Portada (institucionalmente establecida)

# Resumen

(máximo de 5 páginas, incluyendo al final Palabras clave y Códigos UNESCO)

# Contenido

[1 Resumen 2](#_Toc80356329)

[2 Contenido 3](#_Toc80356330)

[3 Introducción 4](#_Toc80356331)

[3.1 Antecedentes 4](#_Toc80356332)

[3.1.1 Reconocimiento de ecuaciones 4](#_Toc80356333)

[3.1.2 Herramientas de solución simbólica de ecuaciones 10](#_Toc80356334)

[3.2 Justificación 12](#_Toc80356335)

[4 Objetivos 13](#_Toc80356336)

[5 Metodología 14](#_Toc80356337)

[5.1 Análisis del estado del arte 14](#_Toc80356338)

[5.2 Elección de la arquitectura 14](#_Toc80356339)

[5.3 Seguimiento de mejores prácticas 15](#_Toc80356340)

[5.4 Elección de licencia de código abierto 15](#_Toc80356341)

[6 Resultados y discusión 16](#_Toc80356342)

[6.1 Arquitectura 16](#_Toc80356343)

[*6.2* *Front end* 16](#_Toc80356344)

[*6.3* *Back end* 17](#_Toc80356345)

[6.4 Segmentación 17](#_Toc80356346)

[6.5 Clasificación 21](#_Toc80356347)

[6.6 Análisis estructural 23](#_Toc80356348)

[6.7 Resolución 25](#_Toc80356349)

[6.8 Distribución de eventos 25](#_Toc80356350)

[6.9 Resultados de los tests 25](#_Toc80356351)

[7 Conclusiones 26](#_Toc80356352)

[8 Líneas futuras 27](#_Toc80356353)

[9 Bibliografía 29](#_Toc80356354)

[10 Planificación temporal y presupuesto 32](#_Toc80356355)

[11 Índice de figuras (opcional) 33](#_Toc80356356)

[12 Índice de tablas (opcional) 34](#_Toc80356357)

[13 Abreviaturas 35](#_Toc80356358)

# Introducción

## Antecedentes

En la actualidad existen multitud de proyectos, herramientas y librerías que cubren tanto el campo del reconocimiento de ecuaciones mediante inteligencia artificial como la resolución de éstas mediante cálculo numérico y simbólico.

A continuación, se incluye explicación de los principales desarrollos que cubren ambos aspectos, así como ejemplos de interés de ambos.

### Reconocimiento de ecuaciones

La investigación en reconocimiento de notación matemática tiene más de cinco décadas de recorrido [1]. Las soluciones desarrolladas se pueden clasificar según varios parámetros: naturaleza de las entradas, flujo de reconocimiento y salidas.

#### Clasificación según la naturaleza de los datos de entrada

En el campo de reconocimiento de ecuaciones se pueden distinguir dos enfoques distintos que difieren en el método de recogida de datos.

El primer método denominado *online* recoge datos de tabletas digitalizadoras y similares. El método de reconocimiento se basa en los trazos realizados con el bolígrafo digital y utilizan, entre otros datos de las direcciones de los trazados y el orden (marcas de tiempo) en el que éstos se realizan.

La gran ventaja de estos métodos es que cuentan con una gran cantidad de información añadida para realizar las predicciones. Por el contrario, generan la necesidad de tener dispositivos para la recogida de los datos y, por tanto, no se pueden utilizar en texto escrito en formato analógico.

Por otro lado, el método *offline* utiliza imágenes como fuente de datos lo que lo convierte en un método más flexible. Es relativamente sencillo convertir registros de *online* a *offline*. Sin embargo, el problema de inferir los trazos y marcas de tiempo para convertir registros *offline* a *online* constituiría un problema a resolver en sí mismo.

**Reconocimiento *online***

El reconocimiento *online* ha sido ampliamente más investigado que el reconocimiento *offline*. Como referente cabe destacar el CROHME (Competición de reconocimiento *online* de expresiones matemáticas manuscritas, por sus siglas en inglés). Se han llevado a cabo 6 competiciones de CROHME en 2011 [2], 2012 [3], 2013 [4], 2014 [5], 2016 [6] y 2019 [7]. En todas ellas se han incluido categorías de reconocimiento de expresiones *online* mientras que solo la última introdujo categoría *offline.*

Actualmente se pueden ver varias implementaciones, tanto comerciales como de código abierto de sistemas que compitieron en las distintas ediciones del CROHME. Entre ellas WIRIS MathType [[1]](#footnote-1) y SESHAT [8] y MyScript Calculator

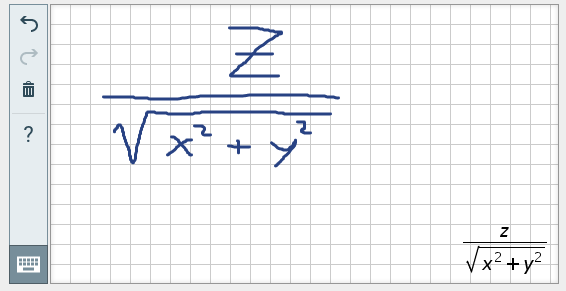


Figura : Reconocimiento «online» WIRIS MathType

La aplicación de WIRIS MathType dispone de versión de escritorio y es incrustable en diversos formatos y *frameworks* web.

SESHAT es parte del resultado de una tesis doctoral llevada a cabo en la Universidad Politécnica de Valencia. Es un desarrollo de código abierto bajo licencia GNU alojado en GitHub[[2]](#footnote-2). También hay una implementación disponible en formato web[[3]](#footnote-3).

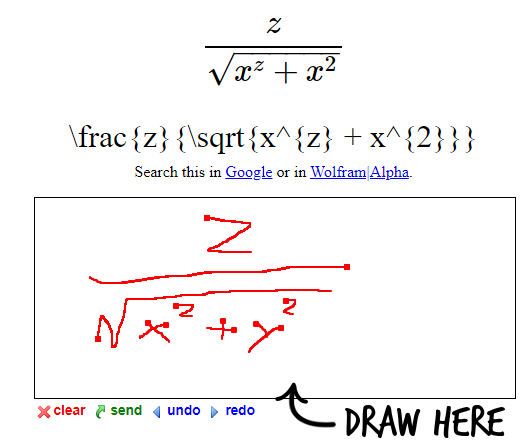


Figura : Reconocimiento «online» SESHAT

MyScript[[4]](#footnote-4) Calculator es una app móvil (de pago) disponible en iOS y Android. La aplicación es capaz de resolver ecuaciones sencillas.

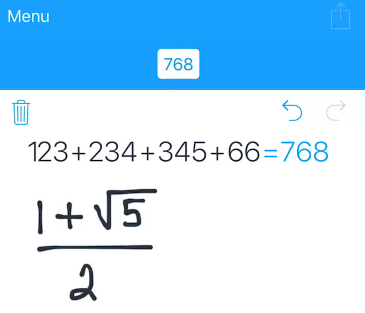


Figura Reconocimiento «online» MyScript

**Reconocimiento *offline***

La entrada de datos tipo *offline* si bien tiene también una larga trayectoria las publicaciones al respecto son más limitadas y sus resultados en la última competición del CROHME inferiores. Esto es debido principalmente a la ventaja que proporciona a nivel estructural conocer las marcas de tiempo de los trazos realizados.

Entre las implementaciones disponibles encontramos Mathpix Snip[[5]](#footnote-5): Mathpix Snip dispone de aplicaciones de escritorio para Windows, Linux y MacOS, así como apps para Android y iOS.

