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
27 de octubre de 2021

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

Índice de figuras

Capítulo 1

Introduction

Capítulo 2

Tareas

2.1. Problemas

2.1.1. Estimación de Pose en Humanos

La tarea de *Estimación de Pose en Humanos* (HPE por sis siglas en inglés) ha sido uno de los tópicos de gran importancia en el campo de Visión por Computadora. Debido a la búsqueda de automatización y entendimiento de diversas actividades humanas, sus utilidades causan impacto directo en las implementaciones tecnológicas del mundo real, tales como, la predicción de intención (vigilancia), sistemas de autónomos y de asistencia en la conducción automóviles, animación, simulaciones, interacción Humano-Computadora (HCI), realidad virtual aumentada (VR y AR), videojuegos, salud o asistencia médica o hasta análisis de movimiento en deportes. La tarea de *Estimación de Pose* no solo se limita a el cuerpo humano, también, puede ser empleado en objetos como carros o animales, vease la imagen 2.1.

Con el crecimiento acelerado de *Aprendizaje Profundo* en los últimos años gracias a las capacidades actuales de potencia de cómputo los métodos basados bajo este enfoque han sobrepasado a las métodos tradicionales, sin embargo aún existen distintos problemas y retos que siguen presentes como la oclusión y la ambiguedad de los datos o la dificultad de su obtención para realizar entrenamientos.

El problema de *Estimación de Pose Humanos* consiste en predecir las partes del cuerpo o las posiciones de las articulaciones de una persona a través de una imagen, video. Este problema ha sido cuidadosamente estudiado a lo largo de los años y diversas recopilaciones de investigaciones han sido escritas. En la tabla 2.1 se resumen algunas de las más recientes y que describen dos formas generales de abordar el problema. La primera de ellas

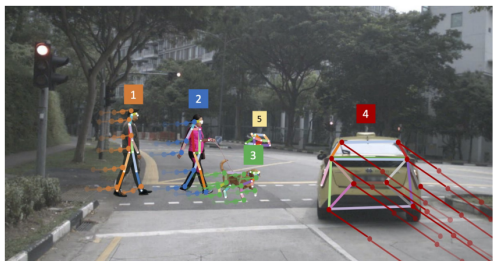


Figura 2.1: OpenPifPaf: Escena del mundo real desde la perspectiva de un carro autónomo. Todos los actores son detectados y seguidos, esto incluye a las personas, el carro y el perro. [44]

la "Tradicionalista", cuyos métodos usan enfoques clásicos de visión por computadora o la segunda basada en técnicas de aprendizaje profundo que involucran comúnmente modelos convolucionales. El trabajo realizado en esta tesis está basado en el segundo método, usando técnicas de aprendizaje profundo y modelos actuales capaces de capturar información temporal, específicamente enfocado en modelos *Transformers* [84].

Título	**Año**	**Métodos cubiertos**	**Descripción**
A survey of computer vision-based motion capture [57]	2001	Tradicionales	Investigacion general sobre métodos de captura de movimientos basados en visión en humanos. Incluye estimación de pose, seguimiento y reconocimiento de acciones.
A survey of advances in vision-based human motion capture and analysis [58]	2006	Tradicionales	Incluye una revisión de los métodos de captura de movimiento del año 2001 al 2006.
Vision-based human motion analysis: An overview. [66]	2007	Tradicionales	Investigacion general sobre métodos de captura de movimientos usando datos sin marcadores de dispositivos de captura.
Advances in view-invariant human motion analysis: A review [40]	2010	Tradicionales	Estudio de métodos de estimación de pose en 3D, comportamiento y reconocimiento/representación de acciones.
Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments [32]	2012	Tradicionales	Métodos de estimación de pose 3D y reconocimiento de acción usando datos multi-vista
A survey of human pose estimation: the body parts parsing based methods [49]	2015	Tradicionales	Estudios de estimación de pose enfocados principalmente a las técnicas de localización de las distintas partes del cuerpo.
Human pose estimation from monocular images: A comprehensive survey [27]	2016	Ambos	Enfocado en la estimación de pose usando datos monoculares incluyendo las metodologías usadas en procesos tradicionales y basados en aprendizaje profundo.
3D human pose estimation: A review of the literature and analysis of covariates [73]	2016	Deep-Learning	Revisión general del estado del arte de estimación de pose 3D usando imágenes y videos RGB.
Monocular human pose estimation: a survey of deep learning-based methods [12]	2020	Deep-Learning	Revisión y clasificación general de los métodos de estimación de pose basados en aprendizaje profundo desde el 2014 usando solo datos monoculares.
The progress of human pose estimation: a survey and taxonomy [59] of models applied in 2D human pose estimation	2020	Deep-Learning	Revisión de los métodos basados en aprendizaje profundo para estimación de pose 2D
Deep Learning-Based Human Pose Estimation: A Survey [95]	2020	Deep-Learning	Estudio general del estado del arte de estimación de pose 2D y 3D.

Cuadro 2.1: Listado de diversas investigaciones de *Estimación de Pose en Humanos* que abarcan tanto enfoques tradicionales como basados en aprendizaje profundo. Tabla basada en el trabajo de Zheng, Wu, Yang, Zhu, Chen, Liu, Shen, Kehtarnavaz, and Shah.

Taxonomía de Estimación de Pose 2D y 3D

Estimación de Pose en 2 y 3 dimensiones

Existen dos grandes grupos que dividen las metodologías seguidas para la *Estimación de Pose en Humanos*; estimación de pose en 2 y 3 dimensiones. Cómo el nombre lo sugiere, la *Estimación de Pose 2D* (**2D HPE** por sus siglas en inglés) consiste en localizar articulaciones

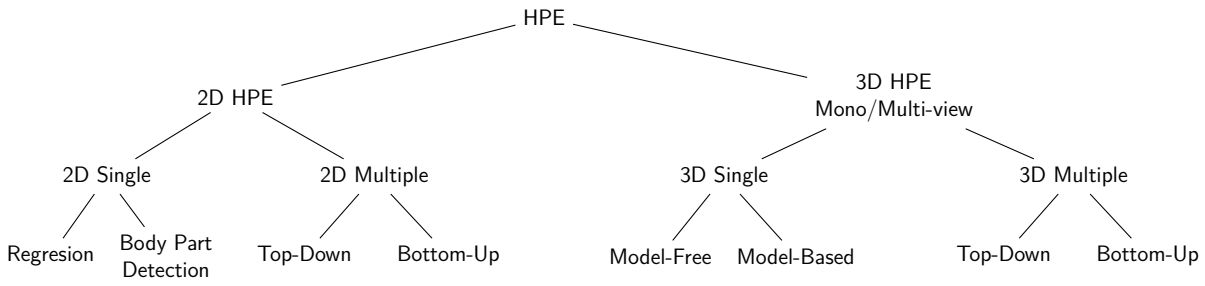


Figura 2.2: M1 caption for diagram

o partes del cuerpo directamente en imágenes, por tanto, el marco de referencia de las posiciones de cada articulación es la propia imagen. En la *Estimación de Pose 3D* (**3D HPE**) las elementos detectados pasan a estar en 3D y se busca un marco de referencia que mejor se ajuste las dimensiones espaciales de las articulaciones y personas. Comúnmente se usa un cubo unitario cuyo centro corresponde a la articulación que indica la cadera o *hip* como se encuentra en la literatura, véase la figura 2.2.

Por otro lado, en muchos escenarios las imágenes contienen más de una persona o se necesita hacer seguimiento de multiples individuos. Por lo regular cuando aparecen más de un persona en una imagen (**MPPE**, Multiple Person Pose Estimation) se opta por identificar cada cuerpo en la imagen y posteriormente resolver individualmente la estimación de pose para cada una de las entidades identificadas (**SPPE**, Single Pose Estimation). La detección de los cuerpos se realiza en etapas previas usando modelos de detección de objetos y entrenados para detectar cuerpos humanos tales como *MobileNet* [68] [34] [71] o *YOLO* (You Only Look Once) [67] [6].

Además, el proceso de estimación de pose puede ser realizado en multiples etapas. Es decir, un modelo end-to-end puede realizar la tarea completamente o en caso contrario, dividir la tarea en multiples etapas y usar modelos especializados para resolver cada etapa. Por ejemplo en estimación de pose 3D es común predecir la pose 2 dimensiones usando una red entrenada para esta tarea y posteriormente pasar a 3 dimensiones usando la información previa [53].

Estimación de Pose 2D Single.

Para la estimación de pose en 2D dónde solo es involucrada una sola personas se usan dos enfoques, los métodos basado en regresión (*Regresion-Based*) y los métodos basados en detección de partes del cuerpo humano ((*Detection-Based o Body Part Detection*)). Los métodos basados en regresión estiman las posiciones relacionando directamente la imagen con las coordenadas de las articulaciones del modelo de cuerpo humano usado. En cambio, los métodos basados de detección identifican primeramente las partes del cuerpo ya sea a través de marcas como recuadros de las posiciones o usando mapas de calor que indican las

posiciones de las articulaciones.

Estimación de Pose 3D Single.

Existen dos métodos generales para la estimación de pose 3D en una sola personas, los *Generativos* y los *Discriminativos*. Los métodos Generativos también conocidos como (*model-based*, basado en modelos) usan alguna representación de modelos del cuerpo humano en conjunto con información apriori como los movimientos y el contexto en el que se ejecutan. Las poses son predecidas usando la imágen un conjunto de representaciones obtenidos de establecer un función de probabilidad usando información tal como descriptores de la imagen, la estructura del cuerpo humano, los parámetros de las cámaras y y un conjunto de restricciones derivadas del contexto. Los modelos Discriminativos simplemente aprenden a predecir directamente la pose usando la datos de entrada sin usar información de modelos humanos.

Estimación de Pose 2D-3D Multiple.

Bottom-Up: En este enfoque se inicia localizando entidades semánticas y luego agrupandolas para formar una persona [89] [11] [36]. Claramente, usando este procedimiento el problema de rendimiento de usar un estimador de pose por cada persona desaparece, pues todas las entidades de los cuerpos son detectados a la vez y posteriormente agrupados para formar cada persona, véase la figura 2.3. Sin embargo, el modelo generado tiende a presentar problemas cuando existen personas ocluyéndose unas a otras. Uno de los trabajos más conocidos que siguen este marco es OpenPose [7], el cual se ha convertido en una completa herramienta de referencia para la estimación de pose logrando realizar tareas como seguimiento en tiempo real y detección de articulaciones en 3D en formato single-person con opción de triangularizar desde distintas vistas de cámaras, también es posible detectar en tiempo real la pose en 2 dimensiones tanto el cuerpo humano como gestos con manos y rostros.

