TRABAJO DE FIN DE GRADO

Grado en Física

Inteligencia Artificial para aprender Hamiltonianos cuánticos

Samuel Blasco Osés

Director:

David Zueco Láinez

Departamento de Física de la Materia Condensada

Facultad de Ciencias, Universidad de Zaragoza

Curso 2023-2024

Índice

1. Introducción

A lo largo del grado nos hemos habituado a que, para realizar distintas medidas en el laboratorio, se seleccionaban unos parámetros, y se medían distintas curvas obtenidas en función de estos. Pero en muchos experimentos, hay varios de esos parámetros que no son conocidos y se deben extraer como resultado de analizar las curvas obtenidas. Esto requiere de mucho tiempo y, en ocasiones, de modelos matemáticos muy complejos como los métodos numéricos, los cuales tienen un coste computacional muy elevado.

Además, día a día se realizan innumerables experimentos en los distintos laboratorios del mundo, de los cuales, muchos requieren un presupuesto muy elevado y tan solo permiten una toma de medidas muy limitada.

Con este trabajo pretendemos solventar estos dos problemas mediante el uso de redes neuronales. La idea es demostrar que se pueden entrenar modelos basados en ellas, con datos sintéticos (nunca se han medido en un laboratorio), y mediante este entrenamiento, realizar el proceso inverso de obtener los distintos parámetros en base a las curvas experimentales. De esta forma estaríamos ahorrando dinero y recursos en la toma de medidas, y no requeriríamos de una potencia computacional tan elevada como la de los métodos de resolución numérica.

En este trabajo, nos centraremos en los datos obtenidos de [1], donde se hace uso de moléculas magnéticas como el compuesto DPPH para medir la interacción entre la luz y la materia en resonadores ópticos. En este tipo de experimentos, la variable más difícil de obtener es el conocido factor g, que determina el acoplamiento luz-materia; sobre todo, es especialmente complicado de medir para los casos de acoplamiento débil.

2. Electrodinámica cuántica de una cavidad (cQED)

El sistema que vamos a estudiar es una cavidad resonante, en la que tenemos una excitación generada por un campo EM, y en la que se introduce un átomo, el cual se puede aproximar como un sistema de dos niveles (qubit). Un sencillo esquema del sistema puede verse en la Fig. 2.1.

Este sistema tendrá un Hamiltoniano constituido por tres términos:

$$H = H_{EM} + H_{at} + H_{int} \tag{2.1}$$

donde, el primer término es la contribución del campo EM de la cavidad, el segundo término es la parte del átomo de dos niveles, y el tercero se corresponde al término de interacción entre el átomo y el campo EM.

A continuación, vamos a obtener cada uno de los términos.

2.1. Cavidad (campo EM)

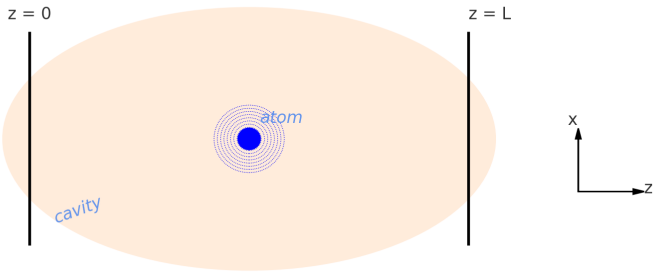


Figura 2.1: Esquema de una cavidad

Nuestro punto de partida son las ecuaciones de Maxwell, y concretamente nos centraremos en las definiciones del campo eléctrico y magnético en función del potencial magnético vector:

$$\vec{B} = \nabla \times \vec{A} \tag{2.2}$$

$$\vec{E} = -\frac{\partial \vec{A}}{\partial t} - \nabla\phi \tag{2.3}$$

Para mayor comodidad y poder simplificar los cálculos, vamos a trabajar en el gauge de Coulomb ($\nabla \cdot \vec{A} = 0$), así como en ausencia de fuentes ($\rho = 0$, $\vec{j} = 0$).

De esta forma, se resuelve la ecuación de ondas,

$$(\partial_t - c^2\nabla)\vec{A} = 0 \tag{2.4}$$

y teniendo en cuenta que la cavidad se puede aproximar como unos espejos, lo que nos da unas condiciones de contorno para el campo eléctrico,

$$\vec{E}(z = 0) = \vec{E}(z = L) = 0 \tag{2.5}$$

Con estas condiciones y usando el método de separación de variables ($A \sim f(z)q(t)$), se tiene que

$$\begin{cases} f(z) = \sin kz, \ k = \frac{\pi}{L}n, \ n \in \mathbb{Z} \\ q(t) = e^{i\omega t} \end{cases} \tag{2.6}$$

Así pues, definiendo

$$\vec{A} = N f(z) q(t)\, \hat{x} \tag{2.7}$$

y utilizando (2.2) y (2.3), para el caso de n modos, tenemos que:

$$\vec{E} = -N \sum_n \sin k_n z\, \dot{q}_n(t)\, \hat{x} \tag{2.8}$$

$$\vec{B} = N \sum_n k_n \cos k_n z\, q_n(t)\, \hat{y} \tag{2.9}$$

con N un constante de normalización.

Ahora calculamos la energía,

$$E = \frac{1}{2}\int \varepsilon_0 E^2 + \frac{1}{\mu_0}B^2 dz = \frac{N^2}{2}\sum_n \varepsilon_0 \dot{q}_n^2(t)\frac{L}{2} + \frac{L}{2\mu_0}k_n^2 q_n^2 = \frac{1}{2}N^2\varepsilon_0\frac{L}{2}\sum_n \dot{q}_n^2(t) + \frac{k_n}{\varepsilon_0\mu_0}q_n^2(t)$$

Si deshacemos la constante N, obtendremos la energía como una suma de osciladores armónicos [2, Capítulo 10.3],

$$E = \frac{1}{2}\sum_n \dot{q}_n(t)^2 + \omega_n^2 q_n^2(t)\ , \quad \omega_n = c\,k_n = c\frac{\pi}{L}n, \ n \in \mathbb{Z} \tag{2.10}$$

Para obtener el hamiltoniano, hacemos uso de los operadores creación y aniquilación, que se definen como [2, Capítulo 10.3]:

$$\begin{cases} a_n = \frac{1}{\sqrt{2\hbar\omega_n}}\left(\omega_n q_n + i p_n\right) \\ a_n^\dagger = \frac{1}{\sqrt{2\hbar\omega_n}}\left(\omega_n q_n - i p_n\right) \end{cases} \implies \begin{cases} q_n = \sqrt{\frac{\hbar}{2\omega_n}}\left(a_n + a_n^\dagger\right) \\ \dot{q}_n \equiv p_n = -i\sqrt{\frac{\hbar\omega_n}{2}}\left(a_n - a_n^\dagger\right) \end{cases} \tag{2.11}$$

Y de esta forma, llegamos al hamiltoniano de una suma de osciladores armónicos:

$$H = \sum_n \hbar\omega_n (a_n^\dagger a_n + \frac{1}{2}) \tag{2.12}$$

Pero esta ecuación puede ser modificada, como se explica en [3, Capítulo 2.1]. Esto es porque el hamiltoniano escrito de esta forma, tiene un infinito, el cual se debe a que la mínima energía del oscilador armónico cuantizado es $1/2$ de cuanto y no cero. Como tenemos una suma de infinitos osciladores, este infinito es la suma de la energía de punto cero de todos ellos. Puesto que los procesos en los que este hamiltoniano participa, describen el intercambio de energía entre el campo y otro sistema, y solo nos interesa el intercambio de energía, podemos omitir el término de energía de punto cero. Finalmente,

$$H = \sum_n \hbar\omega_n a_n^\dagger a_n \tag{2.13}$$

2.2. Átomo

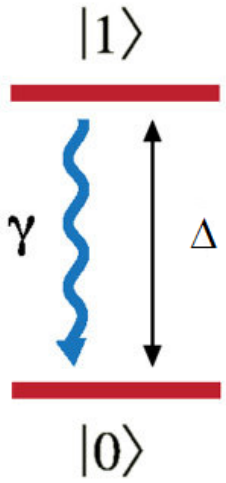


Figura 2.2: Sistema de dos niveles

Tenemos un átomo que se comporta como un sistema de dos niveles (como el de la Fig. 2.2), con un estado fundamental $|0\rangle$, y un estado excitado $|1\rangle$; cuyas energías son $E_0 = \hbar\omega_0$ y $E_1 = \hbar\omega_1$. Además, se define la diferencia entre los dos niveles energéticos como $\Delta = \omega_1 - \omega_0$.

Sabemos que,

$$|0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad , \quad |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{2.14}$$

y que las energías son los autovalores del hamiltoniano con autoestados $|0\rangle$ y $|1\rangle$, respectivamente, llegamos a que:

$$\begin{aligned} H &= E_0|0\rangle\langle 0| + E_1|1\rangle\langle 1| = \begin{pmatrix} E_1 & 0 \\ 0 & E_0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{2}(E_1+E_0) + \frac{1}{2}(E_1-E_0) & 0 \\ 0 & \frac{1}{2}(E_1+E_0) - \frac{1}{2}(E_1-E_0) \end{pmatrix} \\ &\left\{ E = \frac{1}{2}(E_1+E_0) \right\} \Rightarrow H = E\,\mathbb{I} + \frac{1}{2}\hbar\Delta \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{aligned} \tag{2.15}$$

El término de energía constante no contribuye, ya que, como se ha mencionado previamente, nos interesan los intercambios de energía. De esta forma,

$$H = \frac{1}{2}\hbar\Delta\sigma^z \tag{2.16}$$

2.3. Término de interacción

Si suponemos que el átomo se comporta como un dipolo, su interacción con el campo será de la forma, $H = \vec{d} \cdot \vec{E}$, donde $\vec{d} = e\,\vec{r}$, es el momento dipolar eléctrico.

Sabemos que el potencial atómico es central y, por tanto, invariante bajo $r \to -r$. Por otra parte, las funciones de onda tienen paridad bien definida, lo que, unido a que $\vec{r}$ es antisimétrico, hace que los términos de la diagonal desaparezcan:

$$\langle j|\vec{r}|j\rangle = \int d^3r |\Psi_j(\vec{r})|^2 \vec{r} = 0 \tag{2.17}$$

Sin embargo, los elementos de fuera de la diagonal resultan:

$$\begin{aligned} e\,\langle 0|\vec{r}|1\rangle &= \int d^3r\, \Psi_0(\vec{r})^*\, \vec{r}\, \Psi_1(\vec{r}) \equiv d \\ e\,\langle 1|\vec{r}|0\rangle &= \int d^3r\, \Psi_1(\vec{r})^*\, \vec{r}\, \Psi_0(\vec{r}) \equiv d^* \end{aligned} \tag{2.18}$$

Por lo que,

$$|\vec{d}| = \begin{pmatrix} 0 & |d| \\ |d^*| & 0 \end{pmatrix} = \sigma^x |d| = \sigma^x e|\vec{r}| \tag{2.19}$$

Lo siguiente que necesitamos hacer es escribir el campo eléctrico en función de los operadores creación y aniquilación, luego, sustituimos (2.11) en (2.8)[1]. Con esto, y usando aproximación de cavidad de modo simple (usamos solo un modo), llegamos a que,

$$\vec{E} = \vec{E}_{rms}(a + a^\dagger) \tag{2.20}$$

Para obtener $\vec{E}_{rms}$, haremos uso del *Teorema del Virial*:

$$\frac{1}{2}E = \frac{1}{2}\langle 0|E|0\rangle = \frac{1}{4}\hbar\omega = \frac{1}{2}\int_V dV \varepsilon_0 \langle 0|\vec{E}^2|0\rangle = \frac{1}{2}V\varepsilon_0 E_{rms}^2 \tag{2.21}$$

$$\Longrightarrow |\vec{E}_{rms}| \equiv E_{rms} = \sqrt{\frac{\hbar\omega}{2V\varepsilon_0}} \tag{2.22}$$

Con todo esto, obtenemos el hamiltoniano de la interacción:

$$H = e|\vec{r}|E_{rms}\sigma^x(a + a^\dagger) \tag{2.23}$$

si introducimos la variable g, que se define como el acoplo entre el campo y el átomo,

$$g = \frac{e|\vec{r}|E_{rms}}{\hbar} = e|\vec{r}|\sqrt{\frac{\omega}{\hbar V\varepsilon_0}} \tag{2.24}$$

el hamiltoniano resulta finalmente,

$$H = \hbar g \sigma^x (a + a^\dagger) \tag{2.25}$$

[1]Se ha usado que $\dot{q}(t) = i\omega q(t)$ (recordemos que $q(t) = e^{i\omega t}$), en vez de usar directamente la definición de $\dot{q}(t)$

2.4. Modelo de Rabi y aproximación RWA

Si sumamos los tres términos obtenidos anteriormente, y mantenemos la aproximación de modo único (utilizada para calcular el término de interacción), llegaremos a lo que se conoce como *Hamiltoniano del modelo cuántico de Rabi*,

$$H_{Rabi} = \frac{1}{2}\hbar\Delta E\sigma^z + \hbar\Omega a^\dagger a + \hbar g\sigma^x(a + a^\dagger) \tag{2.26}$$

siendo ΔE la diferencia entre las frecuencias de los dos niveles del átomo, y Ω la frecuencia de resonancia de la cavidad.

Es conocido que, $\sigma^x = \sigma^+ + \sigma^-$. Si sustituimos en (2.26), nos queda

$$H_{Rabi} = \frac{1}{2}\hbar\Delta E\sigma^z + \hbar\Omega a^\dagger a + \hbar g(\sigma^+ a + \sigma^- a^\dagger + \sigma^- a + \sigma^+ a^\dagger) \tag{2.27}$$

Los dos últimos sumandos del término de interacción, podemos eliminarlos haciendo uso de la aproximación RWA (aproximación de onda rotante, por sus siglas en inglés), que brinda conservación numérica. Además, esos dos términos (como se explica en [2, Capítulo 14.8]) no cumplen la conservación de la energía, porque son términos que indicarían la absorción de un fotón que bajase el nivel de energía del átomo, y la emisión de un fotón por un aumento en el nivel energético del átomo, procesos que difieren completamente de la emisión/absorción típica.

Con la eliminación de estos términos, tenemos el *Hamiltoniano de Jaynes-Cumming*:

$$H_{JC} = \frac{1}{2}\hbar\Delta E\sigma^z + \hbar\Omega a^\dagger a + \hbar g(\sigma^+ a + \sigma^- a^\dagger) \tag{2.28}$$

2.5. Ecuaciones del movimiento y función de transmisión

En el laboratorio, tanto el átomo como la cavidad, sufren "pérdidas" en forma de tasas de decaimiento, γ para el átomo y κ para el resonador, esto se debe a que el sistema no está completamente aislado. Por otra parte, experimentalmente se tiene una colección de N espines, y no un único átomo, lo que implica tener un sumatorio para los términos atómico y de interacción:

$$H_{JC} = \sum_{n=1}^{N}\frac{1}{2}\hbar\Delta E\sigma_n^z + \hbar\Omega a^\dagger a + \sum_{n=1}^{N}\hbar g_n(\sigma_n^+ a + \sigma_n^- a^\dagger) \tag{2.29}$$

Además, en el resonador, el campo EM es guiado por una excitación coherente de amplitud α, que tiene una tasa de decaimiento $\sim \kappa$, la cual nos añade un término en el hamiltoniano:

$$H_{guiado} \approx \hbar\alpha\kappa(ae^{i\omega t} + a^\dagger e^{-i\omega t}) \tag{2.30}$$

Vamos a escribir las ecuaciones del movimiento para este sistema, para ello, haremos uso de la *representación de Schrödinger* para los operadores $\langle a\rangle$ y $\langle\sigma^-\rangle$.

$$\frac{d}{dt}\langle a\rangle = -\frac{i}{\hbar}\langle[a, H_{JC} + H_{guiado}]\rangle \tag{2.31}$$

$$\frac{d}{dt}\langle\sigma^-\rangle = -\frac{i}{\hbar}\langle[\sigma^-, H_{JC}]\rangle \tag{2.32}$$

El término de guiado, solo se añade en el hamiltoniano para el operador a, porque es el que hace referencia a la cavidad, y el guiado se realiza solo al campo EM.

Como se ha mencionado antes, el átomo y la cavidad tienen un término de decaimiento, luego al final de cada ecuación se añade un término proporcional a este.

De esta forma, resolviendo (2.31) y (2.32), y añadiendo las pérdidas, llegamos a:

$$\frac{d}{dt}\langle a\rangle = -i\Omega\langle a\rangle - i\sum_{n=1}^{N} g_n\langle\sigma_n^-\rangle - i\alpha\kappa e^{-i\omega t} - \kappa\langle a\rangle \tag{2.33}$$

$$\frac{d}{dt}\langle\sigma_n^-\rangle = -i\Delta E\langle\sigma_n^-\rangle + ig_n\langle a\,\sigma_n^z\rangle - \gamma\langle\sigma_n^-\rangle \tag{2.34}$$

Para resolver este sistema de ecuaciones, suponemos soluciones del tipo,

$$\begin{cases}\langle a\rangle(t) = \langle a\rangle(0)e^{-i\omega t}\\ \langle\sigma_n^-\rangle(t) = \langle\sigma_n^-\rangle(0)e^{-i\omega t}\end{cases} \tag{2.35}$$

Y si usamos la aproximación para el límite semi-clásico de grandes colecciones de espines ($N\to\infty$), podemos olvidarnos de las correlaciones entre el campo del resonador y los espines individuales ($\langle a\,\sigma_n^z\rangle \approx \langle a\rangle\langle\sigma_n^z\rangle$) [1, Capítulo 4.6]. Llegamos a que,

$$0 = -i(\Omega-\omega)\langle a\rangle - i\sum_{n=1}^{N} g_n\langle\sigma_n^-\rangle - i\alpha\kappa - \kappa\langle a\rangle \tag{2.36}$$

$$0 = -i(\Delta E-\omega)\langle\sigma_n^-\rangle + ig_n\langle a\rangle\langle\sigma_n^z\rangle - \gamma\langle\sigma_n^-\rangle \tag{2.37}$$

Por último, asumimos que la diferencia de población ($(\Delta P)_e$) equivale a, $\langle\sigma_n^z\rangle = -(\Delta P)_e$ (para tiempos suficientemente altos, es el estado de equilibrio) [1, Capítulo 4.6]. Y también, vamos a unificar la colección de espines como,

$$\langle\sigma^-\rangle = \frac{1}{g_N}\sum_{n=1}^{N} g_n\langle\sigma_n^-\rangle \tag{2.38}$$

con $g_N = \sqrt{\sum_{n=1}^{N}|g_n|^2}$.

La función que a nosotros nos va a interesar para este trabajo, es la función de transmisión, puesto que es la que se mide en el laboratorio. Mediante la teoría de *input output* [4], esta función se obtiene como:

$$t(\omega) = -\frac{\langle a\rangle}{\alpha} \tag{2.39}$$

Una vez hechas las aproximaciones y resolviendo (2.36) y (2.37), se obtiene finalmente

$$t(\omega) = \frac{-\kappa}{(\Omega-\omega) - i\kappa - i\frac{g_N^2(T)}{i(\Delta E-\omega)+\gamma}} \tag{2.40}$$

donde hemos definido un acoplo dependiente de la temperatura,

$$g_N^2(T) := g_N^2(\Delta P)_e \tag{2.41}$$

3. Redes neuronales

Una vez sabemos cuál es nuestro sistema físico, el objetivo de este trabajo es obtener los parámetros $\kappa, \Delta E, \gamma$ y g de una curva de transmisión $t(\omega)$[2] (*Ec.* 2.40), es decir, queremos hacer un proceso de regresión a varios parámetros de una función.

Para llevar a cabo esto, vamos a hacer uso de las redes neuronales, las cuales son modelos de inteligencia artificial, pertenecientes a la categoría de aprendizaje automático (*machine learning*).

3.1. Unidad básica: la neurona

La unidad básica de procesamiento de una red neuronal, es la neurona. Tiene este nombre por estar basada en las conexiones sinápticas de las neuronas de los organismos biológicos [5].

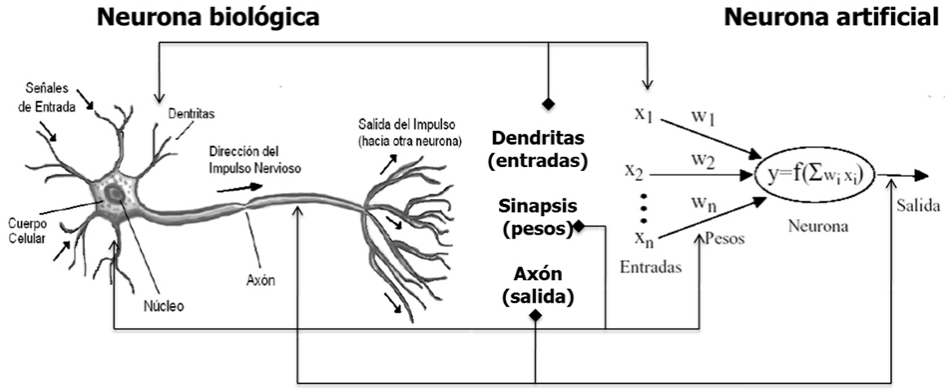


Figura 3.1: Comparativa entre una neurona real y una artificial.

