

Thesis Title

by
Student Name

Professor SuperProf, Advisor

A thesis submitted in partial fulfillment
of the requirements for the
Degree of Bachelor of Arts with Honors
in Physics

WILLIAMS COLLEGE
Williamstown, Massachusetts
21 de noviembre de 2023

Abstract

Your abstract will summarize your thesis in one or two paragraphs. This brief summary should emphasize methods and results, not introductory material.

Executive Summary

Your executive summary will give a detailed summary of your thesis, hitting the high points and perhaps including a figure or two. This should have all of the important take-home messages; though details will of course be left for the thesis itself, here you should give enough detail for a reader to have a good idea of the content of the full document. Importantly, this summary should be able to stand alone, separate from the rest of the document, so although you will be emphasizing the key results of your work, you will probably also want to include a sentence or two of introduction and context for the work you have done.

Acknowledgments

The acknowledgment section is optional, but most theses will include one. Feel free to thank anyone who contributed to your effort if the mood strikes you. Inside jokes and small pieces of humor are fairly common here . . .

Índice general

Abstract	I
Executive Summary	II
Acknowledgments	III
1. Introduction	1
2. Marco Teórico	2
2.1. Planteamiento del Problema	2
2.2. Detección de Patologías en Pulmones Mediante Imágenes de Rayos-X	2
2.2.1. Datos	5
2.3. De RNN's a Transformers	6
2.3.1. Redes Neuronales Recurrentes más comunes	7
2.3.2. Compuertas LSTM y GRU	10
2.3.3. Mecanismos de Atención	14
2.4. El modelo Transformer	20
2.4.1. El Codificador y Decodificador	21
2.4.2. Multi-Head Self-Attention	21
2.4.3. Información Posicional	24

2.4.4. Problemas típicos en el entrenamiento de Transformers	25
3. Vision Transformers en Detección de Patologías en Pulmones con Rayos X	33
3.1. Detección de Patologías en Pulmones con Rayos X	33
3.2. Arquitecturas usadas	36
3.2.1. CNN y Transfer Learning	36
3.2.2. Vision Transformers	37
3.2.3. ViT con cabezas de atención Flexibles	39
3.2.4. Métricas de Evaluación	41
3.2.5. Resultados	44
4. Conclusiones y Trabajo Futuro	50
A. An appendix	51
A.1. About the bibliography	51

Índice de figuras

2.1.	RNN - Grafo Computacional	7
2.2.	RNN - CFG	8
2.3.	RNN - Image Captioning	9
2.4.	Computo del estado oculto y salida de una Red Neuronal Recurrente.	11
2.5.	Descripción.	12
2.6.	Descripción.	14
2.7.	Descripción.	15
2.8.	Modelo seq2seq propuesto por Bahdanau et al. con <i>Additive/Concat Attention</i>	17
2.9.	Modelo Transformer generalizado como modelo Secuencia a Secuencia	20
2.10.	Etapa Codificadora del Modelo Transformer. Pseudocódigo	21
2.11.	Etapa Decodificadora del Modelo Transformer. Pseudocódigo	22
2.12.	Esquema de entrenamiento e inferencia del modelo Transformer en un problema de Machine Translation.	23
2.13.	2000 Vectores de Positional Encoding con dimensiones de embedding=500. .	24
2.14.	Noam-Warmup con $warmup_steps = 4000$ y $d_m = 512$	25
2.15.	Learning rate sobre X 18000 iteraciones usando RAdam y lineal, exponencial warmup con Adam	27
2.16.	Histograma de gradientes del Algoritmo Adam con y sin etapa de <i>WarmUp</i> y usando inicialización <i>T-Fixup</i> . Imagen original de Huang et al.	28

ÍNDICE DE FIGURAS

VII

2.17. Visualización de 8 cabezas de atención sobre una tarea de Machine-Translation. Las matrices de atención tienden a ser ralas al tener carencia de relaciones relevantes entre diversas representaciones a distancias lejanas.	29
2.18. Transformer-XL. Para tratar con secuencias largas divide el proceso en se- cuencias más cortas creando estados ocultos intermedios y usandolos en el cálculo de las próximas secuencias. Figura obtenida de [19].	30
2.19. Fast-Former. Reemplaza la atención tradicional del transformer por una ite- rativa. En cada paso crea una consulta y clave global usando atención sobre estos mismos. Figura obtenida de [101]	32
3.1. Modelo ViT [23]. Representación original del ViT. Las imágenes a procesarse son descompuestas en una serie de parches (3x3 parches en este ejemplo). Los parches generados son proyectados a una dimension linear agregandose información posicional.	38
3.2. Scheme of the extension of the 15 pathology detector including a new branch for Tuberculosis detection.	43
3.3. ROC Curves (plots of FPR versus TPR) for the pathology and healthy classes.	48
3.4. Some X-ray images (top row) with different pathologies and their associated GradCam images (bottom row). The first column shows a grown heart, the second column corresponds to Pneumonia, the third column shows an unusual tissue in the lung, and the last column a case of COVID-19.	49

Capítulo 1

Introduction

Capítulo 2

Marco Teórico

2.1. Planteamiento del Problema

Aquí Especificar brevemente los problemas y cómo se piensan resolver

2.2. Detección de Patologías en Pulmones Mediante Imágenes de Rayos-X

A pesar de diversos esfuerzos para desarrollar métodos basados en aprendizaje máquina basados en análisis de imágenes de Rayos-X y Tomografías Computarizadas aún no están listos para uso clínico. Limitaciones como sesgos debido a bases de datos pequeñas o recopilaciones de diversas fuentes sin un tratamiento o normalización entre estos, así como enfoques de detección en enfermedades específicas dejando de lado la posible contribución a los modelos la inclusión de otras enfermedades. Por ello el trabajo presentado en este escrito se concentra en desarrollar un modelo basado en aprendizaje profundo atacando estos problemas. El modelo desarrollado es entrenado para la detección de 15 patologías de pulmones, incluyendo *COVID-19*.

El padecimiento por *COVID-19* es una enfermedad contagiosa causada por el *Síndrome Respiratorio Agudo Severo Coronavirus 2* o *SARS-CoV-2* por sus siglas en inglés (*Severe Acute Respiratory Syndrome Coronavirus 2*) reportada por primera vez en diciembre del año 2019 como un nuevo tipo de pneumonia viral [34]. Pocos meses después, en marzo del 2020 el *COVID-19* fue declarado como pandemia a nivel mundial por la Organización Mundial de la Salud (*WHO*) [61]. Los métodos más eficaces de detección de *COVID-19* son la prueba clínica de Reacción en Cadena de Polimerasa con Transcripción Inversa (*RT-PCR*) también

llamada genéricamente *molecular photocopying test* pues es usada para amplificar-copiar pequeños segmentos de DNA y detectar material genético de un organismo en específico como el virus *SARS-CoV-2* y mediante la búsqueda de anticuerpos desarrollados por el organismo como respuesta a la enfermedad con la Prueba Rápida de Anticuerpos (*RAT*) [30, 3, 74, 44]. Puesto que los anticuerpos tardan en generarse entre los 10 y 20 días después de la infección [50, 62, 89], la prueba tipo *PCR* es preferida como método de detección temprana. En la ausencia de prueba *PCR*, los pacientes con sintomatología similar a la provocada por *COVID-19* solo pueden ser diagnosticados con pneumonia atípica como padecimiento. Por ello, diversos métodos de análisis de imágenes basados en técnicas de Inteligencia Artificial han sido desarrollados para la detección de *COVID-19* usando imágenes de Rayos X y Tomografías Computarizadas. Reportes clínicos indican que imágenes de Rayos X y Tomografías Computarizadas de pecho pueden mostrar efectos de afectaciones por *COVID-19*. Dichos efectos pueden ser apreciados en los pulmones incluso en casos donde la prueba PCR resulta en Falso Negativo [2, 100]. Así, la principal motivación es desarrollar métodos alternativos que ayuden a la detección de *COVID-19* dada la limitada disponibilidad, creciente demanda, costo asociado y la obtención de resultados inmediatos de la aplicación de técnicas como *PCR* en todo el mundo.

Un práctico y exitoso enfoque en la implementación de Redes Neuronales Profundas (DNNs) de dominio específico [46] basados en técnicas de clasificación es usar Transferencia de Conocimiento (Deep Transfer Learning, DTL) [49, 60, 86]. Esta técnica fue inicialmente desarrollada para implementar modelos de dominio en específico en situaciones en las que se cuenta con una cantidad de datos limitada, pero también se ha visto que usar técnicas de DTL resulta en un método efectivo para entrenar modelos de dominio en específico aún cuando se tiene suficientes datos. Particularmente, en problemas de clasificación, la Transferencia de Conocimiento consiste en reusar modelos de Aprendizaje Profundo entrenados en problemas de dominio general en donde los conjuntos de datos son lo suficientemente grandes. Dado que la tarea de clasificación contiene un gran número de clases y datos, los modelos entrenados pueden generalizar y extraer mejores características de bajo nivel. Por lo que, el uso de Transferencia de Conocimiento también ha sido efectivo en el desarrollo de modelos específicos para el análisis de imágenes médicas [21, 42, 84], tales como la detección de *COVID-19* a partir de imágenes de Rayos X, problema atacado ampliamente por la comunidad usando Transferencia de Conocimiento en Modelos basados en Aprendizaje Profundo [1, 2, 84].

Un popular método para la detección de *COVID-19* es *CoroNet* [93], un modelo con arquitectura basado en redes neuronales profundas. *CoroNet* clasifica imágenes de radiografías de Rayos X en 4 clases: *COVID-19*, neumonía bacterial, neumonía viral e imágenes sin ninguno de los tres padecimientos anteriores. Bressem et al. presenta un estudio sobre cuales arquitecturas usadas como configuración inicial de la red *backbone* encargada de tareas principalmente de extracción de características es más adecuada para desarrollar modelos de clasificación con imágenes radiográficas. Limitan el estudio a la detección de solo 5 patologías: cardiomegalia, edema, consolidación, atelectasia y derrame pleural encontrado que es mucho más

importante el tamaño de lote que la red usada como *backbone*, es decir, la mejoría de los modelos dependían mucho más del proceso de entrenamiento que de los datos y la arquitectura. En [107] se reporta un ejemplo de clasificador de *COVID-19*, neumonía y no *COVID-19*. Otro reciente estudio compara redes neuronales para la clasificación de radiografías de pecho entre *COVID-19* y neumonía [77]. Para ello construyen clasificadores binarios usando redes pre-entrenadas con la base de datos ImageNet, considerando los modelos VGG16 y VGG19, DenseNet121, Inception-ResNet-V2, InceptionV3, ResNet50 y Xception. Con base a los resultados los modelos basados en DenseNet121 121 tuvieron el mejor accuracy (99,48 %) seguidos muy creca por ResNet50 y VGG19 con 99,32 % y 99,18 % respectivamente. Los resultados anteriores son muy cercanos a los reportados por *InstaCovNet-19*, un clasificador binario de *COVID-19* y no *COVID-19* [29]. Por otro lado, Bassi and Attux proponen una estrategia de Transferencia de Conocimiento que tomas los modelos DenseNet121 y DenseNet201 (los sufijos 121, y 201 hacen referencia a el número de capas del modelo DenseNet) como backbone, posteriormente reemplazan las neuronas correspondientes a la capa de salida para clasificar radiografías de Rayos X entre *COVID-19*, neumonía, y sin ninguna de las patologías anteriores. De acuerdo a Shoeibi et al. la mayoría de los modelos de redes neuronales basados en aprendizaje profundo tienen un accuracy dentro del 90 % al 100 % para clasificación binaria entre *COVID-19* y no *COVID-19*.

A pesar de los esfuerzos en desarrollar diversos métodos de aprendizaje máquina para la detección de *COVID-19* basados en Rayos X y Tomografías Computarizadas, los resultados obtenidos y reportados hasta el momento aún no están listos para uso clínico [70]. Shuja et al. presenta una relación de los conjuntos de datos abiertos de *COVID-19* categorizándolos por tipo (imágenes biomédicas, datos textuales y de audio), aplicaciones y métodos aplicados de *IA*, *Big Data* y estadísticos. Sin embargo, las imágenes en los datasets mencionados son reducidos y limitados a regiones específicas alrededor del mundo. Por otro lado, Greenspan et al. mencionan que la mayoría de los modelos reportados fueron probados en bajo en esquema de diagnóstico bastante estrecho, puesto que los modelos deberían ser capaces de detectar *COVID-19* en conjunto con una amplia variedad de patologías. De acuerdo a Roberts et al. los fallos comunes son, entre otros, el sesgo en datasets pequeños o datasets no normalizados recolectados de una larga variedad de fuentes. Roberts et al. también argumenta la importancia de desarrollar modelos no solo para la clasificación binaria de *COVID-19*, sino además poder distinguirlo de otros tipos de neumonías virales y bacteriales. El trabajo descrito a continuación se centra en atacar el problema en los términos anteriores, no solamente haciendo la distinción entre *COVID-19* y otros tipos de neumonías sino a través de diversas enfermedades en afán de ayudar a clínicos en el diagnóstico de otras patologías mas allá de neumonía.

2.2.1. Datos

CXR8 Dataset

El dataset es extraído de las bases de datos del hospital clínico de investigación *NIH Clinical Center* perteneciente al instituto *National Institutes of Health*. Está formado por alrededor del 60 por ciento de imágenes de Rayos X frontales de pecho capturas en dicho hospital y es considerado como uno de los conjuntos de datos más representativos. Contiene 112,120 imágenes de Rayos X frontales (frontal-view) de un total de 30,805 pacientes únicos. Cada imagen contiene etiquetas correspondientes a 14 enfermedades (más de una etiqueta puede estar asociada a una imagen) extraídas usando técnicas de Procesamiento de Lenguaje Natural (NLP) de los reportes médicos realizados por radiólogos asociados con un accuracy mayor al 90 por ciento [97]. Las 14 patologías incluidas son las siguientes: Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural-thickening, Cardiomegaly, Nodule, Mass y Hernia.

COVIDx Dataset

Es una base de datos de uso público construida a través de las contribuciones de la comunidad científica con la finalidad de construir método que ayuden a combatir el padecimiento y propagación de COVID-19 [92]. Contiene 13,975 imágenes de Rayos X de un total de 13,870 casos de pacientes (las cifras pueden variar dado a su constante actualización) recopiladas a través de 5 repositorios de datos; *COVID-19 Image Data Collection* [18], *Figure 1 COVID-19 Chest X-ray Dataset Initiative* [95], *ActualMed COVID-19 Chest X-ray Dataset Initiative* [94], *RSNA Pneumonia Detection Challenge dataset* [59], *COVID-19 radiography database* y [58]. Los conjuntos de datos contienen casos de pacientes con y sin padecimientos de pulmonía causada por COVID-19.

SIIM-FISABIO-RSNA COVID-19 Detection Dataset

Es un conjunto de datos creado, organizado y liberado por la *Society for Imaging Informatics in Medicine (SIIM)* en conjunto con la *Foundation for the Promotion of Health and Biomedical Research of Valencia Region (FISABIO)*, *Medical Imaging Databank of the Valencia Region (BIMCV)*, y *Radiological Society of North America (RSNA)* [81] a través de un concurso en la plataforma de *Kaggle*. Esta competición está enfocada en localizar e identificar anomalías en radiografías de pecho y clasificarlas en cuatro casos de neumonía causada por COVID-19 [43]:

- *Typical appearance:* Contiene observaciones típicas a neumonía causada por COVID-19. Sin embargo, estas pueden presentarse en conjunto con otros infecciones, reacciones

a medicamentos, u otras causas de lesiones agudas en los pulmones.

- *Indeterminate appearance*: Presenta observaciones indeterminadas a neumonía causada por *COVID-19* y que pueden ocurrir en conjunto con a una variedad de condiciones infecciosas y no infecciosas.
- *Atypical appearance*: Observaciones atípicas a las reportadas a neumonía causada por *COVID-19* y otros diagnósticos alternativos deben ser considerados.
- *Negative for pneumonia*: No se encuentran observaciones de neumonía causada por *COVID-19*. Sin embargo, las radiografías pueden no presentar aún elementos visibles en etapas tempranas de neumonía por *COVID-19*.

Tuberculosis X-ray (TBX11K) dataset

Es uno de los datasets más grandes actualmente con imágenes de Rayos X con Tuberculosis. Este dataset contiene 11200 imágenes de Rayos X con anotaciones marcando los bounding boxes de las áreas donde con observaciones de este padecimiento. Se clasifican en 4 distintas categorías; saludable, Tuberculosis activa (active TB), Tuberculosis latente (latent TB), con padecimientos otros a Tuberculosis (unhealthy but non-TB). Las imágenes son de calidad mucho mayor a la mayoría de los datasets anteriores con una resolución de 3000x3000 pixeles, Liu et al. mencionan que usar una resolución de 500x500 es suficiente para entrenar modelos de aprendizaje profundo de detección y clasificación de Tuberculosis.

2.3. De RNN's a Transformers

Las **Redes Neuronales Recurrentes** o **RNN** (por sus siglas en Inglés) basadas en el trabajo de Rumelhart et al. datan del año 1986. Este tipo de redes están especializadas en el procesamiento de datos que contienen información temporal, mejorando los resultados obtenidos por otros tipos de redes como *Redes FeedForward* o *Redes Convolucionales*.

La idea principal detrás de estos modelos de redes es el concepto de *Parameter Sharing*. Usando *Parameter Sharing* un modelo puede generalizar mejor cuando la información está contenida en diferentes partes de una secuencia. Así, el modelo no necesita aprender independientemente todas las reglas que forman la secuencias, sino que ahora, la salida para cada elemento perteneciente a un tiempo t está determinada por la salida del elemento anterior $t - 1$. Resultando en una recurrencia con las mismas reglas de actualización aplicadas a cada elemento en el tiempo. La ecuación 2.1 representa este proceso; $h^{(t)}$ es el estado de la recurrencia definida por una función f sobre un elemento $x^{(t)}t$ de la secuencia X en el tiempo t y θ son los parámetros compartidos.

$$h^{(t)} = f(x^{(t)}, h^{(t-1)}; \theta) \quad (2.1)$$

En una *RNN* vista como un *gráfico computacional dirigido y acíclico*, cada nodo representa un estado en la recurrencia y procesa la información de la secuencia X con los mismos parámetros θ en cada paso, observe la figura 2.1.

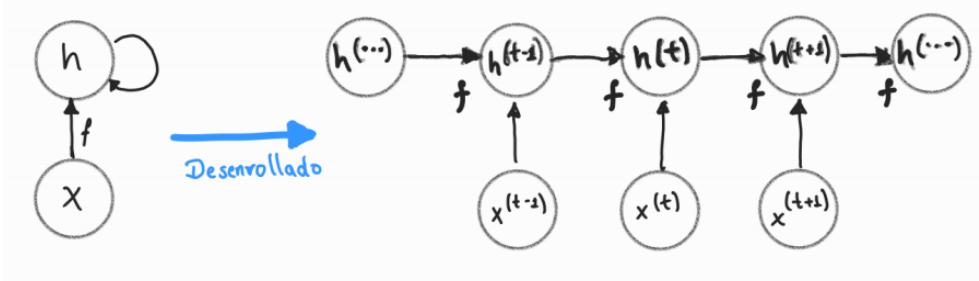


Figura 2.1: Grafo computacional generado por una *RNN* al “desenrollar” la recurrencia. Usando los parámetros compartidos en cada nodo y con cada elemento $x^{(t)}$ de la secuencia genera un nuevo estado oculto $h^{(t)}$ para retroalimentar nuevamente la entrada del siguiente nodo.

2.3.1. Redes Neuronales Recurrentes más comunes

Existen diversas formas como construir *Redes Neuronales Recurrentes*, estas pueden producir una salida en cada paso de tiempo o tener solo una al final de la recurrencia o tener conexiones entre unidades ocultas. La manera más común de implementar una *RNN* está ilustrada en la figura 2.2a. En esta figura, cada etapa de la recurrencia es retroalimentada por la activación del estado oculto previo. Así, $h^{(t)}$ contiene información codificada de elementos previos de la secuencia que puede ser usada en el futuro para obtener una salida $O^{(t+1)}$. En la figura 2.2b se cambia la retroalimentación de $h^{(t)}$ por $o^{(t)}$. Nótese que en este caso, la red es entrenada para obtener un valor en específico $o^{(t)}$ lo que provocaría que gran parte de la información de los estados ocultos pasados $h^{(t-1)}, h^{(t-2)}, \dots$ no se transmita. La diferencia entre los dos esquemas anteriores es que la red 2.2a es entrenada para decidir qué información debe transmitir en el futuro a través de los estados ocultos, en cambio, en la figura 2.2b cada estado está conectado con el pasado a través de la predicción del paso anterior, perdiendo así gran parte de la información codificada en cada estado oculto $h^{(t)}$. Este no sería un problema si la salida $O^{(t-1)}$ fuese lo suficientemente Enriquecedora y en altas dimensiones.