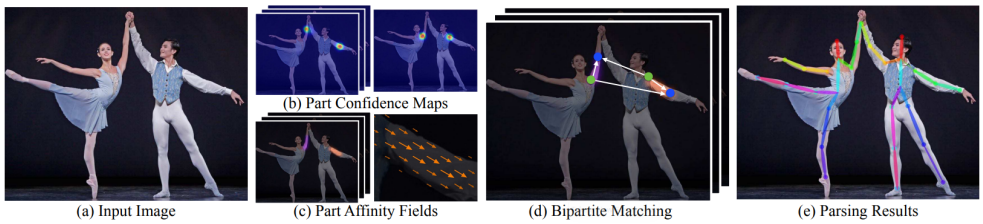


Figura 2.3: OpenPose: Trabaja bajo un marco Bottom-Pp. Primero predice los mapas de confianza para cada parte del cuerpo *b*) y codifican la orientación y ubicación de las extremidades a través *b*) y *c*) y finalmente asocian cada miembro identificado y reconstruyen la pose del cuerpo *d*) y *e*) [7].

Top-Down: El procesamiento se realiza primero detectando los personas individualmente en la imagen usando un bounding-box proporcionado por algun detector de objetos [60]

[86], véase la figura 2.5. El principal problema de este enfoque es que si la detección de la persona falla ya no hay nada más que hacer, y el costo computacional depende de la cantidad de personas en la imagen, puesto que para cada persona detectada es necesario correr un estimador de pose entrenado para detectar una sola persona. En contraparte a Openpose, Alpha-Pose [23] sigue el un marco top-down a través de 3 componentes esenciales; *Symmetric Spatial Transformer Network* (SSTN), *Parametric Pose Non-Maximum-Suppression*(NMS) y *Pose-Guided Proposals Generator* (PGPG) que les permiten minimizar el problema de la detección incorrecta o redundante de bounding-boxes de las personas.

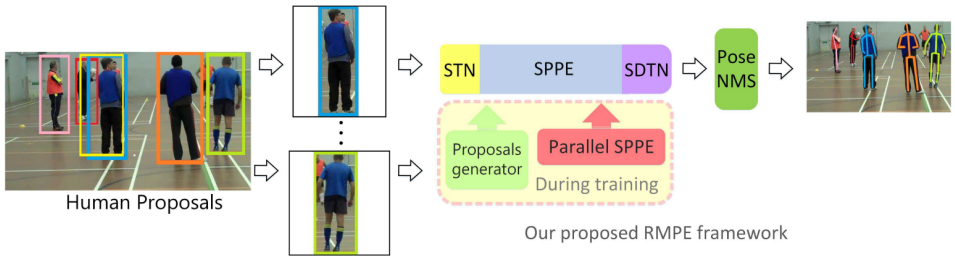


Figura 2.4: AlphaPose (RMPE): Trabaja bajo un marco Up-Down. Consiste en 3 principales componentes; El primero, *Symmetric Spatial Transformer Network* (SSTN) recibe los imágenes de las poses a procesar y genera propuestas de poses el *Parametric Pose Non-Maximum-Suppression*(NMS) se encarga de eliminar las redundancias e inconsistencias y finalmente el *Pose-Guided Proposals Generator* (PGPG) es usado como un modelo de aumentación de datos [23].

La estimación de pose en 3d también puede ser realizado en dos marcos, *monocular* cuando solo se tiene una imagen de reference del sujeto de prueba en un tiempo y pose exacta [53] [63] [10] [33] [13] y multi-vista cuando se tienen imágenes desde diferentes perspectivas del sujeto de prueba en misma la pose y tiempo exacto [38] [21] [81]. La ventaja de usar técnicas que puedan aprovechar la información contenida en diferentes perspectivas es que ayuda a reducir en gran medida la ambigüedad ocasionada por las oclusiones, una parte puede no ser visible desde un ángulo pero desde otro si. Sin embargo, la cantidad de conjunto de datos existentes para estimación de pose multi-vista es reducida.

Modelación del Cuerpo Humano

En la solución de los problemas de Estimación de Pose se plantea una arquitectura base para el cuerpo humano a partir de la cual se adecuarán las estimaciones. Los modelos usados para la representación del cuerpo humano son 3: *kinematic model*, *Planar Model* y *Volumetric Model*. véase la figura

Skeleton base model, stick figure o kinematic model: Es uno de los modelos más simples y mayormente usados. Consiste en grafo de nodos que representan las articulaciones del cuerpo humano, comúnmente entre 10 y 30 nodos [24]. Así, un hueso es representado como

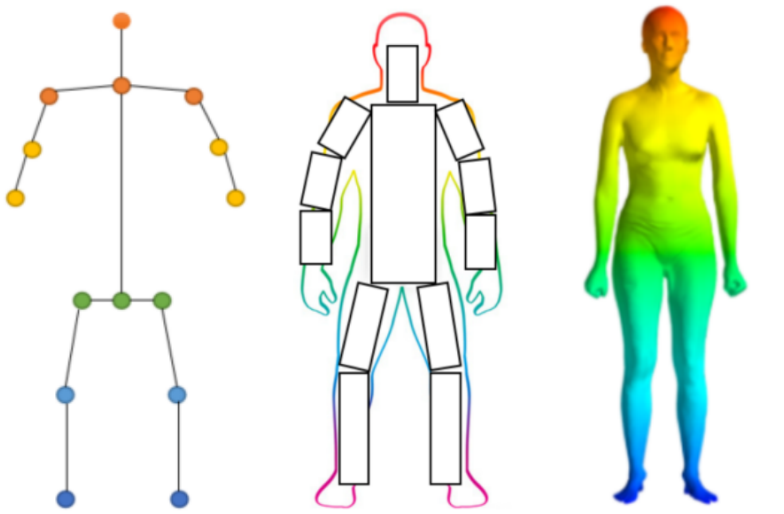


Figura 2.5: Representaciones de la arquitectura del cuerpo humano. Imagen obtenida de Fang et al.

una conexión entre dos articulaciones. Al ser solo una representación de la estructura del cuerpo carece de demás información como texturas o formas.

Contour-base model o Planar Model: Es uno de los primeros modelos usados para problemas de Estimación de Pose. Los miembros y torso del cuerpo humano son representados como un conjunto de rectángulos que proveen información tanto de los límites de cada miembros como sus longitudes y anchor [41] [19].

Volumen base model: Es usado para representar el cuerpo humano y su volumen, es decir es una representación 3D del cuerpo conseguida a través de un mallado de figuras geométricas y capturados por escaner 3D [75].

Tipos de datos.

En los últimos años, debido a la masificación de dispositivos inteligentes, el acceso a una cámara digital no resulta un problema mayor para la muchas de las personas, así, la mayoría de los trabajos realizados sobre Estimación de Pose usan *Imágenes RGB* gracias a su fácil captura y acceso. Sin embargo, esto solo es cierto en el contexto de la Estimación de Pose 2D, puesto que el etiquetado de datos usando las coordenadas de la imagen como referencia no representa demasiado problema. En la Estimación de Pose 3D, la complejidad de la obtención de los datos se incrementa. Para llevar a cabo el proceso de obtención de los datos es necesario usar equipo especializado y costoso para la captura de movimiento en acompañamiento de las cámaras de video [37]. Hay dos mecanismos de captura de movimiento,

los ópticos y los no ópticos. Generalmente, en los primeros el sujeto de prueba no necesita de aditamentos complejos como el uso de trajes (exoesqueleto) y solo un software con ayuda de cámaras, sensores y marcadores se encargan de registrar e interpretar el movimiento a un modelo digital. Si bien aventajan en una reducción de coste su precisión es menor que los no ópticos. El *Kinect*, fabricado por *Microsoft* es uno de los dispositivos más usados para generar imágenes de infrarrojos (*IR-image*) siendo de fácil encontrarlo y adquirirlo gracias a su bajo conste. [79] [39].

Por otra parte, existen dispositivos basados en tecnologías como *LIDAR* cuya función es generar imágenes de profundad. Al igual que el *Kinect*, cada vez son de más fácil acceso debido a su comercialización en dispositivos inteligentes en el último año por compañías como Apple [96]. El segundo tipo se puede dividir en dos clases; los mecánicos los cuales usan giroscopios y acelerómetros para registrar el movimiento y los electromagnéticos, cuyo funcionamiento es a través de la generación un campo electromagnético y para posteriormente capturar las alteraciones en este que realiza el sujeto al moverse [69].

Conjuntos de Datos más usados para estimación de pose 2D y 3D

MPII Human Pose Dataset: Es un dataset para estimación de pose en modalidades de single-person o multi-person en 2 dimensiones [2]. Incluye al rededor de 25 mil imágenes conteniendo 40 mil de ellas de personas con anotaciones de las articulaciones del cuerpo realizando diversas actividades. En total se agrupan en en 410 actividades recolectadas de videos de *YouTube*.

Microsoft COCO Dataset: Es un dataset publicado por *Microsoft* para tareas de detección de objetos, segmentación y subtitulado de imágenes (image captioning), su última version disponible corresponde a la del año 2017 [47]. Contiene al rededor de 80 categorías de imágenes y de estas 66,808 mil son de personas con un aproximado de 250 mil con total de aproximado de 273,469 anotaciones de cuerpos humanos. Sin embargo, No todas las imágenes contienen anotaciones de las 17 articulaciones, por lo que los modelos tienen que predecir cuales y cuántas articulaciones están presentes. Ambas modalidades single-person y multi-person en 2 dimensiones están disponibles en las imágenes.

HumanEva Dataset: Dataset usado para estimación de pose en 3 dimensiones en modalidad de single-person. Contiene diversos secuencias de videos grabadas con cámaras en formato RGB y escala de grises. Está compuesto de dos sub-datasets *HumanEva I* y *HumanEva II* la principal diferencia es el sistema de captura, el primero fue a través de software marcadores con 6 cámaras y el segundo a través de hardware con 8 cámaras, ambos dentro de un ambiente controlado.

Human3.6M Dataset: Contiene aproximadamente 3.6 millones de imágenes de poses humanas correspondientes a la cantidad de frames de secuencias de videos sobre 11 diferentes actores, 6 hombres y 3 mujeres realizando 17 distintas actividades [37]. Cada video fue realizado a usando un total de 10 cámaras de motion capture en un ambiente controlado

interno.

TotalCapture Dataset: Similar a Human3.6M, contiene contiene aproximadamente 1.9 millones de frames en videos en modalidad single-person y multi-vista calibrado a través de 8 cámaras.

Conjuntos de Datos Sinteticos

SURREAL (Synthetic hUmans foR REAL tasks) (2017): Es un dataset en modalidad single-person para estimación de pose 2D y 3D. Los datos originales son tomados del dataset Human3.6M y aleatoreamente muestreados (pose de la persona, apariencia, luz ambiental, posición de la cámara, tipos de fondos en las imágenes, texturas, etc.) para crear sintéticamente personas y escenas [82].

JTA Dataset (2018): Dataset sumamente grande creado a partir de simulaciones de *Grand Theft Auto V* desarrollado por *Rockstar North* [22] para estimación de poses 2D y 3D. Contiene al rededor de 500 mil frames y 10 millones de poses de poses en escenarios urbanos todos ellos con anotaciones completas de posiciones en 3D.

