UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

GNN vs CNN en datos representados por grafos

PROYECTO DE TESIS I

Autor: Jesús Miguel Yacolca Huamán

Asesor: Victor Melchor Espinoza

Noviembre, 2021

Resumen

En esta tesis se realizará una comparación entre las Redes Neuronales basadas en Grafos y las Redes Neuronales Convolucionales, GNN y CNN respectivamente por sus siglas en inglés, en un problema de clasificación de imágenes. Se tendrá en cuenta la rapidez en la convergencia a una obtención óptima de los parametros. A su vez se comparará la capacidad de predictibilidad de ambas arquitecturas de Redes Neuronales.

Índice general

Índice de figuras

Índice de Acrónimos

GNN	Graph Neural Network
GCN	Graph Convolutional Network
CNN	Convolutional Neural Network
FNN	Feed-Forward Neural Network
FC	Fully-connected layer
MLP	Multi-layer Perceptron
NN	Neural Network
DNN	Deep Neural Network
ML	Machine Learning
ETC	Etcétera

Agradecmientos

Para la realización de esta tesis se agradece al asesor de tesis por la ayuda a la creación de la misma. También debo agradecer a mi familia por haberme apoyado en el transcurso de mi estadia en la universidad en la estudio. A su vez dar las gracias a los profesores que me han impartido clases que han sido buenas y dieron lo mejor de si para transmitir conocimientos a lo largo de mi carrera profesional.

Capítulo 1

Introducción

Las Redes Neuronales basadas en Grafos (GNN) son una arquitectura de Red Neuronal (NN) la cual ha sido recientemente introducida en el campo del Machine Learning que esta especializada en el tratamiento de datos con una estructura de Grafo. Un ejemplo de datos con esta estructura son las imágenes y el texto. La primera como una red interconectada donde cada arista puede ser la intensidad de cada pixel. En el caso del texto se puede estructurar como una serie de nodos consecutivos conectados uno tras otro.

Por otro lado las Redes Neuronales Convoluciones (CNN) han demostrado ser buenas en los problemas relacionados a imágenes. Ahora se comparan estas arquitecturas con el fin de saber cual da mejores resultados.

1.1. Motivación

La motivación para este trabajo es la de dar una revisión e implementación de esta arquitectura que esta creciendo en popularidad como son las GNN. Revisar para así conocer los alcances que pueda tener. Implementar para poder probar su eficacia. A su vez se comparará con un tipo de arquitectura más conocido y con una gran cantidad de estudios como son las CNN. Esta última ha demostrado su valia para el procesamiento de imágenes. Sin embargo las GNN tienen un subtipo conocido como GCN que podría tener unos resultados similares. En especial se comparará ambas para el tratamiento de imágenes con alta resolución.

1.2. Objetivos de la Investigación

1.2.1. Objetivo General

Evaluar las arquitecturas GNN, en particular GCN, y CNN en un problema de clasificación de imágenes para de esta forma poder determinar cuál es la que da mejores métricas y es más eficiente.

1.2.2. Objetivos Especificos

- Determinar la eficiencia de ambas arquitecturas al tratar el problema con imágenes de alta resolución.
- Interpretar los resultados obtenidos por los analisis en un problema de clasificación.
- Examinar los tiempos de entrenamiento para ambas arquitecturas.

Capítulo 2

Planteamiento del Problema

2.1. Descripción del Problema

El problema se centra en conocer si las GNNs, en concreto un tipo particular de esta conocido como GCNs, pueden ser mejores o tener resultados comparables a los que se puedan obtener con las conocidas CNNs en un problema de clasificación de imágenes el cual es un problema donde estas últimas han demostrado buenos resultados.

2.2. Formulación del Problema

Se usará una versión especializada de las GNNs en los problemas relacionados a imágenes que hace uso de la convolución. Este tipo se llama GCN. Y se comparará el tiempo de entrenamiento, la velocidad de convergencia a una elección óptima de parametros que de una alta tasa de aciertos en los datos de entrenamiento y, finalmente, comparar los aciertos en los datos de test con el que se obtendrá de la resolución del ismo problema con una CNN.

2.3. Justificación del Estudio

Las redes neuronales (NN) han demostrado ser uno de los metodos del Machine Learning que mejores resultados esta dando. Un tipo particular de red neuronal es la Red Neural Profunda (DNN). Esta esta compuesta por capas, dos de las más conocidas son la capa convolucional o recurrente.

Además muchos de los tipos de datos en importantes componentes de la vida actual como las redes sociales o ramas de la ciencia como la biología tienen una estructura de grafo.

A la vez las imágenes también pueden ser tratadas como un grafo donde cada pixel actua como un nodo.

En estos datos las CNNs han demostrado tener gran performance gracias a caracteristicas tales como la compartición de pesos, el uso de multiples capas la localización de conexiones (LeCun, Yoshua Bengio, y G. Hinton, 2015).

Sin mebargo, otro tipo de red neuronal conocido como GNN que esta especializado en el tratamiento de datos con estructura de grafo, podría dar mejores resultados en el tratamiento de imágenes con alta resolución.

El tratamiento de este tipo de imágenes suele ser complicado debido a la cantidad de pixeles que estas contienen. Aunque la importancia de su tratamiento es crucial pues esta vinculado a importantes areas como la medicina o la biología.

