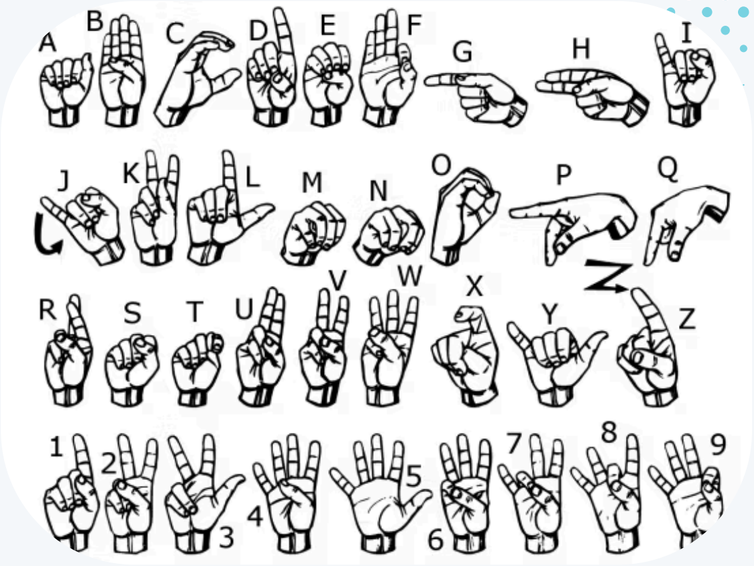
RECONOCIMIENTO Y TRADUCCIÓN DE FINGERSPELING (ASL)

Ariela Mishaan, Alina Carias, Diego Soto, Ignacio Mendez, Marcos Díaz

¿QUÉ SE QUIERE LOGRAR?

- Traducir fingerspelling en ASL (deletreo con las manos) a texto.
- Input: Secuencias temporales de keypoints (x,y,z) extraídos con MediaPipe.
- Output: oraciones en lenguaje natural que describen la secuencia.
- Tipo de problema:
  - Series temporales multivariadas de alta dimensión.
  - Reconocimiento de gestos y secuencias, similar a escritura a mano o reconocimiento de voz.

El objetivo principal es que a partir de la trayectoria de las manos en 3D, el modelo aprenda a “leer” lo que se est+a deñetrando en ASL y se convierta a texto.



¿POR QUÉ IMPORTA?

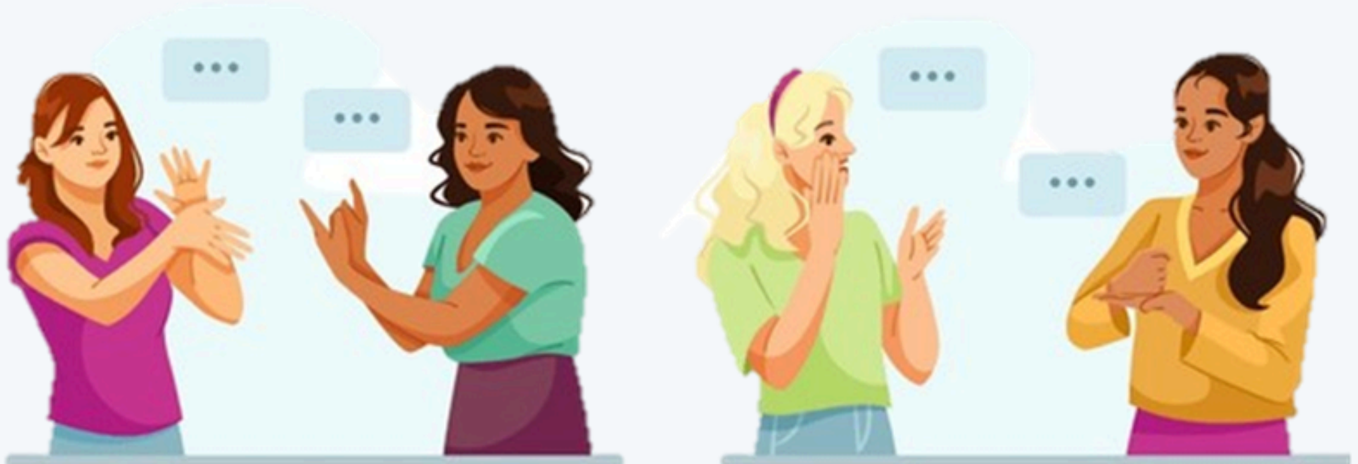
- Apoya a la comunicación accesible entre personas sordas y oyentes

