Universidad Internacional de Valencia

Diseño de un Sistema para el Reconocimiento de Caracteres Escritos a Mano utilizando Aprendizaje Profundo

Titulación:
Máster en Inteligencia artificial
Curso académico 2023 – 2024

Alumno/a: Guardado Alonso Luis
D.N.I: 20454689F

Director/a de TFM: Junior Altamiranda

Convocatoria:

Tercera

14 de octubre de 2024

De:
Planeta Formación y Universidades

Agradecimientos

A mi mujer le agradezco haber cuidado de nuestra familia y nuestro hogar durante este difícil año. Si no fuese por ella la realización del máster que culmina con este trabajo sencillamente no habría sido posible.

Les agradezco a mis lectoras beta, ya que seis ojos ven más que dos. Especialmente dos tan miopes como los míos.

Le agradezco a mi tutor por su inestimable guía y consejo que ha conducido este trabajo a buen puerto.

Finalmente, le agradezco a la inteligencia artificial generativa, sin ella este trabajo no sería lo que es hoy.

Índice

Índice de figuras

Índice de tablas

Resumen

El reconocimiento de escritura manuscrita representa un desafío significativo en la intersección entre la visión por computador, el procesamiento del lenguaje natural y el aprendizaje automático. Este trabajo presenta un estudio exhaustivo sobre la evolución de los métodos empleados en el reconocimiento de caracteres manuscritos, centrándose en los resultados obtenidos con el conjunto de datos IAM Handwriting Database.

Se analiza la progresión desde los modelos ocultos de Márkov hasta las arquitecturas más recientes basadas en redes neuronales profundas. En particular redes neuronales convolucionales, redes neuronales recurrentes y transformadores. El estudio aborda en profundidad los fundamentos teóricos de estas técnicas.

El trabajo incluye una exploración practica de los hiperparámetros de la arquitectura convolucional recurrente, la cual permite una comparación directa de su rendimiento. Los resultados obtenidos se analizan en el contexto de la evolución histórica del campo para proporcionar una visión completa del estado actual del reconocimiento de escritura manuscrita y sus posibles direcciones futuras.

Recientemente un nuevo actor ha tomado protagonismo en el campo del procesamiento del lenguaje natural: los grandes modelos del lenguaje. Estos modelos también han demostrado ser efectivos en el reconocimiento de texto manuscrito. Pese a que entrenar estos modelos está fuera de las capacidades del *hardware* disponible para el desarrollo de este estudio, se probarán varios modelos comerciales para evaluar su rendimiento sobre el conjunto de datos IAM.

El código fuente para realizar los entrenamientos puede consultarse en el **Apéndice II**.

Abstract

Handwriting recognition represents a significant challenge at the intersection of computer vision, natural language processing, and machine learning. This work presents a comprehensive study on the evolution of methods used in handwritten character recognition, focusing on the results obtained with the IAM Handwriting Database.

The progression from hidden Markov models to the most recent architectures based on deep neural networks is analyzed. In particular, convolutional neural networks, recurrent neural networks, and transformers. The study delves into the theoretical foundations of these techniques.

The work includes a practical exploration of the hyperparameters of recurrent convolutional architecture, allowing for a direct comparison of their performance. The obtained results are analyzed in the context of the field's historical evolution to provide a comprehensive view of the current state of handwritten character recognition and its potential future directions.

Recently, a new player has taken center stage in the field of natural language processing: large language models. These models have also proven to be effective in handwritten text recognition. Although training or retraining these models is beyond the capabilities of our hardware, in this work we will test several commercial models to evaluate their performance on the IAM dataset.

The source code to perform the training can be found in **Apéndice II**.

1.Introducción

En la era digital actual, donde la tecnología permea cada aspecto de nuestras vidas, la interacción entre el mundo analógico y el digital sigue siendo un desafío fascinante y complejo. El reconocimiento de escritura a mano (HWR por sus siglas en inglés), un campo que se encuentra en la intersección de la visión por computador, el procesamiento del lenguaje natural y el aprendizaje automático, ejemplifica perfectamente este desafío. El presente trabajo se sumerge en las profundidades de este campo, explorando desde un punto de vista teórico y práctico las distintas técnicas de aprendizaje automático que se han empleado para mejorar la precisión y eficiencia en el reconocimiento de texto manuscrito.

La escritura a mano, una de las formas más antiguas y personales de comunicación humana, sigue siendo relevante en numerosos contextos: desde la digitalización de documentos históricos hasta la automatización de procesos en sectores como la banca, la salud o la administración pública. Sin embargo, la variabilidad inherente en los estilos de escritura, la calidad de los trazos y las condiciones de captura de imágenes hacen que el reconocimiento automatizado de texto manuscrito sea un problema particularmente desafiante en el campo de la inteligencia artificial.

Este trabajo se centra en el conjunto de datos IAM Handwriting Database, desarrollado por Marti & Bunke, (2002), una colección de referencia en el campo del reconocimiento de escritura a mano. Este conjunto de datos, que contiene muestras de escritura de cientos de escritores diferentes, proporciona un terreno fértil para el desarrollo y evaluación de algoritmos de aprendizaje automático.

A lo largo de esta investigación, se explorarán diversas arquitecturas con las que se ha tratado de resolver la tarea de reconocimiento planteada en IAM. Se estudiarán desde los modelos ocultos de Márkov (HMM por sus siglas en inglés) apoyados en perceptrones multicapa (MLP por sus siglas en inglés) hasta modelos de atención y transformadores, pasando por las redes neuronales recurrentes (RNN por sus siglas en inglés) y convolucionales (CNN por sus siglas en inglés). Se mostrará la evolución de los resultados de los distintos métodos y se realizará una descripción teórica de todos ellos.

También se entrenarán algunos de los algoritmos más significativos, explorando su espacio de hiperparámetros para entenderlos en profundidad. De entre todas las pruebas realizadas se seleccionarán los mejores modelos para compararlos con el estado del arte.

1.1. Estado del arte

El campo del reconocimiento de escritura a mano ha experimentado una evolución notable en las últimas dos décadas, para ilustrarla se empleará la tarea de reconocimiento de textos manuscritos contenida en el conjunto de datos IAM. Esta base de datos, que se ha convertido en un estándar de facto para evaluar el rendimiento de los sistemas de reconocimiento de texto manuscrito, ha sido testigo de una progresión significativa en las técnicas y arquitecturas empleadas, reflejada en la constante reducción de la tasa de error por carácter (CER por sus siglas en inglés), como ilustra la **Figura 1**. En la cual se ha recopilado todos los artículos que han reportado resultados sobre la tarea. Se ha elegido la métrica CER ya que es la que se utiliza mayoritariamente en la literatura, siendo puntuales los artículos que utilizan la tasa de error por palabra (WER por sus siglas en inglés) en su lugar. Debido a ello algunos artículos, como (Bertolami & Bunke., 2008), han quedado excluidos de este análisis. Para ilustrar la evolución del campo se han resaltado con distintos colores las diferentes arquitecturas empleadas y con diferentes colores los distintos métodos de evaluación. La figura contiene las citas de los artículos que mejoraron el estado del arte, el resto de citas pueden consultarse en el **Apéndice I**.

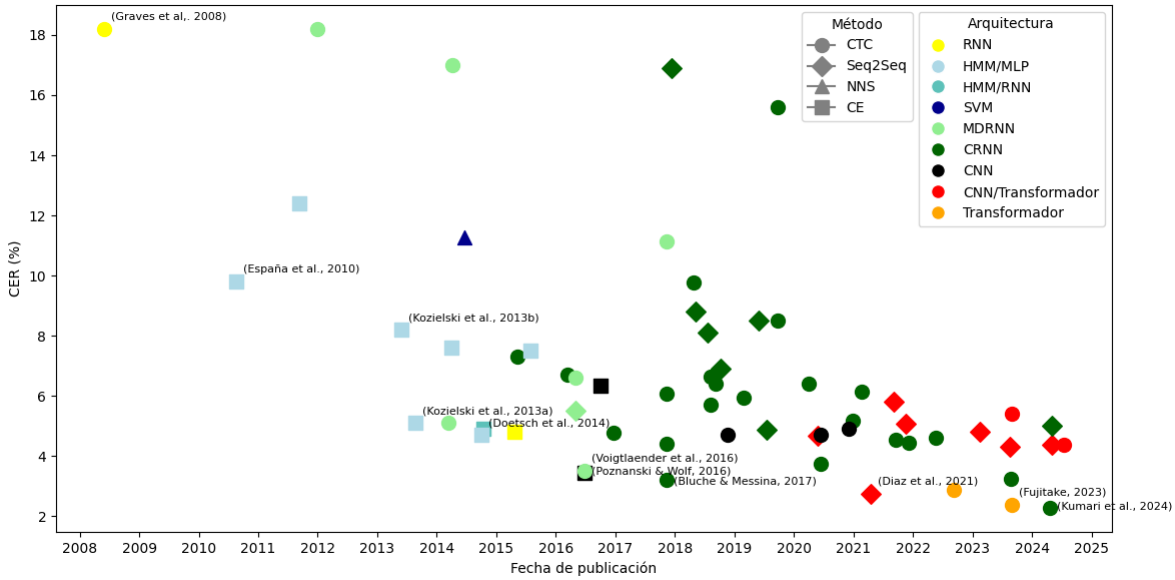


Figura 1: Evolución de la CER sobre el conjunto de datos IAM. Se muestran en distintos colores las diferentes arquitecturas empleadas y con diferentes colores los distintos métodos de evaluación. Se han etiquetado los artículos que mejoraron el estado del arte.

Los primeros autores en acometer la tarea del reconocimiento de caracteres manuscritos con el conjunto de datos IAM fueron Graves et al. (2008), que lograron una CER de 18%. Diseñaron una arquitectura de redes neuronales recurrentes y la entrenaron con la función de pérdidas de la clasificación temporal conexionista (CTC por sus siglas en inglés). Este singular enfoque no ha sido repetido por otros autores ya que, aunque las RNNs han sido ampliamente usadas, suelen ir acompañadas de un módulo de visión CNN o ser multidimensionales (MDRNNs).

En los años siguientes, se exploraron enfoques híbridos que combinaban las fortalezas de diferentes arquitecturas. Los modelos de Márkov combinados con perceptrones multicapa (HMM/MLP) o redes recurrentes (HMM/RNN) surgieron como intentos de fusionar la robustez estadística de los HMM con la capacidad de aprendizaje de características de las redes neuronales. Estos enfoques híbridos, ejemplificados por los trabajos de España et al., (2010), Kozielski et al., (2013a, 2013b) y Doetsch et al. (2014), que lograron empujar el estado del arte hasta CERs de 9,8%, 8,2%, 5,1% y 4,7%, respectivamente.

En 2014 el equipo de Almazán et al. propuso un singular método basado en máquinas de soporte vectorial (SVM por sus siglas en inglés) apoyadas por búsqueda de vecinos cercanos (NNS por sus siglas en inglés) que no tuvo ninguna repercusión. En este estudio no se tratará esta aproximación debido a su efímero recorrido.

El enfoque basado en redes neurales recurrentes puras no obtuvo tracción principalmente debido a que estas redes trabajan con series unidimensionales, mientras que las imágenes contienen patrones bidimensionales. No obstante, este paradigma cambió cuando Graves et al. (2007) introdujeron las redes recurrentes multidimensionales, las cuales permiten trabajar directamente sobre las imágenes. Este tipo de redes empezaron teniendo unos resultados modestos (Liwicki et al., 2012; Louradour & Kermorvant, 2014), aunque en los siguientes años demostraron ser una técnica tan válida como los HMM. De hecho es la técnica utilizada por Voigtlaender et al. (2016) para alcanzar una CER de 3,5%, récord que sería roto a los pocos días por Poznanski & Wolf (2016) quienes lo bajaron a 3,44% utlizando una CNN.

Pese al éxito cosechado por las MDRNNs, esta arquitectura sería abandonada durante la segunda mitad de la década de 2010 en favor de una arquitectura conceptualmente más sencilla: las redes neuronales convolucionales recurrentes (CRNNs por sus siglas en inglés). Las cuales constan de dos partes: un codificador convolucional que se encarga de extraer características de las imágenes y un decodificador recurrente que se encarga de analizar la secuencia de características extraídas. Este tipo de arquitecturas han dominado completamente el campo durante un lustro y han logrado establecer los récords de 3,2% (Bluche & Messina, 2017) y 2,27% (Kumari et al., 2024), siendo este último el estado del arte actualmente.

No obstante, esa dominancia comenzó a resquebrajarse tras la introducción de los transformadores (Vaswani et al., 2017), los cuales son una potente alternativa a las redes neuronales recurrentes. Aunque las RNNs no desaparecieron completamente (de hecho, actualmente conservan el récord), sí tuvieron que ceder protagonismo a las arquitecturas basadas en transformadores, las cuales lograron establecer los récords de 2,75% (Diaz et al., 2021) y 2,38% (Fujitake, 2024). Este último es especialmente interesante porque se basa en una nueva tendencia al alza: los grandes modelos del lenguaje (LLM por sus siglas en inglés).

El estado actual del arte en el reconocimiento de escritura a mano sobre el conjunto de datos IAM refleja no solo los avances en arquitecturas de redes neuronales, sino también mejoras significativas en técnicas de preprocesamiento, estrategias de aumento de datos y métodos de optimización. Con cada iteración, los investigadores se acercan más a la precisión humana en esta tarea desafiante, abriendo nuevas posibilidades para la digitalización y análisis automatizado de documentos manuscritos.

1.2. Planteamiento del problema

El reconocimiento de escritura manuscrita consiste en la capacidad de un sistema computacional para interpretar y entender la escritura a mano, ya sea en forma de textos individuales (como palabras y frases) o de signos específicos (como números y caracteres). Es un problema que plantea un reto en los campos de la inteligencia artificial y el procesamiento de imágenes. A pesar de los avances en las técnicas de aprendizaje profundo y el procesamiento del lenguaje natural, la variabilidad inherente a la escritura humana (variaciones en el estilo, la inclinación, la presión y la rapidez con la que se escribe) plantea obstáculos considerables para lograr un reconocimiento preciso y robusto.

Los sistemas actuales de HWR han logrado resultados prometedores en tareas específicas y con conjuntos de datos limitados. Sin embargo, su rendimiento disminuye significativamente cuando se enfrentan a escritura cursiva sin restricciones, variaciones en el estilo de escritura, y vocabularios extensos. Además, la mayoría de los enfoques existentes se centran en el reconocimiento a nivel de caracteres o palabras, sin aprovechar plenamente la información contextual y lingüística disponible a nivel de oración o documento.

El principal problema que se busca abordar es la baja precisión y reducir la tasa de error en el reconocimiento automático de textos manuscritos, especialmente en textos que presentan variaciones estilísticas significativas o que están escritos por varias personas con distintas caligrafías. Esto se traduce en retos como:

- **Variabilidad de la escritura**: Diferentes estilos de escritura, incluyendo diferentes tamaños, inclinaciones y presiones, dificultan el reconocimiento.

- **Ruidos en las imágenes**: Los documentos manuscritos pueden contener manchas, o sombras que afectan la calidad de los datos.

- **Escasez de datos etiquetados**: La falta de conjuntos de datos grandes y diversos que representen adecuadamente la variabilidad de la escritura a mano cuando se utilizan técnicas de aprendizaje supervisado.

Actualmente se busca abordar las limitaciones en el reconocimiento del texto manuscrita, aprovechando técnicas de aprendizaje profundo para mejorar la precisión y la adaptabilidad de los sistemas automatizados. La resolución de este problema no solo permitirá una mejor digitalización de documentos manuscritos, sino que también abrirá nuevas posibilidades en este campo.

1.3. Justificación

El reconocimiento del texto manuscrito es un área activa de investigación, y con los avances en técnicas de aprendizaje profundo y la disponibilidad de conjuntos de datos para su estudio, se espera que la precisión y la aplicabilidad de los modelos planteados continúen mejorando. A continuación, se presentan los principales puntos que permiten estudiar este problema utilizando aprendizaje profundo:

1. **Eficiencia de modelos profundos:** El aprendizaje profundo ha demostrado ser altamente eficaz en tareas de reconocimiento de patrones, incluidas las imágenes.

    - **Redes neuronales convolucionales (CNNs)**: Estas son eficientes para extraer características de imágenes. Se utilizan para reconocer patrones en las imágenes de escritura manuscrita.

    - **Redes neuronales recurrentes (RNNs)**: Son útiles para procesar secuencias y pueden manejar la variación en la longitud de la entrada (por ejemplo, diferentes longitudes de palabras).

    - **Transformadores**: Estas arquitecturas combinan la recurrencia con mecanismos de atención y encajes posicionales. Este maridaje facilita enormemente la paralelización del entrenamiento y ha permitido escalar modelos hasta los billones de parámetros.

    - **Modelos híbridos**: Combinaciones de CNNs con RNNs, (CRNNs) o con transformadores que extraen características y luego de manera eficiente.

2. **Manejo de la variedad:** El texto manuscrito presenta una gran variabilidad debido a diferentes estilos de escritura, inclinaciones, tamaños y presiones. Los modelos de aprendizaje profundo, a través de su capacidad de generalización, pueden aprender a manejar estas variaciones y reconocer caracteres independientemente de la persona que los escribe.

3. **Extracción automática de características:** Capacidad de extraer automáticamente características relevantes de los datos. Esto significa que la red puede aprender a identificar características importantes (como trazos, curvas y estilos) sin necesidad de un preprocesamiento exhaustivo.

4. **Mejora continua con datos:** Los modelos de aprendizaje profundo son escalables y pueden mejorar continuamente a medida que se les proporciona más datos. Esto es especialmente relevante en el campo de la escritura manual, donde la disponibilidad de grandes conjuntos de datos etiquetados, como IAM o RIMES, permite entrenar modelos más robustos y precisos.

    - **Recolección de datos:** Se requieren grandes conjuntos de datos de ejemplos de escritura manual. Los datos pueden incluir imágenes de texto escrito a mano y sus correspondientes transcripciones.

    - **Preprocesamiento:** Las imágenes se pueden escalar, recortar y normalizar para facilitar el proceso de entrenamiento. Además, el proceso típico de reconocimiento de texto manuscrito pasa por segmentar todo el texto en líneas y procesarlas separadamente.

    - **Aumento de datos:** Se pueden aplicar técnicas como rotación, traslación o distorsiones para aumentar la diversidad del conjunto de datos.

5. **Resultados de estado del arte:** El uso de aprendizaje profundo ha llevado a logros significativos en comparación con métodos más tradicionales, como los basados en el procesamiento de imágenes y algoritmos de aprendizaje automático clásico. Muchos estudios e implementaciones han demostrado que las arquitecturas profundas pueden superar a los métodos tradicionales en términos de precisión y tiempo de inferencia.

6. **Aplicaciones Prácticas:** El reconocimiento de escritura manuscrita tiene aplicaciones prácticas en diversas áreas, como la digitalización de documentos, la entrada de datos en dispositivos móviles, asistentes virtuales y la educación. La implementación de modelos de aprendizaje profundo en estos contextos puede mejorar la eficiencia y la experiencia del usuario.

    - **Digitalización de documentos**: Transformar documentos escritos a mano en texto digital.

    - **Interfaces de usuario**: Permitir que los usuarios ingresen información a través de escritura a mano en dispositivos móviles o tabletas.

    - **Reconocimiento de formularios**: Extraer texto escrito a mano en formularios y documentos administrativos.

El reconocimiento de caracteres manuscritos utilizando aprendizaje profundo representa un avance significativo en el campo de la inteligencia artificial, ofreciendo una solución precisa, flexible y adaptable a una amplia gama de aplicaciones. Su capacidad para aprender de los datos y adaptarse a diferentes estilos de escritura lo convierte en una tecnología con un gran potencial para transformar la forma en que se interactúa con la información.

1.4. Objetivos

En esta sección se establecen los objetivos tanto generales como específicos del presente trabajo.

1.4.1. Objetivo general

El objetivo principal de este trabajo es realizar el Diseño de un Sistema para el Reconocimiento de Caracteres Escritos a Mano utilizando Aprendizaje Profundo.

1.4.2. Objetivos específicos

- Estudiar los conceptos teóricos del reconocimiento de caracteres escritos a mano.
- Estudiar la evolución de los métodos más usados para el reconocimiento de caracteres manuscritos a lo largo de la historia del campo, sobre el conjunto de datos IAM.
- Estudiar los modelos de aprendizaje profundo utilizados para el reconocimiento de caracteres escritos a mano.
- Comparar los resultados obtenidos a partir de los diferentes modelos planteados para el reconocimiento de caracteres escritos a mano.
- Evaluar los resultados obtenidos con el modelo planteado para el reconocimiento de caracteres escritos a mano.

1.5. Estructura del documento

La estructura de este trabajo es presentada a continuación.

En el Capítulo 1, se comienza con una introducción que establece el ámbito de estudio, presentando el tema y, posteriormente, su motivación. Seguidamente, en el estado del arte se abordan las investigaciones previas que se han realizado sobre el tema y sus respectivas metodologías, resultados y conclusiones. Después, se describe el planteamiento del problema, los objetivos y la estructura, que establecen el marco en el que se desarrolla el trabajo, fijan las expectativas sobre los resultados y proporcionan al lector una guía sobre cómo está estructurado el documento.

En el Capítulo 2, en el contexto, se explican los conceptos que sustentan los diferentes métodos utilizados en HWR y sus técnicas asociadas para proporcionar los conocimientos previos esenciales para comprender el trabajo. En este capítulo se exponen las bases de sistemas cásicos como los HMM y basados en redes neuronales como las CNNs, RNNs y transformadores. Además, se presentan las métricas que se emplean para evaluarlos.

En el Capítulo 3, se presentan los materiales y métodos que detallan el conjunto de datos y la metodología empleados para analizar y desarrollar el modelo. También se describen las aplicaciones de las distintas técnicas utilizadas, así como las técnicas de validación y evaluación.

En el Capítulo 4, se presentan los resultados en función de los objetivos establecidos en el trabajo.

En el Capítulo 5, se muestra el análisis de los resultados del estudio para conocer la eficacia y eficiencia del modelo propuesto.

El Capítulo 6 contiene las conclusiones del trabajo y se presentan futuras líneas de investigación.

2. Marco teórico

El reconocimiento de escritura a mano es un campo interdisciplinario que combina elementos de procesamiento de imágenes, aprendizaje automático y reconocimiento de patrones. Este marco teórico proporciona una base sólida para comprender los fundamentos, técnicas y métricas utilizadas en el desarrollo y evaluación de sistemas de reconocimiento de texto manuscrito.

En las siguientes secciones, se explorarán en detalle las métricas de evaluación estándar, así como los diversos modelos y arquitecturas que han sido fundamentales en el avance de este campo. Desde los clásicos modelos ocultos de Márkov hasta las más recientes arquitecturas basadas en transformadores, cada enfoque ha contribuido significativamente a la mejora de la precisión y eficiencia en el reconocimiento de escritura a mano.

2.1. Modelo oculto de Márkov (HMM)

El modelo oculto de Márkov (Markov, 1913) es una técnica estadística que ha desempeñado un papel crucial en el desarrollo de sistemas de reconocimiento de escritura a mano, especialmente antes del auge de las técnicas de aprendizaje profundo. Los HMM son particularmente eficaces para modelar secuencias temporales o espaciales de datos, lo que los hace adecuados para el reconocimiento de texto manuscrito.

Un HMM es un modelo probabilístico que asume que el comportamiento observable, o_t, del sistema está gobernado por un estado oculto, q_t, que cumple la propiedad de Márkov, es decir que su valor actual depende únicamente de su valor inmediatamente anterior, q_{t-1}. En la **Figura 2** se ilustra el funcionamiento de un HMM. La hipótesis de Márkov en general no es cierta en el lenguaje, no obstante, es una aproximación que históricamente ha dado buenos resultados.

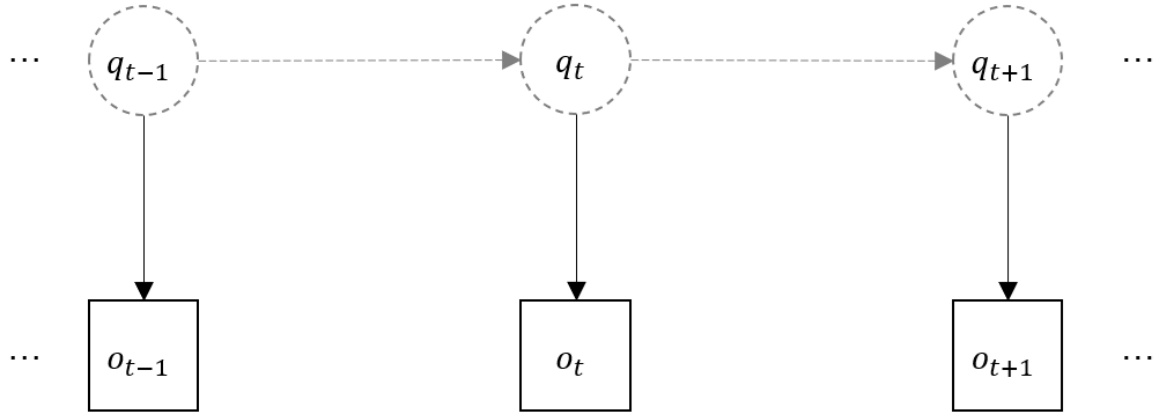


Figura 2: Visualización de la inferencia de un HMM. En gris y con línea punteada los estados ocultos, en negro los observables.

