REDES NEURONALES GRUPO 8

1.
INTRODUCCIÓN

Introducción

- Red neuronal configurable
  - Capas ocultas
  - Neuronas por capa
  - Cantidad de épocas para calcular error

- División de la información
  - Conjunto de entrenamiento
  - Conjunto de testeo

- Se alimenta la red con patrones del conjunto de entrenamiento
  - Algoritmo *forward* en forma de *batches* sucesivos (épocas)

- Cálculo del error de entrenamiento
  - Decidir si la red ha aprendido o sobre-aprendido

2.
IMPLEMENTACIÓN

Implementación

Lenguaje
Python

Clases

1. Neural Network
    a. Matrices de pesos de las distintas capas
    b. Learning rate
    c. Patrones de entrenamiento y testeo
    d. Algoritmo forward
    e. Algoritmo backpropagation
    f. Funciones
        i. Tanh
        ii. Sigmoid
2. Properties

Implementación

Patrones de aprendizaje

- Normalizar los patrones de entrada
- Normalizar las salidas esperadas de los patrones
  - Para calcular el error

Función de normalización

- Funcional lineal
- Entrada
  - Mínimo -> **-1**
  - Máximo -> **1**
- Salida
  - Tanh
    - Mínimo -> **-0.8**
    - Máximo -> **0.8**
  - Exp
    - Mínimo -> **0.1**
    - Máximo -> **0.9**

Implementación

Aprendizaje

- Patrones de entrenamiento aleatorios
- Entrenamiento batch
- Parámetros de configuración
  - Learning rate
  - Momentum
- Algoritmos
  - Forward
  - Back propagation

3.
RESULTADOS

Evaluación

- Mismo conjunto de entrenamiento y testeo
  - Previamente generados a partir del archivo de datos del terreno
  - División porcentual (50%) aleatoria

- Diversas arquitecturas
  1. [ ] - Ninguna capa oculta
  2. [10]
  3. [30]
  4. [10, 10]
  5. [30, 30]
  6. [20, 20, 20]
  7. [10, 10, 10, 10, 10]

Parámetros de aprendizaje

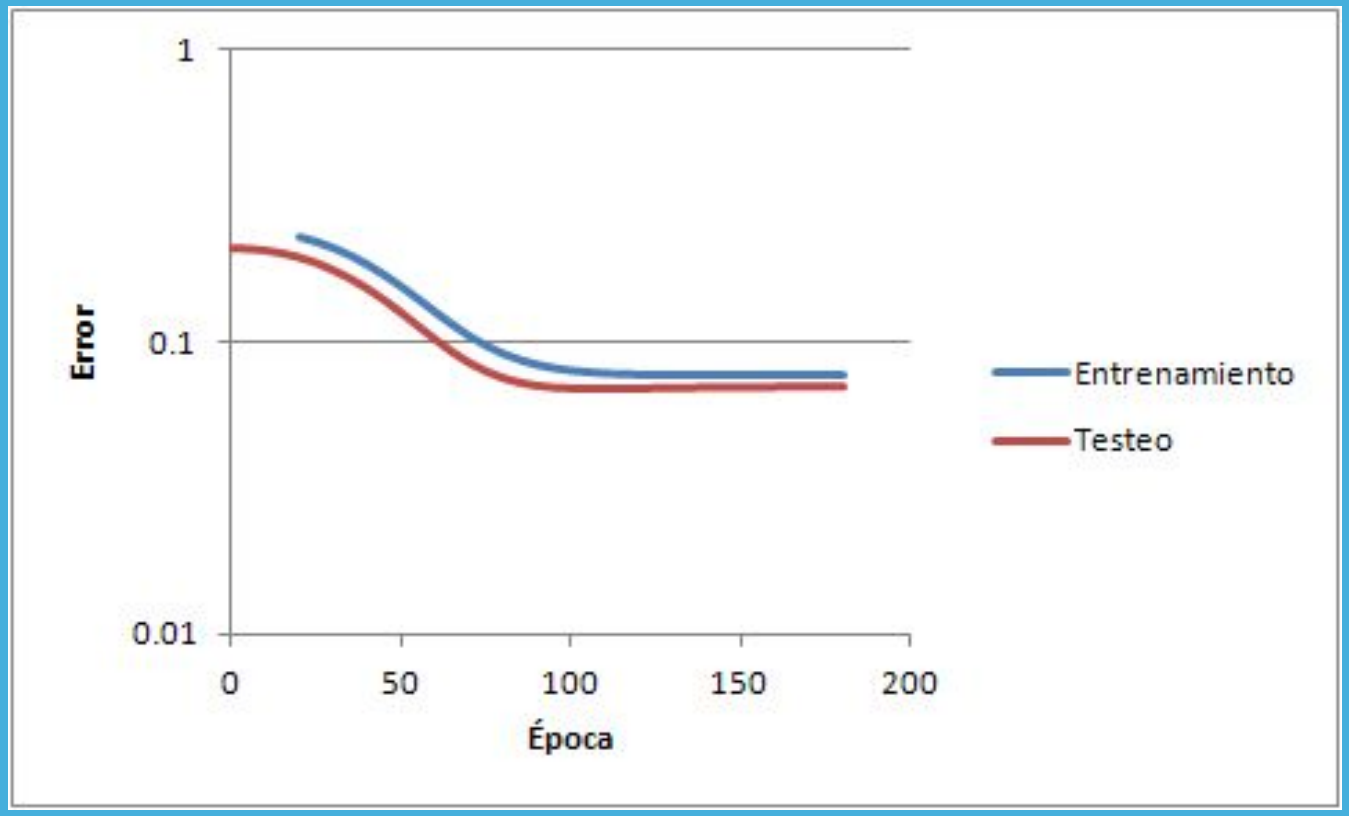
- Función de activación
  - g(h): tanh(bh) con b = 1
  - Ƞi = 0,0001

- Variaciones
  - Etha adaptativo, actualizable cada 2 épocas
    - a = 5e-7
    - b = 0,7
    - probabilidad de deshacer (p) = 0,75
  - Momentum
    - Ɑ = 0,9
  - Suma de 0,1 a la derivada
    - Evitar que de cero