Capítulo 3

Marco Teórico

3.1. Conceptos Previos

3.1.1. Redes Neuronales (NN)

Una Red Neuronal es un algoritmo del Aprendizaje Supervisado que, en su forma más simple, es una serie de capas compuestas por neuronas que estan conectadas entre si. Esto viene inspirado de la biología donde las neuronas transmiten información entre si mediante pulsos electricos en sus conexiones. Esta transmisión de información en las Redes Neuronales se da de manera similar asignando un peso a cada conexión, que en una Red Neuronal simple se conecta cada neurona en una capa con todas las neuronas en la capa siguiente. Los pesos se pueden colocar de manera simplificada en un vector para cada nerona. La unión de estos pesos en forma de vector de cada neurona en una capa forma una matriz conocida como "Matriz de pesosrepresentada con la letra W.

Este algoritmo realiza un proceso de aprendizaje el cual le permite ajustar sus parametros en base a los datos que se le proporcionen. Estos parametros son las matrices de peso de cada capa. De esta forma el proceso de aprendizaje consiste en actualizar estas matrices de pesos. Existen varios algoritmos que realizan estas tarea. Sin embargo, el más usado es el algoritmo de "Back-propagation". Este realiza un proceso de optimización sobre los parametros para minimizar una función de coste. Esta función mide la diferencia entre las salidas que se desean obtener y las que se obtienen al

culminar el proceso del cálculo hacia adelante o "Forward calculation". El proceso de optimización conocido como "Backward propagation.º propagación hacia atras pues se construye una función en base a las derivadas parciales de las capas anteriores.

Las capas de las Redes Neuronales se pueden dividr en en tres grupos de acuerdo a su función y relación con los datos que se le proporcionen:

- **Capa de entrada:** Es la que se relaciona con los datos de los que se busca una predicción.
- **Capas ocultas:** Estas dan la robustez a la red para procesar los datos y la cantidad de neuronas y capas se determina de tal forma que se alcance un nivel de predicci”on aceptable.
- **Capa de salida:** De esta capa salen los valores predichos por la Red Neuronal.

Aunque cone el pasar del tiempo y la realización de más estudios se han creado tipos de capas especializadas en ciertas tareas como la predicción de texto (RNNs) o clasificación de imágenes (CNNs). A continuación se presenta tres tipos que son de vital importancia para este estudio:

- **Feed-Forward Neural Network (FNN):** Es el tipo más simple de Red Neuronal. Consiste en las 3 capas ya mencionadas. Además posee la restricción de que una neurona de una capa solo puede conectarse a las neuronas de las capas vecinas. Un ejemplo de esto se puede ver en la Figura 6.5

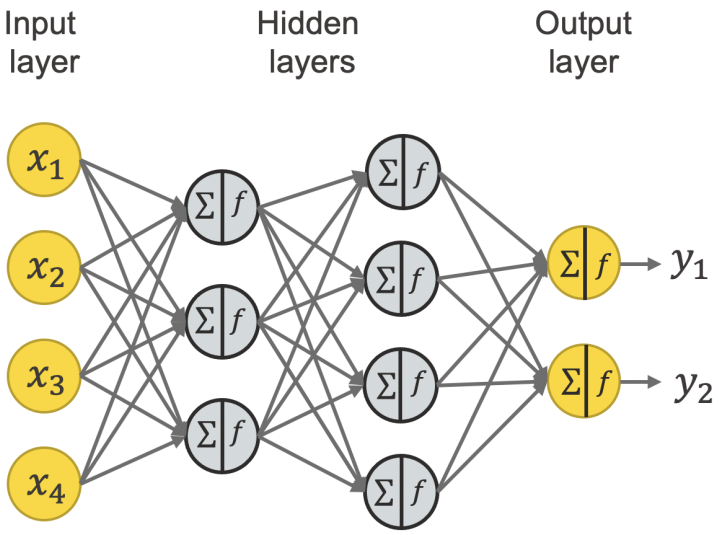

FIGURA 3.1: Feed-Forward Neural Network

- **Red Neuronal Convolucional (CNN):** Esta es un tipo particular de la anteriormente mencionada. Pero a diferencia de esta la conectividad local se preserva. [1]. Este tipo de capa también esta compuesta por subcapas como:

  - Capa Convolucional
  - Capa de reducción de muestra o Pooling Layer
  - Capa simple donde todas las neuronas estan conectadas con las de la capa inmediatamente anterior y posterior.

- **Red Neuronal basada en Grafos (GNN):** Esta capa esta especialmente diseñada para tratar con tipos de datos gráficos o con una estructura no euclidea. Tiene subtipos que pueden aprovechar la operación de convolución conocidos como GCN.

3.1.2. Red Neuronal Convolucional (CNN)

Una Red Neuronal Convolucional esta compuesta, como se mencionó en la sección anterior, por tres tipos diferentes de capas: una capa convolucional, una capa Pool y una capa completamente conectada o Fully-conncected (FC).

Las CNNs se centran en la idea primaria que la entrada conntiene una imagen que enfocan la arquitectura a construir en un método que ajuste con precisión la necesidad de lidiar con una forma particular de la información. Sin embargo, una de las caracteristicas de las CNNs es que las capas intermedias estan compuestas por neuronas organizadas en 3 partes conocidas como la dimensión espacial de la entrada. [2]

A continuación se realizará una revisión de los 3 tipos de caps que componen esta arquitectura especializada en el procesamiento de imágenes y las cuales son cruciales para el entendimiento de esta.

