UNIVERSIDAD DE GUADALAJARA

Red Universitaria e Institución Benemérita de Jalisco

CUCEI

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

ANÁLISIS DE ALGORITMOS

PROYECTO FINAL

Equipo: Bogo-Sort Fans

Alumnos:

Cesar Alejandro Prieto Franco

Corona Espinoza Daniel Joel

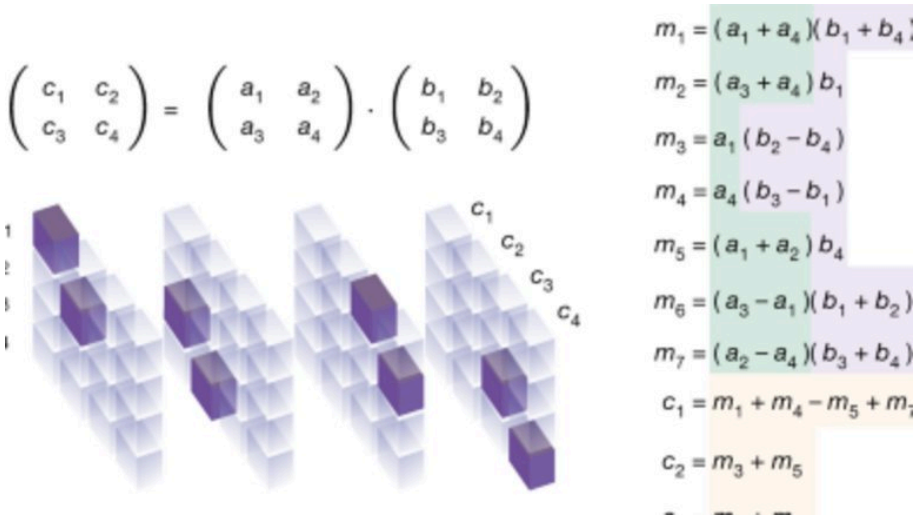
Fecha:
26 de Noviembre de 2025

Introducción

Para este proyecto hemos explorado una variedad de algoritmos y sus técnicas durante el semestre, observando las aplicaciones y utilidades de cada uno, asi como sus características para decidir cual era la mejor opción dependiendo de cosas como la complejidad en tiempo y memoria, por supuesto verificando que fuera el algoritmo que mejor se adaptara al problema en cuestión.

En principio al momento de estar en el tema de fuerza bruta elegimos el algoritmo de String Matching siendo un problema en el que se comparan subcadenas de forma iterativa para encontrar patrones dentro de cadenas de texto más grandes, nos fue interesante debido a su amplia variedad de aplicaciones en editores de texto, buscadores de documentos o en menor proporción se pueden encontrar en motores de búsqueda.

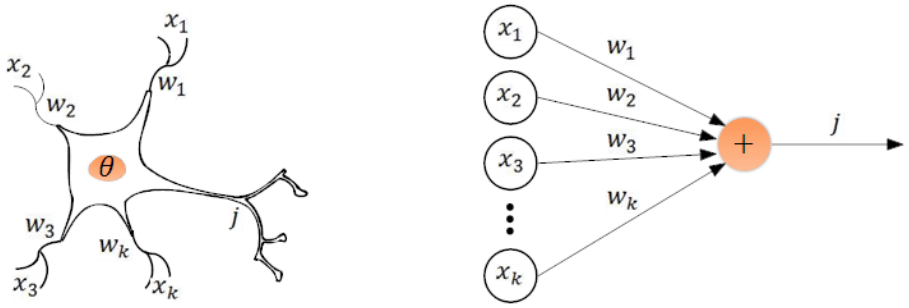
Pasando al siguiente tema de técnicas de Divide y Vencerás, siendo este una estrategia que consta de dividir un problema complejo en subproblemas más simples, permitiendo resolver o “atacar” un problema de manera más sencilla para encontrar una solución más compleja.
Para este tema hicimos un cambio de algoritmo, siendo el algoritmos de Strassen que consta de multiplicar matrices de forma más eficiente en términos de complejidad comparándolo con una multiplicación de matrices tradicional, estas teniendo una complejidad temporal de O(N^3), strassen logra reducir tal complejidad a O(N^2.807).



