Informe de Trabajo Profesional

GenPer: Software de tomografía por microondas

Autor: Estanislao María Ledesma

(Padrón: 96622)

Tutor: Dr. César Federico Caiafa

Índice

Introducción

GenPer ("**Gen**erador de **per**mitividades") es un software capaz de reconstruir imágenes de permitividades de dieléctricos a partir de sus datos de dispersión obtenidos de tomografías por microondas (problema de dispersión inversa o inverse scattering problem - ISP). Este software está basado en la red neuronal desarrollada por Zhun Wei en [8].

Debido a factores de dispersión múltiple, este tipo de problemas se clasifican como no lineales, y por lo general se utilizan algoritmos iterativos. Esto implica gran costo computacional, razón por la cual, se han propuesto otros enfoques para lograr su resolución. Entre ellos, la utilización de métodos de machine learning como redes neuronales convolucionales (CNN) combinados con técnicas de preprocesamiento aplicadas a las mediciones antes del entrenamiento del set de datos [1]. Esto resulta disruptivo, ya que dejan de utilizarse los modelos matemáticos, para resolverse mediante algoritmos que utilizan la estadística para aprender de ejemplos.

Este tipo de problema puede ser aplicado a la tomografía por microondas (MWT) sobre tejido blando, ya que el hueso, la grasa, el músculo, el tejido blando y las potenciales anormalidades o problemas del tejido (un tumor, por ejemplo), poseen distintos valores de permitividades. A pesar de que estas diferencias sean pequeñas, se ha comprobado que este tipo de tomografías es capaz de aplicarse a extremidades, cerebro y corazón, y puede diagnosticar cáncer de mama o pulmón [2].

El repositorio con el código puede hallarse en [14]

Descripción del problema

La configuración del problema puede ser representada según la siguiente figura:

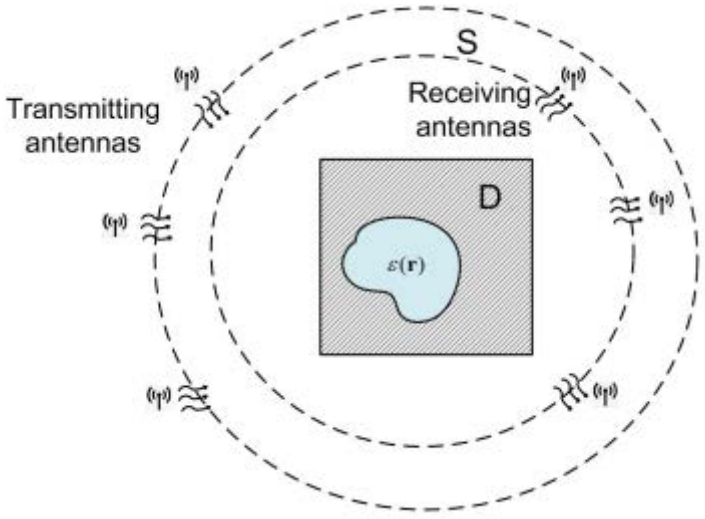

Fig. 1: Esquema típico de un problema de dispersión inversa [1]

Se considera un dominio D de dos dimensiones en modo transversal electromagnético donde se ubica el dieléctrico, cuyo campo eléctrico es perpendicular al mismo. Se consideran N_i fuentes lineales situadas a r_p^i, con p = 1, 2, …, N_i que inciden sobre el mismo. Por cada incidencia, el campo disperso es medido por N_r antenas situadas a r_p^s, con p = 1, 2, …, N_r.

El problema se encuentra planteado según dos ecuaciones, la primera es la ecuación integral del campo eléctrico (EFIE) o ecuación de Lippmann-Schwinger:

$$E^t(r) = E^i(r) + k_0^2 \int_D g(r, r')\, I(r')\, dr',\ r \in D, \text{ (1)}$$

donde $E^t(r)$ y $E^i(r)$son el campo eléctrico total e incidente, respectivamente, $k_0 = \omega\sqrt{\mu_0\,\varepsilon_0}$es el número de onda de medio homogéneo, y $g(r, r')$ es la función de Green bidimensional. Su versión discretizada (para D en M x M subunidades, con centros en r_n, n = 1, 2, …, M^2) es:

$$\overline{E}^t = \overline{E}^i + \widehat{G}_D \cdot \bar{I} \text{ (1.1)}$$

(notación: $\bar{I}$ es un vector y $\widehat{G}_D$es una matriz),

donde $\overline{E}^t$,$\overline{E}^i$ e $\bar{I}$ son vectores de M^2 dimensiones, con el n-ésimo elemento siendo $\overline{E}^t(n) = E^t(r_n)$, $\overline{E}^i(n) = E^i(r_n)$ e $\bar{I}(n) = I(r_n)$, respectivamente. La matriz $\widehat{G}_D$ es de M^2xM^2 dimensiones, con $\widehat{G}_D(n, n') = k_0^2 A_{n'} g(r_n, r_{n'})$, donde $A_{n'}$ es el área de las n'-ésimas subunidades, siendo n = 1, 2, …,M^2 y n' = 1, 2, …,M^2.

La densidad de corriente está definida como:

$$I(r) = \xi(r)\, E^t(r), \text{ (2)}$$

donde $\xi(r) = \varepsilon_r(r) - 1$, siendo $\varepsilon_r(r)$ la permitividad en r. Su versión discretizada es:

$$\bar{I} = diag(\bar{\xi}) \cdot \overline{E}^t, \text{ (2.1)}$$

donde $\bar{\xi}$ es un vector de M^2 dimensiones y $diag(\bar{\xi})$ es la matriz de MxM dimensiones cuya diagonal son las componentes del vector $\bar{\xi}$.

La segunda ecuación del problema es:

$$E^s(r) = \int_D g(r, r')\, I(r')\, dr',\ r \in S, \text{ (3)}$$

donde $E^s(r)$ es el campo eléctrico disperso en la superficie S. Su versión discretizada (para D en M x M subunidades, con centros en r_n, n = 1, 2, …, M^2) es:

$$\overline{E}^s = \widehat{G}_s \cdot \bar{I}, \text{ (3.1)}$$

donde $\overline{E}^s$ es un vector de N_r dimensiones, con el q-ésimo elemento siendo $\overline{E}^s(q) = E^s(r_q)$. La matriz $\widehat{G}_D$ es de $N_r x M^2$ dimensiones, siendo $\widehat{G}_S(q, n') = k_0^2 A_{n'} g(r_q, r_{n'})$, q = 1, 2, …, N_r.

En un problema de dispersión inversa, el objetivo consiste en obtener las permitividades relativas $\varepsilon_r(r)(r \in D)$ a partir de las mediciones de campo eléctrico disperso $E_p^s(r)$ de las N_i incidencias. Si el operador ψdenota la resolución del problema directo, la ecuación no lineal se puede expresar así:

$$E_p^s(r) = \psi_p(\varepsilon_r) \quad (4)$$

Al no tener una solución exacta esta ecuación cuando hay presencia de ruido, se suele resolver una función objetivo:

$$Min: f(\varepsilon_r) = \sum_{p=1}^{N_i} \left| \psi_p(\varepsilon_r) - E_p^s \right|^2 + \alpha T(\varepsilon_r), \quad (5)$$

donde $T(\varepsilon_r)$es la regularización utilizada para balancear en ajuste de datos y estabilidad de la solución, αes el coeficiente constante de regularización. Se sabe que (5) es no lineal, no convexa. Se excluyen casos donde las partes con frecuencias altas de los objetos son dominantes, ya que estos provocan la mayor cantidad de ruido y, generalmente, la mayor parte de la información en las imágenes se encuentra en las bandas de frecuencia baja. Además, las redes neuronales son capaces de restablecer estos valores a través del aprendizaje y capacidad de representación.

Para resolver este tipo de problemas no lineales, por lo general, se suelen aplicar métodos numéricos iterativos, pero suelen tomar mucho tiempo y no son aplicables en tiempo real. Es por esto que se utilizan redes neuronales, como las convolucionales, que sirven especialmente para procesamiento de imágenes. Para esto se prepara un set de entrenamiento con imágenes y sus respectivas mediciones de campo eléctrico y permitividad. Aunque se podrían utilizar directamente los valores, se puede simplificar el aprendizaje a través de la utilización de algún método no iterativo previo al entrenamiento de la red neuronal. A continuación, se describe el método propuesto en [1].

Se aplica lo que se llama un esquema de backpropagation (distinto al algoritmo de backpropagation para calcular errores en redes neuronales). Este consiste en asumir que la corriente inducida I^b es proporcional al campo campo eléctrico de dispersión E^s:

$$\overline{I}^b = \widehat{G}_s^H \cdot \overline{E}^s \quad (6)$$

(notación: $\widehat{G}_s^H$ es la conjugada transpuesta de $\widehat{G}_s$)

Se define una función costo a partir de (3.1):

$$F^b(\chi) = \left| \overline{E}^s - \widehat{G}_s \cdot (\chi \cdot \widehat{G}_s^H \cdot \overline{E}^s) \right|^2 \quad (7)$$

Para obtener el mínimo de $F(\chi)$, se requiere que su derivada sea 0, con lo cual se puede hallar una solución para χ:

$$\chi = \frac{(\overline{E}^s)^T \cdot (\widehat{G}_s \cdot (\widehat{G}_s^H \cdot \overline{E}^s))^*}{\left| \widehat{G}_s \cdot (\widehat{G}_s^H \cdot \overline{E}^s) \right|^2} \quad (8)$$

(notación: $(\overline{E}^s)^T$ es la transpuesta de $\overline{E}^s$ y A^* es la conjugada compleja de A)

Lo que permite obtener $\overline{I}^b$ de (6) y, consecuentemente, $\overline{E}^{t,b}$ puede hallarse a partir de (1.1):

$$\overline{E}^{t,b} = \overline{E}^i + \widehat{G}_D \cdot \overline{I}^b \quad (8)$$

Para cada incidencia p, se puede aplicar (2.1):

$$\overline{I}^b = diag(\overline{\xi}^b) \cdot \overline{E}^{t,b} \quad (9)$$

Y combinando todas las incidencias de (9), se reduce a un problema de cuadrados mínimos dado por la fórmula:

$$\overline{\xi}_1^b(n) = \frac{\sum_{p=1}^{N_i} \overline{I}_p^b(n) \cdot \left[\overline{E}_p^{t,b}(n) \right]^*}{\sum_{p=1}^{N_i} \left| \overline{E}_p^{t,b}(n) \right|^2} \quad (10)$$

Objetivos

Los principales objetivos del presente trabajo consisten en:

- Realizar una implementación análoga a la disponible en [8], que permita resolver problemas de dispersión inversa para reconstruir imágenes, utilizando un lenguaje con licencia de código abierto y sobre una plataforma moderna y poderosa para aplicaciones de deep learning como lo es Pytorch.
- Permitir que el software tenga capacidad de procesamiento tanto CPU como GPU, lo que permitirá acelerar el tiempo de cómputo [9].
- Hacer disponible el software para lograr experimentos de mayor escala que los expuestos en [1] con aplicaciones en desarrollos científicos futuros.

Tecnologías

- **Lenguaje**: *Python*
- **Biblioteca para deep learning**: *PyTorch*
- **Biblioteca para análisis y representación de datos**: *Numpy / Matplotlib*

Tomografía de microondas

La tomografía es una técnica que se utiliza para generar una representación interna de un objeto o cuerpo a través de una imagen en una escala de grises o colores. [4]

Hoy en día, existen varios métodos para obtener tomografías. Entre ellos, los más usados son tomografía computada (CT), tomografía por emisión de positrones (PET) y resonancia magnética (MRI). Todos estos son sumamente útiles para obtener información de los tejidos como hipoxia, metabolismo, isquemia y malignidades. Pero, a su vez, presentan sus desventajas, como lo son los elevados precios de sus equipos y la radiación que emiten y afectan a los individuos.[2]

Fig. 2: Escáner PET/CT [3]

Por otra parte, la tomografía por microondas obtiene imágenes dadas por las permitividades de los tejidos a través de la aplicación de un campo electromagnético. Esto se logra a través de una serie de antenas emisoras y receptoras ubicadas alrededor del cuerpo del cual se quiere tomar la tomografía. Estas antenas captan información del campo eléctrico y mediante un software, se

obtienen los valores de permitividades. Esto se conoce como problema de dispersión inversa y es muy costoso, computacionalmente hablando. [4] [1]

Dado el avance tecnológico de los últimos tiempos, empezando por las telecomunicaciones (móviles), abaratando los costos de las antenas, y de la computación a través de deep learning y el procesamiento de datos, haciendo posible la generación de imágenes en menor tiempo, hacen de esta técnica una asequible y accesible. Sumando a esto que el campo electromagnético utilizado es no ionizante, contrario a la medicina nuclear y las tomografías computadas que utilizan radiación ionizante, hacen que la utilización de métodos de tomografía por microondas sean menos invasivos y potencialmente peligrosos para la salud. Se considera que la exposición a microondas es comparable a la generada por la utilización de un teléfono celular. [2]

Ahora bien, el problema yace en encontrar un software capaz de generar las imágenes de permitividades correctamente y de manera rápida ya que, tradicionalmente, los problemas de dispersión inversa se resuelven mediante métodos iterativos muy costosos computacionalmente hablando. GenPer cumple con ese propósito a través de la utilización de, primero, el preprocesamiento (backpropagation en la sección anterior) para generar una imagen preliminar (difusa) y, luego, la utilización de una red neuronal de tipo U-Net para obtener la imagen final.

Red neuronal U-Net

Usualmente, para tareas de deep learning como segmentación de imágenes, se utilizan redes neuronales de tipo convolucional. Este tipo de redes basa su eficacia en tener un set de entrenamiento lo suficientemente grande. Para tareas de procesamiento de imágenes biomédicas, se suele necesitar información de cada píxel de la imagen, no solamente asignar una clasificación a cada una. Además, no suelen haber grandes sets de imágenes para entrenar la red. Es por esto que surge la red neuronal de tipo U-Net. Esta cuenta con la característica de ser capaz de hacer una predicción de cada píxel a través de los píxeles adyacentes como entrada. Además, al trabajar sobre pequeñas regiones de cada imagen, el set de entrenamiento se hace más grande. [6]

La arquitectura U-Net consiste de dos caminos, el primero o encoder, es el camino contractivo. Está compuesto por capas de tipo convolucionales que ayudan a obtener el contexto de la imagen, y capas de tipo max pooling que reducen el tamaño (la resolución) de la imagen. Este camino obtiene información sobre qué hay presente en la imagen. El segundo camino o decoder, es el camino expansivo y es simétrico al anterior. Está compuesto por capas de tipo convolucional para obtener nuevamente el contexto de la imagen y capas de tipo convolucional traspuesta, que ayudan a situar ese contexto en la imagen de mayor definición. [7] Además, las capas de tipo convolucional y

convolucional traspuesta suelen estar acompañadas de capas de tipo batch normalization, que sirve para estandarizar las entradas de cada capa, y ReLU, que introduce la no linealidad en la red.

En el caso de GenPer, la estructura de la red es como se puede ver en la siguiente figura:

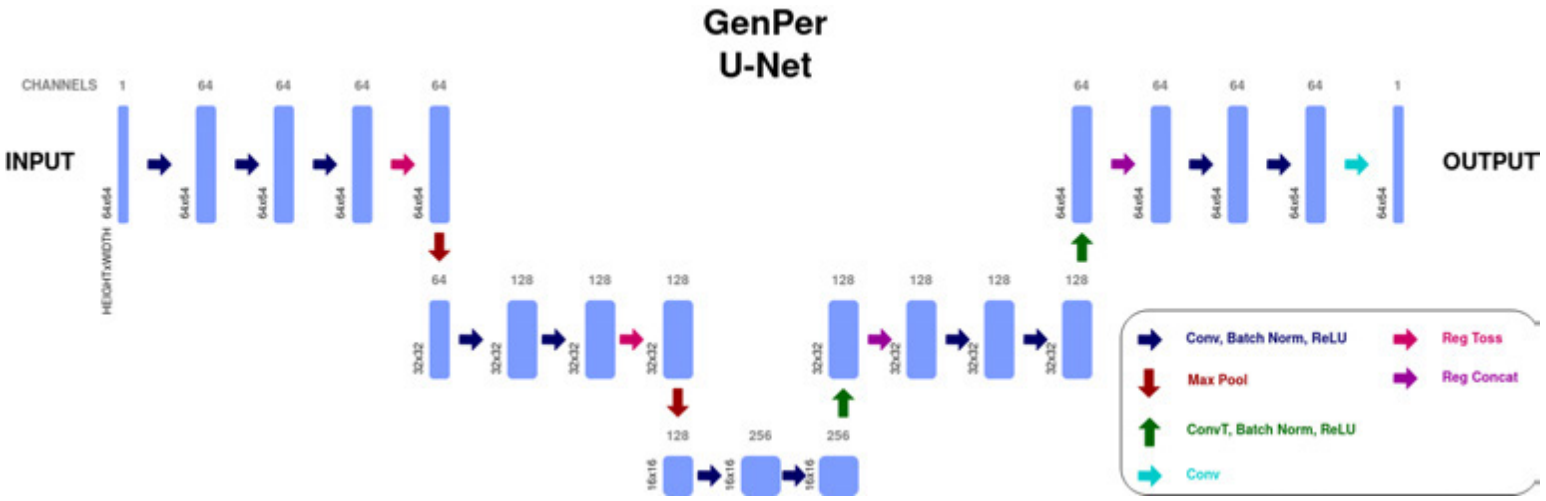

Fig. 3: Arquitectura de la red neuronal UNet de GenPer

La red neuronal tiene 46 capas en total y 3 niveles. Por cada nivel, hay por lo menos dos bloques de capas de convolución, batch normalization y ReLU. Para descender en la red, se utilizan capas de tipo Max pooling, mientras que para ascender, bloques de convolución traspuesta, batch normalization y ReLU. A su vez, en el camino descendente, se utilizan dos capas customizadas de tipo register toss, que guardan información en un registro. Esta información es obtenida nuevamente luego en el camino ascendente en las capas de tipo register concatenation.

A medida que se desciende en la red, aumenta la cantidad de canales, lo cual ayuda a obtener detalles más complejos de la imagen. Lo contrario resulta con las dimensiones de las imágenes, las cuales reducen su definición. En el camino ascendente, estas operaciones son revertidas, obteniéndose finalmente una salida con las mismas dimensiones de la entrada.

Funcionamiento

El software consta de 4 comandos básicos. El primero, genera un número customizable de imágenes de 64 x 64 con una cantidad aleatoria de círculos con distintos valores de permitividad (también parametrizable). Las imágenes obtenidas son similares a las siguientes:

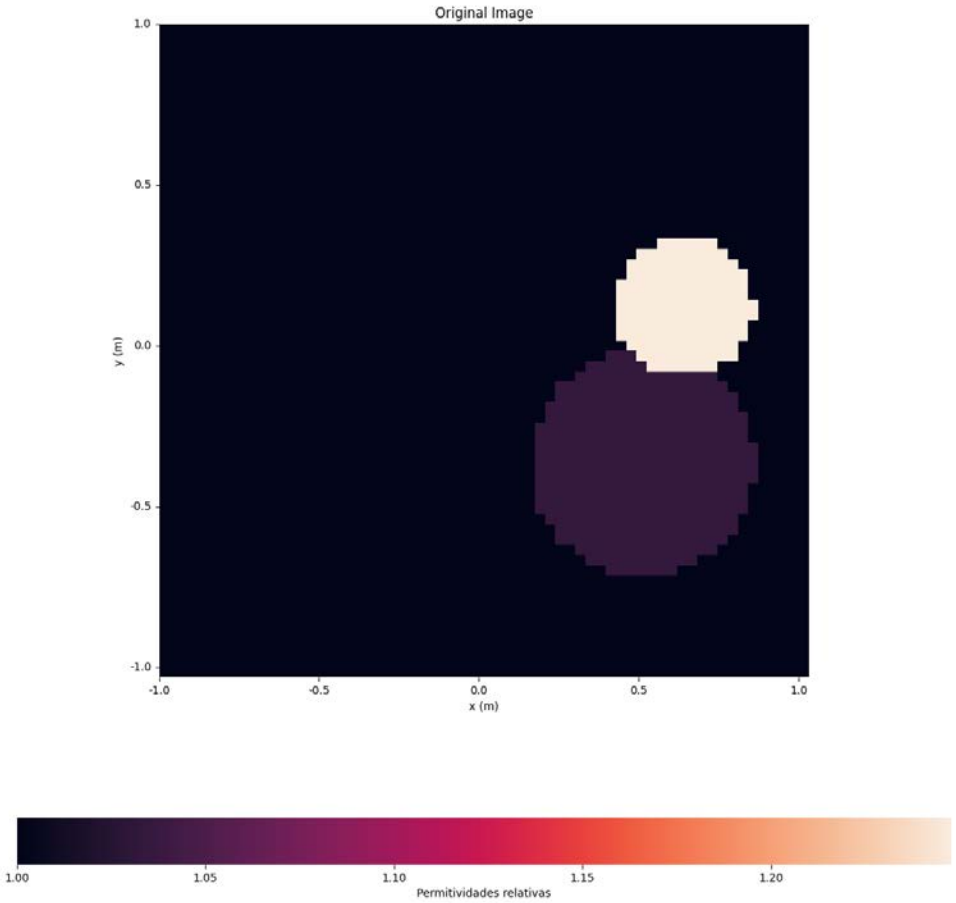

Fig.4: Imagen generada por GenPer con 2 círculos de diferente permitividad

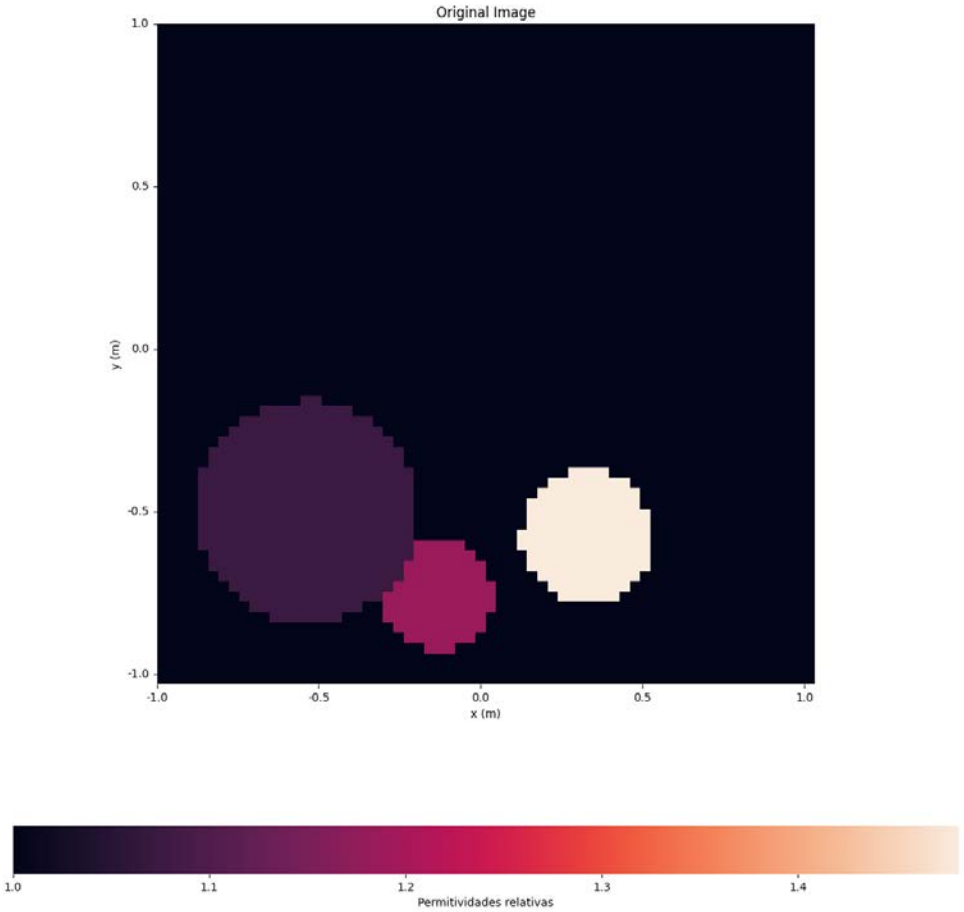

Fig.5: Imagen generada por GenPer con 3 círculos de diferente permitividad

De estas, a su vez, se obtiene el campo eléctrico emitido por 16 antenas en 32 receptores. Estos valores serán los que usará el GenPer para poder obtener las imágenes originales.

El segundo comando es el preprocesamiento. Como entrada tiene los datos de campo eléctrico obtenidos de cada imagen y a través de este se obtiene una imagen preliminar similar a la original, pero mucho más difusa.

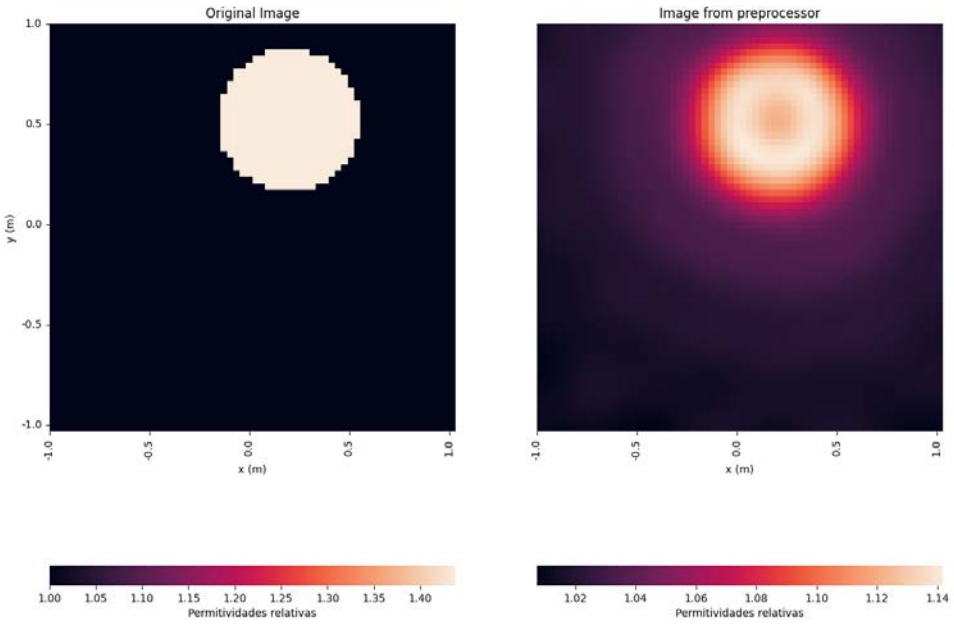

Fig. 6: A la izquierda, la imagen original con 1 círculo, a la derecha, la imagen obtenida por el preprocesador

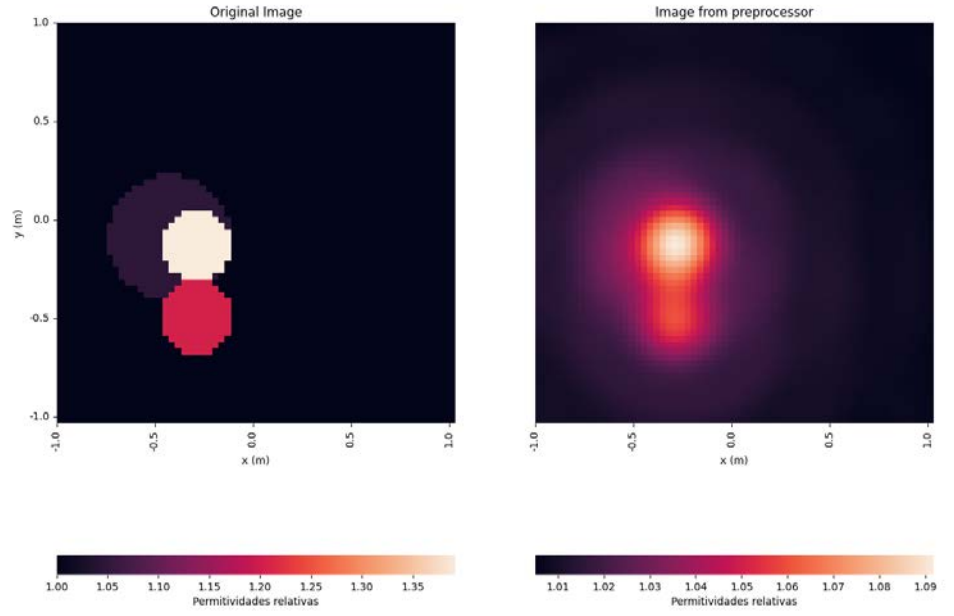

Fig. 7: A la izquierda, la imagen original con 3 círculos, a la derecha, la imagen obtenida por el preprocesador

El tercer comando es para entrenar la red neuronal. Esta, en lugar de tener los valores de campo eléctrico por entrada, tiene a la imagen preprocesada. El set de datos es dividido en 3 partes con proporciones configurables, obteniéndose un set de entrenamiento, un set de validación y un set de testing. El entrenamiento tiene como parámetros configurables el número de iteraciones y el

tamaño del batch. El mismo utiliza un algoritmo de AdamW para optimizar y el error se calcula mediante el error cuadrático medio.

El último comando sirve para hacer el testing de la red neuronal previamente entrenada con el set de testing. Se obtiene un valor para el error total del set, el error medio y el desvío estándar.

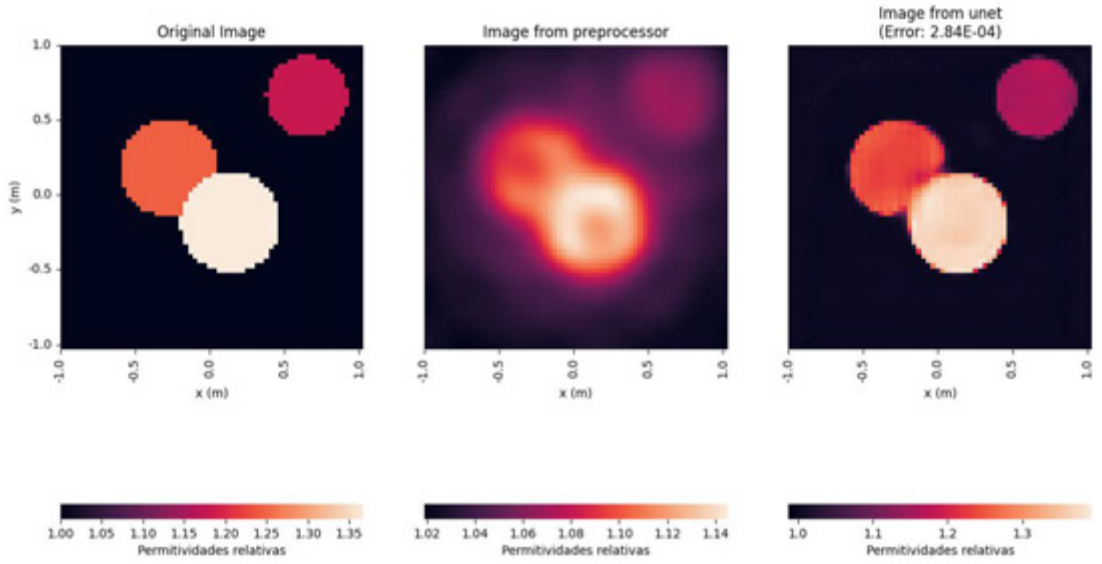

Fig. 8: A la izquierda, la imágen original, en el centro, la imágen preprocesada y a la derecha la salida de la red neuronal con su correspondiente valor de error.

Resultados experimentales

Ejecución básica

Se realizaron varias ejecuciones con distintos parámetros para observar la variación del error según el número de imágenes, tamaño del batch, el número de iteraciones, las proporciones del set de datos y algunos parámetros físicos (como el rango de permitividades). A continuación, se agruparon los resultados experimentales según el parámetro modificado.

Cantidad de imágenes

Se ejecutó GenPer variando la cantidad de imágenes desde 100 hasta 10.000 con un tamaño de batch de 8.

Ejecución 1

Parámetro	Valor

Imágenes	100
Imágenes por batch	8
Iteraciones	50
Set de entrenamiento	36 (70 %)
Set de validación	7 (15 %)
Set de testing	7 (15 %)
Rango de permitividades	1 - 1,5

Tabla 1: Parámetros de ejecución 1

Métrica	**Valor**
Duración del entrenamiento	0:06:49
Error total	$4,95\ x\ 10^{-1}$
Error medio	$4,95\ x\ 10^{-1}$
Desvío estándar	0

Tabla 2: Resultados de ejecución 1

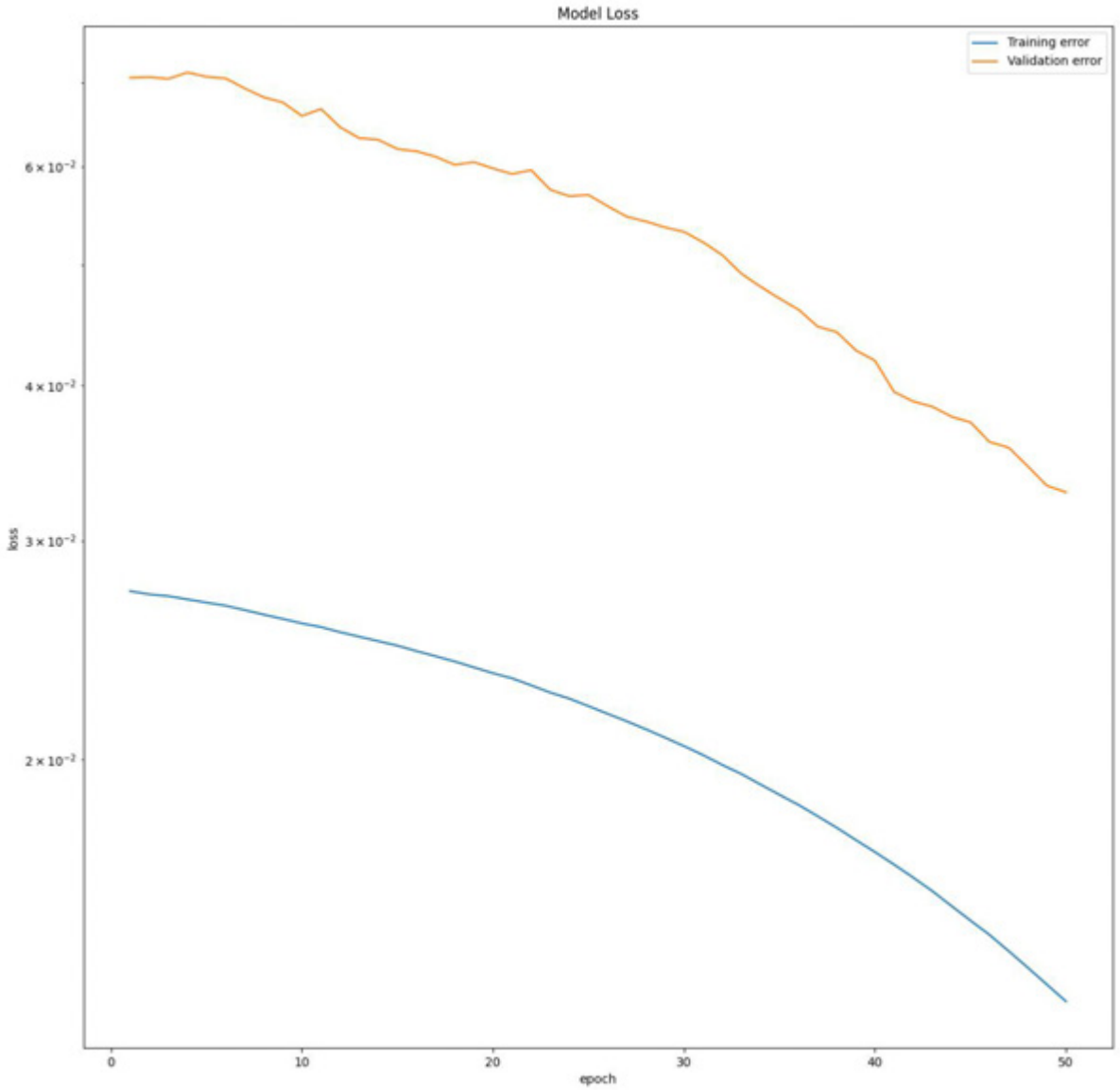

Fig. 9: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 1

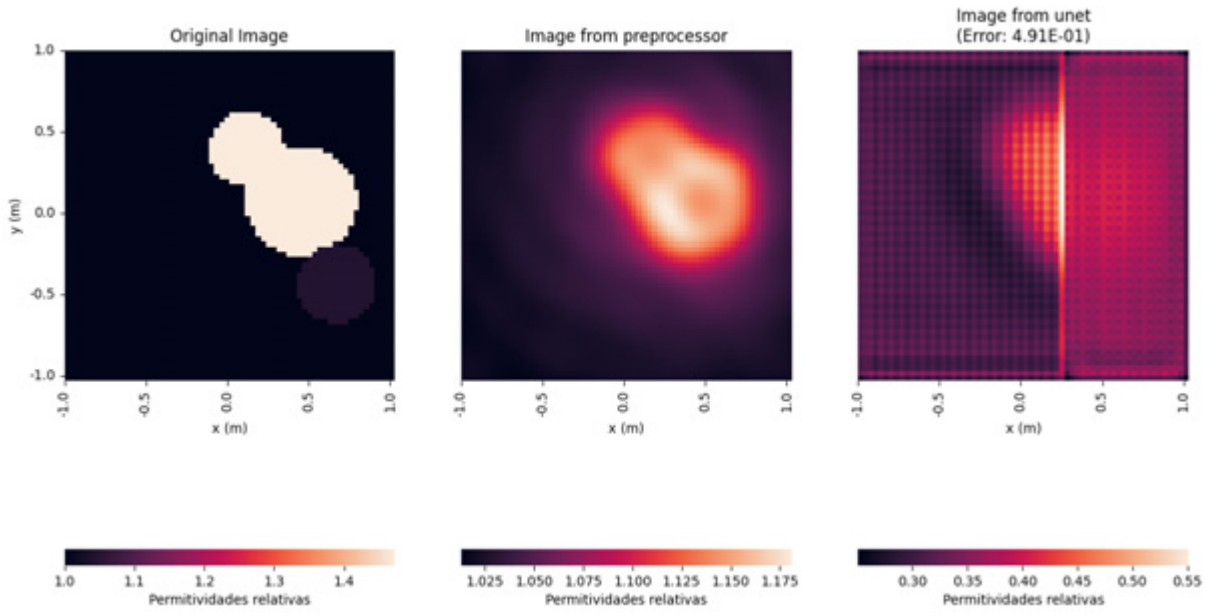

Fig. 10: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 1

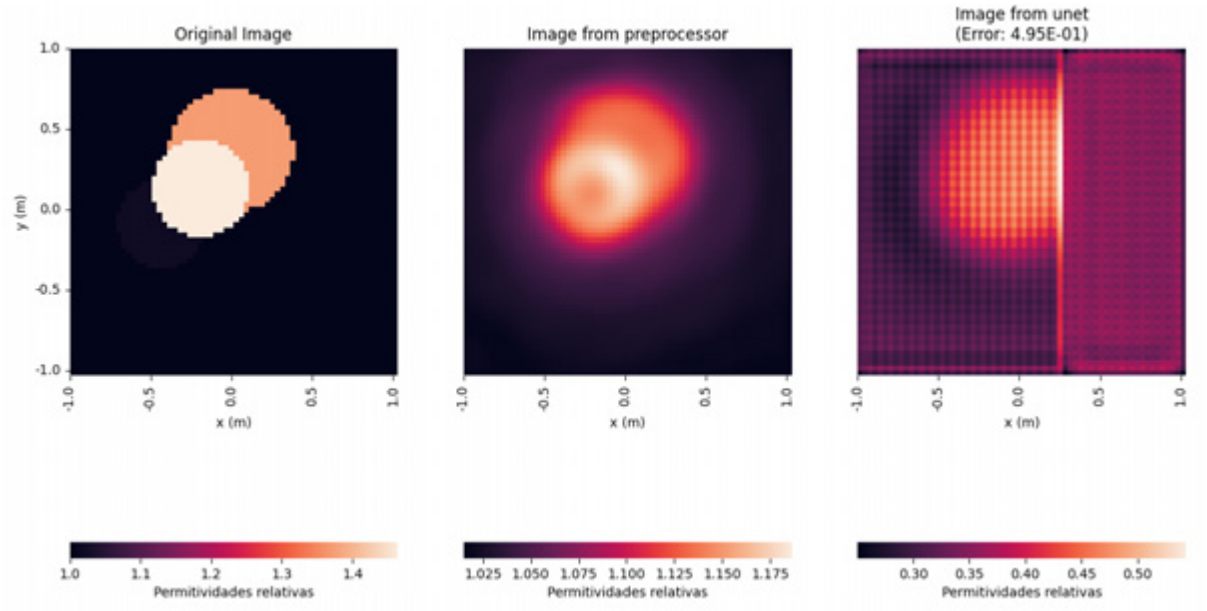

Fig. 11: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 1

Ejecución 2

Parámetro	Valor
Imágenes	500
Imágenes por batch	8
Iteraciones	50

Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 3: Parámetros de ejecución 2

Métrica	**Valor**
Duración del entrenamiento	0:27:58
Error total	$7,30\ x\ 10^{-3}$
Error medio	$8,11\ x\ 10^{-4}$
Desvío estándar	$3,52\ x\ 10^{-4}$

Tabla 4: Resultados de ejecución 2

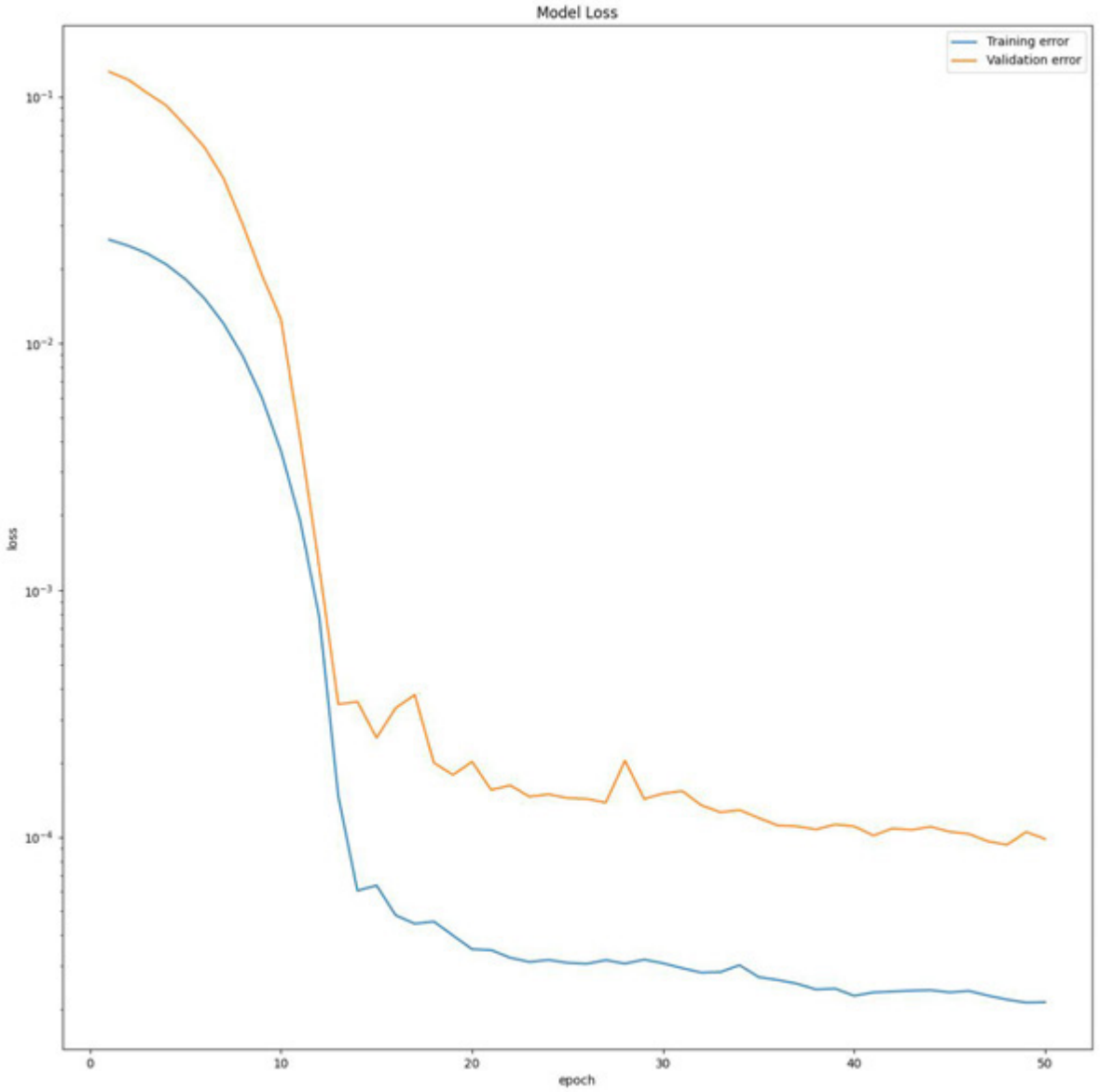

Fig. 12: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 2

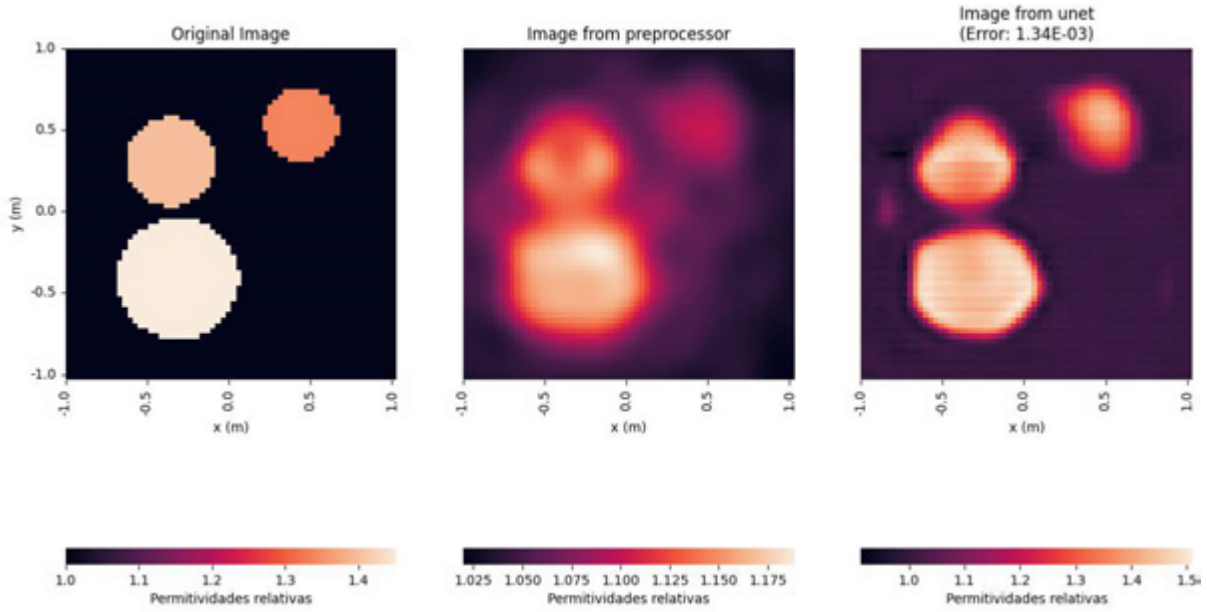

Fig. 13: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 2

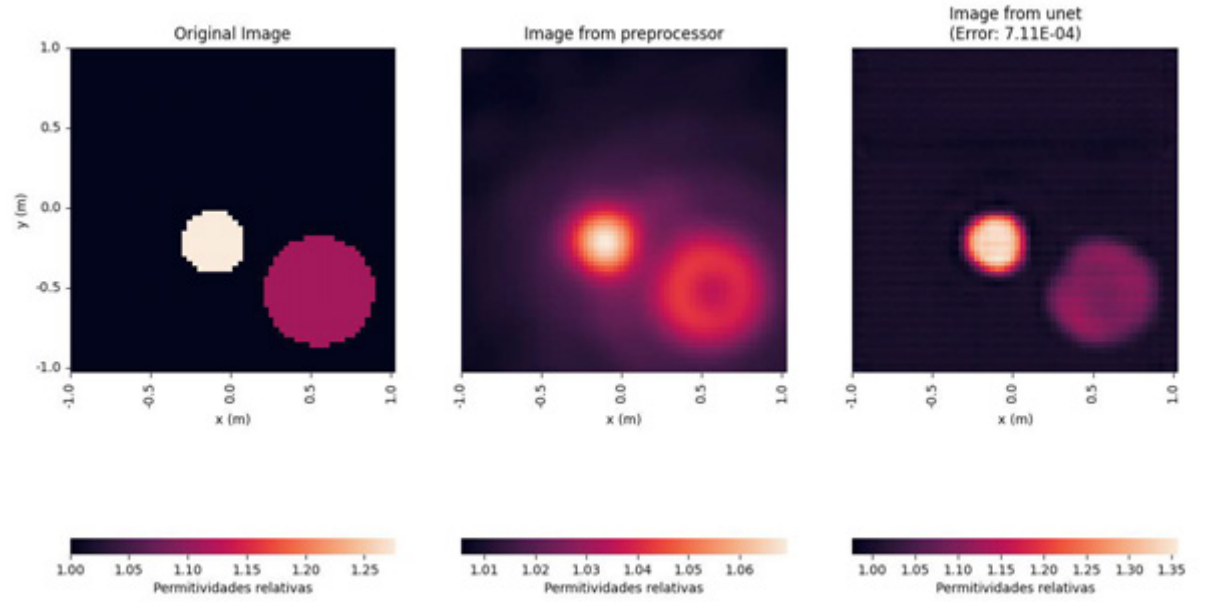

Fig. 14: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 2

Ejecución 3

Parámetro	Valor
Imágenes	1.500
Imágenes por batch	8
Iteraciones	50
Set de entrenamiento	1050 (70 %)

Set de validación	225 (15 %)
Set de testing	225 (15 %)
Rango de permitividades	1 - 1,5

Tabla 5: Parámetros de ejecución 3

Métrica	**Valor**
Duración del entrenamiento	1:12:23
Error total	$1,18\ x\ 10^{-2}$
Error medio	$4,22\ x\ 10^{-4}$
Desvío estándar	$1,54\ x\ 10^{-4}$

Tabla 6: Resultados de ejecución 3

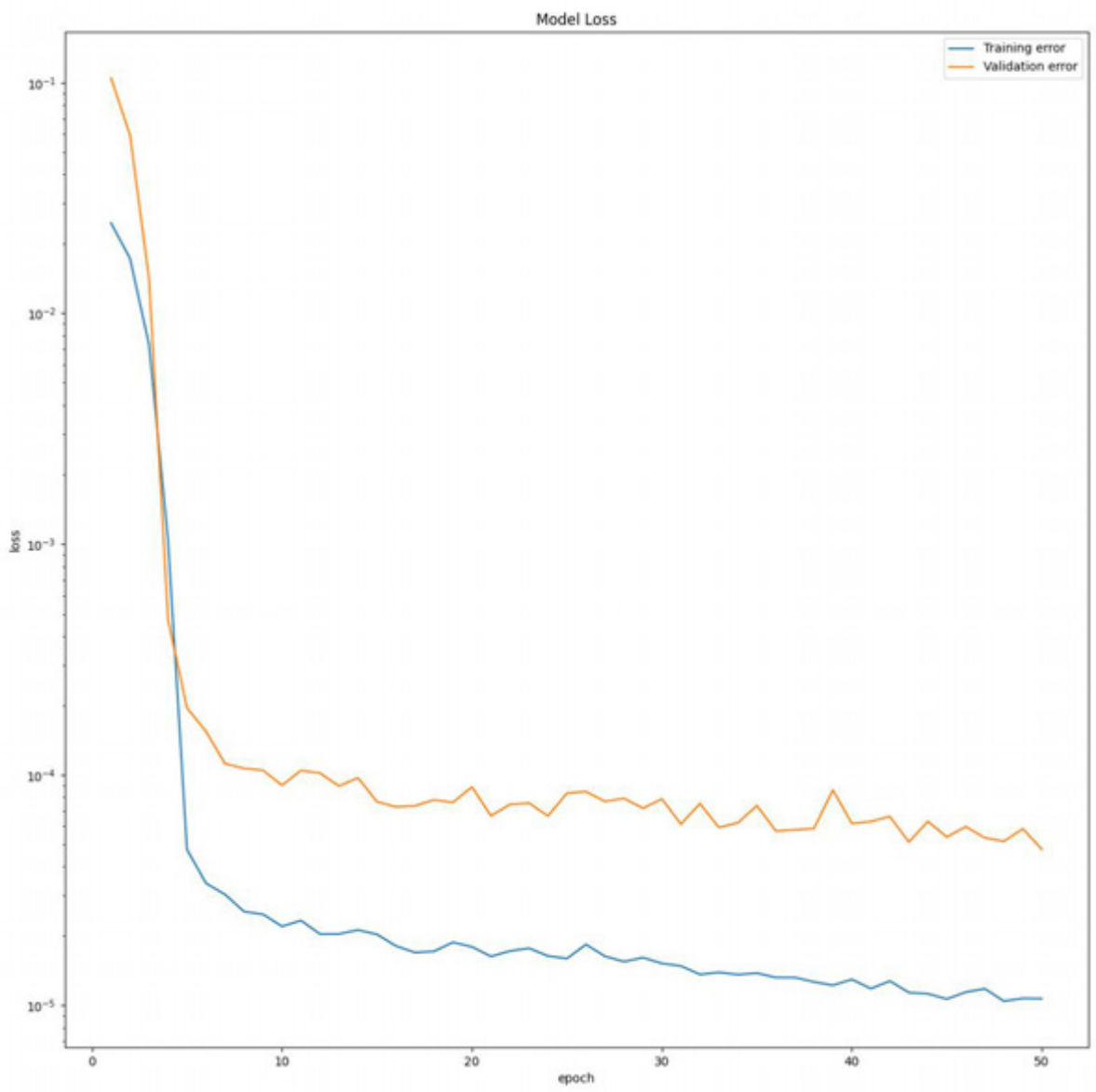

Fig. 15: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 3

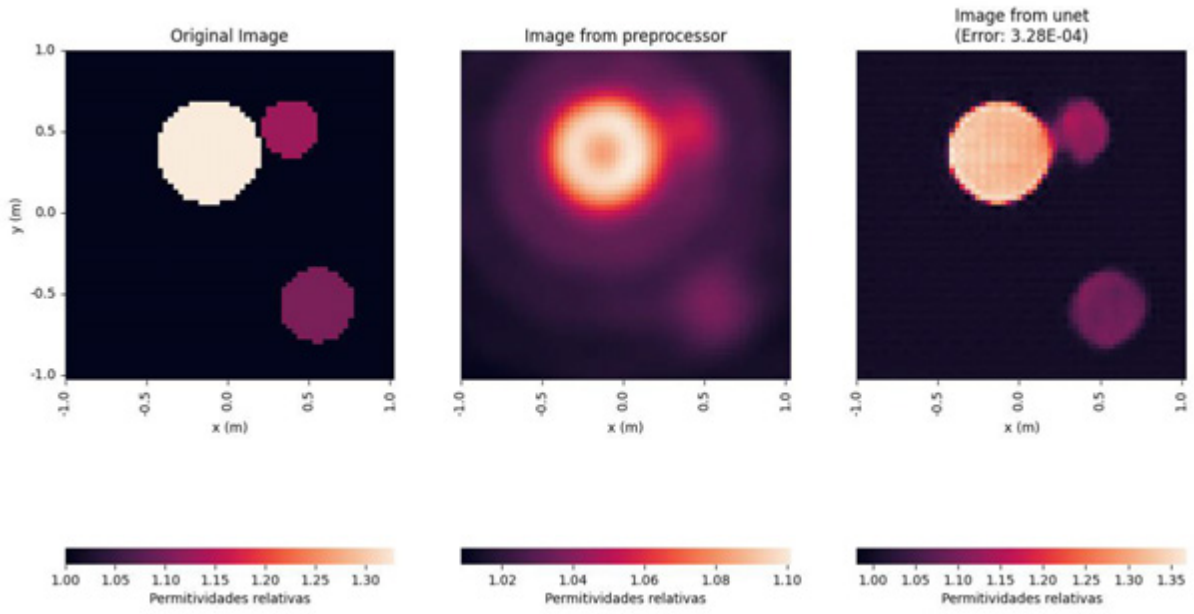

Fig. 16: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 3

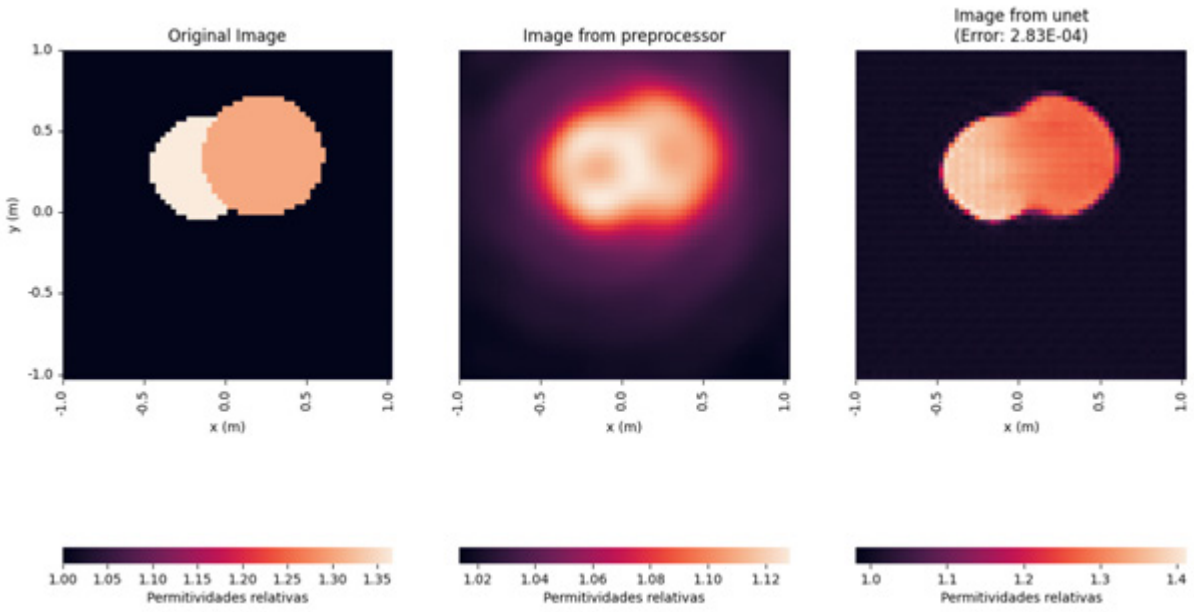

Fig. 17: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 3

Ejecución 4

Parámetro	**Valor**
Imágenes	5.000
Imágenes por batch	8
Iteraciones	50
Set de entrenamiento	3.500 (70 %)

