Reducción de complejidad computacional en simulaciones de dinámica de fluidos mediante Aprendizaje Profundo y Representación Simbólica

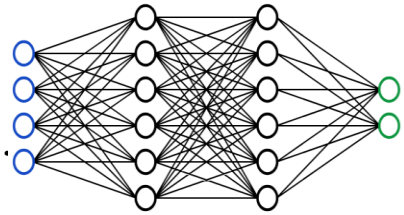
Presentación TFG – Inteligencia Artificial
Jose Antonio Guerrero Barberán
UOC

Introducción: Contexto y justificación

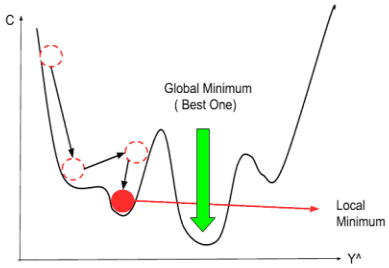
- **Importancia de las PDE**: Fundamentales para modelar fenómenos en ciencia e ingeniería, las PDE representan cambios en múltiples variables interrelacionadas.
- **PDE vs. ODE**: A diferencia de las ecuaciones diferenciales ordinarias (ODE) con una sola variable, las PDE abordan múltiples variables, siendo clave en fenómenos físicos complejos.
- **Aplicaciones de PDE**: Cruciales en física, ingeniería, economía, etc., para modelar procesos como propagación de calor, fluidos y deformaciones sólidas.
- **Métodos Numéricos para PDE**: Debido a la complejidad, las soluciones analíticas son poco comunes, recurriendo a métodos numéricos para aproximaciones precisas.
- **Dinámica de Fluidos Computacional (CFD)**: Especialización que utiliza métodos numéricos para resolver la ecuación de Navier-Stokes, modelando fluidos en estado líquido y gaseoso.
- **Desafíos de Granularidad en Simulaciones**: La precisión aumenta con la granularidad, pero también lo hace el coste computacional.
- **Propósito del Trabajo**: Utilizar este contexto como marco de aplicación y estudiar cómo el Aprendizaje Computacional puede mejorar la eficiencia y precisión en simulaciones de fluidos.

Introducción: Inteligencia Artificial y Redes neuronales

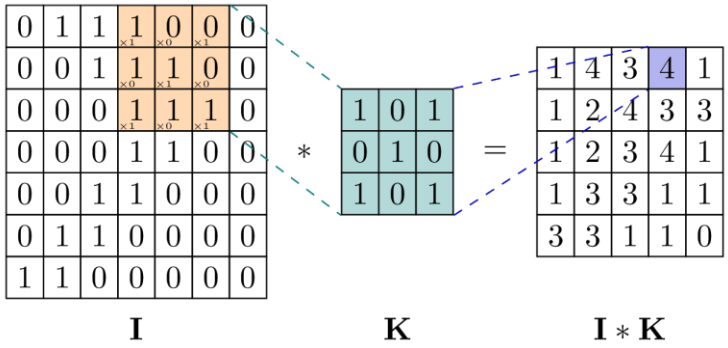
- **Aplicación de IA en Física y CFD**: Exploración de herramientas de inteligencia artificial (IA) y aprendizaje computacional en Dinámica de Fluidos Computacional (CFD) y física.
- **Capacidad Predictiva de IA**: Los sistemas de IA pueden aprender y predecir comportamientos complejos mediante la optimización de sus parámetros.
- **Redes Neuronales**: Fundamento de los modelos de aprendizaje profundo, con cada capa aplicando una función que transforma un vector de entrada en un vector de salida. Aprendizaje Profundo con la concatenación de múltiples capas.



- **Optimización de Modelos Neuronales**: Utilización del descenso de gradiente para ajustar iterativamente los pesos y sesgos de la red, minimizando una función de coste.
- **Desafíos en el Entrenamiento**: Ajuste del ratio de aprendizaje, manejo de la explosión y desaparición de gradientes, y prevención de sobreajuste.

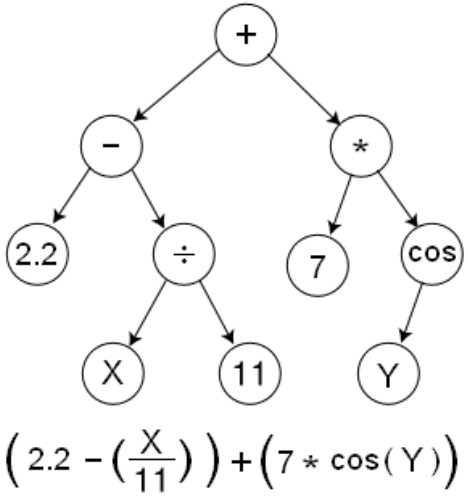


Introducción: Redes neuronales convolucionales



- **Fundamento de las CNNs**: Uso de filtros para procesar datos multidimensionales.
- **Mecanismo de Filtros**: Aplicación de filtros a través de operaciones de producto interno.
- **Detección de Patrones**: Capacidad de capturar patrones invariantes en distintas posiciones.
- **Padding y Stride**: Técnicas para configurar dimensiones y regular el desplazamiento de filtros.
- **Convolución Traspuesta**: Utilizada para el upsampling y expansión de datos.
- **Optimización de Filtros**: Ajuste mediante descenso de gradiente para extraer información relevante.
- **Aplicaciones de CNNs**: Reconocimiento de imágenes, detección de objetos, generación de imágenes.

