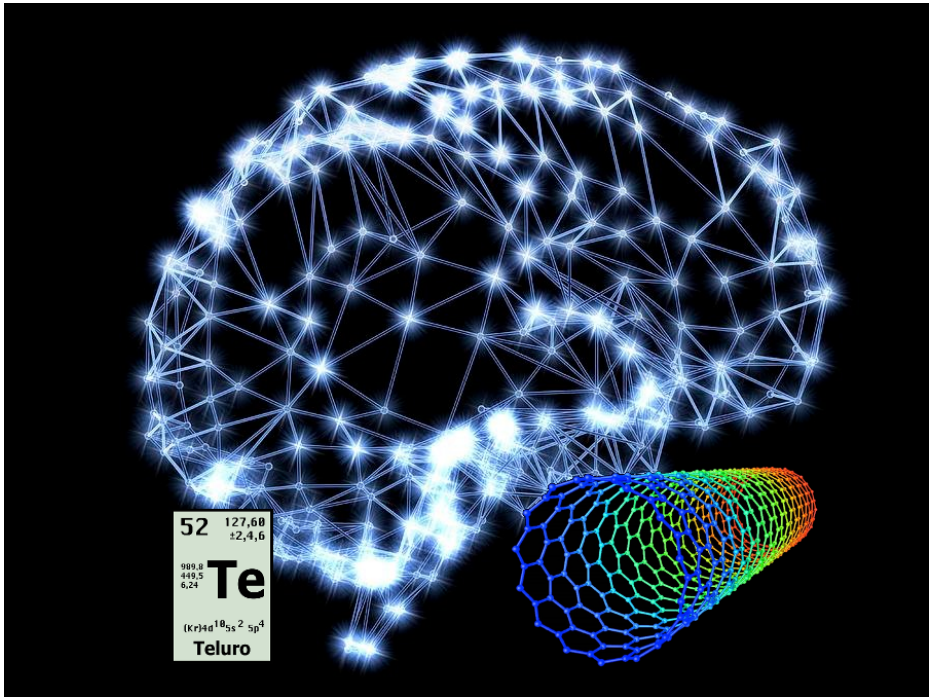


Antonio Díaz Pozuelo

adpozuelo@uoc.edu

TFG – Grado en Ingeniería Informática

Universitat Oberta de Catalunya

Enero de 2016

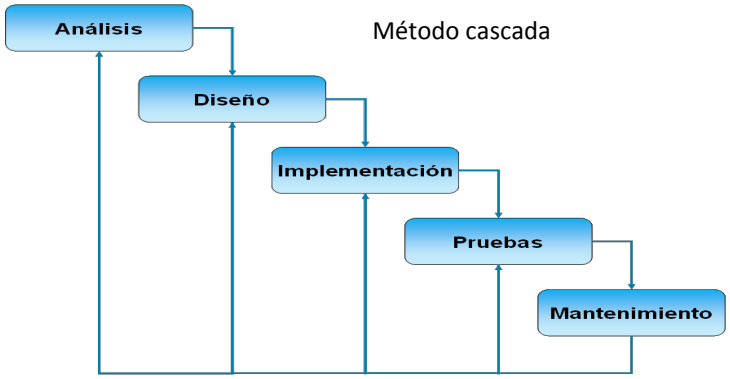
High-Dimensional Neural Network Potentials

INTRODUCCIÓN

- El objetivo del presente TFG (Trabajo Final de Grado) consiste en diseñar e implementar una red neuronal pre-alimentada FFNN (*Feed-Forward Neural Network*) sobre una GPU (*Graphical Processor Unit*) del fabricante NVIDIA con arquitectura CUDA (*Compute Unified Device Architecture*). Dicha red neuronal debe ser capaz de aprender a partir de las propiedades de determinados conjuntos de sistemas de átomos (posiciones en una secuencia temporal y energía para una temperatura y densidad dadas) y predecir las propiedades macroscópicas de otro sistema de átomos diferente, en condiciones de temperatura y/o densidad, a los utilizados para el aprendizaje.

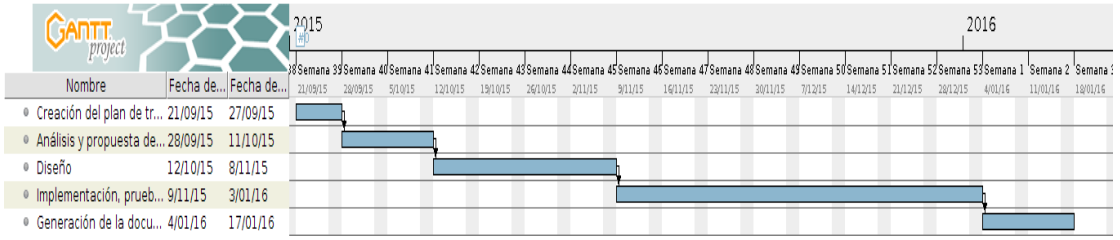
PLANIFICACIÓN

- Para realizar este proyecto se utilizará un enfoque metodológico en cascada, ya que cada etapa del proyecto nos dará como resultado un artefacto que se utilizará en la siguiente fase. Por lo tanto, los artefactos de cada etapa nos marcarán un desarrollo lineal e iterativo, con objeto de corregir y ajustar las fases según se desarrolle el proyecto.
- Podemos dividir el desarrollo del proyecto en varias etapas:
  - Análisis y propuesta de soluciones.
  - Diseño.
  - Implementación.
  - Pruebas y simulaciones.
  - Mantenimiento.



Método cascada

Diagrama de Gantt



ANÁLISIS Y PROPUESTA DE SOLUCIONES

- Exposición del problema:
  - Dado un sistema de átomos (configuración) se quieren calcular las propiedades macroscópicas (energía, presión, conductividad, etc.) del mismo.
- Posibles soluciones:
  - Utilizar interacciones efectivas entre pares/tripletes de partículas ajustadas a cálculos mecano-cuánticos y/o propiedades experimentales. Éste es un procedimiento habitual cuando no se requiere gran precisión en sistemas relativamente simples.
  - Resolver la ecuación de Schrödinger para un conjunto de átomos dado - empleando la teoría del funcional de la densidad (DFT) - y obtener las energías y las fuerzas correspondientes. Estos métodos, conocidos como *ab initio*, son computacionalmente muy costosos y sólo pueden aplicarse a unos cientos de átomos.
  - Utilizar redes neuronales NN (*Neural Networks*) dada su habilidad para representar funciones arbitrarias. Estas redes se consideran "aproximadores universales" y aprenden a partir de datos *ab initio* para conjuntos pequeños de átomos permitiendo un estudio detallado en condiciones donde el cálculo requiere muestras (número de átomos/moléculas) grandes.

ANÁLISIS Y PROPUESTA DE SOLUCIONES

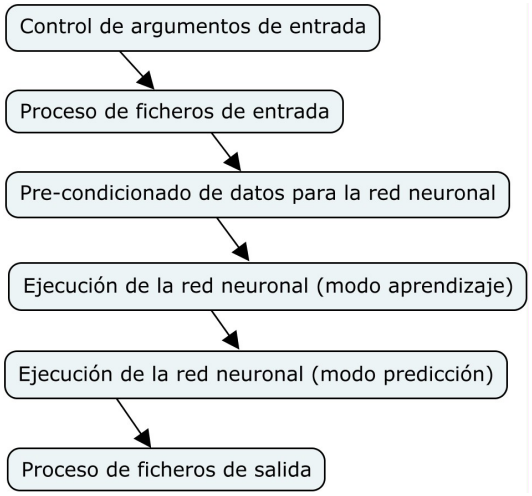
- Solución elegida:
  - Para resolver el problema dado se utilizará la solución ofrecida por las redes neuronales debido a que:
    - Se desea aprender sobre el diseño y desarrollo de redes neuronales.
    - Es el campo menos desarrollado de los tres citados anteriormente.
    - Existe la necesidad en el Instituto de Química Física Rocasolano, del Consejo Superior de Investigaciones Científicas, de disponer de esta metodología. Por lo tanto, esta solución podrá tener un carácter práctico en el futuro.
- Se tendrán las siguientes consideraciones:
  - La red neuronal será pre-alimentada FFNN.
  - La red neuronal será multi-capa.
  - La red neuronal se construirá para sistemas de altas dimensiones.

Arquitectura computacional y tecnología

- La construcción de la solución elegida se puede realizar de forma secuencial en su totalidad o paralelizando las partes que lo permitan. Se optará por paralelizar las partes que admitan este paradigma con objeto de obtener el mejor rendimiento posible.
- Se puede optar por paralelizar utilizando la CPU o la GPU.
- Se optará por la paralelización utilizando la GPU dado que:
  - Se desea aprender el desarrollo de aplicaciones utilizando esta tecnología.
  - La paralelización en GPU es una tecnología en continua expansión en el ámbito científico.
  - La eficiencia de las GPUs es superior respecto de las CPUs.
- Se decide utilizar GPUs de la marca NVIDIA con arquitectura CUDA dado que:
  - Se desea aprender a implementar aplicaciones en CUDA_C/C++.
  - Se trata de GPUs de ámbito general que se pueden encontrar en gran parte del ecosistema informático existente.
  - Disponen de un SDK (*Software Development Kit*) maduro y basado en el lenguaje de programación C.