Set de validación	750 (15 %)
Set de testing	750 (15 %)
Rango de permitividades	1 - 1,5

Tabla 7: Parámetros de ejecución 4

Métrica	**Valor**
Duración del entrenamiento	4:09:38
Error total	$2,85\ x\ 10^{-2}$
Error medio	$3,06\ x\ 10^{-4}$
Desvío estándar	$1,13\ x\ 10^{-4}$

Tabla 8: Resultados de ejecución 4

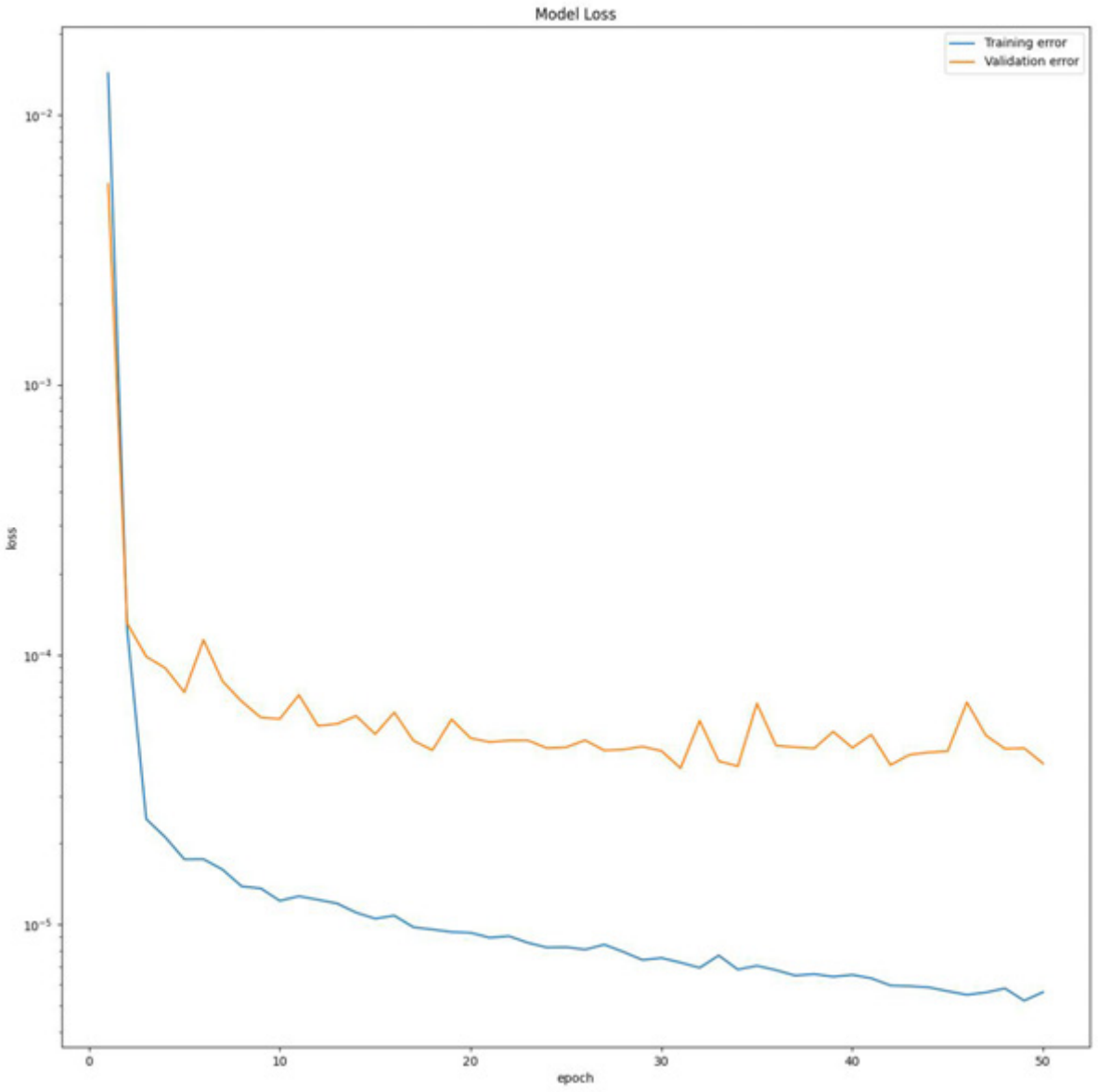

Fig. 18: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 4

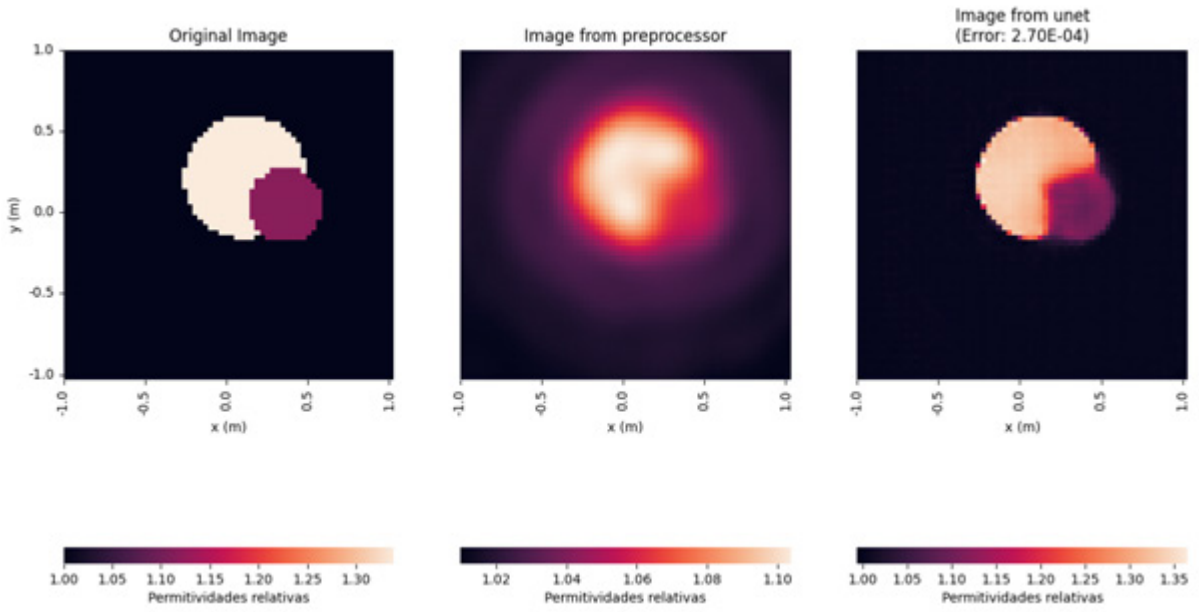

Fig. 19: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 4

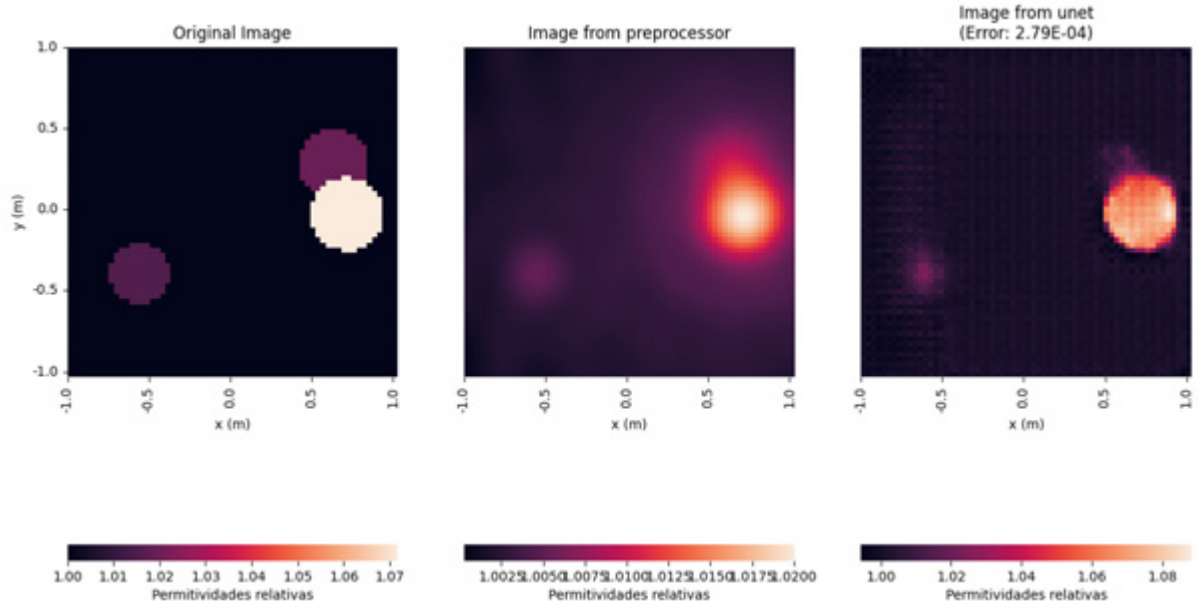

Fig. 20: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 4

Ejecución 5

Parámetro	**Valor**
Imágenes	10.000
Imágenes por batch	8
Iteraciones	50

Set de entrenamiento	7.000 (70 %)
Set de validación	1.500 (15 %)
Set de testing	1.500 (15 %)
Rango de permitividades	1 - 1,5

Tabla 9: Parámetros de ejecución 5

Métrica	**Valor**
Duración del entrenamiento	8:32:31
Error total	$4,58\ x\ 10^{-2}$
Error medio	$2,45\ x\ 10^{-4}$
Desvío estándar	$1,19\ x\ 10^{-4}$

Tabla 10: Resultados de ejecución 5

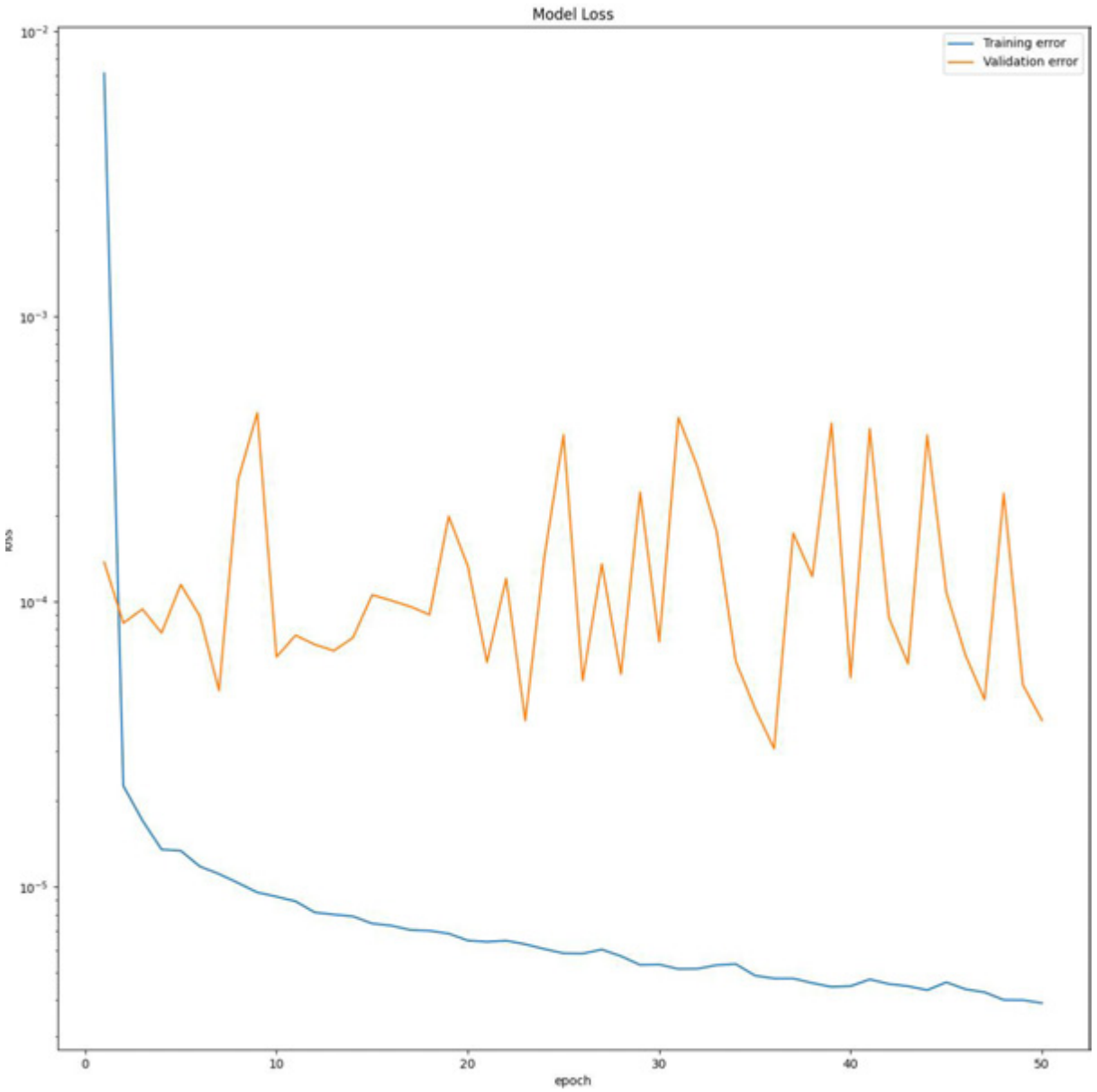

Fig. 21: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 5

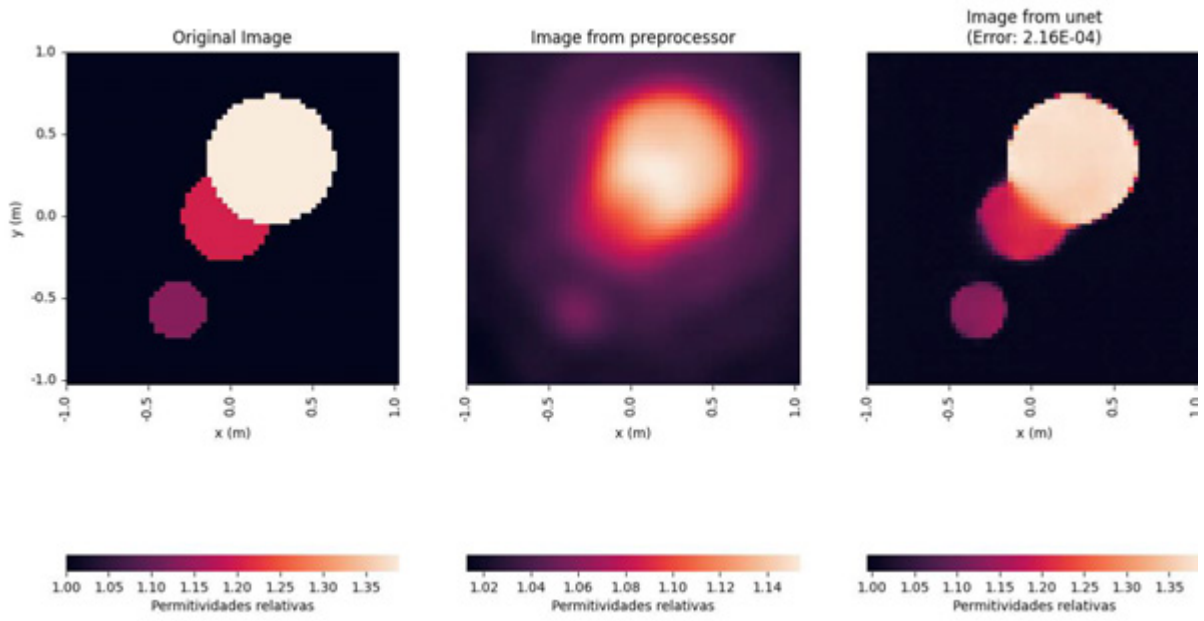

Fig. 22: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 5

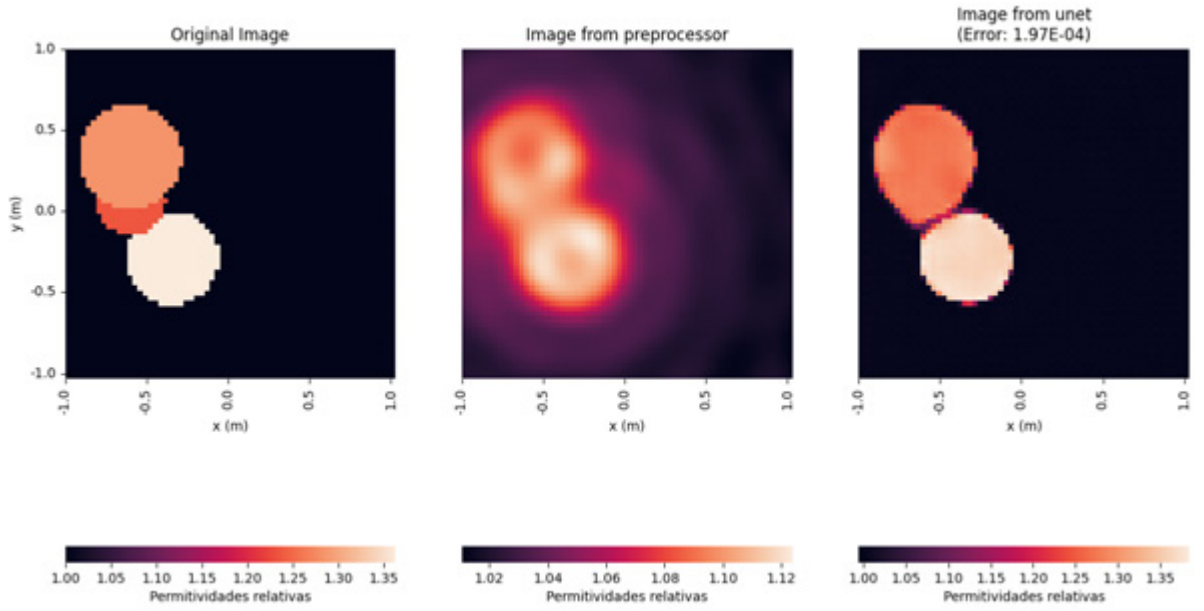

Fig. 23: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 5

Tamaño del batch

Se ejecutó GenPer variando el tamaño del batch desde 1 hasta 32 para 500 imágenes, utilizando potencias de 2 como se acostumbra.

Ejecución 6

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 11: Parámetros de ejecución 6

Métrica	Valor
Duración del entrenamiento	0:41:25
Error total	$4,80\ x\ 10^{-2}$
Error medio	$6,40\ x\ 10^{-4}$
Desvío estándar	$7,92\ x\ 10^{-4}$

Tabla 12: Resultados de ejecución 6

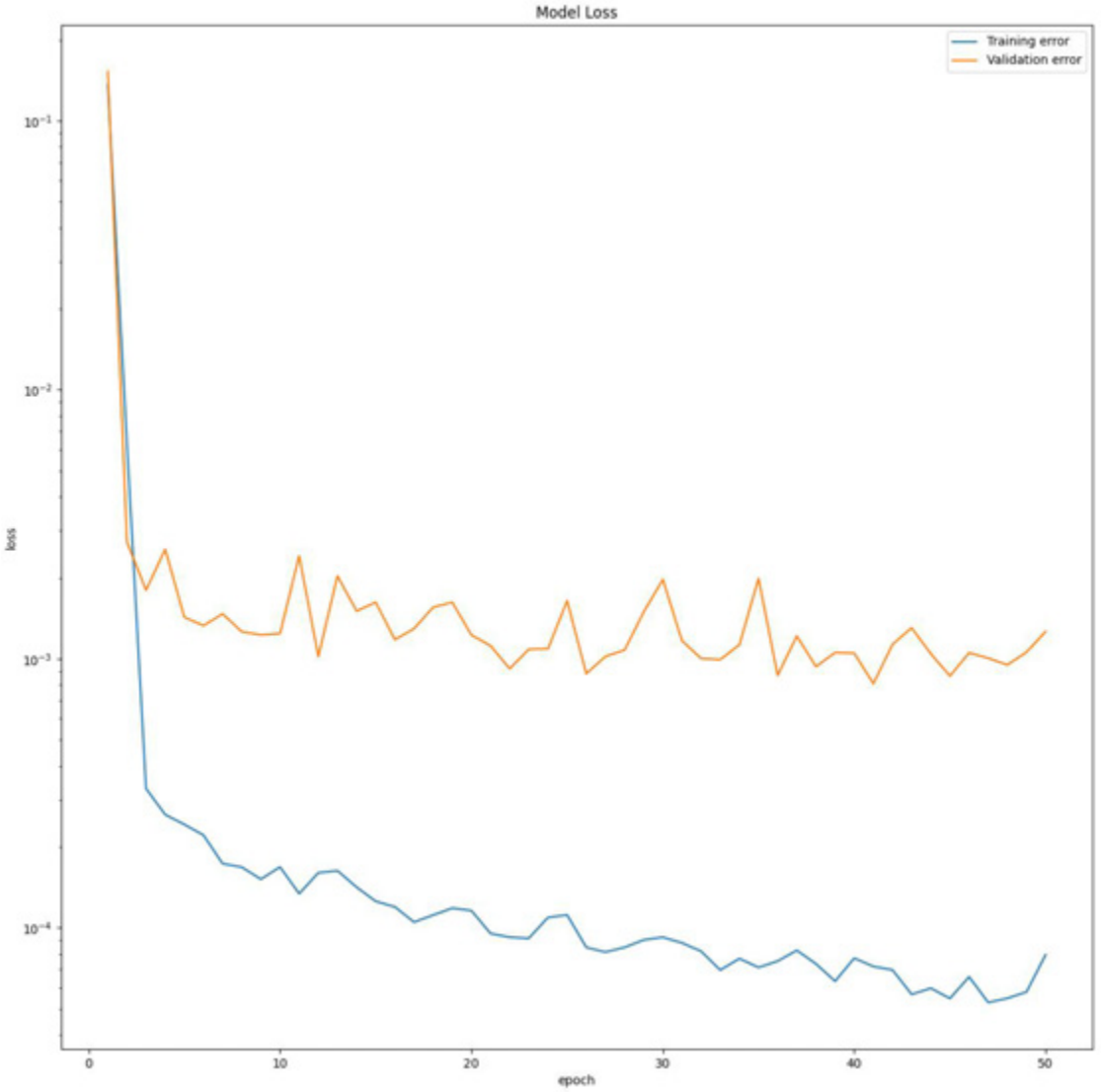

Fig. 24: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 6

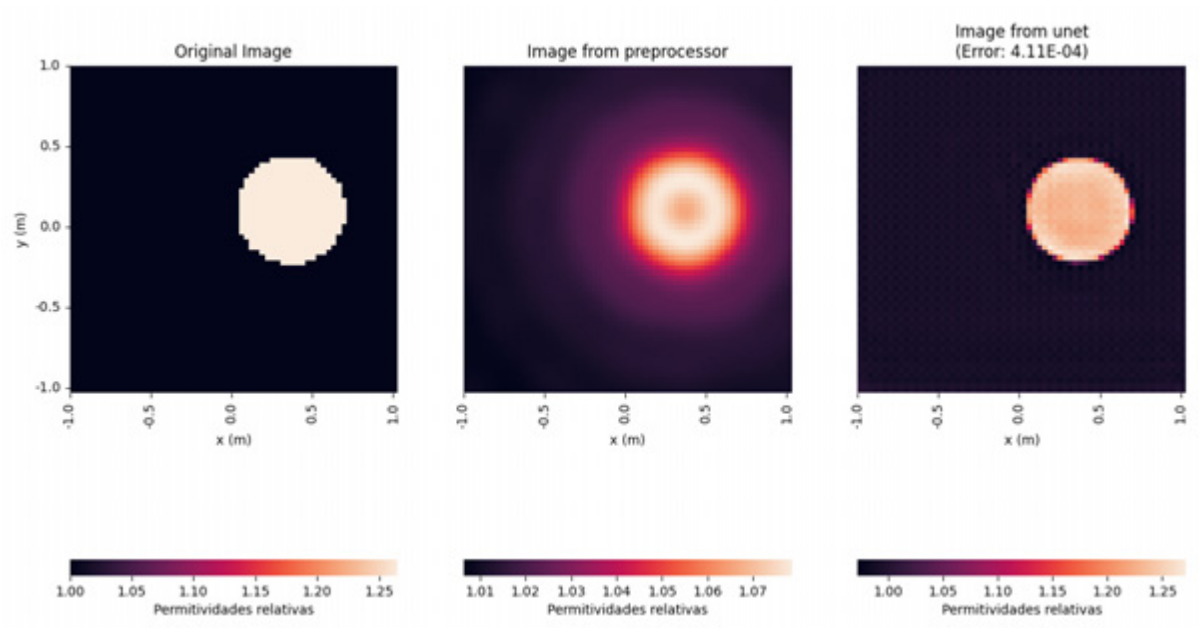

Fig. 25: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 6

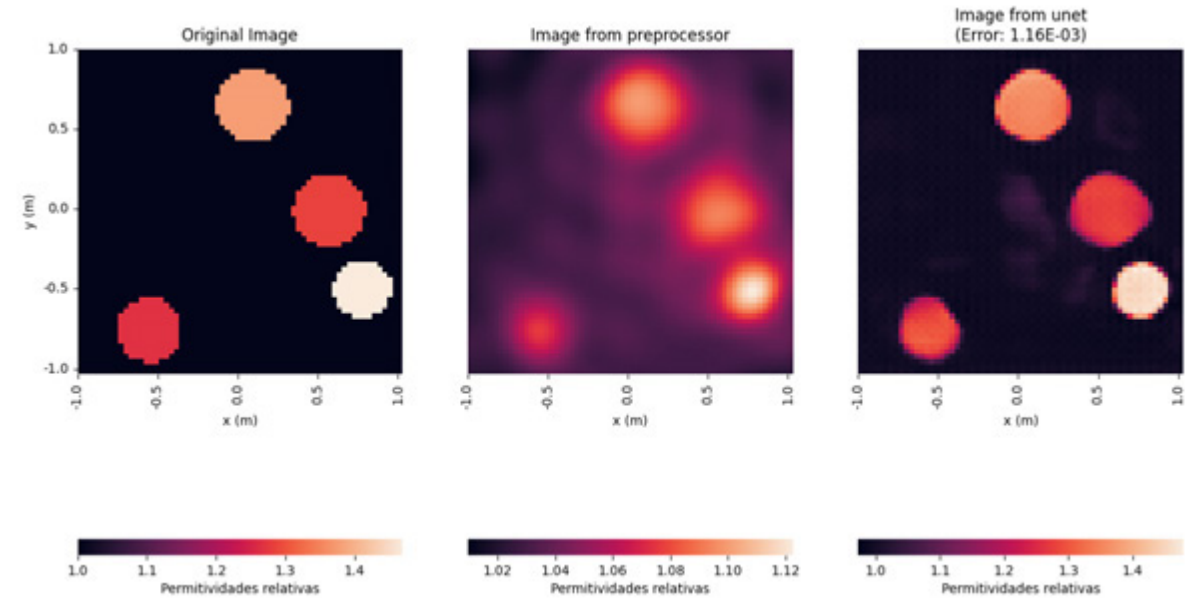

Fig. 26: Resultado del testeo de la red para una imagen de 4 círculos en ejecución 6

Ejecución 7

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	2
Iteraciones	50
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 13: Parámetros de ejecución 7

Métrica	**Valor**
Duración del entrenamiento	0:33:30
Error total	$1,95\ x\ 10^{-2}$
Error medio	$5,27\ x\ 10^{-4}$
Desvío estándar	$3,60\ x\ 10^{-4}$

Tabla 14: Resultados de ejecución 7

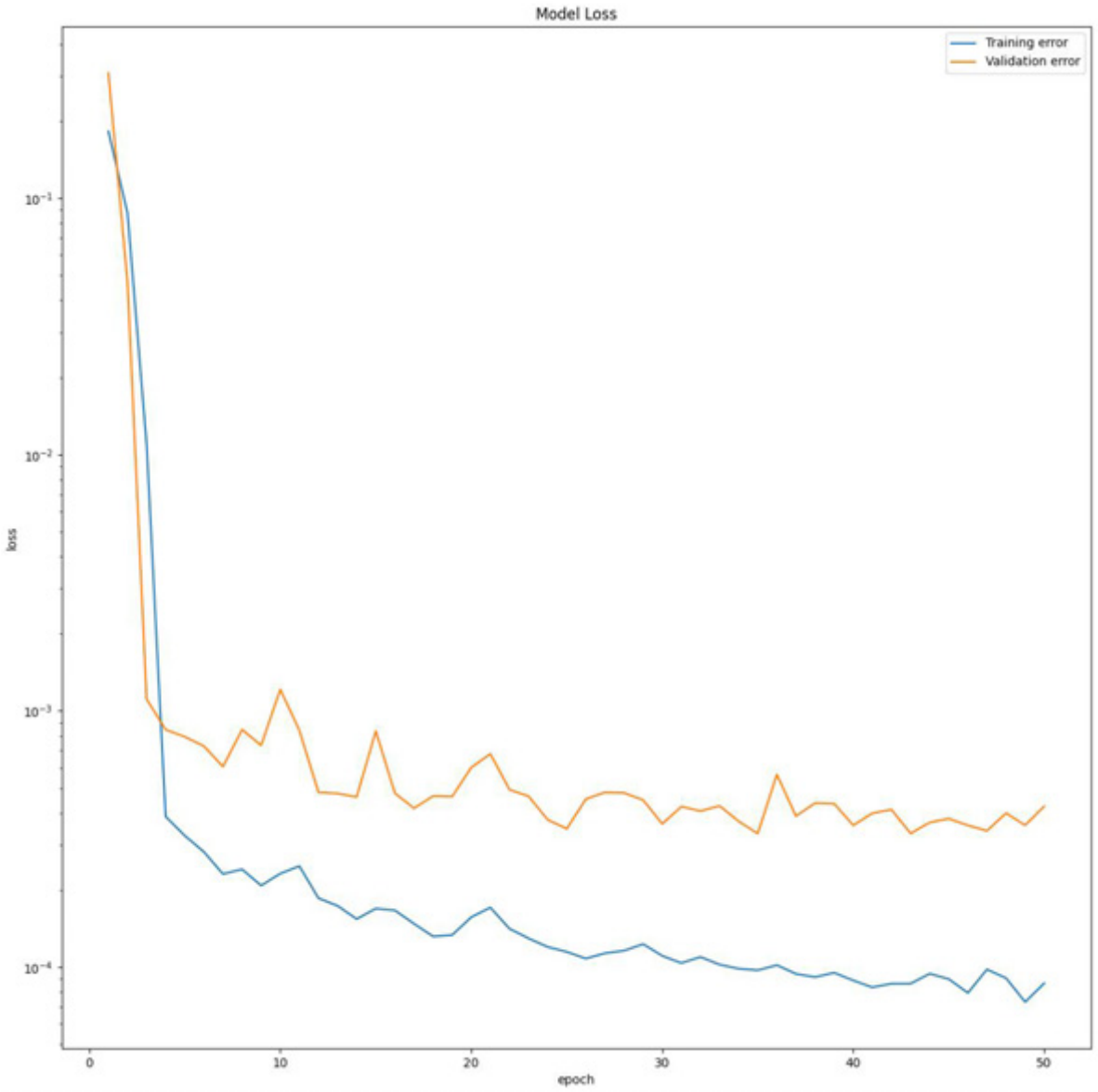

Fig. 27: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 7

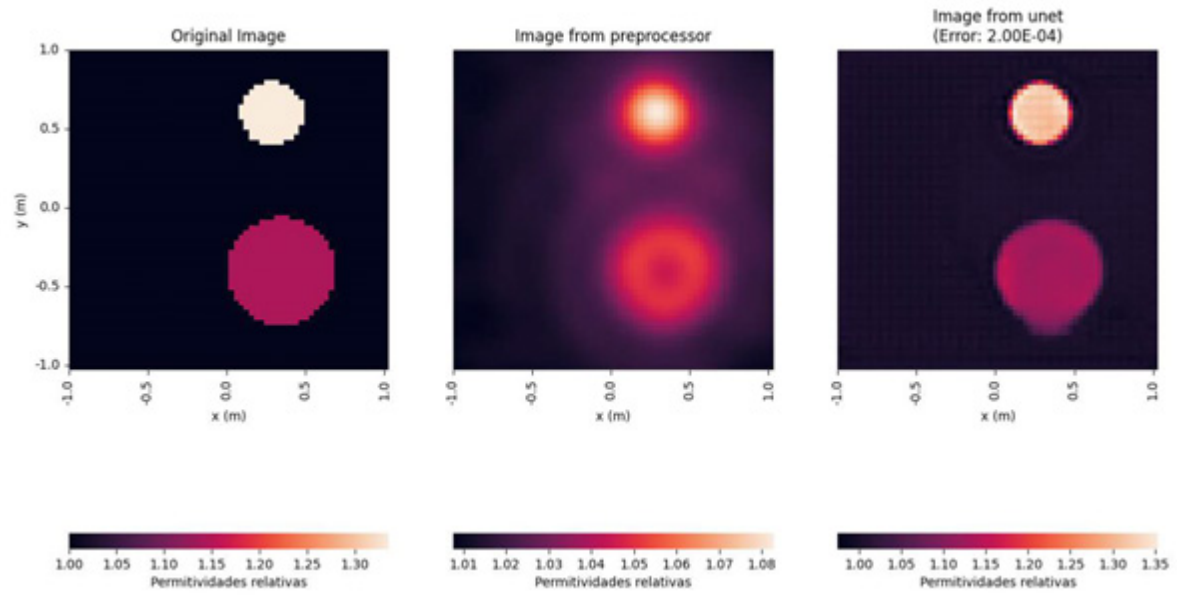

Fig. 28: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 7

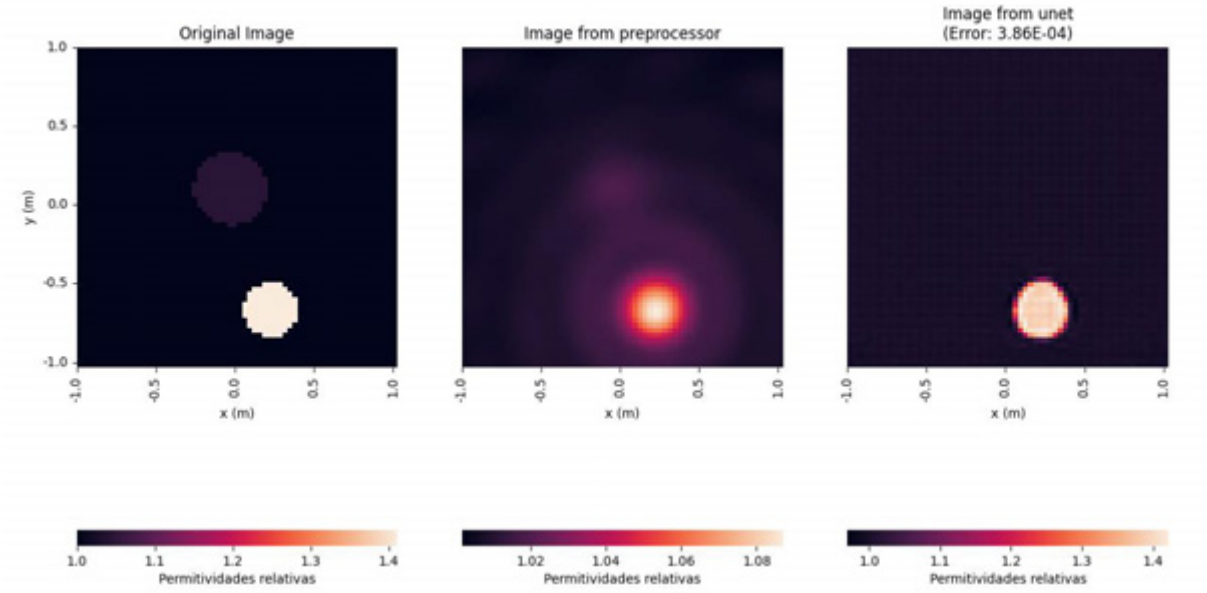

Fig. 29: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 7

Ejecución 8

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	4
Iteraciones	50
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 15: Parámetros de ejecución 8

Métrica	**Valor**
Duración del entrenamiento	0:29:21
Error total	$1,46\ x\ 10^{-2}$
Error medio	$8,11\ x\ 10^{-4}$
Desvío estándar	$4,27\ x\ 10^{-4}$

Tabla 16: Resultados de ejecución 8

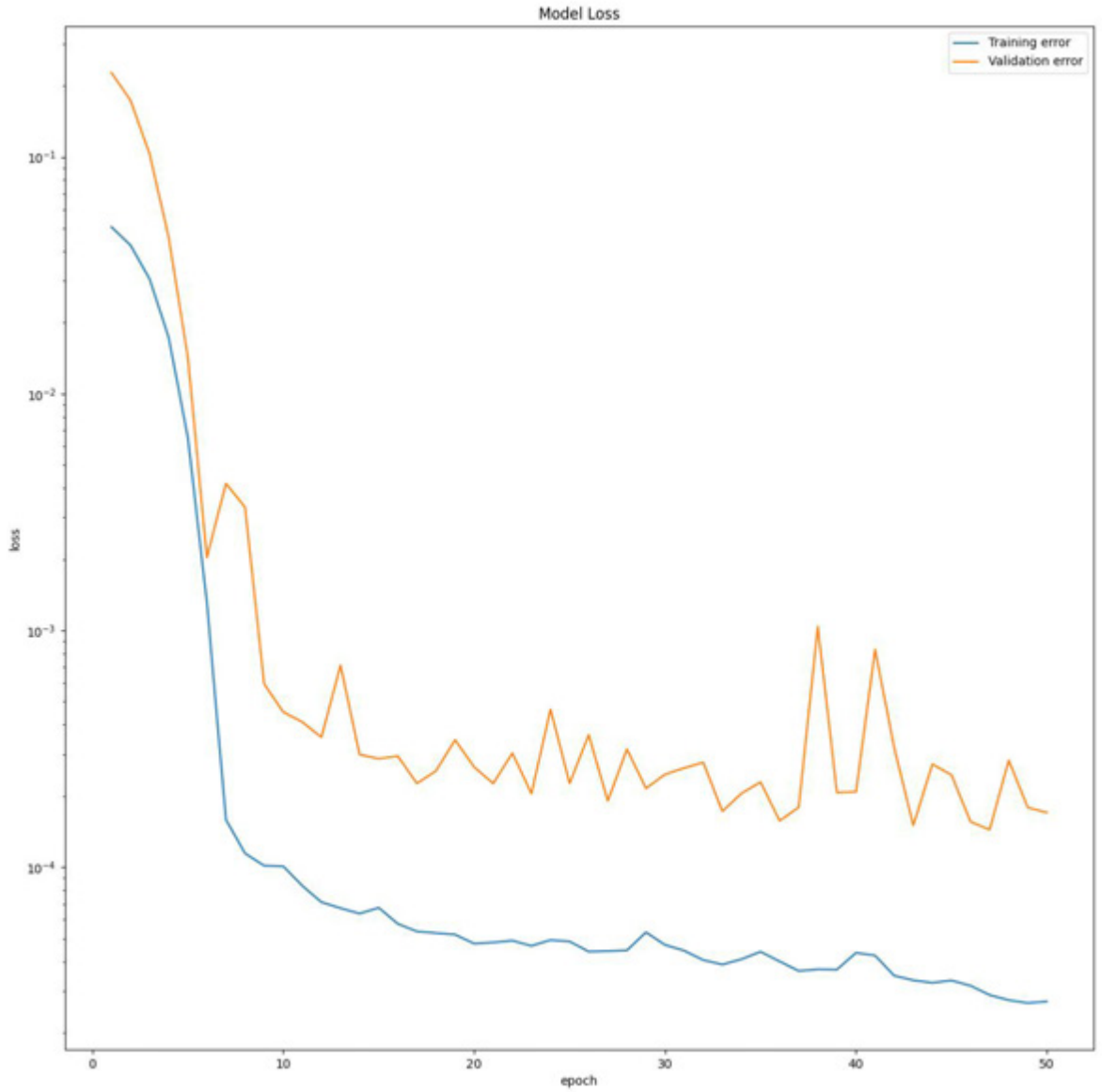

Fig. 30: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 8

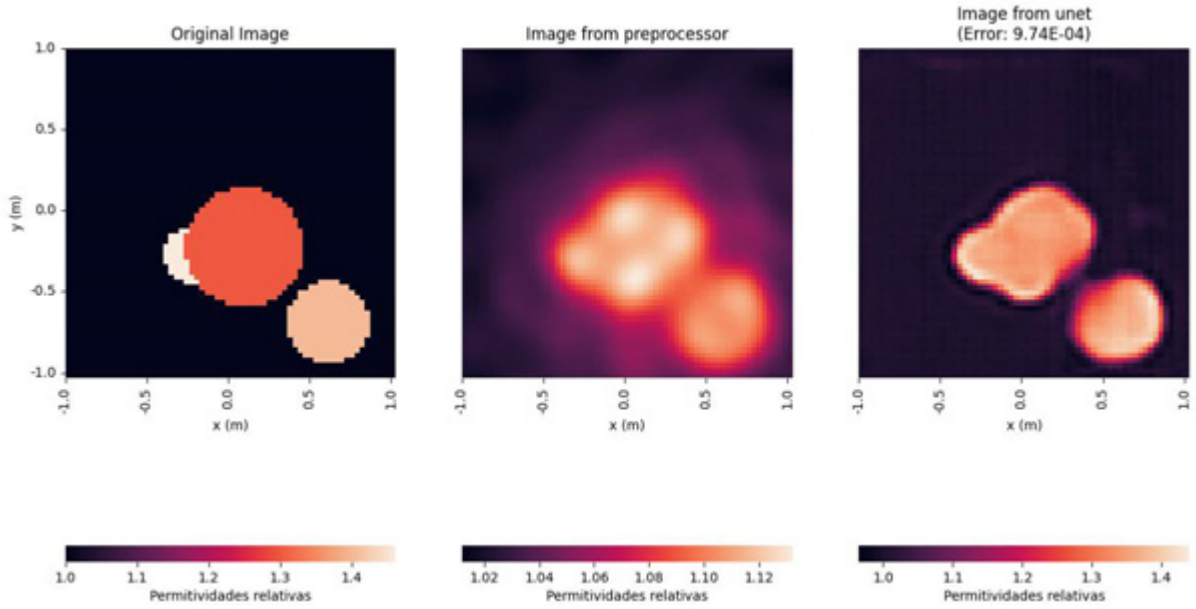

Fig. 31: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 8

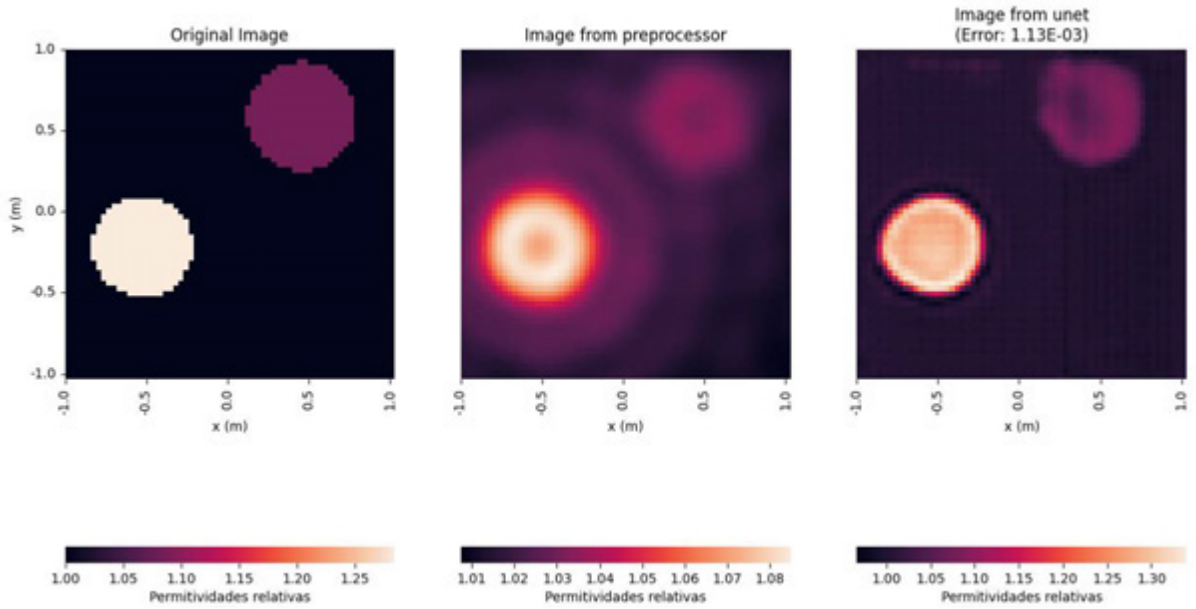

Fig. 32: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 8

Ejecución 9

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	8
Iteraciones	50
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 17: Parámetros de ejecución 9

Métrica	**Valor**
Duración del entrenamiento	0:26:53
Error total	$7,43\ x\ 10^{-3}$
Error medio	$8,26\ x\ 10^{-4}$
Desvío estándar	$4,41\ x\ 10^{-4}$

Tabla 18: Resultados de ejecución 9

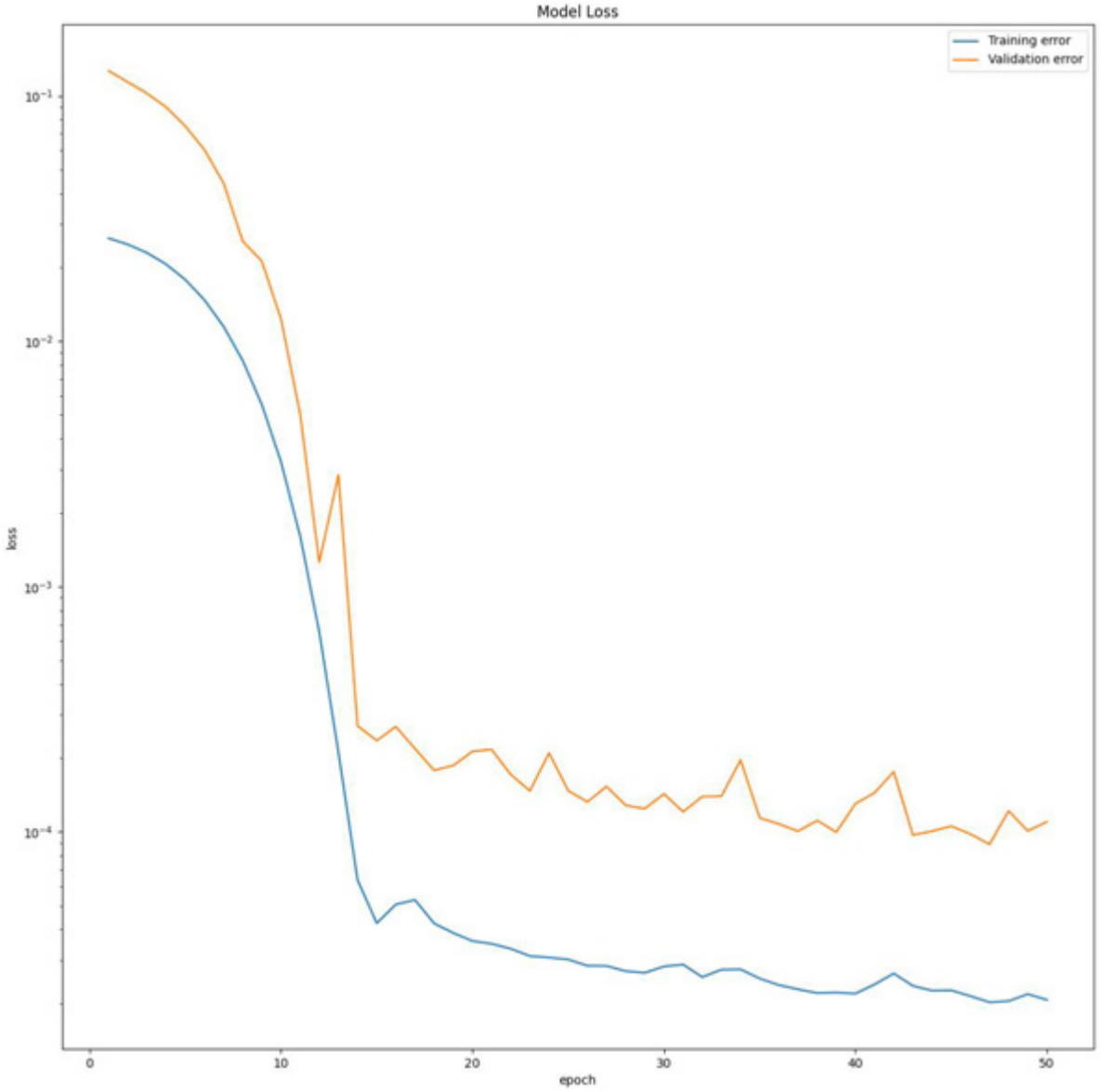

Fig. 33: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 9

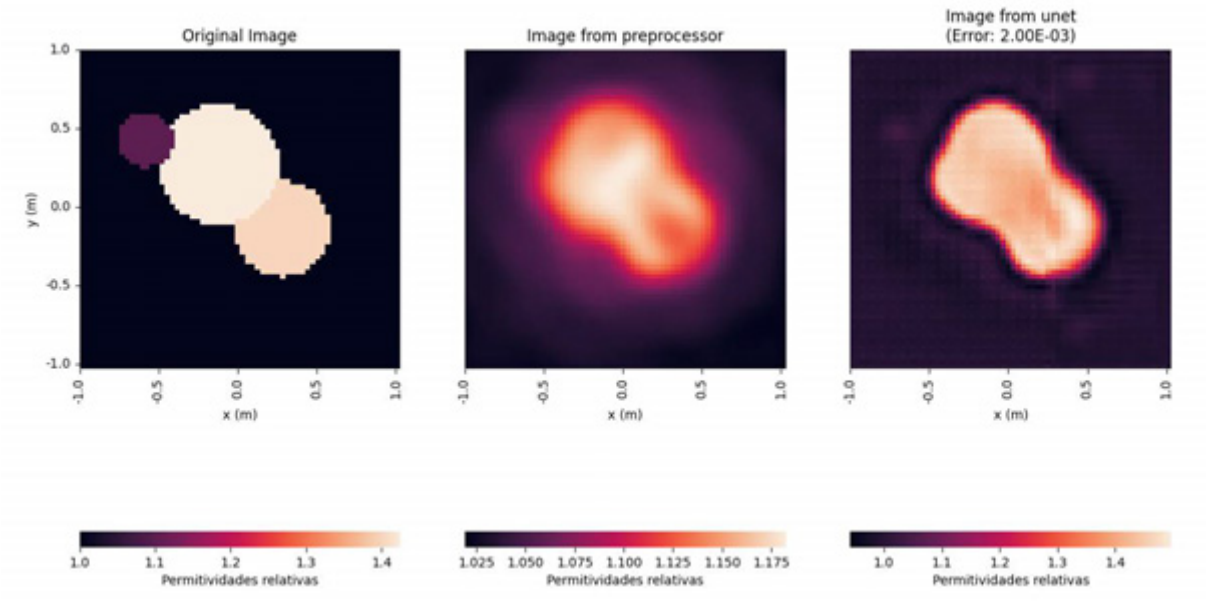

Fig. 34: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 9

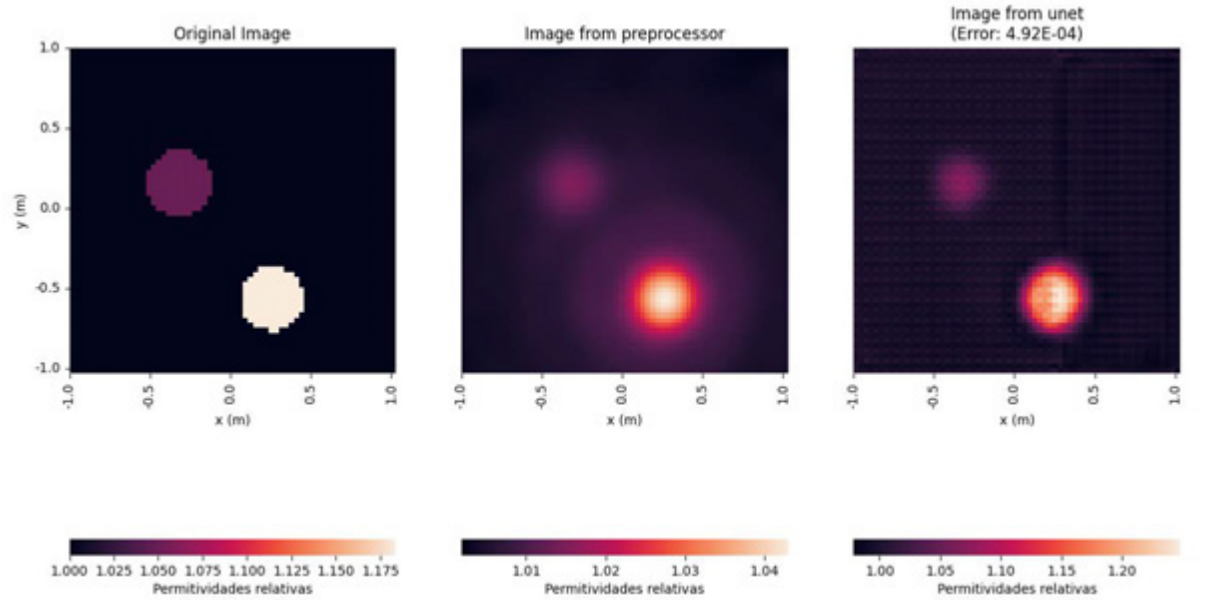

Fig. 35: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 9

Ejecución 10

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	16
Iteraciones	50
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 19: Parámetros de ejecución 10

