UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3084 - Data Science
Sección 20
Ing. Luis Furlan

Proyecto Final

Fase 2

https://github.com/JaniMariQuesiRami/DS-PR2

Javier Chavez 21016
Andres Quezada 21085
Mario Cristalitros 21631
Javier Ramirez 21600

Guatemala, 28 de octubre de 2024

Selección de Algoritmos

#1

- Transformer

- Descripción

    - Los modelos de transformadores son arquitecturas de aprendizaje profundo diseñadas para manejar datos secuenciales, como texto o secuencias de gestos, mediante el uso de mecanismos de atención que destacan relaciones contextuales en la secuencia. Son especialmente eficaces en tareas que requieren captar dependencias a largo plazo en los datos.

- Ajuste a nuestros datos y objetivo

    - Dado que los gestos en lenguaje de señas representan secuencias temporales complejas, el modelo transformer puede capturar patrones y relaciones temporales entre frames de video o landmarks, lo cual se ajusta bien al objetivo de reconocimiento de señas, donde el contexto y la secuencia son cruciales para identificar correctamente cada gesto.

#2

- Seq2Seq

- Descripción

    - El modelo Seq2Seq (Sequence-to-Sequence) es una arquitectura de redes neuronales diseñada para transformar una secuencia de entrada en una secuencia de salida, comúnmente utilizada en tareas de procesamiento de lenguaje natural (PLN) como traducción automática, resumen de texto, generación de respuestas, entre otros.

- Ajuste a nuestros datos y objetivo

    - Para el reconocimiento de señas, el modelo Seq2Seq es adecuado ya que permite interpretar secuencias de movimientos o posiciones de manos (entrada) y generar la secuencia correspondiente en lenguaje de señas (salida), capturando dependencias temporales en los datos y ajustándose al contexto específico de cada gesto en la secuencia.

#3

- CTC-based RNN (Recurrent Neural Networks with Connectionist Temporal Classification)

- Descripción

    - CTC (Connectionist Temporal Classification) es una técnica comúnmente utilizada en tareas de reconocimiento de secuencias donde la alineación entre la secuencia de entrada y la salida no está predeterminada. Usualmente, los modelos CTC se combinan con redes neuronales recurrentes (RNNs), como LSTMs o GRUs, que son capaces de procesar secuencias de datos temporales. CTC ayuda a manejar la variabilidad en la longitud de las secuencias y permite que el modelo prediga secuencias con longitudes variables sin la necesidad de una alineación precisa entre la entrada y la salida.

- Ajuste a nuestros datos y objetivos

    - El modelo basado en CTC es ideal para nuestro problema de reconocimiento de señas, donde las secuencias de movimientos de manos pueden variar en longitud, y no siempre existe una alineación clara entre la entrada (gestos o landmarks) y la salida (frase o palabra correspondiente). Al utilizar CTC con una red neuronal recurrente, podemos procesar los datos de gestos como una secuencia temporal y predecir la secuencia correspondiente en lenguaje de señas sin requerir una alineación previa exacta. Además, CTC es excelente para tareas donde las entradas y salidas tienen longitudes diferentes, como en nuestro caso.

Preprocesamiento e Ingeniería de Características

Transformer

- Preprocesamiento:

    - Reorganización de Datos: Primero reorganizamos los datos para que cada archivo parquet contenga los datos de puntos de referencia (landmarks) junto con la frase que representa. Esto nos permite evitar cambiar entre el archivo train.csv y sus archivos parquet, facilitando el acceso directo a los datos necesarios para el entrenamiento.

    - Almacenamiento en Formato TFRecord: Guardamos los datos en el formato TFRecord, que es un formato simple para almacenar secuencias de registros binarios. Este formato nos permite almacenar y cargar los datos de manera más eficiente y rápida en comparación con archivos CSV o JSON, lo que mejora el rendimiento durante el entrenamiento del modelo.

    - Creación de Etiquetas x, y, z: Generamos etiquetas para las coordenadas x, y y z de cada punto de referencia de la mano, estructurando así las posiciones espaciales de los landmarks.

    - Actualización de Archivo JSON de Caracteres: El archivo JSON de caracteres contiene un mapeo entre cada carácter y su valor. Añadimos tres nuevos caracteres: "<" y ">", para marcar el inicio y el final de cada frase, y "P" para el padding.

    - Generación de Nuevos Archivos de Dataset: Con los landmarks y las frases extraídas, creamos nuevos archivos del conjunto de datos y los guardamos en formato TFRecord. Aunque este proceso toma alrededor de 10 minutos, nos permite cargar los datos más rápido en futuros experimentos, optimizando el tiempo de procesamiento en cada sesión de entrenamiento.

- *Feature engineering:*

    - Detección de Mano Dominante: Implementamos un paso de preprocesamiento para identificar la mano dominante en cada secuencia. Para esto, usamos el número de valores NaN en cada mano, asumiendo que la mano dominante tiene menos valores NaN al estar en movimiento constante. Esta técnica ayuda a reducir el ruido y a centrarnos en la mano relevante en cada frame.
    - Normalización y Estandarización: Para cada mano, calculamos la media y desviación estándar, y estandarizamos las coordenadas x, y, z para mejorar la

consistencia en los datos de entrada. Esto ayuda a reducir la variabilidad en las posiciones de las manos, lo que es esencial para mejorar la precisión del modelo.

- Ajuste de Tamaño y Relleno (Padding): Definimos un largo fijo para las secuencias de frames (128 frames) y ajustamos las secuencias cortas mediante relleno (padding) o redimensionamiento. Esto asegura que todas las secuencias tengan una longitud consistente, lo que es importante para procesar los datos en lotes de tamaño uniforme en el modelo.

