Neural Machine Translation:

Seq-to-Seq con arquitecturas de deep learning

Felipe Ramírez Herrera

VNIVERSITAT DE VALÈNCIA

Agenda

Contribución

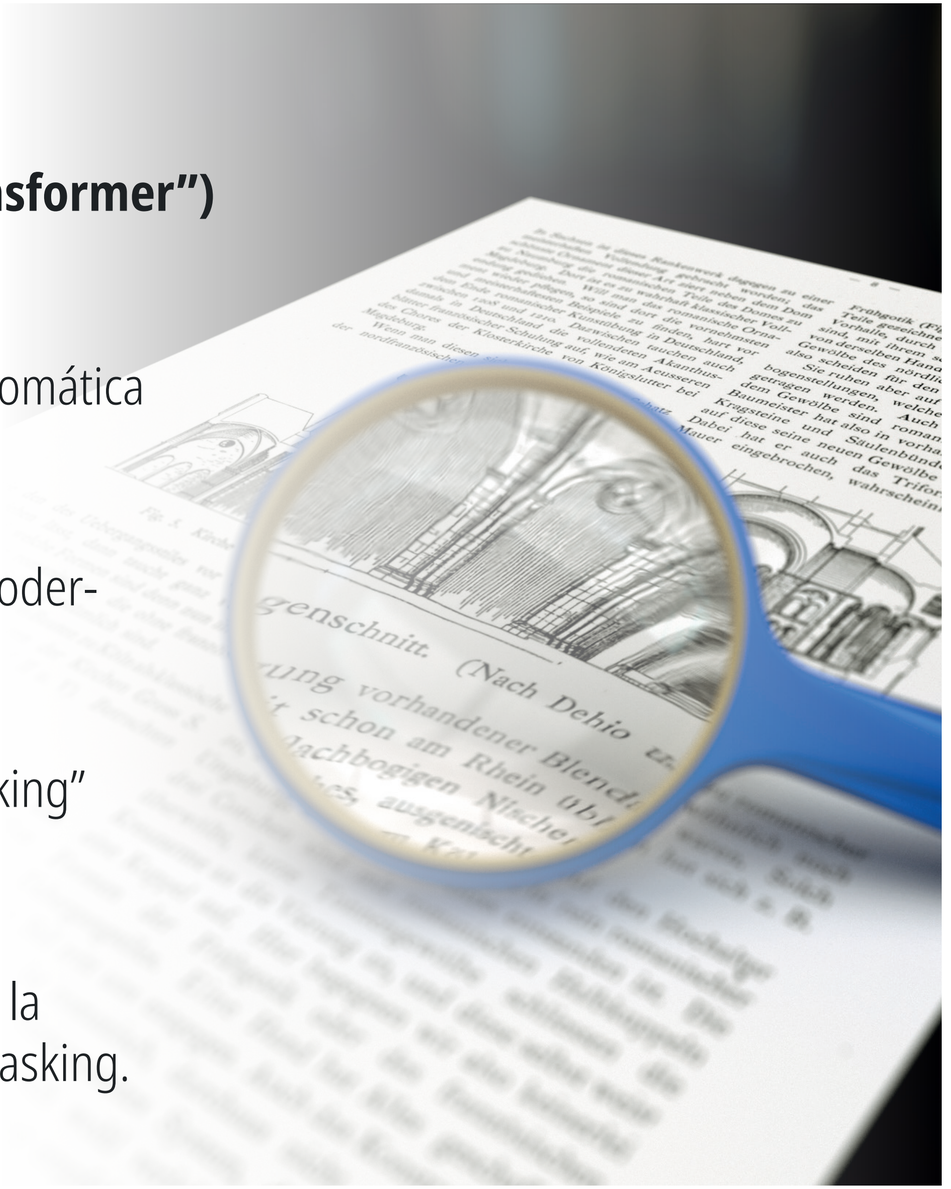
¿Se puede modificar una arquitectura Encoder-Decoder (p. ej. “transformer”) agregando un decodificador más?

1. Aplicar modelos de redes profundas para la tarea de traducción automática (Seq-to-Seq) en los siguientes idiomas “EN a ES” y “EN a FR”.

2. Implicaciones de implementar “dual-tasking”[1] en arquitecturas “encoder-decoder”.

3. Comparar una solución de composición de dos modelos “single tasking” contra un modelo “dual tasking”.

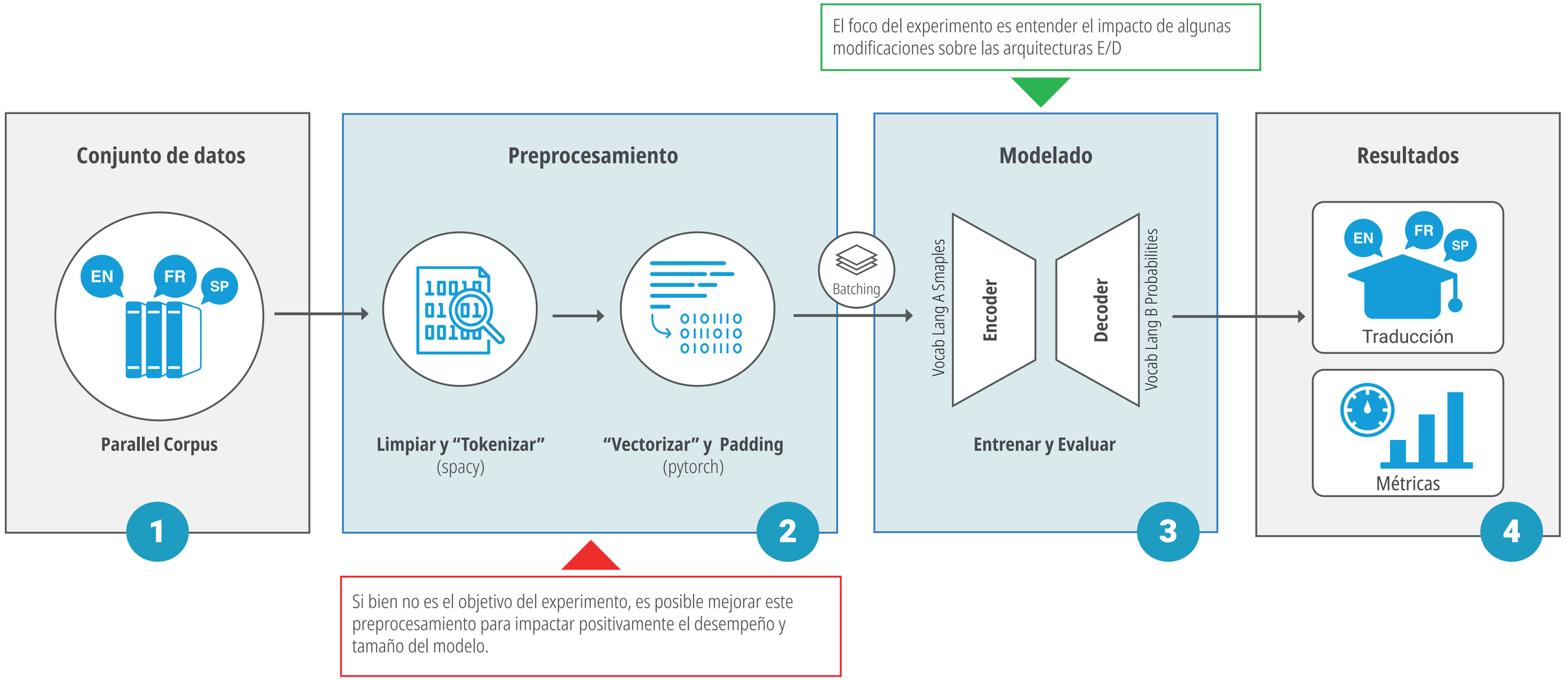
4. Entender las consecuencias del weight-sharing con el fin de reducir la cantidad de parámetros y su impacto sobre una arquitectura “dual tasking.



Nota: La teoría sobre Transformers fue presentada durante la clase del 20 de Enero de 2024 por Miguel Ángel Fernández Torres. (1) Definir la arquitectura con dos decodificadores implicaría “dual-tasking”, mientras que definirla con dos codificadores o “cruzar” los decodificadores conllevaría “multi-modality” a nivel de entrada o salida.

Tarea: Neural Machine Translation (NMT)

Está diseñada originalmente como una tarea de aprendizaje de extremo a extremo. Procesa directamente una secuencia de origen a una secuencia de destino (seq-to-seq).



Nota: El objetivo de aprendizaje es encontrar la secuencia de destino correcta dada la secuencia de origen, lo cual puede verse como un problema de clasificación de alta dimensionalidad que intenta mapear las dos oraciones en el espacio semántico. En todos los principales modelos modernos de NMT, este proceso se puede dividir en un paso de codificación y un paso de decodificación, y así separar funcionalmente todo el modelo. (Yang, Wang, & Chu, 2020)

Conjunto de datos

Un corpus paralelo contiene traducciones del mismo documento en dos o más idiomas, alineadas al menos a nivel de oración. Estos tienden a ser más raros que los corpora menos comparables[1]. Es muy poco frecuente su uso en literatura o ejemplos.

Conjunto de datos

Conjunto de datos	Espacio requerido en SSD	Número de documentos	Número total de líneas	Número de líneas utilizadas	Vocabulario (EN)
UN-Parallel Corpus V1	~5,8 GB	86 307	11 365 709	500 000	334 953 817 tokens

Entorno de prueba

Elemento	Descripción
CPU	AMD® Ryzen ® 9 5950X 4.9 Ghz con 16 Cores / 32 Hilos, Caché L3: 64 MB.
GPU	NVIDIA ® GeForce ® RTX 3060 12 GB de memoria GDDR6 y 3584 CUDA Cores.
Memoria	64 GB de Memoria DDR4 (4 módulos)
Almacenamiento	Kingston® NV1 NVMe M.2 PCIe Gen3, 1TB.
Sistema operativo	WSL2 sobre Microsoft Windows ® 11 Professional (Ubuntu® 22.04.2 LST)
Ambiente	Miniconda. Python 3.10.13, Pytorch 2.2.0 / NVidia CUDA 12.1.105

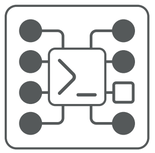
Nota: (1) La versión actual del fully-aligned corpus contiene contenido que fue producido y traducido manualmente entre 1990 y 2014, incluyendo alineaciones a nivel de oración. Véase https://conferences.unite.un.org/uncorpus

Particionamiento de los datos y otras configuraciones

Datos de entrenamiento

Conjunto de datos	Número de ejemplos	Número de ejemplos con max_seq < 102[2]	Ejemplos para entrenamiento	Ejemplos para validación	Ejemplos para prueba
UN-Parallel Corpus V1[1]	500 000 (~4.40%)	440 944 (~88.19%)	308 660 (~70%)	88 630 (~20%)	43654 (~10%)

Notas:
(1) Sólo se consideran los ejemplos (oraciones) para EN, ES y FR, muestra tomada secuencialmente.
(2) Se acota para asegurar que el modelo y los datos generados por el mismo no excedieran el tamaño de la memoria de la GPU y que los tiempos de entrenamiento fueran razonables, max_seq_length es un hiperparámetro dependiente de los datos y afecta diferentes capas de las arquitecturas.



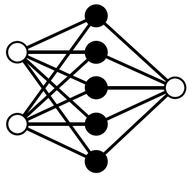
Hiperparámetros y configuración

min_seq_length	max_seq_length	EN-Vocab Size	ES-Vocab Size	FR-Vocab Size	Batch Size
3 tokens + UNK, PAD, BoS y EoS	102	74 563	91 234	86 462	32

Optimizadores	Scheduler	Early Stopping	Clipping	Loss Measure (ST)	Loss Measure (DT)
1 x ADAM LR: 0.0001	LROnPlateau	5	15	Cross-Entropy (CE)	$CE_{ES} + CE_{FR}$

Notas: Cuando los gradientes son demasiado grandes, pueden causar problemas como la explosión del gradiente, lo que puede hacer que el entrenamiento falle o se vuelva extremadamente lento. El clipping de gradientes implica limitar el valor de los gradientes a un cierto umbral. Un valor de clipping más alto significa que se permiten gradientes más grandes, lo que puede acelerar el entrenamiento, pero también aumenta el riesgo de problemas de explosión del gradiente.

Configuración de las arquitecturas

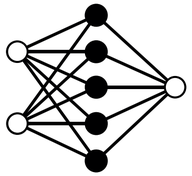


Transformer

Input Vocab Size	LANG	D_MODEL[1]	LAYERS[2]	N_HEADS	DROPOUT	HIDDEN_DIM[3]
74563	EN	256	3	8	0.1	512

Output Vocab Size	LANG	D_MODEL[1]	LAYERS[2]	N_HEADS	DROPOUT	HIDDEN_DIM[3]
91234	ES	256	3	8	0.1	512
86462	FR	256	3	8	0.1	512

Nota: Se implementa con Multihead Attention Mechanism (Vaswani et al., 2017). Originalmente (1) D_Model = 512, (2) Layers = 6 y (3) Hidden_Dim = 2048. Se utiliza la siguiente nomenclatura: TFRMR_ST_COMP para una composición de dos modelos single task (ST) independientes, TFMR_DT para un modelo dual task (DT) con dos decodificadores y TFMR_DT_WS para un modelo dual task (DT) con dos decodificadores y al que se le aplica la técnica de Weight-Sharing (WS) para reducir el número de parámetros.



Gated CNN

Input Vocab Size	LANG	D_MODEL	LAYERS[2]	KERNEL_SIZE	GLU_UNITS	DROPOUT	HIDDEN_DIM
74563	EN	256	5	3	5	0.25	512

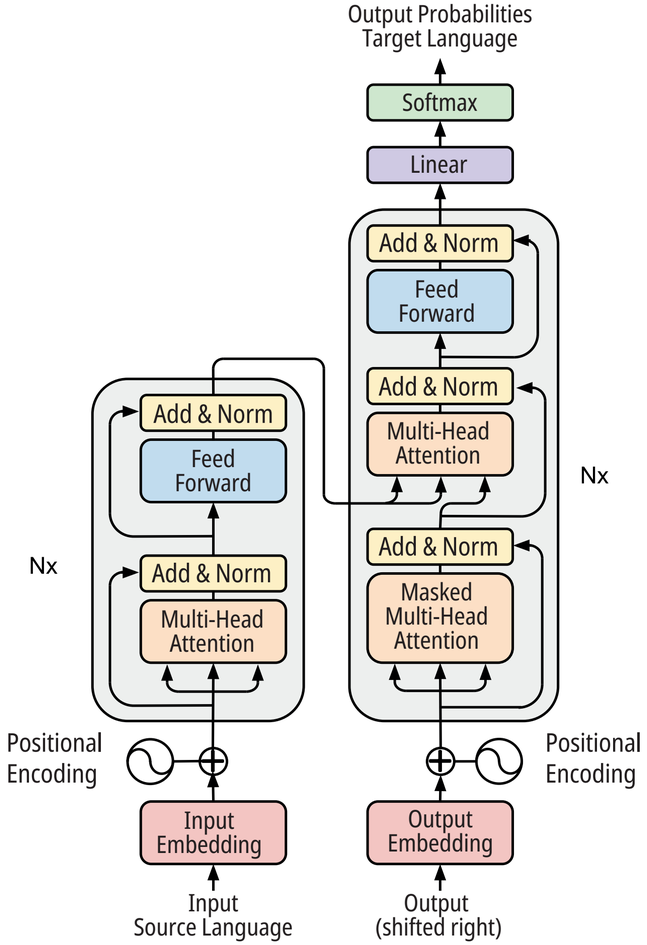
Output Vocab Size	LANG	D_MODEL	LAYERS[2]	KERNEL_SIZE	GLU_UNITS	DROPOUT	HIDDEN_DIM
91234	ES	256	5	3	5	0.25	512
86462	FR	256	5	3	5	0.25	512

Nota: Se implementa con Scaled Dot-Product Attention Mechanism (Vaswani et al., 2017). Originalmente: (2) Layers = 10. Se utiliza la siguiente nomenclatura: GCNN_ST_COMP para una composición de dos modelos single task (ST) independientes, GCNN_DT para un modelo dual task (DT) con dos decodificadores y GCCN_DT_WS para un modelo dual task (DT) con dos decodificadores y al que se le aplica la técnica de Weight-Sharing (WS) para reducir el número de parámetros.