A pesar de que los datasets sintéticos son mucho grandes que los otros, actualmente no son tan aceptados en la comunidad y su uso benchmark como benchmark no es visto principalmente dado que son datos sinteticos. Sin embargo puesto que en los datos sintéticos se tiene mucho mayor control del ambiente puede solventar varias de las debilidades de los otros datasets como oclusiones, cambios de luz, tipos y colores de ropa, distintos contextos de fondos de imagen llevando modelos más robustos y con mejor generalización. La mayoría de la bases de datos son obtenidas a través de algunos pocos sujetos de prueba y la estructura fisiológica de individuos no es perfecta. Esta variación es causada por diversos diversos factores como el sexo, la raza, edad, lugar de nacimiento y desarrollo, enfermedades, factores genéticos, entre otros, además de que los movimientos recreados entre sujetos no siempre son hechos de la misma manera aunque las circunstancias o ambiente esté controlado.

Con los datos son obtenidos bajo un ambiente controlado es posible obtener imágenes fieles para el entrenamiento. Aún así, los modelos al ser desplegados en ambientes reales se enfrentan a imprevistos de los que no se puede tener control; oclusiones de diversas partes del cuerpo que no fueron provistas durante el entrenamiento ya sea por el mismo sujeto o algún objeto extraño en la captura, movimientos extraños o rápidos como correr o dar una patada donde el modelo no puede los puede identificar o el equipo de captura no pueda obtener que se ven como fotogramas borrosos.

Métricas de Evaluación

Percentage of Correct Parts (PCP): Mide la tasa de detección de extremidades. Una extremidad o parte de un cuerpo es considerada detectada si el promedio de la distancia

de las posiciones entre dos articulaciones predichas y la distancia de las posiciones de las articulaciones de la extremidad real es menor que cierto umbral [25]. El umbral comúnmente tomado corresponde al 50 % de la distancia de la longitud de la extremidad real en cuestion:

$$\frac{||c_s^{(n)} - \hat{c}_s^{(n)}|| + ||c_e^{(n)} - \hat{c}_e^{(n)}||}{2} \leq \alpha \; ||c_s^{(n)} - c_e^{(n)}|| \tag{2.1}$$

$||c_s^{(n)}$ y $||c_e^{(n)}$ representan las coordenadas de las dos articulaciones (inicial y final respectivamente) de la n-ésima extremidad. $||\hat{c}_s^{(n)}$ y $||\hat{c}_e^{(n)}$ son las coordenadas inicial y final de las dos articulaciones predichas de la n-ésima extremidad. α funciona como el umbral de error. Actualmente, ya no se usa esta métrica debido a que penaliza mayormente las partes del cuerpo más pequeñas. Mientras mayor es su *PCP* mejor es el modelo.

Percentage of Detected Joints (PDJ): Esta métrica fue propuesta para sobrellevar la limitante antes mencionada de *PCP*. Mide la tasa de detección de articulaciones del cuerpo. Una articulación es correctamente detectada si la distancia entre la posición de la articulación detectada y la posición real de la articulación está dentro de cierta fracción del diámetro del torso, es decir la distancia entre la cadera derecha (right hip) y el hombro izquierdo (left shoulder) [72] [80]:

$$||c^{(i)} - \hat{c}^{(i)}|| \leq \alpha \; ||c_{rh} - c_{ls}|| \tag{2.2}$$

$c^{(i)}$ es la coordenada de la i-ésima articulación y $\hat{c}^{(i)}$ es la coordenada de la articulación predicha correspondiente. c_{rh} es la coordenada de la cadera derecha y c_{ls} es la coordenada del hombro izquierdo. El problema con esta métrica es que cuando la persona se es capturada de lado en una imagen 2D el diámetro del torso tiende a ser cero así como la distancia entre la cadera derecha e izquierda.

Percentage of Correct Key-points (PCK): Es similar a la métrica *PDJ* pero en vez de tomar el diámetro del torso se toma la distancia de la diagonal del rectangulo externo que rodea todas las articulaciones del cuerpo [93].

$$||c^{(i)} - \hat{c}^{(i)}|| \leq \alpha \; dia g_{bbox} \tag{2.3}$$

La ecuación 2.3 correspondiente al cálculo de la detección correcta de articulación para la métrica de *PCK* es similar a *PDJ*. Una tercer variación es considerando una proporción de la longitud del segmente de la cabeza como umbral [1], *head-normalized probability of the correct keypoint* o *PCKh*, con la finalidad de tener un umbral independiente de las distancias de las articulaciones y una posible elección es usar el tamaño de la cabeza del sujeto a prueba.

Average Precision (AP) y Average Recall (AR): Inicialmente introducido como *Average Precision of Keypoints (APK)* [93] mide la exactitud y rendimiento de la detección de las articulaciones de acuerdo a la proporción de verdaderos positivos sobre el total de positivos detectados (precision) $\frac{TP}{TP+FP}$ y la proporción de verdaderos positivos sobre el total de positivos (recall) $\frac{TP}{TP+FN}$ penalizando tanto detecciones no encontradas como falsas detecciones. Al igual que los anteriores hay variantes como Mean Average Precision (mAP) que es la media de la precisión del modelo para todas las clase. Todos basados en alguna medida de similaridad como la propuesta en *Object Key-points Similarity (OKS)* que mide el promedio de cercania de las articulaciones predichas y las reales. Definen una similaridad entre articulaciones (Keypoint Similarity) [47] como la distancia entre las articulaciones predichas normalizadas por la escala del área que forma la persona y una constante de regularización determinada para cada articulación:

$$KS = \exp\left(-\frac{||c^{(n)} - \hat{c}^{(n)}||^2}{2s^2k_n^2}\right) \tag{2.4}$$

s y k_n corresponden al factor de escala equivalente a la raíz cuadrada del área segmentada del objeto y a la constante de regularización por cada tipo de articulación cuya su función conjunta es regular la importancia de cada articulación.

Mean Per Joint Position Error (MPJPE): Calcula el error en milímetros determinado por la distancia euclidiana entre los articulaciones predichas y las reales sobre cada tipo de articulación de la imagen o imágenes en caso de videos. Para distintos datasets de predicción de pose en 3D existen más de un pre-procesamiento antes calcular el error *MPJPE* conocidos como protocolos. Por ejemplo para *Human3.6M* el Protocolo #1 consiste en alinear las coordenadas de las articulaciones con respecto a la raíz, generalmente la que corresponde al centro de la cadera (hip). El Protocolo #2 calcula el error después de realizar un alineamiento mediante una transformación rígida usando *Procrustes Analysis* [29], también abreviado como *P-MPJPE*.

$$MPJPE = \frac{1}{M}\sum_{i=1}^{M}||c^{(i)} - \hat{c}^{(i)}|| \tag{2.5}$$

M corresponde a el total de articulaciones. Generalmente el error es calculado por tipo, clase o acción representado en los videos.

Al igual que en estimación de pose 2D la métrica de PCK o 3DPKC es usada para estimación de pose en 3D, al igual que su area bajo la Curva (AUC) considerando típicamente un umbral de 150mm que se aproxima a la mitad del tamaño de la cabeza [56]. Por otro lado, datasets usados para estimación de pose 3D son basados en secuencias de videos y una variante a *MPJPE* es introducida [63] para considerar la suavidad de las transiciones

entre poses midiendo la velocidad de las articulaciones (en milímetros por segundo), *Mean Per Join Velocity Error (MPJVE)*, que corresponde al error *MPJPE* primer derivada de las secuencias de pose 3D.

Mean Joint Angle Error (MJAE): Es similar al error *MPJPE* pero usando los ángulos de cada articulación. Mide el error como promedio sobre todos los ángulos de la diferencia absoluta entre la articulación estimada y la real de acuerdo a su posición angular:

$$MPJPE = \frac{1}{M}\sum_{i=1}^{M} |(c^{(i)} - \hat{c}^{(i)}) \quad \text{mód } \pm 180^{\circ}| \tag{2.6}$$

2.1.2. Lung Pathologies detection

Capítulo 3

Transformers

3.1. De RNN's a Transformers

Las **Redes Neuronales Recurrentes** o **RNN** (por sus siglas en Inglés) basadas en el trabajo de Rumelhart, Hinton, and Williams 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 3.1 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.

$$h^{(t)} = f(x^{(t)}, h^{(t-1)}; \theta) \tag{3.1}$$

En una *RNN* vista como un *gráfo computacional dirigído 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 3.1.

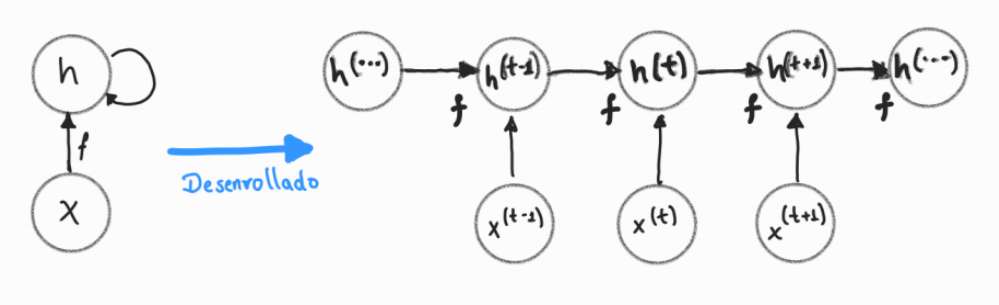
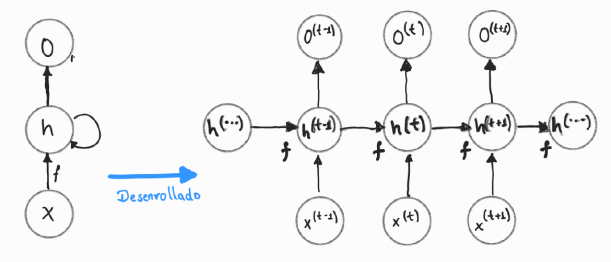


Figura 3.1: Grafo computacional generado por una *RNN* al "desenrollar" la recurrencia. Usando los parámetros compartidos en cada nodo y con cada elemento $x^{(t)}$ de la secuencia genera un nuevo estado oculto $h^{(t)}$ para retroalimentar nuevamente la entrada del siguiente nodo.

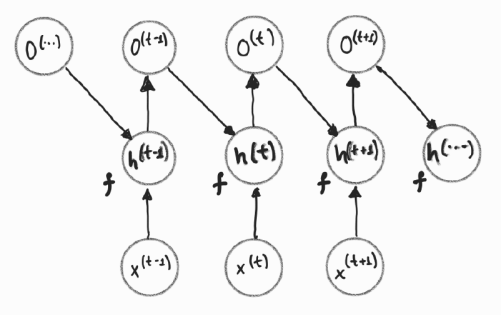
3.1.1. Redes Neuronales Recurrentes más comunes

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 3.2a. 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 3.2b 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 $h^{(t-1)}, h^{(t-2)}, ...$ no se transmita. La diferencia entre los dos esquemas anteriores es que la red 3.2a es entrenada para decidir que información debe transmitir en el futuro a través de los estados ocultos, en cambio, en la figura 3.2b cada estado esta 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.