- Nota: Los próximos gráficos son con escala logarítmica con base 10

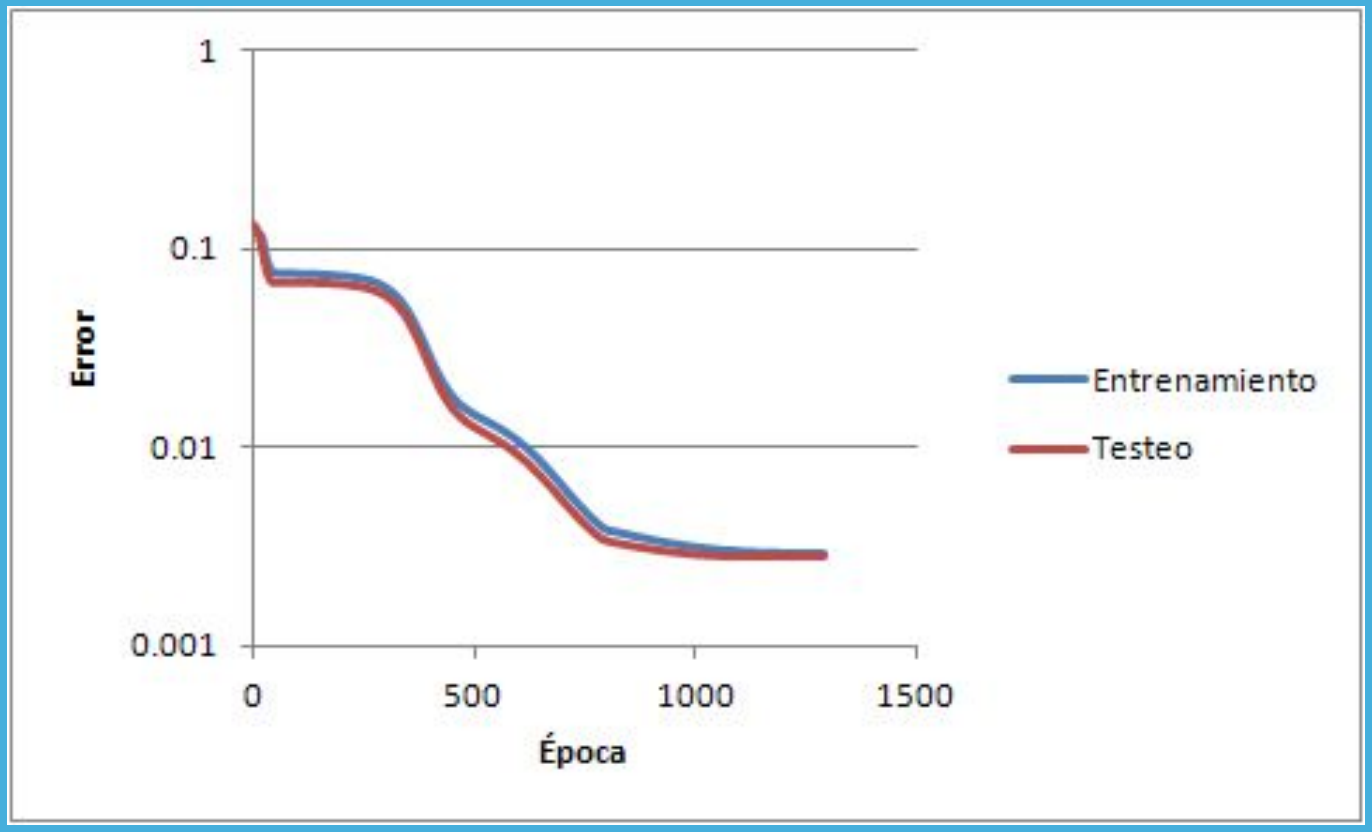
Error de la red con la arquitectura 1

[ ] - Ninguna capa oculta



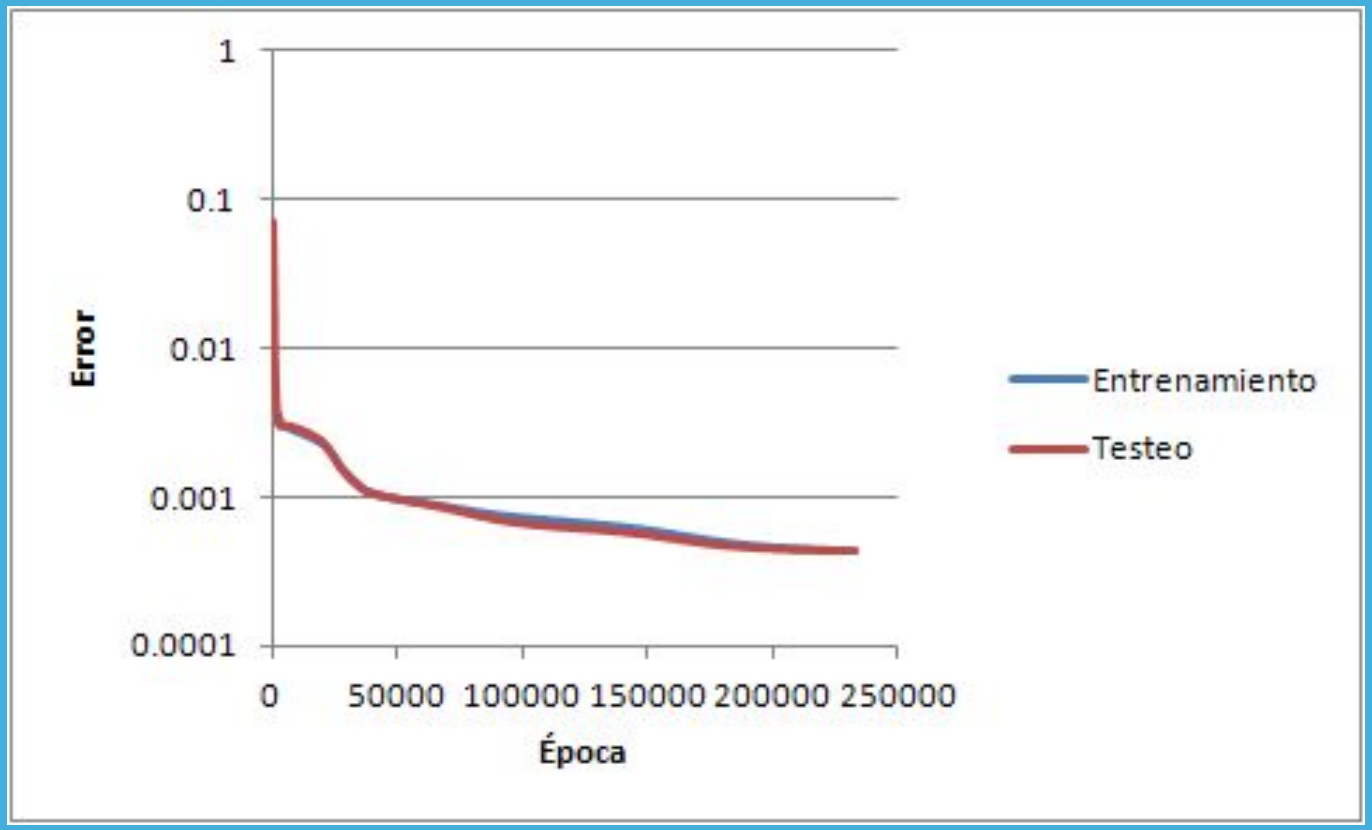
Error de la red con la arquitectura 2

[10]