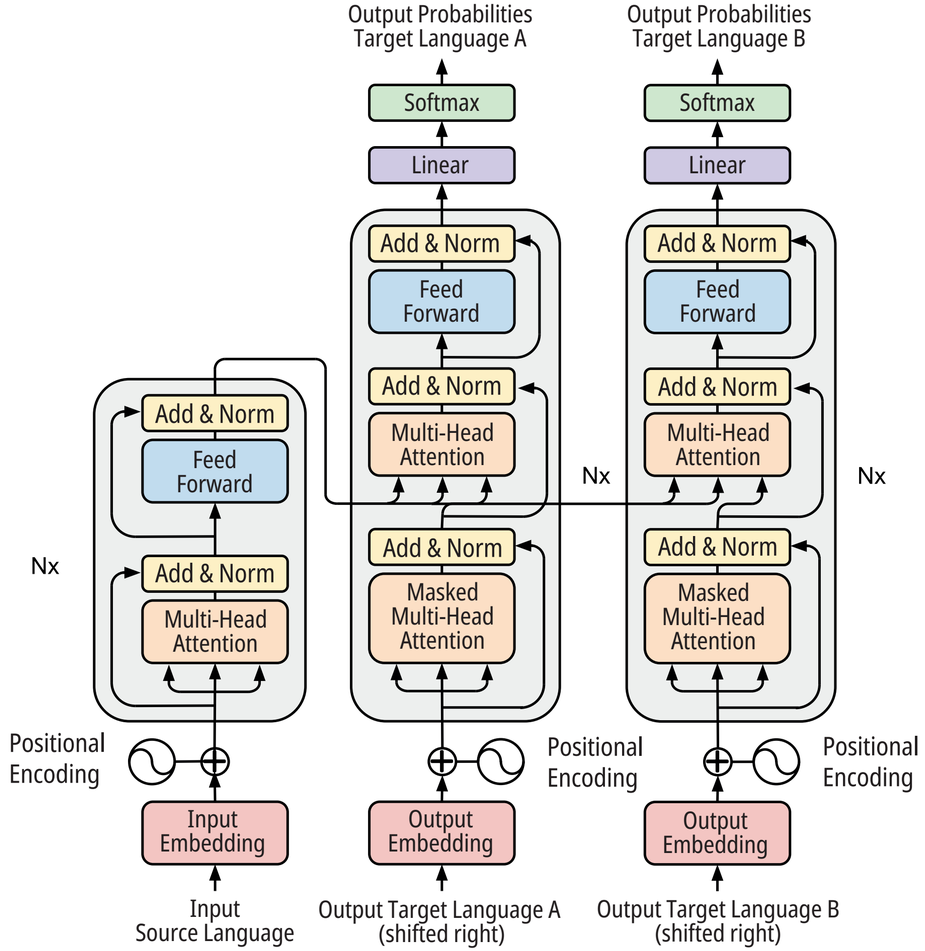
Transformers: Self-Attentive Networks

Se propone una variante de la arquitectura del transformer clásico para soportar dos tareas (dual tasking)

A Transformer Clásico[1] (TFMR_ST)



B Decodificadores independientes[2] (TFMR_DT)

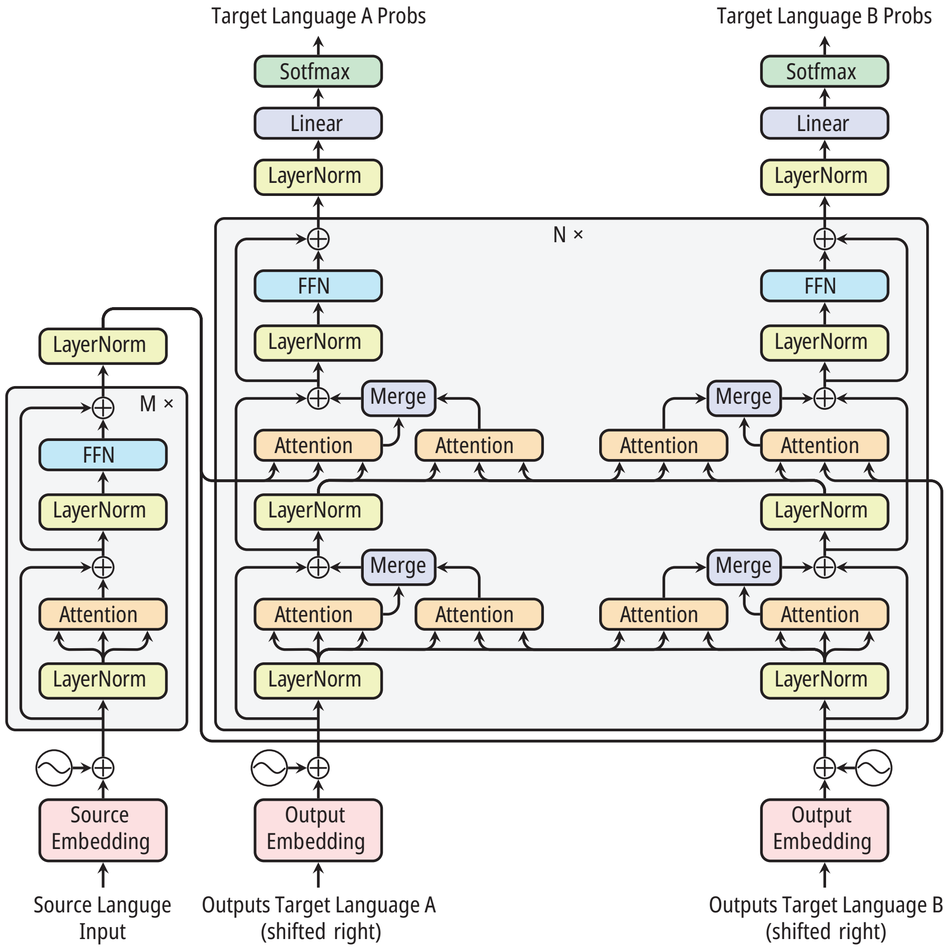


Notas: (1) Tomado de Vaswani et al. (2017). (2) Basado en taxonomía de Anastasopoulos and Chiang (2018) y Sperber et al. (2020).

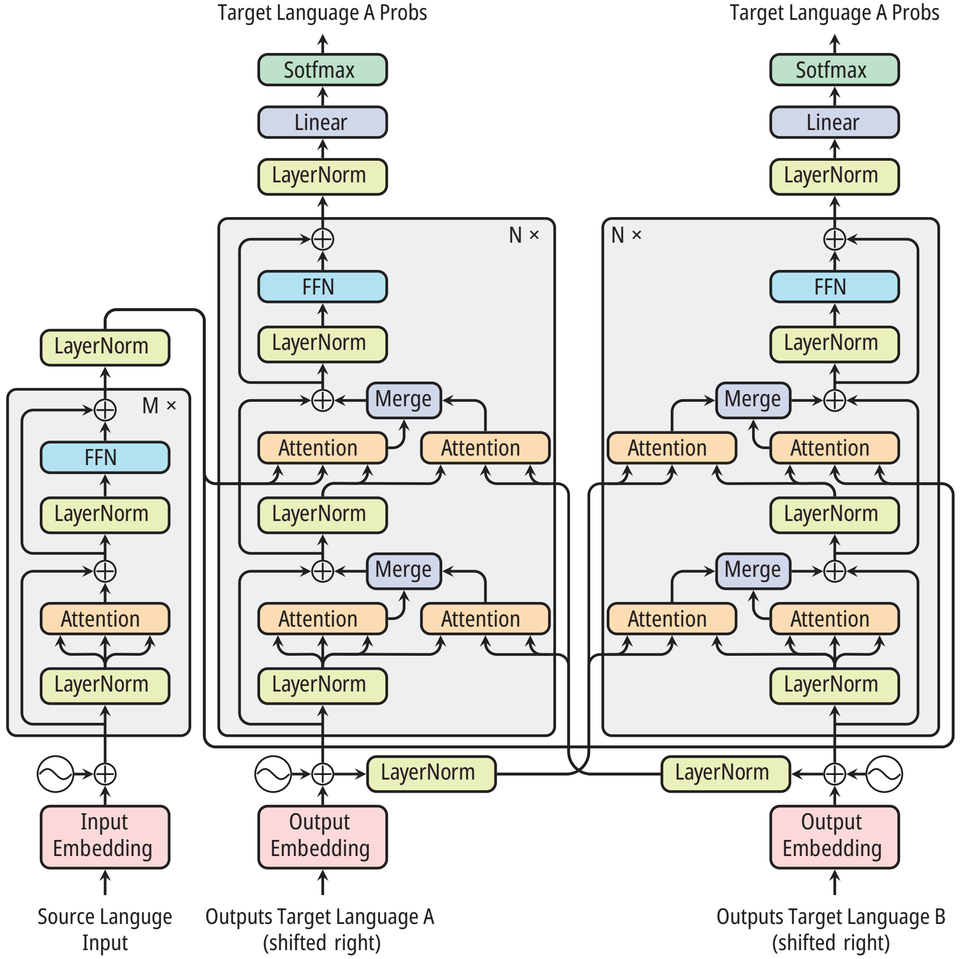
Transformers: Self-Attentive Networks

Se puede complicar todavía más para incorporar tareas multimodales (p. ej. ASR y ST).

C Decodificadores en paralelo[1] (no implementado)



D Decodificadores cruzados[1] (no implementado)

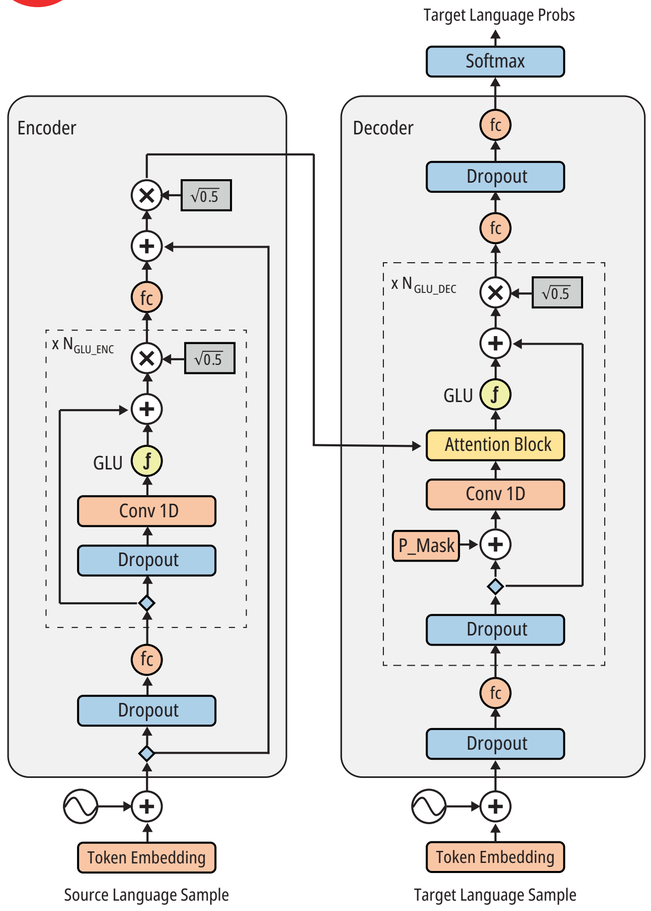


Nota: (1) Basado en Le et al. (2020).

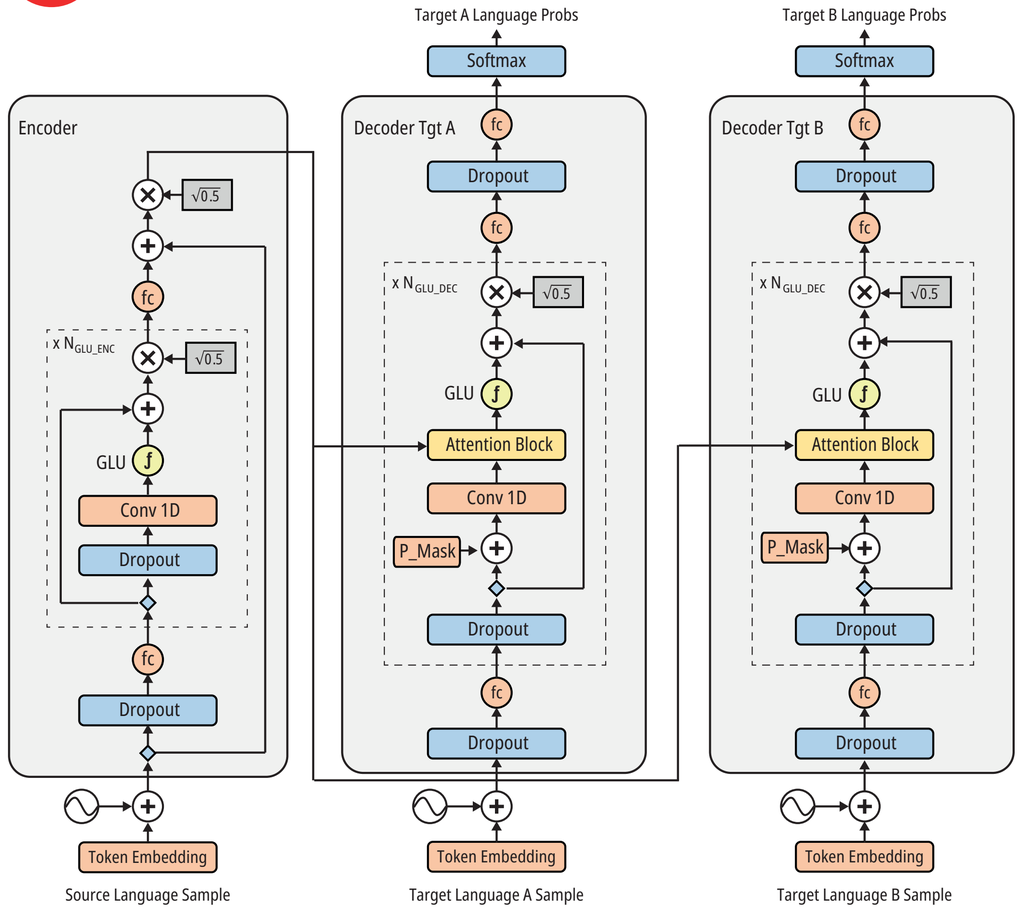
Gated Convolutional Self-Attentive Network

Las redes de convolución fueron introducidas parcialmente en NLP para superar la ineficiencia computacional de los modelos recurrentes[1]. Son una alternativa efectiva a otras arquitecturas de modelado de secuencias y del SOTA[2].