Métrica	**Valor**
Duración del entrenamiento	0:26:11
Error total	$2,82\ x\ 10^{-3}$
Error medio	$7,04\ x\ 10^{-4}$
Desvío estándar	$1,91\ x\ 10^{-4}$

Tabla 20: Resultados de ejecución 10

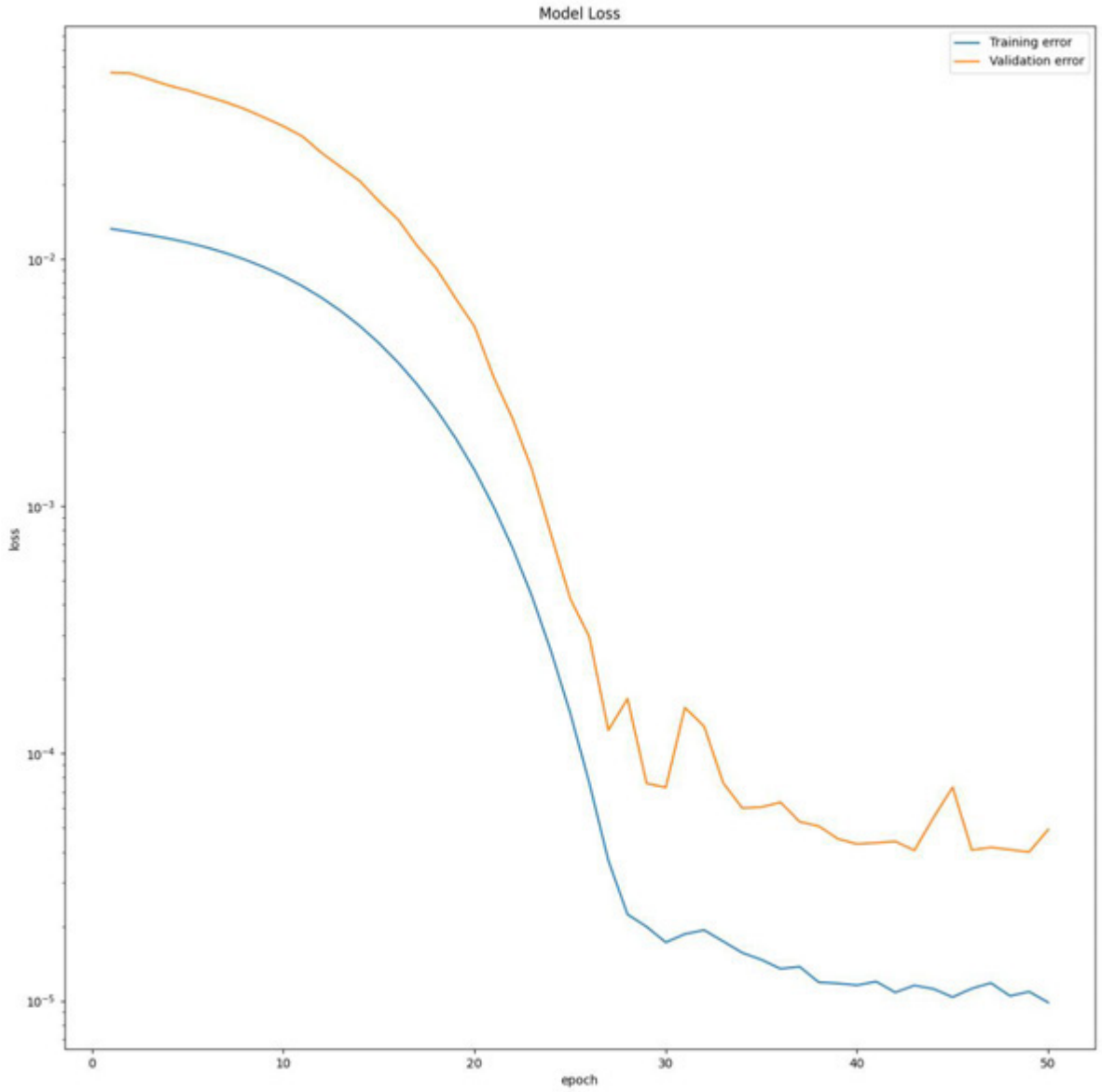

Fig. 36: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 10

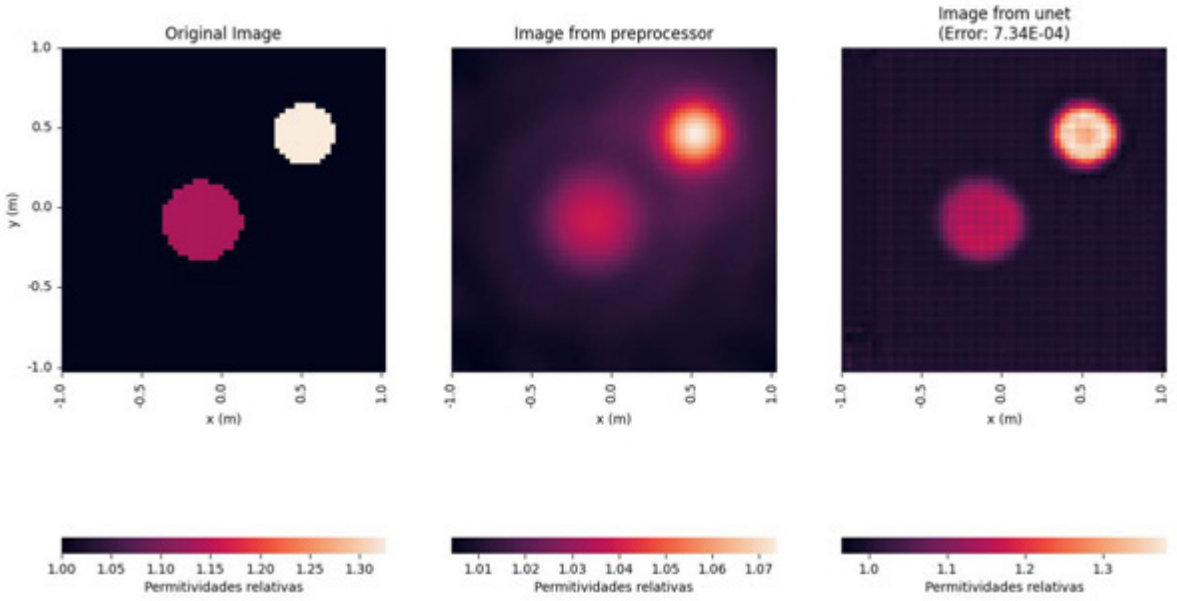

Fig. 37: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 10

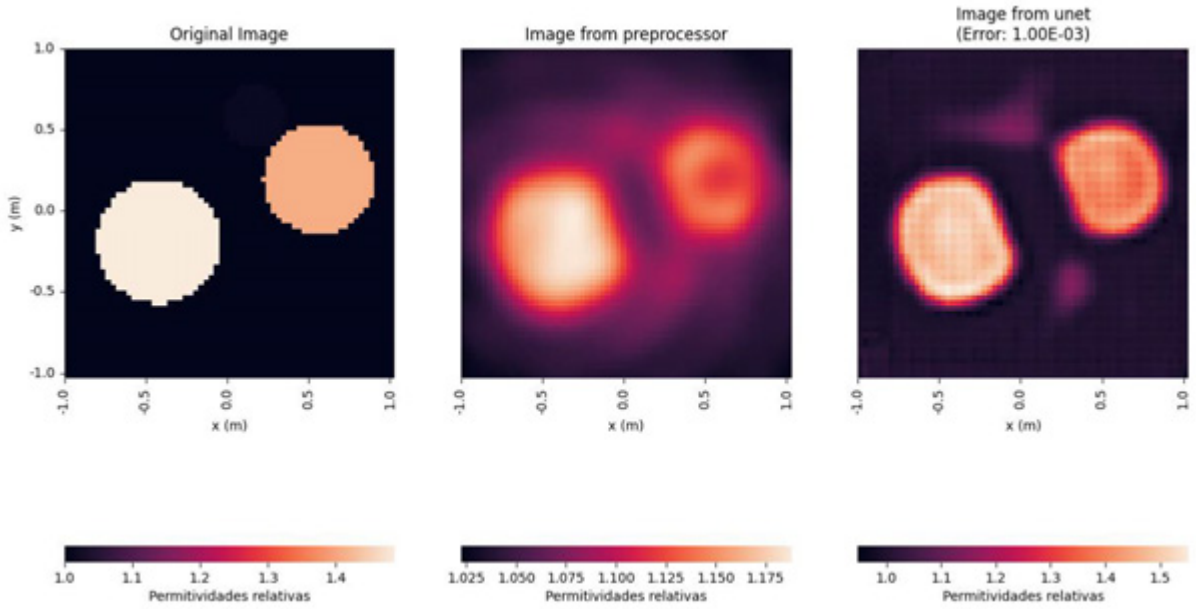

Fig. 38: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 10

Ejecución 11

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	32
Iteraciones	50

Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 21: Parámetros de ejecución 11

Métrica	**Valor**
Duración del entrenamiento	0:25:42
Error total	$1,38\ x\ 10^{-2}$
Error medio	$6,90\ x\ 10^{-3}$
Desvío estándar	$1,33\ x\ 10^{-4}$

Tabla 22: Resultados de ejecución 11

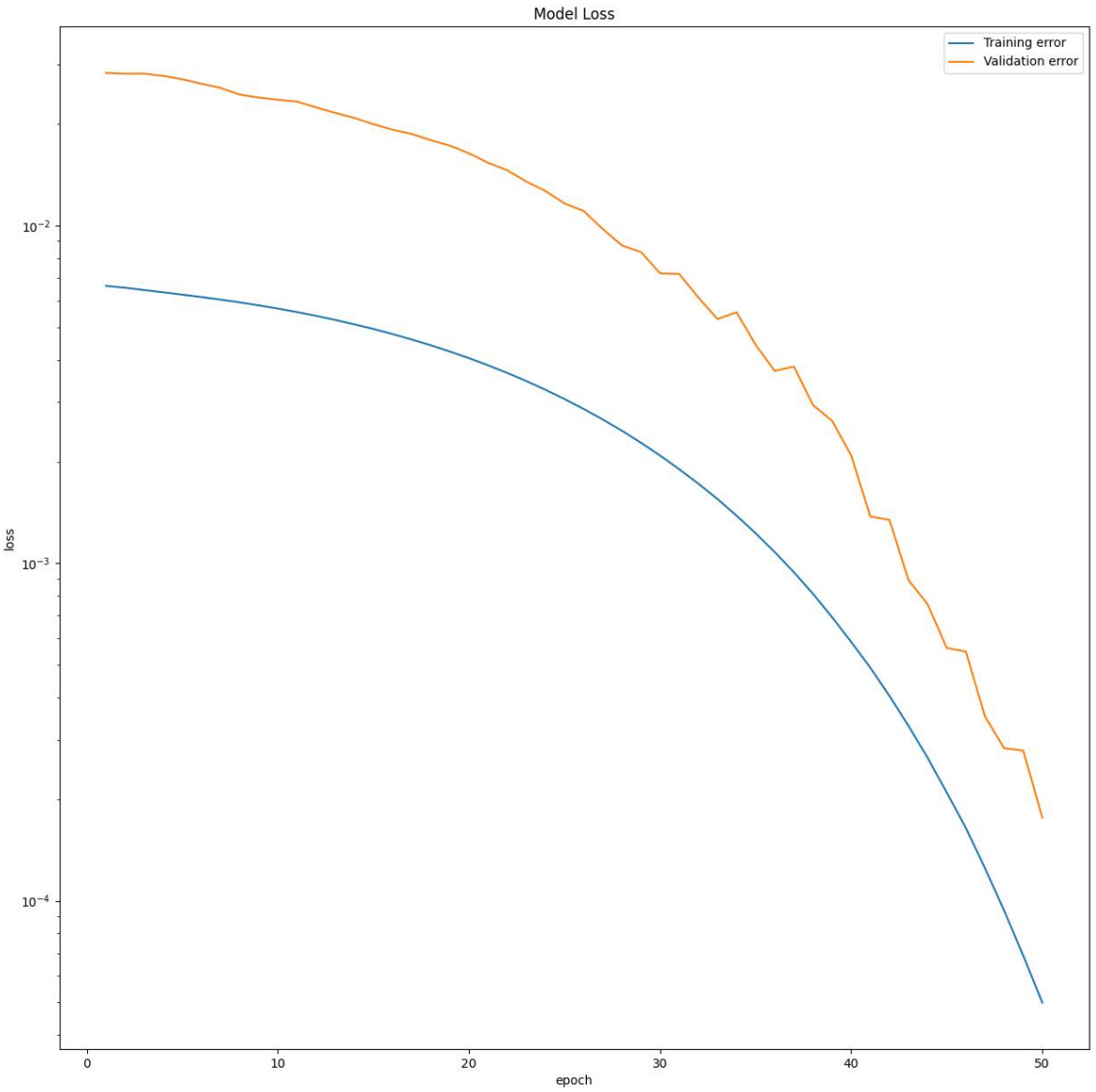

Fig. 39: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 11

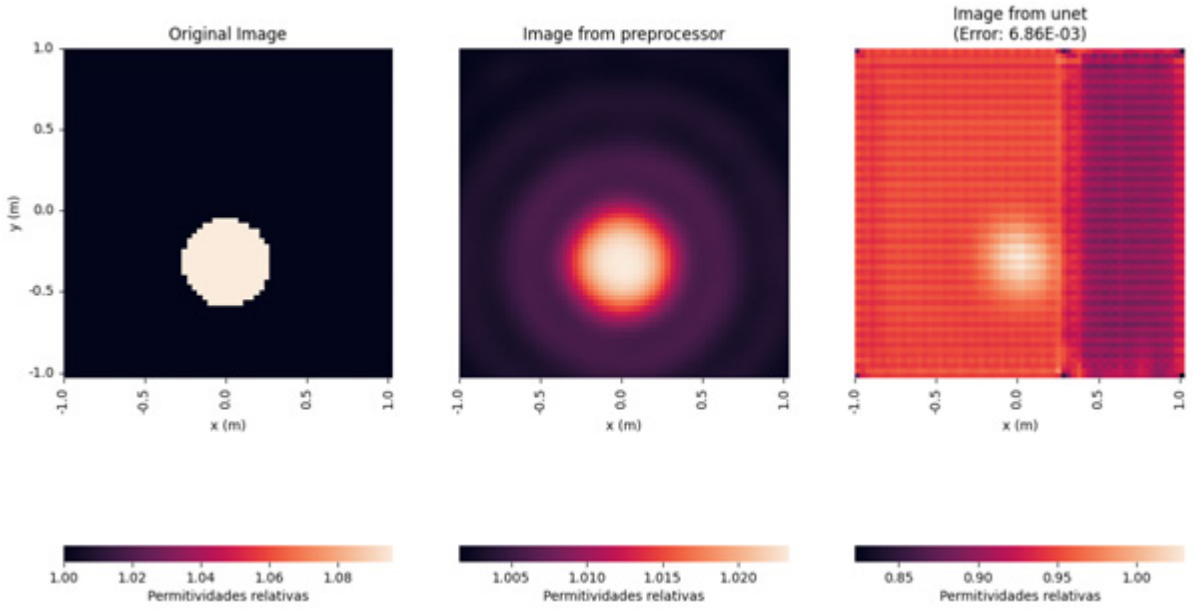

Fig. 40: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 11

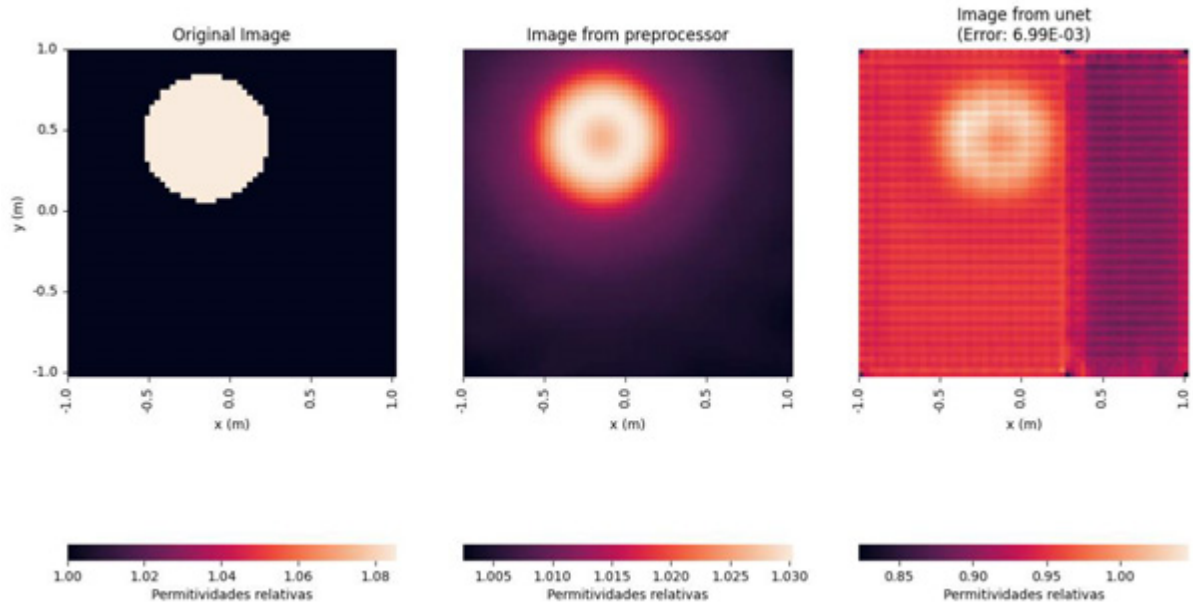

Fig. 41: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 11

Iteraciones

Se ejecutó GenPer variando la cantidad de iteraciones de 10 a 200 con 500 imágenes y una batch de tamaño 1. Para la ejecución se fue cambiando el número de iteraciones a medida que se terminaba de ejecutar, retomando de la última.

Ejecución 12

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	10
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 23: Parámetros de ejecución 12

Métrica	Valor
Duración del entrenamiento	0:10:21
Error total	$1,00\ x\ 10^{-1}$
Error medio	$1,34\ x\ 10^{-3}$
Desvío estándar	$1,49\ x\ 10^{-3}$

Tabla 24: Resultados de ejecución 12

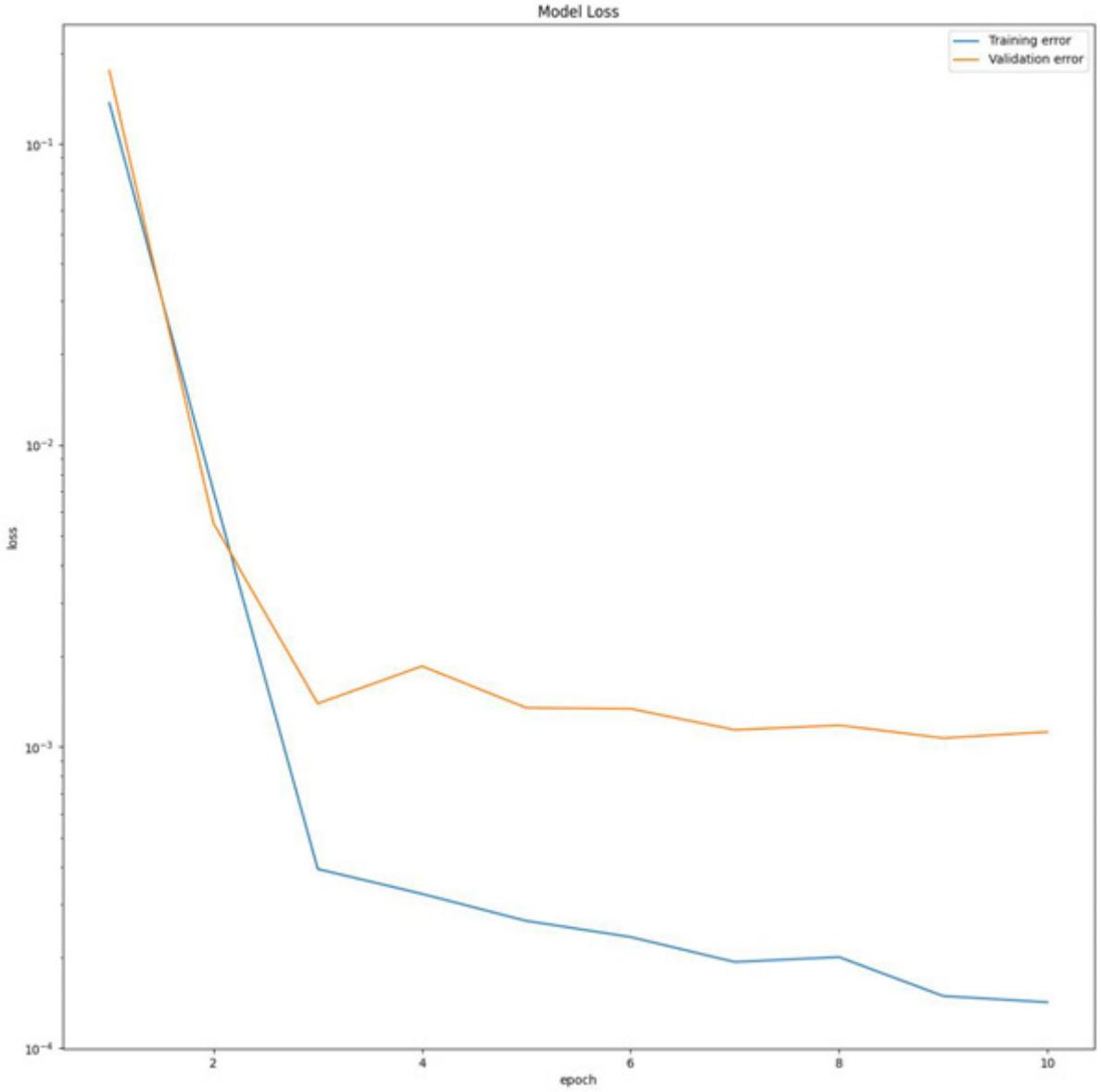

Fig. 42: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 12

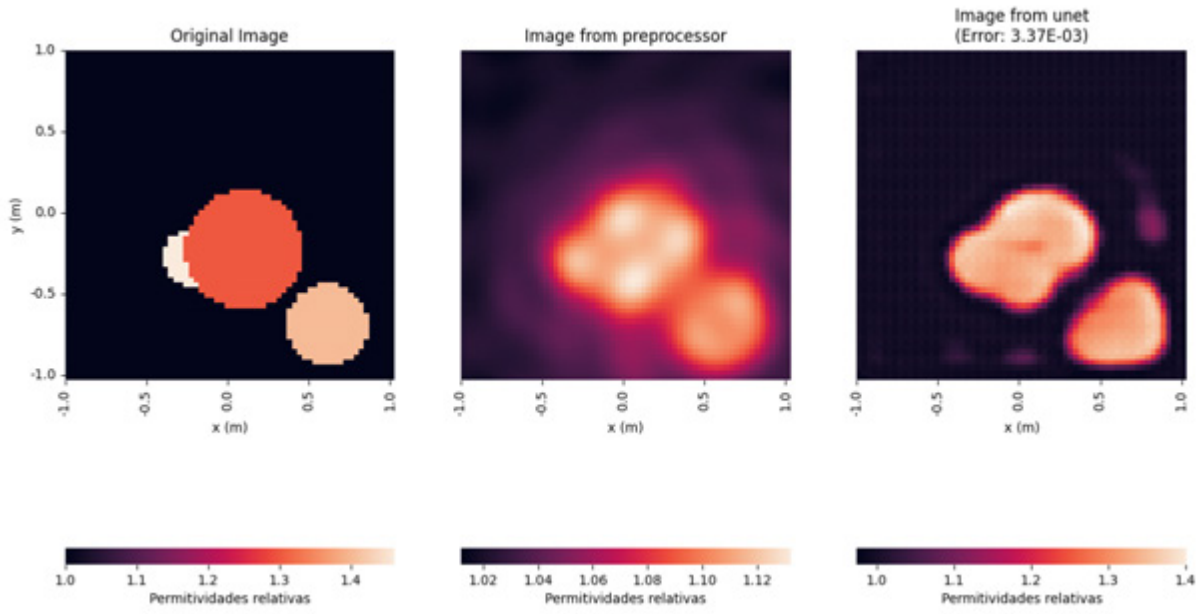

Fig. 43: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 12

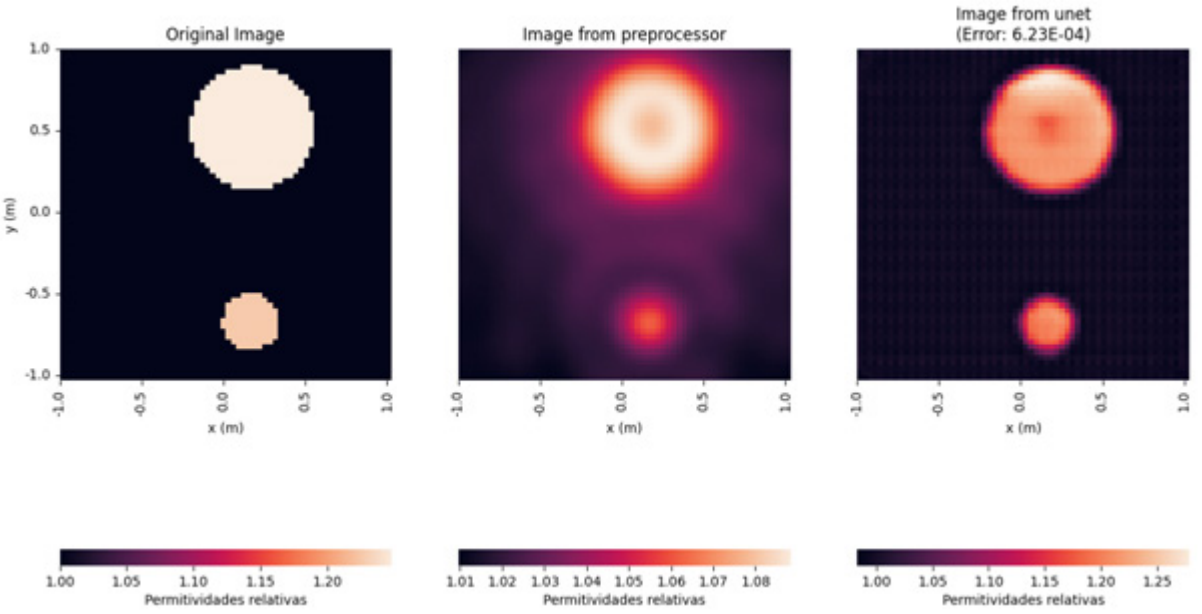

Fig. 44: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 12

Ejecución 13

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 25: Parámetros de ejecución 13

Métrica	**Valor**
Duración del entrenamiento	0:44:35
Error total	$7,93\ x\ 10^{-2}$
Error medio	$1,06\ x\ 10^{-3}$
Desvío estándar	$1,60\ x\ 10^{-3}$

Tabla 26: Resultados de ejecución 13

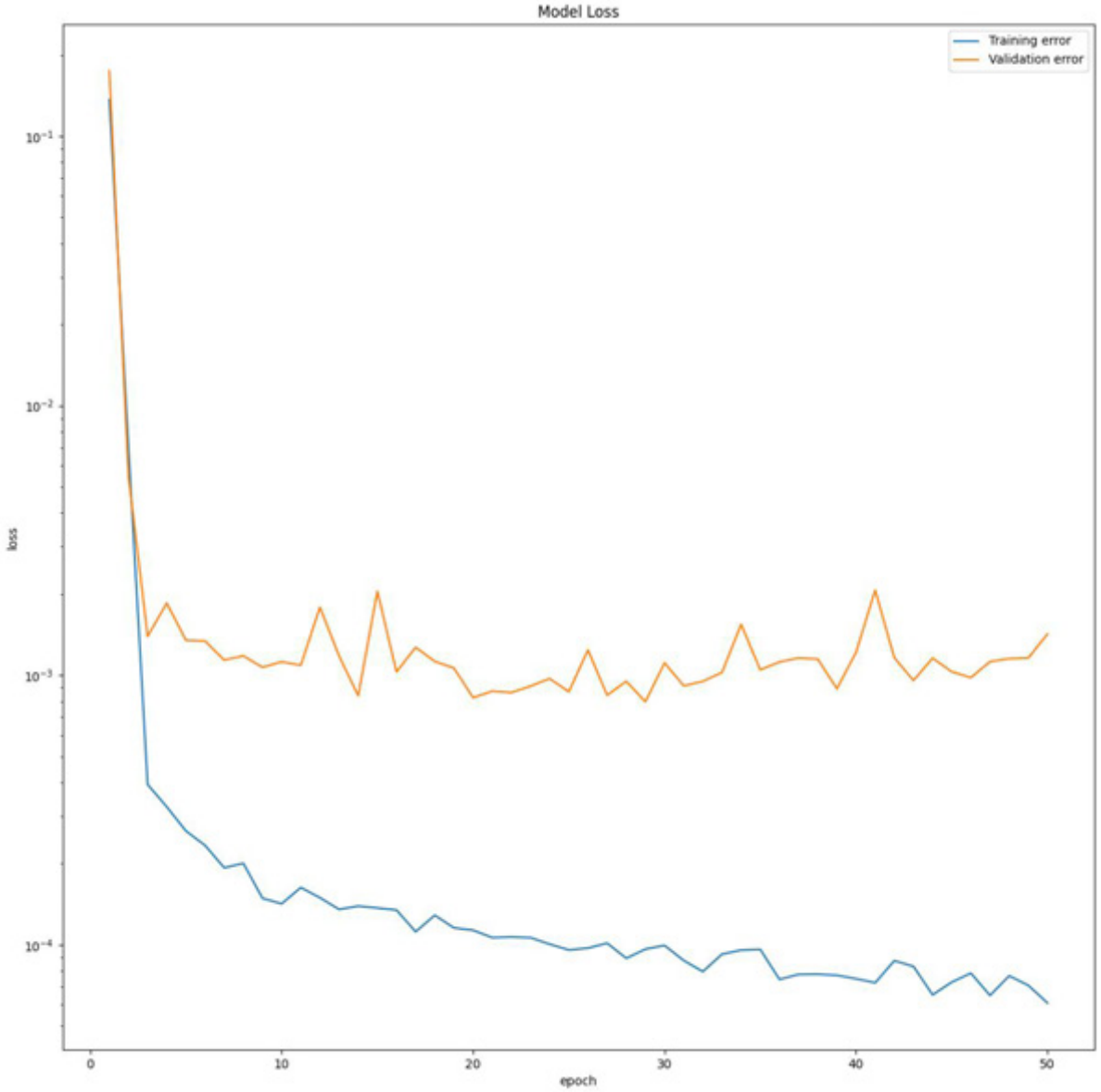

Fig. 45: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 13

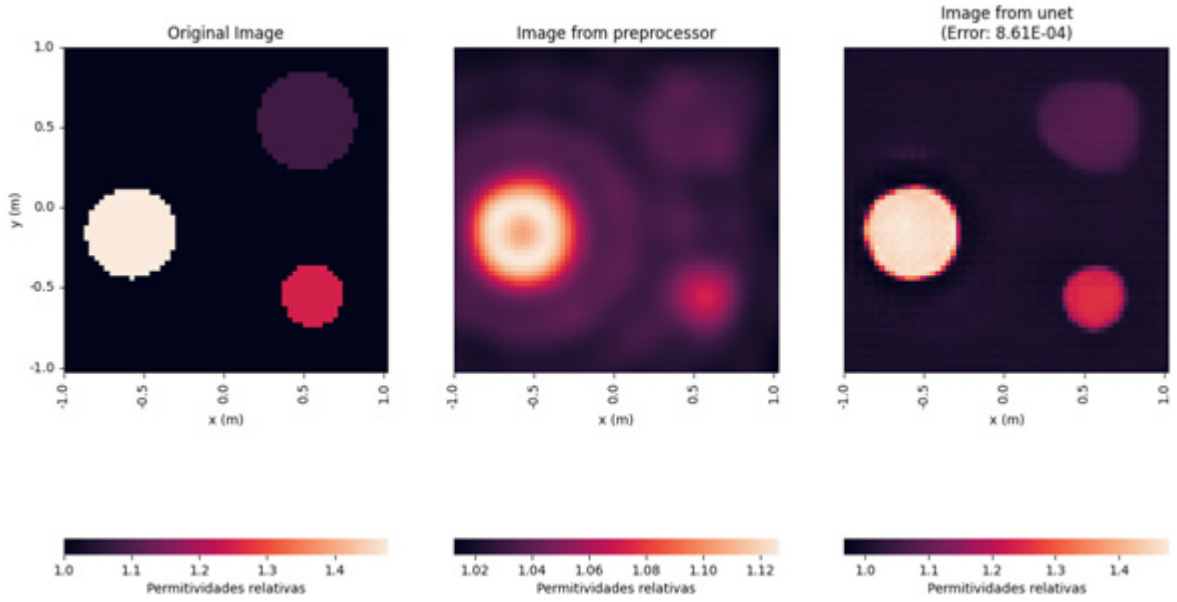

Fig. 46: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 13

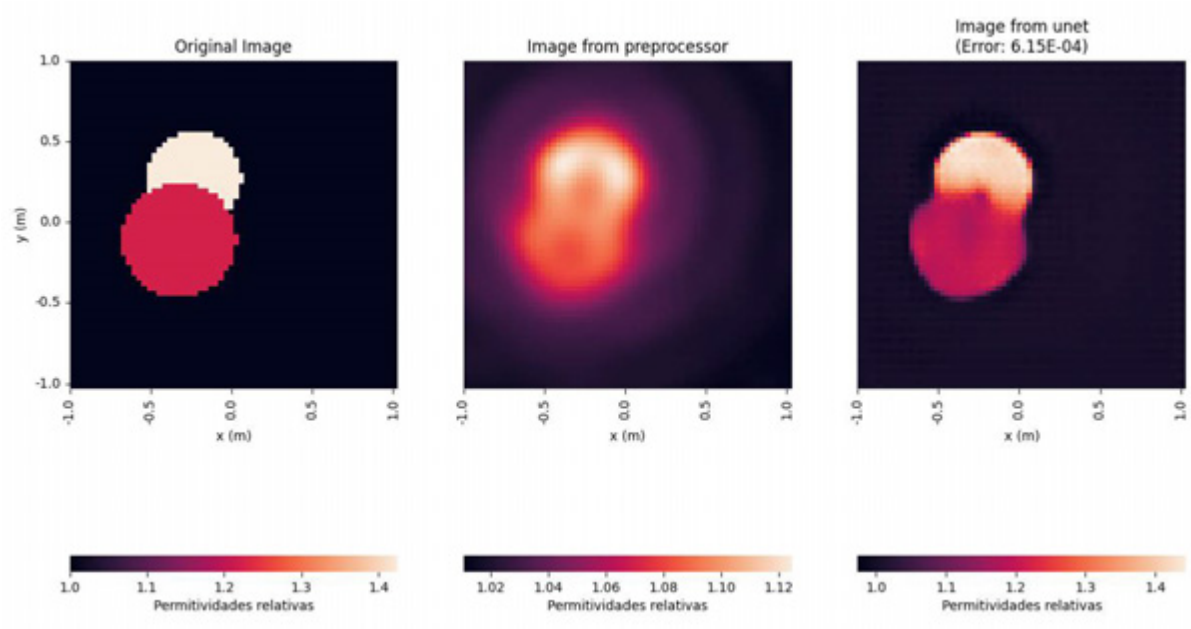

Fig. 47: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 13

Ejecución 14

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	75

Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 27: Parámetros de ejecución 14

Métrica	**Valor**
Duración del entrenamiento	1:09:34
Error total	$7,93\ x\ 10^{-2}$
Error medio	$1,06\ x\ 10^{-3}$
Desvío estándar	$1,60\ x\ 10^{-3}$

Tabla 28: Resultados de ejecución 14

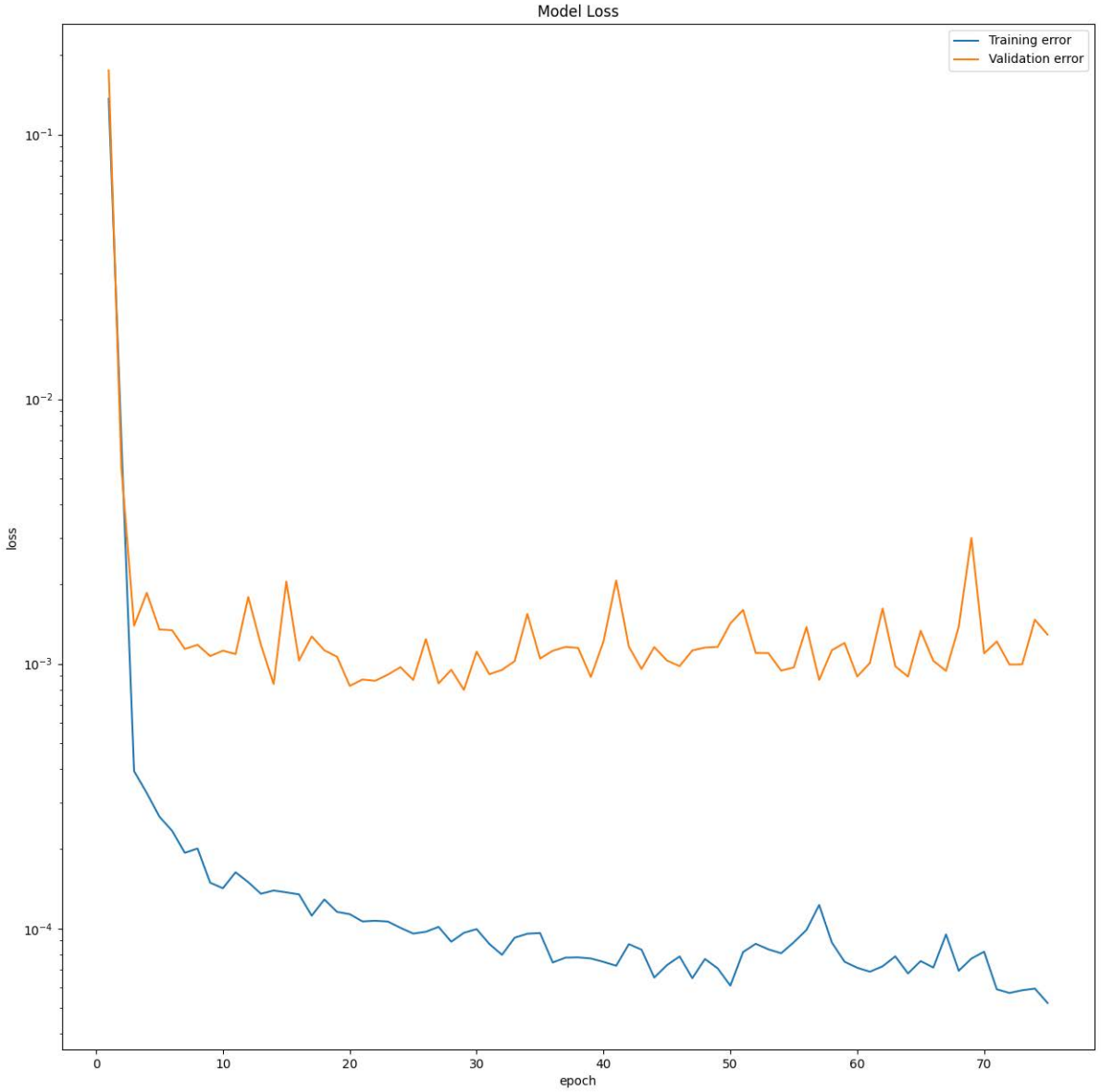

Fig. 48: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 14

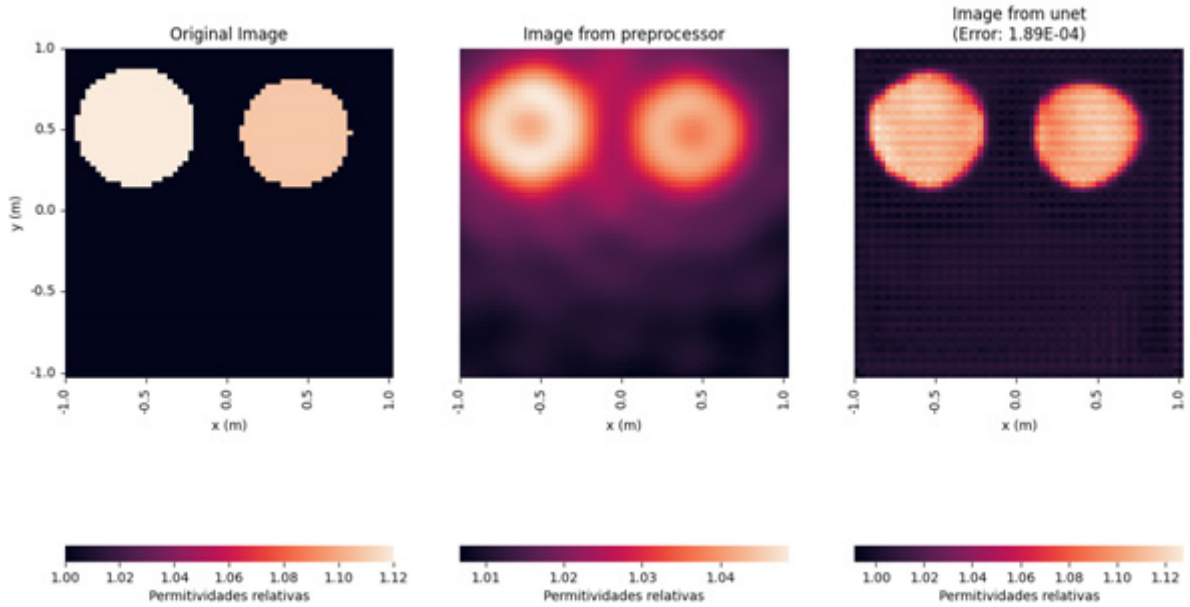

Fig. 49: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 14

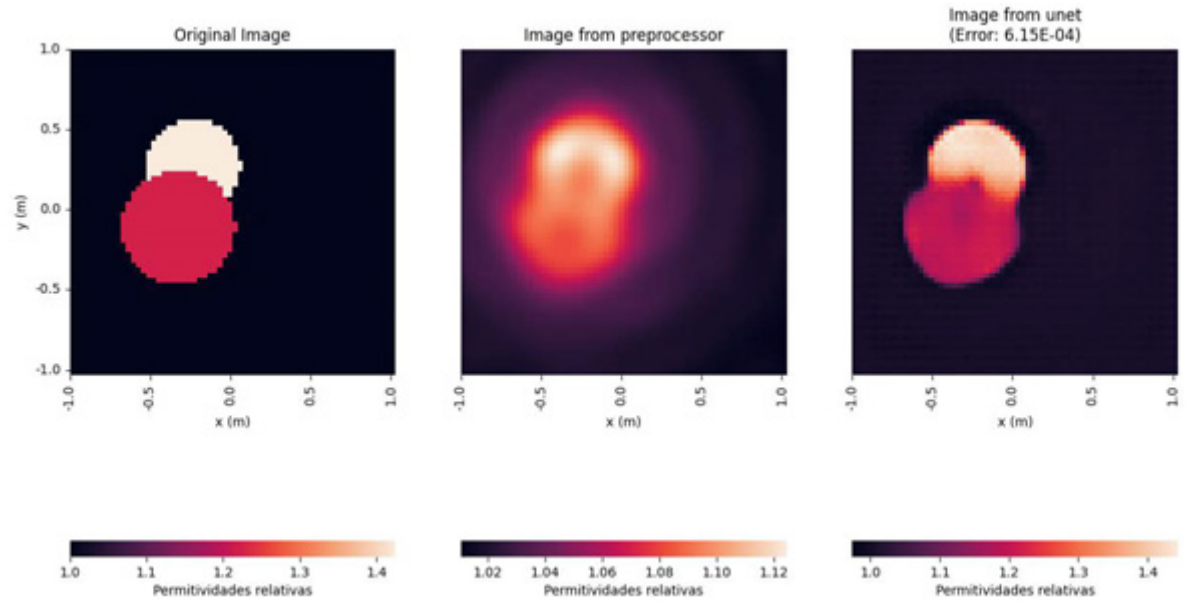

Fig. 50: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 14

Ejecución 15

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	100
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 29: Parámetros de ejecución 15

Métrica	**Valor**
Duración del entrenamiento	1:31:32
Error total	$7,62\ x\ 10^{-2}$
Error medio	$1,02\ x\ 10^{-3}$
Desvío estándar	$1,48\ x\ 10^{-3}$

Tabla 30: Resultados de ejecución 15

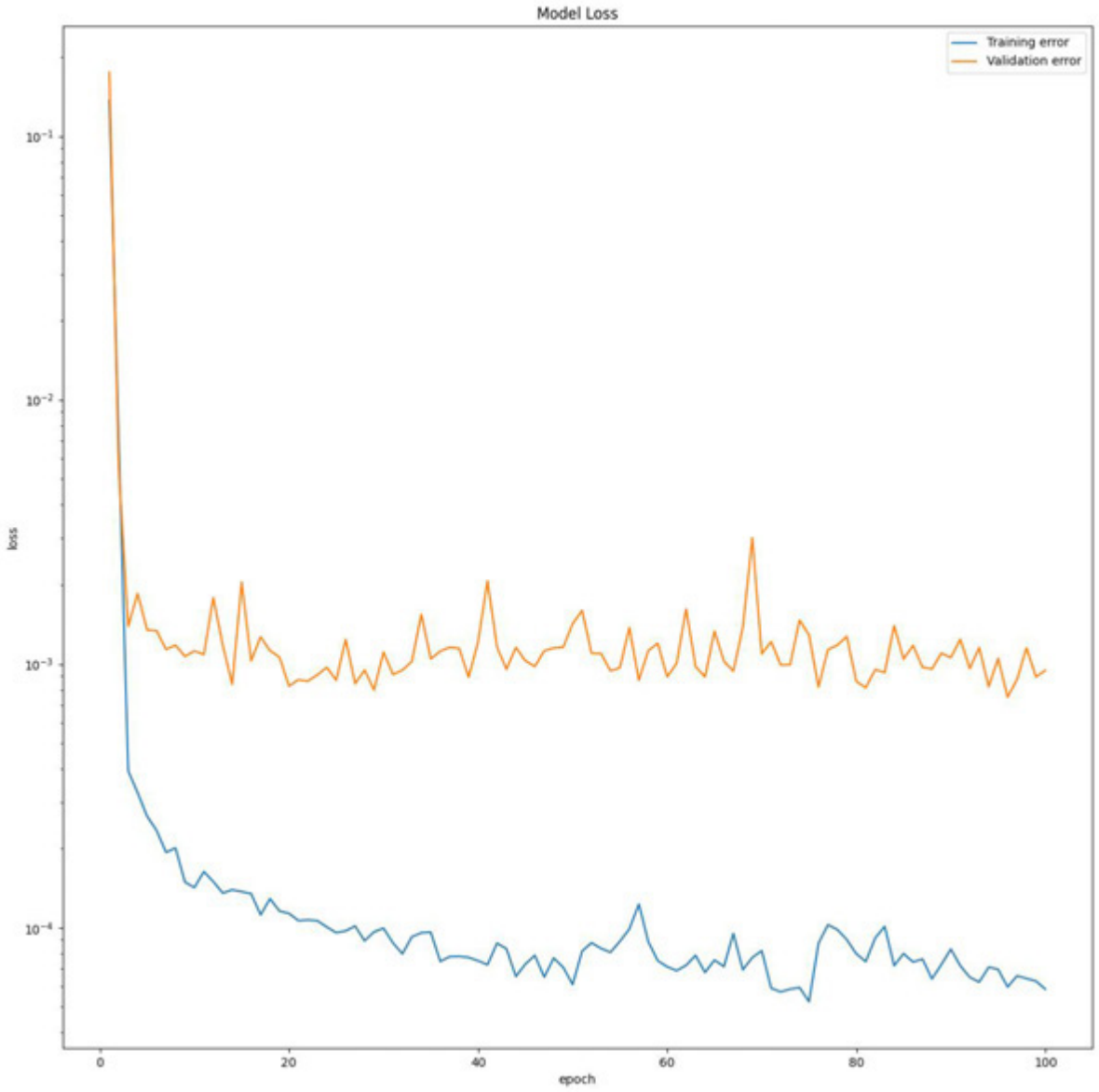

Fig. 51: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 15

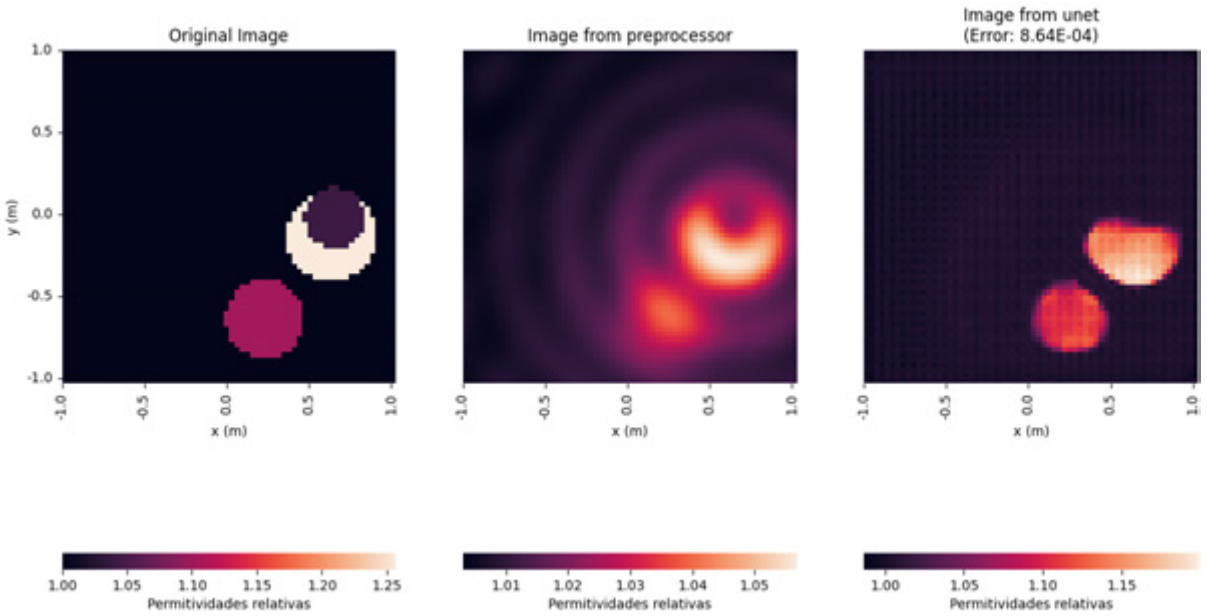

Fig. 52: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 15

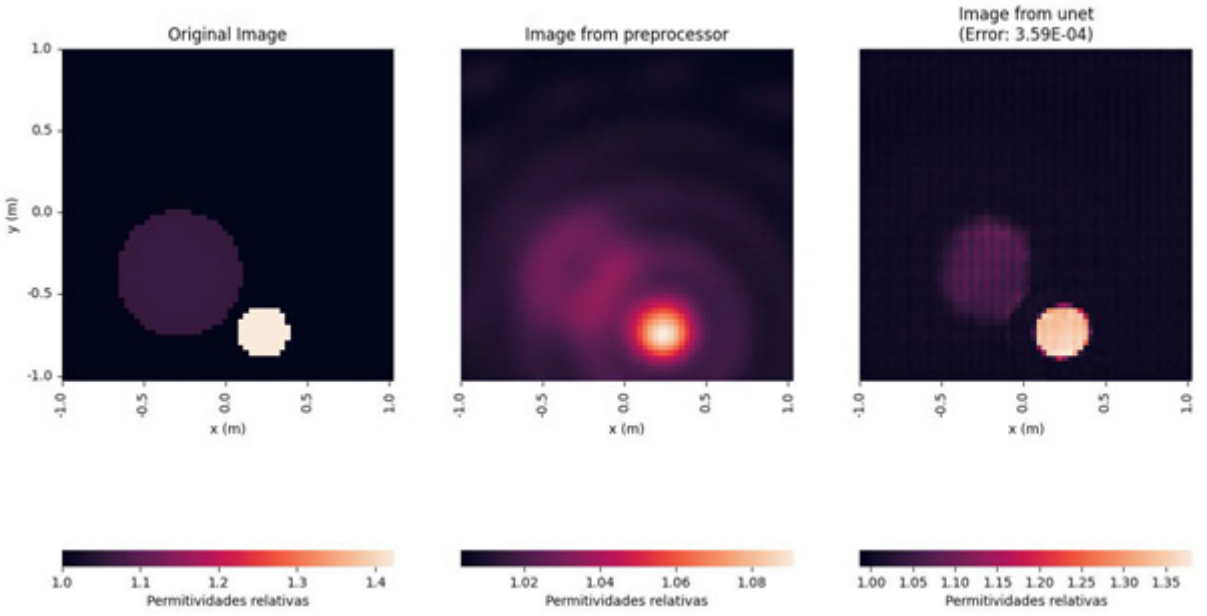

Fig. 53: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 15

Ejecución 16

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	150
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 31: Parámetros de ejecución 16

Métrica	**Valor**
Duración del entrenamiento	2:11:06
Error total	$7,62\ x\ 10^{-2}$
Error medio	$1,02\ x\ 10^{-3}$
Desvío estándar	$1,48\ x\ 10^{-3}$