Introducción: Representación simbólica



- **Enfoque Alternativo**: Representación simbólica como alternativa al aprendizaje profundo.
- **Estructura de Árbol**: Uso de árboles para representar expresiones matemáticas.
- **Optimización con Algoritmos Genéticos**: Selección iterativa de árboles basada en el coste.
- **Interpretabilidad**: Facilidad para traducir el resultado en ecuaciones matemáticas comprensibles.
- **Contraste con Aprendizaje Profundo**: Mayor interpretabilidad frente a mayor coste computacional en el entrenamiento.
- **Aplicación en Física**: Valor en campos donde la interpretación de modelos es crucial.

Metodología

- **Generación de Datos**: Uso de datos de simulaciones de CFD en alta y baja resolución.
- **Transformación Clave**: Conversión de timesteps de baja resolución (X) en aproximaciones de alta resolución (Y').
- **Entrenamiento de Modelos**: Uso de estrategias supervisadas para optimizar la solución.
- **Estrategias de Optimización**:
  - Minimizar la diferencia entre ground-truth y la salida del modelo.
  - Minimizar la diferencia entre groun-truth y la suma de la salida del modelo y una interpolación bicúbica.
  - Minimizar la diferencia entre residuos de las ecuaciones físicas. Alineación con condiciones físicas ('physics-driven').
- **Evaluación y Análisis Comparativo**: Rendimiento de modelos, viabilidad y comparación con simulaciones tradicionales.
- **Estudio de Complejidad Computacional**: Comparación de complejidad computacional entre modelos.

Planificación

Nombre	Duración	Inicio	Fin
Fase 1	20días?	27/09/2023	24/10/2023
Definición y planificación	20días?	27/09/2023	24/10/2023
Fase 2	27días?	24/10/2023	29/11/2023
Investigación del State-Of-The-Art	14días?	24/10/2023	10/11/2023
Obtención de datos	14días?	10/11/2023	29/11/2023
Fase 3	18días?	30/11/2023	25/12/2023
Implementación de modelos y entrenamiento	13días?	30/11/2023	18/12/2023
Análisis de resultados	6días?	18/12/2023	25/12/2023
Fase 4	16días?	26/12/2023	16/01/2024
Elaboración de memoria	16días?	26/12/2023	16/01/2024
Fase 5	11días?	17/01/2024	31/01/2024
Preparación de defensa	10días?	17/01/2024	30/01/2024
Defensa	1día?	31/01/2024	31/01/2024

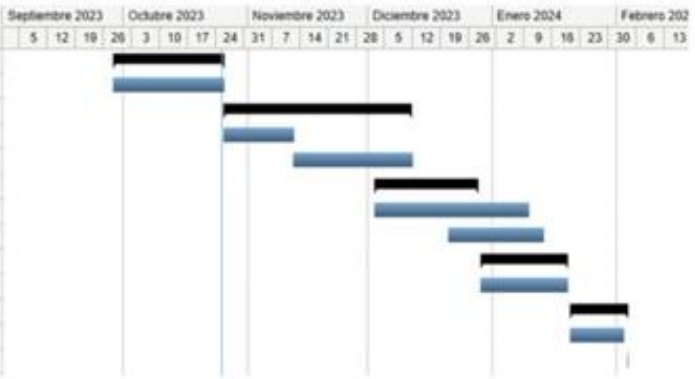


Figura 6: Diagrama de Gantt del desarrollo del trabajo

- **Etapas de Planificación**:
  - Definición y planificación del trabajo.
  - Investigación del estado del arte.
  - Obtención de datos mediante simulaciones.
  - Implementación y entrenamiento de modelos.
  - Análisis de resultados y evaluación.
  - Elaboración de memoria y preparación de defensa.
- **Productos Obtenidos**:
  - Conjunto de datos de simulaciones de dinámica de fluidos. (Disponibles en Kraggle)
  - Algoritmos de entrenamiento en PyTorch.
  - Algoritmo para árboles de representación simbólica.
  - Modelos entrenados para upsampling.
  - Estudio comparativo de rendimiento y coste.
- **Disponibilidad de Recursos**: Acceso a todos los recursos en el repositorio GitHub del proyecto.

Definición del Dominio del Problema en Simulaciones de CFD

- **Ecuación de Navier-Stokes**: Fundamento de las simulaciones de CFD para fluidos incompresibles en 2D.
- **Geometría de Simulación**: Área constante de 255x255 celdas representando un cuadrado de lado 1.
- **Condiciones de Simulación**: 320 iteraciones representando 10 segundos, con densidad fija y viscosidad ajustada para reflejar un flujo turbulento (Re $\approx 1x10^4$)
- **Condiciones inciales aleatorias**: Geometría constante con condiciones iniciales variables.

$$\nabla \vec{v} = 0$$

$$\frac{\partial \vec{v}}{\partial t} + (\nabla \vec{v})\vec{v} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{v}$$