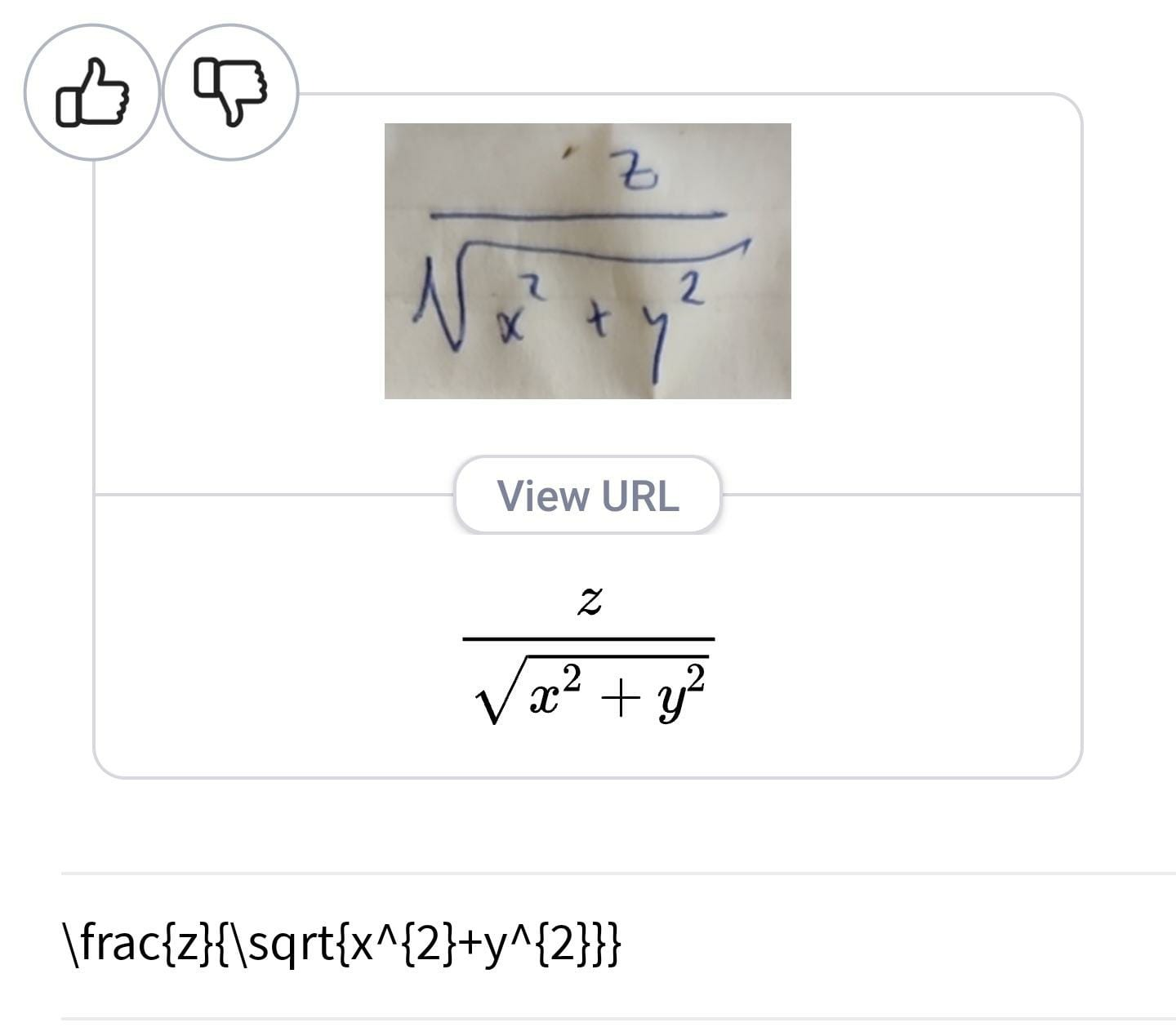


Figura : Reconocimiento «offline» Mathpix Snip

#### Clasificación según el flujo de resolución

El reconocimiento de expresiones matemáticas se puede dividir en tres procesos principales: segmentación, donde se divide la imagen en los distintos símbolos que la componen; clasificación, donde se identifica a qué clase pertenece cada símbolo; y reconocimiento de la estructura, donde se buscan las relaciones entre los distintos símbolos.

Las distintas soluciones propuestas se pueden clasificar según el orden en el que estas fases se ejecutan.

Así, es posible distinguir soluciones *top-down* [9] en las que se realiza primero el análisis estructural para después utilizar dicha información en la segmentación y clasificación de símbolos.

En las soluciones *bottom-up* se realiza primero la segmentación, seguida de la clasificación y por último el análisis estructural.

Por último, algunas soluciones realizan simultáneamente el análisis estructural y la clasificación de símbolos [10].

Dentro de cada una de estas fases existen enfoques distintos.

En la fase de segmentación se encuentran soluciones basadas en la proyección recursiva de las imágenes (ya sean los datos de entrada o generadas a partir de los mismos) sobre los ejes vertical y horizontal para encontrar las divisiones [11]. Otros enfoques utilizan los cuadros delimitadores para separar los símbolos [12].

En algunos casos se obtienen varias posibles segmentaciones, a las que luego se aplican el resto de fases para seleccionar la segmentación que produzca una mayor probabilidad de acierto [13]. De hecho, en las competiciones de CROHME celebradas entre 2011 y 2014 la mayoría de los sistemas participantes utilizaba la información obtenida de la clasificación de símbolos para guiar el proceso de segmentación [14].

En la fase de clasificación se pueden encontrar distintos tipos de clasificadores. Los más comunes suelen ser:

* Modelos de *machine learning* (aprendizaje automático), como KNN (k vecinos cercanos), arboles de decisión, SVM (Máquinas de vectores de soporte)…
* Modelos de redes neuronales como CNN (redes neuronales convolucionales) para datos *offline* y RNN (redes neuronales recurrentes) para el caso de datos *online*
* Modelos probabilísticos como los HMM (modelos ocultos de Márkov) [10].

En algunos casos la clasificación se realiza en conjunto con el análisis estructural, como se discute en la siguiente sección

En los casos en los que la segmentación se ha hecho de forma independiente a la clasificación, la segmentación se suele utilizar para construir un árbol de relaciones con el que luego se construye la secuencia de salida.

Se pueden encontrar casos donde el método de análisis estructural se realiza con métodos de gramática libre de contexto probabilística. Estos métodos ampliamente utilizados en áreas similares como procesamiento de lenguaje natural (NLP) o traducción de textos, definen conjuntos de categorías y símbolos y unas reglas de relaciones entre unos y otros para formar combinaciones válidas según dichas reglas. Las reglas de relaciones llevan asignadas una probabilidad de ocurrencia. Al predecir una secuencia de símbolos se pueden utilizar algoritmos como el CYK (Cocke-Younger-Kasami) para comprobar si la secuencia es válida según las reglas de la gramática, así como para comprobar la probabilidad que tiene la secuencia de ser correcta.

El problema principal de dichos métodos es que son computacionalmente caros. El algoritmo CYK tiene una complejidad de O(n3), donde n es la longitud de la secuencia predicha. Aunque existen técnicas como la programación dinámica que elimina posibles caminos, reduciendo la complejidad, estos modelos siguen resultando lentos cuando se quieren hacer predicciones en «tiempo real».

Para los modelos que realizan todas las fases en conjunto se suelen utilizar modelos de codificador-descodificador con capas de atención. Estos mecanismos son ampliamente utilizados en conversiones de tipo secuencia a secuencia donde los tamaños de las secuencias de entrada y salida son variables.

La función del codificador es extraer las características principales de los datos de entrada codificándolas como un vector de longitud –generalmente– fija. A continuación, el decodificador transforma el vector codificado en una secuencia, de longitud variable.

En estos casos el codificador suele ser una red neuronal convolucional para datos *offline*. Existen multitud de redes utilizadas en estos modelos. Aunque varía según el set de datos utilizados para el entrenamiento, en la siguiente figura se pueden ver los nombres de los distintos tipos de redes convolucionales utilizados en las últimas décadas.

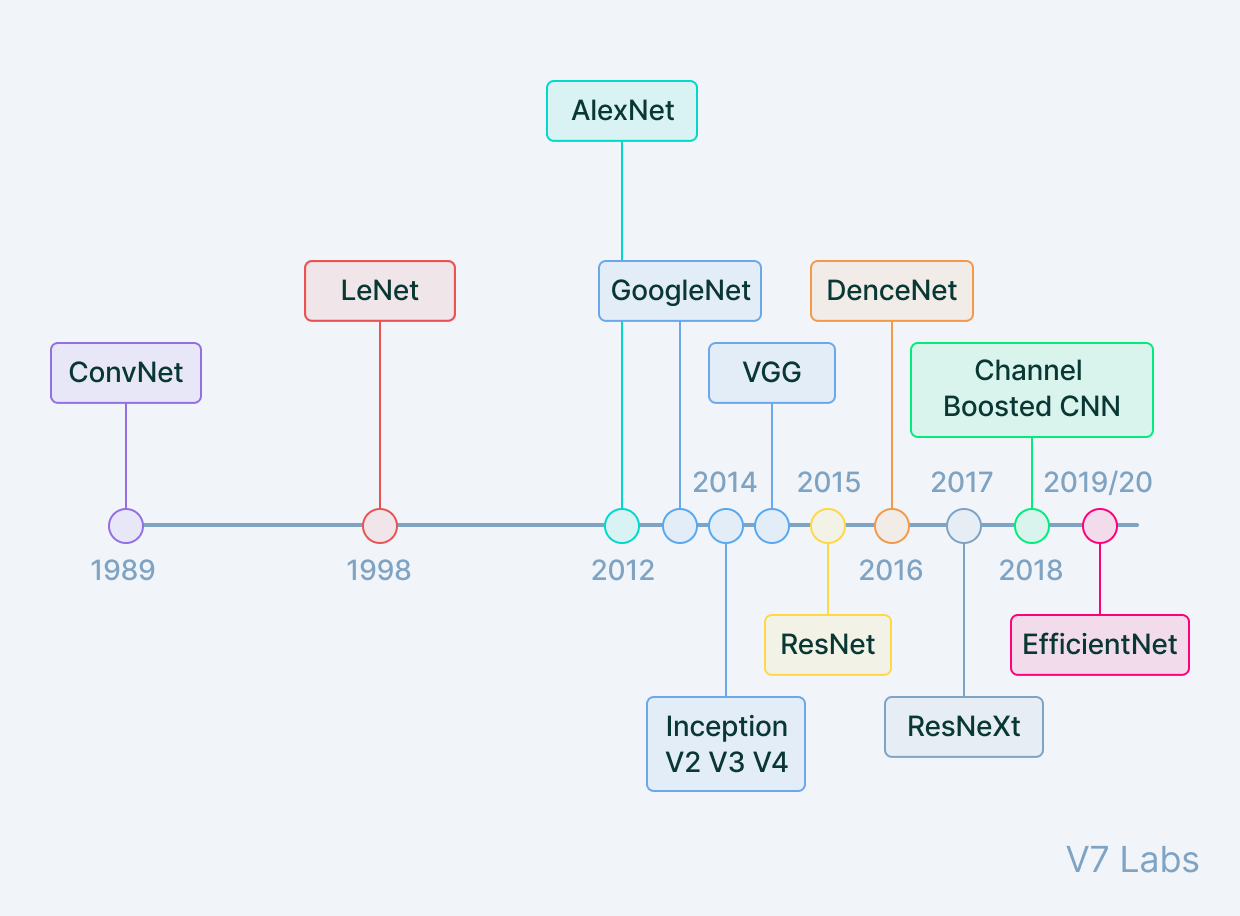


Figura : Redes neuronales convolucionales según año de invención[[6]](#footnote-6)

Para datos *online* se suelen utilizar redes neuronales recurrentes (RNN) ampliamente utilizadas en el reconocimiento de secuencias. Para predecir cada símbolo o relación espacial, las redes neuronales recurrentes tienen en cuenta las predicciones anteriores. Para sopesar las diferentes predicciones anteriores según importancia (en general ligadas con la cercanía) se utilizan estructuras llamadas células de memoria a corto y largo plazo (LSTM). Para tener en cuenta las predicciones de los símbolos anteriores y posteriores se utilizan estructuras bidireccionales, que replican las LSTM en ambos sentidos y después sopesan la importancia de la predicción obtenida en cada sentido. Estas estructuras se llaman memoria a corto y largo plazo bidireccionales (BLSTM). Otro tipo de redes con comportamiento similar a las LSTM son las unidades recurrentes cerradas (GRU). Existen soluciones que implementan ambos tipos.