Capa convolucional

Esta capa es la más importante que posee esta arquitectura. Además es la que mayor coste computacional genera. Esta realiza el proceso de la convolución sobre la imagen siguiendo unos parametros que se centran en el empleo de lo que se conoce como kernels de aprendizaje. Estos suelen ser de una dimensionalidad reducida aunque de todas formas recorriendo completamente la entrada. La capa convolucional opera de la siguiente forma sobre las entradas. A cada entrada se le aplica una multiplicación entre los pesos, que son los componentes del kernel, y una región asociada de la entrada para luego sumarlos dando un mapa en forma de matriz de la misma. Una imagen que ilustra este procedimiento es la Figura 3.2. Esta capa posee tres hiperparametros que ayudan a controlar a la misma los cuales son el padding (P), el stride (S) y las dimensiones de la entrada (V). A partir de estos es posible calcular las dimensiones del kernel necesario para generar una salida de tamaño determinado (R) usando la siguiente fórmula:

$$\frac{(V - R) + 2P}{S + 1} \tag{3.1}$$

El proceso de entrenamiento de este tipo de red también hace uso del algoritmo de Backpropagation.

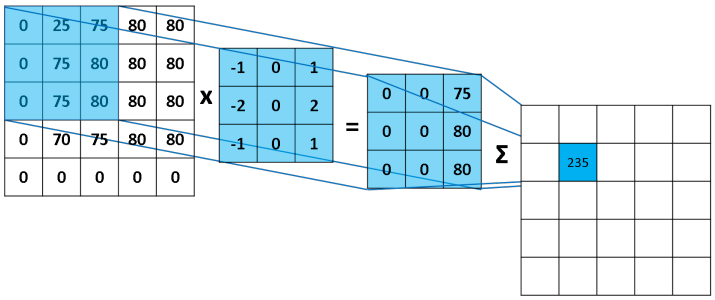

FIGURA 3.2: Proceso de una Capa Convolucional sobre una entrada

Capa de reducción de muestra o Pooling Estas capas también hacen parte de la arquitectura de una CNN y normalmente se le puede encontrar después de una convolucional. Siendo así la salida de la capa convolucional es la entrada de la capa de Pooling. En efecto este tipo de capas se usan para realizar una reducción de la dimensión de sus entradas mediante el uso de funciones como calcular el valor más común o el mayor en una región donde se aplique. La importancia de esta capa radica en que estas pueden reducir la complejidad del módelo. Existen varios tipos de estas capas como son Max pooling basado en hallar el valor máximo de la región u otro como uno llamado Average pooling basado en hallar la media de la región. La imagen que se puede ver en la Figura 3.3 ofrece una explicación visual de como funciona una capa Pool, en concreto una Max pool.

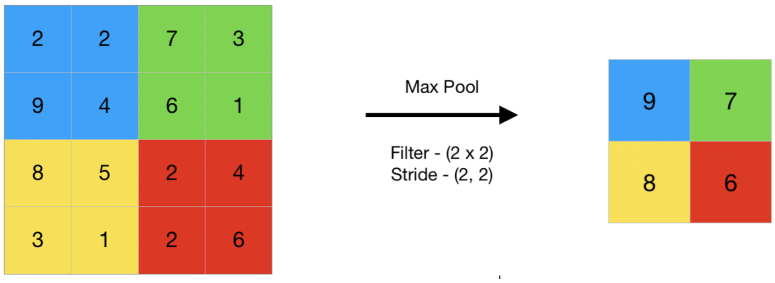

FIGURA 3.3: Proceso de una Capa Max pool sobre una entrada

capa Fully-Connected Esta capa ya se vió en la seccianterior. Esta capa como parte de la arquitectura de las CNNs se coloca al final pues es esta capa no encapsula bien la información espacial. También es usada para aplanar la salida

de las capas de Pooling que llegan en una dimensisuperior a 1. Sin embargo, esta no es indispensable para la creación de la arquitectura de las CNNs pues recientemente algunos diseños reemplazan esta con otra capa llamada General Average Pooling [5].

Una visualización de como funcionan estas capas conjuntamente se puede ver en la imagen de la Figura 3.4

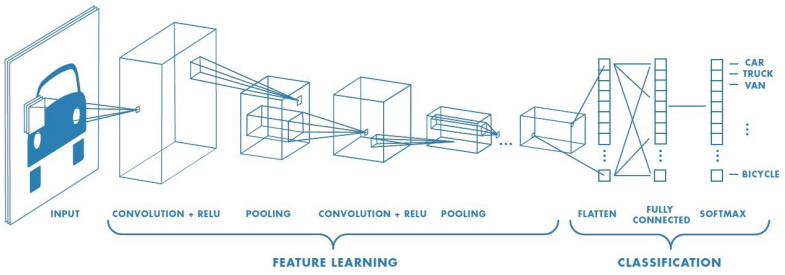

FIGURA 3.4: Fumcionamiento de una CNN

3.1.3. Red Neuronal basada en Grafos (GNN)

Los grafos son una estructura de datos que esta muy presente en la vida real pues esta sirve para representar la connexión entre grupos de objetos. Y es por esto que se han realizado estudios sobre redes neuronales que operan sobre grafos. Este tipo de red neuronal recive el nombre de GNN. Estos estudios llevan realizandose más de una decada. Sin embargo, son los avances recientes los que han aumentado sus capacidades en el poder de representación de la data. Gracias a esto se esta empezando a ver utilidades en campos como la simulación de sistemas físicos como en la detección de nooticias falsas. [3]

Un grafo puede ser representado mediante un conjunto de nodos y las aristas que los unen. Esto a su vez puede representarse con una matriz conocida como Matriz de adyacencia. En este sentido las im<”agenes también pueden ser representadas como grafos. En esta representación cada pixel es un nodo y se rcoloca una arista entre este y los vecinos que posee. Aynque esta representación puede ser muy redundante pues solo se crea una banda en la matriz de adyacencia.