Cada una de estas neuronas, recibe unos parámetros de entrada o *input* ($\vec{x}$), realiza unos procesos matemáticos, y devuelve unos valores de salida o *output*. Cada uno de los parámetros de entrada lleva asociado un valor de "importancia", el cual se conoce como peso ($\vec{w}$). Como añadido, a cada neurona se le incluye un parámetro extra llamado *bias* o sesgo (b), el cual actúa como una corrección que permite manejar mejor distintas situaciones. Al final, la neurona realiza la siguiente operación matemática,

$$z = \sum_i x_i \cdot w_i + b \tag{3.1}$$

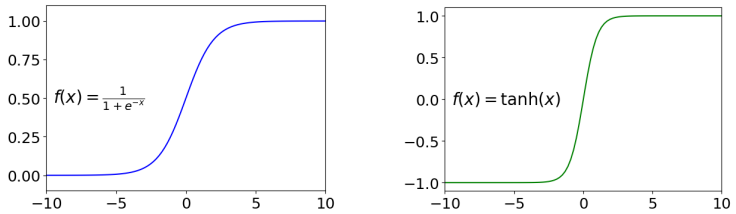
3.2. Función de activación

Una vez realizada esta operación, antes de devolver un resultado, la neurona pasa el valor de salida a una función de activación. Estas funciones de activación tienen el propósito de introducir no linealidad en el modelo, permitiendo al modelo aprender mayor variedad de problemas.

[2]Como no es cómodo trabajar con funciones complejas, se usará el módulo $|t(\omega)|$, aunque siempre se hablará de la función $t(\omega)$ y no mencionaremos la palabra *módulo*.

Las principales funciones de activación son [5-9]:

- **Sigmoide**: $f(x) = \frac{1}{1+e^{-x}}$. Devuelve un valor en el rango $(0, 1)$, suele ser útil para ejercicios de clasificación que devuelven probabilidades.
- **Tangente hiperbólica**: $f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$. Es similar a la sigmoide, pero tiene el beneficio de estar centrada en 0.
- **Rectified Linear Unit (ReLU)**: $f(x) = \text{máx}(0, x)$. Es la más popular actualmente, ya que reduce el tiempo computacional y permite "apagar" neuronas para ciertas entradas.
- **Softmax**: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$. Se usa para modelos que requieran clasificar más de dos cosas.



(a) Sigmoide (b) Tanh

(c) ReLU (d) Softmax

Figura 3.2: Algunas funciones de activación

3.3. Redes neuronales densas

Como una única neurona no es de mucha utilidad, se necesita un conjunto de neuronas para formar una red y poder realizar trabajos complejos.

Las redes neuronales se crean siguiendo un modelo secuencial que consta de tres capas principales: la capa de entrada, las capas ocultas[3] y la capa de salida. Todas las neuronas de una capa están enlazadas con cada una de las neuronas de la capa que le precede (excepto la capa de entrada, ya que no tiene una capa previa a ella), formando así una red densa. Podemos ver un esquema general de una red neuronal en la Fig. 3.3.

[3]Las redes con más de una capa oculta se conocen como redes profundas (*Deep Learning*).

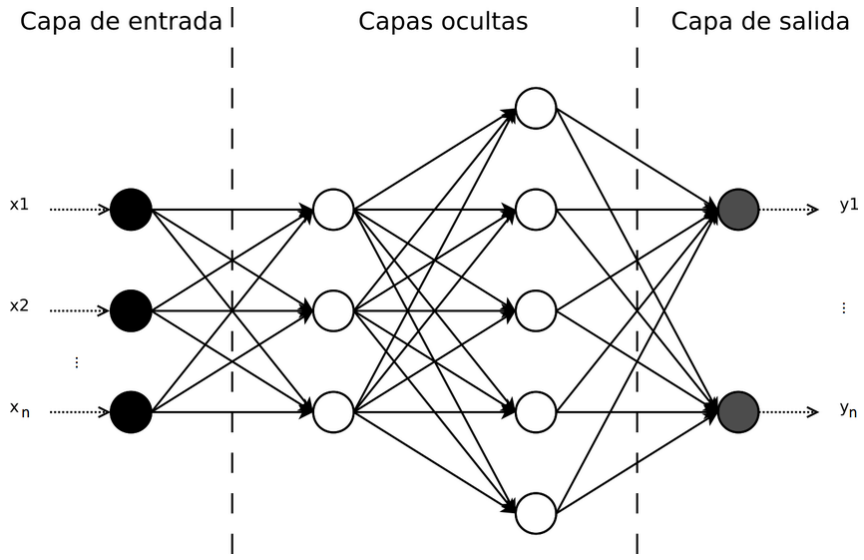


Figura 3.3: Esquema general de una red neuronal densa.

Para entrenar a la red, se le dan unos datos de entrenamiento como entrada y realiza una propagación hacia adelante. Cuando los datos llegan a la capa de salida, se calcula la **función de pérdida**, que cuantifica qué tan mal está haciendo la red sus predicciones comparando los valores obtenidos con el valor real que se debería obtener. La finalidad de la red es minimizar esta función de pérdidas al máximo, y para ello, se hace uso del algoritmo de *backpropagation*, el cual calcula el gradiente de la pérdida haciendo uso de la regla de la cadena, y lo propaga hacia atrás hasta las capas de entrada para actualizar e ir ajustando los parámetros.

3.4. Hiperparámetros

Todo este proceso de propagarse hacia delante, calcular la función de pérdida, hacer el *backpropagation* y actualizar los parámetros, depende de varios componentes conocidos como **hiperparámetros**, los cuales deben ser ajustados y seleccionados los más adecuadamente posible, para optimizar lo máximo el proceso de aprendizaje.

3.4.1. Pérdidas/Coste

La primera variable a elegir es la ya mencionada función de pérdida. El término coste también suele ser utilizado para hablar de la función de pérdida, pero técnicamente, la pérdida es el resultado de una instancia, y el coste es la pérdida a lo largo de todo el conjunto de datos.

Principalmente se usan estas cuatro funciones como pérdidas [5-9]:

- **Error cuadrático medio (MSE)**:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  donde y_i es el valor real, $\hat{y}_i$ es la predicción, y n es el número total de muestras.

- **Error absoluto medio (MAE)**:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Estas dos son las más utilizadas en problemas de regresión, ya que evalúan directamente la diferencia entre el valor real y la predicción. Aunque de las dos, la favorita suele ser el MSE.

- **Entropía Cruzada Binaria (Binary Cross-Entropy)**:

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

- **Entropía Cruzada Categórica (Categorical Cross-Entropy)**:

$$CCE = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

Ambas son típicamente usadas en problemas de clasificación, la primera cuando la clasificación es binaria, y la segunda cuando hay más de dos opciones.

3.4.2. Optimizador

Lo siguiente que se debe elegir es el optimizador, ya que es quien va a seleccionar el método utilizado para actualizar los parámetros tras su paso por el *backpropagation*. Los principales optimizadores son [5, 7, 9]:

- **SGD (Descenso Estocástico de Gradiente)**: Es el optimizador más básico, en él los parámetros de la red se actualizan en la dirección opuesta al gradiente de la pérdida.
- **AdaGrad (Algoritmo de Gradiente Adaptativo)**: Ajusta la tasa de aprendizaje para cada parámetro de forma independiente, en función de la magnitud de los gradientes previos. Es útil para problemas con datos atípicos, pero puede tener una desventaja al acumular gradientes que reducen demasiado la tasa de aprendizaje a lo largo del tiempo.
- **RMSProp (Propagación de la Raíz Cuadrada Media)**: Es una mejora sobre AdaGrad. Mantiene una media móvil exponencial de los cuadrados de los gradientes anteriores y ajusta la tasa de aprendizaje para cada parámetro de acuerdo con esa media. Esto ayuda a resolver el problema de AdaGrad, haciéndolo más adecuado para problemas no convexos.
- **Adam (Estimación de Momento Adaptativa)**: Combina las ideas de AdaGrad y RMSProp al utilizar tanto la media móvil de los gradientes, como la media móvil de los cuadrados de los gradientes. Adam ajusta la tasa de aprendizaje de cada parámetro de manera adaptativa y es conocido por ser robusto, eficiente y funcionar bien en una amplia variedad de problemas.

Los dos más utilizados son el SGD para tareas de clasificación, y Adam para problemas de regresión.

3.4.3. Tamaño de lote y épocas

Por último, se deben ajustar dos parámetros, los cuales son muy importantes, porque afectan al tiempo total de entrenamiento, la eficiencia computacional, la estabilidad del aprendizaje y la capacidad del modelo para generalizar a nuevos datos.

El tamaño de lote o *batch size* es el número de muestras procesadas en una única iteración durante el entrenamiento.

Y las épocas o epochs[4] se refieren al número de veces que todo el conjunto de datos de entrenamiento se va a procesar completamente. Más epochs permiten que el modelo aprenda mejor los patrones de los datos, pero un número excesivo puede causar sobreajuste (overfitting).

3.5. El problema del *overfitting* y *underfitting*

Cuando se entrena una red neuronal, si no se hace una buena elección de los hiperparámetros, pueden surgir dos problemas [5-10]:

- *Overfitting* (sobreentrenamiento/sobreajuste[5]). Se trata del caso en el que el modelo "memoriza" los datos de entrenamiento, y por ello, no es capaz de generalizar a nuevos datos.
- *Underfitting* (subajuste). Es el caso opuesto al anterior, se trata de un problema que implica que la red no haya tenido el tiempo suficiente para entrenar y, por lo tanto, no ha aprendido todavía a optimizar los parámetros. Suele ocurrir por la elección de un número de épocas o un set de entrenamiento pequeños.

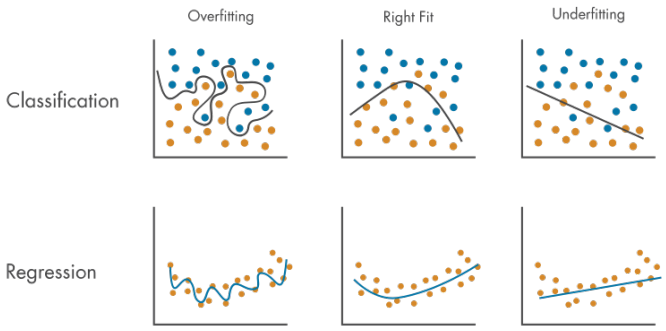


Figura 3.4: Ejemplo práctico de overfitting y underfitting, comparado con el caso óptimo

Para crear nuestra red neuronal haremos uso de las librerías TensorFlow y Keras, de Python, para evitar complejidades innecesarias, ya que ambos han sido desarrollados por profesionales y tienen un funcionamiento de lo más óptimo [6, 7, 9].

[4] A lo largo del trabajo se usarán los nombres en inglés, batch size y epochs, por mantener la nomenclatura típica.
[5] La traducción literal es más bien sobreajuste, pero como es un problema dado al entrenar una red, es más adecuado el uso de sobreentrenamiento. A lo largo del trabajo se usará la notación en inglés.

4. Creación de la red

Conocido el sistema físico, el funcionamiento de las redes neuronales y nuestro objetivo, es hora de entrenar modelos basados en redes neuronales.

4.1. Estudio previo

El primer paso antes de crear un modelo y entrenarlo, es hacer un estudio de todas las variables que podemos ajustar, de manera que nuestra red neuronal sea la más óptima para el problema que deseamos tratar. Los pasos a seguir son los siguientes:

- Lo primero que debemos elegir es el tamaño de la base de datos que será usada para entrenar a la red y de éstos datos, qué porcentaje serán datos de entrenamiento y cuál serán datos de validación y test. Este apartado es muy importante ya que como hemos mencionado en 3.5, la red se sobreajustará y no sabrá generalizar a nuevos datos.
- Una vez creada la base de datos, debemos escalar estos datos, para que todos estén en los mismos rangos y así la red pueda aprender más fácilmente, por lo que debemos elegir un escalado. Los escalados más comunes son[11]:
  - *StandardScaler*: elimina la media y escala unitarimente respecto de la varianza, es decir, re-escala los datos siguiendo una función normal centrada en 0 con una desviación estándard de 1. Muy útil para datos que se distribuyen de forma Gaussiana.
  - *MinMaxScaler*: escala los datos para que estén en un rango deseado. Responde bien para datos con una desviación estándar pequeña.
  - *MaxAbsScaler*: muy similar al anterior, escala los datos según su valor absoluto máximo, por lo que quedan en el rango $[0, 1]$.
  - *RobustScaler*: es un escalador diseñado para no ser afectado por valores atípicos (de ahí el nombre *Robust*). Elimina la mediana y escala en el rango intercuartílico, el cual se basa en percentiles.
- Para finalizar, debemos ajustar los hiperparámetros:
  - Lo primero que debemos estudiar es cuántas capas ocultas debe tener nuestra red, y cuántas neuronas habrá en cada una de las capas. También se deberá elegir una función de activación para las neuronas.
  - Los siguientes hiperparámetros a elegir son aquellos que se explican en 3.4.2 y 3.4.3, que son: el optimizador, el *batch size* y las *epochs*.
  - Por último, elegiremos una función de coste/pérdidas y, opcionalmente[6], una métrica para valorar el funcionamiento de la red.

[6]En la regresión es un parámetro opcional, ya que la propia función coste nos indica la eficiencia del modelo. Por el contrario en problemas de clasificación se añade como métrica la precisión (*accuracy*) para valorar mejor la eficiencia.

Una vez conocidas todas las variables y parámetros que debemos ajustar, explicamos los métodos utilizados para determinar cada uno, y la elección tomada.[7]

Lo primero que se escogió fue la división de los datos, donde escogimos una de las particiones más comunes [6] que es: 80 % de datos de entrenamiento, 10 % de datos de validación y el otro 10 %, datos de test.

Para continuar, elegimos como optimizador Adam, como función de activación ReLU, y la función de pérdidas y la métrica elegidas fueron el MSE y MAE, respectivamente.

Conocido esto, creamos un modelo básico de 1 capa oculta de 256 neuronas, con *batch size* = 100 y *epochs* = 1000, y lo entrenamos para distintos valores de tamaño de la base de datos, donde obtuvimos que el mejor tamaño era de 10000, ya que nos daba la mejor relación entre pérdidas y tiempo de entrenamiento.

Una vez seleccionado el tamaño de la base de datos, tocaba elegir el número de capas ocultas y la cantidad de neuronas en cada una de ellas. Como a la salida tenemos 4 neuronas correspondientes a los parámetros a predecir, estimamos usar múltiplos de este número $(16, 32, 64, 126, 256)$, y para el número de capas ocultas llegamos a estudiar hasta un máximo. Para las redes con más de 1 capa oculta, se eligieron combinaciones de números de neuronas del grupo anterior, de forma descendente, es decir, el número de neuronas en la capa siguiente debe ser inferior al de la capa que le precede. De esta forma obtuvimos que la mejor red era la de 5 capas ocultas con $16, 32, 64, 126$ y 256 neuronas, respectivamente.

Y, para finalizar, se hizo una búsqueda en malla de la mejor combinación de *batch size* y *epochs*, donde se probaron todas las combinaciones de, *batch size* = $\{10, 25, 50, 100, 250, 500, 1000\}$ y *epochs* = $\{10, 50, 100, 250, 500, 1000, 2500, 5000, 10000\}$. El resultado obtenido fue la combinación, *batch size* = 250, *epochs* = 1000, que devolvía unas pérdidas pequeñas para un coste computacional pequeño.

La siguiente tabla muestra un resumen de las variables e hiperparámetros elegidos:

Tamaño base de datos	10000
Datos de entrenamiento (80 %)	8000
Datos de validación (10 %)	1000
Datos de test (10 %)	1000
Escalado	RobustScaler
Número de capas ocultas	5
Neuronas en cada capa	256, 128, 64, 32, 16

Función de activación	ReLU
Batch size	250
Epochs	1000
Coste/pérdidas	MSE
Métrica	MAE
Optimizador	Adam

Tabla 4.1: Variables e hiperparámetros seleccionados en nuestro modelo.

[7] Si se desea ver un análisis completo con gráficas, puede verse en Anexo B.

4.2. Entrenamiento del modelo

Para enseñar a nuestro modelo, generaremos una base de datos sintética, que constará de distintas combinaciones, generadas aleatoriamente, de los parámetros $\kappa, \Omega, \Delta E, \gamma, g$ de la función $t(\omega)$. Los rangos en los que se generan son los siguientes:

- $\kappa, \gamma,\ g \in \left[10^{-4}, 10^{-2}\right]$
- $\Delta E \in [0.5, 1.5]$
- $\Omega = 1$ (es una constante, ya que su valor será conocido en el laboratorio)

La función $t(\omega)$ se generará con 300 cortes (valores de ω) distribuidos entre 0.85 y 1.15. Tanto ω como los parámetros, están en unidades adimensionales en su entrenamiento.

Se han escogido esos valores de los parámetros intentando tener un modelo lo más genérico, que incluya el mayor número de combinaciones posibles, y trabajando mayormente en acoplo débil, puesto que es más difícil de medir experimentalmente. Y se ha elegido discretizar ω en 300 puntos, pensando que en el laboratorio cada medida requiere mucho tiempo y se hará un número limitado de cortes.

4.2.1. Datos sin ruido

Comenzaremos entrenando la red con las curvas originales, es decir, tal cual se obtienen al sustituir los parámetros en (2.40).

Para valorar la precisión en la predicción, vemos en la Fig. 4.1 cómo ha predicho la red los distintos parámetros. Es bastante apreciable que no tiene ningún problema con κ, y que g lo predice bastante bien. Por otra parte, ΔE lo predice muy bien en los valores centrales, pero fatal en los bordes, esto es porque como trabajamos en un rango de ω más corto que el de ΔE, la red no puede ver el comportamiento de la función fuera de los límites en los que se trabaja. En cuanto a γ, parece que no es capaz de predecirlo, esto se debe a que es un parámetro que afecta muy ligeramente en la forma de la función, por lo que la red no tiene suficiente información para poder predecir este parámetro.

A continuación, en la Fig. 4.2, podemos ver algunos ejemplos de curvas originales frente a las obtenidas por el modelo. Se puede ver que las curvas tienen una forma de tipo *Lorentziana* y que, en general, cuando solo se presenta un pico, la red predice perfectamente la curva. Sin embargo, cuando empiezan a aparecer deformaciones (conocidas como *fanos*), la red parece fallar ligeramente al predecir la altura en la que se encuentra el pequeño pico.

Por otra parte, cuando tenemos curvas en resonancia, esto es $\Delta E \approx \Omega$, la red es muy imprecisa y rara vez logra seguir perfectamente la forma de doble pico, esto se justifica con que alrededor de la resonancia cualquier pequeña variación en ΔE cambia mucho la forma de la curva, esto hace que aunque la predicción de la red sea muy buena, visualmente parece que haya errado mucho en su predicción.

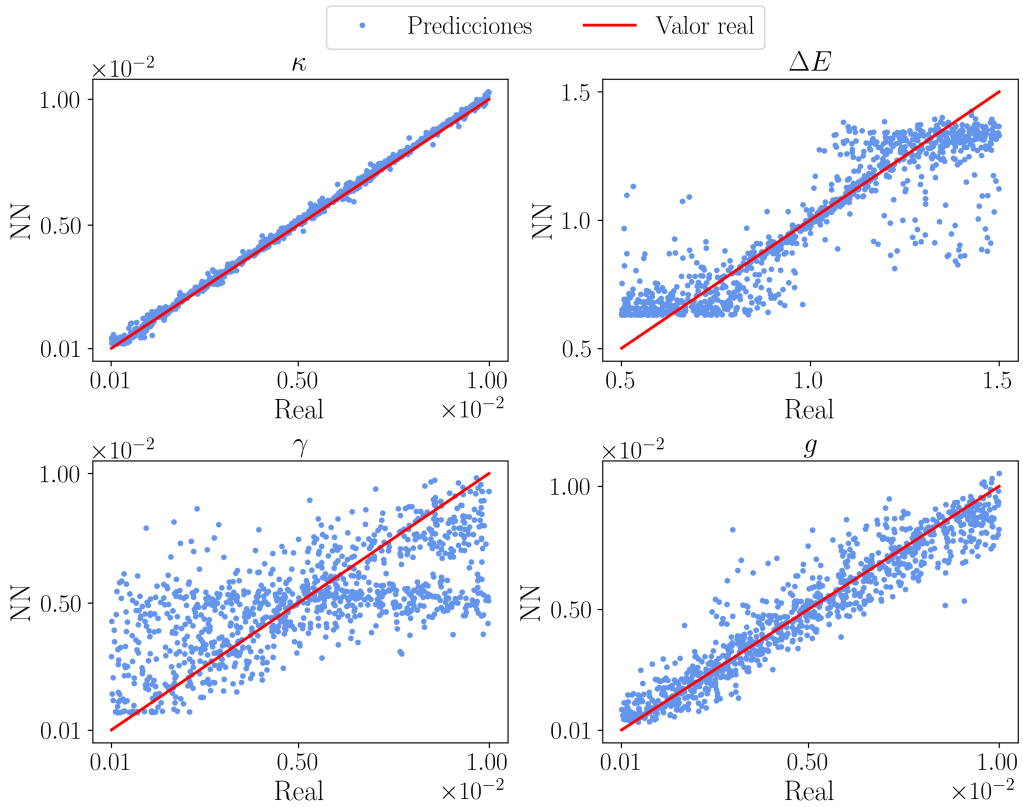


Figura 4.1: Valor real frente a predicción de la red para los 4 parámetros a predecir. La línea roja indica la predicción perfecta y los puntos azules las predicciones de la red.

