótesis H4, la cual establece que mediante la técnicas de regularización se puedo construir un modelo basado en DL más robusto, eliminando la presencia de overfitting.

   A pesar del gran desbalance en la cantidad de datos de las clases de TFs y No TFs, se pudo consolidar un modelo estable. Esto queda demostrado a través del analizar la función de pérdida en validación, eliminando los saltos abruptos y aplanando la curva asintóticamente.

3. Respecto al objetivo O3, que plantea la realización de análisis de calidad de predicción y cobertura para medir la certidumbre del modelo. Por lo que *DeepReg* no solo elimina el problema del *overfitting*, sino que además sus predicciones tienden a ser más certeras que *DeepTFactor*.

   Utilizando un análisis cuantitativo de cobertura y otro análisis cualitativo que mide varianza y sesgo, el modelo de *DeepReg* muestra poco sesgo y poca varianza y encuentra más TFs con evidencia experimental con respecto a sus predicciones.

   De manera adicional, se plantearon dos objetivos más de los cuales se puede concluir que:

4. Para el objetivo O4 el modelo se aplicó a factores de virulencia dando excelentes resultados, solamente el modelo se reentrenó sin cambiar su arquitectura resolviendo un problema de clasificación en otro dominio (ver Apéndice A). Por lo que se puede afirmar que *DeepReg* es flexible para otras tareas de clasificación - predicción.

5. Finalmente para el objetivo O5, se exploró la posibilidad de crear un modelo de lenguaje predictivo basado en las nuevas arquitecturas *Transformer*. Los resultados son esperanzadores simplemente siguiendo una metodología práctica y sencilla en las familias predictivas *BERT*.

   Cabe mencionar que si desea seguir trabajando en este tema, se requiere de un poder de cómputo especial dado que los parámetros tienden a crecer de manera exponencial.

   En este punto surgen nuevas preguntas:

   - ¿Hasta qué punto se puede decir que una tarea de clasificación - predicción ya está resuelta?
   - ¿Se podrá generar un modelo híbrido ML-DL, donde las predicciones basadas en modelos de DL que ya no se pueden dar, se podría crear una herramienta auxiliar basada en ML para completar las predicciones, o viceversa?

   Otras tareas pendientes serían:

   - Las predicciones sin evidencia experimental se tendrían que analizar biológicamente y comprobar si realmente son candidatos fuertes a ser TFs.

- Explorar otras técnicas de reciente estudio para mejorar la tasa de cobertura de modelos basados en DL como son: el meta aprendizaje y las técnicas de aprendizaje no supervisado.

Apéndice A

Apéndice

El dogma central de la biología molecular.

El dogma central de la biología molecular describe el flujo de información genética dentro de las células y cómo esta información se convierte en proteínas, las cuales son fundamentales para la estructura y función celular. Este dogma establece tres pasos principales: la replicación del ADN, la transcripción y la traducción.

La replicación del ADN es esencial para mantener la integridad genética en los organismos. El proceso de replicación ocurre cuando las hebras de ADN existentes sirven como plantilla para la síntesis de nuevas hebras de ADN hijas, siguiendo modelos conservativos y semi conservativos.

La ADN polimerasa participa en la iniciación de la replicación así también las helicasas desenrollan las hebras de ADN para exponer las bases y permitir su apareamiento con desoxinucleótidos.

Como parte del dogma central están los procesos de transcripción y traducción en la síntesis de proteínas, los genes codificadores contienen la información necesaria para la construcción de proteínas y esta información se transcribe en ARNm durante el proceso de transcripción. Una cadena de ADN actúa como molde para la síntesis de una cadena complementaria de ARN mediante la polimerización de monómeros de rNTP.

En segundo lugar, la transcripción es el proceso mediante el cual se copia la información genética del ADN a una molécula de ARN mensajero (ARNm). Durante la transcripción, la enzima ARN polimerasa recorre una hebra de ADN y sintetiza una cadena complementaria de ARNm utilizando ribonucleótidos. La transcripción es crucial porque el ARNm lleva la información genética desde el núcleo de la célula hasta los ribosomas en el citoplasma, donde tiene lugar la traducción.

Finalmente, la traducción es el proceso mediante el cual la secuencia de nucleótidos en el ARNm se traduce en una secuencia de aminoácidos en una proteína. Este proceso ocurre en los ribosomas, que actúan como ”fábricas de proteínas”. Los aminoácidos son transportados al ribosoma por moléculas de ARN de transferencia (ARNt), que contienen anticodones complementarios a los codones del ARNm. Los ribosomas, junto con otras moléculas de ARN ribosómico (ARNr) y proteínas, coordinan la unión de aminoácidos para formar una cadena polipeptídica que luego se pliega para formar una proteína funcional.

Métricas de clases desbalanceadas.

Para determinar las métricas para modelos de clases desbalanceadas es necesario definir las condiciones de predicción para clasificación binaria:

- Predicciones correctas:

  A Verdaderos Positivos (*True positives*, TP): El modelo predice de manera correcta un caso positivo y el caso estaba etiquetado como un caso positivo o bien el caso efectivamente es positivo. eg. *El modelo predice que la secuencia A es un TF, cuando efectivamente es un TF.*

  B Verdaderos Negativos (*True negatives*, TN): El modelo predice de manera correcta un caso negativo y el caso estaba etiquetado como un caso negativo o bien, el caso efectivamente es negativo. eg. *El modelo predice que la secuencia A no es un TF, cuando efectivamente no es un TF.*

- Predicciones incorrectas:

  C Falsos Negativos (*False negatives*, FN): El modelo predice de manera incorrecta, como un caso positivo y el caso estaba etiquetado como un caso negativo o bien el caso realmente es negativo. eg. *El modelo predice que la secuencia A es un TF, cuando realmente no lo es.*

  D Falsos Positivos (*False positives*, FP): El modelo predice de manera incorrecta, como un caso negativo y el caso estaba etiquetado como un caso positivo o bien el caso realmente es positivo. eg. *El modelo predice que la secuencia A no es un TF, cuando realmente sí lo es.*

Las métricas más usadas.

a) Precisión (*Positive predictive value*, PPV): esta métrica evalúa el rendimiento de un modelo de clasificación con el cociente de predicciones de verdaderos positivos sobre el total de las mismas más los falsos positivos. Esta métrica es más significativa cuando existe una gran disparidad entre el número de etiquetas positivas y negativas. Con ella se mide qué tan bueno es el modelo en predecir casos realmente positivos tomando como referencia lo que el modelo determina que es positivo.