Un HMM se compone de los siguientes elementos (cuyas relaciones pueden visualizarse en la **Figura 3**):

- Un conjunto de estados ocultos, $Q = \{1,2,\dots,N\}$, que representan cada uno de los N posibles valores que puede tomar la variable oculta.
- Un conjunto de valores, $V = \{v_1, v_2, \dots, v_M\}$, que representan los posibles valores que pueden tomar las observaciones. En el caso del HWR, serían las posibles letras que componen el alfabeto o palabras que componen el vocabulario.
- Las probabilidades, $\pi = \{\pi_i\}$, de que el estado inicial de la secuencia sea i.
- Las probabilidades, $A = \{a_{ij}\}$, de transicionar del estado i al j:

$$a_{ij} = P(q_t = j | q_{t-1} = i) \quad (1)$$

- Las probabilidades, $B = \{b_j(v_k)\}$, de que la observación v_k sea producida por el estado j:

$$b_j(v_k) = P(o_t = v_k | q_t = j) \quad (2)$$

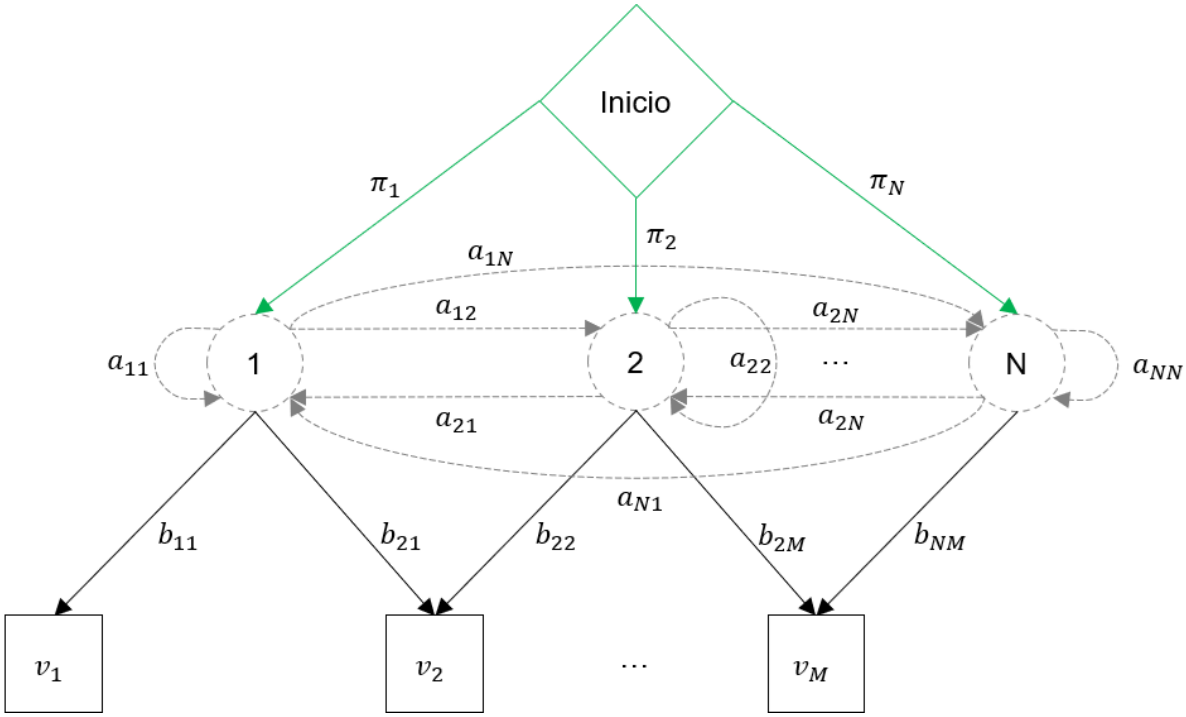


Figura 3: Visualización los parámetros un HMM. En gris y con línea punteada los estados ocultos y sus probabilidades de transición. En negro los observables y las probabilidades de observación. En verde las probabilidades del estado oculto inicial.

El entrenamiento de un HMM consiste en, dada una secuencia de observables, $O = (o_1, o_2, \dots, o_T)$, encontrar los parámetros (π, A, B) que mejor la explican. Además, en el campo del procesamiento del texto manuscrito, se suele combinar este procedimiento con redes neuronales que se encargan de estimar la función de probabilidades B.

2.2. Red neuronal convolucional (CNN)

La semilla de la que provienen las redes neuronales convolucionales fue plantada por Fukushima (1980) mediante su modelo Neocognitrón. Este fue recibiendo mejoras con los años que permitieron su compatiblidad con el algoritmo de entrenamiento hacia atrás (LeCun et al., 1998) y su paralelización mediante GPUs (Ciresan et al., 2011). Sin embargo, el auténtico punto de inflexión en el uso de las CNNs en tareas de visión vino de la mano de AlexNet (Krizhevsky et al., 2012), la cual barrió a los otros métodos en la competición ImageNet de 2012.

La arquitectura de las CNNs se inspira en el funcionamiento del córtex visual biológico, lo que les confiere una habilidad única para procesar datos con una estructura similar a una cuadrícula, como es el caso de las imágenes. Esta similitud biológica se traduce en tres principios fundamentales que definen su funcionamiento: campos receptivos locales, pesos compartidos y submuestreo espacial. Estos principios permiten a las CNNs capturar eficientemente las características locales y las relaciones espaciales en las imágenes de caracteres manuscritos, aspectos cruciales para su correcta identificación.

En el corazón de una CNN se encuentran las capas convolucionales, que aplican filtros, $\boldsymbol{K}$, (también llamados *kernels*) a las imágenes de entrada, $\boldsymbol{x}$. Estos filtros actúan como detectores de características, capaces de identificar desde trazos simples en las primeras capas hasta formas más complejas en las capas más profundas. Al resultado de la convolución se le añade un sesgo, $\boldsymbol{b}$, y finalmente se le aplica una función de activación, f, para obtener las salidas de la capa:

$$\boldsymbol{y} = f(\boldsymbol{K} \otimes \boldsymbol{x} + \boldsymbol{b}) \quad (3)$$

La operación de convolución se puede expresar matemáticamente como

$$[\boldsymbol{K} \otimes \boldsymbol{x}]_{ij\mu} = \sum_{n,m,\nu} K_{ij\mu}\, a_{i+n,j+m,\nu} \quad (4)$$

Donde índices latinos representan las coordenadas espaciales de la imagen y los índices griegos representan el número de filtros (o los canales de color de la imagen original). Para un único filtro y canal se puede visualizar la convolución mediante la **Figura 4**.

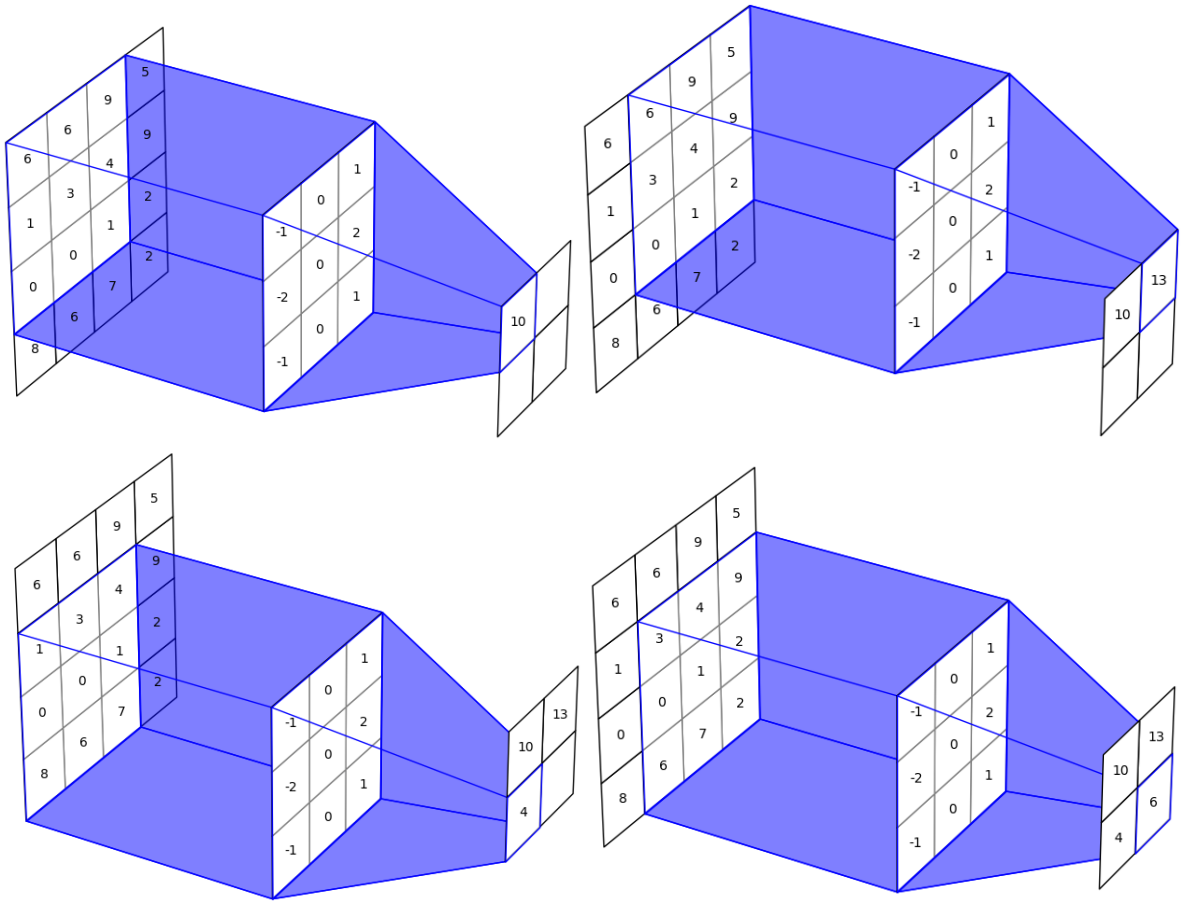


Figura 4: Visualización de una capa convolucional con un único filtro 3×3.

Si las dimensiones de los filtros son mayores a la unidad la convolución tiene un problema de definición en los bordes de la imagen. Este fenómeno queda bien ilustrado en **Figura 4**, donde puede verse que la convolución de un filtro 3×3 sobre una matriz de 4×4 termina produciendo una matriz 2×2 al no poder convolucionar los bordes. La solución más empleada es añadir un marco con ceros alrededor de la imagen para poder calcular las convoluciones de todos los pixeles, como se muestra en la **Figura 5**.

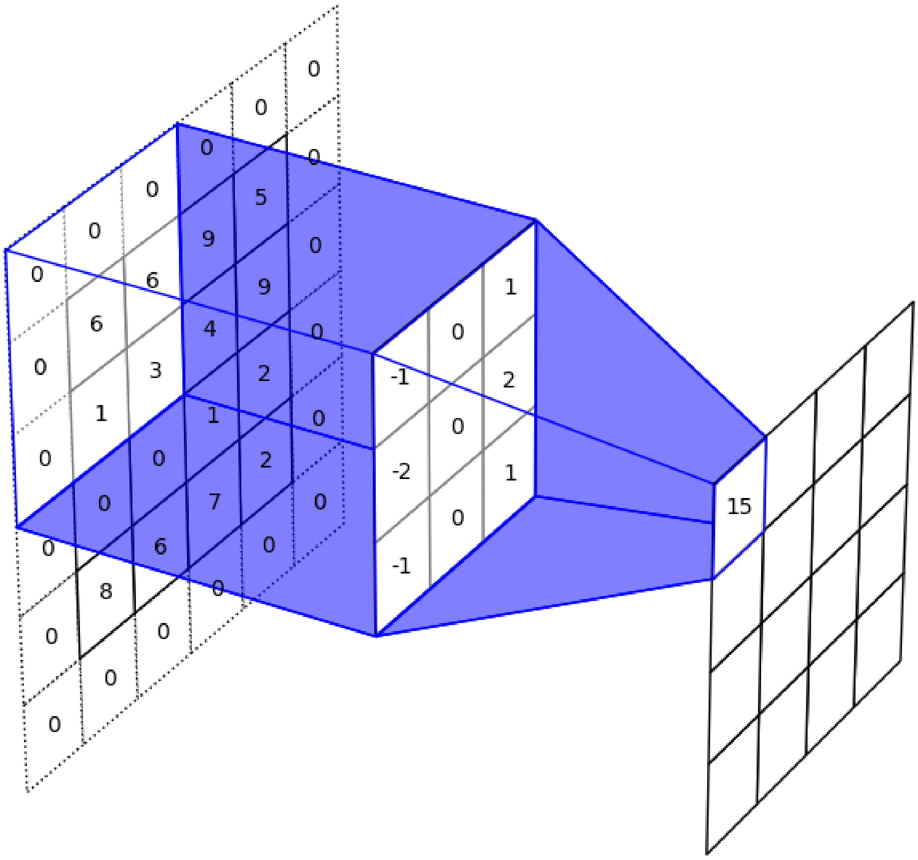


Figura 5: Visualización de una capa convolucional con un único filtro 3×3 y relleno de ceros.

De la ecuación (*4*) se sigue fácilmente que las redes neuronales convolucionales, en general, trabajan con volúmenes de datos. Usualmente estos volúmenes suelen ir disminuyendo en las dimensiones espaciales a la par que se hacen más profundos a medida que atraviesan las diferentes capas convolucionales que componen la red. De esta forma la CNN actúa como un extractor de características que permite la clasificación por las siguientes capas de la red. Para aumentar la profundidad del volumen de datos basta con añadir un número creciente de filtros a las capas que componen la red. Mientras que para reducir las dimensiones espaciales se utiliza el *stride*, el cual define un mecanismo para saltarse convoluciones, y por lo tanto reducir la dimensión del volumen de salida. Este mecanismo puede visualizarse en la **Figura 6**. Si se denotan s_x y s_y los valores de *stride* en las direcciones X e Y se puede modificar la ecuación (*4*) para incluir este efecto de la siguiente forma:

$$[\boldsymbol{K} \otimes \boldsymbol{x}]_{ij\mu} = \sum_{n,m,\nu} K_{ij\mu}\, a_{i+ns_x, j+ms_y, \nu} \qquad (5)$$

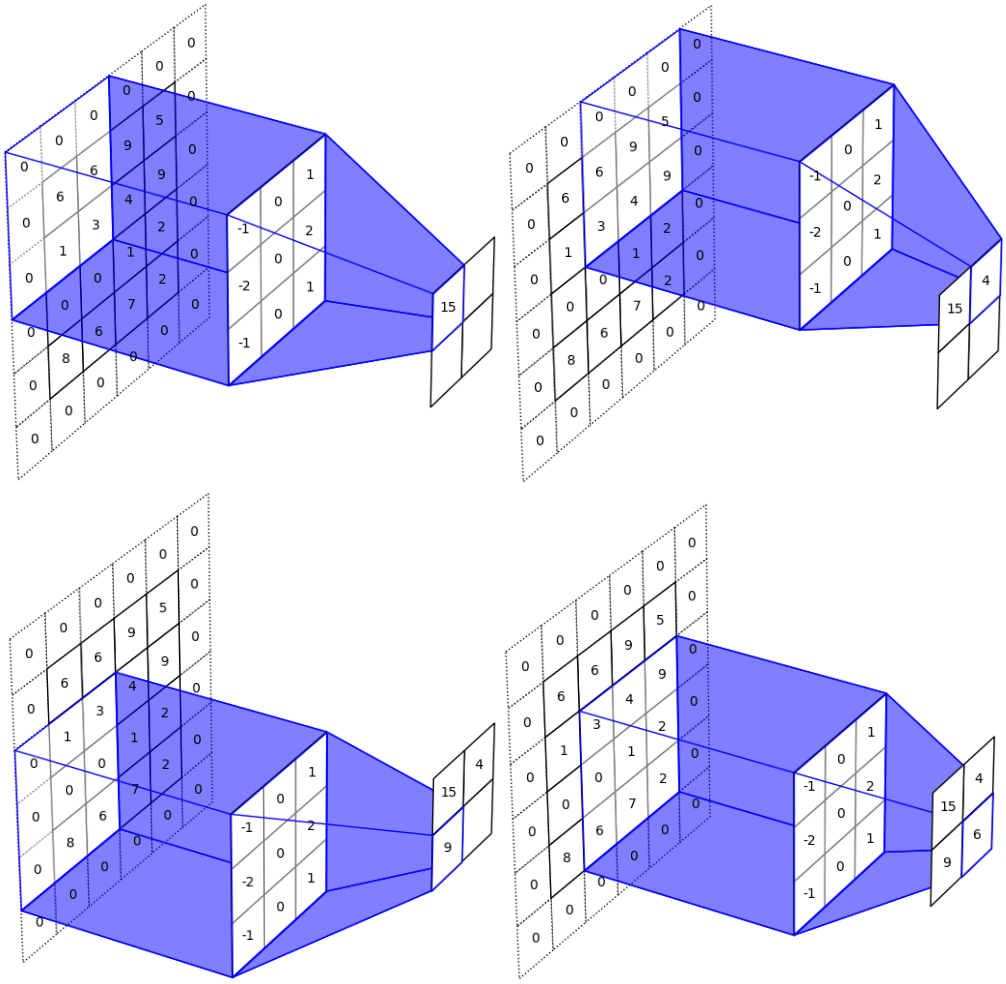


Figura 6: Visualización de una capa convolucional con un único filtro 3×3 y relleno de ceros y *stride* 2×2.

Aunque la operación de reducir la dimensionalidad puede realizarse por las mismas capas convolucionales, se suelen utilizar capas específicas para esta tarea, denominadas capas de *pooling*. Estas utilizan filtros identidad para no distorsionar los datos. Además, estas capas suelen utilizar una función de agregación sobre las dimensiones espaciales distinta a la suma, siendo la más utilizada el máximo, lo que se conoce como una capa *max pooling*.

La arquitectura convolucional típica se compone de dos modelos: el modelo base y el superior. El modelo base es la red convolucional propiamente dicha y se compone de varios bloques convolucionales (serie de capas convolucionales apiladas) separados por capas de *pooling*. El objetivo de este modelo es extraer características de interés para comprender la imagen. A continuación, esas características son analizadas por el modelo superior para resolver la tarea de entrenamiento. En el campo del reconocimiento de la escritura a mano los modelos superiores normalmente están compuestos por capas recurrentes o transformadores.

2.3. Red neuronal recurrente (RNN)

Las redes neuronales recurrentes (Williams et al., 1986) representan una clase poderosa de arquitecturas de aprendizaje profundo especialmente diseñadas para procesar secuencias de datos. En el contexto del reconocimiento de la escritura a mano, las RNNs ofrecen la ventaja única de poder considerar el contexto temporal o espacial de los trazos, lo que las hace particularmente útiles para el reconocimiento de escritura cursiva o para interpretar secuencias de caracteres.

La característica distintiva de las RNNs es su capacidad para mantener un estado interno, $\boldsymbol{h_t}$, que se actualiza a medida que se procesa cada elemento de la secuencia de entrada. Este estado interno actúa como una forma de «memoria» que permite a la red capturar dependencias a largo plazo en los datos. La salida de una capa recurrente se calcula en dos pasos. En el primero, se actualiza el estado interno en base a la nueva observación, $\boldsymbol{x_t}$, y al estado interno previo, $\boldsymbol{h_{t-1}}$. En el segundo, se calcula la salida, $\boldsymbol{y_t}$, utilizando únicamente el nuevo estado interno:

$$\boldsymbol{h_t} = f_h\left(\boldsymbol{W^{hh}h_{t-1}} + \boldsymbol{W^{hx}x_t} + \boldsymbol{b_h}\right) \quad (6)$$
$$\boldsymbol{y}_t = f_y\left(\boldsymbol{W^{hy}h_t} + \boldsymbol{b_y}\right)$$

Donde:

- $\boldsymbol{W^{hh}}$ es la matriz de pesos para las conexiones recurrentes (de estado a estado).
- $\boldsymbol{W^{hx}}$ es la matriz de pesos para las conexiones de entrada a estado oculto.
- $\boldsymbol{W^{hy}}$ es la matriz de pesos para las conexiones de estado oculto a salida.
- $\boldsymbol{b_h}$ y $\boldsymbol{b_y}$ son vectores de sesgo.
- $\boldsymbol{f_h}$ y $\boldsymbol{f_y}$ son funciones de activación.

La clave de esta formulación es que el estado oculto actual depende no solo de la entrada actual, sino también del estado anterior, lo que produce la recursión que da nombre a la arquitectura. Dicha recursión permite a la red mantener información a lo largo del tiempo. El fujo de información a través de una neurona recurrente se ilustra en la **Figura 7**.

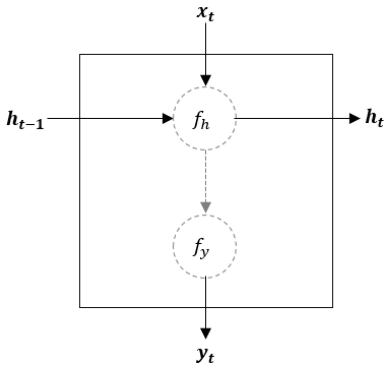


Figura 7: Flujo de información de una neurona recurrente.

2.3.1. Redes recurrentes multidimensionales

En el contexto del reconocimiento de textos manuscritos, las entradas de la red recurrente son las características extraídas por una red convolucional que analiza la imagen, las cuales se analizan secuencialmente en el eje de escritura (que en nuestra cultura es el horizontal). Aunque antes de que este tipo de arquitecturas se convirtieran en el estándar para el campo se investigó otro tipo de redes recurrentes, las multidimensionales.

Las redes recurrentes multidimensionales surgieron como una extensión natural de las redes recurrentes unidimensionales (Graves et al., 2007), con el objetivo de capturar dependencias espaciales en múltiples dimensiones simultáneamente. Esta característica las hace particularmente interesantes para el análisis de imágenes de texto manuscrito, donde la información relevante no solo se encuentra a lo largo del eje horizontal de escritura, sino también en las relaciones verticales entre líneas y en la estructura bidimensional general del texto.

En una MDRNN, la idea fundamental es extender el concepto de recurrencia a múltiples dimensiones. Mientras que en una RNN estándar, el estado oculto se propaga en una sola dirección (típicamente de izquierda a derecha en el caso del texto); en una MDRNN, el estado oculto se propaga en múltiples direcciones. Para una imagen de texto manuscrito, esto significa que la red puede procesar la información tanto horizontal como verticalmente, e incluso en direcciones diagonales. Para ello se añade un término $\boldsymbol{W^{hh}h_{t-1}}$ por cada eje. En el caso del procesamiento horizontal y vertical la ecuación (*6*) se modificaría de la siguiente forma:

$$\boldsymbol{h_{ij}} = f_h\left(\boldsymbol{W^{hh}h_{i,j-1}}+\boldsymbol{W^{hv}h_{i-1,j}} + \boldsymbol{W^{hx}x_{ij}} + \boldsymbol{b_h}\right) \quad (7)$$

Donde $\boldsymbol{W^{hh}}$ y $\boldsymbol{W^{hv}}$ son las matrices de pesos para las conexiones recurrentes en los ejes horizontal y vertical, respectivamente. En la **Figura 8** se ilustra el flujo de información a través de una neurona recurrente con dos ejes.

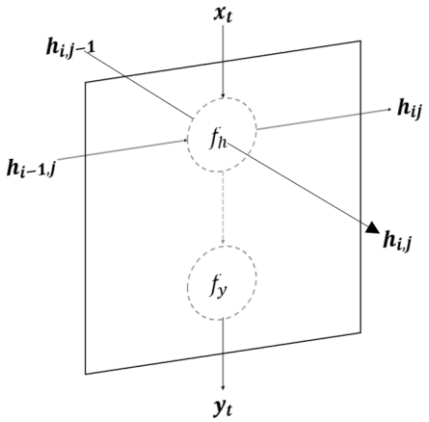


Figura 8: Flujo de información de una neurona recurrente bidimensional.

A pesar de su potencial teórico, las MDRNNs no se han convertido en el estándar en el campo del reconocimiento de textos manuscritos. Esto se debe en gran parte al éxito de las arquitecturas que combinan redes convolucionales para la extracción de características con RNN unidimensionales (típicamente LSTM bidireccionales) para el modelado secuencial. Estas arquitecturas híbridas CNN-RNN han demostrado ser más eficientes computacionalmente y más fáciles de entrenar, mientras que aún logran capturar efectivamente tanto las características locales como las dependencias secuenciales en el texto manuscrito.

2.3.2. Memoria corto-largo plazo (LSTM)

Las RNNs básicas sufren del problema de desvanecimiento o explosión del gradiente durante el entrenamiento, especialmente cuando se trata de capturar dependencias a largo plazo. Para abordar este problema, se han desarrollado arquitecturas más avanzadas como las LSTM (Hochreiter & Schmidhuber, 1997) y las *Gated Recurrent Units* (Cho et al., 2014).