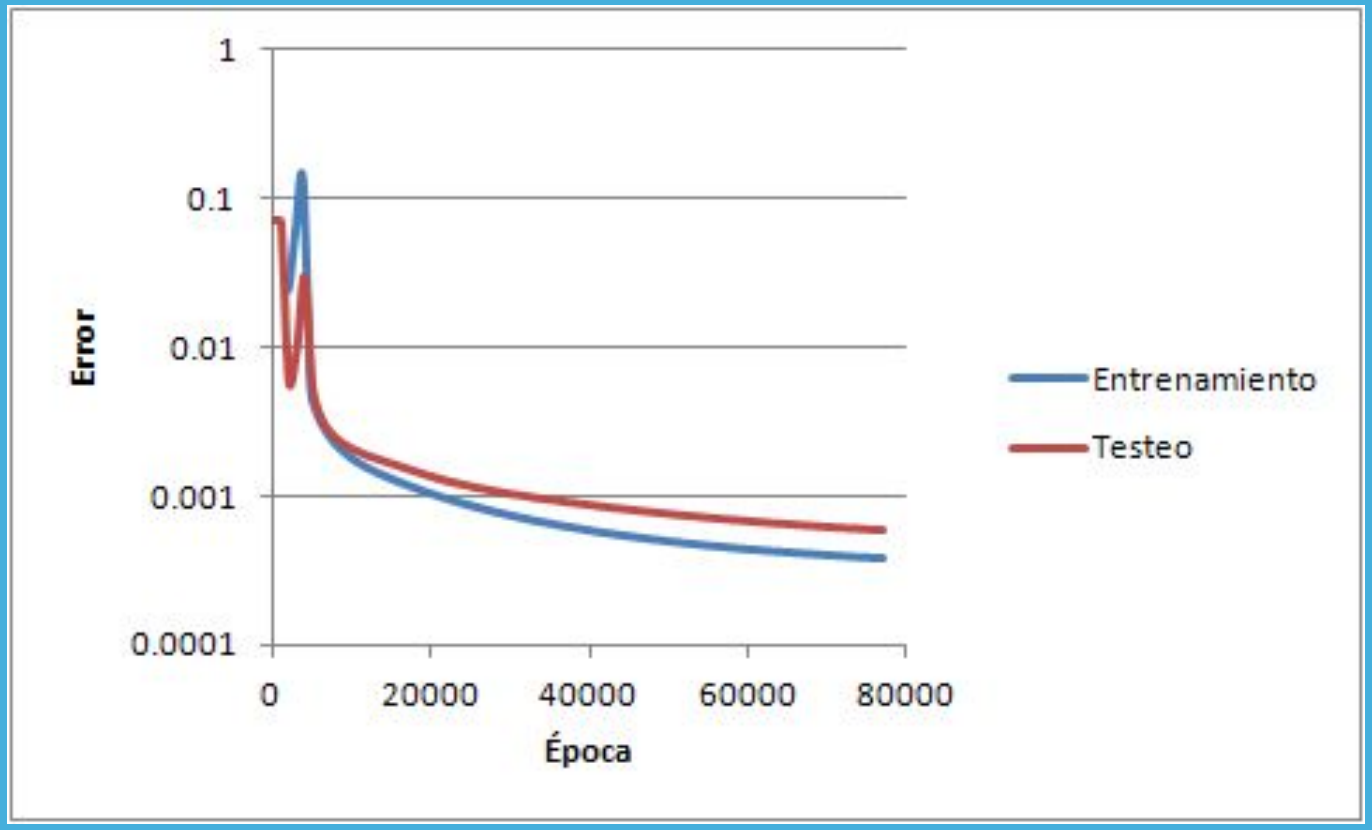
Error de la red con la arquitectura 3

[30]



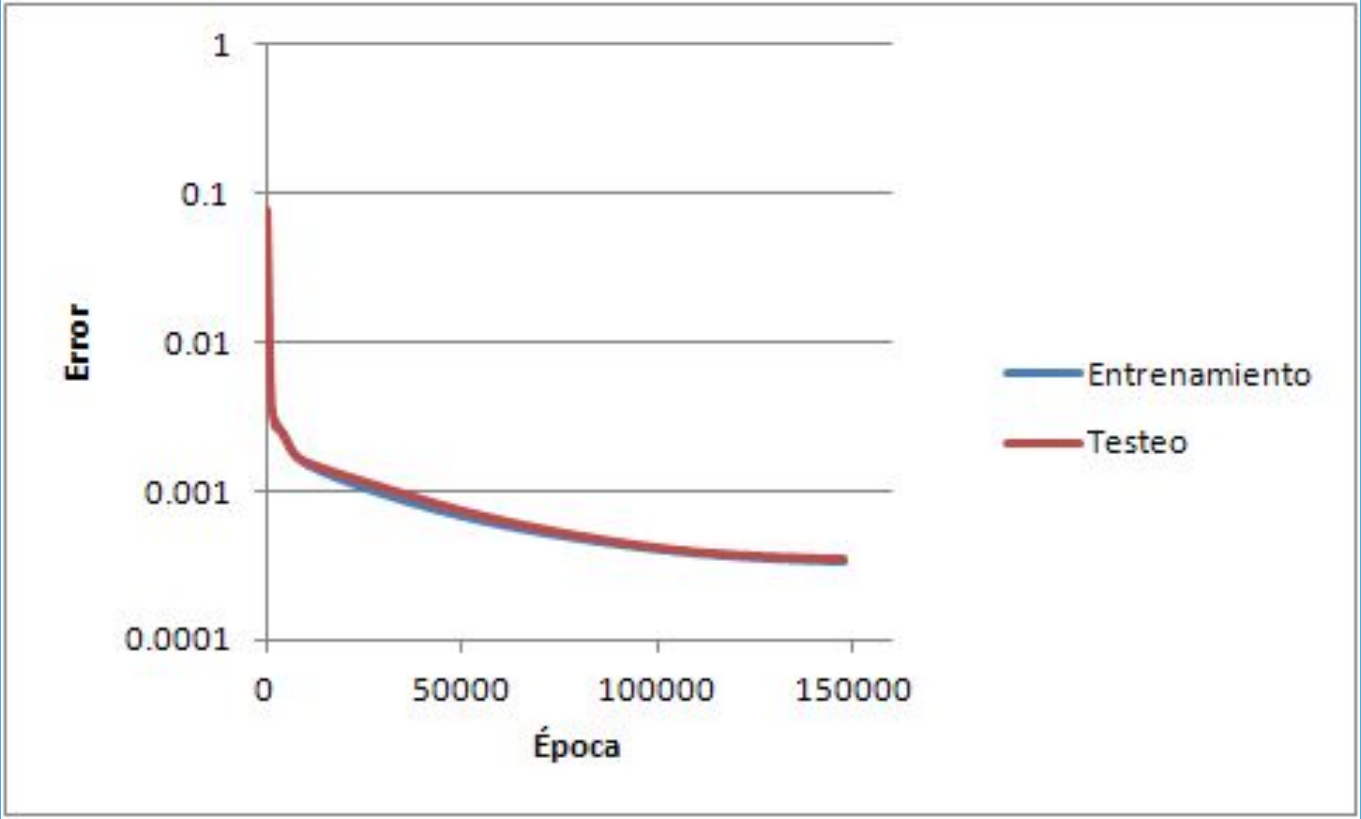
Error de la red con la arquitectura 4

[10, 10]



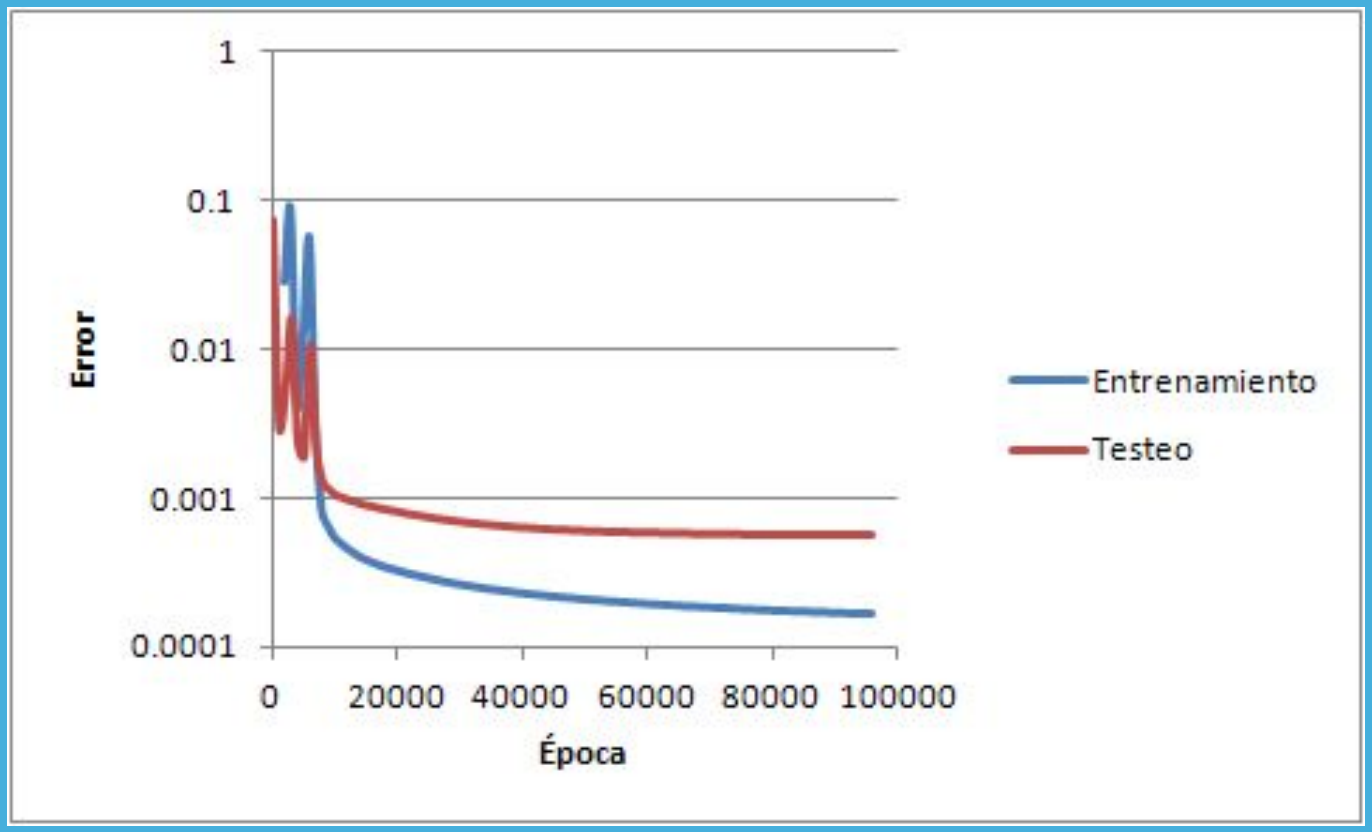
Error de la red con la arquitectura 5

[30, 30]