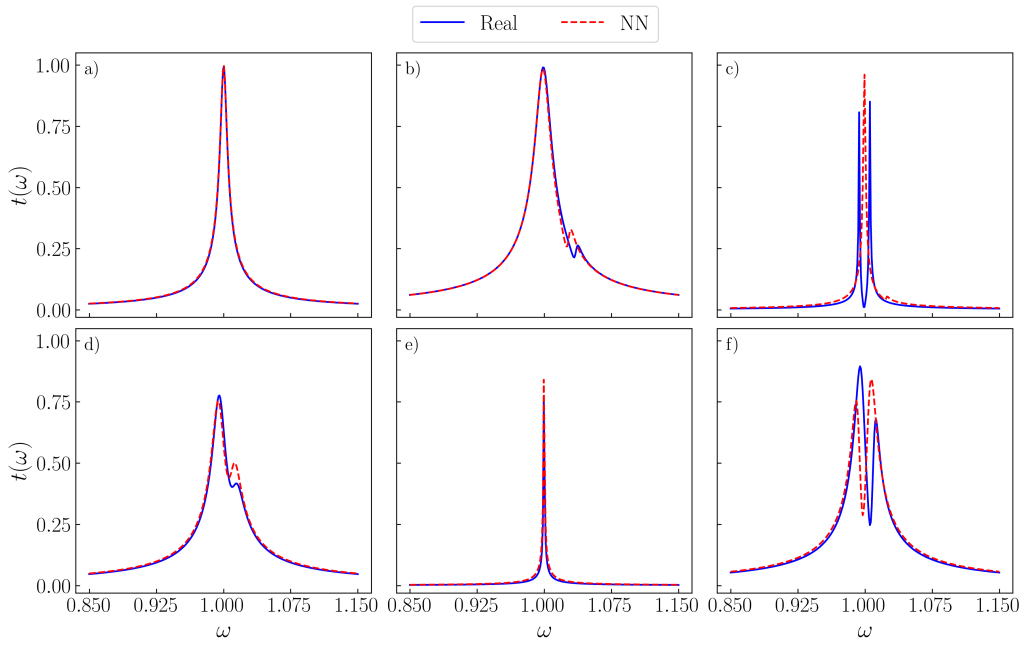


Figura 4.2: Distintos ejemplos de curvas $t(\omega)$ reales obtenidas con los parámetros originales (azul), frente a las curvas obtenidas al sustituir los parámetros predichos por la red (rojo).

4.2.2. Datos con ruido

Sabemos que en el laboratorio las cosas no son perfectas y, por ello, las medidas serán distorsionadas mediante ruido de fondo y de los propios aparatos de medida. Por ello, vamos a añadir ruido a las curvas para que el modelo aprenda de medidas más realistas.

El ruido va a ser añadido generando una distribución gaussiana cuya amplitud (σ) es un número aleatorio entre 0 y el 25 % del valor de la función t(ω) para cada ω.

Al igual que en el apartado anterior, evaluamos la precisión del modelo viendo la predicción de cada parámetro frente al valor real (Fig. 4.3). Si comparamos las predicciones obtenidas con el ruido respecto de las que no lo tenían, vemos claramente que, excepto κ que sigue siendo predicha a la perfección y γ que siguen sin predecirse, ΔE y g han perdido notablemente la precisión que tenían al ser predichas. Este comportamiento es algo que cabe esperar, puesto que estamos variando la forma de la curva, haciendo que la red tenga más dificultades para predecir adecuadamente.

Por último, en la Fig. 4.4 vemos algunas curvas y sus predicciones, y curiosamente, hemos percibido mayor precisión en la forma de las curvas a pesar de que los parámetros se predigan peor. Posiblemente, este comportamiento se deba a que la red ha aprendido a seguir la forma de la curva de forma certera, pero como varias combinaciones de parámetros pueden dar curas prácticamente iguales, esto habrá generado los errores en la predicción de parámetros.

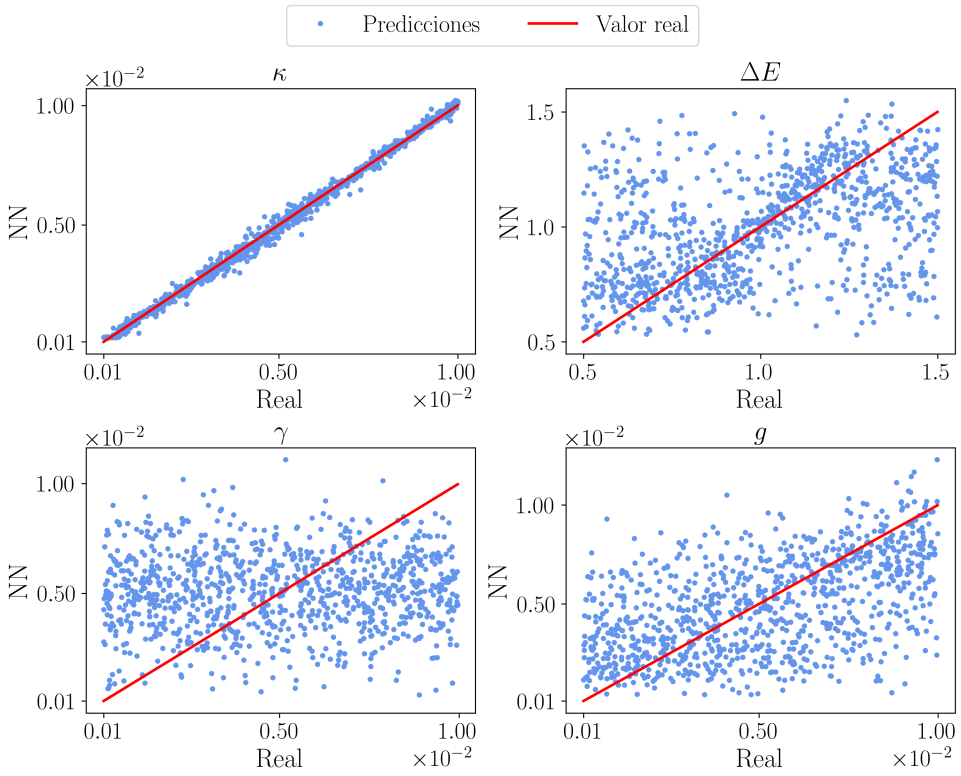


Figura 4.3: Predicción de cada parámetro (puntos azules), frente al valor real de cada uno (línea roja), para los datos con ruido.

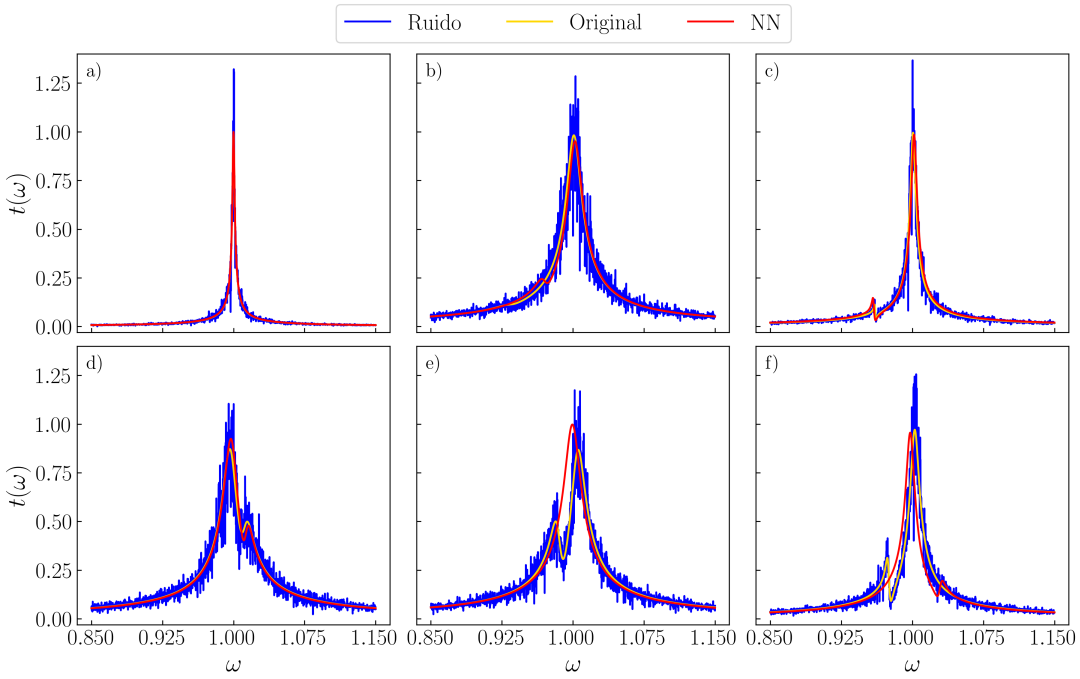


Figura 4.4: Curvas originales (amarillo), curva con ruido (azul) y la predicción del modelo (rojo) para los datos con ruido.

5. Resultados con datos experimentales

Una vez hemos entrenado entrenado nuestra red neuronal y hemos visto su comportamiento en distintas situaciones, es hora de recibir los datos experimentales y así poder evaluar la calidad del modelo.

Los datos experimentales han sido aportados del *LER 5* de la tesis doctoral de Marcos Rubín [1, Capítulo 5.3]. Estos datos contienen 41 valores del campo magnético (entre 65 y 85 mT), cada uno con 2500 cortes en frecuencia (entre 2.09 y 2.14 GHz). Véase el *contour plot* de la Fig. 5.1 para una vista general.

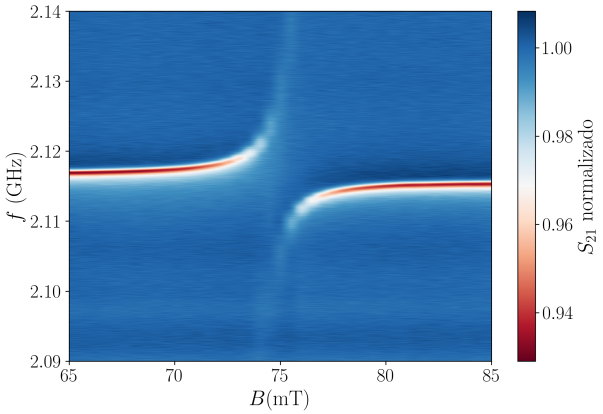


Figura 5.1: *Contour plot* de los datos experimentales, donde se representan las frecuencias medidas para cada valor del campo magnético y su correspondiente medida de S_{21}.

La función medida difiere ligeramente de (2.40), ya que la función usada es,

$$S_{21}(\omega) = 1 - \frac{\kappa_c}{i(\Omega - \omega) + \kappa + \frac{g_N^2(T)}{i(\Delta E - \omega) + \gamma}} \tag{5.1}$$

donde se han separado las pérdidas internas (κ_i) y las pérdidas propias del resonador (κ_c), dando una tasa de decaimiento total, $\kappa := \kappa_i + \kappa_c$. Por lo que en las ecuaciones del movimiento, el término de guiado es proporcional a κ_c.

Además, como conocemos el valor del campo magnético, usando (5.2)[8] tenemos el valor de ΔE, lo que nos elimina una variable a ser predicha.

$$\Delta E = \frac{\mu_B g_s B}{h} \tag{5.2}$$

donde μ_B es el magnetón de Bohr ($\mu_B = 9.274\, JT^{-1}$), g_s es el fator g obtenido en [1] ($g_s = 2.01928$), B el campo magnético, y h la constante de Planck.

Teniendo en cuenta estos cambios, volvimos a crear y entrenar nuestro modelo. Siguiendo los valores obtenidos experimentalmente en [1] y manteniendo la forma de nuestros rangos genéricos, se eligió un rango de valores para la nueva variable, $\kappa_c \in [10^{-6}, 10^{-4}]$.

Tras crear los nuevos modelos adaptados a los datos experimentales[9], veamos qué tal predicen los valores experimentales.

5.1. Modelo entrenado sin ruido

Comenzamos viendo los resultados de nuestra red entrenada con rangos genéricos[10] de los parámetros.

Si observamos la Fig. 5.2, vemos que la red es totalmente incapaz de predecir la curva, alejándose totalmente de la forma experimental.

Para intentar mejorar estos resultados, se entrenó un nuevo modelo, pero esta vez con unos parámetros acotados en un entorno cercano de los resultados obtenidos experimentalmente por [1, Capítulo 5.3 & Apéndice A] en la resonancia. Los rangos usados son los siguientes[11]: $\kappa \in [10^{-4}, 10^{-3}]$, $\kappa_c \in [10^{-5}, 10^{-4}]$, $\gamma \in [5 \cdot 10^{-3}, 8 \cdot 10^{-3}]$, $g \in [10^{-2}, 1.5 \cdot 10^{-2}]$.
También tenemos el valor de la frecuencia del resonador, $\Omega = 2.116451\, GHz$.

Mirando ahora la Fig. 5.3, vemos que las predicciones son significativamente mejores que antes, lo que es un gran avance.

[8] Usamos h en lugar de $\hbar$ para obtener directamente la frecuencia de ΔE en Hz.
[9] Gráficas de la predicción de parámetros y algunas curvas en C.3.
[10] Los rangos se han mencionado en 4.2
[11] A diferencia del rango general, aquí sí tenemos unidades y todo está en GHz.

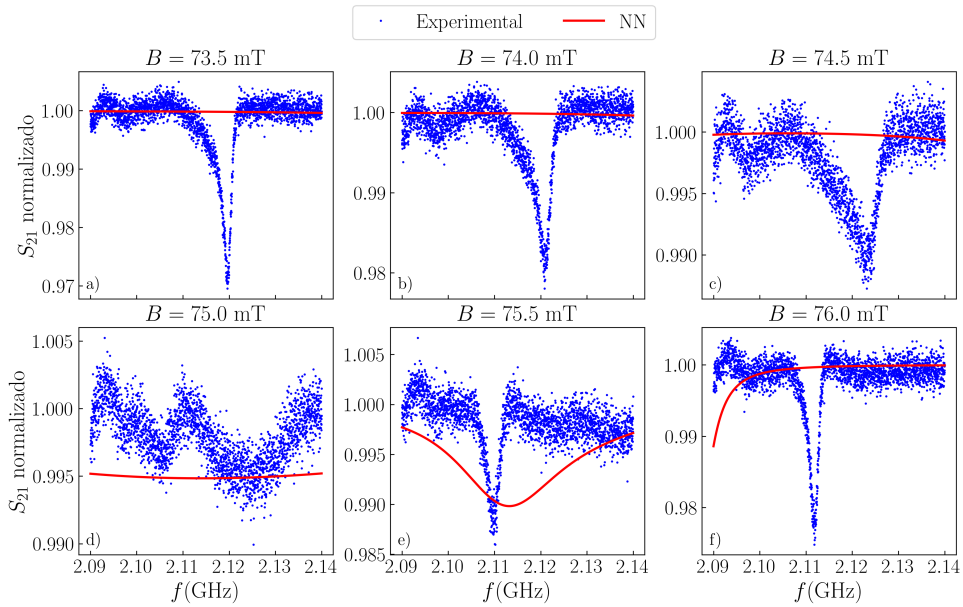


Figura 5.2: Predicción de curvas experimentales con modelo sin ruido entrenado en un rango genérico.

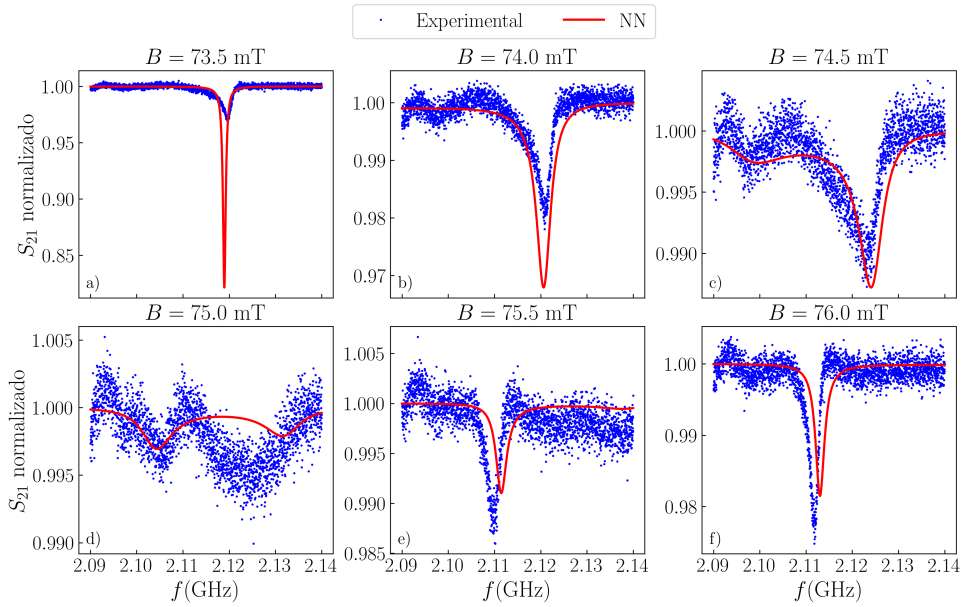


Figura 5.3: Predicción de curvas experimentales con modelo sin ruido para un modelo acotado.

Como queremos mejorar la precisión del modelo al máximo, aprovechamos que en [1] nos dan el valor de γ, y se nos indica que es una constante del material ($\gamma = 6..5 \pm 0.5\, MHz$), por lo que podemos fijarlo y así reducimos la complejidad eliminando una variable a predecir.

Con este cambio, al entrenar una red con los rangos genéricos de los parámetros, los resultados no mejoraban prácticamente, así que pasamos directamente a entrenar un nuevo modelo acotado. En la Fig. 5.4 tenemos los resultados para este nuevo modelo, y curiosamente, ha empeorado ligeramente la predicción a la hora de seguir las curvas (veremos en 5.3 que la predicción de parámetros también es mejor sin fijar γ).

Para terminar, quisimos acotar aún más los parámetros y ver si la red podía ser todavía más precisa (aunque se podría considerar "hacer trampas", ya que estamos dándole prácticamente los resultados al modelo). Los nuevos rangos de los parámetros serán: $\kappa \in [2 \cdot 10^{-4}, 6.5 \cdot 10^{-3}]$, $\kappa_c \in [1.5 \cdot 10^{-5}, 3.5 \cdot 10^{-5}]$, $g \in [1.15 \cdot 10^{-2}, 1.35 \cdot 10^{-2}]$.

Los resultados de este segundo acotamiento los podemos ver en Fig. 5.5, y como cabía esperar, los resultados han mejorado bastante, pero ni de esta forma logra predecir al 100 % la forma de la curva en todos los casos, donde más precisión tiene es en las dos curvas más cercanas a la resonancia.