Las LSTM, en particular, han demostrado ser muy efectivas en tareas de procesamiento de secuencias. La arquitectura LSTM introduce un mecanismo de compuerta que permite a la red aprender selectivamente qué información almacenar, olvidar y mostrar. Para ello construye un segundo estado oculto, el estado de la celda, $\boldsymbol{c_t}$. Este se actualizará procesando el estado oculto recurrente anterior, junto con la entrada actual y el estado de celda previo. El procesamiento se realiza mediante tres compuertas: la compuerta de olvido que produce una máscara, $\boldsymbol{f_t}$, que filtra la parte del estado de celda anterior que se olvida; la compuerta de entrada que produce candidatos a incorporar al estado de celda, $\bar{\boldsymbol{c}}_{\boldsymbol{t}}$, junto con sus pesos, $\boldsymbol{i_t}$, y la celda de salida que produce una máscara, $\boldsymbol{o_t}$, que determina qué parte del nuevo estado de celda se expresa en el estado oculto. Para calcular estas máscaras se emplea la misma formulación recurrente que se ha visto anteriormente:

$$\begin{aligned} \boldsymbol{f_t} &= \sigma\left(\boldsymbol{W^{fh}h_{t-1}} + \boldsymbol{W^{fx}x_t} + \boldsymbol{b_f}\right) \\ \boldsymbol{i_t} &= \sigma\left(\boldsymbol{W^{ih}h_{t-1}} + \boldsymbol{W^{ix}x_t} + \boldsymbol{b_i}\right) \\ \boldsymbol{o_t} &= \sigma\left(\boldsymbol{W^{oh}h_{t-1}} + \boldsymbol{W^{ox}x_t} + \boldsymbol{b_o}\right) \\ \bar{\boldsymbol{c}}_{\boldsymbol{t}} &= tanh\left(\boldsymbol{W^{ch}h_{t-1}} + \boldsymbol{W^{cx}x_t} + \boldsymbol{b_c}\right) \\ \boldsymbol{c_t} &= \boldsymbol{f_t} \circ \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \circ \bar{\boldsymbol{c}}_{\boldsymbol{t}} \\ \boldsymbol{h_t} &= \boldsymbol{o_t} \circ tanh(\boldsymbol{c_t}) \end{aligned} \quad (8)$$

Donde:

- σ denota la función sigmoide:

$$f(\boldsymbol{z}) = \frac{1}{1 + e^{\boldsymbol{z}}} \quad (9)$$

- ∘ denota la multiplicación elemento a elemento (producto Hadamard):

$$[\boldsymbol{A} \circ \boldsymbol{B}]_{ij} = A_{ij}B_{ij} \quad (10)$$

En la **Figura 9** se ilustra el flujo de información a través de una celda LSTM.

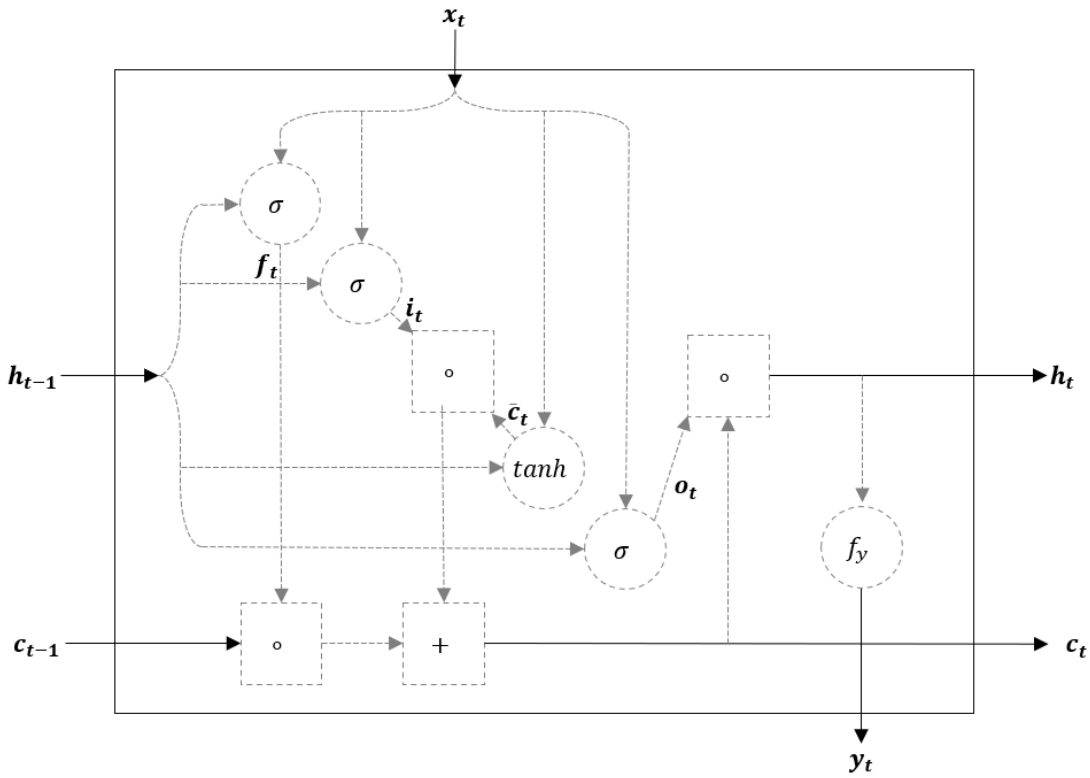


Figura 9: Flujo de información de una celda LSTM. Los círculos representan la ponderación mediante matriz de pesos y sesgo, seguidos de la función de activación indicada. Los cuadrados representan únicamente las operaciones indicadas.

2.4. Transformadores

Los transformadores, introducidos por Vaswani et al. (2017), representan un cambio de paradigma en el procesamiento de secuencias, incluyendo el reconocimiento de escritura a mano. A diferencia de las arquitecturas recurrentes, los transformadores procesan la secuencia completa en paralelo, lo que permite un entrenamiento más eficiente y la captura de dependencias a largo plazo de manera más efectiva.

El modelo transformador se basa en el mecanismo de atención, que permite al modelo enfocarse en diferentes partes de la entrada al generar cada elemento de la salida. La atención se calcula como:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V} \quad (11)$$

Donde $\boldsymbol{Q}$ (*query*), $\boldsymbol{K}$ (*key*) y $\boldsymbol{V}$ (*value*) son matrices derivadas de la entrada, y $\sqrt{d_k}$ es la dimensión de las claves.

La arquitectura completa del transformador consta de múltiples bloques formados por una capa de atención de múltiples cabezas seguida de una capa prealimentada. Cada capa de atención de múltiples cabezas se puede expresar como:

$$MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(\boldsymbol{head_1}, \ldots, \boldsymbol{head_h})\boldsymbol{W^O} \quad (12)$$

Donde cada cabeza se calcula como:

$$\boldsymbol{head_i} = Attention\left(\boldsymbol{QW_i^Q}, \boldsymbol{KW_i^K}, \boldsymbol{VW_i^V}\right) \quad (13)$$

Y $\boldsymbol{W^O}$, $\boldsymbol{W_i^Q}$, $\boldsymbol{W_i^K}$, $\boldsymbol{W_i^V}$ son las matrices de pesos de la salida, las consultas, las claves y los valores de la cabeza i-ésima, respectivamente.

La capa prealimentada realiza dos transformaciones lineales separadas por una activación ReLU:

$$FFN(\boldsymbol{x}) = \max(0, \boldsymbol{xW_1} + \boldsymbol{b_1})\boldsymbol{W_2} + \boldsymbol{b_2} \quad (14)$$

Un componente crucial de la arquitectura transformador, especialmente relevante en el contexto de HWR, son los encajes posicionales. Dado que los transformadores procesan la entrada en paralelo, necesitan una forma de incorporar información sobre la posición relativa de cada elemento en la secuencia. Esto se logra mediante los encajes posicionales, que se suman a los encajes de entrada antes de pasarlos a las capas de atención. Una forma elegante de implementarlos es utilizando funciones sinusoidales de diferente frecuencia:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \quad (15)$$

Donde pos es la posición en la secuencia, i es el índice del vector de encajes y d_{model} es la dimensión del modelo. El efecto de este encaje puede visualizarse mediante la **Figura 10**:

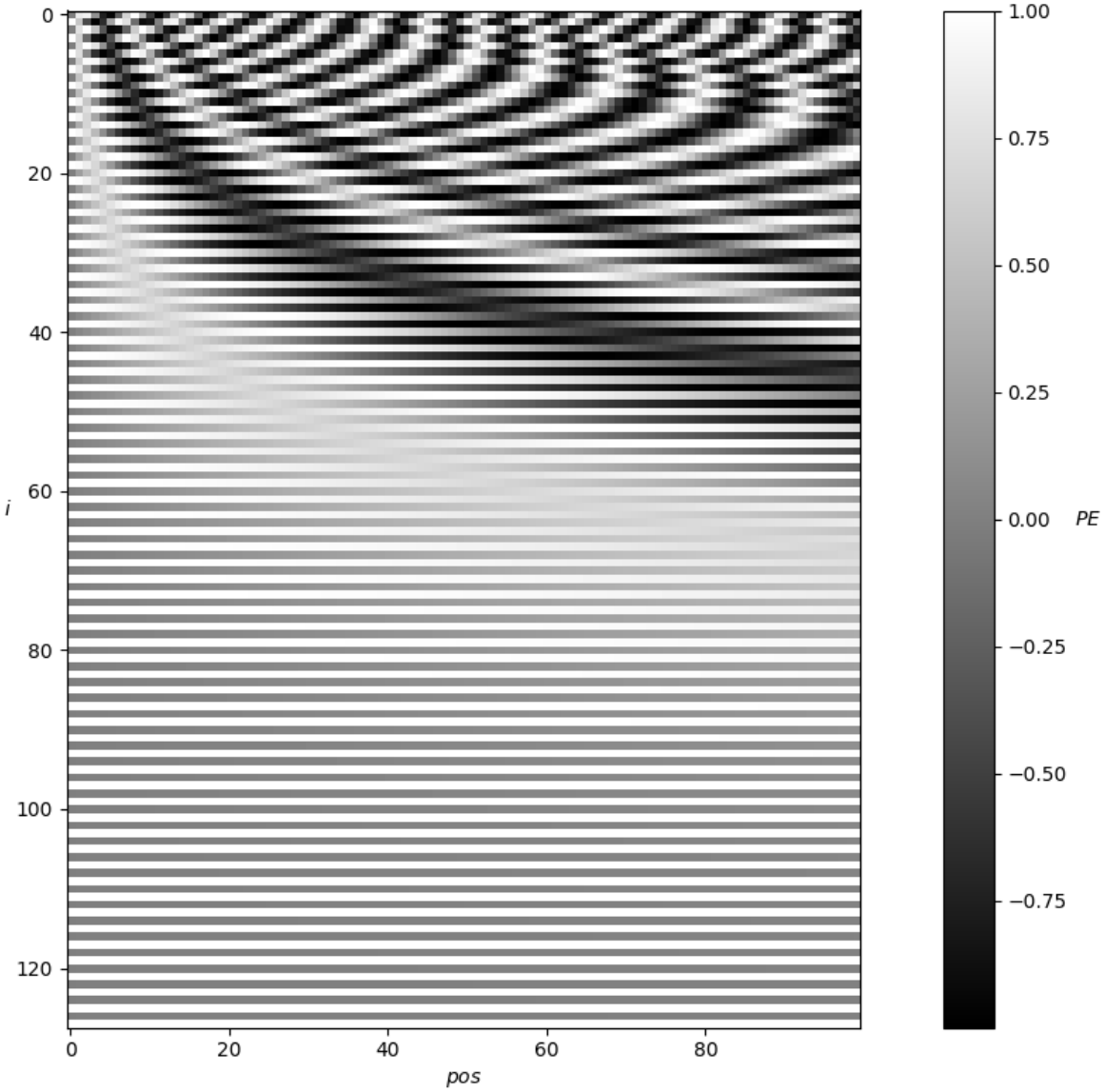


Figura 10: Visualización de los valores del encaje posicional para $d_{model} = 128$ y una longitud máxima de la secuencia de 100.

La aplicación de transformadores en HWR ha demostrado ser prometedora, como evidencian los trabajos de Diaz et al. (2021) y Fujitake (2024), que lograron una CER de 2,75% y 2,38%, respectivamente, sobre el conjunto de datos IAM utilizando una arquitectura basada en transformadores. Sin embargo, es importante notar que los transformadores, especialmente en sus variantes más grandes, pueden requerir conjuntos de datos significativamente más grandes y recursos computacionales más intensivos para su entrenamiento efectivo en comparación con las arquitecturas CNN-RNN tradicionales.

2.4.1. Gran modelo del lenguaje (LLM)

Los Grandes Modelos de Lenguaje (LLM, por sus siglas en inglés) representan la culminación de los avances en el procesamiento del lenguaje natural y han comenzado a mostrar un potencial significativo en el campo del reconocimiento de escritura a mano. Estos modelos, basados en arquitecturas transformador de gran escala, se entrenan en vastos corpus de texto para adquirir un conocimiento profundo del lenguaje y la capacidad de generar texto coherente y contextualmente apropiado. Una vez adquirido este conocimiento pueden reentrenarse para una tarea específica, como es el caso del reconocimiento del texto manuscrito (Fujitake, 2024; Li et al., 2021).

Los LLMs se caracterizan por su proceso de entrenamiento en dos fases: un preentrenamiento en un corpus de datos masivo y diverso, seguido de un ajuste fino para tareas específicas. Durante el preentrenamiento, estos modelos son expuestos a vastas cantidades de texto no etiquetado, lo que les permite aprender representaciones lingüísticas ricas y generalizables. Este proceso se realiza típicamente mediante tareas de modelado de lenguaje, donde el modelo aprende a predecir la siguiente palabra (Radford et al., 2018) en una secuencia o a reconstruir partes enmascaradas de un texto (Devlin et al., 2018).

Una tendencia notable en el desarrollo de los LLMs es el aumento constante en el número de parámetros, impulsado por la observación de que modelos más grandes exhiben comportamientos más inteligentes e incluso habilidades emergentes. Estas son capacidades que no están presentes en modelos más pequeños y que surgen de manera aparentemente espontánea al aumentar la escala. Este fenómeno es posible gracias a la paralelización de los transformadores, que permiten entrenar los modelos sobre grandes volúmenes de datos de forma eficiente.

Otro desarrollo significativo en el campo de los LLMs es la tendencia hacia la multimodalidad. Modelos como GATO (Reed et al., 2022), introducido por DeepMind, demuestran la capacidad de los transformadores para manejar múltiples tipos de datos de entrada, incluyendo texto, imágenes o datos de control robótico. Esta capacidad multimodal tiene implicaciones potencialmente revolucionarias para el campo del HWR, ya que permite la integración de información visual y textual en el proceso de reconocimiento.

2.5. Métodos de optimización

En el campo del aprendizaje automático tan importante como el modelo para representar los datos es el algoritmo que permite al modelo optimizar sus parámetros para ajustarse a ellos. En esta sección, se explorarán los diferentes métodos que se han empleado en el campo del reconocimiento de la escritura a mano para optimizar los modelos.

2.5.1. Entropía cruzada (CS)

La entropía cruzada es una función de pérdida ampliamente utilizada en problemas de clasificación, incluyendo el reconocimiento de escritura a mano. Esta función mide la diferencia entre la distribución de probabilidad predicha por el modelo, q, y la distribución de probabilidad real de los datos, p, sobre un conjunto de clases, $C = \{1,2, \dots, n\}$:

$$H(p,q) = -\sum_{c \in C} p(c) \log q(c) \quad (16)$$

En el campo del HWR la entropía cruzada se ha empleado principalmente para entrenar modelos ocultos de Márkov y fue abandonada en favor de la CTC cuando los modelos basados en redes neuronales profundas ganaron tracción.

2.5.2. Clasificación temporal conexionista (CTC)

La clasificación temporal conexionista es una función de pérdida introducida por Grave et al. (2006) para abordar el problema de alinear secuencias de longitud variable en tareas de reconocimiento de patrones secuenciales, como el reconocimiento de escritura a mano. La CTC es particularmente útil en escenarios donde la segmentación precisa de los datos de entrada no está disponible o es difícil de obtener.

En el contexto del HWR, la CTC permite que los modelos aprendan a reconocer secuencias de caracteres sin necesidad de una segmentación explícita de la imagen de entrada en caracteres individuales. Esto es especialmente valioso para el reconocimiento de escritura ligada, donde los límites entre caracteres pueden ser ambiguos.

La idea central detrás de la CTC es permitir que el modelo produzca una distribución de probabilidad sobre todas las posibles secuencias de etiquetas para cada paso de tiempo en la secuencia de entrada. Para lograr esto, la CTC introduce un símbolo especial, comúnmente denotado como «*blank*» o «-», que representa la ausencia de un carácter en un paso de tiempo dado.

Matemáticamente, la función de pérdida CTC se define como el negativo del logaritmo de la probabilidad de la etiqueta correcta, sumado sobre todas las posibles alineaciones:

$$L_{CTC} = -\sum_{(\boldsymbol{l},\boldsymbol{x}) \in S} log \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l})} p(\boldsymbol{\pi}|\boldsymbol{x}) \quad (17)$$

Donde:

- $\boldsymbol{x}$ es la secuencia de entrada.
- $\boldsymbol{l}$ es la secuencia de etiquetas objetivo.
- S es el conjunto de pares de secuencias de entrada y objetivo.
- $\boldsymbol{\pi}$ es una posible alineación.
- $\mathcal{B}$ es una función que aplica alineaciones a etiquetas eliminando repeticiones y símbolos de cambio de carácter.
- $p(\boldsymbol{\pi}|\boldsymbol{x})$ es la probabilidad de una alineación dada una secuencia de entrada.

El cálculo eficiente de esta suma se realiza mediante programación dinámica, utilizando el algoritmo avance-retroceso.

Durante el entrenamiento, la CTC permite que el modelo aprenda a alinear automáticamente las entradas con las etiquetas, sin necesidad de alineaciones explícitas. En la fase de inferencia, la decodificación puede realizarse de varias maneras:

- Decodificación voraz: Se selecciona el carácter más probable en cada paso de tiempo y luego se aplica la función $\mathcal{B}$ para obtener la secuencia final.

- Búsqueda en haz: Se mantienen múltiples hipótesis de secuencias y se selecciona la más probable al final.

- Decodificación basada en diccionario: Se incorpora conocimiento lingüístico para favorecer secuencias de palabras válidas.

En la **Figura 11** se muestra un ejemplo de decodificación voraz. En primer lugar, la red neuronal profunda analiza la imagen y extrae las probabilidades de que cada carácter se corresponda con cada uno de los posibles valores del vocabulario. Nótese que el número de caracteres predichos en esta fase está dado por la arquitectura (en este caso 256) y no por la imagen (que en este caso contiene 55 caracteres). En segundo lugar, se selecciona vorazmente los caracteres cuya probabilidad sea mayor. Finalmente, se reconstruye la secuencia mediante la función $\mathcal{B}$, la cual elimina las repeticiones consecutivas y el carácter «*blank*». La visualización incluye un zoom que muestra cómo se trascribe «n't d» eliminando el apostrofe y el espacio repetido y otro que muestra cómo se transcriben las dos «e» de «*been*» al estar separadas por el carácter «*blank*».

La CTC ha demostrado ser especialmente efectiva cuando se combina con arquitecturas de redes neuronales recurrentes, como las LSTM bidireccionales, o con modelos basados en transformadores. En el campo del HWR, la CTC ha contribuido significativamente a mejorar el rendimiento de los sistemas de reconocimiento, que permiten el entrenamiento de modelos de extremo a extremo que pueden manejar directamente imágenes de texto manuscrito sin un preprocesamiento extensivo.

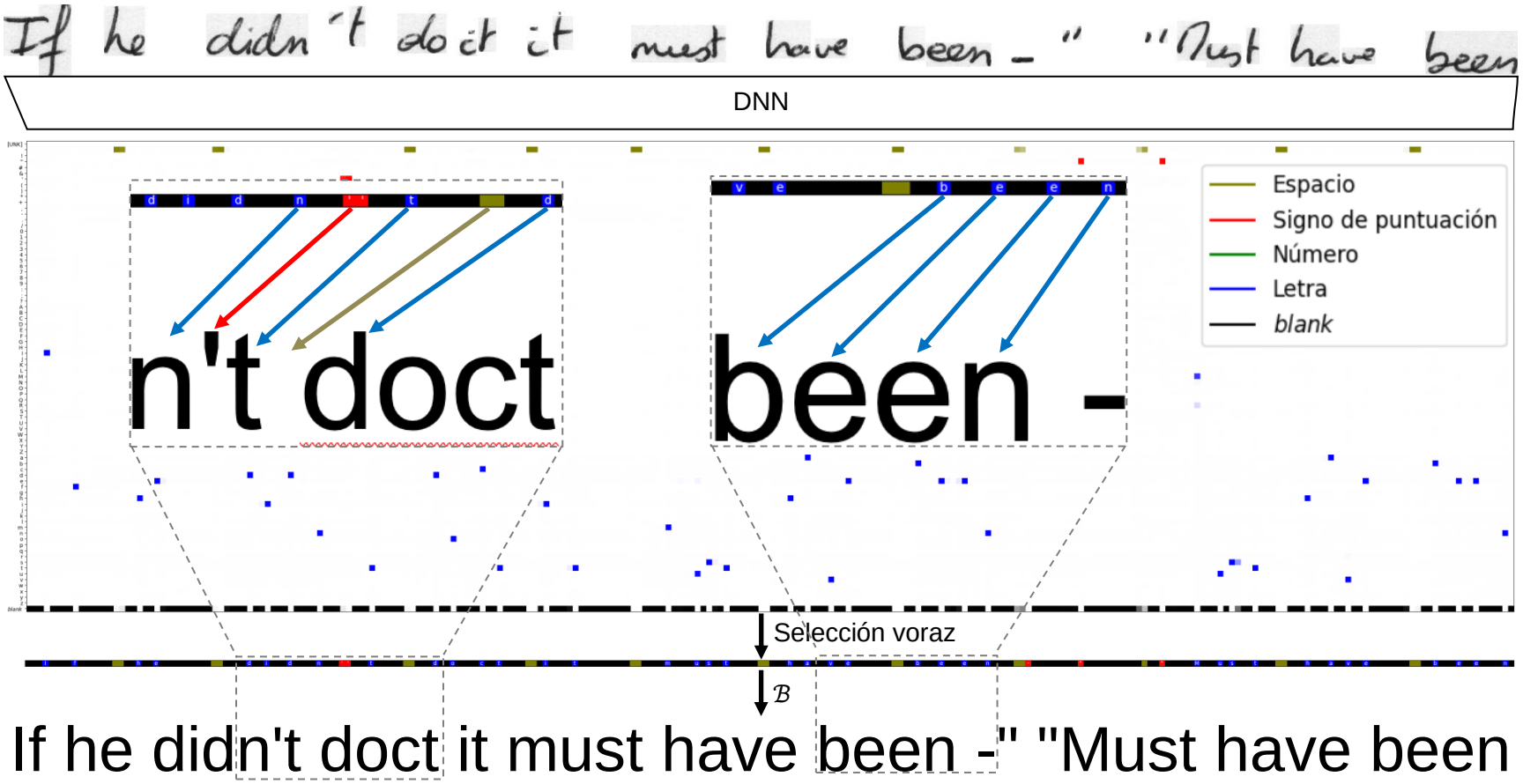


Figura 11: Visualización de la decodificación de una secuencia mediante CTC voraz.

Sin embargo, es importante señalar que la CTC también tiene algunas limitaciones. Por ejemplo, asume independencia condicional entre las predicciones en diferentes pasos de tiempo, lo que puede no ser siempre una suposición válida en el contexto del lenguaje natural. Además, la CTC puede tener dificultades con secuencias muy largas debido a problemas numéricos en el cálculo de las probabilidades.

A pesar de estas limitaciones, la CTC sigue siendo una herramienta fundamental en el campo del HWR y ha sido utilizada en muchos de los sistemas de reconocimiento de escritura a mano más avanzados hasta la fecha.

2.5.3. Seq2Seq

Los modelos *Sequence-to-Sequence* (Sutskever et al., 2014) representan un enfoque de aprendizaje profundo que ha ganado popularidad en tareas de procesamiento de secuencias, incluido el HWR. Estos modelos consisten típicamente en un codificador que procesa la secuencia de entrada y un decodificador que genera la secuencia de salida.

La optimización de modelos Seq2Seq para HWR a menudo implica el uso de mecanismos de atención para permitir que el modelo se enfoque en partes relevantes de la imagen de entrada durante la decodificación. Así como la implementación de estrategias de búsqueda en haz durante la inferencia para mejorar la calidad de las transcripciones generadas.

En el contexto del HWR, el codificador suele ser una red convolucional que procesa la imagen de entrada, mientras que el decodificador sería una red recurrente o, más comúnmente, un transformador que genera la transcripción carácter por carácter.

2.6. Regularización de redes neuronales

La regularización es una técnica crucial en el entrenamiento de redes neuronales profundas, especialmente en tareas complejas como el reconocimiento de escritura a mano. Su objetivo principal es prevenir el sobreajuste, permitiendo que el modelo generalice mejor a datos no vistos. En el contexto del HWR, donde la variabilidad en los estilos de escritura es significativa, las técnicas de regularización son particularmente importantes para desarrollar modelos robustos. A continuación, se describen dos técnicas de regularización ampliamente utilizadas: el *dropout* y normalización de lotes.

2.6.1. Dropout

La técnica de *dropout* (Hinton et al., 2012) es una evolución del abandono (Canning & Gardner, 1988; Sompolinsky, 1987) y supone una técnica de regularización simple pero efectiva que reduce el sobreajuste en redes neuronales. La idea fundamental detrás del abandono es «apagar» aleatoriamente un cierto porcentaje de pesos cada paso de entrenamiento. Es decir, en lugar de trabajar con los pesos $\boldsymbol{W}$ la red se entrena con los pesos diluidos:

$$\widehat{W}_{ij} = \begin{cases} W_{ij}, & con\ probabilidad\ c \\ 0, & con\ probabilidad\ 1-c \end{cases} \quad (18)$$

El *dropout* es un caso particular de esta técnica cuando en lugar de eliminar pesos individuales se eliminan filas completas de la matriz de pesos:

$$\widehat{\boldsymbol{W}}_i = \begin{cases} \boldsymbol{W}_i, & con\ probabilidad\ c \\ 0, & con\ probabilidad\ 1-c \end{cases} \quad (19)$$

La ventaja de eliminar una fila completa es que es equivalente a «apagar» una neurona entera, lo cual permite realizar las operaciones del abandono de forma mucho más eficiente. Y por lo tanto acelerar el entrenamiento. La **Figura 12** ilustra el efecto de ambas técnicas. En la fase de inferencia, todas las neuronas se mantienen activas, pero sus salidas se multiplican por c para mantener la misma magnitud esperada que durante el entrenamiento.