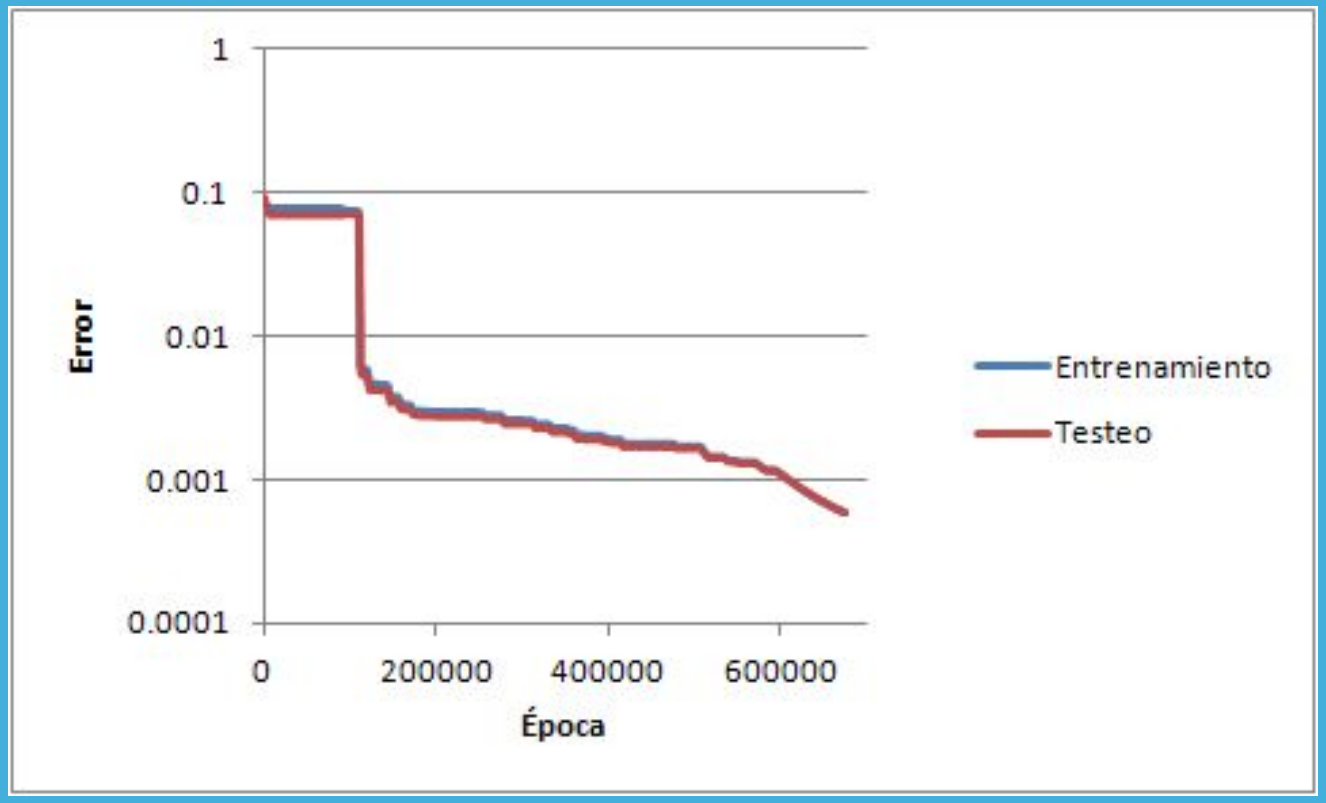
Error de la red con la arquitectura 6

[20, 20, 20]



Error de la red con la arquitectura 7

[10, 10, 10, 10, 10]