DISEÑO

- Se realiza de forma modular por lo que cada parte de la solución se pueda diseñar e implementar en base a descomposiciones (especificaciones) de otras partes más genéricas.
- Dada la alta cantidad de información a procesar y almacenar, cada módulo deberá ser lo máximo posible independiente de los otros con objeto de optimizar los recursos (memoria RAM). Esto permitirá realizar ejecuciones independientes de cada módulo obteniendo resultados de las pruebas que retro-alimentarán el ciclo diseño → implementación → pruebas.
- Algoritmo general:



Módulo 1 → Control de argumentos de entrada

- El objetivo de este módulo es controlar los argumentos de entrada de la aplicación y generar los nombres de los ficheros de entrada a procesar.
- Datos de entrada:
  - Las temperaturas (en grados Kelvin) de los sistemas de átomos con las que la red neuronal debe aprender y a predecir.
- Se deben generar, por cada temperatura de entrada dos nombres de ficheros: uno para el fichero (*history*) que contiene la secuencia temporal de conjuntos de átomos y otro (*energy*) que contiene la energía, ya calculada utilizando simulaciones mecano-cuánticas, de cada sistema de átomos.

Módulo 2 → Proceso de ficheros de entrada

```
timestep         1        64         2        1    0.001000
   14.508000      0.000000      0.000000
    0.000000     14.508000      0.000000
    0.000000      0.000000     14.508000
      Te         1  127.600000    0.000000
    9.2039780617E+00      9.7824159389E+00      2.8602546664E+00
    6.9020853329E-01      5.7969950169E-01     -5.0218420389E+00
    2.1141000000E-01      7.3537400000E-01     -2.0765000000E-01
      Te         2  127.600000    0.000000
    4.9346167759E+00      2.1517051280E+00      4.2864128284E+00
   -6.4354145474E-01      3.1454012737E+00     -7.9618288613E-01
    1.6940500000E-01      6.5414700000E-01      2.2044600000E-01
      Te         3  127.600000    0.000000
    1.0175378176E+01      4.5515028593E+00      1.3390518543E+01
    5.0135202338E-01     -5.8780197567E-01     -1.1360254390E+00
   -2.0113400000E-01      5.2877400000E-01      5.7874100000E-01
      Te         4  127.600000    0.000000
    9.8613298836E-01      9.5423831100E+00      1.4415460722E+00
    1.0302747776E+00     -3.7611708550E+00      1.1845292580E+00
    6.0212800000E-01     -1.2605300000E-01     -2.3624700000E-01
      Te         5  127.600000    0.000000
    1.0231880453E+01      7.7367275856E-01      6.2895434144E+00
   -2.0277714305E+00     -1.4420043594E+00     -2.8902172356E+00
    5.0040300000E-01      2.7809400000E-01      1.2903300000E-01
      Te         6  127.600000    0.000000
    1.0756549941E+01      2.8265929146E+00      1.0011055345E+01
   -1.5319025498E+00     -1.4049868508E+00      2.0238728085E+00
   -3.4564200000E-01      1.6534200000E-01      4.2873300000E-01
      Te         7  127.600000    0.000000
    8.9678764656E+00      6.6238054132E+00      4.7118632652E-01
   -3.4923324006E+00      1.7661316123E-01      1.5661413212E-01
    2.8351200000E-01     -2.8377000000E-01     -3.7302300000E-01
```

HISTORY.Te.673K.

```
  0.000000    -185.488060
  0.001000    -185.470680
  0.002000    -185.453920
  0.003000    -185.437850
  0.004000    -185.422500
  0.005000    -185.407910
  0.006000    -185.394180
  0.007000    -185.381330
  0.008000    -185.369410
  0.009000    -185.358520
  0.010000    -185.348730
  0.011000    -185.340070
  0.012000    -185.332590
  0.013000    -185.326300
  0.014000    -185.321260
  0.015000    -185.317470
  0.016000    -185.314960
```

E.Te.673K

- El objetivo de este módulo es leer los distintos ficheros de entrada de aprendizaje y de predicción y crear un único fichero de salida compuesto por la energía de cada secuencia temporal (conjunto de átomos) junto a los datos estrictamente necesarios de dicha secuencia.
- El fichero que contiene la secuencia temporal está compuesto por conjuntos de átomos.
- Cada conjunto de átomos se encuentra descrito por su número (*timestep*), por el número de átomos que contiene, por el tamaño (Å) de la caja de simulación (*box*) en la que están los átomos y por la descripción de cada átomo dentro de la caja. A su vez, cada átomo esta descrito por su nombre, un número que le identifica, su peso molecular (u.m.a), sus coordenadas (Å), su velocidad (Å / ps) y su fuerza (u.m.a * Å / ps^2).
- El fichero que contiene la energía de cada conjunto de átomos está compuesto por el tiempo de cada paso que ocupa el conjunto (en ps), dentro de la secuencia temporal, y de la energía (en eV) de dicho conjunto de átomos.

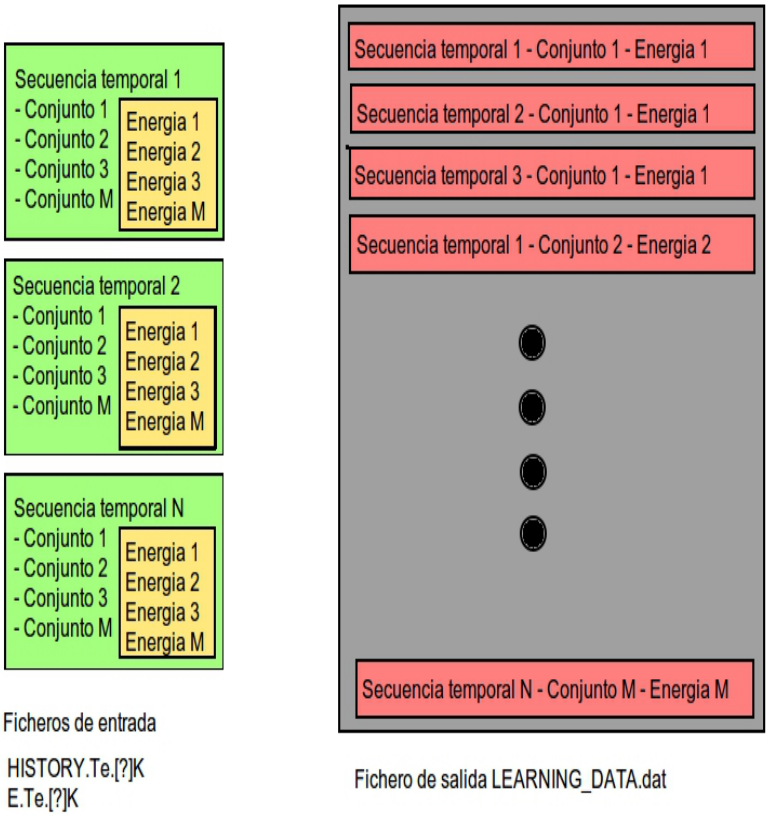
Módulo 2 → Proceso de ficheros de entrada

- Se realiza una selección de características mediante la cual se reducirá la dimensionalidad del conjunto de datos.
- Para el fichero que contiene las secuencias temporales de átomos se seleccionarán las siguientes características:
  - Número de átomos que contiene la caja.
  - Dimensiones de la caja.
  - Coordenadas del átomo.
  - Fuerza ejercida sobre el átomo.
- Para el fichero de energías de las cajas de átomos se seleccionarán las siguientes características:
  - Energía del conjunto de átomos.

```
-185.488068
14.5080003738 14.5080003738 14.5080003738
9.2039785385e+00 9.7824163437e+00 2.8602547646e+00
2.1141000092e-01 7.3537397385e-01 -2.0765000582e-01
4.9346165657e+00 2.1517050266e+00 4.2864127159e+00
1.6940499842e-01 6.5414702892e-01 2.2044600546e-01
1.0175377846e+01 4.5515027046e+00 1.3390518188e+01
-2.0113399625e-01 5.2877402306e-01 5.7874101400e-01
9.8613297939e-01 9.5423831940e+00 1.4415460825e+00
6.0212802887e-01 -1.2605300546e-01 -2.3624700308e-01
1.0231880188e+01 7.7367275953e-01 6.2895436287e+00
5.0040298700e-01 2.7809399366e-01 1.2903299928e-01
1.0756549835e+01 2.8265929222e+00 1.0011054993e+01
-3.4564200044e-01 1.6534200311e-01 4.2873299122e-01
8.9678764343e+00 6.6238055229e+00 4.7118633986e-01
2.8351199627e-01 -2.8376999497e-01 -3.7302300334e-01
```

data/LEARNING_DATA.dat

Módulo 2 → Proceso de ficheros de entrada



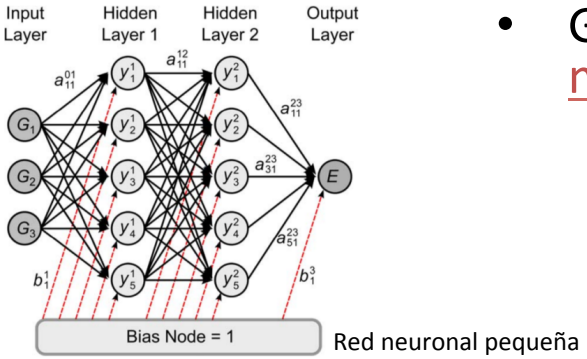
Estructura del fichero de salida del aprendizaje

