Data Science

Proyecto 2 – Investigación Bibliográfica

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

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

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**

   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

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

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 son flexibles con el largo de las secuencias (no es necesario que todas tengan la

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

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

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