Data Science

Proyecto 2 – Entrega 2

I. Investigación

Resumen del problema a resolver

El problema es reconocimiento y traducción de fingerspelling (ASL) a partir de secuencias temporales de keypoints (x,y,z) por frame extraídos con MediaPipe. Es un problema supervisado de series temporales multivariadas y reconocimiento de gestos, similar a reconocimiento de escritura (trayectorias), reconocimiento de gestos con sensores y modelado de series temporales con alta dimensionalidad (análogo a la gran cantidad de puntos clave que se tienen registrados en el dataset) (Paparaju, 2018).

Implicaciones prácticas:

- Cada frame (una observación) es un vector alto dimensional (concatenación de todos los keypoints).
- Importante modelar la dependencia temporal (movimiento)
- Posible modelar estructura espacial (conexiones anatómicas entre extremidades o puntos clave).
- Etiqueta objetivo: secuencia de letras o palabras por secuencia (grupo determinado de frames).

Estructura general del pipeline

1. **Lectura y preprocesamiento de las secuencias**
   a. Agrupación de frames por sequence_id
   b. Normalización de coordenadas
   c. Padding/truncation de secuencias a longitud fija[1].

      Los modelos de deep learning requieren que todos los ejemplos de un batch tengan la misma dimensión. Como cada secuencia puede tener longitudes distintas, se usa padding (rellenar con ceros) para las secuencias más cortas.

[1] Código de referencia para hacer padding para secuencias:
https://www.tensorflow.org/api_docs/python/tf/keras/utils/pad_sequences

Esto permite procesar múltiples secuencias simultáneamente en una misma pasada por la red (GeeksForGeeks, 2025).

2. **Reducción de dimensionalidad (feature extraction)**[2]
   Feature extraction es una técnica para reducir la dimensionalidad o la complejidad de la data para mejorar la eficiencia de los algoritmos de machine learning. Simplifica los datos para mantener solo las variables o atributos más importantes. Mientras más características tena que manejar un modelo, menos eficiente será. Las aplicaciones más comunes de feature extraction son procesamiento de imágenes, procesamiento de lenguaje natural (NLP) y procesamiento de señales (Winland, s.f.).

   Cómo funciona (Winland, s.f.)

   1. El modelo toma la data input.
   2. El **feature extractor** transforma la data a representación numérica (en el caso del presente proyecto, esta condición ya se cumple).
   3. La data numérica se guarda en **feature vectors** para que el modelo realice los algoritmos de reducción de dimensionalidad.
   4. Después de la extracción, se estandariza la data usando **feature normalization**.

3. **Modelado secuencial**
   Se prevé que el modelo principal que se utilizará para resolver el problema planteado es uno de Transformers. Se podrá comparar con otras arquitecturas, como TCN o LSTM. En caso de malos resultados, se integrarán arquitecturas adicionales como GNN. Ver sección siguiente para más detalle sobre las arquitecturas.

4. **Decodificación**
   La salida del modelo será una secuencia de texto, representando la frase que describe la secuencia en ASL.

Algoritmos más utilizados en problemas similares

1. **Reducción de dimensionalidad**

[2] Tutorial con clasificación de métodos por tema (Kumar, 2025): https://www.datacamp.com/tutorial/feature-extraction-machine-learning

a. **Análisis de componentes principales (PCA)**
   Reduce la cantidad de features en datasets grandes a sus principales componentes. PCA es popular por su habilidad de crear data original no correlacionada (linealmente independiente). Es una buena solución para evitar sobreajuste (Winland, s.f.).

b. **Autoencoders**[3]
   Los autoencoders son un tipo de arquitectura de red neuronal diseñada para comprimir (encode) data de un input a sus componentes principales y luego reconstruir (decode) el input original de su representación comprimida. Utiliza machine learning no supervisado. Estos modelos están diseñados para descubrir variables "latentes" en la data de input: variables ocultas o aleatorias que, aunque no son directamente observables, afectan fundamentalmente la manera en que los datos se distribuyen (Bergmann & Stryker, s.f.).

   La principal diferencia de un autoencoder con modelos encoder-decoder como CNN y RNN, es que los encoder-decoder toman un input y lo transforman a un output distinto. En los autoencoders, el **input y el output describen la misma data**. La ventaja de los autencoders sobre PCA es que estos capturan correlaciones **no lineales** (Bergmann & Stryker, s.f.).

2. **Modelado secuencial**

   Los modelos más populares para modelar secuencais son:

a. **RNN, LSTM y GRU**
   Son los modelos base más utilizados en tareas de secuencias de keypoints. Algunas variables para tomar en cuenta: BiLSTM (bidirectional LSTM – ve la secuencia completa para aprender dependencias hacia adelante y hacia atrás), Stacked LSTM (múltiples capas para capturar distintos niveles de abstracción).

b. **Redes Convolucionales Temporales (TCN)**
   Son una alternativa a las LSTM. Usan convoluciones 1D con dilatación para capturar dependencias a largo plazo sin recurrencias. Ventaja: paralelización más eficiente, utilidad cuando las secuencias son muy largas. Las TCN también

[3] Código de referencia (GeeksForGeeks, 2025): https://www.geeksforgeeks.org/machine-learning/auto-encoders/

son flexibles con el largo de las secuencias (no es necesario que todas tengan la misma longitud. Usualmente se utilizan para series temporales y modelado de secuencias (NLP) [4] (Yadav, 2024).

c. Transformers

Los transformers son un tipo de arquitectura de red neuronal muy reciente, especial para procesar data secuencial, usualmente relacionados con los LLM (large language models), pero también útiles para datos como series temporales, visión por computadora, reconocimiento de voz, entre otros (Cihan, Koller, Hadfield, & Bowden, 2020). Utilizan mecanismos de atención, que examinan secuencias completas simultáneamente y deciden cómo y cuándo concentrarse en pasos específicos de esa secuencia (Stryker & Bergmann, s.f.).

La principal ventaja de los transformers es que mejoran significativamente la habilidad de los modelos de comprender dependencias a largo plazo. Esto, a su vez, permite la paralelización (pueden realizar varios pasos computacionales simultáneamente) (Stryker & Bergmann, s.f.).

d. Graph Neural Networks (GNN)

Dado que los datos con los que se está trabajando en el presente proyecto tienen estructura de grafo (conexiones entre articulaciones), se pueden utilizar modelos como GNN. Estos modelos aprenden cómo se mueven las articulaciones relativas con el tiempo. Los principales casos de uso para las GNN son NLP (modelar relaciones entre palabras u oraciones en documentos), visión por computadora, segmentaciones de imágenes, detección de objetos, bioinformática, entre otros (Awan, 2022).

e. Modelos híbridos

Es posible combinar varios de los modelos mencionados para aprovechar las características de cada uno, según los datos que se tengan y el problema que se quiera resolver.

A continuación, se puede ver un a tabla de comparación con las características principales de los modelos para datos secuenciales (GeeksForGeeks, 2025).