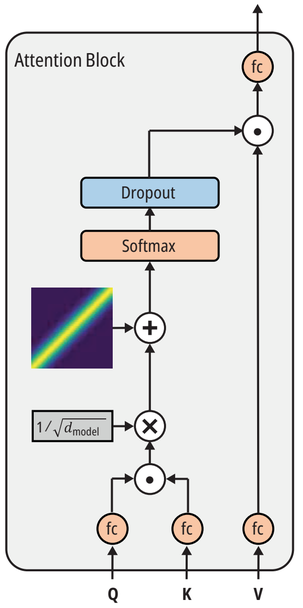
A **GCNN_ST[3]** (single task)



B **GCNN_DT[4]** (dual task)



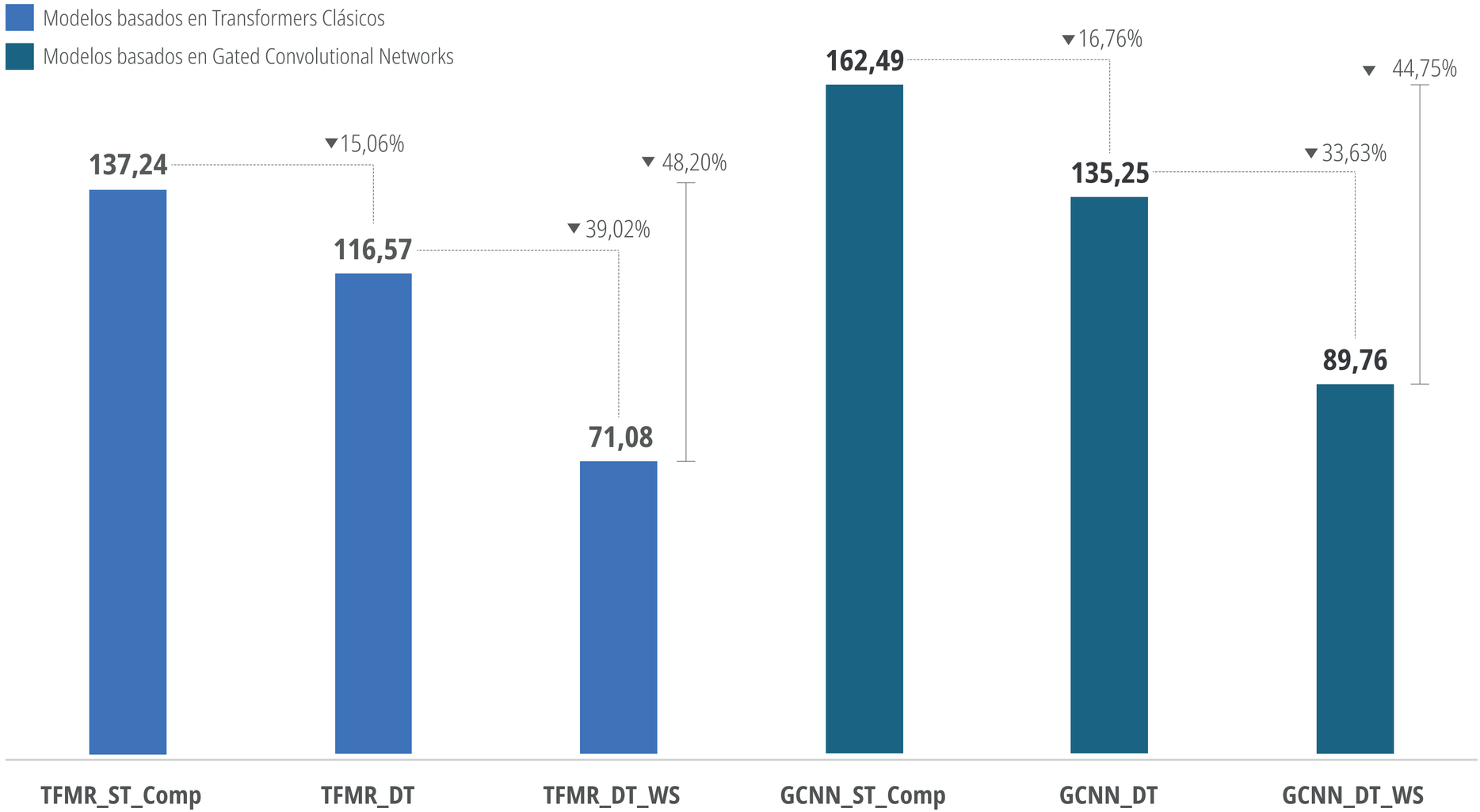
C

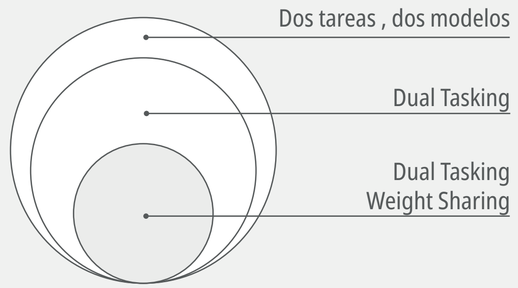


Nota: (1) Véase (Bradbury et al., 2017) (2) Véase (Hu et al., 2023) (3) Basado en Dauphin et al. (2017). (4) Propuesto por el alumno.
Ambos con código fuente basado en: https://github.com/facebookresearch/fairseq y https://github.com/porcelainruler/Language-Translation

Cantidad de parámetros

Efecto de las técnicas aplicadas sobre número de parámetros entrenables de los modelos propuestos.





Cantidad de parámetros entrenables

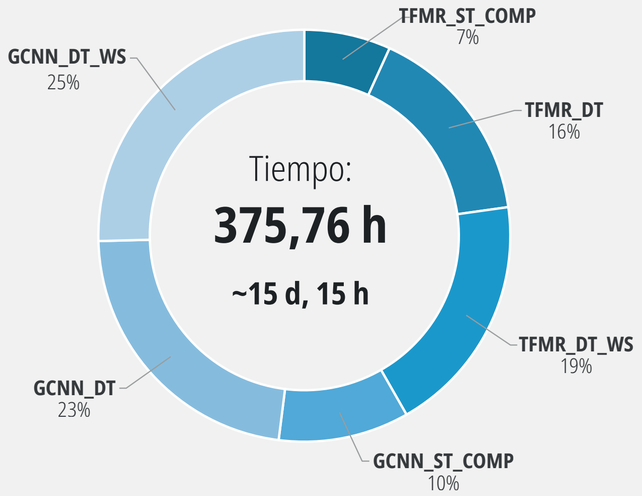
- Implementar el modelo como “Dual Tasking“ o “Dual Decoder” sí reduce la candidad de parámetros en promedio de **15,91%**.
- Uso de Weight Sharing reduce en promedio **36,32%**.
- El efecto promedio de ambas estrategias es de una reducción de **46,47%** de la cantidad de parámetros entrenables.

Fuente: Elaboración propia del estudiante. Cifras absolutas dadas en millones de parámetros.

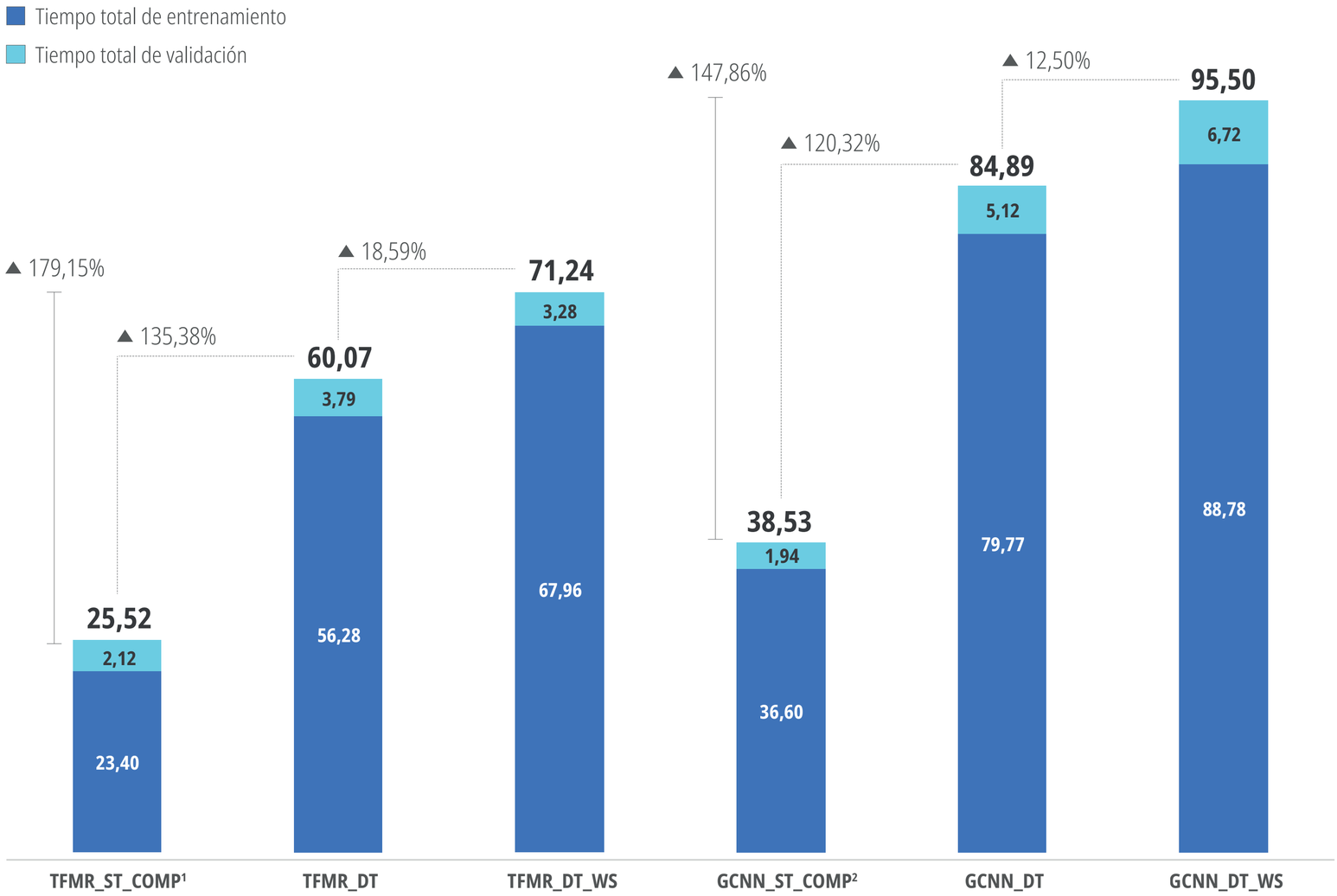
La escala de un modelo es uno de los temas más importantes y muchas veces condiciona la “capacidad” del modelo para generalizar. Dado un presupuesto computacional fijo, entrenar un modelo más grande durante menos pasos es mejor que entrenar un modelo más pequeño durante más pasos.

Tiempo de entrenamiento y validación

Menor tiempo conduce a un uso más eficiente de los recursos computacionales y a la construcción rápida de modelos efectivos.



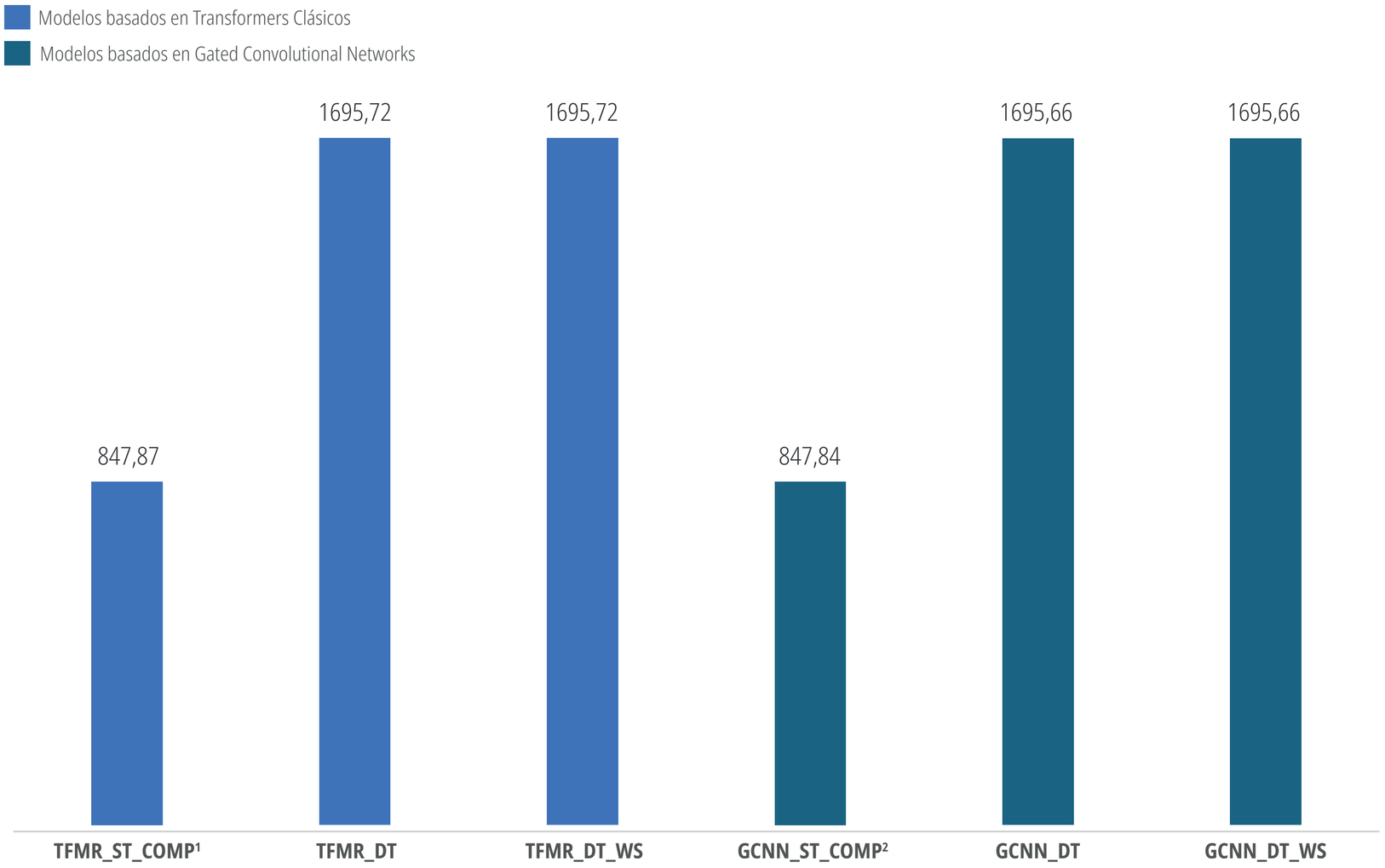
- Entrenar el 4.4% del corpus tomó 15 días y 15 horas. Entrenar todo el corpus, asumiendo linealidad y que es posible hacerlo en el hardware actual con 16 épocas, tomaría aproximadamente 355 días y 20 horas (8540 h).
- El consumo energético es de **319,39 KWh**. Esto es equivalente al consumo mensual de un hogar promedio o 63,88 EUR (1KWh = 0,2 EUR).
- Es más factible entrenar dos modelos individuales que un único modelo con doble decodificador en el hardware de consumo.

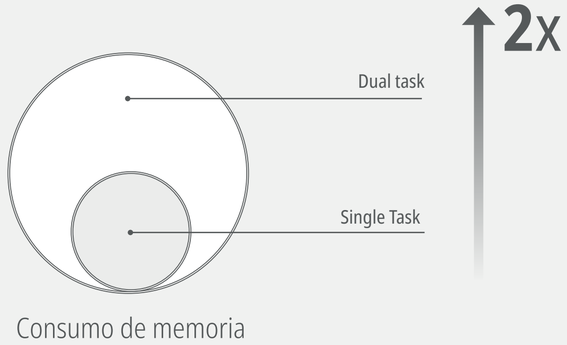


Nota: Tiempo promedio dado en horas tras efectuar 16 epochs. Tipo de cambio al 08/04/2024. (1) y (2) Se componen dos modelos ST independientes en una solución de NMT.
Fuente: Elaboración propia del estudiante.

Consumo de memoria

La cantidad de memoria disponible en la unidad de procesamiento (CPU o GPU) puede limitar el tamaño del modelo y el tamaño de los lotes de datos que se pueden procesar simultáneamente.