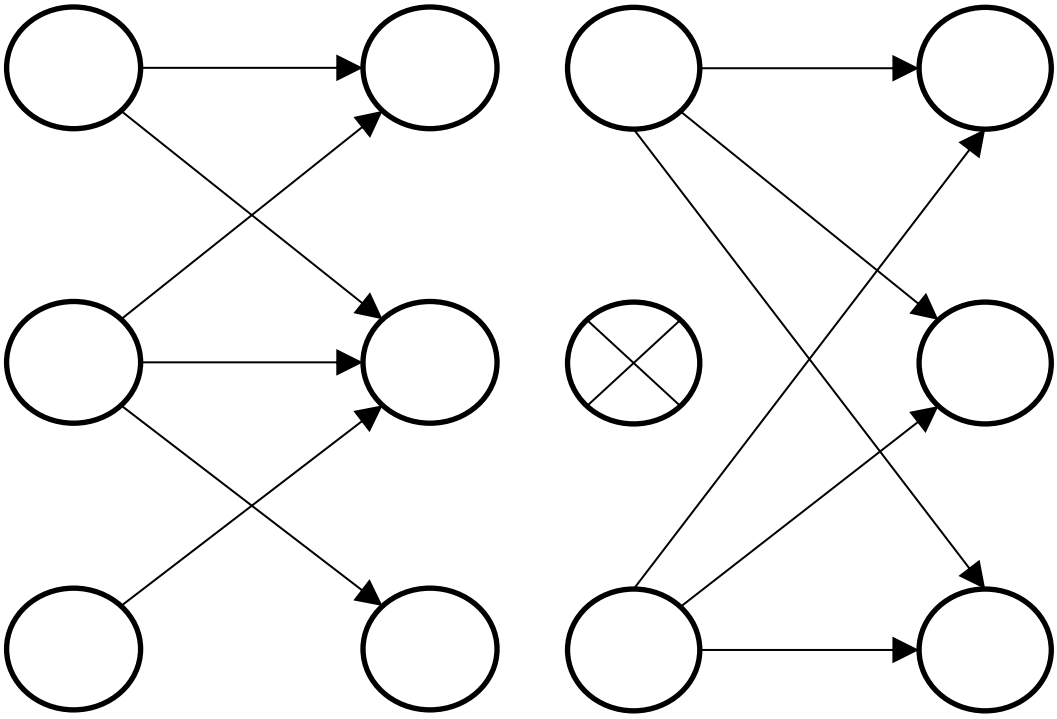


Figura 12: Visualización del abandono (izquierda) y *dropout* (derecha).

2.6.2. Normalización de lotes

La normalización de lotes (Ioffe & Szegedy, 2015) es una técnica que aborda el problema de la varianza interna de las capas. Este fenómeno ocurre cuando la distribución de las entradas a una capa cambia durante el entrenamiento, lo que puede ralentizar significativamente el proceso de aprendizaje.

La idea central de la normalización de lotes es normalizar las entradas de cada capa para que tengan media cero y varianza unitaria:

$$\boldsymbol{y} = \gamma \frac{\boldsymbol{x} - \boldsymbol{\mu}_B}{\sqrt{\boldsymbol{\sigma}_B^2 + \varepsilon}} + \beta \quad (20)$$

Donde:

- $\boldsymbol{\mu}_B$ y $\boldsymbol{\sigma}_B$ son la media y desviación típica del lote, $\boldsymbol{B} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, de tamaño m.
- ϵ es un pequeño valor para evitar divisiones por cero.
- γ y β son parámetros aprendibles que permiten al modelo reescalar y desplazar la distribución normalizada si es necesario.

2.7. Métricas de evaluación

El reconocimiento del lenguaje no puede evaluarse utilizando métricas clásicas de clasificación debido al número infinito de palabras que pueden generarse como combinaciones de caracteres. Por ello, en este campo se les da la vuelta a las métricas y se evalúa en las tasas de fallo, en lugar de las de éxito. Los errores en la transcripción de textos pueden ser de tres tipos:

- Por sustitución, S, cuando un elemento (ya sea un carácter o una palabra) se ha identificado incorrectamente.
- Por omisión, D, cuando un elemento ha sido eliminado completamente de la transcripción.
- Por inserción, I, cuando un elemento extra es añadido a la transcripción.

Para tener en cuenta estos tres factores se emplean las métricas de ratios de error, en las que se divide la suma de errores producidos por los tres factores y se normaliza frente al número total de elementos. En particular se puede expresar las ratios de error por palabra (WER) y por carácter (CER) como

$$WER = \frac{S + D + I}{N_W} \quad (21)$$

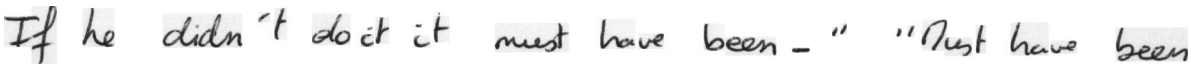
$$CER = \frac{S + D + I}{N_C} \quad (22)$$

Donde N_W y N_C son el número total de palabras o carácter, respectivamente. Si se denota el número de elementos correctamente transcritos como C se puede expresar el número total de elementos de la siguiente forma:

$$N = S + D + C \quad (23)$$

De donde se sigue que las ratios de error por palabra y carácter pueden ser mayor que la unidad si el número de inserciones supera al de elementos correctos.

Para mejor comprensión de ambas métricas se puede consultar el ejemplo de la **Figura 13**. En ese ejemplo se ha transcrito incorrectamente «doct» en lugar de «do it». Este error se descompone en un carácter eliminado, « » y dos incorrectamente transcritos, «it», en el caso del CER o en una palabra eliminada, «it», y otra incorrectamente transcrita, «do», en el caso del WER.



If he didn't doct it must have been -" "Must have been

$$CER = \frac{2 + 1 + 0}{55} = 5{,}4\%$$

If he didn't doct it must have been -" "Must have been

$$WER = \frac{1 + 1 + 0}{13} = 15\%$$

Figura 13: Ejemplo del cálculo de las métricas CER y WER. Se ha marcado en verde los caracteres y palabras correctamente identificados y en rojo los fallos.

Tanto la WER como la CER son métricas complementarias que ofrecen perspectivas diferentes sobre el rendimiento de los sistemas de reconocimiento de escritura a mano. Mientras que la WER proporciona una visión general de la precisión a nivel de palabra, la CER ofrece una evaluación más detallada a nivel de carácter. Debido a ello muchos artículos presentan ambas para evaluar sus resultados.

La factorización de los distintos tipos de error en un único número produce que las ratios de error no sean del todo intuitivas de interpretar. Para suplir este inconveniente se utilizará una tercera métrica, la precisión por líneas:

$$Precisión = \frac{N_{L_{corr}}}{N_L} \quad (24)$$

Donde $N_{L_{corr}}$ representa el número de líneas correctamente transcritas y N_L el número total de líneas. Esta métrica permite saber qué fracción de líneas se transcriben correctamente, aunque no da información sobre cómo de mal se han transcrito las que no.

3. Material

En este capítulo se describirá tanto el *hardware* sobre el que se han ejecutado los experimentos como el conjunto de datos con el que se ha entrenado.

3.1. *Hardware* y *software*

Todos los entrenamientos se han realizado utilizando la CPU de un portátil de uso personal, concretamente una AMD Ryzen 7 5800H la cual consta de 8 núcleos a 3,2 GHz.

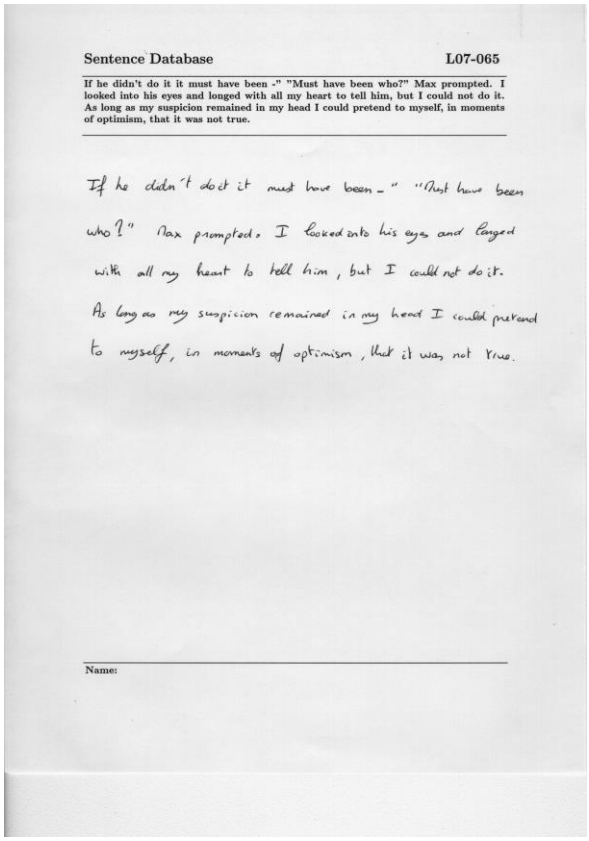
Los modelos fueron implementados en Keras versión 3.3.3 y Tensorflow versión 2.17.0. El código empleado puede consultarse en el **Apéndice II**.

3.2. Conjunto de datos IAM

Para este trabajo se ha utilizado el conjunto de datos IAM (Marti & Bunke, 2002), que contiene muestras de texto manuscrito en inglés. Esta base de datos se fundamenta en el corpus Lancaster-Oslo/Bergen (LOB), una colección de 500 textos en inglés de aproximadamente 2.000 palabras cada uno.

Los textos del corpus LOB se dividieron en fragmentos de tres a seis oraciones, con un mínimo de 50 palabras cada uno. Estos fragmentos se imprimieron en formularios que luego fueron completados a mano por los participantes. En la **Figura 14** puede verse un ejemplo de formulario. Se pidió a los escritores que utilizaran su escritura cotidiana para obtener muestras lo más naturales y sin restricciones posible. No se impusieron limitaciones en cuanto al instrumento de escritura, por lo que el conjunto de datos incluye texto producido con bolígrafos, plumas y lápices de diversos grosores de trazo.

El texto manuscrito de los formularios fue segmentado por líneas y palabras automáticamente. Debido a ello en algunos casos la segmentación no es correcta. Los autores del conjunto de datos (Marti & Bunke, 2002) dejan a discreción de los usuarios como la gestión de las líneas y palabras incorrectamente segmentadas. Puesto que la fracción de errores de segmentación es pequeña, en este estudio se emplearán únicamente las líneas correctamente segmentadas para realizar los entrenamientos. La distribución de tamaños de las líneas segmentadas puede encontrarse en la **Figura 15**. La distribución se centra en 1799×116 aunque tiene una cola larga hacia anchos menores y altos mayores. La variabilidad vertical es debido al diferente tamaño de la caligrafía de los autores. La variabilidad horizontal es menor debido a que las líneas se suelen adaptar al tamaño de la página. No obstante, debido a que algunos párrafos terminan antes de que la línea final se complete, existe un gran rango de valores para el tamaño horizontal.



Sentence Database L07-065

If he didn't do it it must have been -" "Must have been who?" Max prompted. I looked into his eyes and longed with all my heart to tell him, but I could not do it. As long as my suspicion remained in my head I could pretend to myself, in moments of optimism, that it was not true.

If he didn't do it must have been -" "Must have been who?" Max prompted, I looked into his eyes and longed with all my heart to tell him, but I could not do it. As long as my suspicion remained in my head I could pretend to myself, in moments of optimism, that it was not true.

Name:

Figura 14: Ejemplo de un formulario del conjunto de datos IAM.

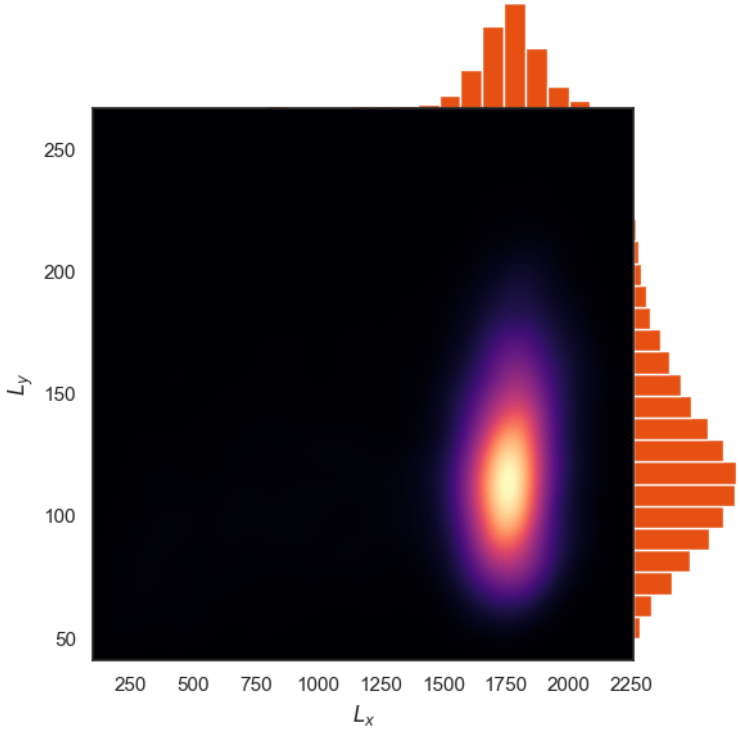


Figura 15: Distribución de tamaños de las segmentaciones por línea.

En total, el conjunto de datos IAM consta de 1.539 formularios producidos por 657 escritores diferentes. Estos formularios contienen un total de 13.353 líneas de texto manuscrito, de las cuales 11.344 están correctamente segmentadas. El conjunto de datos incluye 115.320 (96.456 correctamente segmentadas) instancias de palabras provenientes de un vocabulario de 13.542 (12.215 correctamente segmentadas) palabras únicas.

La base de datos proporciona etiquetas para cada línea de texto y palabra, que se generaron automáticamente a partir de los textos fuente y se corrigieron manualmente para adaptarse a los errores tipográficos producidos por los escritores. Originalmente las etiquetas omiten los espacios y separan las palabras mediante el símbolo «|». Por ejemplo, la primera línea de la **Figura 14** presenta la siguiente etiqueta: «If|he|didn't|do|it|it|must|have| been|-|"|"|Must|have|been». Nótese que en el conjunto de datos IAM considera los signos de puntuación como palabras independientes, por lo que el separador no siempre se corresponde con un espacio en la imagen. Para tener esto en cuenta se preprocesaron las etiquetas para incluir los espacios en las posiciones correctas. Es decir, para la misma línea se emplea la etiqueta «If he didn't do it it must have been -" "Must have been» . La lista completa de etiquetas puede encontrarse en el **Apéndice II**.

El conjunto de datos también define una tarea, la *Large Writer Independent Text Line Recognition Task,* que consiste en 9.862 líneas de texto distribuidas en una partición de entrenamiento, dos de validación y una de prueba según la **Tabla 1**. Los formularios han sido distribuidos de tal forma que cada autor contribuya a una única partición. De esta forma se evitan fugas de información entre las particiones.

Partición	***Número de líneas***	***Número de autores***
Entrenamiento	6.161	283
Validación 1	900	46
Validación 2	940	43
Prueba	1.861	128
Total	9.862	500

Tabla 1: Distribución de líneas y escritores de la *Large Writer Independent Text Line Recognition Task*

Nótese que no todos los escritores están incluidos en la tarea, de hecho, incluso algunas líneas de formularios incluidos tampoco están presentes en la tarea. Para no desaprovechar esos datos se han incluido todas las líneas de formularios no presentes en las particiones de validación y prueba dentro de la partición de entrenamiento. Para ello previamente se comprobaron que los autores de dichas líneas no estuvieran incluidos previamente en las otras particiones. Además, se utilizará una única partición de validación unificando las dos propuestas. En la **Tabla 2** puede verse la distribución de las particiones que se emplearán en este trabajo, una vez eliminadas las incorrectamente segmentadas.

Partición	*Número de líneas*	*Número de autores*
Entrenamiento	8.166	440
Validación 1	1.539	89
Prueba	1.480	128
Total	11.185	657

Tabla 2: Distribución de líneas y escritores empleadas en este trabajo.

Esta base de datos es particularmente útil para tareas de reconocimiento de escritura manuscrita que utilizan conocimientos lingüísticos más allá del nivel del léxico, ya que se puede extraer automáticamente información lingüística adicional del corpus LOB subyacente.

4. Métodos

En esta sección se explorará en profundidad las arquitecturas con las que se ha enfrentado la tarea del reconocimiento de caracteres manuscritos, así como los tratamientos realizados a las imágenes y etiquetas previos al entrenamiento. Para cada arquitectura se mostrará la red de la cual se parte y se expondrán todas las posibles modificaciones que se han probado.

4.1. Preprocesamiento

El procedimiento que se va a realizar para analizar las imágenes se basa principalmente en extraer características espaciales que permitan identificar los caracteres y procesarlas a lo largo del eje de escritura para recuperar el texto completo. Por ello, interesa que el primer índice de la representación matricial de las imágenes se corresponda con el eje horizontal y el segundo con el vertical. Sin embargo, la representación matricial original de las imágenes tiene los índices invertidos. La forma más sencilla de adaptar las imágenes originales es trasponiéndolas.

Como puede verse en la **Figura 15**, las imágenes tienen una gran variabilidad de tamaños, lo cual supone un problema si se pretende utilizar el descenso del gradiente con lotes, ya que es necesario que todas las imágenes del lote tengan las mismas dimensiones. Para solventar este obstáculo es necesario estandarizar las dimensiones de las imágenes. Para ello se fijan unas dimensiones y reescalan todas las imágenes a dichas dimensiones. Por ejemplo, dada una imagen original, **Figura 16 a),** con unas dimensiones de 1853×89 pixeles mientras que la imagen redimensionada, **Figura 16 b),** tiene 1024×128 pixeles. Otros autores (Retsinas et al., 2022) han encontrado beneficioso mantener la ratio de aspecto de las imágenes al ajustar sus dimensiones. Salvo que la ratio de aspecto de la imagen original sea la misma que la de la imagen redimensionada será necesario añadir píxeles en una u otra dimensión hasta alcanzar las dimensiones deseadas. Este relleno se realiza de forma simétrica por ambos lados. Manteniendo como ejemplo la imagen anterior, al redimensionarla manteniendo la ratio de aspecto se obtendría una imagen de 1024×49 píxeles, por lo que para alcanzar el tamaño deseado es necesario añadir 79 píxeles a la segunda dimensión, **Figura 16 c)**.

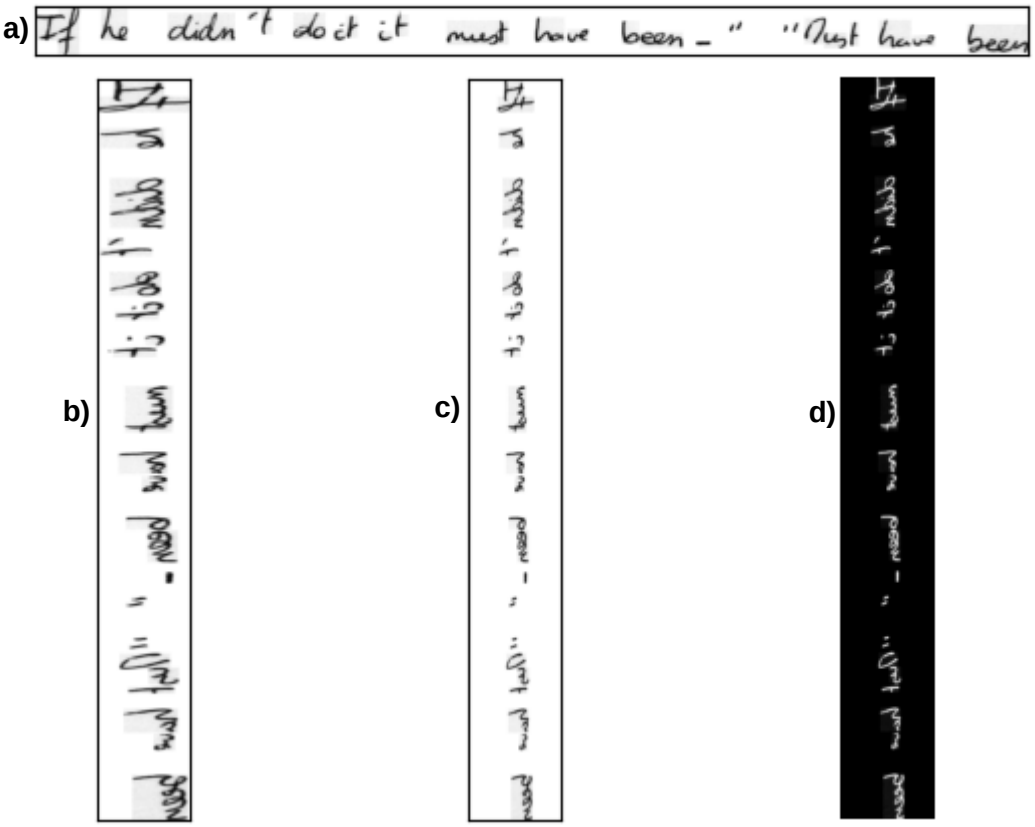


Figura 16: a) Ejemplo de una línea sin procesar. **b)** Imagen transpuesta con tamaño ajustado sin preservar la ratio de aspecto **c)** Imagen transpuesta con tamaño ajustado preservando la ratio de aspecto **d)** Imagen transpuesta con tamaño ajustado preservando la ratio de aspecto y colores en negativo.

Como es natural, las imágenes del conjunto de datos IAM están escritas con tinta oscura sobre fondo blanco. Debido a ello al procesar las imágenes se obtiene una matriz llena de unos con un puñado de ceros. Se estudiará el efecto de invertir las imágenes, **Figura 16 d)**, tanto en el rendimiento de la red como en el tiempo de entrenamiento.

En su artículo, Retsinas et al. (2022) proponen añadir espacios al principio y al final de las etiquetas durante el entrenamiento para compensar los espacios añadidos a la imagen al ajustar el tamaño sin modificar la ratio de aspecto. Para añadir estos espacios se probarán dos estrategias: añadirlos a todas las etiquetas o añadirlos de forma selectiva a aquellas imágenes cuya ratio de aspecto original sea menor al objetivo y, por lo tanto, se añada espacio en blanco delante y detrás del texto. En la **Figura 17** se muestra un ejemplo de las imágenes afectadas si se fija una ratio de aspecto de 1024:128. Como puede verse la afectación es pequeña, tan solo 492 de las 11.344 líneas correctamente segmentadas.

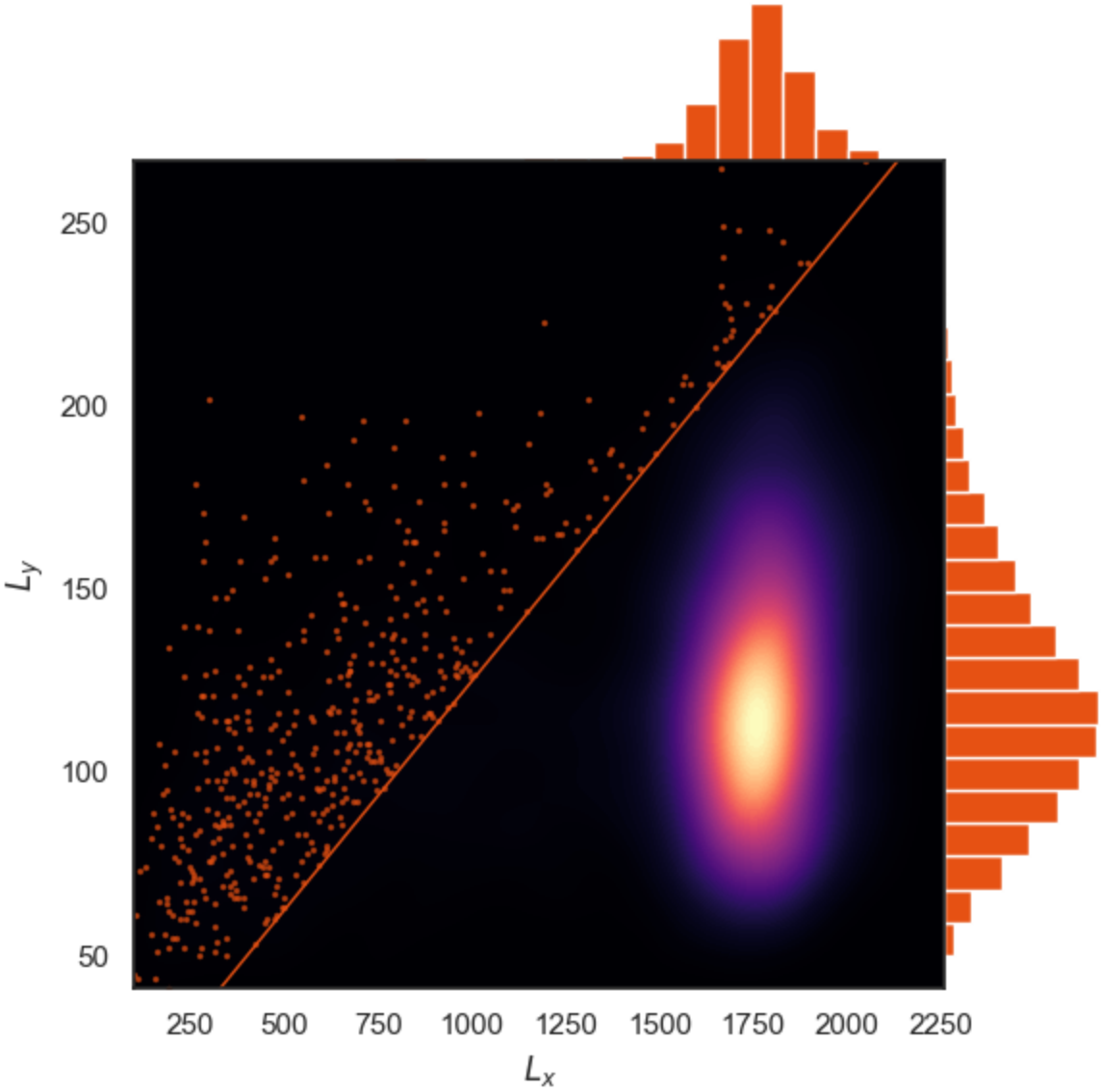


Figura 17: Distribución de tamaños de las segmentaciones por línea. Se han marcado la línea que corresponde con la ratio de aspecto 1024:128 y las imágenes cuya ratio es menor.