   Clasificación Binaria:

$$precision = \frac{TP}{TP + FP}$$

   Clasificación multiclase:

$$precision = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{TP}{TP + FP} \right]_i$$

b) Exhaustividad (*recall, sensitivity, true positive rate*, TPR): mide el desempeño del modelo con el cociente entre las predicciones de verdaderos positivos sobre el total de las mismas más los falsos negativos. Con ella se mide qué tan bueno es el modelo en predecir casos positivos tomando como referencia la casos reales positivos.

   Clasificación Binaria:

$$recall = \frac{TP}{TP + FN}$$

   Clasificación multiclase:

$$recall = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{TP}{TP + FN} \right]_i$$

c) Especificidad o selectividad (*Specificity*): mide el desempeño del modelo con el cociente entre las predicciones de verdaderos negativos sobre el total de las mismas más los falsos positivos. Con

ella se mide qué tan bueno es el modelo en predecir casos negativos tomando como referencia la casos reales negativos.

Clasificación Binaria:

$$specificity = \frac{TN}{TN + FP}$$

Clasificación multiclase:

$$specificity = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{TN}{TN + FP} \right]_i$$

d) Valor F1 (F1 score): Es una métrica de uso común en clases desbalanceadas y combina los conceptos de recall y specificity, para evaluar el rendimiento de un modelo de clasificación. Matemáticamente hablando, el valor F1 es un promedio armónico entre las métricas de precisión y exhaustividad, y se calcula como:

$$f1 - score = 2 \times \frac{precision \times recall}{recall + precision}$$

e) Coeficiente de correlación Matthews (*phi coefficient*, MCC): Propuesto por Karl Pearson considerando todos los valores de una matriz de confusión y se calcula como

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) + (TP + FN) + (TN + FP) + (TN + FN)}}$$

Extensión aplicativa de DeepReg: Predicción de factores de virulencia

Los factores de virulencia son proteínas producidas por organismos patógenos que contribuyen a su capacidad para invadir, colonizar y causar daño al huésped. La virulencia se estudia en patogénesis y epidemiología para diagnosticar, tratar y prevenir enfermedades.

Existen diferentes factores de virulencia como adhesinas, invasinas, toxinas, evasivas y de supervivencia. Por ejemplo, dos factores de virulencia bastante conocidos son la proteína E del virus SARS-

Metrics	64 Batch	32 Batch
Acuracy	0.91959	0.9039
Precision	0.9181	0.9021
AUC	0.9622	0.9484
Recall	0.9228	0.9079

Tabla A.1: Resultados en predicción de factores de virulencia usando DeepReg.

CoV-2 que produce la enfermedad respiratoria COVID-19 [Zhou et al., 2023, Santos-Mendoza, 2023]. La proteína E forma un canal iónico indispensable para que el virus haga daño al huésped, es por esa razón que la proteína E se ha vuelto objeto de estudio para fines terapéuticos.

Otro factor de virulencia es la ureasa de la bacteria Helicobacter pylori cuya actividad está relacionada al daño epitelial gástrico mediante la liberación de amonio [Ansari and Yamaoka, 2019, Valenzuela-Valderrama et al., 2019]. También la Ureasa está relacionada con el cáncer de estómago y la gastritis.

Para dar cumplimiento al O4, se busca extender el mismo modelo DeepReg para otra naturaleza de datos, en este caso, para los factores de virulencia. De esta manera se puede comprobar sobre la extensibilidad y flexibilidad del modelo para otro tipo de datos.

Para la tarea de clasificación y predicción de factores de virulencia se empleó un set de datos de 28,834 factores de virulencia obtenidos de la base de datos VFDB, descritos en el trabajo [Liu et al., 2021] y otro set del mismo tamaño para el set de datos negativo. Se utilizó una proporción de $80:15:5$ para crear los conjuntos de entrenamiento, validación y prueba y se obtuvieron los siguientes resultados:

Capitulo de libro

[Ledesma et al., 2022]: Ledesma, L., Hernandez-Guerrero, R., & Perez-Rueda, E. (2022). Prediction of DNA-Binding Transcription Factors in Bacteria and Archaea Genomes. *Methods in molecular biology* (pp. 103-112). https://doi.org/10.1007/978-1-0716-2413-5_7

Artículo

[Ledesma-Dominguez et al., 2024]: Ledesma-Dominguez, L., Carbajal-Degante, E., Moreno-Hagelsieb, G., & Perez-Rueda, E. (2024). DeepReg: a deep learning hybrid model for predicting transcription factors in eukaryotic and prokaryotic genomes. *Scientific Reports, 14*(1). https://doi.org/10.1038/s41598-024-59487-5