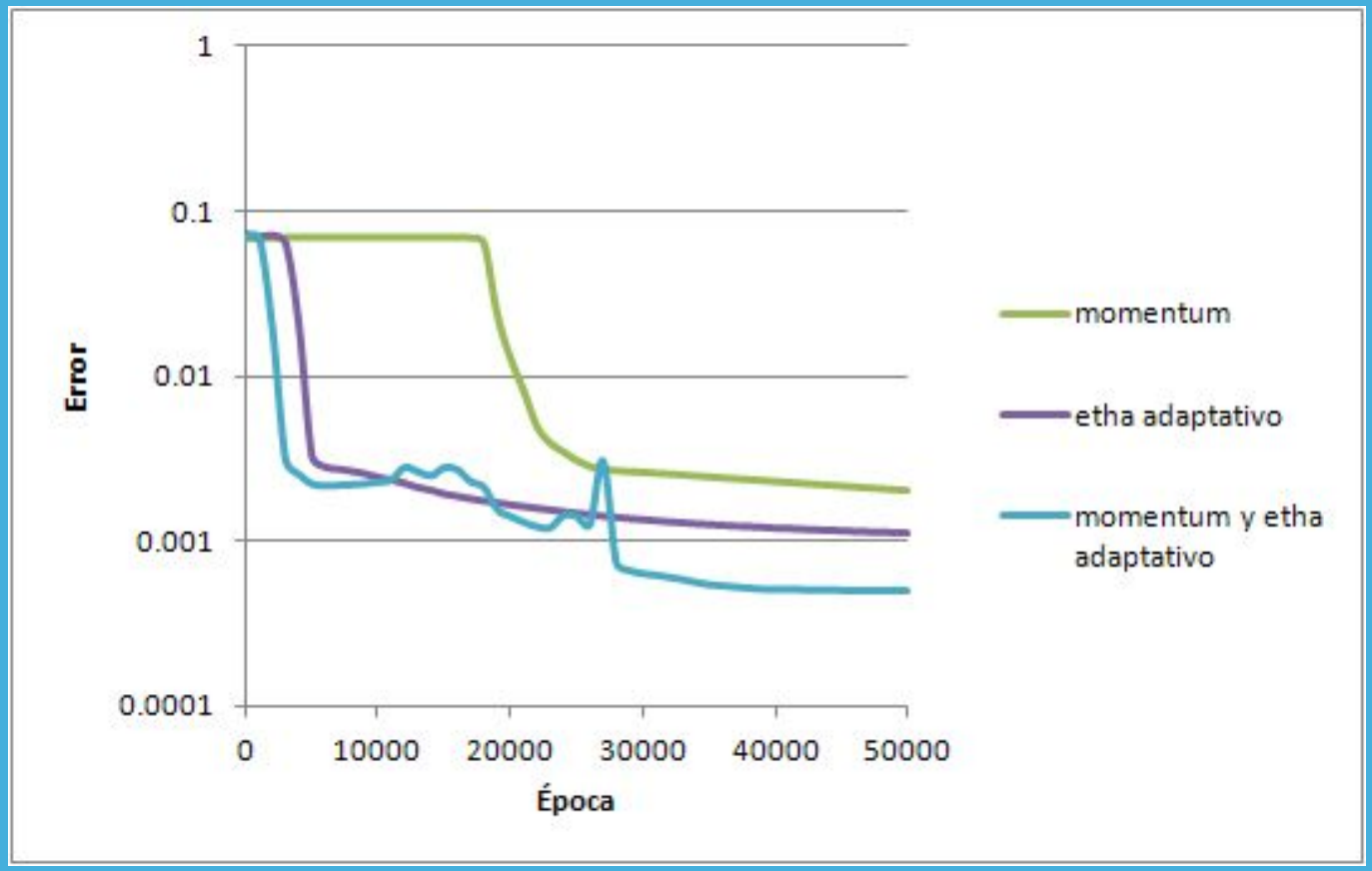


Comparación de errores de la red

Arquitectura	Mín. error de entrenamiento alcanzado	Mín. error de testeo alcanzado
1 - [0]	0.07628695	0.06989342
2 - [10]	0.00293081	0.00287611
3 - [30]	0.00044182	0.0004365
4 - [10, 10]	0.00038096	0.00058133
5 - [30, 30]	0.00033776	0.00034388
6 - [20, 20, 20]	0.00017006	0.00057585
7 - [10, 10, 10, 10, 10]	0.00059725	0.00058852

Aprendizaje de la red con distintas variaciones

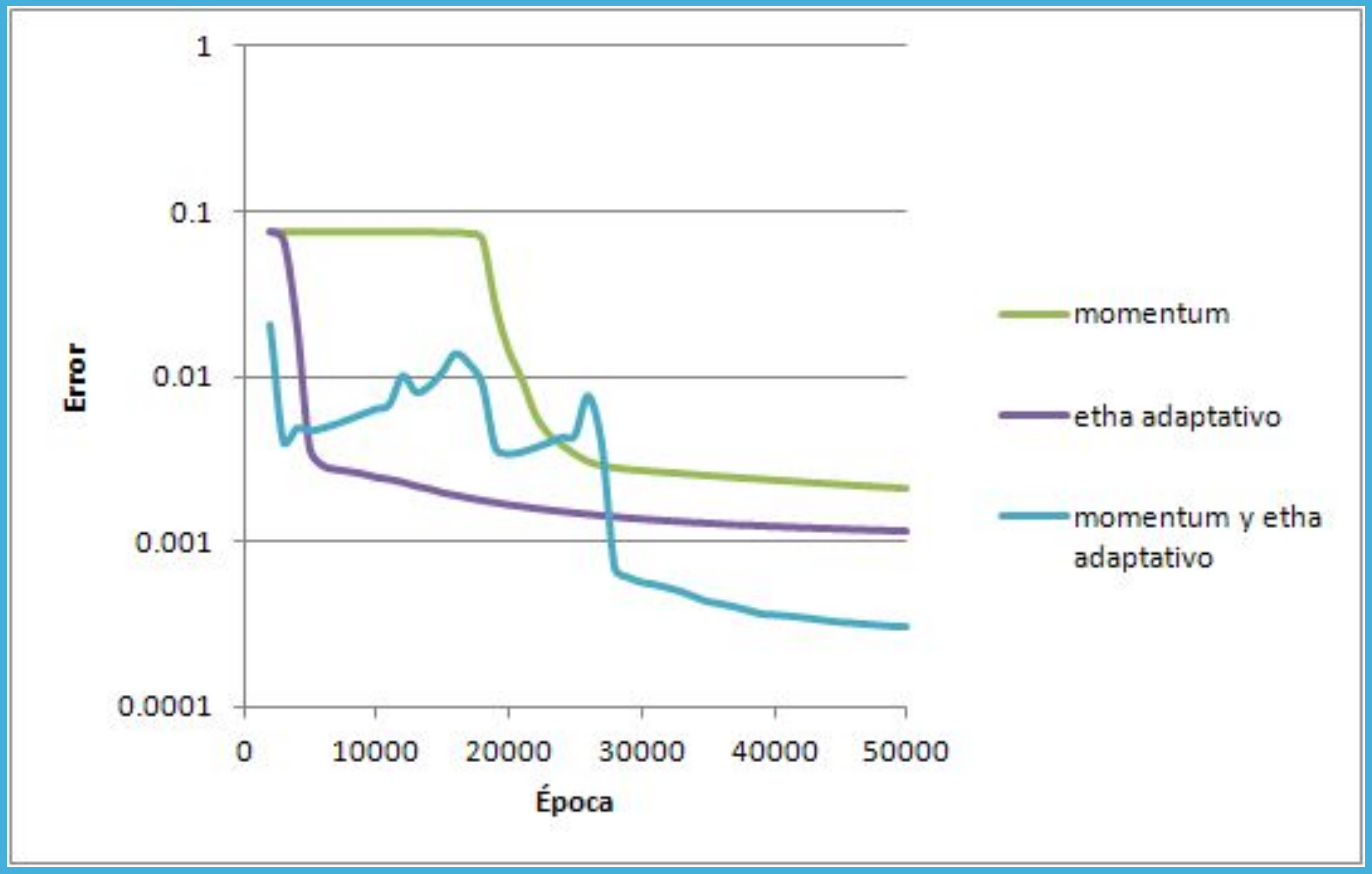
Error de entrenamiento en [10, 10]



Error de entrenamiento con distintas variaciones del algoritmo backpropagation *utilizando tanh como función de activación.*

Aprendizaje de la red con distintas variaciones

Error de testeo en [10, 10]

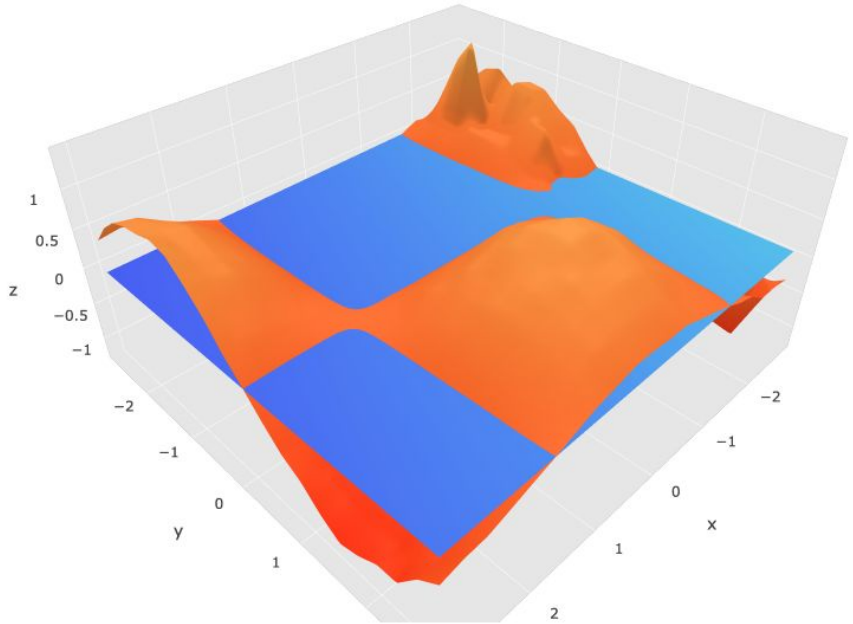


Error de testeo con distintas variaciones del algoritmo backpropagation *utilizando tanh como función de activación.*

