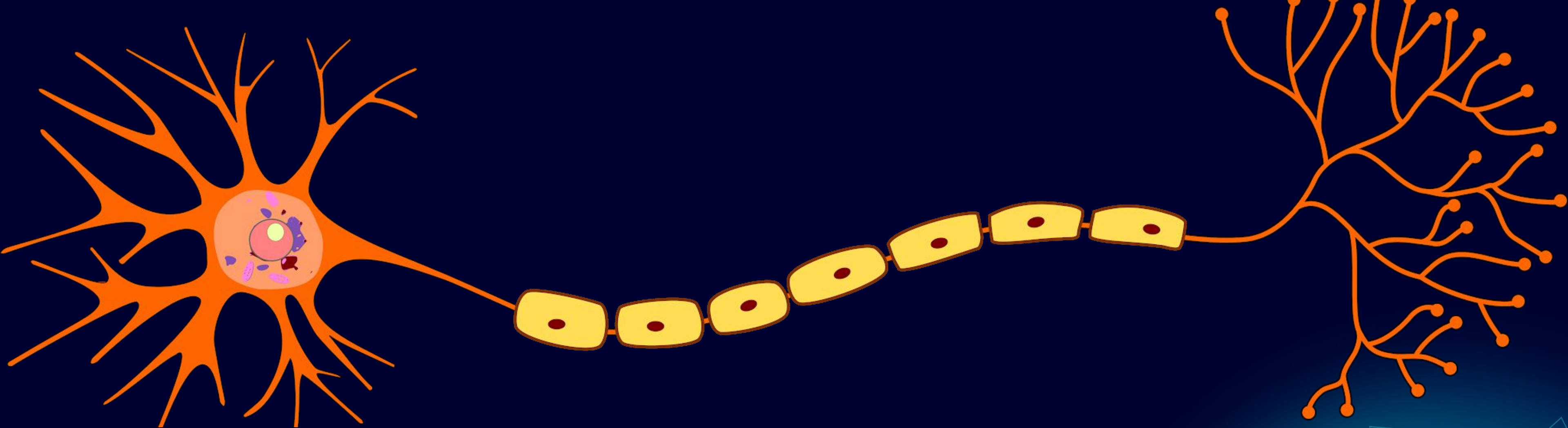
TP3

Perceptron simple y multicapa

72.27 - Sistemas de Inteligencia Artificial



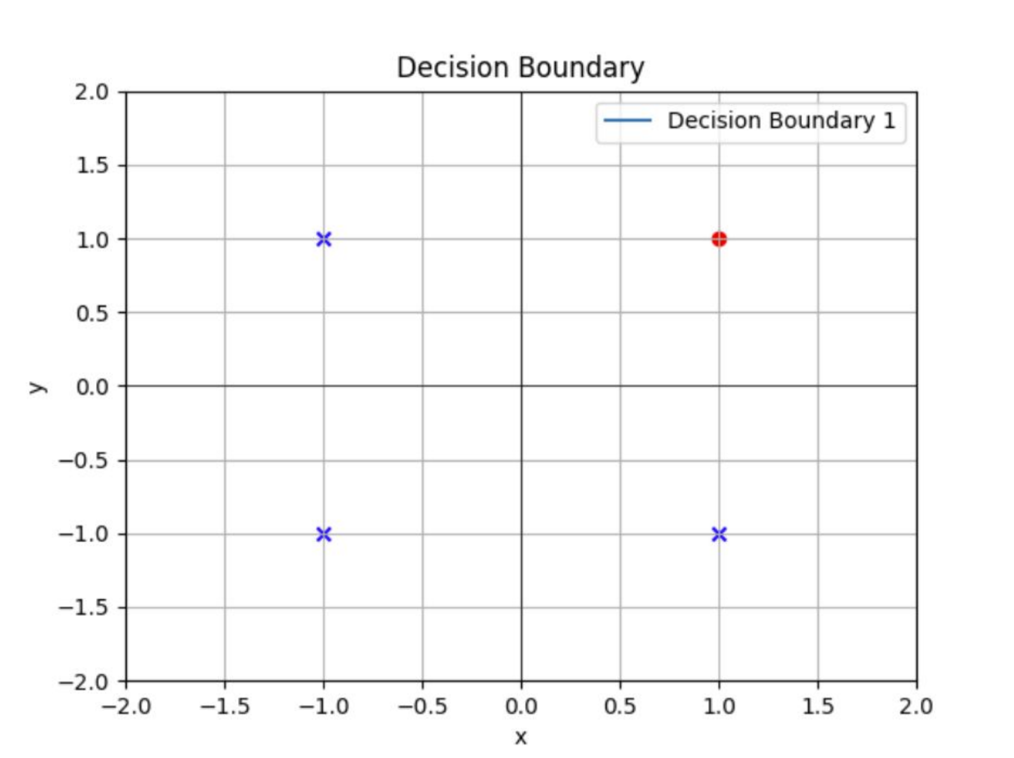
1