[4] Código de referencia para una serie temporal con TCN: https://medium.com/@amit25173/temporal-convolutional-network-an-overview-4d2b6f03d6f8

Parámetro	RNN	LSTM	GRU	Transformers
Arquitectura	Estructura simple con loops.	Células de memoria con gates de input, forget y output.	Combina input y forget gates en el update gate, menos parámetros	Usa mecanismos de atención sin recurrencia.
Secuencias argas	No mantiene dependencias a largo plazo por vanishing gradients.	Bueno para capturar dependencias de largo plazo.	Mejor que las RNN pero un poco peor que LSTM para dependencias de largo plazo.	Maneja secuencias largas efectivamente, usando mecanismos de autoatención.
Tiempo de entrenamiento	Rápido pero menos acertado con data compleja.	Lento por la cantidad de operaciones de memoria.	Más rápido que las LSTM pero más lento que las RNN.	Necesita poder de computación pero permite entrenamiento paralelo.
Uso de memoria	Bajos requerimientos de memoria.	Consumo alto de memoria por complejidad de arquitectura.	Bajo consumo de memoria comparado a LSTM pero alto comparado con RNN.	Alto consumo de memoria por el multi-head attention y feed-forward layers.
Cantidad de parámetros	Baja cantidad de parámetros en general.	Más parámetros que las RNN por la cantidad de gates y células de memoria.	Menos parámetros que las LSTMs por la estructura simplificada.	Gran cantidad de parámetros por las capas de multi-head attention.
Facilidad de entrenamiento	Suceptible al vanishing gradient problem, no bueno para secuencias largas.	Más fácil de entrenar para secuencias largas por manejo bueno de gradiente.	Más simple que las LSTMs y más fáciles de entrenar que las RNN.	Necesita poder computacional alto y GPUs.
Casos de uso	Útil para secuencias simples como precios.	Ideal para series temporales, generación de texto y tareas que necesitan dependencias a largo plazo.	Aplicaciones similares a las LSTM pero preferido cuando la eficiencia computacional es importante.	Se usa para tareas de NLP como traducción, sumarización, visión por computadora y procesamiento de lenguaje de voz.
Paralelismo	Limitado.	Mismas limitaciones que las RNNs; el procesamiento secuencial restringe el paralelismo.	Mismas limitaciones que las RNNs; el procesamiento secuencial restringe el paralelismo.	Alto paralelismo permitido por el mecanismo de atención y el diseño no secuencial.
Performance con secuencias largas	Bajo	Bueno	Moderado a bueno	Excelente

II. Resultados

Corrida 1

Tabla 1: configuración de corrida 1	
Parámetro	**Valor**
Optimizador	AdamW
Learning rate	$1 + 10^{-4}$
Warmup Ratio	0.05
Scheduler	Coseno
Gradient Clipping	1.0
Label smoothing	0.05
Batch size	16
Épocas ejecutadas	6

En la corrida 1 ambos modelos de entrenaron con la misma configuración de optimización y regulación. Se empleó AdamW con una tasa de aprendizaje inicial de $1 + 10^{-4}$ y scheduler coseno con 5% de warm-up. Se aplicó gradient clipping (1.0) y label smoothing (0.05). El batch size fue de 16. En esta corrida se registraron 6 épocas efectivas de entrenamiento. La misma información se puede ver resumida en la Tabla 1.

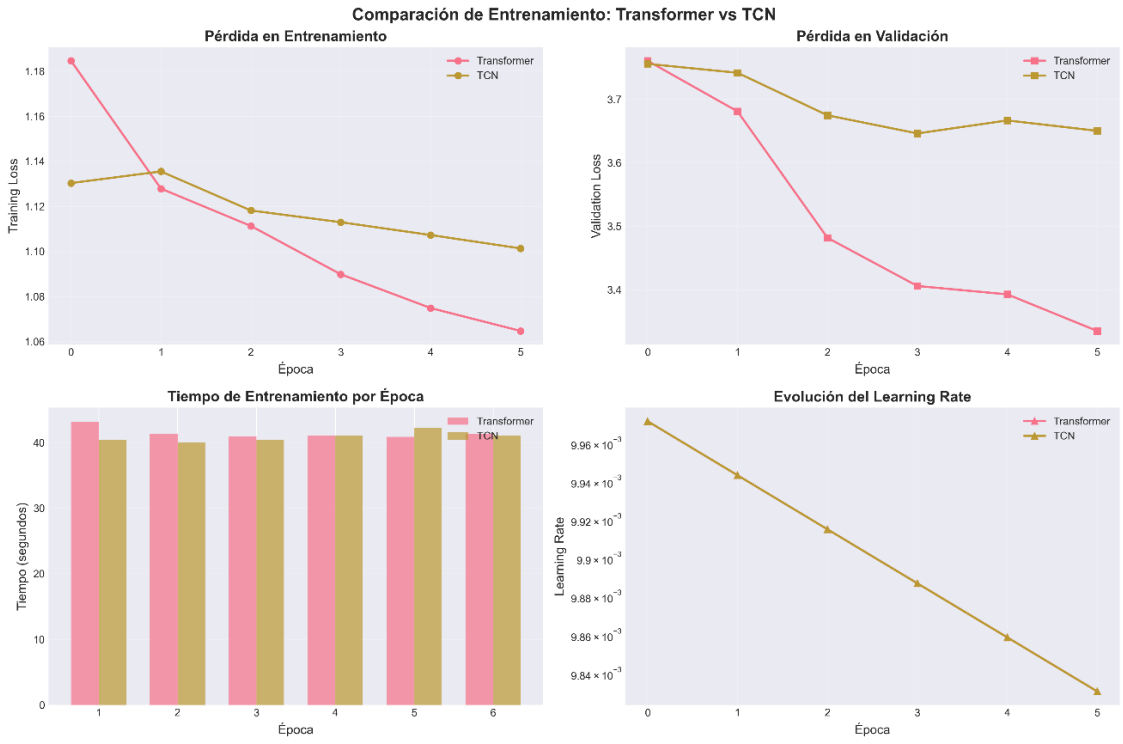

Ilustración 1: Curvas de pérdida (Corrida 1)

Las curvas de la ilustración 1 muestran que el Transformer reduce de forma monótona la pérdida de entrenamiento y también la pérdida de validación a lo largo de 6 épocas. Por otro lado, TCN presenta una disminución leve en entrenamiento mientras que la pérdida de validación permanece alta y casi plana. En tiempo de cómputo, TCN resulta consistentemente más rápido por época que Transformer. La tasa de aprendizaje decrece de forma sincronizada en ambos modelos debido al scheduler coseno.

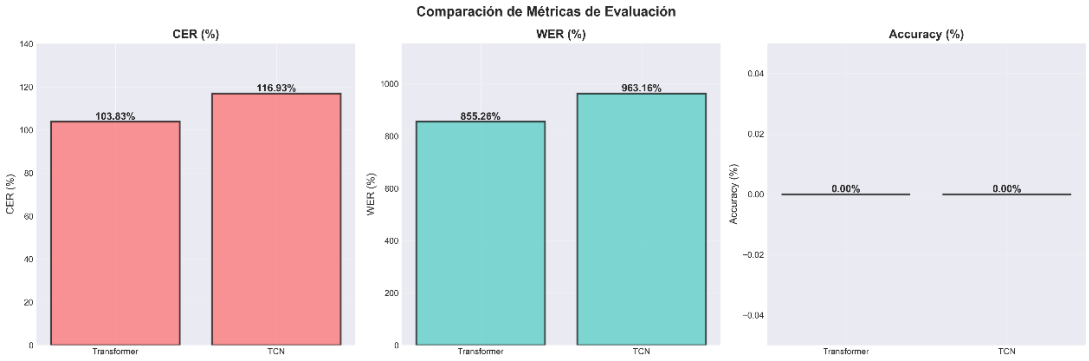

Ilustración 2: Comparación de métricas (Corrida 1)

En la evaluación, el Transformer obtiene CER = 103.83% y WER = 855.26%, mientras que TCN registra CER = 116.93% y WER = 963.16%. El Accuracy de palabra completa es 0.00% para ambos modelos. Por lo tanto, en esta corrida el Transformer supera al TCN en CER y WER, aunque ninguno logra aciertos exactos de secuencia completa.