También se aplican mecanismos de atención, que simulan el comportamiento de los humanos, que realizan las predicciones atendiendo a zonas concretas de los datos de entrada, permitiendo resolver secuencias grandes como suma de secuencias pequeñas, manteniendo la precisión y sin elevar excesivamente el coste computacional.

Por último, algunos sistemas usan mecanismos de cobertura para asegurar que no se interprete múltiples veces un mismo símbolo (por ejemplo, si apareciera en dos zonas de atención adyacentes) y no queden símbolos sin predecir (por no aparecer en ninguna zona de atención o aparecer divididos entre varias).

#TODO: Añadir citas de todos los sistemas mencionados

### Herramientas de solución simbólica de ecuaciones

Las herramientas de cálculo simbólico también tienen una historia que abarca varias décadas. La primera versión de Wolfram Mathematica fue publicada en 1988 y la primera versión de Maple en 1982. Desde entonces numerosos avances y alternativas has ido surgiendo.

La capacidad de resolución de ecuaciones diferenciales aportada por las versiones actuales de Maple (2021.1) y Mathematica (12.3.1) son superiores a las de otros sistemas de cálculo simbólico.

Por ejemplo, podemos ver los resultados obtenidos por ambas soluciones para las 1940 ecuaciones diferenciales del libro de Kamke [15] para ambos lenguajes [16] y del subset Kamke 88 (de ecuaciones diferenciales ordinarias de primer orden en forma normal) para la librería de Python SymPy [17] en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Sistema | Versión | Resultado (%) |
| Mathematica | 12.3.1 | 88,04 |
| Maple | 2021.1 | 88,06 |
| SymPy | >=1.6.2 | 39,77 |

Para SymPy se han tenido en cuenta solo los resultados que no fueron expresados en términos de potencias que aumentarían el ratio a 68,18%. Para la versión de SymPy se ha supuesto que tenía que ser menor o igual a la última versión publicada [18] antes de la publicación de los resultados.

A pesar de los mejores resultados de los dos sistemas estos poseen el problema de ser programas bajo licencia, lo cual aumenta los costes y reduce la facilidad de integración y personalización a la hora de ser integradas en sistemas mayores.

La selección de SymPy como comparación se debe a su gran popularidad como alternativa alcanzando las 8.500 estrellas (*stars*) y las 1.300 bifurcaciones (*forks*) en GitHub [19], con mucha diferencia con otras alternativas populares como SageMath u otros proyectos de código abierto. SymPy además cuenta con la ventaja de ser una librería de Python, por lo que para integrarla en el desarrollo de aplicaciones escritas en este lenguaje solo hace falta importarla.

Python además cuenta con grandes ventajas sobre otros lenguajes. En la siguiente tabla se pueden ver los resultados de la encuesta sobre lenguajes de StackOverflow (el foro más popular de programación) de 2020, en la que participaron más de 65.000 desarrolladores, con respecto a Python [20]:

|  |  |  |
| --- | --- | --- |
| Pregunta | Porcentaje | Posición |
| Lo usan y quieren seguir | 66,7 | 3º |
| Quieren aprender | 30 | 1º |

También se puede observar el número de *pull requests* (peticiones de validación) en el segundo trimestre de 2021 [21]:

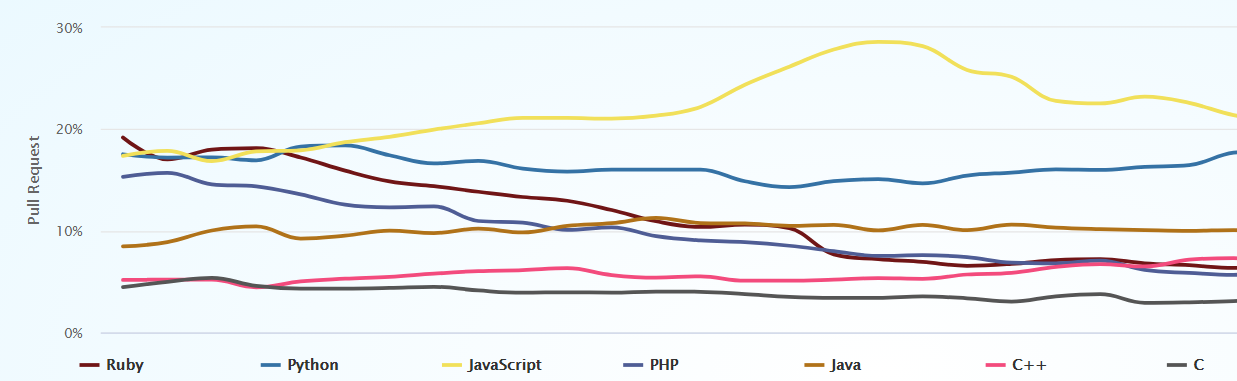


Figura 6: Peticiones de validación en GitHub por lenguaje en el segundo trimestre de 2021

Donde se puede ver que Python es el segundo lenguaje. O el número de búsquedas en Google en la última década [22]:

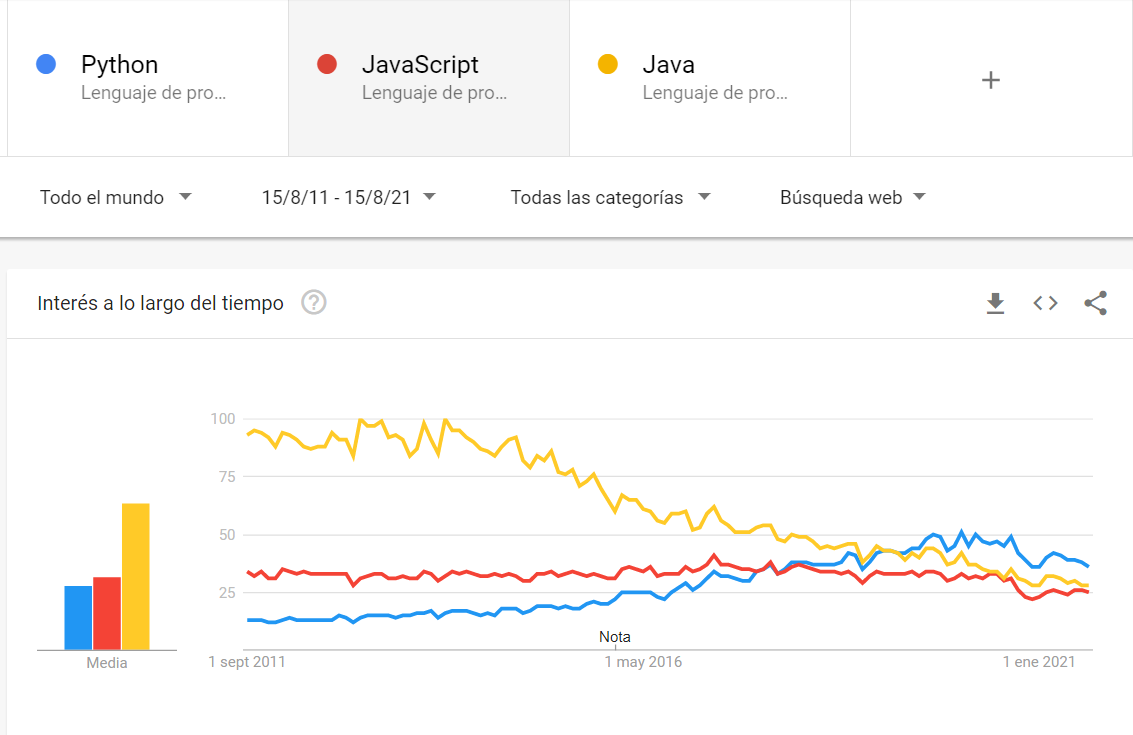


Figura 7: Búsquedas en Google de lenguajes de programación en el periodo 2011-2021

Donde se puede observar el crecimiento de Python hasta la primera posición.

Existen, además, estudios que indican la posibilidad de abordar el problema de resolución de ecuaciones diferenciales mediante el empleo de sistemas de *Deep learning* [23], por ejemplo, del tipo secuencia a secuencia (codificador-descodificador) como los que se han comentado en los apartados de reconocimiento de ecuaciones.