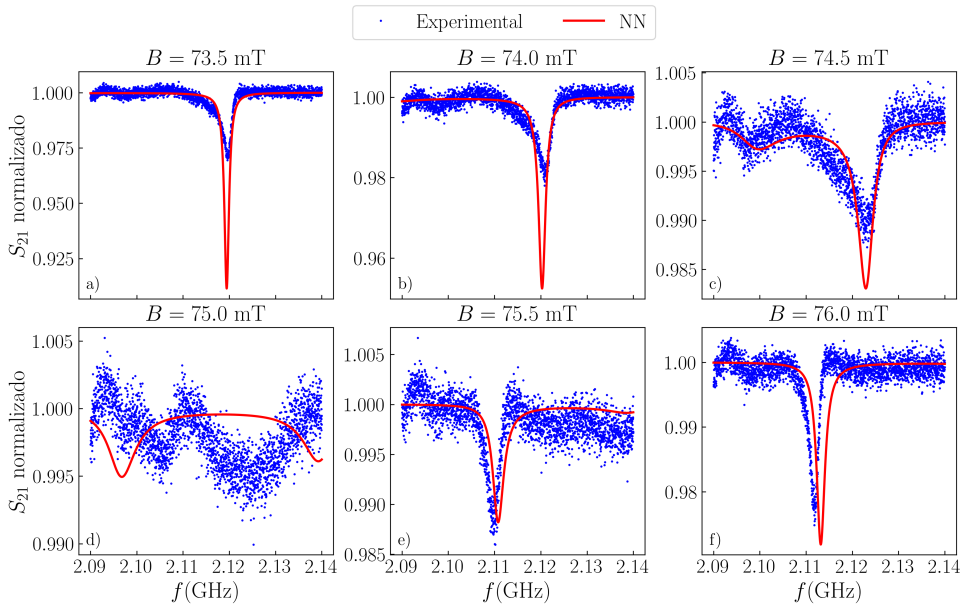


Figura 5.4: Predicción de curvas experimentales fijando γ, en el modelo sin ruido acotado

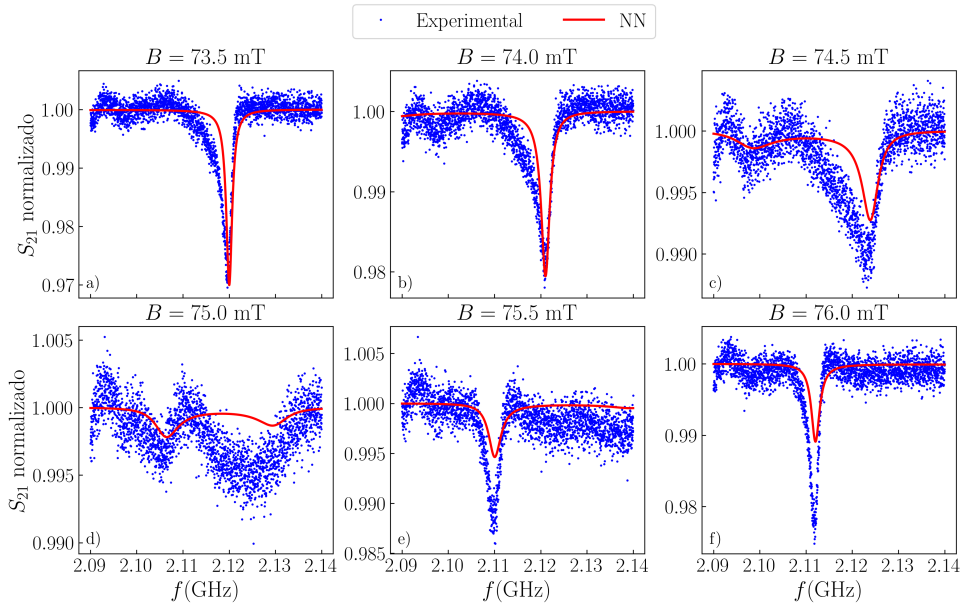


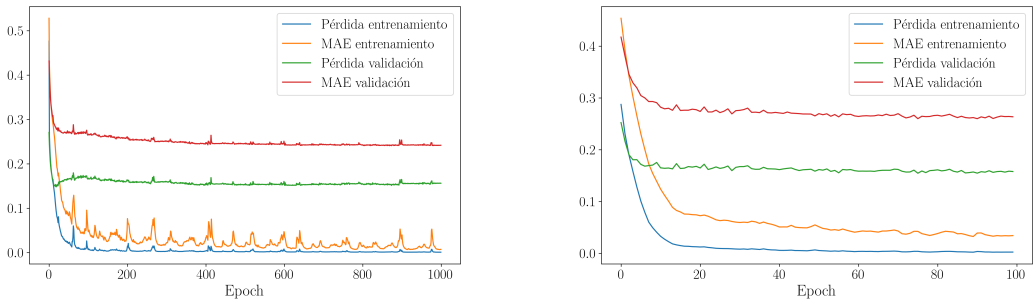
Figura 5.5: Predicción de curvas experimentales fijando γ, cuando realizamos un segundo acotamiento al modelo sin ruido.

5.2. Modelo con ruido

Lo primero que debemos hacer antes de entrenar a la red neuronal, es seleccionar el método para añadir ruido a los datos, ya que el método usado en 4.2.2 no es válido para añadir ruido a la función $S_{21}(\omega)$ (5.1).

Tras realizar varias pruebas y comparar con la forma de los datos experimentales, decidimos que el mejor método para añadir el ruido es añadir un número generado por una distribución gaussiana cuya amplitud (σ) depende de la posición del pico invertido: cuando el mínimo tiene un valor mayor a 0.975, tomaremos un número aleatorio entre 0 y el 7.5 % del valor del mínimo; si el mínimo se encuentra por debajo del valor anterior y por encima de 0.85, la amplitud se generará hasta el 5 % del valor del mínimo; y para el resto de curvas se usará un valor de hasta el 1 %.

Con esto hecho, ya podemos entrenar nuestro modelo[12]. Sin embargo, se observó algún problema con el entrenamiento, ya que el comportamineto de las curvas de aprendizaje (MSE vs Epochs) era irregular y mostraba un patrón que podría indicar *overfitting*. Así pues, hicimos un nuevo estudio de hiperparámetros, pero solo para la combinación (*batch size, epochs*), el cual nos dió como mejor combinación, ($batch\ size = 50$, $epochs = 100$).



(a) (b)

Figura 5.6: Pérdida y MAE en función de las epochs antes (a) y después (b) del nuevo estudio de hiperparámetros. Se puede ver cómo en la primera parece haber un claro problema en el entrenamiento, mientras que en la segunda todo es más plano y como suele ser la gráfica.

Los resultados obtenidos para la red entrenada con parámetros en el rango genérico, fueron ligeramente mejores que los obtenidos por el modelo entrenado sin ruido, algo medianamente esperable. Por el contrario, para el modelo con el primer acotamiento, aunque los resultados son muy buenos, son algo peores que los del modelo entrenado sin ruido. Y como era de esperar, el segundo acotamiento nos devuelve unas curvas ajustadas de forma excepcional.

Mostramos en Fig. 5.7 la gráfica para el modelo del primer acotamiento.

[12] Se entrenó directamente con γ fijo, puesto que brindaba mejores resultados y es un parámetro íntrinseco de la muestra.

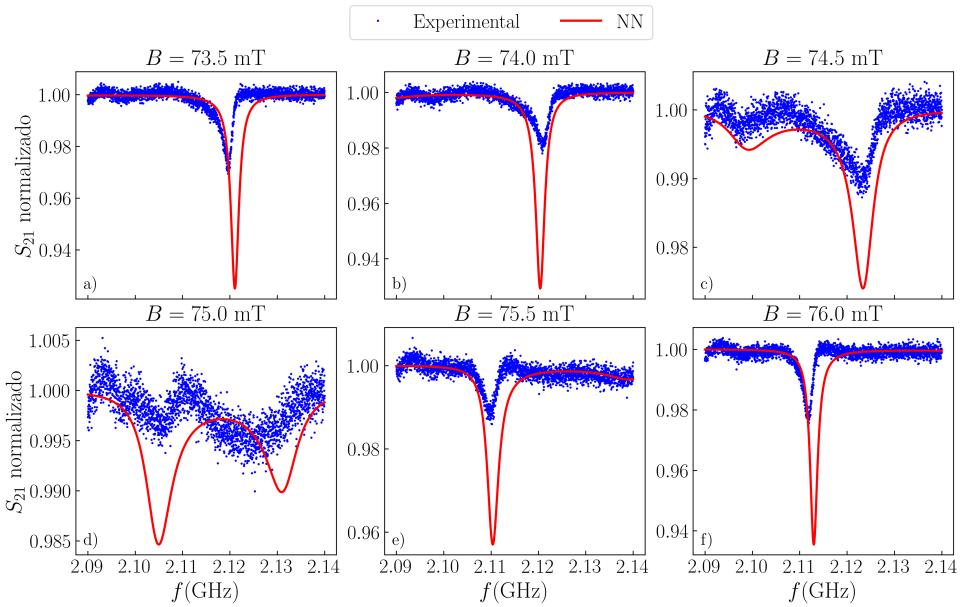


Figura 5.7: Predicción de curvas experimentales, con γ fijo, para el modelo entrenado con ruido gaussiano y con el primer acotamiento.

5.3. Resultados finales

Como el objetivo final de este trabajo es usar las Redes Neuronales para predecir parámetros, vamos a ver qué tal han predicho los parámetros cada modelo. Conociendo los resultados obtenidos de los ajustes experimentales en [1]:

$$\kappa = 415.5 \pm 0.8\,kHz,\ \kappa_c = 30.4 \pm 0.1\,kHz,\ g = 12.491 \pm 0.8\,MHz,\ \gamma = 6.5 \pm 0.5\,MHz$$

En la siguiente tabla (Tabla 5.1), se muestran nuestros resultados para cada uno de los modelos. En ella podemos ver que el modelo más acertado ha sido el que se ha entrenado con ruido y usando el segundo acotamiento, este modelo ha predicho los tres parámetros con diferencias relativas **inferiores al 3 %**, lo que es un resultado prácticamente perfecto. Además, el modelo sin fijar γ, entrenado sin ruido y con el primer acotamiento, también nos ha devuelto unos resultados muy satisfactorios, teniendo en cuenta que el rango de los parámetros es más amplio que el del segundo acotamiento, y que ha predicho 4 parámetros en lugar de 3.

	Modelo	κ (kHz)	κ_c (kHz)	g (MHz)	γ (MHz)
Sin ruido	General	10085.3 ± 10116.2	165.3 ± 119.9	38.362 ± 29.575	13.9 ± 19.6
	Acotado 1	439.38 ± 233.6	38.4 ± 19.8	11.264 ± 1.042	6.8 ± 1.5
	General (γ fijo)	6479.8 ± 4659.8	121.4 ± 91.8	28.507 ± 25.477	
	Acotado 1 (γ fijo)	565.4 ± 179.7	42.2 ± 16.4	11.533 ± 2.603	
	Acotado 2 (γ fijo)	336.1 ± 37.5	20.6 ± 6.6	12.523 ± 0.998	
Ruido	General (γ fijo)	7254.6 ± 5474.2	76.2 ± 17.0	6.378 ± 2.184	
	Acotado 1 (γ fijo)	571.4 ± 220.5	52.8 ± 18.3	11.954 ± 1.223	
	Acotado 2 (γ fijo)	412.6 ± 104.8	31.1 ± 12.3	12.400 ± 0.415	

Tabla 5.1: Resultados finales obtenidos por nuestros modelos de redes neuronales, usando el valor medio y la desviación estándar de cada parámetro.

6. Conclusiones

Tras la realización de este laborioso trabajo, podemos extraer diversas conclusiones.

Durante el entrenamiento de la red inicial (cuando ajustábamos a (2.40)), aprendimos que el parámetro γ no era fácil de predecir para la red porque al encontrarse en un denominador dominado por la resta $(\Delta E - \omega)$ en la mayoría de curvas, la relevancia de este quedaba en segundo plano y no afectaba apenas en la forma de la curva, imposibilitando su correcta predicción. La única excepción eran las curvas que estaban en resonancia o en sus inmediaciones (recordemos que esto es $\Delta E \approx \Omega$), esto hacía que la resta que antes dominaba el denominador quedase prácticamente anulada, permitiendo así al parámetro γ tomar parte importante de la forma de la función, pudiendo así ser predicho por el modelo.

Por otra parte, después de estudiar y evaluar los datos experimentales, debemos mencionar varios aspectos.

El primer aspecto ha sido la elección de los rangos genéricos en los que se crearon los parámetros sintéticos. Inicialmente, la idea era diseñar la red para aprender a predecir en casos de acoplamiento débil, que son los más complicados a la hora de obtener los parámetros, De esta forma, se escogió un rango de valores con el fin de estudiar este caso. Sin embargo, los datos que se nos proporcionaron eran del caso de acoplamiento fuerte, para el cual no estaba muy bien entrenado nuestro modelo.

De aquí extraemos la conclusión de que, si hubiéramos tenido a disposición un tiempo mayor y unos recursos computacionales superiores a los de un ordenador de sobremesa común, podríamos haber realizado un entrenamiento más profundo con una base de datos mucho más grande, teniendo unos rangos de parámetros mucho más amplios, permitiendo aumentar el número de casos cubiertos al entrenar a la red.

La siguiente conclusión que se extrae de este trabajo, es que si conoces un poco el sistema físico que se está midiendo y puedes acotar el rango en el que se mueven las variables que a predecir, se puede realizar un entrenamiento sencillo y con poco coste computacional, el cual trae resultados muy satisfactorios.

Otro aspecto descubierto, es que cada sistema físico es distinto y que las redes neuronales pueden ser muy curiosas. Decimos esto, ya que hicimos un segundo entrenamiento de la red con datos a los que se les introdujo ruido, para que se parecieran más a unos datos experimentales, y aún así, el modelo que habíamos entrenado sin ruido mostró mejores resultados que el entrenando con ruido, para el caso de las variables acotadas a los resultados de [1].

Por último, destacar que, en gran medida, se ha cumplido el objetivo de este trabajo, que era ver si mediante la creación y entrenamiento de un modelo basado en redes neuronales con datos sintéticos, podíamos hacer un proceso inverso de predicción de parámetros a partir de unas curvas experimentales.

Referencias

[1] Marcos Rubín Osanz. «Pump-probe experiments on superconducting resonators coupled to molecular spins». Tesis doctoral. Universidad de Zaragoza, 2024. URL: https://zaguan.unizar.es/record/136041/?ln=es.

[2] Wolfgang P. Schleich. *Quantum Optics in Phase Space*. 1st. Berlin, Germany: WILEY-VCH Verlag Berlin GmbH, 2001. ISBN: 3-527-29435-X. DOI: 10.1002/3527602976.

[3] P. Lambropoulos y D. Petrosyan. *Fundamentals of Quantum Optics and Quantum Information*. Springer-Verlag Berlin Heidelberg, 2007. ISBN: 978-3-540-34571-8. DOI: 10.1007/978-3-540-34572-5.

[4] Crispin W. Gardiner y Peter Zoller. *Quantum Noise: A Handbook of Markovian and Non-Markovian Quantum Stochastic Methods with Applications to Quantum Optics*. Berlin, Germany: Springer, 2004. ISBN: 978-3-540-22301-6. URL: https://link.springer.com/book/9783540223016.

[5] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Yorktown Heights, NY, USA: Springer International Publishing, 2018. ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0.

[6] Sergio Gutiérrez Rodrigo, Luis Martín Moreno y David Zueco Láinez. *Actividad complementaria: Introducción a la Inteligencia Artificial*. Universidad de Zaragoza. Enlace a Drive con los contenidos impartidos: https://drive.google.com/drive/folders/1gZTyxXia-VkObZHcSN3XpaenJLXZgdvf?usp=sharing.

[7] Harrison Kinsley y Daniel Kukieła. *Neural Networks from Scratch in Python*. Self-published, 2020. URL: https://nnfs.io/.

[8] Michael Nielsen. *Neural Networks and Deep Learning*. Self-published, 2015. URL: http://neuralnetworksanddeeplearning.com.

[9] **IBM**. *Introduction to Deep Learning with Keras*. URL: https://www.coursera.org/learn/introduction-to-deep-learning-with-keras.

[10] A. Aylin Tokuç. *Underfitting and Overfitting in Machine Learning*. URL: https://www.baeldung.com/cs/ml-underfitting-overfitting.

[11] Scikit-learn. *Compare the effect of different scalers on data with outliers*. URL: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html.

ANEXOS

A. Códigos

Todos los códigos, imágenes, referencias, y demás archivos usados y creados durante la elaboración del TFG, se encuentran en el siguiente repositorio de GitHub.

B. Estudio previo detallado

Tamaño de la base de datos

Lo primero que debemos estudiar es el tamaño de la base de datos que vamos a utilizar para entrenar a nuestro modelo. Es de esperar, que una base de datos mayor será la que minimice los errores, pero debemos tener en cuenta también el tiempo que cuesta entrenar a la red.

Para realizar esta tarea, partiremos de una red neuronal con una única capa oculta con 256 neuronas, a cuya entrada tendrá 300 neuronas correspondientes a los valores de ω, y en su salida tendrá 4 neuronas correspondientes a los 4 parámetros que deseamos predecir (κ, ΔE, γ y g). Como hiperparámetros utilizaremos: $batch\,size = 100$ y $epochs = 1000$ (un número suficientemente grande de épocas para entrenar).

Con la red ya creada, la entrenaremos con bases de datos de entre 100 y 150000 curvas. Podemos ver en la Fig. B.1, como los menores valores de la función pérdida se dan para las bases de datos más grandes, pero también aumenta mucho el tiempo de entrenamiento. Por ello, usaremos una base de datos con 10000 curvas, ya que tiene unas pérdidas prácticamente idénticas a las de 50000 curvas (el menor valor de MSE), pero su tiempo de entrenamiento es notablemente menor, siendo este de 2 minutos frente a 7 minutos.

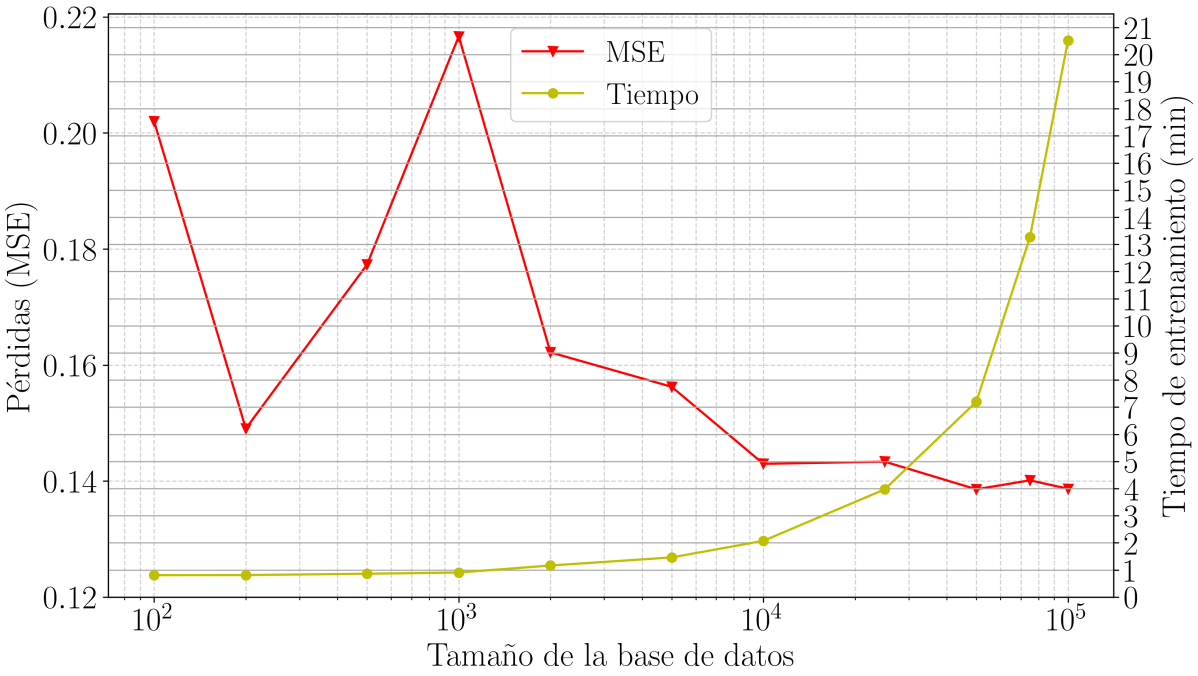


Figura B.1: Pérdidas (rojo) y tiempo de entrenamiento (amarillo), en función del tamaño de la base de datos.

Número de capas y neuronas

Lo siguiente que debemos estudiar a la hora de optimizar la red, es el número de capas ocultas y cuántas neuronas deben haber en cada una de ellas.

Partiremos de una base de datos con 10000 curvas, y al igual que en el caso anterior, fijaremos los hiperparámetros en $batch\,size = 100$ y $epochs = 1000$.

Como nuestro modelo debe ser simple, ya que se trata de un problema de regresión, imponemos la condición de que para varias capas ocultas, las capas consecutivas deben tener un número inferior de neuronas que la capa que le precede. Además, estudiaremos capas cuyos números de neuronas sean 16, 32, 64, 128 o 256, y llegaremos hasta un máximo de 5 capas ocultas.

