

TP3

Perceptrón Simple y Multicapa

Grupo 2

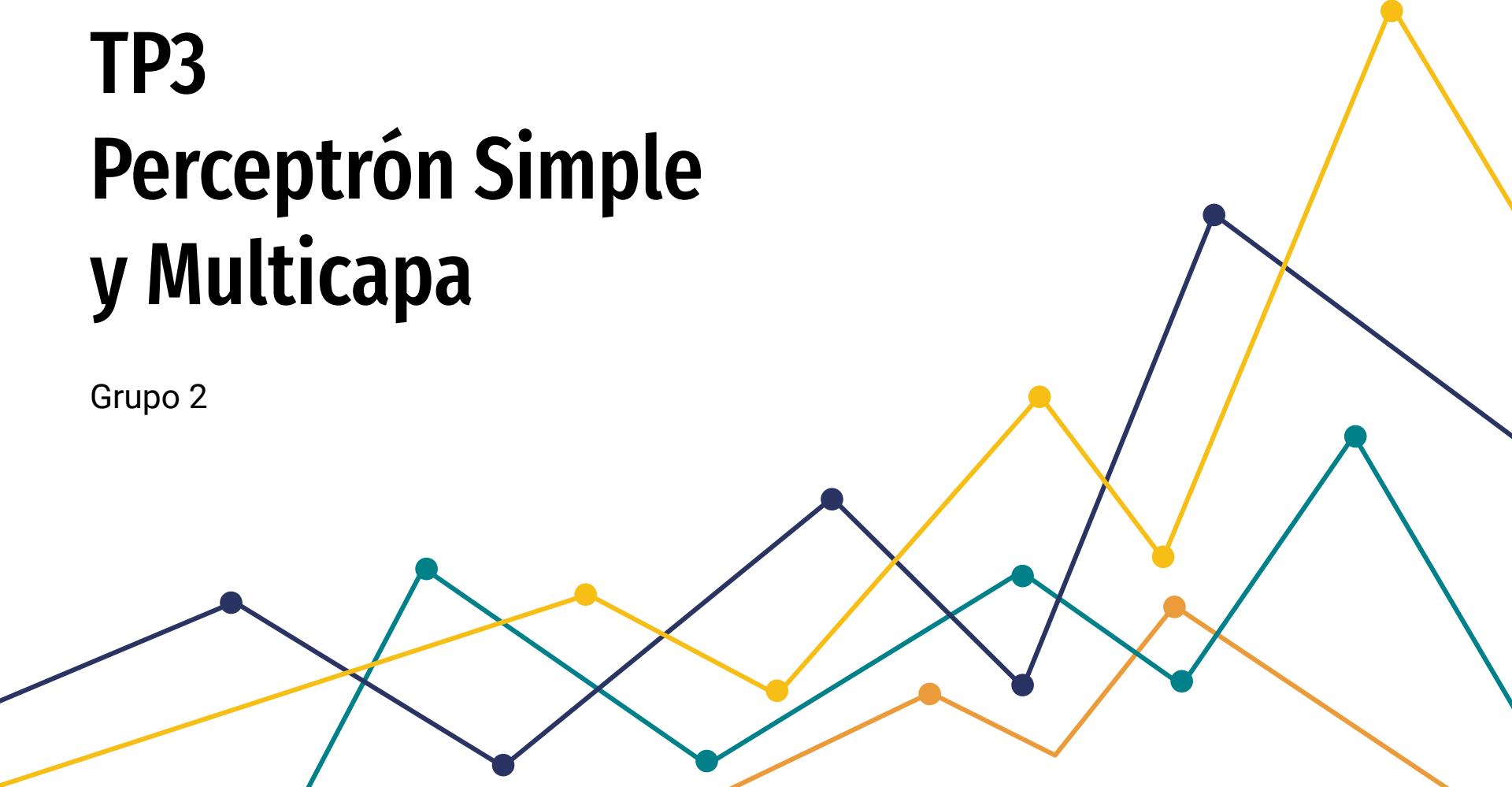


Tabla de contenidos

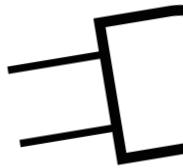
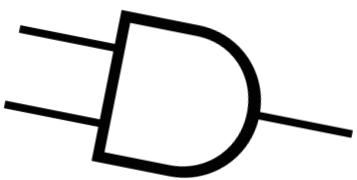
- 1 Ejercicio 1
- 2 Ejercicio 2: Comparación de error
- 3 Ejercicio 2: Comparación de proporciones
- 4 Ejercicio 3: Perceptrón multicapa
- 5 Ejercicio 3: Pares
- 6 Reconocimiento de números

Ejercicio 1

? Preguntas:

- ¿Qué puede decir acerca de los problemas que puede resolver el perceptrón simple escalón en relación a los problemas planteados en la consigna?