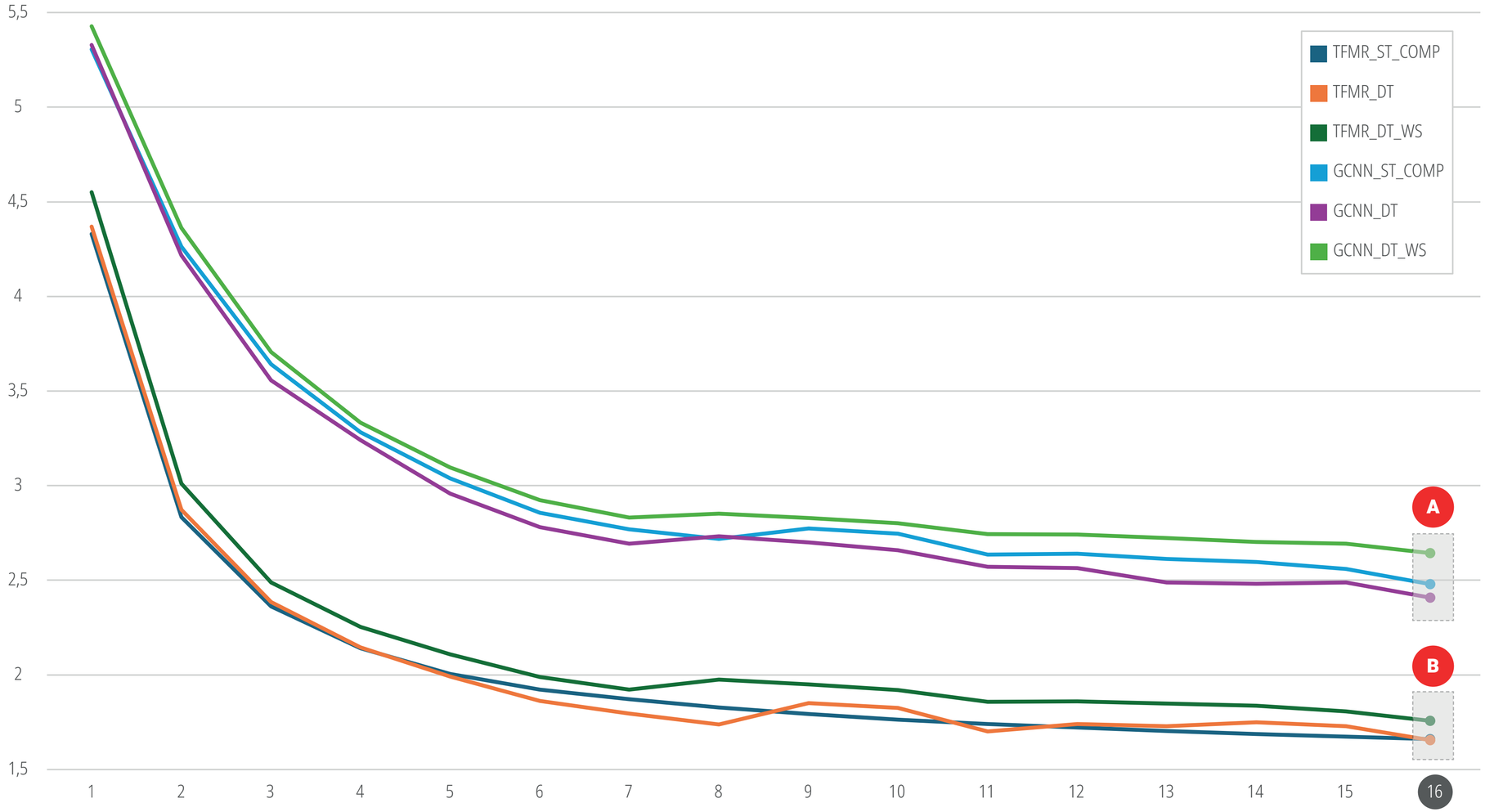


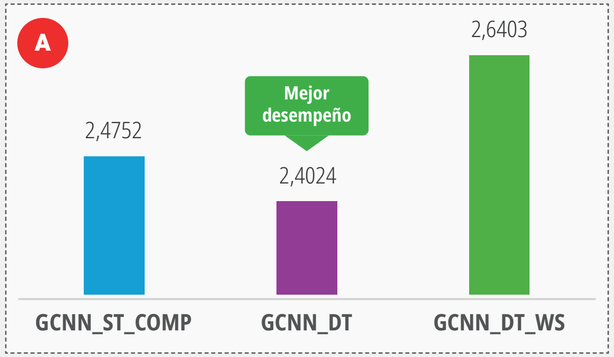
- El procesamiento multitarea conlleva “n” resultados dependiendo del número de decodificadores presentes en el modelo.
- Existe una mínima diferencia entre los modelos TFMR y GCNN, donde los últimos consumen menos memoria debido a que no requieren máscaras de fuente, objetivo, memoria y padding.
- Weigth Sharing / Weight Tying no ofrece mejora sobre el consumo de memoria durante el ciclo de entrenamiento, sólo impacta positivamente el tamaño del modelo (conteo de pesos), pero no de los “logits” resultantes.

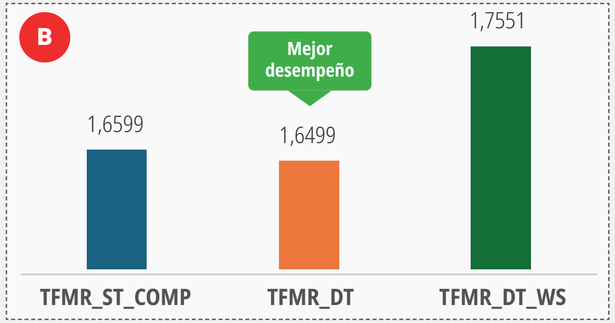
Notas: Consumo de memoria promedio por batch dado en MB tras efectuar 16 ejecuciones. (1) y (2) Se componen dos modelos ST independientes en una solución de NMT.
Fuente: Elaboración propia del estudiante.

Pérdida - Traducción EN a ES (entrenamiento)

Si bien los modelos “dual task” reducen ligeramente la pérdida durante el entrenamiento, “weight-sharing“ la empeora.





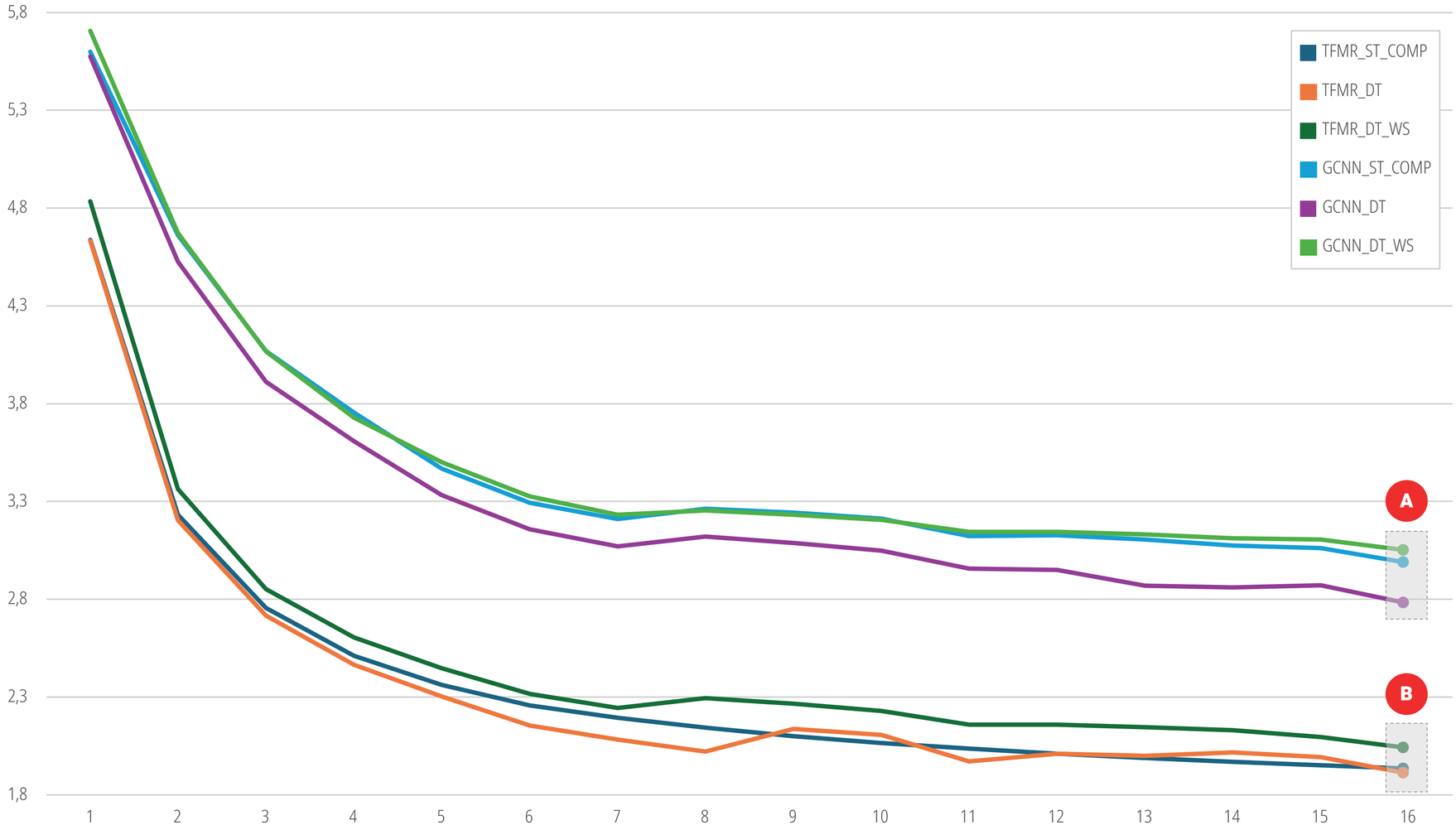


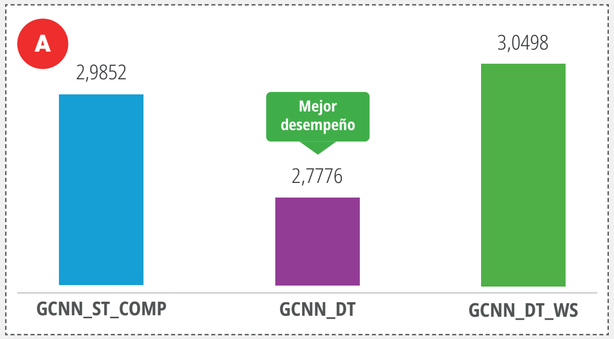
- Comparativamente, una menor pérdida significa que el modelo está haciendo predicciones más precisas o está más cerca de la solución óptima.
- Para el lenguaje ES los modelos “Dual Task” son mejores a nivel de la fase de entrenamiento.

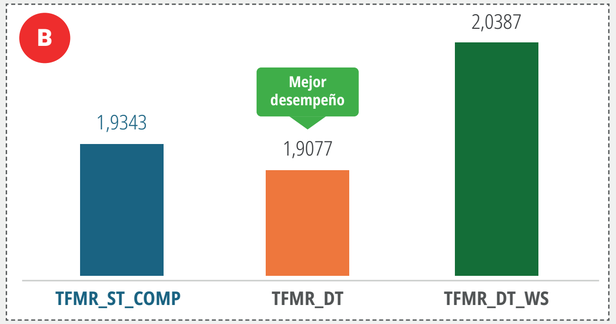
Fuente: Elaboración propia del estudiante.

Pérdida - Traducción EN a FR (entrenamiento)

Si bien los modelos “dual task” reducen ligeramente la pérdida durante el entrenamiento, “weight-sharing“ la empeora.





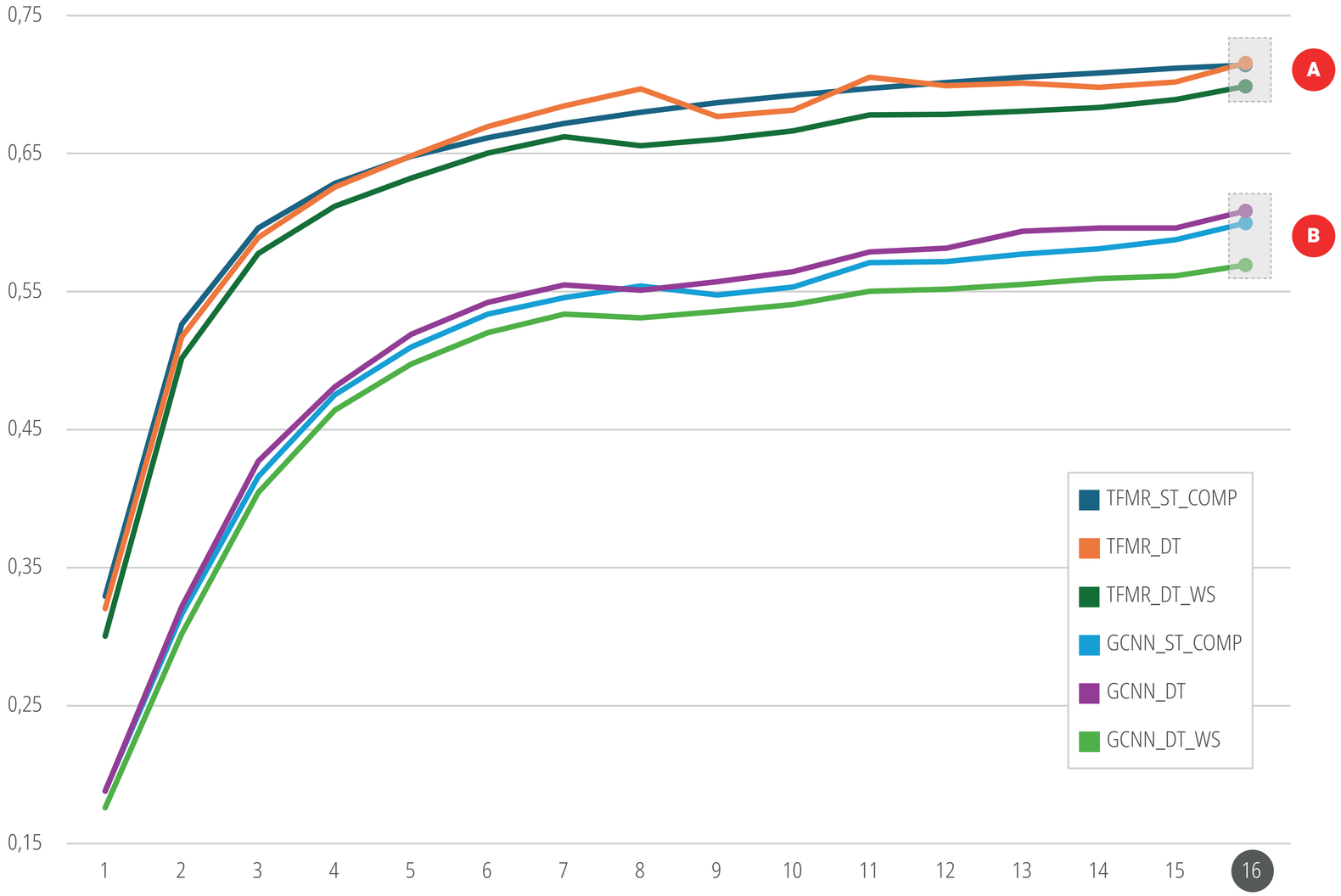


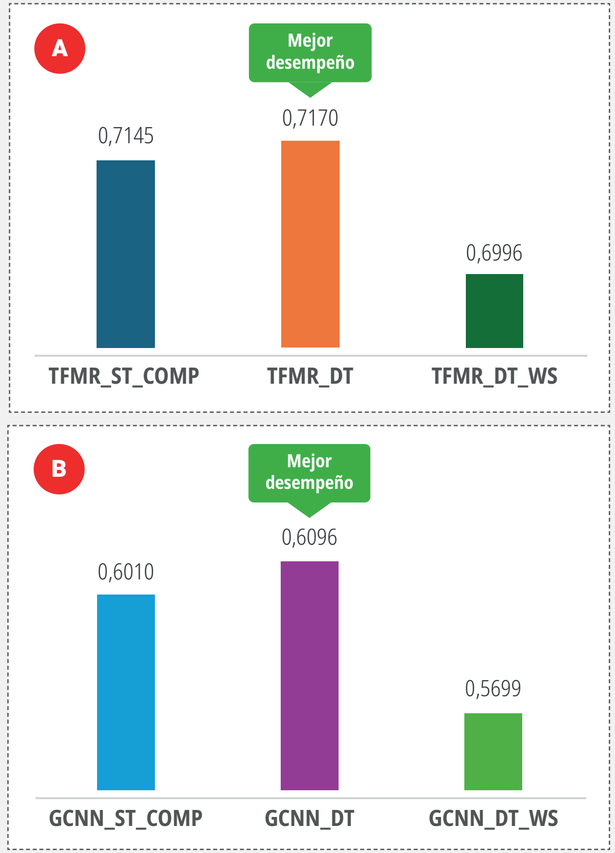
- Para el lenguaje FR los modelos “Dual Task” sin aplicarles weight sharing son mejores a nivel de la fase de entrenamiento.

Fuente: Elaboración propia del estudiante.

Exactitud - Traducción EN a ES (entrenamiento)

Comparar la exactitud entre modelos proporciona una medida objetiva del rendimiento relativo de cada uno.