4.
COMPARACIÓN DE TERRENOS

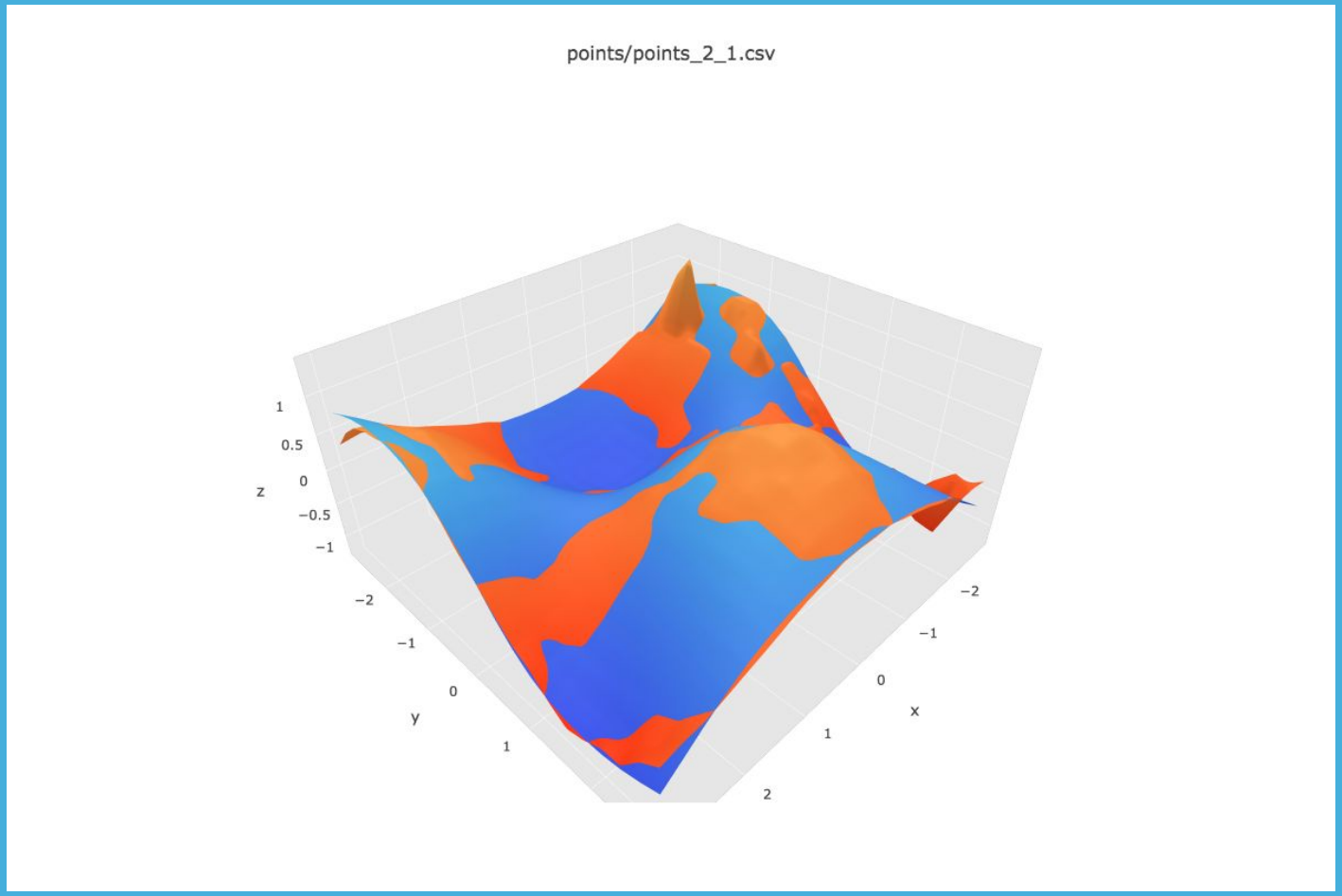
Terrenos arquitectura 1

Datos conocidos Datos obtenidos [ ] - Ninguna capa oculta



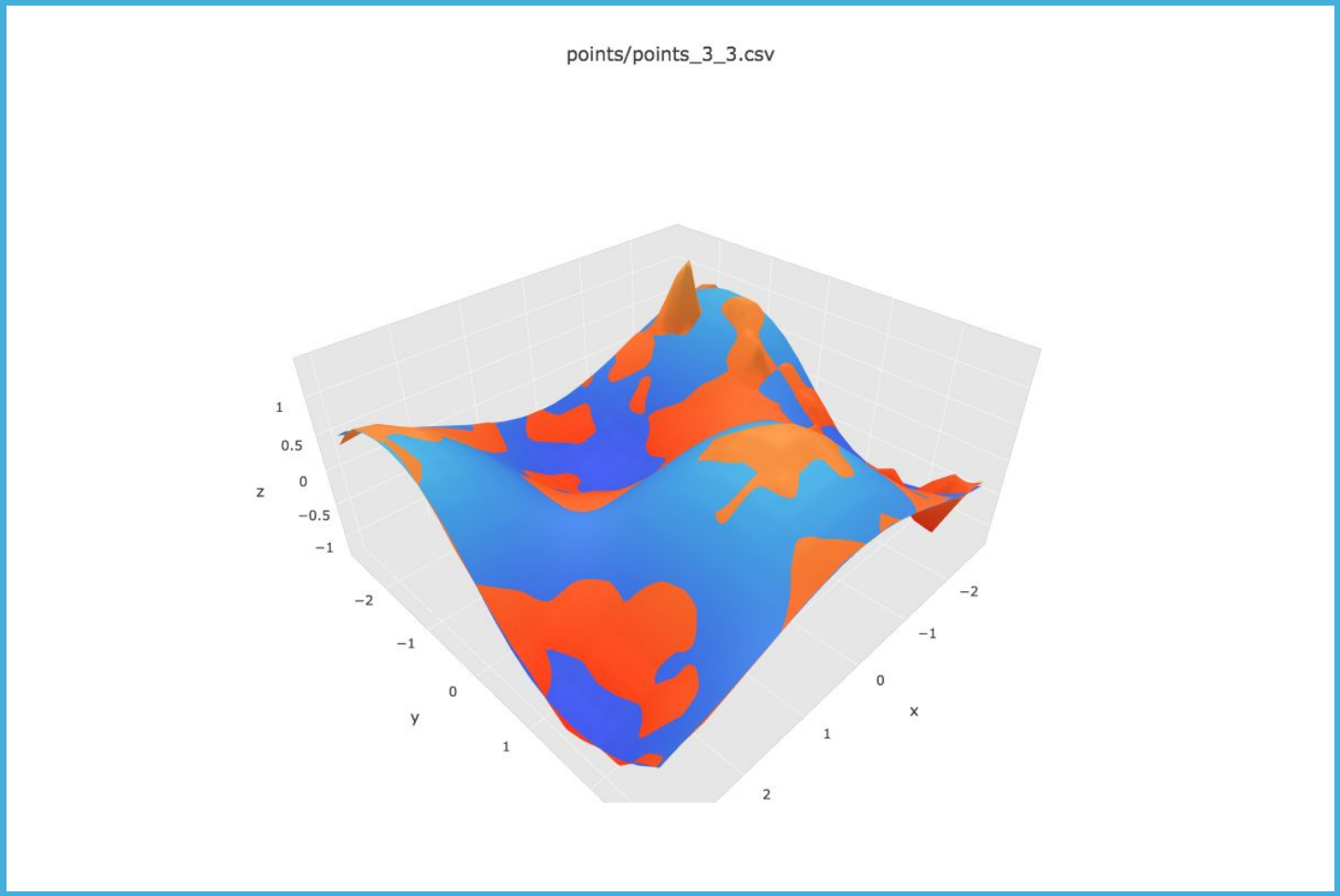
Terrenos arquitectura 2

Datos conocidos Datos obtenidos [10]