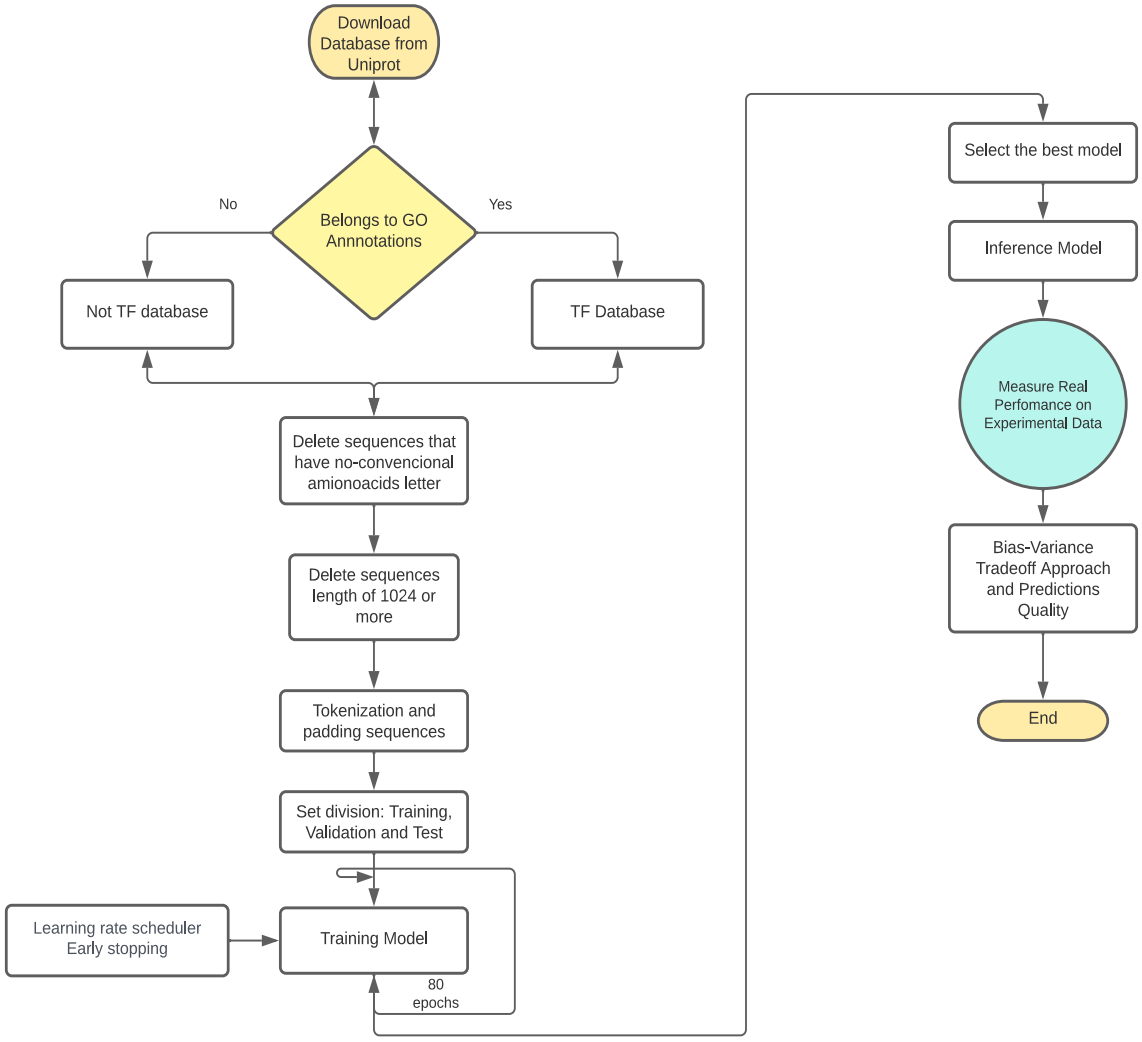


Figura A.1: *Workflow* de la metodología empleada en este trabajo, esta imagen es extraida del trabajo publicado en [Ledesma-Dominguez et al., 2024], ubicado en la Figura 1. La primera parte del diagrama ejemplifica la construcción de la arquitectura de *DeepReg*, así como el uso de las técnicas de reguralización. La segunda parte muestra la selección del mejor modelo para su analisis cuantitativo y cualitativo sobre las predicciones en los organismos de referencia.

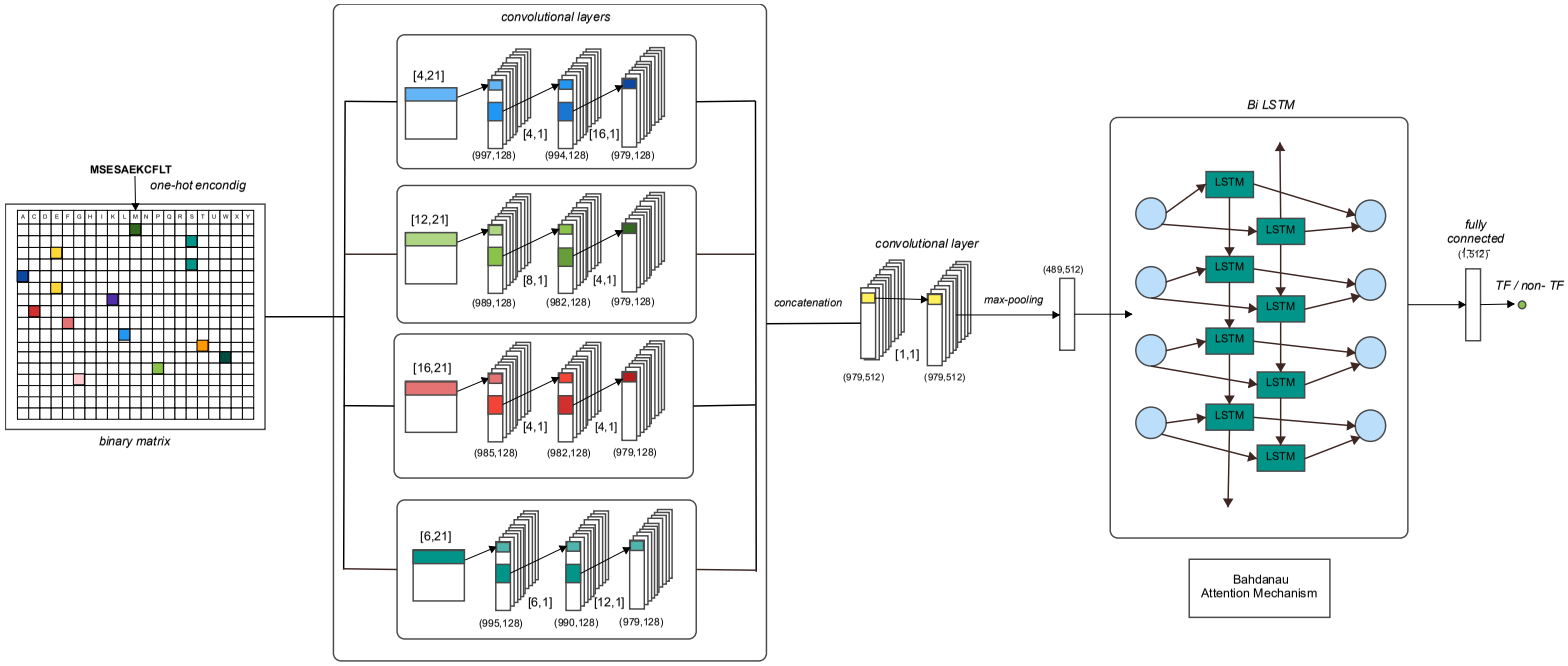
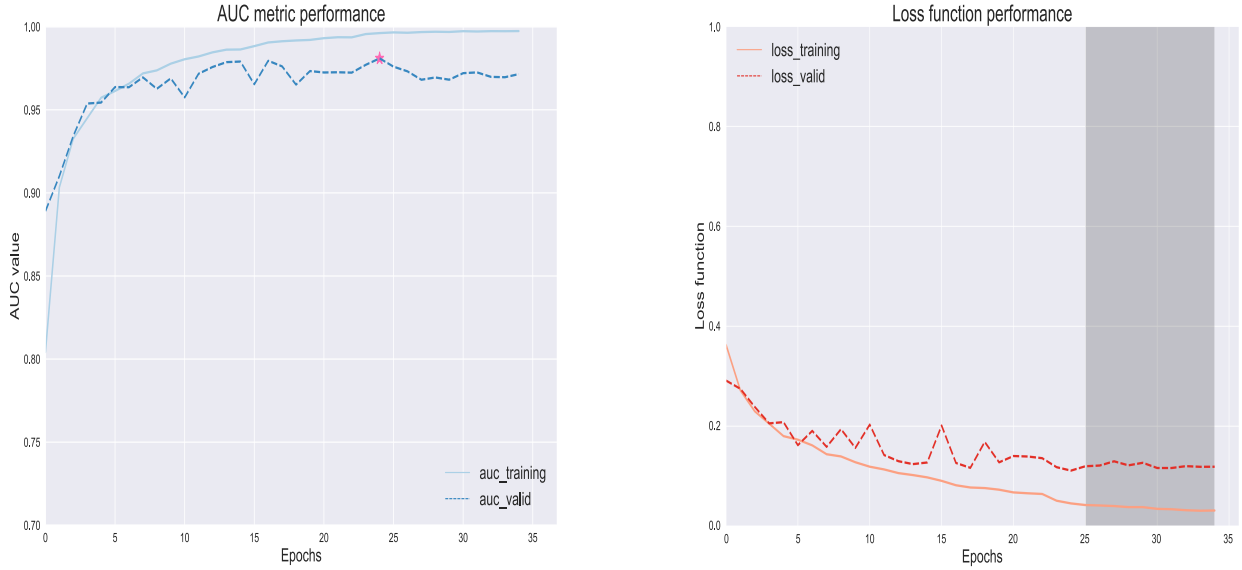