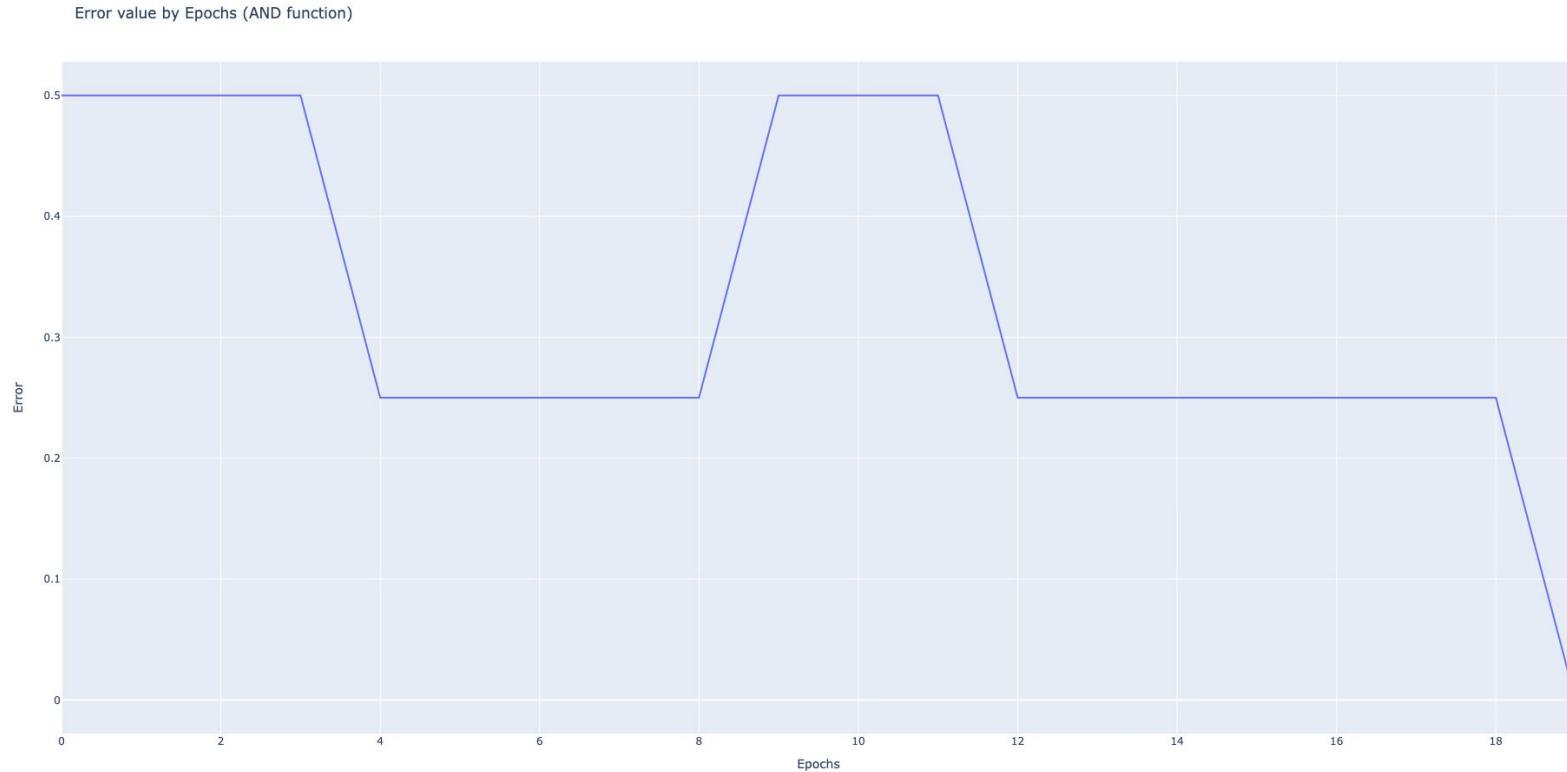
AND



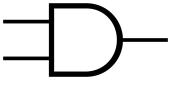
AND



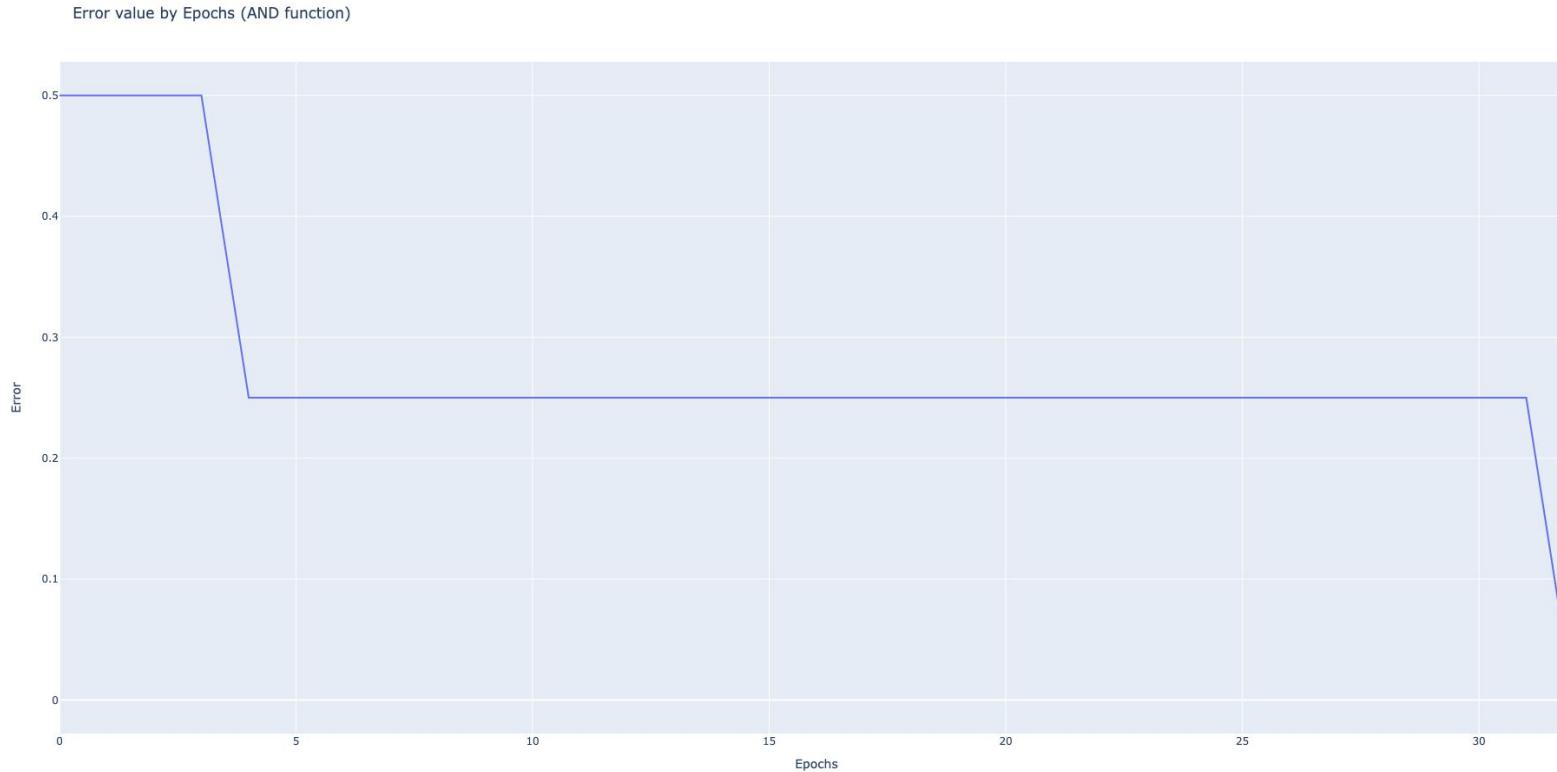
Ejercicio 1



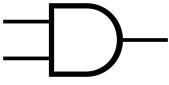
AND



Ejercicio 1



AND



Ejercicio 1

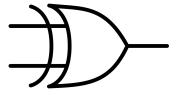




XOR

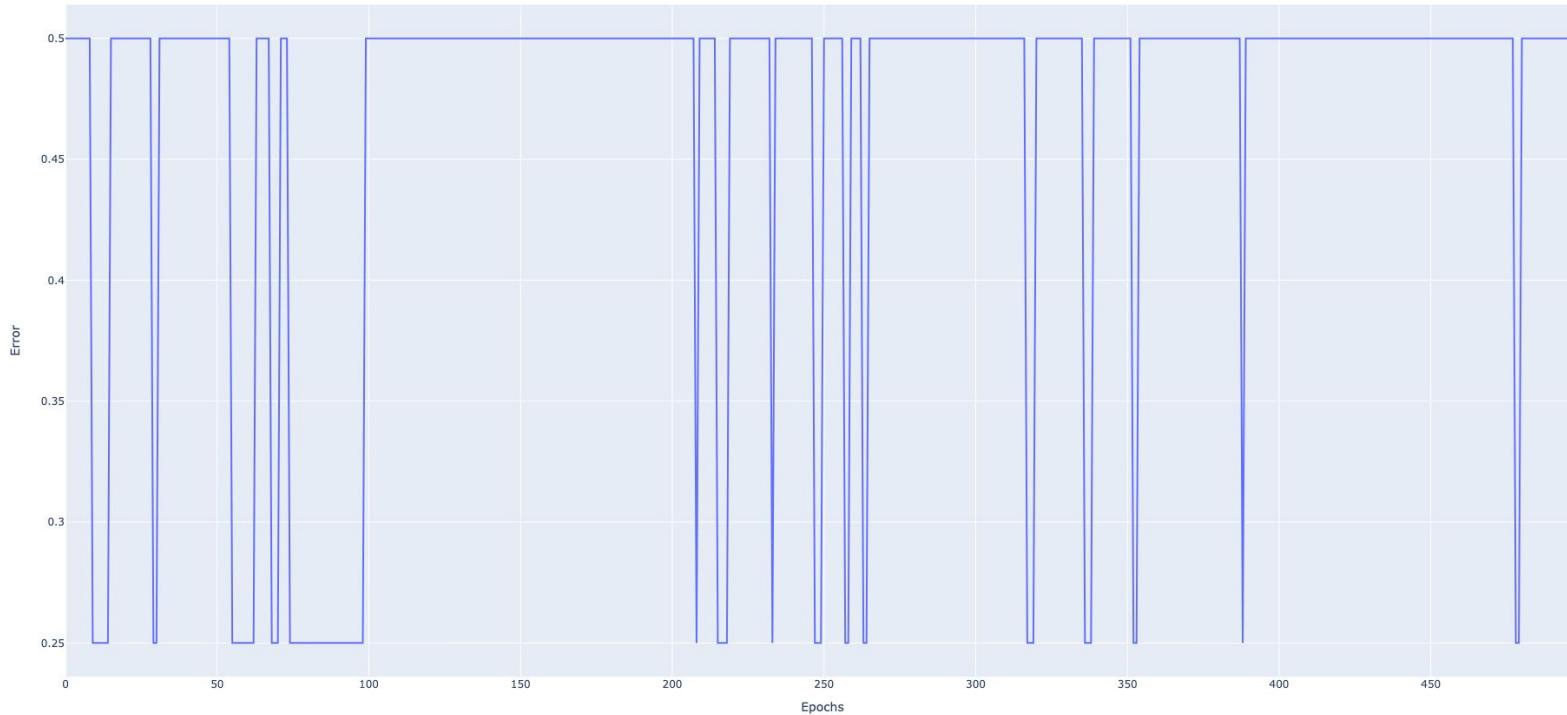


XOR

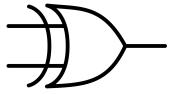


Ejercicio 1

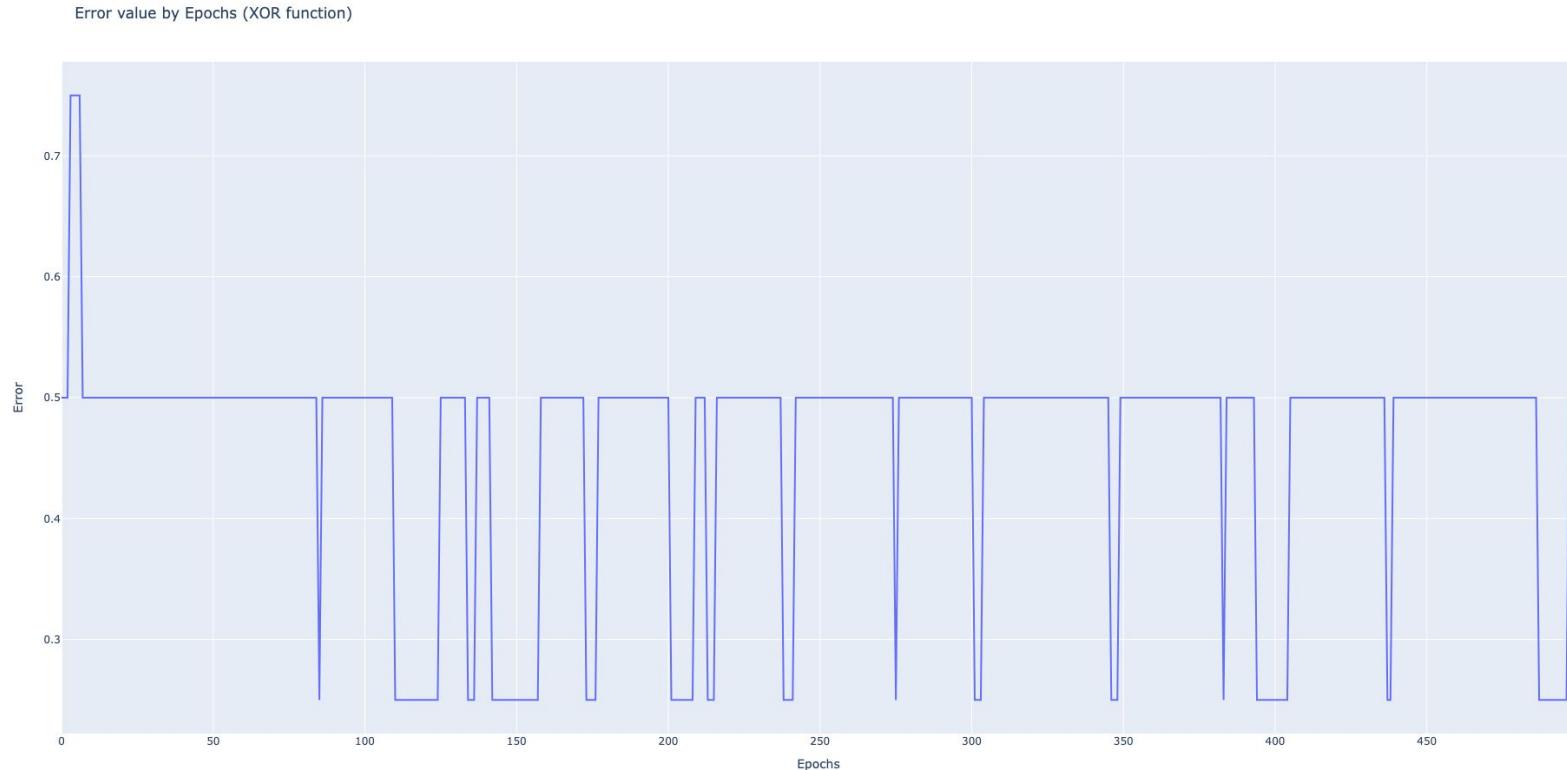
Error value by Epochs (XOR function)



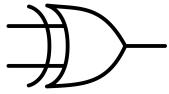
XOR



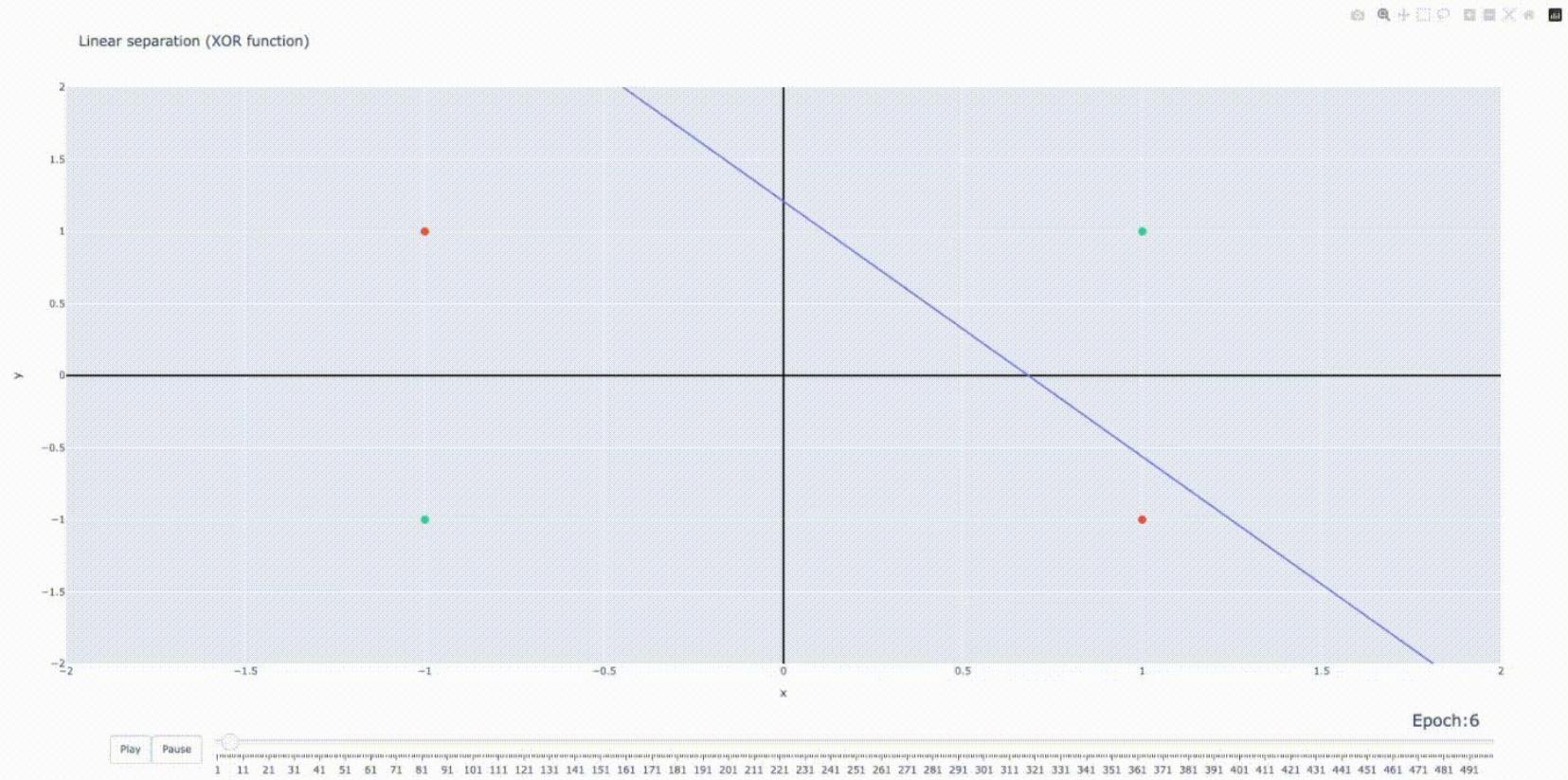
Ejercicio 1



XOR



Ejercicio 1



Ejercicio 2

? Preguntas:

- Evaluar la capacidad de cada uno de los perceptrones para aprender la función cuyas muestras están en los archivos.
- Evaluar la capacidad de generalización del perceptrón simple no lineal utilizando, de los datos provistos, un subconjunto para entrenar y otro para testear.
- ¿Cómo elegirían el mejor conjunto de entrenamiento? ¿Qué efecto tiene la elección en la capacidad de generalización del perceptrón?

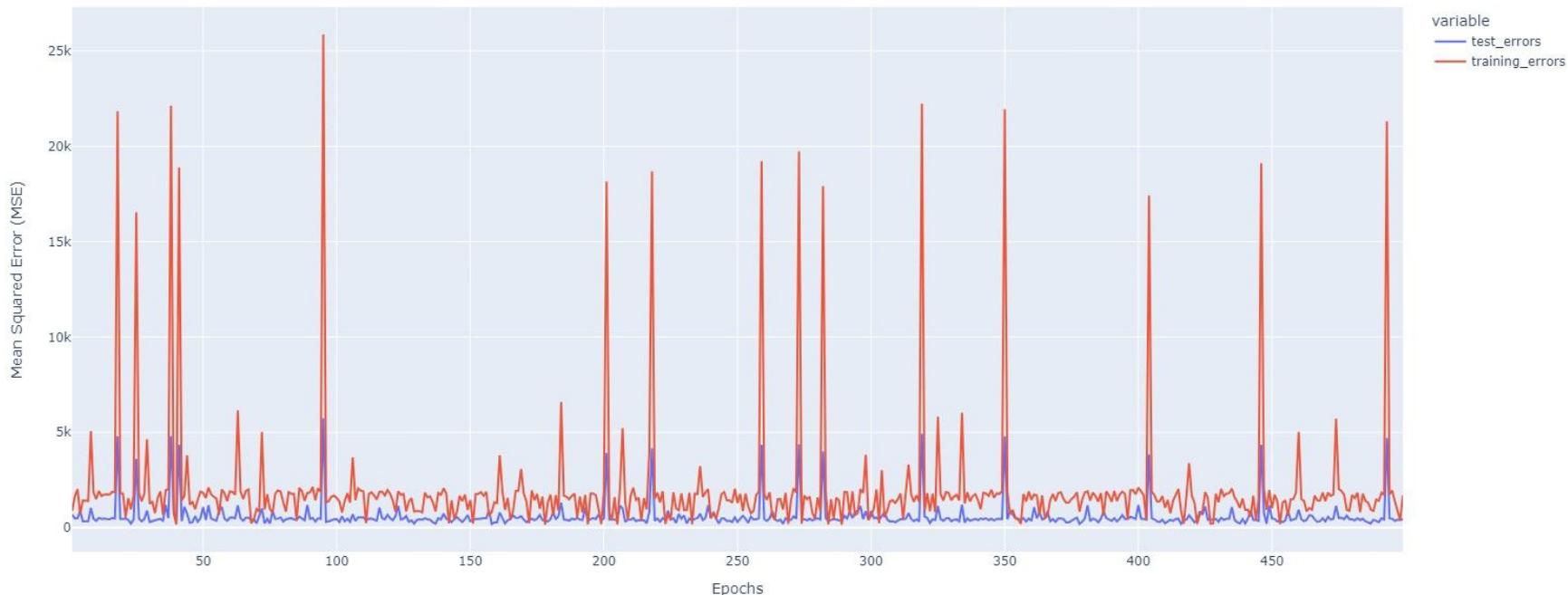
Perceptrones Lineales: Comparación de error entre entrenamiento y prueba

Perceptrón Lineal

Perceptrones Simples: Comparación de errores

```
dataset: TP3-ej2-conjunto.csv
activation_function = IDENTITY
learning_rate = 0.1
beta = 1
split_ratio = 0.8
```

Testing and Training Mean Squared Error Comparison

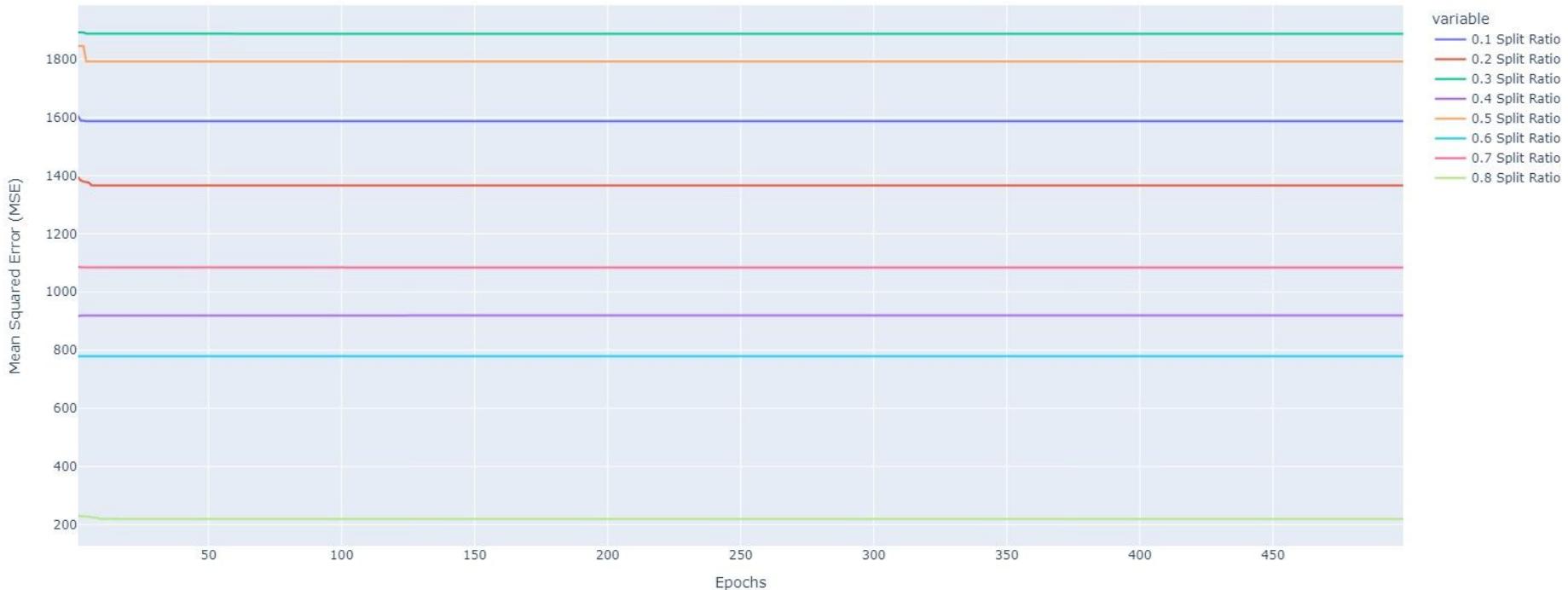


Perceptrón Lineal

Perceptrones Simples: Comparación con distintas proporciones

```
dataset: TP3-ej2-conjunto.csv  
activation_function = IDENTITY  
learning_rate = 0.1  
beta = 1
```