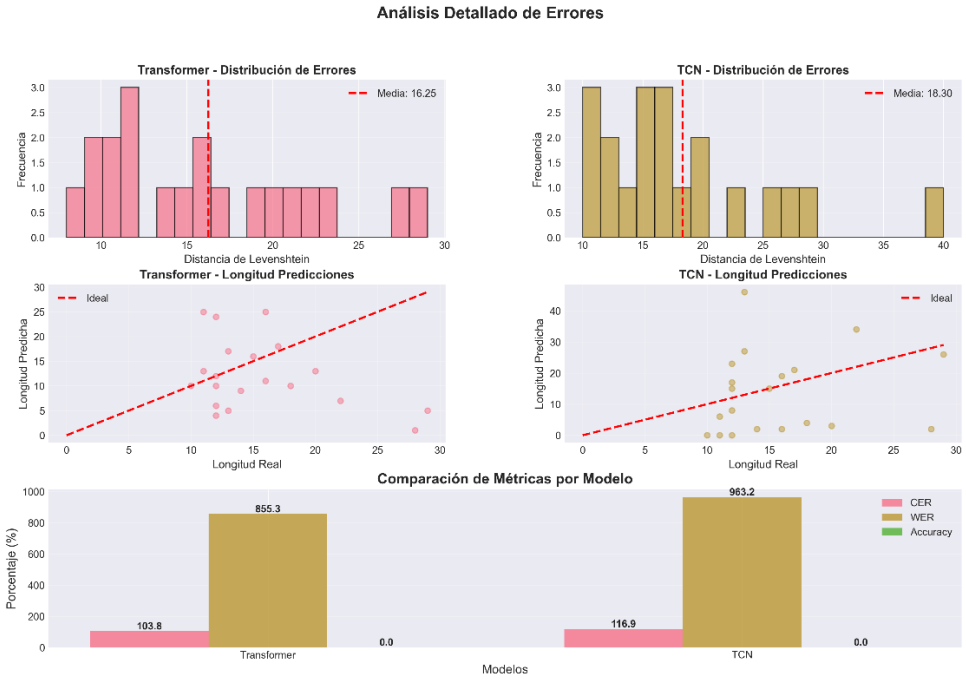

Ilustración 3: Distribución de distancias (Corrida 1)

En la ilustración 3 se presenta la distribución de errores por distancia de Levenshtein y la dispersión de longitudes. El promedio de distancia es 16.25 para Transformer y 18.30 para TCN, indicando menores ediciones requeridas en el primero. Los diagramas de dispersión longitud real vs. longitud predicha se alejan de la diagonal ideal en ambos modelos, evidenciando desajustes de longitud. En el panel inferior se confirma la ventaja del Transformer sobre TCN en CER y WER, con Accuracy nulo en los dos casos.

Tabla 2: Prediccion vs realidad (Corrida 1)

	modelo	real	pred
1	Transformer	mio-footwear/refugee	5ww384-8 od.t
2	Transformer	www.filmdb.it	4-6ym
3	Transformer	www.lemon.co.jp	n.d.rwlswptna_e9
4	Transformer	https://www.pintradingdb.com	w
5	Transformer	981396 3sl	9w630.stmr
6	Transformer	6169 valley view parks	11w-wg.
7	Transformer	2245 winston hill	7-tton adwgtjuew g
8	Transformer	la-mejor-dama-honor/equindiee	889w8
9	Transformer	sasha quinn	17-80w-*oiomo
10	TCN	mio-footwear/refugee	2@5
11	TCN	www.filmdb.it	a,-@.6en2d9nw2bsr8?9w3i4w-e
12	TCN	www.lemon.co.jp	t-o72i510the9rr
13	TCN	https://www.pintradingdb.com	sl
14	TCN	981396 3sl	
15	TCN	6169 valley view parks	ssnc7 k0riw1a1he2r2ed3ajoaf e19r25
16	TCN	2245 winston hill	rsts5rrsew osaopnre-7
17	TCN	la-mejor-dama-honor/equindiee	679r78[-w9s- o8u f-kc7%r33
18	TCN	sasha quinn	
19	TCN	thien-dao-game-thu	9vo!
20	TCN	288-035-2251	

Como evidencia cualitativa, la Tabla 2 presenta ejemplos representativos de la Corrida 1 extraídos de predictions.csv (20 pares por modelo). Se observa que, aun cuando algunas letras o segmentos coinciden, las secuencias predichas difieren sustancialmente de las reales, en concordancia con los valores altos de WER/CER y con la Accuracy de 0%.

En la corrida 1, bajo la misma configuración de entrenamiento para ambos modelos, el Transformer mostró mejor convergencia en validación y menores errores promedio que TCN, aunque el TCN fue más eficiente en tiempo por época. Ninguno de los modelos alcanzó aciertos exactos de palabra completa, y las longitudes predichas presentan desviaciones importantes respecto de las reales.

Corrida 2

Tabla 3: configuración de corrida 2	
Parámetro	**Valor**
Optimizador	AdamW
Learning rate	$1 + 10^{-4}$
Warmup Ratio	0.05
Scheduler	Coseno
Gradient Clipping	1.0
Label smoothing	0.05
Batch size	16
Épocas ejecutadas	10

En la Corrida 2 se mantuvo la misma configuración de optimización y regularización utilizada previamente. Se entrenaron Transformer y TCN con AdamW, learning rate 1×10^{-4}, scheduler coseno con 5% de warm-up, gradient clipping 1.0 y label smoothing 0.05. El batch size fue 16 y se completaron 10 épocas de entrenamiento. La Tabla 3 se puede observar la información resumida.

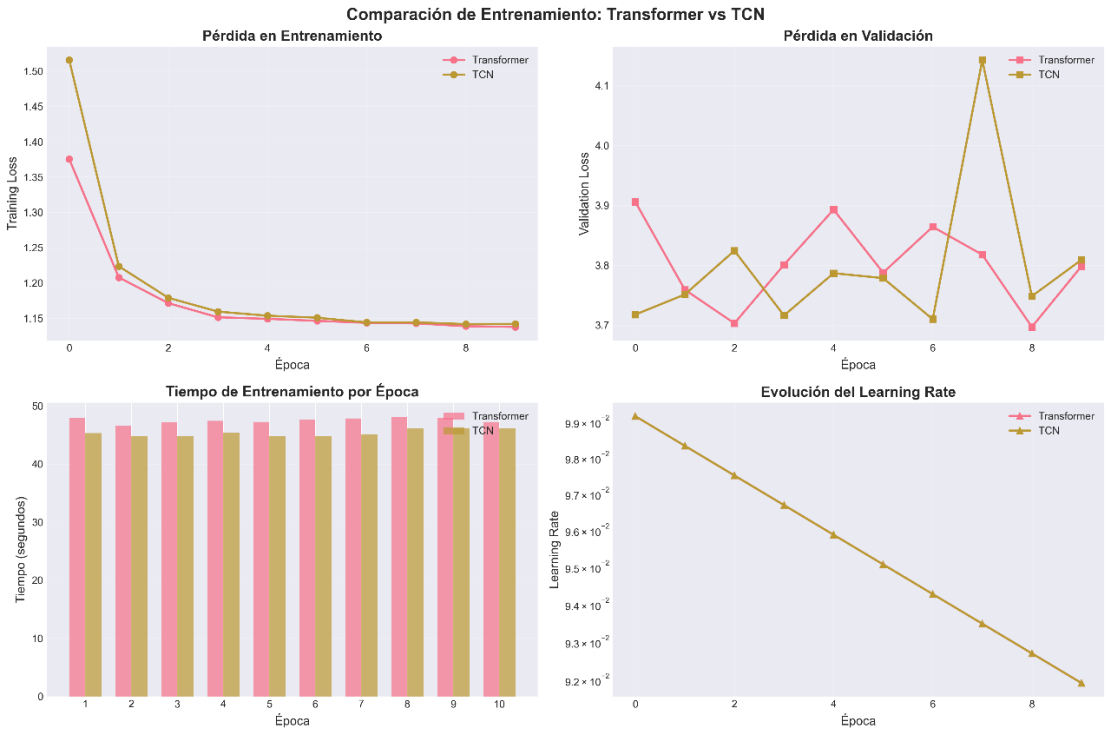

Ilustración 4: Curvas de pérdida (Corrida 2)

La pérdida de entrenamiento desciende con rapidez en las primeras épocas y se estabiliza alrededor de 1.14 en ambos modelos. En validación, las curvas oscilan en

el rango, el TCN presenta un pico marcado hacia la época 7, mientras que el Transformer se mantiene dentro de la misma banda con mínimos locales en torno a 3.70–3.75. En tiempo por época, TCN es ligeramente más eficiente que Transformer. La tasa de aprendizaje decrece de manera monótona conforme al scheduler coseno.