Por otro lado, la *RNN* representada en la figura 2.2c tiene una sola salida al final de la recurrencia. Al contrario de las anteriores, este tipo de redes pueden ser usadas para resumir información contenida en la secuencia para finalmente predecir un único valor final.

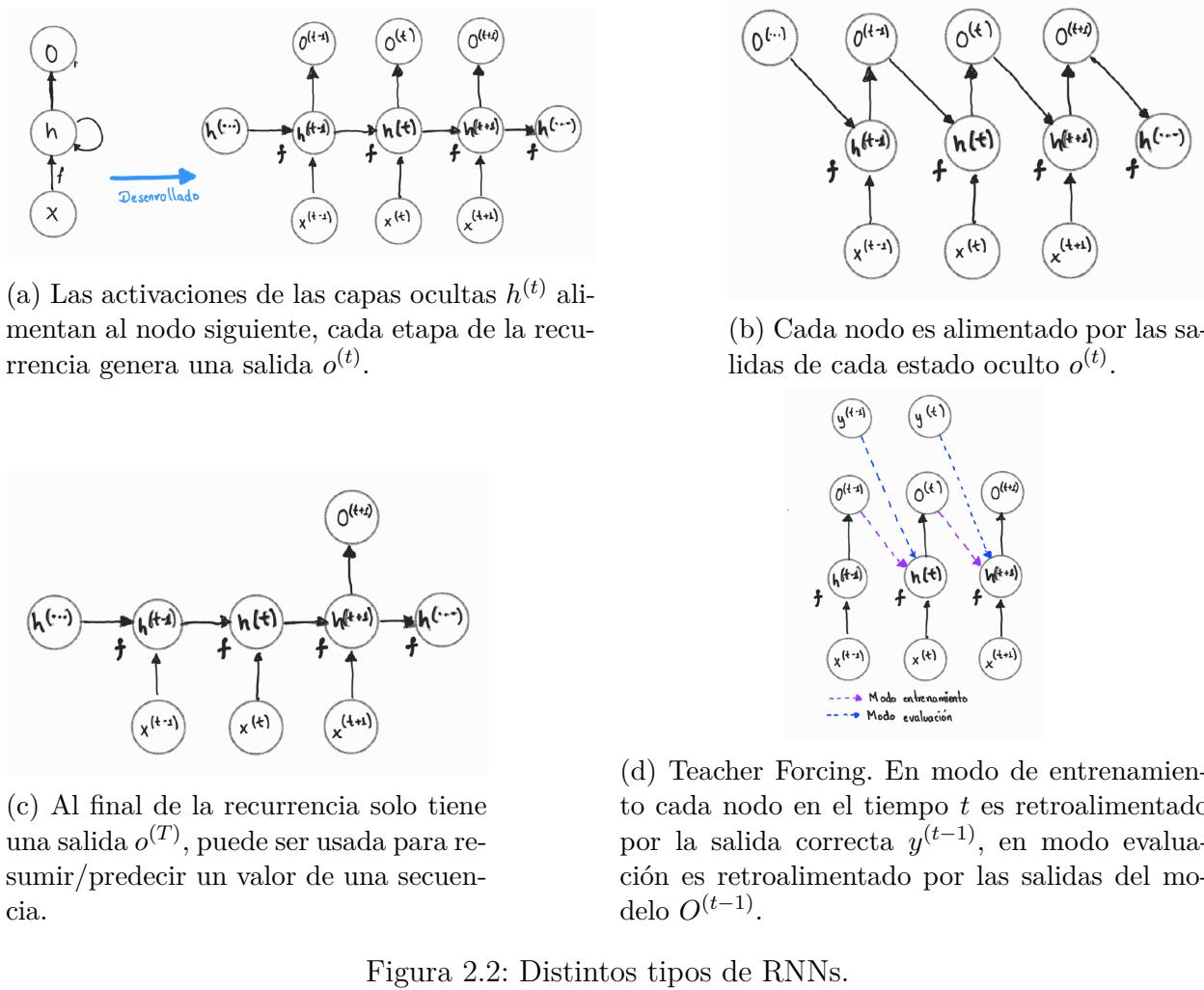


Figura 2.2: Distintos tipos de RNNs.

El *Análisis de Sentimiento* en textos es una tarea común que puede ser representada con este esquema de red. En la figura 2.2d vemos un modelo de *RNN* entrenado mediante el proceso de Teacher Forcing; durante el entrenamiento la red es retroalimentada con las salidas esperadas del modelo $y^{(t)}$ en el tiempo $t+1$. La ventaja de esta red es que al ser eliminadas las conexiones entre estados ocultos, las funciones de pérdida basadas en comparar la predicción en el tiempo t con el valor objetivo $y^{(t)}$ pueden ser desacopladas. Por tanto, el entrenamiento puede ser paralelizado al calcular el gradiente para cada tiempo t por separado, puesto que ya tenemos el valor ideal para esta salida.

Finalmente, en la figura 2.3 la *Red Neuronal Recurrente* es modificada para esta vez no procesar una secuencia, sino, procesar un solo vector en cada paso. El estado oculto previo $h^{(t-1)}$ retroalimenta al siguiente paso t así como la predicción esperada $y^{(t)}$, que a su vez, es usada para calcular la función de costo del paso anterior $L^{(t-1)}$. Esta estructura de red puede ser implementada en tareas como *Image Captioning*, en donde la entrada es una imagen y

la salida una secuencia de palabras que describen esta misma.

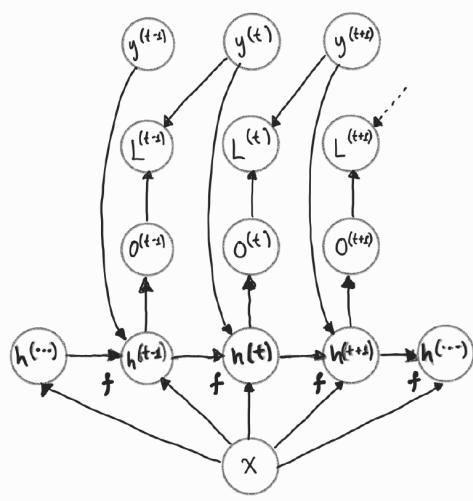


Figura 2.3: Modelo usado para tareas de *Image Captioning*, la entrada es una sola imagen y la red predice una secuencia de palabras que describen dicha imagen La salida esperada $y^{(t)}$ sirve como objetivo para la función de costo del paso anterior y como entrada en cada paso.

Los modelos ejemplificados anteriormente son construidos de forma *causal*, es decir, la secuencia es procesada en un solo sentido en donde la información pasada es transmitida hacia estados futuros. Sin embargo, este flujo de información puede ser insuficiente para resolver todas las tareas. En *Modelo de Lenguaje* se aprende la estructura estadística del lenguaje con el que fue entrenado y su meta es predecir la siguiente palabra, n-grama o letra dado un contexto antes visto. En otros términos, dada una secuencia de texto de longitud T $x^{(1)}, x^{(2)}, \dots, x^{(T)}$ con $x \in \mathcal{R}^{1 \times d}$ donde d es la dimensión de la codificación de las palabras, la meta es predecir la probabilidad conjunta de la secuencia:

$$P(x^{(1)}, x^{(2)}, \dots, x^{(T)}) = \prod_{t=1}^T P(x^{(t)} | x^{(t)}, \dots, x^{(t-1)}) \quad (2.2)$$

Con ello, un modelo de lenguaje basado en *Redes Neuronales Recurrentes* es capaz de predecir un siguiente elemento $\hat{x}^{(t)}$ simplemente obteniéndolo de la secuencia mediante:

$$\hat{x}^{(t)} \approx P(x^{(t)} | x^{(t-1)}, \dots, x^{(1)}) \approx P(x^{(t)} | h^{(t-1)}) \quad (2.3)$$

donde $h^{(t-1)}$ es el estado oculto que almacena la información pasada hasta el tiempo t tal y como se definió en 2.1.

Sin embargo, la información previa de la secuencia codificada en $h^{(t)}$ no siempre contiene los elementos necesarios para que el modelo pueda predecir correctamente el siguiente elemento, observe la siguiente oración:

« *Ella estaba muy _____, después de que Alejandra vió el amanecer en la playa* »

En la oración anterior, el espacio en blanco puede ser completado con algún adjetivo calificativo; *contenta*, *enojada*, *maravillada*, etc. Gracias a la información provista por la parte final de la oración, podemos deducir que de las 3 opciones la menos probable de elegir es *enojada*. Es decir, usamos información del futuro que no pudo haber sido vista por una red (que procesa la información en forma causal) para tomar la mejor elección. Una ligera modificación fácilmente aplicable a estos modelos es que las secuencias sean procesadas en ambas direcciones, las **Redes Neuronales Recurrentes Bidireccionales** [75].

Una *RNN Bidireccional* procesa la secuencia en ambos sentidos (una *RNN* en un sentido y otra en el otro), capturando información del pasado en el estado oculto $\overrightarrow{h}^{(t)}$ cuando la recurrencia es del inicio al final de la secuencia e información del futuro en $\overleftarrow{h}^{(t)}$ cuando la recurrencia es del final al inicio de la secuencia. Finalmente, el estado oculto $h^{(t)}$ es una concatenación de ambos estados $\overrightarrow{h}^{(t)}$ y $\overleftarrow{h}^{(t)}$, vea la ecuación 2.4. Por lo cual, la salida $o^{(t)}$ ahora puede ser calculada con información tanto del futuro como del pasado 2.5.

$$\begin{aligned}\overrightarrow{h}^{(t)} &= f(x^{(t)}, \overrightarrow{h}^{(t-1)}; \theta_f) \\ \overleftarrow{h}^{(t)} &= f(x^{(t)}, \overleftarrow{h}^{(t+1)}; \theta_b) \\ h^{(t)} &= \text{Concat}(\overrightarrow{h}^{(t)}, \overleftarrow{h}^{(t)})\end{aligned}\tag{2.4}$$

$$o^{(t)} = g(h^{(t)}; \theta_{out})\tag{2.5}$$

2.3.2. Compuertas LSTM y GRU

Hasta el momento, se ha hecho mención de las salidas $o^{(t)}$ y estados ocultos $h^{(t)}$ solo como el resultado de operaciones aplicadas por dos funciones; g y f respectivamente. Existen varias alternativas de construir una *RNN*, una de las maneras más comunes es usando 2.6 y 2.7:

$$h^{(t)} = \phi(x^{(t)}W_x + h^{(t-1)}W_h + b)\tag{2.6}$$

$$o^{(t)} = x^{(t)}W_{out} + b\tag{2.7}$$

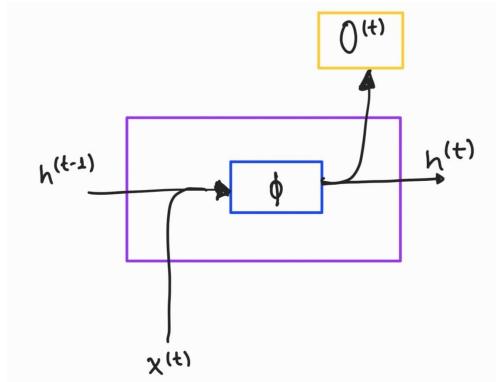


Figura 2.4: Computo del estado oculto y salida de una Red Neuronal Recurrente.

Donde los parámetros compartidos de la red ahora son descritos por las matrices $W_x \in \mathbb{R}^{d \times k}$, $W_h \in \mathbb{R}^{k \times k}$ y $W_{out} \in \mathbb{R}^{k \times q}$ con k como la dimensión del estado oculto, q la dimensión de las salidas $o^{(t)}$, $b \in \mathbb{R}^{1 \times q}$ el parámetro de sesgo y ϕ es la función de activación. De esta manera, los pesos de los parámetros aprendidos en la matriz W_h determinan cómo será usada la información del pasado, codificada en $h^{(t-1)}$. Posteriormente, es incluida a la codificación de la información del tiempo actual t calculada por W_x . La figura 2.4 representa gráficamente la lógica usada para calcular los estados ocultos y las salidas de la red.

Sin embargo el cálculo de los estados ocultos mediante 2.6 presenta algunos problemas. La interacción entre la información del pasado y la actual siempre es "plana", es decir, la información fluye a través del tiempo de la misma manera sin forma de dar prioridad o ignorar parte de esta. Por lo que resulta una tarea un poco más complicada preservar información relevante a en cada paso o desechar información que ya no es útil para la red. También, causado por este mismo flujo de los datos, la información del pasado poco a poco es opacada por nueva información, impidiendo que se puedan encontrar dependencias de información en secuencias largas en tiempos distantes; comúnmente se hace referencia a este problema como *The Short-term Memory Problem* en inglés [7]. Aunado a problemas como el *Desvanecimiento o Explosión del Gradiente* [39] [64], y acentuándose aun más debido a las matrices de pesos compartidos en la recurrencia. Dichas multiplicaciones en la recurrencia tienen similitud al método de potencia, en donde cualquier componente en la matriz inicial que no esté alineada con el vector propio asociado al mayor valor propio son eventualmente descartados [25, pp. 390-392]), por ende, los resultados de este producto tendrán a ser cercanos a cero (desvanecerse) o explotar dependiendo de la magnitud de la matriz de pesos.

Una manera de solventar los problemas anteriores son las **Redes Neuronales con Compúertas**, creadas con la idea de crear conexiones a través del tiempo de tal manera de tener gradientes que no se desvanezcan o exploten, convirtiéndose además en un mecanismo para olvidar información pasada y decidiendo automáticamente cuándo y cuánto de la información debe prevalecer.

LSTM

Long Short-Term Memory, LSTM por sus siglas en inglés, fue propuesta en 1997 por Hochreiter and Schmidhuber, como un método de preservar dependencias de información relevante distantes a corto plazo. Las *LSTM* introducen un nuevo componente la *Celda de Memoria* cuya función es guardar información a través del tiempo y es controlada por distintas compuertas. Las compuertas aprenden a distinguir que información es relevante y cual no. Hay 3 de ellas, la *Compuerta de Entrada*, la *Compuerta de Olvido* y la *Compuerta de Salida*. La *Compuerta de Entrada* $I^{(t)}$ (véase la figura 2.5) determina cuanta información actual debe ser contemplada a través de la *Memoria Candidata* $\tilde{C}^{(t)}$ para actualizar la *Celda de Memoria* $C^{(t)}$. La *Compuerta de Olvido* indica qué información del pasado debe ser desechara de la *Celda de Memoria* $C^{(t-1)}$ y la *Compuerta de Salida* ayuda a determinar el nuevo estado $h^{(t)}$ via la *Celda de Memoria* actual $C^{(t)}$.

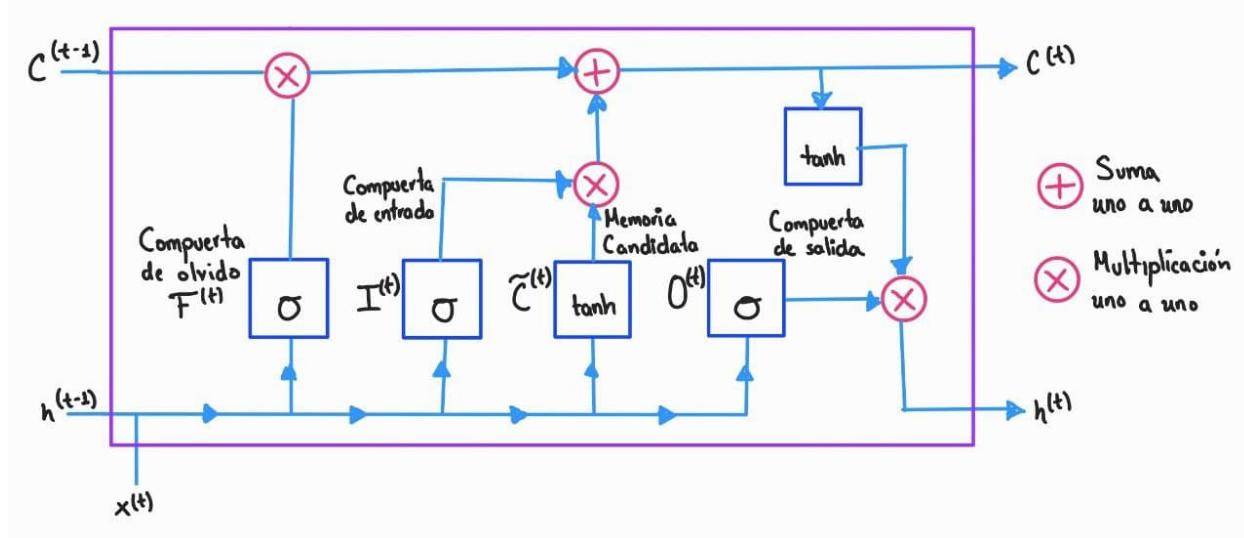


Figura 2.5: Descripción.

Las ecuaciones 2.8 rigen el comportamiento de Compuertas de Entrada, Salida y Olvido.

$$\begin{aligned} I^{(t)} &= \sigma(x^{(t)}W_{xi} + h^{(t-1)}W_{hi} + b_i) \\ F^{(t)} &= \sigma(x^{(t)}W_{xf} + h^{(t-1)}W_{hf} + b_f) \\ O^{(t)} &= \sigma(x^{(t)}W_{xo} + h^{(t-1)}W_{ho} + b_o) \end{aligned} \quad (2.8)$$

Donde $W_{xi}, W_{xf}, W_{xo} \in \mathbb{R}^{d \times k}$, $W_{hi}, W_{hf}, W_{ho} \in \mathbb{R}^{k \times k}$ y $b_i, b_f, b_o \in \mathbb{R}^{1 \times k}$

La Memoria Candidata y la Celda de memoria son actualizadas mediante:

$$\begin{aligned}\tilde{C}^{(t)} &= \tanh(x^{(t)}W_{xi} + h^{(t)}W_{hc} + b_c) \\ C^{(t)} &= F^{(t)} \odot C^{(t-1)} + I^{(t)} \odot \tilde{C}^{(t)}\end{aligned}\tag{2.9}$$

Donde $W_{xi} \in \mathbb{R}^{d \times k}$, $W_{hc} \in \mathbb{R}^{k \times k}$ y $b_c \in \mathbb{R}^{1 \times k}$

Y finalmente el estado oculto $h^{(t)}$ esta dado por:

$$h^{(t)} = O^{(t)} \odot \tanh(C^{(t)})\tag{2.10}$$

σ y \odot denotan la función de activación sigmoide y la multiplicación uno a uno respectivamente.