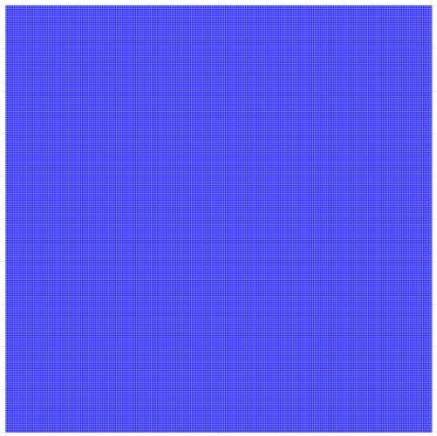


Figura 7: Visualización de malla utilizada

Framework

- **OpenFOAM**: Software clave para simulaciones de CFD, integrado con Matlab y FEATool Multiphysics.
- **Python como Lenguaje de Programación**: Elección por su simplicidad, legibilidad y recursos en IA.
- **Librerías Relevantes**: TensorFlow, PyTorch, Matplotlib, Pandas, Numpy.
- **Uso de PyTorch**: Por su claridad y flexibilidad, facilitando la implementación y el debugging de modelos.
- **Hardware**: Procesador AMD Ryzen 5 5600X, 32GB RAM, Nvidia GeForce 3080.

Datos

- **Condiciones de Contorno**: No-Slip Wall

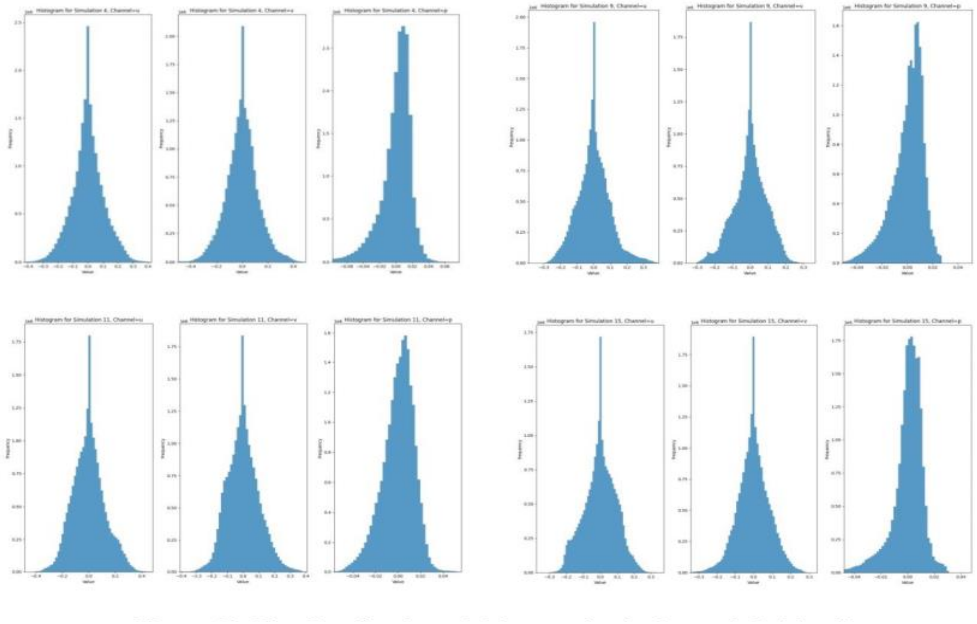
$$u(x = 0, x = 1) = 0$$
$$v(y = 0, y = 1) = 0$$

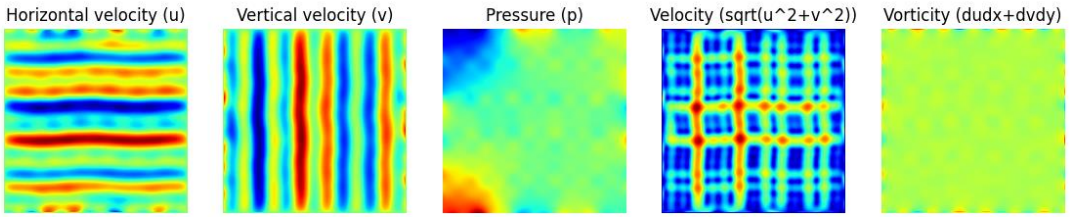
- **Condiciones Iniciales Pseudoaleatorias**: Generación de un comportamiento fluido caótico y turbulento.

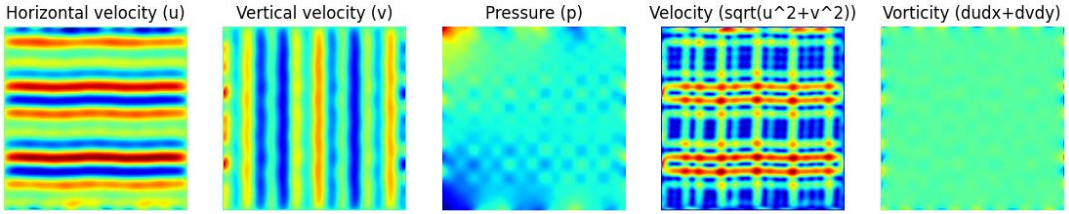
$$\begin{cases} u_0 0(x,y) = rand() * sin(rand() * 8 * (2 * \pi) * y + rand() * 2 * \pi) \\ v_0 0(x,y) = rand() * sin(rand() * 8 * (2 * \pi) * x + rand() * 2 * \pi) \\ p_0 = 0 \end{cases}$$

- **Volumen de Simulaciones**: 35 simulaciones, totalizando aproximadamente 14 GB de datos.
- **Preparación de Datos**: Escalado por desviación típica para precisión en el cálculo de residuos.
- **División de Datos**: 18 simulaciones para entrenamiento, 9 para evaluación, 8 para pruebas.