Tabla 32: Resultados de ejecución 16

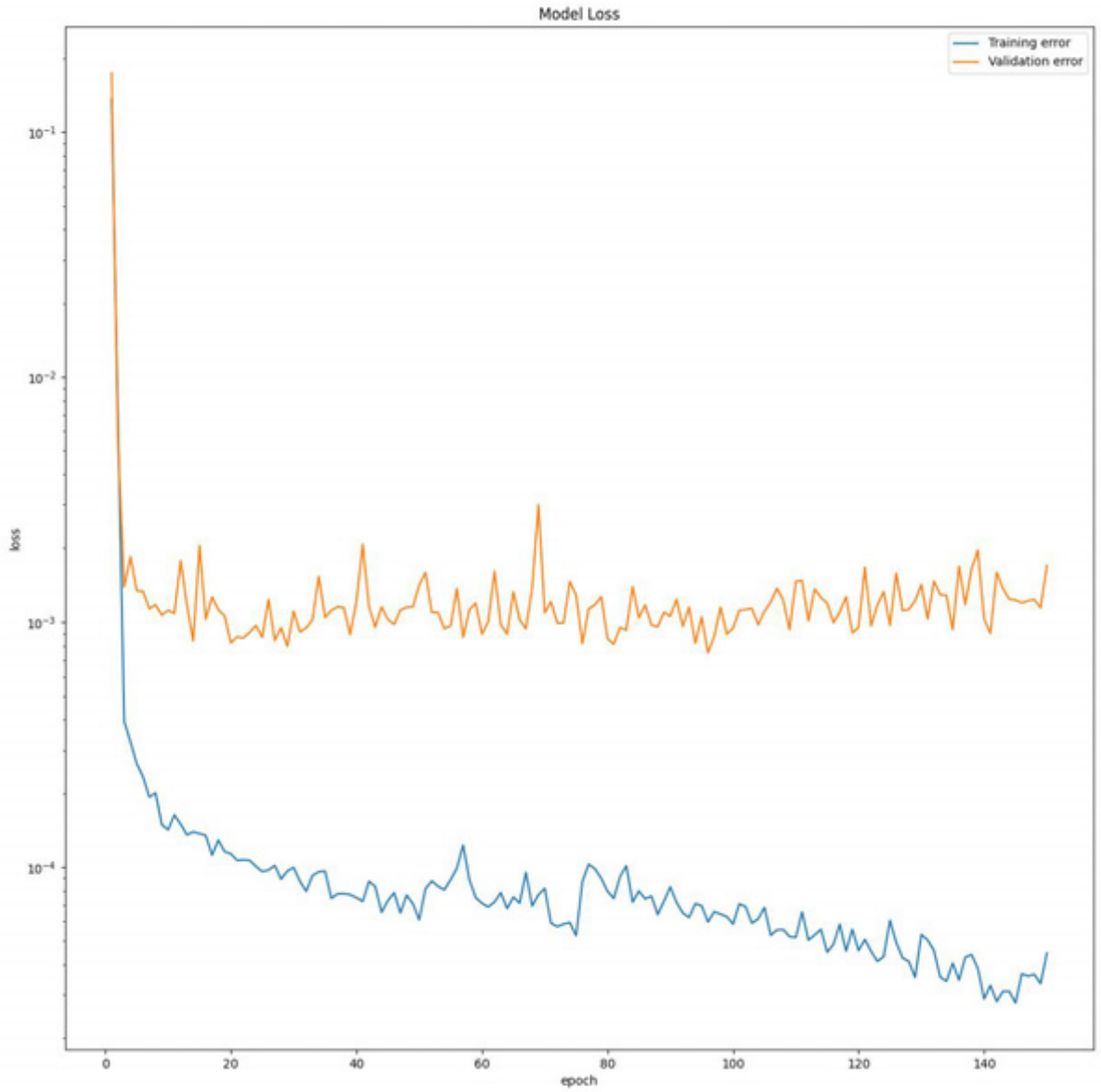

Fig. 54: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 16

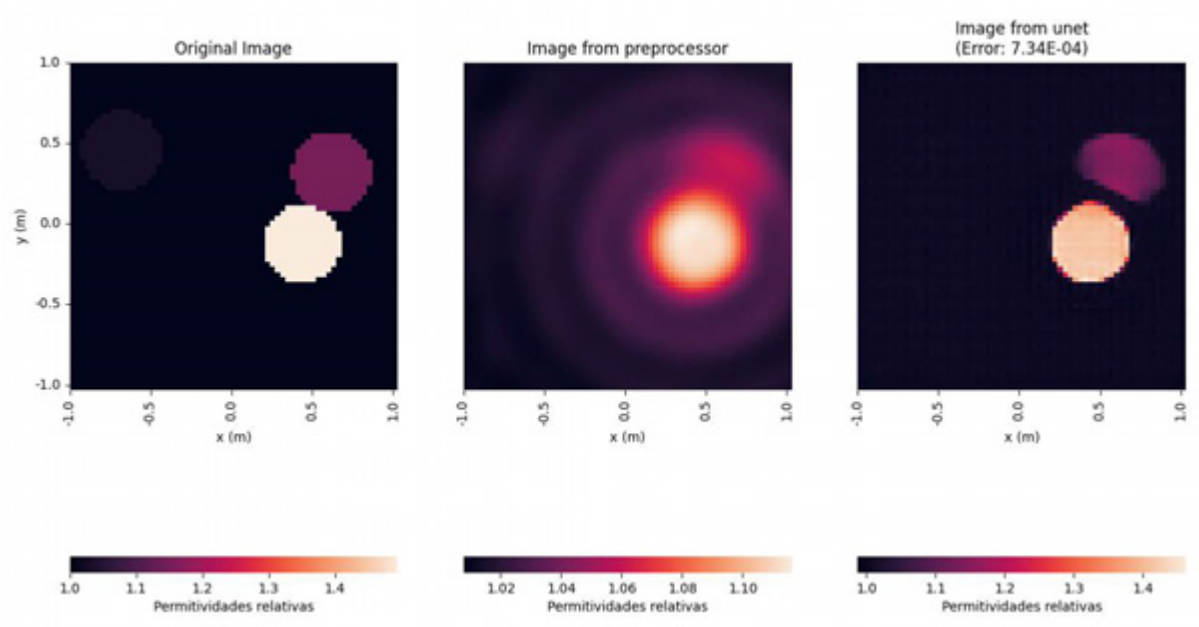

Fig. 55: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 16

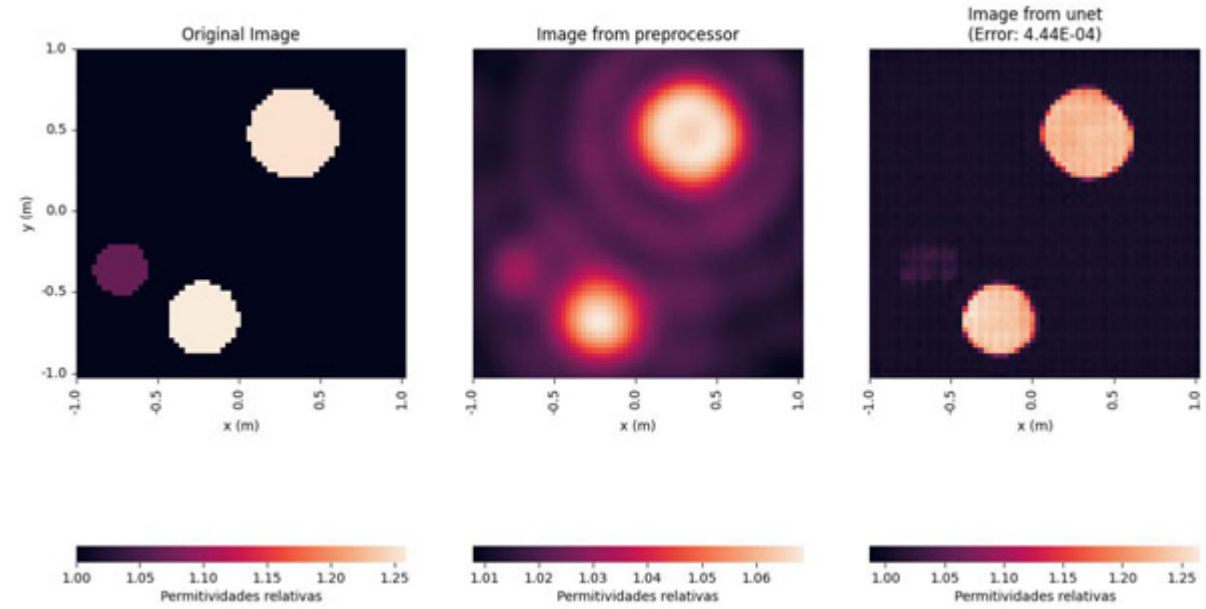

Fig. 56: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 16

Ejecución 17

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	200

Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 33: Parámetros de ejecución 17

Métrica	**Valor**
Duración del entrenamiento	2:49:42
Error total	$7,15\ x\ 10^{-2}$
Error medio	$9,54\ x\ 10^{-4}$
Desvío estándar	$1,27\ x\ 10^{-3}$

Tabla 34: Resultados de ejecución 17

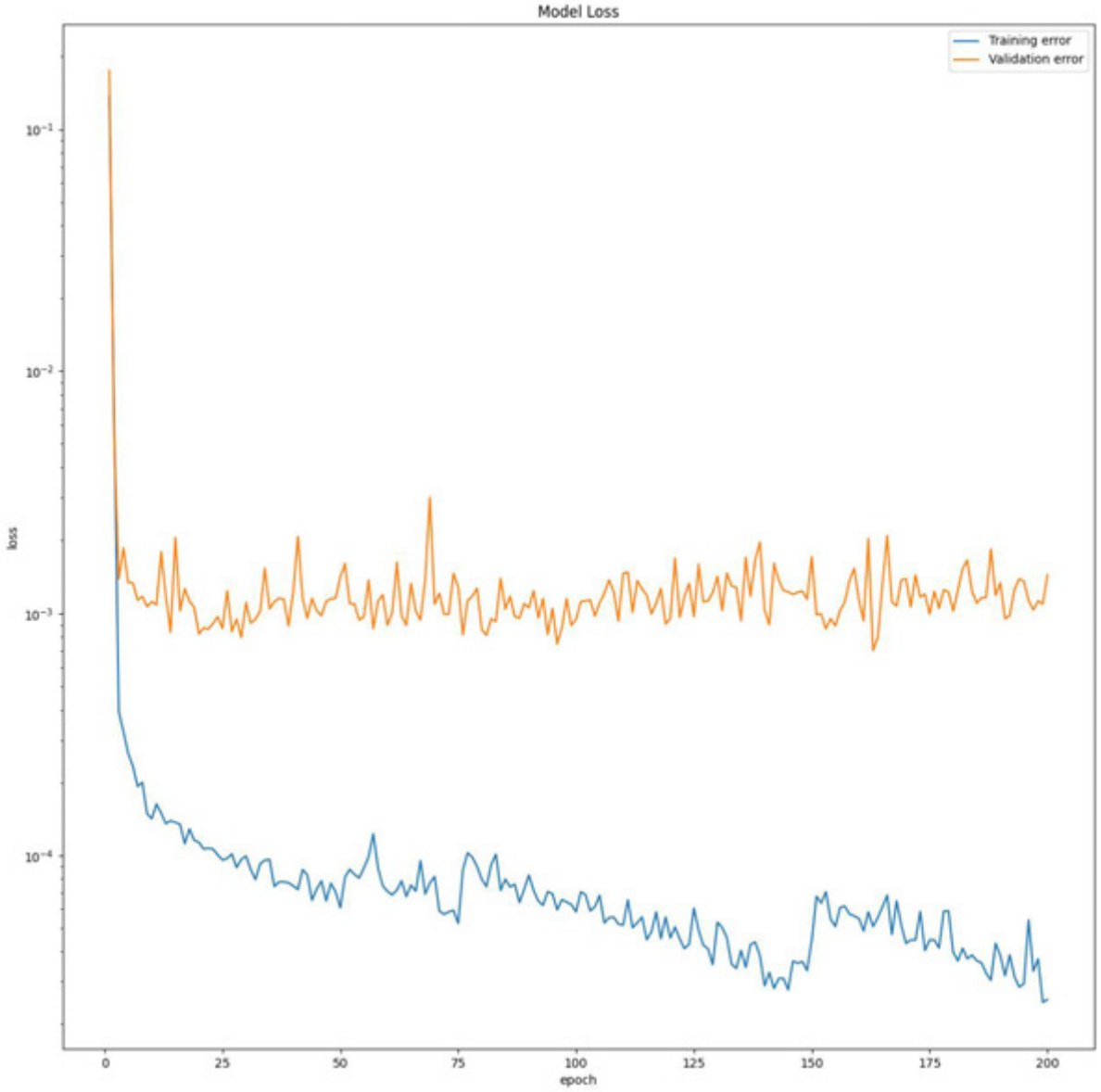

Fig. 57: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 17

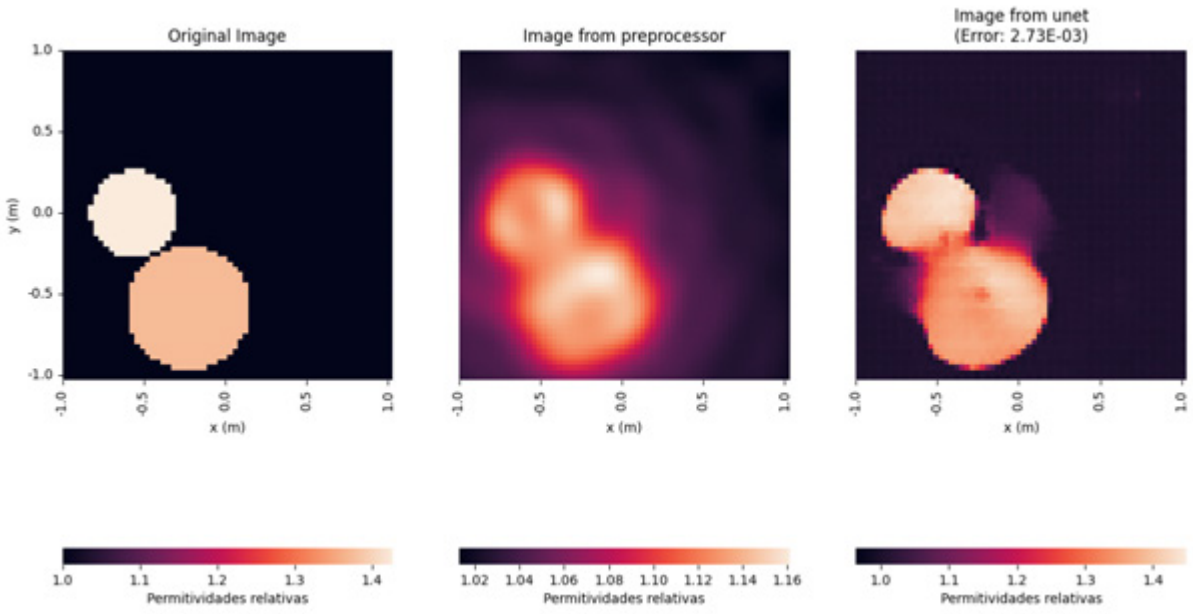

Fig. 58: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 17

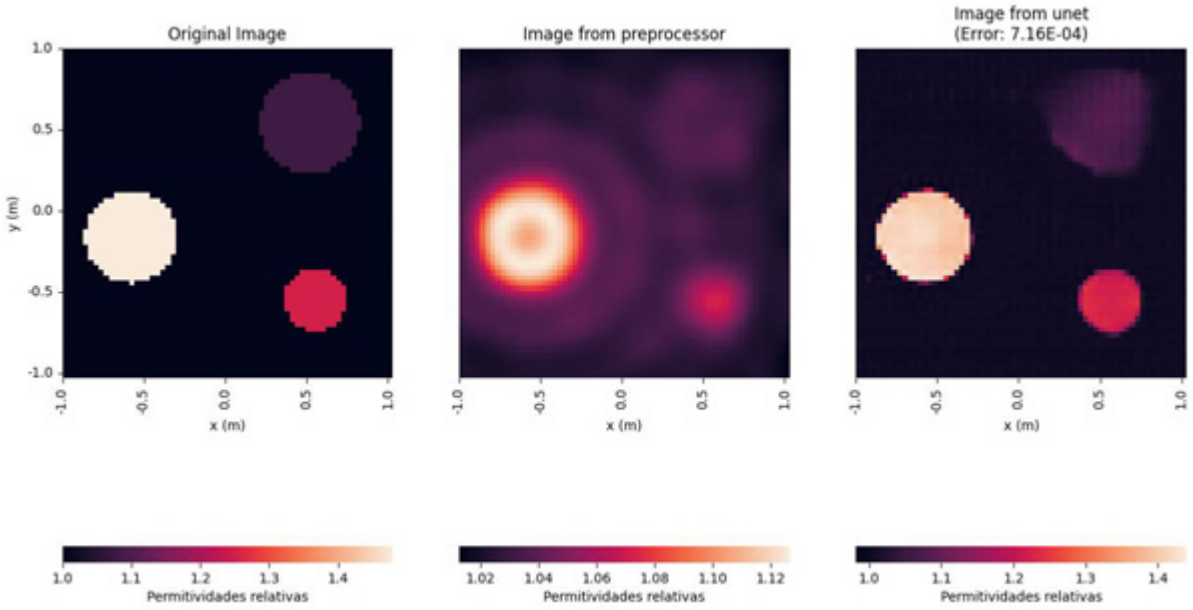

Fig. 59: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 17

Proporciones del set de datos

Se ejecutó GenPer variando las proporciones en las que se divide el set de datos para el entrenamiento, validación y testing para 500 imágenes, un batch de 1 y 50 iteraciones.

Ejecución 18

Parámetro	Valor

Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	300 (60 %)
Set de validación	150 (20 %)
Set de testing	150 (20 %)
Rango de permitividades	1 - 1,5

Tabla 35: Parámetros de ejecución 18

Métrica	**Valor**
Duración del entrenamiento	0:39:14
Error total	$7,69\ x\ 10^{-2}$
Error medio	$7,69\ x\ 10^{-4}$
Desvío estándar	$5,65\ x\ 10^{-4}$

Tabla 36: Resultados de ejecución 18

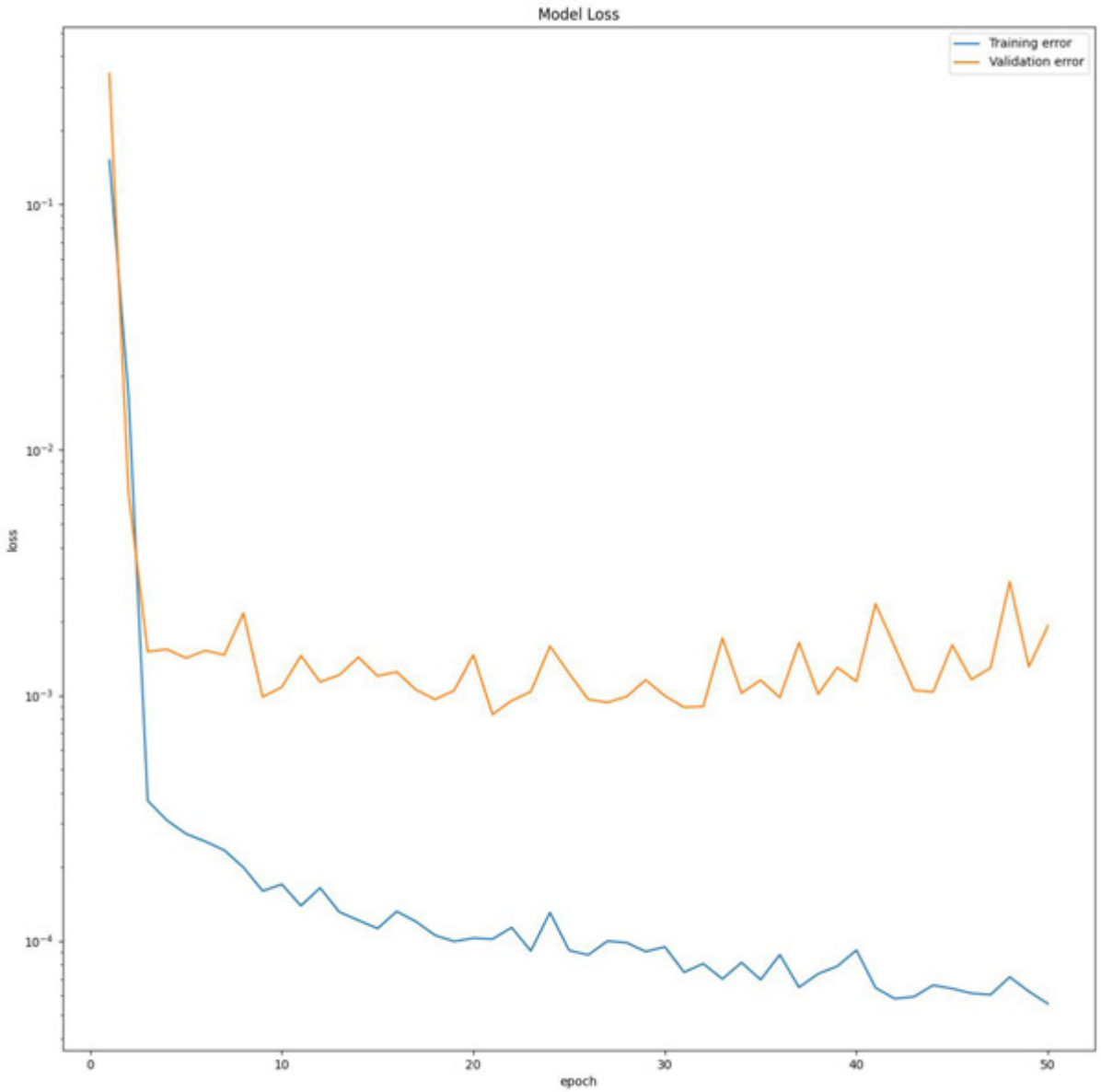

Fig. 60: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 18

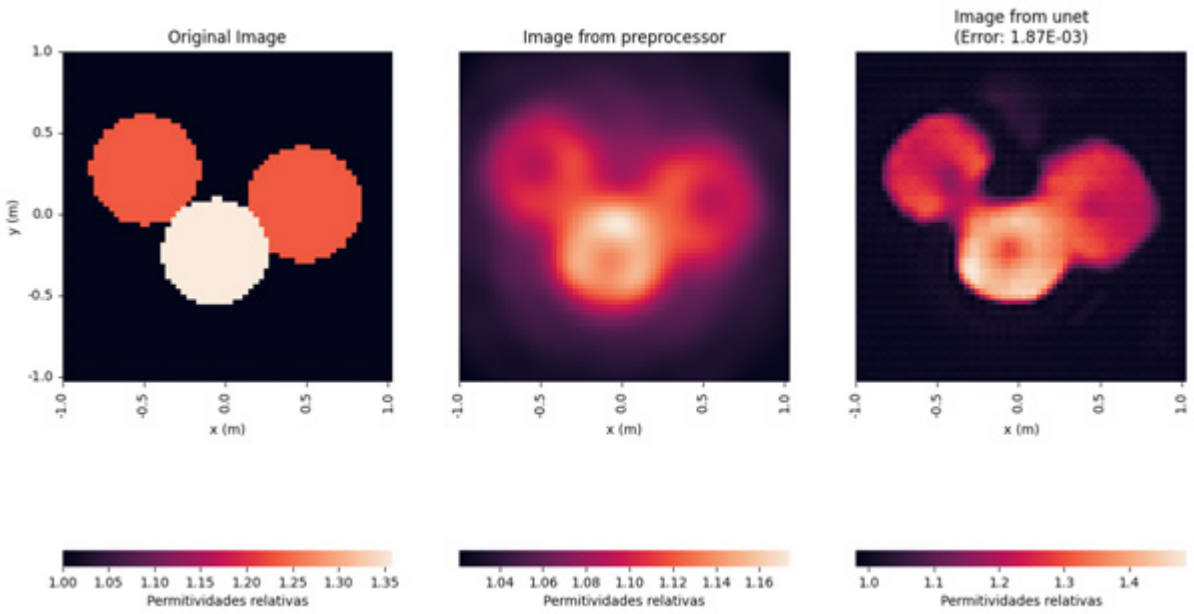

Fig. 61: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 18

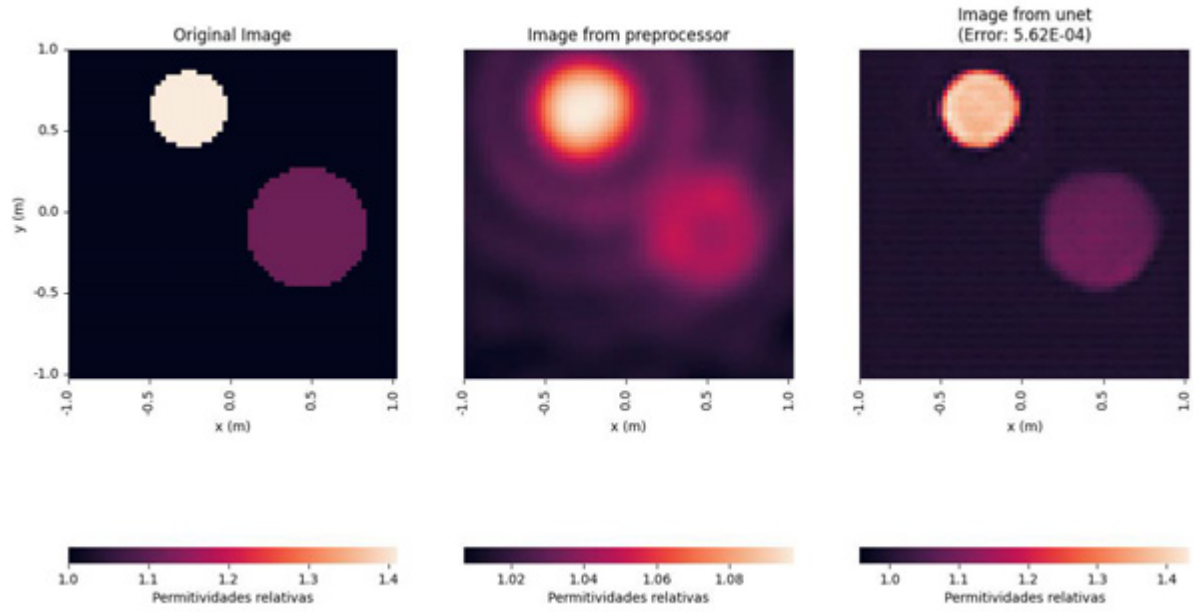

Fig. 62: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 18

Ejecución 19

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 37: Parámetros de ejecución 19

Para ver los resultados de la ejecución, remitirse a la ejecución 6.

Ejecución 20

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	400 (80 %)
Set de validación	50 (10 %)
Set de testing	50 (10 %)
Rango de permitividades	1 - 1,5

Tabla 38: Parámetros de ejecución 20

Métrica	**Valor**
Duración del entrenamiento	0:39:18
Error total	$4,47\ x\ 10^{-2}$
Error medio	$8,94\ x\ 10^{-4}$
Desvío estándar	$1,28\ x\ 10^{-3}$

Tabla 39: Resultados de ejecución 20

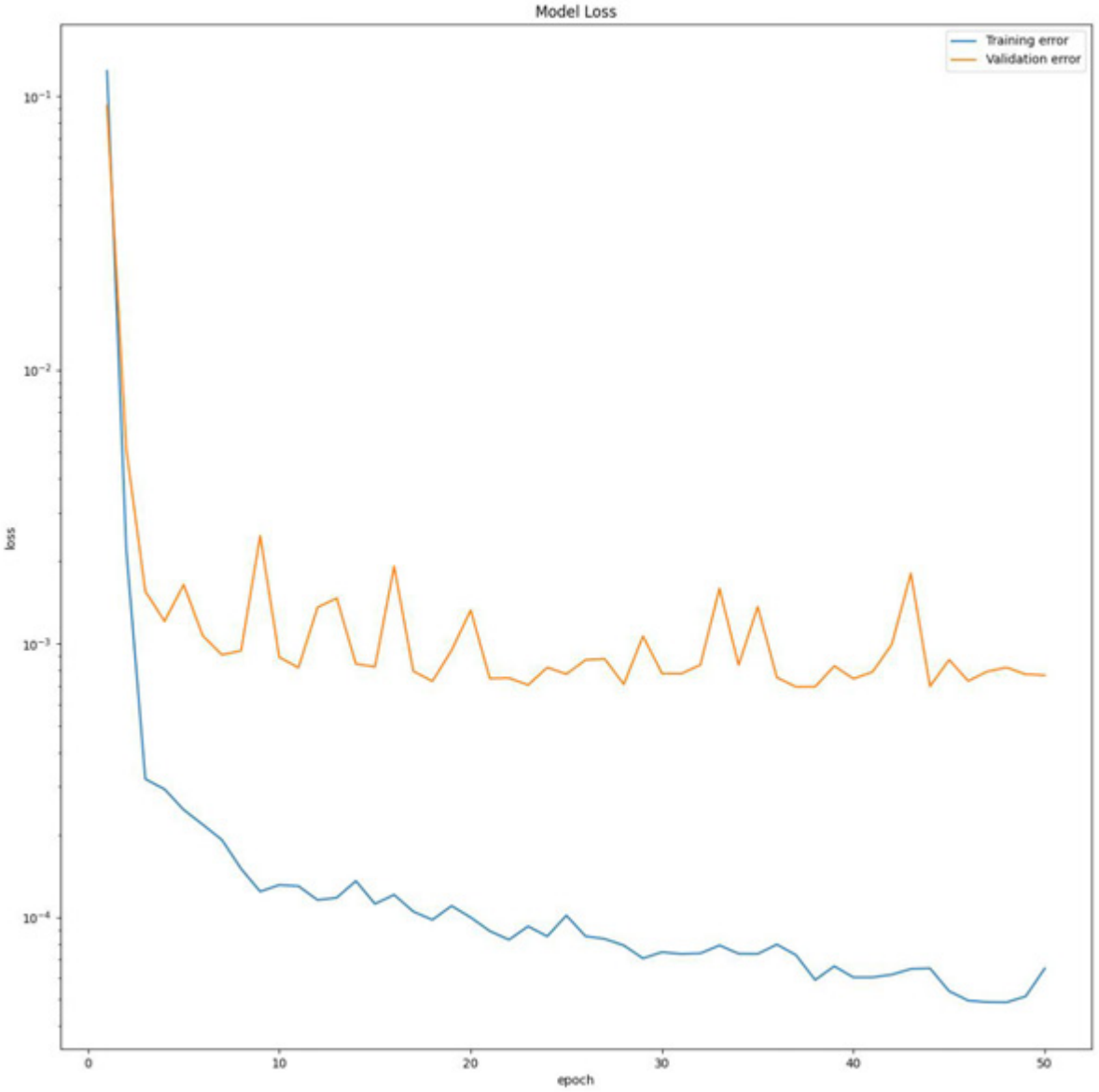

Fig. 63: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 20

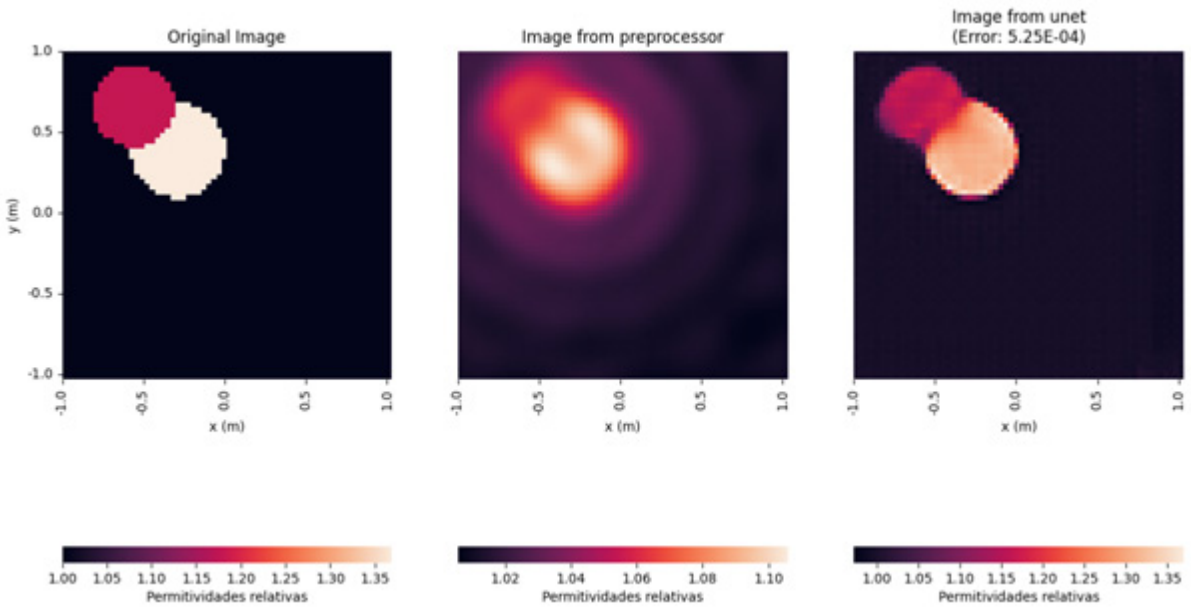

Fig. 64: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 20

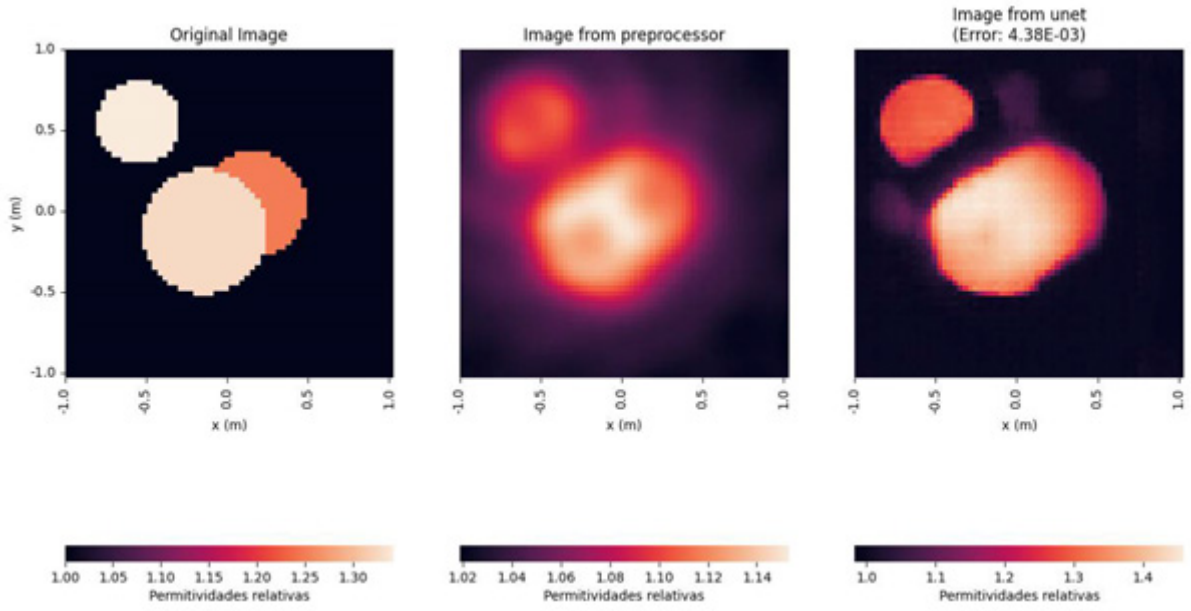

Fig. 65: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 20

Ejecución 21

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	450 (90 %)

Set de validación	25 (5 %)
Set de testing	25 (5 %)
Rango de permitividades	1 - 1,5

Tabla 40: Parámetros de ejecución 21

Métrica	**Valor**
Duración del entrenamiento	0:44:18
Error total	$2,21\ x\ 10^{-2}$
Error medio	$8,85\ x\ 10^{-4}$
Desvío estándar	$9,63\ x\ 10^{-4}$

Tabla 41: Resultados de ejecución 21

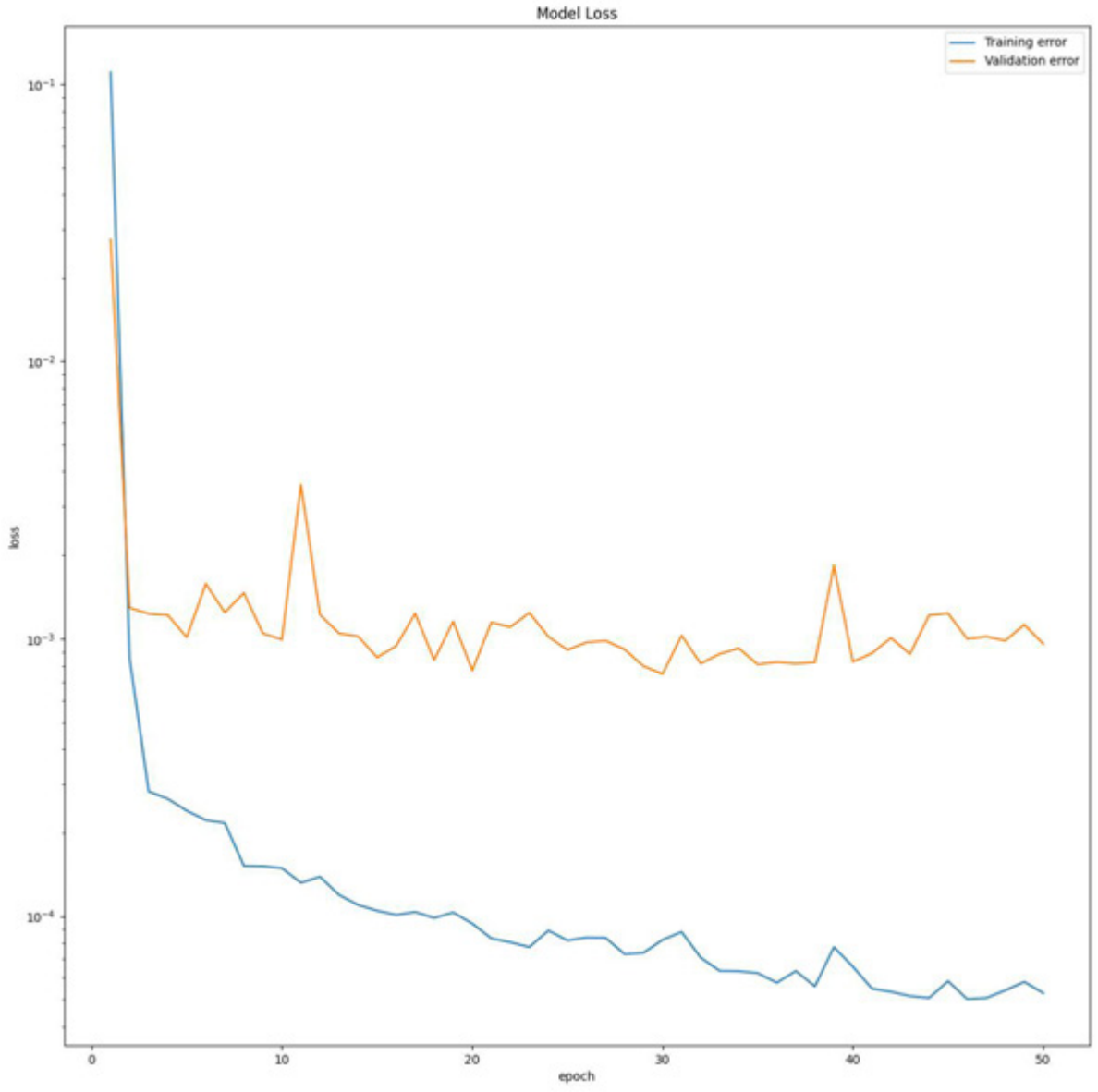

Fig. 66: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 21

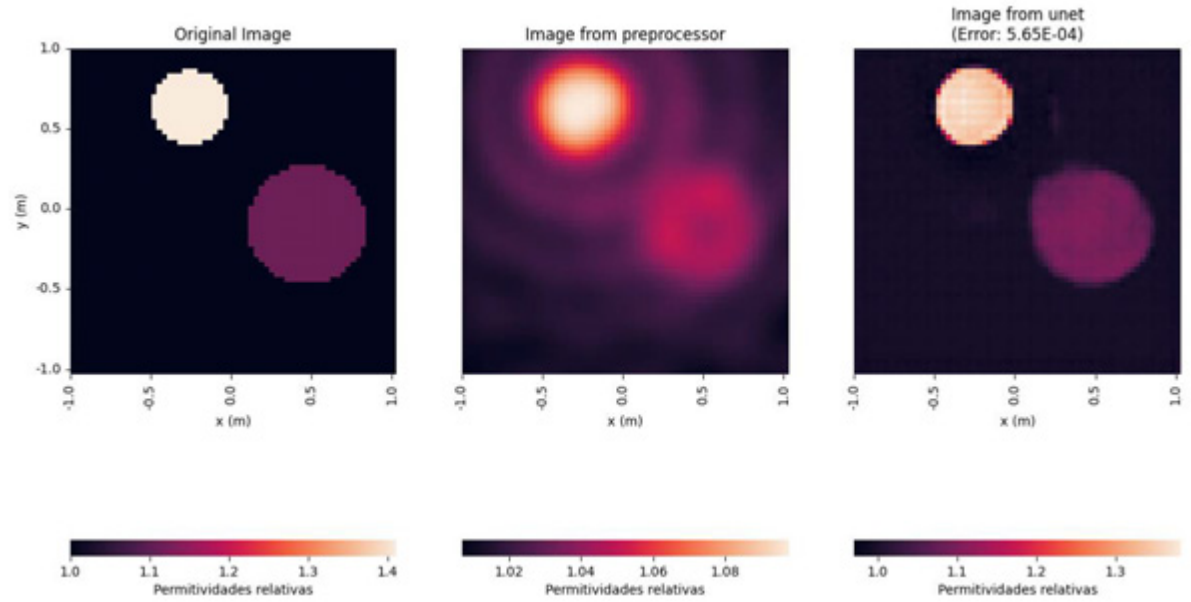

Fig. 67: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 21

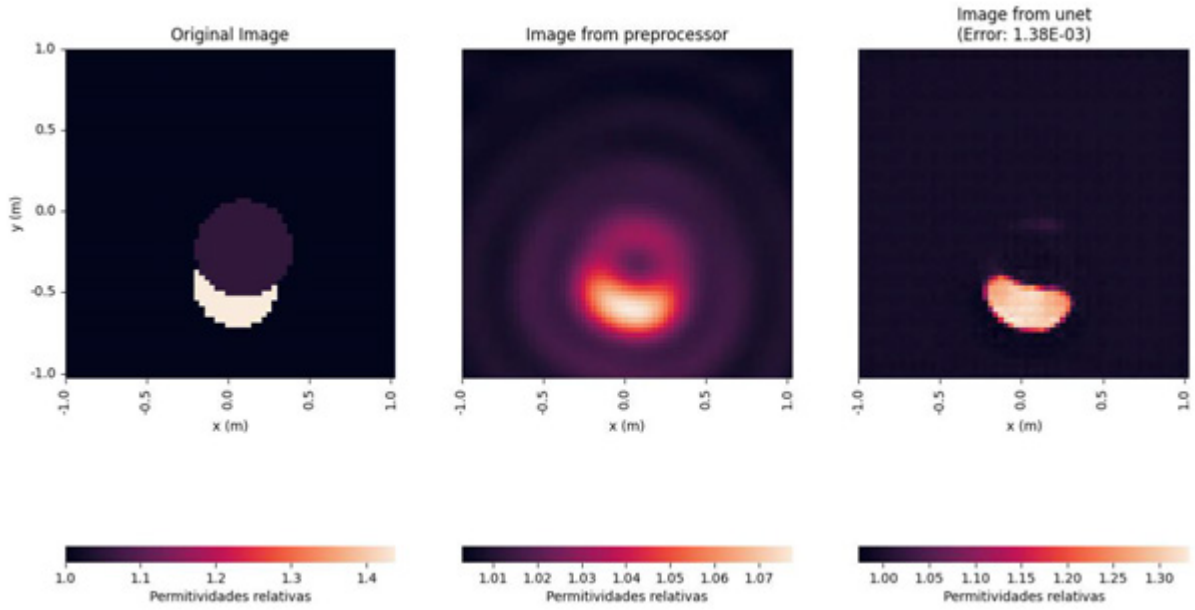

Fig. 68: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 21

Rango de permitividades

Se ejecutó GenPer variando el rango de permitividades (mayor y menor contraste) para 500 imágenes, un batch de 1 y 50 iteraciones.

Ejecución 22

Parámetro	Valor
Imágenes	500

Imágenes por batch	1
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,005

Tabla 42: Parámetros de ejecución 22

Métrica	**Valor**
Duración del entrenamiento	2:38:25
Error total	$2,26\ x\ 10^{-5}$
Error medio	$3,01\ x\ 10^{-7}$
Desvío estándar	$7,00\ x\ 10^{-8}$

Tabla 43: Resultados de ejecución 22

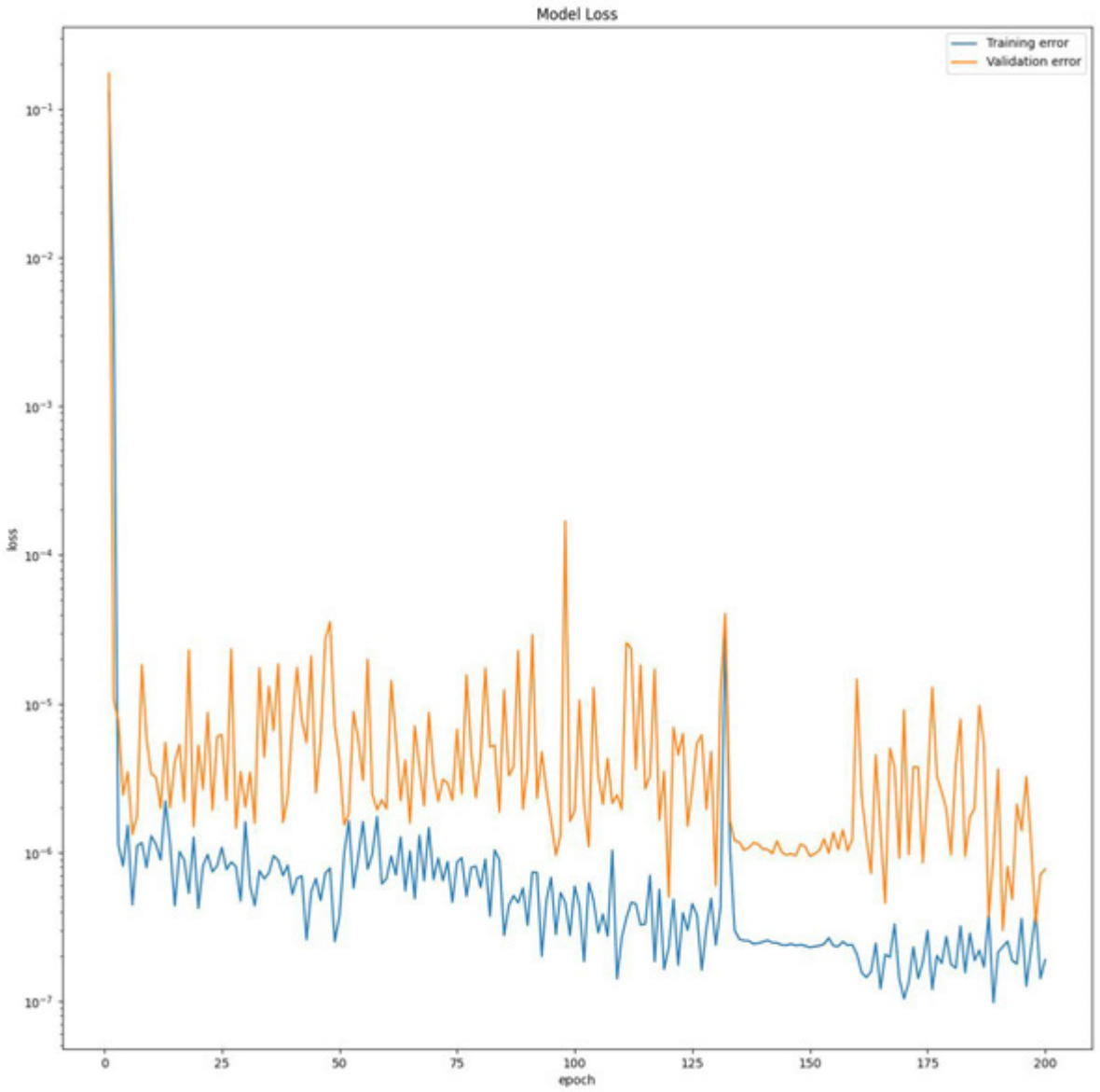

Fig. 69: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 22

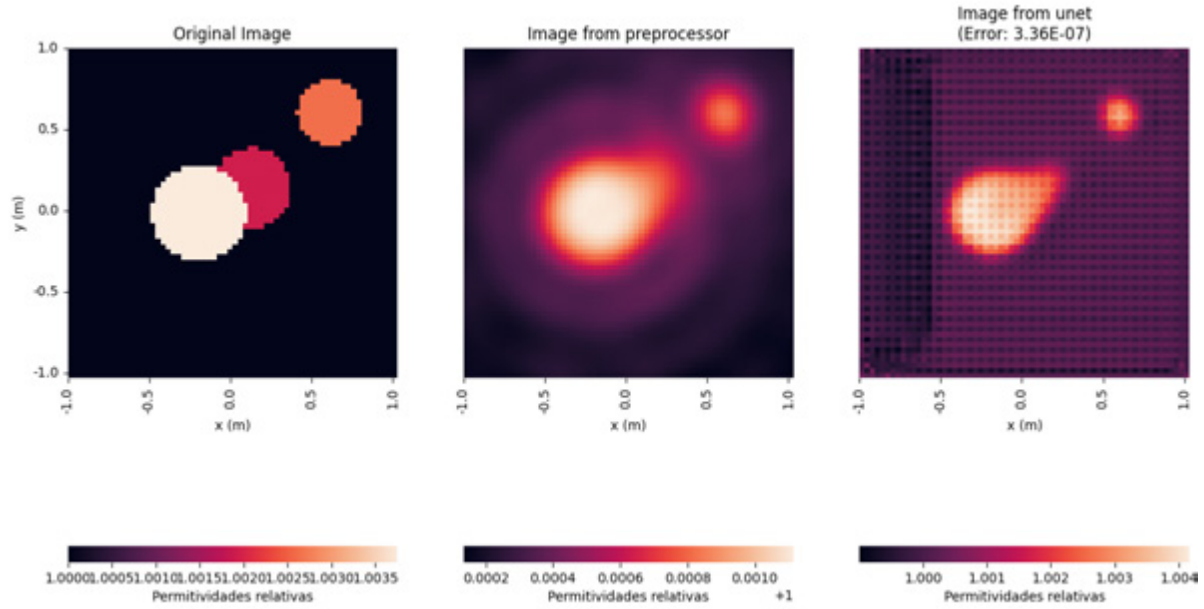

Fig. 70: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 22

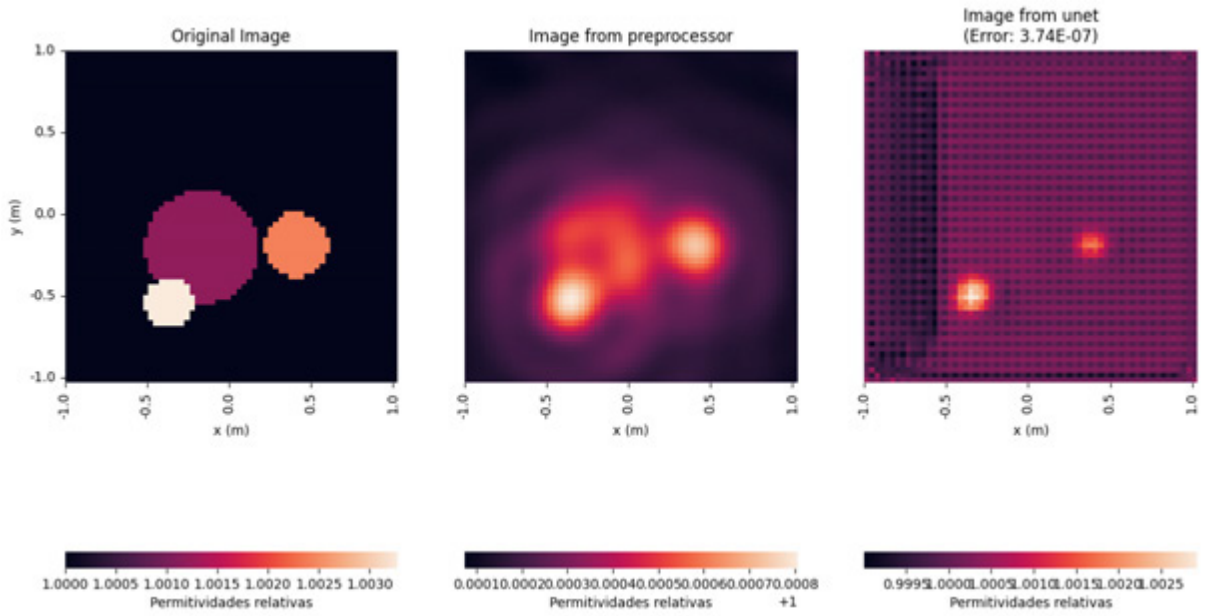

Fig. 71: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 22

Ejecución 23

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	200

Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,05

Tabla 44: Parámetros de ejecución 23

Métrica	**Valor**
Duración del entrenamiento	2:34:08
Error total	$8,56\ x\ 10^{-4}$
Error medio	$1,14\ x\ 10^{-5}$
Desvío estándar	$1,75\ x\ 10^{-5}$

Tabla 45: Resultados de ejecución 23

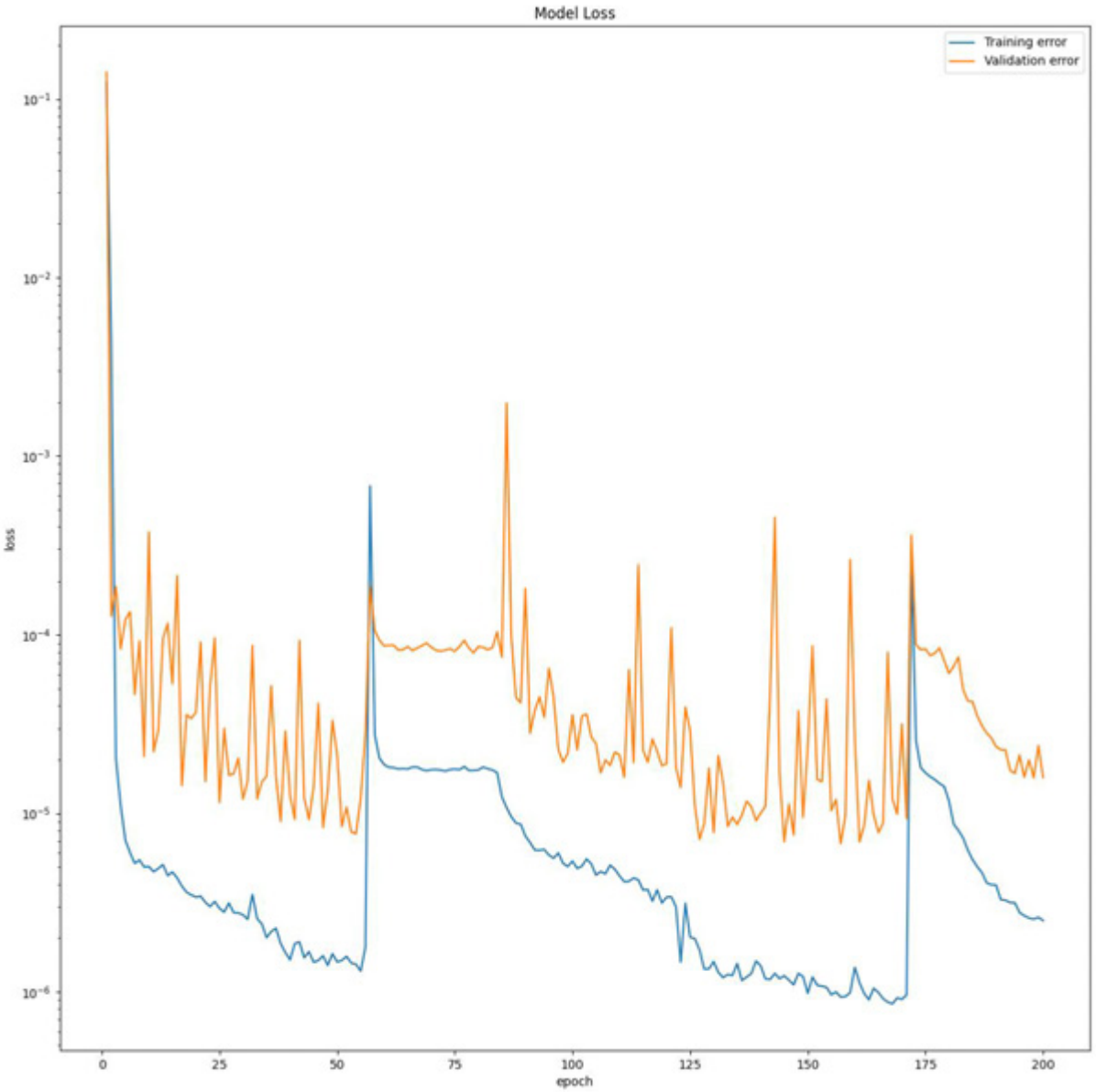

Fig. 72: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 23

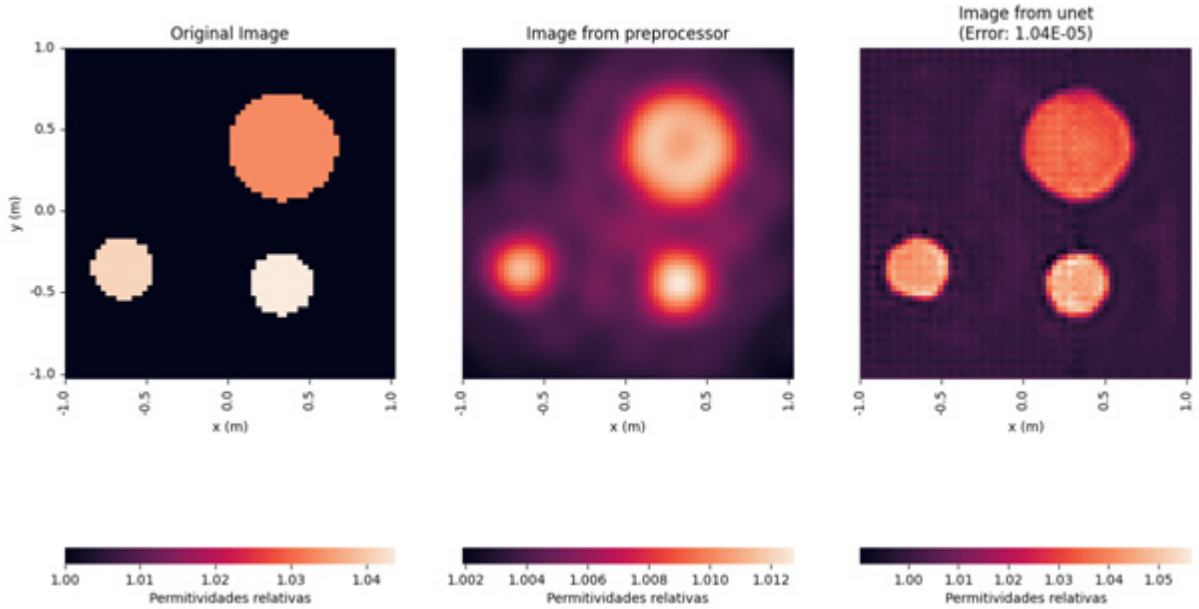

Fig. 73: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 23

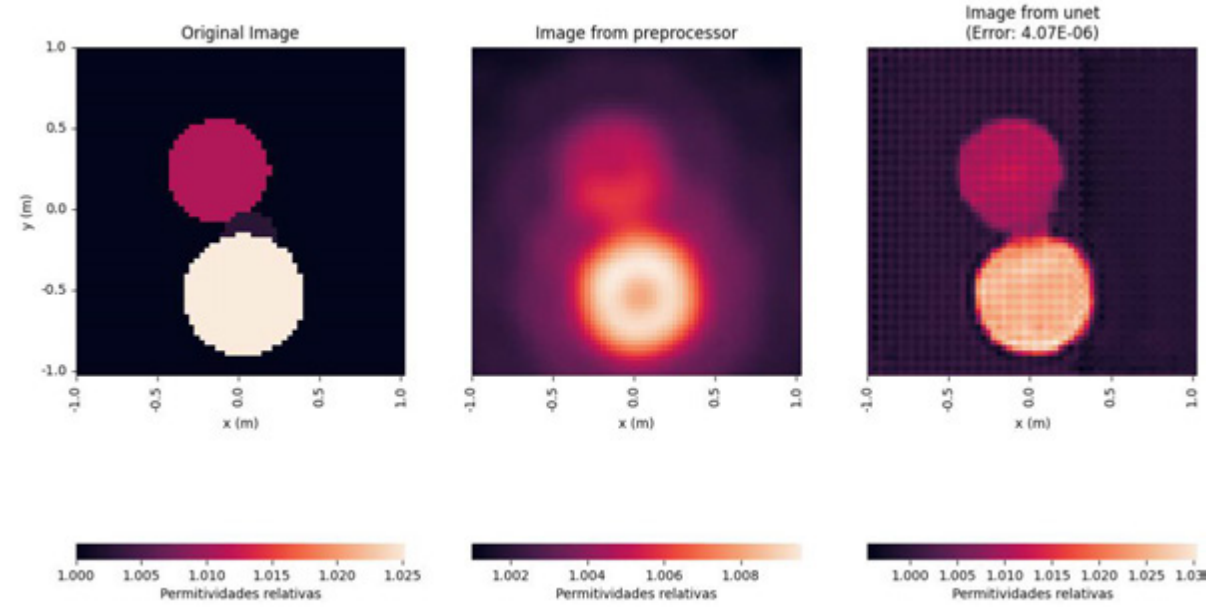

Fig. 74: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 23

Ejecución 24

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	200
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 46: Parámetros de ejecución 24

Para ver los resultados de la ejecución, remitirse a la ejecución 16.