Existen 3 tipos de problemas a resolver en grafos. Estos son a nivel de grafo, nodo o arista. Un ejemplo de un problema a resolver a nivel de grafo relacionado a imágenes es la clasificación de estas. En cuanto a una tarea a nivel de nodo es la segmentación de imágenes. Y un ejemplo a nivel de arista es establecer relaciones entre los objetos de una imagen.

Para representar un grafo se puede utilizar la matriz de adyacencia pero esta no es eficiente pues ocupa mucha memoria y no es única para un mismo grafo llegando a tener varias de estas. Una alternativa a esto es una lista de adyacencia donde el elemento $K - esimo$ es ula representación de la arista que concecta un nodo i con uno J de la forma (i, j). Un ejemplo de esto se puede ver en la Figura 3.5.

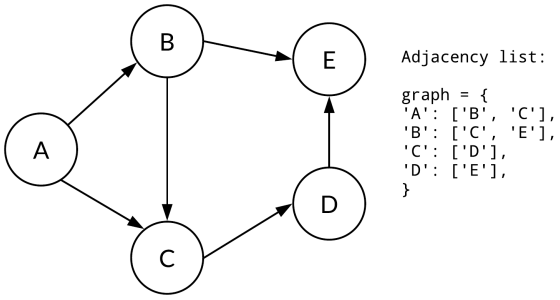

FIGURA 3.5: Representación de la lista de adyacencia de un grafo

Un modelo simple de GNN es pasar los nodos, las aristas o el contexto global del grafo a traves de MLP. De esta forma se puede resolver las distintas problematicas que se mostraron anteriormente y así realizar predicciones. Por ejemplo, si queremos realizar predicciones en el provlema a nivel de nodo, solo es necesario aplicar el MLP entrenado a este. Una forma de visualizar esto se puede ver en la Figura 3.6.

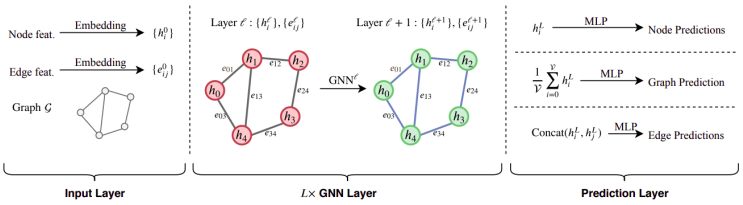

FIGURA 3.6: Representación del funcionamiento de una GNN

A veces no se disponen de la información de todas las partes que se utilizan para representar el grafo, como la de los nodos o aristas, afortunadamente este problema puede solventarse gracias a una técnica coonocida como la transferencia de mensajes. Esta consiste en, generalmente, sumar la información contenida en los objetos adyacentes al que necesitamos suministar información. Por ejemplo, si necesitamos información en los nodos pero solo tenemos el de los vertices se puede obtener esta información como la suma de los vectores de las aristas de las cuales este nodo es parte.

El proceso de transferir mensajes también se puede realizar entre distintas capas de la GNN. Por eje,plo, se puede transferir información de los nodos de la capa anterior sumando a los vectores de los nodos de la nueva capa los vectores de los nodos vecinos a este en la capa anterior por cada nodo en la nueva capa.

Existen diversos subtipos de GNNs. Sin embargo, esta tesis se centrará en las GCNs.

3.1.4. Red Convolucional en Grafos (GCN)

Esta red es parte de las GNNs. Esta hace uso de la generalización de la operación de convolución de las CNNs ,que opera sobre un grafo en forma de rejilla que es como se puede representar a una imagen como grafo, a un tipo más general de grafo. Esto puede resultar complicado pues un grafo, en general, son invariantes bajo el orden de los nodos; es decir, no importa el orden en que estos se elijan. Esta última propiedad otorga flexibilidad en la representación de datos, pero la convolución depende de la posición absoluta de los pixeles. Además de esto la estructura de un nodo a otro cambia sustancialmente pues puede tener diferente número de nodos vecinos. [4]

Una forma de conseguir utilizar la operación de convolución en grafos es mediante la definición de filtros sobre estos como se hace en las CNNs.

Primero se mostrará el Lapalaciano de un grafo que se puede definir como sigue:

$$L = D - A \tag{3.2}$$

Donde A es la matriz de Adyacencia del grafo y D es una matriz diagonal que se define como sigue:

$$D_v = \sum_u A_{vu} \tag{3.3}$$

Esto escencialmente viene a ser que para el nodo v el valor es la suma de esa fila en la matriz de adyacencia. Este Laplaciano representa la misma información que A. Además a partir de A se puede hallar L y viceversa.