- Aplicaciones en:
  - Herramientas de educación inclusiva
  - Interfaces hombre-máquina basadas en gestos
  - Asistentes en tiempo real o sistemas de interpretación.

- Retos intersantes de Data Science:
  - Alta dimensionalidad
  - Dependencia temporal + estructura espacial.

¿QUÉ DATOS USA EL MODELO?

- Cada frame = un vector grande con todos los keypoints concatenados (x,y,z).
- Una secuencia = varios frames para una palabra/frase en ASL.
- Etiqueta objetivo:
  - Secuencia de caracteres/palabras que describe el gesto.
- Puntos clave:
  - Alta dimensionalidad → riesgo de sobreajuste.
  - Estructura tipo grafo: conexiones entre articulaciones.



ESTRUCTURA GENERAL DEL PIPELINE

LECTURA Y AGRUPACIÓN

Agrupación de frames por sequence_id

PREPROCESAMIENTO

- Normalización de coordenadas (escala/traslación).)
- Padding / truncation de secuencias a lonjitud fija

TOKENIZACIÓN DEL TEXTO

Secuencias objetivo con longitud máxima

MODELADO SECUENCIAL

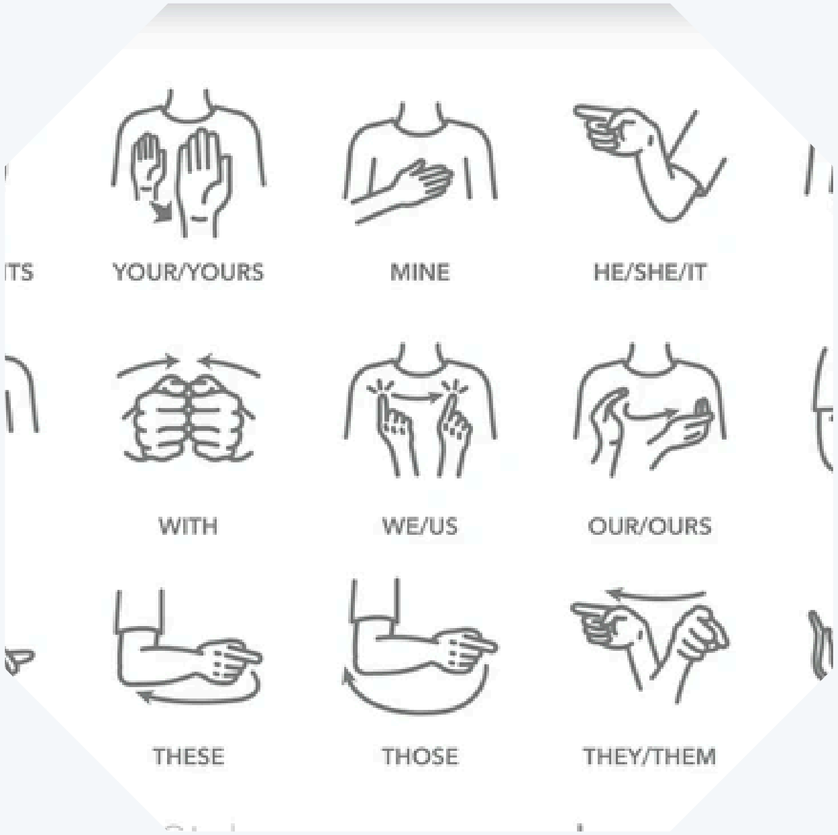
Transformadores (Transformers A y B)

DECODIFICACIÓN

Generación de texto

Todo el pipeline está pensado para que las secuencias, aunque tengan longitudes distintas, se pueden meter en batches y entrenar eficientemente.