- La lectura de los conjuntos de átomos de los ficheros de entrada se realiza de forma secuencial por la propia naturaleza de dichos ficheros.
- La escritura de dichos conjuntos tras la selección de características se realiza en el mismo orden secuencial de la lectura.
- Se crea un sesgo temporal relativo a las temperaturas seleccionadas.
- Con objeto de evitar el sesgo en el aprendizaje se debe realizar un intercalado, en la escritura del fichero de salida para el aprendizaje, entre los conjuntos de átomos de las distintas secuencias temporales.

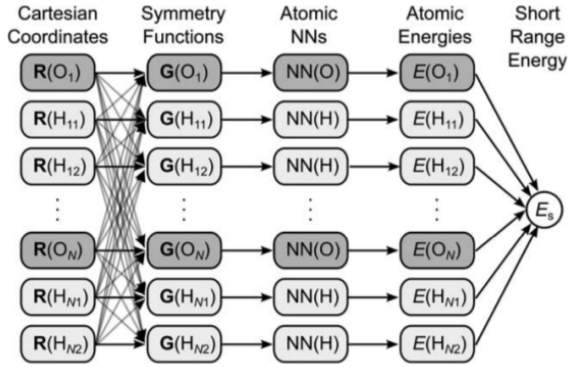
Módulo 3 → Pre-condicionado de datos para la red neuronal

- El objetivo de este módulo es leer las configuraciones de átomos de los ficheros correspondientes, y realizar las siguientes operaciones:
  - Generar los valores de representación de cada átomo mediante funciones de simetría.
  - Seleccionar las características exclusivamente necesarias.
  - Comprobar que los valores de representación de los átomos caracterizan correctamente la configuración.
  - Normalizar los datos de entrada para la red neuronal.
  - Crear los ficheros de salida y gráficas.

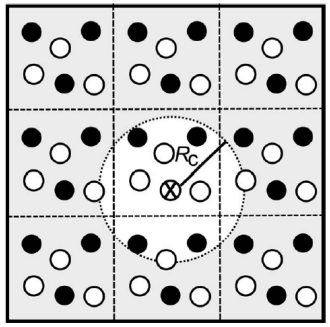
Módulo 3 → Pre-condicionado de datos para la red neuronal



Red neuronal pequeña



Red neuronal alta dimensionalidad



Cutoff del átomo

- Generar los valores de representación de cada átomo mediante funciones de simetría:
  - La necesidad de utilizar funciones de simetría emerge de la condición de alta dimensionalidad de la red neuronal.
  - La red neuronal pequeña el vector de entradas $\{G_i\}$ corresponde a las coordenadas cartesianas de los átomos y sólo existe una red neuronal para cada configuración.
  - La red neuronal de alta dimensionalidad el vector de entradas $\{G_i\}$ corresponde al conjunto de valores de representación, existiendo una red neuronal por átomo.
  - Si en lugar de utilizar los valores de representación se emplearan las coordenadas atómicas como entrada directa a la red neuronal ésta no sería "transferible" a sistemas con diferente número de partículas.
  - La operación a realizar consiste en transformar las coordenadas de un átomo en múltiples valores de representación (obtenidos de evaluar las funciones de simetría) que caracterizarán a dicho átomo dentro de un "entorno limitado esférico" (*cutoff*).
  - Dicho *cutoff* define el entorno energéticamente relevante del átomo y se establece como el radio de la esfera que limita el entorno.

Módulo 3 → Pre-condicionado de datos para la red neuronal

- Se utilizarán las siguientes funciones de simetría:
  - ✓ Función de *cutoff*: decrece cuando se incrementa la distancia (R_{ij}) entre el átomo central i y su vecino j, lo que refleja cualitativamente la reducción de las interacciones entre átomos según se distancian.

$$f_{c,1}(R_{ij}) = \begin{cases} 0.5 \cdot \left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right] & \text{for} \quad R_{ij} \leq R_c \\ 0.0 & \text{for} \quad R_{ij} > R_c \end{cases},$$

Donde:
R_{ij}: distancia entre el átomo central i y el átomo vecino j.
R_c: radio del *cutoff,* se define como 6Å pero se utilizará como valor de referencia R_c^2 = 36Å, con objeto de evitar el cálculo de raíces cuadradas en la determinación de distancias inter-atómicas.

  - ✓ Función radial: define el orden radial de los átomos dentro del *cutoff*.

$$G_i^2 = \sum_{j=1}^{N_{atom}} e^{-\eta\left(R_{ij}-R_s\right)^2} \cdot f_c\left(R_{ij}\right).$$

Donde:
η: parámetro que define el ancho de las gausianas.
R_s: parámetro que define el centro de las gausianas.

  - ✓ Función angular: define el orden angular de los átomos dentro del cutoff.

$$G_i^3 = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} \left[\left(1 + \lambda \cdot \cos\theta_{ijk}\right)^{\zeta} \cdot e^{-\eta\left(R_{ij}^2 + R_{ik}^2 + R_{jk}^2\right)} \cdot f_c\left(R_{ij}\right) \cdot f_c\left(R_{ik}\right) \cdot f_c\left(R_{jk}\right)\right]$$

Donde:
η: parámetro que define el ancho de las gausianas.
ζ: parámetro que define la distribución de los ángulos.
λ: parámetro que cambia la forma de la función del coseno.
θ_{ijk}: ángulo, respecto al átomo central i, que forman los tres átomos i, j y k.

Módulo 3 → Pre-condicionado de datos para la red neuronal

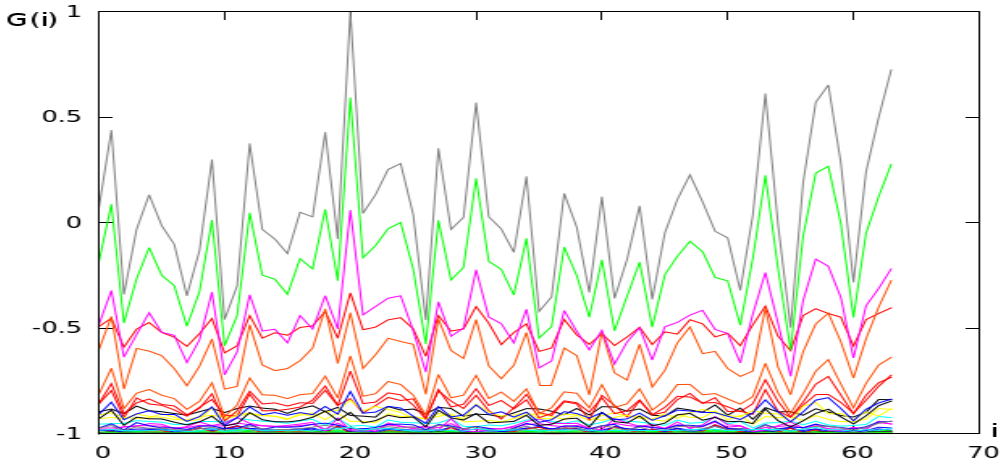
- El cálculo de funciones de simetría consistirá en evaluar cada unas de las funciones anteriores para cada átomo de la configuración y para cada combinación de parámetros posible.
  - $\eta = \{ 0, 0.04, 0.14, 0.32, 0.71, 1.79 \}$
  - $R_s = \{ 0, 1, 2, 3, 4, 5 \}$
  - $\zeta = \{ 1, 2, 4, 16 \}$
  - $\lambda = \{ -1, 1\}$
- Cada átomo quedará representado por el conjunto de valores obtenidos del cálculo anterior, que son los denominados valores de representación.
- Dichos cálculos se realizarán en paralelo utilizando la GPU.
- A los valores de representación de cada átomo se le realiza una selección de características con el objetivo de reducir la dimensionalidad de los datos. Se debe comprobar que:
  - Para el mismo átomo no existen dos valores de representación iguales.
  - Para una misma función de simetría y para todos los átomos el valor de representación es cero.