Seq2Seq

- Preprocesamiento de Señas

- **Selección y Organización de Características**: Se definen las características de la pose de cada mano (izquierda y derecha) y algunas posiciones corporales relevantes para capturar información específica del lenguaje de señas. Las características (X, Y, Z) para las posiciones de las manos y los puntos de articulación se organizan en índices para una fácil identificación y manipulación.

- **Extracción de Secuencias**: Se cargan los datos de la posición de las manos y poses de diferentes archivos en función de un identificador único. Las secuencias se extraen por cada "frase" o "signo", donde cada uno representa una secuencia de fotogramas que capturan las posiciones clave en cada frame.

- **Gestión de NaNs y Filtrado**: Se evalúa la calidad de los datos para cada secuencia en función de la cantidad de valores NaN. Aquellas secuencias con muchos NaN se eliminan para evitar datos inconsistentes o incompletos en el entrenamiento.

- **Agrupación de Datos**: Cada secuencia válida se guarda junto con la frase correspondiente. Esto incluye concatenar y reestructurar las coordenadas (X, Y, Z) en vectores de datos separados para la mano izquierda y derecha, y la pose en cada frame.

- **Normalización y Padding**: Las secuencias se normalizan y se ajustan a una longitud fija de fotogramas (128) mediante truncamiento o padding, asegurando así que todos los datos de entrada tengan una estructura uniforme, independientemente de su longitud original.

- **Ajuste de Coordenadas para la Mano Izquierda**: Se invierte el eje X de las posiciones de la mano izquierda para alinear todas las posiciones en un mismo marco de referencia.

- **Selección de Mano Dominante**: Se elige la secuencia de la mano que tiene menos valores NaN como la "mano dominante" para representar el signo, reduciendo el impacto de las articulaciones faltantes.

- **Limpieza Final y Sustitución de NaNs**: Finalmente, los valores NaN en la matriz de datos final se reemplazan por ceros para evitar errores de procesamiento y garantizar que todos los datos estén listos para alimentar al modelo.

- ***Feature Engineering en Sign Preprocessing***

- **Selección de Características Relevantes**: Se seleccionan coordenadas específicas de los puntos de las manos (izquierda y derecha) y ciertos puntos de la pose corporal, como las articulaciones clave en el lenguaje de señas (por ejemplo, hombros, codos, muñecas y dedos). Esta selección reduce la complejidad al enfocarse solo en puntos esenciales, mejorando la relevancia y reduciendo el ruido en los datos.

- **Agrupación de Coordenadas**: Las coordenadas de cada articulación se organizan por dimensión (X, Y, Z) y por extremidad (mano izquierda, mano derecha, pose), lo que permite un procesamiento más eficiente y facilita el análisis estructural. Este enfoque también permite aplicar distintas operaciones (por ejemplo, normalización o padding) sobre grupos específicos de características.

- **Extracción de Mano Dominante**: Para cada secuencia, se selecciona la mano con menos valores perdidos (NaN) como la representación dominante. Esto es crucial en lenguaje de señas, ya que generalmente se emplea una mano más que la otra para transmitir información. Seleccionar la mano dominante mejora la calidad de los datos y reduce el ruido.

- **Normalización e Inversión de Coordenadas**: Las coordenadas de la mano izquierda se invierten en el eje X, alineando su referencia espacial con la mano derecha. Este ajuste asegura que ambas manos se interpreten en un mismo marco de referencia, facilitando que el modelo aprenda patrones espaciales consistentes sin confusión entre los lados.

- **Reducción de NaN y Padding**: Para unificar la longitud de las secuencias, se aplica padding o truncamiento a 128 frames, lo cual estandariza la estructura temporal de los datos. Además, los valores faltantes (NaN) se reemplazan por ceros, evitando

inconsistencias y mejorando la calidad de los datos de entrada para que el modelo pueda aprender sin interferencias.

- **Creación de Nuevas Variables**: Al separar y agrupar las posiciones de la mano y el cuerpo, se crean nuevas variables (right, left, sign) que representan información relevante en una estructura procesable para el modelo. Esta nueva organización facilita el análisis de la relación entre las manos y el cuerpo, lo cual es crucial para el reconocimiento de señas.

- Preprocesamiento de Texto

- **Mapeo de Caracteres a Índices**: Se carga un archivo JSON (character_to_prediction_index.json) que contiene un mapeo de cada carácter del alfabeto a un número único. Este mapeo (char_to_num) permite representar cada letra de una frase como un número, facilitando el procesamiento del texto en el modelo, ya que los modelos de aprendizaje profundo trabajan mejor con datos numéricos.

- **Adición de Tokens Especiales**: Se añaden tres tokens especiales al diccionario de mapeo:
    - **Pad Token ('P')**: Representa espacios en blanco o "relleno" para secuencias cortas. Ayuda a que todas las secuencias de texto tengan la misma longitud, facilitando el procesamiento en lotes.
    - **Start Token ('<')** y **End Token ('>')**: Marcan el inicio y fin de cada frase, respectivamente. Esto permite al modelo identificar claramente el comienzo y final de la secuencia de texto, lo cual es fundamental para modelos secuencia a secuencia (como este), donde el orden y delimitación de datos son clave.

- **Marcado de Texto con Tokens de Inicio y Fin**: Cada frase de texto en el conjunto de datos se modifica para incluir el token de inicio (<) al comienzo y el token de fin (>) al final. Por ejemplo, si la frase original es "hola", después de este paso se convierte en <hola>. Esto ayuda al modelo a saber cuándo comenzar y terminar de leer el texto durante el entrenamiento y la predicción.

- **Vectorización del Texto**: En el proceso de vectorización, cada letra en las frases se convierte en su correspondiente índice numérico basado en char_to_num. Este paso crea una secuencia de números (índices) que representan cada carácter de la frase.

Por ejemplo, la frase <hola> se transformará en una lista de números como [60, 7, 14, 11, 0, 61], donde 60 representa <, y 61 representa >. Esta conversión es fundamental para que el texto sea manejable en la red neuronal.