GRU

Gated Recurrent Units o **GRU** por sus siglas en inglés, fueron propuestas en 2014 [14] como una alternativa computacionalmente más rápida y con similar rendimiento que las *LSTM* [17]. A diferencia de anteriores, las *GRU* prescinden de la *Celda de Memoria* y utilizan un par de compuertas (la *Compuerta de Actualización* y la de *Olvido*) para decidir que información aún es necesaria que esté codificada dentro del estado oculto, véase la ecuación 2.11. La *Compuerta de Olvido* permite decidir que del pasado aún debe ser transmitido a futuros estados o de otro modo ser desechara. La *Compuerta de Actualización* indica que información nueva es relevante y necesita ser incorporada al no estar codificada dentro del estado oculto, véase la ecuación 2.12.

$$\begin{aligned}R^{(t)} &= \sigma(x^{(t)}W_{xR} + h^{(t-1)}W_{hR} + b_R) \\ Z^{(t)} &= \sigma(x^{(t)}W_{xZ} + h^{(t-1)}W_{hZ} + b_Z)\end{aligned}\tag{2.11}$$

$$\begin{aligned}\tilde{h}^{(t)} &= \tanh(x^{(t)}W_{xh} + (R^{(t)} \odot h^{(t-1)})W_{hh} + b_h) \\ h^{(t)} &= Z^{(t)} \odot h^{(t-1)} + (1 - Z^{(t)}) \odot \tilde{h}^{(t-1)}\end{aligned}\tag{2.12}$$

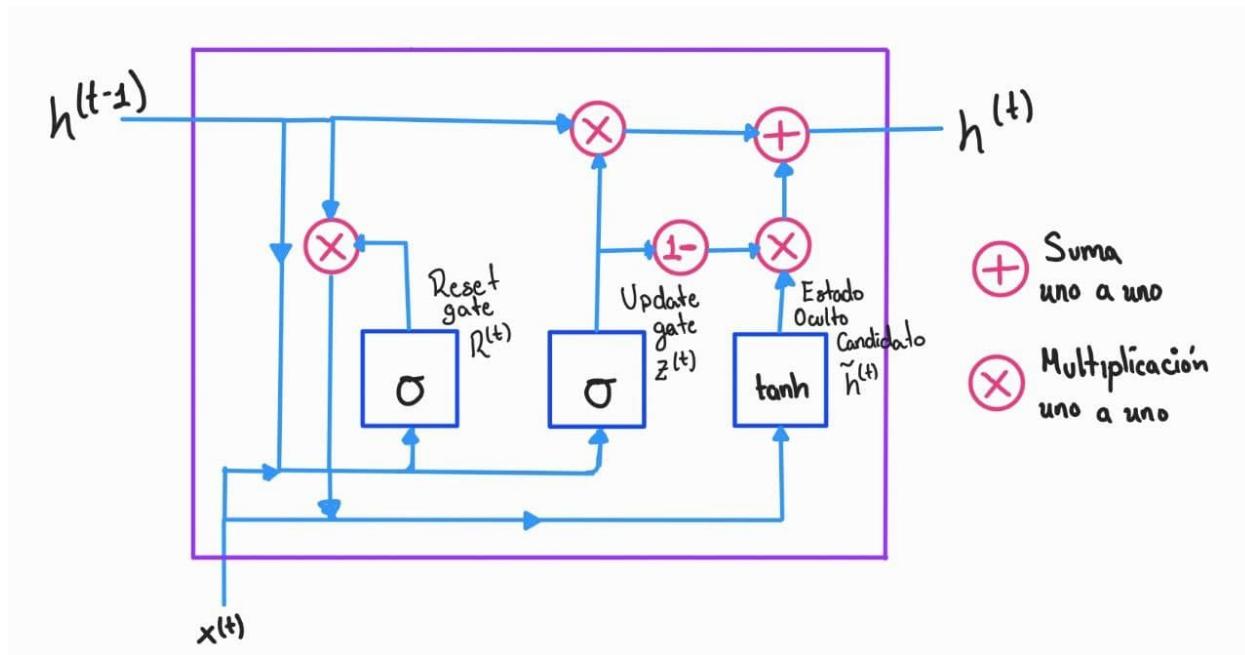


Figura 2.6: Descripción.

2.3.3. Mecanismos de Atención

Una de las arquitecturas comunes vistas previamente es la mostrada en la figura 2.2c cuya información procesada es resumida en una sola salida. Este tipo de red es usada como parte de las soluciones en tareas de reconocimiento de voz (*Speech Recognition*), traducción de lenguaje (*Machine Translation*) o asistencia en respuestas automáticas (*Question Answering*), entre otros, típicamente bajo modelos Secuencia a Secuencia (*Sequence to Sequence, Seq2Seq*) [15]. Los modelos *Seq2Seq* están formados por dos redes neuronales como la mostrada en 2.7. La primera se comporta como un *codificador* al resumir la entrada y producir un vector de salida de tamaño fijo llamado *vector de contexto*. La segunda red se comporta como un *decodificador*, este es inicializado y condicionado con el *vector de contexto* para obtener una transformación de la entrada no necesariamente del mismo tamaño de secuencia, debido a que en tareas como traducir una oración de un lenguaje a otro donde la traducción no siempre contiene las misma cantidad de palabras usadas en el idioma original.

Por ejemplo, en tareas de *Machine Translation* el *codificador* esta formado por una *RNN* Bidireccional que lee y procesa un conjunto de vectores $X = (x^{(1)}, x^{(2)}, \dots, x^{(T_x)})$ para obtener un vector de contexto C . La forma más común es como en 2.13:

$$\begin{aligned} h^{(t)} &= f_{bi}(x^{(t)}, h^{(t-1)}; \theta_f, \theta_b) \\ C &= q(h^{(1)}, h^{(2)}, \dots, h^{(T)}) \end{aligned} \tag{2.13}$$

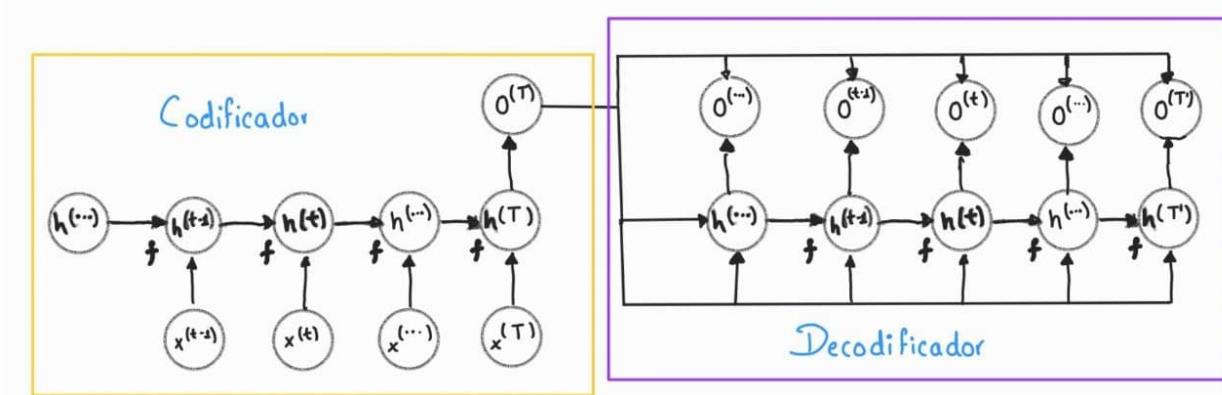


Figura 2.7: Descripción.

Recordemos que $h^{(t)}$ es el estado oculto generado por la concatenación de los dos estados ocultos generados por la *RNN Bidireccional*, f_{bi} y q son funciones no lineales, ya sea, una *LSTM* para f_{bi} y $q(h^{(1)}, h^{(2)}, \dots, h^{(T)}) = h^{(T)}$, equivalente a tomar solo el ultimo estado oculto como vector de contexto C . El *decodificador* es entrenado para predecir la siguiente palabra $y^{(t')}$ dado el vector de contexto C y todas las palabras previas predichas. En otras palabras, el decodificador define la probabilidad conjunta modelada por una *RNN*:

$$p(Y) = \prod_{t=1}^{T_y} p(y^{(t)} | \{y^{(1)}, \dots, y^{(t-1)}\}, C) \quad (2.14)$$

$$p(y^{(t)} | \{y^{(1)}, \dots, y^{(t-1)}\}, C) = g(y^{(t-1)}, s^{(t)}, C; \theta_g) \quad (2.15)$$

donde g es una función no lineal que emite la probabilidad de $y^{(t)}$ y $s^{(t)}$ es el estado oculto del *decodificador* 2.16.

$$s^{(t)} = f(s^{(t)}, y^{(t-1)}, C; \theta_s) \quad (2.16)$$

Sin embargo, cuando las secuencias son bastante largas el *vector de contexto* emitido por el *codificador* no es lo suficientemente grande como para resumir correctamente la secuencia y por tanto, la información inicial de la entrada es olvidada, teniendo escasa presencia en estados ocultos más lejanos. En 2015 Bahdanau et al. observaron estos efectos y propusieron una forma de minimizarlos, los **Mecanismos de Atención**.

La función principal de los **Mecanismos de Atención** es permitir que el *decodificador* pueda acceder al historial completo de los estados ocultos del *codificador*, así, ahora podrá contar con un mecanismo para selectivamente centrarse en las distintas partes de la secuencia que tienen mayor influencia sobre una la salida esperada a cierto tiempo.

Por tanto, las palabras predichas no son calculadas por un único *vector de contexto* generado por el *codificador*, sino que para cada objetivo $y^{(t)}$ se calcula un nuevo *vector de contexto* $c^{(t)}$:

$$p(y^{(t)} | \{y^{(1)}, \dots, y^{(t-1)}\}, c^{(t)}) = g(y^{(t-1)}, s^{(t)}, c^{(t)}; \theta_g) \quad (2.17)$$

$$s^{(t)} = f(s^{(t)}, y^{(t-1)}, c^{(t)}; \theta_s) \quad (2.18)$$

Dado que cada estado oculto $h^{(t)}$ contiene mucho mejor la información que se encuentran alrededor del t-ésimo término, se puede generar cada vector de contexto como una suma pesada de sobre los estados ocultos del *codificador*. Estos pesos nos ayudan a determinar que tan importante es la información codificada por cada estado oculto y al momento de obtener la salida del t-ésimo valor “prestar atención” a aquellos que son más relevantes para esta predicción:

$$c^{(t)} = \sum_{i=1}^{T_x} \alpha_{t,i} h^{(i)} \quad (2.19)$$

aquí cada peso $\alpha_{t,i}$ indica que tan bien se “alinean” los términos $y^{(t)}$ y $x^{(i)}$, y son calculados por una *función de alineamiento* que denota que tan importante es el estado oculto del *codificador* $h^{(t)}$ para el estado oculto del decodificador $s^{(i)}$.

$$\alpha_{t,i} = align(y^{(t)}, x^{(i)}) = \frac{\exp(score(s^{(t-1)}, h^{(i)}))}{\sum_{k=1}^{T_x} \exp(score(s^{(t-1)}, h^{(k)}))} \quad (2.20)$$

Bahdanau propone aprender esta alineación usando una *Red feed-forward* con una sola capa oculta y la función \tanh como activación:

$$score(s^{(t)}, h^{(i)}) = v_a^\top \tanh(W_a[s^{(t)}; h^{(i)}]) \quad (2.21)$$

con v_a y W_a como matrices de pesos a aprender durante el entrenamiento, $[s^{(t)}; h^{(i)}]$ representa una concatenación de los estados ocultos del *codificador* y decodificador. En la figura 2.8 podemos ver gráficamente el modelo usado por *Bahdanau*.

Los modelos de atención pueden ser vistos de manera más general como un mapeo de una secuencia de llaves k hacia una distribución de atención α de acuerdo a una consulta q aplicándose a un conjunto de valores V para selectivamente propagar la información contenida en V . Si bien, los términos de consulta, llaves y valores (query, keys, values) son en

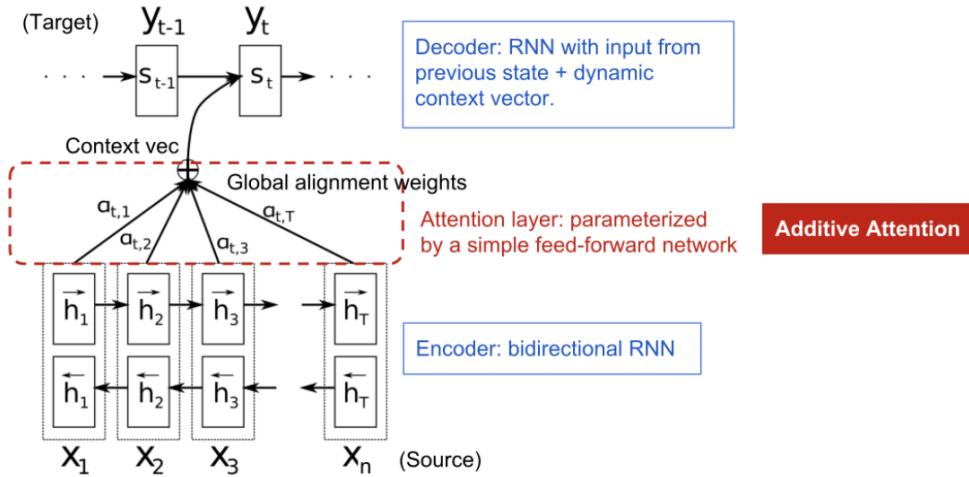


Figura 2.8: Modelo seq2seq propuesto por Bahdanau et al. con *Additive/Concat Attention*

ámbitos de los *Sistemas de Recuperación de Información* su relación en términos de la atención aplicada por Bahdanau es muy similar; las llaves son los estados ocultos del *codificador* y la consulta es el estado oculto del decodificador en cuestión, en este caso el mapeo de entre llaves y valores es la misma:

$$A(q, K, V) = \sum_i p(a(K - i, q)) * v_i \quad (2.22)$$

En la ecuación 2.22, p es una función de distribución que mapea los puntajes de la función de alineación a a pesos de atención. Comúnmente se usan las funciones *softmax* o *logistic sigmoid* puesto que nos aseguran que los pesos de atención producidos estarán dentro del rango $[0, 1]$ y la suma de ellos es igual a 1, por lo que los pesos pueden ser interpretados como una probabilidad que indica que tan relevante es cierto elemento. Algunas variaciones en donde se consideran solo los términos relevantes como *sparsemax* Martins and Astudillo o *sparse entmax* Martins et al. permiten trabajar y enfocarse en solo algunas relaciones de alineamiento. Tay et al. proponen una función de distribución de pesos $M = \tanh(E) \odot \text{sigmoid}(N)$ con E como una matriz en donde cada entrada representa la similaridad entre estados ocultos y N una medida negativa (disimilaridad), por lo que podemos usar $\text{sigmoid}(N)$ como información para “de-atender” los alineamientos de E .

Las funciones de alineamiento se encargan de comparar y extraer la relación entre las representaciones de las llaves (keys) y consultas (queries), por ejemplo usando el producto punto y el coseno como función de similaridad. Bahdanau calcula esta relación a través de una red neuronal 2.20, lo que evita asumir que ambas representaciones están en el mismo espacio, como lo hace las funciones de alineamiento como el producto punto o la similaridad coseno. La tabla 2.1 muestra una recopilación de funciones de alineamiento.

Nombre	Función de Alineación	Cita
Similarity / Content-Base	$a(k_i, q) = sim(k_i, q)$	Graves et al.
Dot Product ¹	$a(k_i, q) = q^\top k_i$	Luong et al.
Scaled Dot Product	$a(k_i, q) = \frac{q^\top k_i}{\sqrt{d_k}}$	Vaswani et al.
General	$a(k_i, q) = q^\top W k_i$	Luong et al.
Biased General	$a(k_i, q) = k_i(Wq + b)$	Sordoni et al.
Activated General	$a(k_i, q) = act(q^\top W k_i + b)$	Ma et al.
Generalized Kernel	$a(k_i, q) = \phi(q)^\top \phi(k_i)$	Choromanski et al.
Additive\Concat ²	$a(k_i, q) = v^\top act(W[q; k_i] + b)$	Bahdanau et al., Luong et al.
Deep	$\begin{aligned} a(k_i, q) &= v^\top E^{(L-1)} + b^L \\ E(l) &= act(W_l E^{(l-1)} + b^l) \\ E(1) &= act(W_0 k_i + W_1 q + b^l) \end{aligned}$	Pavlopoulos et al.
Location-based	$a(k_i, q) = act(Wq)$	Luong et al.
Feature-based	$a(k_i, q) = v^\top act(W_0 \phi(K) + W_1 \phi(K) + b)$	Li et al.

Cuadro 2.1: Distintos tipos de funciones de alineación. (Tabla basada en [9] y [99]).

$a(k_i, q)$ representa la función de alineación entre k_i y q y act es una función de activación. sim es una función de similaridad, Graves et al. propone la función coseno.

Los parámetros v, W, W_1, W_2 son parámetros aprendidos por la red neuronal.

¹ El factor de escala $\frac{1}{\sqrt{d_k}}$ ayuda a estabilizar cuando el gradiente es muy pequeño. d_k es el tamaño de la cabeza de atención.

² La función de activación propuesta por Bahdanau et al. es la función $tanh$ como se ve en 2.21

De acuerdo a como es aplicado los distintos tipos de atención Chaudhari et al. los dividen en 4 grandes grupos; por número de secuencias, por nivel de abstracción, por número de posiciones y por número de representaciones. Estos grupos no son mutualmente excluyentes por tanto una aplicación de atención puede pertenecer a más de una.

En la categoría *por número de secuencias* se identifican 3 tipos, el primero de ellos, **Distintivos** (*Distinctive*) es cuando la clave (key) y valor (value) pertenecen a distintas secuencias de entrada y salida respectivamente, como es el caso del modelo propuesto por

Bahdanau et al.. El segundo tipo, **Co-Atención** (*co-attention*) utiliza distintos secuencias al mismo tiempo para conocer los pesos de atención entre estas entradas. Por ejemplo, en tareas en donde se necesita trabajar con datos multi-modales como procesar imágenes y texto simultáneamente. En tareas como *Visual Question Answering* se puede aplicar un mecanismo de atención conjunto tanto para las imágenes y el texto para identificar las regiones de la imagen y las palabras del texto que son más relevantes. El tercer tipo es **auto-atención** (*Self Attention*), fue propuesto por Yang et al. y es uno de los puntos claves para los modelos *Transformers* [90]. Es comúnmente usada en tareas que solo requieren una salida resumen y no una secuencia como *Clasificación de texto*. La clave (key) y valor (value) pasan a ser las mismas y la atención es calculada sobre los mismos elementos pertenecientes a la secuencia de entrada, buscando así, encontrar las relaciones entre las palabras de la misma oración.

La segunda Categoría agrupa la atención por el nivel de abstracción en la que es aplicada, a un **solo nivel** o en **múltiples niveles**. La información a procesar muchas veces puede ser representada en distintos niveles de abstracción, es decir, en texto, podemos separar los datos a nivel de letras, n-gramas, palabras, oraciones, párrafos, etc., por tanto, es posible atender de manera jerárquica a las palabras que forman una oración para posteriormente prestar atención a las oraciones que conforman un texto más largo. Yang et al. utiliza este procedimiento para generar un vector de características usado posteriormente en un etapa de clasificación.

En la tercera categoría la atención es realizada en diversas partes de la secuencia; la suma pesada sobre todos los puntajes de las entradas usada por Bahdanau et al. se le denomina **Atención suave** (*Soft-Attention*). Una alternativa es la **Atención dura** (*Hard-Attention*) [105] que calcula la atención no sobre todas los puntajes de alineamiento sino en una parte de estos, para ello se usa una distribución multinomial parametrizada por los pesos de la atención. A pesar de que es más eficiente que la *atención suave* resulta difícil de entrenar al no ser completamente diferenciable. Otra opción a la *atención dura* es la **Atención Local** (*Local Attention*) cuya idea es aplicar atención sobre una ventana elegida ya sea centrada con respecto a la tiempo actual (alineamiento monotónico) o predicha por una función (alineamiento predictivo). La *atención local* fue propuesta por Luong et al. así como la *Atención Global* la cual es similar a la *atención suave*.

La última categoría divide los modelos de atención por las formas de representación de las entradas sobre las que la atención es aplicada. Distintos modelos pueden beneficiarse de procesar los datos creando vectores de características distintos, cada uno de ellos deriva de algún tipo de representación de la entrada. por tanto, es posible atender a diferentes representaciones y formar un vector final usando una combinación pesada de estos a través de dichos pesos de atención. Chaudhari et al. llama a este tipo de modelos de atención como **multi-representational AM**. En la segunda categoría, **milti-dimensional attention**, la atención no es aplicada sobre los diversos vectores de características sino a un nivel más interno, sobre sus dimensiones. Pesando cada característica de un vector de características permite seleccionar aquellas que mejor lo describen para un contexto dado. EN *NLP*, resulta

bastante útil cuando se trata con *polisemia*, en donde una palabra o frase puede tener más de un significado.

2.4. El modelo Transformer

A finales del año 2017 se presentó un nuevo modelo que vino a revolucionar el área de Procesamiento de Lenguaje Natural, El Transformer [91]. Una de sus principales características es la capacidad de procesar la información de alguna secuencia de forma paralela, caso contrario a las Redes Neuronales Recurrentes, donde la información se procesa recurrentemente. Gracias a ello, la capacidad de *recuerdo* no se ve afectado por el problema de *El desvanecimiento del Gradiente* específicamente cuando el problema es trabajar con secuencias bastante largas.