Perceptron simple

Función de activación step

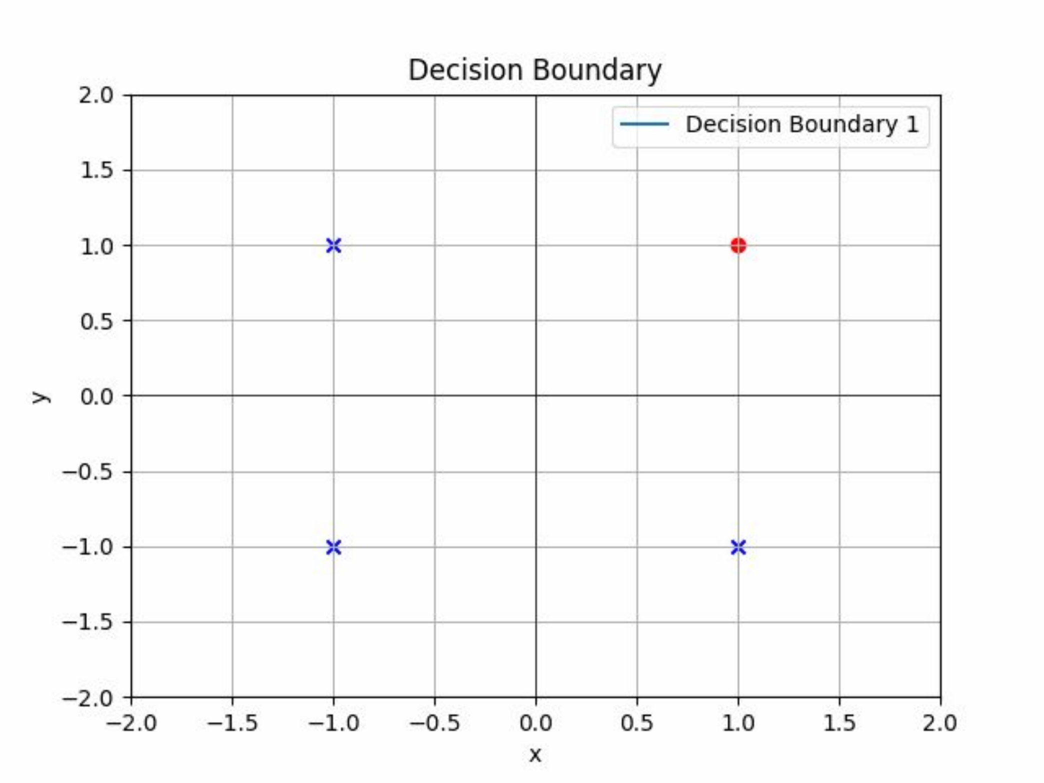
AND

AND



A	B	A **AND** B
0	0	0
0	1	0
1	0	0
1	1	1

AND



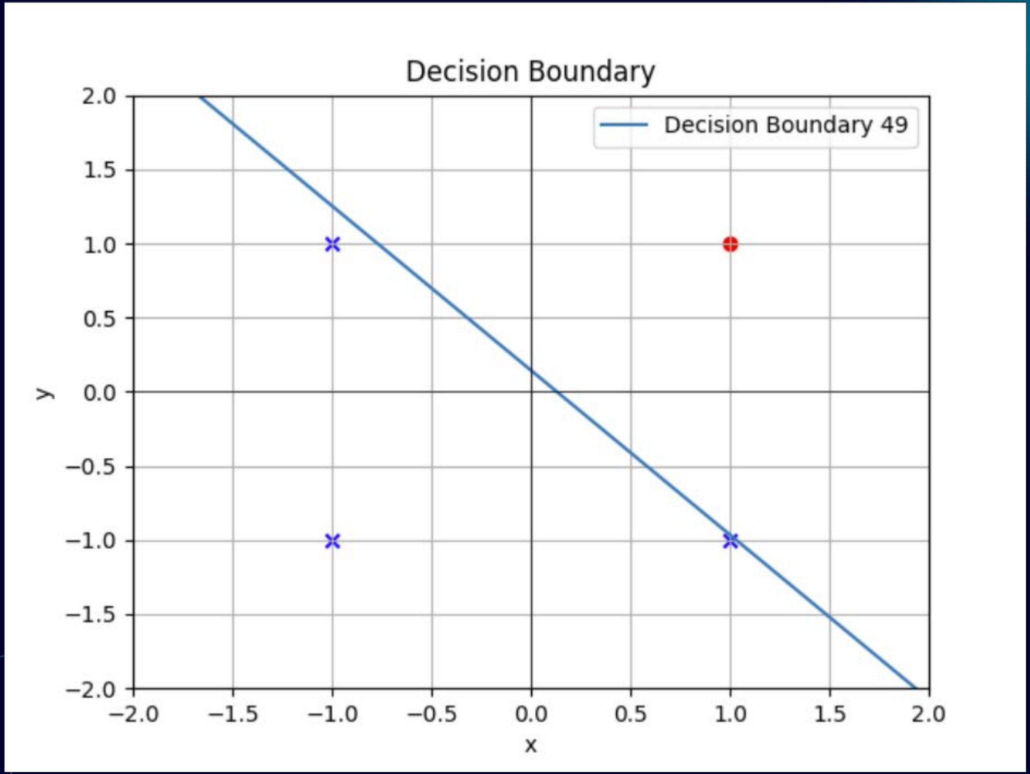