Testing Mean Squared Error with Different Split Ratio Comparison



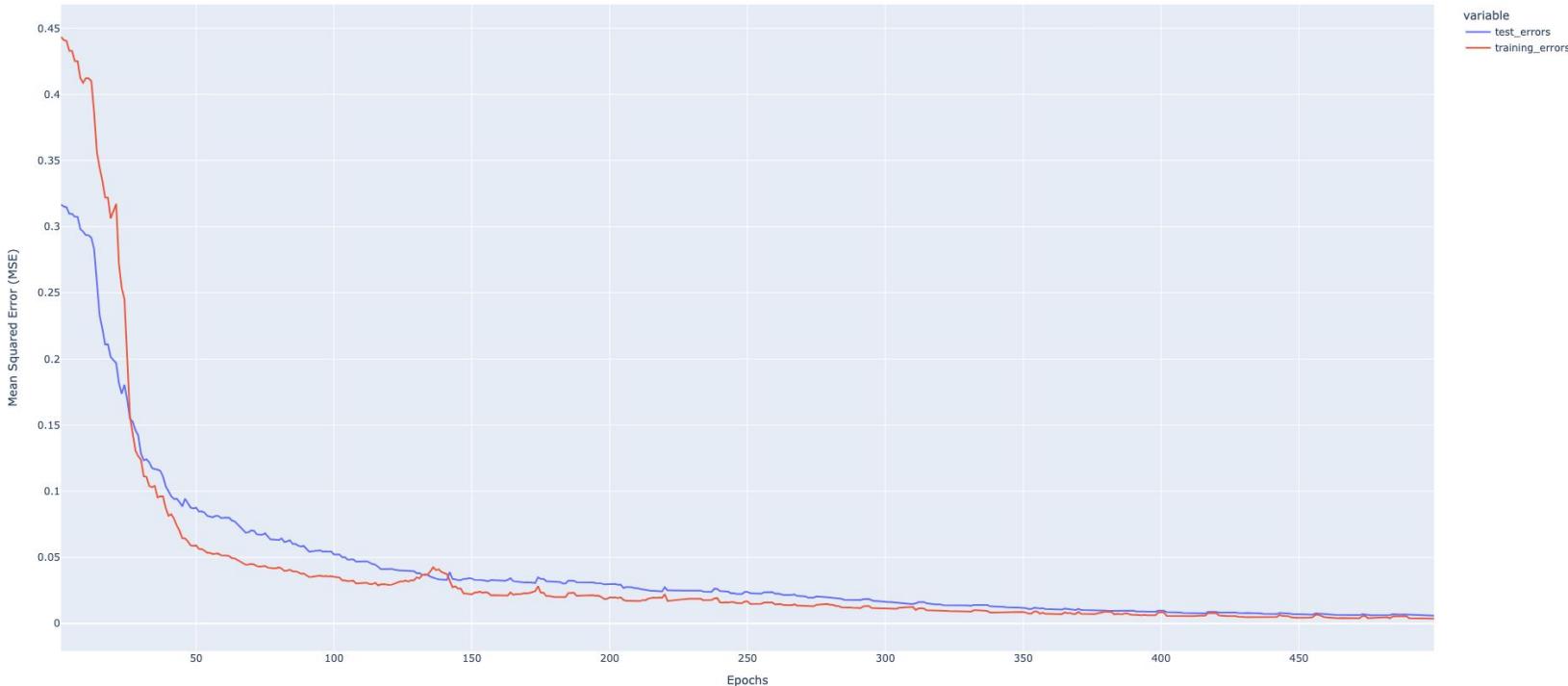
Perceptrones No Lineales: Comparación de error entre entrenamiento y prueba

Perceptrón No Lineal

Perceptrones Simples: Comparación de errores

```
dataset: TP3-ej2-conjunto.csv
activation_function = SIGMOID
learning_rate = 0.1
beta = 1
split_ratio = 0.5
normalized (0, 1)
```

Testing and Training Mean Squared Error Comparison

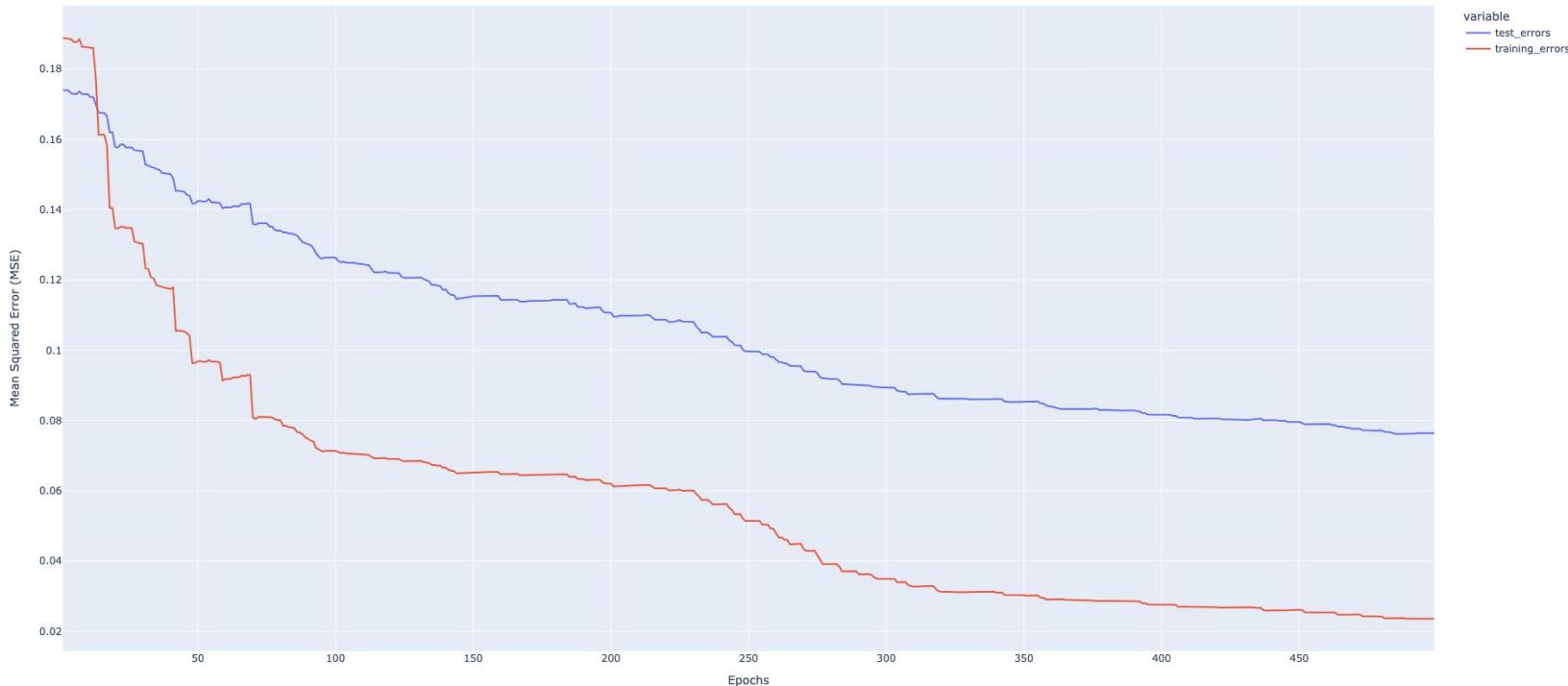


Perceptrón No Lineal

Perceptrones Simples: Comparación de errores

```
dataset: TP3-ej2-conjunto.csv
activation_function = SIGMOID
learning_rate = 0.1
beta = 1
split_ratio = 0.5
normalized (0, 1)
```

Testing and Training Mean Squared Error Comparison

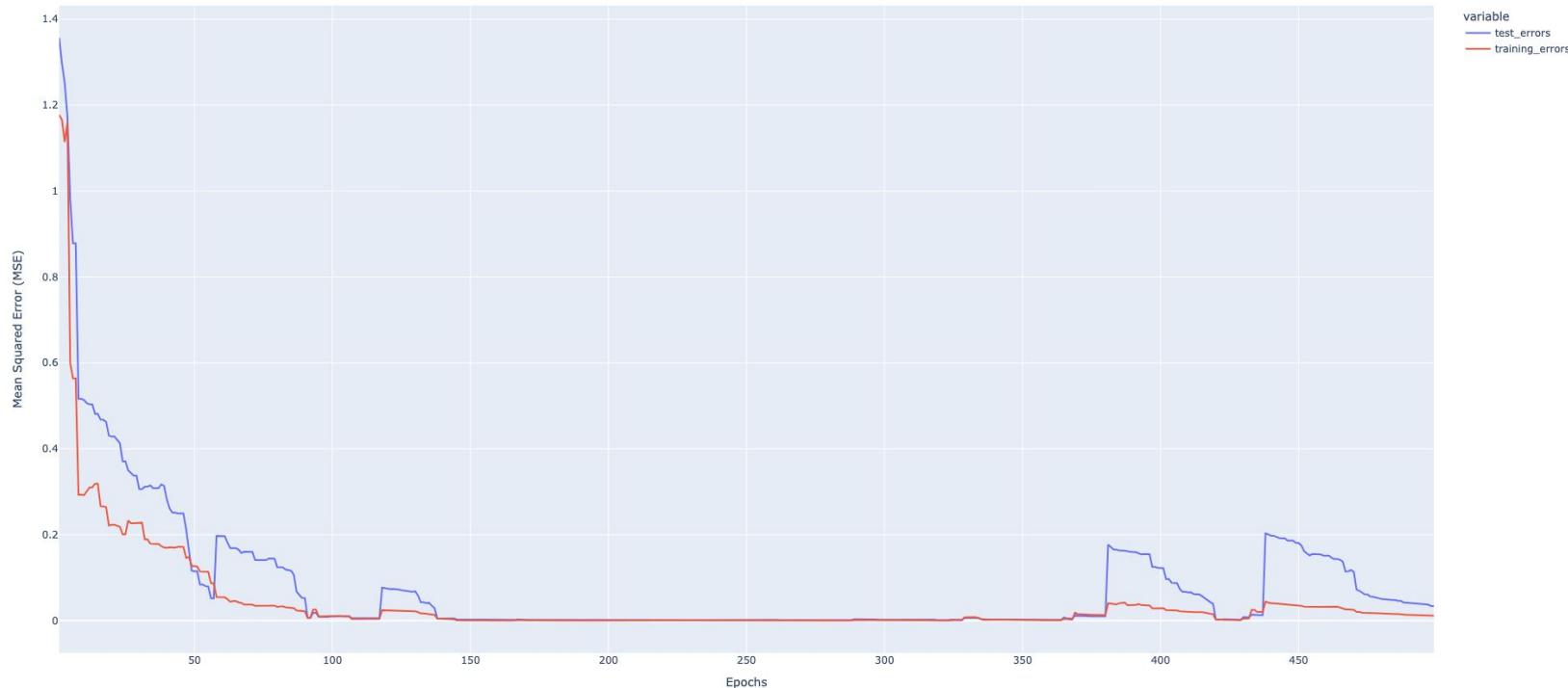


Perceptrón No Lineal

Perceptrones Simples: Comparación de errores

```
dataset: TP3-ej2-conjunto.csv
activation_function = TAN_H
learning_rate = 0.1
beta = 1
split_ratio = 0.5
normalized (-1, 1)
```

Testing and Training Mean Squared Error Comparison

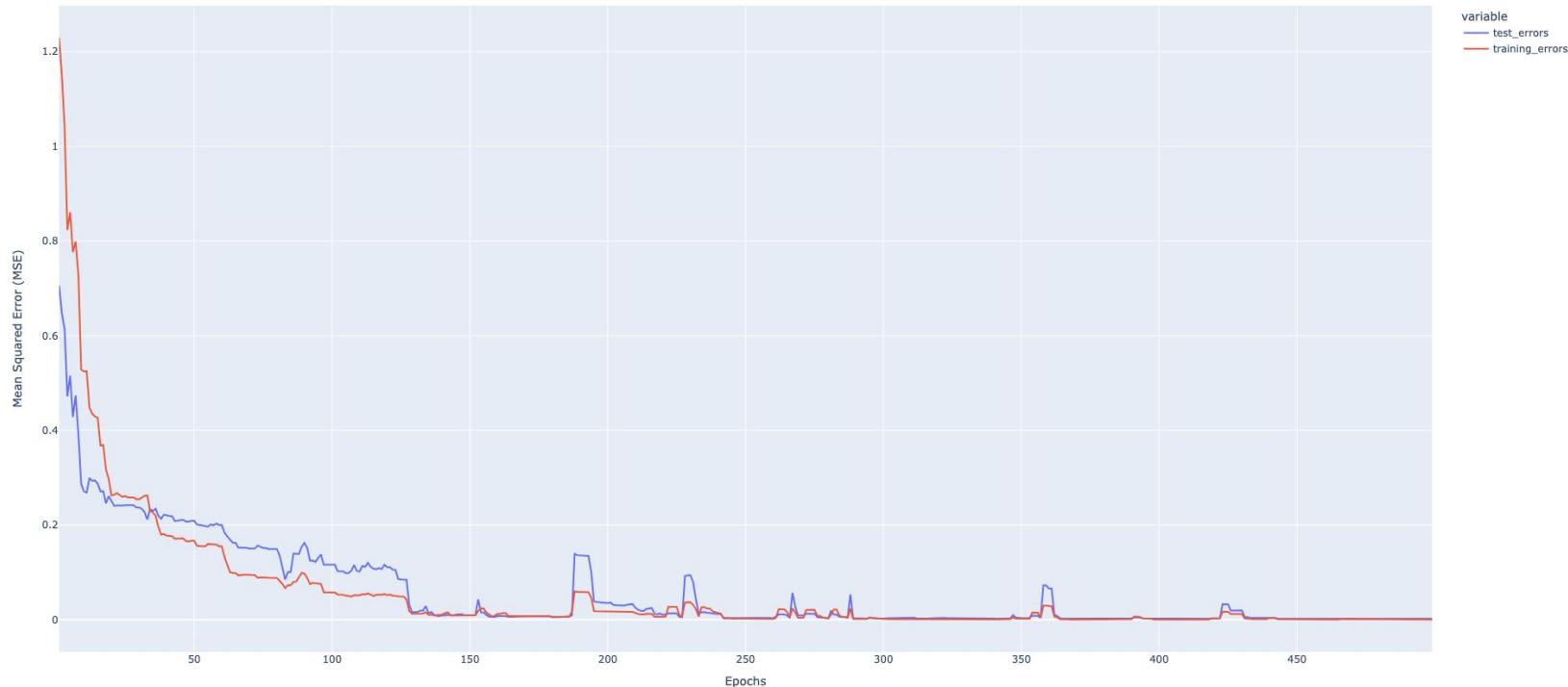


Perceptrón No Lineal

Perceptrones Simples: Comparación de errores

```
dataset: TP3-ej2-conjunto.csv
activation_function = TAN_H
learning_rate = 0.1
beta = 1
split_ratio = 0.5
normalized (-1, 1)
```