El Transformer puede ser visto como otro modelo *seq2seq* (Secuencia a Secuencia) 2.9, formado en por dos etapas, la primera encargada de codificar la información de entrada y la segunda de decodificarla, pero su principal característica es que aplica mecanismos de *Self-Attention* para capturar las dependencias globales entre la entrada y la salida. Dada una secuencia de entrada $X = (x_1, x_2, \dots, x_n)$ con n como el tamaño de la secuencia, el codificador produce una representación intermedia $Z = (z_1, z_2, \dots, z_n)$ al igual que los modelos *seq2seq*. El decodificador usa la secuencia Z para generar la secuencia de salida $Y = (y_1, y_2, \dots, y_m)$ uno a la vez (en modo inferencia), con m como el tamaño de la secuencia de salida. Nótese, que el generar una salida a la vez el decodificador tiene que ser auto-regresivo. Usa la salida anterior y_{i-1} como entrada adicional para generar la siguiente salida y_i . Por ello, durante entrenamiento el modelo es alimentado con entradas y salidas desfasadas en un tiempo.

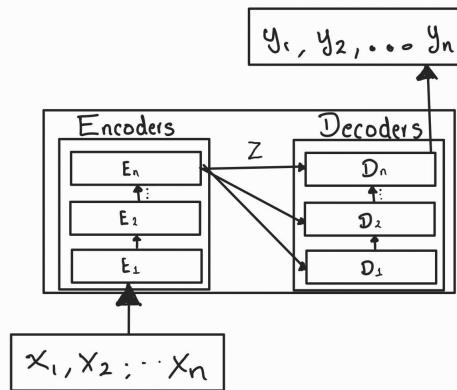


Figura 2.9: Modelo Transformer generalizado como modelo Secuencia a Secuencia

2.4.1. El Codificador y Decodificador

El *Modelo Transformer* está formado por multiples codificadores y decodificadores apilados e inter-conectados, Como observamos en la figura 2.9. El codificador consta de dos capas, la primera de ellas aplica *Self-Attention* múltiples veces sobre la misma entrada (*Multi-Head Self Attention*) y la segunda capa representada solo por una red *Feed-Forward* cuya entrada es la salida de la capa anterior. Véase la figura 2.10.

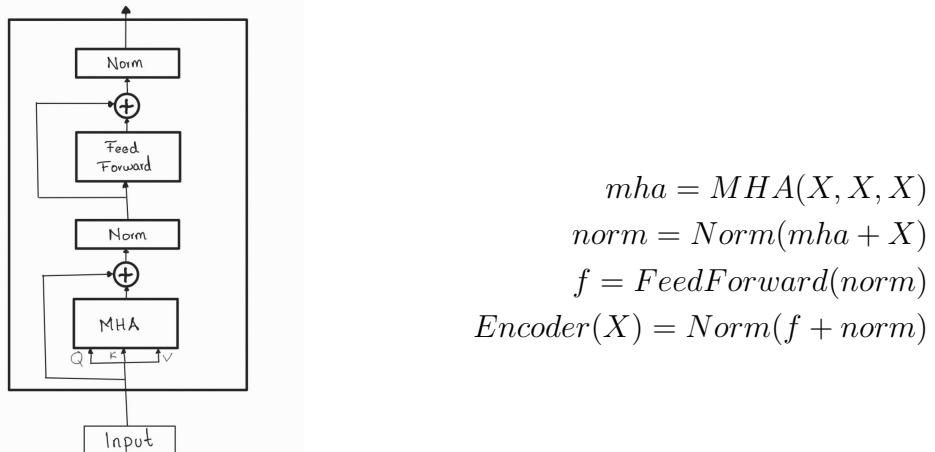


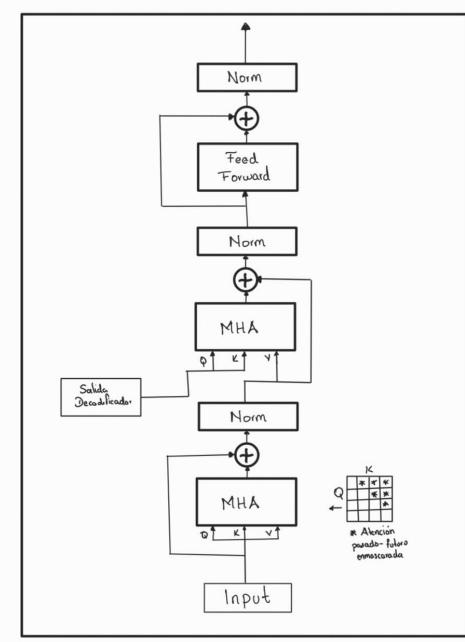
Figura 2.10: Etapa Codificadora del Modelo Transformer. Pseudocódigo

El decodificador tiene una estructura similar al codificador con una etapa adicional intermedia de *Multi-Head Attention* aplicada sobre la salida de la pila de codificadores. También, la primer capa de atención sufre un ligero cambio un su forma de operación, necesitando enmáscarar (al momento en que se realiza el entrenamiento) la atención prestada del pasado al futuro. Esto es debido a que el decodificador se encarga de generar una secuencia (en modo inferencia) uno a la vez usando solamente la salida anterior y por tanto no tiene conocimiento de salidas futuras, observe la figura 2.12.

2.4.2. Multi-Head Self-Attention

En la sección 2.3.3 se detalla una generalización de la atención y diversas variantes usadas a lo largo de la literatura. El modelo original que introdujo a los Transformers usa en especial la variante *Scaled Dot-Product Attention*[91]:

$$\text{Attention}(q, k, v) = \text{softmax}\left(\frac{qk^\top}{\sqrt{d_k}}\right)v \quad (2.23)$$



$$\begin{aligned}
 mha_1 &= MHA(X, X, X) \\
 norm_1 &= Norm(mha_1 + X) \\
 mha_2 &= MHA(enc_{out}, enc_{out}, norm_1) \\
 norm_2 &= Norm(mha_2 + X) \\
 f &= FeedForward(norm_2) \\
 decoder(X) &= Norm(f + norm_2)
 \end{aligned}$$

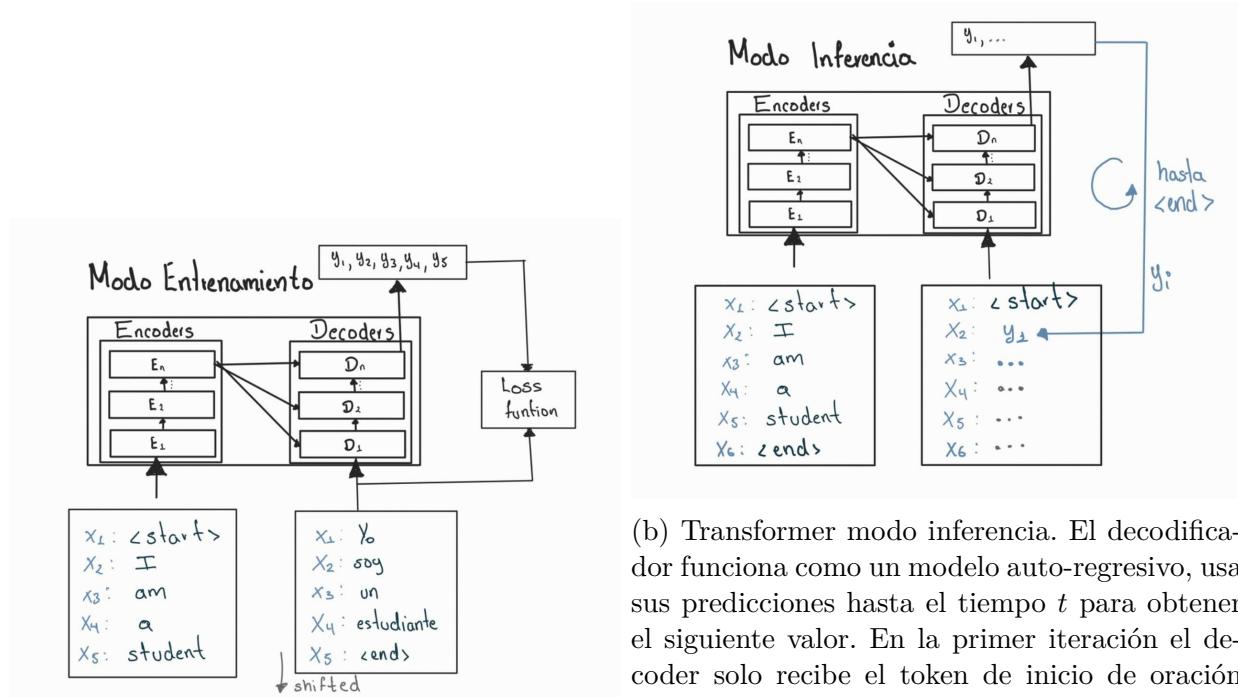
Figura 2.11: Etapa Decodificadora del Modelo Transformer. Pseudocódigo

El Transformer está basado en la idea de aplicar atención múltiples veces, al usar varias cabezas de atención, Multihead-Self-Attention (MHA), permite al modelo conjuntamente atender a información en distintas posiciones desde h diferentes subespacios de representación. 2.24

$$mha(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \text{head}_3, \dots, \text{head}_h)W^O \quad (2.24)$$

Todas las cabezas de atención son concatenadas y resumidas para ser devueltas a las dimensiones del espacio de entrada original, principalmente para mantener consistencia en las dimensiones usadas en cada etapa de codificación y decodificación del modelo a través de $W^O \in \mathbb{R}^{hd_v \times d_m}$. W^O es entrenado conjuntamente para aprender a resumir la información capturada por cada cabeza de atención. $Q, K \in \mathbb{R}^{n \times d_m}$ y $V \in \mathbb{R}^{n \times d_v}$ es la representación consulta, clave y valor de los embeddings de entrada de cada capa de atención del codificador y decodificador como se observa en las figuras 2.10 2.11. n es el tamaño de la secuencia, d_m y d_v son los tamaño del embedding y h el número de cabezas de atención.

En el caso del modelo transformer tenemos un conjunto embeddings sobre las cuales se aplica atención, si bien, no representan necesariamente las consultas, llaves, y valores utilizados para la atención generalizada, podemos obtener estas representaciones transportándolos a sus espacios respectivos a través de alguna transformación aprendida conjuntamente con el entrenamiento del modelo.



- (a) Transformer modo entrenamiento. Las entradas en el decodificador son recorridas un elemento a la vez en el futuro, con el fin de que aprende a predecir la siguiente palabra dado un contexto previo agregado como entrada al decodificador. El decodificador termina su predicción en el momento que el token $<end>$ es obtenido.
- (b) Transformer modo inferencia. El decodificador funciona como un modelo auto-regresivo, usa sus predicciones hasta el tiempo t para obtener el siguiente valor. En la primera iteración el decodificador solo recibe el token de inicio de oración $<start>$ por lo que podrá predecir la primera palabra de la oración gracias a que fue entrenada en el futuro, con el fin de que aprende a predecir la siguiente palabra dado un contexto previo agregado como entrada al decodificador. El decodificador termina su predicción en el momento que el token $<end>$ es obtenido.

Figura 2.12: Esquema de entrenamiento e inferencia del modelo Transformer en un problema de Machine Translation.

Por tanto, para el conjunto de Embeddings $E_Q \in \mathbb{R}^{n \times d_m}$, $E_K \in \mathbb{R}^{n \times d_m}$ y $E_V \in \mathbb{R}^{n \times d_v}$ donde n es el número embeddings, d_m y d_v son las dimensiones de cada uno, la atención en cada cabeza i se calcula como:

$$\begin{aligned} Q_i &= E_Q W_i^Q \\ K_i &= E_K W_i^K \\ V_i &= E_V W_i^V \end{aligned} \tag{2.25}$$

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i \tag{2.26}$$

donde $W_i^Q, W_i^K \in \mathbb{R}^{d_m \times d_k}$, $W_i^V \in \mathbb{R}^{d_m \times d_v}$ y $d_k = d_v = d_m/h$.

el término de escalamiento $\sqrt{d_k}$ ayuda a evitar que la magnitud de los productos puntos calculados entre cada consulta y llave crezcan demasiado, y que la función *softmax* pueda ser más estable al evitar regiones donde los gradientes son muy pequeños[91].

2.4.3. Información Posicional

En los modelos basados en *Redes Recurrentes* la información se procesan uno a uno en cada paso de tiempo. Los modelos basados en *Transformers* procesan la información en conjunto, perdiendo la noción de la temporalidad de los datos. Una solución es agregar dicha información perdida a través de vectores que codifiquen el tiempo/posición de los datos sumándolos con los vectores de embeddings. Estos vectores llamados *Positional Encodings* [24] siguen un patrón en específico que el modelo aprende a identificar y lo ayuda a determinar la posición de cada elemento de la secuencia y por tanto calcular a qué distancia se encuentra cada uno de los demás.

Por lo regular se usa una onda senoidal y cosenoidal para lugares pares e impares, formando una progresión geométrica desde 2π hasta $10000 \cdot 2\pi$ 2.27:

$$\begin{aligned} PE(pos, 2i) &= \sin(pos/10000^{2i/d_m}) \\ PE(pos, 2i + 1) &= \cos(pos/10000^{2i/d_m}) \end{aligned} \quad (2.27)$$

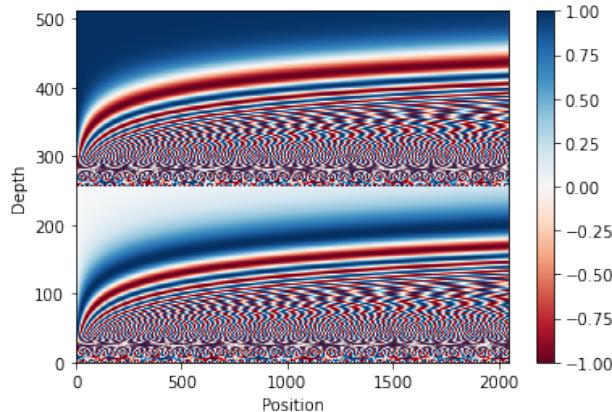


Figura 2.13: 2000 Vectores de Positional Encoding con dimensiones de embedding=500.

Poner una figura completa de todo el esquema del Transformer

2.4.4. Problemas típicos en el entrenamiento de Transformers

Learning Rate WarmUp y Layer Normalization

A pesar de que la arquitectura del modelo Transformer no es tan compleja, puesto que tanto el codificador como el decodificador están formados por pilas de capas de atención y MLP, el entrenamiento de este tipo de modelos muchas veces no resulta tan trivial. Regularmente requiere de una combinación de técnicas para lograr su convergencia a valores aceptables y en conjunto con una gran cantidad de datos, tamaños de lotes de procesamiento grandes y una gran cantidad de tiempo de procesamiento en gpu [66].

Learning Rate WarmUp es una de las primeras técnicas usadas y descritas en el proceso de entrenamiento por Vaswani et al.. Usando el algoritmo Adam como optimizador se varía el factor de aprendizaje de acuerdo a la fórmula:

$$lrate = d_m^{-0,5} \cdot \min \left(step_num^{-0,5}, step_num \cdot warmup_steps^{-0,5} \right) \quad (2.28)$$

En el esquema anterior, más conocido como *Noam-Warmup*, el modelo original es entrena-
do incrementando linealmente el factor de aprendizaje en los primeros $warmup_steps = 4000$ pasos. Posteriormente, decrementa proporcionalmente al inverso del la raíz cuadrada del paso $step_num$ actual, véase la figura 2.14.

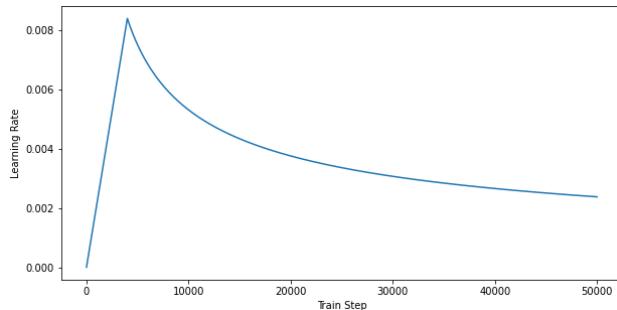


Figura 2.14: Noam-Warmup con $warmup_steps = 4000$ y $d_m = 512$

Si bien, la razón por la que funciona este tipo de técnica no está del todo claro, se presume que usar *Learning Rate WarmUp* ayuda a reducir la varianza del factor de aprendizaje adaptativo durante las primeras etapas del entrenamiento del modelo. Liu et al. demostraron que el segundo momento del algoritmo de Adam durante etapas tempranas de optimización es proporcional a una integral divergente, lo que provoca las actualizaciones inestables, llevando al modelo fuera de las regiones donde un mejor mínimo existe. Con esto en mente Liu et al. proponen el algoritmo de optimización *RAdam* (Rectified Adam) como una alternativa a usar *Learning Rate WarmUp* y mitigar este efecto durante la fase inicial del entrenamiento de los modelos.

El *Learning Rate WarmUp* comúnmente es usado en conjunto con algoritmos de optimización estocásticos como *RMSprop* o *Adam*. En vez configurar el *factor de aprendizaje* α con un decremento constante, la estrategia de *Learning Rate WarmUp* configura este factor con valores muy pequeños en los primeros pasos de entrenamiento. Durante las primeras etapas del entrenamiento el factor de aprendizaje es incrementado hasta un límite que es ligeramente superior o inferior al valor inicial de α del optimizador usado y posteriormente es decrementado progresivamente hasta la convergencia del modelo.

Así, en cada paso del algoritmo de optimización el cuál está parametrizado por el factor de aprendizaje α , puede ser aplicado un factor de *warmup* $\omega \in [0, 1]$ que sirve para reducir α y a la vez el paso de optimización en cada tiempo, reemplazando $\alpha_t = \alpha\omega_t$. La forma mas sencilla es usar un factor **linear warmup** parametrizado por un periodo de “calentamiento” τ .

$$\omega_t^{linear,\tau} = \min\left(1, \frac{t}{\tau}\right) \quad (2.29)$$

Ma and Yarats proponen 3 formas de aplicar la técnica de *warmup*:

Exponential warmup aplica un decaimiento exponencial

$$\omega_t^{expo,\tau} = 1 - \exp\left(-\frac{1}{\tau}t\right) \quad (2.30)$$

recomienda elegir $\tau = (1 - \beta_2)^{-1}$ tal que no se tan diferente del segundo momento de corrección de bias del algoritmo de *Adam* β_2 .

$$\omega_t^{expo,untuned} = 1 - \exp(-(1 - \beta_2)t) \quad (2.31)$$

Similar al decaimiento exponencial proponen usar *linear warmup* sobre $\tau = 2(1 - \beta_2)^{-1}$ iteraciones para preservar un efecto similar de des-aceleración con el paso del tiempo.

$$\omega_t^{linear,untuned} = \min\left(1, \frac{1 - \beta_2}{2}t\right) \quad (2.32)$$

Por otro lado, Huang et al. mencionan que usar la técnica de *Learning Rate WarmUp* para mitigar la varianza del optimizador Adam no es del todo la solución y que el problema radica precisamente en la arquitectura del modelo Transformer, principalmente en las capas de normalización [12] [104]. En particular Xiong et al. encuentran que para un modelo Transformer de cualquier tamaño con capas de normalización entre bloques residuales (*Post-LN Transformer*), la escala de la norma del gradiente que incide en la última capa

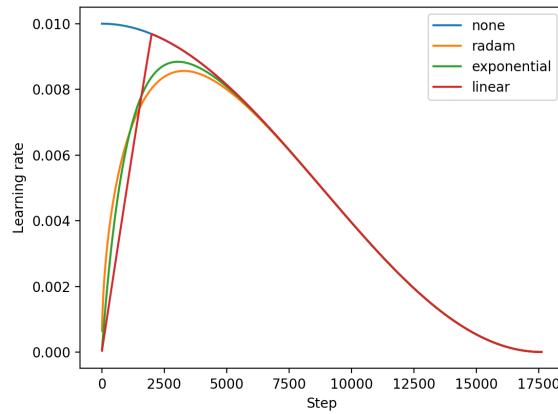


Figura 2.15: Learning rate sobre X 18000 iteraciones usando RAdam y lineal, exponencial warmup con Adam

de normalización permanecen igual al no depender de la cantidad de bloques del transformer. Por el contrario, si la capa de normalización es colocada justo antes de la conexión residual (*Pre-LN Transformer*) la magnitud de la norma del gradiente decrece conforme el tamaño del modelo incrementa, guiando así, al problema de desvanecimiento de gradiente. Huang et al. proponen eliminar las capas de normalización del modelo Transformer que en conjunto con la inestabilidad del algoritmo de optimización de Adam provocan la dificultad de entrenamiento durante desde las primeras etapas. Para ello, estandarizan la siguiente inicialización (*T-Fixup*) de pesos del modelo, permitiendo evitar la etapa de *WarmUp* y las capas de normalización en el Transformer. La figura 2.16 muestra una comparativa de los histogramas usando la inicialización *T-Fixup* e usar el algoritmo de Adam con y sin etapa de *Warmup*:

- Aplicar initialization tipo *Xavier* para todos los pesos del modelo. Excepto el proceso de generación de embedding adecuados al tamaño del modelo d_m .
- Usar una inicialización tipo *Gaussiana* con $\mathcal{N}(0, d_m^{\frac{1}{2}})$ para los pesos de generación de embeddings.
- Escalar las matrices W_i^V y W^O en cada bloque de atención en el decodificador, los pesos en de cada capa MLP del decodificador y los pesos de generación de embeddings tanto del codificador como decodificador por $9N^{-\frac{1}{4}}$ donde N es el número de bloques del Transformer.
- Escalar las matrices W_i^V y W^O de cada bloque de atención del codificador y los pesos de cada capa de MLP del codificador por $0,67N^{-\frac{1}{4}}$

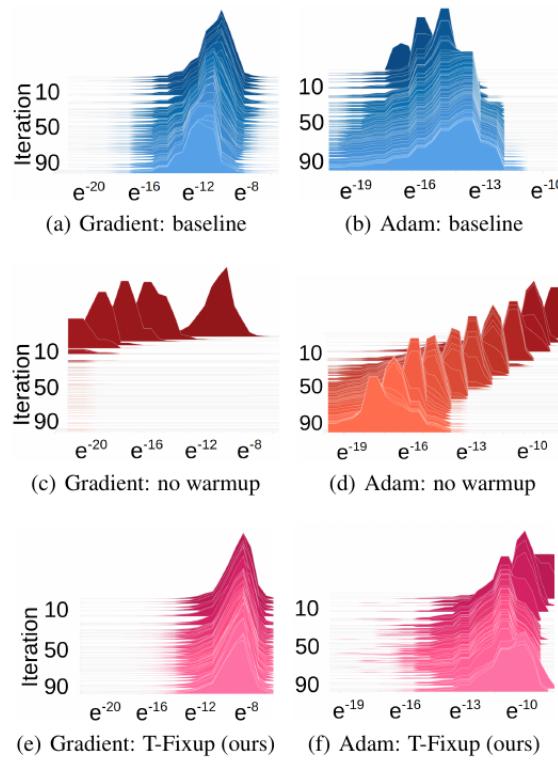


Figura 2.16: Histograma de gradientes del Algoritmo Adam con y sin etapa de *WarmUp* y usando inicialización *T-Fixup*. Imagen original de Huang et al.

