Operador Neuronal de Fourier

Óscar Alvarado
OscarAlvarado@ciencias.unam.mx

Resumen

En este trabajo se presenta una metodología para resolver sistemas de ecuaciones diferenciales del tipo Navier-Stokes mediante aprendizaje profundo, creando una red con capas de Fourier y redes neuronales convolucionales. Se hicieron pruebas con datos simulados mediante algoritmos convencionales de discretización del espacio de la ecuación de Burger en una dimensión y de las ecuaciones de Navier-Stokes en 2 dimensiones espaciales y una dimensión temporal. Se obtuvieron resultados de la norma l2 cercanos a 0.01 para el conjunto de datos de la ecuación de Burger y cercanos a 0.1 para el conjunto de datos de las ecuaciones de Navier-Stokes.
Palabras clave: CFD, Deep Learnig, Aprendizaje Profundo, Navier-Stokes, Burger, Fourier.

1. Introducción

Los fenómenos de transporte han sido estudiados desde hace décadas y es de gran interés para la comunidad científica, ya que en la vida diaria nos vemos rodeados de estos fenómenos. El estudio de estos fenómenos inevitablemente lleva a las ecuaciones de Navier-Stokes, ecuaciones que gobiernan esa área de manera generalizada. Por años se han estudiando estas ecuaciones mediante metodología de discretización del espacio y resolución iterativa de sistemas de ecuaciones lineales algebráicos, lo que para sistemas grandes conlleva a un tiempo que a veces puede ser imposible de disponer, aún con buenas estructuras computacionales.

Para resolver esto, se han presentado diferentes estrategias de aprendizaje máquina y se ve todo un futuro por delante [1]. Tal como lo explican en [2], se ha desarrollado un trabajo [3] en los últimos años que ha presentado mejoras respecto a los otros algoritmos, del cuál nos basamos en este trabajo para poder desarrollar los resultados presentados. En dicho trabajo desarrollan la idea de aprender el operador que mapea de un espacio a a un estado u que corresponde a la evolución de un estado a otro mediante las leyes de la física, por lo que hablan de alimentar a la red con esta evolución dada por algoritmos convencionales y aprender ese comportamiento.

En los artículos [4][5][6] y [7] se hablan de otras estrategias para realizar esta tarea.

1.1. Datos

Para este trabajo se usaron dos conjuntos de datos diferentes, uno para la ecuación de Burger y otro para las ecuaciones de Navier-Stokes[8]. Para la ecuación de Burger se tienen 1200 experimentos diferentes, cada uno consiste en un arreglo de 1024 elementos planos, es decir, en una dimensión, como una lista. Cada uno de estos experimentos contiene un estado a y un estado u, que corresponden a un estado inicial y un estado final, respectivamente. El estado inicial corresponde a las condiciones iniciales del objeto de estudio y el estado final corresponde a la solución de la ecuación de Burger con estas condiciones iniciales. Un ejemplo de este conjunto de datos se muestra en la Figura 1.

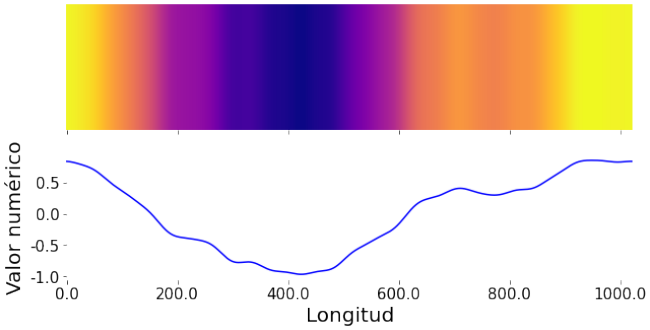

FIGURA 1: Ejemplo de arreglo de los datos de la ecuación de Burger.