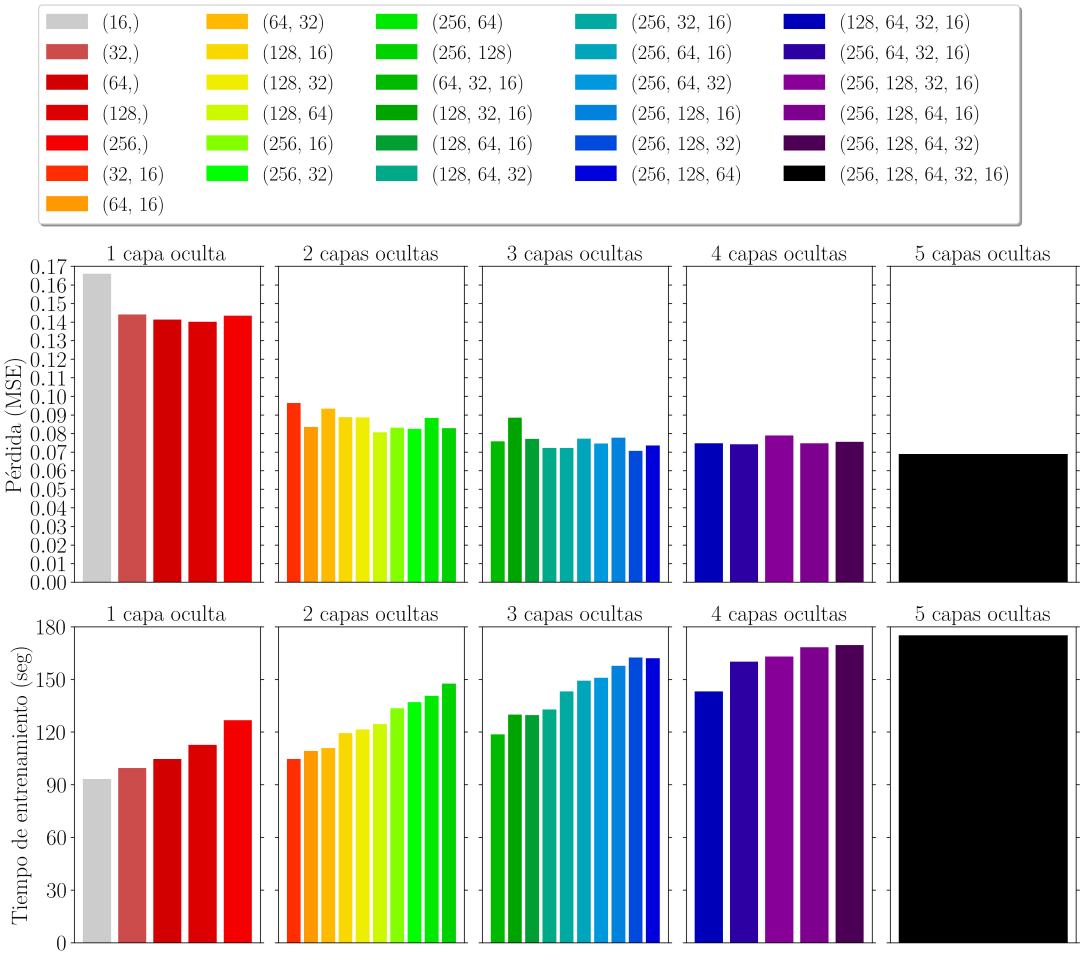


Figura B.2: En la parte de arriba se muestran los valores de la pérdida en función de la combinación de neuronas en cada capa oculta, siendo cada gráfico para un número de capas ocultas, yendo de una cinco capas. La parte inferior, se corresponde a los tiempos de entrenamiento para cada combinación.

Si analizamos en detalle la Fig. B.2, vemos que el mejor valor de MSE se obtiene con la red de 5 capas, aunque hay varias combinaciones de red de 3 capas que también tienen $MSE \approx 0.07$. Sin embargo, estas combinaciones tienen tiempos de entrenamiento de entre 2 y 3 minutos, luego, por esa diferencia de tiempo, preferimos ganar algo de precisión y usaremos una red neuronal de 5 capas ocultas con 256, 128, 64, 32 y 16 neuronas, respectivamente.

Hiperparámetros

A continuación, tenemos que estudiar cuál es la combinación más óptima de los hiperparámetros (*batch size* y *epochs*).

Para hacer esta comprobación, usaremos un dataset de 10000 curvas, y una red neuronal de 5 capas ocultas como la que hemos explicado en Número de capas y neuronas. Se hace una búsqueda en malla, probando todas las combinaciones de, $batch\,size = \{10, 25, 50, 100, 250, 500, 100\}$ y $epochs = \{10, 50, 100, 250, 500, 1000, 2500, 5000, 10000\}$.

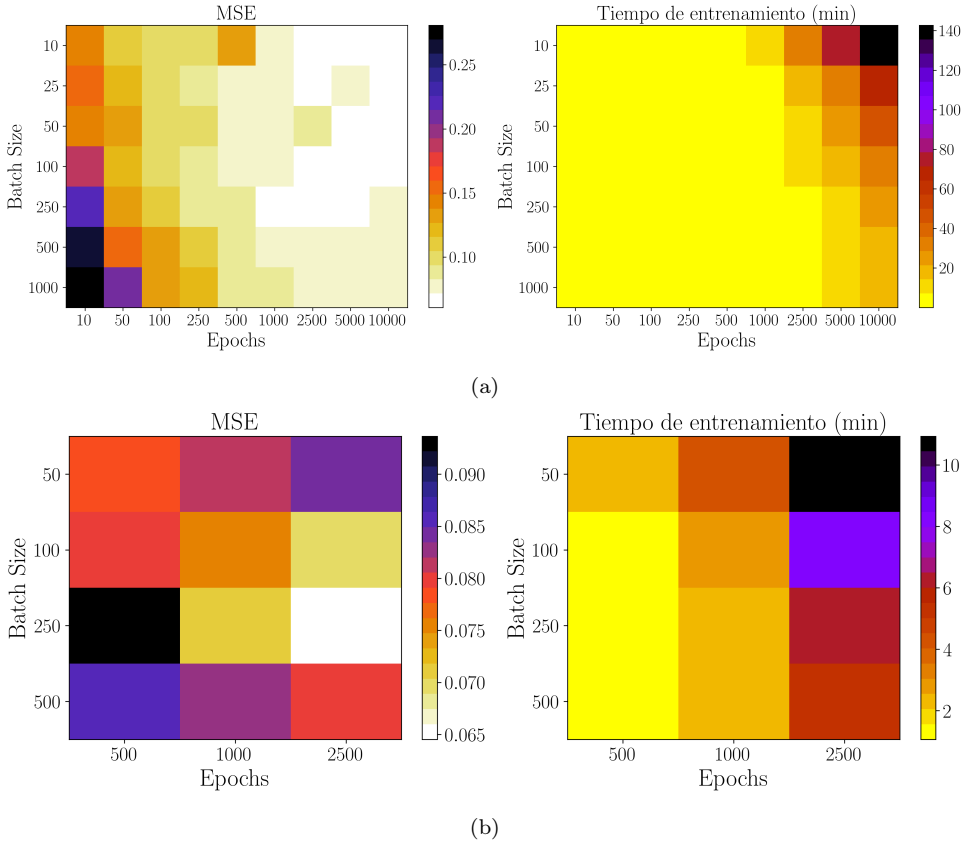


Figura B.3: Valores de MSE y tiempo de entrenamiento para las distintas combinaciones de *batch size* y *epochs*. En (a) se muestra el resultado completo, y en (b) vemos un acotamiento para ver aquellas combinaciones cuyos tiempos de entrenamiento no superen los 15 minutos y sus valores de MSE sean inferiores a 0.1.

Solo hay que observar un poco la Fig. B.3 para ver que las mejores combinaciones de hiperparámetros mirando la función pérdida son: ($batch\,size = 250$, $epochs = 1000$), ($batch\,size = 250$, $epochs = 2500$) y ($batch\,size = 100$, $epochs = 2500$). Entre estas combinaciones, la más eficiente es ($batch\,size = 250$, $epochs = 1000$), ya que con $MSE \approx 0.07$, solo tarda 2 minutos en ser entrenado el modelo, mientras que las otras dos combinaciones tardan más de 6 minutos; por lo que nos quedaremos con esta combinación de hiperparámetros.

Comprobación del *overfitting* y *underfitting*

Por último, se debe verificar que nuestro modelo no está sufriendo *overfitting* ya que el *overfitting* es un problema muy recurrente en las redes neuronales.

Para verificar que nuestra red no está "memorizando" los datos de entrenamiento, vamos a dibujar las pérdidas en función de las épocas. Como vemos en la Fig. B.4, nuestras funciones de pérdidas avanzan como hemos explicado en 3.5, y por lo tanto, podemos asegurar que nuestro modelo está bien entrenado y no sufre ni *overfitting* ni *underfitting*.

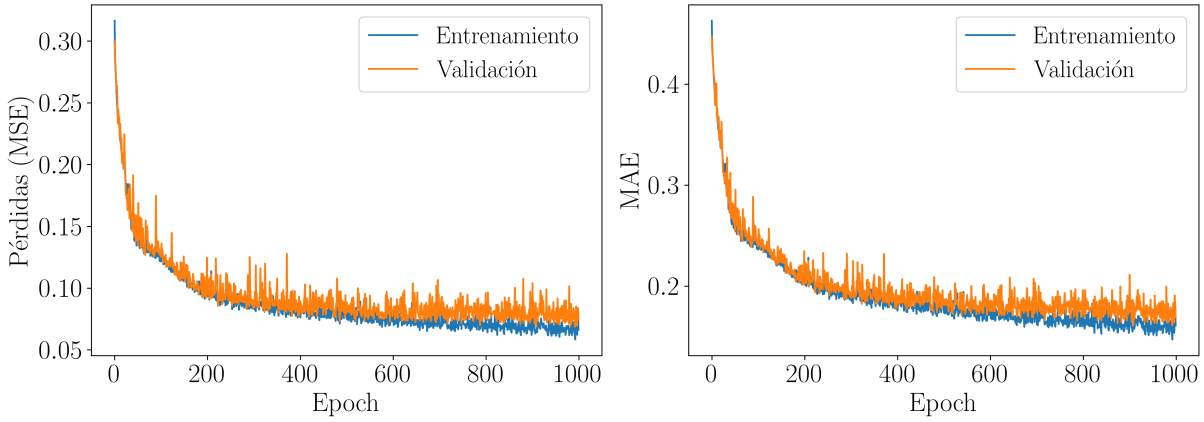


Figura B.4: Verificación de que no hay sobreajuste de la red a los datos de entrenamiento. Se dibujan el MSE y el MAE frente a las *epochs* para los datos de entrenamiento y validación.

C. Gráficas extras

C.1. Curvas con acoplo fuerte

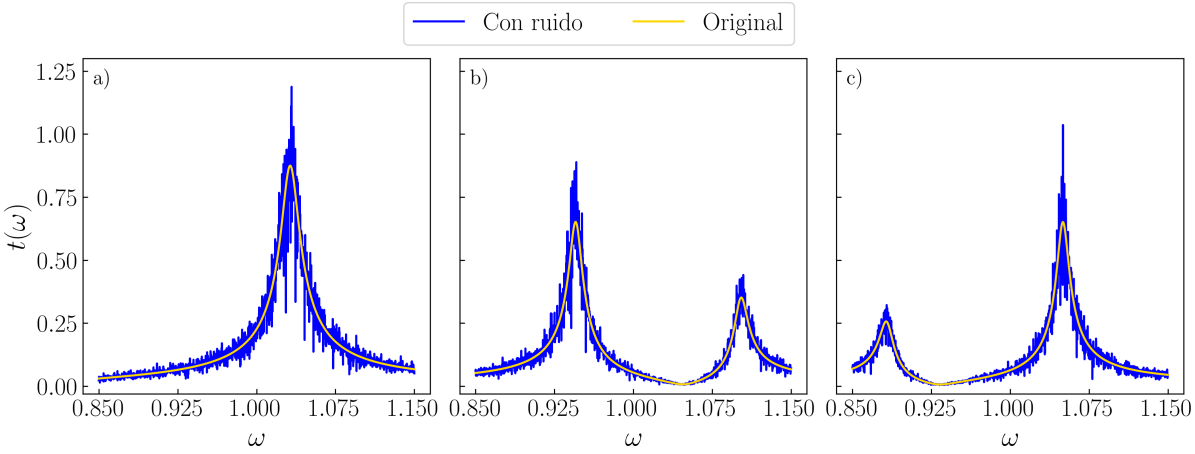


Figura C.1: Ejemplo de curvas con ruido añadido con un rango de g que favoreciese el acoplamiento fuerte ($g \in [10^{-3}, 10^{-1}]$).

C.2. Contour plot

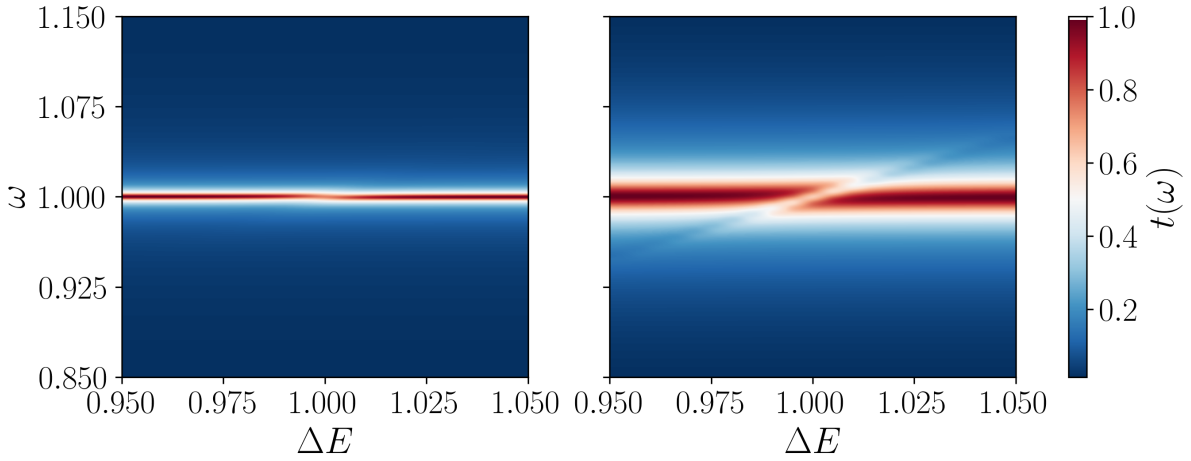


Figura C.2: Contour plot de las curvas de entrenamiento sin ruido

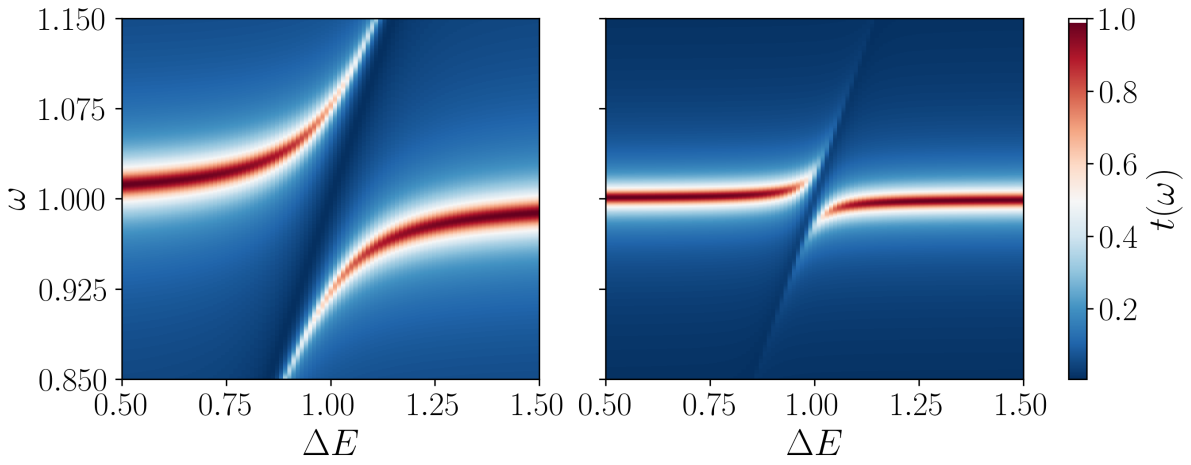


Figura C.3: Contour plot de las curvas de entrenamiento sin ruido, usando valores de g más grandes ($g \in [10^{-3}, 10^{-1}]$), favoreciendo el acoplo fuerte.

C.3. Entrenamiento de modelos adaptados a datos experimentales

C.3.1. Sin ruido

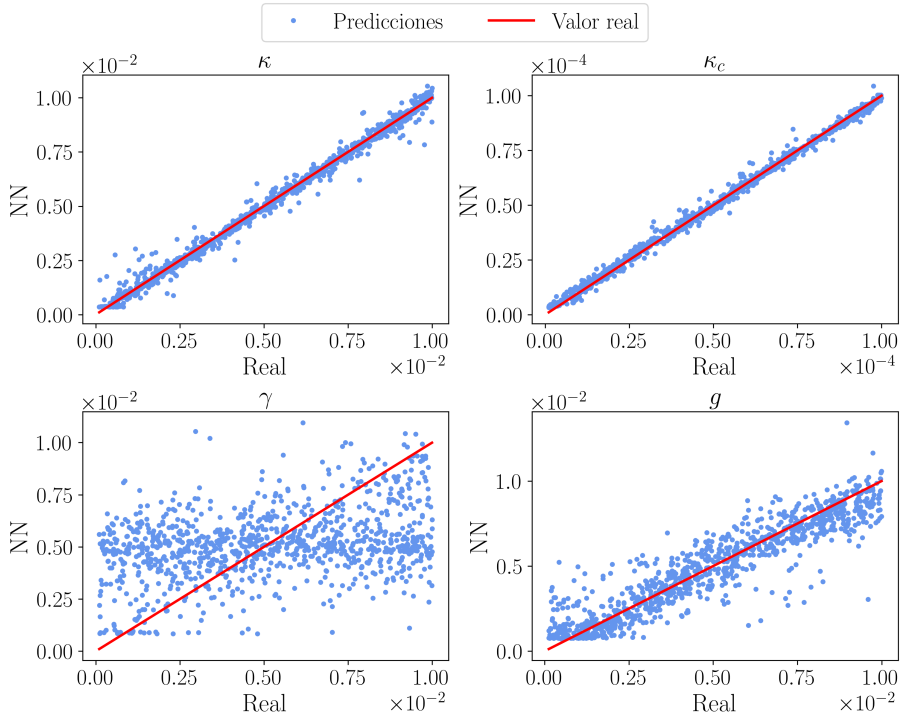


Figura C.4: Predicción de los parámetros para modelo sin ruido y rango genérico.

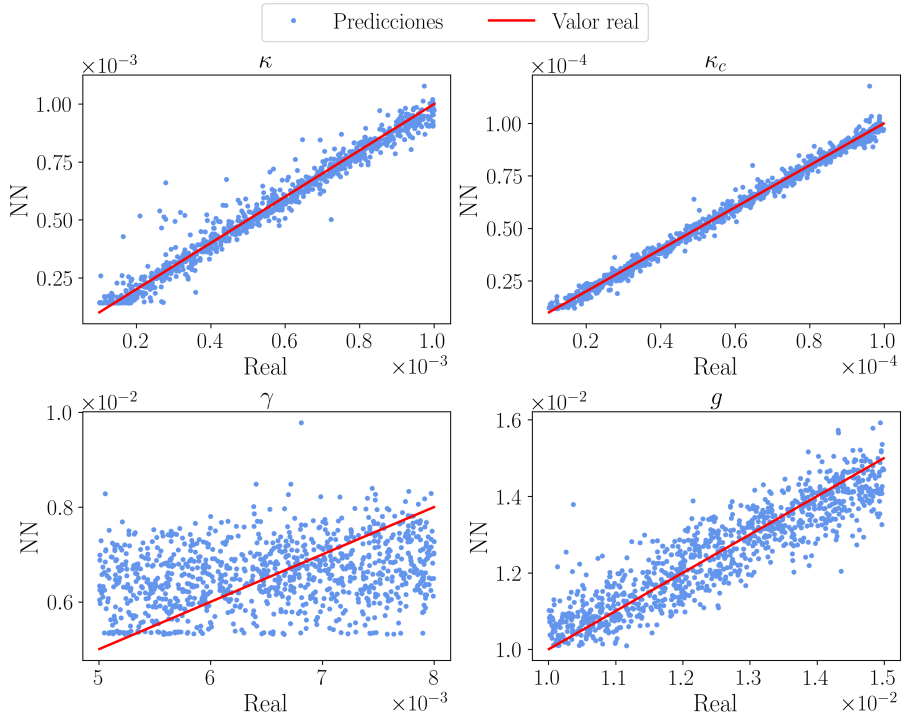


Figura C.5: Predicción de los parámetros para modelo sin ruido y rango acotado.