Cálculo de la Atención

Además de lo específico y delicado del entrenamiento del Modelo Transformer su costo en tiempo computacional y de memoria también representa un serio problema a la hora de optimizar e inferir. Debido principalmente a que en el proceso de atención debe focalizar cada token con respecto a todos los demás, lo que lleva a que su complejidad crezca cuadráticamente con respecto a el tamaño de la secuencia.

Varías técnicas han sido propuestas para reducir este problema, muchas de ellas involucran en reducir la atención a vecindades de representaciones, aproximar la matriz de atención con otras matrices de transformaciones a través de kernels o sustituir completamente la operación *softmax* por otra función.

Atención de vecindades:

Parmar et al. partitionan la información de las representaciones de las consultas asignándolas a diferentes bloques de memoria, restringiéndose a vecindarios locales alrededor de cada consulta. Principalmente basados en cómo las redes convolucionales trabajar. Sin embargo

esta solución es parcial y solo aplicable a secuencias de datos con relaciones cortas, como imágenes.

Child et al. factorizan la matriz de atención para reducir su complejidad de $O(n^2)$ a $O(n\sqrt{n})$ por medio de matrices ralas, separando la atención a través de diferentes pasos al parametrizar la atención por una conectividad de distintos patrones elegidos previamente bajo el supuesto de que las matrices de atención son ralas, puesto que no contienen dependencias de relevancia sobre representaciones distantes como se observa en la imagen 2.17.

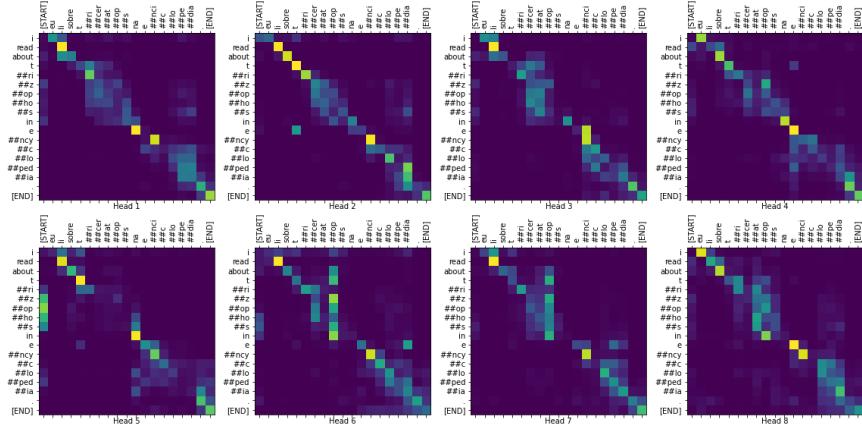


Figura 2.17: Visualización de 8 cabezas de atención sobre una tarea de Machine-Translation. Las matrices de atención tienden a ser ralas al tener carencia de relaciones relevantes entre diversas representaciones a distancias lejanas.

Sukhbaatar et al. proponen reducir el ancho de la atención basándose en que cada representación no necesita prestar atención sobre todas las demás sino que debería ser adaptativo. Así, para cada cabeza de atención se agrega una función de enmascaramiento que controla la flexibilidad del ancho una ventana. La ventana formada cambia dinámicamente de tamaño dependiendo de la representación en cuestión. Beltagy et al. siguen un estrategia similar, implementando atención local con ventanas dilatadas distintas para cada cabeza, permitiendo atender contextos menos locales en cada ocasión y atención global sobre preseleccionados localizaciones. Dada la dificultad de su implementación sin usar ciclos para iterar sobre los elementos seleccionados a atender, implementan su propio kernel en *CUDA* con las operaciones optimizadas para realizar esta tarea.

Dai et al. mencionan que si el problema es el procesamiento de grandes secuencias por qué no dividirlas en secuencias más pequeñas y procesarlas individualmente y así evitar usar grandes cantidades de memoria en su procesamiento. El principal problema de este enfoque es que cada secuencia es procesada individualmente y la información de previas secuencias es ignorada evitando que esta fluya a través de las próximas secuencias. Para solucionar este inconveniente introducen un mecanismo de recurrencia en la arquitectura del transformer. Durante el entrenamiento (véase la figura 2.18) un estado oculto es calculado de las secuen-

cias previas y guardado en memoria para extender el contexto al momento de procesar la siguiente secuencia. Durante el proceso de evaluación, el resultado de las operaciones del transformer pueden ser reusado y no calculado nuevamente desde cero, permitiendo reducir considerablemente el tiempo de evaluación.

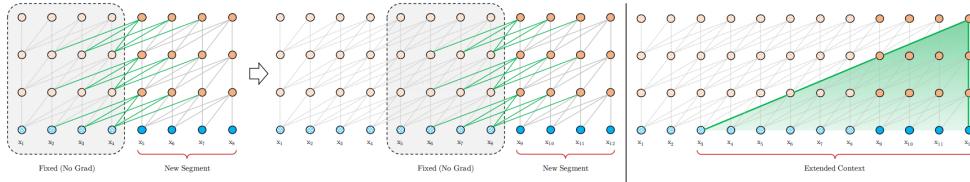


Figura 2.18: Transformer-XL. Para tratar con secuencias largas divide el proceso en secuencias más cortas creando estados ocultos intermedios y usandolos en el cálculo de las próximas secuencias. Figura obtenida de [19].

Kitaev et al. reducen el problema de realizar la operación de softmax sobre toda la matriz $Q_i K_i^\top$ a calcularlo individualmente por cada consulta q_j , guardando solo una vez en memoria este valor en cada iteración y recalculándolo cuando se necesite de nuevo al utilizar *Back-Propagation* usando de capas reversibles. Si bien, computacionalmente es más costoso permite usar mucho menos memoria que la solución original. Por otro lado, dado que el resultado de la función softmax depende en mucho mayor medida en los elementos dominantes de la matriz, solo es necesario fijarse en las llaves más cercanas a la consulta en cuestión. *LSH (Local Sensitive Hashing)* resuelve este problema permitiendo encontrar rápidamente los vecinos más cercanos en espacios de altas dimensiones, con la restricción de que $W_i^Q = W_i^K$ dado que se necesita conservar la similaridad entre consultas y llaves, algo que sería más difícil si sus matrices de proyección W_i^Q y W_i^K fuesen muy distintas.

Aproximaciones a la Atención original:

También Xiao et al. proponen un modelo para compartir pesos de capas adyacentes (Shared Attention Network - SAN). Cada π capas continuas en el codificador comparten la misma matriz de atención y en el decodificador se comparte la proyección de los pesos de atención sobre la representación de los valores V_i , en otras palabras se comparte directamente la cabeza de atención $head_i$. Dado que no es tan fácil conocer que capas deben compartir pesos, establecen un proceso iterativo de entrenamiento basados en calcular que tan diferentes son las capas del transformer usando la *Divergencia de Jensen-Shannon*. Si la similaridad entre dos capas es mayor a cierto umbral se indica que dichas capas deben compartir pesos. Se repite un nuevo entrenamiento y se calcula nuevamente la similaridad entre capas, y así sucesivamente hasta convergencia. Podemos notar que este proceso de entrenamiento y ajuste de pesos es muy costoso, un nuevo entrenamiento es requerido por cada ajuste de compartición de pesos, pero el modelo resultante es menos complejo y el tiempo en modo de evaluación o inferencia se ve reducido considerablemente.

Wang et al. bajo la hipótesis de que la matriz de atención tienen rango mucho menor

que n proponen obtener los valores de cada cabeza haciendo una aproximación a ella. Para ello, se hace uso de dos matrices entrenables conjuntamente con el modelo, E y $F \in \mathbb{R}^{n \times k}$ con $k \ll n$, tal que, $head_i = softmax(\frac{Q_i(E_i K_i)}{\sqrt{d_m}}) F_i V_i$. Con ello la dimensión correspondiente al tamaño de las secuencias se ve reducido bajo el supuesto que podemos representar la información secuencial en un espacio más pequeño sin gran pérdida de información.

Choromanski et al. por el contrario descomponen la operación de atención sobre los valores $head = softmax(\frac{QK^\top}{\sqrt{d_k}V})$ en una multiplicación matricial más simple $head = Q'k'^\top V$ con Q' y $k'^\top \in \mathbb{R}^{n \times r}$ y $r \leq n$. Para ello construyen Q' y k' como dos matrices usando kernels tal que su producto forma una aproximación a la función softmax aplicada al producto de Q y K . Notemos que con ello podemos reducir el costo computacional y en memoria simplemente reduciendo el producto $Q'(k'^\top V)$ de derecha a izquierda.

Finalmente autores como Lee-Thorp et al. remplazan completamente el bloque de *Multihead Attention* con bloques que aplican operaciones de Transformada de Fourier o lo largo de la dimensión de los embeddings y de las secuencias. Probando que el usar *FFT* (Fast Fourier Transform) es suficiente para abstraer y modelar las relaciones. Y como Wu et al. que cambian la atención entre todas las consultas y llaves por una sola con todas las llaves. Para esto, a través de atención sumarizan todos las consultas en una consulta global. Este proceso es repetido con las llaves y valores como se observa en la figura 2.19

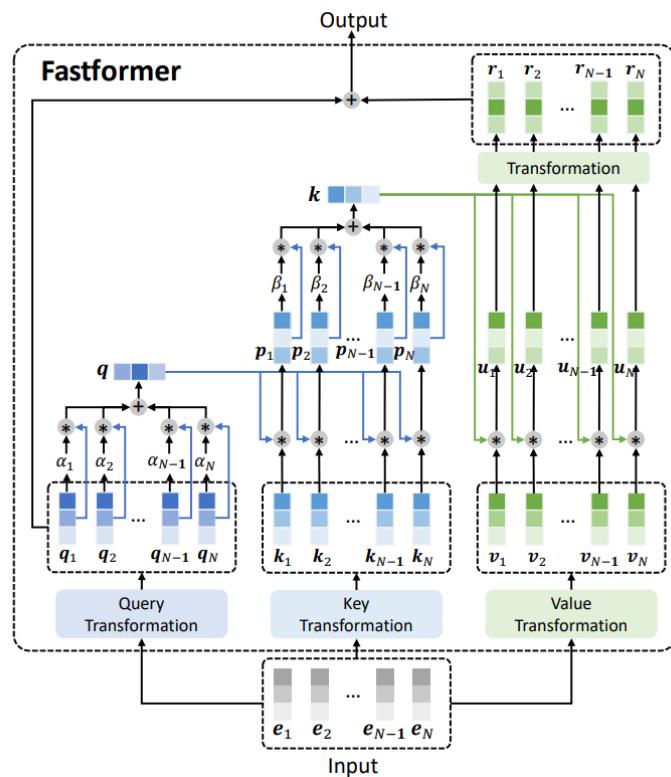


Figura 2.19: Fast-Former. Reemplaza la atención tradicional del transformer por una iterativa. En cada paso crea una consulta y clave global usando atención sobre estos mismos. Figura obtenida de [101]

Capítulo 3

Vision Transformers en Detección de Patologías en Pulmones con Rayos X

3.1. Detección de Patologías en Pulmones con Rayos X

Diferentes trabajos se han limitado a clasificar *COVID-2019* y *no COVID-2019* (o neumonía), es por ello que durante este trabajo desarrollamos un sistemas más completo para distinguir 15 diferentes enfermedades pulmonares.

En este trabajo se desarrolla un método basado en aprendizaje profundo usando una red *ResNet50* como *backbone*, una red ampliamente usada por la comunidad para implementar diferentes métodos de análisis de imágenes. Se tomó como base el dataset *ChestX-Ray14* y se realizó una extensión incluyendo radiografías de pacientes infectados y no infectados por *COVID-19*. El entrenamiento usando el este nuevo dataset, consiste en 3 etapas; la primera basada en *data transfer learning* con la red *ResNet50* pre-entrenada con el dataset de ImageNet y remplazando la capa de clasificación por un nuevo clasificador para las 15 patologías, una etapa de *fine-tuning* para ajustar las últimas capas convolucionales y una última etapa *full-tuning* para ajustar los pesos enteramente de la red. Los experimentos completados muestran que esta propuesta es capaz de preservar el rendimiento de los clasificadores entrenados para las 14 patologías previas del estado del arte, y al mismo tiempo, alcanzar el rendimiento de los clasificadores binarios para *COVID-19* mostrados en la literatura actual. Adicionalmente, se demuestra que el modelo y entrenamiento propuesto puede ser usado para fácilmente realizar una extension hacia alguna otra patología, con la implementación de un clasificador binario para la detección de Tuberculosis, una particular neumonía bacterial [83], con rendimiento comparable con diversos métodos de deep learning *ad hoc* [67].

CAPÍTULO 3. VISION TRANSFORMERS EN DETECCIÓN DE PATOLOGÍAS EN PULMONES CON

La evaluación del modelo propuesto para el diagnóstico de las 15 patologías demuestra la obtención de resultados competitivos, en particular para *COVID-19*. Se realiza la comparación del modelo con trabajos recientes que clasifican las primeras 14 patologías (excluyendo *COVID-19*). Los resultados en términos del *Área Bajo la Curva del Operador del Receptor de Características - AUC* muestra un rendimiento competitivo contra el método propuesto para detección de patologías *ChestX-Ray14* y un notable rendimiento en la detección de neumonía para *COVID-19*.

En la ejecución de este trabajo se usa de la base de datos *ChestX* [98] y se incluye imágenes de Rayos-X para COVID-19 y *saludables* (imágenes de personas sin alguna enfermedad de pulmón identificada). El dataset *ChestX-ray14* contiene 112,120 radiografías de tórax de vista frontal (frontal-view) *AP* y *PA* (anterior-posterior y posterior-anterior) de 30,805 pacientes únicos. Cada imagen esta anotada con algunas de las etiquetas correspondientes a las 14 patologías descritas previamente. Este dataset es enriquecido durante el previamiento al proceso de entrenamiento descrito aquí con radiografías de personas con neumonía y *saludables*. La tabla 3.1 resume la composición del dataset completo usado en este trabajo.

Class	Disease	Source	Train	Test
1	Cardiomegaly	1	5897	1069
2	Emphysema	1	3496	1093
3	Effusion	1	13737	4658
4	Hernia	1	4470	86
5	Infiltration	1	17122	6112
6	Mass	1	7862	1748
7	Nodule	1	9250	1623
8	Atelectasis	1	13762	3279
9	Pneumothorax	1	6303	2665
10	Pleural-Thick.	1	8022	1143
11	Pneumonia	1	6031	555
12	Fibrosis	1	9072	435
13	Edema	1	6953	925
14	Consolidation	1	9796	1815
—	Healthy	1	35645	9861
11	Pneumonia	2	5475	594
15	COVID-19	2	2873	1904
—	Healthy	2	8661	1926

Cuadro 3.1: Number of chest radiography's per case in the dataset. Sources (DS): (1) ChestX-ray14, (2) Internet datasets compilation. Pleural-Thick. means Pleural-Thickening.

Data Relabelling

Para lograr un entrenamiento del modelo exitosamente, se usa la versión re-etiquetada de la base de *ChestX-ray14* propuesta por los autores de *CheXNet* [69]. De acuerdo a [69], los datos re-etiquetados permiten mejorar el proceso de entrenamiento significativamente. Cada re-etiquetado es realizado a través del entrenamiento de múltiples modelos de redes neuronales con los datos originales como entrada, los modelos con mejor accuracy en la evaluación del dataset original son conservados. Después, las predicciones individuales de cada modelo son promediadas y un detector binario para cada patología es construido usando un umbral que permite maximizar la métrica F1-score en todas las patologías, (véase la sección ?? para la definición de F1-score). Finalmente, cada dato es re-etiquetado, como positivo si este fue originalmente etiquetado como positivo o el clasificador construido en el paso previo resulta positiva. A pesar de este re-etiquetado, en estos experimentos los datos preservan su estructura de acuerdo a los datos de entrenamiento y validación proporcionados originalmente; para el conjunto de datos originales, solo los datos de entrenamiento re-etiquetados son usados, es decir, solo se usan los datos con etiquetas modificadas en la etapa de entrenamiento, preservando la evaluación intacta al proceso usado por otros los otros métodos. Aún así, se muestra el reporte correspondiente a el conjunto de validación re-etiquetado solo como referencia adjunto en el Apéndice.

Procesamiento de los datos

El procesamiento las imágenes radiográficas sigue un camino simple, se implementa un proceso de equalización por histograma, seguido de un redimensionamiento a 1024×1024 pixeles. Siendo los datos de entrada a el modelo neuronal generado imágenes de 1024 columnas y 1024 filas con 3 canales idénticos. Siguiendo la propuesta de *CheXNet*, se hace uso de multiples funciones de perdida tipo *Binary Cross-Entropy* con distintos pesos para solventar el desbalanceamiento de clases.

$$L(\hat{y}, y) = \sum_{c=0}^{15} \left[w_c^{(p)} y_c \log \hat{y}_c + w_c^{(n)} (1 - y_c) \log (1 - \hat{y}_c) \right], \quad (3.1)$$

donde y es el vector de etiquetas reales, \hat{y} es el vector de etiquetas predichas, $w_c^{(p)}$ y $w_c^{(n)}$ son los pesos para los casos positivos y negativos de cada clase c respectivamente. Dichos pesos son calculados como sigue:

$$w_c^{(n)} = \frac{n_c}{N} \quad \text{and} \quad w_c^{(p)} = 1 - w_c^{(n)}; \quad (3.2)$$

donde N es el tamaño del conjunto de datos, y n_c el número de imágenes radiográficas

con etiquetas c (el número elementos del vector de la c -ésima clase iguales a 1). Nótese que los pesos son inversamente proporcional al número de casos en la clase.