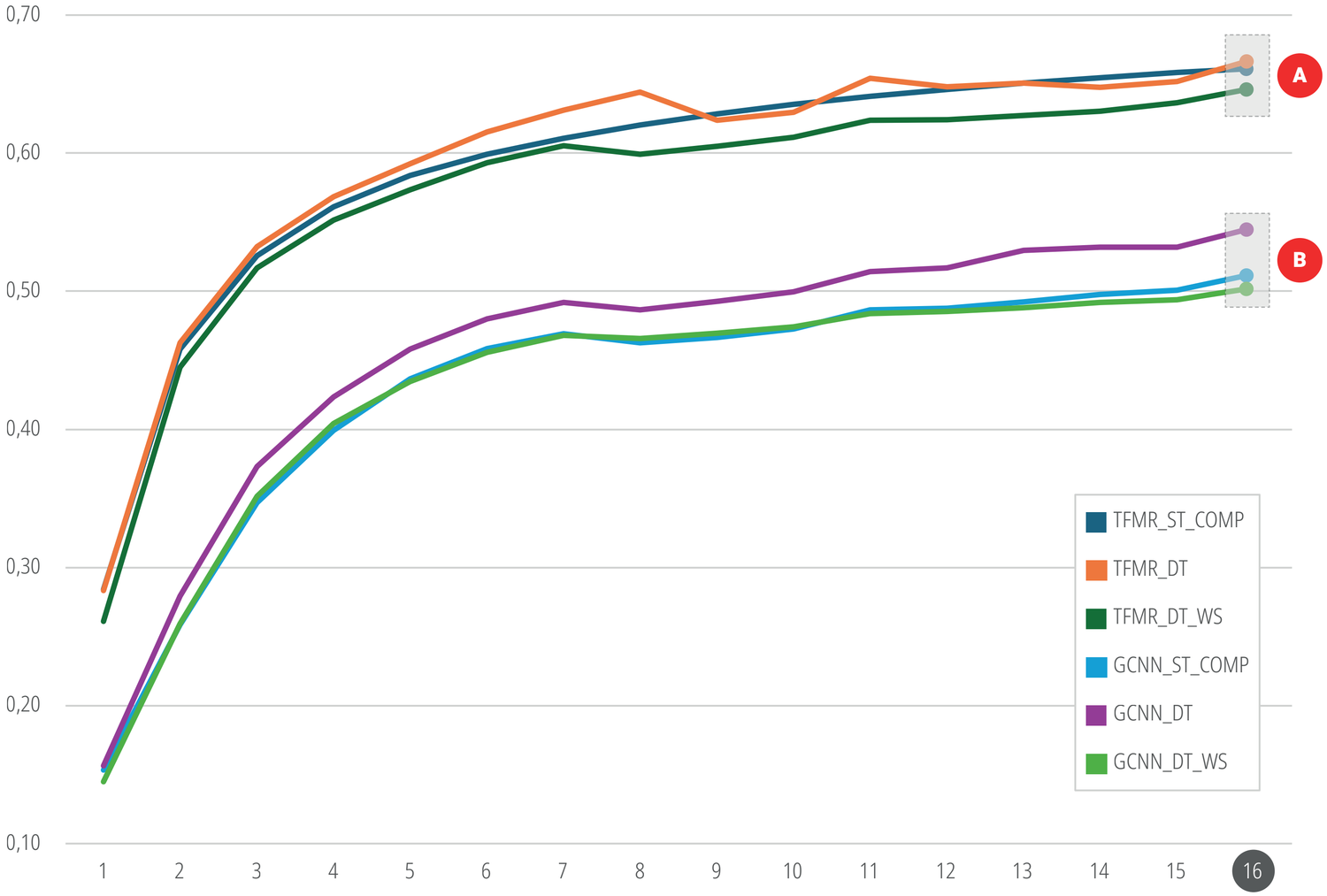


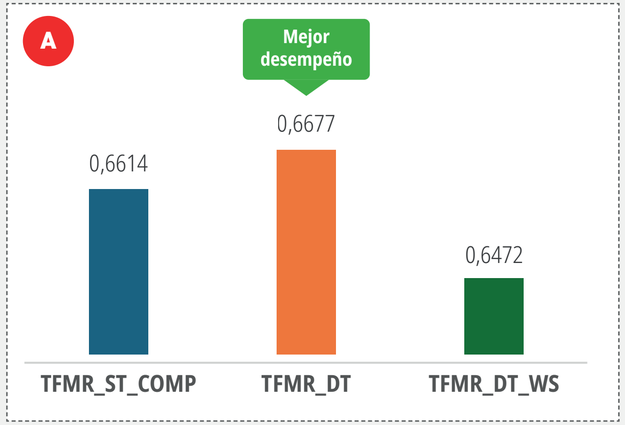
Para el lenguaje ES los modelos "Dual Task" sin weight sharing son ligeramente mejores durante el entrenamiento.

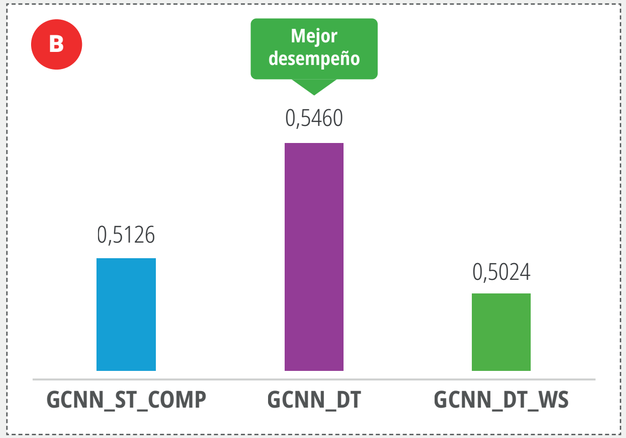
Fuente: Elaboración propia del estudiante.

Exactitud - Traducción EN a FR (entrenamiento)

Comparar la exactitud entre modelos proporciona una medida objetiva del rendimiento relativo de cada uno.





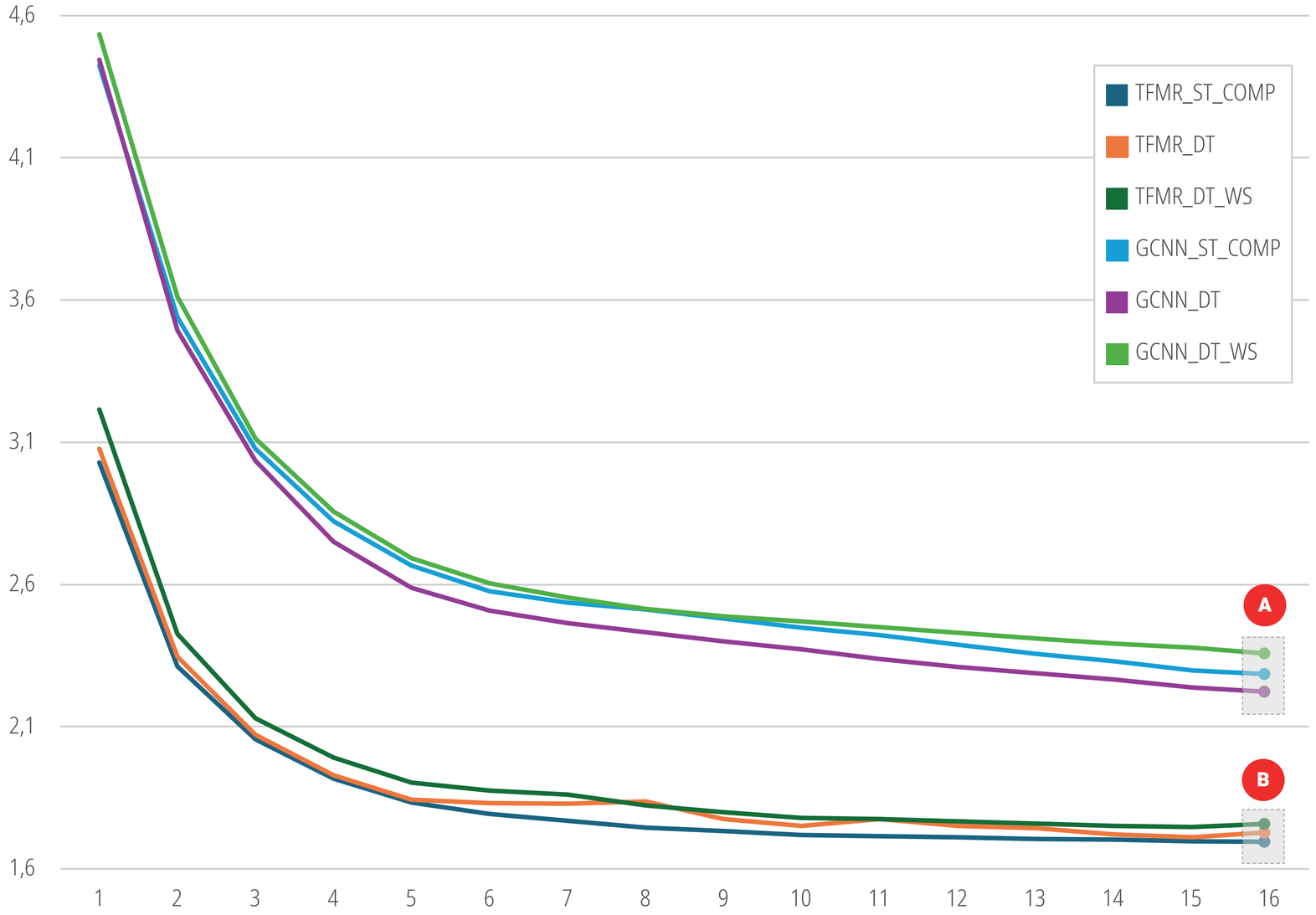


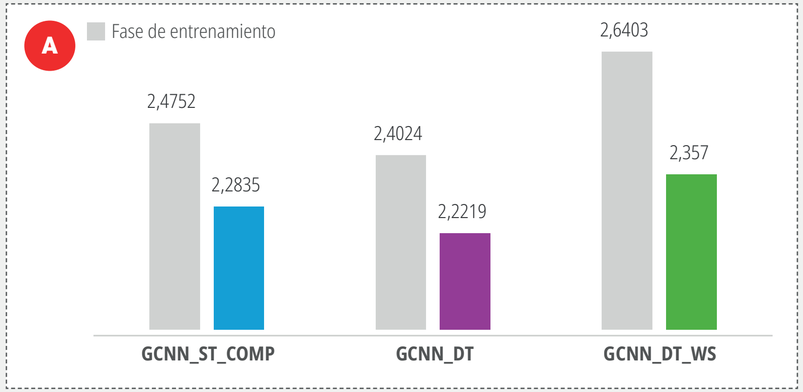
Los modelos “Dual Task” son ligeramente mejores durante el entrenamiento. Weight-Sharing impacta negativamente el desempeño del modelo por la pérdida de capacidad.

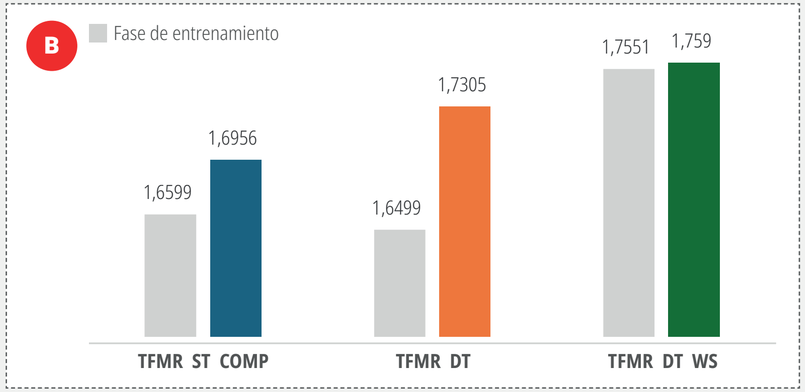
Fuente: Elaboración propia del estudiante.

Pérdida - Traducción EN a ES (post-validación)

Un gap negativo entre la pérdida durante el entrenamiento y la validación es generalmente una señal de alerta.





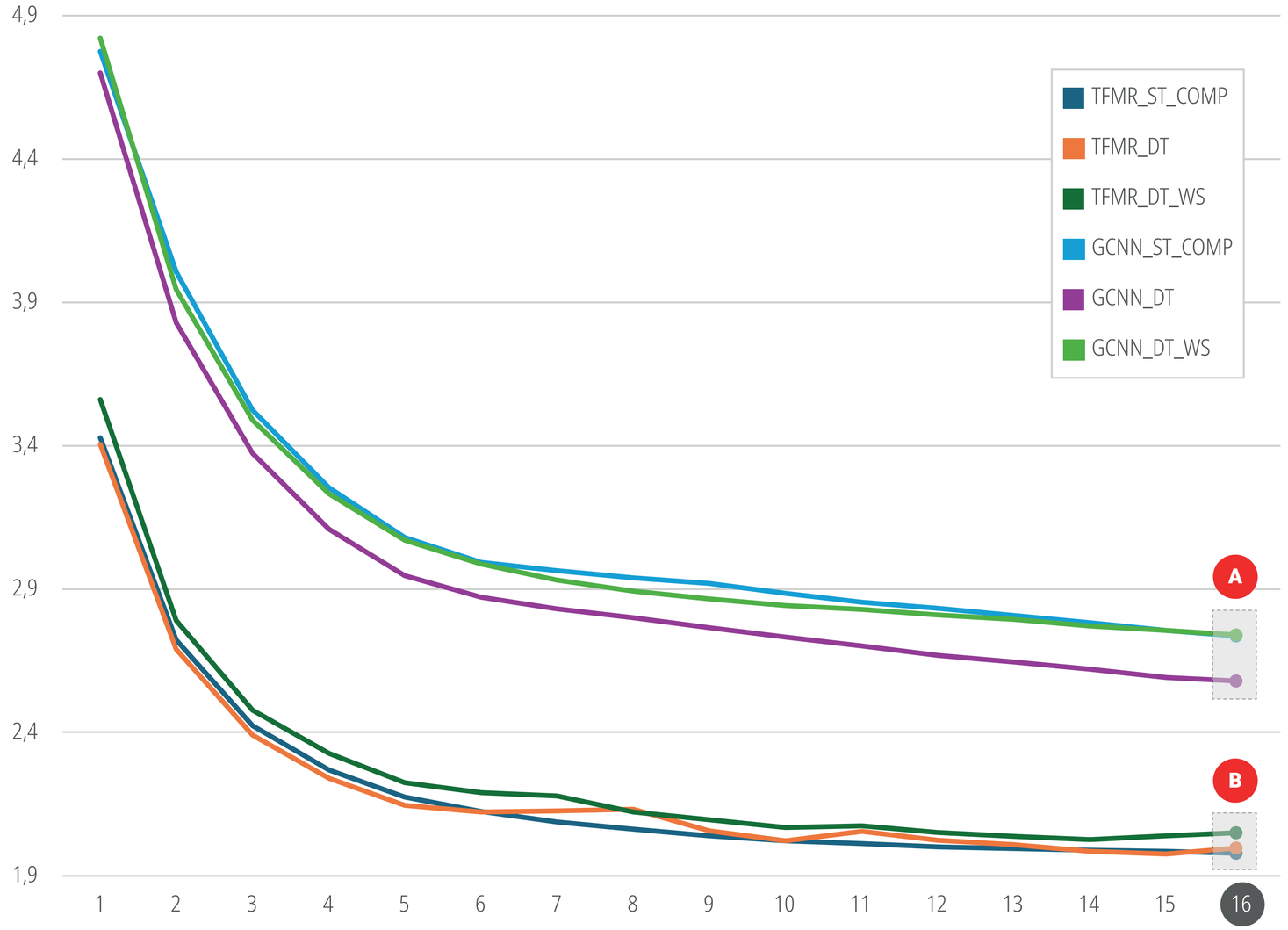


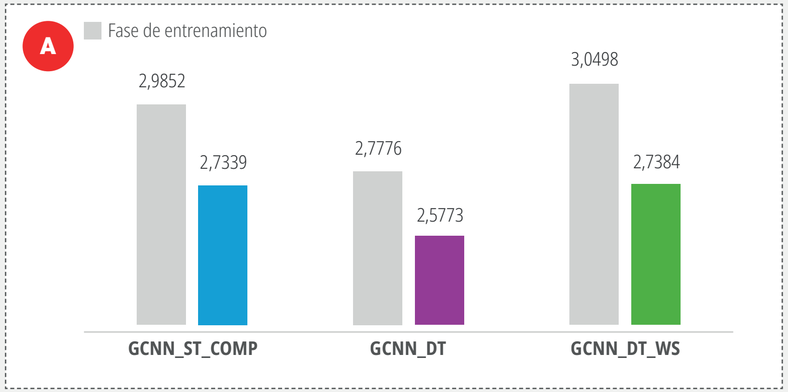
Si la pérdida durante el entrenamiento es mayor que durante la validación, esto es indicativo de un buen ajuste del modelo y una buena generalización (A). Si la pérdida durante el entrenamiento es menor que durante la validación, esto sugiere que hay algún problema en el proceso de entrenamiento o evaluación del modelo y puede ser indicativo de sobreajuste o problemas con el particionamiento de los datos (B).