Para agregar mejores comparativas a este algoritmos decidimos también programarlo en CuPy, la librería con el mismo set de funciones de la librería Numerical Python, mejor conocida com NumPy, pero a diferencia de este que tiene la mayoría de sus funciones adaptadas para ejecutarse en C, CuPy hace algo similar, pero se ejecuta en la GPU, distribuyendo las operaciones en un gran cantidad de núcleos para poder realizar dichas operaciones de forma paralela haciendo que su ejecución se mucho mas rapida.

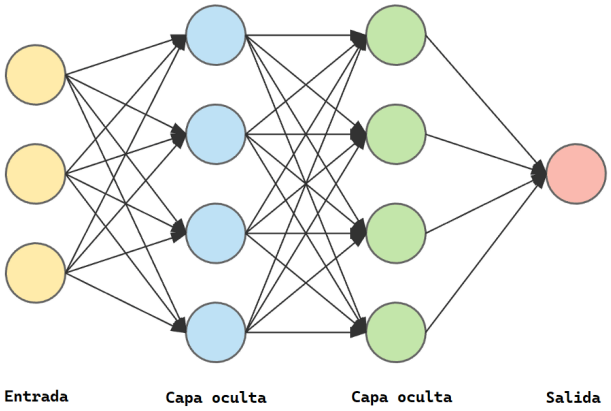
Después de implementar la técnica de divide y vencerás, al pasar a la siguiente etapa del algoritmo, optamos por recomendar ejecutar una Red Neuronal Convolucional.

Una Red Neuronal es un sistema que hace uso de neuronas que son elementos que tratan de imitar el comportamiento de las neuronas, estas reciben datos de entrada y junto con un parámetro y un peso hacen de una operación matemática que se pasa a una funciona de activación para generar una salida de la neurona.

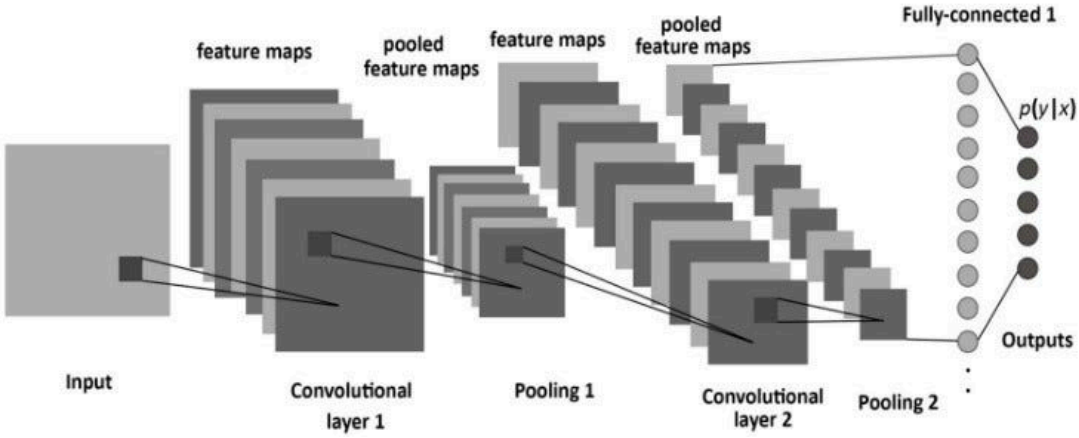


a) Neurona biológica a) Neurona artificial

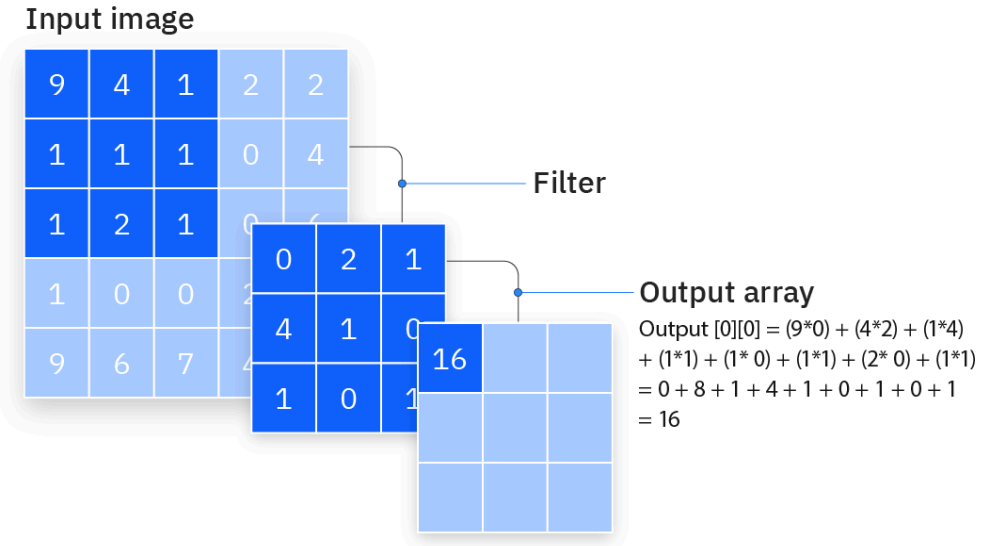
La Red Neuronal cuenta con muchas, la cual cada una representa un conjunto de nodos. La capa de entrada, las capas intermedias y la capa de salida. Cada capa esta interconectada mediante sus grafos los cuales se conectan mediante aristas con peso.