3.2. Arquitecturas usadas

3.2.1. CNN y Transfer Learning

El modelo desarrollado es construido tomando *CheXNet* como referencia [69], el cual clasifica entre 14 enfermedades usando el conjunto de datos *ChestX-Ray14* [98]. El modelo propuesto es construido desde cero y extiende la propuesta de *CheXNet* para incluir la clasificación de imágenes con *COVID-19*.

CheXNet es un modelo basado en redes neuronales para detectar la presencia de 14 diferentes enfermedades de pulmón. *CheXNet* Usas imágenes de Rayos X de vista frontal como entrada y un modelo convolucional pre-entrenado como backbone [69]. El modelo backbone usado por *CheXNet* es la red *DenseNet121* [35] que contiene 121 capas convolucionales entrenadas con la base de datos de *ImageNet* [72]. Las imágenes analizadas por el modelo pueden corresponder a vistas anterior-posterior o posterior-anterior. La implementación de Transferencia de Conocimiento es lograda removiendo la etapa de clasificación (formada por capas densas) mientras que la etapa convolucional previa permanece intacta funcionando como un extractor de características. Posteriormente, una nueva etapa de clasificación construida para la detección de 14 enfermedades es colocada en su lugar. Finalmente, la nueva red compuesta es entrenada usando la base de datos *ChestX-Ray14* manteniendo fijos los pesos correspondientes a la etapa convolucional de la red.

Así como en *CheXNet*, un modelo usado como backbone y entrenado con la base de datos de *ImageNet* define la red entrenada para detectar patologías torácicas. Motivados por Bresssem et al., Shazia et al. que muestran que una red backbone en particular no es un elemento definitivo en el rendimiento de detección del modelo, sino el proceso de entrenamiento. Adicionalmente Huang et al., Luo et al. presentan comparaciones de modelos evaluados en la clasificación de *ImageNet* donde *ResNet50* y *DenseNet201* obtienen un accuracy de 76 % y 74 % respectivamente. Por ello, se realiza la elección del modelo *ResNet50* como backbone en la implementación de la red convolucional. *ResNet50* fue propuesta por He et al. y es una opción popular en la implementación de sistemas de reconocimiento generales dado su eficiencia computacional y relativamente sencillez de entrenamiento (por su grafo de gradientes con poco caminos) en comparación del candidato natural *DenseNet121* usando en *CheXNet*, aque versiones alternativas de *CheXNet* estan disponibles [103]. A pesar de lo anterior, no se descarta el uso e investigación en un futuro de otros modelos como backbone ya sea inclusive *DenseNet* o *efficientNet* [87]. Sabiendo que este mismo procedimiento puede guiarnos a incluir nuevas patologias tales como Tuberculosis Pulmonar [83]; un tipo de neumonia bacterial

CAPÍTULO 3. VISION TRANSFORMERS EN DETECCIÓN DE PATOLOGÍAS EN PULMONES CON AI

mayormente común en países en vías de desarrollo y también frecuentemente reportado en pacientes con síndrome de inmunodeficiencia adquirida (AIDS) [57].

En la red detectora, la etapa clasificadora del modelo *ResNet50* es remplazada con dos capas densas y una operación de *Dropout* intermedia con probabilidad de 25 %. La tabla 3.2 resume la arquitectura de la red.

Layer	Input dimension	Output dimension	Parameters
ResNet50	3,1000,1000	2048,1,1	24,036,431
Flatten	2048	2048	—
Dense	2048	256	524,544
ReLU	256	256	—
Drop-0.25	256	256	—
Dense	256	15	3,855
Sigmoid	15	15	—

Cuadro 3.2: Deep neural network architecture used in the proposed model. The Dense layers include the bias term. Drop-0.25 means a Dropout layer with a probability of 0.25.

3.2.2. Vision Transformers

Los modelos *Vision Transformers* (*ViT*) [23] propuestos en el año 2020 son una variante de los modelos Transformer diseñados para resolver tareas de visión por computadora. En lugar de utilizar convoluciones como tradicionalmente se haría con otros modelos para extraer las distintas características de la imagen, los modelos Vision Transformers descomponen la imagen de entrada en una serie de parches, que posteriormente se tokenizan y se aplican la arquitectura de Transformer estandar presentada anteriormente, véase la figura 3.1.

El mecanismo de atención en el modelo ViT transforma continuamente los vectores de representación de los parches de la imagen, incorporando relaciones semánticas entre los parches de la imagen. Análogamente a como se usaría en el área de Procesamiento de Lenguaje Natural, mientras los vectores avanzan en profundidad en el modelo Transformer, las relaciones semánticas formadas entre los parches son cada vez más complejas. Esto remplaza así, las características extraídas a través de operaciones convolucionales. Para realizar dichas tareas el modelo ViT solo utiliza la parte codificadora de la arquitectura general del Transformer, codificando y extrayendo la información relevante de las imágenes en las etapas de multiatención.

A pesar de que los enfoques dirigidos con modelos ViT y CNN son diferentes, algunas investigaciones indican que la representación de la información en capas inferiores es bastante similar, esto es convoluciones más internas que guardan características locales y cabezas de

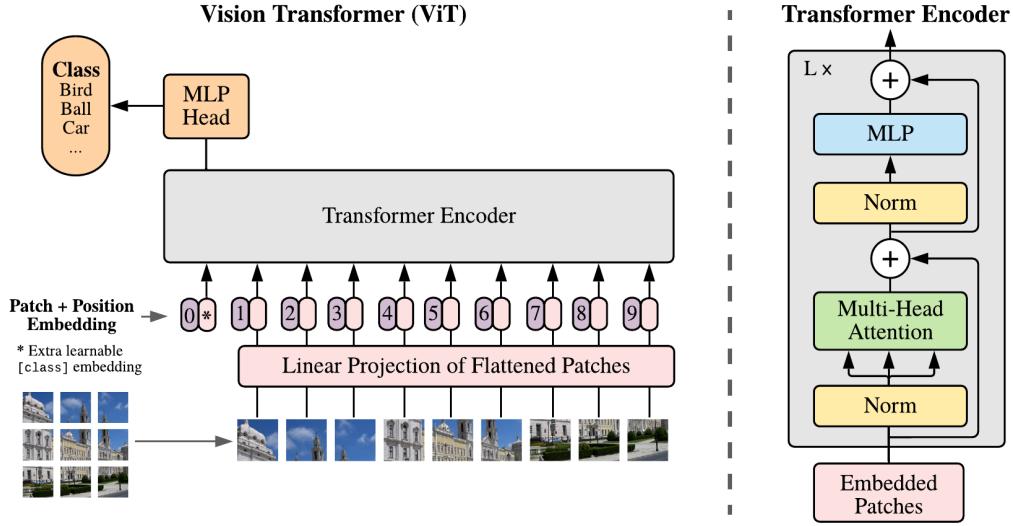


Figura 3.1: Modelo ViT [23]. Representación original del ViT. Las imágenes a procesarse son descompuestas en una serie de parches (3x3 parches en este ejemplo). Los parches generados son proyectados a una dimensión lineal agregándose información posicional.

atención más profundas [68]. Aún así, los modelos ViT incorporan más información global que los modelos CNNs en capas inferiores y los *skip connections* incorporados a los modelos influyen más en los modelos ViTs que en los modelos basados en CNNs. Además, como describen en su artículo Dosovitskiy et al. los modelos ViT se benefician de entrenamientos conjuntos de datos mucho más grandes a comparación de los datos necesarios para obtener resultados favorables en CNNs.

Basados en las propuestas de Dosovitskiy et al., la red clasificadora implementada deriva de estos modelos obtenidos.

Basados en las propuestas de Jorge Santos, la red clasificadora implementada deriva de los modelos obtenidos entrenados en este artículos. El modelo es entrenado usadno el dataset ILSVRC-2012 ImageNet dataset con 1k clases y 1.3M de imágenes, el dataset ImageNet-21k con 21k clases y 14M de imágenes y JFT con 18k clases y 303M de imagenes de alta resolución.

Basados en las propuestas de Dosovitskiy et al., la red clasificadora implementada se deriva de los modelos entrenados que describen. Los modelos se entrena utilizando varios conjuntos de datos; el conjunto de datos ILSVRC-2012 de ImageNet, que consta de 1k clases y 1.3 millones de imágenes, el conjunto de datos ImageNet-21k, que contiene 21k clases y 14 millones de imágenes y el conjunto de datos JFT, que incluye 18k clases y 303 millones de imágenes de alta resolución. Estos conjuntos de datos proporcionan una amplia variedad de ejemplos que permiten al modelo aprender y generalizar eficazmente.

Input size	384×384
Patch size	32×32
Layers	12
Hidden size	768
MLP Size	3072
Heads	12
Params	86M

Cuadro 3.3: Deep neural network architecture used in the extended model with the Tuberculosis branch. Dense layers include the bias term and * indicates trainable layers.

La arquitectura del modelo ViT está basado en las usadas por los transformers implementados en *BERT* [22] y usando la nomenclatura *ViT-Base-patch32-384*. “Base” hace referencia al modelo base de *BERT*, “patch32” a la variante que trabaja con parches de tamaño 32×32 y “384” a la dimensiones de las imágenes de entrada al modelo. En la tabla 3.3 se ejemplifica la arquitectura mencionada. 12 capas codificadoras del Transformer conforman el modelo ViT con 12 cabezas de atención con un tamaño de 768 y finalmente las segmentos MLP de tamaño 3072.

3.2.3. ViT con cabezas de atención Flexibles

Proceso de Entrenamiento

1. **Entrenamiento inicial.** Una vez seleccionada la red convolucional pre-entrenada con la base de datos de ImageNet, se reemplaza la etapa de clasificación por una nueva compuesta por dos capas densas. En la tabla 3.2 se presentan los detalles de la red *ResNet50* usada en este modelo. De esta forma, la red convolucional *ResNet50* funciona como un extracto de características; se transforma los datos originales (las imágenes de radiografías) en una nueva representación que contiene las características que permiten distinguir entre las distintas patologías. El clasificador es implementado agregando dos capas densas a esta red base. En la etapa de entrenamiento la red se comporta como como una red Perceptrón Multicapa (*Multilayer Perceptron*, MLP) donde la entrada es el tensor de características calculados por la red base o backbone. Para entrenar el clasificador, los pesos que corresponden a la red base son congelados y solamente son actualizados los pesos que pertenecen a la etapa de clasificación (las capas densas descritas en la tabla 3.2). El entrenamiento se realiza durante 35 épocas conservando el mejor modelo de acuerdo al accuracy obtenido usando el conjunto de validación.
2. **Fine-tuning.** Hasta este punto, el enfoque usado es un *aprendizaje superficial* y la etapa de extracción de características está completamente desasociado de la etapa de clasificación. La ventaja de implementar el sistema a través de dos redes neuronales (la

CAPÍTULO 3. VISION TRANSFORMERS EN DETECCIÓN DE PATOLOGÍAS EN PULMONES CON...

red backbone y la red MLP) es que podemos mejorar la extracción de características en términos de la tarea de interés. Para ello, se procede a descongelar las últimas capas convolucionales de la red backbone y continuar el entrenamiento en conjunto con las capas densas de la etapa clasificadora. Las capas a descongelar corresponden al último bloque construido de la 5^o etapa convolucional (*layer conv5_3*) [32]. Así, permitimos que el tensor obtenido a la salida de la red backbone sea particularizado a la tarea de clasificación actual. El procedimiento de *Fine-tuning* es realizado por 25 épocas más.

3. **Full-tuning.** Las razones por las cuales solamente son reentrenadas las últimas capas convolucionales son que tenemos que lidiar con el problema del desvanecimiento del gradiente y el sistema completo puede terminar sobre ajustando sus parámetros a la base de datos de entrenamiento. El primer problema no es tan relevante en este punto, el rendimiento obtenido por el modelo es satisfactorio y si no fuese posible mejorar las mejoras los parámetros de las capas convolucionales tampoco sufren un deterioro. El segundo problema es de importancia si la muestra de imágenes radiográficas son suficientemente representativas de las patologías de interés. Puesto que el sistema es suficientemente general para predecir el conjunto de imágenes de prueba correctamente. En este trabajo consideramos que tenemos suficientes datos y por lo tanto, como etapa final del entrenamiento se realiza una afinación completa del modelo. Esto es, entrenando completamente la red, la etapa de extracción de características (backbone) y la etapa de clasificación (MLP). Para evitar el over-fitting, el entrenamiento es continuado solamente por 10 épocas conservando el mejor modelo de acuerdo al accuracy obtenido en el conjunto de validación.

Similar a Rajpurkar et al., usamos, al inicio de cada etapa de entrenamiento, las imágenes de entrenamiento son volteadas horizontalmente con 0.5 de probabilidad como técnica de augmentation de datos.

La salida de la red es un vector de tamaño igual a el número de patologías detectadas, 15. Cada elemento del vector $\tilde{y}_i \in [0, 1]$ puede ser interpretado como la probabilidad que la i-ésima patología esta presente en la imagen analizada. El vector \tilde{y} no necesariamente tiene que sumar 1, puesto que varias patologías pueden estar presentes en la radiografía. Una patología detectada como positiva ocurre si $\tilde{y} > \theta$, donde θ es el umbral. El valor típico para este umbral es de 0,5 y puede ser modificado dependiendo del análisis de la curva del *Receiver Operator Characteristic*, o *Curva ROC*. Como algoritmo de entrenamiento se usa *Adam* [?] con un factor de aprendizaje (LR) de 1×10^{-4} , parámetros de inercia $\beta_1 = 0,9$ y $\beta_2 = 0,999$ con un descenso (LR-decay) de 0,1 si después de 10 iteraciones no se detecta una reducción en el valor de la función de pérdida mayor a 1×10^{-4} (plateau escape).

Transfer Learning para la detección de Neumonía por Tuberculosis

El propósito de esta sección es demostrar que el backbone (el modelo ResNet50) del modelo propuesto y entrenado con las 15 patologías mencionadas anteriormente, puede ser la base para desarrollar detectores para otras enfermedades de pulmón. La idea en esta etapa es no repetir el proceso completo de entrenamiento sino usar en vez una simple estrategia de Transfer Learning. Así, se procede a extender el modelo para detectar (junto con las otras enfermedades) neumonía causada por Tuberculosis [83], un tipo de neumonía bacterial común en países en desarrollo y también frecuentemente reportado en pacientes con síndrome de inmunodeficiencia adquirida (AIDS) [57]. El dataset considerado incluye casos de *Tuberculosis* y *no Tuberculosis* pero es importante aclarar que la condición en particular de los casos de *no Tuberculosis* no es especificada por completo, es decir, incluyen tanto pacientes saludables como pacientes con otras afecciones.

Class	Disease	Source	Train	Test
16	Tuberculosis	3	888	488
—	Non-Tuberculosis	3	6000	1600

Cuadro 3.4: Number of chest radiography's from patients with and without Tuberculosis.

La base de datos (indicada proveniente de la fuente 3 en la tabla 3.4) está compuesta por radiografías provenientes de: TBX11K dataset [48], India (DA and DB) dataset [10], Montgomery County dataset [37], y Shenzhen Hospital dataset [37]. En este trabajo se usa las listas originales para el entrenamiento y Evaluación definidos para este dataset.

Para poder detectar Tuberculosis, se realiza la implementación de un clasificador binario usando como backbone la red ResNet50 entrenada previamente para la detección de las 15 patologías anteriores. La nueva rama de clasificación incluye dos nuevas capas densas (con sus respectivas funciones de activación). Conservamos los parámetros correspondientes al backbone no entrenables y solo se entrena las nuevas capas densas usando la estrategia mencionada en la subsección de **Initial Training** 3.2.3 con el optimizador, factor de aprendizaje y otros parámetros sin cambios y *Weighted Binary Cross-Entropy* como función de pérdida similar a (3.1). Este modelo extendido aún detecta las 15 enfermedades comentadas previamente con una salida binaria extra para Tuberculosis. La arquitectura extendida incluyendo la nueva rama está descrita en la tabla 3.5.

3.2.4. Métricas de Evaluación

Considerando un problema de clasificación binario donde cada radiografía tiene una etiqueta $y = \{1, 0\}$ con 1 indicando la presencia del padecimiento en el paciente y 0 significando que se encuentra sano, el detector puede tener dos posibles resultados: una detección positiva

CAPÍTULO 3. VISION TRANSFORMERS EN DETECCIÓN DE PATOLOGÍAS EN PULMONES CON...

Layer	Input dimension	Output dimension	Parameters
Initial Backbone			
ResNet50	3,1000,1000	2048,1,1	24,036,431
Flatten	2048	2048	—
Dense	2048	256	524,544
ReLU	256	256	—
Drop-0.25	256	256	—
Additional Branch			
Dense*	256	128	38,896
ReLU	128	128	—
Drop-0.20	128	128	—
Dense*	128	1	129
Sigmoid	1	1	—

Cuadro 3.5: Deep neural network architecture used in the extended model with the Tuberculosis branch. Dense layers include the bias term and * indicates trainable layers.

(P) para la enfermedad o una detección negativa (N). La tabla 3.6 muestra la caracterización de la etiqueta predicha de acuerdo a los valores reales (Ground Truth, GT). Si un paciente enfermo es correctamente detectado, tenemos un *Verdadero Positivo* (True Positive, TP) y si la predicción falla, es una *Falso Positivo* (False Negativo, FN). Por otro lado, si una etiqueta positiva es erroneamente predicha en un paciente saludable entonces tenemos un *Falso Positivo* (False Positive, FP), y si el paciente saludable es correctamente predicho es un *Verdadero Negativo* (True Negative, TN). La tabla 3.6 muestra la *Matriz de Confusión* con el conteo de cada tipo de predicción en el conjunto de prueba.

Para este trabajo asumimos que una patología en particular es correctamente detectada si su correspondiente puntaje en el vector de predicho es más significante que cierto umbral. En particular, asumimos un umbral igual para todas las patologías de 0,5. Usando la Matriz de Confusión podemos definir varias métricas de rendimiento.

1. Accuracy (*A*). Esta métrica es quizás la más obvia. Corresponde al razón de los datos predichos correctamente sobre el total.

$$A = \frac{TP + TN}{TP + TN + FP + FN}. \quad (3.3)$$

2. Recall o Sensibility (*R*). Es la fracción de pacientes con enfermedades correctamente detectados. Esta metrica también es conocida como Tasa de Verdaderos Positivos (True Positive Rate, TPR), la tasa de detecciones correctas.

$$R = \frac{TP}{TP + FN} \quad (3.4)$$

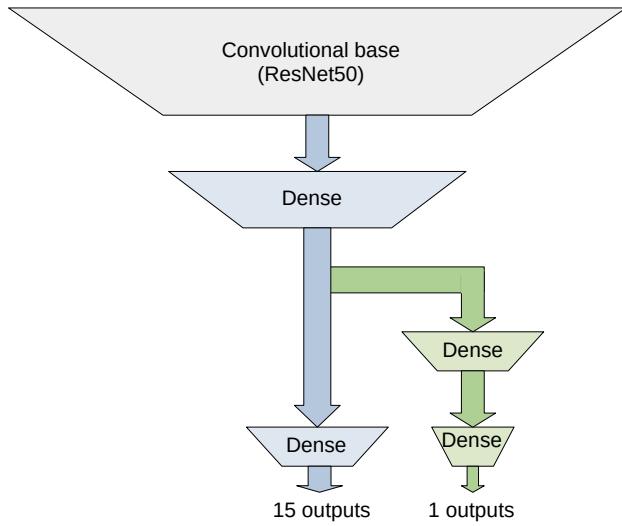


Figura 3.2: Scheme of the extension of the 15 pathology detector including a new branch for Tuberculosis detection.

		Prediction		
		Positive	Negative	
Disease	TP	FN		$\rightarrow R = \frac{TP}{TP+FN}$
GT				
No-Disease	FP	TN		$\rightarrow S = \frac{TN}{FP+TN}$
	$P = \frac{TP}{TP+FP}$	$NPV = \frac{TN}{FN+TN}$		

Cuadro 3.6: Interpretation of the test results (prediction) according to the ground truth (GT). Metrics are calculated as the ratio between the diagonal element and the sum per row or column, as the case may be. R, recall or sensitivity; S, specificity; P, precision; NPV, negative prediction value

3. Specificity (S). Es la fracción de pacientes sin enfermedades que son detectados correctamente.

$$S = \frac{TN}{TN + FP} \quad (3.5)$$

4. False Positive Rate (FPR). La tasa de detecciones falsas de enfermedades, es calculada como:

$$FPR = 1 - S, \quad (3.6)$$

donde S es la Specificity.