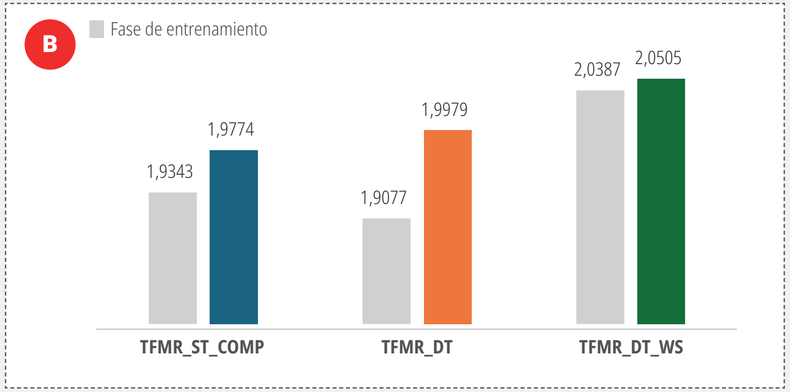
Nota: Una menor pérdida significa que el modelo está haciendo predicciones más precisas o está más cerca de la solución óptima.
Fuente: Elaboración propia del estudiante.

Pérdida - Traducción EN a FR (post-validación)

Un gap negativo entre la pérdida durante el entrenamiento y la validación es generalmente una señal de alerta.





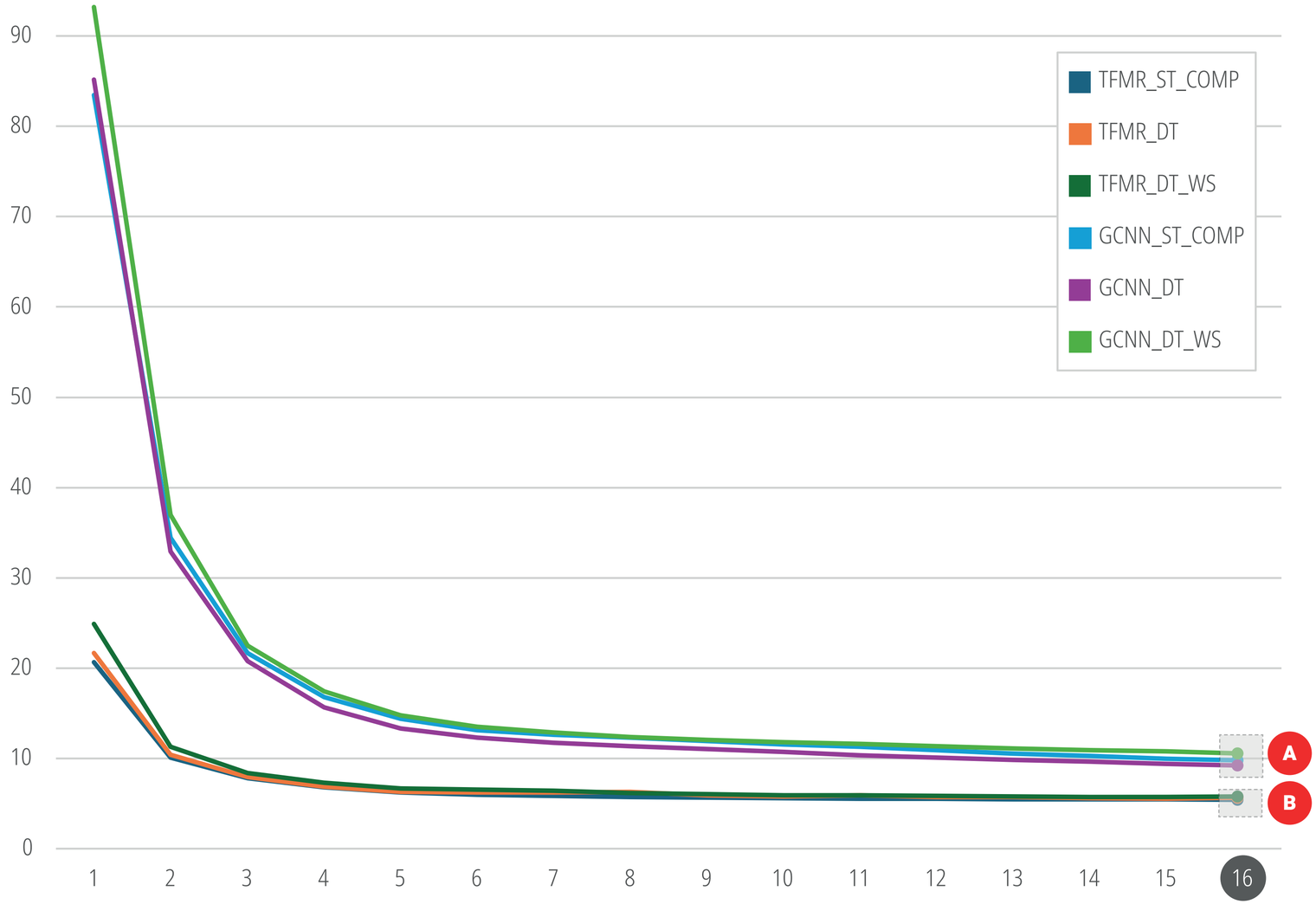


Si la pérdida durante el entrenamiento es mayor que durante la validación, esto es indicativo de un buen ajuste del modelo y una buena generalización (A). Si la pérdida durante el entrenamiento es menor que durante la validación, esto sugiere que hay algún problema en el proceso de entrenamiento o evaluación del modelo y puede ser indicativo de sobreajuste o problemas con el particionamiento de los datos (B). Se invierte la situación de EN a ES.

Nota: Una menor pérdida significa que el modelo está haciendo predicciones más precisas o está más cerca de la solución óptima.
Fuente: Elaboración propia del estudiante.

Perplejidad - Traducción EN a ES (post-validación)

La diferencia entre la perplejidad de entrenamiento y validación es crucial para evaluar la capacidad de generalización del modelo y para identificar problemas como el sobreajuste o el subajuste.



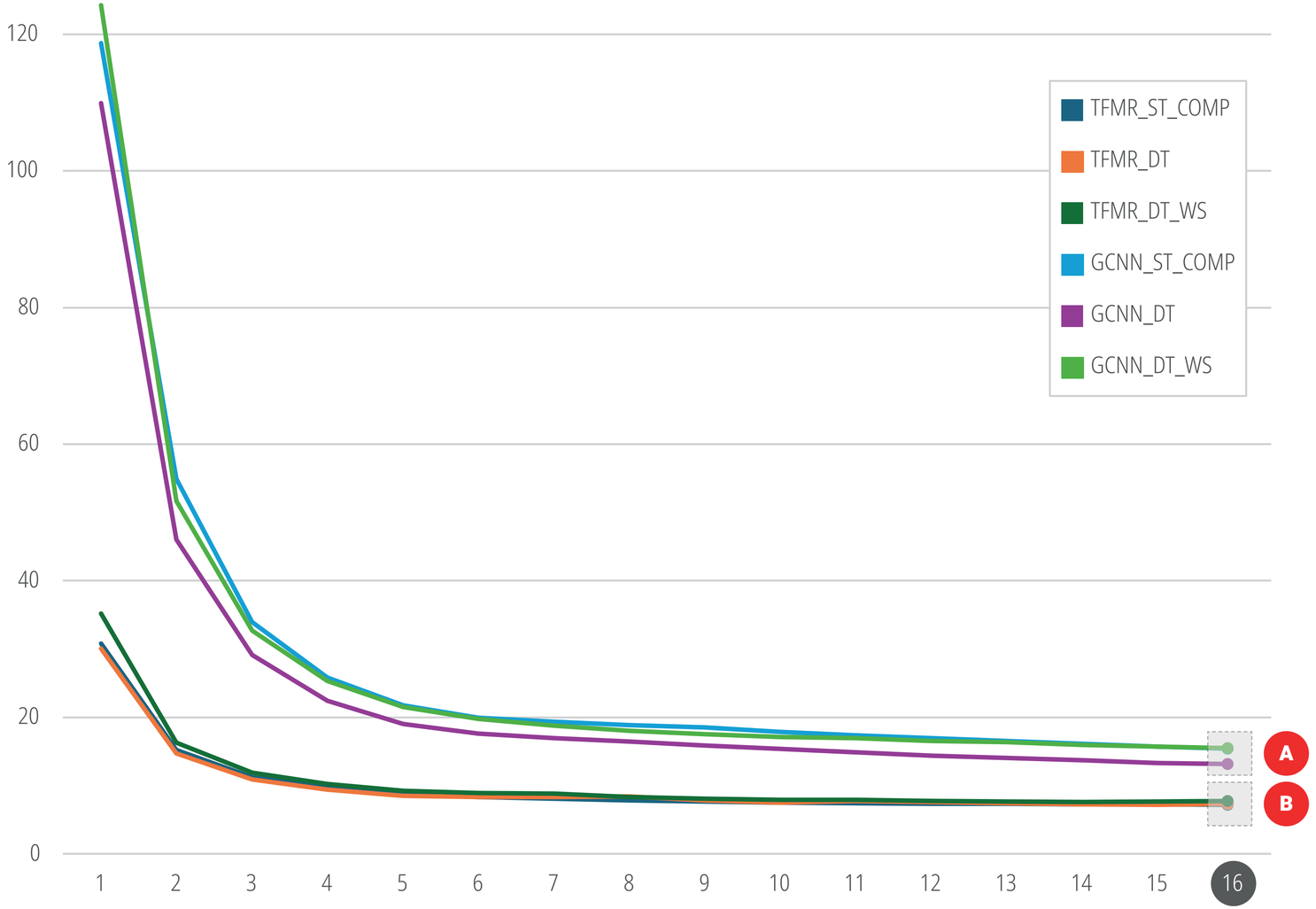
Una diferencia positiva (A) indica que el modelo tiene un mejor desempeño en los datos de entrenamiento que en los datos de validación.Esta discrepancia sugiere que el modelo puede no estar generalizando bien a datos nuevos y desconocidos.

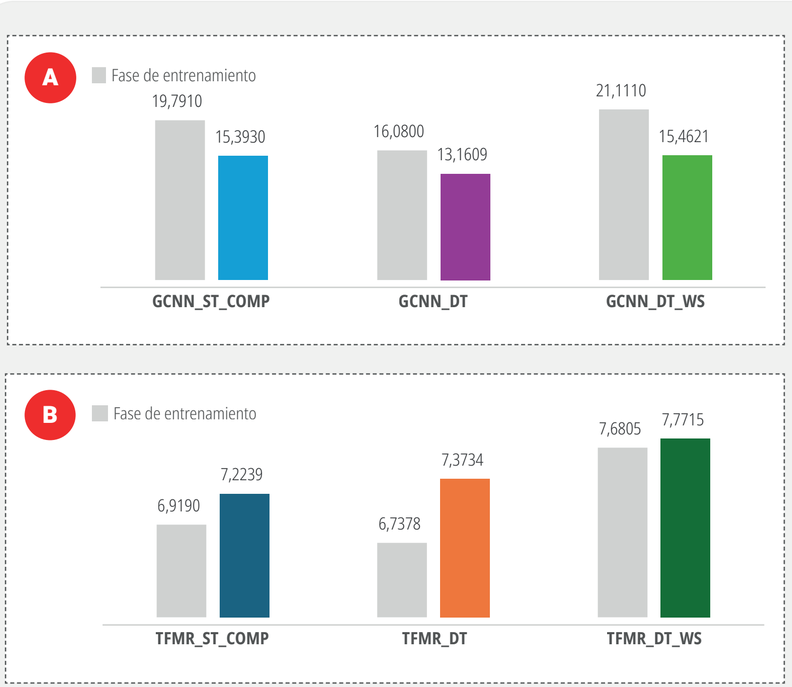
Una diferencia negativa (B) sugiere que el modelo está generalizando mejor de lo esperado, es decir, tiene un rendimiento mejor en datos nuevos y no vistos que en los datos utilizados para entrenamiento. Podría ser un indicio de algún problema en el proceso de entrenamiento o en la selección de datos.

Nota: Una perplejidad baja indica que el modelo puede predecir con confianza las palabras siguientes en una secuencia, mientras que una perplejidad alta indica incertidumbre en estas predicciones. Fuente: Elaboración propia del estudiante.

Perplejidad - Traducción EN a FR (post-validación)

La diferencia entre la perplejidad de entrenamiento y validación es crucial para evaluar la capacidad de generalización del modelo y para identificar problemas como el sobreajuste o el subajuste.





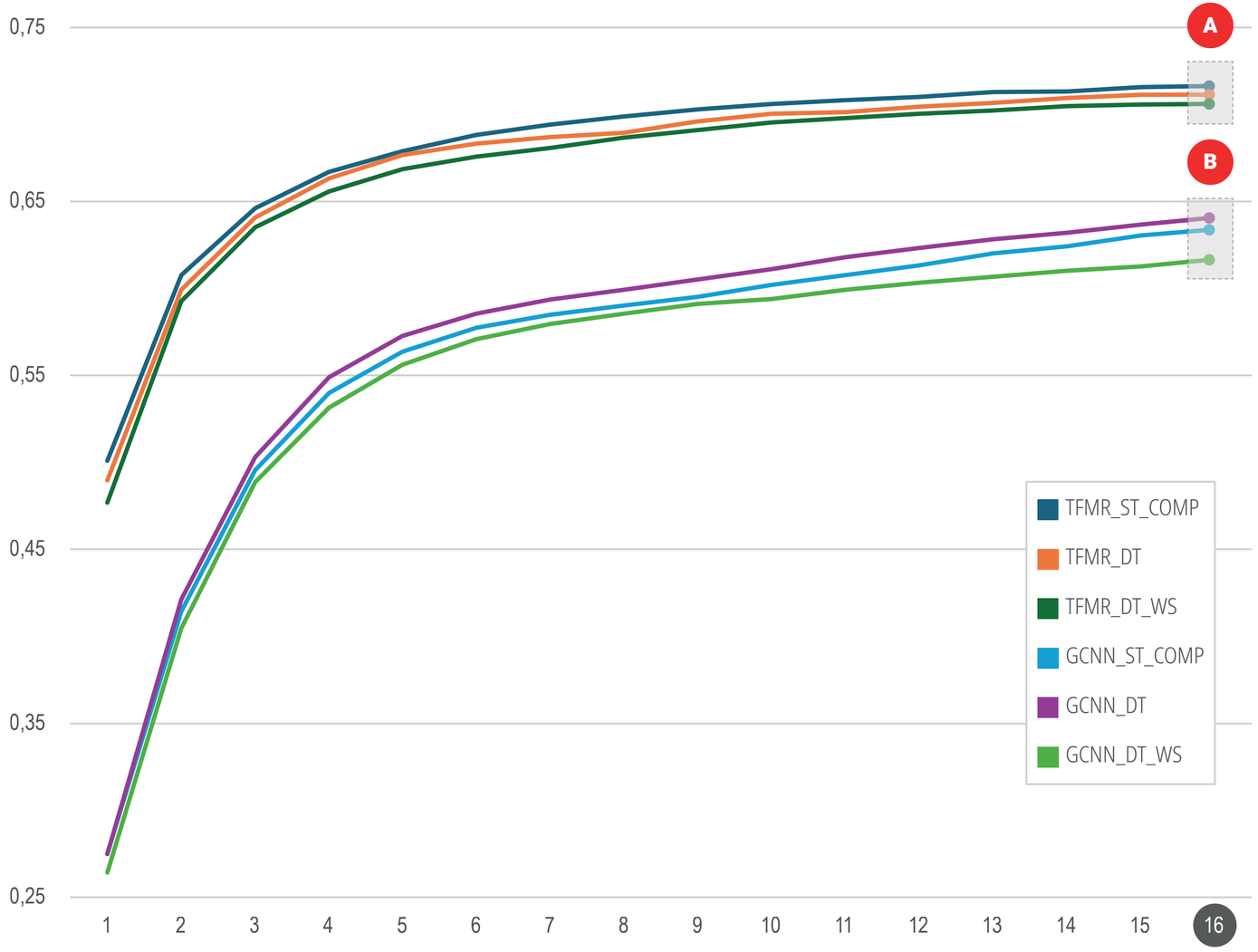
Una diferencia positiva (A) indica que el modelo tiene un mejor desempeño en los datos de entrenamiento que en los datos de validación.Esta discrepancia sugiere que el modelo puede no estar generalizando bien a datos nuevos y desconocidos.