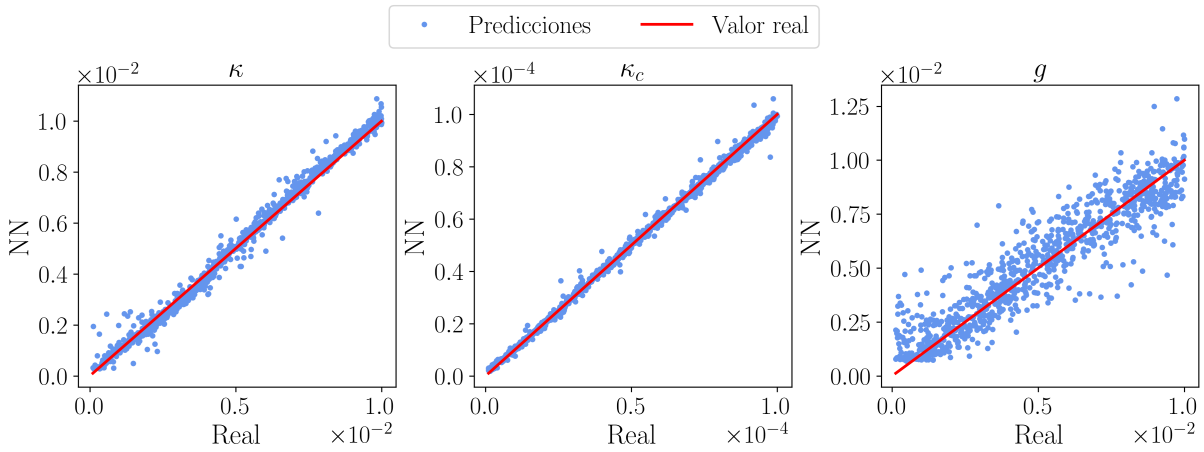


Figura C.6: Predicción de los parámetros para modelo sin ruido, con γ fijado con distintos valores, y rango genérico de parámetros.

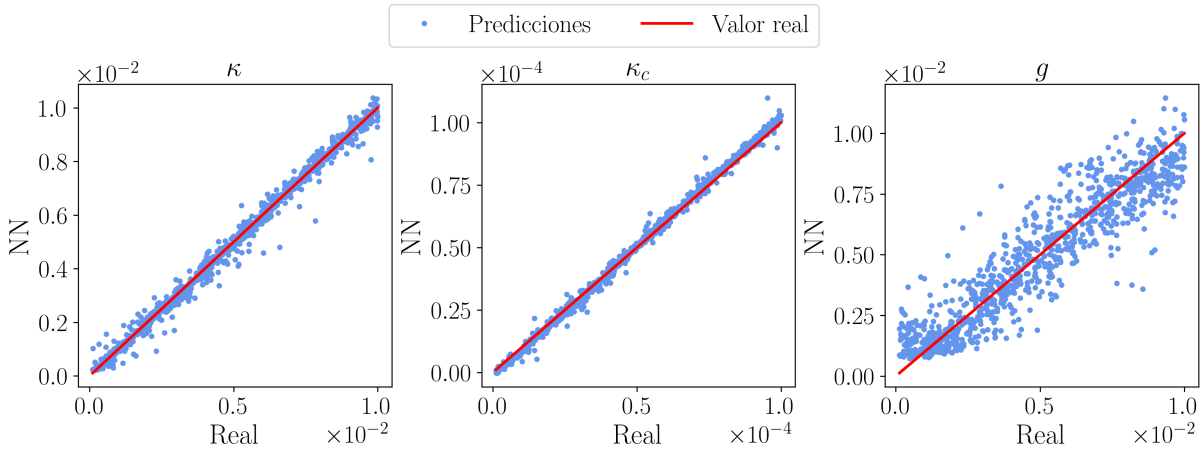


Figura C.7: Predicción de los parámetros para modelo sin ruido, con γ fijo, y rango genérico de parámetros.

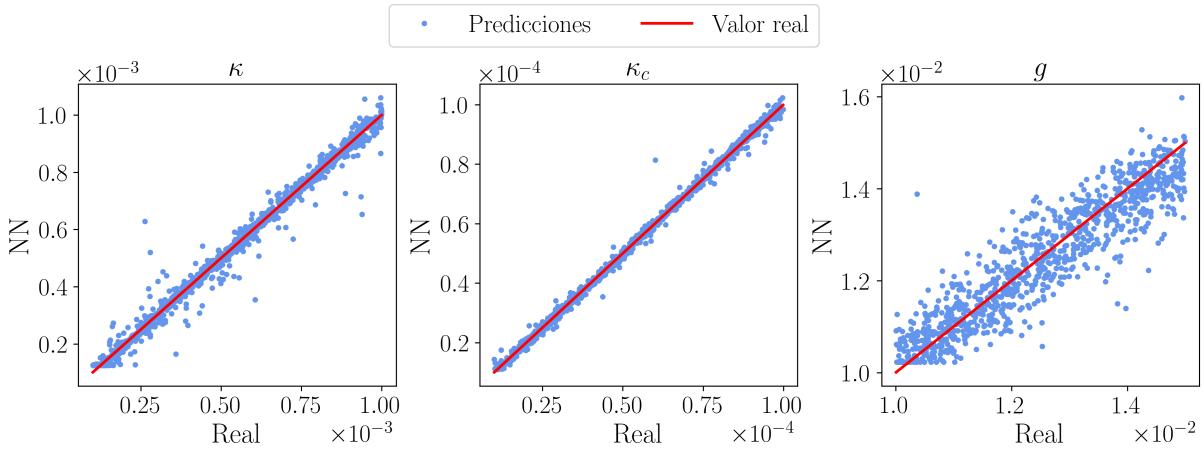


Figura C.8: Predicción de los parámetros para modelo sin ruido, con γ fijado con distintos valores, y rango acotado de parámetros.

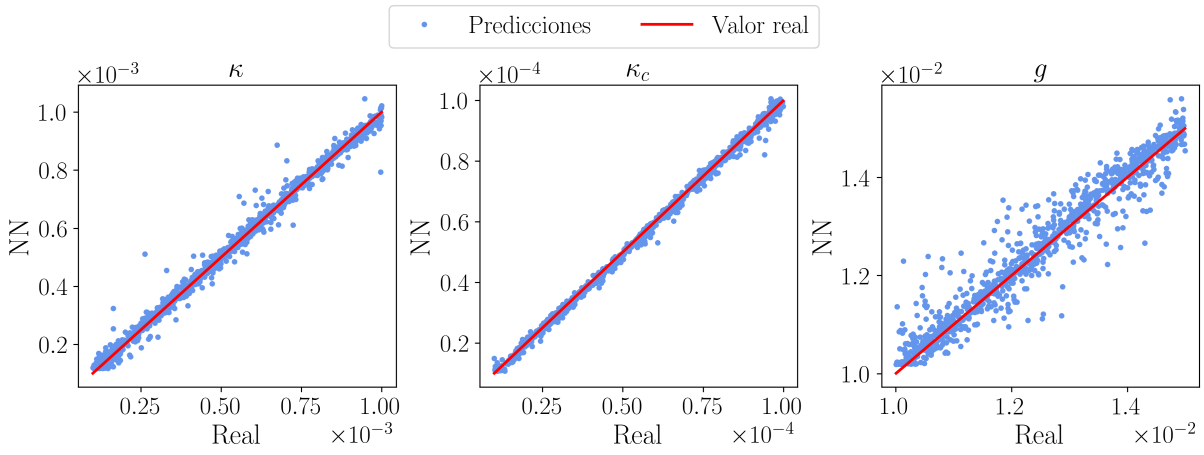


Figura C.9: Predicción de los parámetros para modelo sin ruido, con γ fijo, y rango acotado de parámetros.

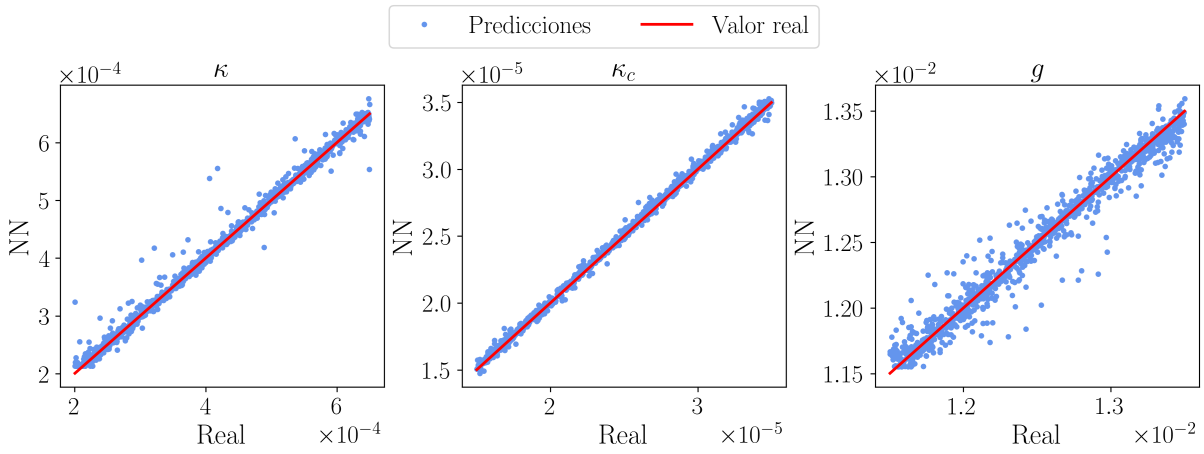


Figura C.10: Predicción de los parámetros para modelo sin ruido, con γ fijo, y rango del segundo acotamiento de parámetros.

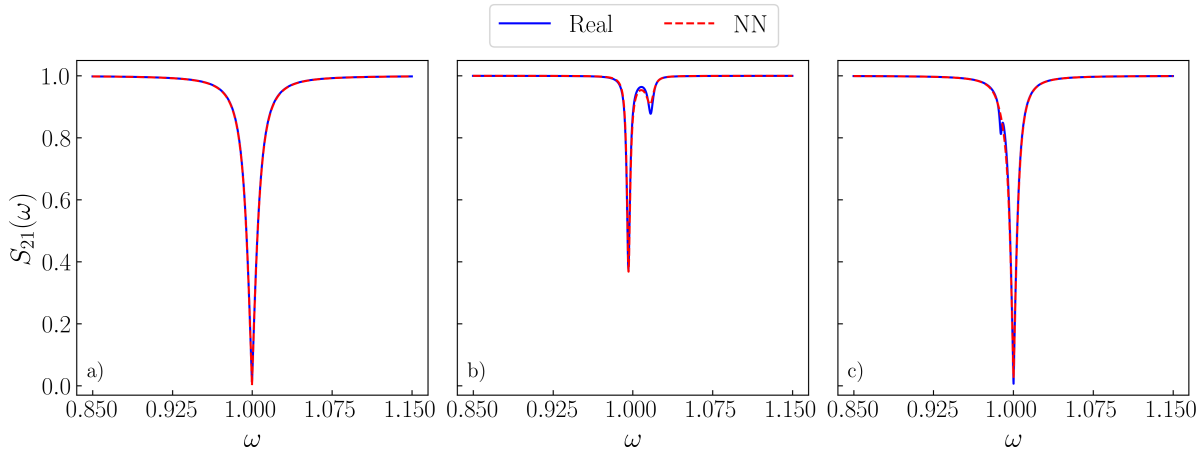


Figura C.11: Algunos ejemplos de curvas y sus predicciones en el modelo sin ruido en rango general de parámetros.

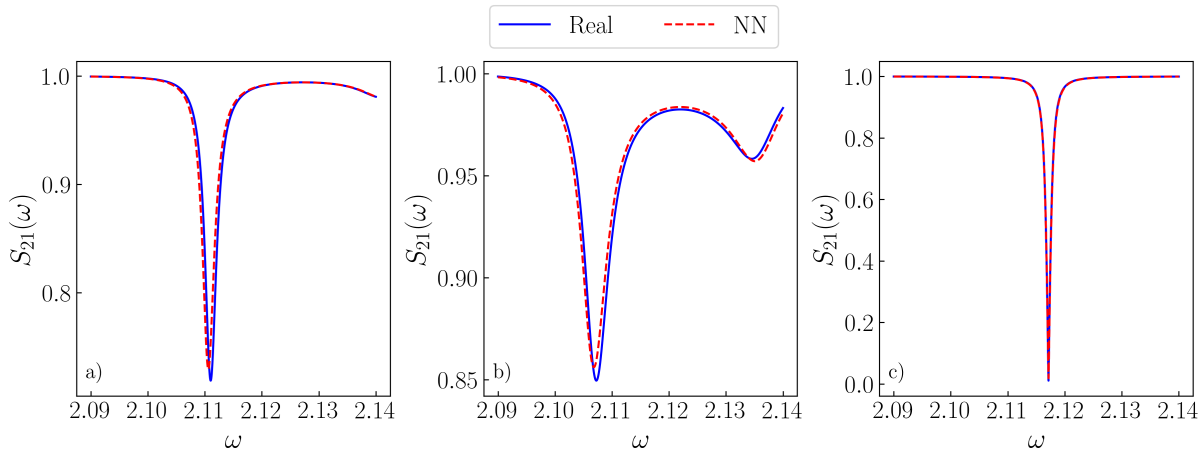


Figura C.12: Algunos ejemplos de curvas y sus predicciones en el modelo sin ruido en rango acotado de parámetros.

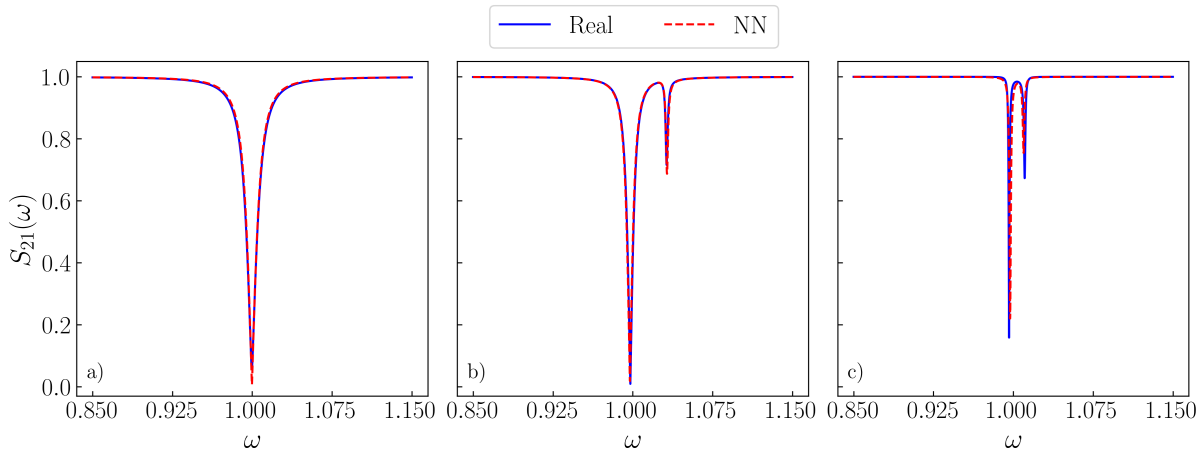


Figura C.13: Algunos ejemplos de curvas y sus predicciones en el modelo sin ruido, con γ fijo, en rango general de parámetros.

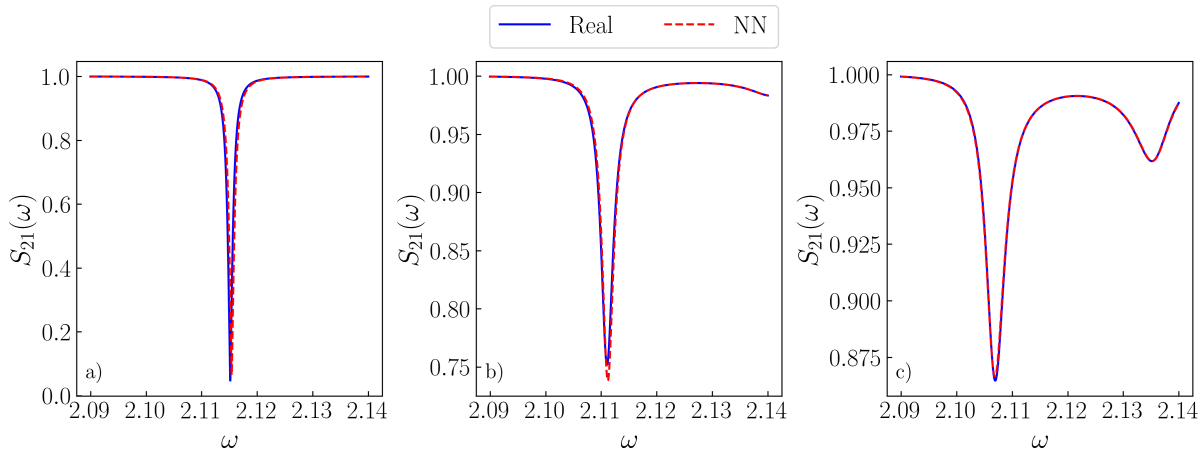


Figura C.14: Algunos ejemplos de curvas y sus predicciones en el modelo sin ruido, con γ fijado con distintos valores, en rango acotado de parámetros.

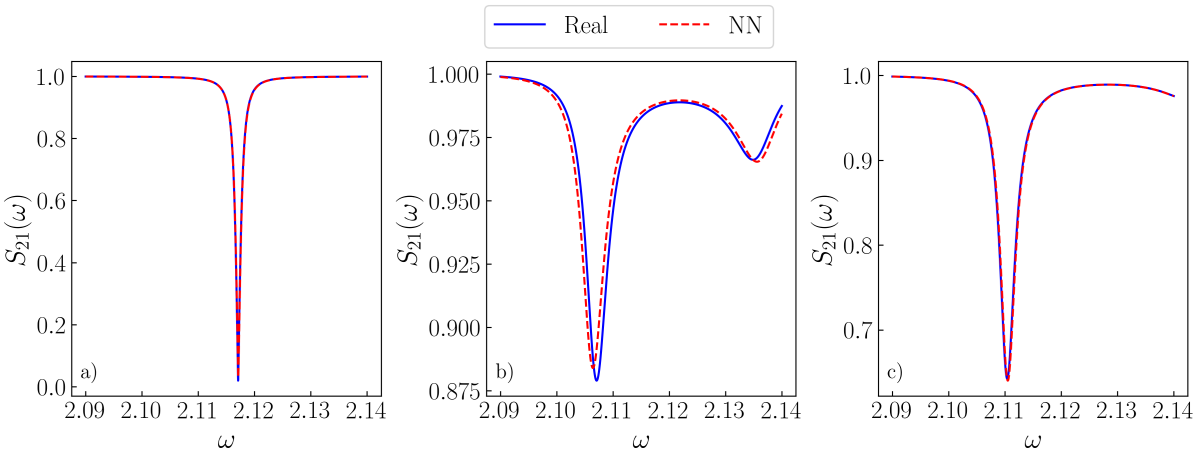


Figura C.15: Algunos ejemplos de curvas y sus predicciones en el modelo sin ruido, con γ fijado, en rango acotado de parámetros.

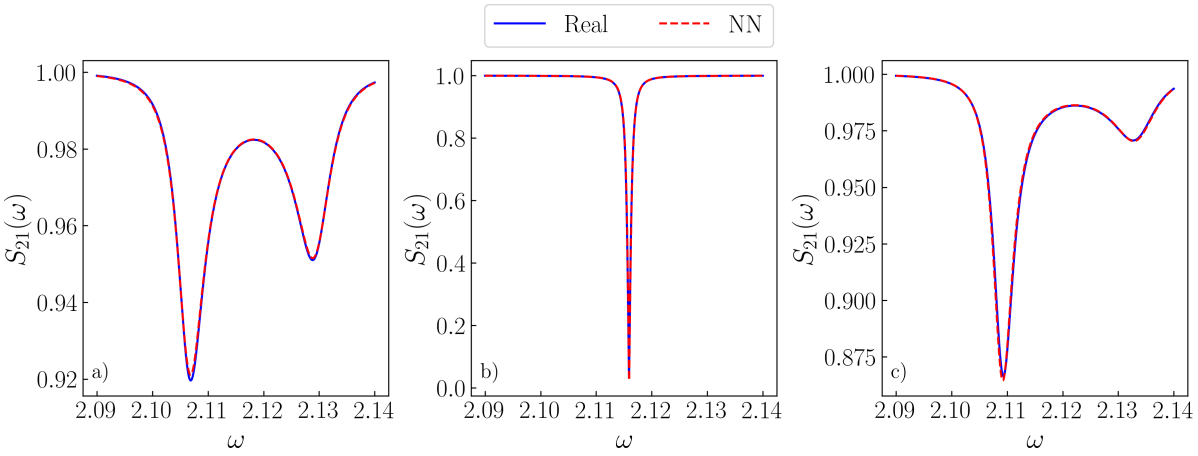


Figura C.16: Algunos ejemplos de curvas y sus predicciones en el modelo sin ruido, con γ fijo, y rango del segundo acotamiento de parámetros.