5. Precision (P). Es la fracción de predicciones positivas que realmente tiene una enfermedad.

$$P = \frac{TP}{TP + FP} \quad (3.7)$$

6. F_1 Score (F_1). Durante la creación de modelos buscamos un balance entre Precision y Recall. El incremento de alguna de estas dos métricas es en compensación de la otra. Un modelo que predice la mayoría de los datos como positivos tendrá un Recall más cercano a 1 pues el conteo de Falsos Negativos será mucho menor pero los Falsos positivos incrementarán reduciendo la Precision del modelo. En caso contrario si el modelo predice la mayoría de los datos como negativos el conteo de Falsos Positivos será menor incrementando la métrica de Precision pero los Falsos Negativos incrementan provocando que el Recall decremente. F_1 -Score busca un balance entre las dos métricas (Precision-Recall Trade-Off). Un imbalance entre ellas indica una sesgo en cualquiera de las dos maneras descritas anteriormente. Por ello, es más informativo como medida de rendimiento usar una media geométrica entre las dos métricas.

$$F_1 = \frac{2 PR}{P + R}. \quad (3.8)$$

7. Area Under Curve of Receiver Operator Characteristic, (AUC-ROC). La Curva Característica Operativa del Receptor es un gráfico que muestra las capacidades de diagnóstico de clasificadores binarios. Como se mencionó anteriormente, una patología es detectada como positiva si su puntaje predicho es más significativo que un umbral dado. Así, ajustando dicho umbral en valores más bajos, permite en general incrementar el TPR, aunque el FPR también puede ser incrementado. La Curva ROC resulta de graficar los valores de TPR contra FPR variando el umbral en el intervalo $[0, 1]$. El AUC corresponde al área bajo la Curva de ROC.

3.2.5. Resultados

La Tabla 3.7 muestra un resumen del rendimiento del modelo propuesto. Las imágenes que conforman los conjuntos de entrenamiento, validación y prueba corresponden a los definidos originalmente en dataset. En ChestX-ray14, se redefinen las etiquetas de entrenamiento de acuerdo a CheXNet [69]; excepto para las imágenes de *COVID-19* y sin patologías identificadas. Es válido usar cualquier modificación de etiquetas en la etapa de entrenamiento siempre y cuando se mantenga el conjunto de datos de prueba sin cambios (imágenes de radiografía y sus etiquetas).

Los resultados presentes se ajustan a el procedimiento de entrenamiento y prueba antes descrito. Las métricas que se incluyen son: *Area Under the the Precision Recovery Curve*

Phatology	Proposal			
	AUC-PR	AUC-ROC	F1-Score	Acc.
Cardiomegaly	0.290	0.875	0.324	0.911
Emphysema	0.413	0.938	0.350	0.886
Effusion	0.494	0.850	0.527	0.807
Hernia	0.050	0.855	0.051	0.957
Infiltration	0.382	0.740	0.458	0.711
Mass	0.290	0.831	0.340	0.877
Nodule	0.249	0.806	0.299	0.876
Atelectasis	0.336	0.794	0.387	0.815
Pneumothorax	0.447	0.906	0.496	0.852
Pleural-Thick	0.157	0.820	0.217	0.852
Pneumonia	0.425	0.863	0.274	0.856
Fibrosis	0.106	0.852	0.154	0.910
Edema	0.187	0.879	0.193	0.788
Consolidation	0.150	0.776	0.234	0.754
COVID-19	0.844	0.991	0.799	0.969
Healthy	0.691	0.736	0.496	0.712
Global-14	0.284	0.842	0.307	0.846
Global-15	0.344	0.852	0.350	0.846

Cuadro 3.7: Summary of the proposed method's performance. The train, validation and test datasets correspond to the originally defined in ChestX-ray14. However, the training labels correspond to the defined by the relabelling computed by CheXNet [69]; see text. The threshold was set equal to 0.5, no extra optimization was performed.

(AUC-PR), *Area Under the Receiver Operating Characteristic Curve* (AUC-ROC), *F1-Score* y *Accuracy* (Acc.). *Global-14* muestra el rendimiento en las patologías de *ChestX-ray-14* (clasificaciones correctas vs clasificaciones incorrectas), *Global-15* muestra el rendimiento incluyendo *COVID-19*. Nótese que el modelo propuesto tiene su mejor rendimiento en la detección de *COVID-19*, la enfermedad anexada. También se han incluido datos de clases sin patologías detectadas (healthy) debido a la importancia de tener un buen desempeño en la detección de sujetos saludables. El rendimiento global sobre los datos de *ChestX-ray14* y *COVID-19* se muestran en el último renglón de la tabla 3.7. Adicionalmente, solo como referencia se incluye en el Apéndice la tabla con los resultados del rendimiento del modelo propuesto con el conjunto de prueba re-etiquetado.

Es importante recalcar que no es apropiado usar *Accuracy* como único criterio de rendimiento bajo este contexto. El *Accuracy* de cada clase es calculado como el número de clasificaciones correctas (TP y TN) entre la suma total de radiografías analizadas. Con clases desbalanceadas, tales como las que analizamos ahora (patología vs no patología), un modelo podría ser engañoso en cuanto a su eficiencia segun su clasificación sobre la o las

CAPÍTULO 3. VISION TRANSFORMERS EN DETECCIÓN DE PATOLOGÍAS EN PULMONES CON AI

clases más representativas. Si el desbalance grande, el modelo puede asignar todos los datos como la clase que representa mayor peso y por tanto tener un buen rendimiento en la métrica de *Accuracy*. A pesar de ello, los datos presentados son resultados que sirven como referencia para el lector. Una manera más apropiada de medir el desempeño del modelo es calcular la gráfica de TPR vs FPR para obtener el área bajo la curva, en otras palabras, calcular AUC-ROC como la métrica de rendimiento [31]. A pesar de que la curva de ROC es preferida sobre la de PR a causa de su convexidad, existe un mapeo uno a uno entre ellas [20], al final, ambas curvas son calculadas usando la misma matriz de confusión. Por tanto, también se incluye la métrica AUC-PR para mejores comparaciones mayormente informativas que solo usar la curva de ROC [73].

La tabla 3.8 muestra una comparación de la métrica AUC-ROC para los modelos DR-CNN [78], CRAL [28], TSNC [11], CheXNet [69] y la propuesta descrita aquí. También se realiza la comparativa del modelo ResNet-38-large-meta (LargeMeta). Este modelo usa ResNet-38 como backbone, un tamaño de imagen de 480x480 pixeles y, a diferencia de los modelos comparados restantes utiliza datos adjuntos a la imagen (metadata). Las dos últimas columnas presentan la AUC-ROC para el modelo propuesto y la detección lograda un grupo de radiólogos [69]. Se indica en negritas el modelo con el mejor rendimiento por patología. En general la métrica AUC-TOC en el dataset original ChestX-ray14 muestra una ligera ventaja para el modelo propuesto sobre el modelo CheXNet. La inclusión de COVID-19 mejora dicha métrica de rendimiento en favor del modelo propuesto. Además, es visible que los resultados obtenidos también superan a aquellos reportados por otros métodos investigados en ???. Es importante notar que la métrica AUC-ROC para CheXNet corresponde a la media de 1000 salidas de usando bootstrapping para un Intervalo de Confianza de 95 % [69]. Una actualización reciente de la implementación de CheXNet en el repositorio [] reporta un AUC-ROC para Global-14 igual a 0.847. La última columna en la tabla 7 presenta el rendimiento promedio para los radiólogos que evalúan la radiografía para ChestX-ray14. Para mayor detalle del cálculo de la evaluación de los radiólogos véase [69]. Las marcas con asterisco (*) indican donde los radiólogos tiene mejor rendimiento que cualquier de los modelos evaluados. La tabla 3.9 resume el rendimiento de la métrica AUC-ROC de los mejores métodos basados en DL (incluyendo el propio analizado) contra rendimiento de los radiólogos. Podemos notar una ventaja si unimos los métodos basados en DL, los cuales superar a los radiólogos en detectar seis patologías (sin tomar en cuenta COVID-19).

La figura 3.3 representa las curvas de ROC correspondientes a nuestra propuesta para todas las patologías incluyendo las clases COVID-19 y Healthy. como pueden verse en cada uno de los gráficos, las curvas se encuentran sobre la diagonal, en particular para COVID-19. La figura ilustra la detección obtenida mediante el método propuesto: una diagnóstico previo y la región de interés detectado usando GradCAM; una técnica de visualización de redes neuronales profundas propuesta en [76]. La primer fila muestra cuatro radiografías de pacientes con diferentes patologías, la segunda fila se encuentra su correspondiente GradCAM indicando la región que contribuye a la detección. Las imágenes radiográficas de la figura están correctamente clasificadas de acuerdo con las etiquetas originales. Las regiones rojas

CAPÍTULO 3. VISION TRANSFORMERS EN DETECCIÓN DE PATOLOGÍAS EN PULMONES CON

	Compared Models						
	CRAL	DR-CNN	TSNC	LargeMeta	CheXNet	Proposal	Radiol.
Cardiomegaly	0.880	0.801	0.887	0.875	0.923	0.875	0.888
Emphysema	0.908	0.773	0.930	0.895	0.937	0.938	0.911
Effusion	0.829	0.797	0.831	0.822	0.864	0.850	0.900*
Hernia	0.917	0.748	0.921	0.937	0.916	0.855	0.985*
Infiltration	0.702	0.751	0.703	0.694	0.734	0.740	0.734
Mass	0.834	0.760	0.833	0.820	0.868	0.831	0.886*
Nodule	0.773	0.741	0.798	0.747	0.780	0.806	0.899*
Atelectasis	0.781	0.766	0.785	0.763	0.809	0.794	0.808
Pneumothorax	0.729	0.778	0.731	0.840	0.889	0.906	0.940*
Pleural-Thick.	0.778	0.759	0.782	0.763	0.806	0.820	0.779
Pneumonia	0.857	0.800	0.881	0.731	0.768	0.863	0.823
Fibrosis	0.830	0.765	0.833	0.816	0.805	0.852	0.897*
Edema	0.850	0.820	0.849	0.846	0.888	0.879	0.910*
Consolidation	0.754	0.787	0.754	0.749	0.790	0.776	0.841*
COVID-19	—	—	—	—	—	0.991	—
Healthy	—	—	—	0.727	—	0.736	—
Global-14	0.816	0.775	0.823	0.807	0.841	0.842	0.872*
Global-15	—	—	—	—	—	0.852	—

Cuadro 3.8: AUC-ROC values for the pathologies and healthy classes. The compared methods correspond to deep neural networks models trained for the 14 pathologies in ChestX-ray14.

son aquellas con mayor contribución a la detección de la enfermedad. La radiografía de la primer columna claramente muestra un corazón de tamaño mayor de lo normal. La segunda columna corresponde a neumonía, la tercera columna muestra una formación inusual en el pulmón (mass), la última columna indica el caso de COVID-19.

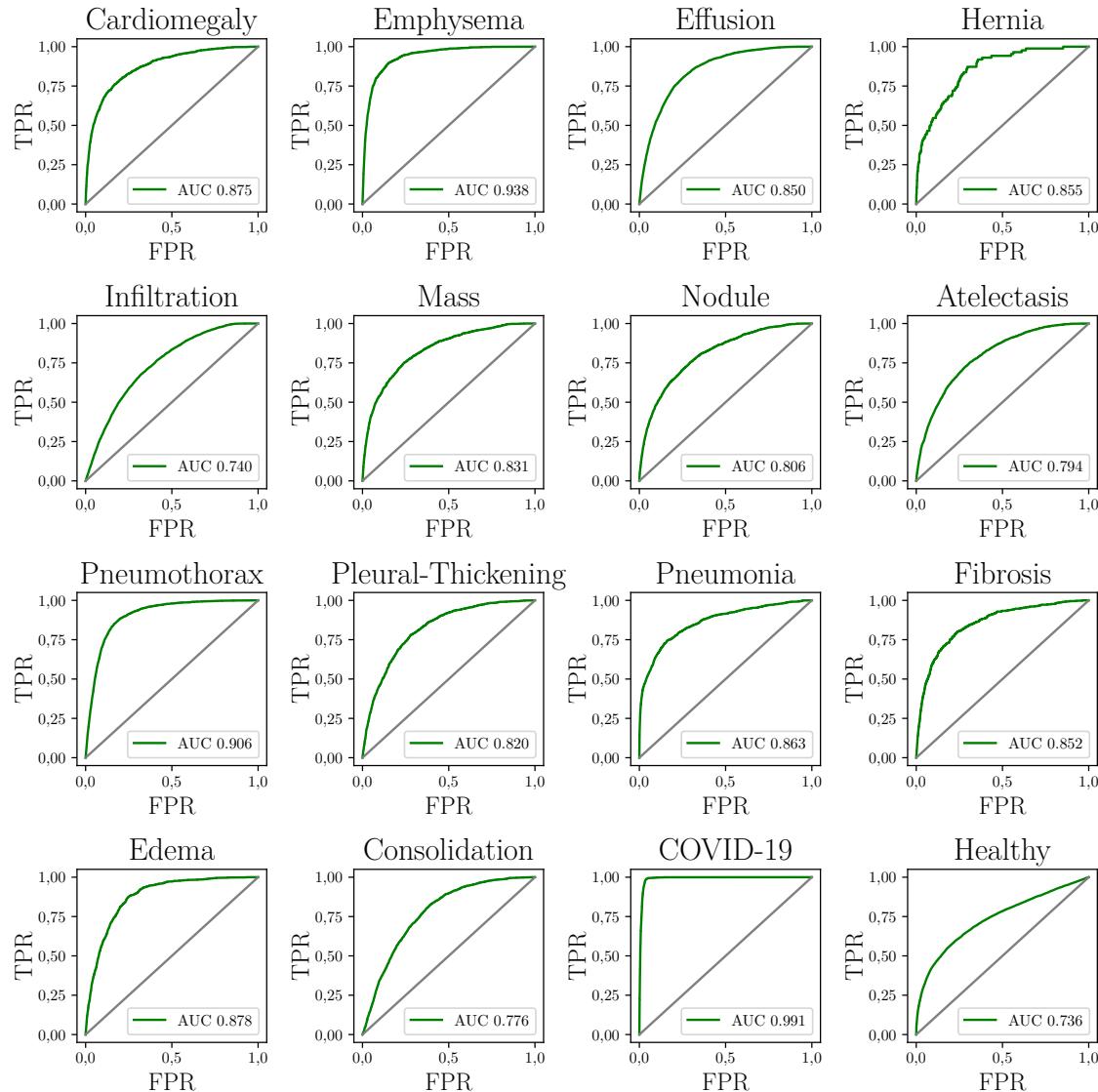


Figura 3.3: ROC Curves (plots of FPR versus TPR) for the pathology and healthy classes.

CAPÍTULO 3. VISION TRANSFORMERS EN DETECCIÓN DE PATOLOGÍAS EN PULMONES CON X-RAY

	Method	AUC-ROC	Radiol.
Cardiomegaly	CheXNet	0.923	0.888
Emphysema	Proposal	0.938	0.911
Effusion	CheXNet	0.864	0.900*
Hernia	LargeMeta	0.937	0.985*
Infiltration	DR-CNN	0.751	0.734
Mass	CheXNet	0.868	0.886*
Nodule	Proposal	0.806	0.899*
Atelectasis	CheXNet	0.809	0.808
Pneumothorax	Proposal	0.906	0.940*
Pleural-Thick.	Proposal	0.820	0.779
Pneumonia	TSNC	0.881	0.823
Fibrosis	Proposal	0.852	0.897*
Edema	CheXNet	0.888	0.910*
Consolidation	CheXNet	0.790	0.841*
COVID-19	Proposal	0.991	—

Cuadro 3.9: Comparison, per pathology, of the deep learning based method with best performance versus human radiologists. Human radiologists lead in eight (indicated with *) of the fifteen pathologies.

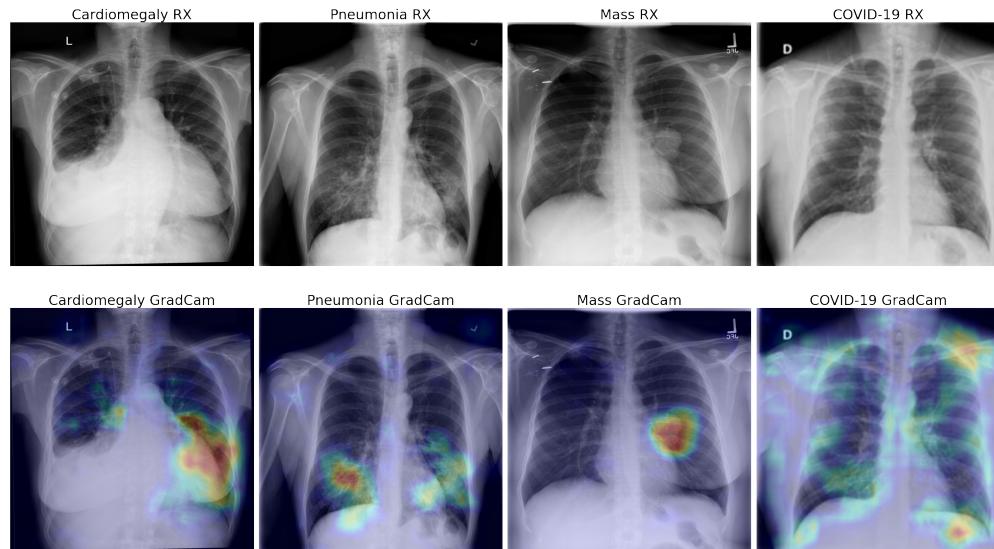


Figura 3.4: Some X-ray images (top row) with different pathologies and their associated GradCam images (bottom row). The first column shows a grown heart, the second column corresponds to Pneumonia, the third column shows an unusual tissue in the lung, and the last column a case of COVID-19.

Capítulo 4

Conclusiones y Trabajo Futuro

Apéndice A

An appendix

Appendices are a good idea for almost any thesis. Your main thesis body will likely contain perhaps 40-60 pages of text and figures. You may well write a larger document than this, but chances are that some of the information contained therein, while important, does *not* merit a place in the main body of the document. This sort of content - peripheral clarifying details, computer code, information of use to future students but not critical to understanding your work . . . - should be allocated to one or several appendices.

A.1. About the bibliography

What follows this is the bibliography. This has its own separate environment and syntax; check out the comments in the .tex files for details. Worth noting, though, is that you may find it helpful to use automated bibliography management tools. BibTeX will automatically generate a bibliography from you if you create a database of references. Other software - for example JabRef on a pc - can be used to make managing the reference database easy. Regardless, once you've created a .bib file you can cite it in the body of your thesis using the \cite tag. For example, one might wish to cite a reference by Bermudez [?]. If you use BibTeX, you can put the relevant information into a referencedatabase (called bibliography.bib here), and then BibTeX will compile the references into a .bbl file ordered appropriately for your thesis based on when the citations appear in the main document.

Bibliografía

- [1] Tarun Agrawal and Prakash Choudhary. Focuscovid: automated covid-19 detection using deep learning with chest x-ray images. *Evolving Systems*, pages 1–15, 2021.
- [2] Tao Ai, Zhenlu Yang, Hongyan Hou, Chenao Zhan, Chong Chen, Wenzhi Lv, Qian Tao, Ziyong Sun, and Liming Xia. Correlation of chest ct and rt-pcr testing for coronavirus disease 2019 (covid-19) in china: a report of 1014 cases. *Radiology*, 296(2):E32–E40, 2020.
- [3] Caroline Apra, Charlotte Caucheteux, Arthur Mensch, Jenny Mansour, Mélodie Bernaux, Agnès Dechartres, Erwan Debuc, Xavier Lescure, Aurélien Dinh, Youri Yordanov, Patrick Jourdain, Nicolas Paris, Alexandre Gramfort, Amélie Aime-Eusebi, Alexandre Bleibtreu, Laurène Deconinck, Christine Katlama, Josselin Lebel, François-Xavier Lescure, Yves Artigou, Amélie Banzet, Elodie Boucheron, Christiane Boudier, Edouard Buzenac, Marie-Claire Chapron, Dalhia Chekaoui, Laurent De Bastard, Alexandre Grenier, Pierre-Etienne Haas, Julien Hody, Michèle Jarraya, Louis Lacaille, Aurélie Le Guern, Jeremy Leclert, Fanny Male, Jérôme Marchand-Arvier, Emmanuel Martin-Blondet, Apolinne Nassour, Oussama Ourahou, Thomas Penn, Ambre Ribardiere, Nicolas Robin, Camille Rouge, Nicolas Schmidt, Pascaline Villie, The AP-HP/Universities/Inserm COVID-19 Research Collaboration, Writing Committee, Data Science Committee, Scientific Committee, and Covidom Regional Center Steering Committee. Predictive usefulness of rt-pcr testing in different patterns of covid-19 symptomatology: analysis of a french cohort of 12,810 outpatients. *Scientific Reports*, 11(1):21233, Oct 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-99991-6. URL <https://doi.org/10.1038/s41598-021-99991-6>.
- [4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.
- [5] Pedro RAS Bassi and Romis Attux. A deep convolutional neural network for covid-19 detection using chest x-rays. *Research on Biomedical Engineering*, pages 1–10, 2021.
- [6] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. URL <https://arxiv.org/abs/2004.05150>.