"limit": 1000,

"learning_rate": 0.02,

"bias": 0,

"epsilon": 0.1

AND



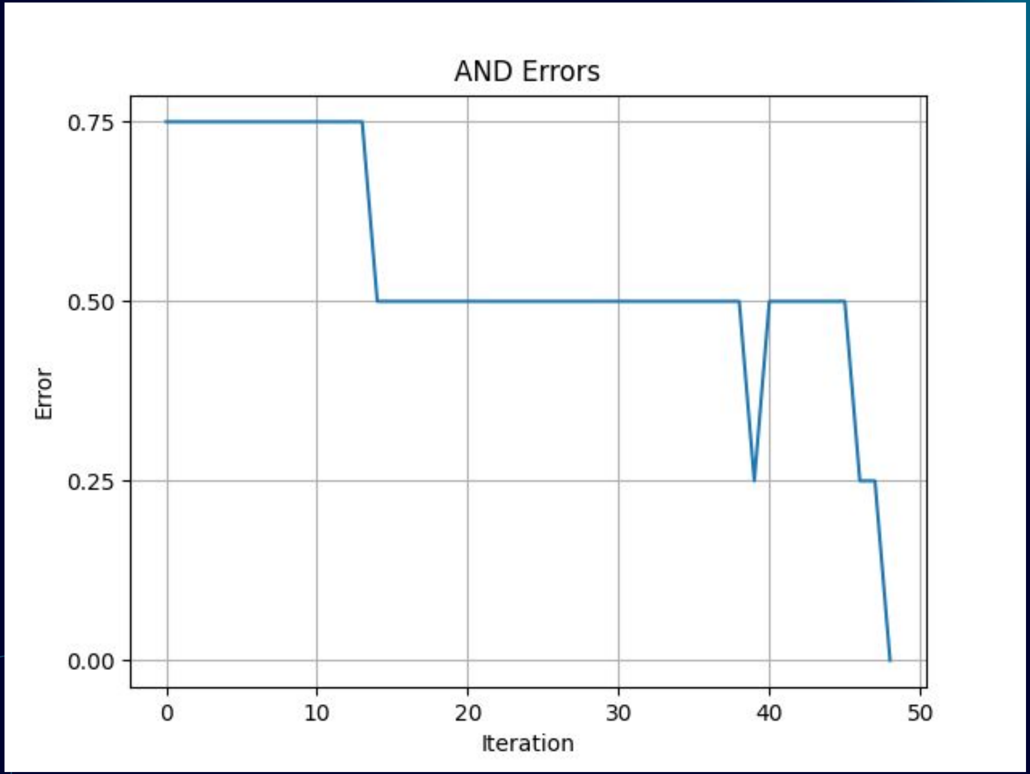
"limit": 1000,

"learning_rate": 0.02,

"bias": 0,