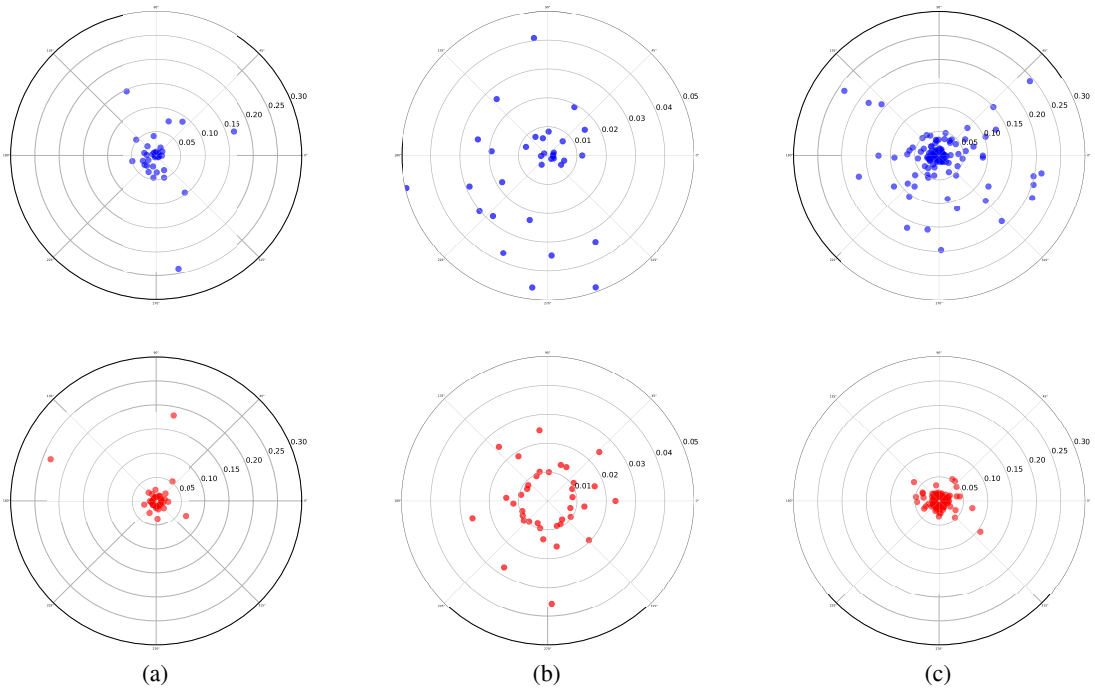
Figura A.2: La arquitectura del modelo *DeepReg*. Se consideraron tres módulos: (1) 4 capas CNN como extractor de características, (2) una red BiLSTM para predicir patrones secuenciales basados en orden y frecuencia, y (3) el módulo del mecanismo de atención. Cada secuencia está tokenizada y pasa a través de una codificación *one-hot enconding*.



(a) (b)

Figura A.3: (a) Métrica AUC y (b) función de pérdida del rendimiento de la arquitectura. En (a), el eje X muestra el número de épocas frente al valor del AUC, durante la validación y entranamiento del modelo, y el valor de AUC máximo se representa con una estrella. En (b), el eje X muestra el número de épocas frente a la función de pérdida, y el área sombreada muestra la estabilidad de la función de pérdida, evitando saltos abruptos.