Para las Redes Neuronales Convolucionales usan un sistema muy similar, pero enfocado en el procesamiento de imágenes, estas toman matrices en las cuales cada elemento cuenta con un valor que representa un píxel.



La capa de convolución en una Red Neuronal Convolución es la parte más importante de las CNN ya que esta toma la capa de entrada y aplica un filtro o “kernel”, el cual no es más que otra matriz cuadrada que cuenta con valores que simbolizan los pesos del filtro, este filtro se recorre por toda la matriz de la capa de entrada haciendo operaciones elemento por elemento, la salida es la suma de la multiplicación de estos elementos, en caso de haber capas intermedias, esta salida será la entrada de otra capa intermedia.



Entre las capas intermedias pueden ir muchos tipos de capas, todo depende del tipo de CNN que quiera realizar y su objetivo.

A diferencia de una red neuronal tradicional, donde cada neurona se conecta con todas las de la siguiente capa, las CNN trabajan sobre estructuras locales, reduciendo la complejidad y mejorando la eficiencia al procesar información espacial como imágenes.

A continuación se describen las capas más relevantes:

Capa Convolucional (Convolutional Layer): Es el núcleo de una CNN y la encargada de extraer características relevantes de la entrada. Esta capa aplica filtros (o kernels) que

recorren la imagen realizando la operación de convolución. Cada filtro aprende a detectar un tipo de patrón, como bordes, texturas, gradientes o formas específicas.

Capa de Activación (Activation Layer): Después de la convolución se aplica una función de activación que introduce no linealidad, permitiendo que la red represente patrones complejos. La más utilizada es ReLU (Rectified Linear Unit),

Capa de Submuestreo o Pooling (Pooling Layer): Reduce las dimensiones espaciales del mapa de características, conservando la información relevante y eliminando ruido. El tipo más común es MaxPooling, que toma el valor máximo dentro de una región definida.

Capa Completamente Conectada (Fully Connected Layer)

Ubicada generalmente al final de la red, toma todas las características previamente extraídas y las interpreta para emitir una predicción. Funciona de manera equivalente a una red neuronal tradicional: cada neurona recibe todas las entradas y produce valores que se combinan para representar las diferentes clases.

Capa Softmax (Clasificación)

Convierte el vector final de activación en una distribución de probabilidades sobre las las clases posibles.

Backpropagation (Retropropagación): El algoritmo de backpropagation es el mecanismo mediante el cual la red ajusta sus parámetros en función del error cometido durante las predicciones. Este procedimiento calcula cómo cada peso contribuyó al error final y determina la cantidad exacta de corrección que debe aplicarse.

Gradient Descent

Una vez calculados los gradientes mediante backpropagation, la red necesita actualizar sus parámetros. Este proceso se realiza utilizando el algoritmo de descenso del gradiente, uno de los métodos fundamentales en optimización numérica.

Métricas: Loss y Accuracy

En el entrenamiento de redes neuronales es fundamental evaluar qué tan bien está aprendiendo el modelo. Para ello se utilizan métricas como la pérdida (loss) y la precisión (accuracy).

1. Loss (Función de pérdida)

La pérdida es un valor numérico que mide la diferencia entre la predicción de la red y la etiqueta real. Cuanto mayor sea el error, mayor será la pérdida.

2. Accuracy (Precisión)

La precisión mide la proporción de predicciones correctas entre el total de ejemplos.