Módulo 3 → Pre-condicionado de datos para la red neuronal

- Comprobar que los valores obtenidos por las funciones de simetría representan correctamente al átomo:
  - Los valores de representación se deben poder diferenciar lo máximo posible.
  - Para cada pareja de átomos debe existir una correlación en la magnitud entre los valores de representación en términos de funciones de simetría (que definen su entorno físico) y las fuerzas a las que están sometidos dichos átomos. Si $d_{ij} \rightarrow 0$, enotonces $f_{ij} \rightarrow 0$.
- Implementar la relación entre d_{ij} y f_{ij}, ejecutarla, escribir los resultados en ficheros para poder representarlos en una gráfica y analizar dichas gráficas

$$d_{ij} = \sqrt{\sum_{\alpha} \left(G_i^2 - G_j^2\right)^2}$$

$$F_{ij} = \left|\vec{F}_i - \vec{F}_j\right|$$



valores de representación normalizados

Módulo 3 → Pre-condicionado de datos para la red neuronal

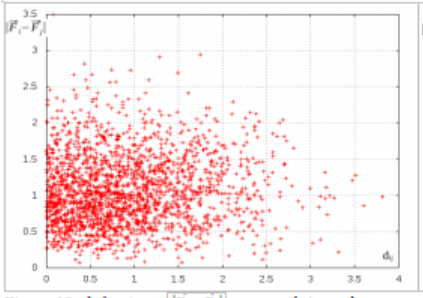


Figura 10: d_{ij} frente a $|\vec{F}_i - \vec{F}_j|$ con un único valor.

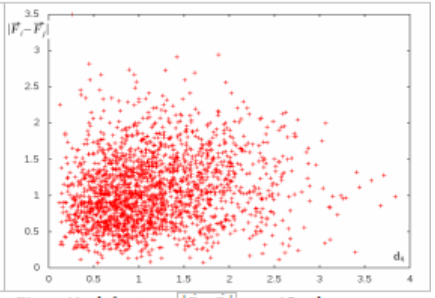


Figura 11: d_{ij} frente a $|\vec{F}_i - \vec{F}_j|$ con 15 valores.

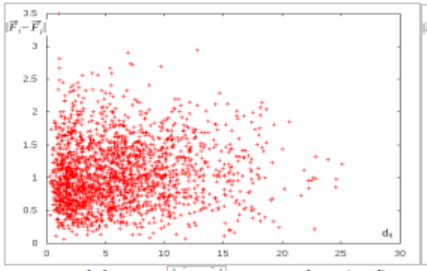


Figura 12: d_{ij} frente a $|\vec{F}_i - \vec{F}_j|$ con 30 valores (total).

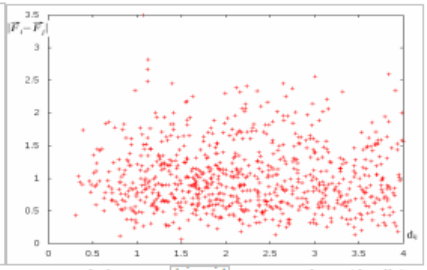


Figura 13: d_{ij} frente a $|\vec{F}_i - \vec{F}_j|$ con 30 valores (detalle).

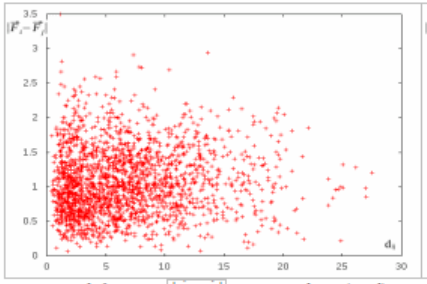


Figura 14: d_{ij} frente a $|\vec{F}_i - \vec{F}_j|$ con 57 valores (total).

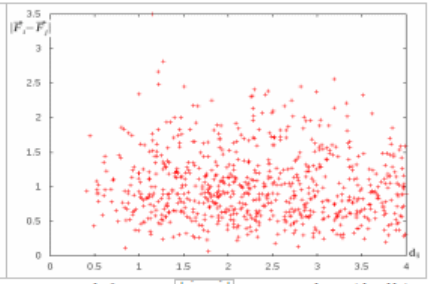


Figura 15: d_{ij} frente a $|\vec{F}_i - \vec{F}_j|$ con 57 valores (detalle).

- Para un único valor de representación la relación de magnitud está descompensada. Se tienen muchos valores nulos o muy cercanos al cero para d_{ij} con $f_{ij} \neq 0$, las fuerzas a que están sometidos los átomos i y j son diferentes.
- Para quince valores de representación la relación empieza a ser más equitativa y los puntos se alejan del cero y se dispersan.
- A partir de treinta valores de representación se representa una relación de magnitud equitativa.

Módulo 3 → Pre-condicionado de datos para la red neuronal

- Con el objetivo de representar una configuración con los mínimos errores y obtener una capacidad de ajuste de la red neuronal lo más eficiente posible, los datos se deben normalizar.
- Se utilizará el método de *ranging* con los límites [-1 , 1], aplicando la función siguiente a todos los valores de representación:

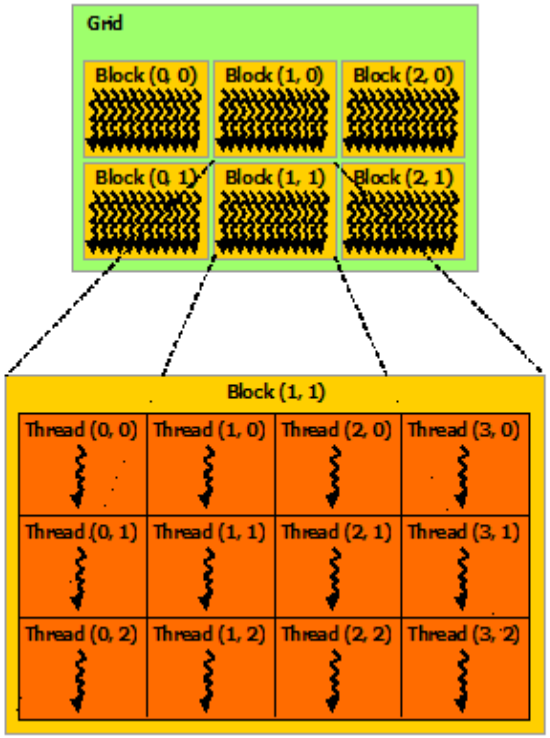
$$G_i^{\text{scaled}} = \frac{2\left(G_i - G_{i,\min}\right)}{G_{i,\max} - G_{i,\min}} - 1,$$

- La decisión relativa al método de normalización (*ranging*) viene dada porque los nodos de la red neuronal utilizarán funciones de estimulación tipo sigmoide centradas en cero, y cuya respuesta estará acotada entre -1 y 1.

$$y = \frac{1}{1 + e^{-x}}$$ Función sigmoide

- Finalmente, se escribirán los ficheros de salida correspondientes.

Módulo 3 → Paralelización GPU



Arquitectura lógica GBT (Grid-Block-Thread) en CUDA.

- Dentro de la arquitectura CUDA se deben configurar las llamadas al código que se ejecutará en la GPU (*kernel*), definiendo el número de bloques y de hilos a utilizar.
- Se decide asignar los bloques a los átomos centrales i (sobre los que se calcula las funciones de simetría) y los hilos a los átomos vecinos de primera instancia j (sobre los que se pivota en primera instancia).
- Este diseño ahorra dos bucles: el de recorrer los átomos centrales y el de pivotar sobre los vecinos de primera instancia.
- La llamada al *kernel* resultará en la siguiente forma:
  - **kernel**<<<número_de_átomos, número_de_átomos>>>(parámetro1, parámetro2, …,parámetro N);

Módulo 3 → Paralelización GPU

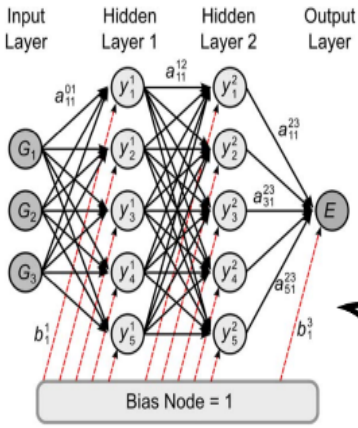
- Reseñas:
  - Se utiliza un acumulador en memoria compartida.
  - El cálculo de distancias entre átomos se debe realizar teniendo en cuenta la periodicidad de la caja.
  - El cálculo del coseno entre los tres átomos se realiza mediante la regla del coseno dado que se tienen las distancias entre átomos.
  - Todas las llamadas a funciones matemáticas dentro de la GPU se realizan con versiones "intrinsics".
  - Se debe minimizar el uso de la operación raíz cuadrada por lo que el radio del *cutoff* (R_c) se utiliza en su forma cuadrática R_c^2. La raíz cuadrada, sólo es necesaria dentro del cálculo del ángulo que forman los tres átomos.
- Mejoras:
  - Si se dispone de GPUs con varios procesadores (multi-GPU) se puede paralelizar las llamadas a los *kernels*: de forma que a cada *grid* se le asigna el átomo central i, a cada bloque el átomo vecino de primera instancia j y a cada hilo el átomo vecino de segunda instancia k.