S. cerevisiae	*N.crassa*	*A. nidulans*
Variance DeepTFactor: 0.0047	*Variance DeepTFactor: 0.0245*	*Variance DeepTFactor: 0.0056*
Variance DeepReg: 0.0039	*Variance DeepReg: 0.0214*	*Variance DeepReg: 0.0012*



(a) (b) (c)

Figura A.4: *Bull-eye bias variance tradeoff*: Analisis cualitativo de las predicciones clasificadas como verdaderos positivos para medir la confiabilidad y certidumbre de lo modelos. (a) *S. cerevisiae* (b) *N. crassa*, and (c) *A. nidulans* sobre valores experimentales del *ground truth*. El rojo es *DeepReg*, y el azul es *DeepTFactor*. Solo se consideraron los verdaderos positivos para esta evaluación, donde el radio con respecto al centro es el puntaje obtenido y el ángulo se distribuye para llenar toda la circunferencia. Para mostrar una distribución clara de los puntos, cortamos los diagramas en cierto radio.

Metrics	DeepReg	DeepTFactor
Accuracy	0.9742	0.9773
Precision	0.9923	0.9656
Recall	0.9770	0.9428
Specificity	0.9591	0.9888
F1-Score	0.9846	0.9541
MCC	0.9066	0.9392

Tabla A.2: Desempeño de *DeepReg* vs *DeepTFactor* en diversas métricas para la clasifiación de TFs.

F1-score	No PCA	PCA 2-elements	PCA 3-elements	PCA 4-elements
Gaussian NB	0.5656	0.5948	0.5281	0.5178
Logistic regression	0.8054	0.6096	0.6313	0.6179
SVM	0.8879	0.6611	0.6509	0.6688
Random Forest	0.7534	0.6585	0.6732	0.6812
Decision Tree	0.7994	0.6934	0.7298	0.7621

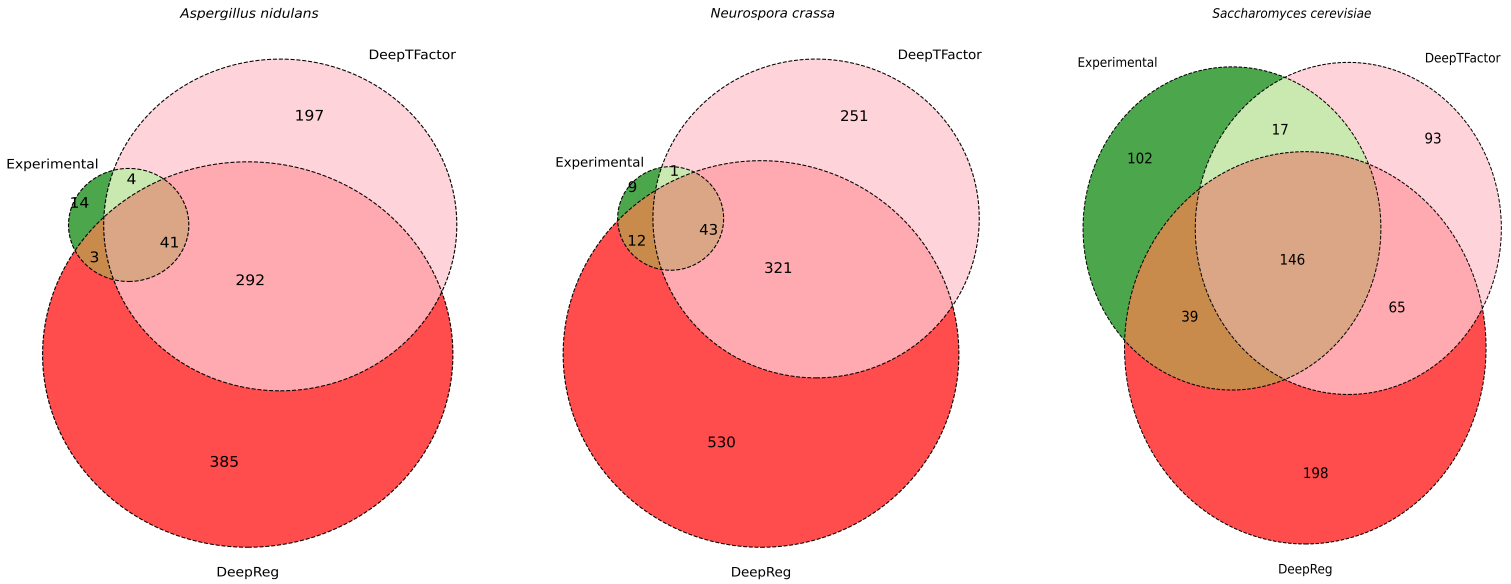
Tabla A.3: Métrica F1-Score: Rendimiento de los métodos tradicionales de ML para la predicción/clasificación de TFs.

AUC	No PCA	PCA 2-elements	PCA 3-elements	PCA 4-elements
Gaussian NB	0.6594	0.5641	0.5978	0.6146
Logistic regression	0.8016	0.6136	0.5477	0.6392
SVM	0.8862	0.6419	0.6797	0.6956
Random Forest	0.7677	0.6563	0.6856	0.6992
Decision Tree	0.7904	0.6848	0.7241	0.7534

Tabla A.4: Métrica valor AUC: Rendimiento de los métodos tradicionales de ML para la predicción/clasificación de TFs.

Method	Accuracy	AUC	F1-Score	Recall	Precision	MCC
(1) Baseline	0.895	0.859	0.932	0.900	0.968	0.704
(2) Baseline + 1CNN	0.919	0.905	0.920	0.9258	0.9144	0.8384
(3) DeepReg - ATT	0.963	0.946	0.976	0.968	0.984	0.902
(4) DeepReg	**0.974**	**0.956**	**0.985**	**0.977**	**0.992**	**0.906**

Tabla A.5: *Ablation analysis* de *DeepReg*. Considerando (1) *Baseline*, (2) Agregando una red CNN, (3) Agregando la red BiLSTM sin el mecanismo de atención, (4) Toda la arquitectura.



(a) (b) (c)

Figura A.5: Análisis de cobertura de predicciones en los organismos de referencia. Predicciones de cada modelo y TFs con evidencia experimental.