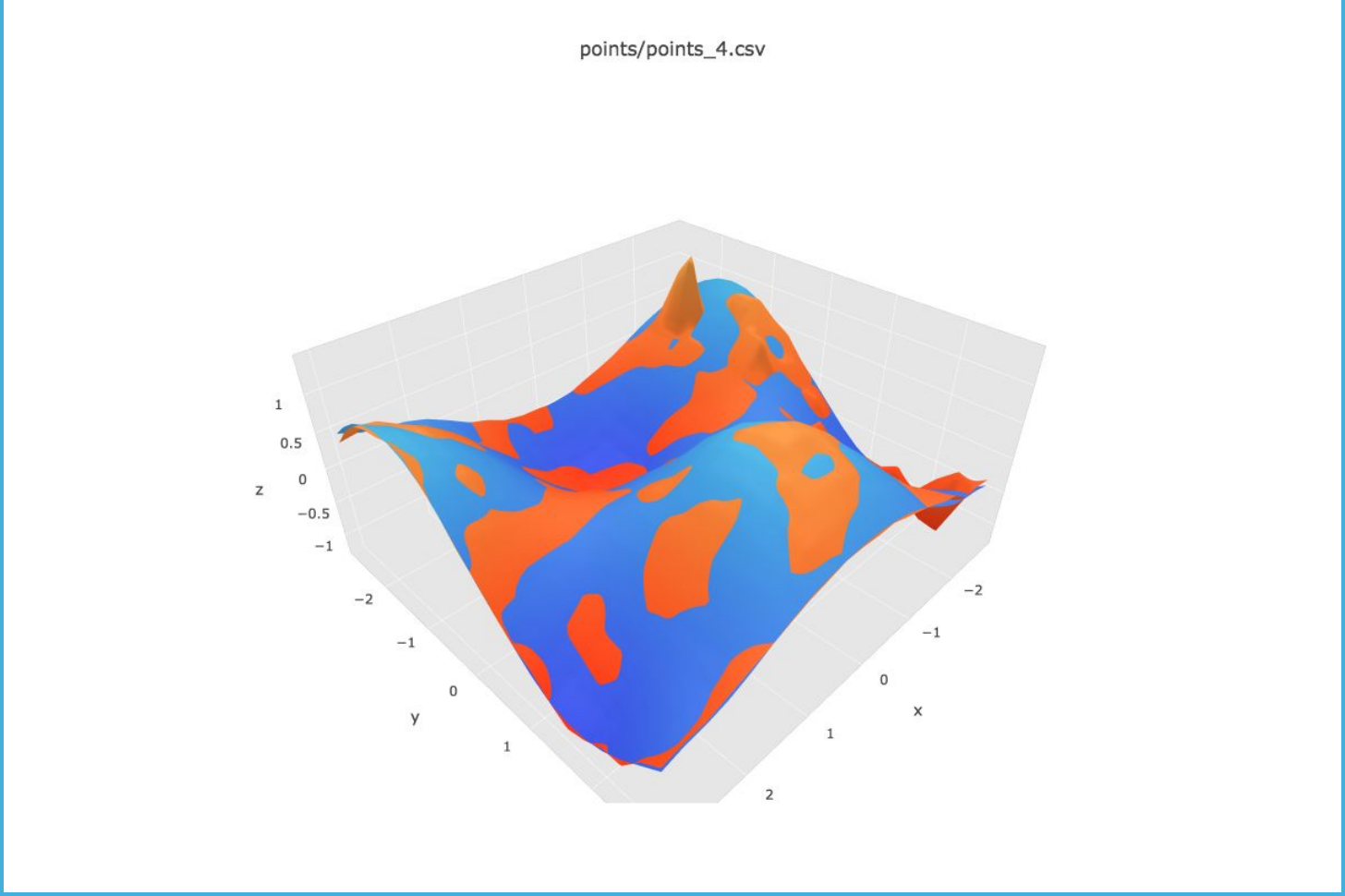
Terrenos arquitectura 3

Datos conocidos Datos obtenidos [30]



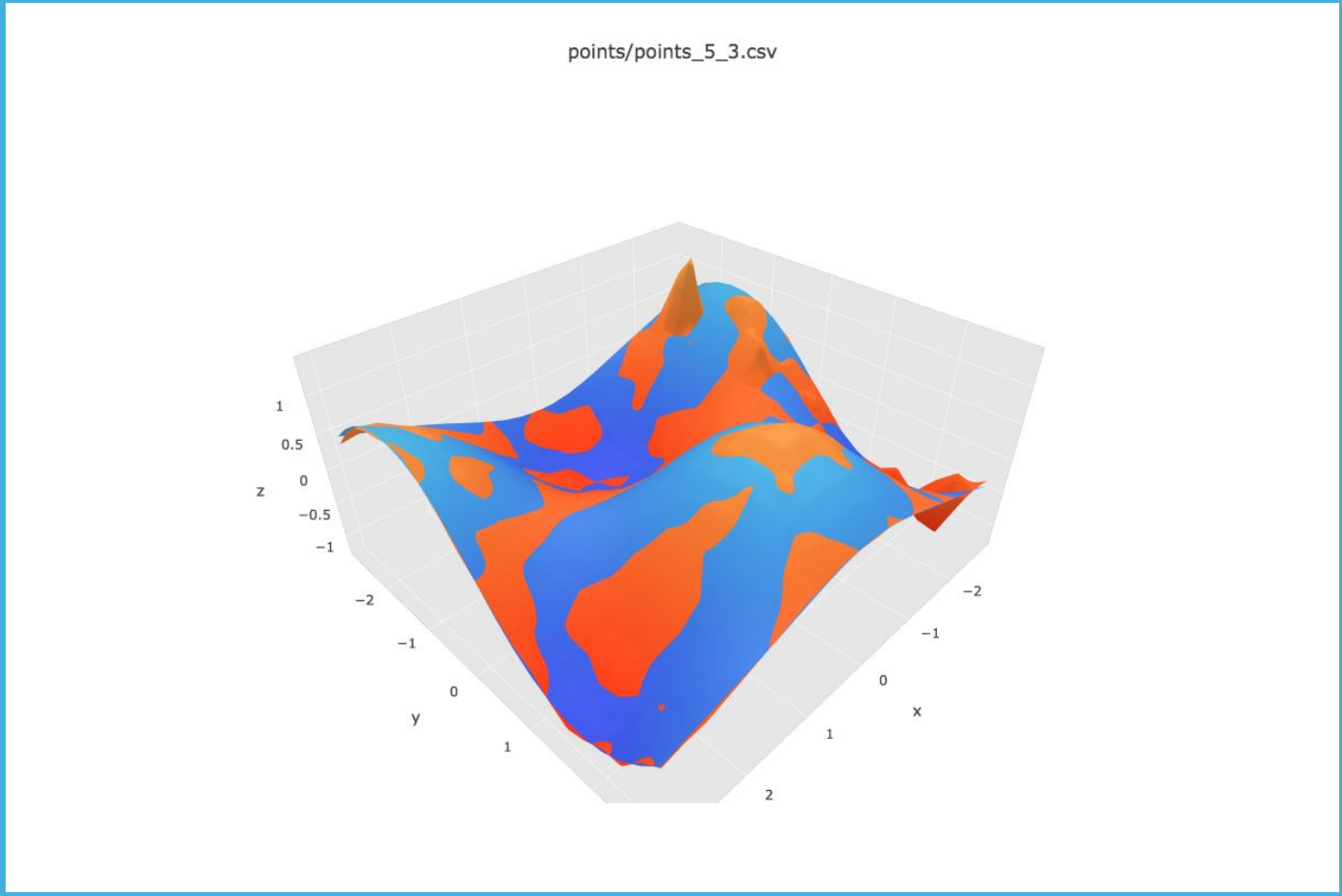
Terrenos arquitectura 4

Datos conocidos Datos obtenidos [10,10]