Datos obtenidos







Aprendizaje profundo: estrategias de optimización

- **Estrategia de Predicción Directa**: Optimización para producir la matriz de tres canales directamente de la entrada de baja resolución.
- **Estrategia de Residuos de Interpolación**: Cálculo de la diferencia entre la interpolación bicúbica y la simulación de alta resolución.
- **Ventajas de la Estrategia de Residuos**: Outputs centrados en cero para facilitar el entrenamiento.
- **Minimización de Residuos Físicos**: Implementación de función de coste específica para residuos de ecuaciones de Navier-Stokes.
- **Objetivo Doble**: Minimizar la desviación y promover coherencia con leyes físicas.
- **Función de Coste Específica**: Incluye cálculo de derivadas respecto al tiempo, que podría capturar información relevante.

APRENDIZAJE PROFUNDO: MODELOS

- **Se acude a la bibliografía del proceso de aumento de resolución de imágenes y videos para buscar una base de modelos con los que realizar el estudio comparativo.**

- **Modelo de referencia 'NaiveTransConv'**:
  - Modelo sencillo para entender la convolución transpuesta.
  - Dos capas de convolución transpuesta en 2D. Cada capa con kernel de 2x2 y stride de 2. 512 canales en la capa profunda.
- **Funcionamiento del Modelo**:
  - Factor de superresolución de 2x por capa, totalizando 4x.
  - Operación sin funciones de activación para mantener la linealidad.
- **Propósito y Aplicación**:
  - Demostrar el funcionamiento básico de superresolución en aprendizaje profundo.
  - Servir de referencia como modelo sencillo y lineal en la comparativa.

Aprendizaje profundo: modelos

- **SRCNN - Super-Resolution Convolutional Neural Network**
- **FSRCNN - Fast Super-Resolution Convolutional Neural Network**
- **VDSR - Very Deep Super-Resolution**
- **ESPCN - Efficient Sub-Pixel Convolutional Neural Network**

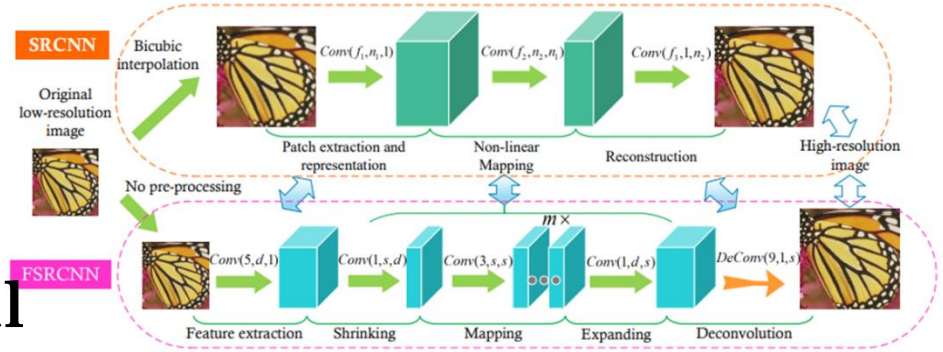


Figura 12:Arquitectura de modelo FSRCNN[33]

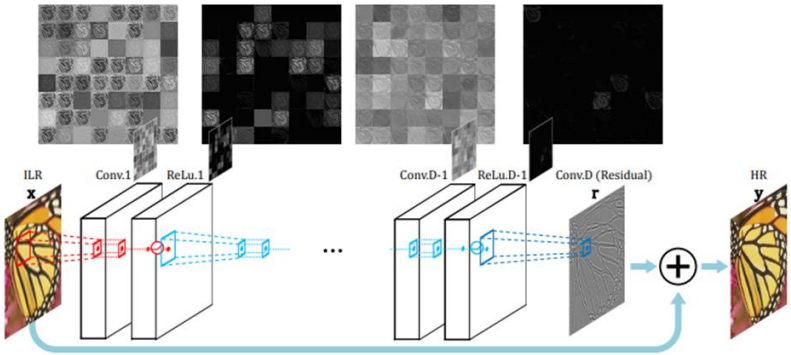


Figura 13: Arquitectura de modelo VDSR[33]

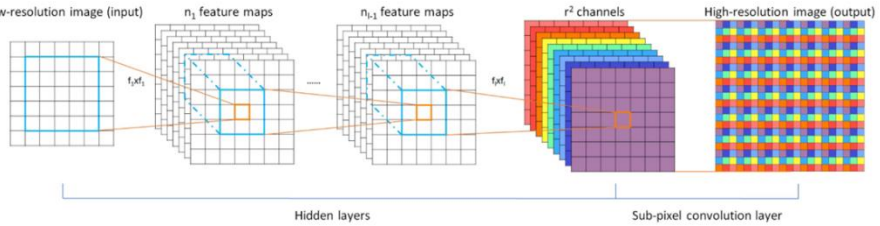


Figura 14: Arquitectura de modelo ESPCN[36]

Aprendizaje profundo: modelos propuestos

- **Modelos propuestos Encoder-Decoder**:
- **Fase de Codificación**:
  - Aumento progresivo de canales a 128 y 256.
  - Uso de kernel de tamaño 3 y padding de 1.
- **Fase de Decodificación**:
  - Transformación de datos densos en imagen de alta resolución.
  - Uso de capas de deconvolución y convolución para 4x upsampling o, en el modelo alternativo, Sub-Pixel deconvolution/Pixel-Shuffle (modelo ESPCN).
- **Variantes Implementadas**:
  - ED_TransConv: Uso de deconvolución.
  - ED_PixelShuffle: Uso de Pixel Shuffle.