Ejecución 25

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 2

Tabla 47: Parámetros de ejecución 25

Métrica	**Valor**
Duración del entrenamiento	2:45:28
Error total	$3,15\ x\ 10^{-1}$
Error medio	$4,20\ x\ 10^{-3}$
Desvío estándar	$4,52\ x\ 10^{-3}$

Tabla 48: Resultados de ejecución 25

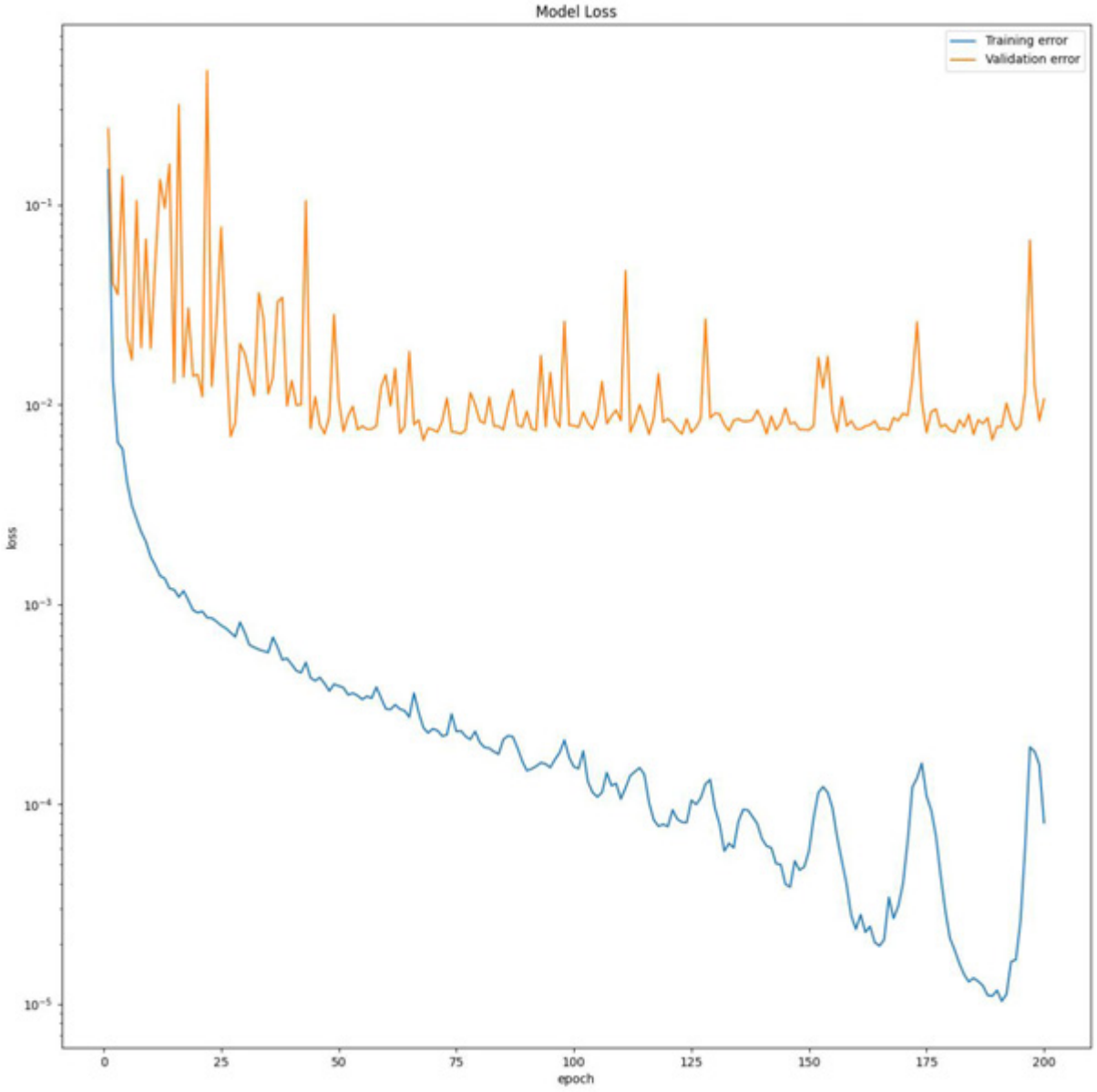

Fig. 75: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 25

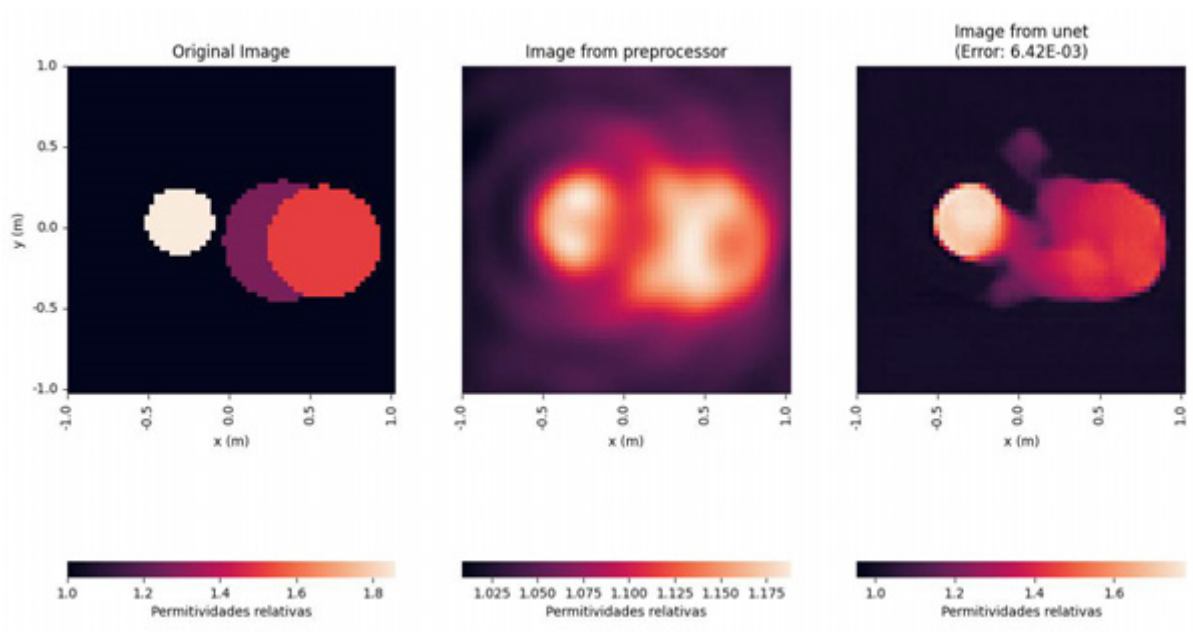

Fig. 76: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 25

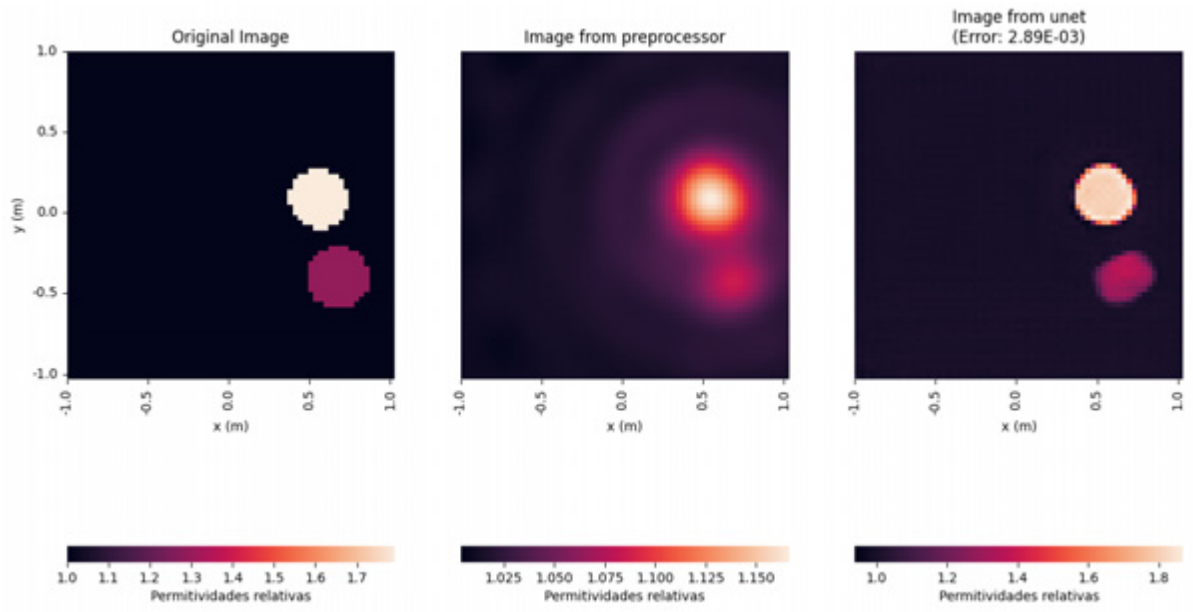

Fig. 77: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 25

Ejecución 26

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	350 (70 %)

Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 5

Tabla 49: Parámetros de ejecución 26

Métrica	**Valor**
Duración del entrenamiento	2:41:28
Error total	$1,49\ x\ 10^{-1}$
Error medio	$1,98\ x\ 10^{-1}$
Desvío estándar	$2,06\ x\ 10^{-1}$

Tabla 50: Resultados de ejecución 26

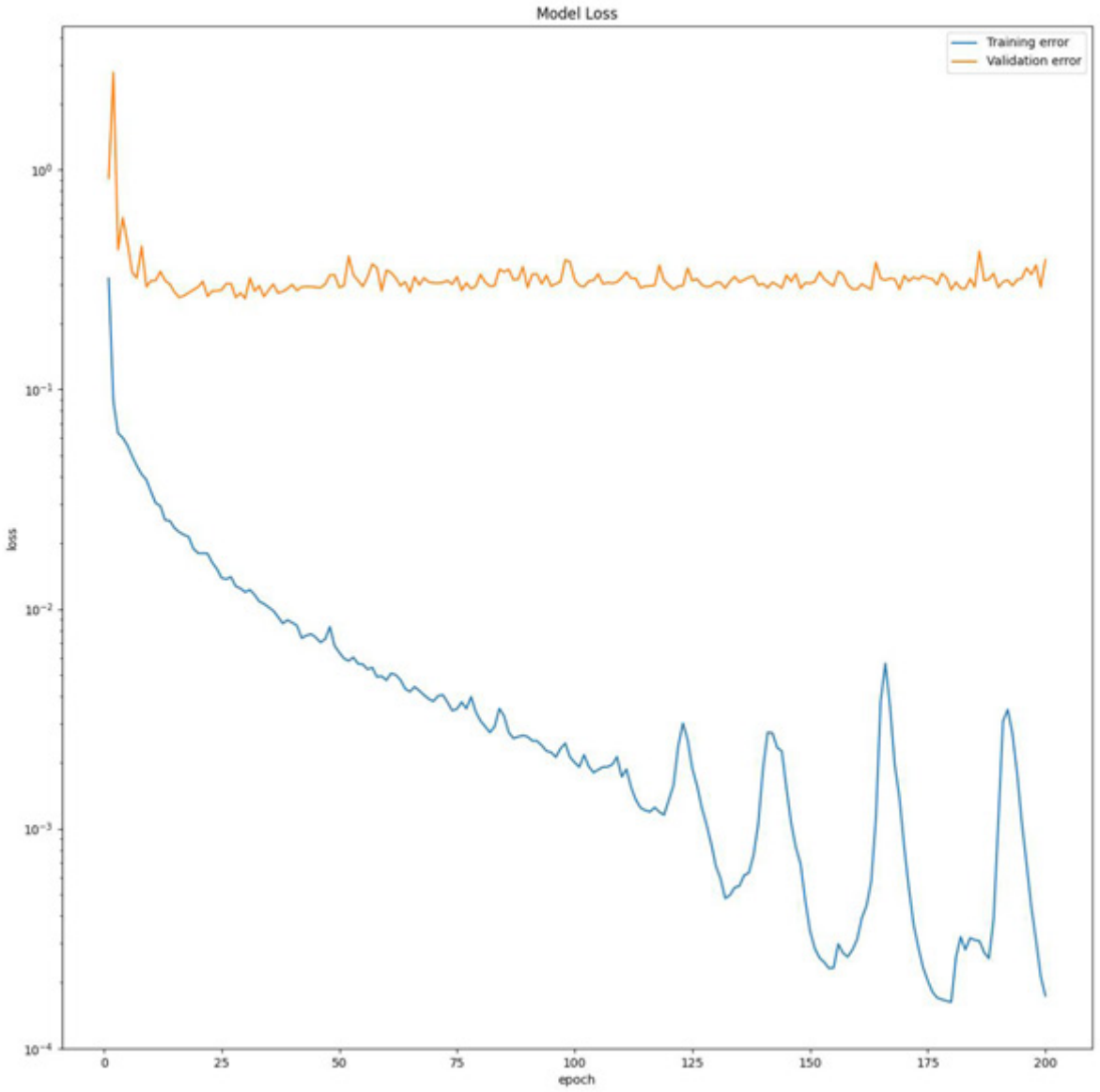

Fig. 78: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 26

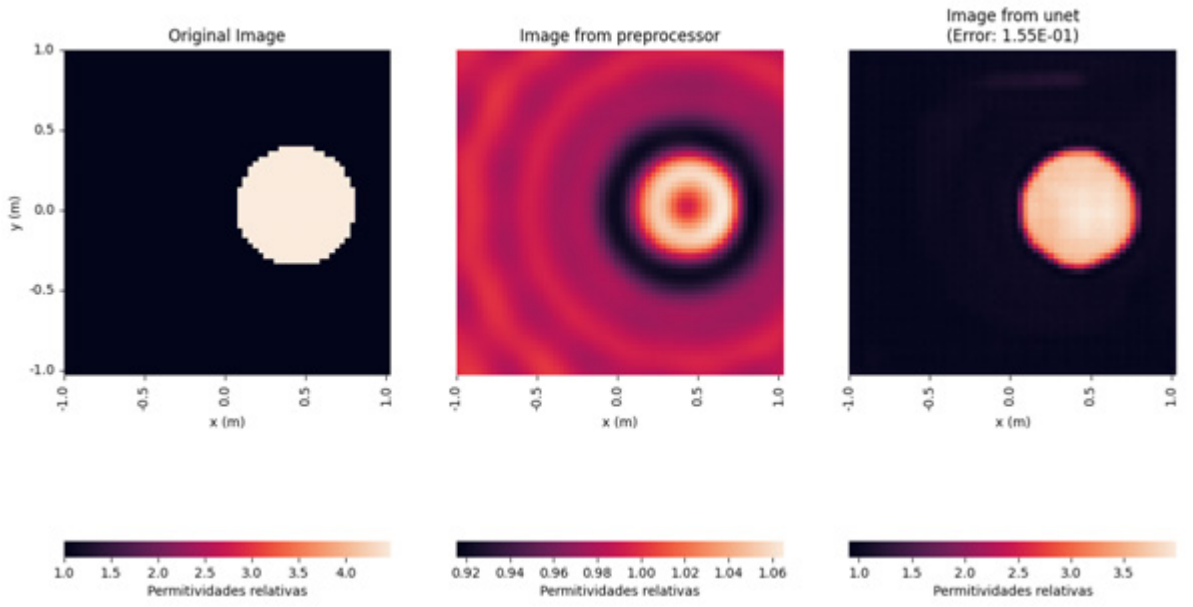

Fig. 79: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 26

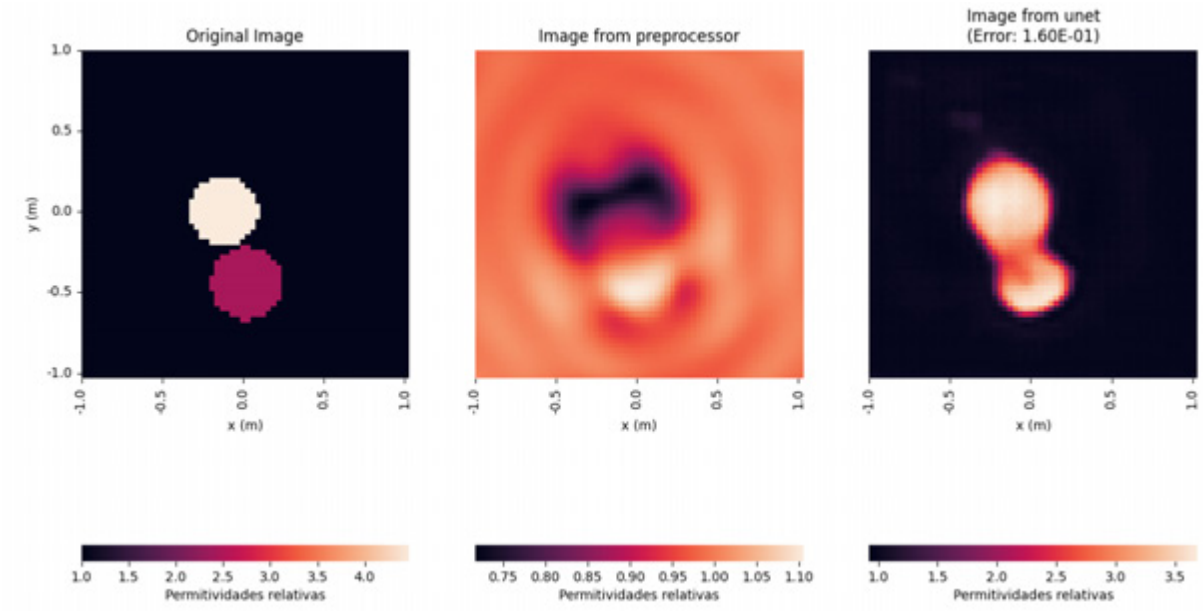

Fig. 80: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 26

Ejecución con GPU

Algunas computadoras actuales poseen una unidad de procesamiento gráfica (GPU) dedicada para realizar operaciones necesarias para renderizar gráficos y así liberar ciclos de CPU para otras tareas. También están optimizadas para deep learning ya que pueden procesar múltiples operaciones en paralelo. En redes neuronales, esto deriva en menores tiempos de entrenamiento ante tamaños de batch mayores [9].

En cuanto a GenPer, está implementado de manera que pueda aprovechar estas ventajas si es ejecutado en un entorno con GPU dedicada. Se realizaron distintas ejecuciones con distintos parámetros en Google Colaboratory, una plataforma desarrollada por Google que permite ejecutar código con una GPU disponible [10].

Ejecución 27

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 51: Parámetros de ejecución 27

Métrica	**Valor**
Duración del entrenamiento	4:13:18
Error total	$4,96\ x\ 10^{-2}$
Error medio	$6,61\ x\ 10^{-4}$
Desvío estándar	$7,05\ x\ 10^{-4}$

Tabla 52: Resultados de ejecución 27

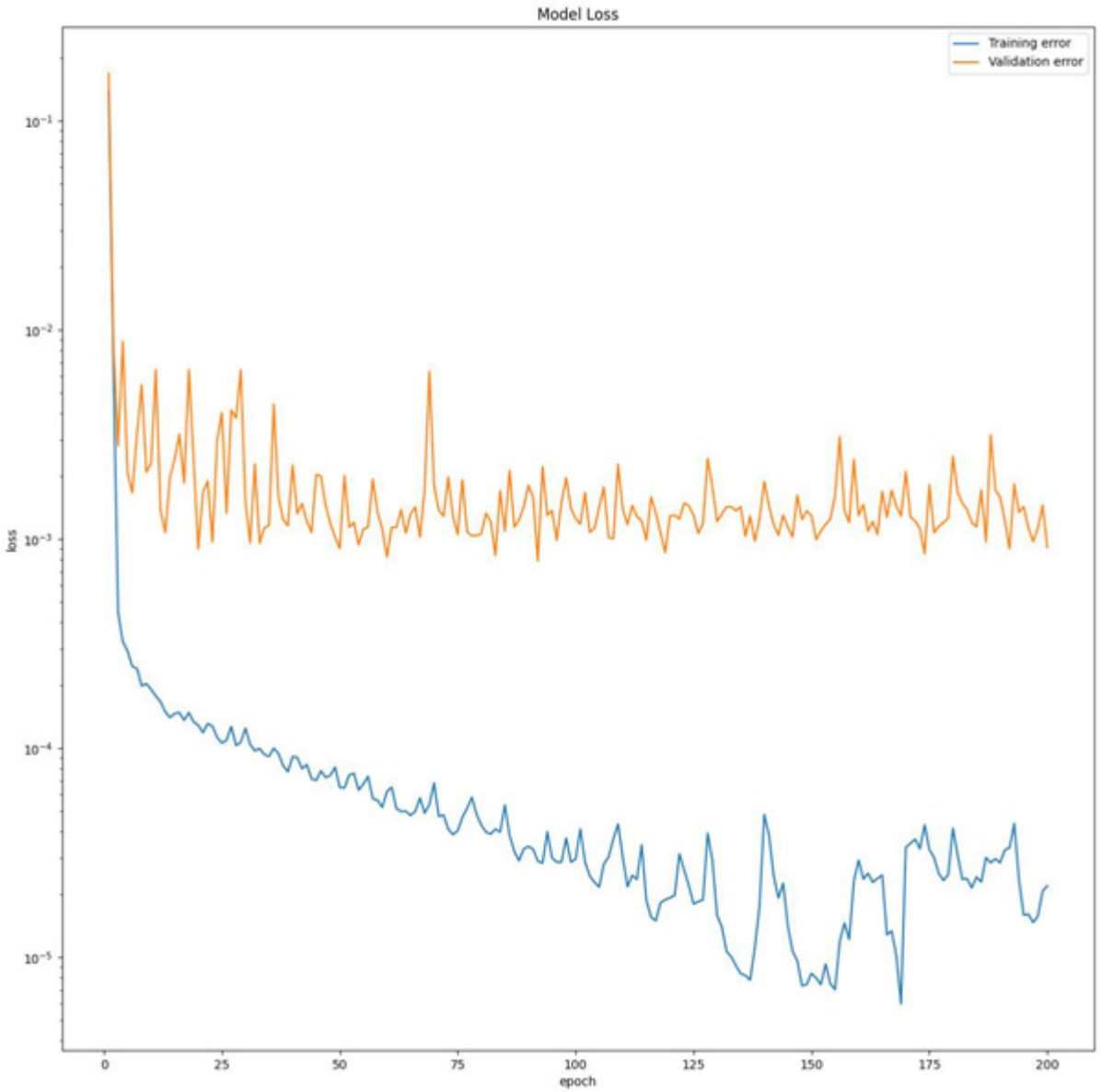

Fig. 81: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 27

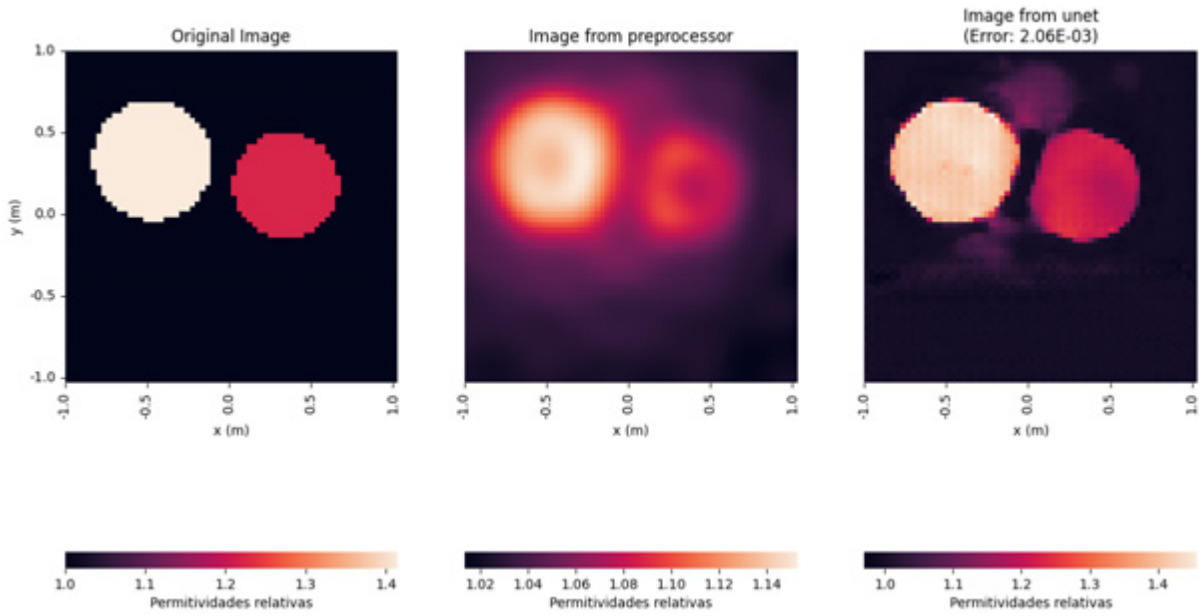

Fig. 82: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 27

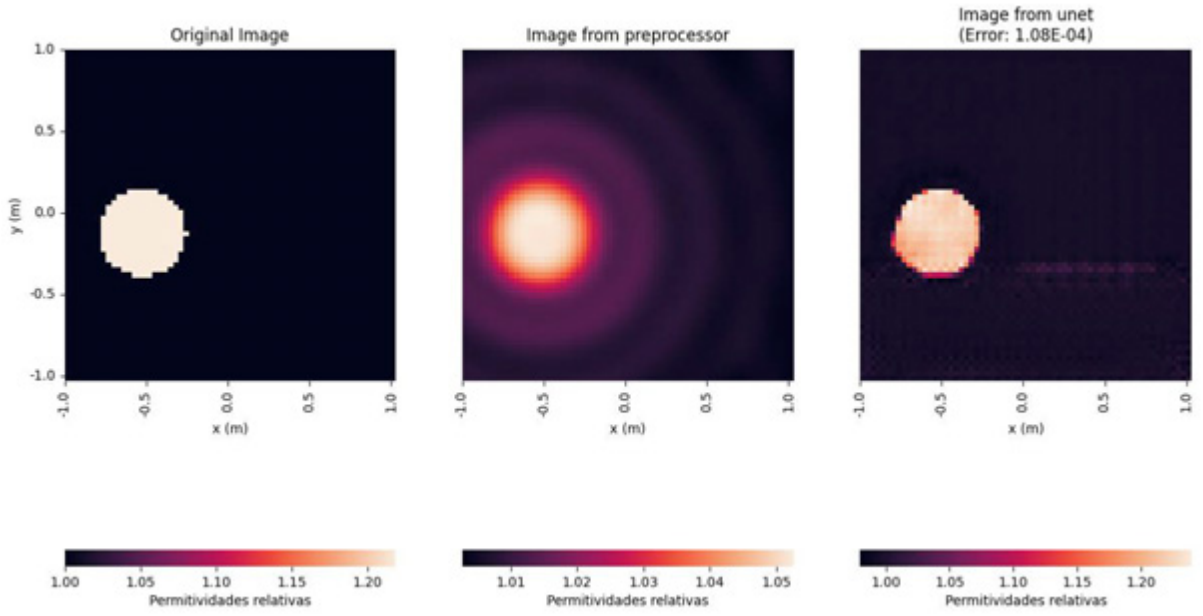

Fig. 83: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 27

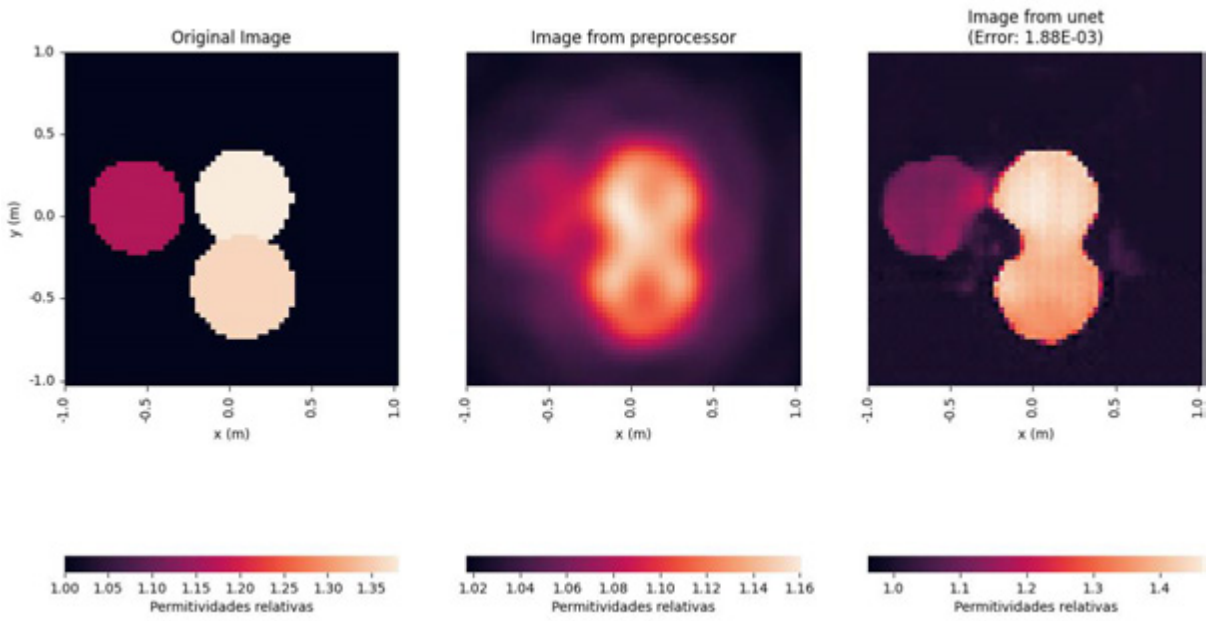

Fig. 84: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 27

Ejecución 28

Parámetro	Valor
Imágenes	500
Imágenes por batch	2
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 53: Parámetros de ejecución 28

Métrica	Valor
Duración del entrenamiento	2:30:13
Error total	$1,93\ x\ 10^{-2}$
Error medio	$5,22\ x\ 10^{-4}$

Desvío estándar	$4,46\ x\ 10^{-4}$

Tabla 54: Resultados de ejecución 28

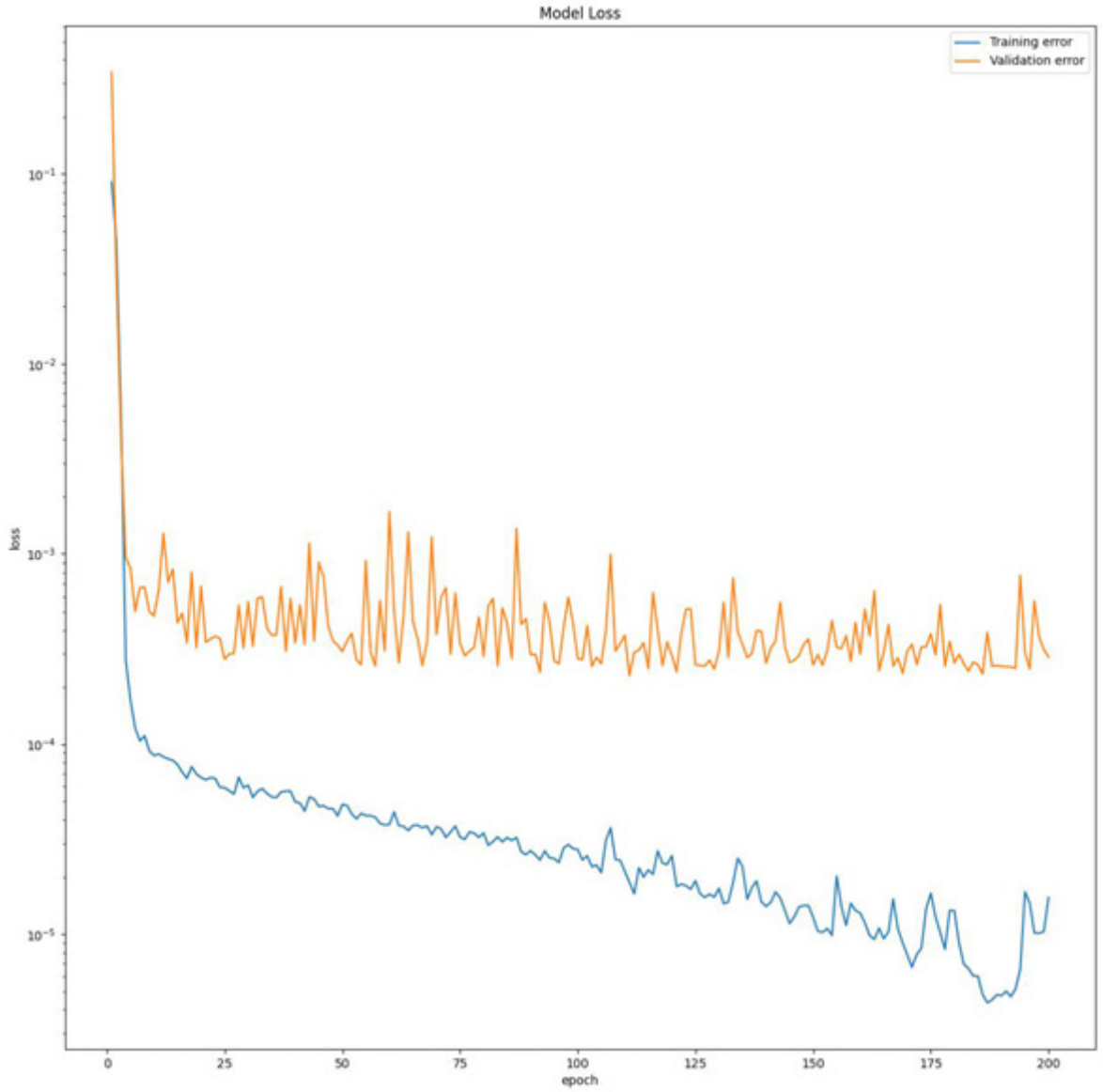

Fig. 85: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 28

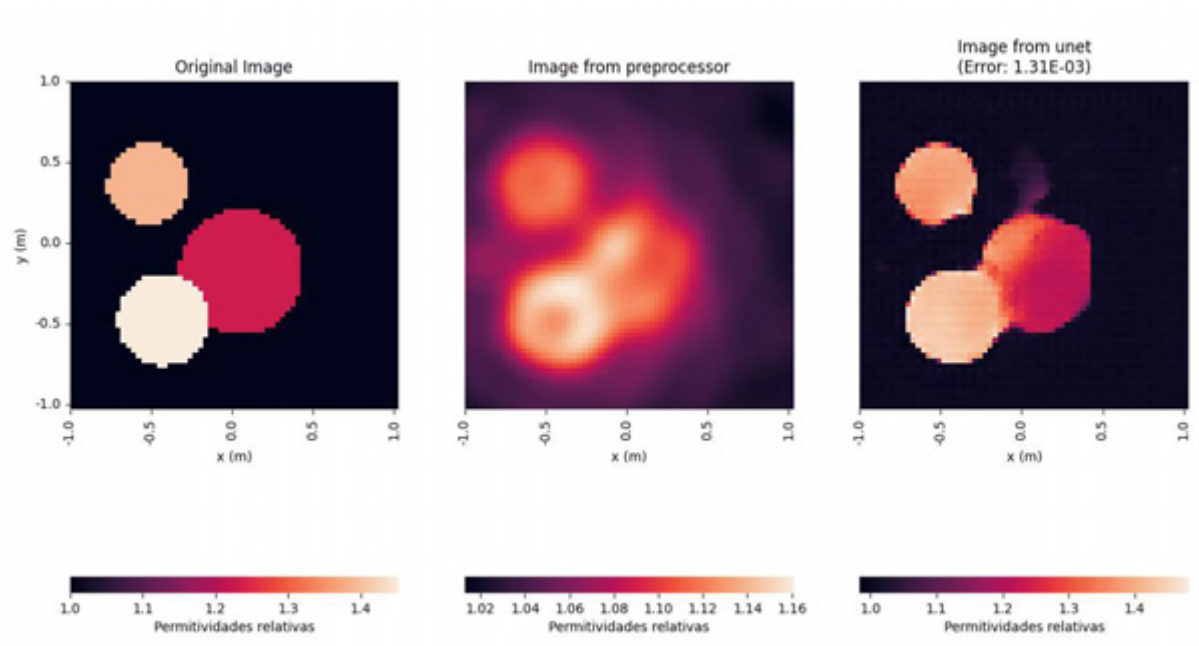

Fig. 86: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 28

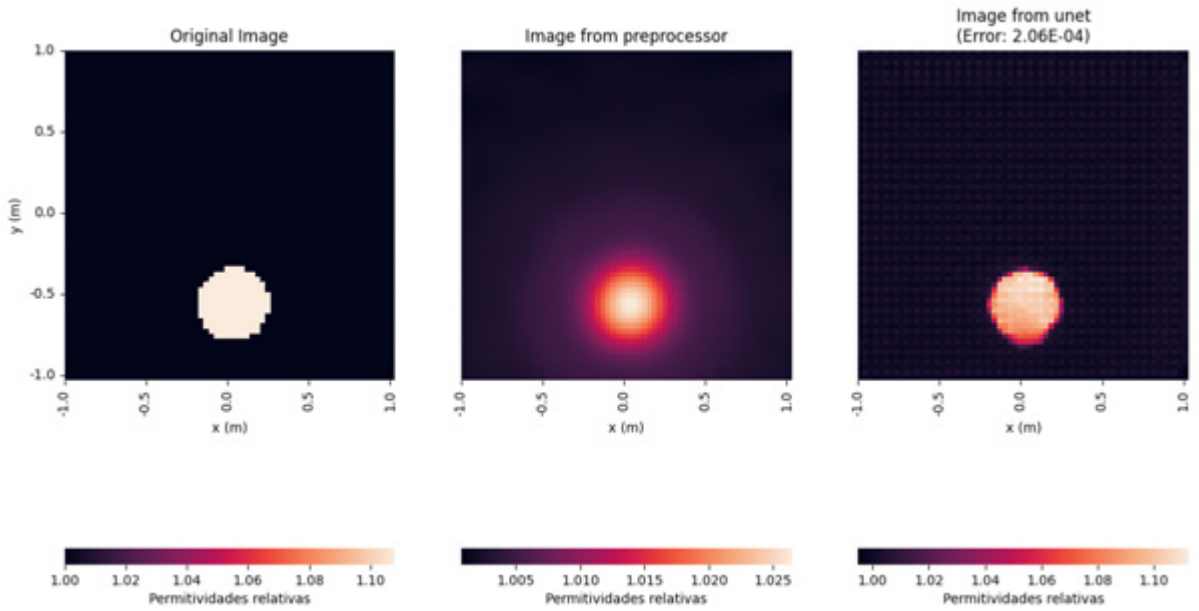

Fig. 87: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 28

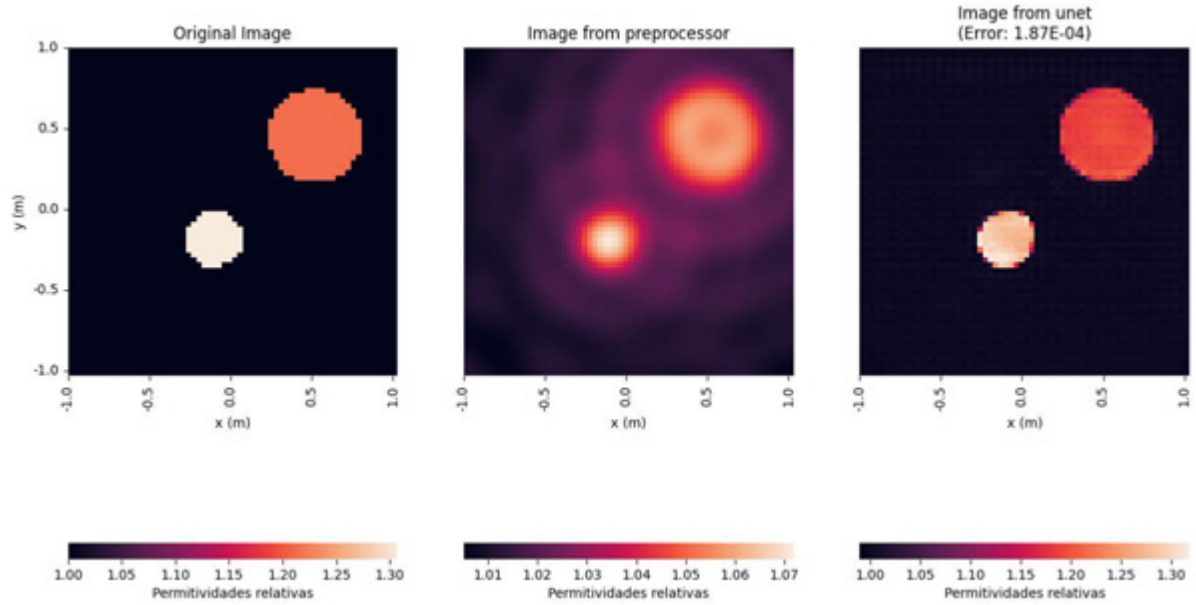

Fig. 88: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 28

Ejecución 29

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	4
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 55: Parámetros de ejecución 29

Métrica	**Valor**
Duración del entrenamiento	1:20:48
Error total	$7,55\ x\ 10^{-3}$
Error medio	$4,19\ x\ 10^{-4}$
Desvío estándar	$2,87\ x\ 10^{-4}$

Tabla 56: Resultados de ejecución 29

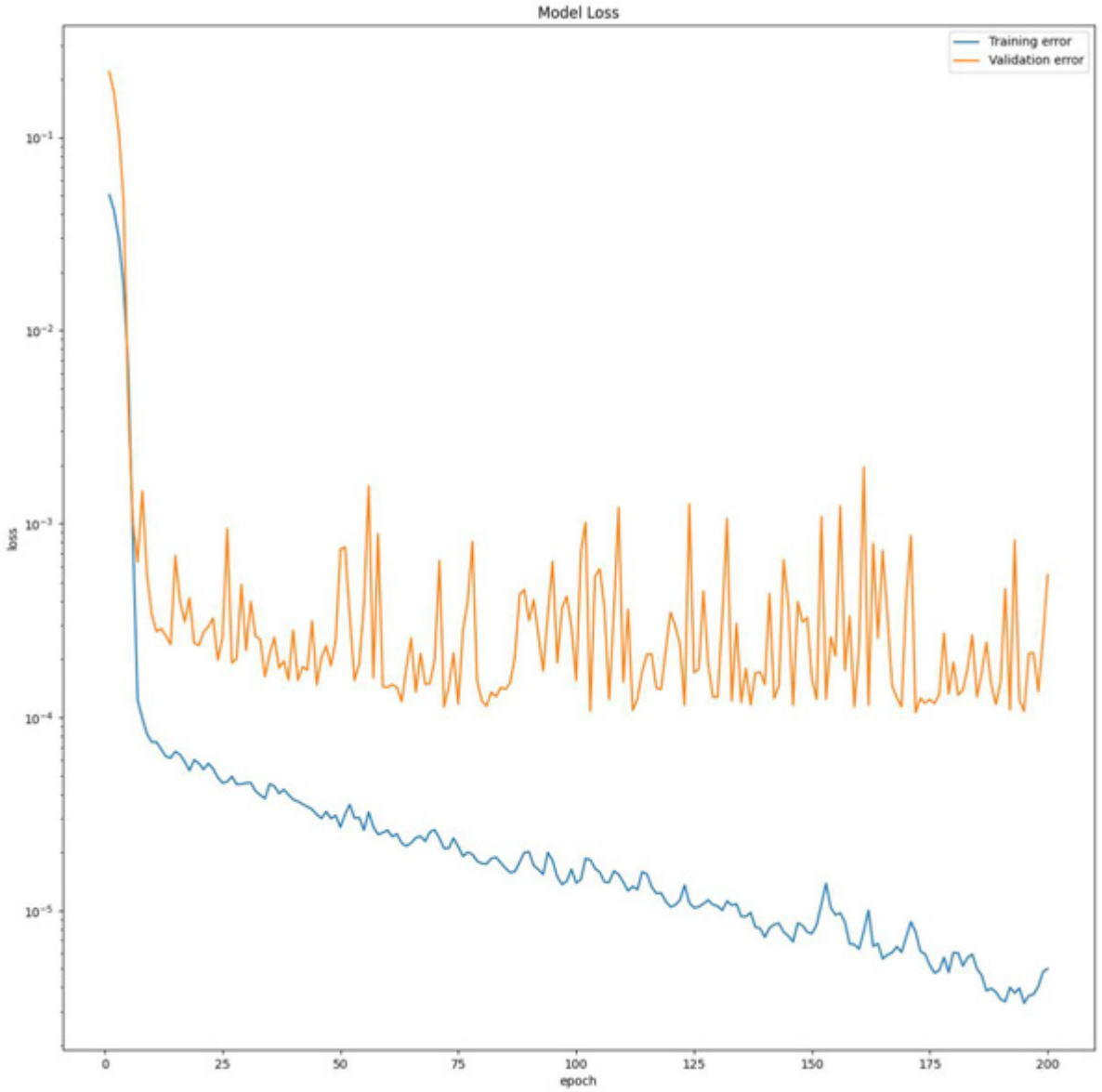

Fig. 89: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 29

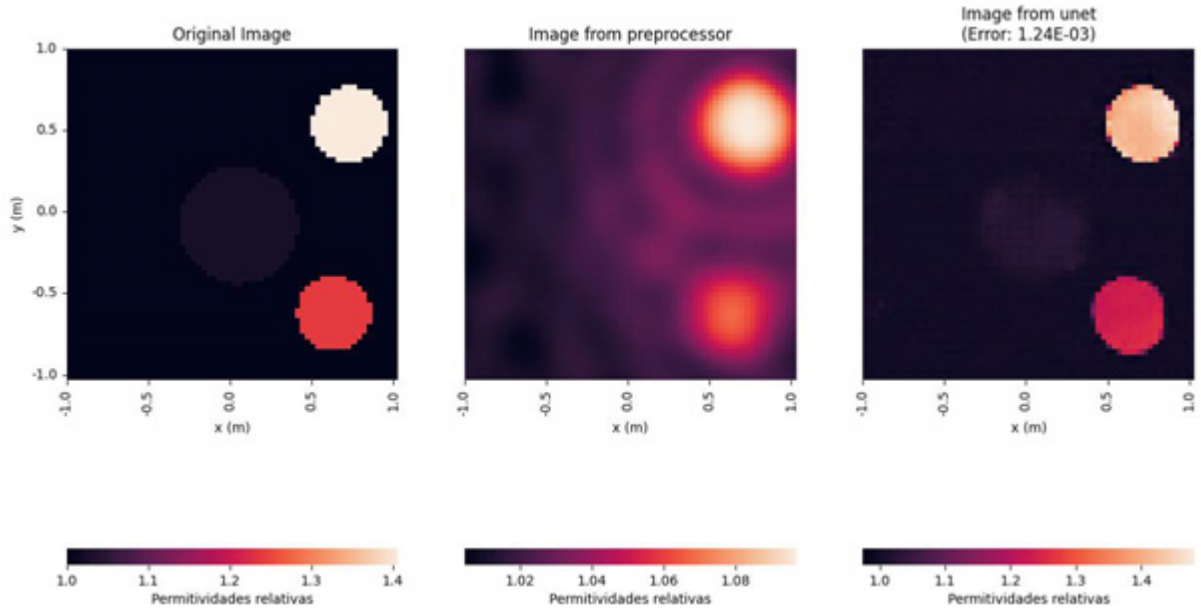

Fig. 90: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 29

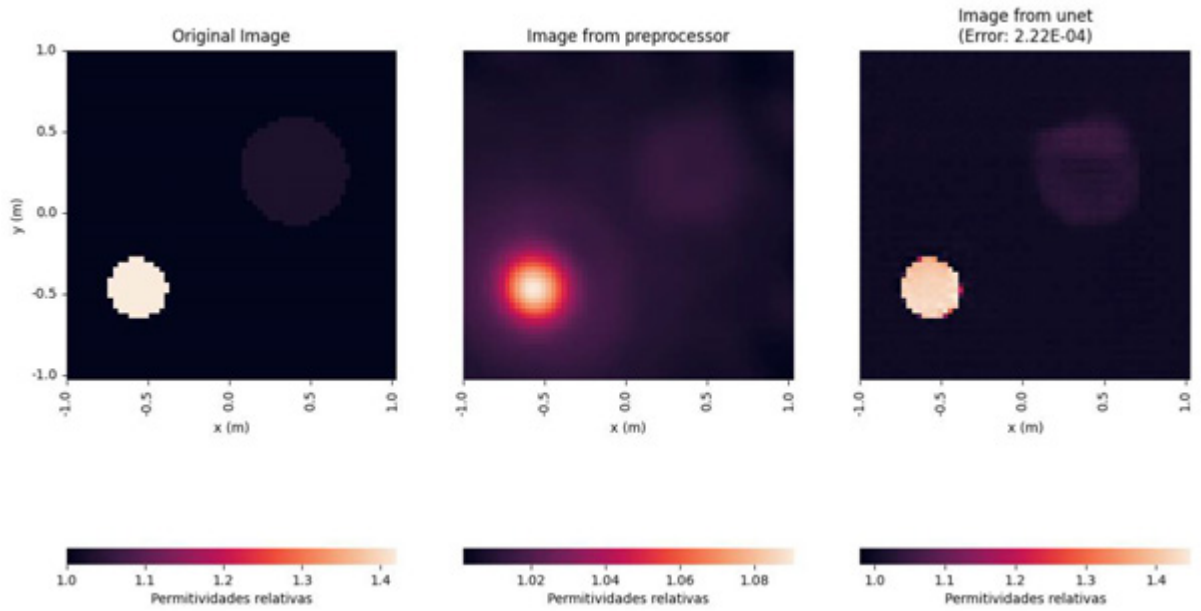

Fig. 91: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 29

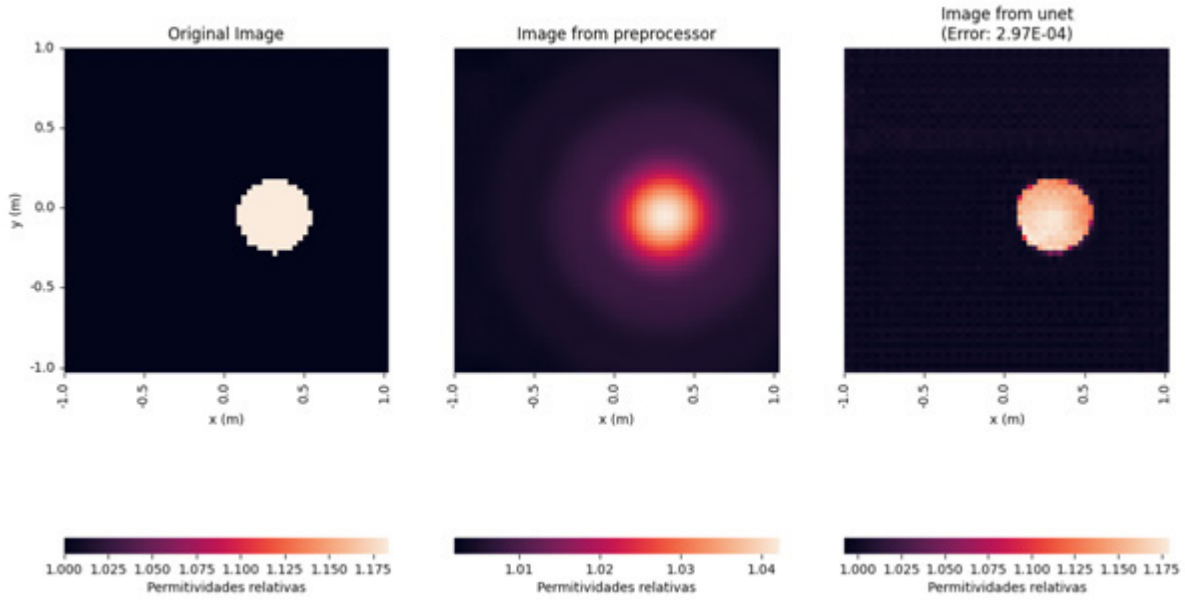

Fig. 92: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 29

Ejecución 30

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	8
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 57: Parámetros de ejecución 30

Métrica	**Valor**
Duración del entrenamiento	1:05:15
Error total	$4,40\ x\ 10^{-3}$
Error medio	$4,89\ x\ 10^{-4}$
Desvío estándar	$1,72\ x\ 10^{-4}$