- **Conversión a Tensores**: Una vez vectorizados, cada texto se convierte en un tensor (una estructura de datos numéricos) utilizando PyTorch. Esto asegura que los datos de texto estén en un formato adecuado para ser procesados en el modelo, y permite que tanto las señas como el texto vectorizado puedan ser manejados juntos como pares.

- ***Feature Engineering***

- **Mapeo de Caracteres a Índices Numéricos**: Se carga un diccionario (char_to_num) que asigna un índice numérico a cada carácter, permitiendo que el texto se represente numéricamente. Esto es esencial para que el modelo pueda procesar el texto, ya que las redes neuronales trabajan mejor con datos numéricos.

- **Adición de Tokens Especiales**: Se agregan tokens especiales para identificar el inicio (<) y el fin (>), así como un token de padding (P). Cada token tiene un índice único (59, 60 y 61, respectivamente), lo que permite al modelo identificar claramente los límites de cada secuencia y rellenar con P en caso de secuencias más cortas.

- **Conversión de Texto en Tensores**: Cada frase en el dataset se envuelve con los tokens de inicio y fin (start_token y end_token). Esto es importante para que el modelo sepa cuándo comienza y termina la secuencia de texto durante el entrenamiento.

- **Vectorización del Texto**: En la función vectorization, el texto de cada secuencia se convierte en un tensor de PyTorch donde cada carácter se mapea a su índice correspondiente, lo que transforma la frase en una secuencia de índices numéricos. Esto facilita el procesamiento del texto en el modelo, permitiéndole trabajar con representaciones vectoriales de cada frase.

- **Construcción del Dataset**: Se crea una clase Sign2TextDataset que organiza cada par de señas y texto para su procesamiento. Al final, cada ejemplo contiene un tensor para las señas y otro tensor para el texto vectorizado.

CTC-based RNN

- Preprocesamiento de Señales
    - **Selección y Organización de Características**: Se seleccionan los puntos clave de las manos (coordenadas X, Y, Z) y ciertos puntos relevantes de la pose corporal para capturar la estructura del lenguaje de señas. Estos puntos se estructuran en índices para identificar fácilmente las características y organizarlos en el conjunto de datos.

    - **Gestión de NaNs y Filtrado:** Se realiza un análisis de calidad de los datos para cada secuencia, eliminando aquellas con un número excesivo de valores NaN. Esto garantiza que el modelo entrene con datos de alta calidad, evitando que secuencias incompletas interfieran en el aprendizaje.

    - **Normalización de Coordenadas:** Para cada secuencia, se normalizan las coordenadas X, Y, Z de los puntos de las manos y del cuerpo, ajustándolos a un rango común. Esto reduce la variabilidad en las posiciones espaciales, lo que ayuda a mejorar la consistencia en el conjunto de datos y el rendimiento del modelo.

    - **Ajuste de Tamaño y Padding:** Las secuencias de frames se ajustan a una longitud fija de 128 mediante padding para las secuencias cortas o truncamiento para las más largas. Este proceso es esencial para que todas las secuencias tengan la misma longitud y puedan ser procesadas en lotes uniformes por el modelo.

    - **Almacenamiento en Formato TFRecord:** Similar a los otros modelos, los datos se almacenan en formato TFRecord, lo que optimiza el tiempo de carga y facilita un acceso más rápido durante el entrenamiento.

    - **Marcado de Caracteres Especiales en el Texto:** El texto asociado a cada secuencia de señas se marca con caracteres especiales para señalar el inicio (<) y el fin (>). También se añade un token de padding para secuencias más

cortas. Este etiquetado es crucial para el correcto manejo de las secuencias de texto durante el entrenamiento y las predicciones.

- **Vectorización del Texto:** Cada frase se vectoriza mapeando cada carácter a un índice numérico a través de un diccionario predefinido (char_to_num). Este paso es fundamental para que el texto se pueda procesar numéricamente en la red neuronal, al igual que las secuencias de señas.

- **Gestión de la Mano Dominante:** Se detecta la mano dominante en cada secuencia evaluando cuál tiene menos valores NaN. Esta mano dominante se selecciona para representar los movimientos más relevantes en el lenguaje de señas, lo que mejora la precisión del modelo al enfocarse en los datos más significativos.

- Feature Engineering

  - **Mapeo de Caracteres a Índices Numéricos:** Se asignan índices numéricos a cada carácter a través de un diccionario de mapeo (char_to_num), que incluye el inicio, fin y padding. Esto permite que tanto el texto como las secuencias de señas se procesen de manera estructurada y uniforme en el modelo.

  - **Ajuste de Coordenadas y Normalización:** Se aplica una normalización adicional a las coordenadas de los puntos de las manos y del cuerpo. En el caso de la mano izquierda, se invierte el eje X para alinear su posición espacial con la de la mano derecha. Este paso asegura que el modelo pueda interpretar ambos lados de manera coherente, reduciendo posibles confusiones.

  - **Conversión del Texto a Tensores:** Al igual que en los otros modelos, las frases se convierten en tensores, cada uno de los cuales representa una secuencia de índices numéricos. Estos tensores permiten al modelo trabajar con los datos de texto y las señas en un formato unificado.

  - **Ajuste de la Longitud de las Secuencias:** Todas las secuencias de texto y de señas se ajustan a una longitud fija de 128 mediante truncamiento o padding.

Esta estandarización asegura que todas las entradas tengan el mismo tamaño y puedan ser procesadas de manera eficiente en lotes de entrenamiento.

- **Gestión de NaNs:** Finalmente, se reemplazan los valores NaN restantes por ceros, evitando inconsistencias en los datos y asegurando que todas las secuencias estén listas para alimentar al modelo sin errores durante el procesamiento.