Módulo 4 → Ejecución de la red neuronal (modo aprendizaje):

- El objetivo de este módulo es ejecutar la red neuronal con todos los conjuntos de átomos de aprendizaje, obteniendo un ajuste óptimo de los parámetros de dicha red:
  1. Leer los valores de representación normalizados de cada átomo para una configuración dada, inicializar una red neuronal por átomo, obtener la predicción atómica de dicha red neuronal y sumar todas las predicciones atómicas resultantes de las redes neuronales obteniendo la predicción de la configuración.
  2. Calcular el error de predicción, pudiéndose dar dos casos:
     1. Si el error supera un determinado umbral, realizar una optimización de los parámetros (comunes a todas las redes neuronales atómicas) y volver al paso 1 con el mismo conjunto de átomos.
     2. Si el error se encuentra dentro del umbral o el método converge se debe volver al punto 1 con un nuevo conjunto de átomos.

- Se definen dos operaciones diferentes:
  - Época de la red neuronal: leer los datos de representación normalizados, inicializar la red neuronal, calcular la predicción y el error de predicción.
  - Optimización: optimizar los parámetros de la red neuronal de modo que se minimice el error de predicción.

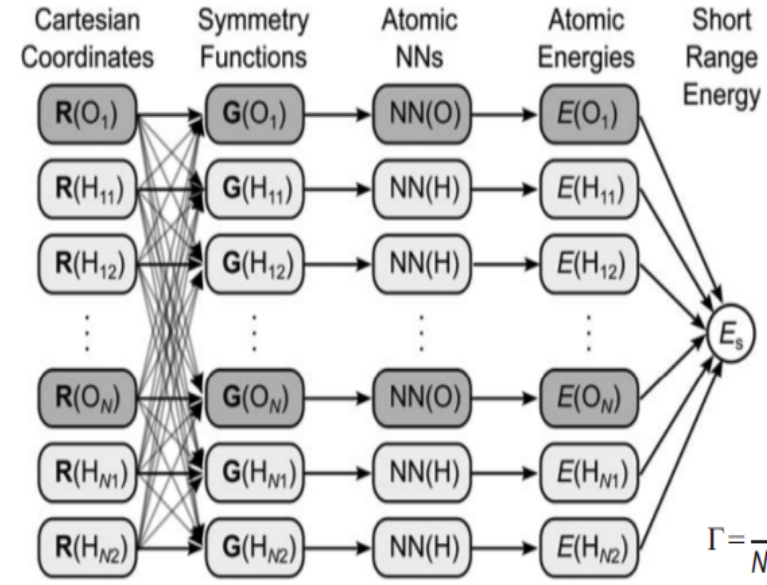
Módulo 4 → Ejecución de la red neuronal (modo aprendizaje):



$$y_i^j = f_i^j\left(b_i^j + \sum_{k=1}^{N_{j-1}} a_{k,i}^{j-1,j} \cdot y_k^{j-1}\right) \quad (5)$$

$$E = f_1^3\left(b_1^3 + \sum_{k=1}^{5} a_{k1}^{23} \cdot f_k^2\left(b_k^2 + \sum_{j=1}^{5} a_{jk}^{12} \cdot f_j^1\left(b_j^1 + \sum_{i=1}^{|G|} a_{ij}^{01} \cdot G_i\right)\right)\right) \quad (6)$$

Época de la red neuronal



$$E_s = \sum_{i=1}^{N_{atom}} E_i. \quad (7)$$

$$\Gamma = \frac{1}{N_{struct}} \sum_{i=1}^{N_{struct}} \left(E_{NN}^i - E_{Ref}^i\right)^2 \quad (8)$$

- Cada átomo inicializa una red neuronal atómica:
- ✓ G_i valores de representación del átomo como capa de entrada (*input*)
- ✓ Dos capas ocultas j.
- ✓ Cinco neuronas por capa y^j_k e y^j_k.
- ✓ Una neurona de salida (*output*) E.
- ✓ N parámetros $a^{j-1}{}_k{}^j{}_i$. Todos los parámetros serán comunes a todas las redes neuronales atómicas y serán inicializados con valores aleatorios en el rango [-1, 1].
- ✓ Una función de estimulación neuronal: $y^j_i = f^j_i$. Para las dos capas ocultas f^j_i será la sigmoide. Para la neurona de salida f^j_i será una función lineal.
- Todas las estimaciones de energía atómicas E_i se suman para obtener la predicción del sistema E_s (7).
- E_s será utilizado para calcular el error de la predicción (8) donde N_{struct} es igual al número de átomos, E^i_{NN} es igual a E_s y E^i_{Ref} es la energía de referencia.

Módulo 4 → Ejecución de la red neuronal (modo aprendizaje):

- La optimización o aprendizaje se basa en ajustar, tras la ejecución de cada época, el valor de los N parámetros $a^{j-1}_{k\,i}$ de la red neuronal con el objetivo de minimizar el error de predicción.
- Se decide utilizar el optimizador COMPLEX-BCPOL de la librería IMSL.
- La ejecución de la optimización de los parámetros se realizará de forma secuencial mediante la librería IMSL.
- Se paralelizará el cálculo de la función de coste para que devuelva el error de predicción de un número determinado (NUMBER_OF_BOXES_TO_OPT) de configuraciones de átomos.
- Kernel:
  - Se crearán NUMBER_OF_BOXES_TO_OPT bloques que representarán las cajas de átomos.
  - Se crearán tantos hilos como átomos se tenga por caja por lo que cada hilo representará una red neuronal atómica.
  - **kernel**<<<NUMBER_OF_BOXES_TO_OPT, número_átomos_por_caja>>>(parámetro1, parámetro2, …,parámetro N);

Módulo 4 → Ejecución de la red neuronal (modo aprendizaje):

- Reseñas:
  - Se utiliza un vector en memoria compartida.
  - Todas las llamadas a funciones matemáticas dentro de la GPU se realizan con sus versiones "intrinsics".
  - Se realiza una reducción binaria del vector de energías atómicas.
  - El parámetro BIAS → OUPUT_NEURON se inicializa como el valor medio de la energía y sus límites inferior y superior como {valor_mínimo_energía, valor_máximo_energía}.
  - El vector de valores de representación de los átomos se genera tras serializar en la CPU todos los valores de representación de cada átomo de cada caja.
- Mejoras:
  - El uso de un filtro digital extendido de Kalman mejoraría la optimización de parámetros ya que se trata de un método dirigido y con memoria.
  - El uso de gradientes y de las fuerzas en el proceso de optimización también mejoraría notablemente el rendimiento de la operación de optimización.

Módulo 5 → Ejecución de la red neuronal (modo predicción):

- El objetivo de este módulo es ejecutar la red neuronal, utilizando los parámetros ya optimizados tras el aprendizaje, con todos los conjuntos de átomos de predicción.
- Se obtiene la predicción de la energía de cada secuencia temporal atómica.

1. Leer los valores de representación normalizados de cada átomo para una configuración dada, inicializar una red neuronal por átomo, obtener la predicción atómica de dicha red neuronal y sumar todas las predicciones atómicas resultantes de las redes neuronales obteniendo la predicción de la configuración (caja).
2. Repetir el paso 1 hasta que no queden configuraciones a predecir.

- Por lo tanto, se define una única operación:
  - Época de la red neuronal: leer los datos de representación normalizados, inicializar la red neuronal y calcular la predicción. No se calcula el error de predicción dado que sólo se quiere obtener la predicción de la energía.