Además, algunos estudios señalan el mejor rendimiento y la resolución de ecuaciones diferenciales que Mathematica o Maple no son capaces de resolver en entornos de tiempo restringido. En entornos en «tiempo real» este factor puede ser crítico. En la siguiente tabla se pueden comprobar los resultados obtenidos en uno de los citados estudios [24] con un *dataset* de 5000 ecuaciones:

|  |  |  |
| --- | --- | --- |
| Sistema | % ODE resuelto primer orden | % ODE resuelto segundo orden |
| Mathematica[[7]](#footnote-7) | 77,2 | 61,6 |
| Red neuronal | 97,0 | 81,0 |

Otra ventaja de utilizar Python para el desarrollo del *solver* de ecuaciones diferenciales es la posibilidad de implementar dichos métodos también en Python, integrándolos de forma sencilla en la aplicación. De hecho, existen artículos que explican cómo crear redes neuronales con este propósito [25] en Python.

## Justificación

El proyecto pretende abarcar el problema completo de la resolución de ecuaciones diferenciales desde la captación de los datos de entrada hasta la generación del código LaTeX con la solución, pasando por la generación del código LaTeX de la ecuación.

Si bien se ha visto que existen herramientas que abarcan todas las partes del problema por separado no existe ninguna solución de extremo a extremo.

Por otro lado, parte de las herramientas que solucionan alguna fase del ciclo del problema son herramientas bajo licencia. El presente proyecto pretende ser un proyecto de código abierto.

Muchas de las referencias expuestas en los anteriores apartados son proyectos de gran calidad, que, sin embargo, se quedan en lo meramente académico al no contar con una herramienta que permita que sean utilizados en la resolución de problemas.

El proyecto, así, constituye una oportunidad para que dichos módulos puedan integrarse en una solución completa sin necesidad de realizar desarrollos adicionales en el resto de partes (interfaz de usuario, reconocimiento o resolución).

# Objetivos

1. Crear una aplicación capaz de tomar fotografías de ecuaciones diferenciales y resolverlas.
2. Escribir código que siga las mejores prácticas de programación para asegurar la mantenibilidad del proyecto.
3. Asegurar la escalabilidad de la aplicación mediante el diseño de un proyecto modular que permita balancear la carga entre los distintos módulos.
4. Adquirir una visión global de la administración de sistemas.

# Metodología

En el desarrollo del proyecto se ha llevado a cabo una metodología AGILE en la que se han realizado *sprints* para el desarrollo de los distintos módulos.

Con el objetivo de mantener una arquitectura modular en mente, tras una fase de análisis del estado del arte global, se han realizado paquetes de trabajo de análisis de las soluciones disponibles y elección de la más adecuada independientes para cada elemento de la arquitectura.

Para asegurar la independencia de los módulos se han diseñado interfaces abstractas, agnósticas con respecto a las tecnologías empleadas, de modo que estas fueran intercambiables en caso de necesidades del proyecto en el presente o el futuro.

## Análisis del estado del arte

El análisis del estado del arte se ha llevado a cabo mediante la lectura de las publicaciones que abarcaban los desarrollos principales.

En el caso del reconocimiento de ecuaciones la existencia de una asociación (ICDAR) que recoge el estado del arte del mismo y realiza concursos (CROHME) periódicamente con el afán de poder unificar *datasets* para poder comparar los resultados de los distintos modelos ha guiado ampliamente la búsqueda de información.

El objetivo de dicha búsqueda no era solamente conocer los mejores sistemas, lo cual señaliza el objetivo al que se puede aspirar y da una idea de la complejidad del tema. También se buscaba conocer los distintos enfoques. Conocer las distintas técnicas empleadas permite abrir la opción de realizar diversos desarrollos, que desde el punto de vista docente y pedagógico enriquecen el proyecto.

En el caso de la resolución de ecuaciones, la existencia de dos herramientas comerciales (Mathematica y Maple) con varias décadas de recorrido genera un marco de referencia con el que comparar las soluciones de código abierto. En este sentido, se abre también el camino de colaborar en las soluciones de código abierto para mejorar las soluciones disponibles de manera que se acerquen a los resultados proporcionados por las herramientas comerciales.

## Elección de la arquitectura

La elección de la arquitectura pretende generar independencia entre los módulos para que se puedan cambiar las piezas y generar desarrollos independientes en los que los colaboradores puedan aportar mejoras al módulo de su elección sin necesidad de comprender ni modificar el resto de módulos.

Por otro lado, la solución de la arquitectura necesitaba ser escalable, para permitir un número creciente de usuarios sin necesidades de cambios estructurales a nivel de código.

Por último, se necesitaban elegir estructuras que permitieran su alojamiento en la nube sin coste para poder disponer de una versión de demostración accesible al público, y así cumplir el objetivo de proporcionar una herramienta accesible al público sin licencia.

## Seguimiento de mejores prácticas

Al tratarse de una aplicación con aspiración a convertirse en un proyecto de código abierto con múltiples colaboradores, el seguimiento de las mejores prácticas de programación para atraer la atención y las ganas de los posibles colaboradores.

#TODO: Mejores prácticas de la arquitectura.

En cuanto al lenguaje, Python cuenta con la ventaja de que las mejores prácticas respecto a estilo están definidas en unos documentos oficiales llamados PEP (propuestas de mejora de Python, por sus siglas en inglés). En este sentido se pueden destacar las propuestas seguidas en este desarrollo:

* PEP 20 [26]: El zen de Python (sobre los principios que guían el estilo en Python)
* PEP 8 [27]: Sobre el estilo general (en particular importaciones, indentación, comentarios y nombres).
* PEP 257 [28]: Sobre *doctrings* (cadenas de caracteres para la documentación de funciones, métodos, clases y módulos).
* PEP 484 [29]: Sobre las anotaciones de tipos.
* PEP 526 [30]: Sobre las anotaciones de variables.

#TODO: Añadir trozos de código que cumplan con las convenciones

Además, para el diseño de la estructura de archivos y carpetas se ha utilizado la estructura recomendada en RealPython [31].

#TODO: Añadir imagen de la estructura de archivos

## Elección de licencia de código abierto

Se elige la licencia GNU GLP v3 para asegurar que el proyecto continúa siendo de código abierto. La licencia por defecto es el copyright, por lo que para compartir el código de forma efectiva se debe elegir una licencia que sustituya a la licencia por defecto.

# Resultados y discusión

(incluyendo la valoración de impactos y de aspectos de responsabilidad legal, ética y profesional relacionados con el trabajo)

## Arquitectura

Para la arquitectura se ha elegido seguir el modelo de arquitectura orientada a eventos. Esta arquitectura permite manejar los flujos de trabajo de forma asíncrona y distribuida. Así, cada parte del sistema escucha los eventos que contienen los datos de entrada necesarios para sus procesos.

La arquitectura orientada a eventos permite escalar horizontalmente las aplicaciones según necesidades. Por ejemplo, si el *solver* de ecuaciones mediante cálculo simbólico requiere de tiempos más largos para encontrar las soluciones se pueden crear más instancias de este módulo que de las demás, no bloqueando los flujos de ejecución del resto de partes.

Esta arquitectura también permite tener un alto nivel de disponibilidad y tolerancia al fallo. Al tener la posibilidad de duplicar instancias en distintos servidores tanto a nivel de localización como de proveedor se reduce la posibilidad de fallo, pues el sistema de distribución de mensajes se encarga de distribuir la carga de trabajo entre las instancias que se encuentran disponibles.

Por otro lado, la arquitectura permitiría calcular las soluciones por varios métodos simultáneamente, dejando la opción de elegir la mejor de las soluciones propuestas o incluir un sistema de ensamblado en el futuro.

Se ha elegido la arquitectura que maximiza el número de módulos para poder acoplar todos los desarrollos futuros. De esta forma se ha elegido implementar un sistema que separa las fases de segmentación, clasificación, análisis estructural y resolución para desarrollar el conjunto de eventos que comunican cada fase con la siguiente. Así, si en el futuro se desarrollara un módulo que realizara la segmentación y la clasificación simultáneamente bastaría con que escuchara a los eventos de peticiones de segmentación y emitiera directamente eventos de clasificación completada, omitiendo los eventos de clasificación.

#TODO: Añadir imágenes y gráficas sobre la estructura de la arquitectura

## *Front end*

Para el *Front end* se ha elegido utilizar Next.JS. Un *framework* de JavaScript basado en React.js, uno de los *frameworks* más populares en programación web mantenido por Facebook, así como una gran comunidad de código libre. En cuanto a desarrollo web JavaScript es el líder indiscutible. Además, es de los lenguajes más utilizados como se puede ver en las cifras de GitHub (Figura 6) y Google (Figura 7). En la encuesta de StackOverflow se pueden ver los siguientes datos sobre JavaScript y React.js [21]:

|  |  |  |  |
| --- | --- | --- | --- |
| Tecnología | Pregunta | % | Puesto |
| JavaScript | Desarrolladores que lo utilizan | 67,7 | 1º |
| JavaScript | Desarrolladores profesionales que lo utilizan | 69,7 | 1º |
| React.js | Desarrolladores que lo utilizan | 35,9 | 2º |
| React.js | Desarrolladores profesionales que lo utilizan | 36,8 | 2º |

Además, Next.js combina carga en servidor y en cliente para poder proteger los datos sensibles, como datos de acceso y conexión con otros sistemas. La carga en servidor permite un mejor indexado en los buscadores.

#TODO: Añadir imágenes de la interfaz

## *Back end*

En el *back end* se ha utilizado Python. Las numerosas ventajas de Python como lenguaje de programación, así como la disponibilidad de SymPy como librería de calculo simbólico han sido comentadas a lo largo del texto. Para conectar con el *front end* se ha utilizado el framework web Flask. Uno de los dos (junto con Django) *frameworks* web más utilizados. Flask es considerado un *microframework* porque tiene las funcionalidades necesarias para la comunicación con el *front end* sin añadir funcionalidades innecesarias que pueden afectar al rendimiento del sistema.