PREPROCESAMIENTO: DECISIONES CLAVE



- Normalización de coordenadas:
  - Reduce varianza por posición y escala de la mano.
- Padding y truncación a longitud fija:
  - Necesario para entrenar modelos de deep learning de batches.
  - Se rellena con ceros las secuencias cortas.
- Tokenización del texto objetivo:
  - Conversión de IDs de caracteres/palabras.
  - Se aplica máscara para que el modelo ignore el padding en la pérdida.

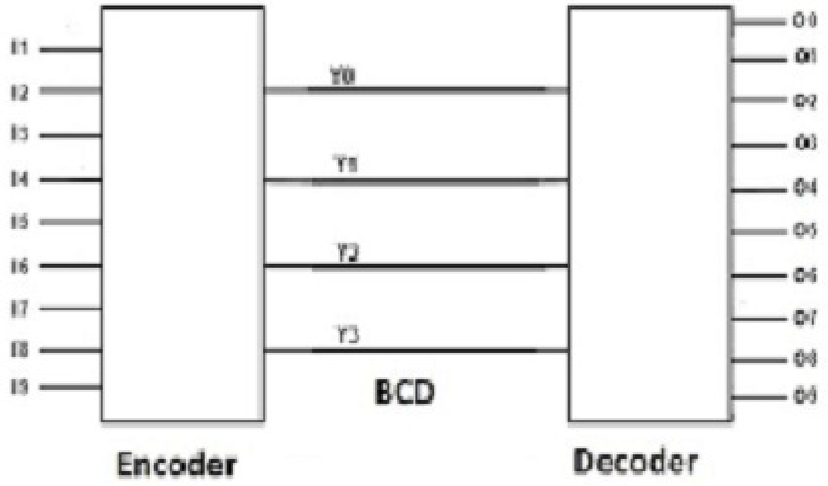
MODELOS CANDIDATOS

- RNN / LSTM / GRU
  - Buen manejo de secuencias pero limitaciones para vanishing gradient.
- Redes Convolucionales Temporanes (TCN)
  - Convoluciones 1D dilatadas, eficientes para secuencias largas.
- Transformers
  - Atención sobre toda la secuencia, paralelismo y manejo de dependencias largas.
- Graph Neural Networks (GNN)
  - Aprovechan la estructura de grafo de las articulaciones.
- Modelos Híbridos
  - Combinar GNN + Transformer, etc.

Aunque hay muchas arquitecturas, nos enfocamos en Transformers porque son el estándar actual para secuencias complejas y permiten explotar la atención en toda la serie temporal.

ARQUITECTURA BASE DEL MODELO TRANSFORMER

- Enfoque seg2seg (secuencia de keypoints → secuencia de texto).
- Landmark Embedding:
  - Proyección del vector de keypoints de cada frame a un espacio latente de 128 dimensiones.
- Codificación posicional:
  - Para preservar el orden temporal de los frames.
- Codificación (Encoder).
  - Autoatención multi-cabeza.
  - Feed-forward + LayerNorm + residuals.
- Decodificador (Decoder):
  - Máscara causal (no mira el futuro)
  - Atención cruzada hacia la representación del encodet.
- Entrenamiento:
  - Optimizador Adam.
  - Pérdida: entropía cruzada sobre la secuencia.



CONFIGURACIONES A Y B

TRANSFOMER A

- Configuración más simple.
- Menor regularización efectiva.

TRANSFOMER B

- Ajuste de tasa de aprendizaje (más estable).
- .Mayor dropout en atención y capas feed-forward.
- Tamaño de batch ligeramente modificado.
- Early stopping para reducir sobreajuste.
- Uso de beam search en la etapa de decodificación.

La arquitectura es prácticamente la misma; se cambian hiperparámetros para estudiar el efecto en la estabilidad y el comportamiento de inferencia.