Las etiquetas se convierten en un vector numérico sustituyendo cada letra por el índice correspondiente al vocabulario de caracteres del conjunto de entrenamiento. Este vocabulario consta de 80 caracteres para la tarea con todos los datos. Del mismo modo que se unifica el tamaño de las imágenes también se unifica el tamaño de las etiquetas al de la mayor del conjunto de datos, es decir, 87. Para lo cual se rellena al final de las etiquetas con el carácter especial de la CTC.

4.2. Aumento de datos

La técnica del aumento de datos consiste en generar datos sintéticos a partir de los datos conocidos. Surgió originalmente como una medida para imputar datos faltantes y reducir el efecto del desbalanceo de clases (Dempster et al., 1977; Rubin, 1987). Actualmente se ha convertido en una herramienta esencial en el campo de la visión por computador (Lecun et al., 1995) en general y el reconocimiento de la escritura a mano (Puigcerver, 2017) en particular.

Los esquemas más básicos de aumentos de datos consisten en aplicar transformaciones geométricas (rotaciones, translaciones, inversiones, deformaciones, …), transformaciones del espacio de color (ajuste del brillo, del contraste o de la saturación) o injección de ruido a una fracción de las imágenes para exponer al modelo a una mayor variablidad en los datos. Con lo cual se consigue reducir el significativamente el riesgo de sobreajuste.

Esquemas de aumento de datos más complejos (Krishnan et al., 2018; Luo et al., 2020; Wigington et al., 2017), han llevado estos beneficios aún más lejos. Estas técnicas pueden adaptarse a las características específicas de un dominio, optimizando aún más el proceso de aumento de datos para tareas particulares como el reconocimiento de texto.

En el presente trabajo se empleará un aumento de datos muy sencillo: únicamente se consideran rotaciones y cizallamientos pequeños ($[-10°, 10°]$ y $[-1,1]$). Esta elección se ve motivada por los tipos de variabilidad principales en la escritura a mano: la inclinación de las líneas y las letras. Para realizar la transformación se expande el tamaño de las imágenes para mantener todo el texto, rellenando con el valor del pixel más frecuente. En la **Figura 18** se muestra un ejemplo de imagen aumentada, en caso de no aplicar el rellenado la imagen transformada sería únicamente lo contenido en el rectángulo discontinuo.

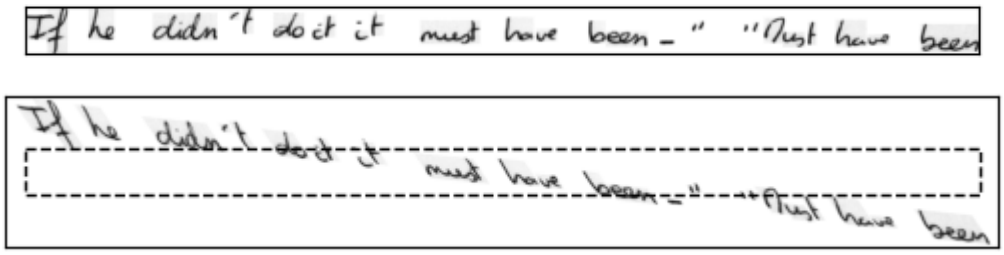


Figura 18: Ejemplo de imagen aumentada (abajo). Se ha añadido el marco de la imagen original (arriba) en línea discontinua para ilustrar el cambio de tamaño de la imagen.

4.3. Arquitectura convolucional recurrente

Se parte de la arquitectura mostrada en la **Figura 19**, la cual consta de una base convolucional para extraer características, un perceptrón multicapa para dar estructura al vector latente de características y una red recurrente para analizar las características. La salida de la RNN se clasifica mediante una capa softmax,

$$softmax(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{C+1} e^{z_j}}, \qquad (25)$$

con tantas salidas como letras en el vocabulario, C, más una (el carácter extra de la CTC) y posteriormente se decodifica de forma voraz con la CTC.

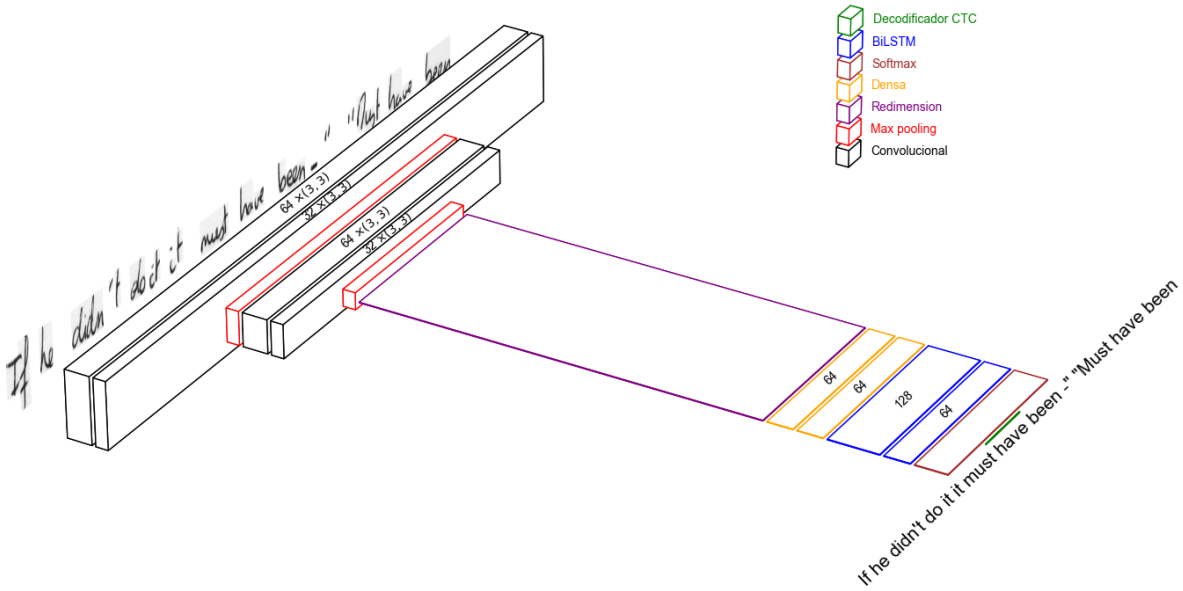


Figura 19: Arquitectura convolucional recurrente inicial. Los volúmenes de las capas se corresponden con los volúmenes de activación de las mismas.

La CNN inicial consta de dos bloques con [64, 32], [64, 32] filtros[1], con activación ReLU, sin *dropout* y seguidos de una capa de *max pooling* con *stride* 2×2. Se realizará un estudio ablativo del número de bloques, capas por bloque y filtros de las capas y del valor de regularización de *dropout*. Además, se probará a añadir regularización por normalización de lotes tanto antes como después de la activación.

[1] En este estudio se denotará la arquitectura convolucional mediante los filtros que la componen, encapsulándolos por bloques mediante corchetes. Es decir, [64, 32], [64, 32] filtros representa a una arquitectura con dos bloques, los cuales están formados por dos capas de 64 y 32 filtros respectivamente.

La función de activación ReLU,

$$ReLU(\boldsymbol{z}) = \max(\boldsymbol{z}, 0), \quad (26)$$

fue introducida por Fukushima K. en 1969. Desde entonces se ha convertido en una de las funciones de activación más utilizadas en redes neuronales profundas. Pero en el ámbito de la visión por computador ha ido siendo desplazada durante los últimos años por una variante, la ReLU con fugas (*Leaky ReLU* en inglés):

$$LReLU(\boldsymbol{z}) = \begin{cases} \boldsymbol{z}, & \boldsymbol{z} \geq 0 \\ a\boldsymbol{z}, & \boldsymbol{z} < 0 \end{cases} \quad (27)$$

Donde a es un parámetro que permite que la activación no se anule para valores negativos. En la **Figura 20** se ilustra la diferencia entre ambas variantes.

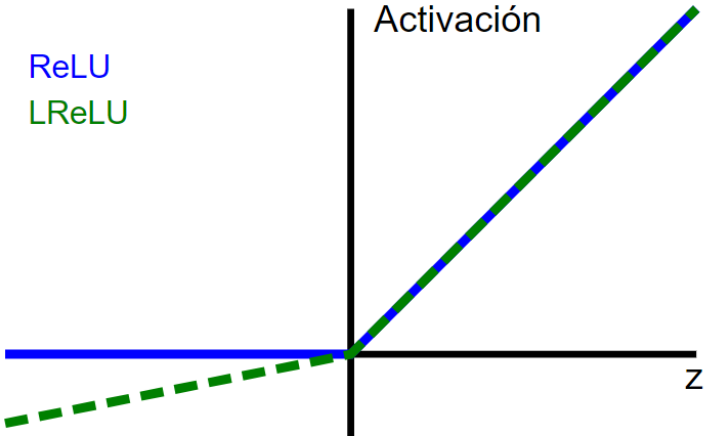


Figura 20: Comparativa entre las activaciones ReLU sin y con fugas.

A la salida del bloque CNN se añade una capa para colapsar las columnas. Inicialmente se empleará una redimensión y se comparará con el *max pooling* por columnas propuesto por Retsinas et al., (2022). Para ello se probarán dos aproximaciones: que reemplace tanto al último *max pooling* como a la redimensión o mantener el *max pooling* y que solo reemplace la redimensión.

Las características extraídas por la CNN son codificadas mediante un MLP, el cual juega un importante rol en la arquitectura cuando el método de aplanamiento es la redimensión. Esto es debido a que al concatenar las características por columnas sin aplicar ninguna función de agregación su número crece enormemente. En particular, para la inicial a la salida de del bloque convolucional se disponen de 1024 características, muchas más de las que el bloque RNN puede procesar con un coste computacional razonable. Sin embargo, se pueden procesar fácilmente mediante capas densas antes de las capas recurrentes. En la arquitectura inicial se usan 2 capas densas de 64 neuronas con activación ReLU una ratio de *dropout* de 0,4 y se realizará un estudio ablativo tanto del número de capas como de neuronas y la ratio de *dropout*.

La RNN inicialmente consta de dos bloques bidireccionales de 128 y 64 celdas LSTM con una ratio de *dropout* interno de 0.25 y se realizará un estudio ablativo tanto del número de capas como de celdas. Además, se probará varias ratios de *dropout*, tanto el de la misma capa como añadiendo otra capa de *dropout*.

Se utiliza como optimizador a Adam con una ratio de aprendizaje de 0,001. Valor que se mantendrá constante en los primeros experimentos y será reducida un factor 10 si no hay mejoras en las pérdidas CTC de validación durante 5 épocas más adelante. Se concluyen los entrenamientos cuando transcurren 10 épocas sin observarse mejoría en las pérdidas de validación.

El entrenamiento de las RNNs suele ser lento y pesado, para asistir en este proceso se ha propuesto la técnica el atajo CTC (Retsinas et al., 2022), el cual consiste en una segunda cabeza en paralelo a la RNN. Esta cabeza es una capa convolucional 1D con activación softmax y tantos filtros como letras tiene el vocabulario más una. Se probará a añadirlo tanto antes como después del MLP. Además, al ser la última capa de procesamiento no es estrictamente necesario mantener el tamaño de la entrada, por lo que se probará tanto con relleno para convolucionar los bordes como sin él.

El número de hiperparámetros de nuestra arquitectura es enorme, por lo que es impracticable probar todas las posibles combinaciones. Para paliar este problema se seguirá una búsqueda voraz y se modificarán de uno en uno para así encontrar el óptimo de cada hiperparámetro. Este método no garantiza alcanzar el óptimo absoluto de la arquitectura, aunque sí permite valorar el efecto individual de cada hiperparámetro.

De hecho, ni siquiera se va a explorar el espacio completo de hiperparámetros (ya que tiene una dimensionalidad potencialmente infinita) sino que se restringirá el estudio al subespacio formado por un pequeño número de hiperparámetros mientras que se mantienen los demás fijos. Por ejemplo, se estudiará el efecto del número de bloques convolucionales, capas por bloque y filtros por capa, pero no se explorará el efecto del tamaño de los filtros, los cuales se mantendrán en un tamaño de 3×3. Otros hiperparámetros destacados que se estudiarán son el número de capas y neuronas del bloque MLP y RNN, la ratio de *dropout*, el método de aplanamiento tras la CNN, la inclusión o no del atajo CTC.

Debido a las limitaciones del *hardware* disponible los entrenamientos con el conjunto completo de datos son demasiado lentos (entre día y medio y dos días) como para realizar una exploración exitosa de los hiperparámetros. Es por esta razón, que se realizará un primer ajuste con una versión reducida del conjunto de datos y posteriormente se probará los mejores valores con el conjunto completo. Por lo tanto, este trabajo se restringirá a entrenar con una única línea de cada formulario, elegida de forma aleatoria.

4.4. Grandes modelos del lenguaje

Actualmente existen un creciente número de grandes modelos del lenguaje multimodales comerciales que pueden realizar tareas de visión por computador. En este estudio se evaluará el rendimiento de algunos de ellos. En concreto de GPT-4o-mini y GPT-4o.

El rendimiento de los LLMs se ve afectado en gran medida por el *prompt* de entrada (Kojima et al., 2024; Yang et al., 2024; Zhou et al., 2022). En este trabajo se utilizará un *prompt* de sistema intencionalmente sencillo para evaluar las capacidades del modelo base:

You are an expert in transcribing handwritten texts. Your task is to convert handwritten text into an accurate and legible digital transcription. The handwriting may vary in clarity, so you should do your best to interpret the text while maintaining fidelity to the original content. Preserve the punctuation and capitalization. Do not add any extra information that is not present in the manuscript. If the text contains abbreviations or spelling errors, try to transcribe them as they appear. Respond only with the transcription.

Las imágenes se enviarán al modelo de una en una por lo que no es necesario reescalarlas al mismo tamaño. Por simplicidad tampoco se invertirá el color de las imágenes por lo que se suministrarán sin ningún preprocesado.

5. Resultados

5.1. Conjunto de datos reducido

Para valorar la influencia de los distintos hiperparámetros sobre el conjunto reducido se emplea únicamente las ratios CER y WER ya que la precisión es demasiado pequeña (oscila entre 0 y 4) para ser discriminativa.

El primer hiperparámetro en fijarse es el tamaño de las imágenes. La norma de las alturas es 116, número que se aproxima a la potencia de dos más cercana, 128, debido a su proximidad. En cuanto al tamaño horizontal, se realiza un estudio ablativo desde 1024 hasta 2048:

Ancho de las imágenes	CER (%)	WER (%)
1024	21,0	56
1552	20,0	57
1748	21,4	59
1750	19,9	**54**
1752	20,6	**54**
	19,2	55
1780	21,0	58
2048	21,9	59

Tabla 3: Comparativa de los diferentes tamaños de las imágenes sobre el conjunto de datos reducido. Los dos resultados para el tamaño de 1752 se corresponden a dos entrenamientos con distintas semillas. Se han marcado en negrita los óptimos.

Como puede verse en la **Tabla 3** no hay gran diferencia en el rendimiento de la red, siendo la horquilla de variabilidad de 2,7 puntos porcentuales. Comparable a la variabilidad debido a la inicialización de la red, que es de al menos 1,4 puntos porcentuales. Por otro lado, el tamaño de las imágenes impacta fuertemente en el tiempo de entrenamiento, el cual va desde 3 horas para un ancho de 1024 hasta 5 horas para un ancho de 2048. Debido a ambos factores se eligió la menor potencia de dos y será el tamaño que se mantendrá en el resto del estudio.

Una vez fijado el tamaño de las imágenes, se pasa ajustar la arquitectura, para ello se empieza por el MLP. Se probó a añadir y a quitar una capa, incluso a eliminarlo por completo. Se encontró que con una capa mejoraba, mientras que con ninguna y tres empeoraba. A continuación, se probó a aumentar el número de neuronas y se encontró que el valor óptimo es de 128, que será el valor que se mantendrá por el resto del estudio. En la **Tabla 4** pueden encontrarse los resultados de estos experimentos.

Capas	Neuronas por capa	CER (%)	WER (%)
Sin MLP		20,9	60
1	32	22,8	60
1	64	18,9	53
1	**128**	**18,4**	**52**
1	256	19,2	53
1	512	20,2	57
1	1024	21,7	59
2	64	21,0	56
3	64	24,6	62

Tabla 4: Comparativa de los diferentes tamaños del MLP codificador sobre el conjunto de datos reducido. Se ha marcado en negrita la configuración óptima.

La regularización mediante *dropout* ha demostrado ser una herramienta valiosa para enfrentarse al reconocimiento de caracteres manuscritos (Pham et al., 2014) así que se procedió a ajustarlo a continuación. Para ello se añadió una capa de *dropout* tras cada capa convolucional y recurrente. Es decir, se aplicó un doble *dropout* a las capas recurrentes: uno mediante la propia capa y otro por una capa consecutiva. Se ajustó en primer lugar el de la CNN y se obtuvo un valor óptimo de 0,1. En segundo lugar se ajustó el externo de la RNN y se obtuvo un valor óptimo de 0,4. Finalmente se ajustó el del MLP (se seguirá refiriendo así a esta parte de la arquitectura, aunque conste de una única capa) y no se observó mejoría sobre el valor de 0,4. En la **Tabla 5** pueden encontrarse los resultados de los experimentos.

dropout			*CER (%)*	*WER (%)*
CNN	MLP	RNN		
0,0	0,4	0,25 y 0,0	18,7	52
0,1	0,2	0,25 y 0,2	19,1	53
0,1	0,3	0,25 y 0,2	19.0	52
0,1	0,4	0,25 y 0,0	18,4	52
0,1	0,4	0,25 y 0,1	17,5	50
0,1	0,4	0,25 y 0,2	17,1	50
0,1	0,4	0,25 y 0,3	17,3	**49**
0,1	**0,4**	**0,25 y 0,4**	**16,8**	**49**
0,1	0,4	0,25 y 0,5	18,0	51
0,1	0,5	0,25 y 0,2	17,7	50
0,1	0,6	0,25 y 0,2	18,0	**49**
0,2	0,4	0,25 y 0,0	18,7	52
0,4	0,4	0,25 y 0,0	19,7	53

Tabla 5: Comparativa de las diferentes ratios de *dropout* sobre el conjunto de datos reducido. Los dos números de la RNN se corresponden con el *dropout* de la capa LSTM y el de la capa de *dropout*. Se ha marcado en negrita la configuración óptima.

A continuación, se ajustó la arquitectura recurrente, para ello se probó a quitar y añadir una capa, y se observaron mejoras con lo segundo. También se probó a variar el número de celdas LSTM en cada capa, obteniendo los mejores resultados para 128, 64 y 64 celdas. En la **Tabla 6** pueden encontrarse los resultados de los experimentos.

Capas	Celdas por capa	CER (%)	WER (%)
1	128	18,0	51
2	128, 64	17,7	50
3	32	20,1	56
3	64	16,8	49
3	128, 64 y 32	16,5	48
3	**128, 64 y 64**	**13,9**	**44**
3	128, 128 y 64	16,3	46
3	128	15,8	45

Tabla 6: Comparativa de las diferentes arquitecturas de la RNN sobre el conjunto de datos reducido. En la columna **Celdas por capa** se muestra el número de celdas LSTM de cada capa o el número de neuronas de todas las capas, cuando es el mismo. Se ha marcado en negrita la configuración óptima.

Una vez afinada la arquitectura recurrente se revisitó su ratio de *dropout*, esta vez utilizando únicamente la ratio interna. Para comparar las nuevas ratios con la proveniente del doble *dropout* se expresa esta como una ratio efectiva:

$$d_{eff} = 1 - (1 - d_{int})(1 - d_{ext}) \quad (28)$$

Donde d_{int} y d_{ext} son las ratios de *dropout* interno y externo, respectivamente.

dropout			CER (%)	WER (%)
CNN	MLP	RNN		
0,1	0,3	0,40	16,8	49
0,1	0,4	0,20	17,1	52
0,1	0,4	0,30	16,2	49
0,1	0,4	0,40	16,4	48
0,1	0,4	0,50	16,2	49
0,1	0,4	0,55*	**13,9**	44
0,1	0,4	0,60	18,1	53
0,1	0,5	0,30	15,5	47
0,1	**0,5**	**0,40**	**13,9**	**43**
0,1	0,5	0,50	14,5	45
0,1	0,5	0,55	18,1	53

Tabla 7: Comparativa de las diferentes ratios de *dropout* sobre el conjunto de datos reducido. Los dos resultados para la ratio de *dropout* 0,1, 0,3 y 0,4 se corresponden a dos entrenamientos con distintas semillas. Se ha marcado en negrita la configuración óptima. * Valor efectivo.

Además de ajustar la ratio de *dropout* de las capas recurrentes, se aprovechó para revisitar la de la capa densa. Se encontró que para una combinación de 0,4 y 0,5 respectivamente el desempeño de la red mejoraba, aunque marginalmente. En la **Tabla 7** pueden encontrarse los resultados de los experimentos.

En cuanto a la arquitectura convolucional se probó a añadir y quitar un bloque, a reducir el número de capas en el primer bloque, a aumentarlo en el último y a variar el número de filtros. No obstante, no se consiguió mejorar la marca inicial con dos bloques con dos capas cada uno 64 y 32 filtros. En la **Tabla 8** pueden encontrarse los resultados de los experimentos.

Filtros convolucionales	CER (%)	WER (%)
[64, 32]	18,8	51
[32], [64, 64]	17,1	49
[32, 32], [64, 64]	19,9	54
[64, 32], [64, 32]	**13,9**	**43**
[32], [64, 64], [128, 128]	19,8	55
[64, 32], [64, 32], [64, 32]	18,8	52

Tabla 8: Comparativa de las diferentes arquitecturas de la CNN sobre el conjunto de datos reducido. Se ha marcado en negrita la configuración óptima.

Además de la normalización por *dropout* se probó a añadir normalización de lotes tanto antes como después de la activación de las capas convolucionales. Como puede verse en la **Tabla 9** en ambos casos empeora, siendo después de la activación cuando peor rinde.

Normalización de lotes	CER (%)	WER (%)
Sin normalización de lotes	**13,9**	**43**
Antes de la activación	18,0	53
Después de la activación	24,3	63

Tabla 9: Comparativa del efecto de la normalización de lotes para el conjunto de datos reducido. Se ha marcado en negrita la configuración óptima.

Hasta aquí los entrenamientos realizados con una ratio de aprendizaje constante. En lo sucesivo se reducirá dinámicamente. En la **Figura 21** se incluye un ejemplo del efecto de la reducción. A partir de la reducción (en torno a la época 125), el entrenamiento tiene un comportamiento más suave y alcanza resultados ligeramente mejores en menos épocas. Todos los resultados mostrados a partir de este punto incorporan este mecanismo.

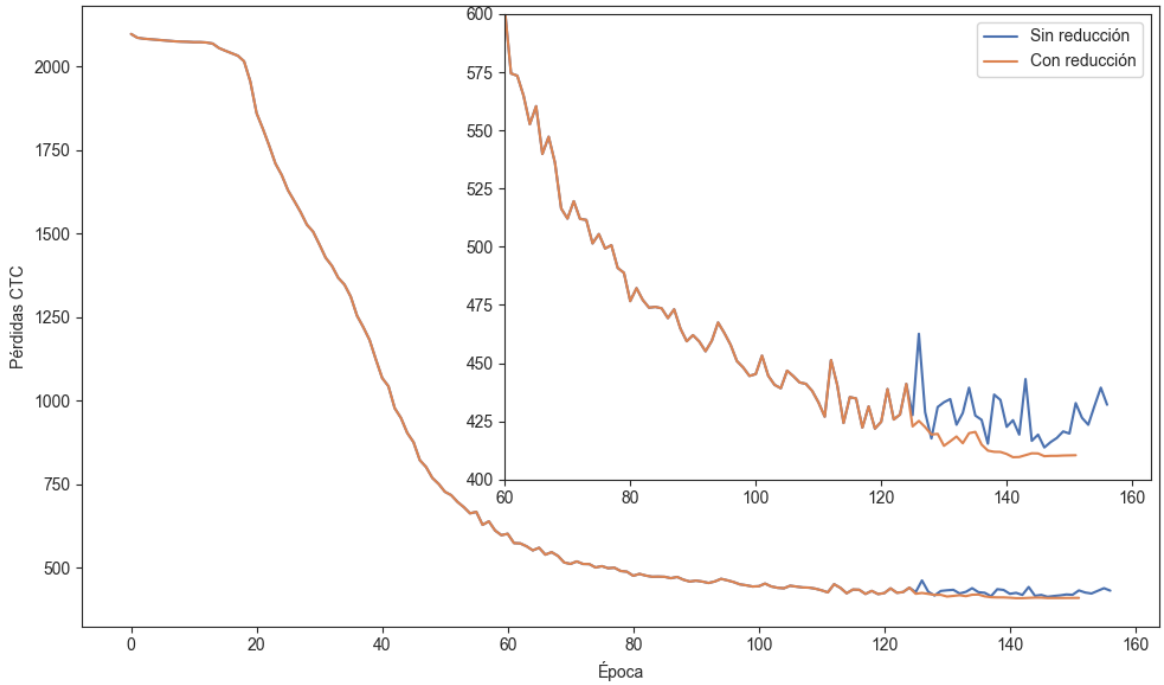


Figura 21: Comparativa de las pérdidas CTC de validación sin reducción de ratio de aprendizaje en meseta y con sobre el conjunto de datos reducido.