Bibliografía

[Bla, 2018] (2018). Rps BLAST search database. https://proteininformationresource.org/rps/blast. [Online; accessed 2024-04-22].

[Abdar et al., 2021] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297. [Online; accessed 2024-05-24].

[Alipanahi et al., 2015] Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology*, 33(8):831–838. [Online; accessed 2024-04-23].

[Altschul, 1997] Altschul, S. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402. [Online; accessed 2024-04-22].

[Ansari and Yamaoka, 2019] Ansari, S. and Yamaoka, Y. (2019). Helicobacter pylori virulence factors exploiting gastric colonization and its pathogenicity. *Toxins*, 11(11):677.

[Apweiler, 2004] Apweiler, R. (2004). Uniprot: The Universal Protein knowledgebase. *Nucleic Acids Research*, 32(90001):115D–119. [Online; accessed 2024-04-22].

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. https://arxiv.org/abs/1409.0473.

[Benson et al., 2007] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Wheeler, D. L. (2007). Genbank. *Nucleic Acids Research*, 36:D25–D30. [Online; accessed 2024-04-22].

[Berman, 2000] Berman, H. M. (2000). The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242. [Online; accessed 2024-04-22].

[Boj et al., 2010] Boj, S. F., Petrov, D., and Ferrer, J. (2010). Epistasis of transcriptomes reveals synergism between transcriptional activators hnf1 and hnf4. *PLoS Genetics*, 6(5):e1000970. [Online; accessed 2024-05-16].

[Brauwers and Frasincar, 2023] Brauwers, G. and Frasincar, F. (2023). A general survey on attention mechanisms in deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3279–3298. [Online; accessed 2024-05-18].

[Bzdok et al., 2024] Bzdok, D., Thieme, A., Levkovskyy, O., Wren, P., Ray, T., and Reddy, S. (2024). Data science opportunities of large language models for neuroscience and biomedicine. *Neuron*, 112(5):698–717. [Online; accessed 2024-04-22].

[Chen et al., 2021] Chen, C., Hou, J., Shi, X., Yang, H., Birchler, J. A., and Cheng, J. (2021). Deepgrn: Prediction of transcription factor binding site across cell-types using attention-based deep neural networks. *BMC Bioinformatics*, 22(1). [Online; accessed 2024-04-25].

[Chung et al., 2023] Chung, C., Verheijen, B. M., Zhang, X., Huang, B., Coakley, A., McGann, E., Wade, E., Dinep-Schneider, O., LaGosh, J., Anagnostou, M.-E., Simpson, S., Thomas, K., Ernst, M., Rattray, A., Lynch, M., Kashlev, M., Benayoun, B. A., Li, Z., Strathern, J., Gout, J.-F., and Vermulst, M. (2023). The fidelity of transcription in human cells. *Proceedings of the National Academy of Sciences*, 120(5). [Online; accessed 2024-05-16].

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805.

[Doroudi, 2020] Doroudi, S. (2020). The Bias-Variance Tradeoff: How Data Science Can Inform Educational Debates. *AERA Open*, 6(4):233285842097720. [Online; accessed 2024-06-03].

[Du et al., 2023] Du, Z., Huang, T., Uversky, V. N., and Li, J. (2023). Predicting TF proteins by incorporating evolution information through PSSM. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20(2):1319–1326. [Online; accessed 2024-04-22].

[Eichner et al., 2013] Eichner, J., Topf, F., Dräger, A., Wrzodek, C., Wanke, D., and Zell, A. (2013). Tfpredict and SABINE: Sequence-Based prediction of structural and functional characteristics of transcription factors. *PLoS ONE*, 8(12):e82238. [Online; accessed 2024-04-22].

[Fukushima, 1980] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202. [Online; accessed 2024-05-16].

[Gal and Ghahramani, 2015] Gal, Y. and Ghahramani, Z. (2015). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. https://arxiv.org/abs/1506.02142.

[Geman et al., 1992] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58. [Online; accessed 2024-06-03].

[Gonzalez, 2016] Gonzalez, D. H. (2016). *Introduction to transcription factor structure and function*, pages 3–11. Elsevier. [Online; accessed 2024-05-16].

[Graves et al., 2014] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. https://arxiv.org/abs/1410.5401.

[Gupta and Rush, 2017] Gupta, A. and Rush, A. M. (2017). Dilated convolutions for modeling long-distance genomic dependencies. Technical report, Cold Spring Harbor Laboratory. [Online; accessed 2024-04-25].

[He et al., 2020a] He, J., Pu, X., Li, M., Li, C., and Guo, Y. (2020a). Deep convolutional neural networks for predicting leukemia-related transcription factor binding sites from DNA sequence data. *Chemometrics and Intelligent Laboratory Systems*, 199:103976. [Online; accessed 2024-04-25].

[He et al., 2020b] He, P., Liu, X., Gao, J., and Chen, W. (2020b). Deberta: Decoding-enhanced bert with disentangled attention.

[He et al., 2020c] He, Y., Shen, Z., Zhang, Q., Wang, S., and Huang, D.-S. (2020c). A survey on deep learning in DNA/RNA motif mining. *Briefings in Bioinformatics*, 22(4). [Online; accessed 2024-04-22].

[Hu et al., 2018] Hu, Y., Qin, Y., and Liu, G. (2018). Collection and Curation of Transcriptional Regulatory Interactions in Aspergillus nidulans and Neurospora crassa Reveal Structural and Evolutionary Features of the Regulatory Networks. *Frontiers in Microbiology*, 9. [Online; accessed 2024-05-30].

[Iandola et al., 2020] Iandola, F. N., Shaw, A. E., Krishna, R., and Keutzer, K. W. (2020). Squeezebert: What can computer vision teach NLP about efficient neural networks? https://arxiv.org/abs/2006.11316.

[Jin et al., 2016] Jin, J., Tian, F., Yang, D.-C., Meng, Y.-Q., Kong, L., Luo, J., and Gao, G. (2016). Planttfdb 4.0: Toward a central hub for transcription factors and regulatory interactions in plants. *Nucleic Acids Research*, 45(D1):D1040–D1045. [Online; accessed 2024-05-02].

[Jumper et al., 2021] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A.,

Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589. [Online; accessed 2024-04-22].