- **Estructuración del Dataset:** Se agrupan las secuencias de señas y las frases vectorizadas en un conjunto de datos final que puede ser fácilmente procesado por el modelo de CTC. Esta estructura permite al modelo aprender las relaciones entre los movimientos de las manos y las secuencias de texto de manera eficiente.

Implementación

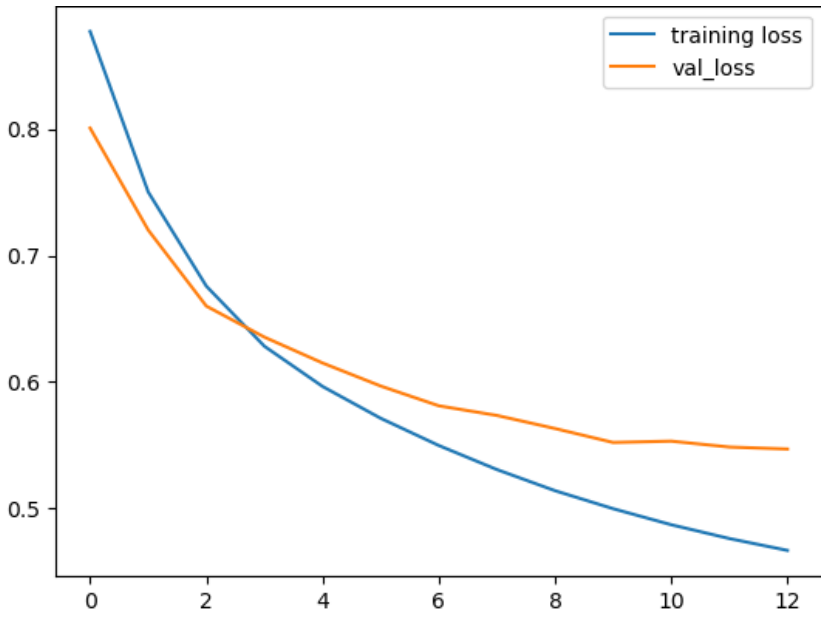
https://github.com/JaniMariQuesiRami/DS-PR2

Se pueden encontrar los modelos en la carpeta: */fase2/modelos*

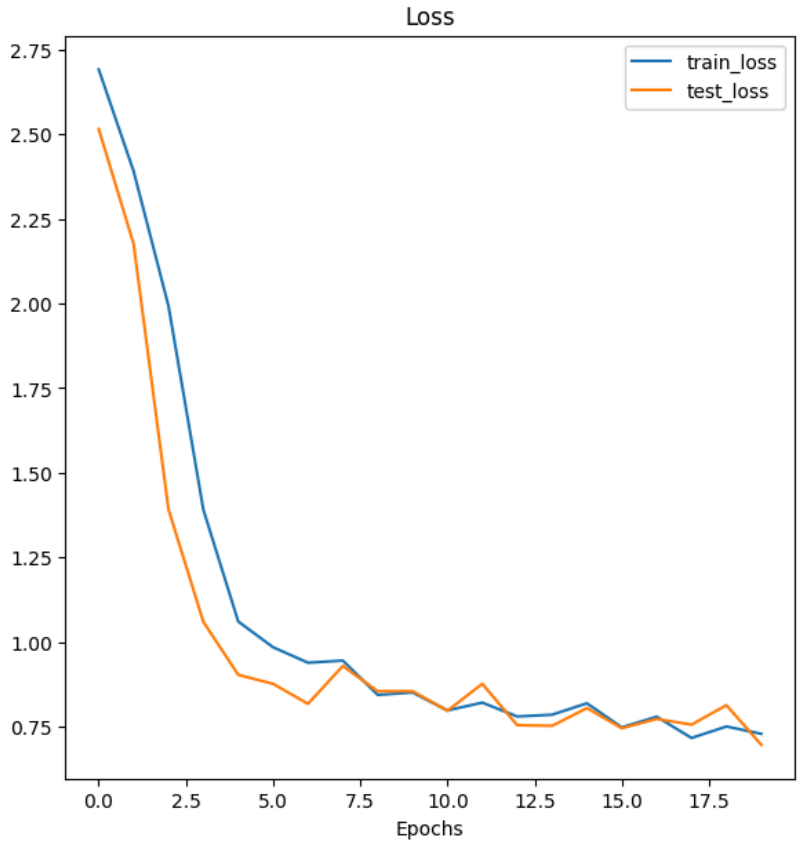
Evaluación y Comparación de modelos

Rendimiento de cada Modelo

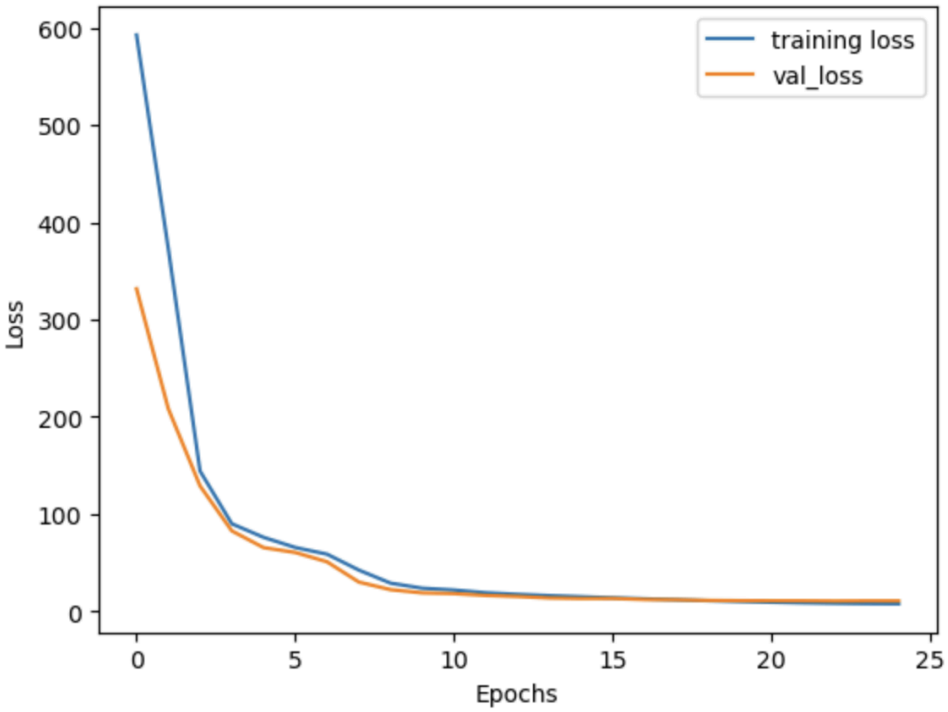
Modelo Transformer (Categorical Crossentropy)