Se probó a añadir la segunda cabeza del atajo CTC y a sustituir la redimensión por un *max pooling* por columna. Como puede verse en la **Tabla 10**, ninguno de los dos mejoró los resultados, aunque se puede decir que es mejor aplicar el atajo sobre las características codificadas por el MLP que directamente sobre las características extraídas por la CNN. Del mismo modo es mejor sustituir tanto el *max pooling* como la redimensión por el *max pooling* por columnas ya que el caso contrario terminó atascado en lo que, en lo sucesivo se denominará metafóricamente como el «Valle de la Muerte». Más adelante se dedicará una sección a este punto.

Método de aplanamiento	Atajo CTC	CER (%)	WER (%)
Redimensionar	**Sin atajo**	**13,9**	**43**
Redimensionar	Antes del MLP	18,0	52
Redimensionar	Después del MLP	15,8	49
Max pooling por columnas	Sin atajo	16,6	49
Max pooling por columnas	Después del MLP	16,5	50

Tabla 10: Comparativa de los del rendimiento de los distintos métodos de reducción de dimensionalidad y del atajo CTC sobre el conjunto de datos reducido. Se ha marcado en negrita la configuración óptima.

También se probó a preservar la ratio de aspecto de las imágenes. Desgraciadamente, la red terminó atascada en el «Valle de la Muerte».

5.2. Tarea con todos los datos

Una vez explorado el espacio de hiperparámetros con el conjunto de datos reducido, se pasa a utilizar el conjunto de la tarea, al cual le añaden los datos excluidos originalmente pero que no se encuentran en las particiones de validación o prueba. Primeramente, se revisó el preentrenamiento, recogiendo los resultados en la **Tabla 11**. Al añadir más datos las métricas mejoran, en especial la precisión por líneas. Por lo tan tanto, en lo sucesivo se añadirá a las tablas.

Ratio de aspecto	Espacios	Color	CER (%)	WER (%)	Precisión (%)
No preservado	No	Positivo	7,9	28	18
Preservado	No	Positivo	**7,3**	27	20
Preservado	**No**	**Negativo**	7,8	28	20
			7,3	**26**	**22**
Preservado	En todas	Positivo	8,0	29	17
Preservado	Selectivos	Positivo	9,2	32	15
			8,4	30	16
Preservado	Selectivos	Negativo	8,1	29	20

Tabla 11: Comparativa de los diferentes preprocesados de las imágenes y etiquetas sobre la partición de la tarea con excluidos. Los resultados que comparten el mismo preprocesamiento son entrenamientos con distintas semillas. Se ha marcado en negrita la configuración óptima.

Al disponer de más datos el entrenamiento que preserva la ratio de aspecto es más fluido y termina en un punto muerto únicamente en una ocasión. El efecto de esta modificación es muy positivo en el rendimiento de la red. Por contra el efecto de añadir espacios es negativo, tanto si se añaden de forma selectiva o en todas las etiquetas. En cuanto a las imágenes negativas el efecto es pequeño pero positivo. El efecto es mayor en el caso en el que se añaden espacios. Esto puede ser debido a que las capas convolucionales añaden un marco de ceros para poder convolucionar los pixeles del borde de la imagen. En el caso de las imágenes negativas este marco puede ser interpretado como espacios extras al ser del mismo color que el resto del fondo.

Al aumentar el número de datos es posible emplear arquitecturas mayores, por lo que se revisitaron las CNNs más grandes. Además, se volvió a probar el aplanamiento mediante *max pooling*. Por un lado, con todos los datos es posible entrenar una red convolucional con tres bloques y mejorar los resultados, como puede verse en la **Tabla 12**. Se probaron arquitecturas aún más profundas, pero en todos los entrenamientos terminaron en el Valle de la Muerte. Por otro lado, el *max pooling* por columnas mantiene su efecto nocivo sobre la arquitectura de [64, 32], [64, 32], aunque para la de [32], [64, 64], [128, 128] filtros sí es beneficioso. Este efecto es debido a que en el primer caso la última capa convolucional tiene 32 filtros, por lo que el número de características por columna es inferior al número de posibles caracteres a predecir, que es 87. Por el contrario, al aumentar el número de filtros en la última capa convolucional hasta 128 permite que la red pueda codificar suficiente

información para identificar todos los caracteres correctamente, aun en el caso de mantener una única característica por columna. En los sucesivos entrenamientos se mantendrá la arquitectura de tres bloques y el aplanamiento por *max pooling*.

Filtros convolucionales	Método de aplanamiento	CER (%)	WER (%)	Precisión (%)
[64, 32], [64, 32]	Redimensionar	7,3	27	20
	Max pooling por columnas	9,7	33	14
[32], [64, 64], [128, 128]	Redimensionar	7,6	28	19
	***Max pooling* por columnas**	**7,0**	**26**	**21**

Tabla 12: Comparativa de las diferentes arquitecturas de la CNN sobre el conjunto de la tarea con todos los datos. Se ha marcado en negrita la configuración óptima.

Todos los modelos mostrados en la **Tabla 12** trabajan con etiquetas sin espacios e imágenes que han sido rescatadas manteniendo la ratio de aspecto y sin inversión de color. Como se vio anteriormente, al trabajar con todos los datos la inversión de color resulta beneficiosa, por lo que se reentrena el modelo con [32], [64, 64], [128, 128] filtros con las imágenes en negativo. En este caso se logra una mejora más sustancial, llegando a una CER de 6,5%, una WER de 24% y una precisión por líneas de 23%.

Pese a que el atajo CTC no mostró mejoría sobre el conjunto reducido se le da una segunda oportunidad sobre la tarea ya que al disponer de más datos se pueden explorar arquitecturas mayores. Esta vez se añade el atajo únicamente después del MLP puesto que sobre el conjunto reducido era la opción más prometedora. Además, se prueba a aplicar relleno o no. Como puede verse en **Tabla 13**, al disponer de más datos y una arquitectura convolucional más grande, el atajo sí muestra una mejora en el rendimiento. Con respecto al uso o no de relleno la diferencia es muy pequeña. Aun así, en los entrenamientos subsiguientes se aplica relleno para mantener la simetría en el número de caracteres predichos tanto por la RNN como por el atajo CTC.

Filtros convolucionales	Atajo CTC	CER (%)	WER (%)	Precisión (%)
[32], [64, 64], [128, 128]	Sin atajo	6,5	24	23
[32], [64, 64], [128, 128]	Sin relleno	**6,3**	24	23
[32], [64, 64], [128, 128]	**Con relleno**	**6,3**	**23**	**26**

Tabla 13: Comparativa del rendimiento del atajo CTC sobre el conjunto de la tarea con todos los datos. Se ha marcado en negrita la configuración óptima.

Las métricas que se recogen en la **Tabla 13** corresponden únicamente a la rama RNN. El atajo solo se utiliza como una asistencia durante el entrenamiento y no se espera que sus predicciones sean particularmente precisas. De hecho, para el caso del relleno alcanza tan solo una CER de 24,5%.

Al adoptar el *max pooling* por columnas como método de aplanamiento la importancia del MLP se diluye. Por lo que a continuación se estudia eliminarlo a la par que se profundiza en la arquitectura convolucional:

MLP	Filtros convolucionales	CER (%)	WER (%)	Precisión (%)
128	[32], [64, 64], [128, 128]	6,3	23	26
	[32], [64, 64], [128, 128, 128, 128]	7,7	28	18
Sin MLP	[32], [64, 64], [128, 128]	6,1	23	24
	[32], [64, 64], [128, 128], [256]	6,0	23	26
	[32], [64, 64], [128, 128], [256, 256]	6,2	24	25
	[32], [64, 64], [128, 128, 128], [256]	6,2	24	25
	[32], [64, 64], [128, 128, 128, 128]	**5,7**	**22**	**27**

Tabla 14: Comparativa del rendimiento del MLP para distintas arquitecturas convolucionales sobre el conjunto de la tarea con todos los datos. Se ha marcado en negrita la configuración óptima.

Como puede verse en la **Tabla 14**, al eliminar el MLP se pueden entrenar arquitecturas convolucionales más profundas, lo cual conduce a un mejor rendimiento del modelo. No obstante, el margen para añadir capas convolucionales no es muy grande, ya que en las arquitecturas con 7 capas que terminan en capas de 256 filtros el sobreajuste produce que el rendimiento baje. Del mismo modo, si se mantiene la capa densa al aumentar la profundidad del módulo convolucional el rendimiento se deteriora. Para ilustrar el crecimiento del nivel de sobreajuste en la **Figura 22** se muestran las curvas de aprendizaje para el modelo más pequeño, el mejor y el peor modelo. En las pruebas sucesivas se partirá del modelo sin MLP y con [32], [64, 64], [128, 128, 128, 128] filtros.

Tras reajustar el tamaño de la CNN se probó a añadir fugas a su función de activación ReLU. En la **Figura 23** puede verse la evolución de la CER en función de la pendiente negativa de la función de activación mientras que en la **Tabla 15** se recogen todas las métricas. Se probaron pendientes negativas desde 0,01 hasta 0,4 y en todos los casos se observó un deterioro del rendimiento del modelo, por ello en los siguientes experimentos se mantuvo la ReLU sin fugas como función de activación.

Pendiente negativa	CER (%)	WER (%)	Precisión (%)
0,00	**5,7**	**22**	**27**
0,01	6,1	23	26
0,05	7,5	28	19
0,10	8,1	30	18
0,20	10,2	37	12
0,40	10,8	37	12

Tabla 15: Comparativa del rendimiento de la pendiente negativa de la ReLU con fugas sobre el conjunto de la tarea con todos los datos. Se ha marcado en negrita la configuración óptima.

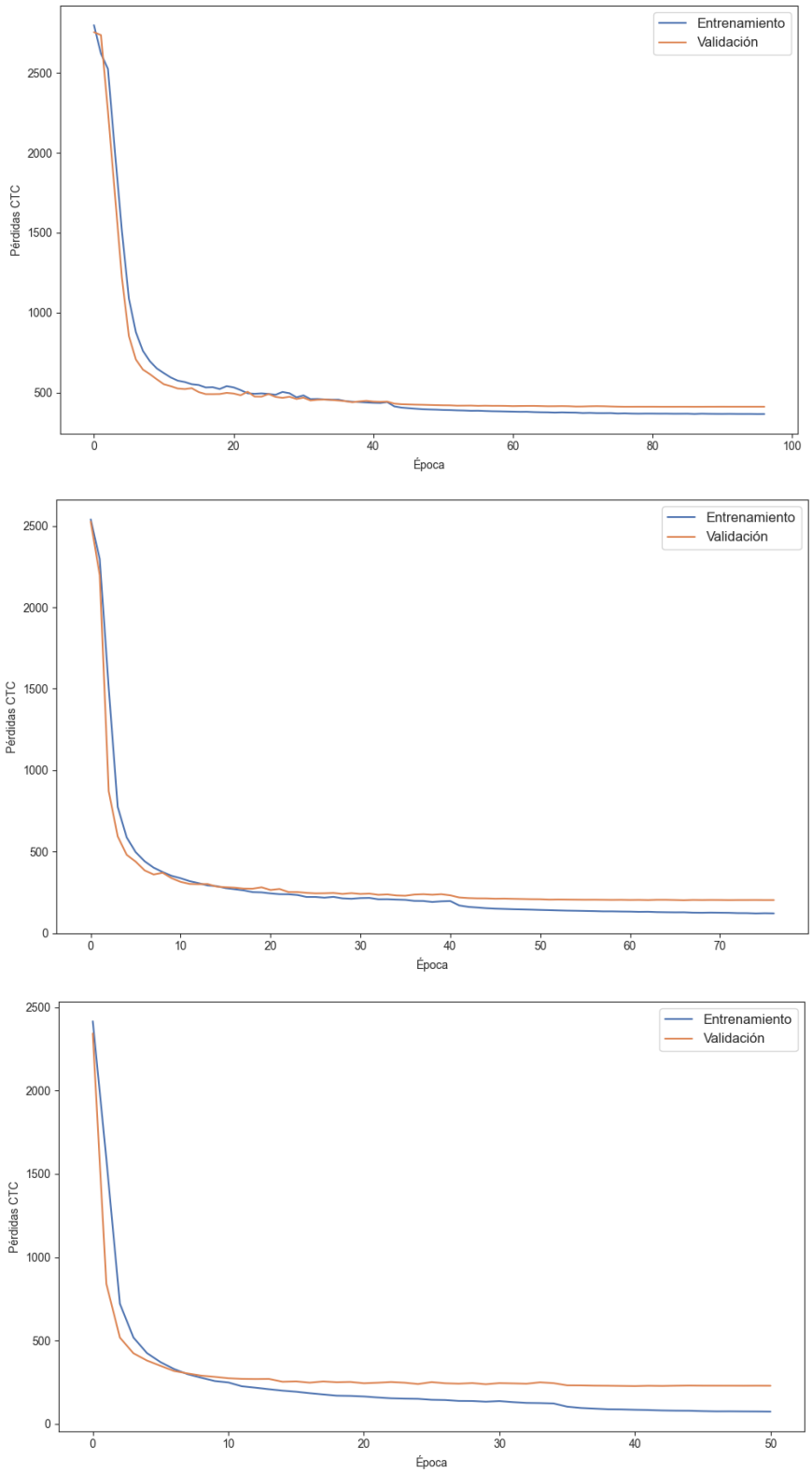


Figura 22: Curvas de aprendizaje para las arquitecturas de [32], [64, 64], [128, 128] (arriba), [32], [64, 64], [128, 128, 128, 128] (centro) y [32], [64, 64], [128, 128], [256, 256] (abajo) filtros sobre el conjunto de la tarea con todos los datos.

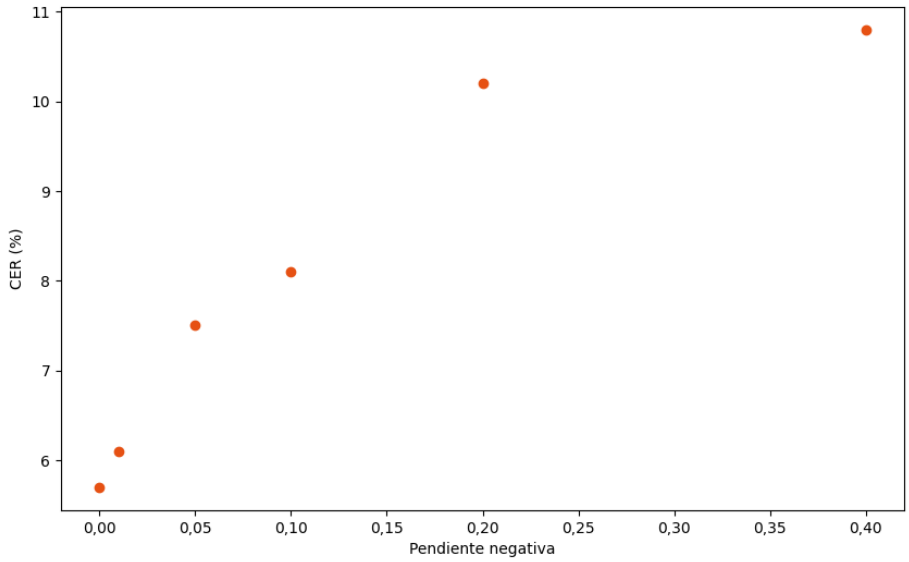


Figura 23: Comparativa del rendimiento de la pendiente negativa de la ReLU con fugas sobre el conjunto de la tarea con todos los datos.

Como puede verse en la **Figura 22**, al aumentar el tamaño del módulo CNN conlleva un aumento del sobreajuste. Este sobreajuste no es un problema para la arquitectura con [32], [64, 64], [128, 128, 128, 128] filtros, que logra mejores resultados que la más pequeña, pero a partir de ahí lastra los resultados de los modelos mayores. Para contrarrestar este efecto se puede aumentar el nivel de regularización, en particular la ratio de *dropout*. Al aumentar la regularización se volvió a entrenar arquitecturas con cuatro bloques. Como puede verse en la **Tabla 16** las tres arquitecturas mejoran al aumentar la ratio de dropout a valores de 0,2-0,3 pero a partir de ahí su rendimiento decae. De todas las pruebas realizadas los mejores resultados se obtienen para la arquitectura con [32], [64, 64], [128, 128], [256] filtros y una ratio de dropout de 0,2.

Filtros convolucionales	*dropout*	CER (%)	WER (%)	Precisión (%)
[32], [64, 64], [128, 128], [256]	0,1	6,0	23	26
	0,2	**5,5**	**21**	**28**
	0,3	5,5	21	27
	0,4	6,4	24	22
[32], [64, 64], [128, 128], [256, 256]	0,1	6,2	24	25
	0,2	5,7	22	27
	0,3	5,7	22	27
[32], [64, 64], [128, 128, 128, 128]	0,1	5,7	22	27
	0,2	5,7	22	28
	0,3	6,7	25	23
	0,4	9,5	32	13

Tabla 16: Comparativa del rendimiento de la regularización por *dropout* para distintas arquitecturas convolucionales sobre el conjunto de la tarea con todos los datos. Se ha marcado en negrita la configuración óptima.

Para concluir, se probó la técnica del aumento de datos. Para ello se entrenó con varios niveles de aumento de datos. Como puede verse en la **Tabla 17** el aumento de datos tiene un efecto positivo sobre el desempeño de la red. Además, se observa que este efecto es máximo al entrenar con un 50% de datos aumentados, mientras que al utilizar más variaciones de los mismos datos la mejora es menor.

Fator de aumento de datos	CER (%)	WER (%)	Precisión (%)
0,00	5,5	21	28
0,50	**4,8**	**19**	29
0,75	5,0	20	**31**

Tabla 17: Comparativa del rendimiento del aumento de datos sobre el conjunto de la tarea con todos los datos. Se ha marcado en negrita la configuración óptima.

5.3. Valle de la Muerte

Para explicar qué significa que un entrenamiento termine en el «Valle de la Muerte» primero se debe entender qué es un entrenamiento «normal». Para ilustrarlo se emplea la **Figura 24**, la cual muestra la evolución de las predicciones de una imagen superpuesta a la curva de aprendizaje un entretenimiento sobre el conjunto de la tarea. Se pueden distinguir fácilmente tres etapas en la gráfica. Inicia con meseta en la cual la red se familiariza con la tarea y no predice más que espacios en blanco y el carácter «*blank*». En este entrenamiento no se añadieron los datos no incluidos originalmente en la tarea ya que con más ejemplos por época la longitud de la meseta disminuye y llegan incluso a desaparecer en la mayoría de entrenamientos. Cuando la red empieza a predecir caracteres se inicia una segunda etapa en la que rápidamente se reducen las pérdidas a la par que mejoran las predicciones. A medida que las predicciones mejoran el entrenamiento se ralentiza y llega a una fase de saturación en la que las predicciones mejoran cada vez más lentamente hasta que el entrenamiento concluye.

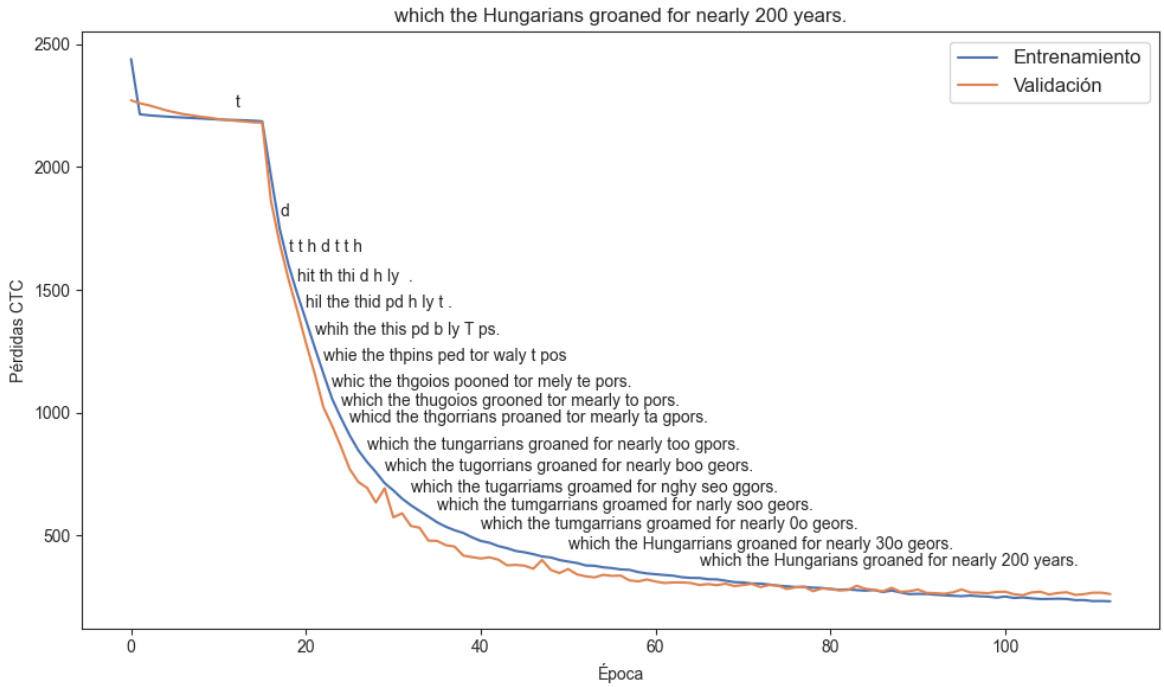


Figura 24: Ejemplo de curva de aprendizaje con la evolución de la predicción.

Una vez entendido el comportamiento nominal de un entrenamiento es posible centrarse en las anomalías. En la **Figura 25** se han recogido todos los entrenamientos patológicos que se han obtenido mientras se preparaba este trabajo. Todos comparten unas características comunes: la planicie inicial se alarga y el descenso rápido se trunca antes de tiempo, lo cual conduce a una fase muy inestable que puede rebotar incluso de vuelta a la planicie inicial.

Además del comportamiento de las curvas de aprendizaje, hay otra característica que tienen en común todos los entrenamientos que terminan en el Valle de la Muerte: las predicciones se vuelven prácticamente independientes de las imágenes. En este estado las redes predicen la misma cadena de texto (o un reducido conjunto de ellas) para todas las imágenes. En la **Tabla 18** se recogen las predicciones y métricas al final del entrenamiento para todos los entrenamientos fallidos.

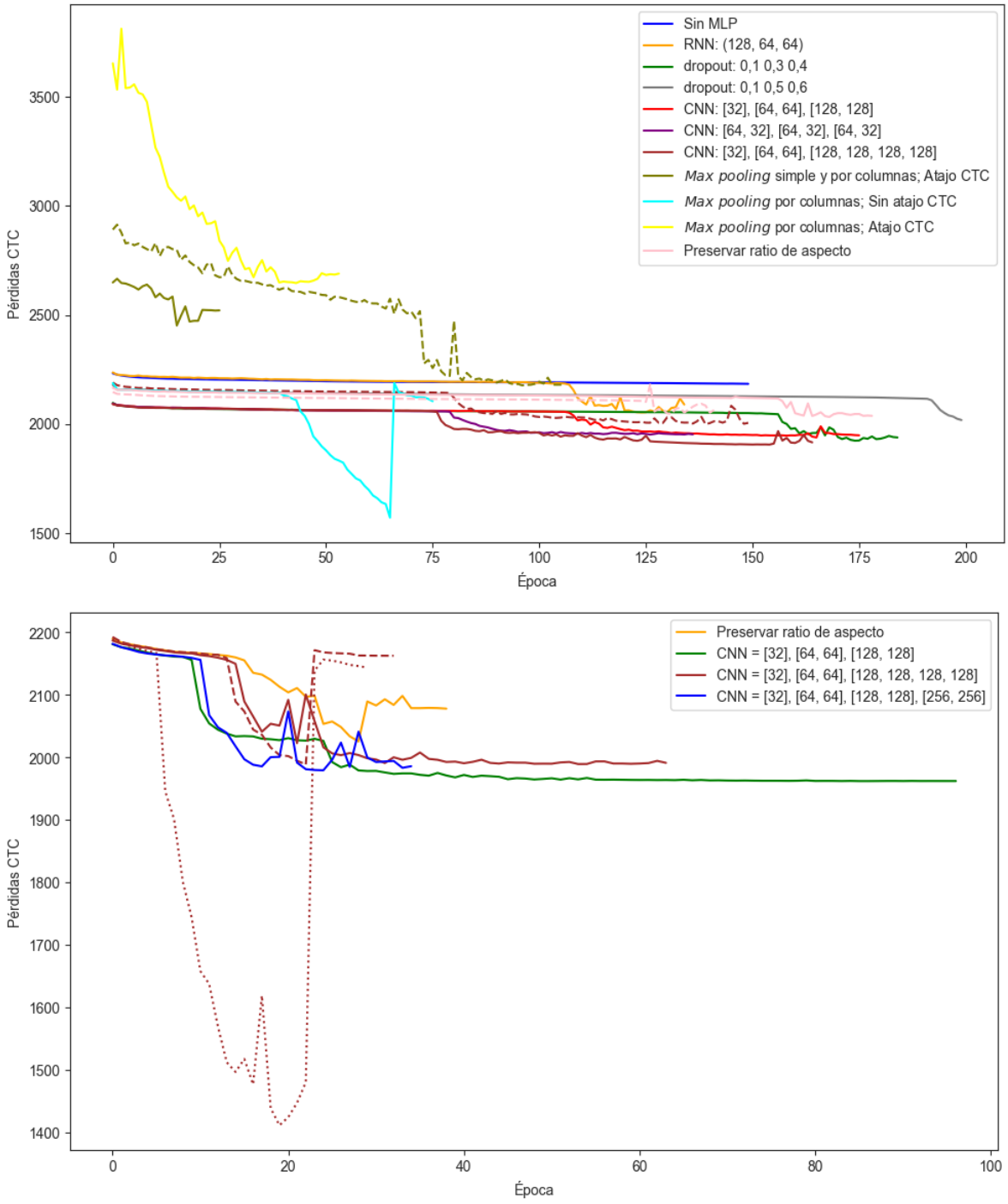


Figura 25: Pérdidas de validación de entrenamientos que terminaron en el Valle de la Muerte. Arriba, sobre el conjunto de datos reducido. Abajo, sobre la tarea con todos los datos. Las curvas con el mismo color y distinto punteado son entrenamientos con los mismos parámetros y distintas inicializaciones.