Para las ecuaciones de Navier-Stokes, se tiene un conjunto de datos muy parecido al mencionado anteriormente, sólo que en este caso se tienen 3 dimensiones en lugar de 1. Los datos corresponden a 1200 experimentos, donde cada uno de estos consiste en arreglos 2-dimensionales de 64×64 y además una dimensión temporal de 20 pasos, que se puede ver como tensores de 3 dimensiones. Estos datos podían partirse en distintos conjuntos a y u dado un intervalo de tiempo, es decir, el conjunto a puede consistir de un arreglo de condiciones iniciales y su evolución temporal a los siguientes 10 pasos de tiempo, donde nos quedaría como conjunto u el experimento pero a los últimos 10 pasos temporales., esto

sin la necesidad de ser mitad y mitad, podrían ser 5 y 15 o 15 y 5. Un ejemplo de condiciones iniciales para este conjunto de datos se muestra en la Figura 2.

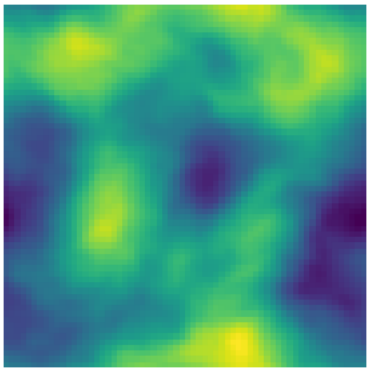

FIGURA 2: Ejemplo de arreglo de los datos de las ecuaciones de Navier-Stokes. Cada experimento cuenta con 20 de estos arreglos, uno por cada paso de tiempo.

Como los datos son generados automáticamente mediante un algoritmo convencional de mallado del espacio, no se tienen inconsistencias o valores nulos.

2. Metodología

Los datos sólo se tuvieron que partir en conjunto de entrenamiento y de pruebas, además de decidir la partición de intervalos de tiempo para los conjuntos a y u. De los 1,200 experimentos diferentes, para el conjunto de entrenamiento se destinaron 1,000 experimentos y 200 para el de pruebas, además de una partición equivalente para cada conjunto a y u, por lo que se entrenó con un conjunto de dimensiones $1,000 \times 64 \times 64 \times 10$ para el conjunto a y $1,000 \times 64 \times 64 \times 10$ para el conjunto u, que a su vez se partieron en conjunto de entrenamiento y de validación que dieron como resultado cuatro conjuntos de dimensiones $800 \times 64 \times 64 \times 10$ y $800 \times 64 \times 64 \times 10$ para el conjunto de entrenamiento, y $200 \times 64 \times 64 \times 10$ y $200 \times 64 \times 64 \times 10$ para el conjunto de validación, de modo que al probar con el conjunto de pruebas, la red nunca haya visto estos datos.

Con los datos particionados de esta manera, lo que siguió fue el armado de la red de acuerdo a la teoría. Este procedimiento se llevó a cabo mediante el *framework* de *pytorch* y la metodología de creación de clases y de bloques vistas en clase. La arquitectura propuesta a partir de la teoría se puede visualizar en la Figura 3.

Esta arquitectura consiste en dos transformaciones lineales, una llamada P a la entrada y otra llamada Q a la salida, que básicamente llevan nuestros arreglos de entrada a una dimensión más grande y más pequeña, respectivamente, de modo que las dimensiones de a y de u sean las mismas. Además de estas dos capas, quedan por ver las capas de Fourier,