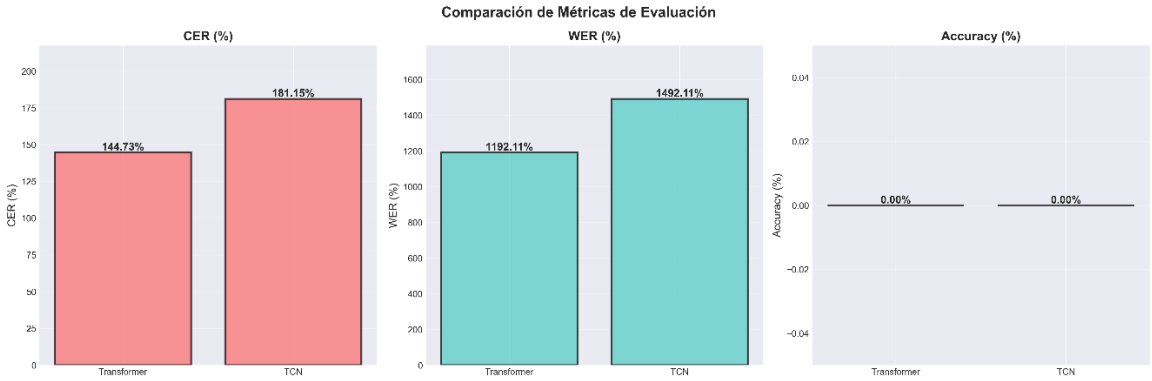

Ilustración 5: Comparación de métricas (Corrida 2)

En evaluación, el Transformer reporta CER = 144.73% y WER = 1192.11%, mientras que el TCN obtiene CER = 181.15% y WER = 1492.11%. El Accuracy de palabra completa es 0.00% para ambos modelos. Por lo tanto, el Transformer vuelve a superar al TCN en CER y WER, aunque ninguno logra aciertos exactos de secuencia completa.

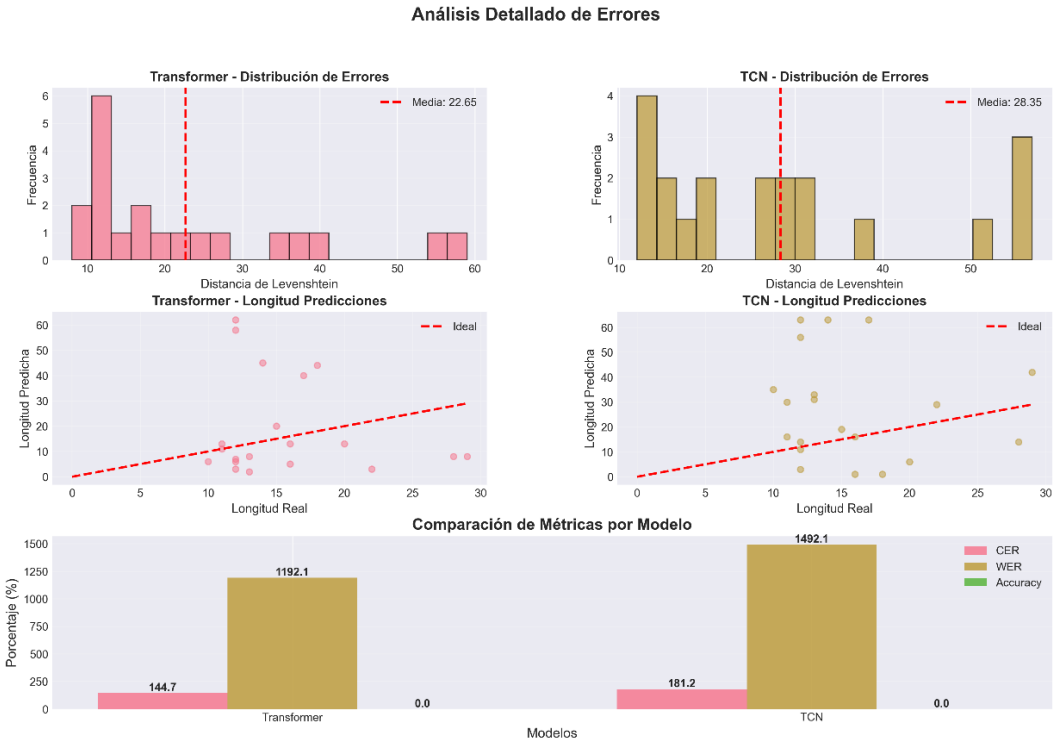

Ilustración 6: Distribución de distancias (Corrida 2)

La ilustración 6 muestra la distribución de distancias de Levenshtein y la dispersión de longitudes. El promedio de distancia es 22.65 para Transformer y 28.35 para TCN, indicando menos ediciones requeridas por el primero. En los diagramas longitud real vs. longitud predicha, ambos modelos se alejan de la diagonal ideal, con sobreestimaciones de longitud especialmente visibles en TCN (predicciones que alcanzan valores entre 40 y 60 caracteres para longitudes reales cercanas a 10–20). El panel inferior confirma la ventaja del Transformer en CER/WER y el Accuracy nulo en ambos.

Tabla 4: Preddiciones vs Realidad (Corrida 2)

	modelo	real	pred
1	Transformer	mio-footwear/refugee	gyweng/l la.r
2	Transformer	www.filmdb.it	to lj0g3
3	Transformer	www.lemon.co.jp	oai7-fjw0jlo3mlftywr
4	Transformer	https://www.pintradingdb.com	i.alltn3
5	Transformer	981396 3sl	=-39-ro
6	Transformer	6169 valley view parks	2ny
7	Transformer	2245 winston hill	3vr6uv.onv9.ljgt33uuug32/l1 rnn-reg ed-o
8	Transformer	la-mejor-dama-honor/equindiee	=-ygl/adr
9	Transformer	sasha quinn	5e2r32 rje2
10	Transformer	thien-dao-game-thu	2-srni-3atergrds'ew3tuc/e/moda3g7bonrluu2o-o
11	TCN	mio-footwear/refugee	0.b -a
12	TCN	www.filmdb.it	ga9i6og956g7ean8. gg/ [ao58a5g06?
13	TCN	www.lemon.co.jp	rg%t93fr/tbayrh/uri
14	TCN	https://www.pintradingdb.com	g-9up0rt0o3 (5
15	TCN	981396 3sl	ina198il9a2k6ter*$4 n66 /ogr !r(b4a
16	TCN	6169 valley view parks	5/-:gredprtr6rrrgu mramolrg-9
17	TCN	2245 winston hill	6og4teha6t39k%tw 3da 6erokegr0otr/ h58gq=tn+y0g4le97tku9m d4-m9
18	TCN	la-mejor-dama-honor/equindiee	m9r0006srrag.iw8/d
19	TCN	sasha quinn	ar49p049/l//u9a6
20	TCN	thien-dao-game-thu	g

La Tabla 4 presenta ejemplos representativos de la Corrida 2 extraídos de predictions.csv. Se aprecia que, aunque ciertos segmentos coinciden, persisten discrepancias globales entre las secuencias reales y las predichas, lo que explica los valores elevados de WER/CER y el Accuracy de 0%.