Conjunto	Hiperparámetros	Predicciones	CER (%)	WER (%)
Reducido	MLP: ()	«if»	95,9	99
	RNN: 128, 64, 32	«b b b b b p f», «b b b b f b f»	78,4	101
	Dropout: 0,1 0,3 0,4	«», «e»	100,0	100
	CNN: [32], [64,64], [128, 128]	«e»	93,8	100
	CNN: [64, 32], [64, 32], [64, 32]	«o»	76,8	112
	CNN: [32], [64,64], [128, 128, 128, 128]	«e e e e e e e e e e e»	97,6	100
		«si»	97,8	100
	Preservar ratio de aspecto	«h»	76,5	163
		«», «e», «o»	95,7	100
	Max pooling simple y por columnas	«»	100,0	100
		«e e e e e e ee e e»	88,7	100
	Max pooling por columnas; Sin atajo CTC	«»	66,8	98
	Max pooling por columnas; Atajo CTC	«»	100	100
Tarea con todos los datos	Preservar ratio de aspecto	«»	100,0	100
	CNN: [32], [64,64], [128, 128]; Redimensionar	«e e o e o e o e ee»	73,9	117
	CNN: [32], [64,64], [128, 128]; *Max pooling* por columnas	«eeeeeeeeeeee»	90,0	100
	CNN: [32], [64,64], [128, 128, 128, 128]	«e e e e ee e e ee»	76,0	109
		«o e e e e e e e e e ee»	74,5	139
		«»	58,6	86
	CNN: [32], [64,64], [128, 128], [256, 256]	«»	74,6	151

Tabla 18: Predicciones y métricas para los entrenamientos que terminaron en el Valle de la Muerte. Los resultados que comparten los mismos hiperparámetros son entrenamientos con distintas semillas.

En la mayoría de los casos con probar un par de inicializaciones se consigue un entrenamiento exitoso, aunque para los modelos más grandes no es el caso. Puesto que el comportamiento de un entrenamiento exitoso de uno que caiga en el Valle de la Muerte diverge durante las primeras épocas se planteó realizar múltiples experimentos sobre la arquitectura convolucional con [32], [64,64], [128, 128, 128, 128] filtros y detener el entrenamiento de forma prematura cuando alcanzaba el Valle de la Muerte. Como puede verse en la **Figura 26** el resultado más prometedor fue el de la semilla 2746317213, que consiguió evadir el Valle de la Muerte durante unas épocas hasta sufrir un olvido catastrófico que lo condujo directamente a él.

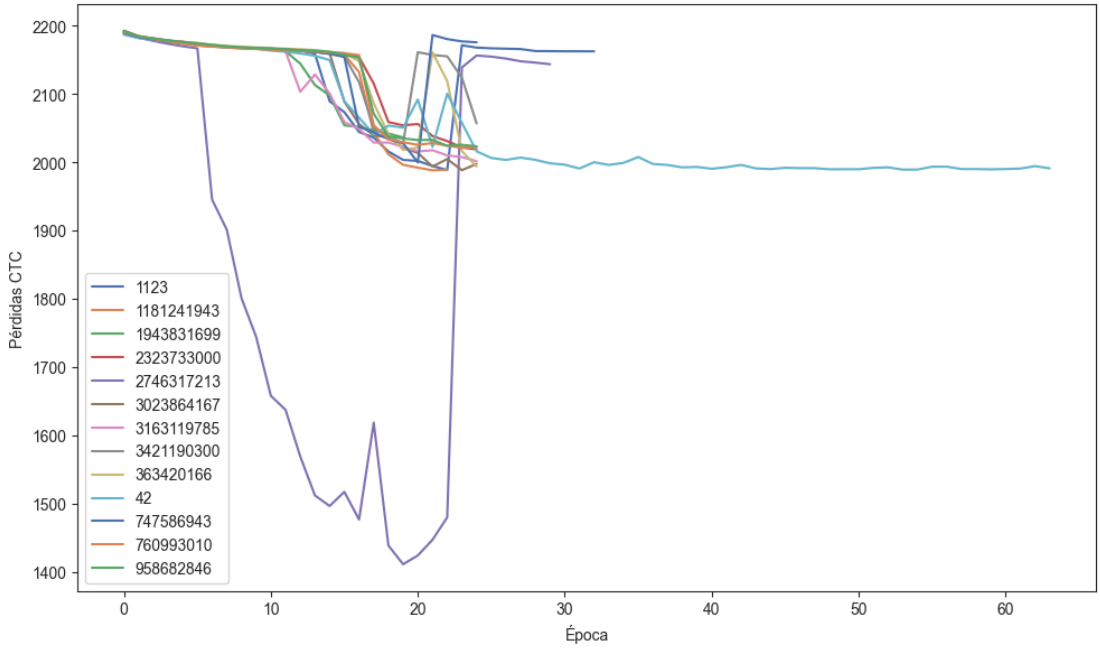


Figura 26: Pédidas de validación de entrenamientos para la CNN con [32], [64,64], [128, 128, 128, 128] filtros sobre la tarea con todos los datos.

Como puede verse en la **Tabla 14**, al incluir el atajo CTC fue posible entrenar modelos con [32], [64,64], [128, 128, 128, 128] e incluso más filtros. La rama CTC, al ser más ligera que la LSTM actúa como un factor protector que evita que la red caiga en el Valle de la Muerte.

5.4. Análisis de la exploración

En este trabajo se ha realizado una amplia exploración del espacio de hiperparámetros de la arquitectura convolucional recurrente, por lo que en esta sección se recopilarán y estudiarán los resultados obtenidos en el proceso. Para ilustrar la exploración realizada puede consultarse la **Figura 27**, donde se muestra el resultado del mejor modelo encontrado hasta el momento a lo largo del estudio en función del hiperparámetro ajustado. También se ha marcado con distinto punteado el conjunto empleado para el entrenamiento, lo cual permite mostrar que el mayor salto en rendimiento se produjo al emplear todos los datos, mientras que los ajustes en la arquitectura han aportado mejoras más moderadas.

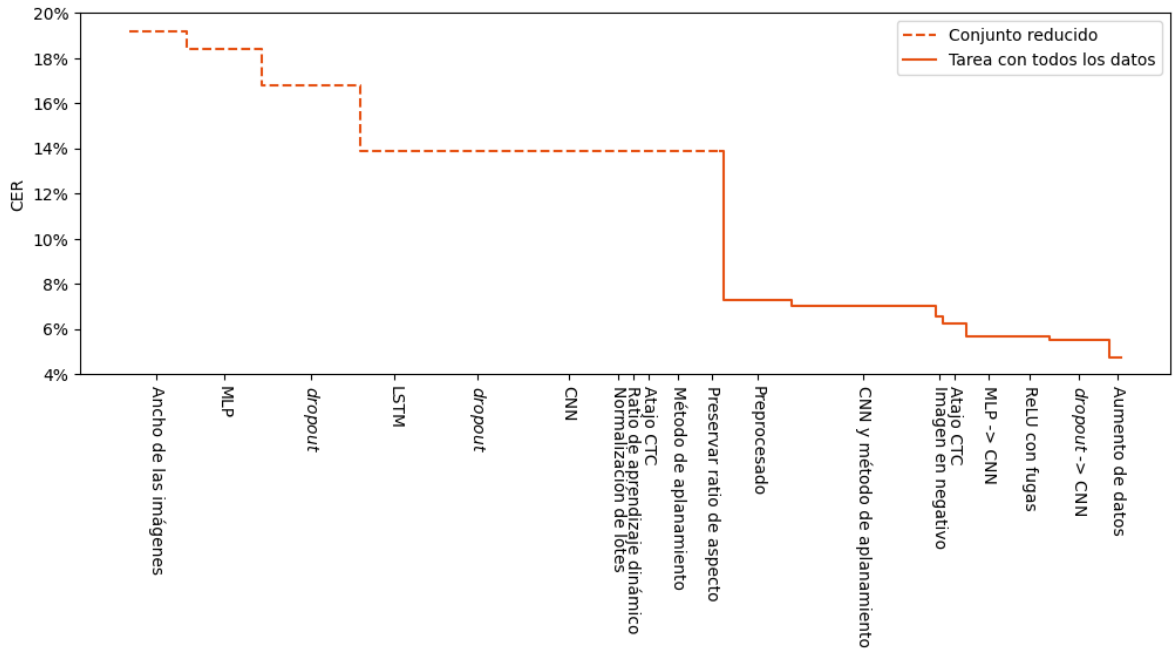


Figura 27: Evolución de la CER a lo largo del estudio.

En este estudio se han realizado un total de 136 entrenamientos por lo que se dispone de una gran cantidad de datos para estudiar las relaciones entre las métricas. Las métricas CER y WER miden el desempeño del modelo con distinto nivel de granularidad al trabajar a nivel de caracteres y palabras respectivamente. Aunque esta granularidad permite que aporten información diferente, cabe esperar cierto grado de correlación entre ambas ya que las palabras incorrectamente transcritas lo son porque contienen caracteres incorrectamente transcritos. Esta correlación queda patente si se grafica una en función de la otra. Como puede verse en la **Figura 28**, los resultados de todos los modelos entrenados en este estudio se ajustan a una ley de potencias, con $r^2 = 0{,}995$. Para el ajuste se han excluido los entrenamientos que terminaron atascados en el Valle de la Muerte ya que se comportan de forma anómala. Aunque las métricas se encuentran muy relacionadas existe cierta dispersión, la cual es mayor en los entrenamientos sobre el conjunto reducido y disminuye al considerar más datos durante el entrenamiento.

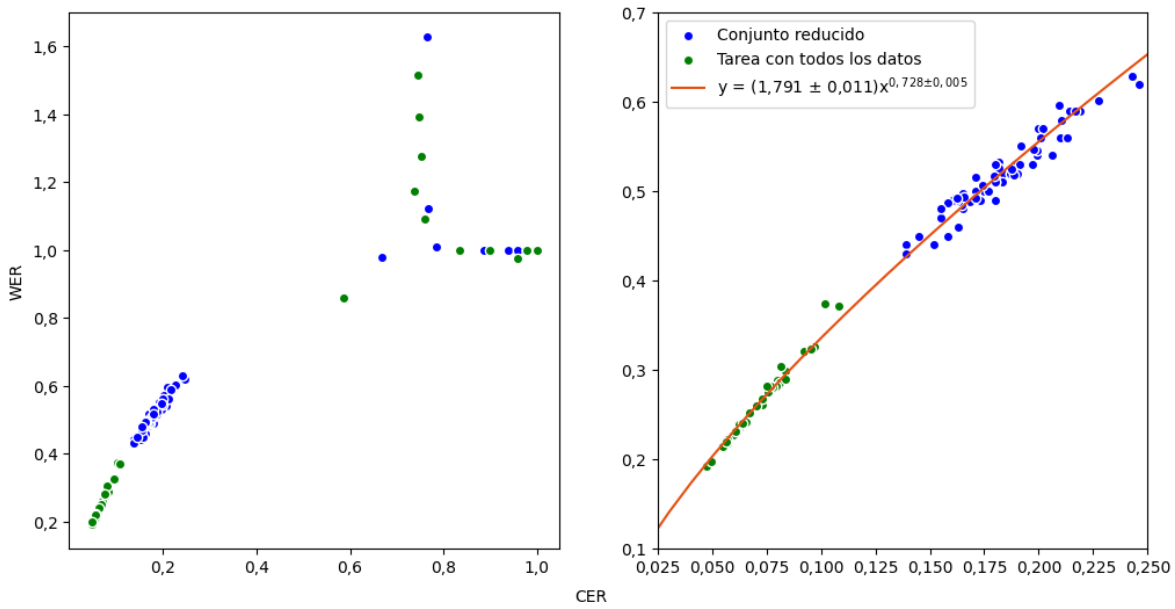


Figura 28: Relación entre las métricas CER y WER. A la izquierda se incluyen todos los resultados. A la derecha se han excluido los casos en los que el entrenamiento terminó en el Valle de la Muerte y se ha ajustado a una ley de potencias con $r^2 = 0{,}995$.

Otra relación relevante es la que existe entre las pérdidas CTC y la ratio CER, la cual se explora en la **Figura 29**. Si se excluyen los resultados del Valle de la muerte puede observarse la relación lineal ($r^2 = 0{,}98$) que liga a ambas magnitudes.

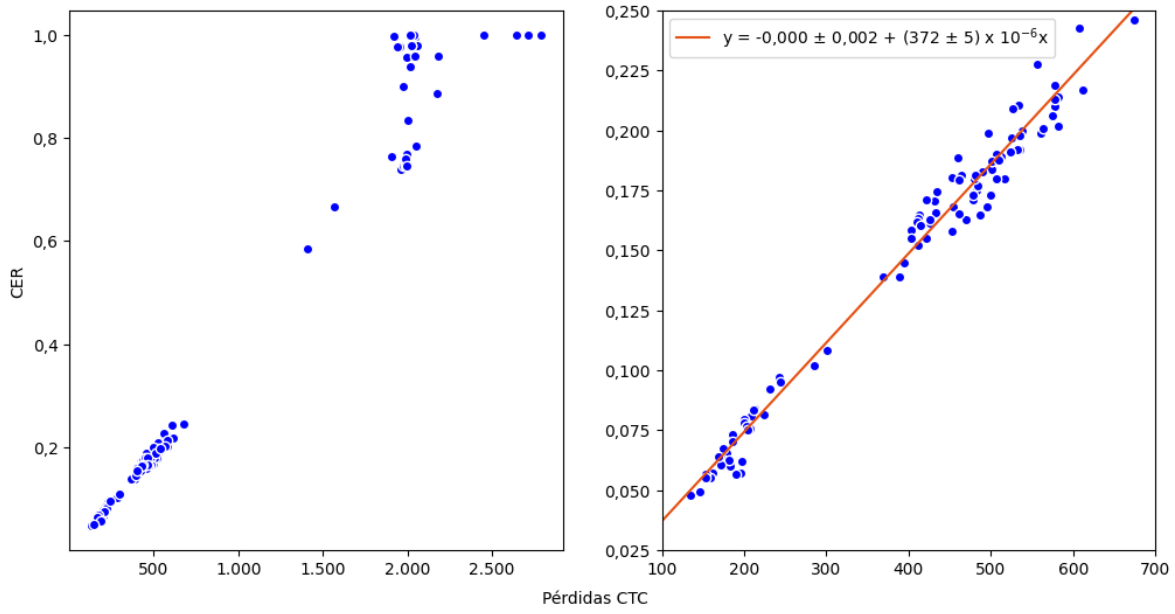


Figura 29: Relación entre las pérdidas CTC y la CER. A la izquierda se incluyen todos los resultados. A la derecha se han excluido los casos en los que el entrenamiento terminó en el Valle de la Muerte y se ha ajustado a una línea recta con $r^2 = 0{,}98$.

5.5. Grandes modelos del lenguaje

Puesto que los LLMs ya están entrenados y no se pretende realizar un estudio sobre la influencia del *prompt* sobre la calidad de las trascripciones se evaluarán los modelos directamente sobre el conjunto de prueba de la tarea IAM. En la **Tabla 19** se recogen las evaluaciones para el modelo GPT-4o en su versión estándar y mini. Como era de esperar los resultados para el modelo grande son mucho mejores.

Modelo	CER (%)	WER (%)	Precisión (%)
GPT-4o-mini	6,7	18	33
GPT-4o	3,6	10	53

Tabla 19: Resultados de los LLMs sobre la partición de prueba del conjunto de la tarea.

5.6. Comparación con el estado del arte

Tras la exploración del espacio de hiperparámetros los mejores resultados se obtuvieron preprocesando las imágenes en negativo y ajustándolas a un tamaño de 1024×128 píxeles manteniendo la ratio de aspecto además de aumentar los datos en un factor 0,5. La mejor arquitectura obtenida (**Figura 30**) consta de cuatro bloques convolucionales con una capa de 32 filtros el primero, dos de 64 el segundo y dos de 128 el tercero y una de 256, tres capas LSTM bidireccionales de 128, 64 y 64 celdas, respectivamente y el atajo CTC. Como método de aplanamiento se utilizó el *max pooling* por columnas.

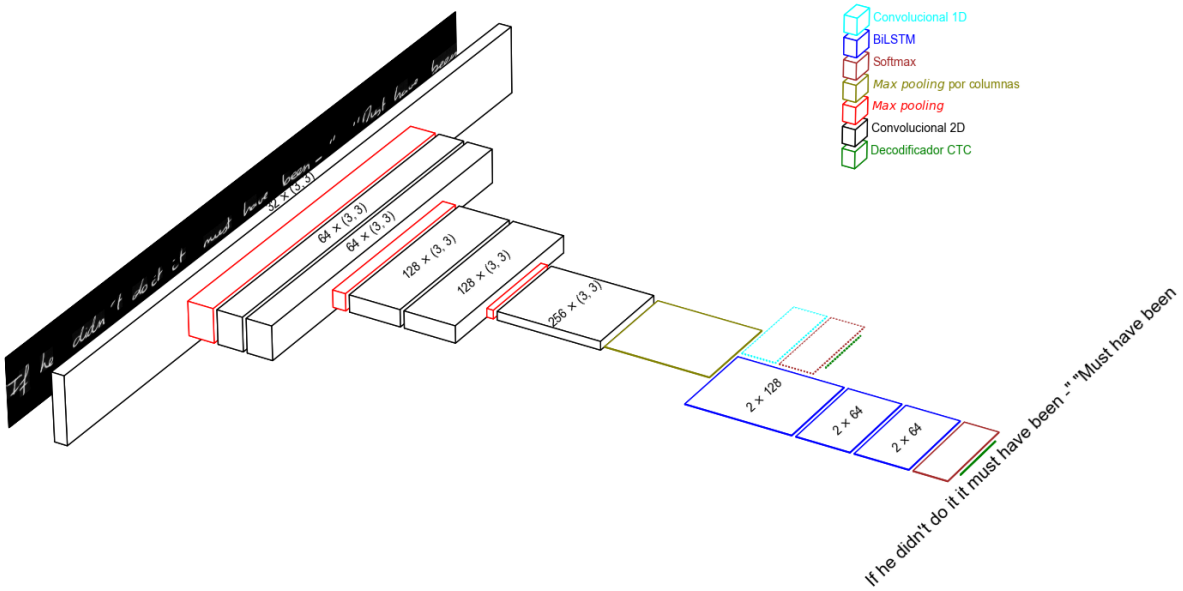


Figura 30: Arquitectura convolucional recurrente final. Los volúmenes de las capas se corresponden con los volúmenes de activación de las mismas. La cabeza del atajo CTC se ha marcado en línea discontinua ya que solo se utiliza durante el entrenamiento.

Durante todo el estudio para comparar los modelos se han medido las métricas sobre la partición de validación. Una vez concluido el ajuste de los hiperparámetros se puede evaluar los resultados finales sobre una partición, la de prueba, que no ha sido empleado en ningún momento de todo el proceso. En la **Tabla 20** se recogen los resultados del mejor modelo convolucional recurrente sobre la partición de prueba, también se incluye la evaluación del mejor LLM para comparar.

Modelo	CER (%)	WER (%)	Precisión (%)
CRNN final	5,6	22	24
GPT-4o	3,6	10	53

Tabla 20: Resultados del mejor modelo CRNN y LLM sobre las particiones de prueba del conjunto de la tarea con todos los datos.

Para contextualizar estos resultados conviene compararlos con el estado del arte, para ello se recupera la visualización presentada anteriormente con la evolución del CER (**Figura 1**) y se añaden los resultados del presente estudio. Como puede verse en la **Figura 31** los resultados del modelo entrenado desde cero en este estudio son comparables con el estado del arte de los últimos años. En cuanto a GPT4o aún siendo un sistema generalista que no se ha entrenado específicamente para la tarea de reconocimiento de caracteres manuscritos presenta un rendimiento mejor que la mayoría de los sistemas especializados.

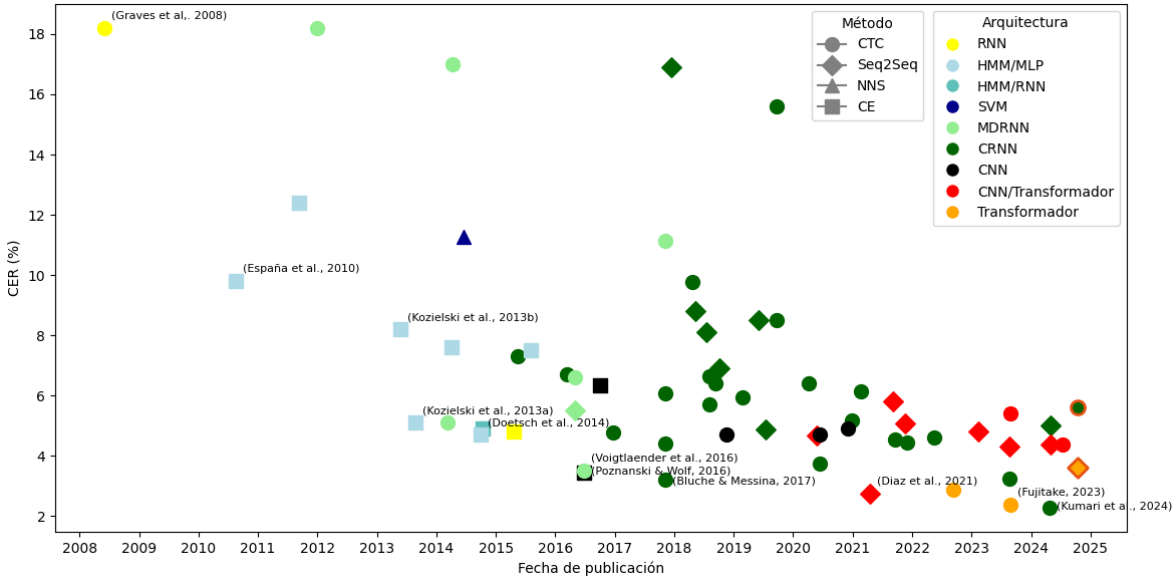


Figura 31: Comparación con el estado del arte de los resultados del estudio (borde naranja) tanto para el modelo entrenado desde 0 (verde) como el LLM evaluado.

6. Conclusiones

En el transcurso de esta investigación, se ha logrado desarrollar un sistema innovador y eficiente para el reconocimiento de caracteres manuscritos. Este sistema se destaca por su rendimiento sobresaliente y, lo que es igualmente importante, por su tamaño considerablemente reducido. Esta característica de compacidad es particularmente significativa, ya que permite que el sistema haya sido entrenado utilizando únicamente recursos de CPU, lo cual amplía considerablemente su accesibilidad y aplicabilidad en diversos contextos tecnológicos.

Para alcanzar este nivel de optimización, se ha llevado a cabo una exhaustiva exploración del espacio de hiperparámetros. Este proceso meticuloso ha permitido identificar la configuración óptima para cada uno de los parámetros clave del sistema. Durante esta fase de investigación, se ha tenido la oportunidad de examinar y validar empíricamente algunas de las propuestas más recientes en el campo, como las presentadas por Retsinas et al. (2022). En particular, se ha analizado en profundidad la efectividad del atajo CTC y la técnica de *max pooling* por columnas. Estos hallazgos han matizado estas propuestas, revelando que su eficacia está intrínsecamente ligada a la arquitectura específica del modelo. Concretamente, se ha descubierto que estas técnicas resultan beneficiosas únicamente cuando la última capa convolucional de la red neuronal posee una cantidad suficiente de filtros.

En el aspecto teórico de este trabajo, se ha realizado una extensa labor de recopilación y descripción de los fundamentos que sustentan todos los métodos empleados en la resolución de la tarea de reconocimiento de texto manuscrito. Para este fin, se ha utilizado como base el conjunto de datos IAM, ampliamente reconocido en el campo. Esta revisión teórica no se ha limitado a los métodos actuales, sino que ha abarcado una descripción detallada de la evolución histórica del campo del reconocimiento de escritura manuscrita. Este enfoque ha permitido contextualizar la contribución del presente trabajo dentro del marco más amplio del progreso científico en esta área.

Un hallazgo particularmente notable de esta investigación ha sido la identificación y caracterización de un fenómeno específico durante el proceso de entrenamiento de los modelos. Se ha observado un comportamiento muy característico en el que el entrenamiento cae en mínimos locales, un fenómeno al que se ha denominado metafóricamente como el «Valle de la Muerte». Esta designación refleja la naturaleza crítica y potencialmente paralizante de esta fase del entrenamiento.

Además de identificar este fenómeno, se ha realizado un descubrimiento significativo en relación con el atajo CTC. Las observaciones indican que esta técnica no solo contribuye a mejorar el rendimiento general del modelo, sino que también desempeña un papel crucial

en la prevención del estancamiento del entrenamiento en el mencionado Valle de la Muerte. Esta doble funcionalidad del atajo CTC representa un avance importante en la optimización de los procesos de entrenamiento de modelos para el reconocimiento de texto manuscrito.