"epsilon": 0.1

AND



"limit": 1000,

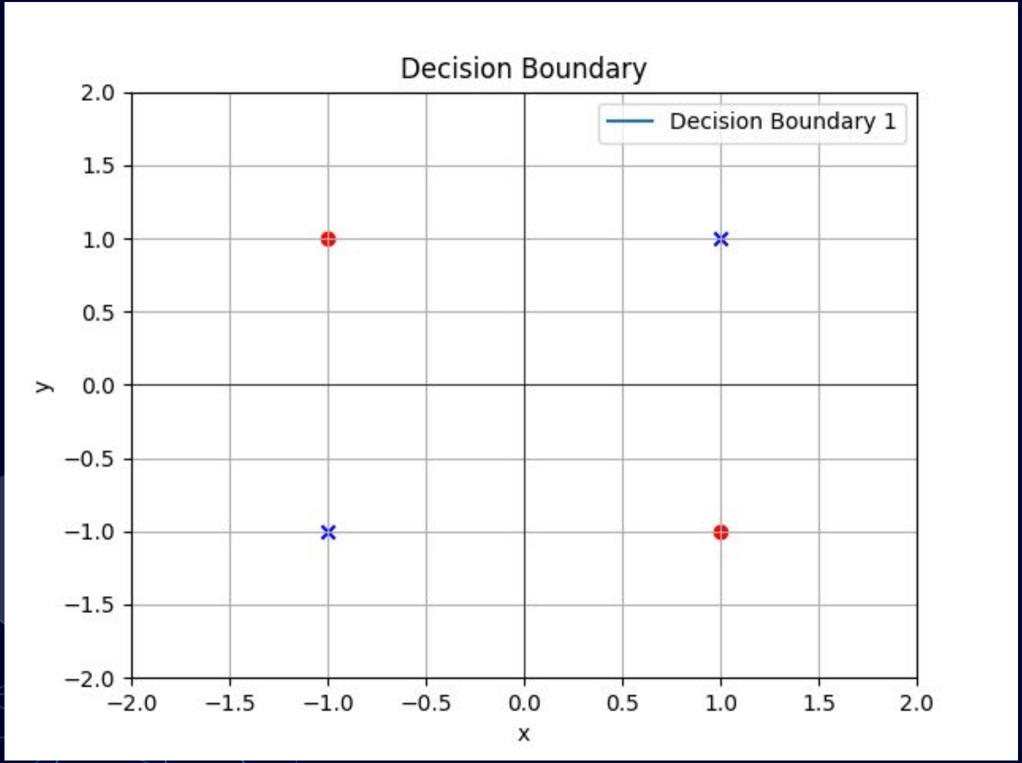
"learning_rate": 0.02,

"bias": 0,

"epsilon": 0.1

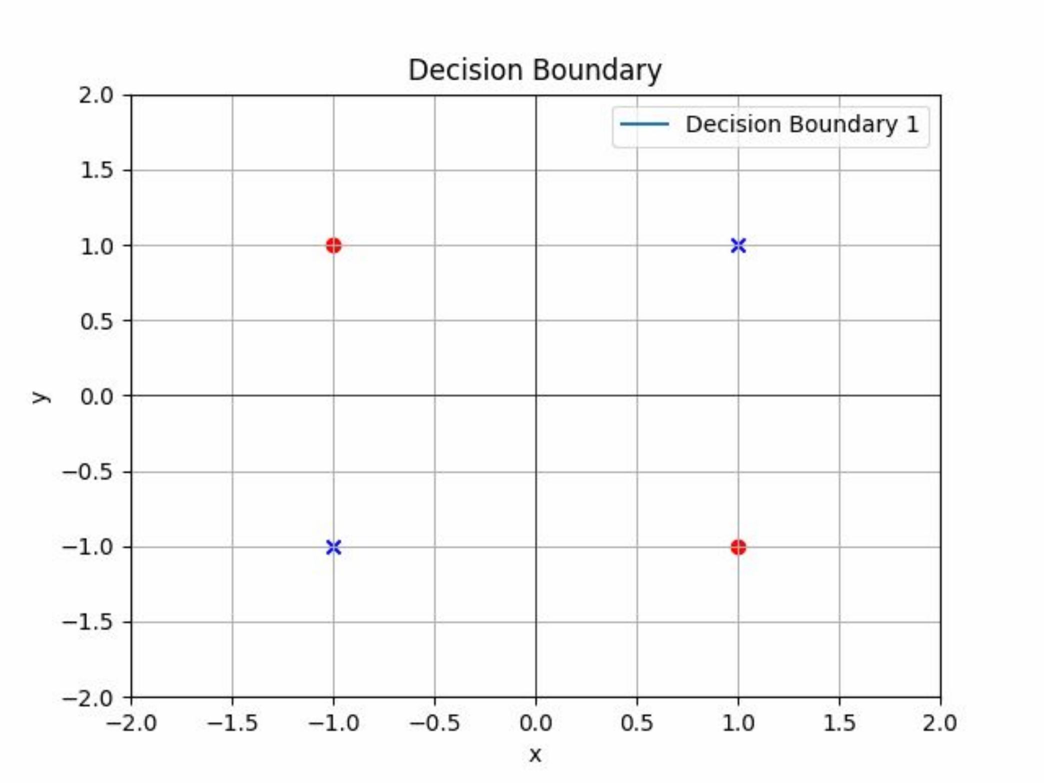
XOR