Ahora se puede formar polinomios con estos de la siguiente forma:

$$p_w(L) = \sum_{i=0}^{d} w_i L^i \tag{3.4}$$

Ahora este polinomio de grado d puede ser un filtro tal y como se tienen en una CNN con los coeficientes w como los pesos. Ahora si consideramos a la informaciasociada a un nodo como un valor único real; entonces, podemos concatenar estos para cada nodo obtenundo un vector x. De esta forma aplicar el filtro obteniendo un vector x' es equivalente a:

$$x' = p_w(L)x \tag{3.5}$$

Un dato importante a considerar aobre el grado d del polinomio es que este afecta a como se realiza el filtro en un nodo v, esto es solo se realiza entre nodos que no esten separados más de d pasos.

Se dice que un algoritmo f es equivariante si dada una permutación P sobre la entrada x se cumple que:

$$f(Px) = Pf(x) \tag{3.6}$$

Se puede ´robar que este polinomio es equivariante del orden de los nodos. Por tanto, esta forma de definir la convolución es equivariante. Además se puede ver que con un polinomio de grado d un nodo v solo afecta otro nodo u como ya se mencionó antes. Esto es similar a una transferencia de información entre los nodos. Si este paso se repitiese k veces, entonces el mensaje se propagará a

todos los nodos del grafo que esten a una distancia k entre si.

Una manera de diferenciar a los diferentes tipos de GNNs es mediante el cálculo del embedding. Como esta tesis se centrará en el uso de las GCN este será el que se presentará a continuación [4]:

$$h_v^{(0)} = x_v \qquad \forall\, v \in V \tag{3.7}$$

Aquí $h_v^{(0)}$ es el embedding inicial del nodo v. Para K iteraciones se pueden calcular los enbeddings como sigue:

$$h_v^{(k)} = f^{(k)} \left(W^{(k)} \cdot \frac{\sum\limits_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)} \right) \qquad \forall\, v \in V \tag{3.8}$$

Notese que tanto $f^{(K)}$ como $W^{(K)}$ y $B^{(K)}$ son los mismos para todos los nodos. Las predicciones pueden obtenerse como sigue:

$$y_v = PREDICT\left(h_v^{(K)}\right) \tag{3.9}$$

Aquí PREDICT suele ser otra red neuronal que trabaja junto a la GCN. Esta formulación de la normalización del embedding es diferente a la dada en el paper original donde se presentaron las GCN.

3.2. Marco Histórico

El área del Aprendizaje de Máquina, Machine Learning en inglés(ML), ha tenido un crecimiento pronunciado en la última década a partir del desarrollo de las Redes Neuronales (NN). Inicialmente se utilizaba el Perceptron Multicapa (MLP) para los problemas de clasificación. Luego se dió un nuevo avance en este tipo de problemas gracias a las Redes Neuronales Convolucionales (CNN) que se especializan en los referidos a imágenes.En el presente un tipo de arquitectura nuevo especializado en los problemas con

datos estructurados como grafos. Esta arquitectura se llama Graph Neural Network (GNN). Esta arquitectura fue presentada en 2009 en el paper titulado "The Graph Neural Network Model"[6]. Ahora las GNNs son potenciadas gracias al uso de la convolución tomando el nombre de Convolutional Graph Network (GCN).

3.3. Investigaciones Relacionadas

Se dará un resumen de los articulos o tesis que sustentan este trabajo dando los resumenes y los objetivos de cada uno.

3.3.1. An Investigation Into Graph Neural Networks [1]

Objetivos

- Dar una revisión de acerca de las GNNs.
- Describir las librerias y frameworks necesarios para crear una GNN y como implementarla.

Resumen

En esta tesis desarrollada por V. Kumar se da una revisión de las GNNs. Dando una sustentación teórica de las mismas. Además se da un resumen de las librerias y frameworks que son necesarios para el desarrollo de las GNNs. También se dan las limitaciones de las mismas. Luego se implementan para resolver distintos tipos de tareas sobre datasets conocidos como lo son MNIST y Cora.

Concluciones

- Es complicado encontrar la libreria correcta para cada experimento.
- A pesar de que las GNNs no dab resultados satisfactorios en muchas condiciones, estas logran conseguir resultados destacables en otros.

3.3.2. An overview of convolutional neural network [2]

Objetivos

- Realizar una revisión de de los fundamentos detras de la arquitectura CNN.
- Describir las aplicaiones de las CNNs.

Resumen

En este paper se realiza una revisión de la arquitectura de las CNNs. Primero se presenta los fundamentos de la arquitectura. Luego se presenta algunas revisiones de la arquitectura más complejas. Finalmente se dan algunos casos de aplicaciones.

Concluciones

- Se revisó la arquitectura de las CNNs y sus aplicaciones.
- Las CNNs han demostrado ser de las mejores arquitecturas para las tareas relacionadas al campo de la Visión Computacional.

3.3.3. A Gentle Introduction to Graph Neural Networks [3]

Objetivos

- Explorar y explicar la arquitecturade las modernas GNNs.