Testing and Training Mean Squared Error Comparison



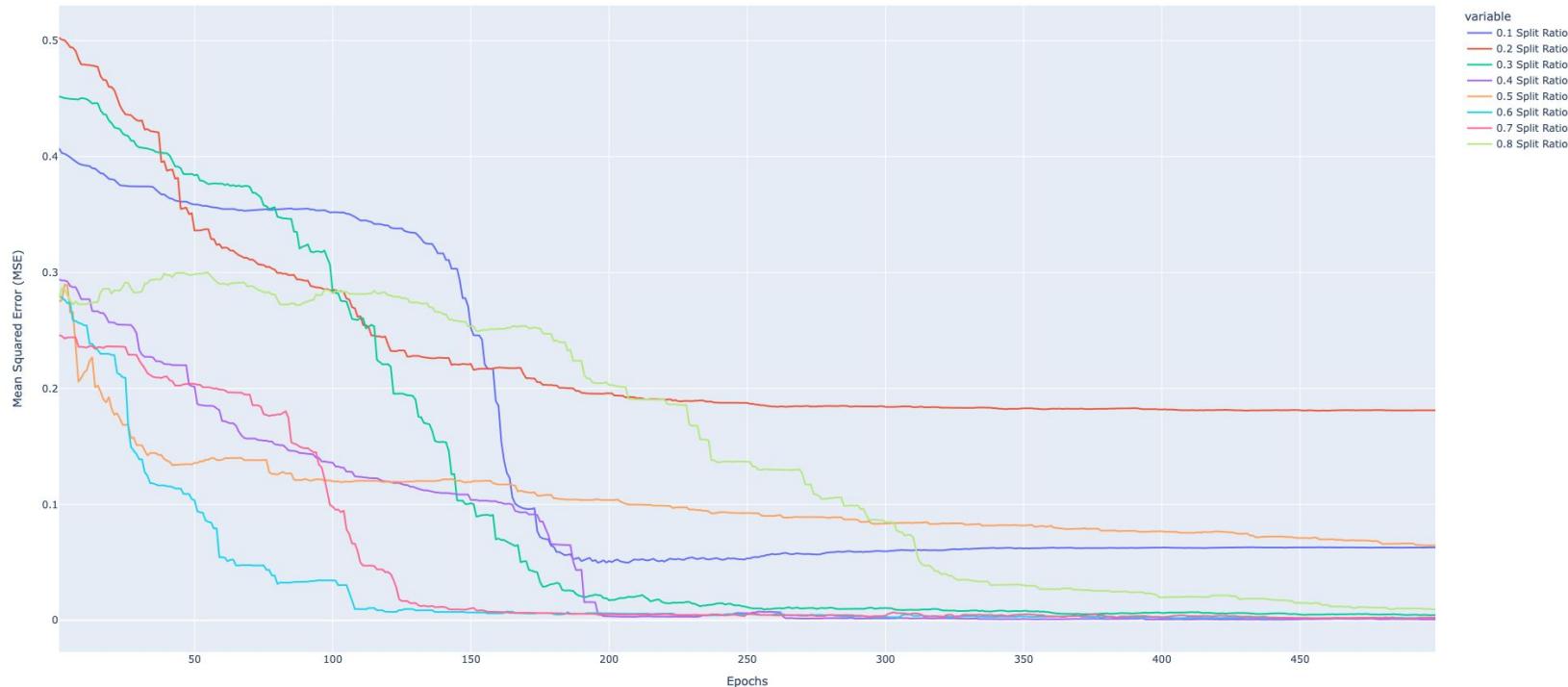
Perceptrones No Lineales: Comparación con distintas proporciones de partición

Perceptrón No Lineal

Perceptrones Simples: Comparación con distintas proporciones

```
dataset: TP3-ej2-conjunto.csv
activation_function = SIGMOID
learning_rate = 0.1
beta = 1
normalized (0, 1)
```

Testing Mean Squared Error with Different Split Ratio Comparison

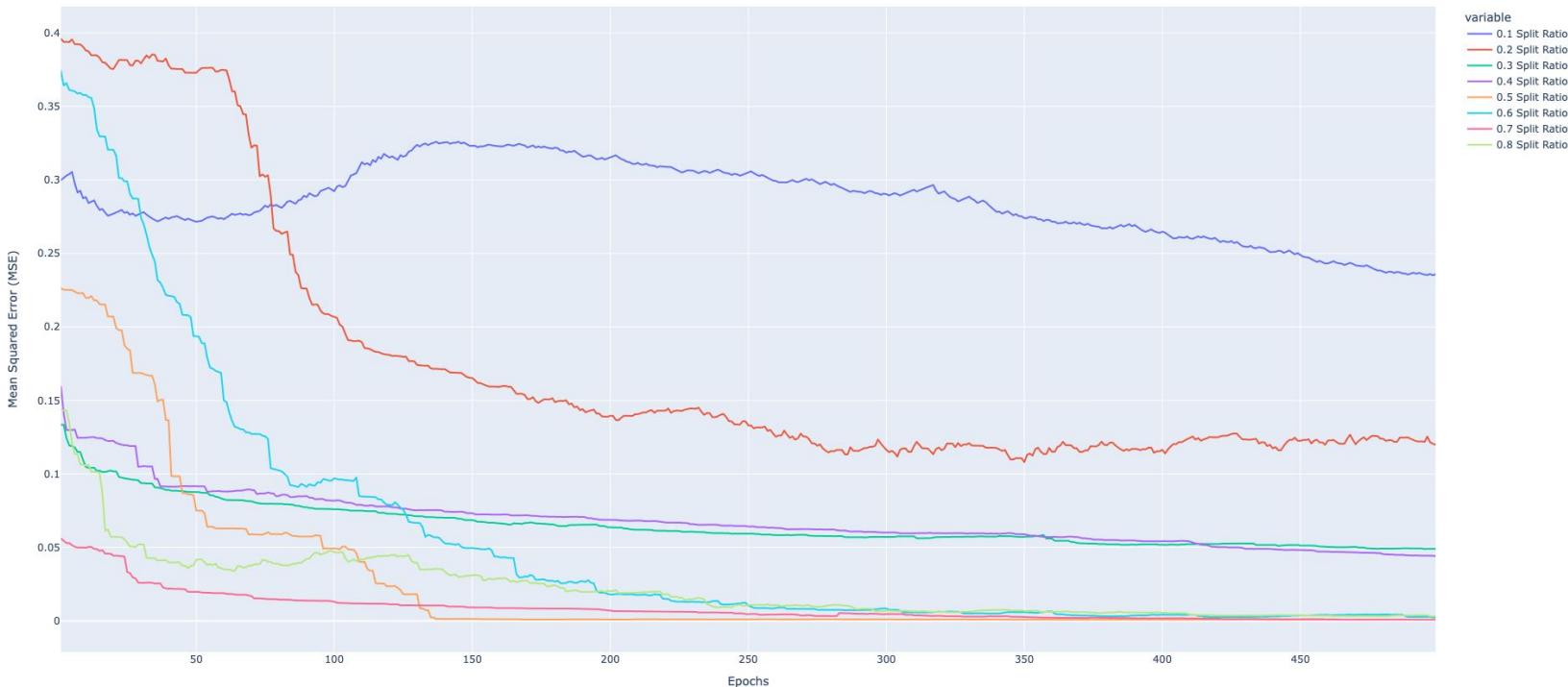


Perceptrón No Lineal

Perceptrones Simples: Comparación con distintas proporciones

```
dataset: TP3-ej2-conjunto.csv
activation_function = SIGMOID
learning_rate = 0.1
beta = 1
normalized (0, 1)
```

Testing Mean Squared Error with Different Split Ratio Comparison



Perceptrón No Lineal

Perceptrones Simples: Comparación con distintas proporciones

```
dataset: TP3-ej2-conjunto.csv
activation_function = SIGMOID
learning_rate = 0.1
beta = 1
normalized (0, 1)
```

Testing Mean Squared Error with Different Split Ratio Comparison

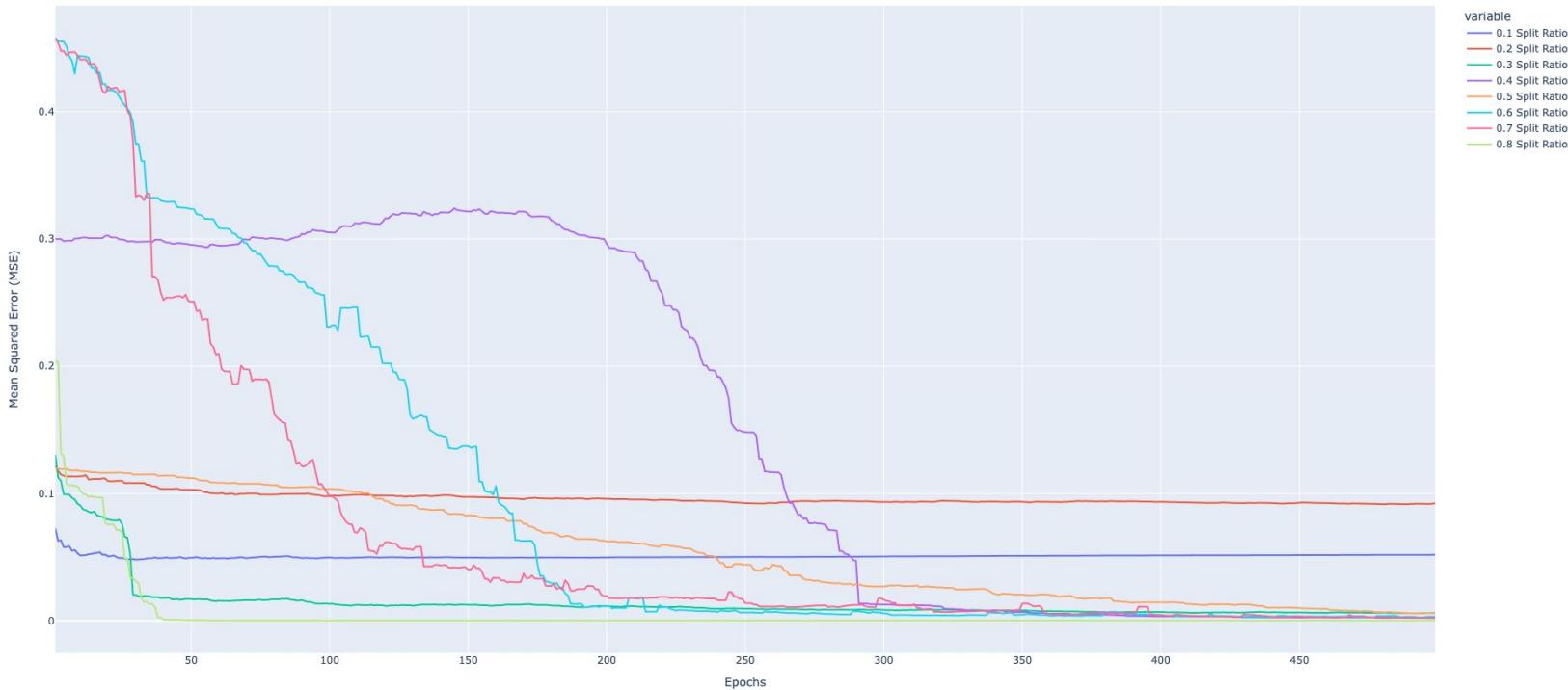


Perceptrón No Lineal

Perceptrones Simples: Comparación con distintas proporciones

```
dataset: TP3-ej2-conjunto.csv
activation_function = SIGMOID
learning_rate = 0.1
beta = 1
normalized (0, 1)
```

Testing Mean Squared Error with Different Split Ratio Comparison

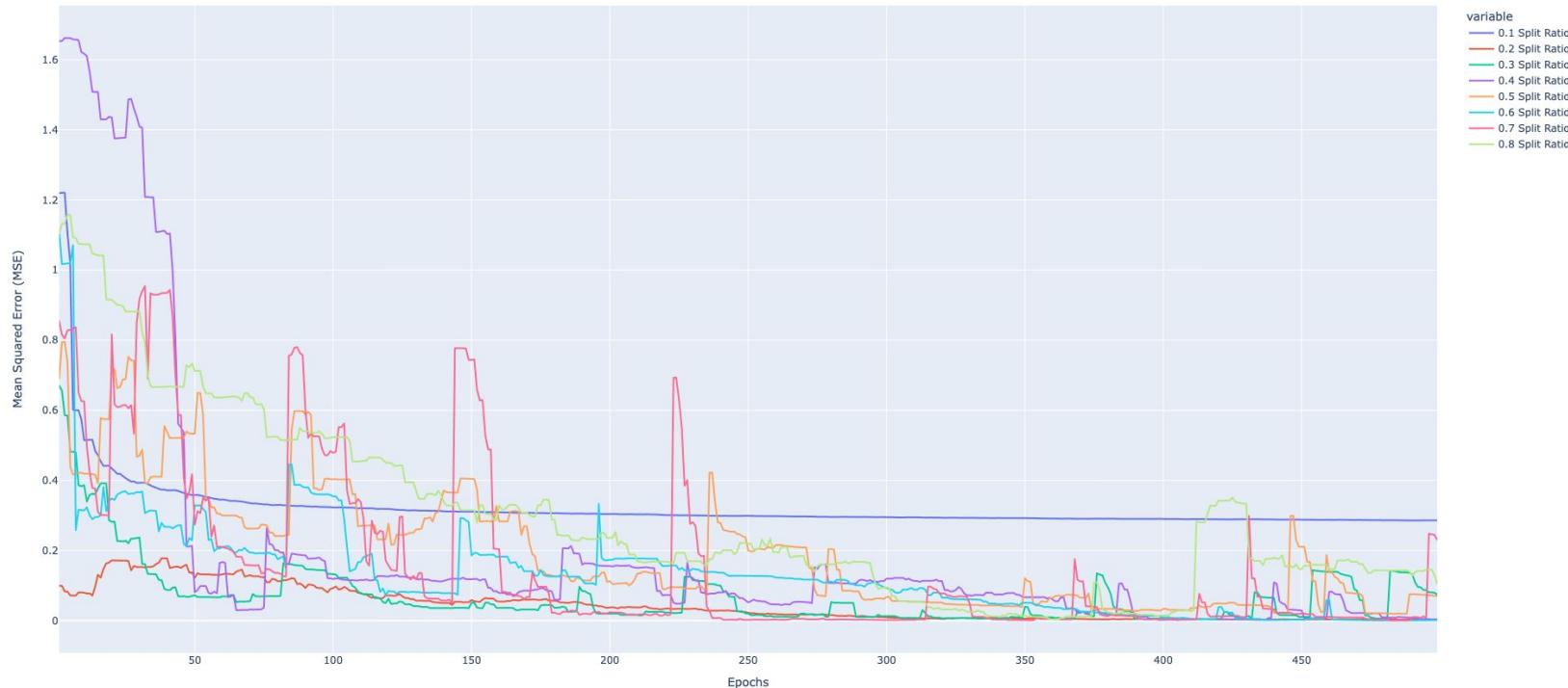


Perceptrón No Lineal

Perceptrones Simples: Comparación con distintas proporciones

```
dataset: TP3-ej2-conjunto.csv
activation_function = TAN_H
learning_rate = 0.1
beta = 1
normalized (-1, 1)
```

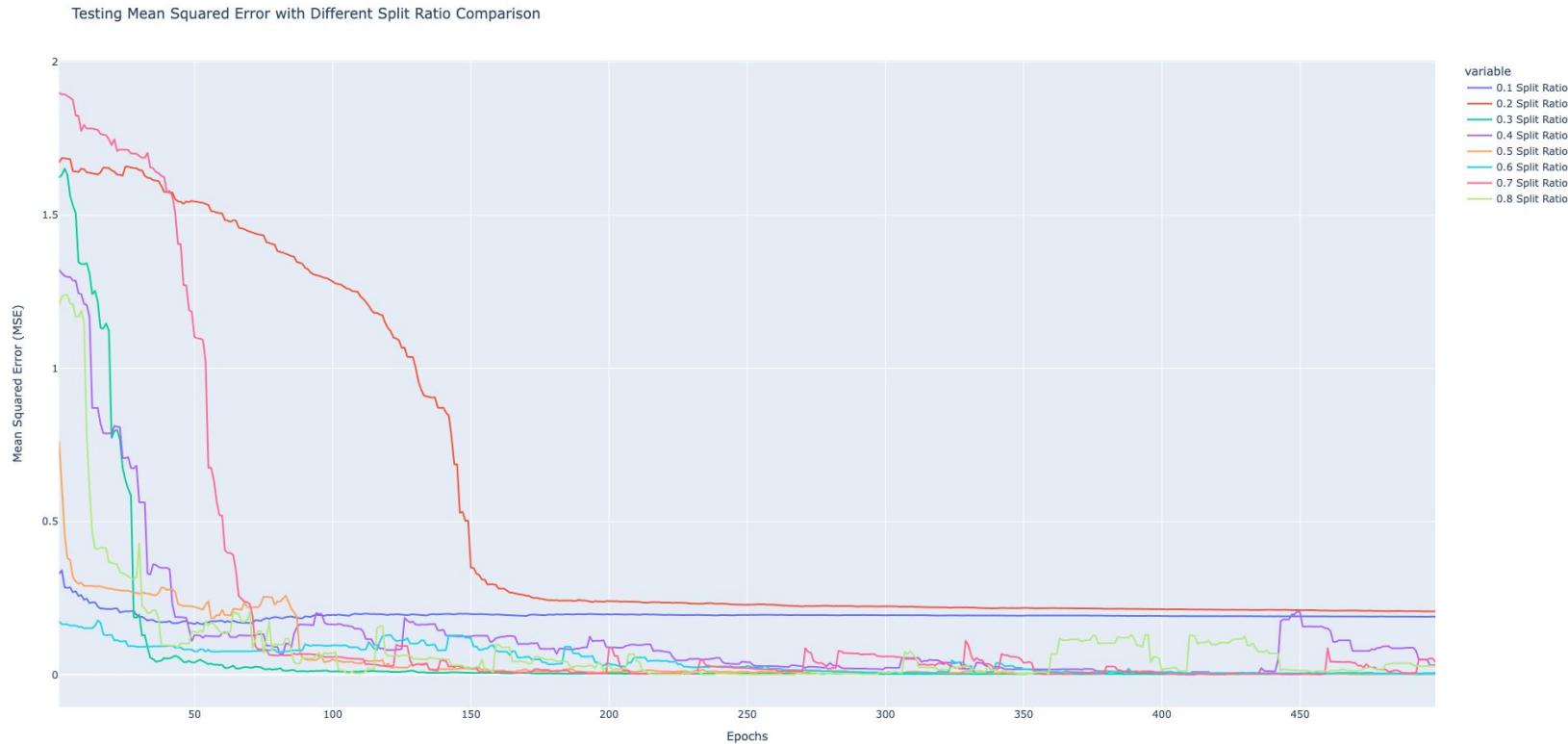
Testing Mean Squared Error with Different Split Ratio Comparison