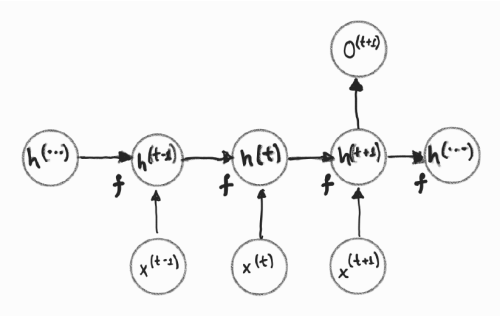
Por otro lado, la *RNN* representada en la figura 3.2c tiene una sola salida al final de la recurrencia. Al contrario de las anteriores, este tipo de redes pueden ser usadas para resumir información contenida en la secuencia para finalmente predecir un único valor final. El *Análisis de Sentimiento* en textos es una tarea común que puede ser representada con este esquema de red. En la figura 3.2d 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.



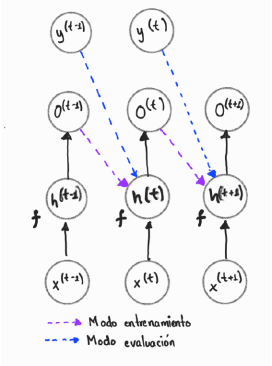
(a) Las activaciones de las capas ocultas $h^{(t)}$ alimentan al nodo siguiente, cada etapa de la recurrencia genera una salida $o^{(t)}$.



(b) Cada nodo es alimentado por las salidas de cada estado oculto $o^{(t)}$.



(c) Al final de la recurrencia solo tiene una salida $o^{(T)}$, puede ser usada para resumir/predecir un valor de una secuencia.



(d) Teacher Forcing. En modo de entrenamiento cada nodo en el tiempo t es retroalimentado por la salida correcta $y^{(t-1)}$, en modo evaluación es retroalimentado por las salidas del modelo $O^{(t-1)}$.

Figura 3.2: Distintos tipos de RNNs.

Finalmente, en la figura 3.3 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.

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 $x^{(1)}, x^{(2)}, ..., x^{(T)}$ con $x \in \mathcal{R}^{1\times d}$ donde d es la dimensión de la codificación de las palabras, la

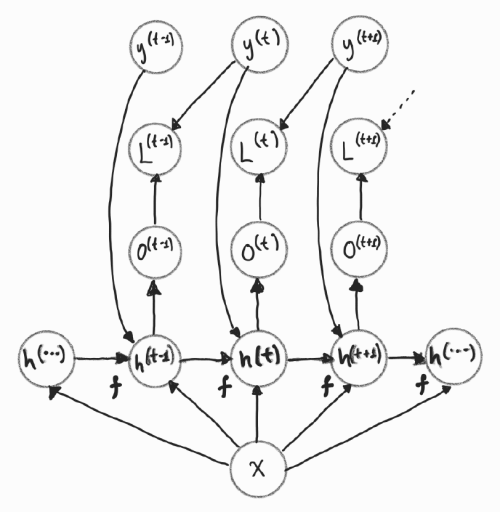


Figura 3.3: Modelo usado para tareas de *Image Captioning*, la entrada es una sola imagen y la red predice una secuencia de palabras que describen dicha imagen La salida esperada $y^{(t)}$ sirve como objetivo para la función de costo del paso anterior y como entrada en cada paso.

meta es predecir la probabilidad conjunta de la secuencia:

$$P(x^{(1)}, x^{(2)}, ..., x^{(T)}) = \prod_{t=1}^{T} P(x^{(t)}|x^{(t)}, \ldots, x^{(t-1)}) \tag{3.2}$$

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:

$$\hat{x}^{(t)} \approx P(x^{(t)}|x^{(t-1)}, \ldots, x^{(1)}) \approx P(x^{(t)}|h^{(t-1)}) \tag{3.3}$$

donde $h^{(t-1)}$ es el estado oculto que almacena la información pasada hasta el tiempo t tal y como se definió en 3.1.

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** [74].

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 3.4. Por lo cual, la salida $o^{(t)}$ ahora puede ser calculada con información tanto del futuro como del pasado 3.5.

$$\begin{aligned} \overrightarrow{h}^{(t)} &= f(x^{(t)}, \overrightarrow{h}^{(t-1)}; \theta_f) \\ \overleftarrow{h}^{(t)} &= f(x^{(t)}, \overleftarrow{h}^{(t+1)}; \theta_b) \\ h^{(t)} &= Concat(\overrightarrow{h}^{(t)}, \overleftarrow{h}^{(t)}) \end{aligned} \tag{3.4}$$

$$o^{(t)} = g(h^{(t)}; \theta_{out}) \tag{3.5}$$

3.1.2. 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 3.6 y 3.7:

$$h^{(t)} = \phi(x^{(t)}W_x + h^{(t-1)}W_h + b) \tag{3.6}$$

$$o^{(t)} = x^{(t)}W_{out} + b \tag{3.7}$$

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 3.4 representa gráficamente la lógica usada para calcular los estados ocultos y las salidas de la red.

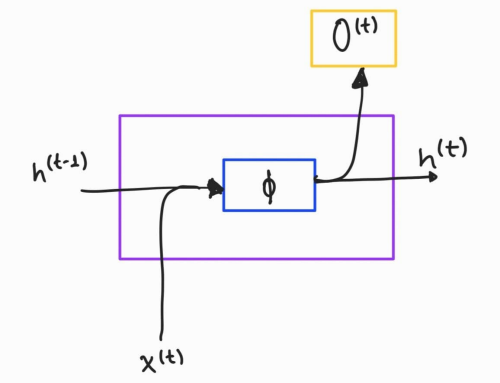


Figura 3.4: Computo del estado oculto y salida de una Red Neuronal Recurrente.

Sin embargo el cálculo de los estados ocultos mediante 3.6 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 util 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 [5]. Aunado a problemas como el *Desvanecimiento o Explosión del Gradiente* [43] [62], 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 [28, 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 Compuertas**, 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 3.5) 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 desechada 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 $C^{(t)}$.

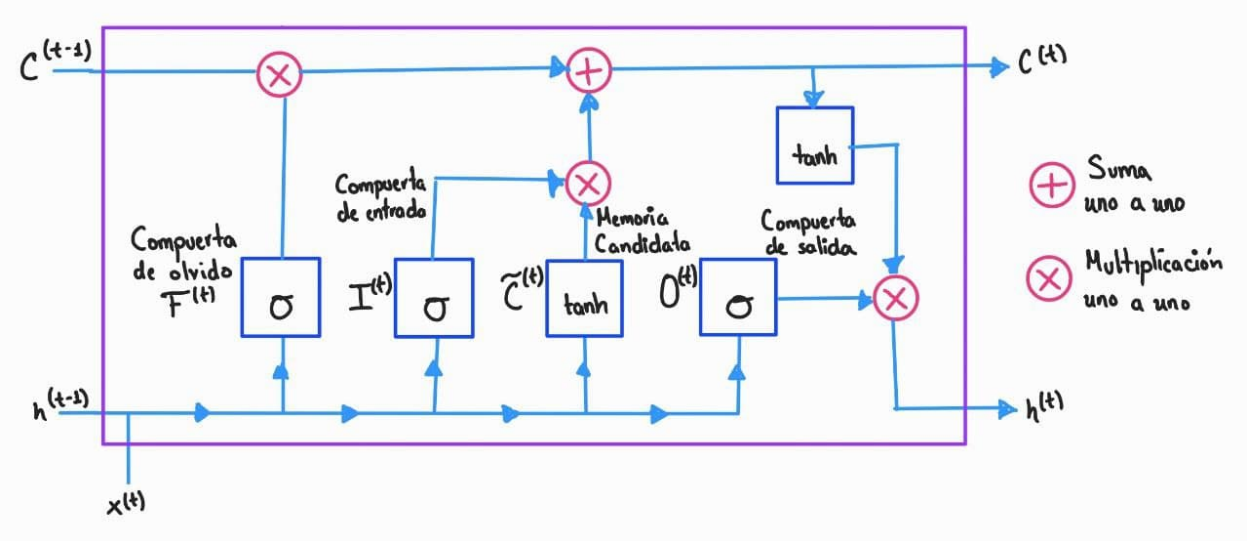


Figura 3.5: Descripción.

Las ecuaciones 3.8 rigen el comportamiento de Compuertas de Entrada, Salida y Olvido.

$$\begin{aligned} I^{(t)} &= \sigma(x^{(t)}W_{xi} + h^{(t-1)}W_{hi} + b_i) \\ F^{(t)} &= \sigma(x^{(t)}W_{xf} + h^{(t-1)}W_{hf} + b_f) \\ O^{(t)} &= \sigma(x^{(t)}W_{xo} + h^{(t-1)}W_{ho} + b_o) \end{aligned} \tag{3.8}$$

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 $b_i, b_f, b_o \in \mathbb{R}^{1xk}$

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

$$\begin{aligned} \tilde{C}^{(t)} &= \tanh\Big(x^{(t)}W_{xi} + h^{(t)}W_{hc} + b_c\Big) \\ C^{(t)} &= F^{(t)} \odot C^{(t-1)} + I^{(t)} \odot \tilde{C}^{(t)} \end{aligned} \tag{3.9}$$

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

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

$$h^{(t)} = O^{(t)} \odot \tanh\Big(C^{(t)}\Big) \tag{3.10}$$

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 [15] como una alternativa computacionalmente más rápida y con similar rendimiento que las *LSTM* [18]. 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 3.11. La *Compuerta de Olvido* permite decidir que del pasado aún debe ser transmitido a futuros estados o de otro modo ser desechada. 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 3.12.

$$\begin{aligned} R^{(t)} &= \sigma(x^{(t)}W_{xR} + h^{(t-1)}W_{hR} + b_R) \\ Z^{(t)} &= \sigma(x^{(t)}W_{xZ} + h^{(t-1)}W_{hZ} + b_Z) \end{aligned} \tag{3.11}$$

$$\begin{aligned} \tilde{h}^{(t)} &= \tanh\Big(x^{(t)}W_{xh} + (R^{(t)} \odot h^{(t-1)})W_{hh} + b_h\Big) \\ h^{(t)} &= Z^{(t)} \odot h^{(t-1)} + (1 - Z^{(t)}) \odot \tilde{h}^{(t-1)} \end{aligned} \tag{3.12}$$

3.1.3. Mecanismos de Atención

Una de las arquitecturas comunes vistas previamente es la mostrada en la figura 3.2c 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*) [16]. Los modelos *Seq2Seq* están formados por dos redes neuronales como la mostrada en 3.7. La primera se comporta como un *codificador* al resumir la entrada y producir un vector