[Kim et al., 2020] Kim, G. B., Gao, Y., Palsson, B. O., and Lee, S. Y. (2020). Deeptfactor: A deep learning-based tool for the prediction of transcription factors. *Proceedings of the National Academy of Sciences*, 118(2). [Online; accessed 2024-04-27].

[Lambert et al., 2018] Lambert, S. A., Jolma, A., Campitelli, L. F., Das, P. K., Yin, Y., Albu, M., Chen, X., Taipale, J., Hughes, T. R., and Weirauch, M. T. (2018). The human transcription factors. *Cell*, 172(4):650–665. [Online; accessed 2024-05-16].

[Lan et al., 2019] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A Lite BERT for Self-supervised Learning of Language Representations. https://arxiv.org/abs/1909.11942.

[Latchman, 2004] Latchman, D. S. (2004). *Families of DNA binding transcription factors*, pages 77–133. Elsevier. [Online; accessed 2024-05-16].

[Ledesma et al., 2022] Ledesma, L., Hernandez-Guerrero, R., and Perez-Rueda, E. (2022). *Prediction of dna-binding transcription factors in bacteria and archaea genomes*, pages 103–112. Springer US, New York, NY. [Online; accessed 2024-05-31].

[Ledesma-Dominguez, 2015] Ledesma-Dominguez, L. (2015). Diseño e implementación de un sistema de consulta sobre modificaciones postraduccionales en las proteínas de la levadura saccharomyces cerevisiae. https://ru.dgb.unam.mx/handle/20.500.14330/TES01000730832.

[Ledesma-Dominguez et al., 2024] Ledesma-Dominguez, L., Carbajal-Degante, E., Moreno-Hagelsieb, G., and Perez-Rueda, E. (2024). Deepreg: A deep learning hybrid model for predicting transcription factors in eukaryotic and prokaryotic genomes. *Scientific Reports*, 14(1). [Online; accessed 2024-04-27].

[Letunic et al., 2009] Letunic, I., Doerks, T., and Bork, P. (2009). Smart 6: Recent updates and new developments. *Nucleic Acids Research*, 37:D229–D232. [Online; accessed 2024-04-22].

[Li and Godzik, 2006] Li, W. and Godzik, A. (2006). Cd-hit: A fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659. [Online; accessed 2024-04-22].

[Li et al., 2019] Li, Y., Huang, C., Ding, L., Li, Z., Pan, Y., and Gao, X. (2019). Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. *Methods*, 166:4–21. [Online; accessed 2024-04-23].

[Liu et al., 2021] Liu, B., Zheng, D., Zhou, S., Chen, L., and Yang, J. (2021). Vfdb 2022: a general classification scheme for bacterial virulence factors. *Nucleic Acids Research*, 50(D1):D912–D917.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. https://arxiv.org/abs/1907.11692.

[Lodish, 2008] Lodish, H., editor (2008). *Molecular cell biology*. Freeman, New York, NY, 6. ed. edition. Glossary S. G-1 - G-24; Index I-1 - I-52.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. https://arxiv.org/abs/1508.04025.

[Martinez-Liu et al., 2021] Martinez-Liu, L., Hernandez-Guerrero, R., Rivera-Gomez, N., Martinez-Nuñez, M. A., Escobar-Turriza, P., Peeters, E., and Perez-Rueda, E. (2021). Comparative genomics of DNA-binding transcription factors in archaeal and bacterial organisms. *PLOS ONE*, 16(7):e0254025. [Online; accessed 2024-05-31].

[Matys, 2003] Matys, V. (2003). Transfac(R): Transcriptional regulation, from patterns to profiles. *Nucleic Acids Research*, 31(1):374–378. [Online; accessed 2024-05-16].

[McGinnis and Madden, 2004] McGinnis, S. and Madden, T. L. (2004). Blast: At the core of a powerful and diverse set of sequence analysis tools. *Nucleic Acids Research*, 32:W20–W25. [Online; accessed 2024-04-22].

[Min et al., 2016] Min, S., Lee, B., and Yoon, S. (2016). Deep learning in bioinformatics. *Briefings in Bioinformatics*, page bbw068. [Online; accessed 2024-04-23].

[Min et al., 2020] Min, X., Ye, C., Liu, X., and Zeng, X. (2020). Predicting enhancer-promoter interactions by deep learning and matching heuristic. *Briefings in Bioinformatics*, 22(4). [Online; accessed 2024-04-25].

[Minchin and Busby, 2013] Minchin, S. and Busby, S. (2013). *Transcription factors*, pages 93–96. Elsevier. [Online; accessed 2024-05-16].

[Nambiar et al., 2020] Nambiar, A., Liu, S., Hopkins, M., Heflin, M., Maslov, S., and Ritz, A. (2020). Transforming the Language of Life: Transformer neural networks for protein prediction tasks. Technical report, Cold Spring Harbor Laboratory. [Online; accessed 2024-05-24].

[Neyshabur et al., 2017] Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. https://arxiv.org/abs/1706.08947.

[NLM, 2004] NLM (2004). Query input and database selection — blasttopics 0.1.1 documentation. National Library of Medicine.

[OpenAI et al., 2023] OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Kaiser, L., Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Kondraciuk, L., Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., Peres, F. d. A. B., Petrov, M., Pinto, H. P. d. O., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss,

C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. (2023). Gpt-4 technical report. https://arxiv.org/abs/2303.08774.

[Ortet et al., 2012] Ortet, P., De Luca, G., Whitworth, D. E., and Barakat, M. (2012). P2tf: A comprehensive resource for analysis of prokaryotic transcription factors. *BMC Genomics*, 13(1). [Online; accessed 2024-04-22].

[Oubounyt et al., 2019] Oubounyt, M., Louadi, Z., Tayara, H., and Chong, K. T. (2019). Deepromoter: Robust promoter predictor using deep learning. *Frontiers in Genetics*, 10. [Online; accessed 2024-04-25].

[Pan et al., 2018] Pan, X., Rijnbeek, P., Yan, J., and Shen, H.-B. (2018). Prediction of RNA-protein sequence and structure binding preferences using deep convolutional and recurrent neural networks. *BMC Genomics*, 19(1). [Online; accessed 2024-04-25].