Tabla 58: Resultados de ejecución 30

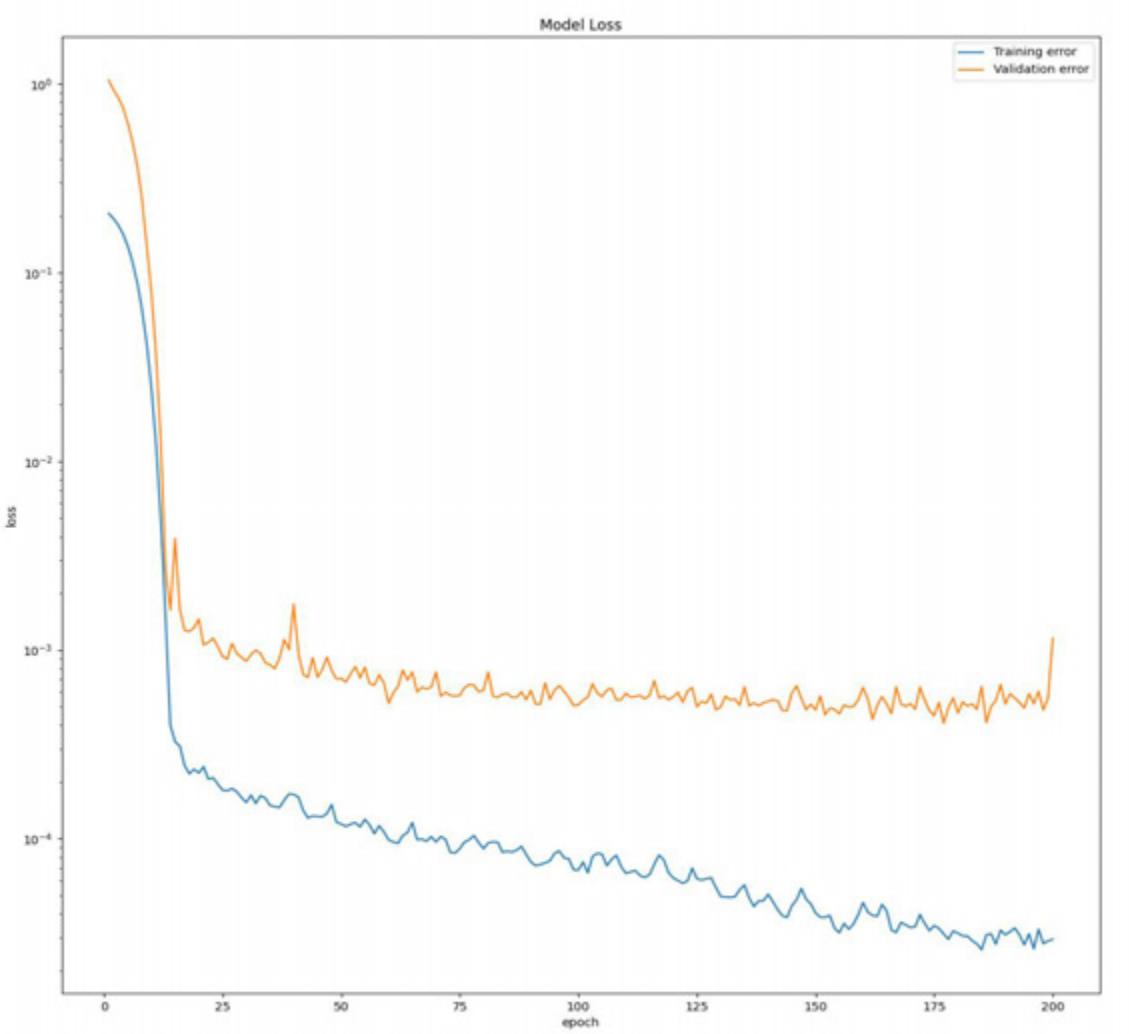

Fig. 93: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 30

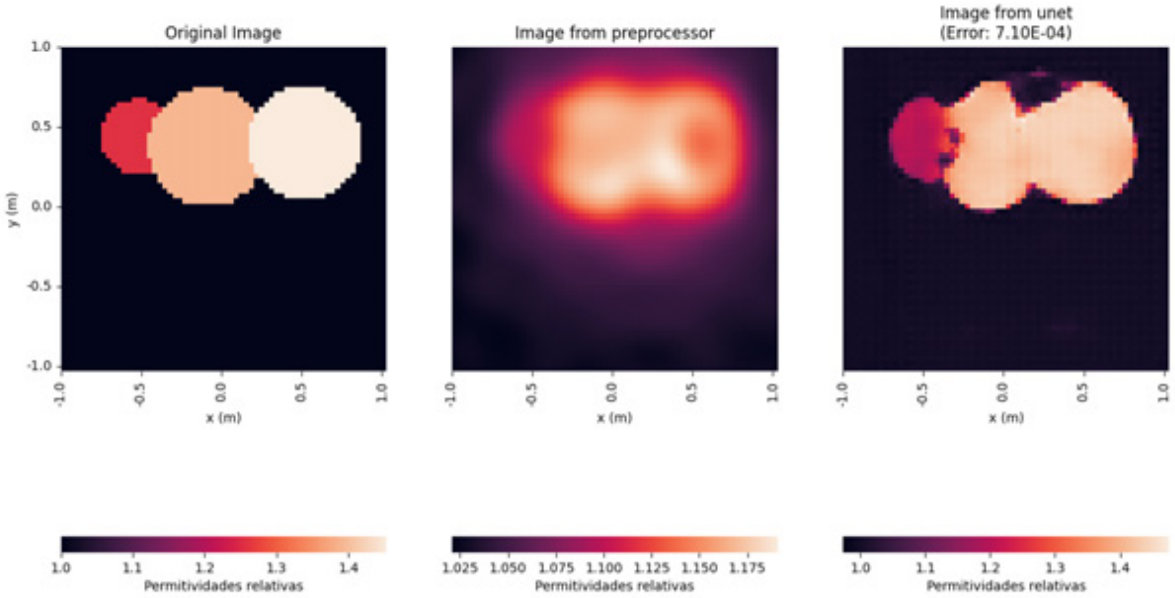

Fig. 94: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 30

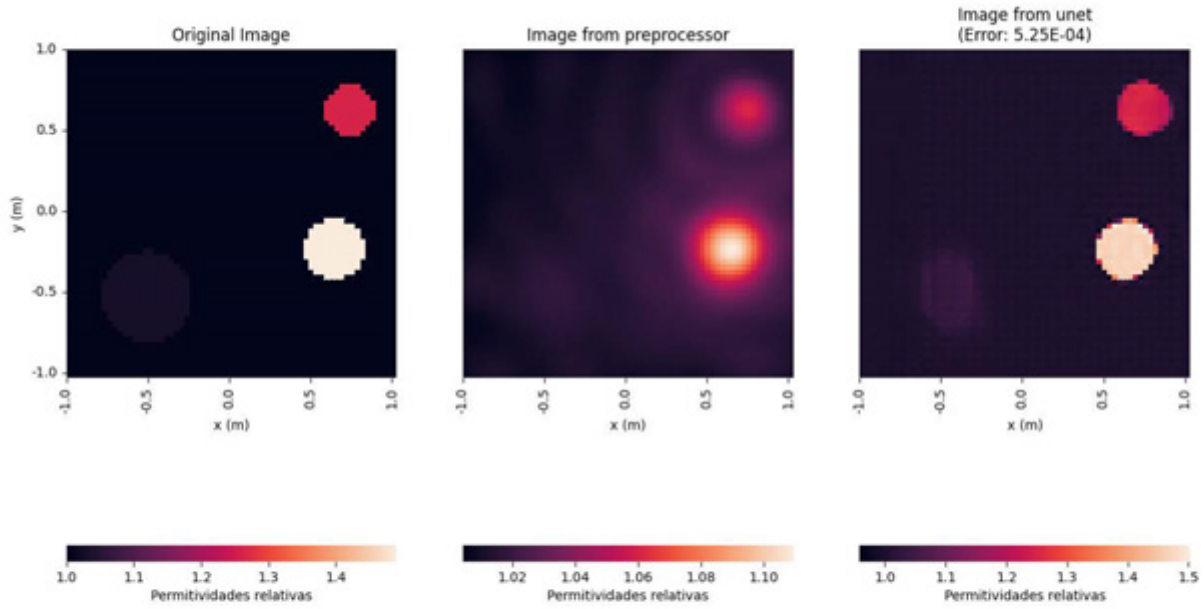

Fig. 95: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 30

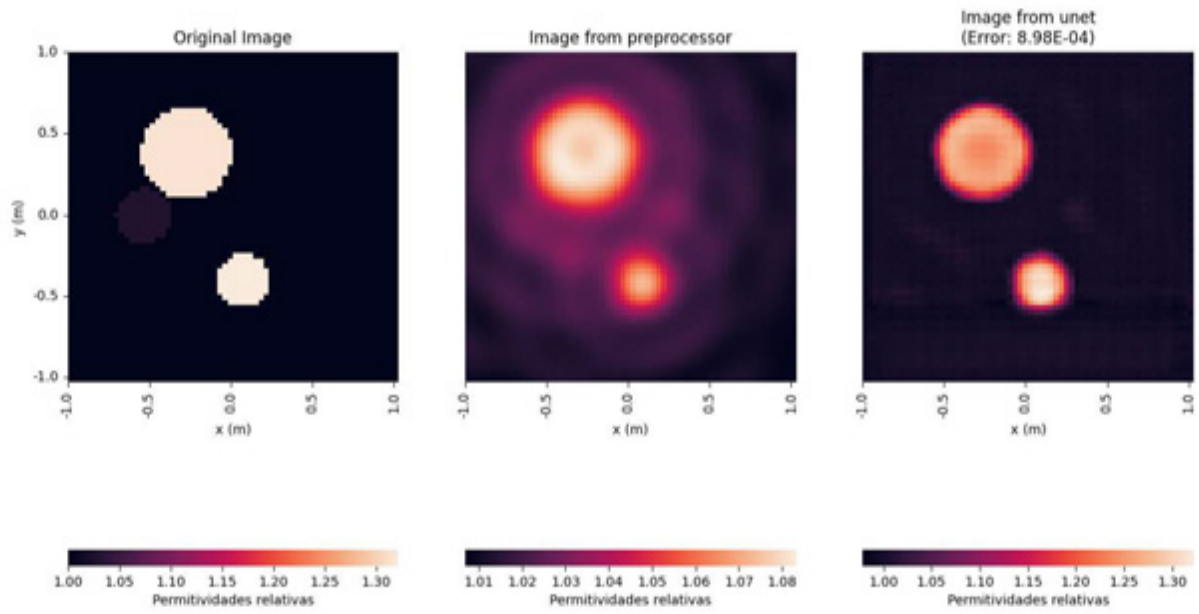

Fig. 96: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 30

Ejecución 31

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	16
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 59: Parámetros de ejecución 31

Métrica	**Valor**
Duración del entrenamiento	0:33:32
Error total	$1,81\ x\ 10^{-3}$
Error medio	$4,52\ x\ 10^{-4}$
Desvío estándar	$1,30\ x\ 10^{-4}$

Tabla 60: Resultados de ejecución 31

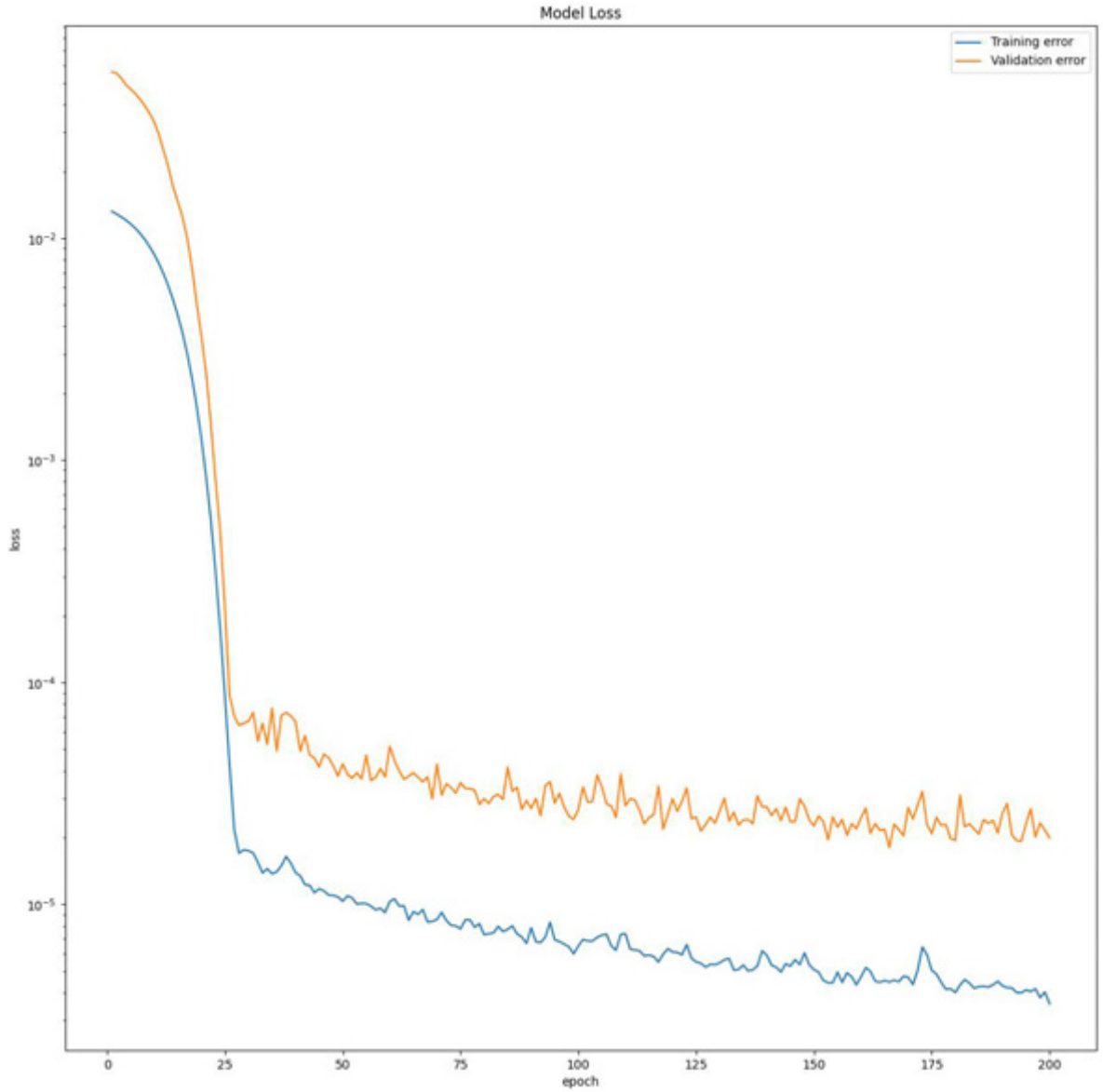

Fig. 97: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 31

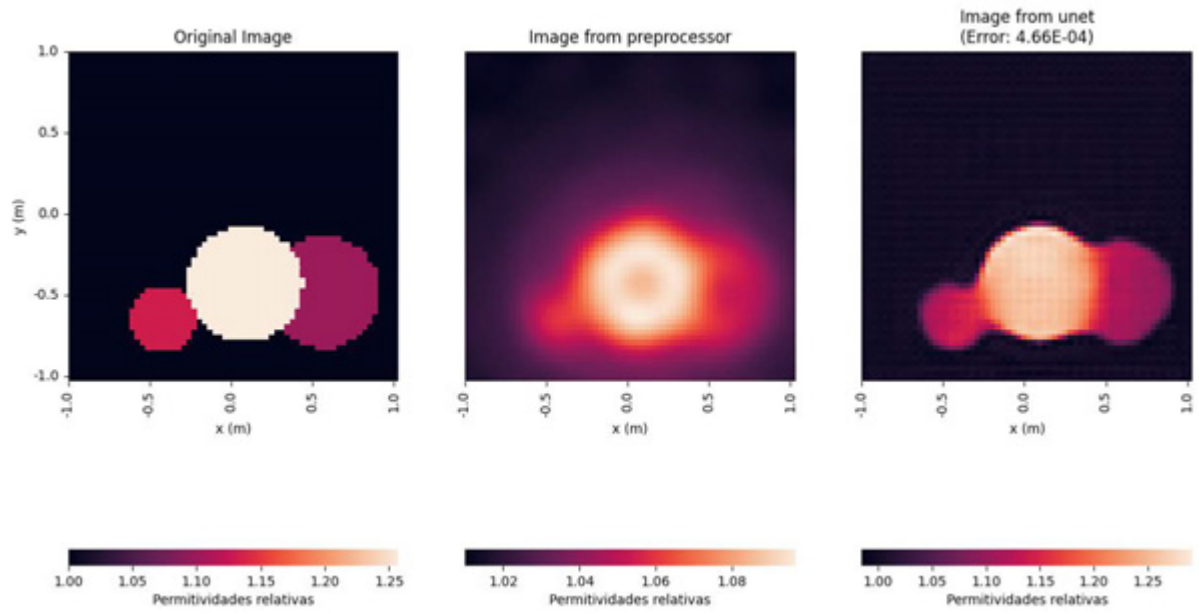

Fig. 98: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 31

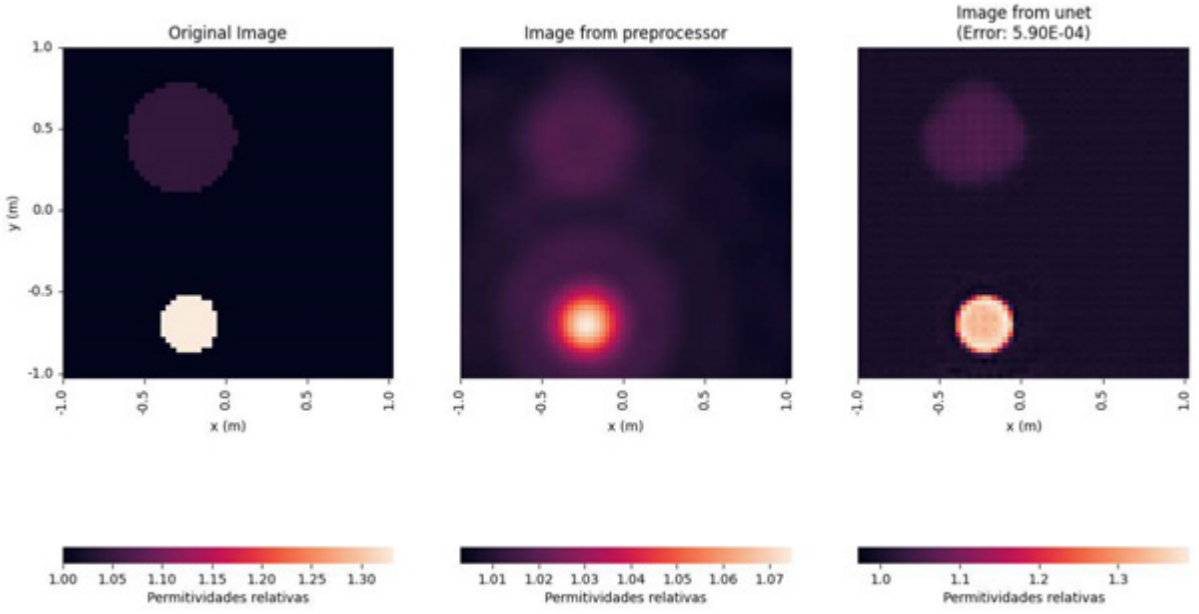

Fig. 99: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 31

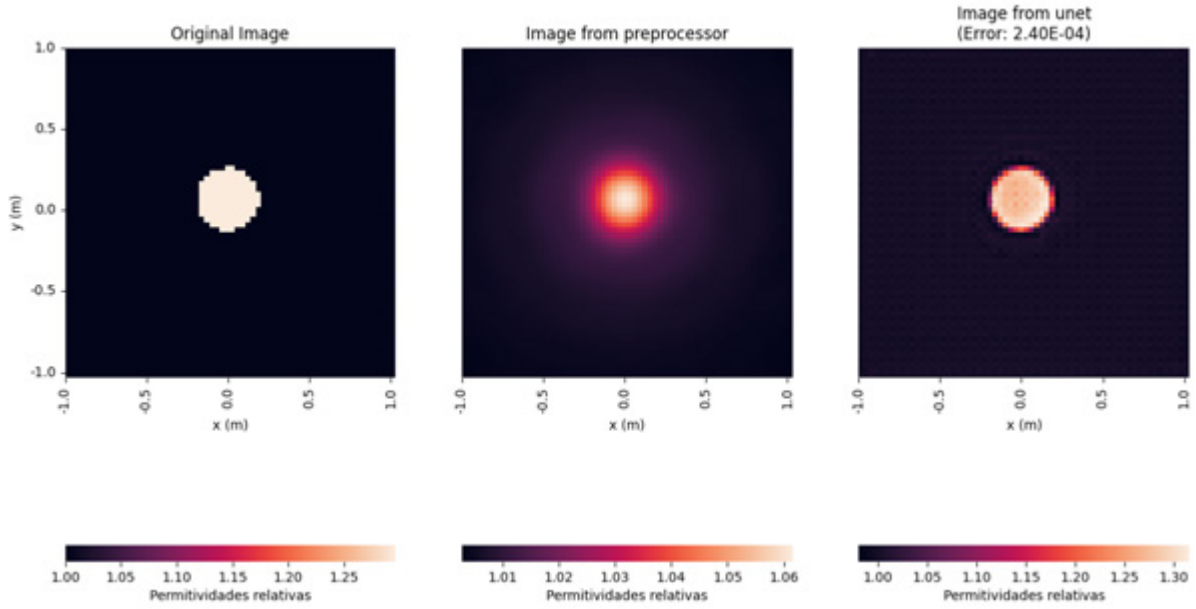

Fig. 100: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 31

Ejecución 32

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	32
Iteraciones	200
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 61: Parámetros de ejecución 32

Métrica	**Valor**
Duración del entrenamiento	0:26:08
Error total	$1,08\ x\ 10^{-3}$
Error medio	$5,42\ x\ 10^{-4}$
Desvío estándar	$8,30\ x\ 10^{-6}$

Tabla 62: Resultados de ejecución 32

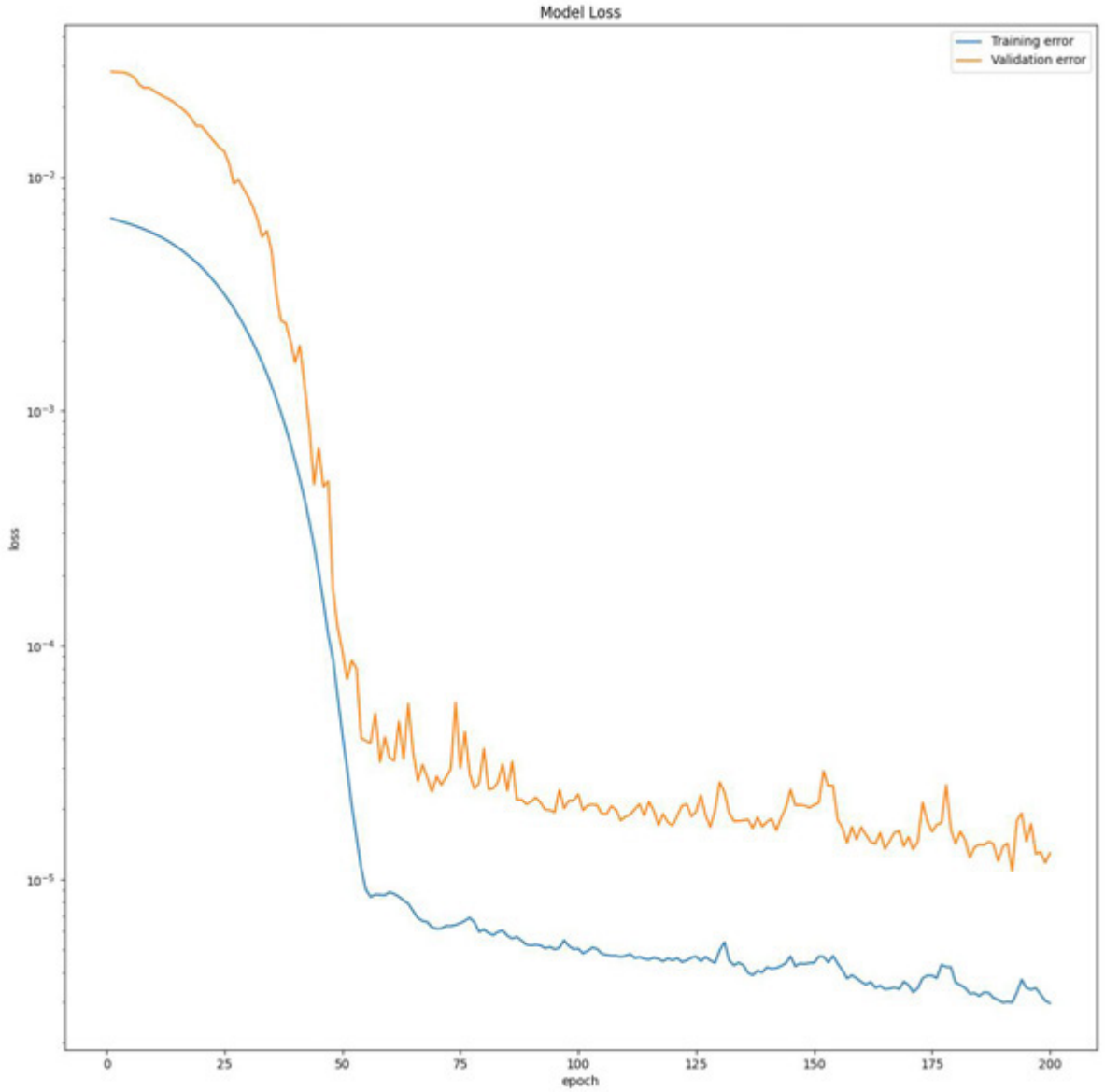

Fig. 101: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 32

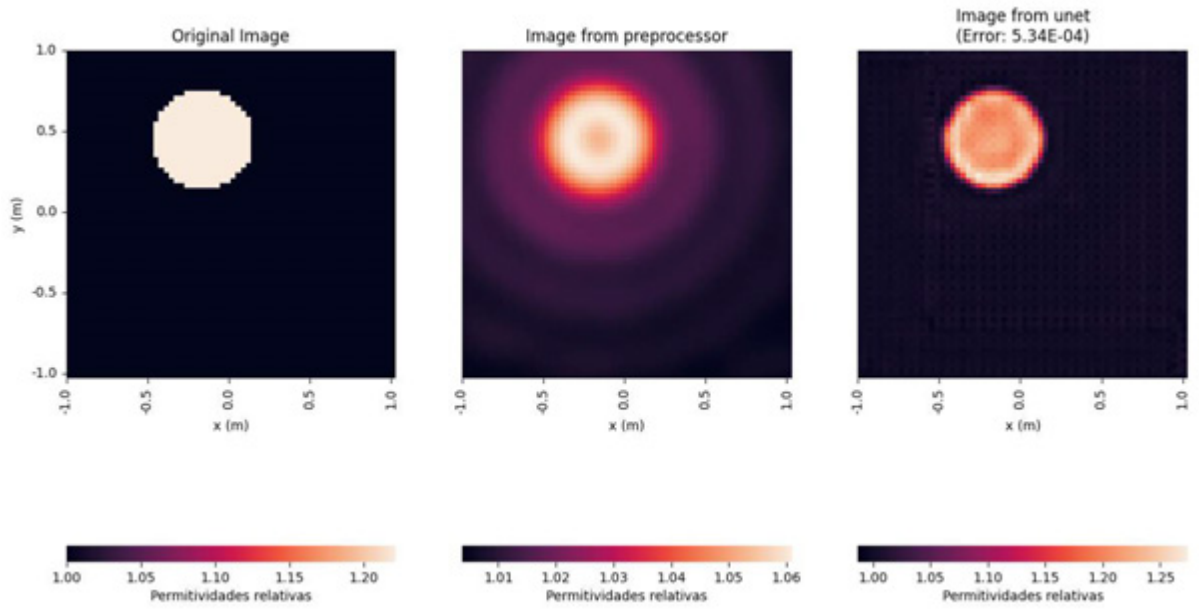

Fig. 102: Resultado del testeo de la red para una imagen de 1 círculo en ejecución 32

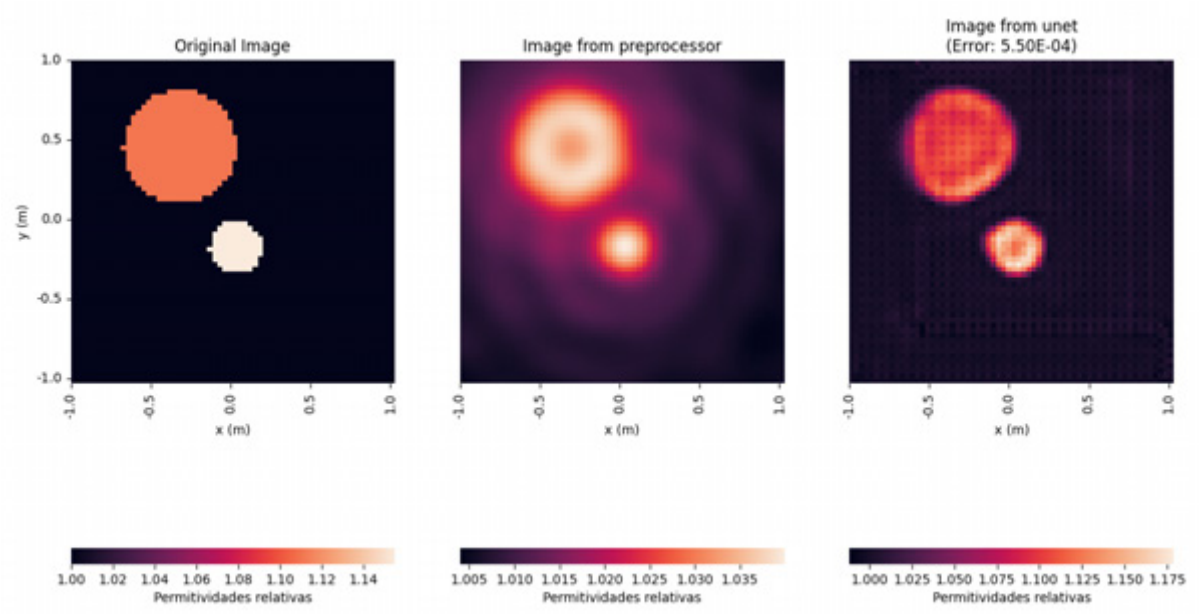

Fig. 103: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 32

Ejecución 33

Parámetro	Valor
Imágenes	5.000
Imágenes por batch	16
Iteraciones	50

Set de entrenamiento	3.500 (70 %)
Set de validación	750 (15 %)
Set de testing	750 (15 %)
Rango de permitividades	1 - 1,5

Tabla 63: Parámetros de ejecución 33

Métrica	**Valor**
Duración del entrenamiento	1:10:08
Error total	$1,79\ x\ 10^{-2}$
Error medio	$3,90\ x\ 10^{-4}$
Desvío estándar	$1,04\ x\ 10^{-4}$

Tabla 64: Resultados de ejecución 33

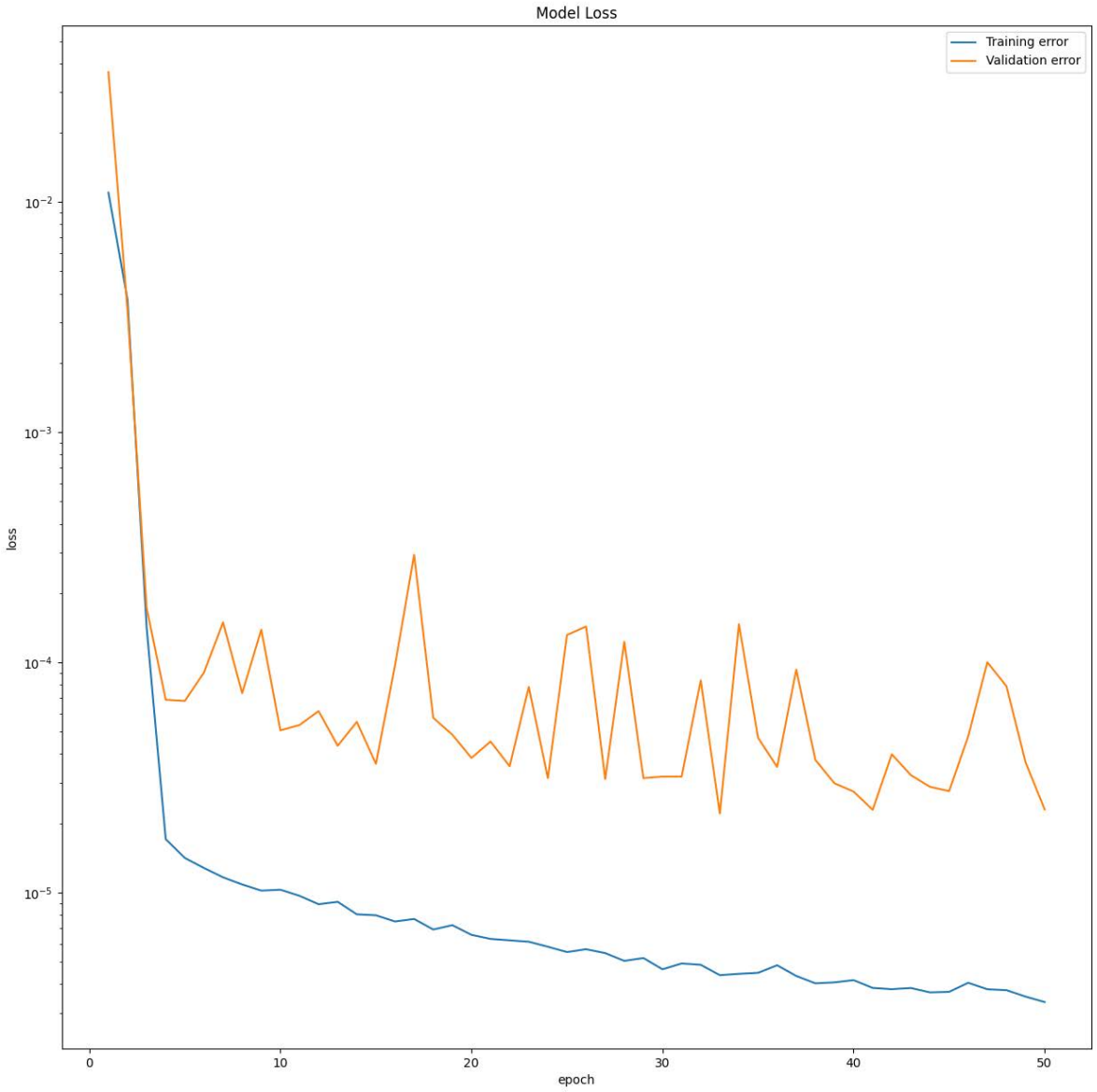

Fig. 104: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 33

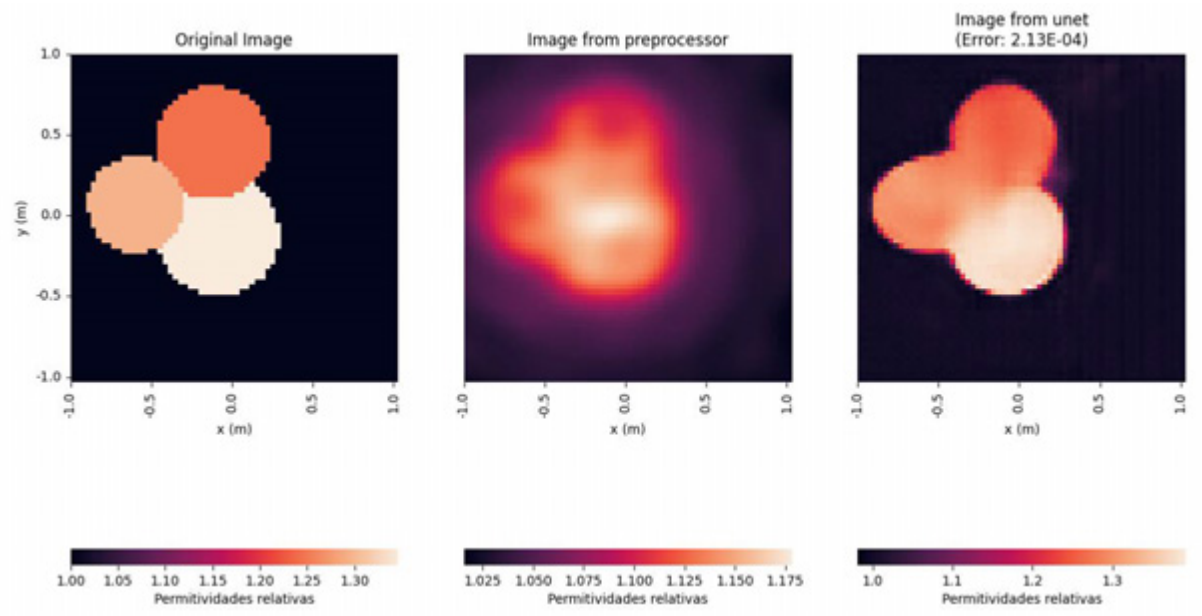

Fig. 105: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 33

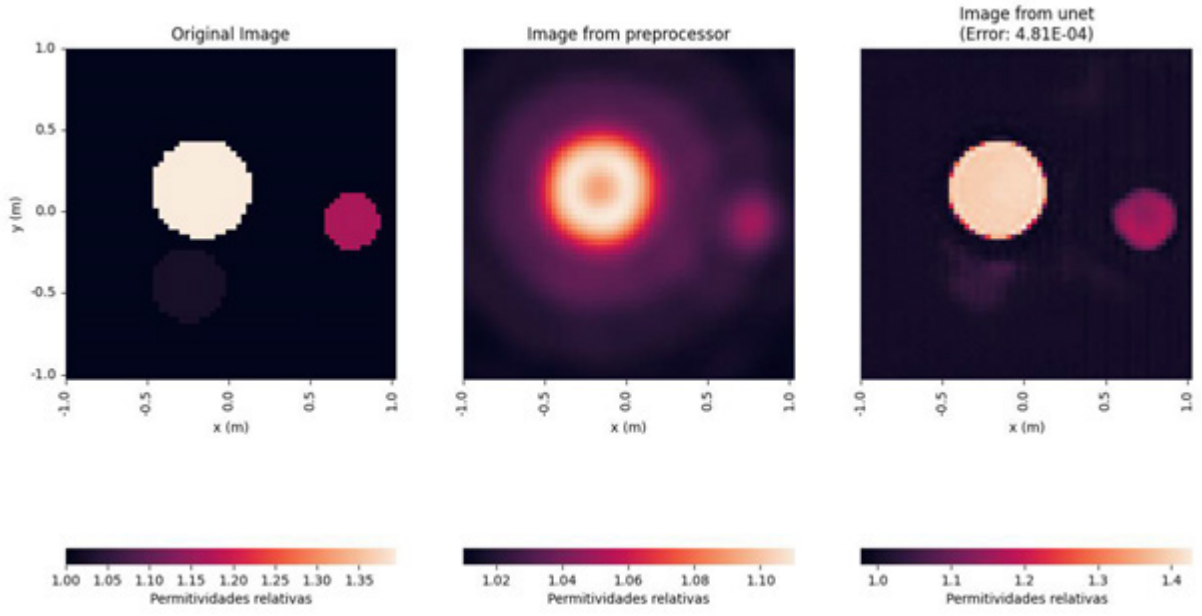

Fig. 106: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 33

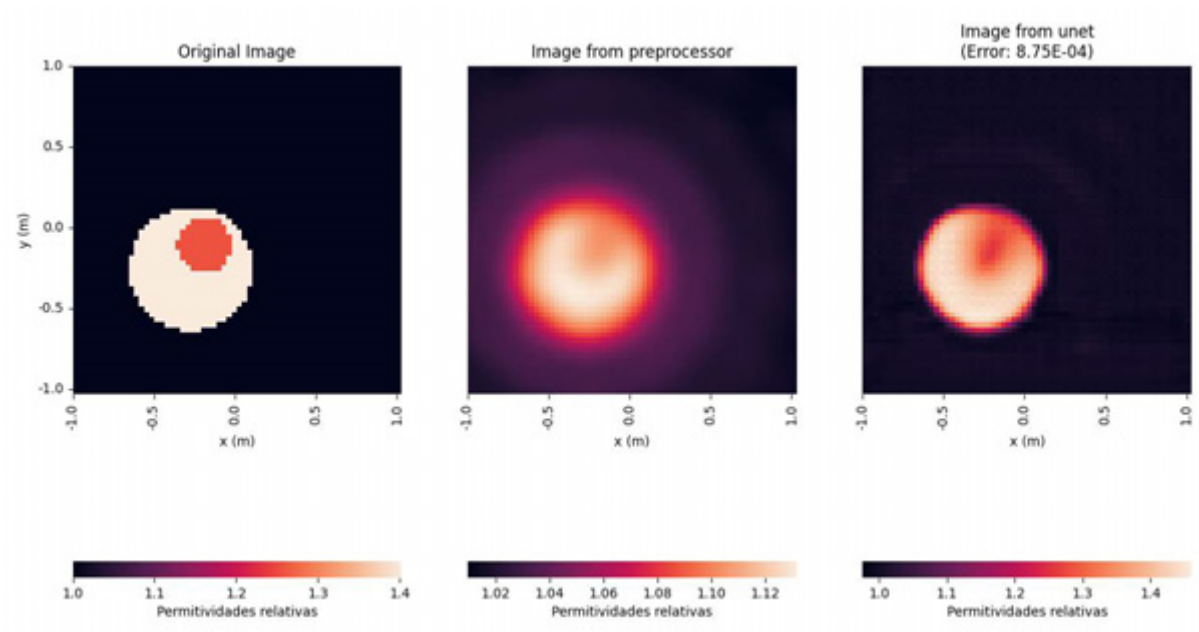

Fig. 107: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 33

Ejecución 34

Parámetro	**Valor**
Imágenes	5.000
Imágenes por batch	16
Iteraciones	100
Set de entrenamiento	3.500 (70 %)
Set de validación	750 (15 %)
Set de testing	750 (15 %)
Rango de permitividades	1 - 1,5

Tabla 65: Parámetros de ejecución 34

Métrica	**Valor**
Duración del entrenamiento	2:19:27
Error total	$1,60\ x\ 10^{-2}$
Error medio	$3,48\ x\ 10^{-4}$
Desvío estándar	$1,10\ x\ 10^{-4}$

Tabla 66: Resultados de ejecución 34

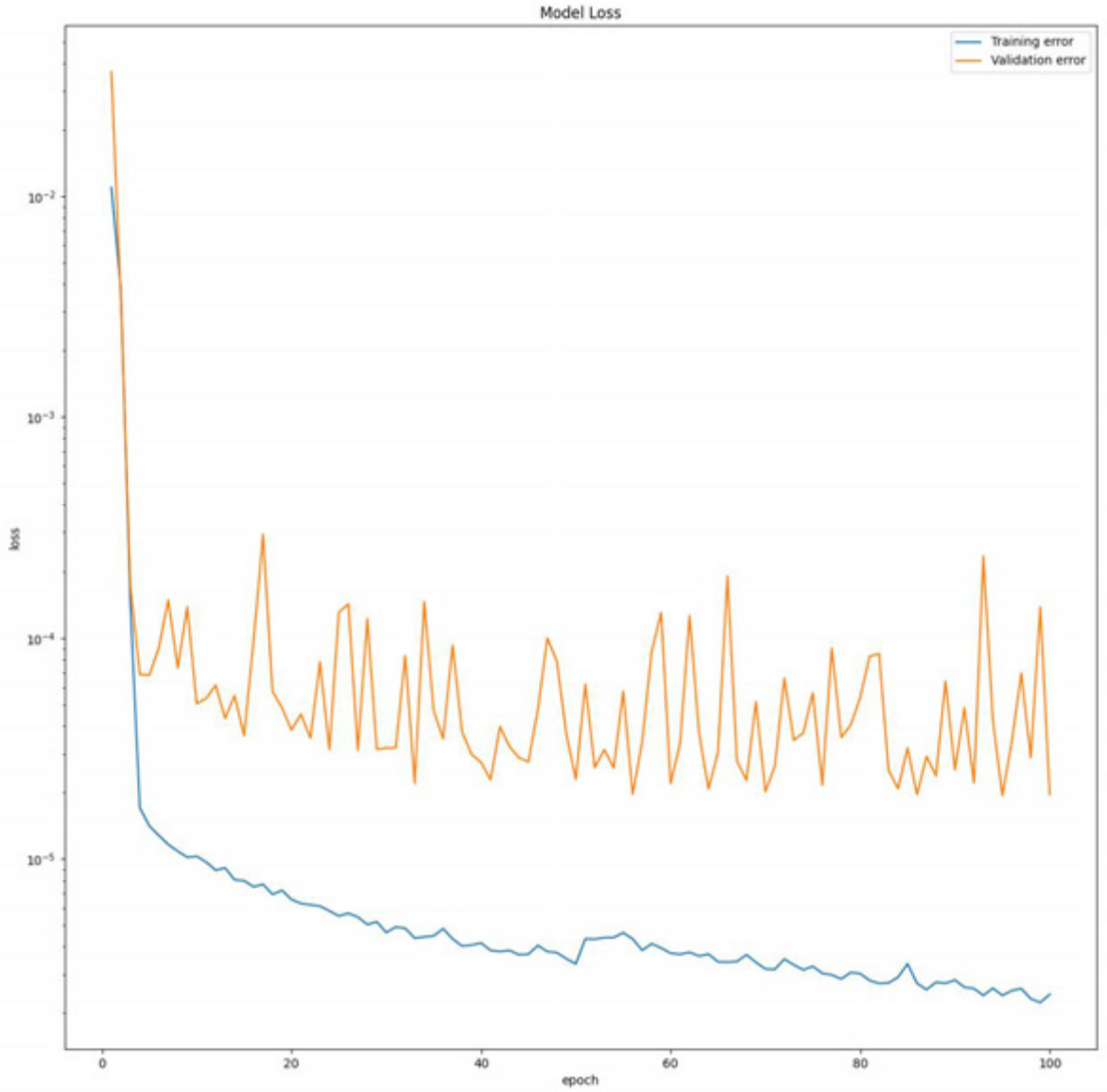

Fig. 108: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 34

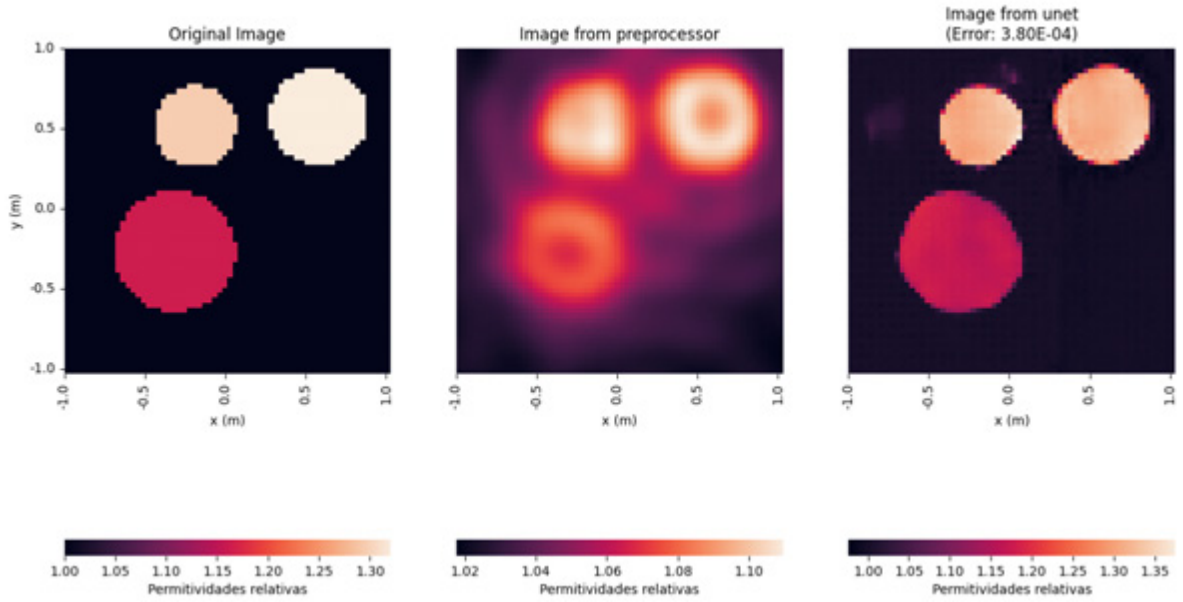

Fig. 109: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 34

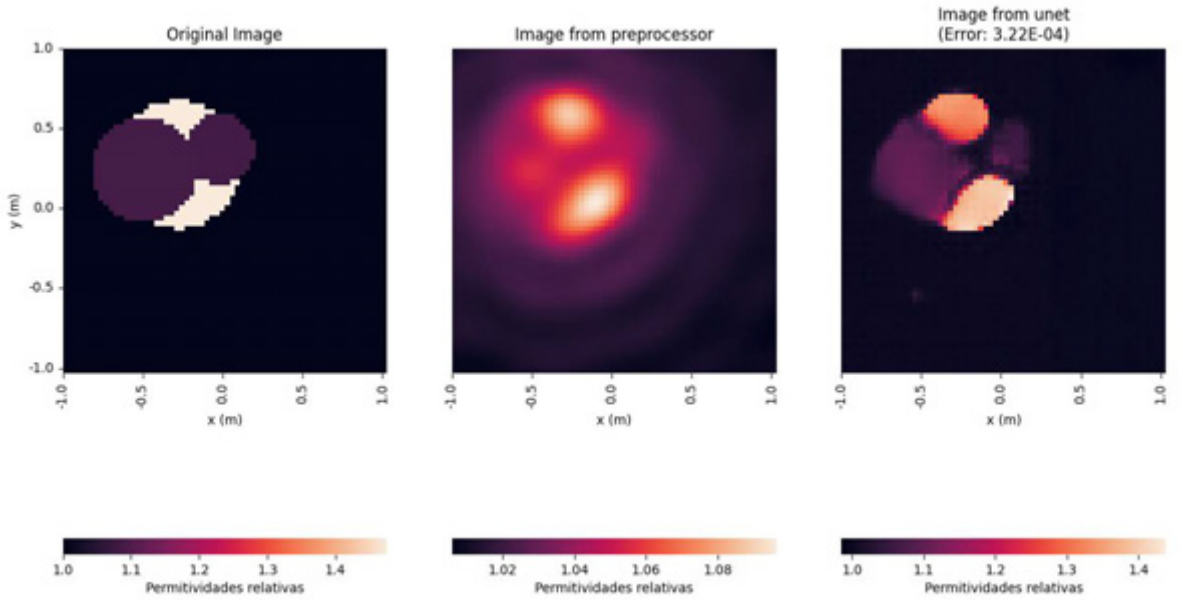

Fig. 110: Resultado del testeo de la red para una imagen de 3 círculos en ejecución 34

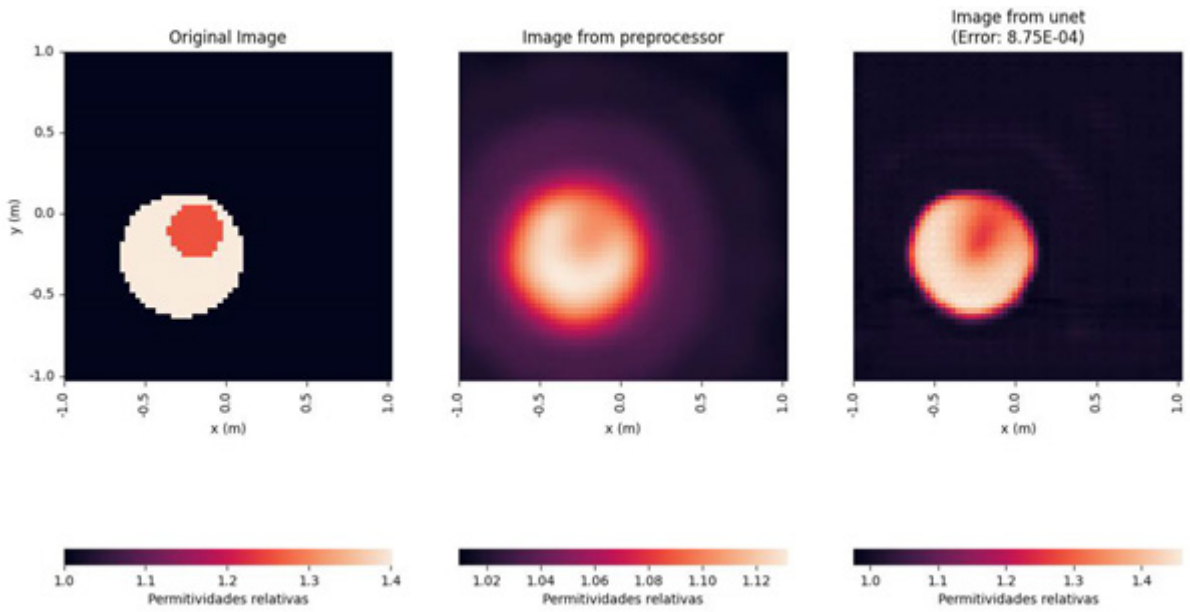

Fig. 111: Resultado del testeo de la red para una imagen de 2 círculos en ejecución 34

Pruebas de validación

Para validar a GenPer frente al software hecho por Wei y Chen en [8], se desarrolló una funcionalidad para leer el archivo obtenido al generar imágenes en el código MATLAB. A partir de este, se obtiene un set de imágenes con círculos equivalente que puede seguir la ejecución normal de GenPer. Al finalizar el testing de la red, se agregó un paso para comparar los resultados obtenidos de MATLAB (a través de un nuevo archivo generado en [8]) y los del código Python. Entre las métricas comparadas están el error medio para entrenamiento, validación y testing y también el desvío estándar de los errores de testing. Además, se generaron algunos gráficos comparativos.