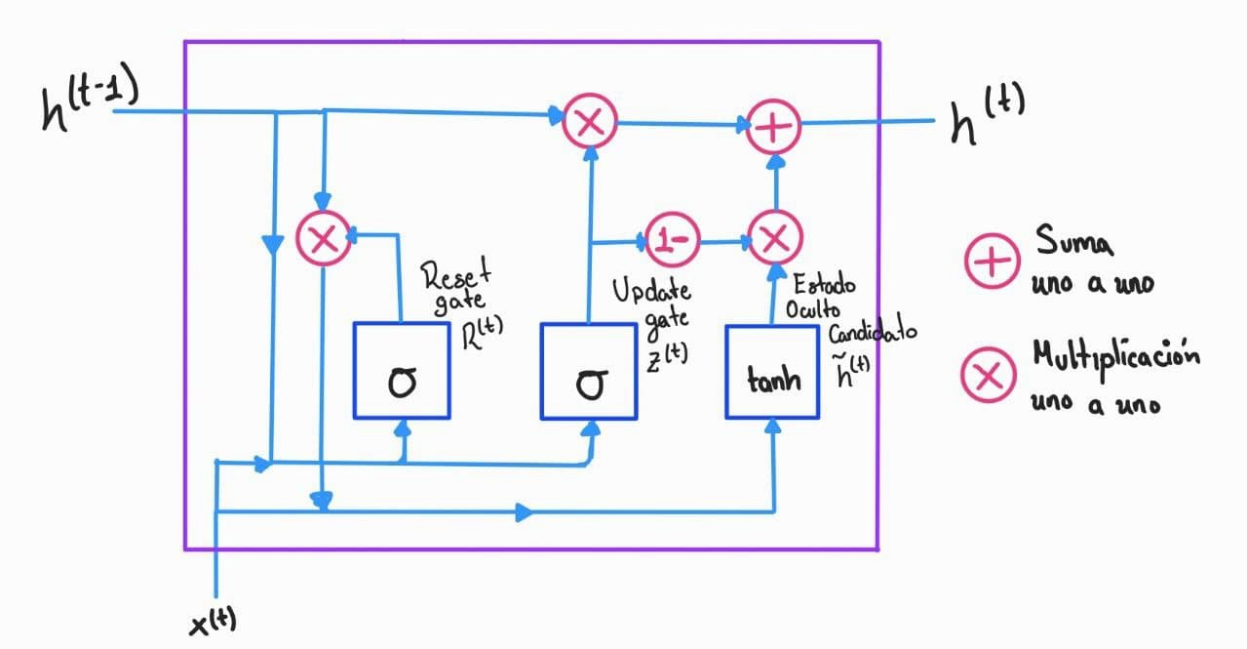


Figura 3.6: Descripción.

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.

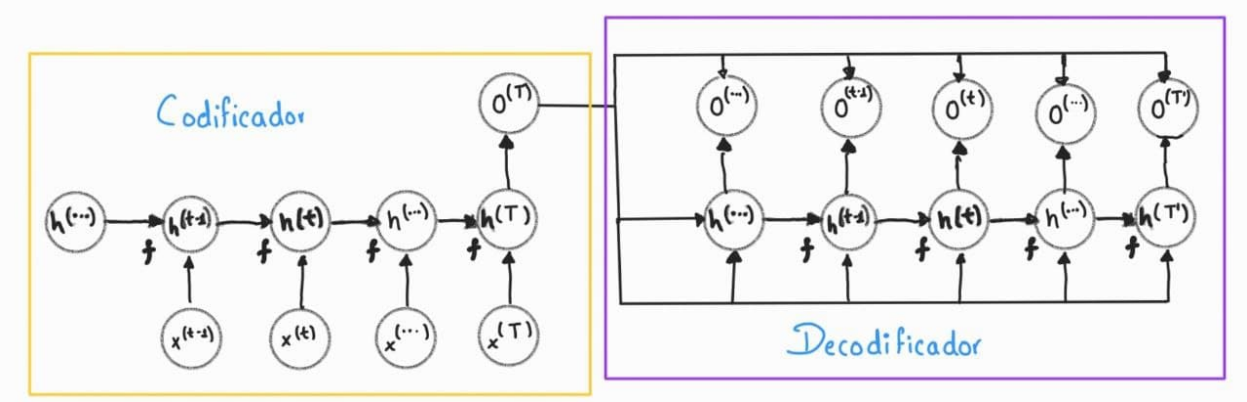


Figura 3.7: Descripción.

Por ejemplo, en tareas de *Machine Translation* el *codificador* esta formado por una *RNN* Bidireccional que lee y procesa un conjunto de vectores $X = (x^{(1)}, x^{(2)}, \ldots, x^{(T_x)})$ para obtener un vector de contexto C. La forma más común es como en 3.13:

$$\begin{aligned} h^{(t)} &= f_{bi}(x^{(t)}, h^{(t-1)}; \theta_f, \theta_b) \\ C &= q(h^{(1)}, h^{(2)}, \ldots, h^{(T)}) \end{aligned} \tag{3.13}$$

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 $q(h^{(1)}, h^{(2)}, \ldots, h^{(T)}) = h^{(T)}$, 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*:

$$p(Y) = \prod_{t=1}^{T_y} p(y^{(t)} | \{y^{(1)}, \ldots, y^{(t-1)}\}, C) \tag{3.14}$$

$$p(y^{(t)} | \{y^{(1)}, \ldots, y^{(t-1)}\}, C) = g(y^{(t-1)}, s^{(t)}, C; \theta_g) \tag{3.15}$$

donde g es una función no lineal que emite la probabilidad de $y^{(t)}$ y $s^{(t)}$ es el estado oculto del *decodificador* 3.16.

$$s^{(t)} = f(s^{(t)}, y^{(t-1)}, C; \theta_s) \tag{3.16}$$

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, Cho, and Bengio 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* $c^{(t)}$:

$$p(y^{(t)} | \{y^{(1)}, \ldots, y^{(t-1)}\}, c^{(t)}) = g(y^{(t-1)}, s^{(t)}, c^{(t)}; \theta_g) \tag{3.17}$$

$$s^{(t)} = f(s^{(t)}, y^{(t-1)}, c^{(t)}; \theta_s) \tag{3.18}$$

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:

$$c^{(t)} = \sum_{i=1}^{T_x} \alpha_{t,i} h^{(i)} \tag{3.19}$$

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 $s^{(i)}$.

$$\alpha_{t,i} = align(y^{(t)}, x^{(i)}) = \frac{\exp\Big(score(s^{(t-1)}, h^{(i)})\Big)}{\sum_{k=1}^{T_x} \exp(score(s^{(t-1)}, h^{(k)}))} \tag{3.20}$$

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

$$score(s^{(t)}, h^{(i)}) = v_a^\top \tanh\Big(W_a[s^{(t)}; h^{(i)}]\Big) \tag{3.21}$$

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 3.8 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 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:

$$A(q, K, V) = \sum_i p(a(K - i, q)) * v_i \tag{3.22}$$

En la ecuación 3.22, 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*

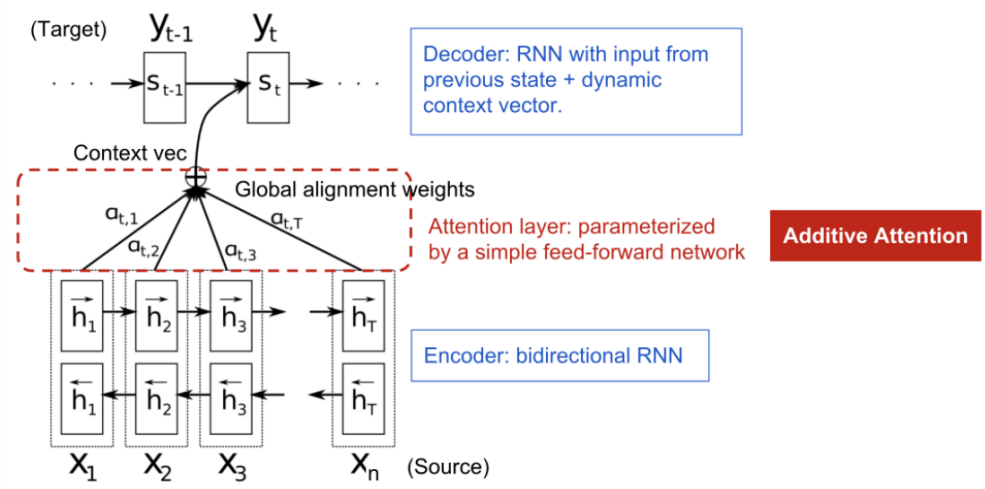


Figura 3.8: Modelo seq2seq propuesto por Bahdanau, Cho, and Bengio con *Additive/Concat Attention*

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, Treviso, Farinhas, Niculae, Figueiredo, and Aguiar permiten trabajar y enfocarse en solo algunas relaciones de alineamiento. Tay, Luu, Zhang, Wang, and Hui proponen una función de distribución de pesos $M = \tanh(E) \odot sigmoid(N)$ con E como una matrix en donde cada entrada representa la similaridad entre estados ocultos y N una medida negativa (disimilaridad), por lo que podemos usar $sigmoid(N)$ 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 3.20, lo que evita asumir que ambas representaciones están en el mismo espacio, como lo hace las funciones de alineación como el producto punto o la similaridad coseno. La tabla 3.1 muestra una recopilación de funciones de alineamiento.

De acuerdo a como es aplicado los distintos tipos de atención Chaudhari, Polatkan, Ramanath, and Mithal 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

Nombre	**Función de Alineación**	**Cita**
Similarity / Content-Base	$a(k_i, q) = sim(k_i, q)$	Graves, Wayne, and Danihelka
Dot Product[1]	$a(k_i, q) = q^\top k_i$	Luong, Pham, and Manning
Scaled Dot Product	$a(k_i, q) = \frac{q^\top k_i}{\sqrt{d_k}}$	Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin
General	$a(k_i, q) = q^\top W k_i$	Luong, Pham, and Manning
Biased General	$a(k_i, q) = k_i(Wq + b)$	Sordoni, Bachman, and Bengio
Activated General	$a(k_i, q) = act(q^\top W k_i + b)$	Ma, Li, Zhang, and Wang
Generalized Kernel	$a(k_i, q) = \phi(q)^\top \phi(k_i)$	Choromanski, Likhosherstov, Dohan, Song, Gane, Sarlós, Hawkins, Davis, Mohiuddin, Kaiser, Belanger, Colwell, and Weller
Additive\Concat [2]	$a(k_i, q) = v^\top act(W[q; k_i] + b)$	Bahdanau, Cho, and Bengio, Luong, Pham, and Manning
Deep	$a(k_i, q) = v^\top E^{(L-1)} + b^L$ $E(l) = act(W_l E^{(l-1)} + b^l)$ $E(1) = act(W_0 k_i + W_1 q + b^l)$	Pavlopoulos, Malakasiotis, and Androutsopoulos
Location-based	$a(k_i, q) = act(Wq)$	Luong, Pham, and Manning
Feature-based	$a(k_i, q) = v^\top act(W_0\phi(K) + W_1\phi(K) + b)$	Li, Kaiser, Bengio, and Si

Cuadro 3.1: Distintos tipos de funciones de alineación. (Tabla basada en [8] y [87]).
$a(k_i, q)$ representa la función de alineación entre k_i y q y act es una función de activación. sim es una función de similaridad, Graves, Wayne, and Danihelka 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, Cho, and Bengio es la función *tanh* como se ve en 3.21

secuencias de entrada y salida respectivamente, como es el caso del modelo propuesto por Bahdanau, Cho, and Bengio. 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 los palabras del texto que son más relevantes. El tercer tipo es **auto-atención** (*Self Attention*), fue propuesto por Yang, Yang, Dyer, He, Smola, and Hovy y es uno de los puntos claves para los modelos *Transformers* [83]. 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 conformar un texto más largo. Yang, Yang, Dyer, He, Smola, and Hovy utiliza este procedimiento para generar un vector de características usado posteriormente en un etapa de clasificación.