- [7] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.
- [8] Keno K Bressem, Lisa C Adams, Christoph Erxleben, Bernd Hamm, Stefan M Niehues, and Janis L Vahldiek. Comparing different deep learning architectures for classification of chest radiographs. *Scientific reports*, 10(1):1–16, 2020.
- [9] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. An attentive survey of attention models. *CoRR*, abs/1904.02874, 2019. URL <http://arxiv.org/abs/1904.02874>.
- [10] Arun Chauhan, Devesh Chauhan, and Chittaranjan Rout. Role of gist and phog features in computer-aided diagnosis of tuberculosis without segmentation. *PloS one*, 9(11):e112980, 2014.
- [11] Bingzhi Chen, Zheng Zhang, Jianyong Lin, Yi Chen, and Guangming Lu. Two-stream collaborative network for multi-label chest x-ray image classification with lung segmentation. *Pattern Recognition Letters*, 135:221–227, 2020. ISSN 0167-8655. doi: <https://doi.org/10.1016/j.patrec.2020.04.016>. URL <https://www.sciencedirect.com/science/article/pii/S0167865520301380>.
- [12] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George F. Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. *CoRR*, abs/1804.09849, 2018. URL <http://arxiv.org/abs/1804.09849>.
- [13] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019. URL <http://arxiv.org/abs/1904.10509>.
- [14] KyungHyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL <http://arxiv.org/abs/1409.1259>.
- [15] Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL <http://arxiv.org/abs/1406.1078>.
- [16] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. *CoRR*, abs/2009.14794, 2020. URL <https://arxiv.org/abs/2009.14794>.

- [17] Junyoung Chung, Caglar Gülcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL <http://arxiv.org/abs/1412.3555>.
- [18] Joseph Paul Cohen, Paul Morrison, and Lan Dao. Covid-19 image data collection, 2020.
- [19] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019. URL <http://arxiv.org/abs/1901.02860>.
- [20] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 233–240, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143874. URL <https://doi.org/10.1145/1143844.1143874>.
- [21] Erkan Deniz, Abdulkadir Şengür, Zehra Kadiroğlu, Yanhui Guo, Varun Bajaj, and Ümit Budak. Transfer learning based histopathologic image classification for breast cancer detection. *Health information science and systems*, 6(1):1–7, 2018.
- [22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL <http://arxiv.org/abs/1810.04805>.
- [23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL <https://arxiv.org/abs/2010.11929>.
- [24] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017. URL <http://arxiv.org/abs/1705.03122>.
- [25] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2017. <http://www.deeplearningbook.org>.
- [26] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL <http://arxiv.org/abs/1410.5401>.
- [27] Hayit Greenspan, Raúl San José Estépar, Wiro J Niessen, Eliot Siegel, and Mads Nielsen. Position paper on covid-19 imaging and ai: From the clinical needs and technological challenges to initial ai solutions at the lab and national level towards a new era for ai in healthcare. *Medical image analysis*, 66:101800, 2020.

- [28] Qingji Guan and Yaping Huang. Multi-label chest x-ray image classification via category-wise residual attention learning. *Pattern Recognition Letters*, 130:259–266, 2020. ISSN 0167-8655. doi: <https://doi.org/10.1016/j.patrec.2018.10.027>. URL <https://www.sciencedirect.com/science/article/pii/S0167865518308559>. Image/Video Understanding and Analysis (IUVA).
- [29] Anunay Gupta, Shreyansh Gupta, Rahul Katarya, et al. Instacovnet-19: A deep learning classification model for the detection of covid-19 patients using chest x-ray. *Applied Soft Computing*, 99:106859, 2021.
- [30] Niharika Gupta, Shine Augustine, Tarun Narayan, Alan O’Riordan, Asmita Das, D. Kumar, John H. T. Luong, and Bansi D. Malhotra. Point-of-care pcr assays for covid-19 detection. *Biosensors*, 11(5):141, May 2021. ISSN 2079-6374. doi: 10.3390/bios11050141. URL <https://pubmed.ncbi.nlm.nih.gov/34062874/> [pmid].
- [31] J A Hanley and B J McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, September 1983.
- [32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [33] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [34] Chaolin Huang, Yeming Wang, Xingwang Li, Lili Ren, Jianping Zhao, Yi Hu, Li Zhang, Guohui Fan, Jiuyang Xu, Xiaoying Gu, et al. Clinical features of patients infected with 2019 novel coronavirus in wuhan, china. *The Lancet*, 395(10223):497–506, 2020.
- [35] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- [36] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4475–4483. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/huang20f.html>.
- [37] Stefan Jaeger, Sema Candemir, Sameer Antani, Yí-Xiáng J Wáng, Pu-Xuan Lu, and George Thoma. Two public chest x-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative imaging in medicine and surgery*, 4(6):475, 2014.

- [38] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *CoRR*, abs/2001.04451, 2020. URL <https://arxiv.org/abs/2001.04451>.
- [39] John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. 2001. doi: 10.1109/9780470544037.ch14.
- [40] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontañón. Fnet: Mixing tokens with fourier transforms. *CoRR*, abs/2105.03824, 2021. URL <https://arxiv.org/abs/2105.03824>.
- [41] Yang Li, Lukasz Kaiser, Samy Bengio, and Si Si. Area attention. *CoRR*, abs/1810.10126, 2018. URL <http://arxiv.org/abs/1810.10126>.
- [42] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.
- [43] Diana E. Litmanovich, Michael Chung, Rachael R. Kirkbride, Gregory Kicska, and Jeffrey P. Kanne. Review of chest radiograph findings of covid-19 pneumonia and suggested reporting language. *Journal of Thoracic Imaging*, 35(6), 2020. ISSN 0883-5993. URL https://journals.lww.com/thoracicimaging/Fulltext/2020/11000/Review_of_Chest_Radiograph_Findings_of_COVID_19.4.aspx.
- [44] Dan Liu, Chenhui Ju, Chao Han, Rui Shi, Xuehui Chen, Demin Duan, Jinghua Yan, and Xiyun Yan. Nanozyme chemiluminescence paper test for rapid and sensitive detection of sars-cov-2 antigen. *Biosensors and Bioelectronics*, 173:112817, 2021. ISSN 0956-5663. doi: <https://doi.org/10.1016/j.bios.2020.112817>. URL <https://www.sciencedirect.com/science/article/pii/S0956566320308034>.
- [45] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *CoRR*, abs/1908.03265, 2019. URL <http://arxiv.org/abs/1908.03265>.
- [46] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.
- [47] Yun Liu, Yu-Huan Wu, Yunfeng Ban, Huifang Wang, and Ming-Ming Cheng. Rethinking computer-aided tuberculosis diagnosis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2643–2652, 2020. doi: 10.1109/CVPR42600.2020.00272.

- [48] Yun Liu, Yu-Huan Wu, Yunfeng Ban, Huifang Wang, and Ming-Ming Cheng. Rethinking computer-aided tuberculosis diagnosis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2646–2655, 2020.
- [49] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. *arXiv preprint arXiv:1602.04433*, 2016.
- [50] Bin Lou, Ting-Dong Li, Shu-Fa Zheng, Ying-Ying Su, Zhi-Yong Li, Wei Liu, Fei Yu, Sheng-Xiang Ge, Qian-Da Zou, Quan Yuan, et al. Serology characteristics of sars-cov-2 infection after exposure and post-symptom onset. *European Respiratory Journal*, 56(2), 2020.
- [51] Chunjie Luo, Xiwen He, Jianfeng Zhan, Lei Wang, Wanling Gao, and Jiahui Dai. Comparison and benchmarking of ai models and frameworks on mobile devices. *arXiv preprint arXiv:2005.05085*, 2020.
- [52] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL <http://arxiv.org/abs/1508.04025>.
- [53] Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Interactive attention networks for aspect-level sentiment classification. *CoRR*, abs/1709.00893, 2017. URL <http://arxiv.org/abs/1709.00893>.
- [54] Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization. *CoRR*, abs/1910.04209, 2019. URL <http://arxiv.org/abs/1910.04209>.
- [55] André F. T. Martins and Ramón Fernandez Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. *CoRR*, abs/1602.02068, 2016. URL <http://arxiv.org/abs/1602.02068>.
- [56] André F. T. Martins, Marcos V. Treviso, António Farinhas, Vlad Niculae, Mário A. T. Figueiredo, and Pedro M. Q. Aguiar. Sparse and continuous attention mechanisms. *CoRR*, abs/2006.07214, 2020. URL <https://arxiv.org/abs/2006.07214>.
- [57] Hiroki Matsuura and Yasufumi Yamaji. Tuberculous pneumonia. *QJM: An International Journal of Medicine*, 111(2):131–131, 2018.
- [58] Radiological Society of North America. Covid-19 radiography database, 2019. URL <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>.
- [59] Radiological Society of North America. Rsna pneumonia detection challenge., 2019. URL <https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data>.

- [60] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.
- [61] World Health Organization et al. Who director-general’s opening remarks at the media briefing on covid-19-11 march 2020, 2020.
- [62] Megan O’Driscoll, Gabriel Ribeiro Dos Santos, Lin Wang, Derek AT Cummings, Andrew S Azman, Juliette Paireau, Arnaud Fontanet, Simon Cauchemez, and Henrik Salje. Age-specific mortality and immunity patterns of sars-cov-2. *Nature*, 590(7844):140–145, 2021.
- [63] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, and Alexander Ku. Image transformer. *CoRR*, abs/1802.05751, 2018. URL <http://arxiv.org/abs/1802.05751>.
- [64] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28(3) of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013.
- [65] John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. Deeper attention to abusive user content moderation. pages 1125–1135, 01 2017. doi: 10.18653/v1/D17-1117.
- [66] Martin Popel and Ondrej Bojar. Training tips for the transformer model. *CoRR*, abs/1804.00247, 2018. URL <http://arxiv.org/abs/1804.00247>.
- [67] Murali Krishna Puttagunta and S Ravi. Detection of tuberculosis based on deep learning based methods. In *Journal of Physics: Conference Series*, volume 1767, page 012004. IOP Publishing, 2021.
- [68] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *CoRR*, abs/2108.08810, 2021. URL <https://arxiv.org/abs/2108.08810>.
- [69] Pranav Rajpurkar, Jeremy Irvin, Robyn L Ball, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis P Langlotz, et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists. *PLoS Medicine*, 15(11):e1002686, 2018.
- [70] Michael Roberts, Derek Driggs, Matthew Thorpe, Julian Gilbey, Michael Yeung, Stephan Ursprung, Angelica I Aviles-Rivero, Christian Etmann, Cathal McCague, Lucian Beer, et al. Common pitfalls and recommendations for using machine learning

- to detect and prognosticate for covid-19 using chest radiographs and ct scans. *Nature Machine Intelligence*, 3(3):199–217, 2021.
- [71] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL <https://doi.org/10.1038/323533a0>.
 - [72] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.
 - [73] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One*, 10(3):e0118432, March 2015.
 - [74] Charles F. Schuler, Carmen Gherasim, Kelly O’Shea, David M. Manthei, Jesse Chen, Don Giacherio, Jonathan P. Troost, James L. Baldwin, and James R. Baker. Accurate point-of-care serology tests for covid-19. *PLOS ONE*, 16(3):e0248729, 2021. doi: 10.1371/journal.pone.0248729. URL <https://app.dimensions.ai/details/publication/pub.1136450856> and <https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0248729&type=printable>.
 - [75] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.
 - [76] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.
 - [77] Anis Shazia, Tan Zi Xuan, Joon Huang Chuah, Juliana Usman, Pengjiang Qian, and Khin Wee Lai. A comparative study of multiple neural network for detection of covid 19 on chest x-ray. *EURASIP J. Adv. Signal Process.*, (50), 2021.
 - [78] Yan Shen and Mingchen Gao. Dynamic routing on deep neural network for thoracic disease classification and sensitive area localization. *CoRR*, abs/1808.05744, 2018. URL <http://arxiv.org/abs/1808.05744>.
 - [79] Afshin Shoeibi, Marjane Khodatars, Roohallah Alizadehsani, Navid Ghassemi, Mahboobeh Jafari, Parisa Moridian, Ali Khadem, Delaram Sadeghi, Sadiq Hussain, Assef Zare, et al. Automated detection and forecasting of covid-19 using deep learning techniques: A review. *arXiv preprint arXiv:2007.10785*, 2020.

- [80] Junaid Shuja, Eisa Alanazi, Waleed Alasmary, and Abdulaziz Alashaikh. Covid-19 open source data sets: a comprehensive survey. *Applied Intelligence*, 51(3):1296–1325, 2021.
- [81] SIIM, FISABIO, BIMCV, and RSNA. Siim-fisabio-rsna covid-19 detection, 2021. URL <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>.
- [82] Alessandro Sordoni, Philip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245, 2016. URL <http://arxiv.org/abs/1606.02245>.
- [83] Sergii Stirenko, Yuriy Kochura, Oleg Alienin, Oleksandr Rokovy, Yuri Gordienko, Peng Gang, and Wei Zeng. Chest x-ray analysis of tuberculosis by deep learning with segmentation and augmentation. In *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*, pages 422–428. IEEE, 2018.
- [84] Abu Sufian, Anirudha Ghosh, Ali Safaa Sadiq, and Florentin Smarandache. A survey on deep transfer learning to edge computing for mitigating the covid-19 pandemic. *Journal of Systems Architecture*, 108:101830, 2020.
- [85] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. *CoRR*, abs/1905.07799, 2019. URL <http://arxiv.org/abs/1905.07799>.
- [86] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer, 2018.
- [87] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.
- [88] Yi Tay, Anh Tuan Luu, Aston Zhang, Shuohang Wang, and Siu Cheung Hui. Compositional de-attention networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/16fc18d787294ad5171100e33d05d4e2-Paper.pdf>.
- [89] Nicolas Vabret, Graham J. Britton, Conor Gruber, Samarth Hegde, Joel Kim, Maria Kuksin, Rachel Levantovsky, Louise Malle, Alvaro Moreira, Matthew D. Park, Luisanna Pia, Emma Risson, Miriam Saffern, Bérengère Salomé, Myvizhi Esai Selvan, Matthew P. Spindler, Jessica Tan, Verena van der Heide, Jill K. Gregory, Konstantina Alexandropoulos, Nina Bhardwaj, Brian D. Brown, Benjamin Greenbaum, Zeynep H. Gümuş, Dirk Homann, Amir Horowitz, Alice O. Kamphorst, Maria A. Curotto de Lafaille, Saurabh Mehandru, Miriam Merad, Robert M. Samstein, Manasi Agrawal, Mark

- Aleynick, Meriem Belabed, Matthew Brown, Maria Casanova-Acebes, Jovani Catalan, Monica Centa, Andrew Charap, Andrew Chan, Steven T. Chen, Jonathan Chung, Cansu Cimen Bozkus, Evan Cody, Francesca Cossarini, Erica Dalla, Nicolas Fernandez, John Grout, Dan Fu Ruan, Pauline Hamon, Etienne Humblin, Divya Jha, Julia Kodysh, Andrew Leader, Matthew Lin, Katherine Lindblad, Daniel Lozano-Ojalvo, Gabrielle Lubitz, Assaf Magen, Zafar Mahmood, Gustavo Martinez-Delgado, Jaime Mateus-Tique, Elliot Meritt, Chang Moon, Justine Noel, Tim O'Donnell, Miyo Ota, Tamar Plitt, Venu Pothula, Jamie Redes, Ivan Reyes Torres, Mark Roberto, Alfonso R. Sanchez-Paulete, Joan Shang, Alessandra Soares Schanoski, Maria Suprun, Michelle Tran, Natalie Vaninov, C. Matthias Wilk, Julio Aguirre-Ghiso, Dusan Bogunovic, Judy Cho, Jeremiah Faith, Emilie Grasset, Peter Heeger, Ephraim Kenigsberg, Florian Krammer, and Uri Laserson. Immunology of covid-19: Current state of the science. *Immunity*, 52(6):910–941, 2020. doi: <https://doi.org/10.1016/j.jimmuni.2020.05.002>.
- [90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL <http://arxiv.org/abs/1706.03762>.
- [91] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL <http://arxiv.org/abs/1706.03762>.
- [92] Linda Wang, Zhong Qiu Lin, and Alexander Wong. Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):19549, Nov 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-76550-z. URL <https://doi.org/10.1038/s41598-020-76550-z>.
- [93] Linda Wang, Zhong Qiu Lin, and Alexander Wong. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):1–12, 2020.
- [94] Linda Wang, Alexander Wong, Zhong Qiu Lin, Paul McInnis, Audrey Chung, Hayden Gunraj, James Lee, Matt Ross, Blake VanBerlo, Ashkan Ebadi, Kim-Ann Git, and Abdul Al-Haimi. Actualmed covid-19 chest x-ray data initiative, 2020. URL <https://github.com/agchung/Actualmed-COVID-chestxray-dataset>.
- [95] Linda Wang, Alexander Wong, Zhong Qiu Lin, Paul McInnis, Audrey Chung, Hayden Gunraj, James Lee, Matt Ross, Blake VanBerlo, Ashkan Ebadi, Kim-Ann Git, and Abdul Al-Haimi. Figure 1 covid-19 chest x-ray data initiative, 2020. URL <https://github.com/agchung/Figure1-COVID-chestxray-dataset>.
- [96] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768, 2020. URL <https://arxiv.org/abs/2006.04768>.

- [97] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3462–3471, 2017. doi: 10.1109/CVPR.2017.369.
- [98] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2097–2106, 2017.
- [99] Lilian Weng. Attention? attention! *lilianweng.github.io/lil-log*, 2018. URL <http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>.
- [100] Ho Yuen Frank Wong, Hiu Yin Sonia Lam, Ambrose Ho-Tung Fong, Siu Ting Leung, Thomas Wing-Yan Chin, Christine Shing Yen Lo, Macy Mei-Sze Lui, Jonan Chun Yin Lee, Keith Wan-Hang Chiu, Tom Wai-Hin Chung, et al. Frequency and distribution of chest radiographic findings in patients positive for covid-19. *Radiology*, 296(2):E72–E78, 2020.
- [101] Chuhuan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fastformer: Additive attention can be all you need. *CoRR*, abs/2108.09084, 2021. URL <https://arxiv.org/abs/2108.09084>.
- [102] Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. Sharing attention weights for fast transformer. *CoRR*, abs/1906.11024, 2019. URL <https://arxiv.org/abs/1906.11024>.
- [103] Jingjing Tian Xinyu Weng, Nan Zhuang and Yingcheng Liu. Chexnet for classification and localization of thoracic diseases (an alternative implementation); consulted august 30, 2021. URL <https://github.com/arnoweng/CheXNet>.
- [104] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. *CoRR*, abs/2002.04745, 2020. URL <https://arxiv.org/abs/2002.04745>.
- [105] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. URL <http://arxiv.org/abs/1502.03044>.

- [106] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *NAAACL 2016*, pages 1480–1489, June 2016. URL <https://www.microsoft.com/en-us/research/publication/hierarchical-attention-networks-document-classification/>.
- [107] Aoxiao Zhong, Xiang Li, Dufan Wu, Hui Ren, Kyungsang Kim, Younggon Kim, Varun Buch, Nir Neumark, Bernardo Bizzo, Won Young Tak, et al. Deep metric learning-based image retrieval system for chest radiograph and its clinical applications in covid-19. *Medical Image Analysis*, 70:101993, 2021.