Módulo 6 → Proceso de ficheros de salida

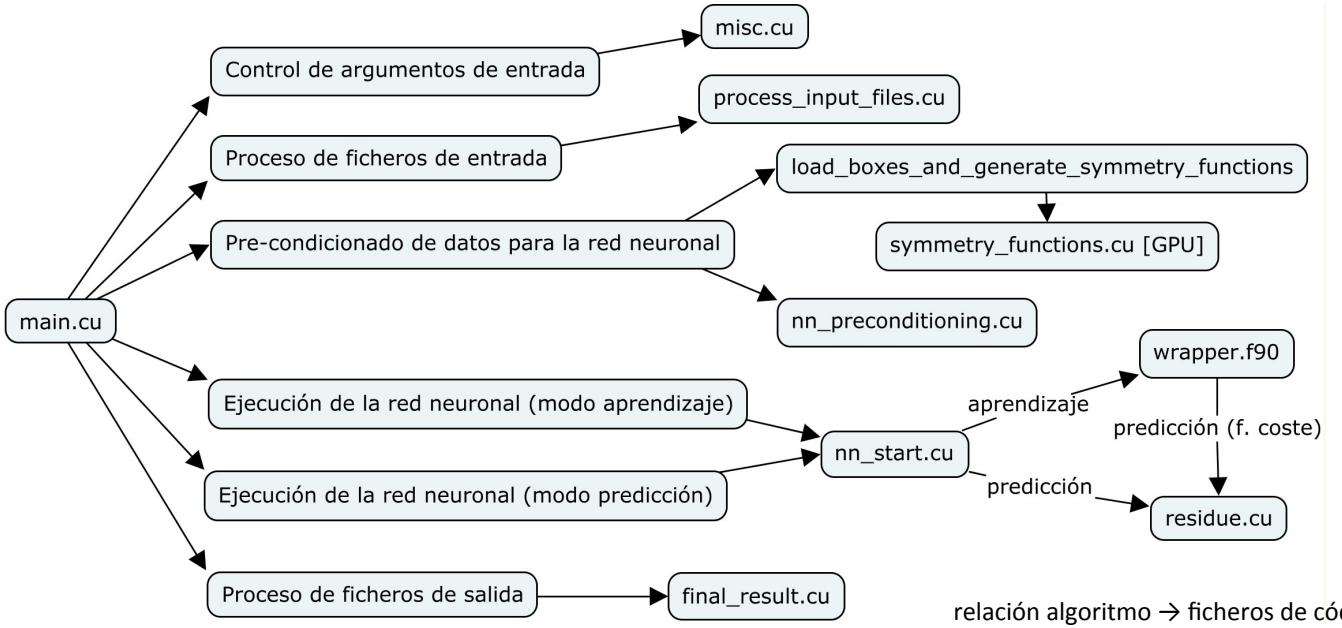
- El objetivo de este módulo es calcular el error medio de predicción y generar las gráficas que indiquen la calidad del aprendizaje y de la predicción.
- Se imprimirá por pantalla el error medio de predicción y se crearán los ficheros “data/LEARNING_PROCESS.png” y “data/FINAL_PREDICTION.png (requiere tener instalado GNUplot).

IMPLEMENTACIÓN

- El programa se ha codificado en los lenguajes de programación CUDA_C/C++ y FORTRAN.
- Como decisiones de documentación del código se genera un fichero de cabecera (.h) para cada fichero de código (.cu). En dicho fichero se encuentra una descripción del propósito (qué hace) general del código y otra para cada una de las funciones que ejecuta dicho código.
- Como decisiones de implementación del código:
  - Para cada fichero de código (.cu) se genera un fichero de cabecera (.h) con las definiciones de las funciones.
  - El fichero del adaptador "wrapper.f90" no tiene fichero de cabecera.
  - Se crea un fichero de configuración (conf.h) en donde se definen todas las variables que servirán de ajuste o serán parámetros de la aplicación.
  - Se genera un fichero de estructuras de datos (structs.h) en donde se definen e implementan todas las estructuras de datos necesarias de la aplicación.
  - Se crea un sistema de mensajes de error y control con objeto de informar sobre posibles fallos en la ejecución de la aplicación.
  - Se define un fichero *Makefile* que compila y ensambla el programa generando todos los ficheros objeto y el binario final.
  - Se crea un fichero de entrada (gnuplot_plot_results.in) para el programa externo GNUPLOT con objeto de generar los ficheros de salida (.PNG) con las gráficas.
  - Se crea un fichero .SH con ejemplos de ejecuciones de la aplicación.
  - Los nombres de variables y de funciones siempre son lo más explícitos posibles respecto a su uso.

Licencia y repositorios

- El código fuente de la aplicación (versión 1.0 estable) posee licencia GPLv3 (*GNU General Public License*) y se puede descargar desde los siguientes repositorios:
  - https://github.com/adpozuelo/HDNNP
  - https://bitbucket.org/adpozuelo/hdnnp/overview
- En ambos repositorios se exponen los requisitos *hardware* y *software*, así como las instrucciones de descarga e instalación.
- Se han incluido ficheros de entrada con datos *ab initio* como ejemplo (50 configuraciones por fichero).
- En la máquina cedida por la UOC para las pruebas y simulaciones sí que se encuentran los ficheros de datos de referencia *ab initio* íntegros.

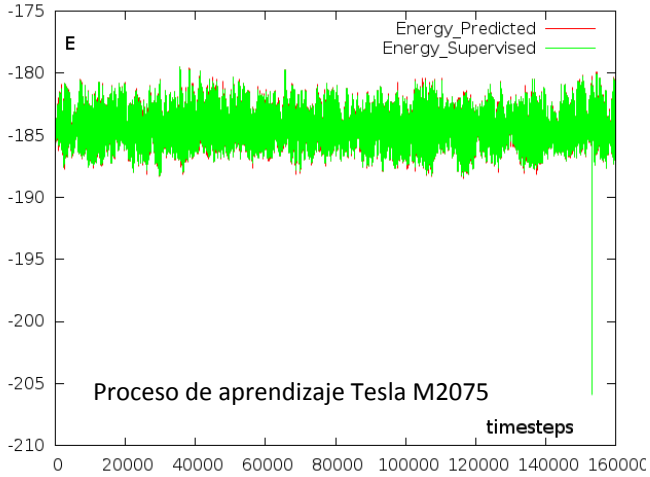


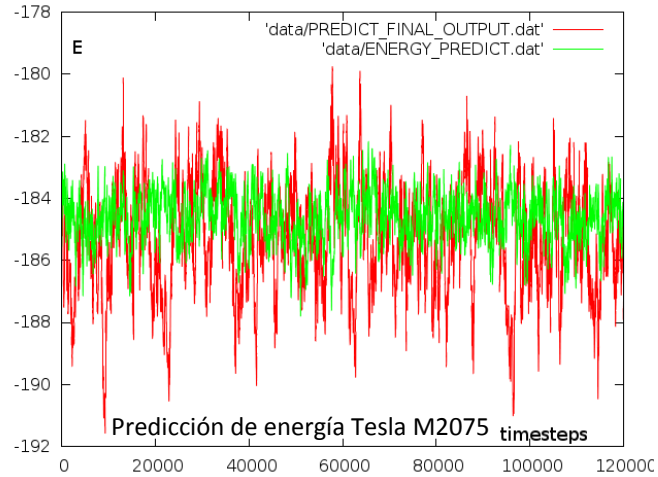
relación algoritmo → ficheros de código.

PRUEBAS Y SIMULACIONES

- Dado el carácter modular de la aplicación no ha sido complejo aislar los problemas y solucionarlos sin que afecten al resto de la aplicación.
- Debido a que los módulos 1, 2 y 3 se pueden ejecutar una única vez y, a partir de entonces, los módulos 4, 5 y 6 son independientes de los primeros se ha podido acelerar el proceso de diseño → implementación → pruebas.
- Señalar la dificultad de depurar errores del código que se ejecuta en la GPU.
- Los problemas afrontados durante las fases de diseño e implementación, respecto con el optimizador de los parámetros de la red neuronal, han obligado a ajustar la red neuronal y el optimizador (con objeto de reducir el tiempo de las simulaciones) a los siguientes parámetros :
  - Número de valores de representación por átomo = 30.
  - Número de neuronas en cada capa oculta de la red neuronal = 4.
  - Número máximo de llamadas a la función de coste = 80000.
  - Sólo se permite predecir un único fichero de configuraciones.
- Se dispone de seis ficheros de datos *ab initio* con las temperaturas 673K, 723K, 773K, 823K, 873K y 973K y de cinco GPUs modelos GTX590, GTX660, GTX960, GTX980 y TeslaM2075, se realizan las siguientes simulaciones:

Dos conjuntos de aprendizaje (673K y 973K) y uno de predicción (773K)

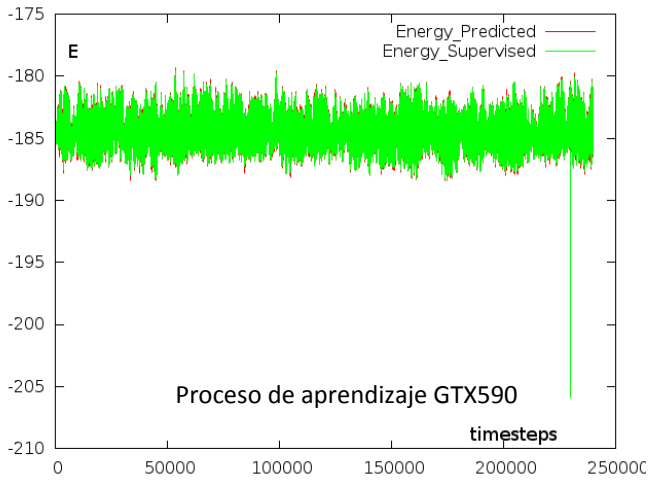


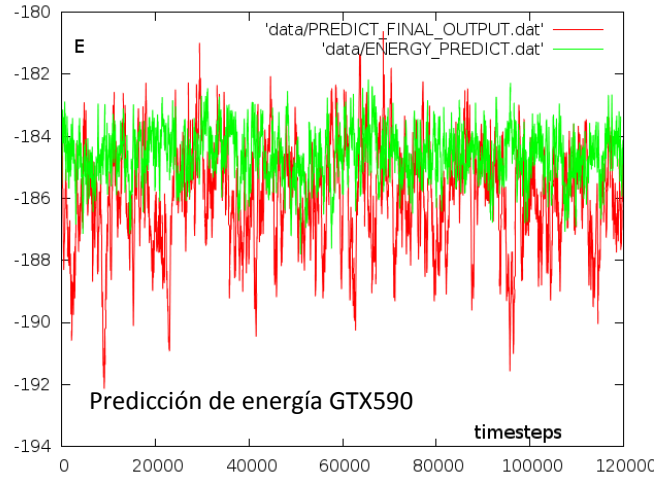


2-Aprendizaje	Tiempo (m)	Ē de predicción
GTX 590	740	0.395462
GTX 660	755	0.527924
GTX 960	383	0.408463
GTX 980	340	0.553131
Tesla M2075	435	0.720734