Perceptrón No Lineal

Perceptrones Simples: Comparación con distintas proporciones

```
dataset: TP3-ej2-conjunto.csv
activation_function = TAN_H
learning_rate = 0.1
beta = 1
normalized (-1, 1)
```



Conclusión

En el caso del conjunto de datos provisto, se pudo ver que resulta mucho mejor utilizar un perceptrón no lineal por sobre uno lineal. Pudimos ver que en el caso del perceptrón lineal el MSE resulta sumamente inestable, con muchos picos, y no tiene una tendencia a decrecer. Esto no sucede en el perceptrón no lineal, cuya función MSE (tanto cuando se utiliza la función sigmoidea como la hiperbólica) tiene semblanza logarítmica a medida que avanza las épocas. Asimismo, en el caso del perceptrón no lineal, resulta mejor utilizar la función tangente hiperbólica antes que la sigmoidea, pues podemos ver su error decrece muy rápidamente a comparación de la sigmoidea. Además, podemos ver que en el caso de la sigmoidea hay overfitting pues el training error da significativamente mejor que el testing error, indicándonos que le está costando generalizar lo aprendido; esto no sucede en cuando se hace uso de la tangente hiperbólica.

Ejercicio 3

? Preguntas:

- ¿Qué problemas puede resolver el perceptrón multicapa que no puede resolver el perceptrón simple?
- ¿Cómo influye el ruido del conjunto de entrenamiento en los resultados finales?

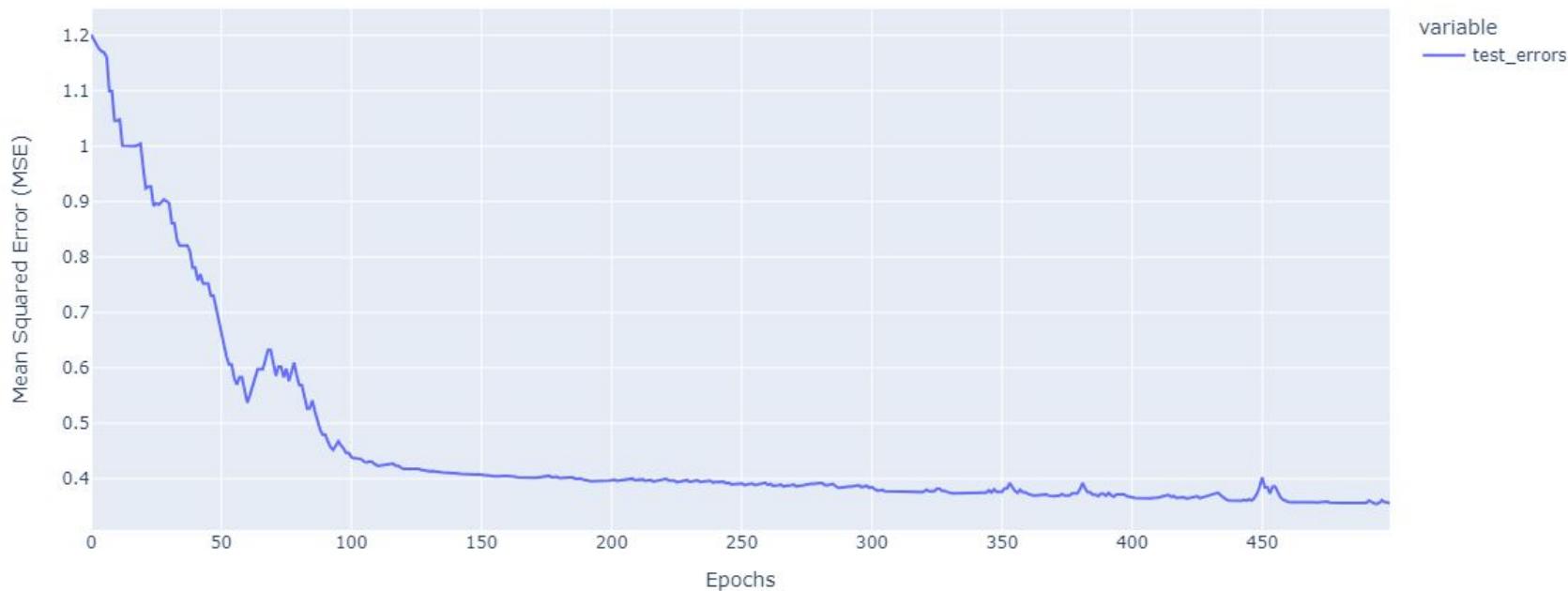
Perceptrón Multicapa: Ejercicio 1

Perceptrón Multicapa: XOR

Ejercicio 1

Testing and Training Mean Squared Error Comparison

```
dataset: XOR
activation_function = TAN_H
learning_rate = 0.1
beta = 1
NArchitecture = [2,2,2,1]
```



Perceptrón Multicapa: Pares

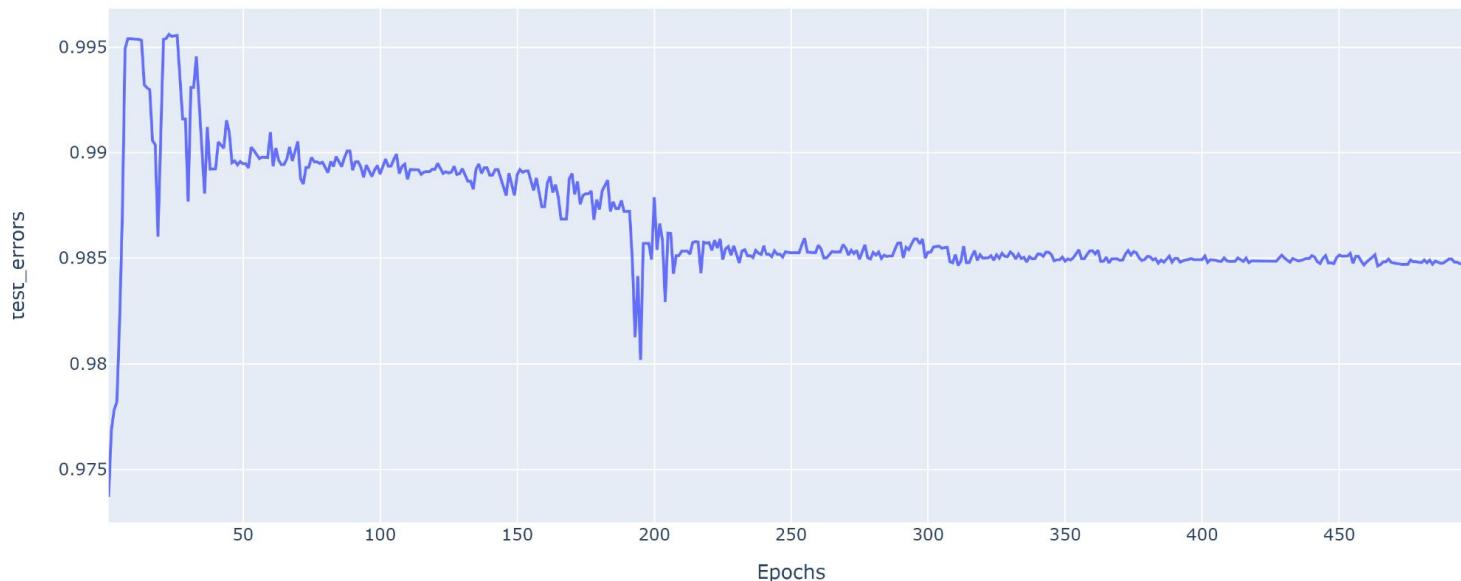
Perceptrón Multicapa: Pares (sin ampliar el dataset)

Ejercicio 2

Mean Squared Error Comparison

Testing Mean Squared Error

```
dataset: TP3-ej3-digitos.txt  
activation_function = TAN_H  
learning_rate = 0.1  
beta = 1  
NArchitecture = [35,2,2,1]
```

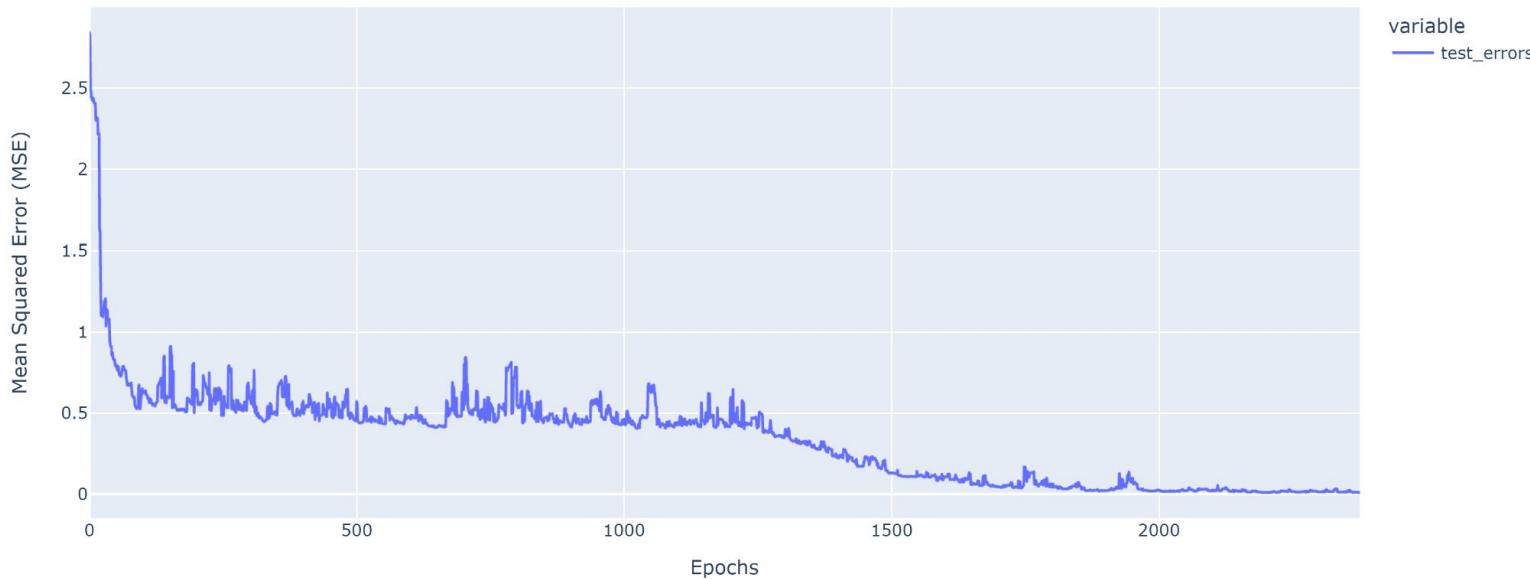


Perceptrón Multicapa: Pares (ampliando el dataset)

Ejercicio 2

Testing Mean Squared Error Comparison

```
dataset: TP3-ej3-digitos.txt
activation_function = TAN_H
learning_rate = 0.1
beta = 1
NArchitecture = [35,2,2,1]
```

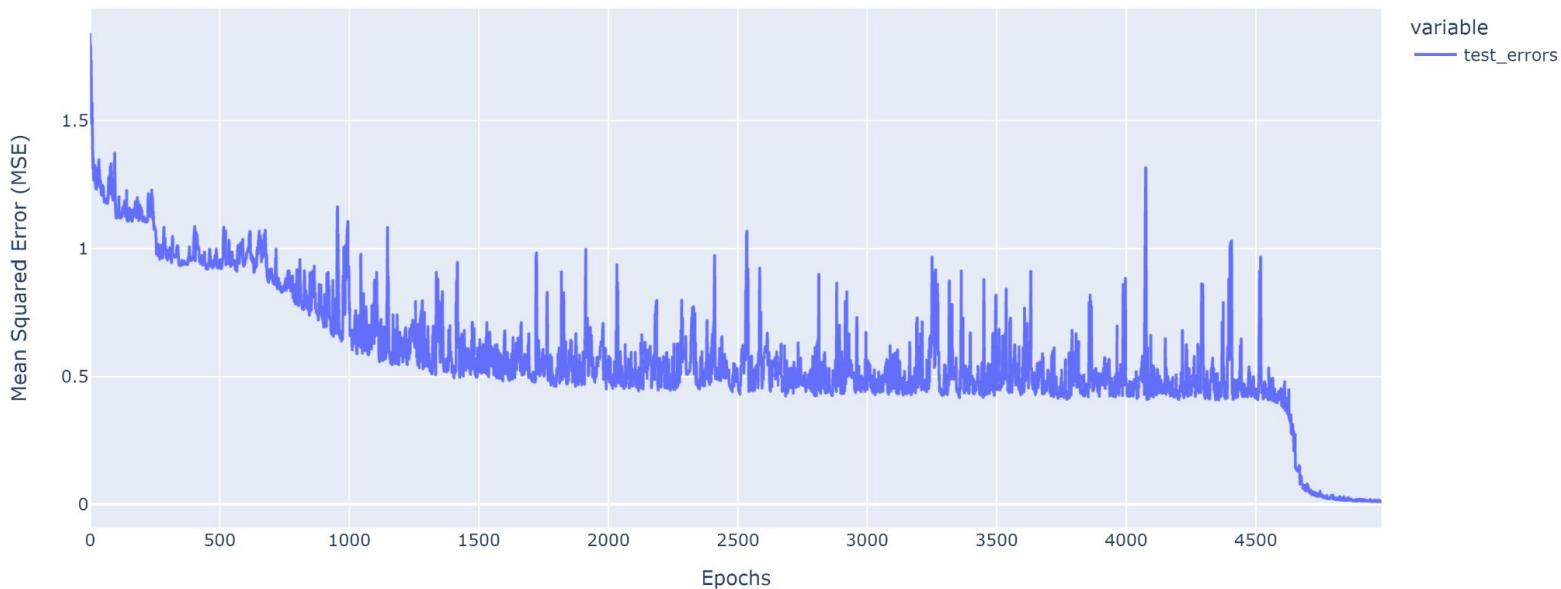


Perceptrón Multicapa: Pares (ampliando el dataset)

Ejercicio 2 (con más capas)

Testing Mean Squared Error Comparison

```
dataset: TP3-ej3-digitos.txt
activation_function = TAN_H
learning_rate = 0.1
beta = 1
NArchitecture = [35,2,2,2,2,2,2,1]
```



Perceptrón Multicapa: Reconocimiento de números

Perceptrón Multicapa: Reconocimiento de números

Si no expandimos el dataset...