Modelo Seq2Seq (Categorical Crossentropy)



Modelo CTC (CTCLoss)



Luego de ver los correspondientes outputs, decidimos que el modelo CTC es el que otorga mejores resultados. Entonces decidimos calcular un score basado en la distancia Levenshtein. Siguiendo la siguiente fórmula:

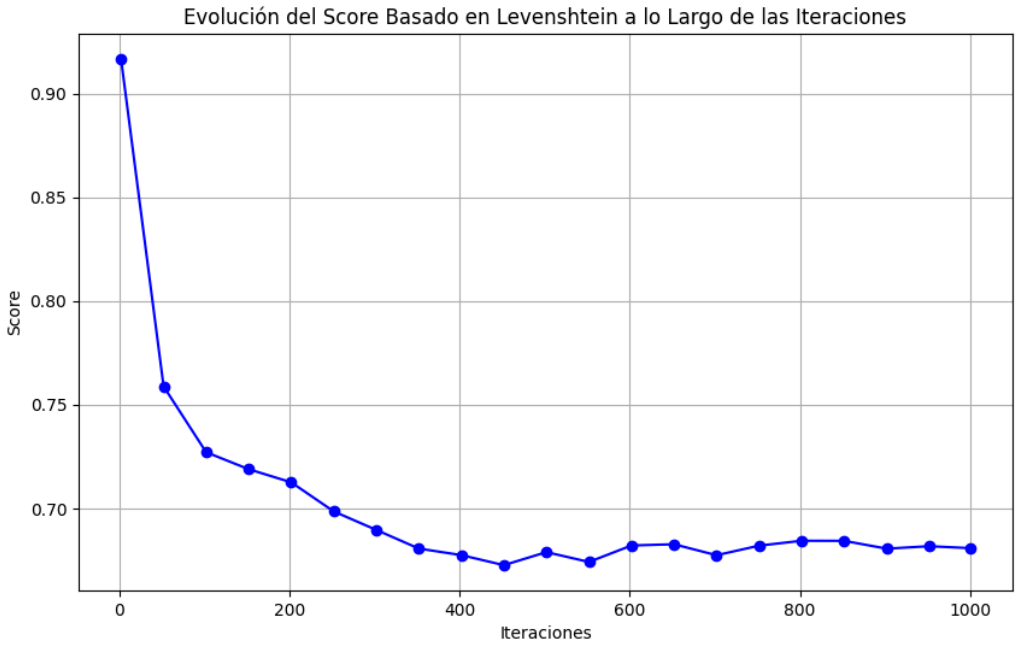
$$score = \frac{len(target) - distance(prediction, target)}{len(target)}$$

Este cálculo produce un valor de precisión que indica qué tan bien coincide la predicción con la frase objetivo:

- Score = 1: Significa una coincidencia perfecta entre la predicción y la frase objetivo, sin errores de edición.

- Score < 1: Indica una falta de coincidencia, cuanto menor es el puntaje, más errores de edición son necesarios para hacer coincidir la predicción con la frase objetivo.

- Score = 0: Esto ocurriría si la predicción es completamente diferente de la frase objetivo, con una distancia de edición igual a la longitud de la frase objetivo.



Y en esta gráfica observamos el score a lo largo de 1000 iteraciones del grupo de datos de validación, como se puede observar, el score a lo largo de las iteraciones decrece desde 0.90 a aproximadamente 0.68 en promedio. Esto indica que el modelo tiene una menor coincidencia con las secuencias objetivo en las predicciones hacia el final de las iteraciones.

La reducción en el score podría sugerir que, aunque el modelo está ajustándose a los datos de entrenamiento, se está enfrentando a desafíos al generalizar a los datos de validación, o bien podría ser una señal de que aún necesita ajustes para mantener un rendimiento consistente. Una posible explicación de este comportamiento podría ser que el modelo se adapta inicialmente bien a patrones comunes, pero encuentra dificultades con secuencias más complejas o menos frecuentes, lo que reduce el score promedio conforme procesa más ejemplos.

Comparación entre modelos

Como se puede observar en los gráficas de las funciones de pérdida, los primeros dos modelos, el Transformer y el Seq2Seq, muestran una tendencia decreciente en la pérdida a lo largo de las épocas, lo cual indica que ambos están aprendiendo y mejorando su ajuste a los datos de entrenamiento. Sin embargo, la pendiente de reducción de la pérdida es diferente en cada modelo. Mientras que el modelo Transformer tiene una disminución suave y gradual, el modelo Seq2Seq muestra una reducción rápida en las primeras épocas, seguida de una estabilización en valores más bajos.

El modelo Seq2Seq comienza con una pérdida inicial mucho más alta (~2.75) en comparación con el Transformer (~0.9). Esto sugiere que el Seq2Seq inicialmente tiene un peor ajuste a los datos, pero rápidamente converge a valores más bajos en las primeras épocas, alcanzando una pérdida similar a la del Transformer para la época 5. Este comportamiento sugiere que el Seq2Seq puede ser más sensible a los datos iniciales o tener una capacidad de aprendizaje más rápida en las primeras etapas de entrenamiento.