[Park et al., 2020] Park, S., Koh, Y., Jeon, H., Kim, H., Yeo, Y., and Kang, J. (2020). Enhancing the interpretability of transcription factor binding site prediction using attention mechanism. *Scientific Reports*, 10(1). [Online; accessed 2024-04-25].

[Pujato et al., 2014] Pujato, M., Kieken, F., Skiles, A. A., Tapinos, N., and Fiser, A. (2014). Prediction of DNA binding motifs from 3d models of transcription factors; identifying TLX3 regulated genes. *Nucleic Acids Research*, 42(22):13500–13512. [Online; accessed 2024-05-16].

[Quang and Xie, 2016] Quang, D. and Xie, X. (2016). Danq: A hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research*, 44(11):e107–e107. [Online; accessed 2024-04-23].

[Radford and Narasimhan, 2018] Radford, A. and Narasimhan, K. (2018). [PDF] improving language understanding by generative pre-training.

[Salekin et al., 2017] Salekin, S., Zhang, J. M., and Huang, Y. (2017). A deep learning model for predicting transcription factor binding location at single nucleotide resolution. In *2017 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*. IEEE. [Online; accessed 2024-04-25].

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. https://arxiv.org/abs/1910.01108.

[Santos-Mendoza, 2023] Santos-Mendoza, T. (2023). The envelope (e) protein of sars-cov-2 as a pharmacological target. *Viruses*, 15(4):1000.

[Sapoval et al., 2022] Sapoval, N., Aghazadeh, A., Nute, M. G., Antunes, D. A., Balaji, A., Baraniuk, R., Barberan, C. J., Dannenfelser, R., Dun, C., Edrisi, M., Elworth, R. A. L., Kille, B., Kyrillidis, A., Nakhleh, L., Wolfe, C. R., Yan, Z., Yao, V., and Treangen, T. J. (2022). Current progress and open challenges for applying deep learning across the biosciences. *Nature Communications*, 13(1). [Online; accessed 2024-04-22].

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681. [Online; accessed 2024-05-16].

[Shen et al., 2022] Shen, W.-K., Chen, S.-Y., Gan, Z.-Q., Zhang, Y.-Z., Yue, T., Chen, M.-M., Xue, Y., Hu, H., and Guo, A.-Y. (2022). Animaltfdb 4.0: A comprehensive animal transcription factor database updated with variation and expression annotations. *Nucleic Acids Research*, 51(D1):D39–D45. [Online; accessed 2024-05-16].

[Smith, 2017] Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. [Online; accessed 2024-06-03].

[Sonnhammer et al., 1997] Sonnhammer, E. L., Eddy, S. R., and Durbin, R. (1997). Pfam: A comprehensive database of protein domain families based on seed alignments. *Proteins: Structure, Function, and Genetics*, 28(3):405–420. [Online; accessed 2024-04-22].

[Suzuki and Yagi, 1994] Suzuki, M. and Yagi, N. (1994). Dna recognition code of transcription factors in the helix-turn-helix, probe helix, hormone receptor, and zinc finger families. *Proceedings of the National Academy of Sciences*, 91(26):12357–12361. [Online; accessed 2024-05-16].

[Teixeira et al., 2022] Teixeira, M. C., Viana, R., Palma, M., Oliveira, J., Galocha, M., Mota, M. N., Couceiro, D., Pereira, M. G., Antunes, M., Costa, I. V., Pais, P., Parada, C., Chaouiya, C., Sá-Correia, I., and Monteiro, P. T. (2022). Yeastract+: A portal for the exploitation of global transcription regulation and metabolic model data in yeast biotechnology and pathogenesis. *Nucleic Acids Research*, 51(D1):D785–D791. [Online; accessed 2024-05-30].

[Thompson et al., 1994] Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). Clustal W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680. [Online; accessed 2024-04-22].

[Valenzuela-Valderrama et al., 2019] Valenzuela-Valderrama, M., Cerda-Opazo, P., Backert, S., González, M. F., Carrasco-Véliz, N., Jorquera-Cordero, C., Wehinger, S., Canales, J., Bravo, D., and Quest, A. F. G. (2019). The helicobacter pylori urease virulence factor is required for the induction of hypoxia-induced factor-1alpha in gastric cells. *Cancers*, 11(6):799.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. https://arxiv.org/abs/1706.03762.

[Wen et al., 2020] Wen, B., Zeng, W., Liao, Y., Shi, Z., Savage, S. R., Jiang, W., and Zhang, B. (2020). Deep learning in proteomics. *PROTEOMICS*, 20(21-22). [Online; accessed 2024-04-23].

[Wingender, 2013] Wingender, E. (2013). Criteria FOR AN UPDATED CLASSIFICATION OF HUMAN TRANSCRIPTION FACTOR DNA-BINDING DOMAINS. *Journal of Bioinformatics and Computational Biology*, 11(01):1340007. [Online; accessed 2024-05-16].

[Yang et al., 2017] Yang, B., Liu, F., Ren, C., Ouyang, Z., Xie, Z., Bo, X., and Shu, W. (2017). Biren: predicting enhancers with a deep-learning-based model using the dna sequence alone. *Bioinformatics*, 33(13):1930–1936.

[Yang et al., 2023] Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Yin, B., and Hu, X. (2023). Harnessing the Power of LLMs in practice: A survey on chatgpt and beyond. https://arxiv.org/abs/2304.13712.

[Zhang et al., 2023] Zhang, S., Fan, R., Liu, Y., Chen, S., Liu, Q., and Zeng, W. (2023). Applications of transformer-based language models in bioinformatics: A survey. *Bioinformatics Advances*, 3(1). [Online; accessed 2024-04-22].

[Zhou and Troyanskaya, 2015] Zhou, J. and Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12(10):931–934. [Online; accessed 2024-04-23].

[Zhou et al., 2023] Zhou, S., Lv, P., Li, M., Chen, Z., Xin, H., Reilly, S., and Zhang, X. (2023). Sars-cov-2 e protein: Pathogenesis and potential therapeutic development. *Biomedicine and Pharmacotherapy*, 159:114242.