Resumen

En este trabajo se detalla de una manera simple y entendible los conceptos principales concernientes a las GNNs. Se divide en 4 partes. En la primera se da una explicación y eje,plos de los tipos de datos que son mejor representados por grafos. En la segunda se da yna explicación de que hace diferente a los grafos de otros tipos de datos. Para la tercera se construye una GNN explicando en este proceso los conceptos de la misma. Por último se provee una representación

animada de una GNN en la cual se puede ajustar sus parametros para un mayor entendimiento de su funcionamiento.

Concluciones

- Los grafos son una estructura de datos poderosa que genera retos diferentes a los vistos con imágenes y texto.
- Se revisó en el articulo algunas de las decisiones de diseño importantes que se deben tener en cuenta al momento de diseñar una GNN.

3.3.4. Understanding Convolutions on Graphs [4]

Objetivos

- Ilustrar los retos de la compuutación relacionada a grafos.
- Explorar las variantes más recientes de las GNNs.

Resumen

En este articulo se da una revisión de la arquitectura de las GNN. Además se da una explicación de la problematica del uso de la convolución convencial en el tratamiento de grafos. En particular se trata el problema de la falta de orden en los nodos. A esto se muestra una solución mediante la extensión de la noción de laplaciano a grafos. También se muestran variantes de las GNNs como lo son las GCNs.

Concluciones

- En el articulo se da una muestra del campo amplio de las GNNs a la vez de mostrar varias técnicas e ideas relacionadas a este campo.
- Se comunicaron las ideas principales para entender las GNNs y algunas de sus variantes.

3.4. Hipótesis

3.4.1. Hipótesis General

Las GNNs, en particular las GCNs, tienen un rendimiento comparable con las CNNs en lo que refiere al problema de clasificación de imágenes.

3.4.2. Hipótesis Específica

- El tiempo de entrenamiento de las CNN es mayor al de las GNNs.
- La arquitectura GCN da mejores resultados que las CNN en el problema de clasificación de imágenes.

Capítulo 4

Metodología de Trabajo

En esta sección se mostrará la forma en la que se procederá a evaluar el problema de la comparativa de las CNNs con las GNNs. En este sentido se dividirá este capítulo en 4 secciones. En la primera se detallará el problema en el cual se compararan. Seguidamente se especificará los datasets a usar. Luego se detallará la elección de los modelos. Finalmente se mostrará la forma de comparar en base a determinadas métricas.

4.1. Problema

Como se ha ido mencionando en secciones anteriores, se eligió el problema de clasificación de imágenes. El motivo de esto se debe a que las CNNs estan especializadas en esta problematica y se busca saber cual es el rendimiento de las GNNs ante estas.

4.2. Datasets

Debido al problema escogido se hará uso del dataset de MNIST pues este es sencillo y no es necesaria la utilización de altos requerimientos de hardware para entrenar un modelo que pueda tener buenos resultados respecto a este dataset.

4.3. Elección del modelo

Primero se tendrá una base de una clasificación realizada por un MLP. Se comparara el modelo más simple de CNN con un subtipo de GNN. Para la elección del subtipo de modelo de GNN se tendrá en cuenta la complejidad del modelo y el que más parecido sea a las CNNs.

4.4. Métricas

Para la comparación se obbtendrá la precisión y la perdida en cada época para ambos modelos. Luego este se graficará para la observación de cual obtiene mejores resultados en los datos de evaluación y la rapidez con la cual los obtiene.

Capítulo 5

Diseño e Implementación

En este capítulo se mostrará las herramientas usadas para la implemntación y evaluación de los modelos. Asi como el diseño e implementaci
on de los modelos. para ver detalladamente el código implementando los modelos de esta tesis, así como los datasets usados y los modelos entrenados revisar el siguiente apéndice A.

5.1. Herramientas

5.1.1. Software

PyTorch

Es un framework de diferenciación automática que se uso para la implementación de los modelos pues cuenta con funciones que facilitan el proceso de creación de redes neuronal complejas como es el caso de las CNNs.

PyTorch Geometric

Es una libreria basada en el framework PyTorch especializada en el diseño de Redes Neuronales basadas en Grafos o GNNs. En la presente tesis se hace uso de este para la implementación de un subtipo de GNN conocido como GCN.

Matplotlib

Es una libreria especializada en la creación y visualización de gráficos. En este trabajo se utilizó para realizar la visualización de las gráficas de la evolución de la función de perdida y la precisión a lo largo de las epocas.

Torchvision

Esta libreria es una extensión del proyecto PyTorch. Se especializa en el tratamiento de imágenes y tiene datasets relacionados al campo de visión artificial. En este caso se extrajo el dataset MNIST con esta libreria.

NumPy

Esta libreria se especializa en la computación científica en Python. En lo referente a este trabajo se utilizan funciones como la media para calcular las métricas o el reescalado de dimensiones para el tratamiento del dataset.

5.1.2. Hardware

Es vital el uso de altos requerimientos de hardware para el proceso de entrenamiento de modelos de aprendizaje automático. Existen varias formas de entrenar usando este tipo de hardware como puede ser ejecutarlo de manera local o remota usando herramientas en la nube como AWS. En este caso se optó por este último mediante el uso de Google Colaboratory. Esta herramienta provee hardware en la nube de manera gratuita bajo limitaciones de tiempo de uso.

5.2. Diseño de los modelos

5.2.1. MLP

Para este tipo de Red Neuronal se consideró como entrada la misma cantidad de neuronas que de pixeles en la imagen. Y solo se considera una capa

oculta de 100 neuronas. Para la salida solo se consideran 10 neuronas pues se trata de clasificación a 10 clases.