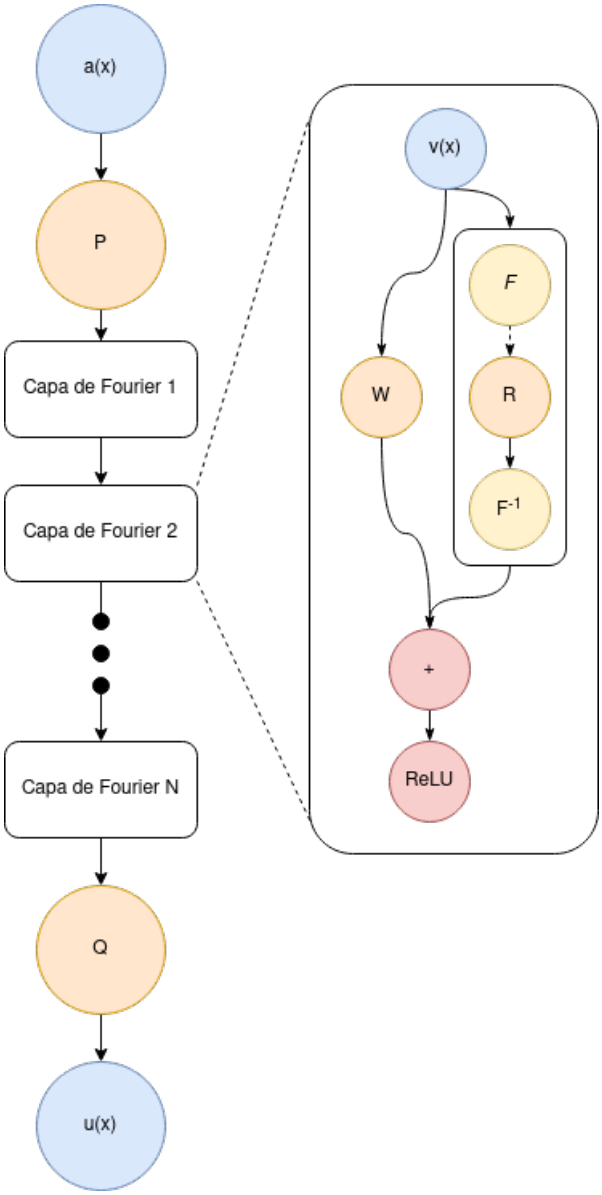

FIGURA 3: Arquitectura de la red utilizada. Las letras P y Q representan capas tipo *Linear*, la letra W representa una convolución, la letra F representa una transformada de Fourier y la letra R representa una multiplicación matricial.

como le llama el autor del artículo, que en la misma Figura 3 se puede visualizar lo que contiene, que básicamente son dos ramificaciones, una que lleva a los datos de entrada a una W que representa a una convolución de una dimensión con 20 filtros, así como los canales de entrada. En la otra ramificación tenemos tres componentes, que representan a una transformada rápida de Fourier, una multiplicación matricial con la cuál obtenemos los modos principales de los datos y por último una transformada rápida de Fourier inversa, con lo que traemos los datos de nuevo al espacio en el que iniciamos desde el espacio de frecuencias, donde se facilitó

la obtención de los modos principales. Al final se hace una suma de ambos resultados de las ramificaciones y se aplica una función de activación ReLU. La teoría relevante para entender el por qué de la estructura de esta red se encuentra en el artículo ya mencionado. La función de pérdida que se utilizó para esta arquitectura fue la norma l2, y como métrica se utilizó el *mse*.

La experimentación consistió en variar el número de capas de Fourier de la arquitectura, principalmente; así como la tasa de aprendizaje, la partición de los pasos temporales de los conjuntos a y u y las dimensiones de mapeo dadas por las capas que llamamos P y Q. Con las capas de Fourier se probaron entre 2 y 8, con los intervalos de tiempo se probó con 5, 10 y 15 para el conjunto a y su complemento de cada uno de estos a 20 para el conjunto u; para la tasa de aprendizaje se probaron valores entre 0.01 y 0.0005, y los pesos para las capas P y Q se variaron entre 10 y 40. El mismo procedimiento se realizó tanto para los datos de la ecuación de Burger como para los de las ecuaciones de Navier-Stokes. Para cada conjunto de datos se probó con diferente número de épocas, ya que hasta cierto punto parecía que el error podría bajar más aún.

Los datos de las ecuaciones de Navier-Stokes con los que se entrenó la red corresponden a un fluido con viscosidad igual a 1×10^{-3}. Y una vez habiendo entrenado la red, se creó otro conjunto con el que se probaron los modelos, los cuáles corresponden a un flujo con viscosidad igual a 1×10^{-4}.

3. Resultados y análisis

Los resultados del proceso de entrenamiento de la red se

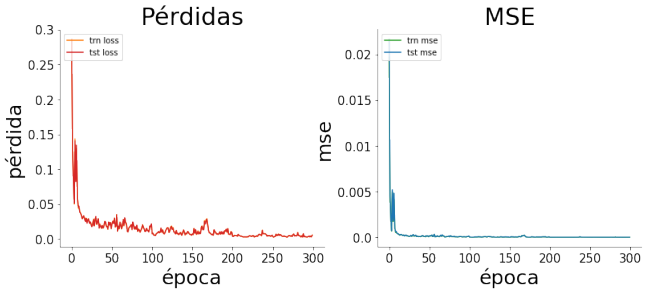

FIGURA 4: Resultados del entrenamiento de la arquitectura propuesta para los datos de la ecuación de Burger.