En la tercer categoría la atención es realizada en diversas partes de la secuencia; la suma pesada sobre todos los puntajes de las entradas usada por Bahdanau, Cho, and Bengio se le denomina **Atención suave** (*Soft-Attention*). Una alternativa es la **Atención dura** (*Hard-Attention*) [92] que calcula la atención no sobre todas los puntajes de alineamiento sino en una parte de estos, para ello se usa una distribución multinoulli 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, Pham, and Manning 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 atendera a diferentes representaciones y formar un vector final usando una combinación pesada de estos a través de dichos pesos de atención. Chaudhari, Polatkan, Ramanath, and Mithal 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 diversas 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.

3.2. 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 [84]. 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) 3.9, 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 $X = (x_1, x_2, \ldots, x_n)$ con n como el tamaño de la secuencia, el codificador produce una representación intermedia $Z = (z_1, z_2, \ldots, z_n)$ al igual que los modelos *seq2seq*. El decodificador usa la secuencia Z para generar la secuencia de salida $Y = (y_1, y_2, \ldots, y_m)$ 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.

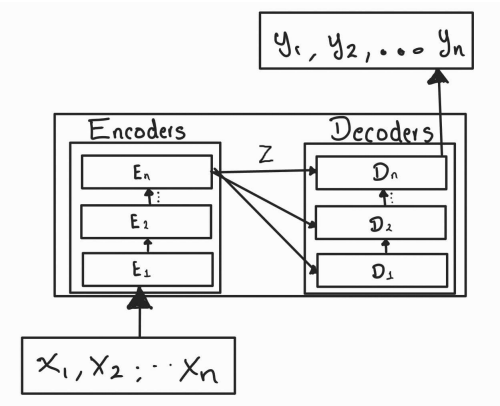
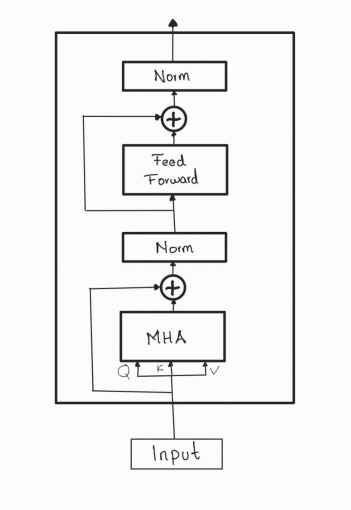


Figura 3.9: Modelo Transformer generalizado como modelo Secuencia a Secuencia

3.2.1. El Codificador y Decodificador

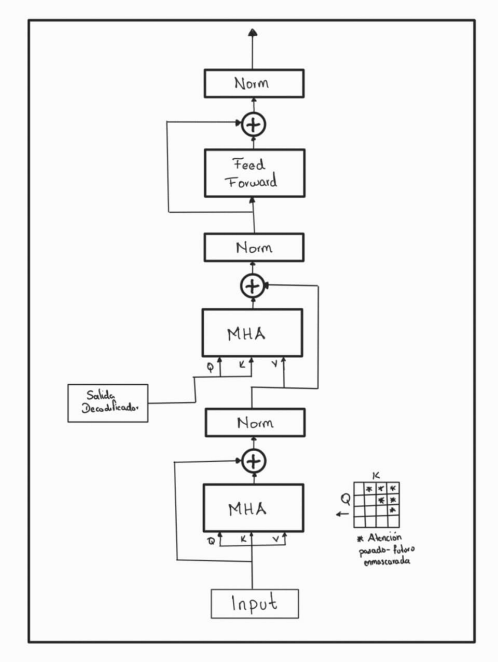
El *Modelo Transformer* está formado por multiples codificadores y decodificadores apilados e inter-conectados, Como observamos en la figura 3.9. El codificador consta de dos capas, la primera de ellas aplica *Self-Attention* múltiples veces sobre la misma entrada (*Multi-HeadSelf Attention*) y la segunda capa representada solo por una red *Feed-Forward* cuya entrada es la salida de la capa anterior. Véase la figura 3.10.

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 3.12.



$$mha = MHA(X, X, X)$$
$$norm = Norm(mha + X)$$
$$f = FeedForward(norm)$$
$$Encoder(X) = Norm(f + norm)$$

Figura 3.10: Etapa Codificadora del Modelo Transformer. Pseudocódigo

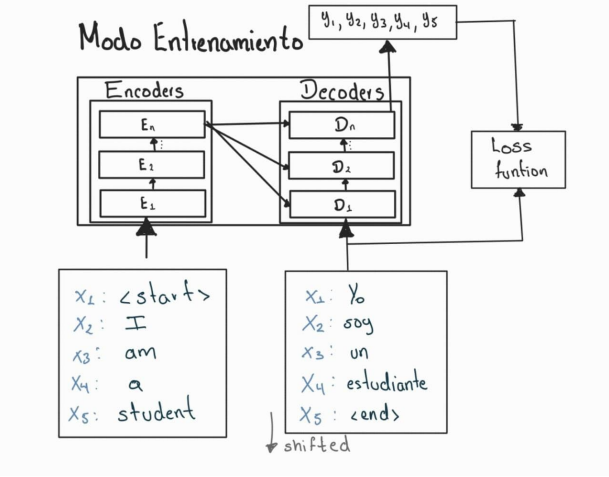


$$mha_1 = MHA(X, X, X)$$
$$norm_1 = Norm(mha_1 + X)$$
$$mha_2 = MHA(enc_{out}, enc_{out}, norm_1)$$
$$norm_2 = Norm(mha_2 + X)$$
$$f = FeedForward(norm_2)$$
$$decoder(X) = Norm(f + norm_2)$$

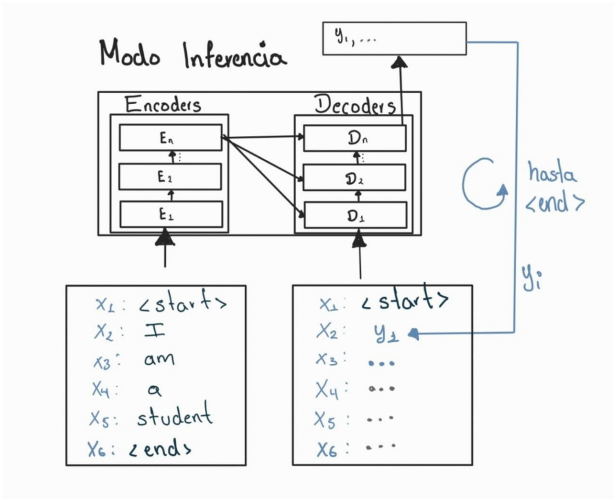
Figura 3.11: Etapa Decodificadora del Modelo Transformer. Pseudocódigo

3.2.2. Multi-Head Self-Attention

En la sección 3.1.3 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*[84]:



(a) Transformer modo entrenamiento. Las entradas en el decodificador son recorridas un elemento en el futuro, con el fin de que aprende a predecir la siguiente palabra dado un contexto previo y las salidas actuales en el momento de la evaluación.



(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 primer iteración el decoder solo recibe el token de inicio de oración $< start >$ por lo que podrá predecir la primer palabra de la oración gracias a que fue entrenado con un desplazamiento hacia el futuro. En la siguiente iteración el nuevo token predicho es agregado como entrada al decodificador. El decodificador termina su predicción en el momento que el token $< end >$ es obtenido.

Figura 3.12: Esquema de entrenamiento e inferencia del modelo Transformer en un problema de Machine Translation.

$$Attention(q, k, v) = softmax(\frac{qk^\top}{\sqrt{d_k}})v \tag{3.23}$$

El Transformer está basado en la idea de de aplicar atención multiples 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. 3.24

$$mha(Q, K, V) = Concat(head_1, head_2, head_3, ..., head_h)W^O \tag{3.24}$$

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 de 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 3.10 3.11. 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.

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:

$$\begin{aligned} Q_i &= E_Q W_i^Q \\ K_i &= E_K W_i^K \\ V_i &= E_V W_i^V \end{aligned} \tag{3.25}$$

$$head_i = Attention(Q_i, K_i, V_i) = softmax\Big(\frac{Q_i K_i^T}{\sqrt{d_k}}\Big) V_i \tag{3.26}$$

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[84].

3.2.3. 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* [26] 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 $10000 \cdot 2\pi$ 3.27:

$$\begin{aligned} PE(pos, 2i) &= \sin\Big(pos/10000^{2i/d_m}\Big) \\ PE(pos, 2i+1) &= \cos\Big(pos/10000^{2i/d_m}\Big) \end{aligned} \tag{3.27}$$

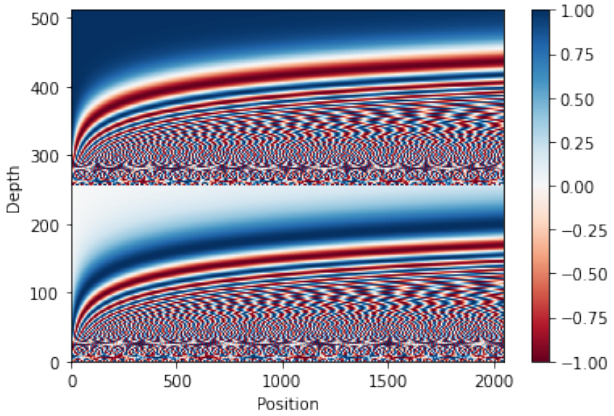


Figura 3.13: 2000 Vectores de Positional Encoding con dimensiones de embedding=500.

Poner una figura completa de todo el esquema del Transformer

3.2.4. 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 [65].