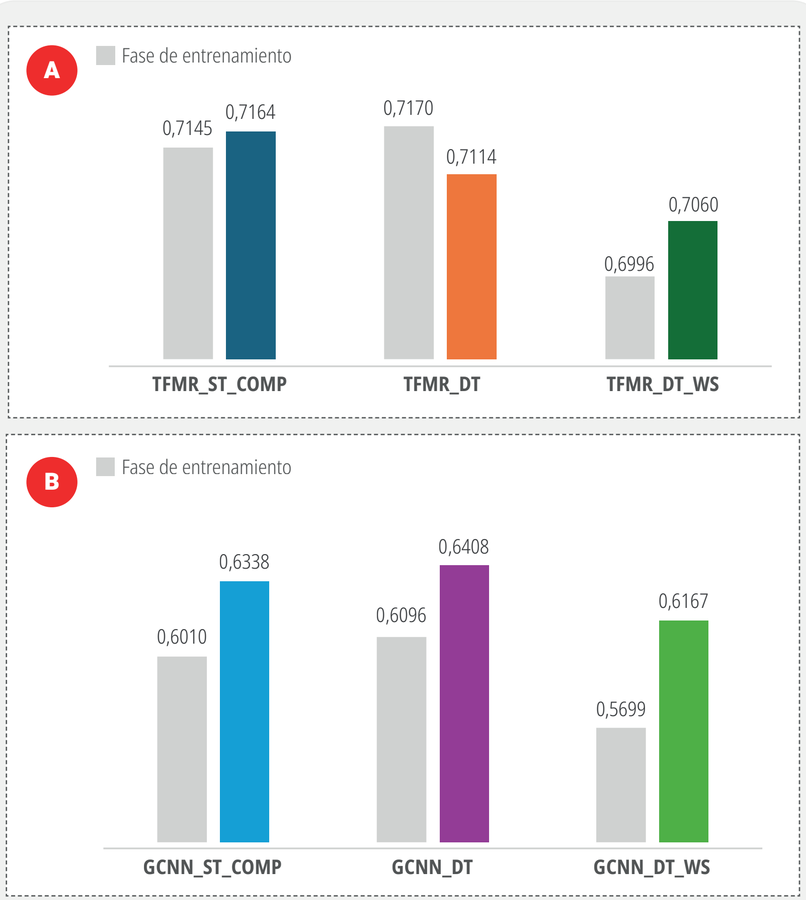
Una diferencia negativa (B) sugiere que el modelo está generalizando mejor de lo esperado, es decir, tiene un rendimiento mejor en datos nuevos y no vistos que en los datos utilizados para entrenamiento. Podría ser un indicio de algún problema en el proceso de entrenamiento o en la selección de datos.

Nota: Una perplejidad baja indica que el modelo puede predecir con confianza las palabras siguientes en una secuencia, mientras que una perplejidad alta indica incertidumbre en estas predicciones. Fuente: Elaboración propia del estudiante.

Exactitud - Traducción EN a ES (post-validación)

El gap[1] entre la exactitud proporciona información sobre la capacidad de generalización y el ajuste del modelo a los datos.



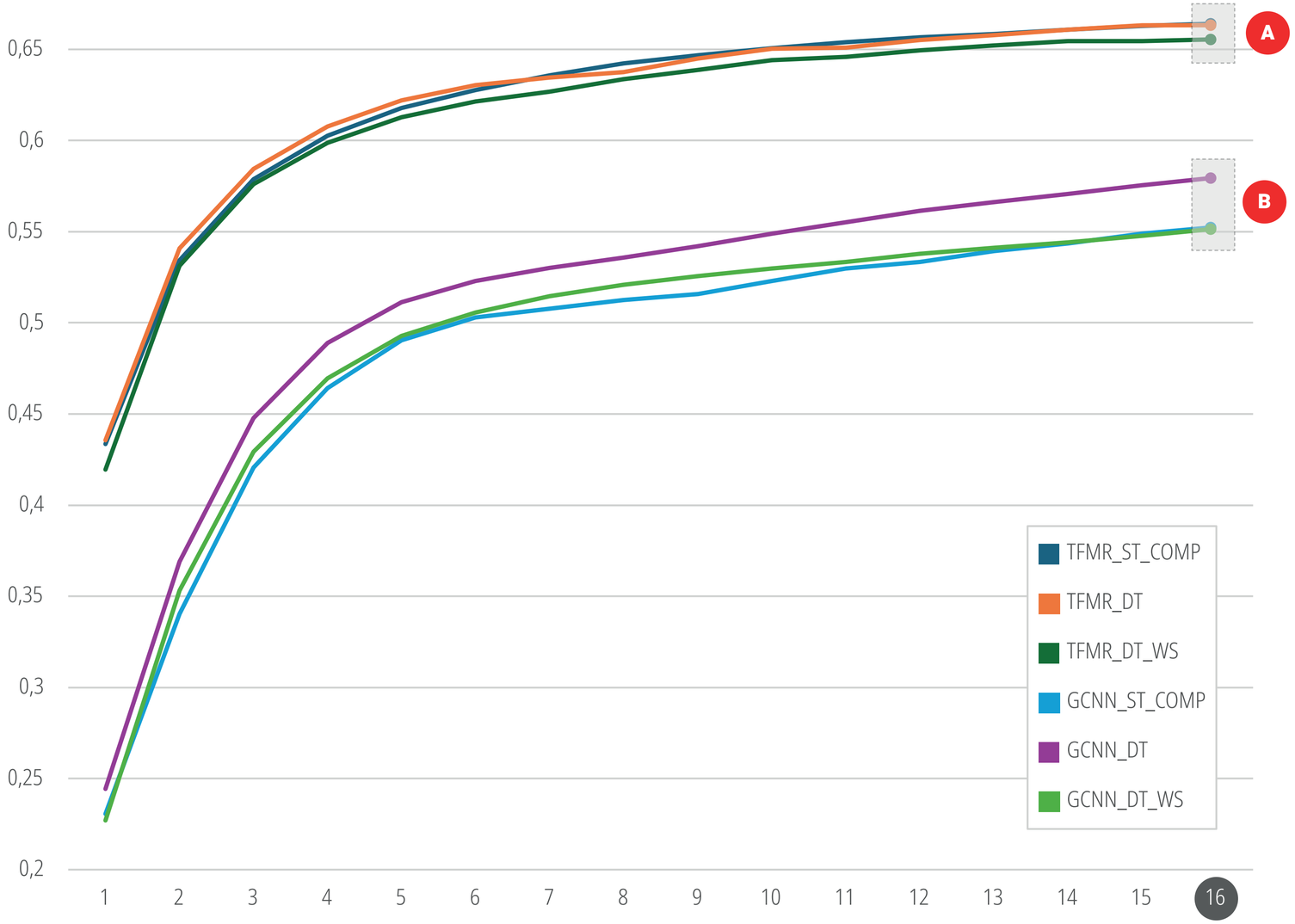


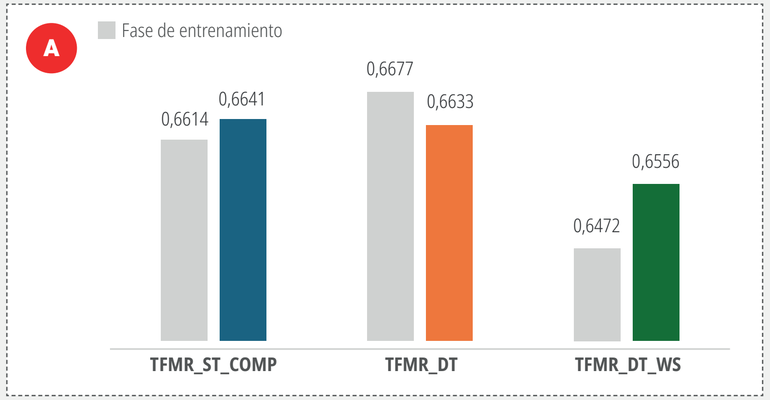
Casi todos los modelos exhiben una precisión mayor durante la validación que durante el entrenamiento (excepto TFMR_DT), esto podría ser indicativo de ciertos problemas o características en los datos o en el proceso de entrenamiento del modelo (overfitting, tamaño o capacidad insuficientes o muestreo arbitrario/desbalanceado del corpus).

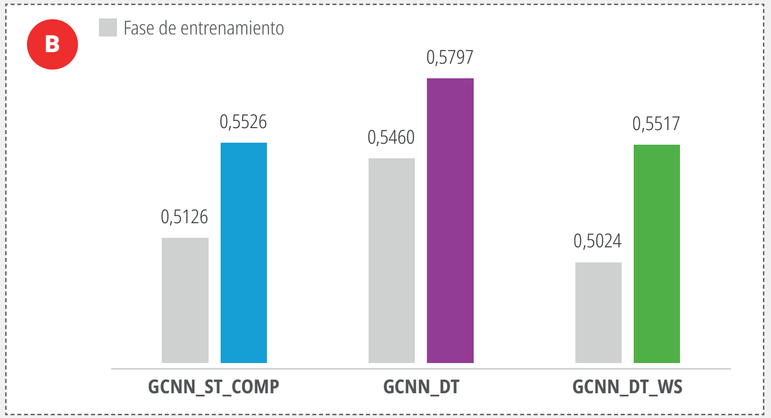
Nota: (1) El gap surge de comparar la métrica de exactitud del modelo durante el entrenamiento versus la validación (ejemplos no vistos).
Fuente: Elaboración propia del estudiante.

Exactitud - Traducción EN a FR (post-validación)

El gap[1] entre la exactitud proporciona información sobre la capacidad de generalización y el ajuste del modelo a los datos.







Como sucede en el caso anterior, casi todos los modelos exhiben una precisión mayor durante la validación que durante el entrenamiento, esto podría ser indicativo de ciertos problemas en los datos o en el proceso de entrenamiento del modelo (overfitting, tamaño o capacidad insuficientes y/o muestreo arbitrario del corpus).

Nota: (1) El gap surge de comparar la métrica de exactitud del modelo durante el entrenamiento versus la validación (ejemplos no vistos).
Fuente: Elaboración propia del estudiante.

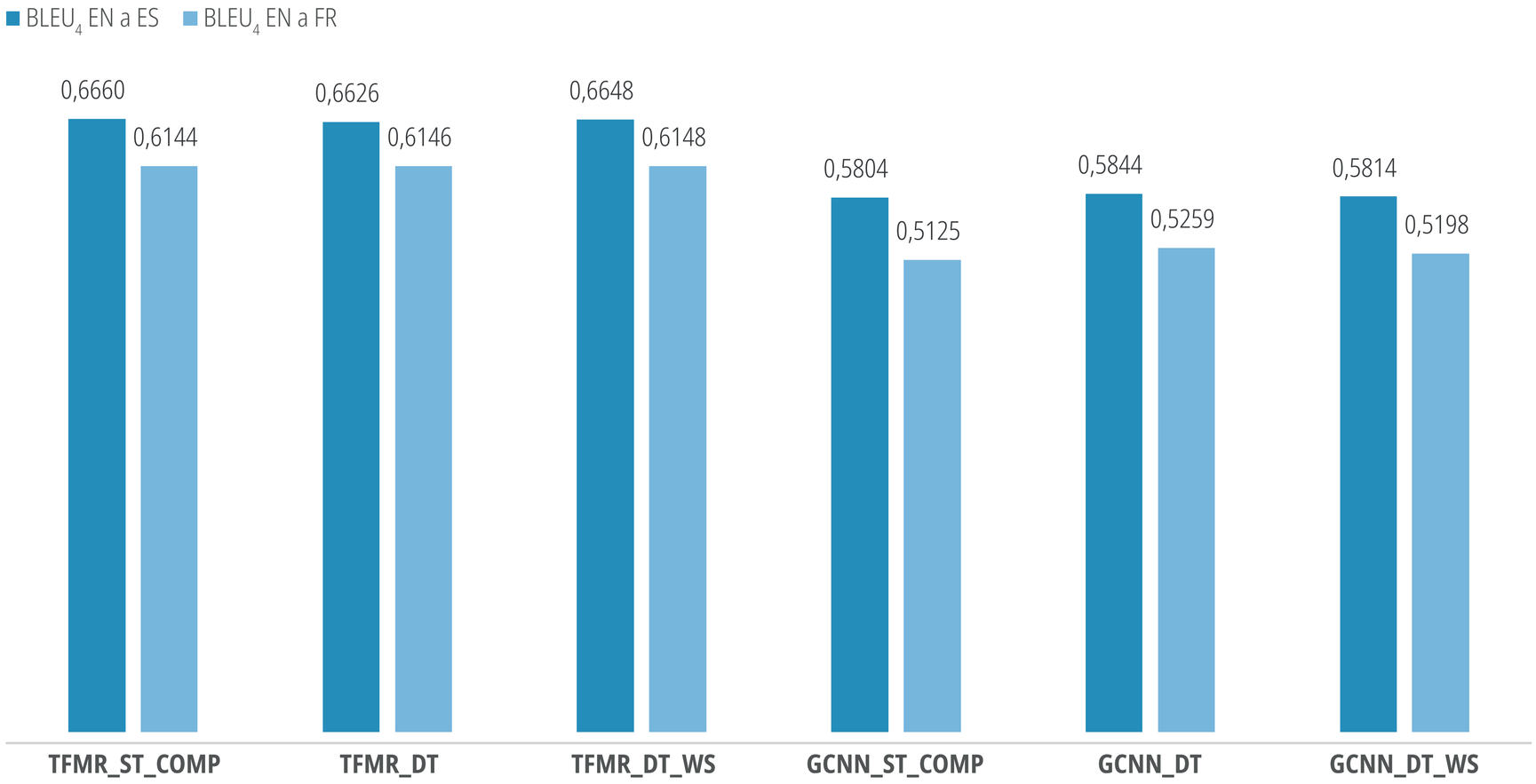
Inferencia - Ejemplos de resultados

Modelo	Origen (EN)	Destino (ES)			Destino (FR)		
		Referencia	Resultado	BLEU-4	Referencia	Resultado	BLEU-4
TFMR_DT	undcp will also continue to assist governments in enhancing judicial cooperation between countries	el pnufid tambien **seguira ayudando** a los gobiernos a fomentar la cooperacion judicial entre los paises	el pnufid **seguira ayudando** tambien a los gobiernos a mejorar la cooperacion judicial entre paises	**0,7995**	le pnucid continuera **egalement a aider** les gouvernements a renforcer lentraide judiciaire	le pnucid continuera **egalement a aider** les gouvernements a renforcer la cooperation judiciaire entre pays	**0,8286**
TFMR_DT_WS			el pnufid **seguira prestando** tambien **ayuda** a los gobiernos para mejorar la cooperacion judicial entre	0,7005		le pnucid continuera **egalement daider** les gouvernements a renforcer la cooperation judiciaire entre pays a	0,7311
TFMR_ST_COMP			el pnufid **seguira prestando** tambien **asistencia** a los gobiernos para mejorar la cooperacion judicial entre	0,7005		le pnucid continuera **egalement daider** les gouvernements a renforcer la cooperation judiciaire entre les pays	0,7121

Modelo	Origen (EN)	Destino (ES)			Destino (FR)		
		Referencia	Resultado	BLEU-4	Referencia	Resultado	BLEU-4
GCNN_DT	undcp will also continue to assist governments in enhancing judicial cooperation between countries	el pnufid tambien seguira **ayudando** a los gobiernos a fomentar la cooperacion judicial entre los paises	tambien seguira **prestando asistencia** a los gobiernos a que aumente la cooperacion judicial en la	0,6662	le pnucid continuera **egalement a aider** les gouvernements a renforcer lentraide judiciaire	il continuera **egalement daider** les gouvernements a ameliorer la cooperation judiciaire en matiere de promotion	0,5411
GCNN_DT_WS			tambien seguira **ayudando** a los gobiernos a que promuevan la cooperacion judicial la cooperacion judicial	**0,7350**		elle continue **egalement a aider** les gouvernements a renforcer la cooperation judiciaire la cooperation judiciaire	**0,6914**
GCNN_ST_COMP			tambien siguen **prestando asistencia** a los gobiernos a que promuevan la cooperacion judicial en la	0,6180		de plus en plus en **faveur de aider** les gouvernements a renforcer la cooperation judiciaire	0,5814

Inferencia - BLEU4

Los modelos que tienen una arquitectura afin, producen resultados con un BLEU-4 similar. A grandes rasgos las técnicas aplicadas parecen no afectar “significativamente” (a ojo) el desempeño del modelo.