muestran en las Figuras 4 y 5 con el máximo número de época con el que se experimentó y para los mejores resultados, que en general fueron con 4 capas de Fourier (aunque mejoraba mientras más capas eran, pero sacrificando muchísima velocidad por poca mejora en resultados), una tasa de aprendizaje de 0.001, variable *width* igual a 20 (muy importante, ya que empeoraba si se alejaba mucho de este valor, ya sea por arriba o por abajo) y un intervalo de tiempo de 10 pasos, que en general no hacía variar tanto a los resultados, pero estaba mejor balanceado de esta manera, ya que uno da 10 pasos de tiempo y recibe otros 10 para el futuro.

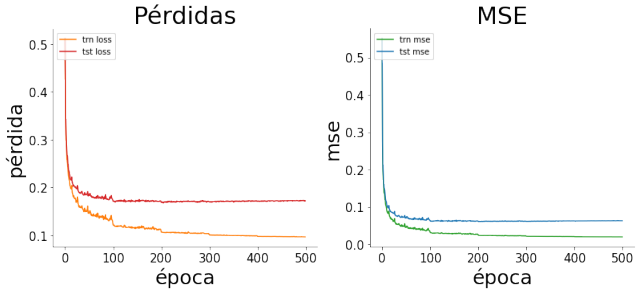

FIGURA 5: Resultados del entrenamiento de la arquitectura propuesta para los datos de las ecuaciones de Navier-Stokes.

La Figura 6(a) muestra el resultado de una predicción del conjunto de pruebas, que nunca vio la red, la Figura 6(b) representa el valor verdadero para los datos con los que se predijo la predicción antes mencionada. La Figura 6(c) muestra el valor absoluto de la diferencia entre las Figuras 6(a) y 6(b), de donde podemos observar que el mayor error se obtuvo en los cambios bruscos en la propiedad física que se esté estudiando, que podría ser la temperatura, por ejemplo.

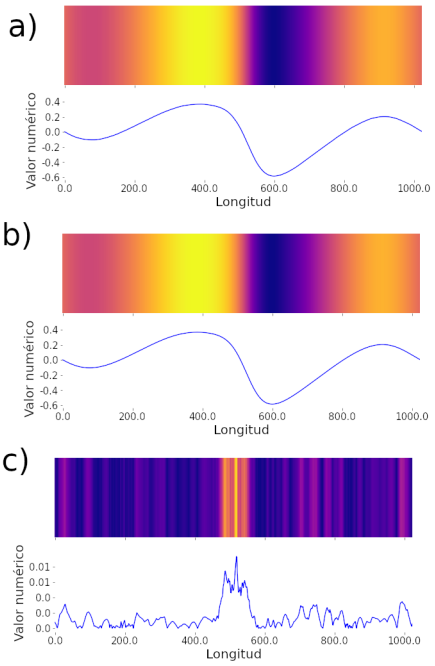

FIGURA 6: a) Se muestra un ejemplo de predicción para la ecuación de Burger.
b) Se muestran un ejemplo del valor verdadero correspondiente a la predicción en (a).
c) Se muestra el valor absoluto de la diferencia entre (a) y (b).

En la Figura 7 Se muestran dos capturas de diferentes tiempos, tanto para la predicción, los valores verdaderos y el valor absoluto de la diferencia entre estos dos, de izquierda a derecha respectivamente. Observamos que mientras más grande sea el valor del tiempo, los errores se van propagando, yendo desde un error mínimo de 0.05 para el tiempo 0 hasta un error máximo de hasta 2.0 para el tiempo 9.