Aprendizaje Profundo: funciones de coste

- **Error Medio Cuadrado (MSE)**:
  - Mide la diferencia cuadrática media entre predicciones y valores reales.

$$MSE(Y, Y') = \frac{1}{N}\sum_{i=1}^{N}(Y_i - Y'_i)^2$$

- **Función de Residuos de Navier-Stokes**:

$$Eq\ de\ continuidad:\ R_0 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

$$Eq\ de\ momento:\ \begin{cases} R_1 = \rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) + \frac{\partial p}{\partial x} - \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ R_2 = \rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) + \frac{\partial p}{\partial y} - \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \end{cases}$$

  - Calcula los residuos de la ecuación de Navier-Stokes.
  - Usa diferencias finitas para aproximaciones de derivadas.
  - Calcula el error medio de la diferencia los residuos en cada punto.
  - Contiene información potencialmente relevante sobre la derivada temporal y busca que el modelo se optimice de forma coherente con las leyes físicas.

Aprendizaje Profundo: entrenamiento

1. **Configuración y Librerías**:
   - Hiperparámetros: Número de épocas, factor de aumento, tamaño del mini-batch.
2. **Carga de Datos**:
   - Datos en formato tensorial, adecuados para simulaciones y timesteps.
3. **Modelo y Optimizador**:
   - Instanciación del modelo.
   - Configuración de optimizador y funciones de coste.
4. **Funciones de Entrenamiento/Evaluación**:
   - Manejo de interpolación y optimización.
   - Downsample de datos de alta resolución para emular datos de baja resolución.
   - Cálculo de pérdidas y ajustes de parámetros.
5. **Bucle de Entrenamiento**:
   - Iteración por epochs.
   - Guardado del mejor modelo en datos de evaluación. Se busca el modelo con mejor capacidad de generalización que luego se evaluará con los datos *test*.

Aprendizaje Profundo: resultados

- **Estrategias de Optimización Comparadas**:
  - Se evalúa el proceso de entrenamiento tras 100 epochs.
  - Inferencia directa vs. predicción de diferencias de interpolación.
  - Resultados más consistentes con la predicción de diferencias.
- **Comparación entre Modelos**:
  - Rápida convergencia con estrategia basada en residuos.
  - No hay diferencias al utilizar la función de coste de Navier Stokes.
- **Ausencia de Sobreajuste**:
  - Mejora en datos de entrenamiento reflejada en datos de evaluación.
  - Estabilidad en el rendimiento de los modelos mostrando convergencia.

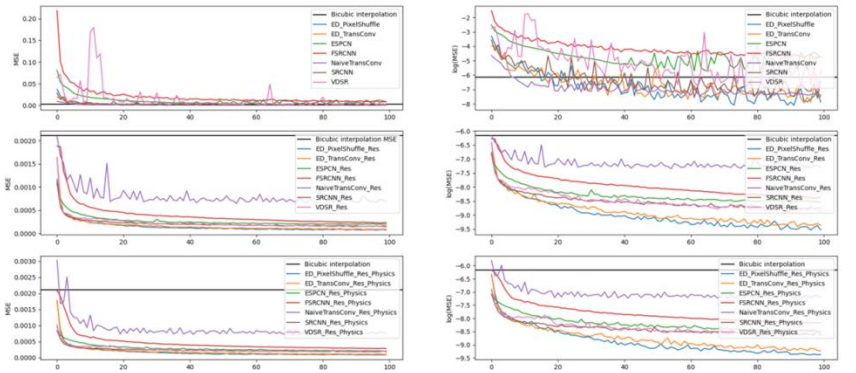


Figura 15: Comparativa de los entrenamientos de cada modelo agrupados por estrategia de optimización.

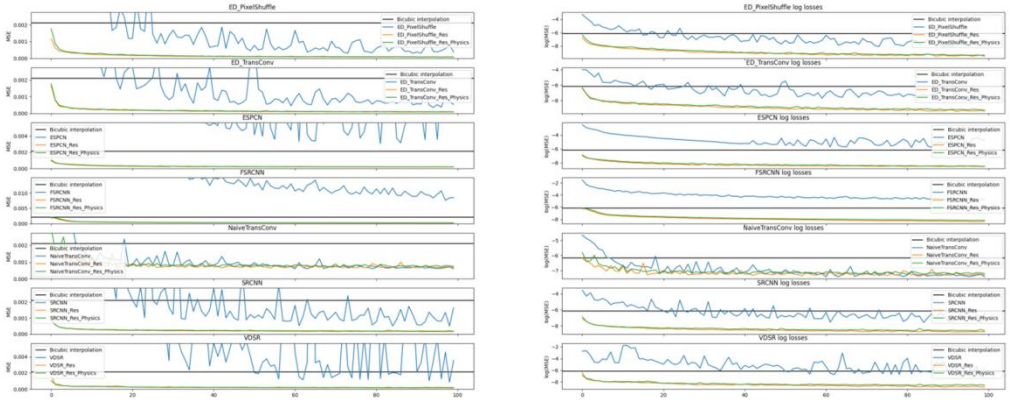


Figura 16: Comparativa de evolución de los entrenamientos agrupados por arquitecturas.