El *back end* está dividido en distintos módulos (6.4, 6.5, **¡Error! No se encuentra el origen de la referencia.** y 6.8) que se comunican mediante eventos gestionados por el bróker de mensajería (6.8).

## Segmentación

El módulo de segmentación está basado en proyecciones recursivas sobre los ejes horizontal y vertical inspirado en el trabajo de C. Faure y Zi-Xiong Wang [32]. Esto permite dividir la estructura de fuera a dentro. Además, durante la segmentación se va generando un árbol de relaciones que luego permite realizar el análisis estructural para reconstruir la ecuación.

Para casos como la raíz cuadrada donde las proyecciones sobre los ejes no aportan información se ejecuta una operación de extracción de máscara basada en la extracción del símbolo cuyo cuadro delimitador abarca partes de los cuadros del resto de símbolos.

En la siguiente figura se puede ver la estructura de clases:

La clase principal de la segmentación (XYSegmentationResults) contiene la estructura completa de la imagen segmentada. En su instanciación recibe la imagen original y empieza el flujo de la segmentación. Durante esta se generan niveles de segmentación (SegmentationLevel), donde el primero es la imagen inicial y cada nivel se construye a partir del anterior. Para ello, los niveles se componen de grupos de segmentación (SegmentationGroup). Cada uno de estos grupos contiene la operación que ha generado el grupo respecto a la imagen de la que parte (proyección sobre ejes, o extracción de máscara), la lista de las imágenes que resultan de la operación, y, para el caso de la proyección sobre el eje de abscisas se guardan las posiciones relativas entre los símbolos en forma de niveles. También se dispone de una operación nula para los nodos terminales a los que se llega en un menor número de pasos.

A continuación, se añade el extracto de código que controla el flujo de la segmentación:

def \_\_division\_step(self):

    for group\_index, image\_group in enumerate(segmentation\_groups):

        for image\_index, image in enumerate(image\_group):

            contours, \_ = cv.findContours(image)

            # If there is more than one contour, division can still happen

            if len(contours) > 2:

                self.\_\_segment\_image(image)

def \_\_segment\_image(self, img) -> SegmentationGroup:

    # Get projection over x axis

    x\_projection = np.matrix(np.amax(img, axis=0))

    # Get all connected components in the projection

    x\_labeled, x\_ncomponents = get\_components(x\_projection)

    if x\_ncomponents > 1:

        return self.\_\_x\_division(img, x\_labeled, x\_ncomponents)

    else:

        # Get projection over y axis

        y\_projection = np.matrix(np.amax(img\_inv, axis=1)).transpose()

        # Get all connected components in the projection

        y\_labeled, y\_ncomponents = get\_components(y\_projection)

        if y\_ncomponents > 1:

            return self.\_\_y\_division(img, y\_labeled, y\_ncomponents)

        # If there are not projected components remove mask

        else:

            return self.\_\_mask\_removal(img)

def \_\_x\_division(img, x\_labeled, x\_ncomponents) -> SegmentationGroup:

    """Function that divides an image into vertical components"""

    # Array of segmented images

    segmented\_images = [None] \* x\_ncomponents

    for component\_index in range(x\_ncomponents):

        cropped\_component = get\_cropped\_component(img, component\_index)

        segmented\_images[component\_index] = cropped\_component

    return SegmentationGroup(X\_SEGMENTATION, segmented\_images)

def \_\_y\_division(img, y\_labeled, y\_ncomponents) -> SegmentationGroup:

    """Function that divides an image into horizontal components"""

    """(...)"""

    return SegmentationGroup(Y\_SEGMENTATION, segmented\_images)

def \_\_root\_removal(img: np.ndarray) -> SegmentationGroup:

    """Function that removes the symbol with biggest bounding box"""

    bound\_rects = get\_bounding\_rects(img)

    # Find biggest box

    biggest\_rect\_idx = get\_biggest\_rect(bound\_rects)

    segmented\_images = split\_mask\_rest(img, bound\_rects,

biggest\_rect\_idx)

    return SegmentationGroup(MASK\_REMOVAL, segmented\_images)

Actualmente, el modelo funciona correctamente para las imágenes correctamente alineadas, sin embargo, cuando las fotos están torcidas o las líneas no son rectas a veces la separación es incorrecta. En las siguientes figuras se pueden ver los niveles de una segmentación correcta en la que se incluyen todos los tipos de operación considerados:

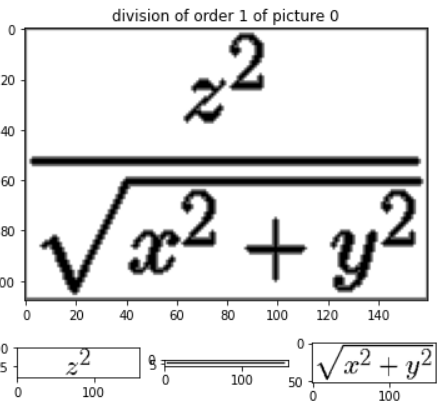


Figura 8: Segmentación de primer orden

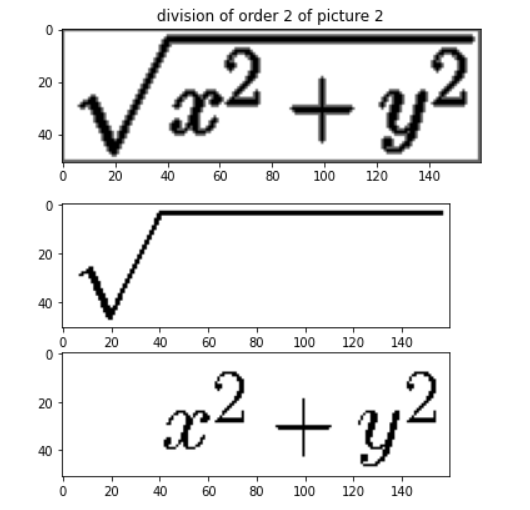
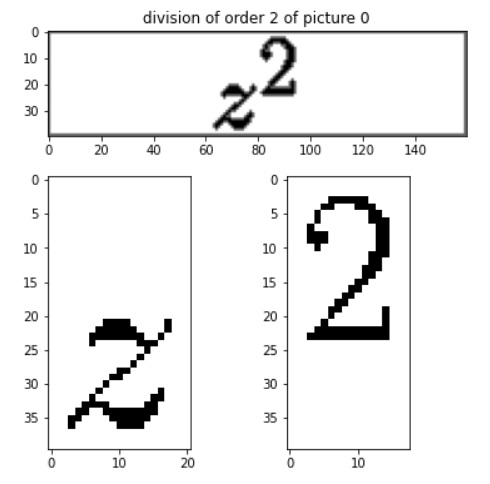


Figura 9: Segmentación de segundo orden

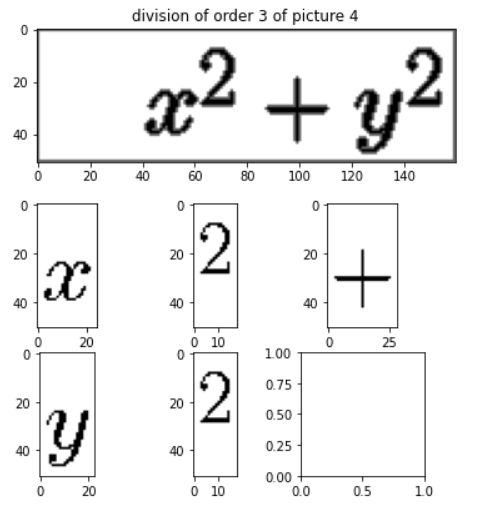


Figura 10: Segmentación de tercer orden

## Clasificación

El módulo de clasificación está compuesto por una red neuronal convolucional adaptada del modelo basado en LeNet presentado por A. Nazemi, N. Tavakolian, D. Fitzpatrick, C. Fernando y C. Y. Suen [33]. La red ha sido ligeramente modificada, en concreto respecto a los *paddings* empleados. En la siguiente figura se puede ver la estructura del modelo:

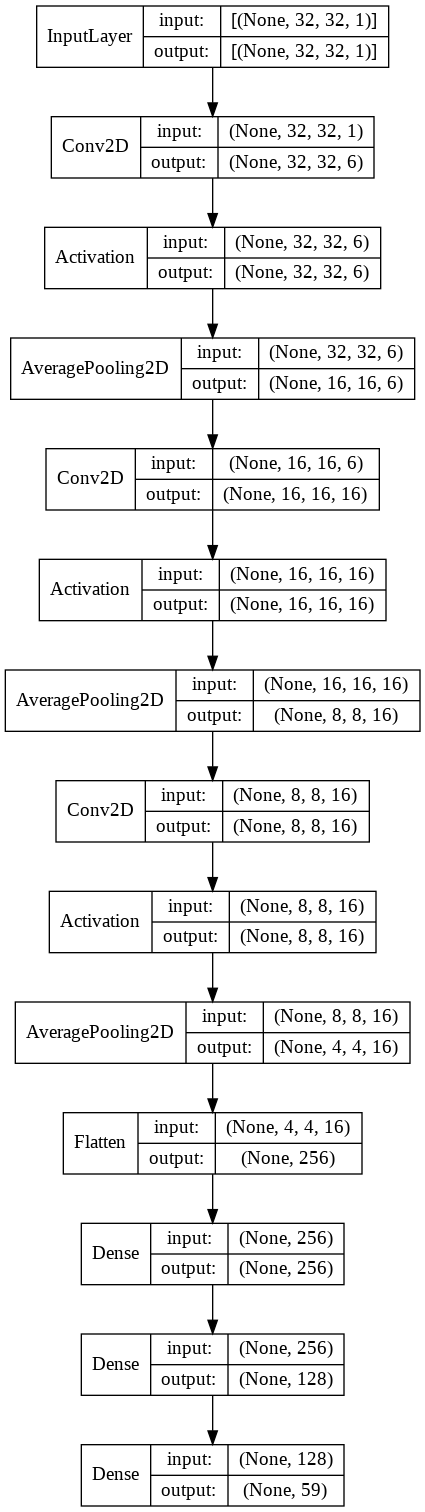
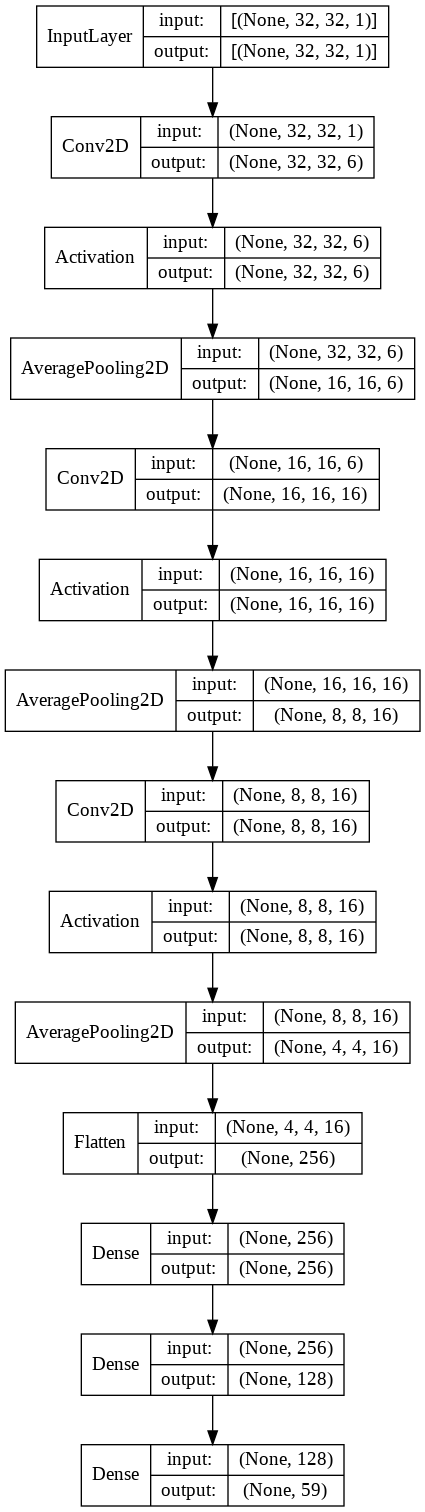


Figura 11: Modelo de capas de la red neuronal

Para el entrenamiento se ha utilizado una combinación del dataset del CROHME (2011-2014) en el que los archivos de trazos se han convertido a imágenes [34] junto con los registros del NIST [35]. De los símbolos recogidos en ambos *datasets* se han seleccionado los más comunes en ecuaciones diferenciales ordinarias.

Los símbolos, así como el número de registros para cada uno se pueden ver en la siguiente tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Símbolo | CROHME | NIST[[8]](#footnote-8) | Símbolo | CROHME | NIST |
| ( | 14.294 |  | M | 2.476 | 20.514+5.743 |
| ) | 14.355 |  | N | 10.862 | 18.737+26.172 |
| [ | 778 |  | O | 449 | 57.819+59.76 |
| ] | 780 |  | P | 2.680 | 19.021+5.217 |
| + | 25.112 |  | Q | 1.230 | 5.584+6.614 |
| - | 33.997 |  | R | 2.671 | 11.318+32.359 |
| / |  |  | S | 1.413 | 48.099+5.834 |
| , | 1.906 + 329[[9]](#footnote-9) |  | T | 3.274 | 22.323+42.020 |
| 0 | 6.914 | 75.166 | U | 1.269 | 28.750+6.149 |
| 1 | 26.520 | 82.753 | V | 1.558 | 10.384+6.232 |
| 2 | 26.142 | 74.256 | W | 556 | 10.527+5.863 |
| 3 | 10.909 | 76.405 | X | 26.594 | 5.934+6.112 |
| 4 | 7.396 | 72.586 | Y | 9.340 | 10.629+5.105 |
| 5 | 3.545 | 67.673 | Z | 5.870 | 2.863+5.902 |
| 6 | 3.118 | 74.016 | Alfa | 2.546 |  |
| 7 | 2.909 | 77.689 | Beta | 2.025 |  |
| 8 | 3.068 | 73.453 | Gamma | 409 |  |
| 9 | 3.737 | 73.253 | Delta | 137 |  |
| A | 12.367 | 14.479+22.873 | Theta | 2.796 |  |
| B | 8.651 | 8.617+11.563 | Lambda | 109 |  |
| C | 5.802 | 23.148+6.078 | Mu | 177 |  |
| D | 4.852 | 10.286+23.281 | Pi | 2.332 |  |
| E | 3.003 | 11.205+57.022 | Sigma | 201 |  |
| F | 3.712 | 20.825+5.454 | Phi | 355 |  |
| G | 1.692 | 5.539+8.115 | Seno | 4.293 |  |
| H | 1.464 | 6.944+19.930 | Coseno | 2.986 |  |
| I | 5.140 | 27.173+5.940 | Tangente | 2.450 |  |
| J | 1.536 | 8.350+4.133 | Logaritmo | 2.001 |  |
| K | 3.074 | 5.323+5.519 | Raíz cuadrada | 8.908 |  |
| L | 1.017 | 11.276+34.790 |  |  |  |

Se puede observar que el número de ocurrencias para cada símbolo difiere ampliamente. Esto implica que se genera un sesgo en la clasificación de símbolos, pues los símbolos que menos aparecen en el *dataset* tienen menor probabilidad de ocurrencia. Además, los símbolos con poca repetición alcanzan un menor nivel de abstracción por lo que es más difícil que sean reconocidos fuera de los casos proporcionados en el dataset.

El desbalanceo del *dataset* es un evento muy frecuente en entrenamiento de redes neuronales y existen diversos métodos para paliarlo. En el caso de *datasets* muy grandes se puede tomar una selección de los símbolos con mayor número de repeticiones para igualar las apariciones.

En otros casos, se pueden introducir repetidas las ocurrencias de los símbolos con menor representación. Esto plantea el problema de que si bien sí reduce el sesgo del sistema a la hora de predecir un resultado, la red realmente no «aprende» características nuevas, y , por tanto, el nivel de abstracción alcanzado para el símbolo es el mismo.

Por último, se puede practicar técnicas de *data augmentation* para ampliar el número de repeticiones. Para símbolos simétricos en el eje vertical u horizontal se pueden aplicar transformaciones de reflexión para duplicar –o cuadriplicar en caso de simetría en ambos ejes– el número de ocurrencias. También se pueden aplicar ligeras rotaciones, o aumentar o disminuir el grosor de los trazos.

Existe, por otro lado, la posibilidad de buscar *datasets* adicionales que complementen los datos que ya se tienen o de eliminar símbolos admitidos, aunque esto reduzca la funcionalidad del sistema.

En cualquier caso, dichas técnicas quedan fuera del alcance del proyecto y se dejan como posibilidad futura de desarrollo.

#TODO: Add results, confusion matrix; explain loss, accuracy, f1 score; grid searching, possible reduce learning rate on plateau, images of convolutional layers.

En la siguiente figura se pueden ver unas predicciones de símbolos:

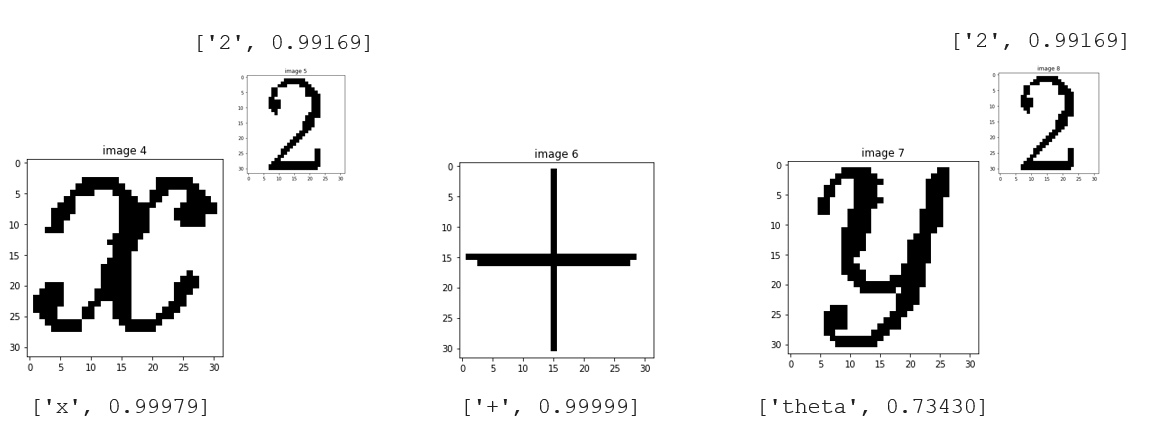


Figura 12: Predicciones de clasificación de símbolos

Las predicciones arrojan una lista con la probabilidad estimada para cada símbolo. En la figura anterior se ha añadido a cada símbolo la clase con mayor probabilidad. Se puede observar que para los símbolos en los que acierta la predicción tiene unas probabilidades (>0.99) muy superiores a la predicción para símbolos erróneos (<0.74).