La Corrida 2 muestra estabilidad en la pérdida de entrenamiento y variación en la pérdida de validación, con un episodio de aumento en TCN. En métricas de evaluación y análisis de errores, el Transformer mantiene mejor desempeño (menor Levenshtein, CER y WER) que TCN, mientras que TCN conserva una ligera ventaja en tiempo por época. No se observan aciertos exactos de palabra completa (Accuracy = 0%), y se registran desajustes de longitud que contribuyen a los altos WER/CER.

Corrida 3

Tabla 3: configuración de corrida 2	
Parámetro	**Valor**
Optimizador	AdamW
Learning rate	$1 + 10^{-4}$
Warmup Ratio	0.05
Scheduler	Coseno
Gradient Clipping	1.0
Label smoothing	0.05
Batch size	16
Épocas ejecutadas	50

En la Corrida 3 se mantuvo la misma configuración de optimización y regularización empleada en las corridas anteriores. Se entrenaron Transformer y TCN con AdamW, learning rate 1×10^{-4}, scheduler coseno con 5% de warm-up, gradient clipping 1.0 y label smoothing 0.05. El batch size fue 16 y se completaron 50 épocas de entrenamiento. La Tabla 5 resume los hiperparámetros.

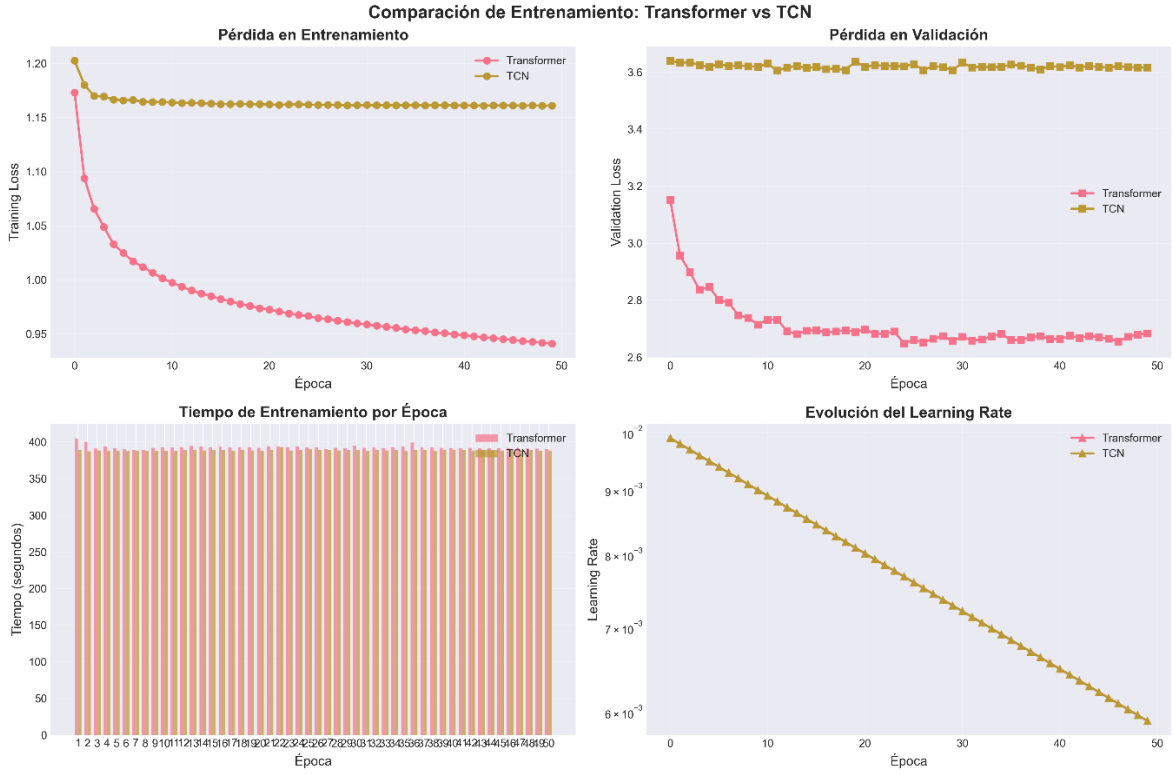

Ilustración 7: Curvas de pérdida (Corrida 3)

El Transformer muestra una disminución sostenida de la pérdida de entrenamiento y una mejora clara en validación, que desciende desde ≈3.1 en las primeras épocas

hasta estabilizarse alrededor de ≈2.70–2.75 a partir de la época 20. En contraste, el TCN permanece prácticamente plano: la pérdida de entrenamiento oscila levemente alrededor de aproximadamente 1.16–1.17, y la pérdida de validación se mantiene en aproximadamente 3.60–3.65 durante toda la corrida. En tiempo por época, TCN resulta ligeramente más rápido que Transformer. La tasa de aprendizaje decrece de forma monótona conforme al *scheduler* coseno.

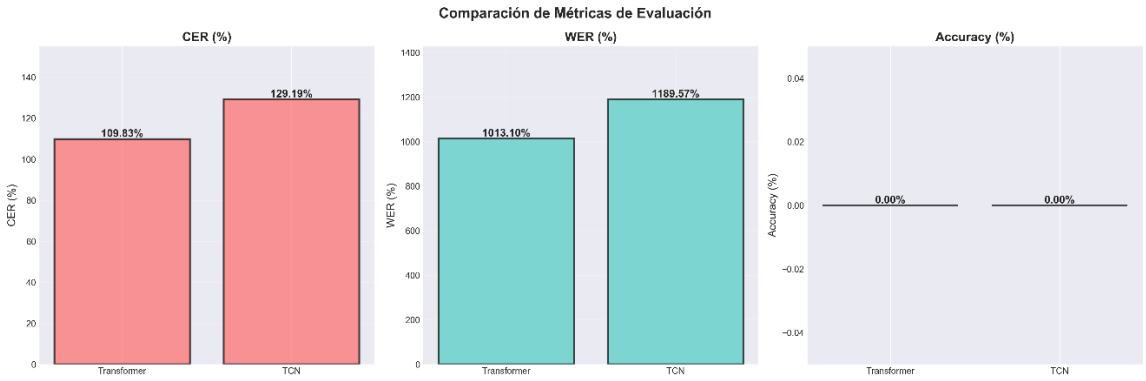

Ilustración 8: Comparación de métricas (Corrida 3)

En evaluación, el Transformer obtiene CER = 109.83% y WER = 1013.10%, mientras que el TCN registra CER = 129.19% y WER = 1189.57%. El Accuracy de palabra completa es 0.00% para ambos modelos. Por lo tanto, el Transformer supera al TCN en CER y WER, si bien ninguno logra aciertos exactos de secuencia completa.