Aprendizaje Profundo: resultados

- **Entrenamiento Extendido y Rendimiento**: Se siguen entrenando los modelos hasta convergencia.
- **Resultados de Rendimiento**: Error medio cuadrado, máximo y residuos físicos para cada modelo y estrategia.
- **Comparación de resultados**: Mejores resultados cuando los modelos tienen un mayor número de parámetros y se optimiza el residuo de la interpolación bicúbica.
- **Modelos propuestos**: ED_TransConv y ED_PixelShuffle con los menores errores y residuos.
- **Impacto Marginal de Residuos Físicos**: Sin diferencia significativa en el rendimiento al minimizar residuos de ecuaciones físicas.

Modelo	MSE	MaxMSE	PhysicsLoss
Bicubic	2.42E-03	2.99E-01	3.15E-03
ED_PixelShuffle	3.97E-04	7.26E-03	2.68e-04
ED_PixelShuffle _Res	**7.79-05**	4.93E-03	1.24e-04
ED_PixelShuffle _Res_Physics	8.09E-04	**4.67E-03**	**1.19E-04**
ED_TransConv	5.39E-04	7.50E-03	3.08E-04
ED_ TransConv _Res	8.48 E-04	4.94E-03	1.30E-04
ED_ TransConv _Res_Physics	1.10E-04	5.56E-03	1.57E-04
ESPCN	7.05E-03	1.45E-02	1.31E-03
ESPCN_Res	2.45E-04	7.68E-03	3.52E-04
ESPCN_Res_Physics	2.58E-04	8.00E-03	3.59E-04
FSRCNN	6.31E-03	1.97E-02	1.79E-03
FSRCNN_Res	2.12E-04	6.76E-03	2.80E-04
FSRCNN_Res_Physics	2.82E-04	7.52E-03	3.10E-04
NaiveTransConv	6.80E-04	1.01E-02	6.63E-04
NaiveTransConv_Res	7.54E-04	1.13E-02	9.44E-04
NaiveTransConv_Res_Physics	8.32E-04	1.12E-02	7.37E-04
SRCNN	1.63E-03	8.80E-03	4.36E-04
SRCNN_Res	1.89E-04	6.87E-03	2.79E-04
SRCNN_Res_Physics	2.20E-04	7.40E-03	2.90E-04
VDSR	3.59E-03	1.12E-02	5.32E-04
VDSR_Res	1.66E-04	6.57E-03	2.30E-04
VDSR_Res_Physics	2.31E-04	7.13E-03	2.81E-04

Tabla 1: Comparación de rendimiento de los modelos tras entrenamiento sobre set de datos *test*.