- El proceso de aprendizaje es correcto dado que la diferencia entre la energía predicha (rojo), realizada tras la optimización de los parámetros de la red neuronal, y la energía supervisada (verde) converge y es pequeña.
- El ajuste en la predicción de la energía no es óptimo dado que la predicción (rojo) no se asemeja a la energía supervisada (verde).
- El error medio de predicción (Ē) es alto lo que explica el mal ajuste representado en la predicción de la energía.
- Se concluye que el sistema aprende correctamente pero no predice con el ajuste deseado.
- Las GPUs más modernas y/o potentes necesitan menos tiempo para realizar la simulación y que todas ellas poseen un error medio de predicción similar (del mismo orden de magnitud).

Tres conjuntos de aprendizaje (673K, 723K y 973K) y uno de predicción (773K)

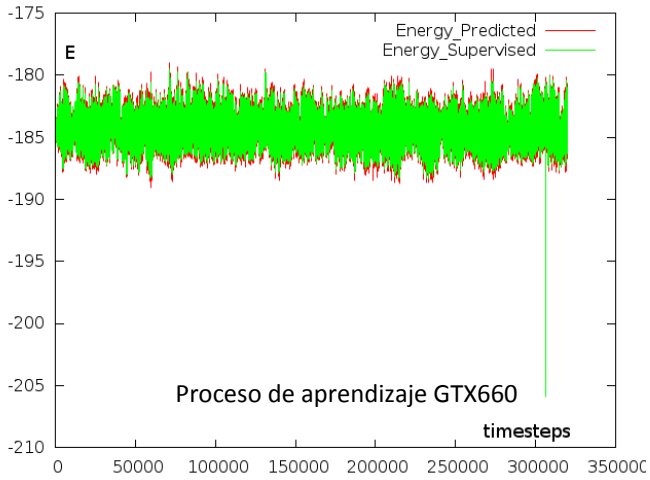


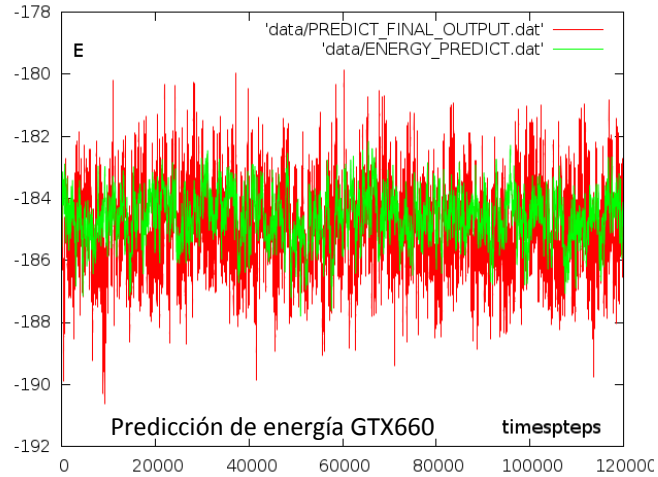


3-Aprendizaje	Tiempo (m)	Ē de predicción
GTX 590	1087	1.399277
GTX 660	1034	1.256638
GTX 960	488	1.067123
GTX 980	307	1.328995
Tesla M2075	576	0.819427

- El proceso de aprendizaje es correcto dado que la diferencia entre la energía predicha (rojo), realizada tras la optimización de los parámetros de la red neuronal, y la energía supervisada (verde) converge y es pequeña.
- El ajuste en la predicción de la energía no es óptimo dado que la predicción (rojo) no se asemeja a la energía supervisada (verde).
- El error medio de predicción (Ē) es alto (incluso más que antes) lo que explica el mal ajuste representado en la predicción de la energía.
- Se concluye que el sistema aprende correctamente pero no predice con el ajuste deseado.
- Las GPUs más modernas y/o potentes necesitan menos tiempo para realizar la simulación y que todas ellas poseen un error medio de predicción similar (del mismo orden de magnitud).

Cuatro conjuntos de aprendizaje (673K, 723K, 873K y 973K) y uno de predicción (773K)

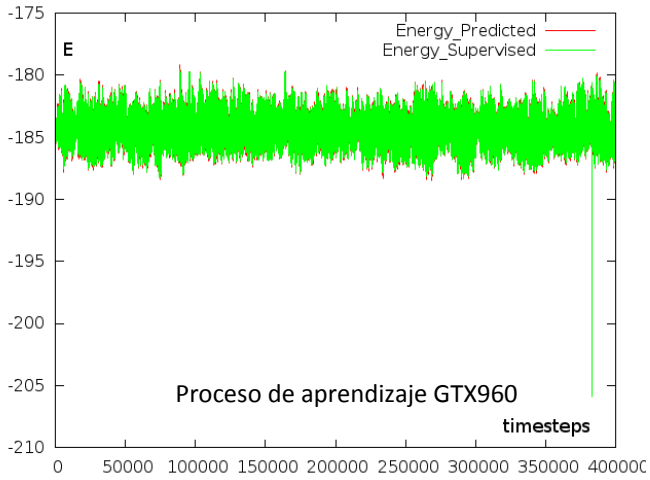


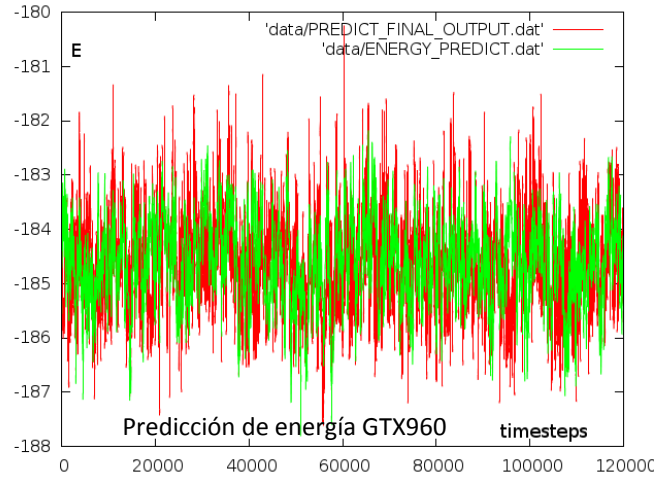


4-Aprendizaje	Tiempo (m)	Ē de predicción
GTX 590	1322	0.320465
GTX 660	1399	0.396881
GTX 960	763	0.638229
GTX 980	409	0.631714
Tesla M2075	732	0.707733

- El proceso de aprendizaje es correcto dado que la diferencia entre la energía predicha (rojo), realizada tras la optimización de los parámetros de la red neuronal, y la energía supervisada (verde) converge y es pequeña.
- El ajuste en la predicción de la energía no es óptimo dado que la predicción (rojo) no se asemeja a la energía supervisada (verde).
- El error medio de predicción (Ē) es alto lo que explica el mal ajuste representado en la predicción de la energía.
- Se concluye que el sistema aprende correctamente pero no predice con el ajuste deseado.
- Se observa una mejoría en el ajuste de la predicción y en el error medio de predicción de la energía respecto de los casos anteriores.