¿CÓMO MEDIMOS EL DESEMPEÑO?

- Character Error Rate (CER):
  - Distancia de edición a nivel de carácter.
  - Sensible a errores ortográficos finos.

- Word Error Rate (WER):
  - Distancia de edición a nivel de palabra.
  - Mide coherencia semántica global.

- Exactitud de secuencia:
  - Porcentaje de secuencias donde la predicción coincide exactamente con la referencia.
  - Métrica muy estricta.

Character Error Rate (CER):

$$CER = \frac{\text{Number of incorrect characters}}{\text{Total number of characters in the reference text}} \times 100\%$$

Word Error Rate (WER):

$$WER = \frac{\text{Number of incorrect words}}{\text{Total number of words in the reference text}} \times 100\%$$

Son métricas típicas de reconocimeinto de voz y traducción automática, adaptadas aquí al problema de fingerspelling.

RESULTADOS CUANTITATIVOS

Modelo	CER (%)	WER (%)	Exactitud (%)	Diagnóstico
Tranformer A	84.97	414.51	0.00	Sobreajuste parcial, más variación
Transformer B	79.77	441.42	0.00	Colapso de salida (modo único)

- Ambos modelos tienen:
  - CER alto y exactitud de secuencia = 0%

- Transformer B:
  - Reduce ligeramente CER
  - Empeora WER y colapsa en la práctica.

Numéricamente, ninguno es “bueno” para producción, pero sirven para entender qué está fallando en el entrenamiento.

TRANSFORMER A

```
Ejemplos de predicciones:

Ejemplo 1:
Real: 'peek out the window'
Pred: 'peesing will a care camper'

Ejemplo 2:
Real: 'keep receipts for all your expenses'
Pred: 'stay hat froy gould'

Ejemplo 3:
Real: 'the aspirations of a nation'
Pred: 'i watched to a stitall succed'

Ejemplo 4:
Real: 'seasoned golfers love the game'
Pred: 'rine at har a doouster bund bater'

Ejemplo 5:
Real: 'we dine out on the weekends'
Pred: 'i want to in hourt hex the wor'
```

- Genera oraciones distintas para entradas diferentes.
- Mezcla de:
  - Bigramas y fragmentos con apariencia de inglés real.
  - Secuencias sin sentido o fonéticamente “rotas”.
- Señales de aprendizaje:
  - Responde a cambios en la entrada.
  - Captura parcialmente patrones sub-léxicos (sílabas, n-gramas).

El modelo A no “traduce bien”, pero cambia lo que dice cuando se cambia la secuencia de entrada. Eso indica que está aprendiendo algo de la dinámica temporal.

TRANSFORMER B

```
Ejemplos de predicciones:

Ejemplo 1:
Real: 'peek out the window'
Pred: 'i want to hold your hand'

Ejemplo 2:
Real: 'keep receipts for all your expenses'
Pred: 'i want to hold your hand'

Ejemplo 3:
Real: 'the aspirations of a nation'
Pred: 'i want to hold your hand'

Ejemplo 4:
Real: 'seasoned golfers love the game'
Pred: 'i want to hold your hand'

Ejemplo 5:
Real: 'we dine out on the weekends'
Pred: 'i want to hold your hand'
```