C.3.2. Con ruido

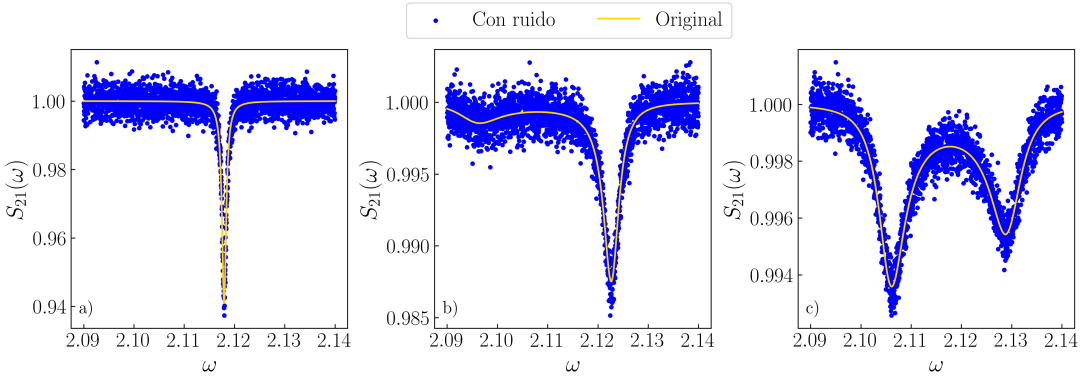


Figura C.17: Ejemplos de curvas tras añadir ruido al usar $S_{21}(\omega)$

Antes de hacer estudio de hiperparámetros

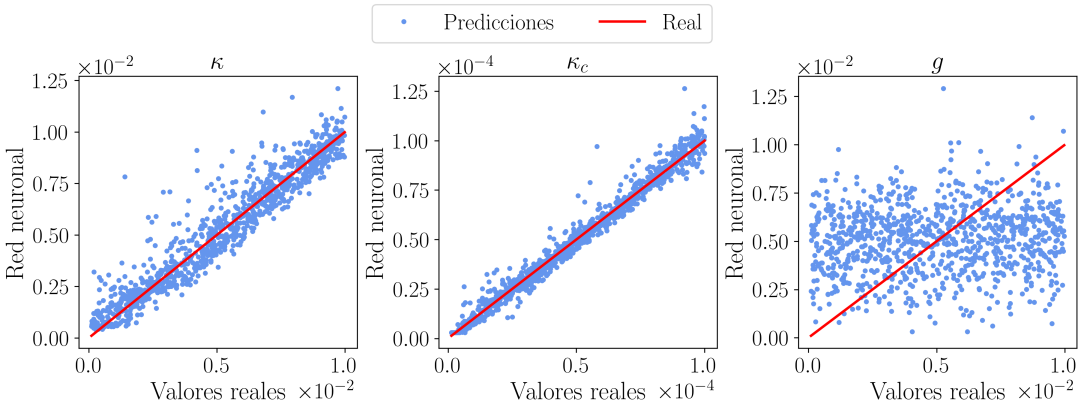


Figura C.18: Predicción de los parámetros para modelo con ruido antes del estudio de hiperparámetros, con γ fijo, en rango general de parámetros.

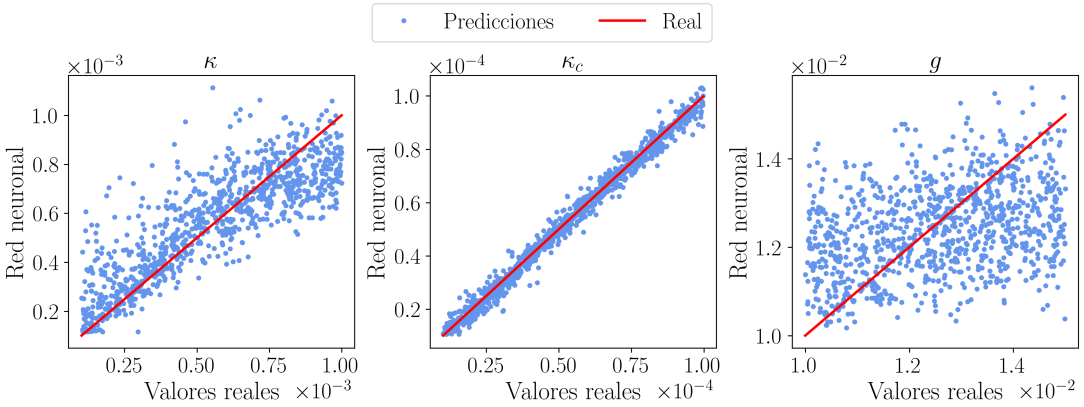


Figura C.19: Predicción de los parámetros para modelo con ruido antes del estudio de hiperparámetros, con γ fijo, en rango acotado de parámetros.

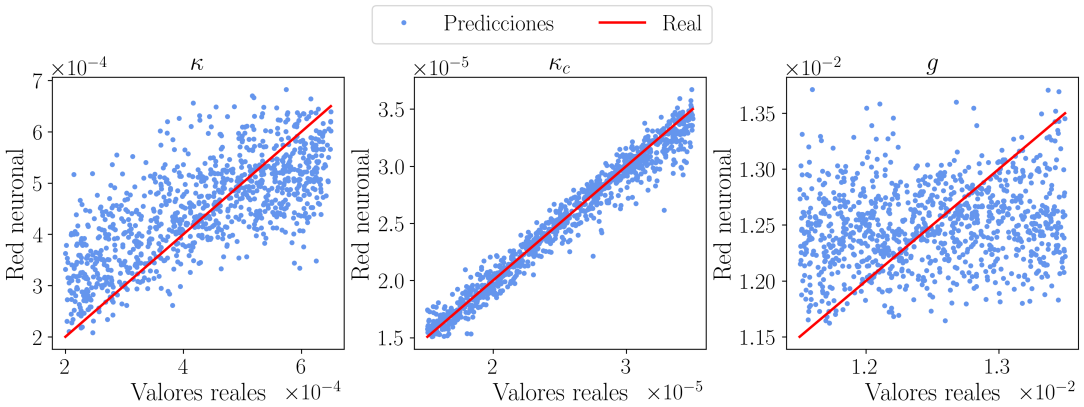


Figura C.20: Predicción de los parámetros para modelo con ruido antes del estudio de hiperparámetros, con γ fijo, y rango del segundo acotamiento de parámetros.

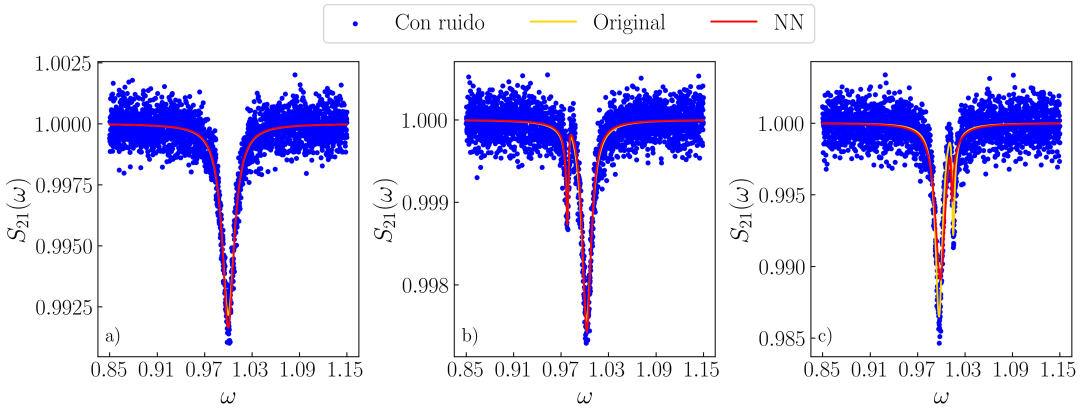


Figura C.21: Algunos ejemplos de curvas y sus predicciones en el modelo con ruido antes del estudio de hiperparámetros, para el rango general de parámetros.

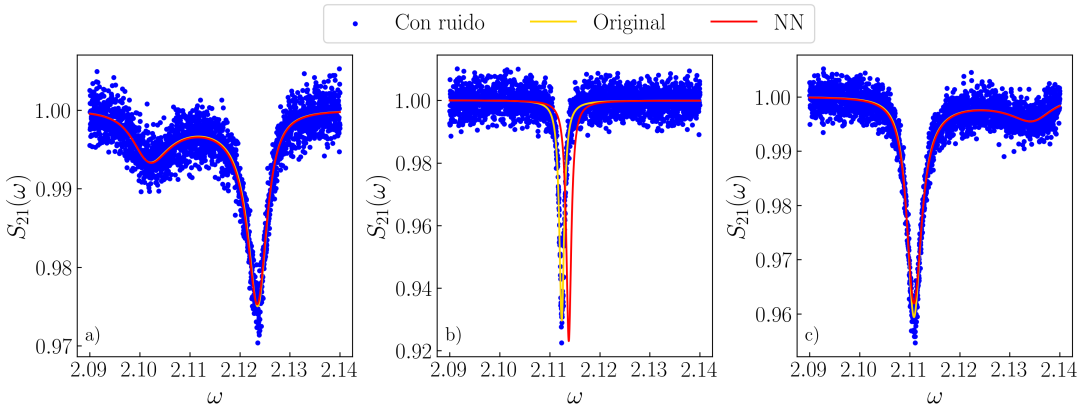


Figura C.22: Algunos ejemplos de curvas y sus predicciones en el modelo con ruido antes del estudio de hiperparámetros, para el rango acotado de parámetros.

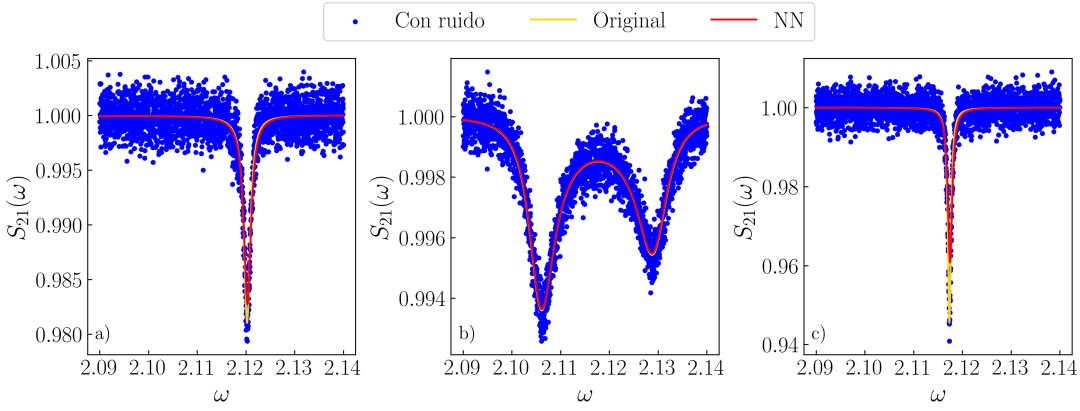


Figura C.23: Algunos ejemplos de curvas y sus predicciones en el modelo con ruido antes del estudio de hiperparámetros, para el segundo rango acotado de parámetros.

Post estudio

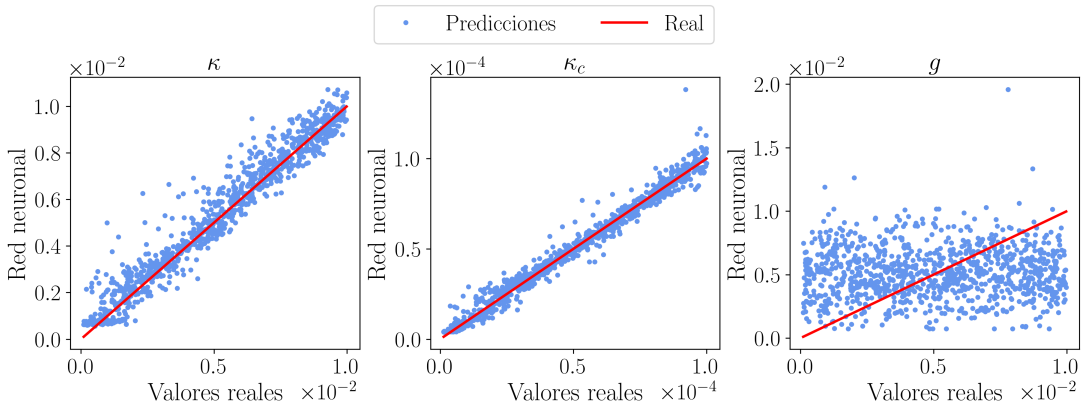


Figura C.24: Predicción de los parámetros para modelo con ruido después del estudio de hiperparámetros, con γ fijo, en rango general de parámetros.

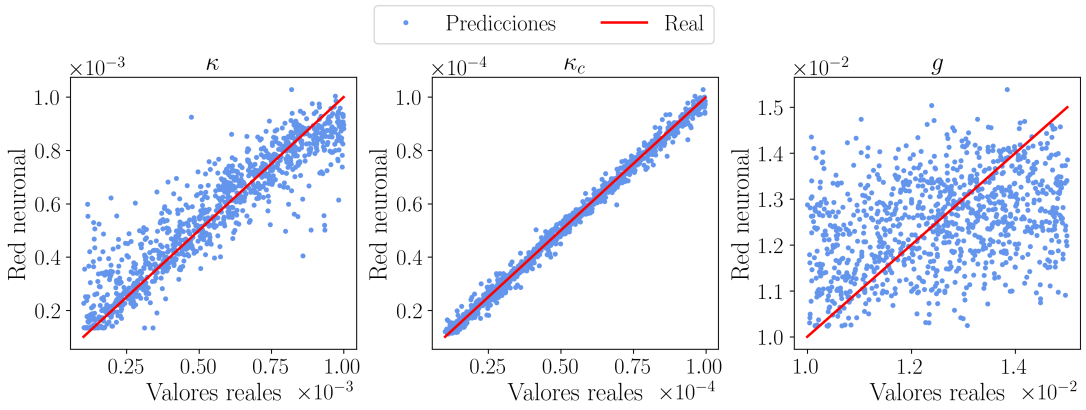


Figura C.25: Predicción de los parámetros para modelo con ruido después del estudio de hiperparámetros, con γ fijo, en rango acotado de parámetros.

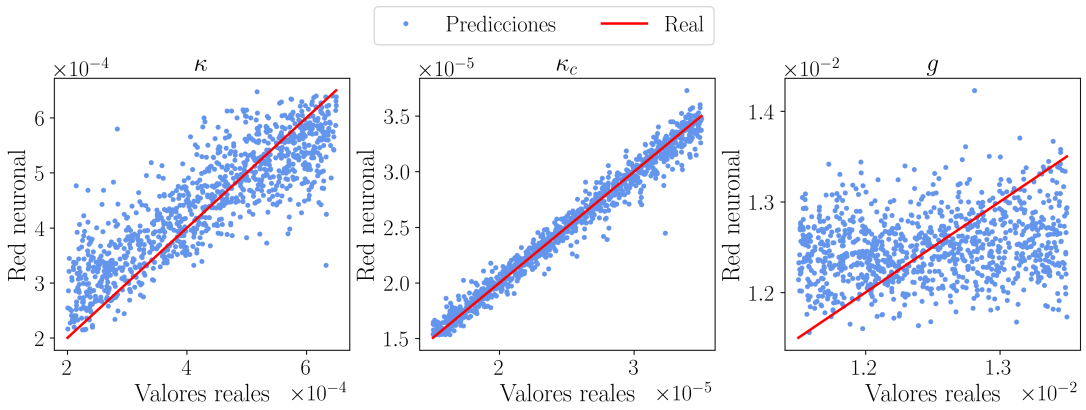


Figura C.26: Predicción de los parámetros para modelo con ruido después del estudio de hiperparámetros, con γ fijo, en el segundo rango acotado de parámetros.

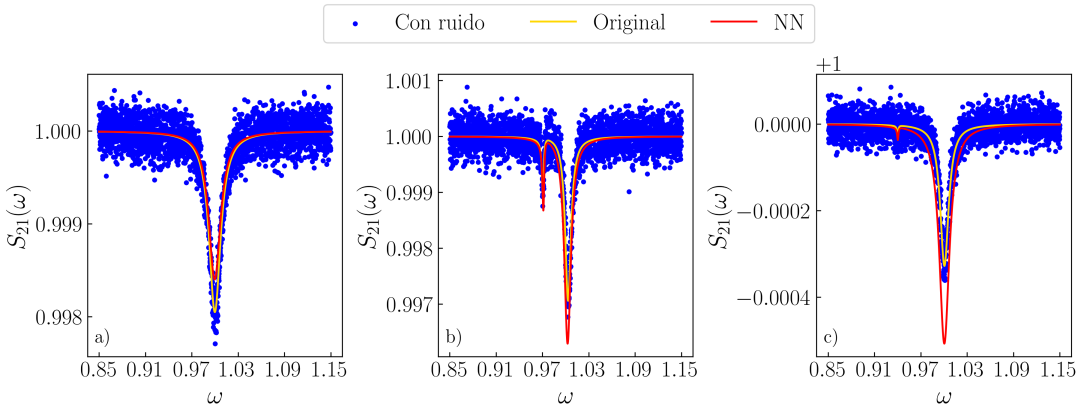


Figura C.27: Algunos ejemplos de curvas y sus predicciones en el modelo con ruido después del estudio de hiperparámetros, para el rango genérico de parámetros.

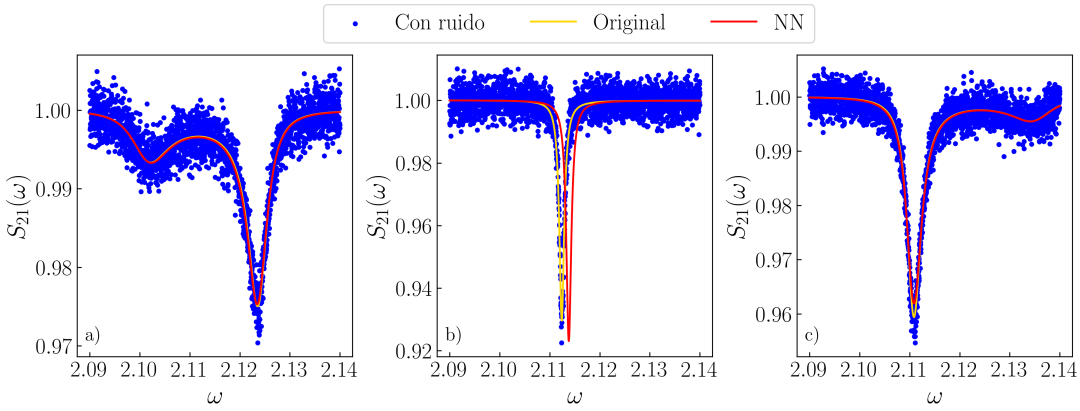


Figura C.28: Algunos ejemplos de curvas y sus predicciones en el modelo con ruido después del estudio de hiperparámetros, para el rango acotado de parámetros.

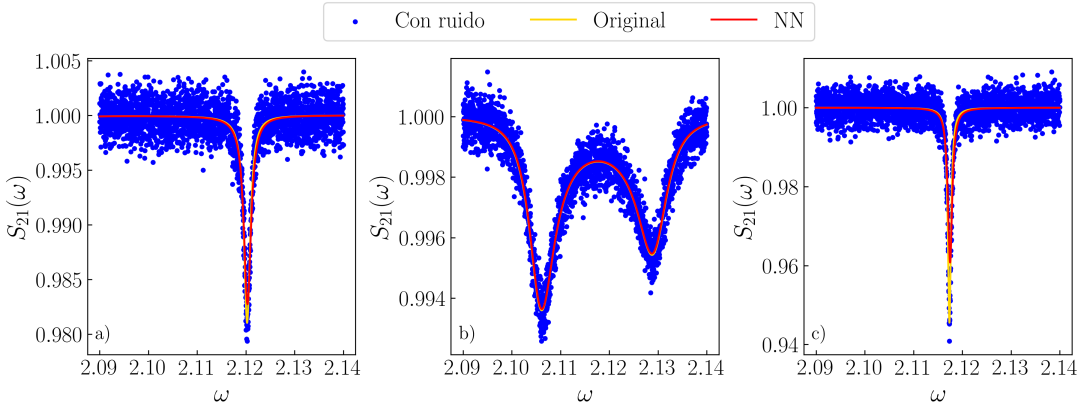


Figura C.29: Algunos ejemplos de curvas y sus predicciones en el modelo con ruidodespués del estudio de hiperparámetros, para el rango acotado de parámetros.