Aprendizaje Profundo: resultados

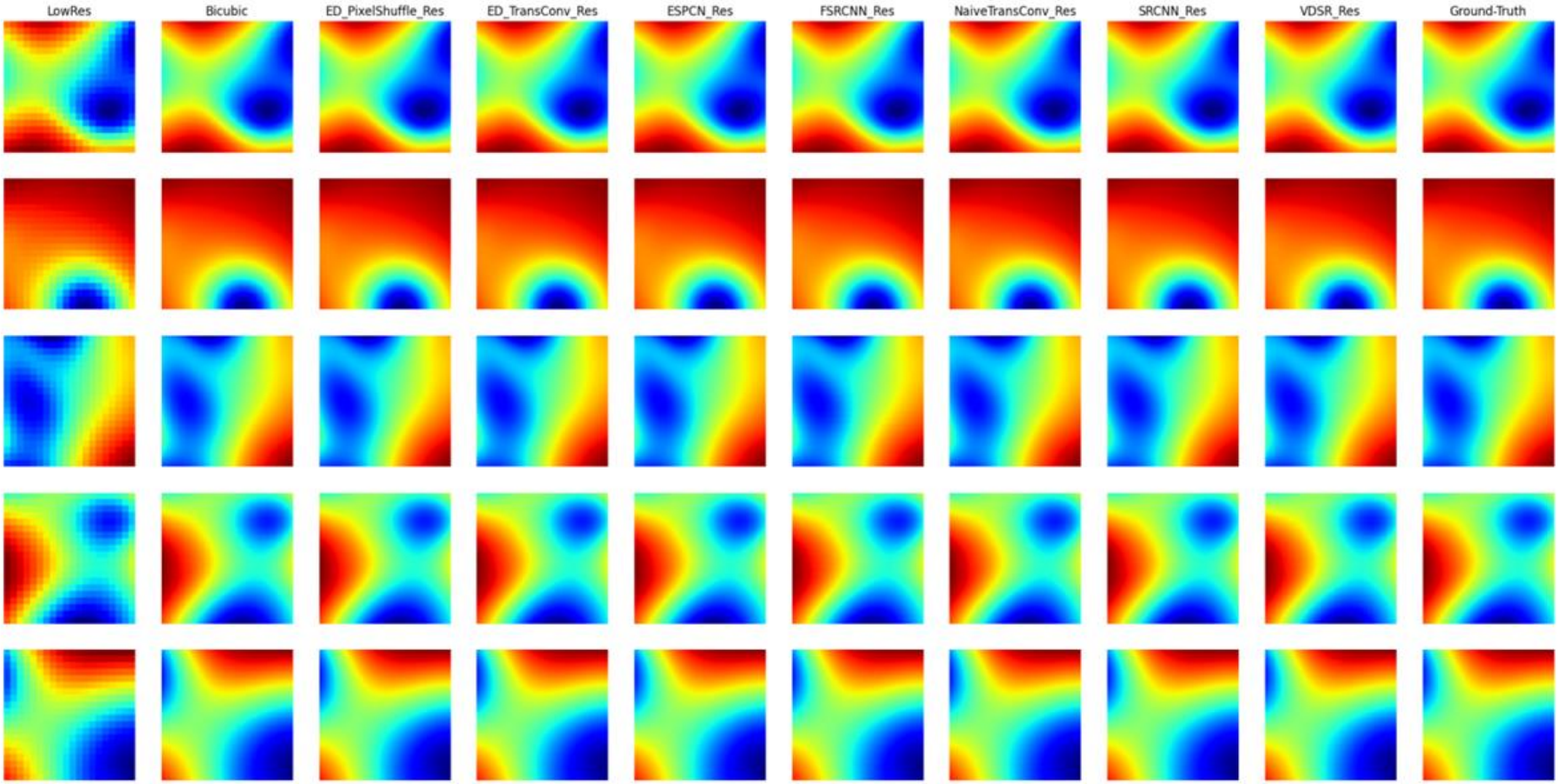


Figura 19: Comparativa visual de reconstrucción de representación de alta resolución para los modelos que se optimizan para predecir la diferencia (residuos) entre interpolación bicúbica y valores de alta resolución.

Aprendizaje Profundo: resultados

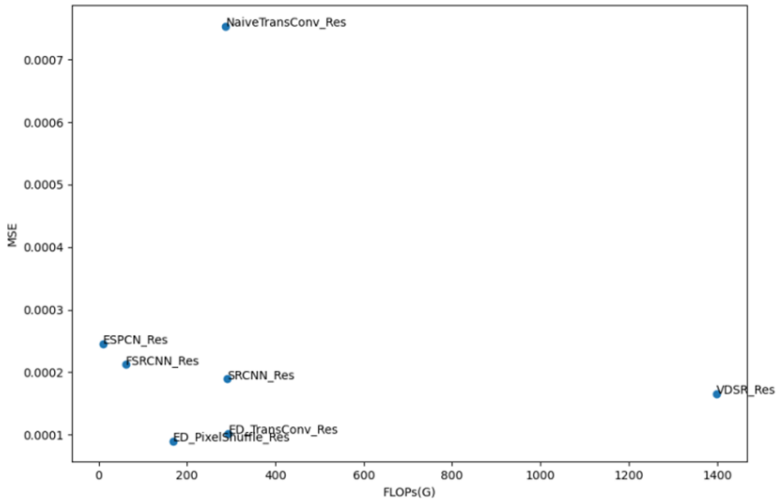


Figura 20: Error cuadrado medio con respecto al número de operaciones.

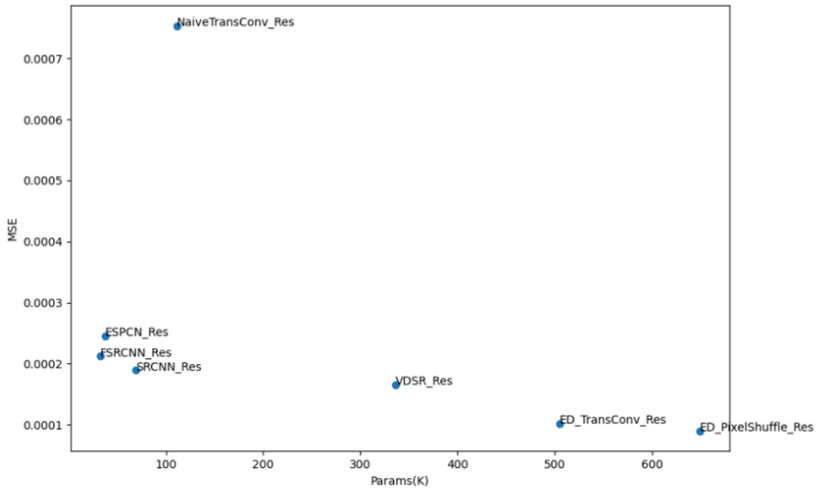


Figura 21: Error cuadrado medio con respecto al número de parámetros.

- **Error Cuadrado Medio vs. Número de Operaciones**: Mejor rendimiento con optimización de residuos.
- **Número de Parámetros vs. Rendimiento**: Relación entre más parámetros y mejor rendimiento.
- **Tiempos de Ejecución Práctica**: Significativamente menores en CPU y GPU comparados con simulaciones de alta resolución.
- **Conclusiones de Eficiencia**: Los modelos de aprendizaje profundo son eficientes y rápidos, ofreciendo una alternativa viable a simulaciones más complejas.

	NaiveTransConv	ED_PixelShuffle	ED_TransConv	ESPCN	FSRCNN	SRCNN	VDSR
Tiempo (s)	41.5	6.2	10.6	1.6	2.7	13.9	64.6

Tabla 4: Muestra comparativa de tiempos de ejecución en CPU.