Perceptrón Multicapa: Reconocimiento de números

Accuracy (Sin expandir el dataset)

```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
split_ratio = 0.8 (80%
training, 20% testing)
noise = 0
architecture = [35, 10, 10]
```

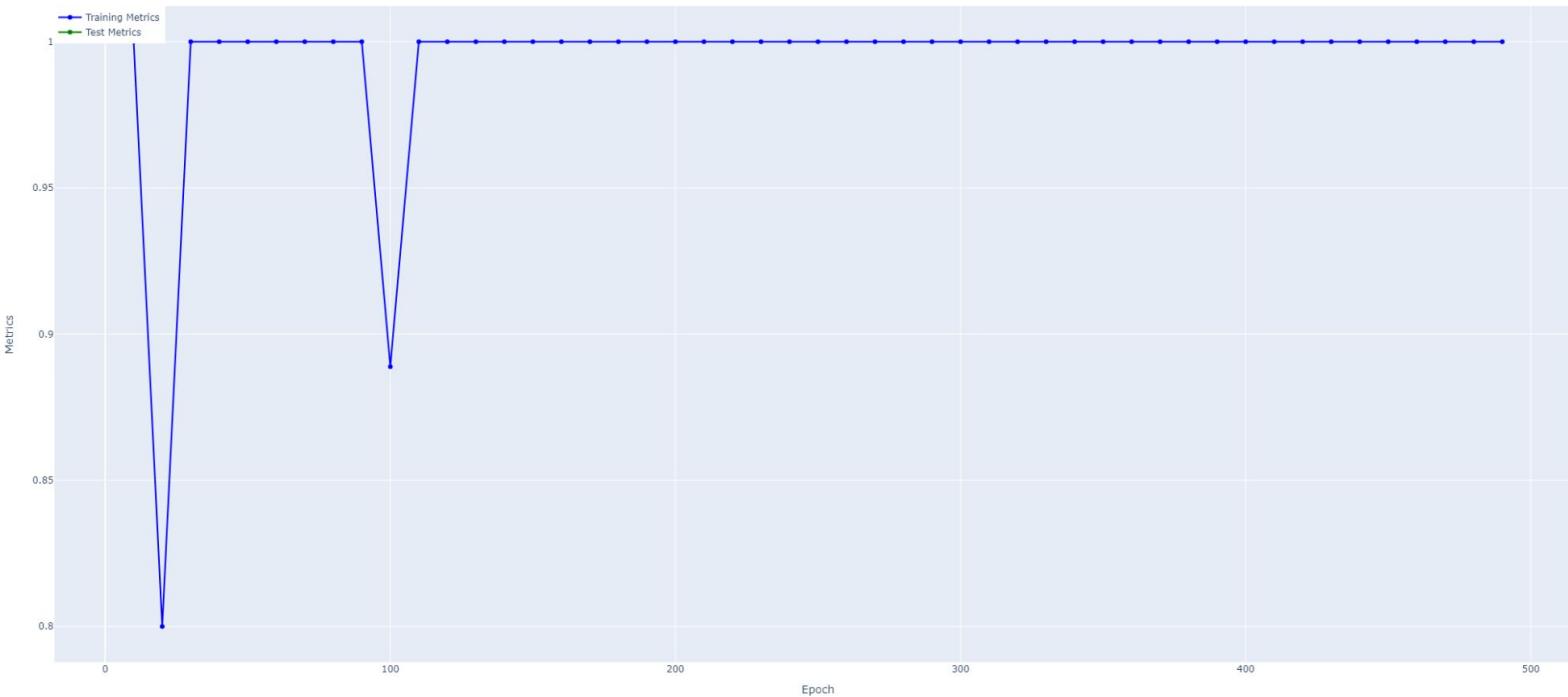


Perceptrón Multicapa: Reconocimiento de números

F1 (Sin expandir el dataset)

```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
split_ratio = 0.8 (80%
training, 20% testing)
noise = 0
architecture = [35, 10, 10]
```

Training and Test Metrics Over Epochs (F1)



Perceptrón Multicapa: Reconocimiento de números

Precision (Sin expandir el dataset)

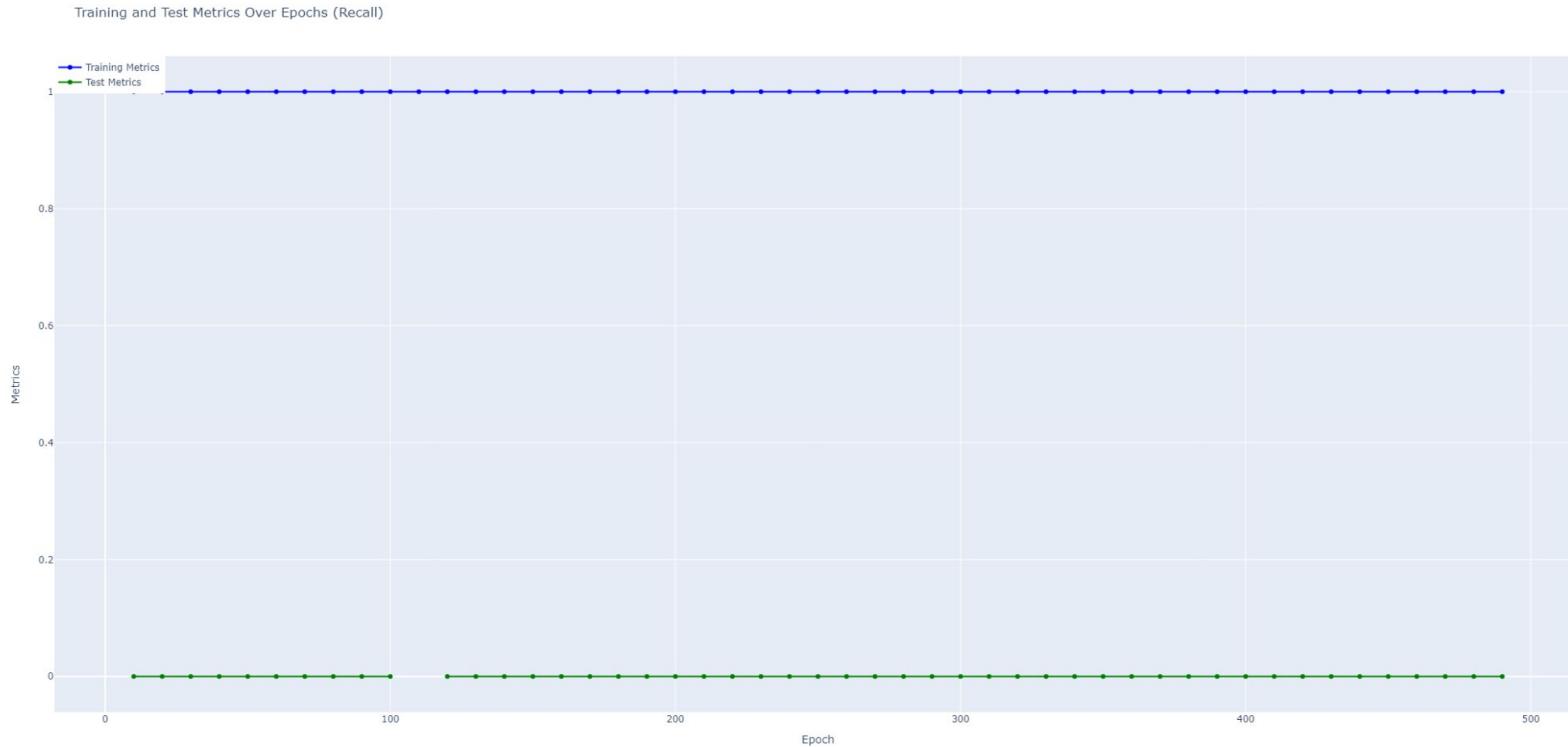
```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
split_ratio = 0.8 (80%
training, 20% testing)
noise = 0
architecture = [35, 10, 10]
```



Perceptrón Multicapa: Reconocimiento de números

Recall (Sin expandir el dataset)

```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
split_ratio = 0.8 (80%
training, 20% testing)
noise = 0
architecture = [35, 10, 10]
```



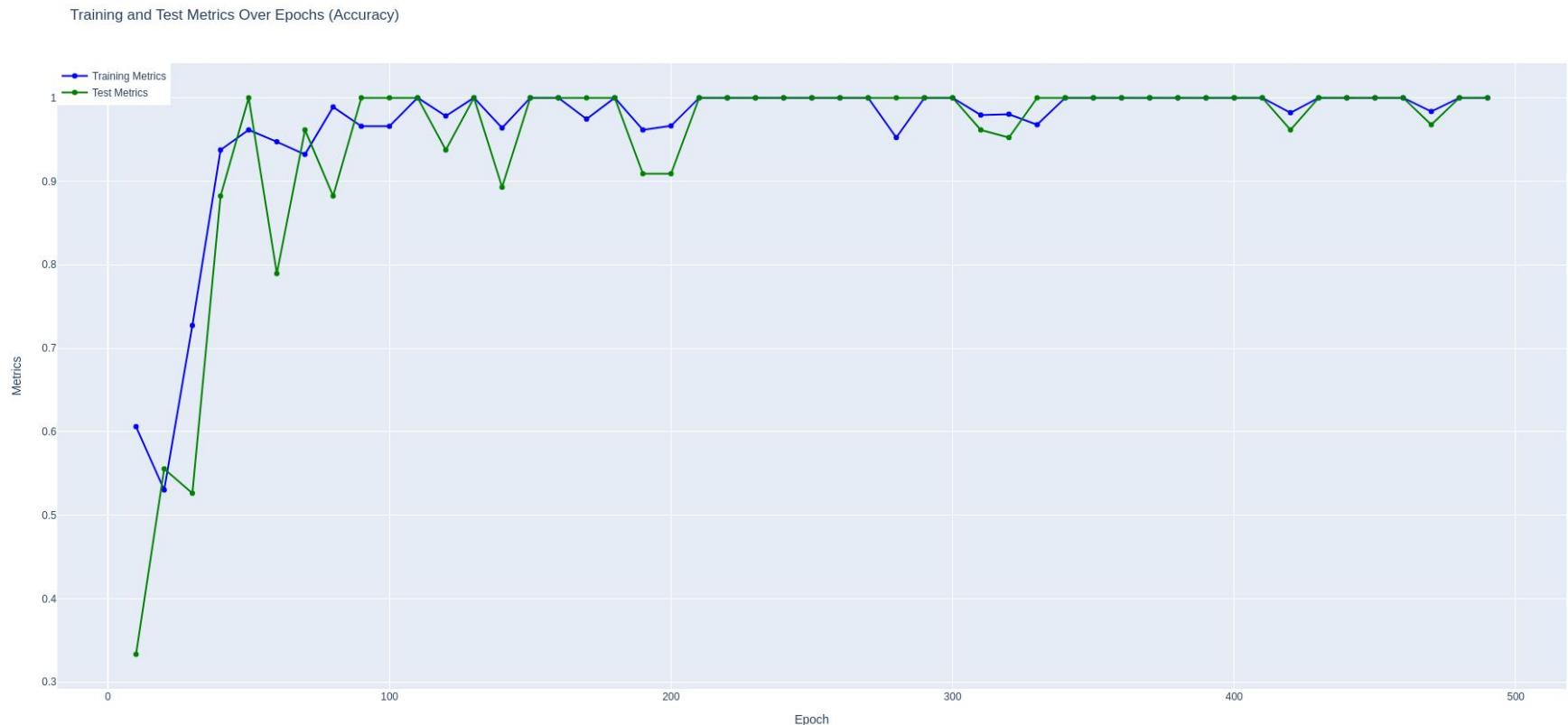
Perceptrón Multicapa: Reconocimiento de números

Si expandimos el dataset...

Perceptrón Multicapa: Reconocimiento de números

Accuracy (Expandiendo el dataset)

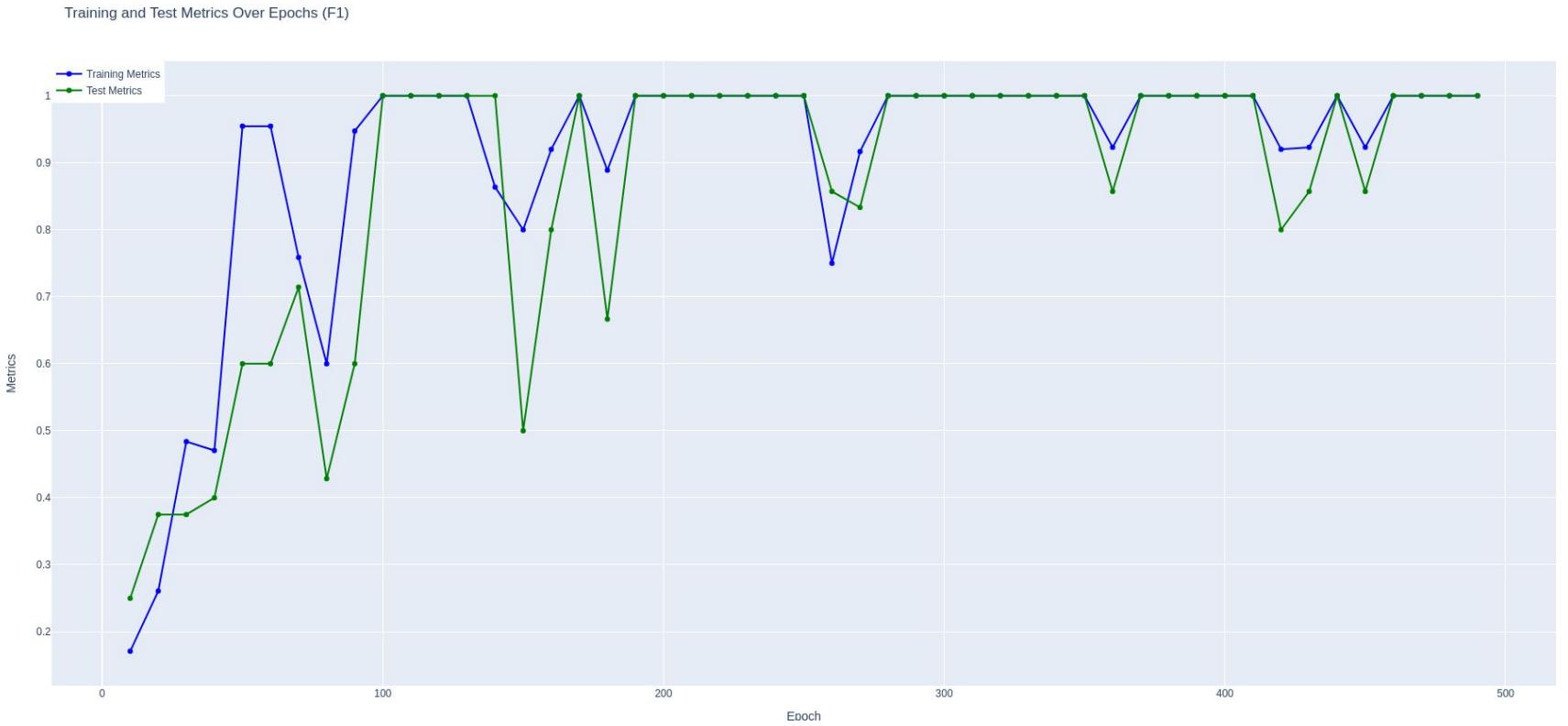
```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
noise = 0.2
architecture = [35, 10, 10]
```



Perceptrón Multicapa: Reconocimiento de números

F1 (Expandiendo el dataset)