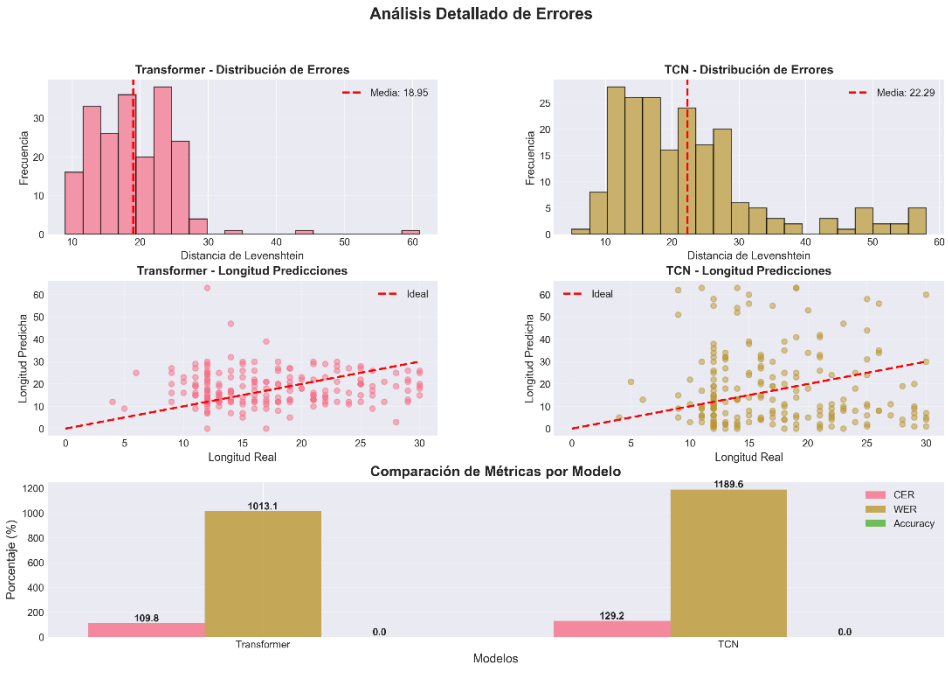

Ilustración 9: Distribución de distancias (Corrida 3)

La ilustración 9 presenta la distribución de distancias de Levenshtein y la dispersión de longitudes. El promedio de distancia es 18.95 para Transformer y 22.29 para TCN, lo que indica que el Transformer requiere menos ediciones para llegar a la secuencia correcta. En los diagramas longitud real vs. longitud predicha, ambos modelos se alejan de la diagonal ideal; el TCN exhibe sobreestimaciones de longitud más pronunciadas (predicciones que alcanzan 40–60 caracteres con longitudes reales en el rango 10–20). El panel inferior confirma la ventaja del Transformer en CER/WER y el Accuracy nulo en ambos casos.

Tabla6: Predicciones vs realidad (Corrida 3)

	modelo	real	pred
1	Transformer	thien-dao-game-thu	999013 n3668
2	Transformer	664468 crested tern	716ney keremey dree
3	Transformer	-1321	www.rtisttrinaicveresereba4036
4	Transformer	ronnie davidson	bm-@vesrhe/eallerst/7354k-iste
5	Transformer	548-627-7820	sendinczioaletbesl
6	Transformer	5693 jacob fork river road	+53-956-61(aeliellel-kgrta
7	Transformer	7837 spring break terrace	www.pwostlang.obrg/2
8	Transformer	jnhswf.com/linn-county-leader	menece)foldy
9	Transformer	302-910-3504	www.sautarnta.com
10	TCN	thien-dao-game-thu	gmc75ed6lb/kkt7d.79yc8
11	TCN	664468 crested tern	t01rp
12	TCN	-1321	mi8srr1lwp5vm6-3ow .-8 7aa2 e cnnh ew.e.ed -n41
13	TCN	ronnie davidson	cly1scm o-3wf@0pu5inoobdhlabo/2derstvue-n l.o8vt4y_uwe8ouoe
14	TCN	548-627-7820	omsu2337e2.w lnwawcd$ltrydeiiiw5i o9ri
15	TCN	5693 jacob fork river road	se7a9o6o
16	TCN	7837 spring break terrace	2
17	TCN	jnhswf.com/linn-county-leader	w(ant+u
18	TCN	302-910-3504	2xesy4di776-l bto2b9 2 /a
19	TCN	jess mullen	%.2%en$r8ossio.h(oh/srta9dtes'
20	TCN	sasha quinn	ci7kl4l 0sd-h0be5

La Tabla 6 muestra ejemplos representativos de la Corrida 3 extraídos de predictions.csv. Aunque ciertos segmentos coinciden con las etiquetas reales, persisten diferencias globales en la secuencia, coherentes con los valores elevados de WER/CER y el Accuracy de 0%.

Con 50 épocas de entrenamiento bajo la misma configuración, el Transformer evidencia mejor convergencia y mejor desempeño en validación que el TCN, además de menores errores promedio (Levenshtein, CER y WER). El TCN conserva una ventaja leve en tiempo por época, pero sin mejoras en validación. No se observan aciertos exactos de palabra completa (Accuracy = 0%), y persisten desajustes de longitud que explican los altos WER/CER.

Corrida 4

Tabla 3: configuración de corrida 2	
Parámetro	**Valor**
Optimizador	AdamW
Learning rate	$1 + 10^{-4}$
Warmup Ratio	0.05
Scheduler	Coseno
Gradient Clipping	1.0
Label smoothing	0.05
Batch size	16
Épocas ejecutadas	50

En la Corrida 4 se mantuvo la misma configuración de optimización y regularización de las corridas previas. Se entrenaron Transformer y TCN con AdamW, learning rate de 1×10^{-4}, scheduler coseno con 5% de warm-up, gradient clipping 1.0 y label smoothing 0.05. El batch size fue 16 y se completaron 50 épocas de entrenamiento. La Tabla 7 resume los hiperparámetros.

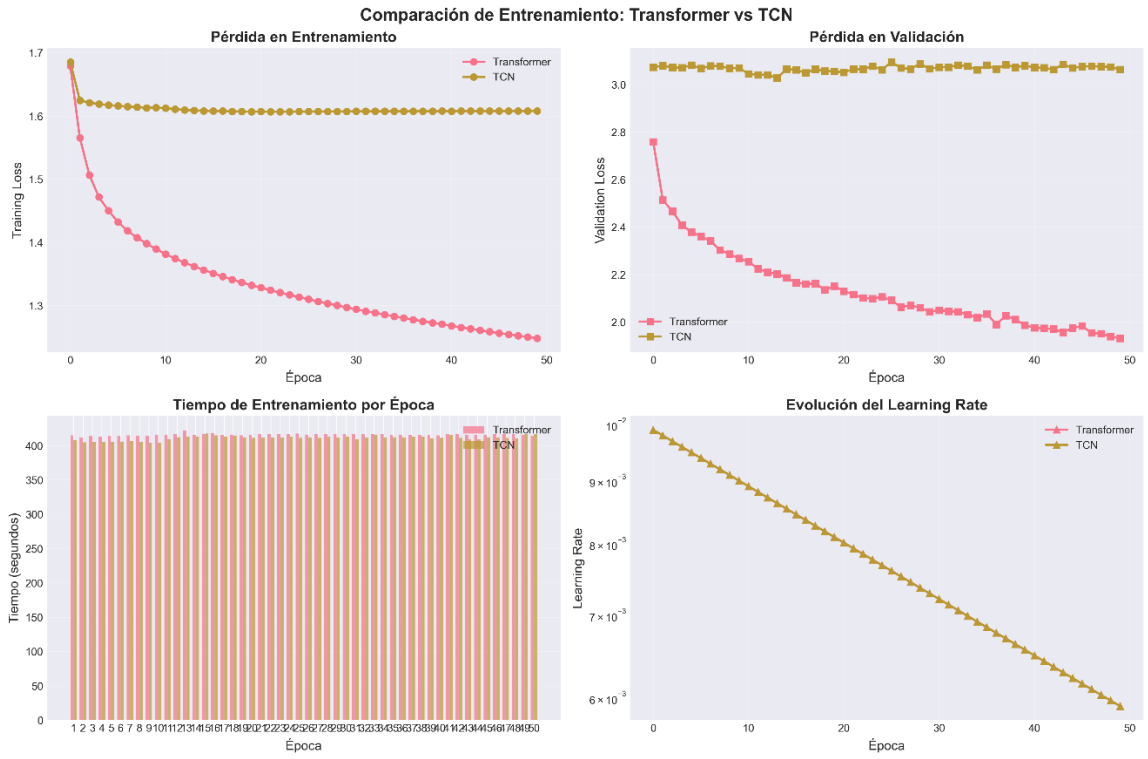

Ilustración 10: Curvas de pérdida (Corrida 4)