Las predicciones de símbolos pertenecientes al alfabeto griego generan mayor confusión por su escasez dentro del dataset. Esto probablemente se podría solucionar accediendo a algún dataset como el GRUHD que contiene más de 600,000 símbolos del alfabeto griego manuscritos [36].

## Análisis estructural

El análisis estructural se realiza recogiendo los datos de la segmentación en la que se forma un árbol con las relaciones entre grupos de segmentación. Junto con los símbolos resultantes de la clasificación se recorre el árbol desde los nodos terminales hacia arriba convirtiendo a LaTeX el grupo en función de la operación de segmentación de la que procede.

Para el caso de las operaciones de extracción de máscara, actualmente el único símbolo considerado es la raíz cuadrada. Para el caso de división horizontal (proyección de componentes sobre el eje Y) se considera que para grupos de más de dos símbolos se consideran las fracciones. Para el caso de dos símbolos se consideran según los símbolos clasificados si pueden tratarse de símbolos con punto («i», «j») o en caso de dos guiones detectados el símbolo «=». En las divisiones verticales (proyección sobre el eje X) se anotan las posiciones relativas de los centroides de los símbolos para distinguir cuando parten de una misma línea base o son superíndices o subíndices. A continuación, se puede ver el código empleado para la reconstrucción y transformación a LaTeX:

class XYParser:

  def parse\_equation(self, predicted\_array, segmentation\_levels):

    self.\_\_parsed\_levels = list()

    self.add\_level(ParsedLevel(predicted\_array))

    for level\_index, segmentation\_level in segmentation\_levels:

      self.add\_level(ParsedLevel())

      expression\_iter = iter(self.previous\_level.parsed\_groups)

      segmentation\_groups = segmentation\_level.segmentation\_groups

      for group\_index, segmentation\_group in segmentation\_groups:

        self.parse\_group(segmentation\_group, expression\_iter)

  def parse\_group(self, segmentation\_group, parsed\_iterator):

    operation = segmentation\_group.segmentation\_operation

    elif operation == SegmentationOperation.MASK\_REMOVAL:

      parsed\_group = ''.join(("{\\", next(parsed\_iterator),

                              "{", next(parsed\_iterator), "}}"))

    elif operation == SegmentationOperation.X\_SEGMENTATION:

      previous\_level = 0

      parsed\_group = ''

      for symbol\_level in segmentation\_group.segmentation\_levels:

        if symbol\_level == previous\_level:

          parsed\_group += next(parsed\_iterator)

        elif symbol\_level < previous\_level:

          if previous\_level > 0:

            parsed\_group = ''.join((parsed\_group, "}",

                                    next(parsed\_iterator)))

          else:

            parsed\_group = ''.join((parsed\_group, "\_{",

                                    next(parsed\_iterator)))

        else:

          if previous\_level >= 0:

            parsed\_group = ''.join((parsed\_group, "^{",

                                    next(parsed\_iterator)))

          else:

            parsed\_group = ''.join((parsed\_group, "}", next(parsed\_iterator)))

        previous\_level = symbol\_level

      parsed\_group += "}" \* abs(symbol\_level)

    elif operation == SegmentationOperation.Y\_SEGMENTATION:

      if len(segmentation\_group.segmented\_images) == 3:

        numerator, \_, denominator = next(parsed\_iterator),

                                    next(parsed\_iterator),

                                    next(parsed\_iterator)

        parsed\_group = ''.join((r"\frac{", numerator, "}{", denominator, "}"))

      elif len(segmentation\_group.segmented\_images) == 2:

        group1, group2 = next(parsed\_iterator), next(parsed\_iterator)

        if [group1, group2] == ['-', '-']:

          parsed\_group = '='

        else:

          parsed\_group = group2

## Resolución

## Distribución de eventos

## Resultados de los tests

# Conclusiones

# Líneas futuras

Debido a la gran escala del proyecto se pueden abordar mejoras en cada uno de los módulos, así como en las distintas vertientes de estos.

En cuanto al reconocimiento de las ecuaciones se pueden abordar diversos caminos. A continuación, se detallan, de forma no exhaustiva, las posibilidades.

Empezando por el módulo de segmentación, se puede continuar la línea por la que se ha empezado.

Para empezar, se podrían implementar métodos de alineación y/o separación de componentes. Como se ha comentado en 6.4 Segmentación actualmente el modelo funciona bien cuando las imágenes están alineadas y las componentes no se superponen. Para casos en los que estas hipótesis no se cumplen el algoritmo no es capaz de separar los símbolos debido a que las proyecciones sobre los ejes no arrojan componentes separadas.

La autoalineación de componentes puede realizarse, por ejemplo, encontrando los cuadros delimitadores de menor área y rotándolos hasta que estos queden alineados respecto de los ejes vertical y horizontal (en el sentido que requiera menor rotación).

Por otro lado, si las componentes se solapan, se puede encontrar el centroide de la imagen, así como los de cada símbolo y expandirlos en la dirección que los une una distancia proporcional a la distancia entre ellos, de esa forma, las componentes mantienen estructuras similares con mayor espaciado.

Existen, también, diversas técnicas de segmentación diferentes a la empleada que podrían probarse, como son los métodos basados en rectángulos delimitadores o en grupos de píxeles como pueda ser el SLIC (algoritmo de agrupación linear iterativa simple) que divide la imagen en función de los grupos de pixeles. Estos métodos requieren un ajuste más fino y por ello no se han obtenido buenos resultados en el contexto de este proyecto. Sin embargo, se deja el código utilizado en el repositorio del proyecto como punto de partida para posibles futuros desarrollos.

Continuando con los módulos, el módulo de clasificación, que probablemente tenga la mayor complejidad se puede dividir en varias partes: *dataset*, preprocesamiento, modelo y entrenamiento.

En cuanto al *dataset*, como se ha comentado en 6.5 Clasificación se pueden añadir nuevos registros, o aumentar los que ya se tienen con diversas técnicas, balanceando el número de muestras de cada símbolo. También se podrían añadir nuevos símbolos para entrenar el modelo en caso de que se quisiera añadir nueva funcionalidad al proyecto (p.ej.: integrales, límites…).

Para el proyecto se ha decidido utilizar imágenes de 32x32 por la velocidad de entrenamiento y la reducción del espacio, si bien las imágenes originales del *dataset* eran de tamaño 45x45 y estaban generadas de datos online, por lo que yendo a la fuente original se podrían obtener versiones aún mayores. Optimizar el tamaño de las imágenes de entrenamiento y entrada podría ser, en sí, un proyecto de desarrollo.

Respecto al preprocesamiento las imágenes se han utilizado con el preprocesamiento por defecto que tenía la fuente. Sin embargo, podrían aplicarse distintas técnicas de normalización para optimizar el rendimiento del clasificador.

El modelo elegido es uno de los muchos referenciados en el trabajo y aún más s pueden encontrar en la literatura asociada. Por la dimensión del proyecto, se ha elegido un modelo relativamente sencillo de configurar y entrenar. Sería interesante implementar un modelo del estado del arte actual como pueda ser el ganador del CROHME 2019.

Además de buscar los modelos de mayor rendimiento, debido al carácter pedagógico de este proyecto, se podrían implementar modelos basados en gramáticas, como forma de aprendizaje. En el terreno práctico estos modelos también tienen resultados de gran rendimiento, y ofrecen la posibilidad de realizar ensamblados de modelos para obtener mejores resultados.

En un estadio más avanzado del proyecto, donde se hubieran implementado diversos modelos, se podría añadir una funcionalidad de comparador, donde se mostraran los diferentes resultados arrojados por cada modelo y así poder analizar sus fortalezas y debilidades. A nivel de aplicación de reconocimiento esto se podría utilizar para sugerir al usuario las distintas combinaciones resultantes, para que pudiera elegir la más adecuada.

En lo que respecta al entrenamiento, hay diversas técnicas que se pueden emplear para mejorar los modelos. En concreto se puede realizar una búsqueda en rejilla en donde se entrene el modelo iterando en distintas combinaciones de los hiperparámetros como el *batch size* o el *lerning rate*, así como los optimizadores y funciones de activación y pérdidas empleadas.

Para que el entrenamiento fuera más preciso se podrían emplear técnicas como la reducción del *learning rate* cuando se estanca el aprendizaje o añadir funciones como el descenso de gradiente o el momento de Nesterov que utilizan la derivada de la función de pérdidas para predecir cuándo se va a alejar de la solución óptima y cambia el *learning rate*.

En cuanto a la fase de análisis estructural, para el modelo empleado va muy ligada a la segmentación. Si bien el modelo actual se puede optimizar, el salto lógico sería buscar otros modelos de análisis estructural más ligados a la clasificación.