C.4. Otras figuras de resultados experimentales

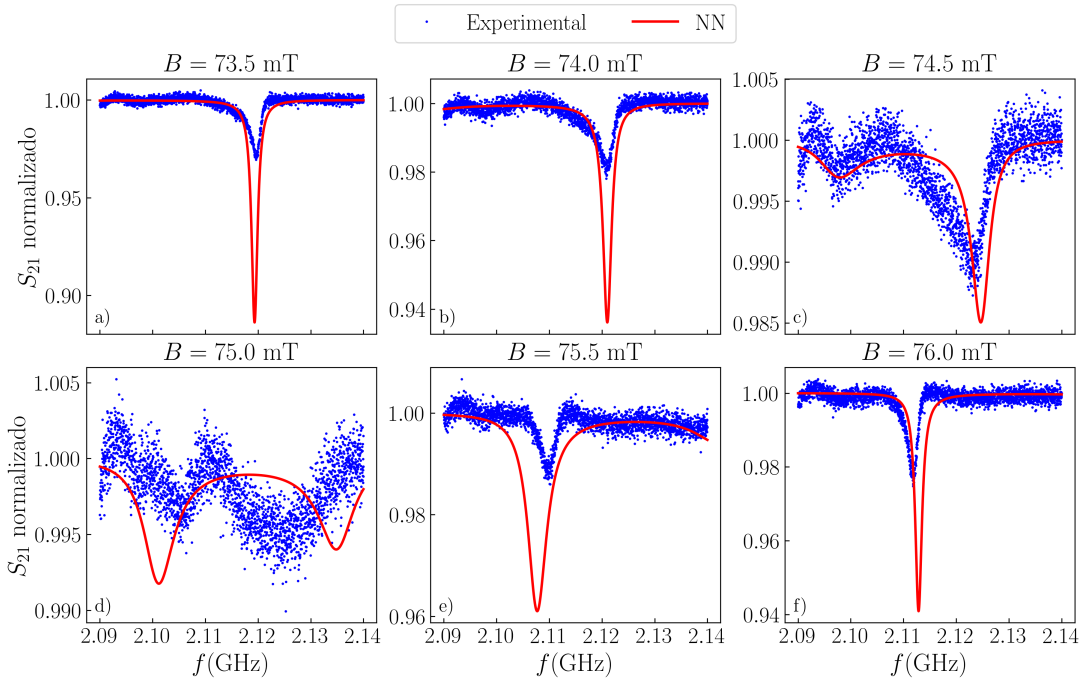


Figura C.30: Predicción de las curvas experimentales con el modelo sin ruido, con γ fijado en distintos valores, para el rango acotado de parámetros

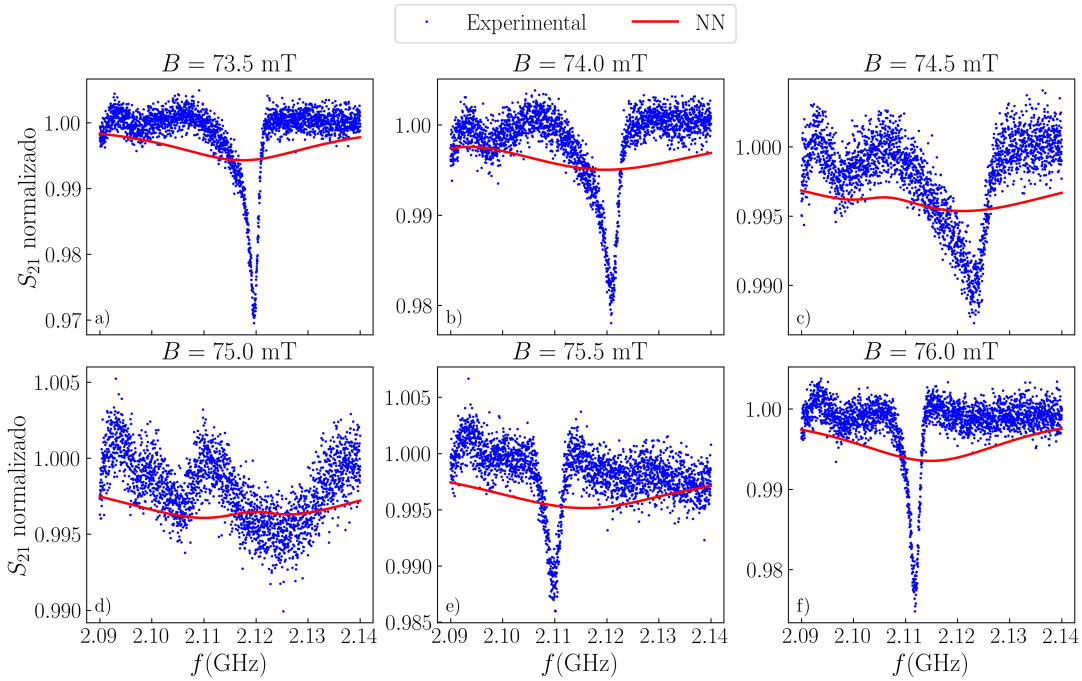


Figura C.31: Predicción de las curvas experimentales con el modelo con ruido antes del estudio de hiperparámetros, con γ fijo, para el rango general de parámetros

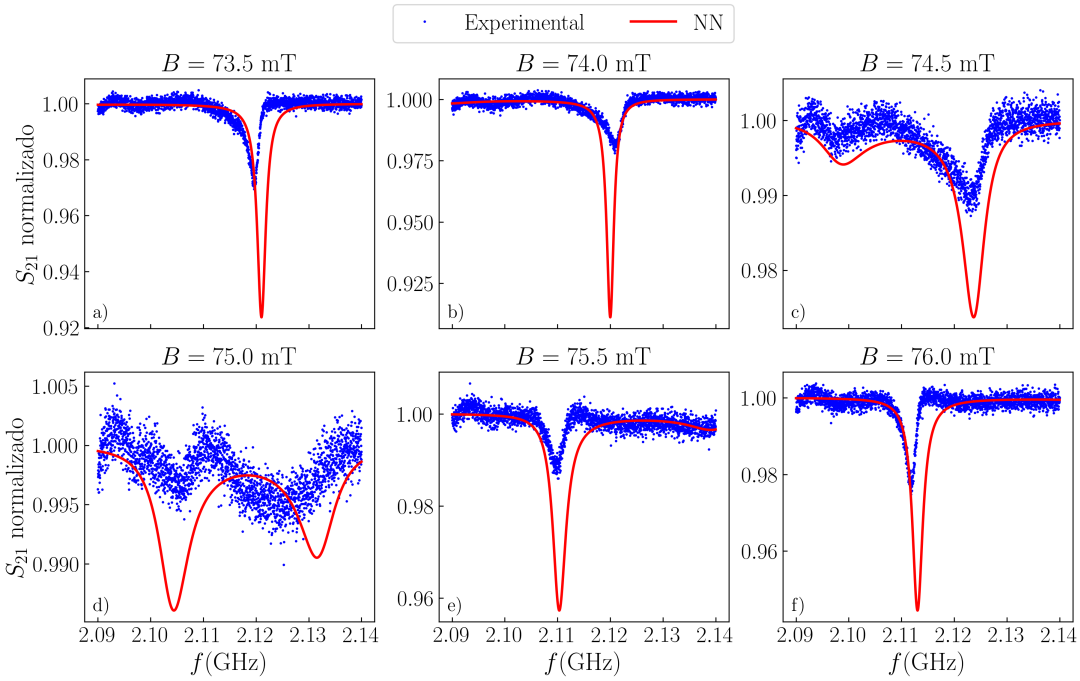


Figura C.32: Predicción de las curvas experimentales con el modelo con ruido antes del estudio de hiperparámetros, con γ fijo, para el rango acotado de parámetros

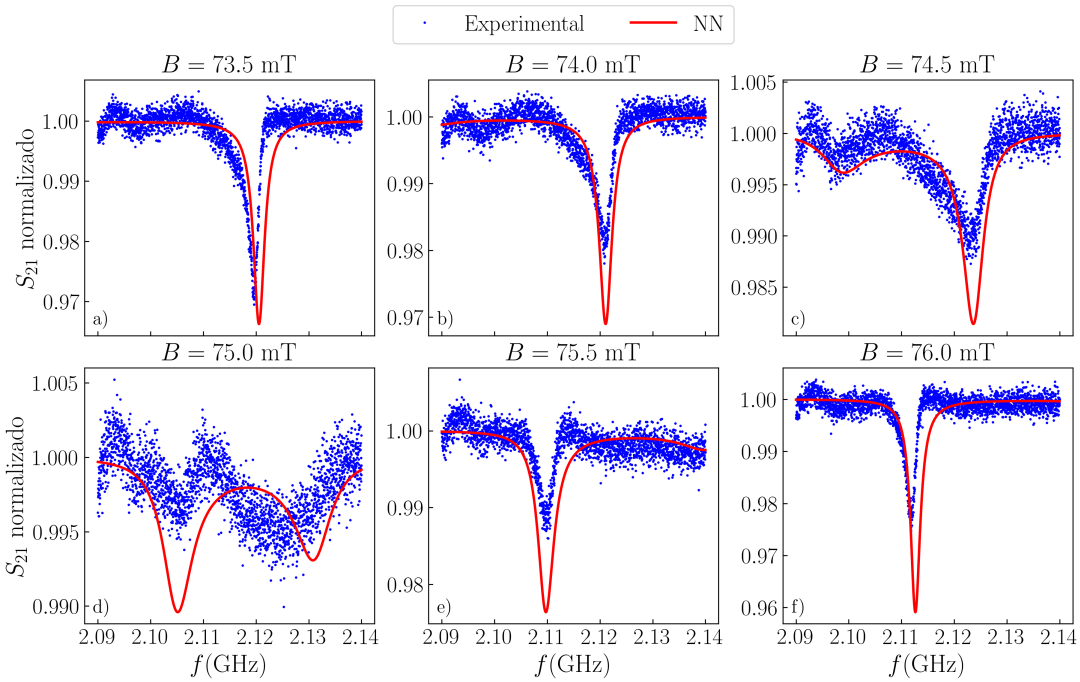


Figura C.33: Predicción de las curvas experimentales con el modelo con ruido antes del estudio de hiperparámetros, con γ fijo, para el segundo acotamiento de parámetros

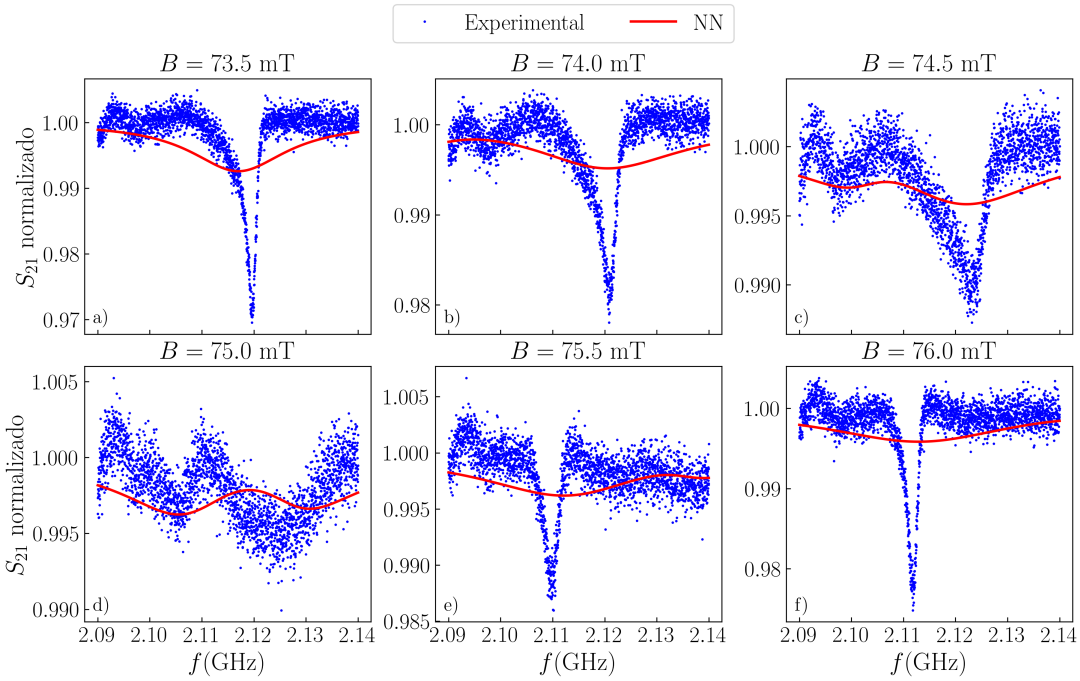


Figura C.34: Predicción de las curvas experimentales con el modelo con ruido después del estudio de hiperparámetros, con γ fijo, para el rango general de parámetros

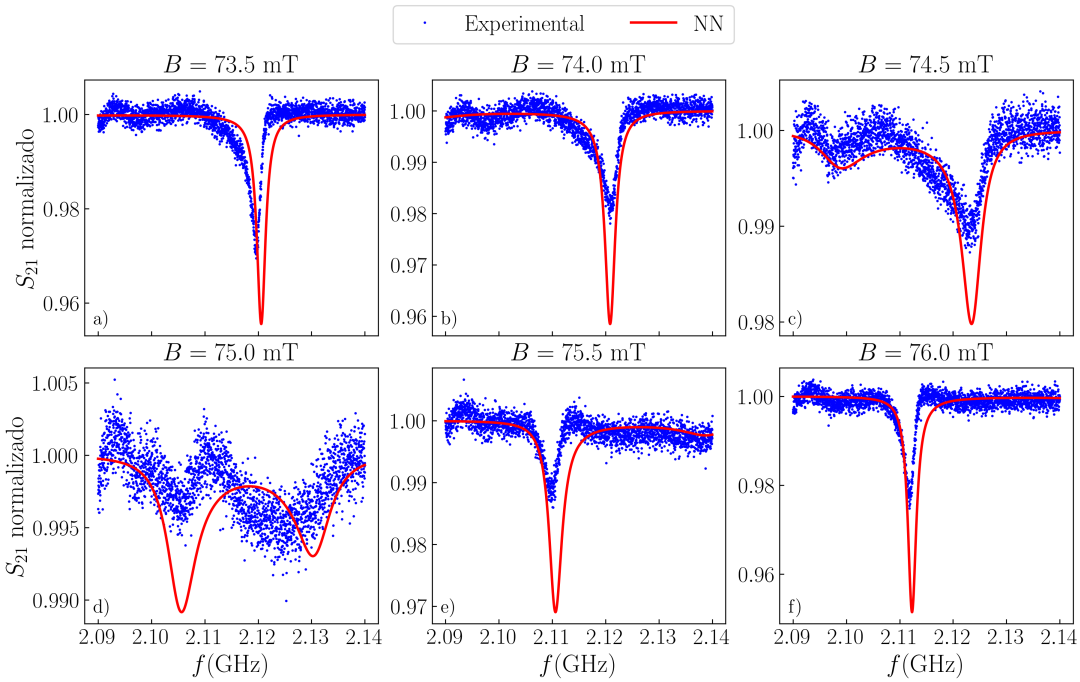


Figura C.35: Predicción de las curvas experimentales con el modelo con ruido después del estudio de hiperparámetros, con γ fijo, para el segundo acotamiento de parámetros

C.5. Segundo estudio de hiperparámetros

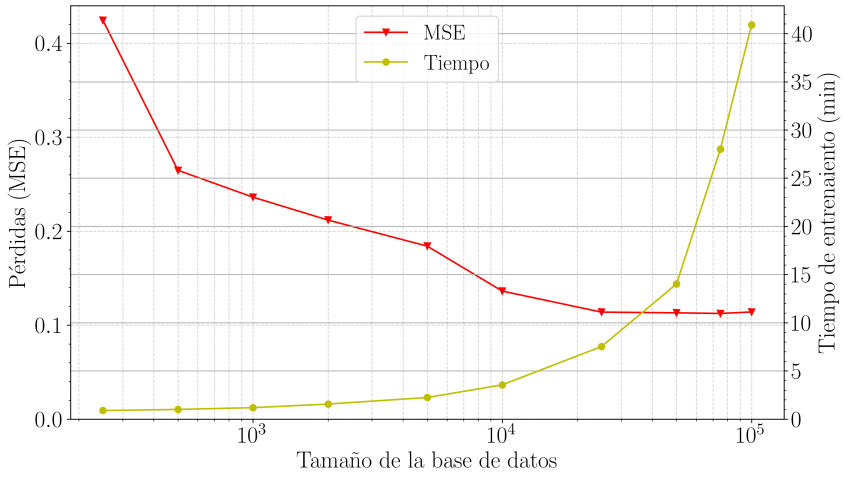


Figura C.36: Estudio del tamaño de la base de datos para el modelo con ruido adaptado a los datos experimentales.

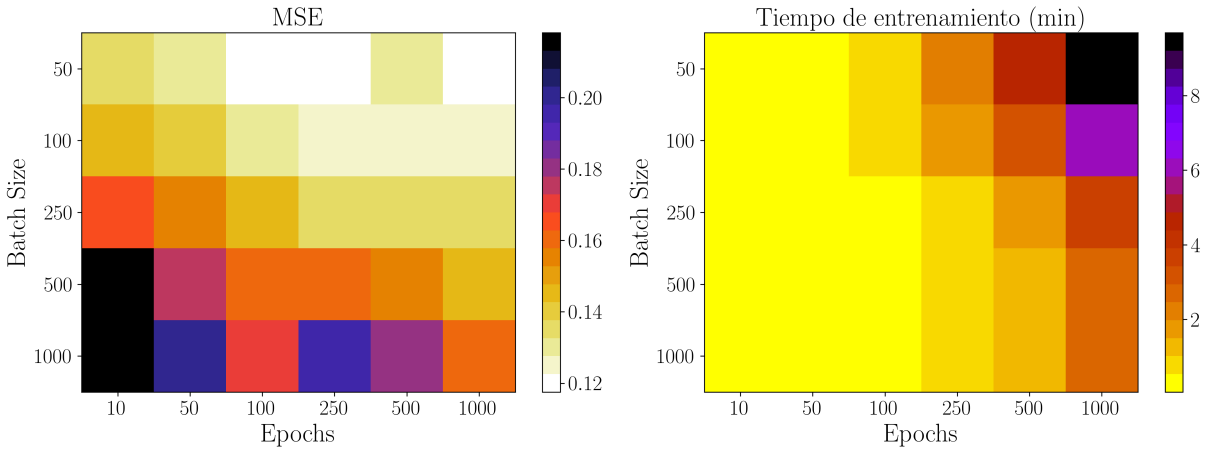


Figura C.37: Estudio de la relación (batch size, epochs) para el modelo con ruido adaptado a los datos experimentales.

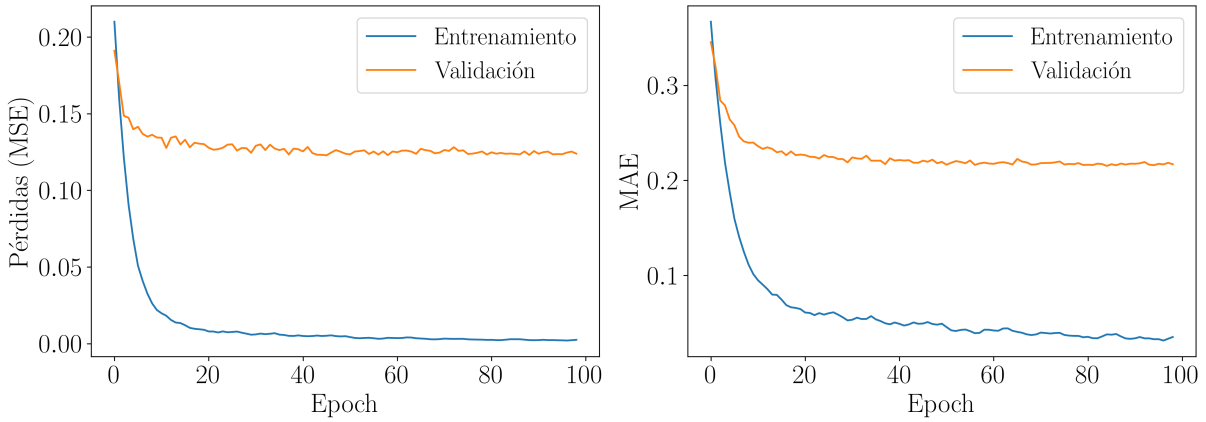


Figura C.38: Comprobación de que no hay overfitting en el nuevo modelo con ruido adaptado a los datos experimentales.