- Una **accuracy alta** indica que el modelo está clasificando correctamente la mayoría de los ejemplos.

- Una **alta accuracy pero alto loss** puede significar que la red está insegura al emitir sus predicciones.

- Una **accuracy baja y loss alto** indica que la red todavía no ha aprendido patrones útiles.

Una forma de poder visualizar los resultados de la CNN es mediante grafos.
Los grafos son una estructura de datos que se compone de nodos que representan valores que están interconectados mediante aristas, las cuales pueden o no tener un peso, el cual en este caso si tendrán ya que representarán la similitud entre los resultados de cada salida de la CNN.

De acuerdo a la técnica voraz que podemos implementar para esta problemática implementamos el MST dado por Kruskal. Un árbol de expansión mínima (MST) es un subconjunto de aristas de un grafo con pesos que conecta todos los nodos sin formar ciclos y con la menor suma de pesos posible. Se utiliza para encontrar la forma más económica de conectar un conjunto de puntos.

Objetivo

El objetivo general del proyecto es poder hacer uso de distintas técnicas de algoritmos para resolver problemáticas reales en el mundo del software y aplicarlas a problemas reales.

Objetivos Específicos:
Implementación de algoritmos de Fuerza Bruta: Implementar algoritmos que prueben todos los casos, soluciones o combinaciones distintas para encontrar una solución, observar el comportamiento de los algoritmos haciendo uso de esta técnica, así como comprobar los restos técnicos y de costo que involucra el uso de estas técnicas.

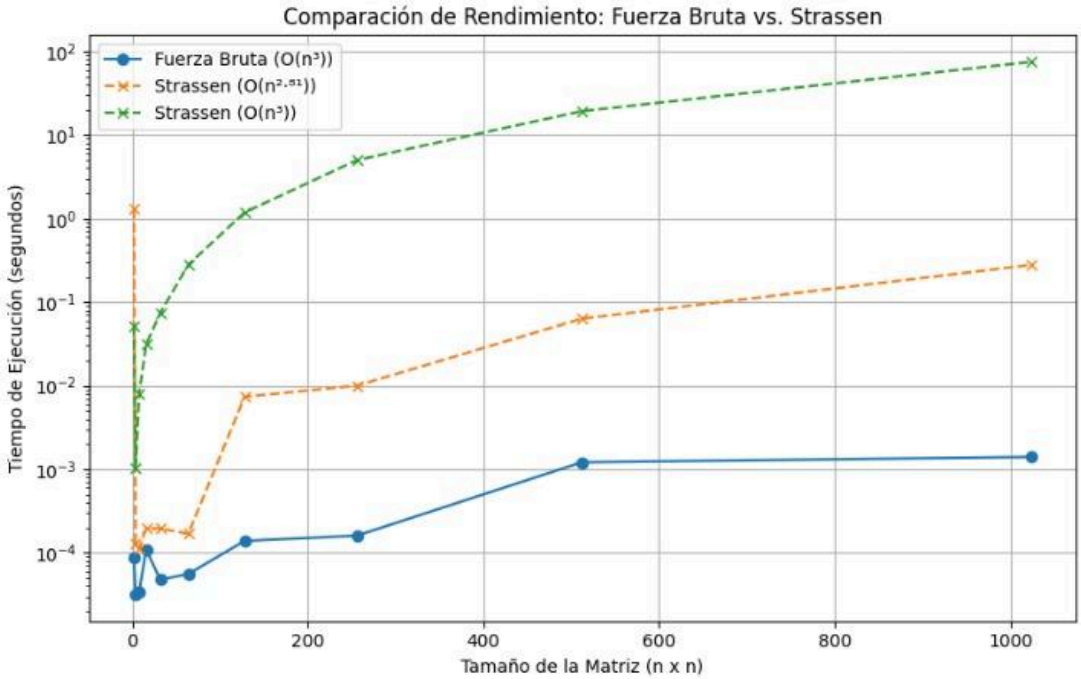
Implementación de Divide y Vencerás: Implementar una técnica para dividir un problema en subproblemas y poder dar soluciones juntando las respuestas a dichos subproblemas, implementados en strassen, ayudarnos a ahorrarnos tiempo y memoria.