5.2.2. CNN

La arquitectura pensada para este modelo se compone de 2 bloques formados de una capa convolucional, una función de activación, ReLU en este caso, y una de pooling. Por último se coloca una capa lineal con 10 neuronas para dar la clase de la imagen.

5.2.3. GNN

En el caso de esta arquitectura se considera la variante de GNN conocida como GCN. Esta variante implementa la operación de convolución sobre un grafo que es lo que se explicó que se necesitaba en la metodología. Se considera 5 capas convolucionales GCN y 2 capas lineales para llegar a una salida de una dimensión que nos de la clase de la entrada.

5.3. Implementación de los modelos

5.3.1. MLP

Para la implementación de esta arquitectura se utiliza la función Linear del modulo nn de PyTorch que contiene los tipos de capa especificados en el diseño de esta y la función de activación. Esto se estructura en una clase que hereda la superclase torch.nn.module.

5.3.2. CNN

Esta arquitectura se implementó gracias al modulo nn de PyTorch usando las funciones Linear para las capas lineales, para la capa convolucional se utiliza Conv2D y para la capa de pooling se usa MaxPool2D que es el max pooling.

Para la función de activación se utiliza la función ReLU. Finalmente esto se estructura en una subclase de torch.nn.module.

5.3.3. GCN

Se implemntó usando PyTorch Geometric que contiene el modulo nn con la capa GCNConv que es la que se necesitaba. Además se usa el modulo functional de PyTorch para la función de activación ReLU, esto en vez del modulo nn pues la primera devuelve un objeto con la propiedad de obtener el tamaño del mismo, lo cual es necesario para la entrada de la capa GCNConv siguiente. Esto se estructura en forma de clase como una clase hija de la superclase torch.nn.module.

5.4. Evaluación

Para la evaluación de la capacidad de los modelos se consideró el dataset MNIST que esta constituido de digitos manuscritos y se clasifican en 10 clases. Este se obtiene gracias al modulo datasets de Torchvision. El tratamiento de este dataset varia con el modelo que se usa y se tomaran solo 5 épocas debido a la falta de recursos computacionales para realizar más pruebas. Esto se presenta en una subsección. También se presenta la métrica usada.

5.4.1. Tratamiento del dataset

MLP

Las imágenes del dataset obtenido por el modulo datasets de Torchvision tienen dos dimensiones, lo cual es problematico para la capa lineal. Por tanto se cambian las dimensiones de esta y se divide en batches para mejorar el entrenamiento.

CNN

Los datos obtenidos por Torchvision son los que se necesitan para el entrenamiento de este tipo de arquitectura que acepta imágenes de 2 dimensiones.

GNN

Para esta arquitectura fue necesario un tratamiento especialde los datos obtenidos pues esta necesita que los datos se estructuren como un grafo. Con este fin los pixeles que esten bajo un umbral se consideran con el valor de -1. En este caso el umbral es 0.También se agrega un padding de 2 a cada imagen con el fin de que cada pixel tenga una vecindad ígual de 8 pixeles. Luego se enumeran los nodos diferentes a -1 de forma ascendente iniciando por 0. Estos se colocan en una lista que contiene dos parametros indicando su posición. Y la lista de adyacencia para cada nodo se compone los nodos que rodean a los que tengan valores diferentes a -1.

5.4.2. Métricas

La métrica que se utiliza para comparar estos modelos es el Accuracy. Esta métrica es de las más sencillas y es la media de errores en validación. También se utiliza los valores de la función de perdida para saber la rapidez de convergencia.

Capítulo 6

Presentación de Resultados

En este apartado se presentan los resultados obtenidos de la comparativa de modelos. En este sentido se mostraran las gráficas de la perdida y el Accuracy.

Primero se muestra como base los resultados obtenidos por el MLP como base de comparación.