Learning Rate WarmUp es una de las primeras técnicas usadas y descritas en el proceso de entrenamiento por Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin. Usando el algoritmo Adam como optimizador se varía el factor de aprendizaje de acuerdo a la fórmula:

$$lrate = d_m^{-0,5} \cdot \text{mín}\left(step\_num^{-0,5}, step\_num \cdot warmup\_steps^{-0,5}\right) \tag{3.28}$$

En el esquema anterior, más conocido como *Noam-Warmup*, el modelo original es entrenado incrementando linealmente el factor de aprendizaje en los primeros $warmup\_steps = 4000$ pasos. Posteriormente, decrementa propocionalmente al inverso del la raíz cuadrada del paso *step_num* actual, véase la figura 3.14.

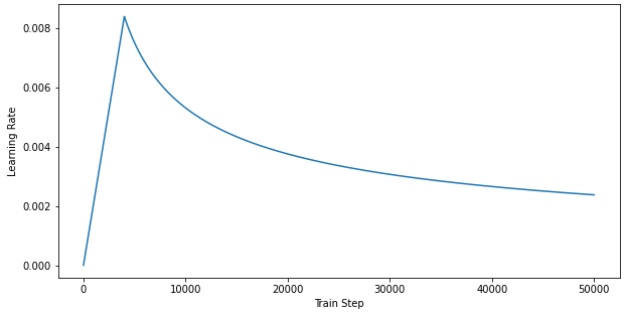


Figura 3.14: Noam-Warmup con $warmup\_steps = 4000$ y $d_m = 512$

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, Jiang, He, Chen, Liu, Gao, and Han 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, Jiang, He, Chen, Liu, Gao, and Han 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, replazando $\alpha_t = \alpha\omega_t$. La forma mas sencilla es usar un factor **linear warmup** parametrizado por un periodo de "calentamiento" τ.

$$\omega_t^{linear,\tau} = \text{mín}\left(1, \frac{t}{\tau}\right) \tag{3.29}$$

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

Exponential warmup aplica un decaimiento exponencial

$$\omega_t^{expo,\tau} = 1 - \exp\left(-\frac{1}{\tau}t\right) \tag{3.30}$$

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.

$$\omega_t^{expo,untuned} = 1 - \exp(-(1-\beta_2)t) \tag{3.31}$$

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.

$$\omega_t^{linear,untuned} = \text{mín}\left(1, \frac{1-\beta_2}{2}t\right) \tag{3.32}$$

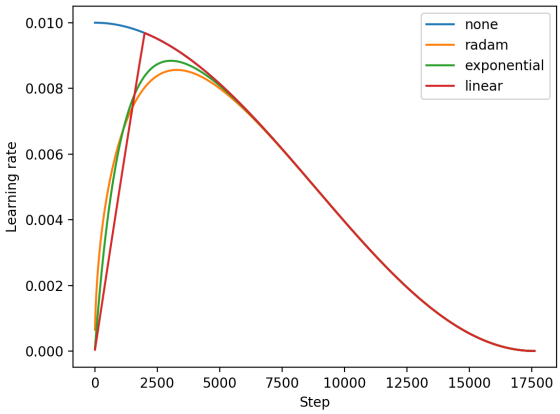


Figura 3.15: Learning rate sobre X 18000 iteraciones usando RAdam y lineal, exponencial warmup con Adam

Por otro lado, Huang, Perez, Ba, and Volkovs 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 [9] [91]. En particular Xiong, Yang, He, Zheng, Zheng,

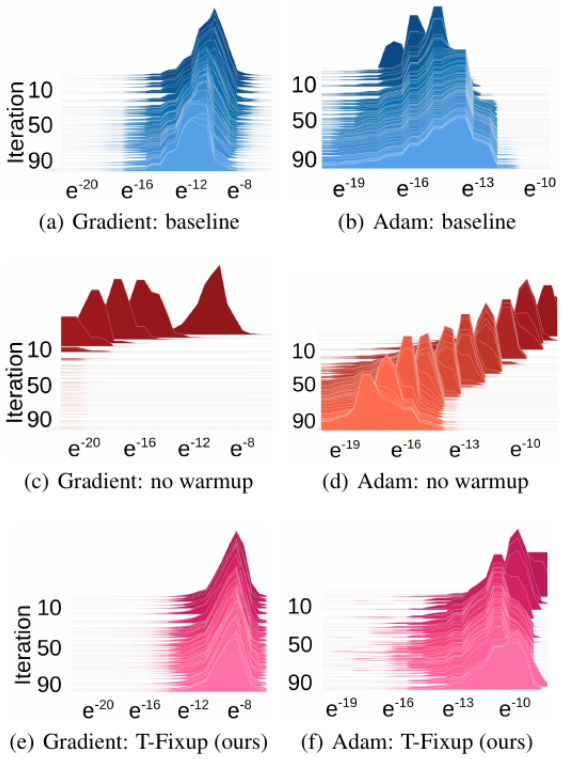
Xing, Zhang, Lan, Wang, and Liu encuentran que para un modelo Transformer de cualquier tamaño con capas de normalización entre bloques residuales (*Post-LN Transformer*), la escala de la norma del gradiente que incide en la última capa de normalización permanecen igual al no depender de la cantidad de 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, Perez, Ba, and Volkovs 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 3.16 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 $\mathbb{N}(0, d_m^{\frac{1}{2}})$ 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 $0{,}67N^{-\frac{1}{4}}$

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.



(a) Gradient: baseline (b) Adam: baseline

(c) Gradient: no warmup (d) Adam: no warmup

(e) Gradient: T-Fixup (ours) (f) Adam: T-Fixup (ours)

Figura 3.16: Histograma de gradientes del Algoritmo Adam con y sin etapa de *WarmUp* y usando inicialización *T-Fixup*. Imagen original de Huang, Perez, Ba, and Volkovs

Atención de vecindades:

Parmar, Vaswani, Uszkoreit, Kaiser, Shazeer, and Ku particionan la información de las representaciones de las consultas asignándolos a diferentes bloques de memoria, restringiéndose a vecindarios locales alrededor de cada consulta. Principalmente basados en cómo las redes convolucionales trabajar. Sin embargo esta solucion es parcial y solo aplicable a secuencias de datos con relaciones cortas, cómo imágenes.

Child, Gray, Radford, and Sutskever 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 3.17.

Sukhbaatar, Grave, Bojanowski, and Joulin 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

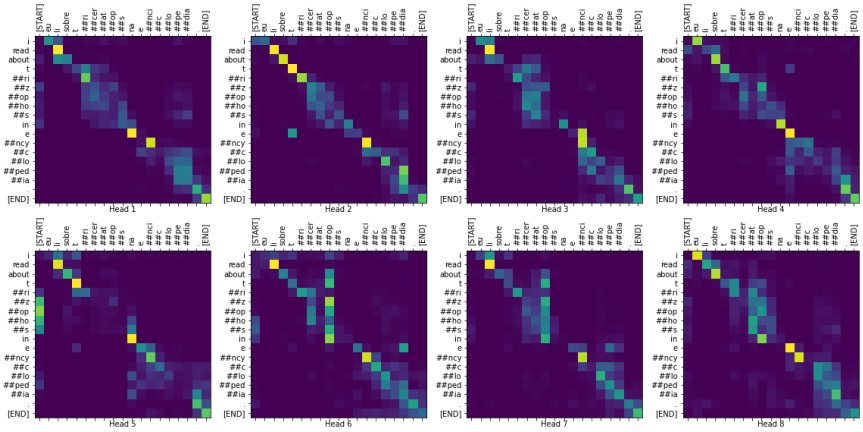


Figura 3.17: Visualización de 8 cabezas de atención sobre una tarea de Machine-Translation. Las matrices de atención tienden a ser ralas al tener carencia de relaciones relevantes entre diversas representaciones a distancias lejanas.

de enmascaramiento que controla la flexibilidad del ancho una ventana. La ventana formada cambia dinámicamente de tamaño dependiendo de la representación en cuestión. Beltagy, Peters, and Cohan siguen un estrategia similar, implementado 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, Yang, Yang, Carbonell, Le, and Salakhutdinov 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 3.18) un estado oculto es calculado de las secuencias 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.

Kitaev, Kaiser, and Levskaya 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

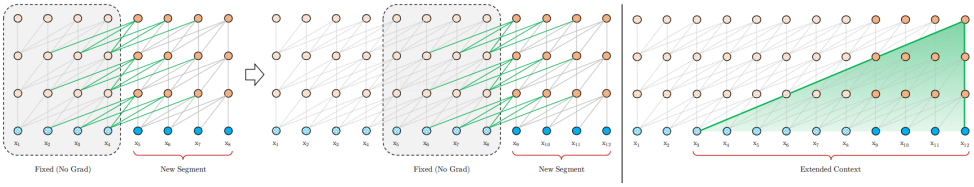


Figura 3.18: Transformer-XL. Para tratar con secuencias largas divide el proceso en secuencias más cortas creando estados ocultos intermedios y usandolos en el cálculo de las próximas secuencias. Figura obtenida de [20].

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 dificil si sus matrices de proyección W_i^Q y W_i^K fuesen muy distintas.

Aproximaciones a la Atención original:

También Xiao, Li, Zhu, Yu, and Liu 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, Li, Khabsa, Fang, and Ma 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 $k << n/$, 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, Likhosherstov, Dohan, Song, Gane, Sarlós, Hawkins, Davis, Mohiuddin, Kaiser, Belanger, Colwell, and Weller 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 $r <= n$. 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, Ainslie, Eckstein, and Ontañón 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, Wu, Qi, Huang, and Xie 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 3.19

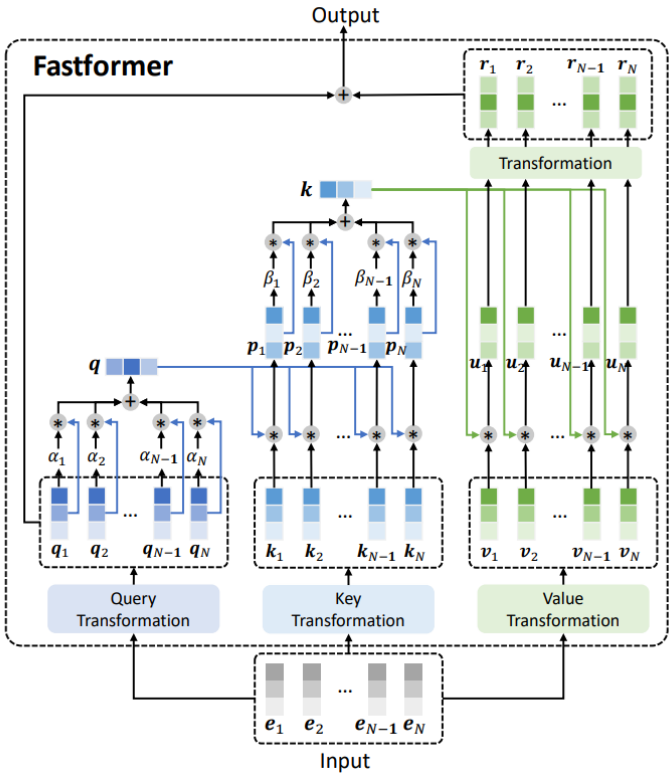


Figura 3.19: Fast-Former. Remplaza la atención tradicional del transformer por una iterativa. En cada paso crea una consulta y clave global usando atención sobre estos mismos. Figura obtenida de [88]

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 nothing, 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] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693, 2014. doi: 10.1109/CVPR.2014.471.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[4] 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`.