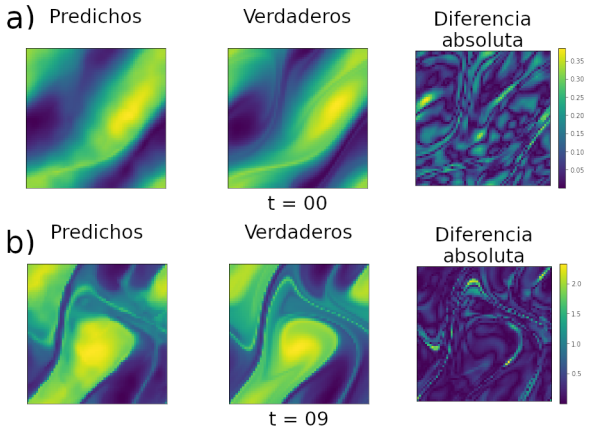

FIGURA 7: Resultados obtenidos para el conjunto de pruebas de los datos de las ecuaciones de Navier-Stokes.
a) Se muestra el valor predicho, el verdadero y el valor absoluto de la diferencia entre los dos anteriores para el tiempo 0, es decir, una predicción a futuro de un paso de tiempo.
b) Se muestra el valor predicho, el verdadero y el valor absoluto de la diferencia entre los dos anteriores para el tiempo 9, es decir, una predicción a futuro de diez pasos de tiempo.

En la Figura 8 se tienen los resultados de probar el modelo con un fluido completamente diferente de aquel con el que se entrenó, se observa que los resultados de la diferencia absoluta entran dentro del rango que había para el fluido con el que se entrenó, e incluso son mejores respecto a la propagación del error en el tiempo.

4. Conclusiones

Los resultados obtenidos parecen tener un comportamiento como el que se esperaría según las leyes de la física, se podría decir que está aprendiendo una aproximación del operador que mapea de a a u. Como trabajo a futuro se propone hacer normalización por lotes para mejorar la propagación en los errores con el paso del tiempo, también se recomienda un aumentado de datos o simulación de otros con una ventana temporal más amplia para poder disminuir el error con el paso de tiempo. Como vimos en las gráficas de resultados, se puede pensar en el uso de menos épocas para hacer más experimentación con los hiperparámetros, ya que tarda mucho, aún cuando se tiene uso de GPU QUADRO RTX 4000.

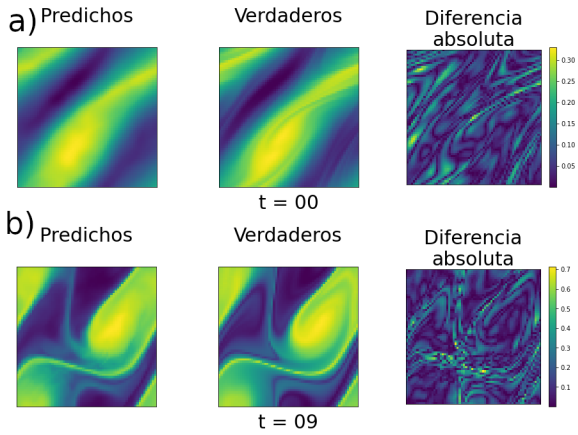

FIGURA 8: Los mismos resultados que la Figura 7 pero probando el modelo con un fluido con viscosidad de 1×10^{-4}.

Dentro de la experimentación a futuro, se propone probar con otras resoluciones de imágenes y ver cómo se comportan los errores respecto a esto, además de probar con resoluciones diferentes a aquellas con las que se esté entrenando la red para ver si en efecto se está aprendiendo un operador general y no únicamente para una resolución y una familia de soluciones.

Por otro lado, se propone que se hagan pruebas con los mismos resultados que se están teniendo, es decir, si se tiene una predicción a futuro con un paso de tiempo de 10, predecir a futuro con esas mismas predicciones. Por último, se propone probar con fluidos más diferentes habiendo entrenado la red con un tipo de fluido fijo, es decir, ya que entrenamos con un fluido con viscosidad de 1×10^{-3}, ver cómo resuelve el comportamiento de fluidos con una viscosidad de 1 o algo así.

Referencias

[1] https://sci-hub.do/10.1017/jfm.2016.803

[2] https://www.infoq.com/news/2020/12/caltech-ai-pde/

[3] https://arxiv.org/pdf/2010.08895.pdf

[4] https://arxiv.org/pdf/2008.10509.pdf

[5] https://www.jmlr.org/papers/volume19/18-046/18-046.pdf

[6] https://arxiv.org/pdf/1904.07200.pdf

[7] https://arxiv.org/pdf/2004.08826.pdf

[8] https://github.com/zongyi-li/fourier_neural_operator