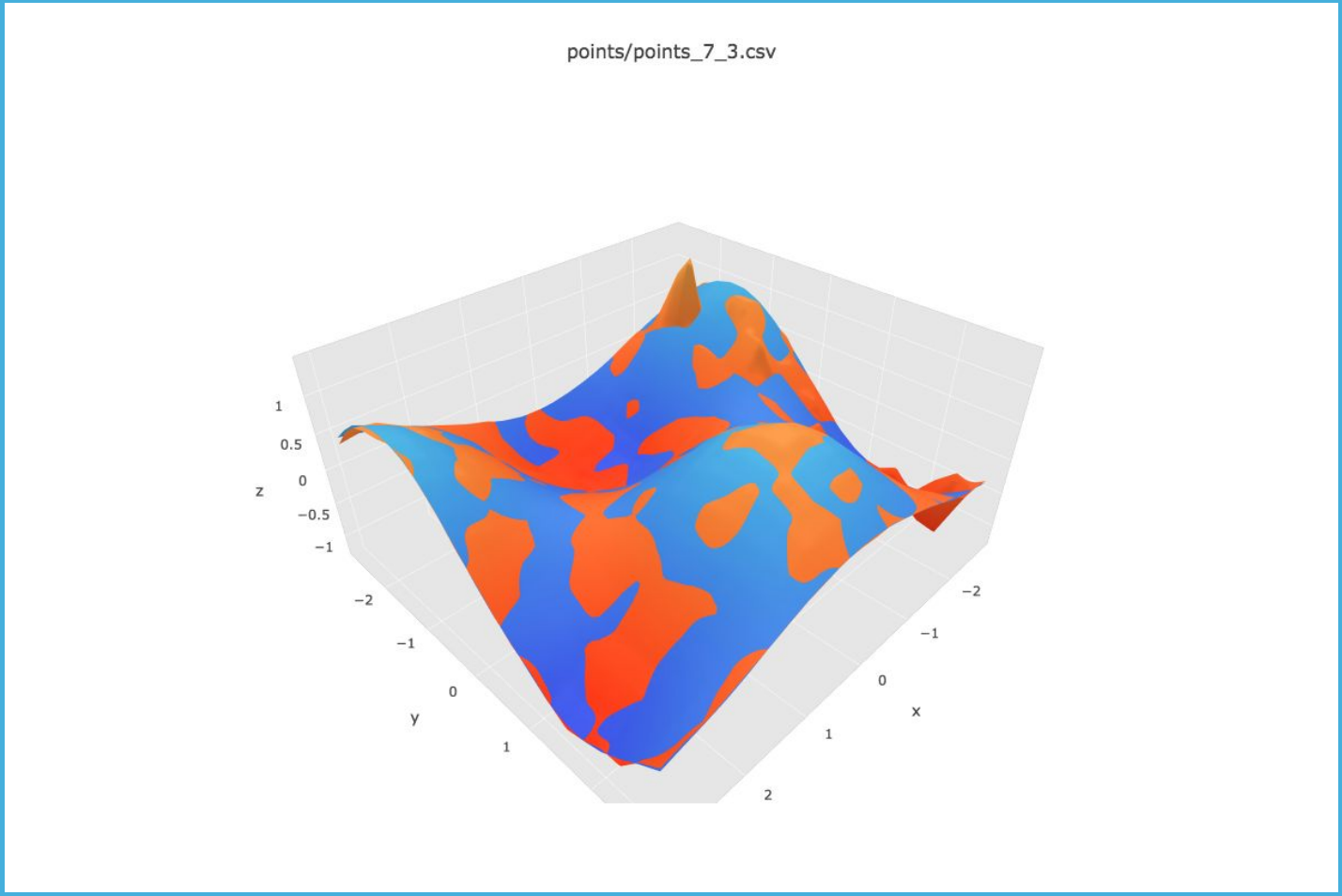
Terrenos arquitectura 5

Datos conocidos Datos obtenidos [30,30]



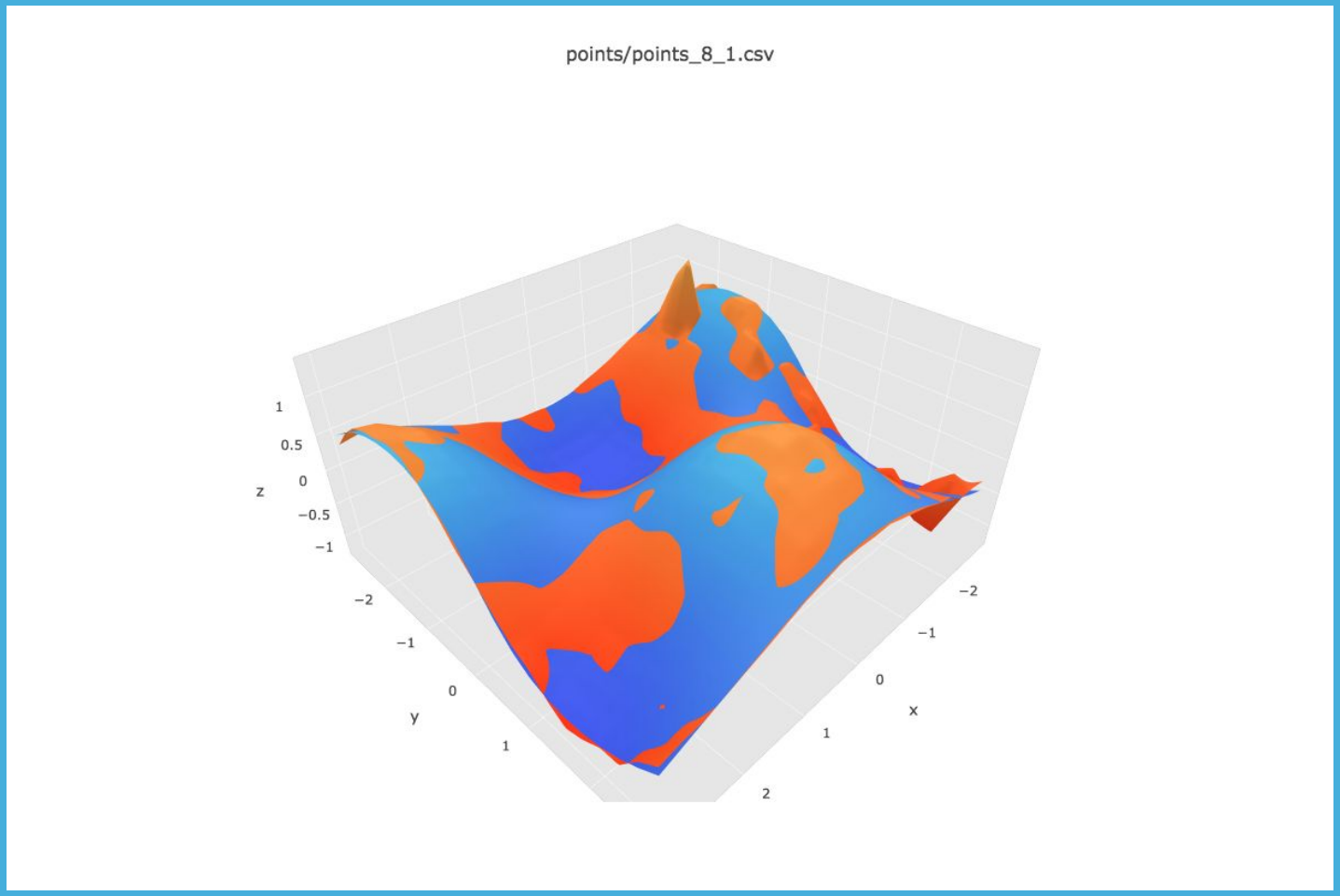
Terrenos arquitectura 6

Datos conocidos Datos obtenidos [20,20,20]



Terrenos arquitectura 7

Datos conocidos Datos obtenidos [10,10,10,10,10]



5.
CONCLUSIONES

- Agregar capas no asegura mejora en el aprendizaje
  - La arquitectura [20, 20, 20] tiene mejores resultados que la arquitectura [10, 10, 10, 10, 10]

- En todos las arquitecturas utilizadas mayor cantidad de neuronas mejoró los resultados

- Conviene usar tanh antes que la función exponencial
  - Permite un rango más amplio de valores de entrada sin saturar
  - Puntos más distinguibles entre sí

- Utilizar etha adaptativo junto con momentum, ayuda a decrecer el error de manera más rápida
  - Sin embargo, el error oscila más

GRACIAS

Preguntas?