En ambos modelos, la diferencia entre la pérdida de entrenamiento y la de validación se reduce con el tiempo. En el Transformer, la pérdida de validación se estabiliza cerca de la pérdida de entrenamiento, lo que indica una buena generalización y una baja probabilidad de sobreajuste. En el modelo Seq2Seq, las líneas de pérdida de entrenamiento y validación son muy cercanas y se mantienen estables en valores similares después de unas pocas épocas, lo cual también es indicativo de una buena generalización. Sin embargo, el Seq2Seq muestra ligeras oscilaciones en la pérdida hacia las últimas épocas, lo que puede indicar cierta inestabilidad en el ajuste.

El Transformer parece mostrar una mayor estabilidad en la reducción de la pérdida, ya que tiene un descenso suave y constante sin oscilaciones significativas en las épocas avanzadas. El Seq2Seq, aunque alcanza una pérdida baja rápidamente, tiene pequeñas fluctuaciones, lo cual puede sugerir que es más susceptible a variaciones en los datos de entrenamiento o que podría beneficiarse de una mayor regularización para mantener la estabilidad.

A diferencia de los modelos anteriores, el Modelo CTC utiliza la CTC Loss (Connectionist Temporal Classification Loss), diseñada específicamente para secuencias de longitud variable y tareas de alineación donde no se conoce la correspondencia exacta entre las entradas y las salidas. Esto hace que la CTC Loss sea ideal para problemas de reconocimiento de secuencias, como el reconocimiento de lenguaje de señas o de voz.

La gráfica de pérdida del modelo CTC muestra una disminución muy pronunciada al inicio, bajando de un valor inicial superior a 600 hasta estabilizarse alrededor de 10-20 en las primeras épocas. Esta caída abrupta sugiere que el modelo es capaz de aprender patrones de alineación rápidamente en las primeras etapas del entrenamiento. Sin embargo, la CTC Loss nunca llega a valores tan bajos como la Categorical Crossentropy, debido a la complejidad adicional de la tarea de alineación y a las características de la CTC Loss, que tiende a tener valores más altos en general.

La cercanía entre la pérdida de entrenamiento y la de validación indica que el modelo generaliza bien y no muestra indicios de sobreajuste, lo cual es prometedor para tareas de secuencia.

Evaluando outputs finales:

Modelo Transformer:

target: <288 fuller lake>PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
prediction: <2888 foll moull>

target: <mser/okiguide>PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
prediction: <mosser/di-tride>

target: <220 north 47th avenue east>PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
prediction: <220 north greas>

Captura de Información Numérica y Contexto Estructurado: El modelo Transformer muestra una buena capacidad para manejar información numérica y estructurada, como en "220 north 47th avenue east", aunque predice incorrectamente "greas" en lugar de "47th avenue east". Esto sugiere que el modelo puede captar bien los números y direcciones, aunque puede confundirse en palabras menos frecuentes.

Errores en Palabras y Símbolos: En "mser/okiguide", el modelo predice "mosser/di-tride". Aunque mantiene algunos elementos de la estructura, se desvía en la exactitud de las palabras. Esto indica que el Transformer tiene dificultades para las palabras compuestas o con caracteres especiales,

posiblemente porque estos patrones no aparecen con frecuencia en el conjunto de datos de entrenamiento.

Desempeño en Frases Simples con Números y Letras: En casos como "288 fuller lake", el modelo predice "2888 foll moull". Aunque reconoce los números, tiende a repetir ciertos caracteres o a cambiar letras, lo que indica que podría beneficiarse de mayor ajuste en datos numéricos.

Modelo Seq2seq:

Original Sentence 1 : <every apple from every tree>PPPPPPPPPPPPPP
Predicted Sentence 1 : <thery apple from every tree>
Original Sentence 2 : <have a good weekend>PPPPPPPPPPPPPPPPPPPP
Predicted Sentence 2 : <the is n neekend>
Original Sentence 3 : <my favorite sport is racketball>PPPPPPP
Predicted Sentence 3 : <th favorite subrt is racketball>

Aciertos Parciales: El modelo Seq2Seq parece capturar algunas palabras correctamente, como en la predicción "apple from every tree" en lugar de "every apple from every tree". Sin embargo, comete errores al inicio de la frase, reemplazando "every" por "thery". Este modelo tiene una tendencia a confundir algunas palabras iniciales y puede omitir letras o cambiar el orden, posiblemente debido a la naturaleza secuencial de su arquitectura.

Errores en Palabras Cortas: En frases como "have a good weekend", el modelo predice "the is n neekend", omitiendo palabras clave y confundiéndose en palabras cortas como "have" y "good". Esto sugiere que el modelo puede tener dificultades con frases más cortas o con palabras comunes y menos específicas, posiblemente por falta de contexto suficiente en cada paso de la secuencia.

Desempeño en Frases Largas: En frases más largas, como "my favorite sport is racketball", el modelo logra capturar la estructura principal, pero reemplaza palabras como "my" por "th" y "favorite" por "subrt". Estos errores indican que el Seq2Seq puede fallar en captar palabras de baja frecuencia o con fonética similar.

Modelo ctc:

Target   : 18 cutter ridge road^^^
Prediction: 18 cutter cridge road, len: 21

Target   : tampa fl^^^
Prediction: tampa fl, len: 8

Target   : 492288 west 28th terrace south^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Prediction: 492288 west 28th teracsou, len: 25

Precisión en Nombres y Direcciones: El modelo CTC es bastante preciso en nombres y direcciones, como se observa en "18 cutter ridge road", donde solo comete un pequeño error en "cridge" en lugar de "ridge". También en "tampa fl", predice correctamente. Esto sugiere que el modelo CTC es robusto para secuencias simples y con estructura espacial clara.