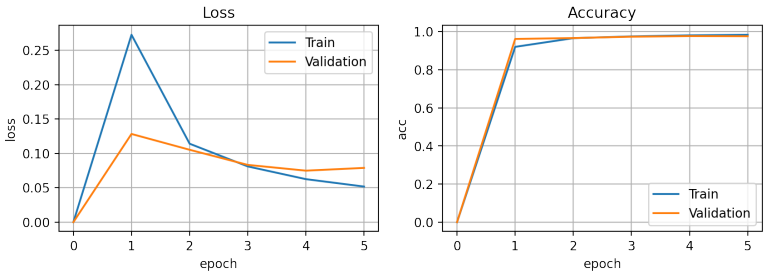

FIGURA 6.1: Resultados del MLP

6.1. Comparativas

6.1.1. MLP y CNN

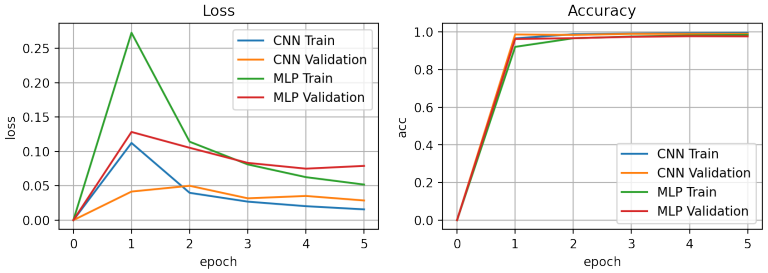

FIGURA 6.2: Comparación entre MLP y CNN

6.1.2. MLP y GCN

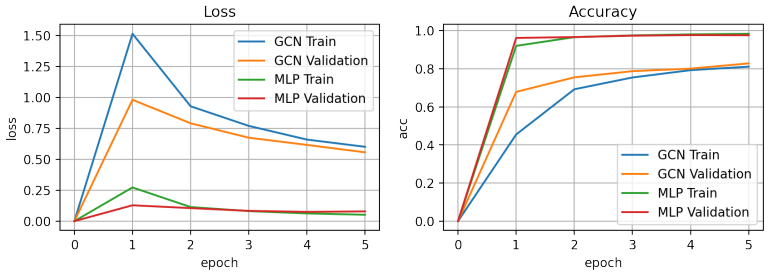

FIGURA 6.3: Comparación entre MLP y GCN

6.1.3. CNN vs GCN

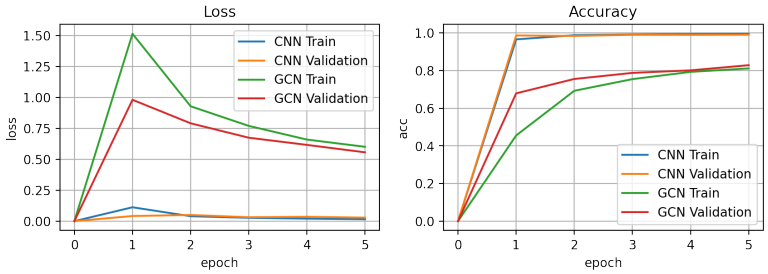

FIGURA 6.4: Comparación entre CNN y GCN

También se realiza un entrenamiento del modelo de GCN con 20 épocas para mostrar mejores resultados. Se muestran los resultados a continuación.

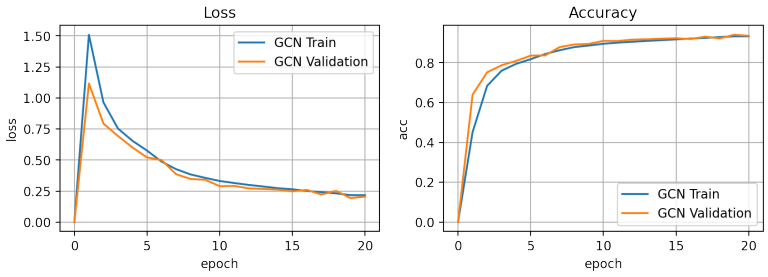

FIGURA 6.5: Modelo de GCN con 20 épocas

6.2. Resultados

- En la comparativa en CNN y MLP se puede observar que el modelo que utiliza CNN converge más rápido que el otro.

- En la comparativa en GCN y MLP se puede observar que el modelo que utiliza MLP converge más rápido que el otro.
- En la comparativa en CNN y GCN se puede observar que el modelo que utiliza CNN converge más rápido que el otro.
- Se puede notar que la convergencia del modelo con GCN tarda más que los otros modelos. Aun que como se puede notar con 20 épocas de entrenamiento ya posee resultados comparables a los otros modelos.

Capítulo 7

Conclusiones y Trabajos Futuros

En este apartado se muestran las conclusiones después de haber realizado la experimentación en los modelos MLP, CNN y GCN. También se muestran los posibles trabajos futuros a realizarse a partir de lo obtenido en este trabajo.

7.1. Conclusiones

- Después de haber realizado las comparativas entre las CNNs y GNNs, en particular la GCNs, se puede concluir que este último modelo obtiene peores resultados en la misma cantidad de épocas.
- Las GCNs tardan más tiempo en converger a mejores resultados que las CNNS en el problema de clasificar las imágenes del dataset MNIST.
- Las GCNs obtuvieron peores resultados que el MLP tras el transcurrir de 5 épocas.
- Se mostró que las GCNs con más épocas de entrenamiento, en el caso que se probó son 20 épocas, consiguen resultados parecidos a los obtenidos con los otros modelos.

Ahora se presentan los trabajos futoros a realizarse a partir de lo obtenido como conclusión en la presente tesis.

7.2. Trabajos Futuros

- Se propone realizar una comparativa con más datasets.
- Se propone realizar entrenamientos que puedan poseer más épocas en el entrenamiento de las GCN.
- Se propone comparar una arquitectura que pueda combinar el uso de las GCNs y CNNs.

Bibliografía

[1] Vishal Kumar. *An Investigation Into Graph Neural Networks*. PhD thesis, Trinity College Dublin, Ireland, 2020.

[2] Shadman Sakib, Nazib Ahmed, Ahmed Jawad Kabir, and Hridon Ahmed. An overview of convolutional neural network: its architecture and applications. 2019.

[3] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B Wiltschko. A gentle introduction to graph neural networks. *Distill*, 6(9):e33, 2021.

[4] Ameya Daigavane, Balaraman Ravindran, and Gaurav Aggarwal. Understanding convolutions on graphs. *Distill*, 6(9):e32, 2021.

[5] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Apéndice A

Implementación realizada

Para visualizar el código, los datasets y los modelos entrenados puede referirse al siguiente repositorio de github:

https://github.com/jesusmiguel123/Tesis