Ejecución 35

Parámetro	Valor
Imágenes	500
Imágenes por batch	1
Iteraciones	203
Set de entrenamiento	450 (90 %)
Set de validación	25 (5 %)
Set de testing	25 (5 %)

Rango de permitividades	1 - 1,5
Algoritmo de optimización	AdamW

Tabla 67: Parámetros de ejecución de la ejecución 35 de validación - establecidos por defecto en [8]

Métrica	**Valor**
Duración del entrenamiento GenPer	2:45:06
Duración del entrenamiento MATLAB	7:52:43
Error medio de entrenamiento GenPer	$5,95\ x\ 10^{-4}$
Error medio de entrenamiento MATLAB	$9,49\ x\ 10^{-2}$
Error medio de validación GenPer	$1,48\ x\ 10^{-3}$
Error medio de validación MATLAB	$9,90\ x\ 10^{-2}$
Error medio de testing GenPer	$9,35\ x\ 10^{-4}$
Error medio de testing MATLAB	$2,79\ x\ 10^{-2}$
Desvío estándar GenPer	$1,34\ x\ 10^{-3}$
Desvío estándar MATLAB	$1,05\ x\ 10^{-2}$

Tabla 68: Resultados de ejecución 35 de validación

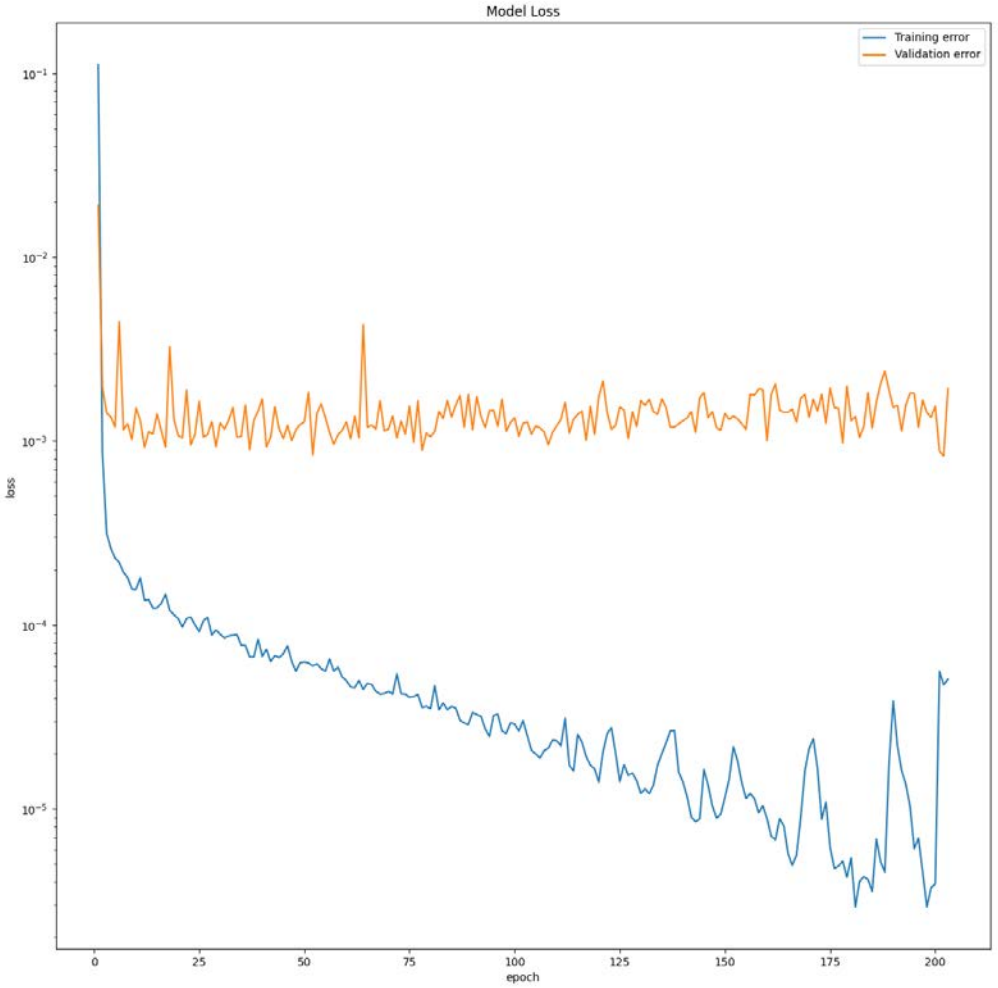

Fig. 112: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 35 de validación para GenPer

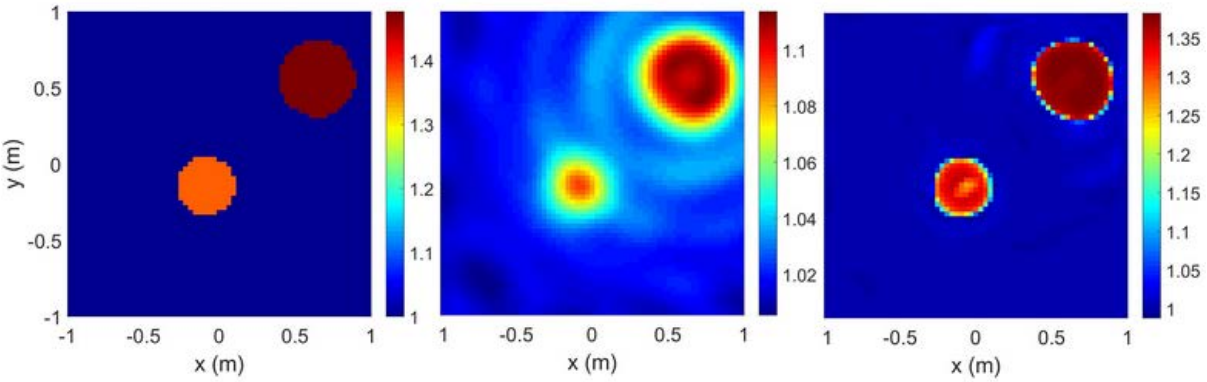

Fig. 113: Resultado del testeo de la red en MATLAB para una imagen de 2 círculos en ejecución 35 de validación

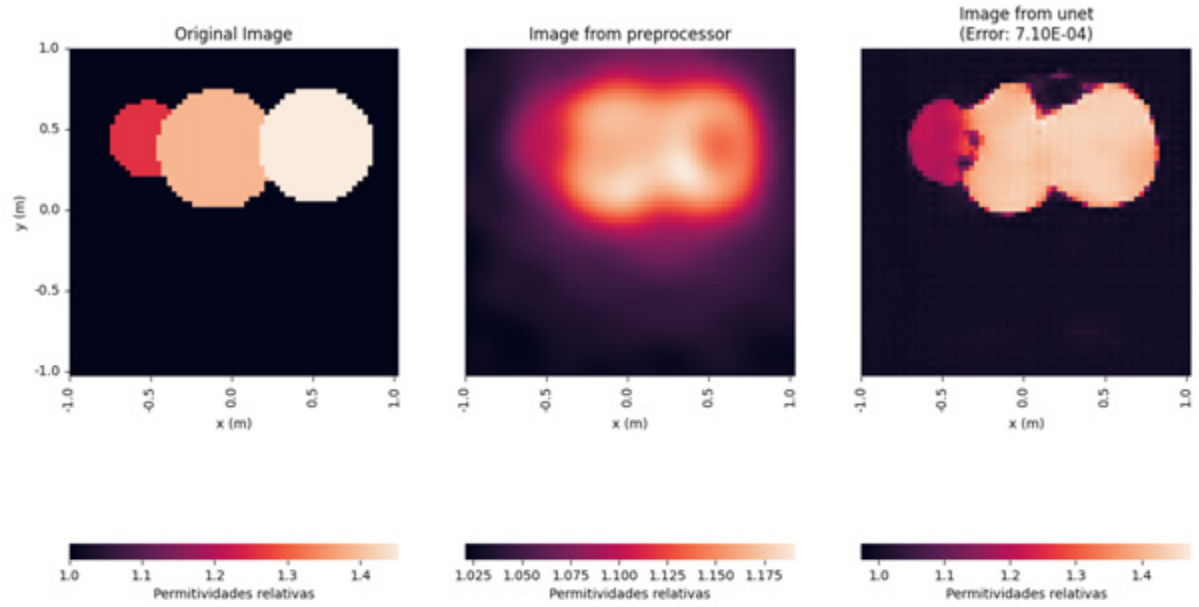

Fig. 114: Resultado del testeo de la red en Python para una imagen de 3 círculos en ejecución 35 de validación

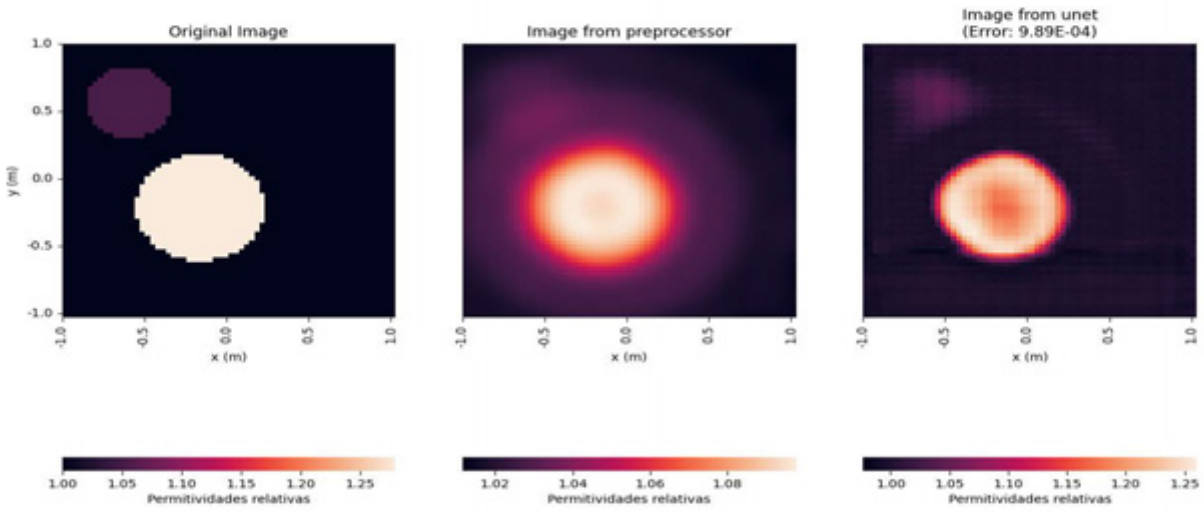

Fig. 115: Resultado del testeo de la red en Python para una imagen de 2 círculos en ejecución 35 de validación

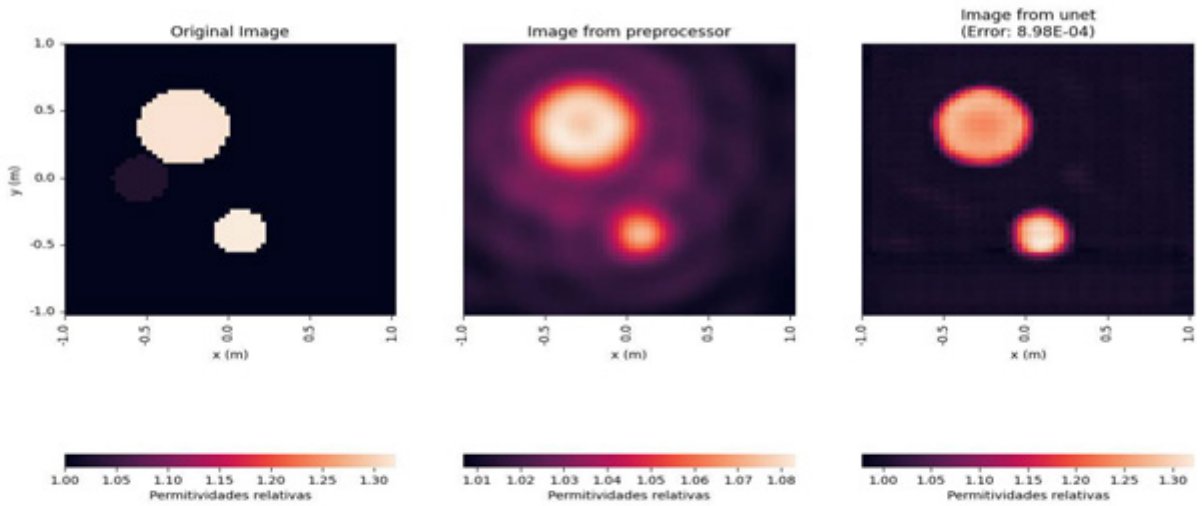

Fig. 116: Resultado del testeo de la red en Python para una imagen de 2 círculos en ejecución 35 de validación

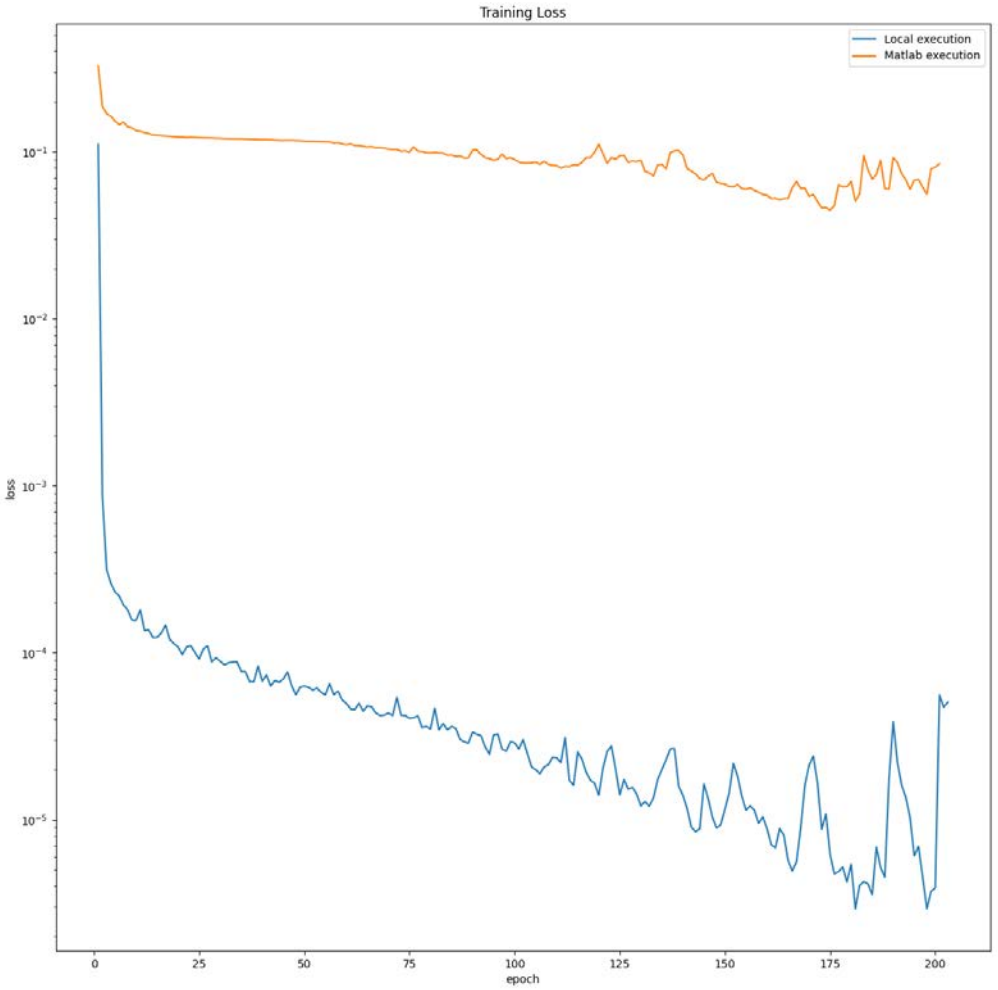

Fig. 117: Gráfico de iteración vs error de entrenamiento (escala logarítmica en eje vertical) en ejecución 35 de validación para GenPer (azul) y MATLAB (naranja)

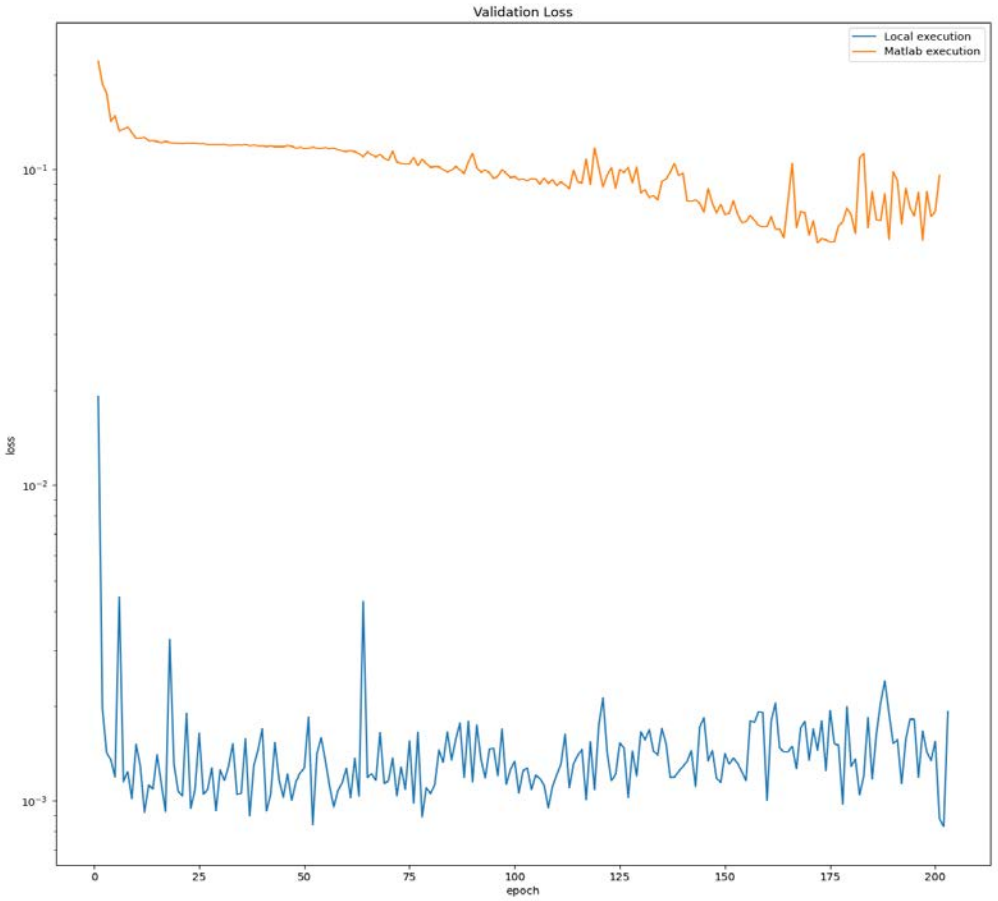

Fig. 118: Gráfico de iteración vs error de validación (escala logarítmica en eje vertical) en ejecución 35 de validación para GenPer (azul) y MATLAB (naranja)

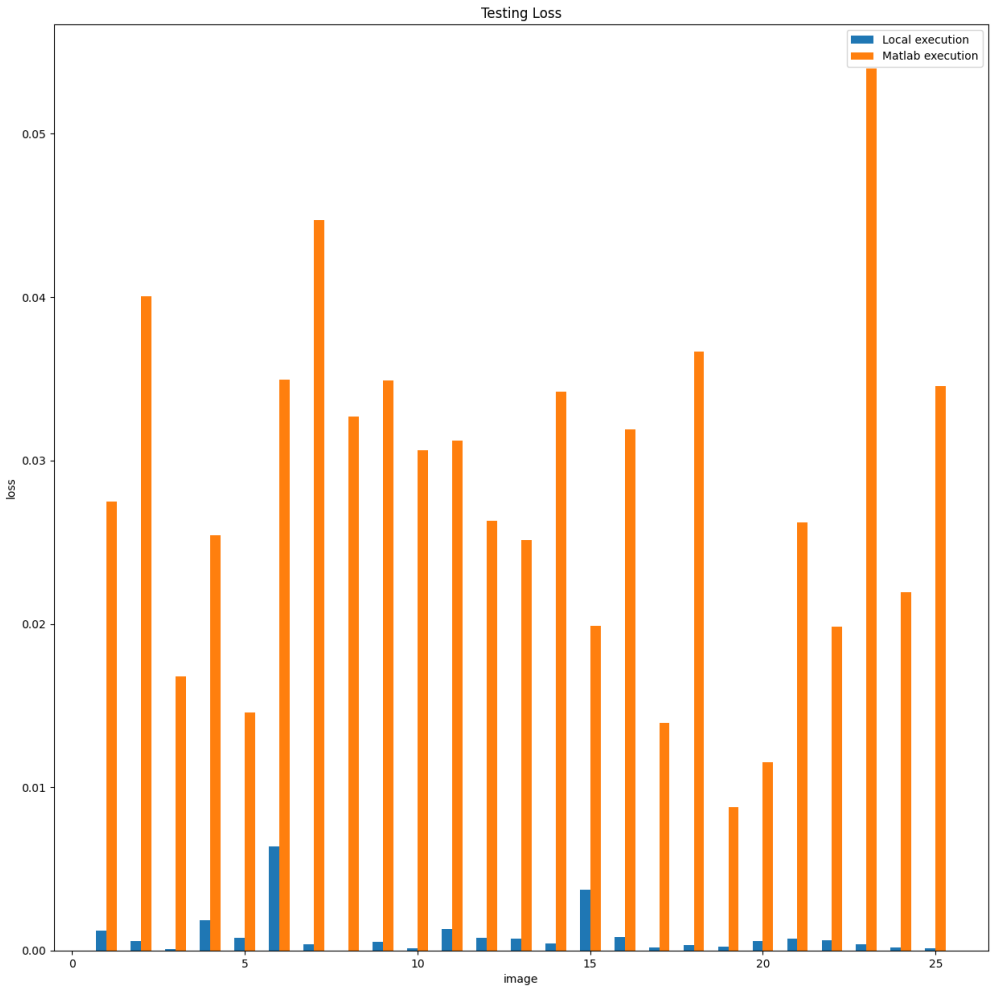

Fig. 119: Gráfico de número de imagen vs error de testing en ejecución 35 de validación para GenPer (azul) y MATLAB (naranja)

Debido a que GenPer emplea un algoritmo de tipo AdamW para optimizar, mientras que el código MATLAB utiliza SGD (Stochastic Gradient Descent) con coeficiente fijo, se optó por hacer una nueva ejecución de validación con este algoritmo en Python y así comparar nuevamente los errores.

Ejecución 36

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	203
Set de entrenamiento	450 (90 %)
Set de validación	25 (5 %)

Set de testing	25 (5 %)
Rango de permitividades	1 - 1,5
Algoritmo de optimización	SGD

Tabla 69: Parámetros de ejecución de la ejecución 36 de validación - establecidos por defecto en [8]

Métrica	**Valor**
Duración del entrenamiento GenPer	2:36:28
Duración del entrenamiento MATLAB	7:52:43
Error medio de entrenamiento GenPer	$9,76\ x\ 10^{-2}$
Error medio de entrenamiento MATLAB	$9,49\ x\ 10^{-2}$
Error medio de validación GenPer	$4,89\ x\ 10^{-1}$
Error medio de validación MATLAB	$9,90\ x\ 10^{-2}$
Error medio de testing GenPer	$1,16\ x\ 10^{-2}$
Error medio de testing MATLAB	$2,79\ x\ 10^{-2}$
Desvío estándar GenPer	$1,81\ x\ 10^{-3}$
Desvío estándar MATLAB	$1,05\ x\ 10^{-2}$

Tabla 70: Resultados de ejecución 36 de validación

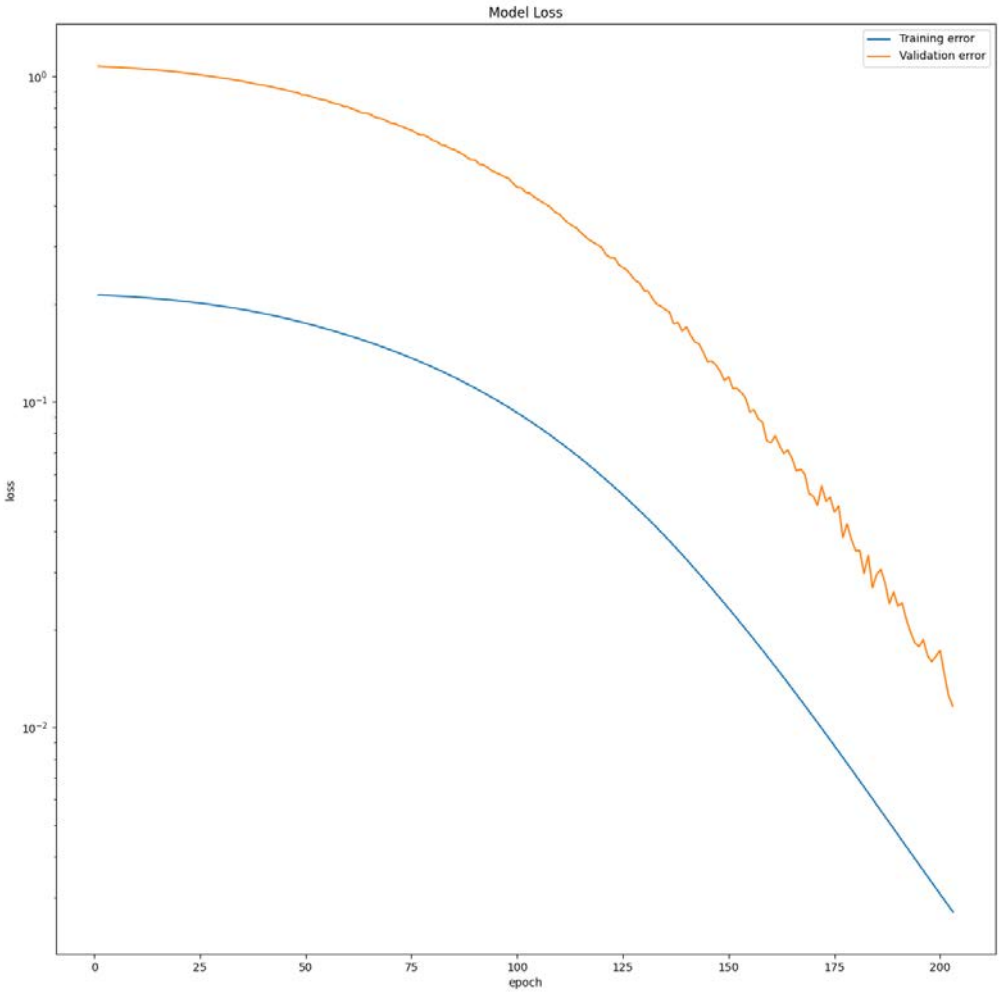

Fig. 120: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 36 de validación para GenPer

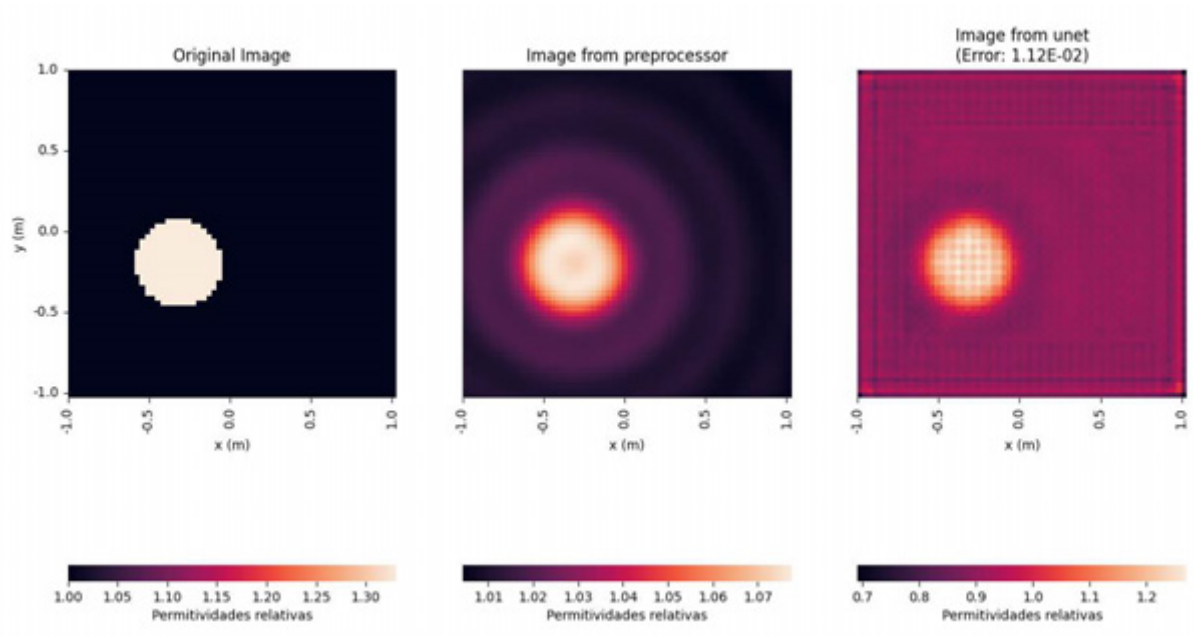

Fig. 121: Resultado del testeo de la red en Python para una imagen de 1 círculo en ejecución 36 de validación

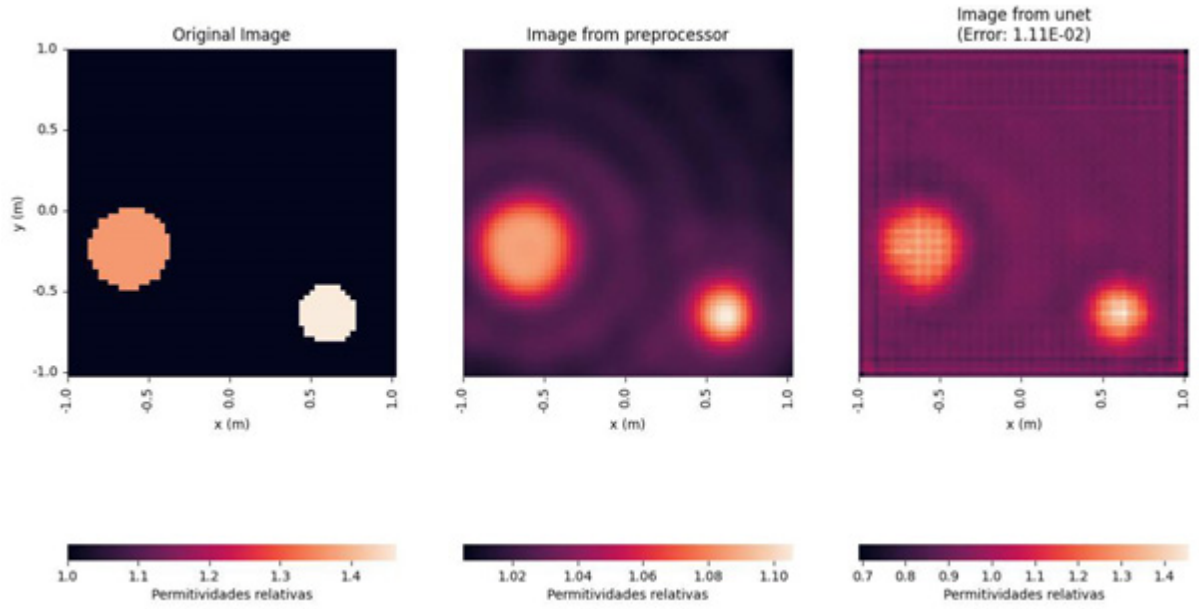

Fig. 122: Resultado del testeo de la red en Python para una imagen de 2 círculos en ejecución 36 de validación

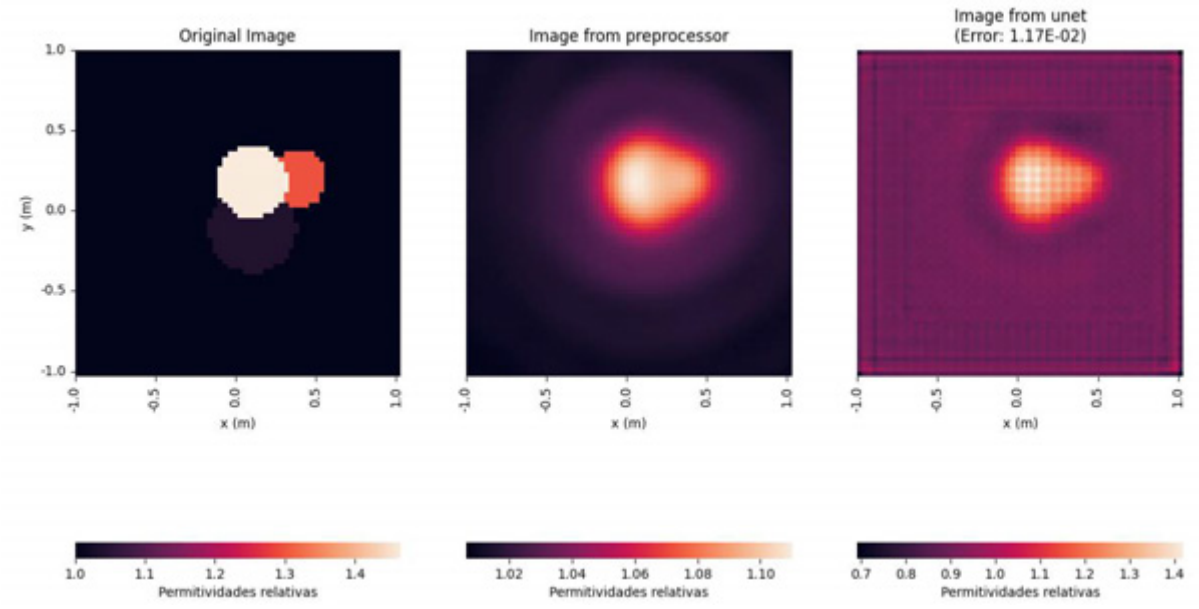

Fig. 123: Resultado del testeo de la red en Python para una imagen de 3 círculos en ejecución 36 de validación

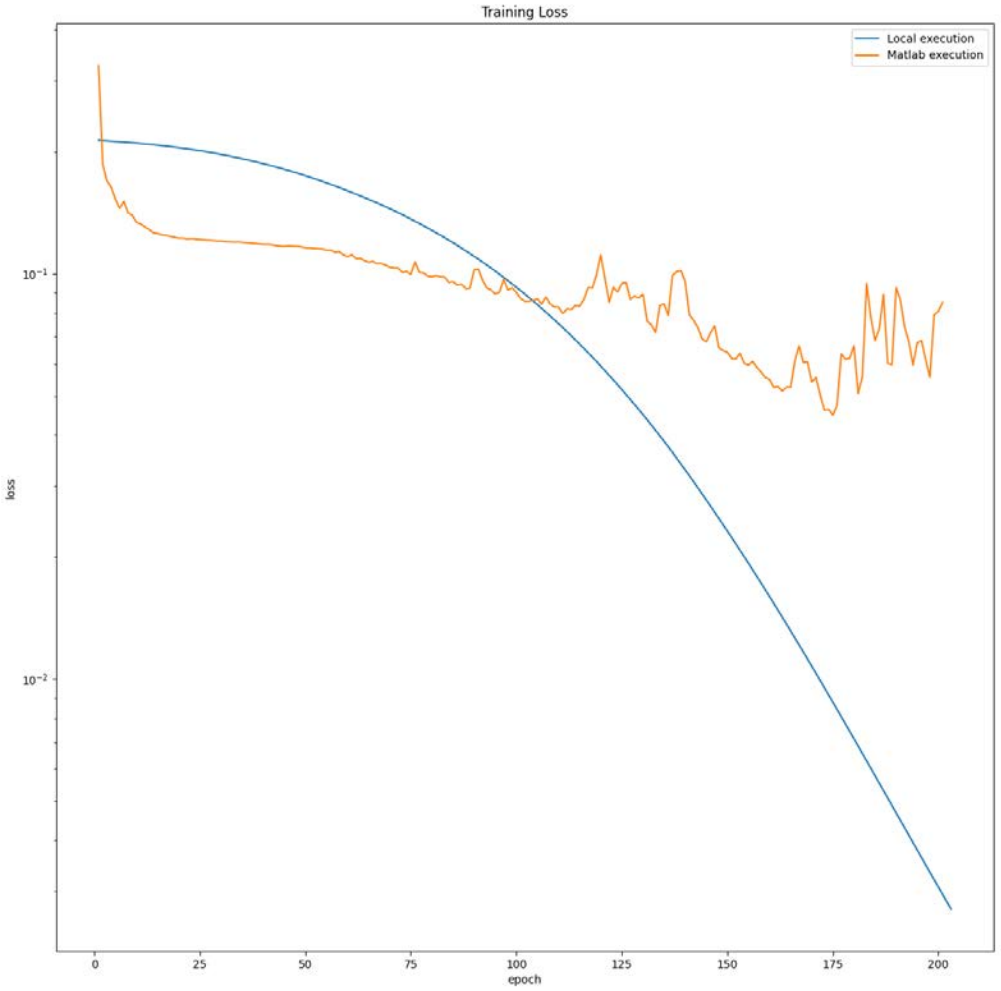

Fig. 124: Gráfico de iteración vs error de entrenamiento (escala logarítmica en eje vertical) en ejecución 36 de validación para GenPer (azul) y MATLAB (naranja)

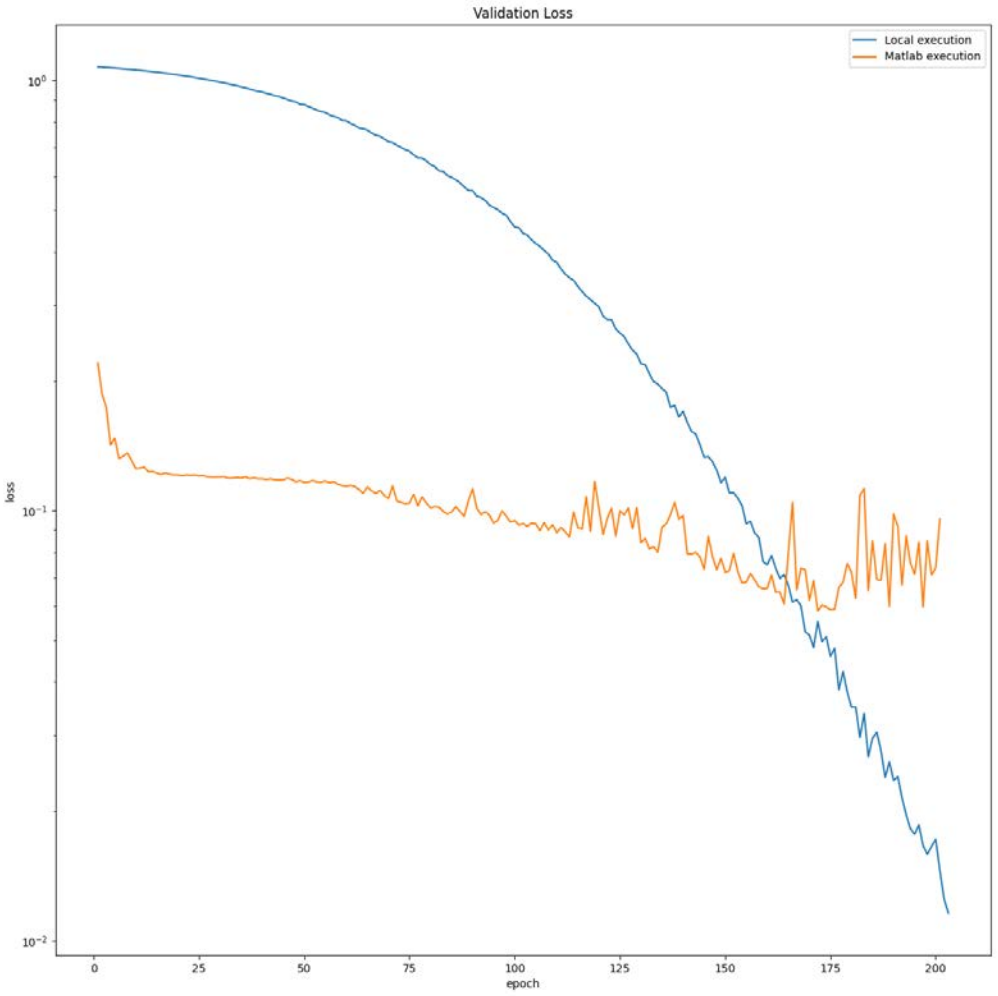

Fig. 125: Gráfico de iteración vs error de validación (escala logarítmica en eje vertical) en ejecución 36 de validación para GenPer (azul) y MATLAB (naranja)

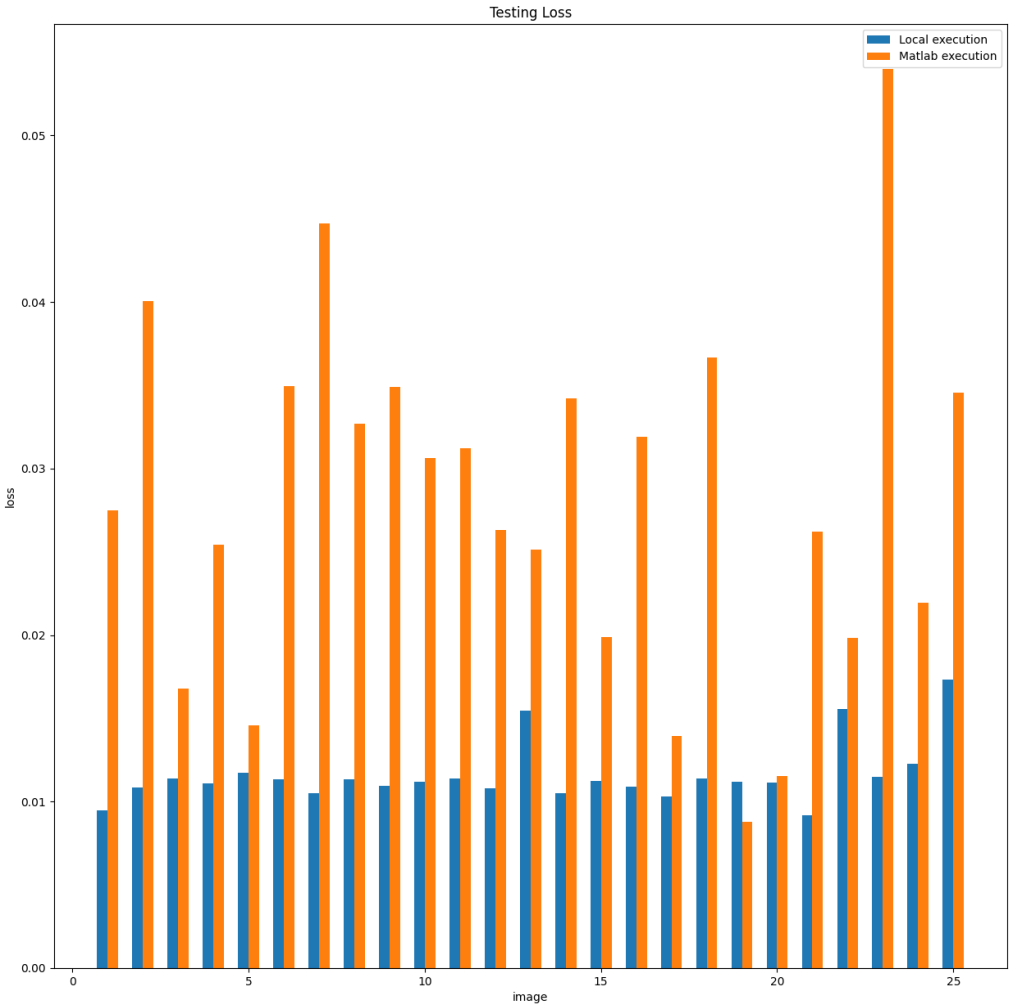

Fig. 126: Gráfico de número de imagen vs error de testing en ejecución 36 de validación para GenPer (azul) y MATLAB (naranja)

Experimentos adicionales

Los experimentos adicionales consistieron en realizar la misma ejecución pero utilizando otras formas en lugar de círculos.

Ejecución con rectángulos

El primero de los experimentos adicionales consistió en utilizar rectángulos en lugar de círculos en las imágenes, los cuales pueden ser múltiples y superpuestos, como se observa en la siguiente representación:

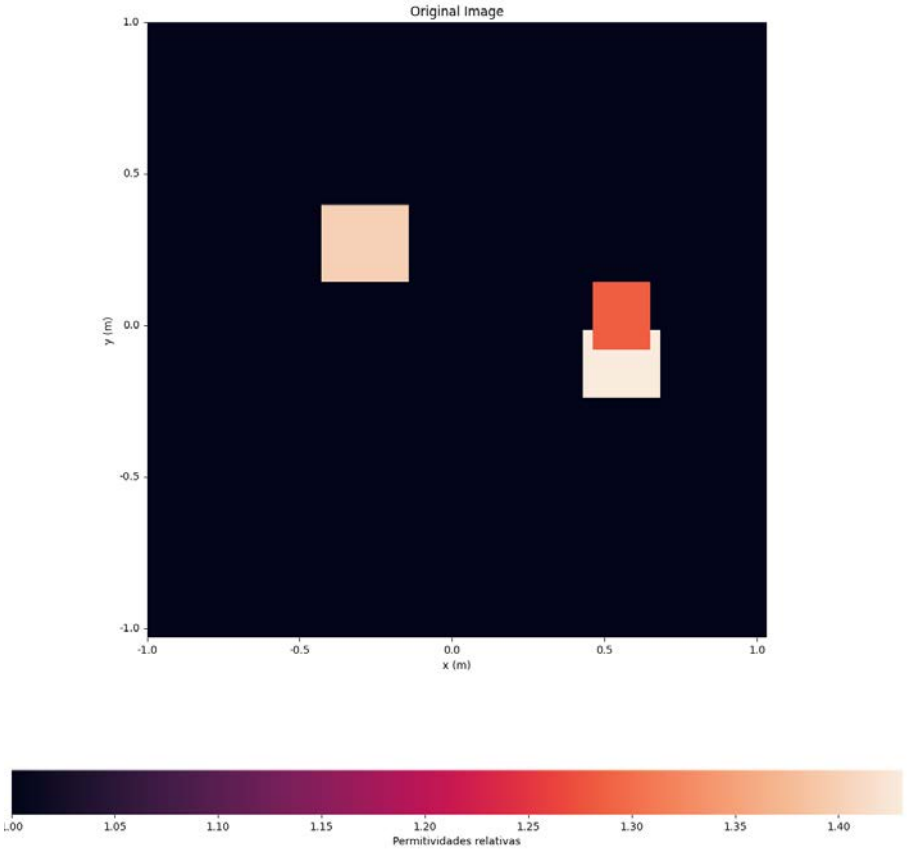

Fig. 127: Imágen generada por GenPer con 3 rectángulos de distintos tamaños, 2 de ellos, superpuestos.

Ejecución 37

Parámetro	**Valor**
Imágenes	500
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	350 (70 %)
Set de validación	75 (15 %)
Set de testing	75 (15 %)
Rango de permitividades	1 - 1,5

Tabla 71: Parámetros de ejecución 37

Métrica	**Valor**

Duración del entrenamiento	0:39:39
Error total	$2,42\ x\ 10^{-2}$
Error medio	$3,23\ x\ 10^{-4}$
Desvío estándar	$4,37\ x\ 10^{-4}$

Tabla 72: Resultados de ejecución 37

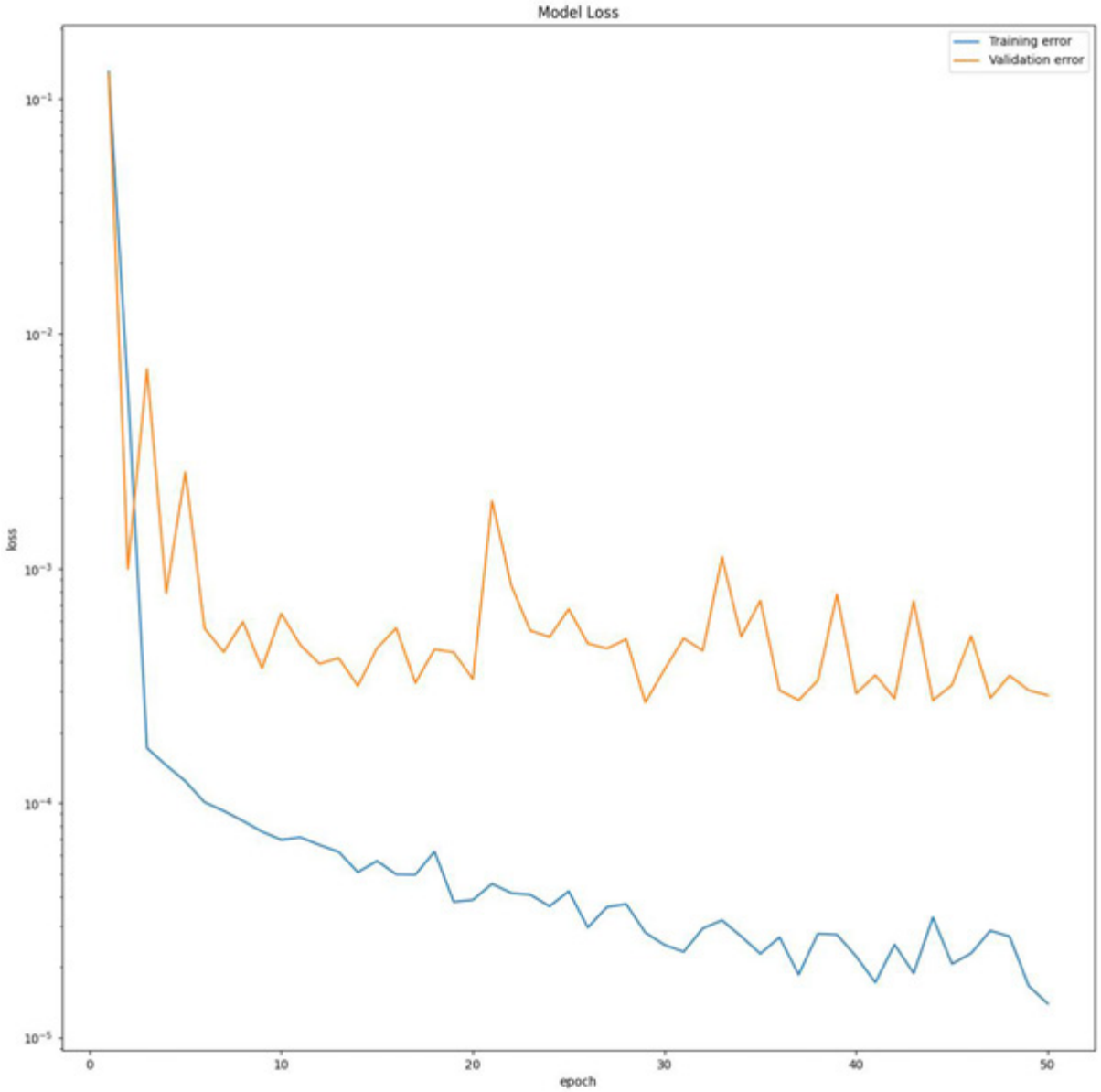

Fig. 128: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 37

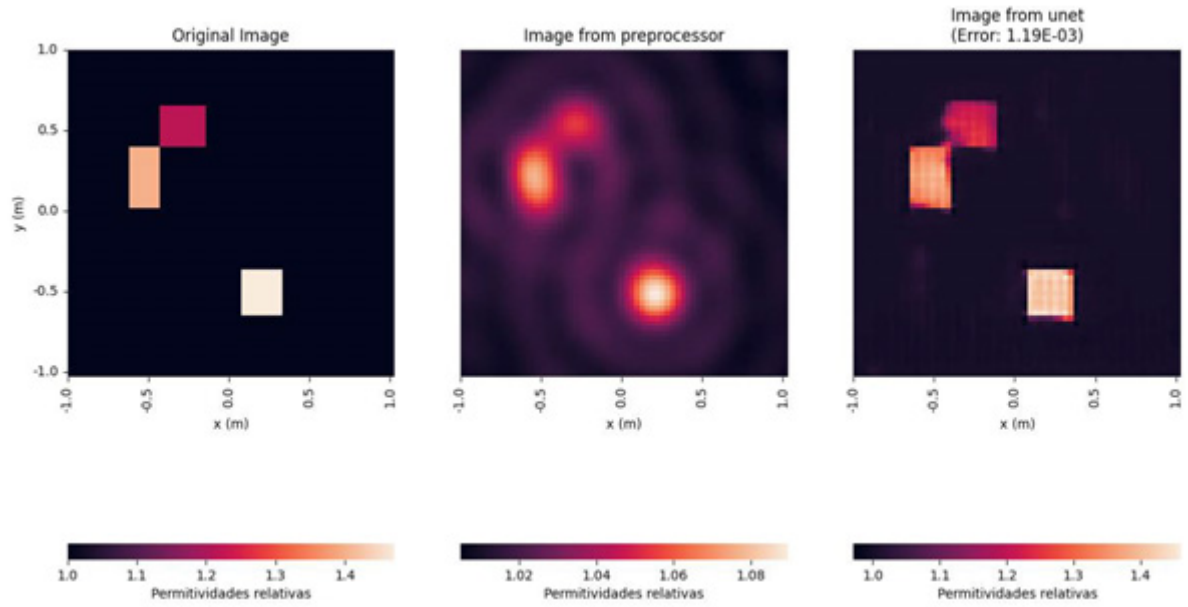

Fig. 129: Resultado del testeo de la red en Python para una imagen de 3 rectángulos en ejecución 37

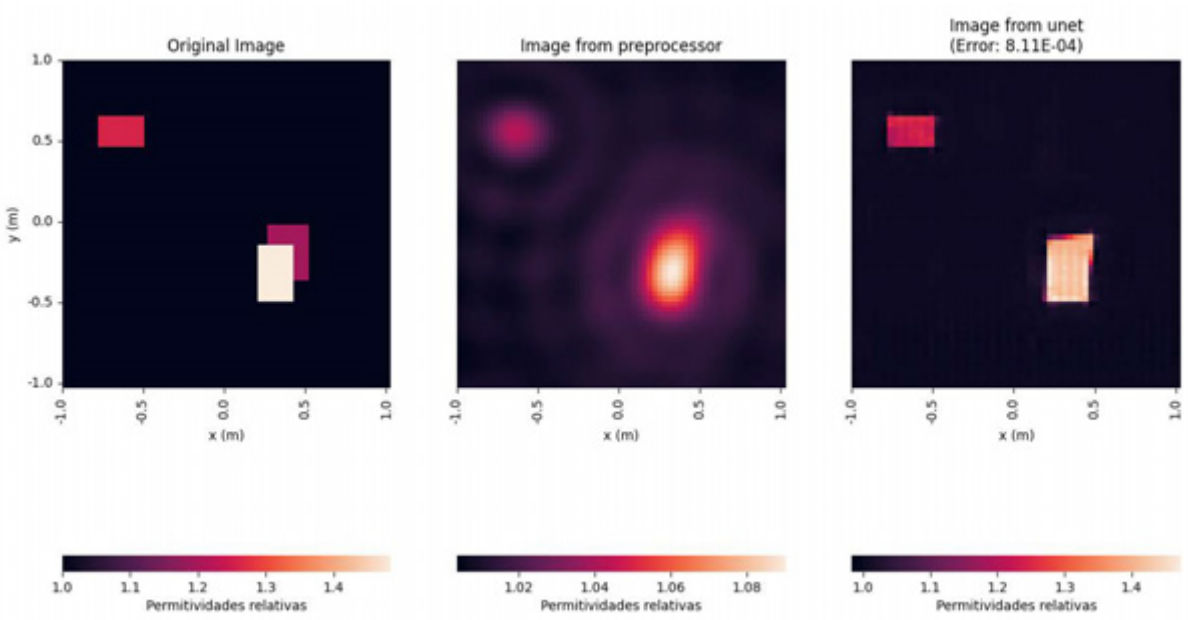

Fig. 130: Resultado del testeo de la red en Python para una imagen de 3 rectángulos en ejecución 37

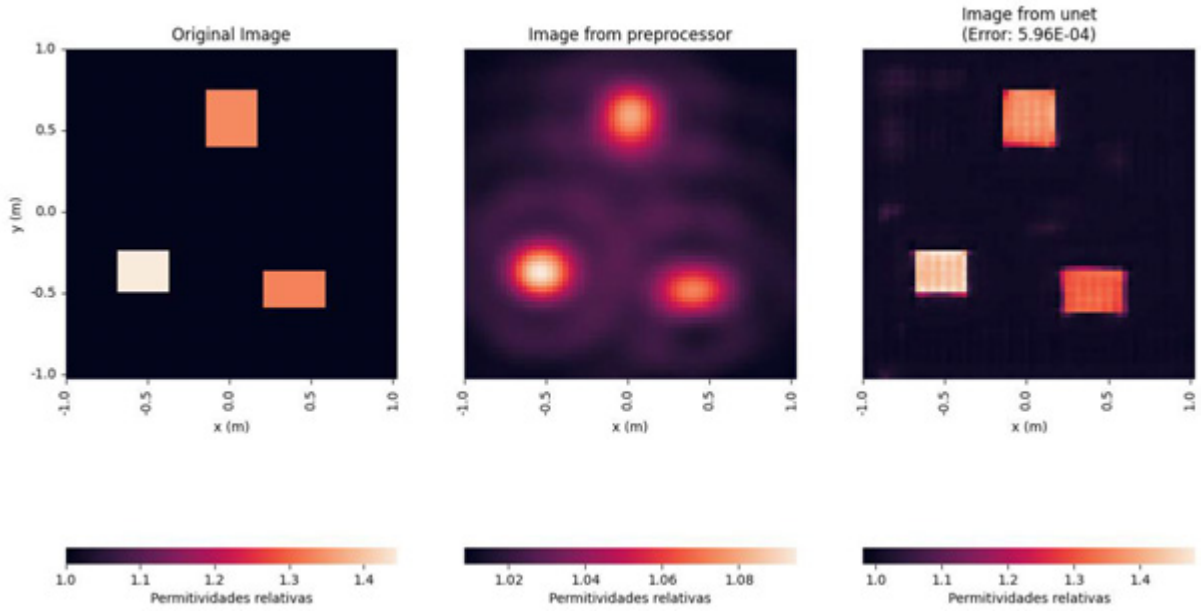

Fig. 131: Resultado del testeo de la red en Python para una imagen de 3 rectángulos en ejecución 37

Ejecución con MNIST

El otro experimento adicional consistió en adaptar el set de datos MNIST de dígitos escritos a mano de [11] para generar imágenes en las cuales los dígitos sean las formas de las mismas. A su vez, con un 50 % de probabilidad, se superpusieron 2 dígitos de manera aleatoria, como se observa en la siguiente representación:

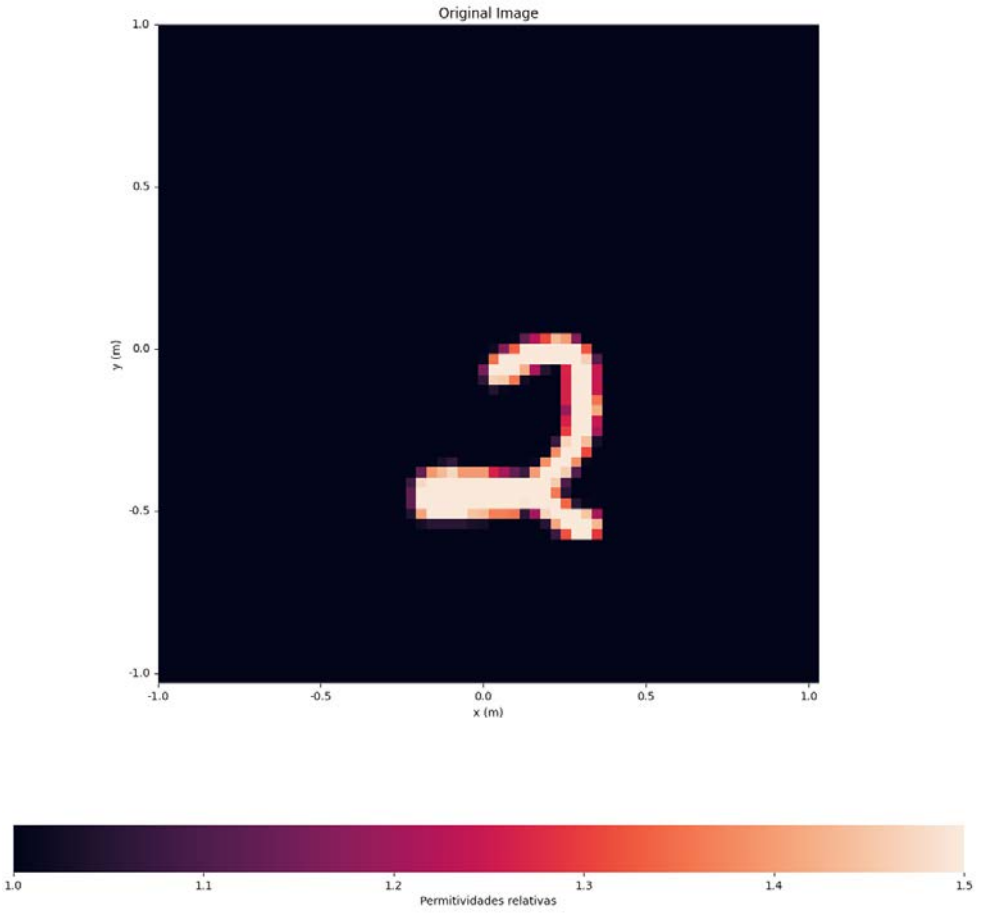

Fig. 132: Imagen sin procesar de un dígito del set de datos MNIST adaptada a GenPer.