El Transformer muestra una disminución sostenida de la pérdida de entrenamiento y una mejora marcada en validación, que desciende de ≈2.75 en las primeras épocas

hasta ≈1.95 hacia el final. En contraste, el TCN permanece casi plano tanto en entrenamiento como en validación durante toda la corrida. En tiempo por época, ambos modelos rondan los ≈410–420 s, con TCN apenas más rápido de forma consistente. La tasa de aprendizaje decrece monótonamente conforme al scheduler coseno.

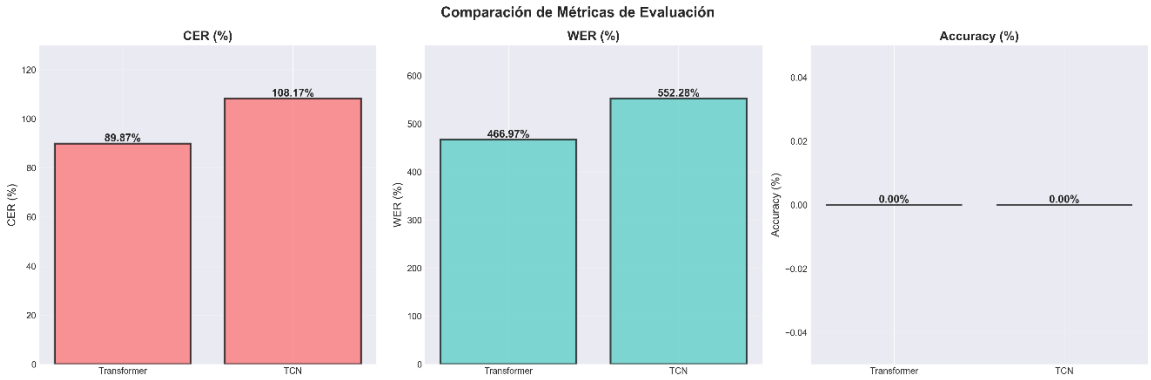

Ilustración 11: Comparación de métricas (Corrida 4)

En evaluación, el Transformer alcanza CER = 89.87% y WER = 466.97%, mientras que el TCN registra CER = 108.17% y WER = 552.28%. El Accuracy de palabra completa es 0.00% para ambos modelos. Por lo tanto, el Transformer vuelve a superar al TCN en CER y WER, aunque ninguno logra aciertos exactos de secuencia completa.

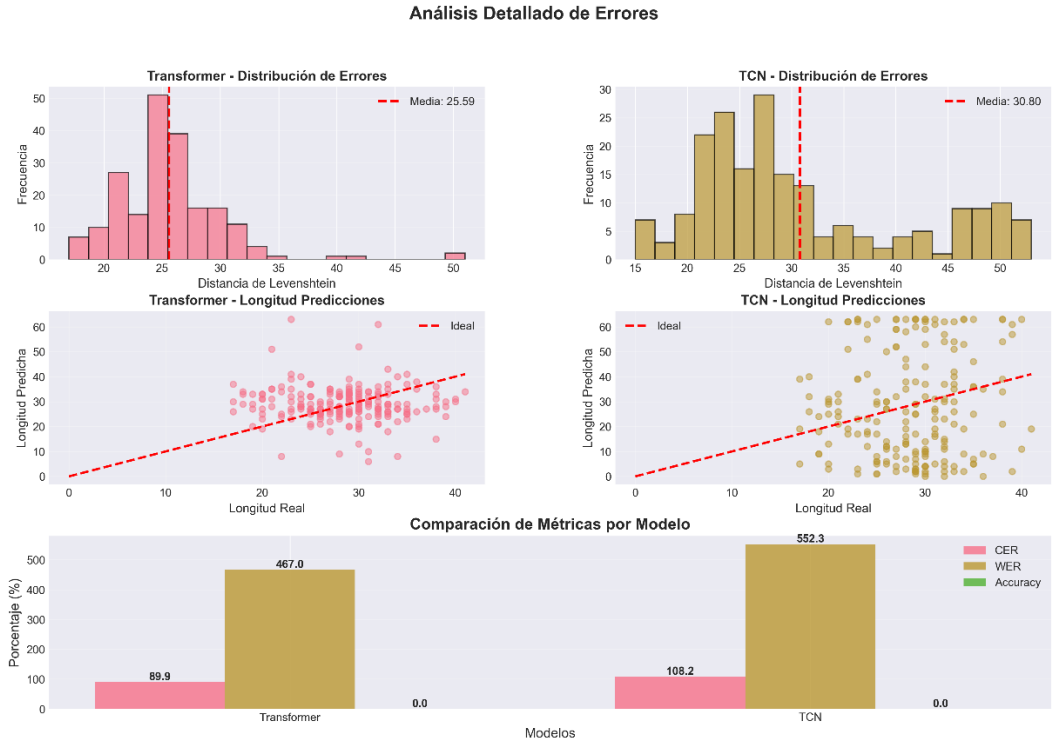

Ilustración 12: Distribución de distancias (Corrida 4)

La ilustración 12 muestra la distribución de distancias de Levenshtein y la dispersión de longitudes. El promedio de distancia es 25.59 para Transformer y 30.80 para TCN, indicando menores ediciones requeridas por el primero. En los diagramas longitud real vs. longitud predicha, el Transformer concentra los puntos alrededor de la diagonal ideal con ligera sobreestimación en el rango de 20–35 caracteres; el TCN presenta una dispersión mayor, con sub- y sobreestimaciones pronunciadas para longitudes reales entre 20–35. El panel inferior confirma la ventaja del Transformer en CER/WER y el Accuracy nulo en ambos.

Tabla 8: Predicciones vs realidad (Corrida 4)

	modelo	real	pred
1	Transformer	peek out the window	oth?s arans my t ou newnoreal
2	Transformer	keep receipts for all your expenses	meit you onerve the wacce
3	Transformer	the aspirations of a nation	the pripurating to modanseceel
4	Transformer	seasoned golfers love the game	so atave imeds knot ake ten
5	Transformer	we dine out on the weekends	the prozod ons knk
6	Transformer	he played a pimp in that movie	a ling tha2 dringh fblestiv:es
7	Transformer	he watched in astonishment	wet my wor@lh t9re ayh fis
8	Transformer	a little encouragement is needed	the be ,eand the is xhswe eastery
9	Transformer	a picture is worth many words	wthe cot seve ther crow foriect
10	TCN	peek out the window	tlq- tnde
11	TCN	keep receipts for all your expenses	ostr gtvgo iac+i ueelaisepgooioc op
12	TCN	the aspirations of a nation	iu
13	TCN	seasoned golfers love the game	w
14	TCN	we dine out on the weekends	wrdeedm ?u i r$ r n krrplss iuda
15	TCN	he played a pimp in that movie	i yoo aon yitts go h he snn oo6t,cape sh cdu mai vr
16	TCN	he watched in astonishment	re+ ot
17	TCN	a little encouragement is needed	i vd
18	TCN	a picture is worth many words	el
19	TCN	flashing red light means stop	ulna ctul
20	TCN	only an idiot would lie in court	s naetns b i siv d=iet a etisnaeaaan nei itrmtbln ke q

La Tabla 8 presenta ejemplos representativos de la Corrida 4 extraídos de predictions.csv. Se aprecia que, aunque existen coincidencias parciales, las secuencias predichas difieren de forma global respecto a las reales, lo cual es consistente con los valores elevados de WER/CER y el Accuracy de 0%.