[5] 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.

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. URL `https://arxiv.org/abs/2004.10934`.

[7] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021. doi: 10.1109/TPAMI.2019.2929257.

[8] 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`.

[9] 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`.

[10] Tianlang Chen, Chen Fang, Xiaohui Shen, Yiheng Zhu, Zhili Chen, and Jiebo Luo. Anatomy-aware 3d human pose estimation in videos. *CoRR*, abs/2002.10322, 2020. URL `https://arxiv.org/abs/2002.10322`.

[11] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7103–7112, 2018. doi: 10.1109/CVPR.2018.00742.

[12] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192:102897, 2020. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2019.102897. URL `https://www.sciencedirect.com/science/article/pii/S1077314219301778`.

[13] Yu Cheng, Bo Yang, Bo Wang, and Robby T. Tan. 3d human pose estimation using spatio-temporal networks with explicit occlusion training. *CoRR*, abs/2004.11822, 2020. URL `https://arxiv.org/abs/2004.11822`.

[14] 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`.

[15] KyungHyun Cho, Bart van Merrienboer, 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`.

[16] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, 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`.

[17] Krzysztof Choromanski, Valerii Likhosherstov, 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`.

[18] Junyoung Chung, Çaglar Gülçehre, 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`.

[19] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59,

1995. ISSN 1077-3142. doi: https://doi.org/10.1006/cviu.1995.1004. URL https://www.sciencedirect.com/science/article/pii/S1077314285710041.

[20] 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.

[21] Junting Dong, Wen Jiang, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Fast and robust multi-person 3d pose estimation from multiple views. *CoRR*, abs/1901.04111, 2019. URL http://arxiv.org/abs/1901.04111.

[22] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Andrea Palazzi, Roberto Vezzani, and Rita Cucchiara. Learning to detect and track visible and occluded body joints in a virtual world. In *European Conference on Computer Vision (ECCV)*, 2018.

[23] Haoshu Fang, Shuqin Xie, and Cewu Lu. RMPE: regional multi-person pose estimation. *CoRR*, abs/1612.00137, 2016. URL http://arxiv.org/abs/1612.00137.

[24] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, Jan 2005. ISSN 1573-1405. doi: 10.1023/B:VISI.0000042934.15159.49. URL https://doi.org/10.1023/B:VISI.0000042934.15159.49.

[25] Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587468.

[26] 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.

[27] Wenjuan Gong, Xuena Zhang, Jordi Gonzàlez, Andrews Sobral, Thierry Bouwmans, Changhe Tu, and El-Hadi Zahzah. Human pose estimation from monocular images: A comprehensive survey. *Sensors (Basel, Switzerland)*, 16(12):1966, Nov 2016. ISSN 1424-8220. doi: 10.3390/s16121966. URL https://pubmed.ncbi.nlm.nih.gov/27898003.27898003[pmid].

[28] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2017. http://www.deeplearningbook.org.

[29] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, Mar 1975. ISSN 1860-0980. doi: 10.1007/BF02291478. URL https://doi.org/10.1007/BF02291478.

[30] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

[31] 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`.

[32] Michael B. Holte, Cuong Tran, Mohan M. Trivedi, and Thomas B. Moeslund. Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments. *IEEE Journal of Selected Topics in Signal Processing*, 6 (5):538–552, 2012. doi: 10.1109/JSTSP.2012.2196975.

[33] Mir Rayat Imtiaz Hossain and James J. Little. Exploiting temporal information for 3d pose estimation. *CoRR*, abs/1711.08585, 2017. URL `http://arxiv.org/abs/1711.08585`.

[34] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL `http://arxiv.org/abs/1704.04861`.

[35] 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`.

[36] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. *CoRR*, abs/1605.03170, 2016. URL `http://arxiv.org/abs/1605.03170`.

[37] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7): 1325–1339, 2014. doi: 10.1109/TPAMI.2013.248.

[38] Karim Iskakov, Egor Burkov, Victor S. Lempitsky, and Yury Malkov. Learnable triangulation of human pose. *CoRR*, abs/1905.05754, 2019. URL `http://arxiv.org/abs/1905.05754`.

[39] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *In Proc. UIST*, pages 559–568, 2011.

[40] Xiaofei Ji and Honghai Liu. Advances in view-invariant human motion analysis: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):13–24, 2010. doi: 10.1109/TSMCC.2009.2027608.

[41] S.X. Ju, M.J. Black, and Y. Yacoob. Cardboard people: a parameterized model of articulated image motion. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 38–44, 1996. doi: 10.1109/AFGR.1996.557241.

[42] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *CoRR*, abs/2001.04451, 2020. URL `https://arxiv.org/abs/2001.04451`.

[43] 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.

[44] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Openpifpaf: Composite fields for semantic keypoint detection and spatio-temporal association. *CoRR*, abs/2103.02440, 2021. URL `https://arxiv.org/abs/2103.02440`.

[45] 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`.

[46] Yang Li, Lukasz Kaiser, Samy Bengio, and Si Si. Area attention. *CoRR*, abs/1810.10126, 2018. URL `http://arxiv.org/abs/1810.10126`.

[47] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL `http://arxiv.org/abs/1405.0312`.

[48] 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`.

[49] Zhao Liu, Jianke Zhu, Jiajun Bu, and Chun Chen. A survey of human pose estimation: The body parts parsing based methods. *Journal of Visual Communication and Image Representation*, 32:10–19, 2015. ISSN 1047-3203. doi: https://doi.org/10.1016/j.jvcir.2015.06.013. URL `https://www.sciencedirect.com/science/article/pii/S1047320315001121`.

[50] 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`.

[51] 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`.

[52] 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.

[53] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A simple yet effective baseline for 3d human pose estimation. *CoRR*, abs/1705.03098, 2017. URL http://arxiv.org/abs/1705.03098.

[54] 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.

[55] 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.

[56] Dushyant Mehta, Helge Rhodin, Dan Casas, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation using transfer learning and improved CNN supervision. *CoRR*, abs/1611.09813, 2016. URL http://arxiv.org/abs/1611.09813.

[57] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001. ISSN 1077-3142. doi: https://doi.org/10.1006/cviu.2000.0897. URL https://www.sciencedirect.com/science/article/pii/S107731420090897X.

[58] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, 2006. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2006.08.002. URL https://www.sciencedirect.com/science/article/pii/S1077314206001263. Special Issue on Modeling People: Vision-based understanding of a person's shape, appearance, movement and behaviour.

[59] Tewodros Legesse Munea, Yalew Zelalem Jembre, Halefom Tekle Weldegebriel, Longbiao Chen, Chenxi Huang, and Chenhui Yang. The progress of human pose estimation: A survey and taxonomy of models applied in 2d human pose estimation. *IEEE Access*, 8: 133330–133348, 2020. doi: 10.1109/ACCESS.2020.3010248.

[60] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016. URL http://arxiv.org/abs/1603.06937.

[61] 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.

[62] 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.

[63] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7745–7754, 2019. doi: 10.1109/CVPR.2019.00794.

[64] 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.

[65] Martin Popel and Ondrej Bojar. Training tips for the transformer model. *CoRR*, abs/1804.00247, 2018. URL `http://arxiv.org/abs/1804.00247`.

[66] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1):4–18, 2007. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2006.10.016. URL `https://www.sciencedirect.com/science/article/pii/S1077314206002293`. Special Issue on Vision for Human-Computer Interaction.

[67] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL `http://arxiv.org/abs/1506.02640`.

[68] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL `http://arxiv.org/abs/1506.01497`.

[69] Daniel Roetenberg, Henk Luinge, and Per Slycke. Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technol. BV Tech. Rep.*, 3, 01 2009.

[70] 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`.

[71] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL `http://arxiv.org/abs/1801.04381`.

[72] Ben Sapp and Ben Taskar. Modec: Multimodal decomposable models for human pose estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3681, 2013. doi: 10.1109/CVPR.2013.471.

[73] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A. Kakadiaris. 3d human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 152:1–20, 2016. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2016.09.002. URL `https://www.sciencedirect.com/science/article/pii/S1077314216301369`.

[74] 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.

[75] H. Sidenbladh, F. De la Torre, and M.J. Black. A framework for modeling the appearance of 3d articulated figures. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 368–375, 2000. doi: 10.1109/AFGR.2000.840661.

[76] 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`.

[77] 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`.

[78] 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`.

[79] Jing Tong, Jin Zhou, Ligang Liu, Zhigeng Pan, and Hao Yan. Scanning 3d full human bodies using kinects. *IEEE Transactions on Visualization and Computer Graphics*, 18 (4):643–650, 2012. doi: 10.1109/TVCG.2012.56.

[80] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659, 2013. URL `http://arxiv.org/abs/1312.4659`.

[81] Hanyue Tu, Chunyu Wang, and Wenjun Zeng. End-to-end estimation of multi-person 3d poses from multiple cameras. *CoRR*, abs/2004.06239, 2020. URL `https://arxiv.org/abs/2004.06239`.

[82] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *CoRR*, abs/1701.01370, 2017. URL `http://arxiv.org/abs/1701.01370`.

[83] 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`.

[84] 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`.

[85] 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`.

[86] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. *CoRR*, abs/1602.00134, 2016. URL `http://arxiv.org/abs/1602.00134`.

[87] Lilian Weng. Attention? attention! *lilianweng.github.io/lil-log*, 2018. URL `http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html`.

[88] Chuhan 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`.

[89] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. *CoRR*, abs/1804.06208, 2018. URL `http://arxiv.org/abs/1804.06208`.

[90] Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. Sharing attention weights for fast transformer. *CoRR*, abs/1906.11024, 2019. URL `http://arxiv.org/abs/1906.11024`.

[91] 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`.

[92] 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`.

[93] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013. doi: 10.1109/TPAMI.2012.261.

[94] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *NAACL 2016*, pages 1480–1489, June 2016. URL https://www.microsoft.com/en-us/research/publication/hierarchical-attention-networks-document-classification/.

[95] Ce Zheng, Wenhan Wu, Taojiannan Yang, Sijie Zhu, Chen Chen, Ruixu Liu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. Deep learning-based human pose estimation: A survey. *CoRR*, abs/2012.13392, 2020. URL https://arxiv.org/abs/2012.13392.

[96] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, abs/1711.06396, 2017. URL http://arxiv.org/abs/1711.06396.