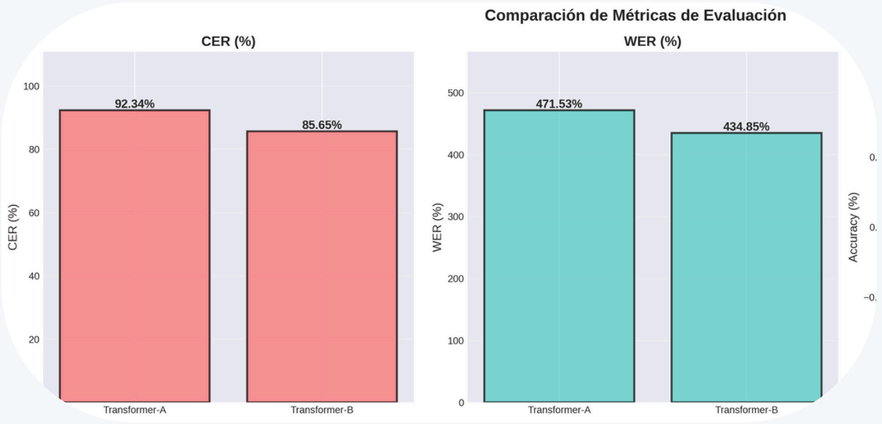
- Se observa un colapso de modo clásico.
- Para entradas diferentes, el decodificador:
  - Produce la misma frase corta y frecuente (“I want to hold your hand”).
- El modelo “aprende” una solución trivial:
  - Emitir siempre la frase que minimiza la pérdida promedio.
- Señar clara de:
  - Sobreajuste severo.
  - Pérdida de sensibilidad a la entrada.

El modelo B deja de “escuchar” los keypoints y se queda repitiendo su frase favorita.

¿QUÉ NOS DICEN LOS RESULTADOS?

- La arquitectura Transformer sí es adecuada para datos secuenciales complejos.
- El problema actual no es la forma del modelo, sino:
  - Estrategia de entrenamiento.
  - Tamaño y calidad del dataset.
  - Falta de regularización bien controlada.
- Transformer A:
  - Muestra una entropía de salida más alta (más diversidad).
  - Sensible a la entrada, aunque con errores altos.
- Transformer B:
  - Ilustra el extremo degenerado que se debe de evitar.

La arquitectura va en buen camino, pero la receta de entrenamiento necesita ajustes fuertes.

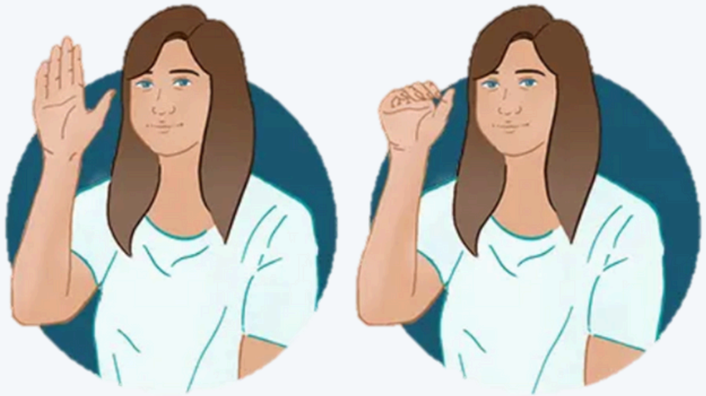


CONCLUSIONES

- La arquitectura Transformer es apropiada para el reconocimiento y traducción de fingerspelling en ASL, gracias a su capacidad para modelar dependencias temporales largas y patrones complejos.
- El bottleneck actual está en la ingeniería de entrenamiento: tamaño de dataset, regularización, tuning de hiperparámetros y estrategias de decodificación.
- El Transformer A es el “menos malo”: mantiene errores altos pero responde a la entrada y genera salidas diversas.
- El Transformer B muestra un colapso de modo, repitiendo la misma frase; esto evidencia sobreajuste severo y la importancia de monitorear las curvas de entrenamiento y validación.

RECOMENDACIONES

- Parada temprana (early stopping):
  - Basada en métrica de validación para cortar antes del sobreajuste.
- Checkpointing:
  - Guardar los pesos del mejor modelo durante el entrenamiento.
- Tasa de aprendizaje:
  - Explorar estrategias como one-cycle o reducción en meseta.
- Aumento de datos en keypoints:
  - Pertubaciones temporales controladas.
  - Escalado y rotación ligera por frame.
- Representación espacial:
  - Insertar una capa GNN por frame que codifique explícitamente la topolgía anatómica.
- Dataset:
  - Ampliar y diversificar el conjunto de entrenamiento (más sujetos, más variabilidad).



Bye

BIBLIOGRAFÍA

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