Con 50 épocas bajo una configuración común, el Transformer muestra mejor convergencia y mejor desempeño en validación que el TCN, además de menores errores (Levenshtein, CER y WER). El TCN conserva una ventaja leve en tiempo por época, pero no traduce esa eficiencia en mejoras de generalización. No se registran aciertos de palabra completa (Accuracy = 0%), y persisten desajustes de longitud que explican los WER/CER elevados.

III. Discusión

A lo largo de las cuatro corridas realizadas, se observa un patrón claro: el modelo Transformer supera al TCN en métricas de error como CER, WER y distancia promedio de Levenshtein. Aunque el TCN muestra una ligera ventaja en tiempo por época, esta eficiencia computacional no se traduce en mejor capacidad de generalización. En todos los casos, el Accuracy de palabra completa se mantiene en 0% para ambos modelos, lo cual concuerda con las discrepancias entre las secuencias reales y las predichas.

Este resultado se explica en parte por la naturaliza estricta de la métrica de Accuracy utilizada, que exige coincidencia exacta a nivel de palabra. Basta una sola edición –inserción, eliminación o sustitución de carácter- para que una predicción parcialmente correcta se considere incorrecta. Por ello, el 0% no implica que los modelos no hayan aprendido nada, sino que los aciertos parciales no alcanzan el umbral exigido por la métrica.

Los valores elevados de CER y WER, incluso superiores al 100%, se deben a desajustes de longitud entre las predicciones y las referencias. Este efecto es más notorio en el TCN, cuyas salidas tienden a ser más largas que las palabras reales, lo que incrementa el número de ediciones necesarias. Por otro lado, el Transformer muestra una reducción sostenida en la pérdida de entrenamiento y validación, lo que sugiere una mejor capacidad de generalización bajo la configuración común usada.

Finalmente, aunque los resultados están condicionados por los insumos disponibles y por el hecho de que ambos modelos fueron evaluados bajo la misma configuración, el desempeño observado favorece el Transformer en términos de convergencia y precisión. El TCN, por su parte, destaca únicamente por su menor tiempo por época, sin que ello compense sus limitaciones en error y generalización.

IV. Conclusiones

En conjunto, las cuatro corridas muestran de forma consistente que el Transformer ofrece un mejor desempeño que el TCN en las métricas de error (CER y WER) y en la distancia promedio de Levenshtein. Aunque el TCN resulta ligeramente más rápido por época, esa eficiencia computacional no se traduce en mejor generalización. Las curvas de validación del Transformer descienden y se estabilizan más abajo, mientras que las de TCN se mantienen altas o casi planas.

Los errores observados se explican, en gran medida, por desajustes de longitud entre predicciones y etiquetas, lo que eleva el número de ediciones y pueden llevar a WER/CER >

100%. En ese contexto, el Accuracy de palabra completa permanece en 0% para ambos modelos, dado que exige coincidencia exacta de toda la secuencia y penaliza cualquier discrepancia parcial.

Bajo la configuración evaluada, el Transformer es la arquitectura que mejor aprende y generaliza para el reconocimiento de secuencias en este conjunto, mientras que el TCN destaca únicamente por el mejor tiempo por época. Estos hallazgos orientan el foro del trabajo posterior hacia el modelo que ya muestra ventajas objetivas en error y convergencia.

V. Recomendaciones para próxima entrega

1. **Ajustes del Transformer:** Sería útil explorar distintas combinaciones de parámetros como la tasa de aprendizaje, el calentamiento inicial, la regularización y el suavizado de etiquetas. También se recomienda implementar mecanismos de parada temprana y guardar los mejores modelos según las métricas de error (CER/WER). Si el modelo sigue mejorando en validación, conviene ampliar el número de épocas para aprovechar ese progreso.
2. **Decodificación y control de longitud:** Para mejorar la calidad de las predicciones, se sugiere usar estrategias de decodificación más robustas, como beam search, acompañadas de ajustes que penalicen o calibren la longitud de las salidas. Además, aplicar reglas simples de posprocesamiento puede ayudar a evitar que las palabras generadas sean demasiado cortas o largas.
3. **Explorar modelos con estructura espacial:** Sería interesante probar arquitecturas que aprovechen mejor la información espacial, como redes de grafos (GNN) o variantes espaciales-temporales (ST-GCN). Estos modelos, ya sea en forma pura o combinados con Transformers, podrían capturar mejor las relaciones entre los puntos clave del cuerpo.
4. **Mejorar el preprocesamiento y la variación de datos:** Para fortalecer la capacidad de generalización, se recomienda aplicar técnicas como la normalización por secuencia o por sujeto, distorsiones temporales, ruido en las coordenadas y eliminación aleatoria de cuadros o articulaciones. También puede ser útil agrupar las secuencias por longitud para reducir el relleno innecesario.
5. **Evaluación más detallada:** Para entender mejor los errores, conviene reportar las métricas CER y WER segmentadas por rangos de longitud y por tipo de carácter. Además, mantener constantes los conjuntos de evaluación y las semillas de aleatoriedad permite hacer comparaciones más justas entre modelos y configuraciones

Referencias

Awan, A. A. (21 de julio de 2022). *A Comprehensive Introduction to Graph Neural Networks*. Obtenido de Datacamp: https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial

Bergmann, D., & Stryker, C. (s.f.). *What is an autoencoder?* Obtenido de IBM: https://www.ibm.com/think/topics/autoencoder

Cihan, N., Koller, O., Hadfield, S., & Bowden, R. (2020). *Sign Language Transformers: Joint End-To-End Sign Language Recognition and Translation.* Guildford, UK: University of Surrey.

GeeksForGeeks. (23 de julio de 2025). *Zero Padding in Deep Learning and Signal Processing*. Obtenido de Geeks For Geeks.org: https://www.geeksforgeeks.org/deep-learning/zero-padding-in-deep-learning-and-signal-processing/

GeeksForGeeks. (23 de julio de 2025). *RNN vs LSTM vs GRU vs Transformers*. Obtenido de Geeks For Geeks.org: https://www.geeksforgeeks.org/deep-learning/rnn-vs-lstm-vs-gru-vs-transformers/

Kumar, R. (11 de febrero de 2025). *Feature Extraction in Machine Learning: A Complete Guide*. Obtenido de DataCamp: https://www.datacamp.com/tutorial/feature-extraction-machine-learning

Paparaju, T. (27 de diciembre de 2018). *Sequence Modelling*. Obtenido de Medium: https://medium.com/machine-learning-basics/sequence-modelling-b2cdf244c233

Stryker, C., & Bergmann, D. (s.f.). *What is a transformer model?* Obtenido de IBM: https://www.ibm.com/think/topics/transformer-model

Winland, V. (s.f.). *What is feature extraction?* Obtenido de IBM: https://www.ibm.com/think/topics/feature-extraction

Yadav, A. (9 de octubre de 2024). *Temporal Convolutional Network - An Overview*. Obtenido de Medium: https://medium.com/@amit25173/temporal-convolutional-network-an-overview-4d2b6f03d6f8