Para la implementación de técnica voraz es integrar una técnica que permita aportar a la resolución de un problema, en este caso nuestro objetivo es implementar para tener una mejor visualización de los resultados de la CNN

Desarrollo:

Para el desarrollo de este proyecto combinamos varias técnicas de programación, más que la evolución del algoritmo, hemos utilizado diferentes algoritmos conforme hemos ido atacando problemas cada vez más complejos.

Como fuerza bruta originalmente utilizamos un algoritmo para cadenas, pero al pasar al algoritmo de divide y vencerás, donde utilizamos strassen para la multiplicación de matrices, nos enfocamos en este tema. En la parte de Divide y vencerás habíamos utilizado strassen como algoritmo base para comparar contra el método de fuerza bruta para multiplicar matrices, donde el método de multiplicación por fuerza bruta tenía una complejidad de O(N^3) mientras que strassen tenía una complejidad de O(N^2.8), y para mejores resultados utilizamos una implementación de Ambos métodos en Cuda:

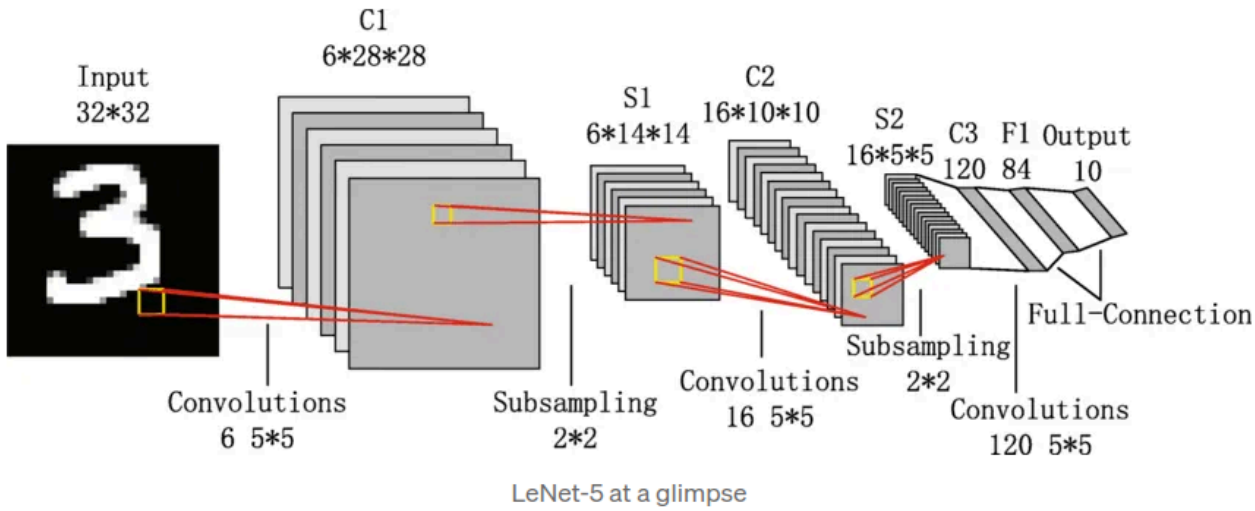


Partiendo de la multiplicación de matrices hechas en la primera parte, por sugerencia decidimos adentrarnos en las Redes Neuronales Convolucionales, utilizando recursos previamente proporcionados como datasets de imágenes, decidimos realizar este proyecto sobretodo porque consideramos que se podían integrar perfectamente las técnicas de multiplicación de matrices ya utilizadas, podemos verlas como el tipo de operación principal en las CNN, en cada capa, iteración, y proceso de la red neuronal se

utilizan demasiadas multiplicaciones de matrices, siendo la capa de convolución la mas importante donde es implementada por el filtro.

La red neuronal implementada consta de múltiples capas intermedias a través de las cuales cada entrada es procesada. La primera capa de una CNN siempre es una capa convolucional. Esta capa se encarga de detectar características simples como bordes o colores. Para la detección de patrones en partes más grandes de la imágen, se emplean capas convolucionales en etapas más avanzadas de la red.

La salida de la CNN es un vector de probabilidades, donde cada elemento indica cuál es la probabilidad de una determinada entrada de ser de cada clase o grupo del conjunto de entrada. En el caso del set de datos MNIST, la salida del programa es una lista de 10 elementos, uno por cada dígito del 0 al 9, indicando qué tan probable es cada imágen de entrada de pertenecer a esa clase en particular.