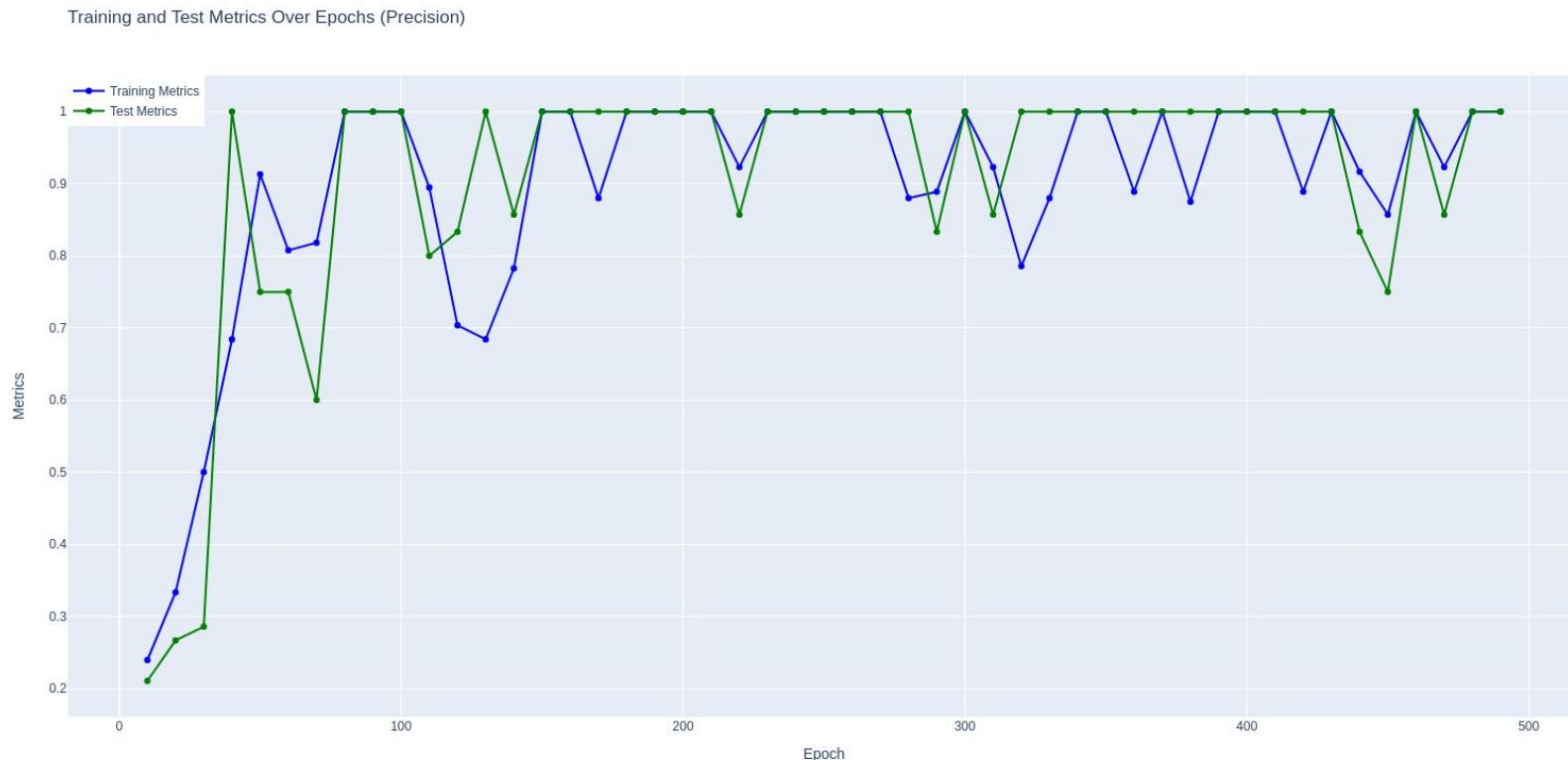
```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
noise = 0.2
architecture = [35, 10, 10]
```



Perceptrón Multicapa: Reconocimiento de números

Precision (Expandiendo el dataset)

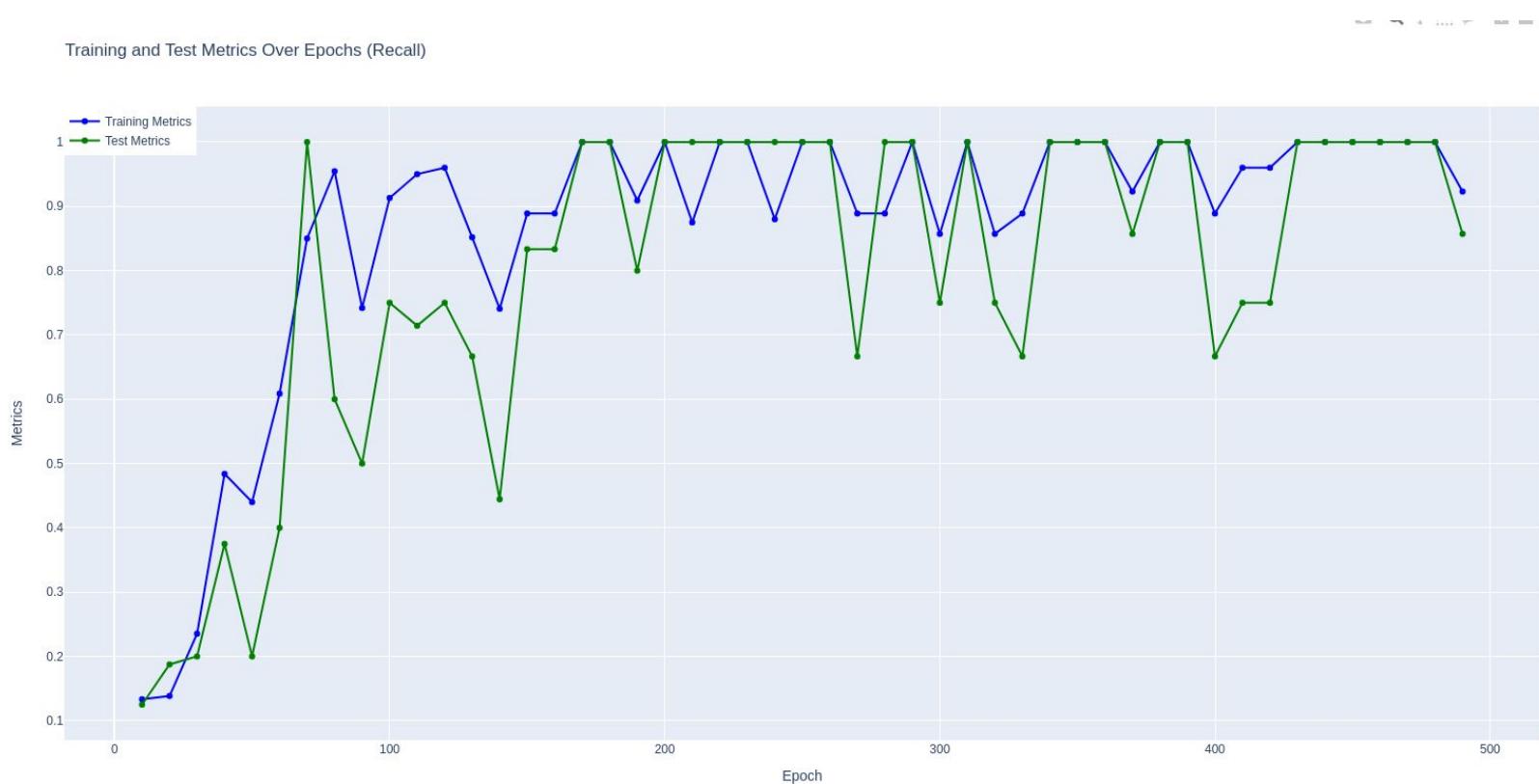
```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
noise = 0.2
architecture = [35, 10, 10]
```



Perceptrón Multicapa: Reconocimiento de números

Recall (Expandiendo el dataset)

```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
noise = 0.2
architecture = [35, 10, 10]
```

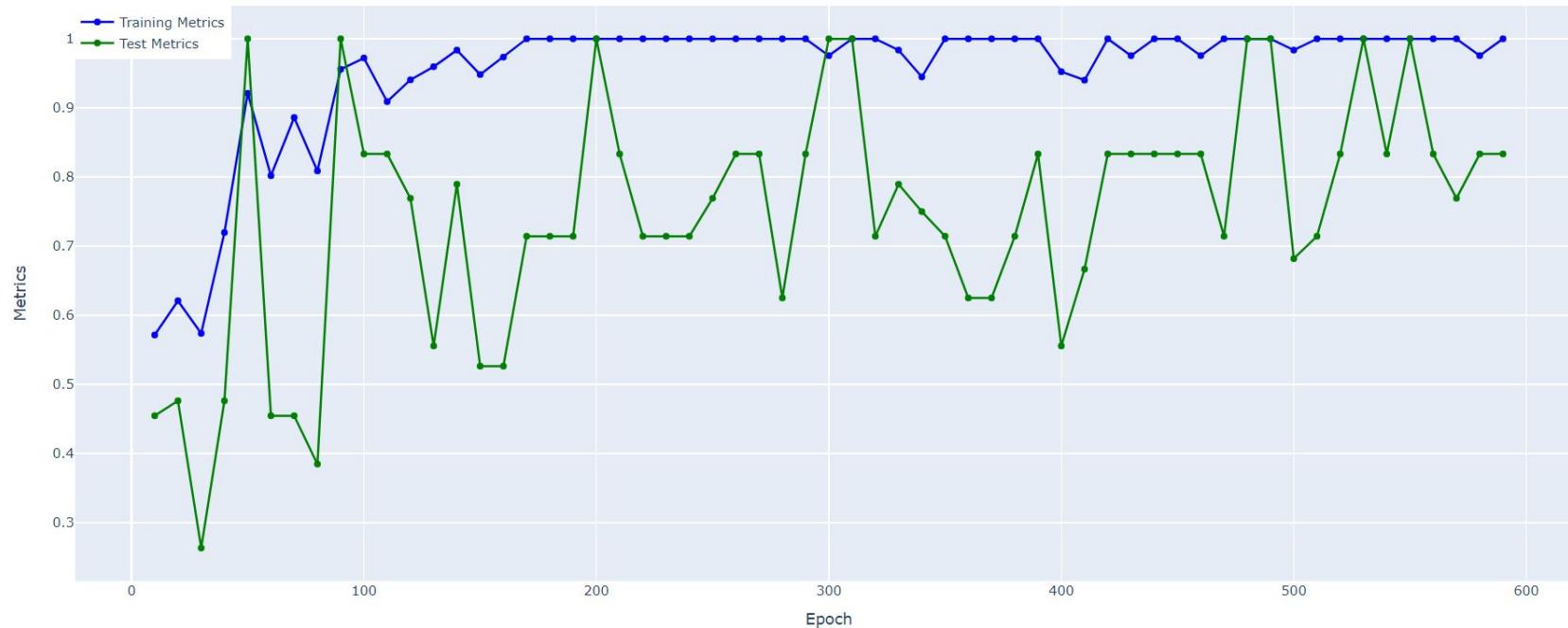


Perceptrón Multicapa: Reconocimiento de números

Accuracy (Expandiendo el dataset y con mucho ruido)

```
dataset: TP3-ej3-digits.txt
activation_function = TAN_H
learning_rate = 0.2
beta = 1
noise = 0.85
architecture = [35, 10, 10]
```

Training and Test Metrics Over Epochs (Accuracy)



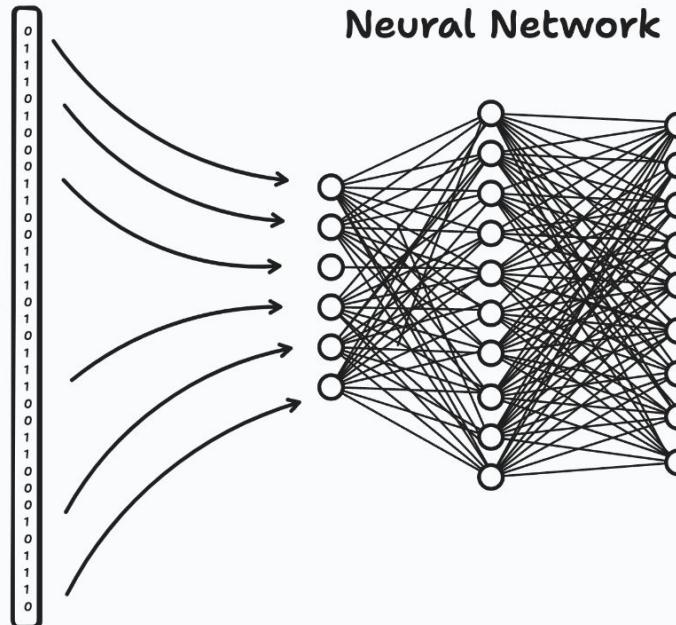
One more thing...