Por otro lado, también se pueden realizar mejoras en el sistema de resolución de las ecuaciones

# Bibliografía

|  |  |
| --- | --- |
| [1] | R. H. Anderson, «Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics,» de *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, New York, Association for Computing Machinery, 1967, p. 436–459. |
| [2] | C. V.-G. D. H. K. J. H. K. a. U. G. H. Mouchere, «CROHME2011: Competition on Recognition of Online Handwritten Mathematical Expressions,» de *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 1497-1500. |
| [3] | C. V.-G. D. H. K. J. H. K. U. G. Harold Mouchère, «ICFHR 2012 Competition on Recognition of On-Line Mathematical Expressions (CROHME 2012),» de *2012 International Conference on Frontiers in Handwriting Recognition*, 2012, pp. 811-816. |
| [4] | H. a. V.-G. C. a. Z. R. a. G. U. a. K. D. H. a. K. J. H. Mouchère, «ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions,» de *2013 12th International Conference on Document Analysis and Recognition*, 2013, pp. 1428-1432. |
| [5] | H. a. V.-G. C. a. Z. R. a. G. U. Mouchère, «ICFHR 2014 Competition on Recognition of On-Line Handwritten Mathematical Expressions (CROHME 2014),» de *2014 14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 791-796. |
| [6] | H. a. V.-G. C. a. Z. R. a. G. U. Mouchère, «ICFHR2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions,» de *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 607-612. |
| [7] | M. a. Z. R. a. M. H. a. V.-G. C. a. G. U. Mahdavi, «ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection,» de *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1533-1538. |
| [8] | «An integrated grammar-based approach for mathematical expression recognition,» *Pattern Recognition,* vol. 51, pp. 135-147, 2016. |
| [9] | A. a. H. J.-P. Belaid, «A Syntactic Approach for Handwritten Mathematical Formula Recognition,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vols. %1 de %2PAMI-6, nº 1, pp. 105-111, 1984. |
| [10] | A. a. R. G. Kosmala, «On-line handwritten formula recognition using statistical methods,» de *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, 1998, pp. 1306-1308 vol.2. |
| [11] | doi.org/10.1007/978-3-642-77281-8\_2, «An Experimental Implementation of a Document Recognition System for Papers Containing Mathematical Expressions,» de *Structured Document Image Analysis*, Springer Berlin Heidelberg, 1992, pp. 36-53. |
| [12] | J. a. H. R. a. P. I. Ha, «Understanding mathematical expressions from document images,» de *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, pp. 956-959. |
| [13] | S. a. N. K. a. A. J. Smithies, «A Handwriting-Based Equation Editor,» de *Proceedings of the 1999 Conference on Graphics Interface '99*, Morgan Kaufmann Publishers Inc., 1999, p. 84–91. |
| [14] | H. a. Z. R. a. G. U. a. V.-G. C. Mouchère, «Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014,» *International Journal on Document Analysis and Recognition (IJDAR),* vol. 19, 2016. |
| [15] | E. Kamke, Differentialgleichungen Lösungsmethoden und Lösungen, Vieweg+Teubner Verlag, Wiesbaden, 1977. |
| [16] | N. M. Abbasi, «Kamke diﬀerential equations. Mathematica 12.3.1 and Maple 2021.1,» 12 Agosto 2021. [En línea]. Available: https://www.12000.org/my\_notes/kamek/mma\_12\_3\_1\_maple\_2021\_1/KEse1.htm#x3-20001. |
| [17] | J. D. Mena, Desarrollo de herramientas de soporte a un curso OCW-UPM sobre resolución de ecuaciones diferenciales mediante simetrí}as de Lie, Madrid, 2020. |
| [18] | O. Benjamin, «Realeases - sympy/sympy,» 8 Agosto 2021. [En línea]. Available: https://github.com/sympy/sympy/releases. |
| [19] | SymPy, «sympy/sympy: A computer algebra system written in pure Python,» 15 Agosto 2021. [En línea]. Available: https://github.com/sympy/sympy. |
| [20] | S. Overflow, «StackOverflow Developer Survey 2020,» 2020. [En línea]. Available: https://insights.stackoverflow.com/survey/2020. |
| [21] | F. B. (Madnight), «GitHub Language Stats,» 27 Mayo 2021. [En línea]. Available: https://madnight.github.io/githut/#/pull\_requests/2021/2. |
| [22] | Google, «Python, JavaScript, Java - Explorar - Google Trends,» 15 Agosto 2021. [En línea]. Available: https://trends.google.es/trends/explore?date=2011-08-15%202021-08-15&q=%2Fm%2F05z1\_,%2Fm%2F02p97,%2Fm%2F07sbkfb. |
| [23] | I. a. L. A. a. F. D. Lagaris, «Artificial neural networks for solving ordinary and partial differential equations,» *IEEE Transactions on Neural Networks,* vol. 9, nº 5, pp. 987-1000, 1998. |
| [24] | F. C. Guillaume Lample, «Deep Learning for Symbolic Mathematics,» *CoRR,* vol. abs/1912.01412, 2019. |
| [25] | C. Davi, «Using Neural Networks to solve Ordinary Differential Equations | by Caio Davi | Towards Data Science,» 23 Febrero 2021. [En línea]. Available: https://towardsdatascience.com/using-neural-networks-to-solve-ordinary-differential-equations-a7806de99cdd. |
| [26] | T. Peters, «PEP 20 -- The Zen of Python | Python.org,» 12 Agosto 2004. [En línea]. Available: https://www.python.org/dev/peps/pep-0020/. |
| [27] | B. W. N. C. Guido van Rossum, «PEP 8 -- Style Guide for Python Code | Python.org,» 5 Julio 2001. [En línea]. Available: https://www.python.org/dev/peps/pep-0008/. |
| [28] | G. v. R. David Goodger, «PEP 257 -- Docstring Conventions | Python.org,» 29 Mayo 2001. [En línea]. Available: https://www.python.org/dev/peps/pep-0257/. |
| [29] | J. L. L. L. Guido van Rossum, «PEP 484 -- Type Hints | Python.org,» 29 Septiembre 2014. [En línea]. Available: https://www.python.org/dev/peps/pep-0484/#type-aliases. |
| [30] | I. L. L. R. G. V. R. Ryan Gonzalez. Philip House, «PEP 526 -- Syntax for Variable Annotations | Python.org,» 9 Agosto 2016. [En línea]. Available: https://www.python.org/dev/peps/pep-0526/. |
| [31] | K. Stratis, «Python Application Layouts: A Reference – Real Python,» 15 Mayo 2018. [En línea]. Available: https://realpython.com/python-application-layouts/. |
| [32] | C. F. a. Z.-X. Wang, «AUTOMATIC PERCEPTION OF THE STRUCTURE OF HANDWRITTEN MATHEMATICAL EXPRESSIONS,» de *Computer Processing of Handwriting*, 1990, pp. 337-361. |
| [33] | A. N. a. N. T. a. D. F. a. C. F. a. C. Y. Suen, «Offline handwritten mathematical symbol recognition utilising deep learning,» *ArXiv,* p. 2019. |
| [34] | X. Nano, «Handwritten math symbols dataset | Kaggle,» 15 Enero 2017. [En línea]. Available: https://www.kaggle.com/xainano/handwrittenmathsymbols. |
| [35] | NIST, «NIST Special Database 19 - 2nd edition,» 2016. |
| [36] | E. a. L. N. a. K. E. a. F. N. a. K. G. Kavallieratou, «The GRUHD database of Greek unconstrained handwriting,» de *Proceedings of Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 561-565. |
| [37] | F. a. H. N. S. a. V. C. a. M. H. a. M. S. JulcaAguilar, «Mathematical Symbol Hypothesis Recognition with Rejection Option,» de *2014 14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 500-505. |
| [38] | F. a. M. H. a. V.-G. C. a. H. N. S. T. Julca-Aguilar, «Top-Down Online Handwritten Mathematical Expression Parsing with Graph Grammar,» de *20th Iberoamerican Congress, CIARP 2015, Montevideo, Uruguay, November 9-12, 2015, Proceedings*, Montevideo, 2015, pp. 444-45. |
| [39] | J. Z. a. J. D. a. S. Z. a. D. L. a. Y. H. a. J. H. a. S. W. a. L. Dai, «Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition,» *Pattern Recognition,* pp. 196-206, 2017. |
| [40] | Y. F. Z. Y. Z. X. L. C. Wu JW., «Image-to-Markup Generation via Paired Adversarial Learning,» de *Machine Learning and Knowledge Discovery in Databases*, Cham, Springer International Publishing, 2019, pp. 18-34. |

# Planificación temporal y presupuesto

# Índice de figuras (opcional)

# Índice de tablas (opcional)

# Abreviaturas

CROHME: *Competition on Recognition of On-line Handwritten Mathematical Expressions* (Competición en reconocimiento *online* de expresiones matemáticas manuscritas).

ICDAR: *International Conference on Document Analysis and Recognition* (Conferencia internacional de análisis y reconocimiento de documentos)

LSTM: *Long-Short Term Memory* (Memoria de corto y largo plazo)

BLSTM: *Bidirectional Long-Short Term Memory* (Memoria de corto y largo plazo bidireccional)

GRU: *Gated Recurrent Unit* (Unidad recurrente cerrada)

RNN: *Recurrent Neural Network* (Red neuronal recurrente)

CNN: *Convolutional Neural Network* (Red neuronal convolucional)

SVM: *Support Vector Machine* (Máquina de vectores de soporte)

HMM: *Hidden Markov Models* (Modelos ocultos de Márkov)

KNN: *K-Nearest Neighbors* (K-vecinos cercanos)

1. https://demo.wiris.com/mathtype/en/index.php [↑](#footnote-ref-1)
2. https://github.com/falvaro/seshat [↑](#footnote-ref-2)
3. https://cat.prhlt.upv.es/mer [↑](#footnote-ref-3)
4. https://www.myscript.com/calculator [↑](#footnote-ref-4)
5. https://mathpix.com/ [↑](#footnote-ref-5)
6. https://www.v7labs.com/blog/convolutional-neural-networks-guide [↑](#footnote-ref-6)
7. Resultados para Mathematica 12.0.0.0 con un límite de tiempo de cálculo de 30s [↑](#footnote-ref-7)
8. Para las categorías del NIST se suman mayúsculas y minúsculas. [↑](#footnote-ref-8)
9. Se suman las categorías de «,» y «’» [↑](#footnote-ref-9)