También se ha observado que las métricas CER y WER se encuentran fuertemente relacionadas.

Finalmente se ha evaluado el desempeño de grandes modelos del lenguaje multimodales, los cuales obtienen unos resultados muy competitivos, aunque aún por detrás de los mejores algoritmos especializados.

6.1. Trabajo futuro

Sería interesante completar este estudio con la exploración de otras arquitecturas que han sido históricamente relevantes como es el caso de los HMM, las MDRNN y los transformadores. Aunque esta línea de investigación requerirá del acceso a una capacidad computacional mayor.

Con más capacidad computacional se podrían investigar modelos más grandes a los estudiados en el presente trabajo. E incluso realizar ajuste fino sobre algún LLM pequeño de código abierto. Cuyo entrenamiento podría complementarse con el uso de técnicas de aumento de datos o mediante conjuntos de datos sintéticos.

También sería interesante evaluar otros LLM comerciales para comparar si rinden tan bien como GPT-4o y probar a mejorar los resultados mediante técnicas de ingeniería de *prompt*.

Al probar modelos de distinto género permitiría poner a prueba la universalidad de la relación entre las métricas de CER y WER. En esta línea también sería interesante investigar otros conjuntos de datos.

Otra dirección inexplorada en este trabajo es el empleo de lexicones para reducir aún más los errores en las predicciones del modelo.

Por último, se requiere una investigación más detallada del fenómeno que se ha bautizado como el «Valle de la Muerte», en particular de cómo evitarlo.

7.Bibliografía

Almazán, J., Gordo, A., Fornés, A., & Valveny, E. (2014). Word spotting and recognition embedded attributes. *Transactions on Pattern Analysis and Machine* (pp. 2552-2566). IEEE.

Barrere, K., Soullard, Y., Lemaitre, A., & Coüasnon, B. (2021). Transformers for Historical Handwritten Text. *16th International Conference on Document Analysis and Recognition.* Lausanne, Switzerland: HAL science. https://hal.science/hal-03485262

Bertolami, R., & Bunke., H. (2008). Hidden Markov model-based methods for offline handwritten text line recognition. *Pattern Recognition, 41*(11), 3452–3460.

Bhunia, A. K., Das, A., Bhunia, A. K., Kishore, P. S., & Roy, P. P. (2019). Handwriting Recognition in Low-resource Scripts using Adversarial Learning. *Conference on* (pp. 4767-4776). IEEE.

Bluche, T. (2015). Deep neural networks for large vocabulary handwritten text recognition. *Ph.D. thesis*. Paris XI: Université Paris Sud.

Bluche, T. (2016). Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. *Advances in Neural Information Processing Systems*, (pp. 838–846).

Bluche, T., & Messina, R. (2017). Gated convolutional recurrent neural networks for multilingual handwriting recognition. *14th IAPR International Conference on Document Analysis and Recognition* (pp. 646-651). Kyoto, Japan: IEEE. https://doi.org/10.1109/ICDAR.2017.111

Bluche, T., Louradour, J., & Messina, R. (2017). Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. *14th IAPR International Conference on Document Analysis and Recognition.* Kyoto, Japan: IEEE. https://doi.org/10.1109/ICDAR.2017.174

Bluche, T., Ney, H., & Kermorvant, C. (2014). A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. *SLSP*.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory.* https://doi.org/10.1145/130385.130401

Canning, A., & Gardner, E. (1988). Partially connected models of neural networks. *Journal of Physics A: Mathematical and General, 21*, 3275-3284. https://doi.org/10.1088/0305-4470/21/15/016

Carbonell, M., Mas, J., Villegas, M., Fornés, A., & Lladós, J. (2019). End-to-end handwritten text detection and transcription in full pages. *International Conference on Document Analysis and Recognition Workshops.* Sydney, NSW, Australia: IEEE. https://doi.org/10.1109/ICDARW.2019.40077

Castro, D., Bezerra, B. L., & Valença, M. (2018). Boosting the Deep Multidimensional Long-Short-Term Memory Network for Handwritten Recognition Systems. *16th*

International Conference on Frontiers in Handwriting Recognition. Niagara Falls, NY, USA: IEEE. https://doi.org/10.1109/ICFHR-2018.2018.00031

Chen, Z., Wu, Y. Y., & Liu, C. (2017). Simultaneous script identification and hand-writing recognition via multi-task learning of recurrent neural networks. *14th IAPR International Conference on Document Analysis and Recognition.* Kyoto, Japan: IEEE.

Cho, K., van Merrienboer, B., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Association for Computational Linguistics*. https://doi.org/10.48550/arXiv.1406.1078

Chowdhury, A., & Vig, L. (2018). An efficient end-to-end neural model for handwritten text recognition. *arXiv*. https://doi.org/arXiv:1807.07965

Chung, J., & Delteil, T. (2019). A computationally efficient pipeline approach to full page offline handwritten text recognition. *International Conference on Document Analysis and Recognition Workshops.* Sydney, NSW, Australia: IEEE. https://doi.org/10.1109/ICDARW.2019.40078

Ciresan, D., Meier, U., Masci, J., Gambardella, L. M., & Schmidhube, J. (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, *2*, pp. 2278-2324.

Constum, T., Tranouez, P., & Paquet, T. (2024). DANIEL: A fast Document Attention Network for Information Extraction and Labelling of handwritten documents. *arXiv*. https://doi.org/arXiv:2407.09103

Coquenet, D., Chatelain, C., & Paquet, T. (2022). End-to-end handwritten paragraph text recognition using a vertical attention network. *Pattern Analysis and Machine Intelligence. 45*, pp. 508-524. IEEE. https://doi.org/10.1109/TPAMI.2022.3144899

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273-297. https://doi.org/10.1007/BF00994018

Davis, B., Morse, B., Price, B., Tensmeyer, C., Wigington, C., & Morariu, V. (2023). End-to-end document recognition and understanding with dessurt. In L. Karlinsky, & T. M. Nishino (Ed.), *Computer Vision – ECCV 2022 Workshops* (pp. 280-296). Cham, Germany: Springer Nature Switzerland.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological), 39*(1), 1-22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*. https://doi.org/10.48550/arXiv.1810.04805

Diaz, D. H., Qin, S., Ingle, R., & Y. (2021). Rethinking text line recognition models. *arXiv*. https://doi.org/arXiv:2104.07787

Doetsch, P., Kozielski, M., & Ney, H. (2014). Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition. *14th International Conference on Frontiers in Handwriting Recognition* (pp. 279–284). Greece: IEEE.

Dreuw, P., Doetsch, P., Plahl, C., & Ney, H. (2011). Hierarchical Hybrid MLP/HMM or rather MLP Features for a Discriminatively Trained Gaussian HMM; A Comparison for Offline Handwriting Recognition. *International Conference on Image Processing.*

Dutta, K., Krishnan, P., Mathew, M., & Jawahar, C. (2018). Improving CNN-RNN hybrid networks for handwriting recognition. In: 16th International Conference on Fron-tiers in Handwriting Recognition. *16th International Conference on Frontiers in Handwriting Recognition.* Niagara Falls, NY, USA: IEEE.

España-Boquera, S. C.-B., Gorbe-Moya, J., & Zamora-Martinez, F. (2011). Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models. *TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. 33.* IEEE.

Fujitake, M. (2024). DTrOCR: Decoder-only Transformer for Optical Character Recognition. *Winter Conference on Applications of Computer Vision.* Waikoloa, Hawaii: IEEE. https://doi.org/10.1109/WACV57701.2024.00784

Fukushima, K. (1969). Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements. *Transactions on Systems Science and Cybernetics, 5*(4), 322 - 333. https://doi.org/10.1109/TSSC.1969.300225

Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics, 36*(4), 193-202.

Grave, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. *Proceedings of the 23rd international conference on Machine learning.* Association for Computing Machinery. https://doi.org/10.1145/1143844.1143891

Graves, A., Fernández, S., & Schmidhuber, J. (2007). Multi-dimensional Recurrent Neural Networks. *Artificial Neural Networks – ICANN 2007.* Springer. https://doi.org/10.1007/978-3-540-74690-4_56

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2008). A Novel Connectionist System for Unconstrained Handwriting Recognition. *Transactions on Pattern Analysis and Machine Intelligence. 31*, pp. 855-868. IEEE. https://doi.org/10.1109/TPAMI.2008.137

Hammerla, N. Y., Vajda, T. P., & Fink, G. A. (2011). Towards Feature Learning for HMM-based Offline Handwriting. *First International Workshop on Frontiers in Arabic Handwriting Recognition.* https://doi.org/10.17877/DE290R-8202

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *International Conference on Computer Vision* (pp. 1026-1034). Santiago, Chile: IEEE. https://doi.org/10.1109/ICCV.2015.123

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*. https://doi.org/10.48550/arXiv.1207.0580

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, (pp. 448-456).

Kang, L., Riba, P., M. Rusiñol, A. F., & Villegas, M. (2020). Pay attention to what you read: nonrecurrent handwritten text-line recognition. *arXiv*. https://doi.org/arXiv:2005.13044

Kang, L., Toledo, J. I., Riba, P., Villegas, M., Fornés, A., & Rusinol, M. (2018). Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition. *German Conference on Pattern Recognition* (pp. 459-472). Springer.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2024). Large language models are zero-shot reasoners. *Proceedings of the 36th International Conference on Neural Information Processing Systems.* New Orleans, LA, USA: Curran Associates Inc.

Kozielski, M., Doetsch, P., & Ney, H. (2013a). Improvements in RWTH's system for off-line handwriting recognition. *12th International Conference on Document Analysis and Recognition.* Washington, DC, USA: IEEE.

Kozielski, M., Rybach, D., Hahn, S., Schluter, R., & Ney, H. (2013b). Open vocabulary handwriting recognition using combined word-level and character-level language models. *2013 International Conference on Acoustics, Speech and Signal Processing.* Vancouver, BC, Canada: IEEE. https://doi.org/10.1109/ICASSP.2013.6639275

Krishnan, P., Dutta, K., & Jawahar, C. (2016). Deep feature embedding for accurate recognition and retrieval of handwritten text. *International Conference*, (pp. 289-294).

Krishnan, P., Dutta, K., & Jawahar, C. (2018). Word spotting and recognition using deep embedding. *13th IAPR International Workshop on Document Analysis Systems.*

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. Burges, L. Bottou, & K. Q. Weinberger (Ed.), *Advances in Neural Information Processing Systems.*

Kumari, L., Singh, S., Singh Rathore, V. V., & Sharma, A. (2023). A comprehensive handwritten paragraph text recognition system: Lexiconnet. In Document Analysis and Recognition. *17th International Conference on Document Analysis and Recognition* (pp. 226-241). San José, California, USA: Springer.

Kumari, L., Singh, S., Singh Rathore, V. V., & Sharma, A. (2024). GatedLexiconNet: A Comprehensive End-to-End Handwritten Paragraph Text Recognition System. *arXiv*. https://doi.org/arXiv:2404.14062

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324.

Lecun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., . . . Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. In J. H. Oh, C. Kwon, & S. Cho (Ed.), *Neural networks: The statistical mechanics perspective* (pp. 261-276). World Scientific.

Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., . . . Wei., F. (2021). TrOCR: Transformer-based optical character recognition with pre-trained models. *arXiv*. https://doi.org/arXiv:2109.10282

Liwicki, M., Grave, A., & Bunke, H. (2012). Neural networks for handwriting recognition. In *Computationa lintelligence paradigms in advanced pattern classification* (pp. 5–24). Springer.

Louradour, J., & Kermorvant, C. (2014). Curriculum Learning for Handwritten Text Line Recognition. *11th IAPR International Workshop on Document Analysis Systems.* Tours, France: IEEE. https://doi.org/10.1109/DAS.2014.38

Luo, C., Zhu, Y., Jin, L., & Wang, Y. (2020). Learn to augment: Joint data augmentation and network optimization for text recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13746–13755). Seattle, WA, USA: IEEE. https://doi.org/10.1109/CVPR42600.2020.01376

Markou, K., Tsochatzidis, L., Zagoris, K., Papazoglou, A., Karagiannis, X., Symeonidis, S., & Pratikakis, I. (2021). A convolutional recurrent neural network for the handwritten text recognition of historical greek manuscripts.

Markov, A. A. (1913). An example of statistical investigation of the text Eugene Onegin concerning the connection of samples in chains. (In Russian.). *Bulletin of the Imperial Academy of Sciences of St. Petersburg, 7*(3), 153-162.

Marti, U., & Bunke, H. (2002). The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. *Journal on Document Analysis and Recognition, 5*, 39-46.

Michael, J., Labahn, R., Grüning, T., & Zöllner, J. (2019). Evaluating sequence-to-sequence models for handwritten text recognition. *International Conference on Document Analysis and Recognition* (pp. 1286–1293). IEEE.

Pastor-Pellicer, J., España-Boquera, S., Castro-Bleda, M. J., & Zamora-Martinez, F. (2015). A combined convolutional neural network and dynamic programming approach for text line network and dynamic programming approach for text line. *Proceedings of the International Conference on Document Analysis and Recognition.*

Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). Dropout improves Reccurent. *14th International Conference on Frontiers in Handwriting Recognition.*

Poulos, J., & Valle, R. (2017). Attention networks for image-to-text. *CoRR, abs/1712.04046*.

Poznanski, A., & Wolf, L. (2016). CNN-N-gram for handwriting word recognition. *Conference on Computer Vision and Pattern Recognition* (pp. 2305-2314). Las Vegas, NV, USA: IEEE.

Ptucha, R., Such, F. P., Pillai, S., Brockler, F., Singh, V., & Hutkowski, P. (2019). Intelligent character recognition using fully convolutional neural networks. *Patt. Recogn., 88*, 604–613. https://doi.org/10.1016/j.patcog.2018.12.017

Puigcerver, J. (2017). Are multidimensional recurrent layers really necessary for handwritten text recognition? *14th IAPR International Conference on Document Analysis and Recognition.* Kyoto, Japan: IEEE.

Puigcerver, J., Martin-Albo, D., & Villegas, M. (2016). *Laia: A deep learning toolkit for HTR*. GitHub repository: https://github.com/jpuigcerver/Laia

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., . . . Freitas, N. d. (2022). A Generalist Agent. *arXive*. https://doi.org/10.48550/arXiv.2205.06175

Retsinas, G., Sfikas, G., Gatos, B., & Nikou, C. (2022). Best Practices for a Handwritten Text Recognition System. In S. Uchida, E. Barney, & V. Eglin (Ed.), *Lecture Notes in Computer Science. 13237.* Springer. https://doi.org/doi.org/10.1007/978-3-031-06555-2_17

Retsinas, G., Sfikas, G., Nikou, C., & Maragos, P. (2021). Deformation-invariant networks for handwritten text recognition. *International Conference on Image* (pp. 949-953). IEEE.

Rubin, D. B. (1987). The Calculation of Posterior Distributions by Data Augmentation: Comment: A Noniterative Sampling/Importance Resampling Alternative to the Data Augmentation Algorithm for Creating a Few Imputations When Fractions of Missing Information Are Modest: The SI. *Journal of the American Statistical Association, 82*(398), 543–546. https://doi.org/10.2307/2289460

Sompolinsky, H. (1987). The theory of neural networks: The Hebb rule and beyond. *Heidelberg Colloquium on Glassy Dynamics* (págs. 485-527). Springer. https://doi.org/10.1007/BFb0057531

Stuner, B., Chatelain, C., & Paquet, T. (2016). Handwriting recognition using cohort of LSTM and lexicon verification with extremely large lexicon. *CoRR, abs/1612.07528*.

Sueiras, J., Ruiz, V., Sanchez, A., & Velez, J. (2018). *Neurocomputing, 289*, 119-128.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems. 2*, pp. 3104–3112. Montreal, Canada: MIT Press.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Conference on Computer Vision and Pattern Recognition.* (pp. 2818-2826). Las Vegas, NV, USA: IEEE.

Tarride, S., & Kermorvant, C. (2024). Revisiting N-Gram Models: Their Impact in Modern Neural Networks for Handwritten Text Recognition. *arXiv*. https://doi.org/arXiv:2404.19317

Tarride, S., Boillet, M., & Kermorvant, C. (2023). Key-value information extraction from full handwritten pages. In G. A. Fink, R. Jain, K. Kise, & R. Zanibbi (Ed.), *Document Analysis and Recognition* (pp. 185–204). Cham, Germany: Springer Nature Switzerland.

Tassopoulou, V., Retsinas, G., & Maragos, P. (2021). Enhancing handwritten text recognition with n-gram sequence decomposition and multitask learning. *25th International Conference on Pattern Recognition* (pp. 10555–10560). IEEE.

Tula, D., Paul, S., Madan, G., Garst, P., Ingle, R., & Aggarwal, G. (2023). Is it an i or an l: Test-time Adaptation of Text Line Recognition Models. *arXiv*. https://doi.org/arXiv:2308.15037

Vapnik, V. (1997). The Support Vector method. En W. Gerstner, A. Germond, M. Hasler, & J. Nicoud (Ed.), *Lecture Notes in Computer Science. 1327.* Springer. https://doi.org/10.1007/BFb0020166

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, (pp. 5998–6008).

Villanova-Aparisi, D., Tarride, S., Martínez-Hinarejos, C. D., Romero, V., Kermorvant, C., & Pastor-Gadea, M. (2024). Reading Order Independent Metrics for Information Extraction in Handwritten Documents. *arXiv*. https://doi.org/arXiv:2404.18664

Voigtlaender, P., Doetsch, P., & Ney, H. (2016). Handwriting Recognition with Large Multidimensional Long Short- Term Memory Recurrent Neural Networks. *15th International Conference on Frontiers.* Shenzhen, China: IEEE. https://doi.org/10.1109/ICFHR.2016.0052

Voigtlaender, P., Doetsch, P., Wiesler, S., Schlüter, R., & Ney, H. (2015). Sequence-discriminative training of recurrent neural networks. *International Conference on Acoustics, Speech and Signal Processing* (pp. 2100-2104). South Brisbane, QLD, Australia: IEEE. https://doi.org/10.1109/ICASSP.2015.7178341

Wang, T., Zhu, Y., Jin, L., Luo, C., Chen, X., Wu, Y., . . . Cai, M. (2020). Decoupled attention network for text recognition. *AAAI, 34*, 12216-12224.

Wick, C., Z¨ollner, J., & Gr¨uning, T. (2021). Transformer for handwritten text recognition using bidirectional post-decoding. *International Conference on Document Analysis* (pp. 112–126). Springer.

Wigington, C., Stewart, S., Davis, B., Barrett, B., Price, B., & Cohen, S. (2017). Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network. *14th IAPR International Conference on Document Analysis and Recognition* (pp. 639-645). Kyoto, Japan: IEEE.

Wigington, C., Tensmeyer, C., Davis, B., Barrett, W., Price, B., & Cohen, S. (2018). Start, follow, read: End-to-end full-page handwriting recognition. In V. Ferrari, C. S. M. Hebert, & Y. Wess (Ed.), *European Conference on Computer Vision* (pp. 372-388). Munich, Germany: Springer.

Williams, R. J., Hinton, G. E., & Rumelhart, D. E. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533-536. https://doi.org/10.1038/323533a0

Yang, C., Wang, X., Lu, Y., & H.Liu. (2024). LARGE LANGUAGE MODELS AS OPTIMIZERS. *International Conference on Learning Representations.* Vienna, Austria.

Yousef, M., & Bishop, T. E. (2020). OrigamiNet: Weakly-Supervised, Segmentation-Free, One-Step, Full Page Text Recognition by learning to unfold. *Conference on Computer Vision and Pattern Recognition* (págs. 14698–14707). Los Alamitos, CA, USA: IEEE.

Yousef, M., Hussain, K., & Mohammed, U. (2020). Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. *108*. https://doi.org/10.1016/j.patcog.2020.107482

Zamora-Martinez, F., Frinken, V., España-Boquera, S., Castro, M. J., Fischer, A., & Bunke., H. (2014). Neural network. *Pattern Recognition*, 1642-1652.

Zhang, Y., Nie, S., Liu, W., Xu, X., Zhang, D., & Shen, H. T. (2019). Sequence-to-sequence domain adaptation network for robust text image recognition. *Conference on Computer Vision and Pattern Recognition.* 2740-2749: IEEE.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2022). Large Language Models Are Human-Level Prompt Engineers. *Conference on Neural Information Processing Systems.* New Orleans, USA.

8. Apéndices

Apéndice I. Referencias del estado del arte

Artículo	*Arquitectura*	*Método*	*CER (%)*
(Graves et al., 2008)	RNN	CTC	**18,2**
(Voigtlaender et al., 2015)	RNN	CE	4,8
(España-Boquera et al., 2011)	HMM/MLP	CE	**9,8**
(Hammerla et al., 2011)	HMM/MLP	CE	30,6
(Dreuw et al., 2011)	HMM/MLP	CE	12,4
(Kozielski et al., 2013b)	HMM/MLP	CE	**8,2**
(Kozielski et al., 2013a)	HMM/MLP	CE	**5,1**
(Zamora-Martinez et al., 2014)	HMM/MLP	CE	7,6
(Doetsch et al., 2014)	HMM/MLP	CE	**4,7**
(Pastor-Pellicer et al., 2015)	HMM/MLP	CE	7,5
(Bluche et al., 2014)	HMM/RNN	CE	4,9
(Almazán et al., 2014)	SVM	NNS	11,27
(Krishnan et al., 2016)	CNN	CE	6,33
(Poznanski & Wolf, 2016)	CNN	CE	**3,44**
(Ptucha et al., 2019)	CNN	CTC	4,70
(Yousef & Bishop, 2020)	CNN	CTC	4,7
(Yousef et al., 2020)	CNN	CTC	4,9
(Liwicki et al., 2012)	MDRNN	CTC	18,2
(Louradour & Kermorvant, 2014)	MDRNN	CTC	17
(Pham et al., 2014)	MDRNN	CTC	5,1
(Bluche et al., 2017)	MDRNN	CTC	6,6
(Voigtlaender et al., 2016)	MDRNN	CTC	**3,5**
(Chen et al., 2017)	MDRNN	CTC	11,15
(Bluche, 2016)	MDRNN	Seq2Seq	5,5
(Bluche, 2015)	CRNN	CTC	7,3
(Puigcerver et al., 2016)	CRNN	CTC	6,7
(Stuner et al., 2016)	CRNN	CTC	4,77
(Puigcerver, 2017)	CRNN	CTC	4,4
(Bluche & Messina, 2017)	CRNN	CTC	**3,2**
(Wigington et al., 2017)	CRNN	CTC	6,07
(Krishnan et al., 2018)	CRNN	CTC	9,78
(Castro et al., 2018)	CRNN	CTC	6,64
(Dutta et al., 2018)	CRNN	CTC	5,7
(Wigington et al., 2018)	CRNN	CTC	6,4
(Bhunia et al., 2019)	CRNN	CTC	5,94
(Carbonell et al., 2019)	CRNN	CTC	15,6

(Chung & Delteil, 2019)	CRNN	CTC	8,5
(Wang et al., 2020)	CRNN	CTC	6,4
(Luo et al., 2020)	CRNN	CTC	3,75
(Tassopoulou et al., 2021)	CRNN	CTC	5,18
(Markou et al., 2021)	CRNN	CTC	6,14
(Retsinas et al., 2021)	CRNN	CTC	4,55
(Coquenet et al., 2022)	CRNN	CTC	4,45
(Retsinas et al., 2022)	CRNN	CTC	4,62
(Kumari et al., 2023)	CRNN	CTC	3,24
(Kumari et al., 2024)	CRNN	CTC	**2,27**
(Poulos & Valle, 2017)	CRNN	Seq2Seq	16,9
(Sueiras et al., 2018)	CRNN	Seq2Seq	8,8
(Chowdhury & Vig, 2018)	CRNN	Seq2Seq	8,1
(Kang et al., 2018)	CRNN	Seq2Seq	6,9
(Zhang et al., 2019)	CRNN	Seq2Seq	8,5
(Michael et al., 2019)	CRNN	Seq2Seq	4,87
(Villanova-Aparisi et al., 2024)	CRNN	Seq2Seq	5
(Kang et al., 2020)	CNN/Transformador	Seq2Seq	4,67
(Diaz et al., 2021)	CNN/Transformador	Seq2Seq	**2,75**
(Wick et al., 2021)	CNN/Transformador	Seq2Seq	5,8
(Barrere et al., 2021)	CNN/Transformador	Seq2Seq	5,07
(Davis et al., 2023)	CNN/Transformador	Seq2Seq	4,8
(Tarride et al., 2023)	CNN/Transformador	Seq2Seq	4,3
(Tarride & Kermorvant, 2024)	CNN/Transformador	Seq2Seq	4,38
(Tula et al., 2023)	CNN/Transformador	CTC	5,4
(Constum et al., 2024)	CNN/Transformador	CTC	4,38
(Li et al., 2021)	Transformador	CTC	2,89
(Fujitake, 2024)	Transformador	CTC	**2,38**

Tabla 21: Resultados para la tarea IAM ordenados cronológicamente y agrupados por arquitectura, método. Se han marcado en negro los resultados récord en el estado del arte.

Apéndice II. Repositorio de código

La implementación del código empleado en este trabajo puede encontrarse en el repositorio https://github.com/Tyrian0/IAM_a_reader. El repositorio contiene los siguientes documentos:

- data/lines.txt: Etiquetas para las líneas con los espacios correctamente asignados.
- callbacks.py: Contiene la clase SavePredictions, que permite guardar las predicciones para cada época del entrenamiento.
- ctc_layer.py: Contiene las clases para calcular las pérdidas CTC.
- hwr_dataset.py: Contiene la clase IAM_dataset que permite cargar y trabajar con el conjunto de datos IAM.
- utils.py: Contiene los métodos necesarios para definir, entrenar y visualizar los modelos.
- requirements.txt: Contiene las versiones de los módulos utilizados.