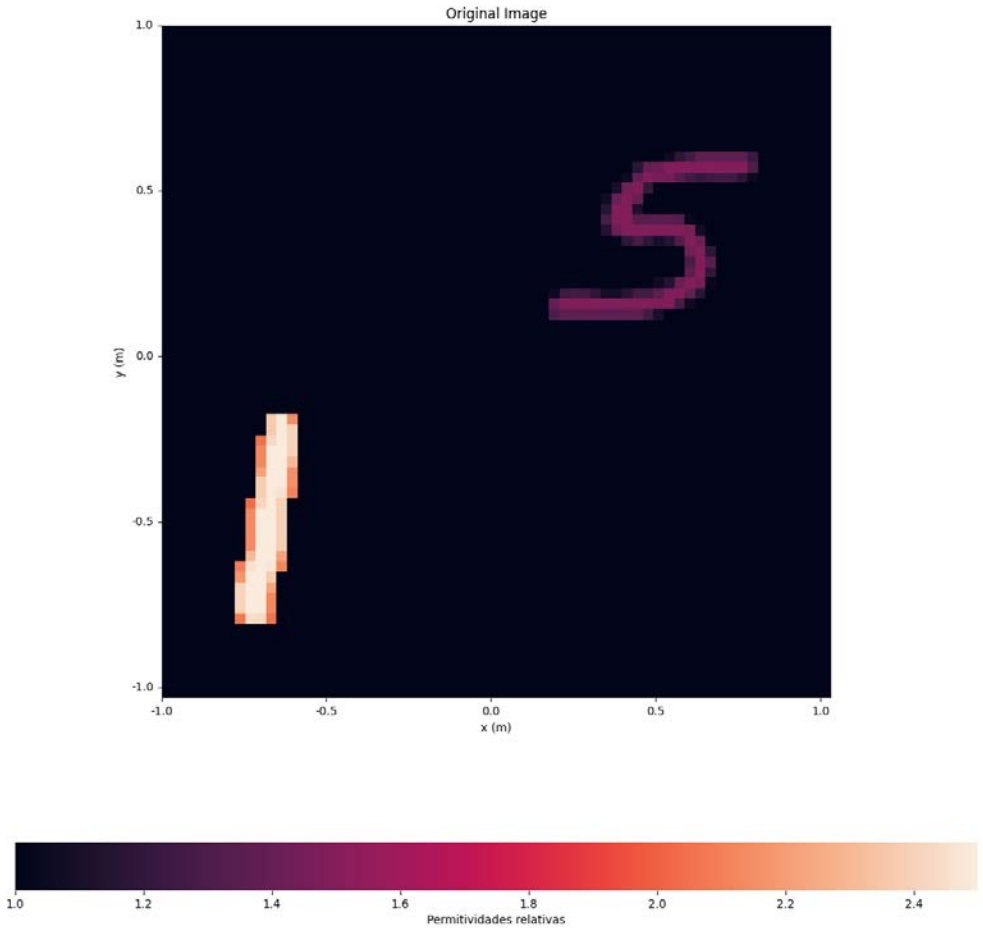

Fig. 133: Imagen sin procesar de dos dígitos superpuestos del set de datos MNIST adaptada a GenPer.

Dado que se trata de un set de datos bastante extenso (son 70.000 imágenes), se optó por utilizar, para el entrenamiento, 50 iteraciones y un tamaño de batch de 50 imágenes. En cuanto a las proporciones del set, se lo utilizó como viene, 60.000 para entrenamiento y validación (para este último se usó el 5 %) y 10.000 para testing.

Ejecución 38

Parámetro	**Valor**
Imágenes	70.000
Imágenes por batch	1
Iteraciones	50
Set de entrenamiento	57.000 (81 %)
Set de validación	6.700 (5 %)
Set de testing	10.000 (15 %)

Rango de permitividades	1 - 1,5

Tabla 73: Parámetros de ejecución 38

Métrica	**Valor**
Duración del entrenamiento	58:56:37
Error total	$4,88\ x\ 10^{-1}$
Error medio	$4,88\ x\ 10^{-3}$
Desvío estándar	$5,32\ x\ 10^{-3}$

Tabla 74: Resultados de ejecución 38

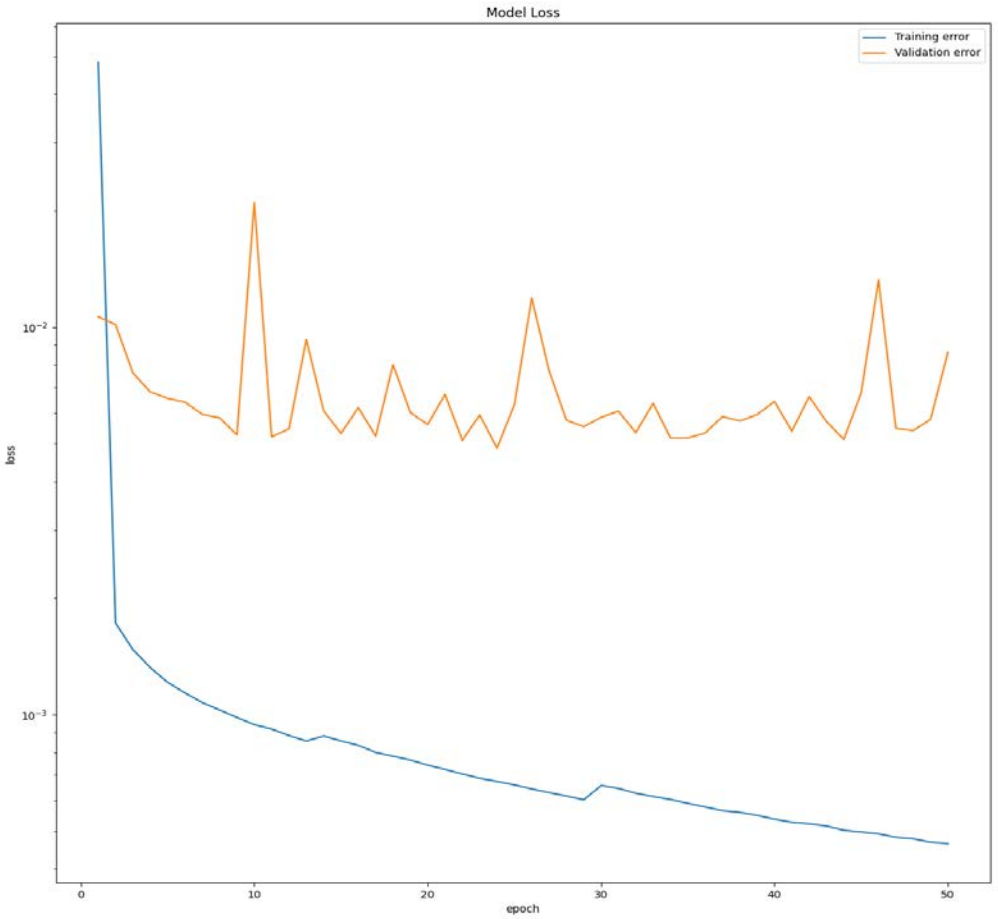

Fig. 134: Gráfico de iteración vs error para entrenamiento y validación (escala logarítmica en eje vertical) en ejecución 38

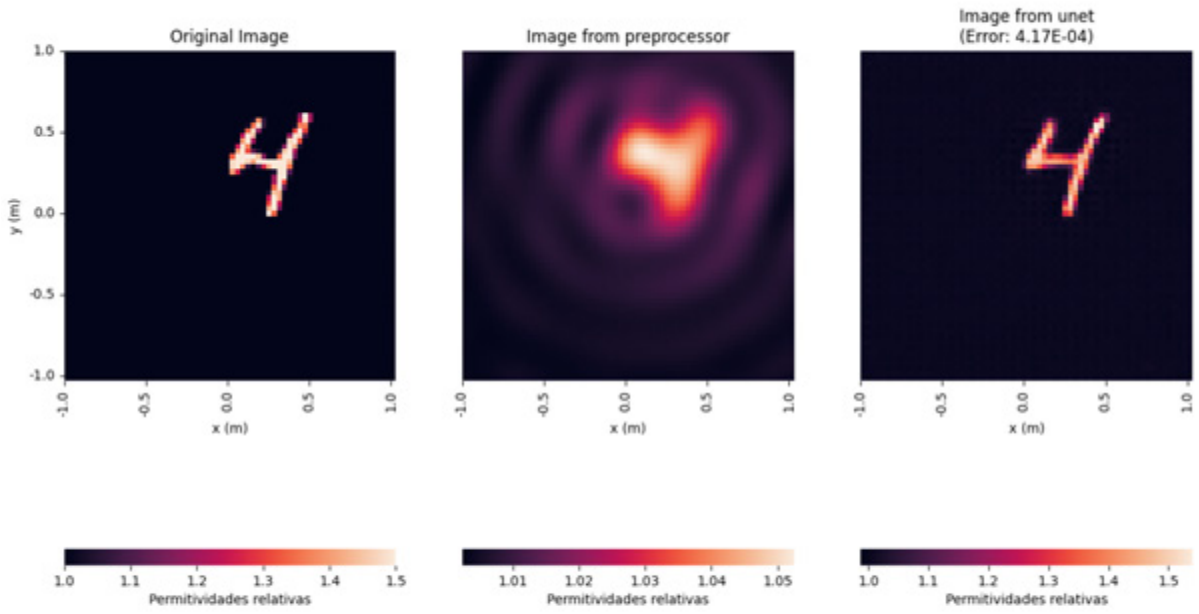

Fig. 135: Resultado del testeo de la red para una imagen de un dígito (cuatro) en ejecución 38

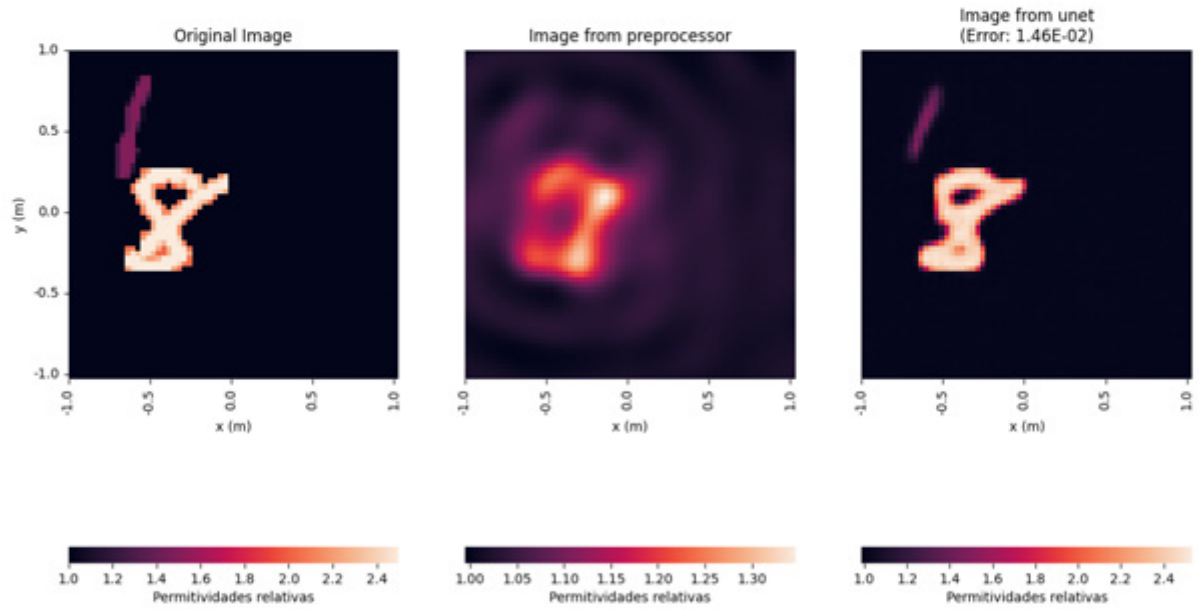

Fig. 136: Resultado del testeo de la red para una imagen de dos dígitos (uno y ocho) en ejecución 38

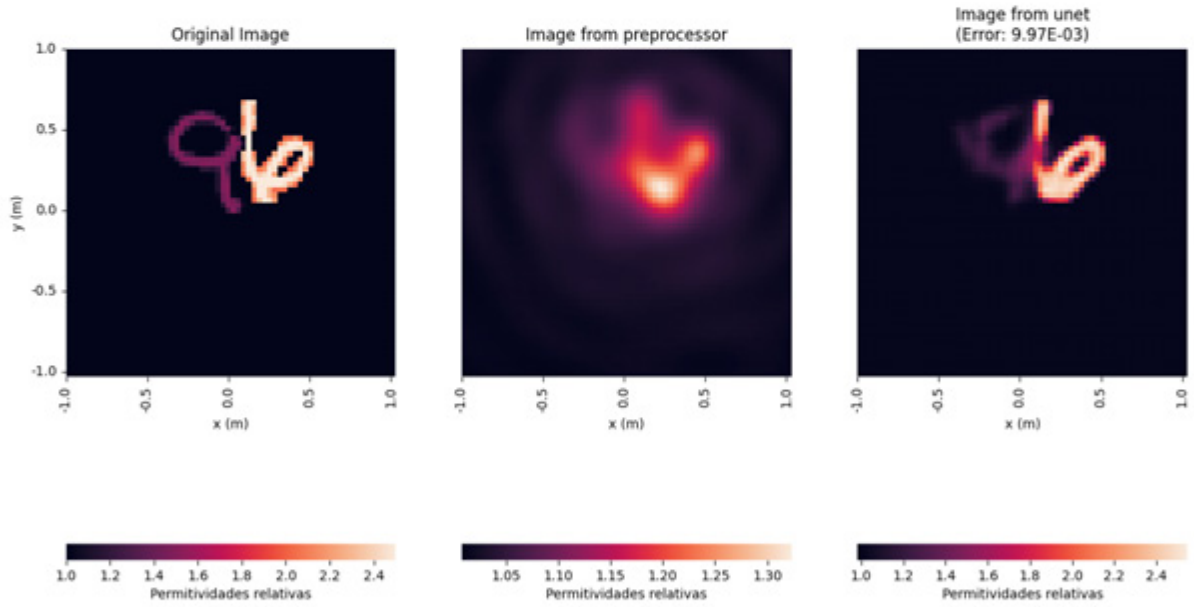

Fig. 137: Resultado del testeo de la red para una imagen de dos dígitos (nueve y seis) en ejecución 38

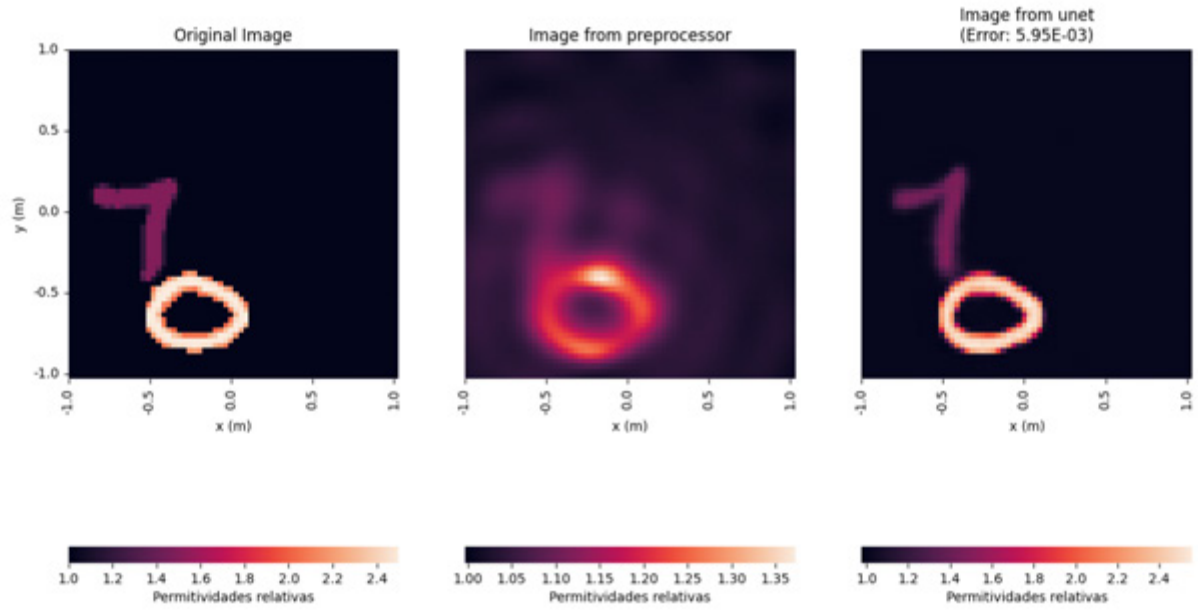

Fig. 138: Resultado del testeo de la red para una imagen de dos dígitos (siete y cero) en ejecución 38

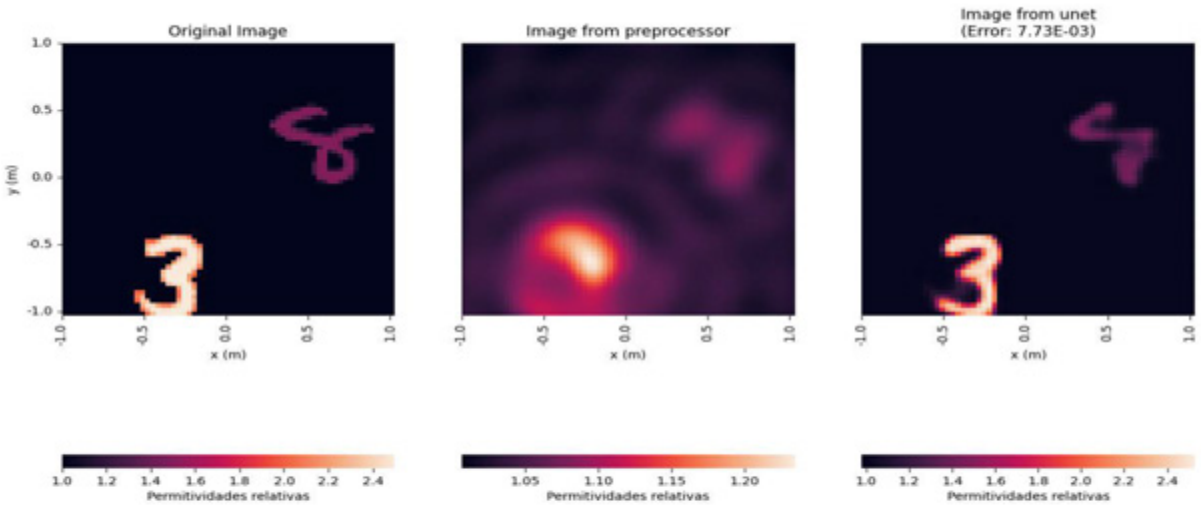

Fig. 139: Resultado del testeo de la red para una imagen de dos dígitos (tres y ocho) en ejecución 38

Gestión del proyecto

En cuanto a la gestión, el proyecto se realizó teniendo como referencia la metodología Scrum [12], siendo el desarrollo iterativo e incremental en trece sprints de dos semanas. Cada uno de ellos incluyó tareas de análisis, desarrollo, pruebas, documentación y gestión. Al ser un equipo de un solo desarrollador, se realizaron reuniones según fuera necesario (aproximadamente cada dos sprints) para demostrar progreso, planificar los próximos pasos, definir prioridades y responder consultas. Las tareas se organizaron en un tablero Kanban [13] que resulta más práctico para una sola persona.

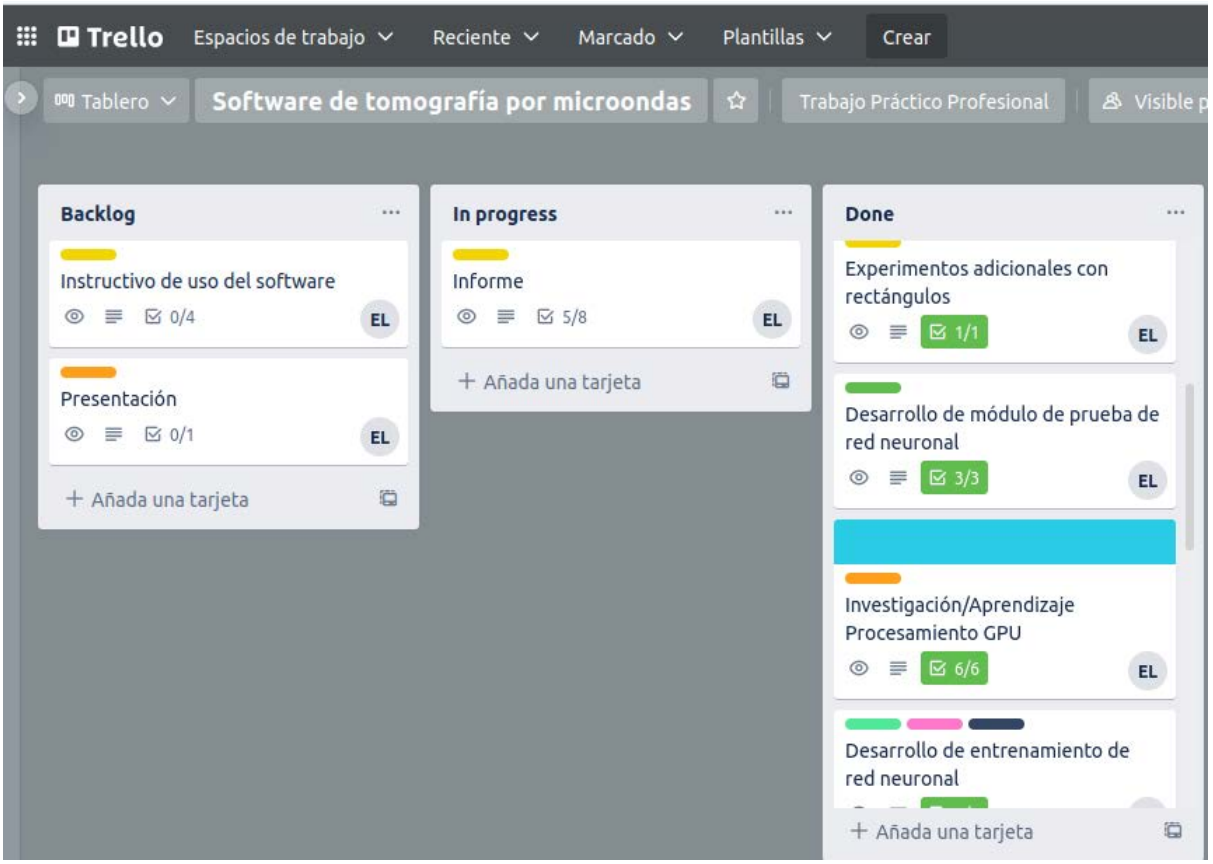

Fig. 140: Tablero Kanban de GenPer

Como resultado de esta gestión, se obtuvo los siguientes resultados con respecto a las estimaciones originales:

Iteración	Tarea	Esfuerzo estimado (hh)	Esfuerzo real (hh)
1	Investigación/Aprendizaje Matlab + PyTorch	40	50
2	Investigación/Aprendizaje PyTorch + Procesamiento GPU	40	30
3	Desarrollo de Generador de imágenes	40	50
4	Desarrollo de Generador de imágenes + pruebas	40	50
5	Desarrollo de Esquema de Backpropagation	40	40
6	Desarrollo de Esquema de Backpropagation + pruebas	40	60
7	Desarrollo de Entrenamiento y Red Neuronal	40	30
8	Desarrollo de Entrenamiento y Red Neuronal	40	30
9	Desarrollo de Entrenamiento y Red Neuronal + pruebas	40	40

10	Desarrollo de módulo de Prueba + Desarrollo GPU	40	40
11	Desarrollo de módulo de Prueba + Pre-procesador MNIST	40	40
12	Documentación + Pruebas de validación y experimentos adicionales	40	40
13	Presentación + Documentación + Informe	10	40
Total		**490**	**540**

Tabla 75: Esfuerzo estimado y real (en horas hombre) en función de la iteración y tareas. En rojo, tareas que no pudieron cumplirse en la iteración y, en verde, tareas que se agregaron a la misma.

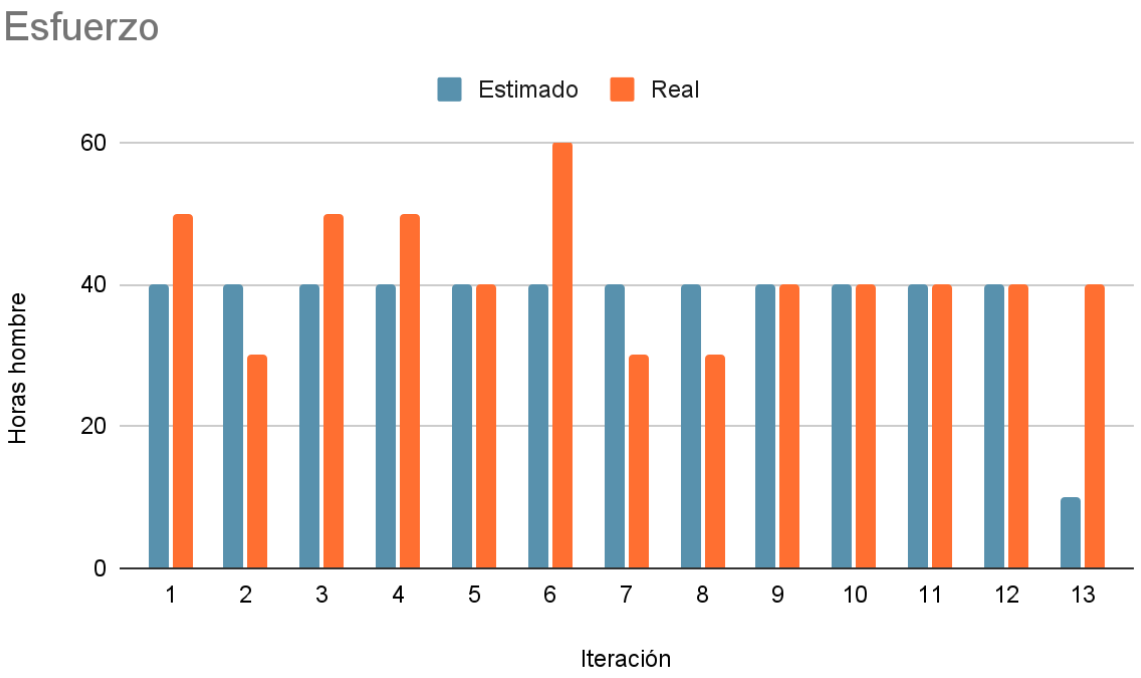

Fig. 141: Gráfico esfuerzo estimado vs esfuerzo real en función del número de iteración

Conclusiones

Las conclusiones se dividirán en distintos títulos a partir de los resultados experimentales que les dieron origen, para luego hacer unas apreciaciones generales.

Variación de la cantidad de imágenes

Se puede observar que, como es de esperarse, a mayor cantidad de imágenes, mayor duración del entrenamiento de la red. Esto sucede ya que el entrenamiento debe procesar mayor cantidad de imágenes cada vez.

En cuanto a los errores de entrenamiento, tienen una caída pronunciada antes de las 10 iteraciones (este valor se va reduciendo a medida que se utilizan más imágenes) para luego disminuir más lentamente. Lo mismo sucede con los errores de validación, que siempre se mantienen por arriba de los anteriores, siguiendo la misma pendiente pero oscilando más a medida que se agregan imágenes. Esto sucede porque el modelo se ajusta al set de entrenamiento y no al set de validación, que sirve a modo de control para evitar el overfitting.

Cantidad de imágenes	Error total	Error medio	Desvío estándar
100	$4,95\ x\ 10^{-1}$	$4,95\ x\ 10^{-1}$	$0,00\ x\ 10^{-0}$
500	$7,30\ x\ 10^{-3}$	$8,11\ x\ 10^{-4}$	$3,52\ x\ 10^{-4}$
1500	$1,18\ x\ 10^{-2}$	$4,22\ x\ 10^{-4}$	$1,54\ x\ 10^{-4}$
5000	$2,82\ x\ 10^{-2}$	$3,06\ x\ 10^{-4}$	$1,13\ x\ 10^{-4}$
10000	$4,58\ x\ 10^{-2}$	$2,45\ x\ 10^{-4}$	$1,19\ x\ 10^{-4}$

Tabla 76: Valores de error total, error medio y desvío estándar en testing con respecto a la cantidad de imágenes

Como se puede observar en la tabla anterior, para cien imágenes, el error medio es bastante alto con respecto a otros casos y las imágenes aparecen bastante difusas, similares a las entradas de la red (imágenes preprocesadas). Esto probablemente se deba principalmente al tamaño del batch con respecto al total del set y, en segundo lugar, a la cantidad de imágenes. Es probable que esta situación mejore un poco en la medida que se agreguen iteraciones al entrenamiento.

Pese a que el valor de error total aumenta a medida que se agregan imágenes (lo cual es de esperarse, ya que la suma de los errores aumentará a medida que sea mayor la cantidad de imágenes), el error medio disminuye considerablemente. De 500 a 1.500 imágenes se reduce a la mitad y los tiempos de ejecución se duplican, pero siguen estando dentro de un rango aceptable para el error obtenido. Además, el desvío estándar también se reduce a la mitad, lo que quiere decir que más imágenes tienen un error cercano a la media. Ya de 5.000 en adelante, los tiempos de ejecución son sumamente extensos para una reducción del error que no lo amerita (28% para 1.500 a 5.000 y 20% para 5.000 a 10.000).

Variación del tamaño de batch

Para empezar, el tiempo de ejecución se redujo a medida que se aumentó el tamaño de batch pero, a partir de un tamaño de 8 imágenes, la duración del entrenamiento se mantuvo constante. Esto quiere decir que si bien se recorre el set de imágenes más rápidamente, a medida que se agranda el batch, el procesamiento del mismo tiende a ser más lento.

En cuanto a los errores de entrenamiento y validación, ocurre el caso inverso al anterior. La caída pronunciada se produce en las primeras iteraciones para un batch de 1 imagen y se va moviendo hacia la derecha en el eje x a medida que se aumenta el batch. En el caso de 32 imágenes la caída pronunciada se da a lo largo de todas las iteraciones.

Tamaño del batch	Error total	Error medio	Desvío estándar
1	$4,80\ x\ 10^{-2}$	$6,40\ x\ 10^{-4}$	$7,92\ x\ 10^{-4}$
2	$1,95\ x\ 10^{-2}$	$5,27\ x\ 10^{-4}$	$3,60\ x\ 10^{-4}$
4	$1,46\ x\ 10^{-2}$	$8,11\ x\ 10^{-4}$	$4,27\ x\ 10^{-4}$
8	$7,43\ x\ 10^{-3}$	$8,26\ x\ 10^{-4}$	$4,41\ x\ 10^{-4}$
16	$2,82\ x\ 10^{-3}$	$7,04\ x\ 10^{-4}$	$1,91\ x\ 10^{-4}$
32	$1,38\ x\ 10^{-2}$	$6,90\ x\ 10^{-3}$	$1,33\ x\ 10^{-4}$

Tabla 77: Valores de error total, error medio y desvío estándar en testing con respecto al tamaño de batch

Según se observa en la tabla anterior, a medida que se aumenta el tamaño del batch, el error total disminuye pero, cuando llega a un batch de 16, el mismo comienza a aumentar. Esto se debe a que el set de datos no es lo suficientemente grande para el batch dado. Lo mismo se puede ver en las imágenes de muestra del set de testing para estos casos, son mayormente difusas no solamente en los círculos, sino también en el fondo (figs. 40 y 41).

Mientras que el error total disminuye, el error medio no sigue la misma tendencia. A partir de un batch de 4 aumenta, lo que quiere decir que los errores fluctúan mucho, hay casos con error bajo y casos con error alto, lo cual probablemente se deba a que el modelo no está lo suficientemente generalizado. En cuanto al desvío estándar, este disminuye constantemente, es decir, el error de las imágenes de testing están más cerca del error medio.

Es probable que los valores de error para batch cada vez más grandes mejoren con un set de datos más grande.

Variación de número de iteraciones

Como es de esperarse, la duración del entrenamiento es proporcional al número de iteraciones, ya que se debe recorrer más veces el set de datos. A su vez, la forma del gráfico de error de entrenamiento/validación en función del número de iteraciones solo se va alargando, continuando la misma tendencia. Comienza con una caída rápida en las primeras iteraciones, para mantenerse estable hasta el final con las características oscilaciones.

Iteraciones	Error total	Error medio	Desvío estándar
10	$1,00\ x\ 10^{-1}$	$1,34\ x\ 10^{-3}$	$1,49\ x\ 10^{-3}$
50	$7,93\ x\ 10^{-2}$	$1,06\ x\ 10^{-3}$	$1,60\ x\ 10^{-3}$
75	$7,93\ x\ 10^{-2}$	$1,06\ x\ 10^{-3}$	$1,60\ x\ 10^{-3}$
100	$7,62\ x\ 10^{-2}$	$1,02\ x\ 10^{-3}$	$1,48\ x\ 10^{-3}$
150	$7,62\ x\ 10^{-2}$	$1,02\ x\ 10^{-3}$	$1,48\ x\ 10^{-3}$
200	$7,15\ x\ 10^{-2}$	$9,54\ x\ 10^{-4}$	$1,27\ x\ 10^{-3}$

Tabla 78: Valores de error total, error medio y desvío estándar en testing con respecto al número de iteraciones

En la tabla anterior se puede observar que, en general, el error total, el error medio y el desvío estándar se mantuvieron bastante constantes a partir de 50 iteraciones. Pese a que hay pequeñas variaciones, no son lo suficientemente grandes como para justificar la ejecución más larga. Lo mismo se puede ver en las imágenes de testing, que a partir de 50 iteraciones son bastante nítidas y los colores se diferencian bien, pero no hay grandes mejoras después de este punto. Para la cantidad de imágenes, 500, y el tamaño de batch, 1, con 50 iteraciones los resultados son aceptables y hasta se podría disminuir este número.

Variación de las proporciones del set de datos

En este caso, la duración del entrenamiento no se vio muy afectada. Esto se debe principalmente a que el set de entrenamiento sumado al set de validación, que son los que intervienen en esta etapa del procesamiento, se mantuvo más o menos constante (alrededor de 450 imágenes). Probablemente este resultado hubiera variado más con un set de datos total mayor.

Los gráficos de los errores de entrenamiento y validación son bastante similares para todos los casos, imitando el comportamiento característico de esta red: una disminución abrupta en las primeras iteraciones y una cola larga bastante estable hacia las últimas iteraciones.

Proporciones del set de datos	Error total	Error medio	Desvío estándar
60 - 20 -20	$7,69\ x\ 10^{-2}$	$7,69\ x\ 10^{-4}$	$5,65\ x\ 10^{-4}$
70 - 15 - 15	$4,80\ x\ 10^{-2}$	$6,40\ x\ 10^{-4}$	$7,92\ x\ 10^{-4}$
80 - 10 - 10	$4,47\ x\ 10^{-2}$	$8,94\ x\ 10^{-4}$	$1,28\ x\ 10^{-3}$
90 - 5 - 5	$2,21\ x\ 10^{-2}$	$8,85\ x\ 10^{-4}$	$9,63\ x\ 10^{-4}$

Tabla 79: Valores de error total, error medio y desvío estándar en testing con respecto a la proporción del set de datos (entrenamiento - validación - testing)

El mejor caso para el error medio se dio con la proporción de 70 % para el set de entrenamiento, pese a no tener el error total más bajo, este se encuentra cercano al obtenido en las demás ejecuciones. Esta proporción es una de las más utilizadas en deep learning.

Para el caso de 60 - 20 - 20, el error medio y el total aumentan, mientras que el desvío estándar disminuye. Esto quiere decir que casi todos los casos tienen un error bastante similar al error medio, lo cual muestra que el entrenamiento no fue suficiente para cubrir todos los casos que después se testearon, es decir, está cercano al underfitting. En las imágenes de muestra, se puede ver que todos los círculos son bastante parecidos en tamaño y color (fig. 61).

Para los dos casos restantes, con mayor proporción de datos de entrenamiento, se observa un incremento del error medio, pero una disminución del error total. Esto se da porque hay casos más precisos pero, a la vez, casos con mayor error. Con lo cual, se puede concluir que el modelo se especializa más en el set de entrenamiento, incurriendo en overfitting, con lo cual no puede adaptarse del todo a imágenes distintas o nuevas. Si se miran las imágenes de ejemplo, hay círculos que no están representados en la salida de la red (figs. 65 y 68).

Variación del rango de permitividades

En cuanto a la duración del entrenamiento, esta se mantuvo constante para todos los casos ya que el tamaño del set y del batch tienen el mismo valor para todos los casos. La forma del gráfico de error de entrenamiento frente al número de iteraciones se va haciendo más curvo (siempre

disminuyendo) a medida que la imagen tiene mayor contraste (mayor diferencia de permitividades). Por otra parte, el error de validación se hace cada vez más constante, manteniendo las características oscilaciones que se aplanan ante un mayor contraste.

Rango de permitividades	Error total	Error medio	Desvío estándar
1 - 1,005	$2,26\ x\ 10^{-5}$	$3,01\ x\ 10^{-7}$	$7,00\ x\ 10^{-8}$
1 - 1,05	$8,56\ x\ 10^{-4}$	$1,14\ x\ 10^{-5}$	$1,75\ x\ 10^{-5}$
1 - 1,5	$7,15\ x\ 10^{-2}$	$9,54\ x\ 10^{-4}$	$1,27\ x\ 10^{-3}$
1 - 2	$3,15\ x\ 10^{-1}$	$4,20\ x\ 10^{-3}$	$4,52\ x\ 10^{-3}$
1 - 5	$1,49\ x\ 10^{-1}$	$1,98\ x\ 10^{-1}$	$2,06\ x\ 10^{-1}$

Tabla 80: Valores de error total, error medio y desvío estándar en testing con respecto al rango de permitividades (mínimo - máximo)

En este caso, los errores no aportan demasiada información. Según la tabla, tanto el error total, como el error medio y el desvío estándar todos aumentan a medida que aumenta el rango de permitividades. Esto es lógico que suceda calculando el error cuadrático medio ya que, una imagen de bajo contraste donde las diferencias de permitividades entre dos pixeles estén en el orden de las milésimas, va a tener un error más pequeño que otra donde la diferencia esté en la unidad.

Si observamos las imágenes obtenidas de la red aplicada al set de testing, se puede ver que ante un rango pequeño de permitividades, la imágen obtenida es más bien difusa (figs. 70, 71, 73 y 74). Al aumentar el contraste, la imagen de salida mejora. Pero para el caso de la ejecución 26 por ejemplo, la imagen preprocesada ya es poco nítida, con lo cual la salida de la red es más pobre aún (figs. 79 y 80). Con lo cual se puede concluir que para rangos pequeños, la red neuronal no funciona muy bien, mientras que para rangos grandes, el preprocesamiento falla.

Procesamiento con GPU

En este caso se varió el tamaño del batch desde 1 a 32 (utilizando potencias de 2), con un set de 500 imágenes y 200 iteraciones pero, esta vez, se utilizó el procesamiento en GPU en lugar de CPU. Acá se pudo comprobar cómo se reduce el tiempo de ejecución drásticamente a medida que se aumenta el tamaño del batch:

Tamaño del batch	Duración del entrenamiento con GPU	Duración del entrenamiento con CPU (aproximado)	Relación
1	4:13:18	2:45:00	1:0,65
2	2:30:13	2:13:00	1:0,89
4	1:20:48	1:57:00	1:1,4
8	1:05:15	1:45:00	1:1,5
16	0:33:32	1:45:00	1:3
32	0:26:08	1:41:00	1:4

Tabla 81: Duración del entrenamiento con GPU y CPU* con respecto al tamaño del batch y su relación. * La aproximación se hizo multiplicando el tiempo obtenido en las ejecuciones 6 a 11 por 4 (ya que allí se realizaron 50 iteraciones y en estas 200)

Como se puede observar en la anterior tabla, el procesamiento para un batch de 1 imagen fue más largo con GPU que con CPU. Esto puede deberse a los límites impuestos por el software donde se ejecutó GenPer, pero a su vez sirve de referencia para indicar que si se quiere mejorar la performance, entonces se debe aumentar el tamaño del batch. Con un batch de 2, la duración fue prácticamente la misma pero, a partir de 4 en adelante, se puede ver claramente la diferencia de tiempos. Con un batch de 32, la duración se redujo a la quinta parte prácticamente. Este resultado impulsó el siguiente experimento, aumentando el set de datos a 5.000 imágenes, dado que para 500, el error es bastante alto y las imágenes son difusas. Esto está relacionado con las conclusiones que obtuvimos con la variación del tamaño de batch.

Para las ejecuciones 32 y 33 se utilizaron 5.000 imágenes, un batch de 16 y un número de iteraciones de 100 y 200, respectivamente. Si comparamos los tiempos con la ejecución 3, que pese a tener un batch de 8, podemos suponer que los tiempos no variarán demasiado (como sucede entre las ejecuciones 8 y 9), el tiempo de ejecución se reduce a casi la mitad y los errores obtenidos son aceptables. Las imágenes de testing de salida de la red son similares a las originales (figs. 106, 107 y 109) y esto se debe en gran parte a la mayor cantidad de imágenes de testing.

Pruebas de validación

Se puede observar que la diferencia en el tiempo de ejecución es de casi el triple en MATLAB, aún utilizando un algoritmo de optimización como SGD. Como la red neuronal está compuesta por las mismas capas en ambos casos y no hay grandes diferencias al respecto, probablemente la diferencia de performance se deba a la forma en la que está implementado el código de la red

neuronal en cada lenguaje. La versión de MATLAB utiliza una librería llamada MatConvNet, cuya última actualización se produjo en el 2018, mientras que GenPer utiliza PyTorch, una librería ampliamente utilizada en el ámbito del deep learning y que está en constante actualización.

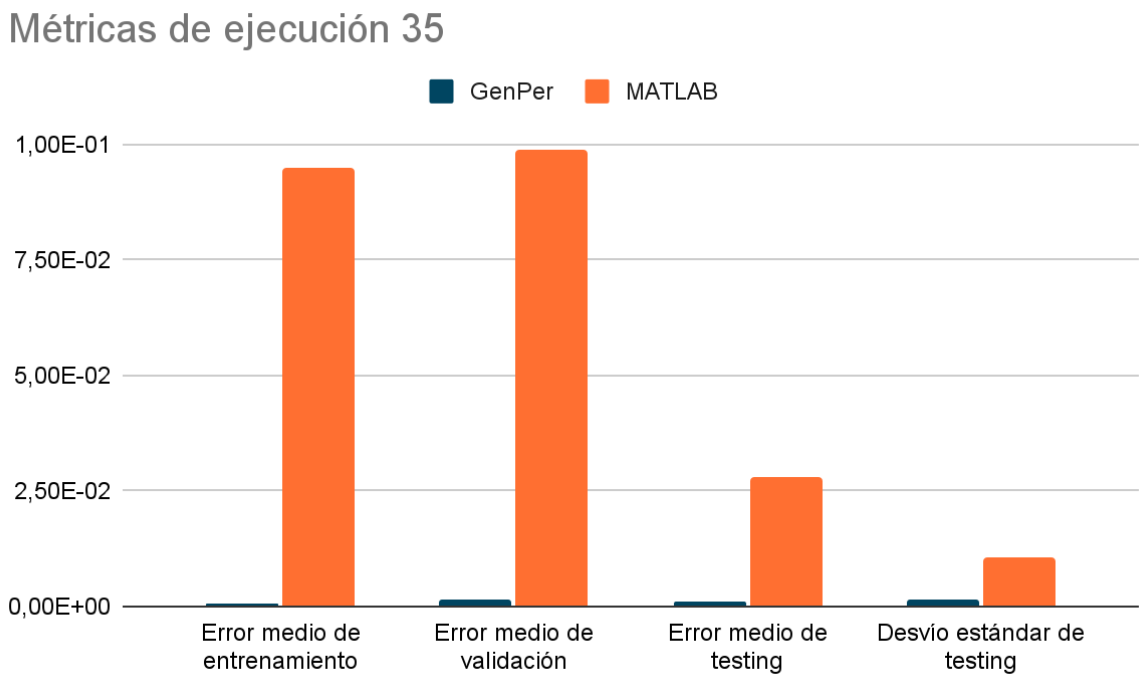

Fig. 142: Métricas de ejecución 35: GenPer vs MATLAB

Si observamos la figura anterior, se puede ver claramente la diferencia entre el desempeño de GenPer frente a la implementación en MATLAB utilizando el algoritmo de optimización AdamW. En todos los casos es, por lo menos, 10 veces menor, lo cual se ve reflejado claramente en los gráficos comparativos del error de cada implementación. Por ejemplo, en la comparación del error de cada imagen de testing, en ningún caso se supera ni iguala al resultado de MATLAB. Si vemos la imagen de testing en el caso de MATLAB (fig. 113) pareciera ser bastante exacta con respecto a la original, pero si nos enfocamos en la escala de colores, se puede observar que no es la misma para ambas imágenes, sino que la salida de la red resulta con permitividades un poco más bajas que la original. Esto es lo que probablemente genere un mayor error. Por otra parte, si vemos las imágenes de testing para GenPer (figs. 114, 115 y 116), pese a ser bastante difusas o faltar algunos círculos (esto puede deberse a las proporciones del set de datos como se vio antes, que pueden presentar cierto overfitting), estas tienen una escala de colores similar a la original, lo que probablemente haga que el error se reduzca.

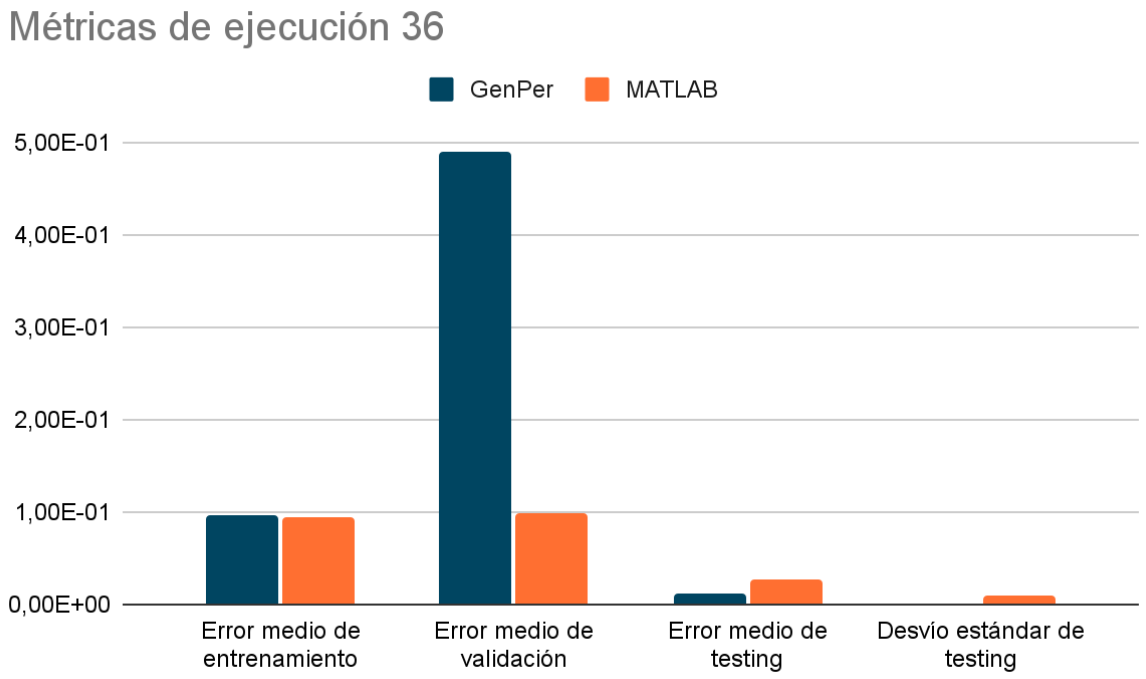

Fig. 143: Métricas de ejecución 36: GenPer vs MATLAB

Ahora bien, si observamos la figura anterior, que representa la comparativa de errores medios y el desvío estándar para cada caso, utilizando un algoritmo de optimización SGD con coeficiente fijo en GenPer, estos presentan valores bastante similares. El único caso en el que difieren un poco es en el error medio de validación. Esto, siendo un paso del entrenamiento, donde intervienen factores aleatorios también, no define si los modelos son similares o no. Además, si se observan las imágenes de prueba de GenPer de la ejecución 36 (figs. 121, 122 y 123), estas presentan el mismo efecto que se describió anteriormente: el rango de permitividades es ligeramente menor, haciendo que aumente el error.

Experimento adicional: rectángulos

Se puede comparar a la ejecución 37 con la ejecución 13, que tiene sus mismos parámetros con imágenes de círculos. En cuanto a tiempo de entrenamiento, si bien es más bajo, esta diferencia puede estar ligada a muchos factores relacionados a la utilización de la CPU, tareas ejecutadas en paralelo, etc. ya que, a priori, no hay ninguna razón para pensar que esta ejecución sea más rápida. En cuanto a los valores de error, en la ejecución 37 estos tienden a ser menores, lo cual se puede explicar si miramos las imágenes de prueba (figs. 129, 130 y 131). Los rectángulos son figuras más simples, los lados son líneas rectas y son más fáciles de delimitar. Mientras que los círculos, al estar

representados con pixeles, son más complejos, sus límites más difusos y las superposiciones más difíciles de delimitar.

Probablemente si se aumentara la cantidad de iteraciones o la cantidad de imágenes, estos resultados serían aún mejores.

Experimento adicional: MNIST

En el caso de la ejecución con el set de MNIST, no se cuenta con ninguna ejecución con parámetros similares ya que es un set de datos bastante extenso. Pero si nos referimos a la ejecución 5, que posee una gran cantidad de datos (8.500 imágenes para entrenamiento y testing), podríamos llegar a hacer una analogía. Aproximando el tiempo de entrenamiento multiplicado por 7 (60.000 / 8.500 ~= 7), entonces llegamos a que el tiempo de ejecución total es de 60 horas, bastante similar al de la ejecución con MNIST.

En cuanto a los errores, esta última tiene valores un poco mayores, lo cual se puede explicar de manera similar a lo ocurrido con las imágenes de rectángulos. Los dígitos de MNIST son más complejos aún que los círculos, su dibujo se encuentra disperso en la imagen y tienden a producir mayores errores. Sin embargo, al ser un set de datos tan extenso, se puede ver en las imágenes de testing que los resultados son sorprendentemente buenos (figs. 135, 136 y 138). En todos los casos se puede definir que hay un dígito siendo representado pero, a su vez, existen varios casos donde el mismo es confundido por otro (figs. 137 y 139). Esto es un problema común en este set de datos y se debe a que la red neuronal utiliza métodos estadísticos para predecir.

Conclusiones finales

Algunas conclusiones finales respecto de los análisis hechos anteriormente:

- No es necesaria una gran cantidad de imágenes para obtener buenos resultados con GenPer. Como se ha visto anteriormente, las redes neuronales de tipo U-Net aprovechan la información de cada píxel de las imágenes para poder entrenarse, con lo cual los set de datos necesarios son más bien pequeños. Unas 1.500 imágenes serían suficientes para lograr buenas predicciones.
- El tamaño de batch dependerá en gran medida del tamaño del set de datos y será aún más importante en el caso de que la ejecución sea mediante GPU, ya que la misma se verá acelerada ante un mayor batch. Para 1.500 imágenes, un batch de 8 probablemente dé buenos resultados.
- El número de iteraciones dependerá del tamaño del batch pero, por lo general, si se utiliza

un algoritmo de tipo AdamW como lo hace GenPer, dentro de las primeras iteraciones se lograrán resultados con errores aceptables y estables. 50 iteraciones obtuvieron buenos resultados en casi todos los casos del presente informe.

- Utilizar la típica proporción del set de datos 70 - 15 - 15 genera errores aceptables sin incurrir ni en overfitting ni underfitting.
- GenPer funciona bien con rangos bajos de permitividades, así como sucede en tejidos blandos. El problema surge cuando el contraste es muy bajo, donde la red neuronal comienza a fallar, o cuando este es muy alto, donde el preprocesamiento comienza a fallar. En estos casos es una buena idea aumentar el número de imágenes y el número de iteraciones.
- GenPer es capaz no sólo de obtener los valores de permitividades para formas circulares, sino también rectangulares y de dígitos, superpuestas o no. Esto indica que podría llegar a utilizarse para obtener formas aún más diversas, como combinaciones de estos últimos o formas más indefinidas.

Referencias

(1) Z. Wei and X. Chen, “Deep learning schemes for full-wave nonlinear inverse scattering problems,” IEEE Transactions on Geoscience and Remote Sensing, 57 (4), pp. 1849-1860, 2019. URL: https://www.ece.nus.edu.sg/stfpage/elechenx/Papers/TGRS_Learning.pdf

(2) Semenov, Serguei. “Microwave tomography: review of the progress towards clinical applications.” Philosophical transactions. Series A, Mathematical, physical, and engineering sciences vol. 367,1900 (2009): 3021-42. doi:10.1098/rsta.2009.0092.
URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2696111/

(3) Biograph Vision
URL: https://www.siemens-healthineers.com/cl/molecular-imaging/pet-ct/biograph-vision

(4) Wu and H. Wang, "Microwave Tomography for Industrial Process Imaging: Example Applications and Experimental Results.," in *IEEE Antennas and Propagation Magazine*, vol. 59, no. 5, pp. 61-71, Oct. 2017, doi: 10.1109/MAP.2017.2731201.
URL: https://www.research.manchester.ac.uk/portal/files/58214111/IEEE_APM_Microwave_Tomography_final.pdf

(5) M. Persson, Xuezhi Zeng and A. Fhager, "Microwave imaging for medical applications," *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, 2011, pp. 3070-3072.
URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.827.5828&rep=rep1&type=pdf

(6) Ronneberger, Olaf & Fischer, Philipp & Brox, Thomas. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. LNCS. 9351. 234-241. 10.1007/978-3-319-24574-4_28.
URL: https://arxiv.org/pdf/1505.04597v1.pdf

(7) Harshall Lamba. (2019). Understanding Semantic Segmentation with UNET.
URL: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

(8) Z. Wei and X. Chen. (2019). "Solving full-wave-nonlinear-inverse-scattering-problems with back-propagation-scheme".
URL: https://github.com/eleweiz/Solving-full-wave-nonlinear-inverse-scattering-problems-with-back-propagation-scheme

(9) Jason Dsouza (2020). "What is a GPU and do you need one in Deep Learning?".
URL: https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d

⑽ Google Colaboratory.

URL: https://colab.research.google.com/

⑾ LeCun, Yann and Cortes, Corinna. "MNIST handwritten digit database." (2010).

URL: http://yann.lecun.com/exdb/mnist/

⑿ Scrum: Metodología de desarrollo de Software, con ciclo de vida iterativo.

URL: https://scrummethodology.com/

⒀ Kanban.

URL: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

⒁ Repositorio git de GenPer.

URL: https://github.com/estanislaoledesma/genper