Number dataset

0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	0	0	0	1
0	1	1	1	0

→ Flatten →



Neural Network

...Training

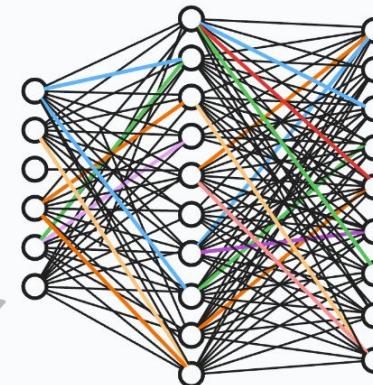
Number dataset

0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	0	0	0	1
0	1	1	1	0

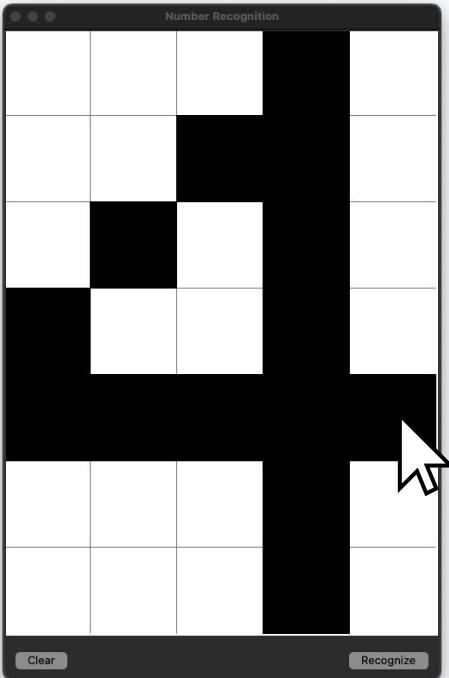
→ Flatten →



Neural Network

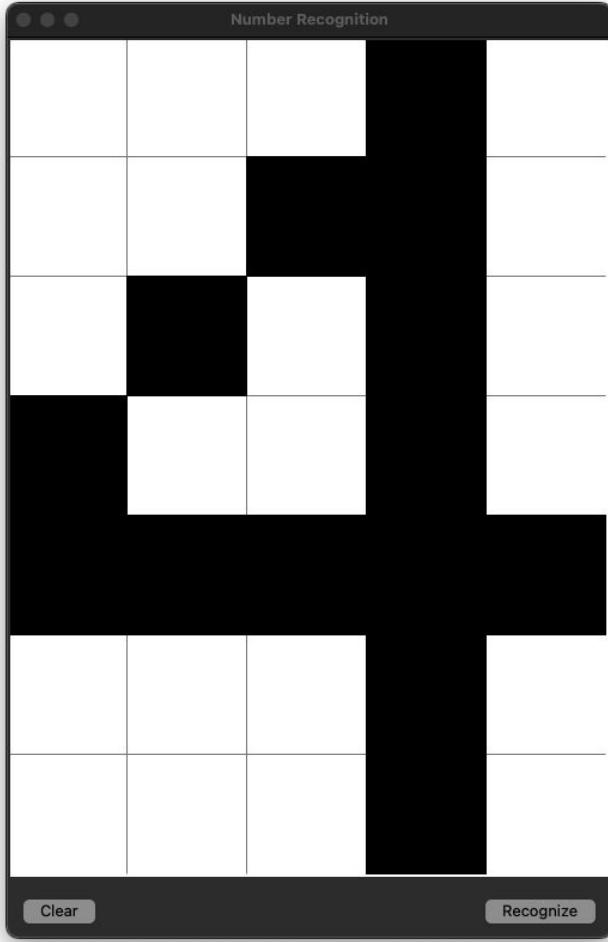


Trained ✓



Neural Network

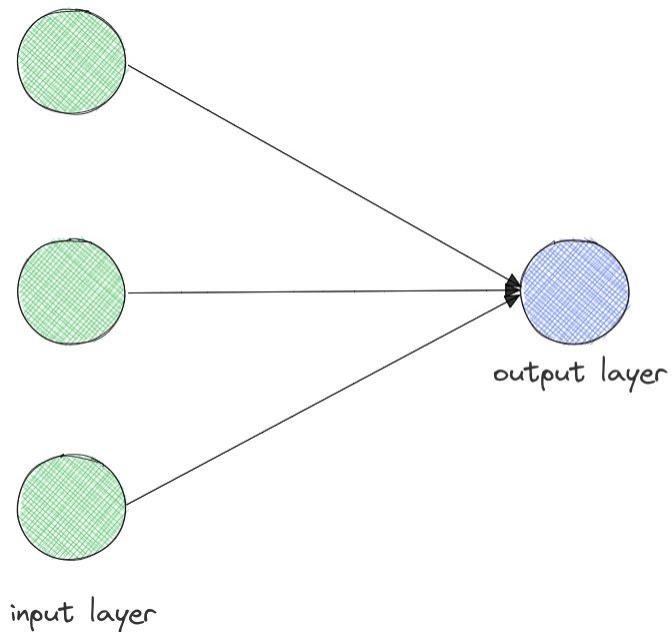


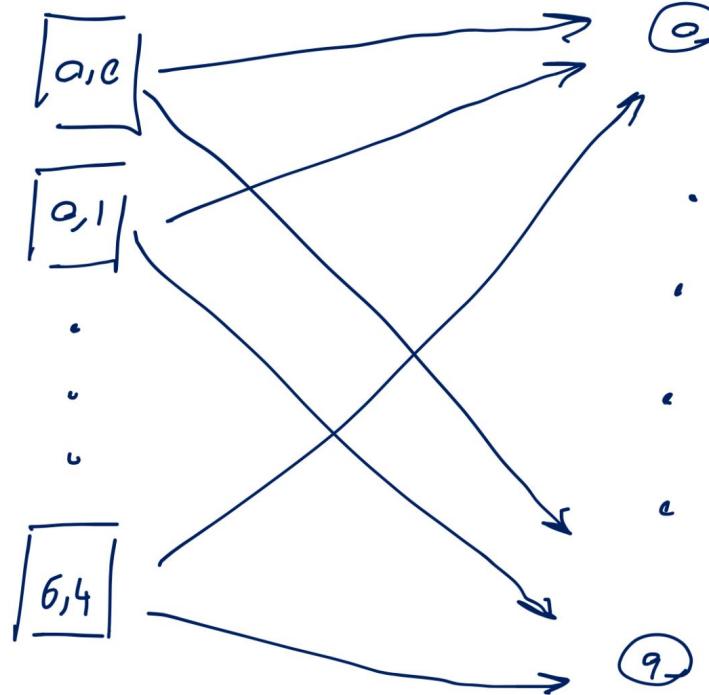
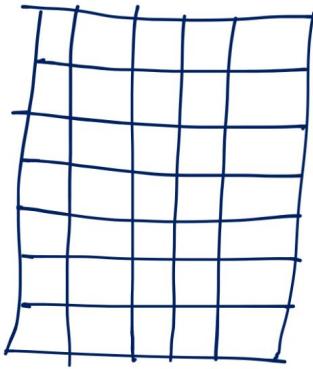


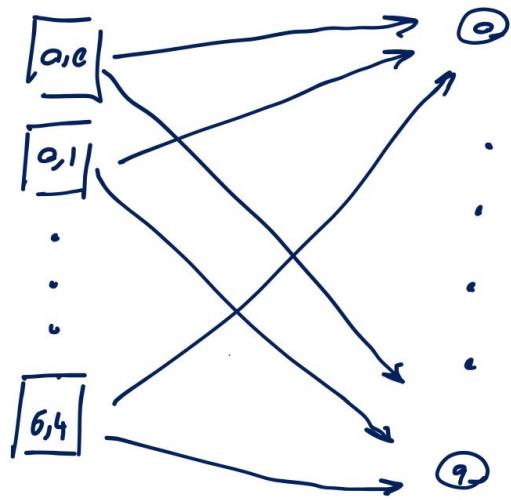
pipenv run python3 cli.py

```
~/Projects/sia/tp3 git:(main) ±16
pipenv run python3 cli.py
[0 1 1 1 0 1 0 0 0 1 1 0 0 1 1 1 0 1 0 1 1 1 0 0 1 1 0 0
 0 1 0 1 1 1 0]
Match: 0
Candidate: 3
Candidate: 6
[0 1 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0
 0 1 0 1 1 1 0]
Match: 3
Candidate: 8
Candidate: 2
[0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 0
 1 0 0 1 1 0 0]
Match: 9
Candidate: 6
Candidate: 3
[1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0
 0 1 0 1 1 1 0]
Match: 5
Candidate: 3
Candidate: 6
[0 1 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0
 0 1 0 1 1 1 0]
Match: 3
Candidate: 8
Candidate: 2
[0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 0
 1 0 0 1 1 0 0]
Match: 9
Candidate: 6
Candidate: 3
[0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 0
 1 0 0 0 0 1 0]
Match: 4
Candidate: 8
Candidate: 1
```

Red neuronal sin hidden layers





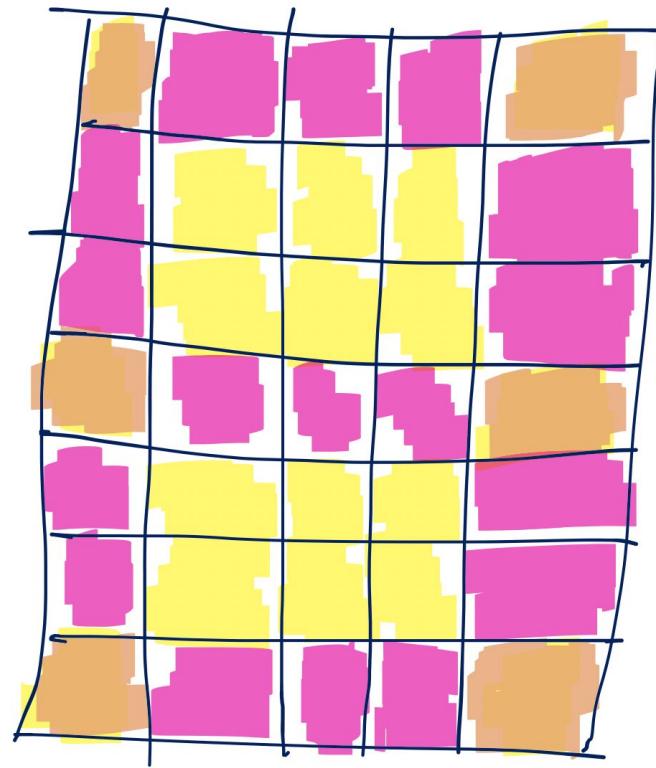


3500 weights
350 per neurone

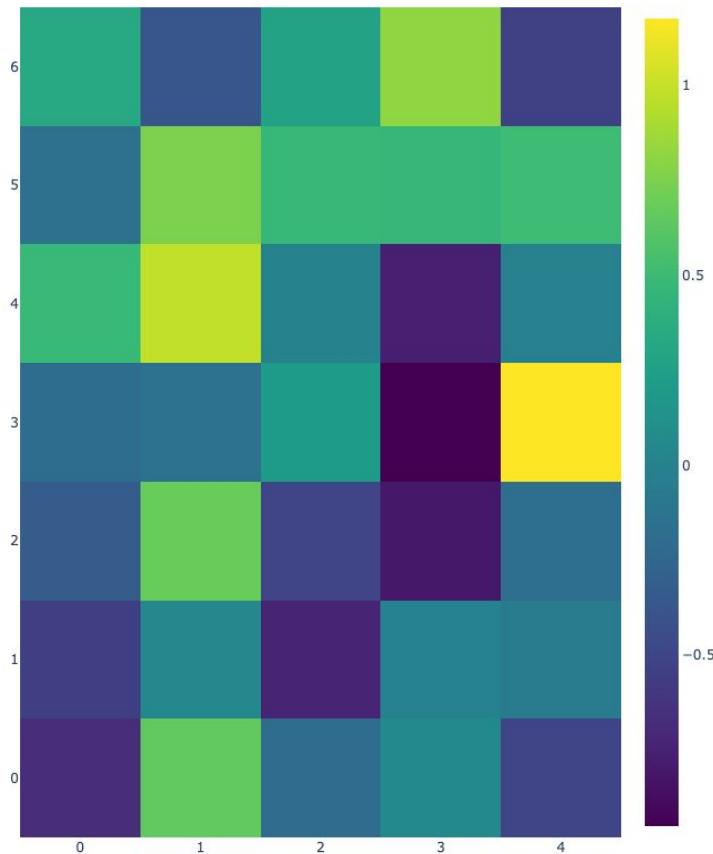


Y si dibujamos los pesos relacionados con cada neurona?

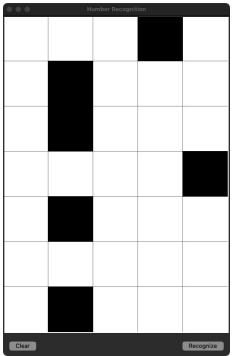
Expectativa



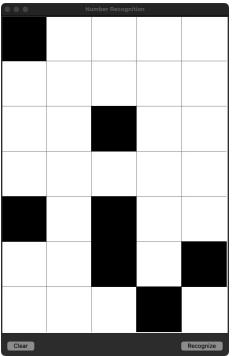
Realidad



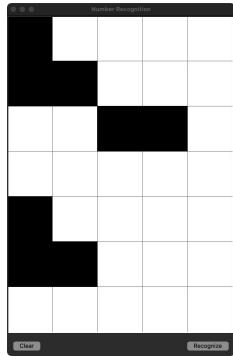
0



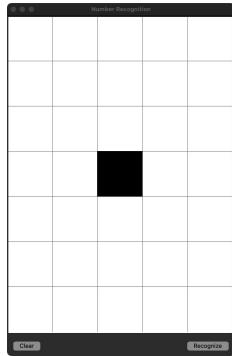
1



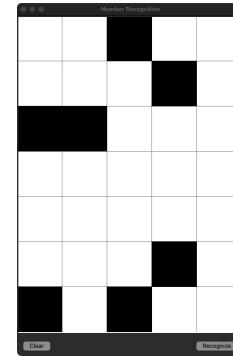
2



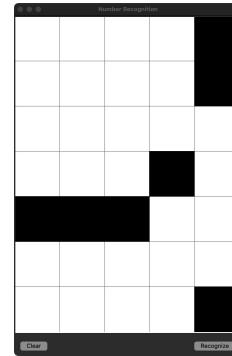
3



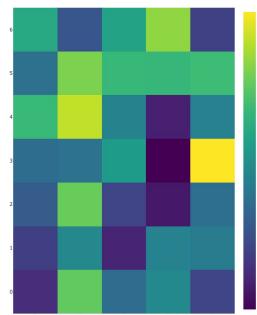
4



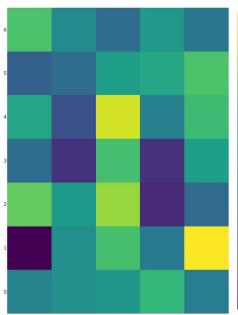
5



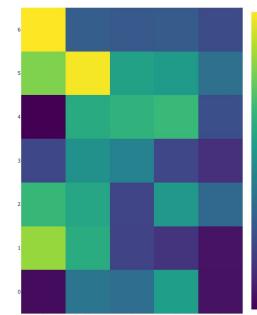
Heatmap Example Digit 0



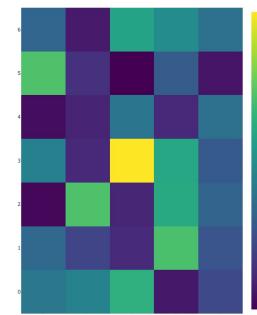
Heatmap Example Digit 1



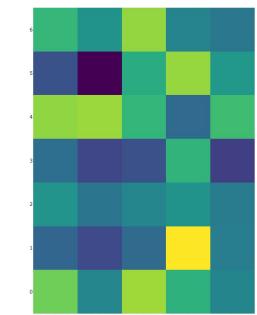
Heatmap Example Digit 2



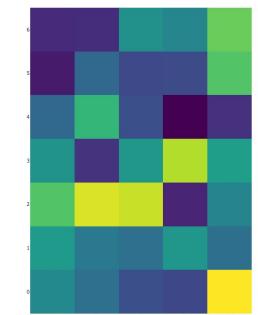
Heatmap Example Digit 3



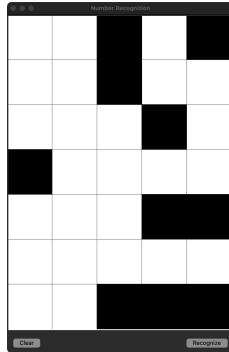
Heatmap Example Digit 4



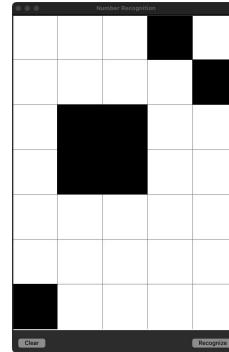
Heatmap Example Digit 5



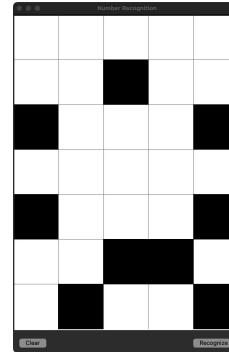
6



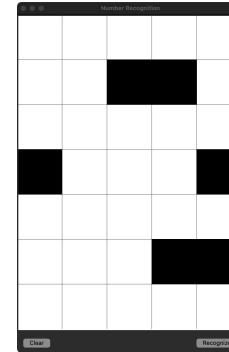
7



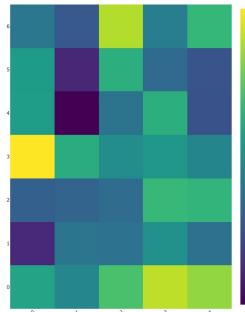
8



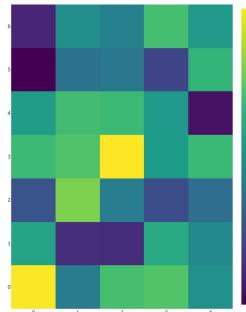
9



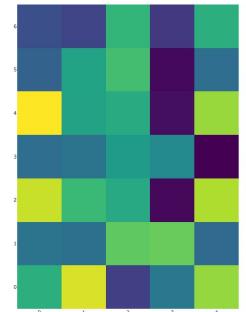
Heatmap Example Digit 6



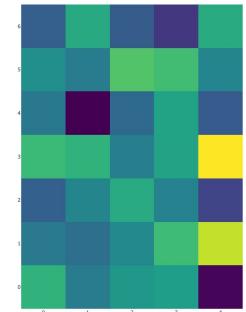
Heatmap Example Digit 7



Heatmap Example Digit 8



Heatmap Example Digit 9



¡Muchas gracias!

Integrantes:

- **Nicolás Matías Margenat, 62028**
 - **Martín Hecht, 62041**
 - **Juan Burda, 62094**
 - **Lautaro Hernando, 62329**
- **Saul Ariel Castañeda, 62493**
 - **Elian Paredes, 62504**