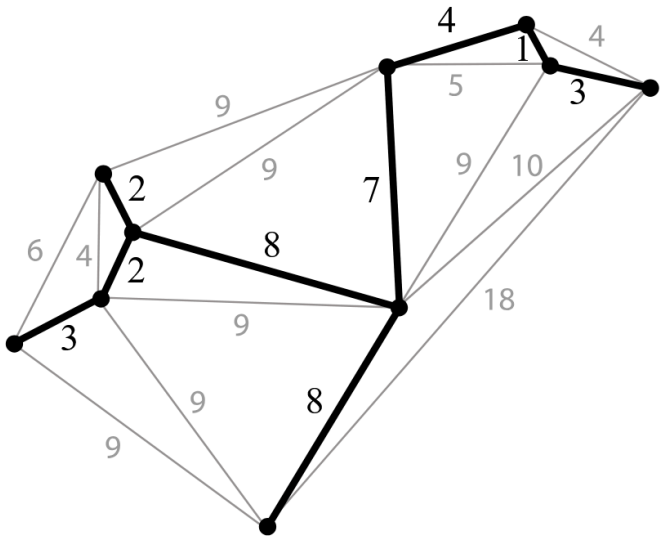


Ejemplo de una red neuronal LeNet-5 aplicada al dataset MNIST

Utilizando estos vectores de probabilidades, generamos una comparativa entre los datos de prueba del dataset seleccionado para este proyecto. La comparación que realizamos se hace a través de la distancia euclidiana, simulando un espacio de n dimensiones, donde n es el número de clases, representando las probabilidades de cada entrada como una dimensión distinta y calculando la distancia entre ambos vectores.

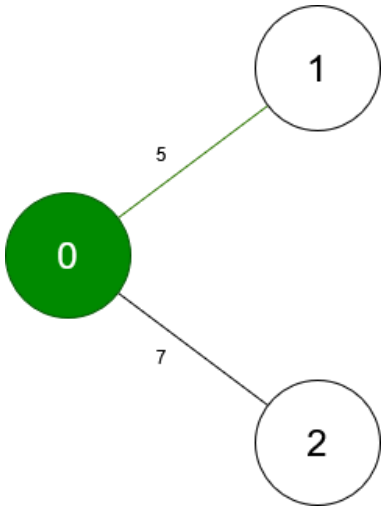
Utilizando las distancias entre cada predicción de pares de elementos, se construye una matriz de resultados, donde cada resultado corresponde a una predicción utilizando las entradas de prueba. Empleamos un algoritmo voraz para generar un árbol de expansión mínima, conectando cada resultado con otros resultados similares, ya sea que resulten en la misma clase o que sus entradas sean parecidas entre sí,

Utilizamos los algoritmos de Prim y Kruskal para determinar el árbol de expansión mínima del grafo generado por las salidas de la CNN. Ambos algoritmos emplean la técnica voraz para resolver sus respectivos problemas, tomando en cada momento el mejor vértice o la mejor arista respectivamente en cada paso de la ejecución.

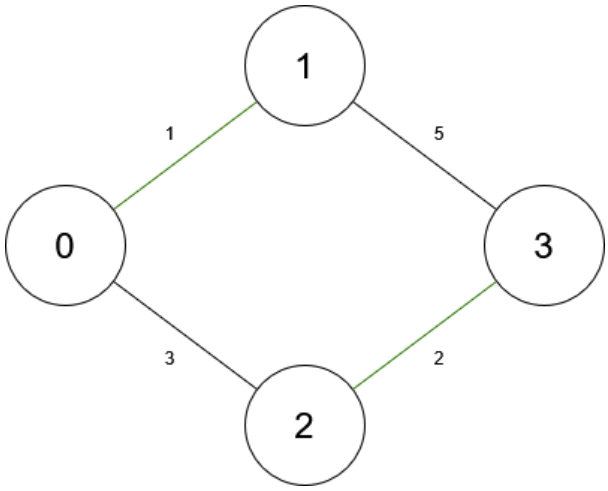


Árbol de expansión mínima.

En el caso de Prim, se toma un vértice inicial, y en cada iteración se agrega a la lista de vértices visitados el vértice conectado a la arista de menor peso que conecta un nodo visitado con un nodo no visitado. En el caso de Kruskal, no existe un nodo inicial, sino que se toman todas las aristas posibles, se ordenan por su peso o magnitud, y se van seleccionando las de menor peso, procurando que ninguna arista agregada forme un ciclo.