Aprendizaje Profundo: viabilidad

- **Capacidad Temporal Ausente**: Explorar modelos con integración temporal, como RNN (LSTM, GRU) o convolución en 3D.
- **Validación de Datos de Baja Resolución**: Metodología está supeditada a captura de patrones relevantes por simulación al haber simulado el mapeo de baja resolución.
- **Enfoque Limitado a la Dimensión Espacial**: Oportunidad de investigación en la interpolación temporal.
- **Limitación en los modelos comparados:**

  Se acude a la literatura de upsampling de imágenes y se toma este marco de referencia, pero existen modelos más sofisticados.

Representación simbólica: modelo

```python
class Node:
    def __init__(self, value, left=None, right=None):
        self.value = value  # Can be an operation or an operand
        self.childs = 0
        self.left = left
        self.right = right
```

```python
operations = {
    '+': operator.add,
    '-': operator.sub,
    '*': operator.mul,
    '/': operator.truediv,
    '^2': power2,
    '^3': power3,
    'sin': np.sin,
    'exp': np.exp,
    'log': np.log,
}
```

- **Estructura del Árbol**: Cada nodo tiene hasta dos hijos y representa una operación o un valor.
- **Operaciones Definidas**: Incluye operaciones básicas como suma, resta, y funciones como seno, exponencial, y logaritmo.
- **Variables de Entrada**: Variables que incluyen posiciones en la malla y valores de velocidad y presión.
- **Objetivo**: Encontrar relaciones matemáticas entre variables de alta resolución y sus equivalentes de baja resolución.

```python
variables = ['x','y',
             'U00','U10','U01','U11',
             'V00','V10','V01','V11',
             'P00','P10','P01','P11',
             'x0','y0','x1','y1']
```

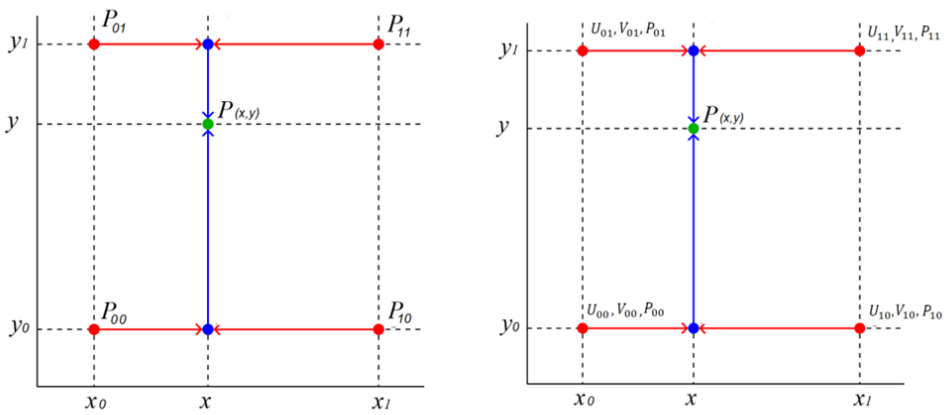


Figura 22: Representación de variables utilizadas en interpolación bilineal (a la izquierda) e interpolación buscada que hace uso de todas las variables (derecha).

Representación simbólica: entrenamiento

- **Proceso de Entrenamiento:**
- **Creación de Población Inicial**: Árboles con estructuras aleatorias.
- **Evaluación y Selección**: Se eligen árboles con mejor ajuste (menor error medio cuadrado).
- **Operaciones Genéticas**: Incluyen mutación de nodos y cruce entre árboles.
- **Iteraciones Generacionales**: Repetición del proceso de selección y mutación a lo largo de varias generaciones.

Representación simbólica: resultados

- **Limitaciones del Modelo**: El modelo solo ha identificado soluciones triviales como la interpolación por vecinos cercanos y bilineal, sin integrar otras variables físicas relevantes.
- **Alto Costo Computacional**: El entrenamiento del modelo ha resultado ser extremadamente costoso en términos de tiempo y recursos computacionales.
- **Desafío en la Diversidad de Soluciones**: Dada la gran cantidad de variables de entrada y la naturaleza aleatoria del algoritmo genético, ha sido difícil obtener ecuaciones válidas al ser el espacio de búsqueda muy amplio.

Conclusiones

- **Aplicación de Aprendizaje Computacional**: Demostración de cómo mejorar la eficiencia en simulaciones de dinámica de fluidos.
- **Alternativa a Métodos Numéricos**: Exploración de modelos de Aprendizaje Profundo como opciones viables cuando los recursos son limitados.
- **Necesidad de Evaluación Específica**: Importancia de adaptar la metodología a cada contexto físico específico.
- **Profundización en Optimización Basada en Navier Stokes**: Identificación de la necesidad de más investigación en esta área.
- **Uso de Sub-Pixel Deconvolution**: Mejora en los resultados utilizando técnicas avanzadas como Pixel-Shuffle.
- **Limitaciones de la Representación Simbólica**: Dificultades encontradas al tratar con grandes cantidades de datos y variables.
- **Futuras Direcciones de Investigación**: Potencial para integrar la dimensión temporal y técnicas más avanzadas en los modelos.
- **Conclusiones Generales**: Potencial de Aprendizaje Profundo para modelización computacional más rápida y económica, con implicaciones en múltiples campos científicos y de ingeniería.

Muchas gracias