Cinco conjuntos de aprendizaje (673K, 723K, 823K, 873K y 973K) y uno de predicción (773K)



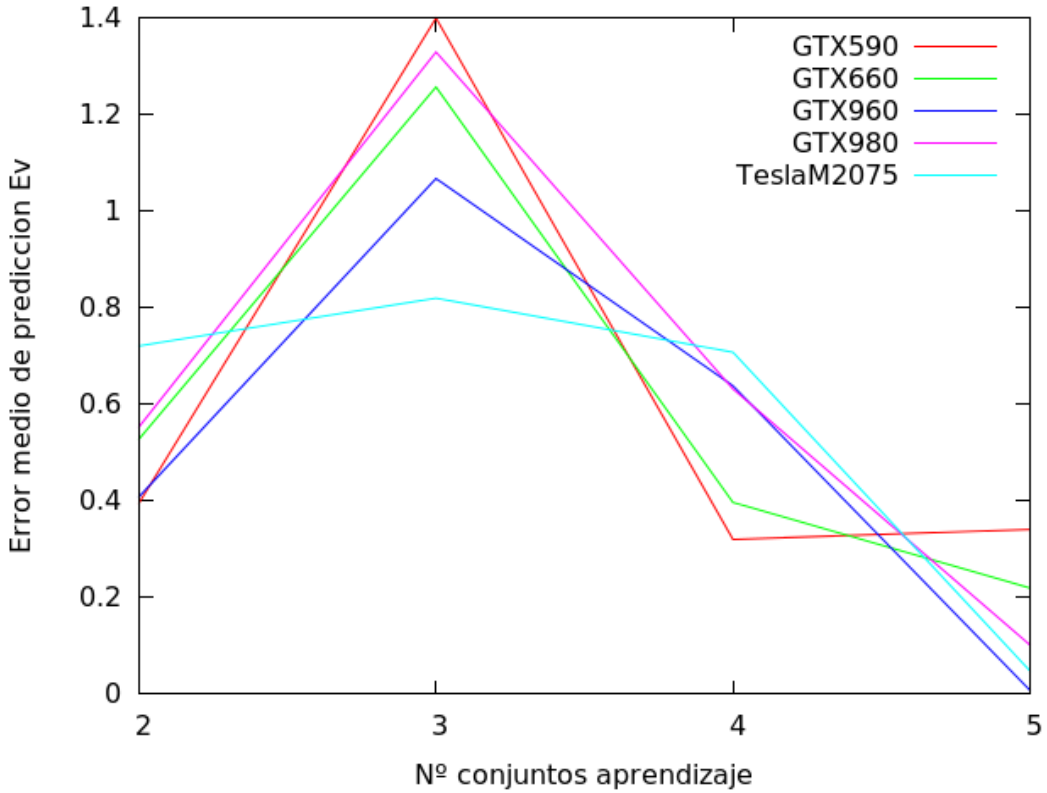


5-Aprendizaje	Tiempo (m)	Ē de predicción
GTX 590	1371	0.340820
GTX 660	1726	0.219604
GTX 960	935	0.007904
GTX 980	603	0.101883
Tesla M2075	1019	0.048141

- El proceso de aprendizaje es correcto dado que la diferencia entre la energía predicha (rojo), realizada tras la optimización de los parámetros de la red neuronal, y la energía supervisada (verde) converge y es pequeña.
- El ajuste en la predicción de la energía es más óptimo que en los casos anteriores dado que la predicción (rojo) se asemeja a la energía supervisada (verde).
- El error medio de predicción (Ē) es más bajo que en los casos anteriores, siendo realmente óptimo en los casos de la GTX960 (0.007904 eV) y de la Tesla (0.048141 eV)
- Se concluye que el sistema aprende correctamente y predice con el ajuste deseado.
- Se observa una notable mejoría en el ajuste de la predicción y en el error medio de predicción de la energía respecto de los casos anteriores.

CONCLUSIONES

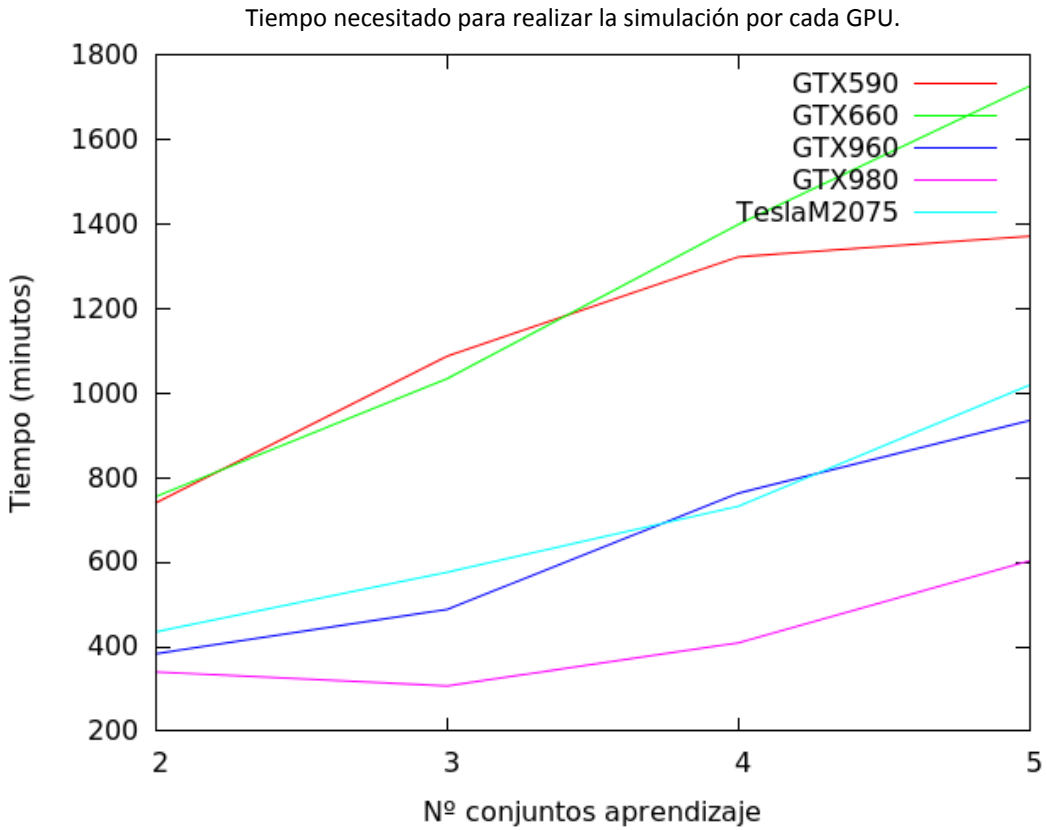
- Error medio de predicción:
  - El error medio de predicción disminuye según se incrementan los conjuntos de aprendizaje con los que se alimenta la red neuronal.
  - Los distintos errores en cada tipo de GPU pueden estar relacionados con la diferencia en el número de núcleos de cada *hardware*.



Error medio de predicción Ē (eV) por cada GPU.

CONCLUSIONES

- Tiempo:
  - El tiempo necesario para realizar la simulación aumenta de forma más o menos lineal respecto al número de conjuntos de aprendizaje utilizados.



CONCLUSIONES

- Como cualquier sistema inteligente la red neuronal HDNNP necesita tener más experiencia (ver más cosas) para tomar mejores decisiones (realizar predicciones).
- El futuro:
  - Desde el punto de vista del ajuste de la red neuronal se realizarán simulaciones variando la configuración de la misma con objeto de comprobar si con más parámetros entre neuronas y BIAS se consigue un mejor ajuste con menos conjuntos de aprendizaje.
  - Desde el punto de vista del diseño, y concretamente del optimizador:
    - Se utilizará una nueva función de coste en el proceso de optimización que, junto a la diferencia de energías predichas y supervisadas, incorpore la diferencia entre los gradientes de dichas energías como estimación de las fuerzas sobre cada átomo y las fuerzas *ab initio* de las que se dispone en los datos de referencia.
    - Se estudiará, diseñará e implementará un filtro extendido de Kalman con objeto de sustituir el actual optimizador.
  - Desde el punto de vista de los datos se realizarán simulaciones variando el número de átomos por configuración, con objeto de comprobar el ajuste de la red neuronal de alta dimensionalidad con conjuntos de átomos de distinto tamaño.

BIBLIOGRAFÍA

1. "Constructing High-Dimensional Neural Network Potentials: A Tutorial Review", Jörg Behler. *Int. J. Quantum Chem*. **2015**, *115*, 1032-1050. DOI: 10.1002/qua.24890
2. "Neural Network Models of Potential Energy Surfaces: Prototypical Examples", James B. Witkoskie and Douglas J. Doren. *Journal of Chemical Theory and Computation.* **2005**, *1 (1)*, 14-23. DOI: 10.1021/ct049976i
3. "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces", Behler y Parrinello, Phys. Rev. Lett. **2007**, *98*, 146401; DOI: 10.1103/PhysRevLett.98.146401
4. "Neural network models of potential energy surfaces", Thomas B. Blank, Steven D. Brown, August W. Calhoun, and Douglas J. Doren. J. Chem. Phys. **1995,** *103*, 4129; DOI: 10.1063/1.469597.
5. "Atom-centered symmetry functions for constructing high-dimensional neural network potentials", Jörg Behler. J. Chem. Phys. **2011**, *134*, 074106; DOI: 10.1063/1.3553717.
6. CUDA Programming - ISBN: 978-0-12-415933-4.
7. CUDA By Example - ISBN 978-0-13-138768-3.
8. CUDA Toolkit Documentation – http://docs.nvidia.com/cuda/index.html.