Prim selecciona el vértice no conectado con el menor peso para conectarse a un vértice conectado (en verde).



Kruskal selecciona las aristas que tengan menor peso dentro de todo el grafo, sin importar si están conectadas o no, mientras no formen ciclos.

Después de haber ejecutado un entrenamiento extensivo con un dataset MNIST de lenguaje de señas, el cual estaba dividido en 24 categorías lo cual culmina en un vector con 24 valores que representan las probabilidades de ser una o varias de las 24 categorías.

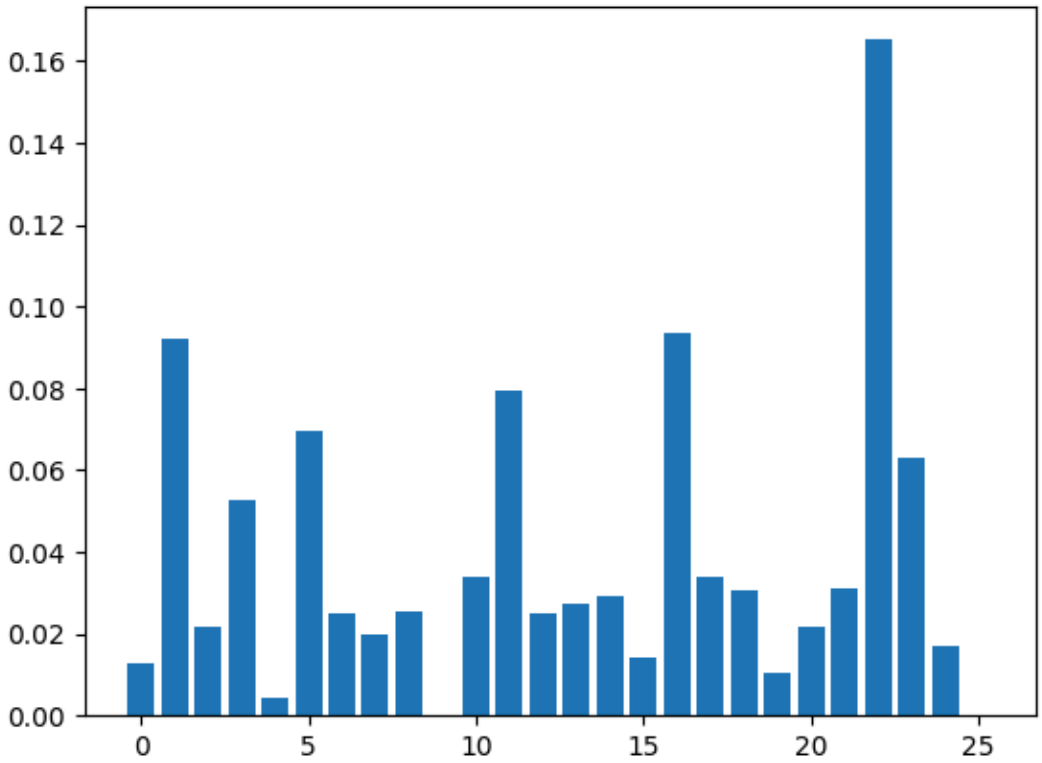
Para comprender y poder visualizar mejor estos valores decidimos clasifcarlos por cercanías en un plano cartesiano, pero al ser 24 valores utilizamos una ecuación para medir distancia en N-dimensiones, llamada distancia euclidiana multidimensional, que se basta en esta formula para calcular distancias:

$$\sqrt{\sum_{i=1}^{p}(Y_{i1}-Y_{i2})^2}$$

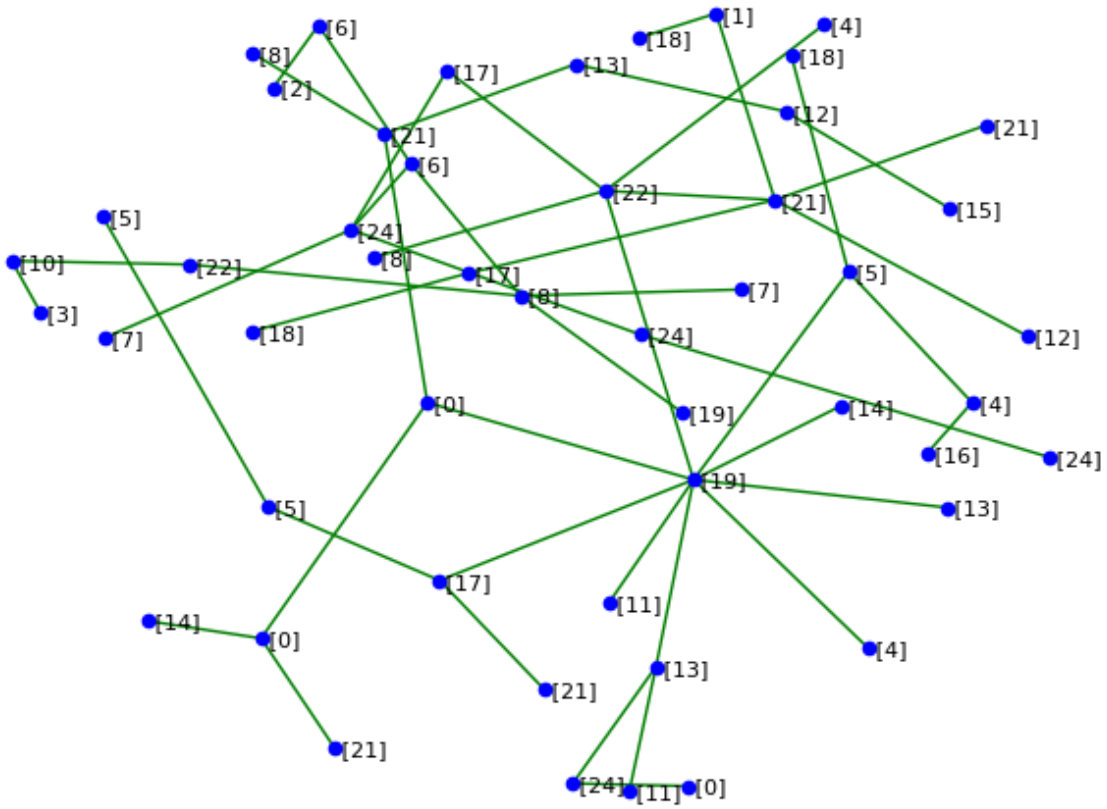
La distancia de cada nodo es usada como peso de conexión entre nodos, por lo cual utilizamos un árbol de expansión mínima para realizar un árbol que represente la categorización de cada uno de los nodos.

Resultados:

Los resultados de pasar un test case por la CNN, devuelve la cantidad probabilidades sobre cada cateogria



Un arbol de expansion minima que representa las similitudes entre



Conclusiones:

Al final, lo que nos llevamos de este proyecto es que la Inteligencia Artificial no es magia, sino pura matemática aplicada y bien optimizada. Nos dimos cuenta de que todo el "cerebro" de una Red Neuronal se reduce a multiplicar matrices masivamente, y ahí es donde el uso de varias técnicas puede jugar un papel muy importante. Fue genial ver cómo piezas que parecían sueltas —como grafos, algoritmos voraces y álgebra lineal— se conectan para crear un sistema que realmente puede "ver" y clasificar imágenes.

Bibliografía

Introduction to convolution neural network. (2017, agosto 21). GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/

What are Convolutional Neural Networks? (2025, noviembre 17). Ibm.com. https://www.ibm.com/think/topics/convolutional-neural-networks

Zhou, V. (s/f-a). CNNs, part 1: An introduction to convolutional neural networks. Victorzhou.com. Recuperado el 27 de noviembre de 2025, de https://victorzhou.com/blog/intro-to-cnns-part-1/

Zhou, V. (s/f-b). Machine learning for beginners: An introduction to neural networks. Victorzhou.com. Recuperado el 27 de noviembre de 2025, de https://victorzhou.com/blog/intro-to-neural-networks/

Wikipedia contributors. (2025c, octubre 9). Convolution. Wikipedia. https://en.wikipedia.org/wiki/Convolution

Wikipedia contributors. (2025d, noviembre 19). LeNet. Wikipedia. https://en.wikipedia.org/wiki/LeNet

Convolutional Neural Network (CNN). (s. f.). TensorFlow. https://www.tensorflow.org/tutorials/images/cnn