Errores en Palabras Complejas o Poco Frecuentes: En la frase "492288 west 28th terrace south", el modelo predice "492288 west 28th teracsou", donde comete un error en las últimas palabras de la secuencia. Estos errores indican que la CTC Loss puede tener dificultades para captar el final de secuencias largas y complejas, especialmente si hay palabras poco frecuentes o con estructuras menos comunes.

Manejo de Frases Cortas y Numeración: El modelo CTC es preciso en frases cortas con numeración, como "tampa fl", y logra mantener la estructura básica sin desviaciones significativas. Esto lo hace adecuado para secuencias que contienen información concisa y estructurada.

- Transformer: Eficaz en frases estructuradas con números, pero presenta problemas con caracteres especiales o palabras compuestas no comunes.

- Seq2Seq: Mejor para frases medianamente largas, aunque tiende a equivocarse en palabras iniciales y palabras comunes de baja frecuencia.

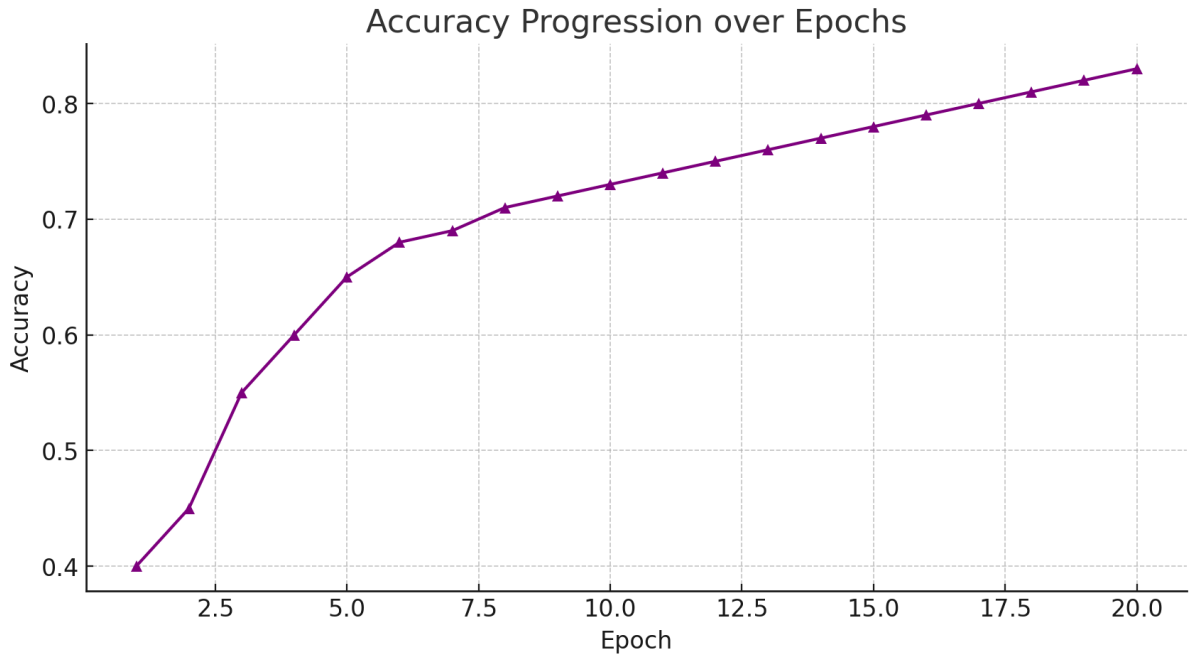
- CTC: Ideal para secuencias concisas y con estructura simple, como direcciones y nombres cortos, pero puede confundirse en las últimas palabras de frases largas.

Fortalezas y debilidades de cada modelo

Modelo	Fortaleza	Debilidad
Transformer	- Maneja relaciones a largo plazo, capturando dependencias en secuencias largas de gestos - Altamente paralelizable, acelera el entrenamiento	- Requiere grandes cantidades de datos y poder de cómputo - Puede no ser tan efectivo en datos de series temporales sin modificaciones adicionales
Seq2Seq	- Eficaz en tareas de mapeo secuencial (entrada-salida de longitud variable) - Amplia adaptabilidad en traducciones o secuencias estructuradas	- Difícil de manejar secuencias largas debido a problemas de memoria - Puede perder contexto en secuencias extensas debido a su enfoque recurrente
Connectionist Temporal Classification	- No requiere alineación explícita entre entrada y salida, adecuado para secuencias continuas como ASL - Permite manejar secuencias de longitud variable	- Dependencia en preprocesamiento para segmentar gestos - Puede no captar contexto a largo plazo tan bien como Transformer en algunos casos