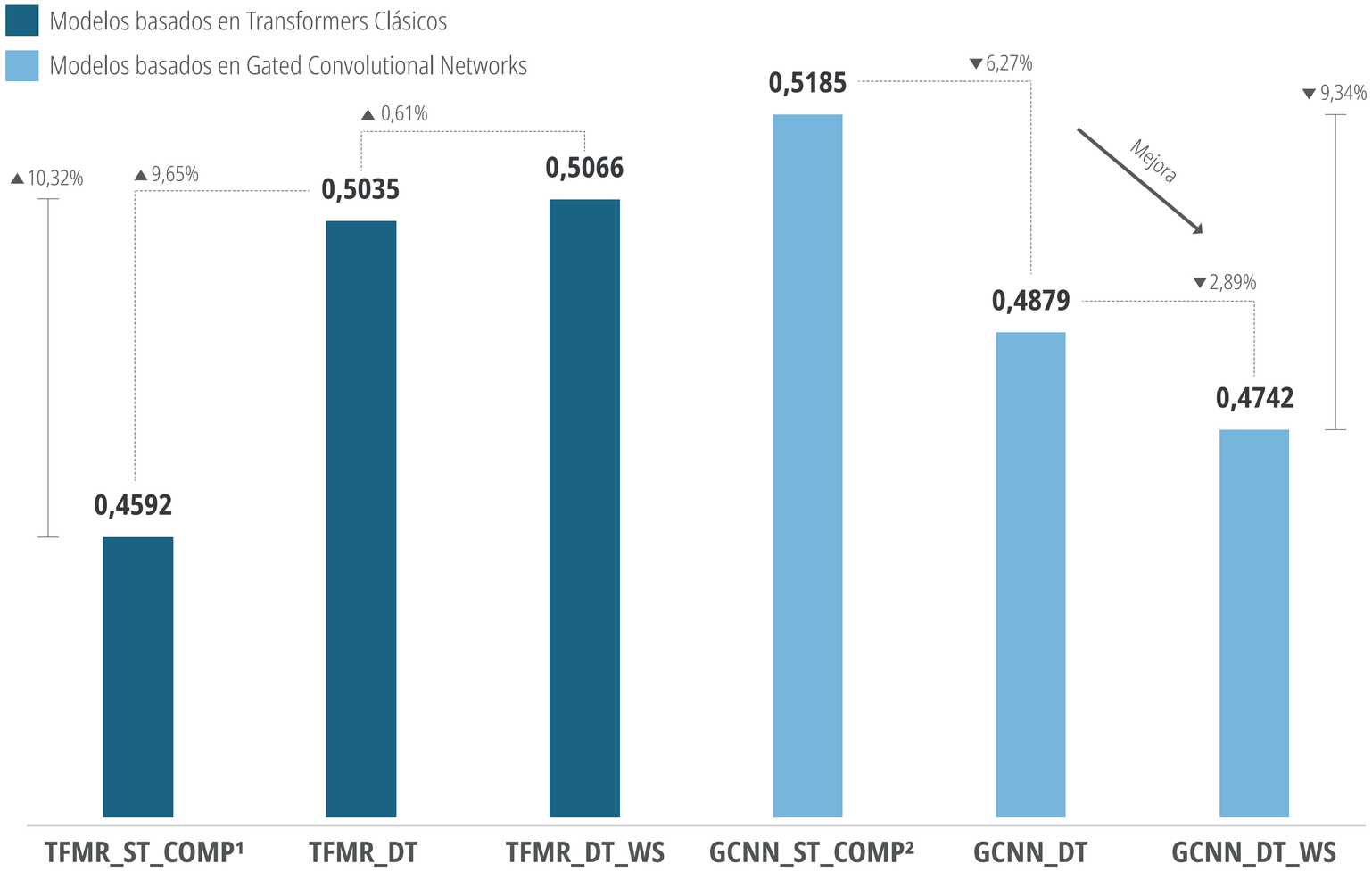


- Si el desempeño se mantiene similar cuando se implementan optimizaciones para reducir su tamaño, podría ser una señal de que ese modelo optimizado es preferible en ciertos contextos.
- **Eficiencia de recursos:** Un modelo más pequeño puede requerir menos memoria y poder de computación, lo que lo hace más eficiente en términos de recursos.
- **Tiempo de inferencia:** Los modelos más pequeños generalmente requieren menos tiempo para realizar inferencias, lo que puede ser crucial en aplicaciones en tiempo real o en dispositivos con recursos limitados.
- **Facilidad de implementación:** Un modelo más pequeño puede ser más fácil de implementar y distribuir en dispositivos con recursos limitados, como dispositivos móviles o sistemas embebidos.

Nota: BLEU (Bilingual Evaluation Understudy) es una métrica comúnmente utilizada para evaluar la calidad de las traducciones generadas por modelos de traducción automática. Es particularmente relevante en modelos de traducción neuronal automática (MTN), ya que proporciona una medida cuantitativa de qué tan cercanas son las traducciones automáticas generadas por el modelo a las traducciones de referencia realizadas por humanos. Fuente: Elaboración propia del alumno.

Inferencia - Tiempo promedio por ejemplo

Comparar el tiempo de inferencia entre diferentes modelos puede ayudar a seleccionar el modelo más adecuado para una aplicación específica.

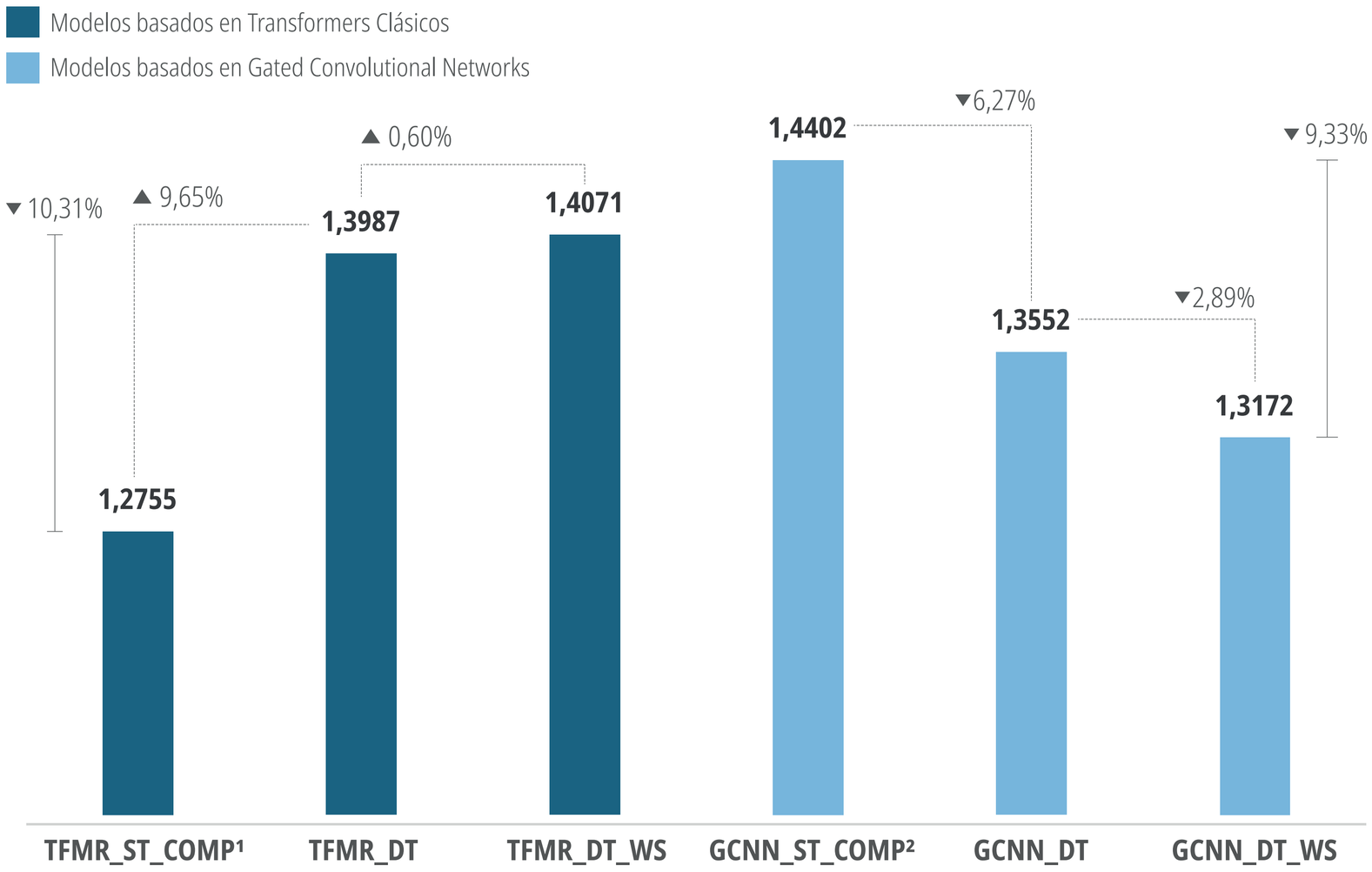


- Al implementar técnicas para reducir el tamaño del modelo podría llevar a un aumento en los tiempos de inferencia. Esto podría suceder si las técnicas introducen una sobrecarga computacional significativa durante la inferencia.
- Por otro lado, al combinar múltiples técnicas para reducir el tamaño del modelo, es posible que se logre una optimización global que resulte en tiempos de inferencia más rápidos.
- Esto podría deberse a que las técnicas individuales se complementan entre sí y compensan cualquier sobrecarga introducida por una técnica particular.
- El impacto en los tiempos de inferencia al implementar técnicas para reducir el tamaño del modelo puede variar dependiendo de la combinación específica de técnicas utilizadas, así como de las características del modelo y del hardware de destino.

Nota: Valores absolutos dados en segundos, tiempo promedio para cada ejemplo utilizando una muestra de 10000 ejemplos seleccionados aleatoriamente una vez y aplicada por igual a todos los modelos. (1) y (2) Se componen dos modelos ST independientes en una solución de NMT. Fuente: Elaboración propia del alumno.

Inferencia - Tiempo total

Comparar el tiempo de inferencia entre diferentes modelos puede ayudar a seleccionar el modelo más adecuado para una aplicación específica.



- Es fundamental que el modelo pueda realizar inferencias en un tiempo razonable para responder rápidamente a los eventos en entornos interactivos puede mejorar la experiencia del usuario al proporcionar respuestas más rápidas y fluidas.
- Un tiempo de inferencia más corto significa que el modelo puede procesar las entradas más rápidamente.
- Al medir el tiempo de inferencia, se puede evaluar qué tan bien utiliza el modelo los recursos disponibles, como la CPU, la GPU o la memoria.

Nota: Valores absolutos datos en horas, para procesar 10000 ejemplos seleccionados aleatoriamente una vez y aplicados por igual a todos los modelos.
(1) y (2) Se componen dos modelos ST independientes en una solución de NMT. Fuente: Elaboración propia del alumno.

Anexo 1: Artefactos (artefactos.zip)

Archivos entregados como parte de la tarea evaluable.

ARCHIVO	DESCRIPCIÓN
dl_main.py	Programa de entrenamiento, se ejecuta desde la línea de comando.
dl_inference.py	Programa de inferencia, se ejecuta desde la línea de comando.
dl_common.py	Constantes globales, algunas variables globales compartidas.
dl_common_preprocessing.py	Funciones de preprocesamiento de datos.
dl_xxxx_models.py	Funciones compartidas entre los modelos.
dl_tfmr_models.py	Definición de los modelos basados en Transformer.
dl_tfmr_routines.py	Rutinas de entrenamiento y prueba, ciclo completo de entrenamiento para Transformers.
dl_gcnn_models.py	Definición de los modelos basados en Gated Convolutional Networks (GCNN).
dl_gcnn_routines.py	Rutinas de entrenamiento y prueba, ciclo completo de entrenamiento para GCNN.
dl_greddy_search.py	Rutinas de prueba de los modelos mediante búsqueda ávida y evaluación del BLEU.

Anexo 2: Salidas de datos (measures.zip)

Archivos entregados como parte de la tarea evaluable.

ARCHIVO	DESCRIPCIÓN
tfmr_LP.csv	Contiene resultados de métricas de pérdida y exactitud de modelos basados en transformers.
tfmr_PF.csv	Contiene resultados de métricas de memoria y duración del entrenamiento y evaluación de modelos basados en transformers.
gcnn_LP.csv	Contiene resultados de métricas de pérdida y exactitud de modelos basados en Gated CNN.
gcnn.PF.csv	Contiene resultados de métricas de memoria y duración del entrenamiento y evaluación de modelos basados en Gated CNN.
metrics_tfmr_st_models.csv	Métricas individuales para modelos de un solo decodificador basados en transformers.
metrics_tfmr_dt_models.csv	Métricas individuales para modelos de doble decodificador basados en transformers.
metrics_gcnn_st_models.csv	Métricas individuales para modelos de un solo decodificador basados en Gated CNN.
metrics_gcnn_dt_models.csv	Métricas individuales para modelos de doble decodificador basados en Gated CNN.
st_inference.csv	Resultados de la inferencia de modelos de un único decodificador.
dt_inference.csv	Resultados de la inferencia de modelos de doble decodificador.
st_samples.csv	Ejemplos de resultados de la inferencia de modelos de un único decodificador.
dt_samples.csv	Ejemplos de resultados de la inferencia de modelos de doble decodificador.

Conclusiones y oportunidades de mejora

La innovación en Deep Learning resulta muy restrictiva en el ámbito académico, pues el acceso a capacidades computacionales de HPC es una limitante para entrenar un modelo predictivo de gran tamaño o utilizando un corpus grande.

Las oportunidades de mejora:

- Implementar HOGWILD! para el ciclo de entrenamiento.
- Completar el ciclo de entrenamiento (más allá de la EPOCH 16) y tratar de incluir la mayor cantidad de documentos del corpus.
- Valorar una depuración más fina del conjunto de datos pues se encontraron letras sueltas y un tratamiento insuficiente de apostrofes en el corpus.
- Implementar modelos con mayor capacidad que permitan representar un mayor número de simbolos del vocabulario y mayor cantidad de unidades.
- Explorar funciones de pérdida alternativas (CE + DICE) y/o optimizadores más nuevos o duales (independientes para el encoder y decoder).
- Utilizar REFORMERS (LSH) / MAMBA.

Referencias

Libros y artículos consultados.

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). **Neural machine translation by jointly learning to align and translate**. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.

Bishop, C., & Bishop, H. (2024). **Deep Learning: Foundations and Concepts**. Switzerland: Springer Cham. doi:https://doi.org/10.1007/978-3-031-45468-4

Bradbury, J., Merity, S., Xiong, C., & Socher, R. (2017). **Quasi-recurrent neural networks**. 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings.

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). **Language modeling with gated convolutional networks**. 34th International Conference on Machine Learning, ICML 2017, 2.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. (2017). **Convolutional Sequence to Sequence Learning**. doi:https://doi.org/10.48550/arXiv.1705.03122

Goodfellow, I., & Bengio, Y. (2016). **Deep Learning**. London, England: MIT Press.

Hu, J. C., Cavicchioli, R., Berardinelli, G., & Capotondi, A. (2023). **Heterogeneous Encoders Scaling In The Transformer For Neural Machine Translation**. http://arxiv.org/abs/2312.15872

Kamath, U., Graham, K., & Emara, W. (2022). **Transformers for Machine Learning: A deep dive**. CRC Press.

Koehn, P. (2020). **Neural Machine Translation**. Cambridge University Press. doi:https://doi.org/10.1017/9781108608480

Le, H., Gu, J., Pino, J., Schwab, D., Wang, C., & Besacier, L. (2020). **Dual-decoder Transformer for Joint Automatic Speech Recognition and Multilingual Speech Translation**. Proceedings of the 28th International Conference on Computational Linguistics. doi:https://doi.org/10.48550/arXiv.2011.00747

Prince, S. (2023). **Understanding Deep Learning**. The MIT Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). **Attention is all you need**. Advances in Neural Information Processing Systems, 2017-December.

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Wang, L., Chen, Z., Xiao, A., Chang, J., Zhang, X., & Tian, Q. (2022). **Weight-Sharing Neural Architecture Search: A Battle to Shrink the Optimization Gap**. In ACM Computing Surveys (Vol. 54, Issue 9). https://doi.org/10.1145/3473330

Yang, S., Wang, Y., & Chu, X. (2020). **A Survey of Deep Learning Techniques for Neural Machine Translation**. doi:https://doi.org/10.48550/arXiv.2002.07526

¡Muchas gracias!