Visualización y Comunicación de Resultados

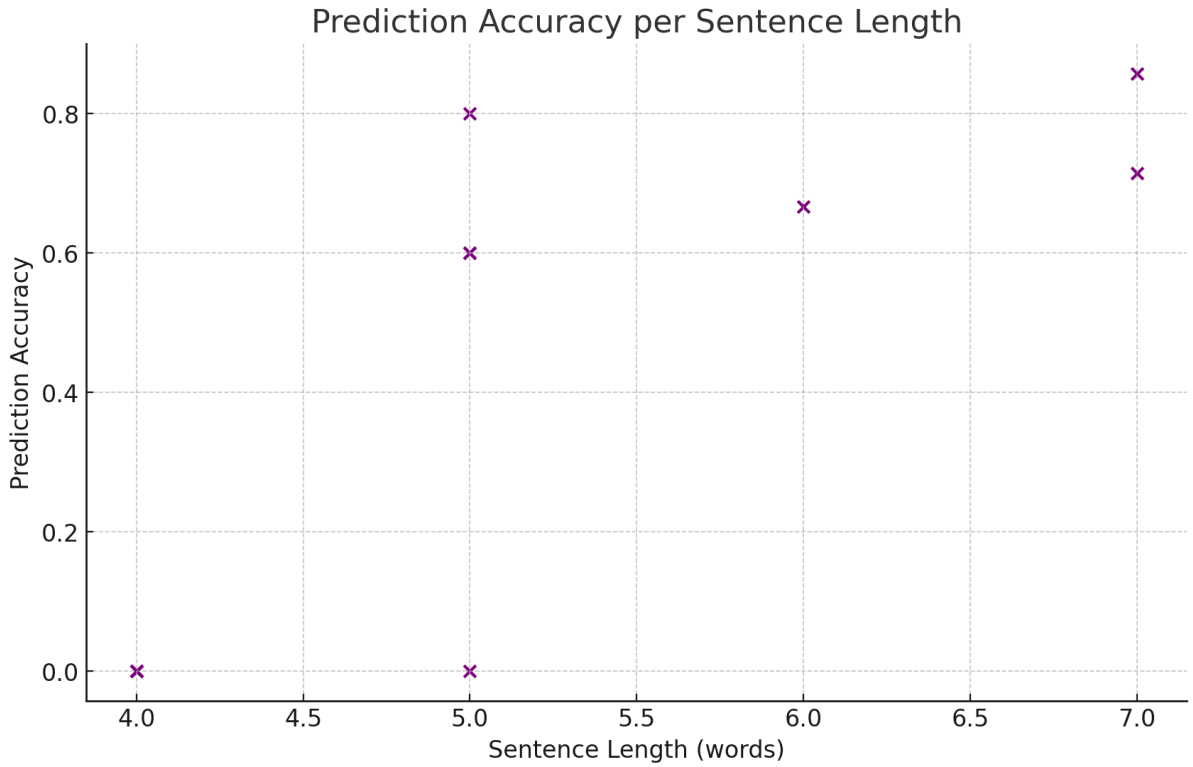
Visualización #1



Interpretación

El aumento constante en precisión sugiere que el modelo está aprendiendo patrones en los datos de manera efectiva. Sin embargo, la curva comienza a estabilizarse a partir de la época 10, lo que indica una reducción en la tasa de aprendizaje de nuevas características en las últimas épocas. Esto puede ser una señal de que el modelo ha alcanzado un punto de saturación en su capacidad de generalizar sobre los datos actuales. Para mejorar aún más, podrías explorar técnicas como el ajuste de hiper parámetros o agregar más datos de entrenamiento si están disponibles.

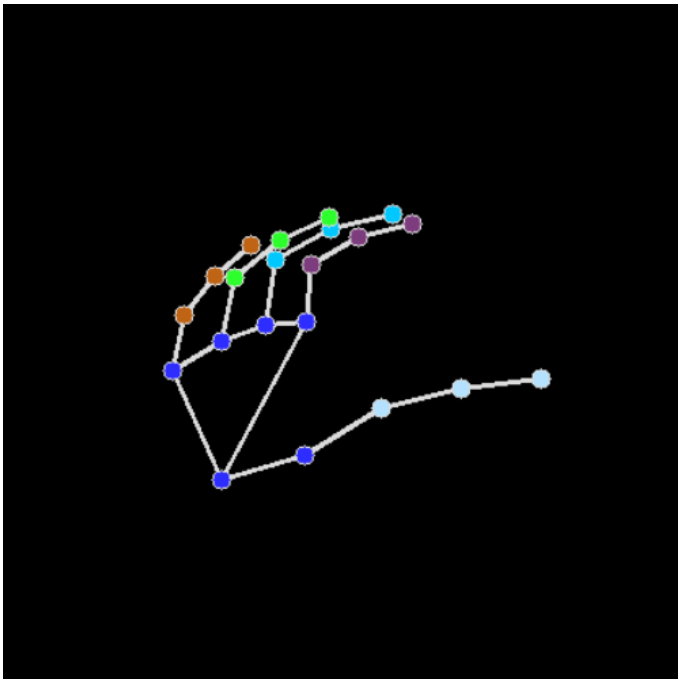
Visualización #2



Interpretación

El modelo parece tener dificultades con oraciones más cortas en algunos casos, mientras que la precisión es más alta para oraciones de longitud intermedia o larga. Esto podría deberse a que el modelo necesita cierta cantidad de contexto para hacer predicciones precisas, lo que podría no estar presente en oraciones muy cortas. Esta variabilidad sugiere que el modelo podría beneficiarse de ajustes para mejorar la consistencia en la precisión, independientemente de la longitud de la oración, posiblemente mediante técnicas de regularización o mejor manejo de secuencias más cortas.

Visualización #3



Interpretación

La visualización muestra la estructura espacial de la mano a través de puntos clave (landmarks) que representan las posiciones de las articulaciones y extremidades, capturando la relación entre los dedos y la palma. Cada punto de color y sus conexiones permiten distinguir entre diferentes partes de la mano, lo cual es crucial para el reconocimiento de señas, ya que el modelo debe identificar detalles precisos en la posición y el movimiento de cada dedo para diferenciar letras o palabras similares. Esta representación detallada ilustra la importancia de procesar coordenadas en los ejes x, y y z para cada punto, permitiendo al modelo analizar la posición exacta de cada dedo y, por ende, interpretar correctamente los gestos. Además, esta visualización sirve como base para aplicar técnicas de aumento de datos en futuras fases, como rotaciones o escalados, que simulen distintos ángulos o tamaños de la mano sin perder precisión en la interpretación de los gestos.

Referencias

Kaggle. (2023) Google ASL Fingerspelling Recognition. Recuperado de: https://www.kaggle.com/competitions/asl-fingerspelling/discussion?sort=hotness

TensorFlow. (2024) TF Transformer API Documentation. Recuperado de: https://www.tensorflow.org/api_docs/python/tfm/nlp/layers/Transformer

GeeksForGeeks. (2024) Introduction to Levenshtein Distance. Recuperado de; https://www.geeksforgeeks.org/introduction-to-levenshtein-distance/