XOR



A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

XOR



El perceptron:

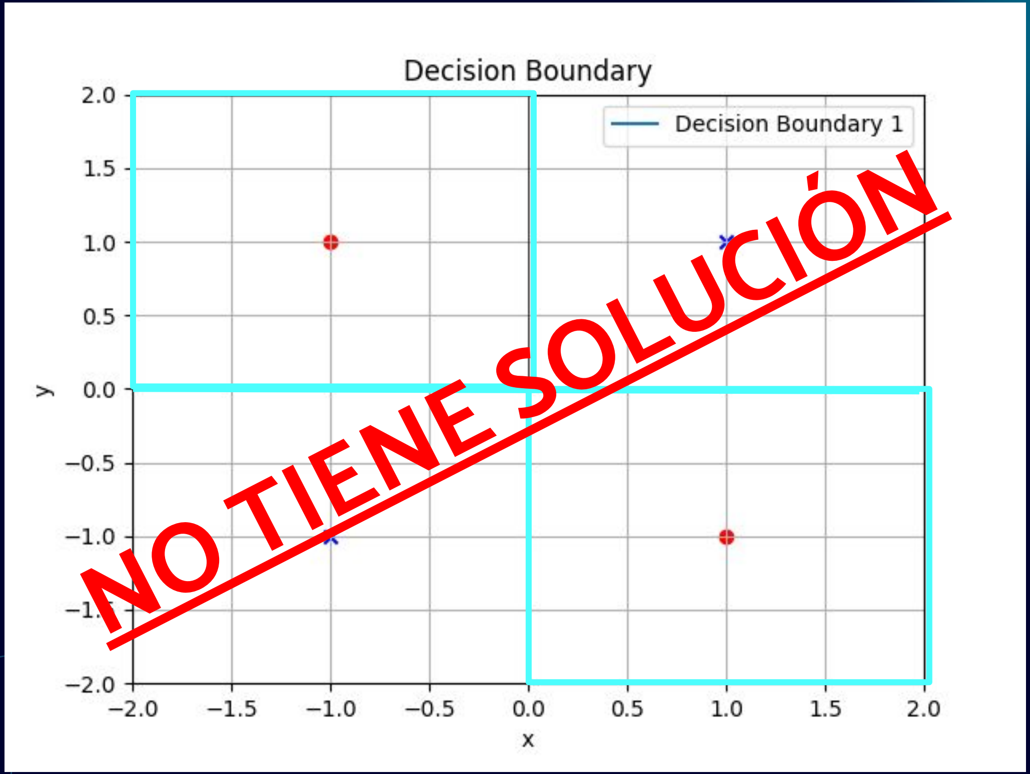
"limit": 1000,

"learning_rate": 0.02,

"bias": 0,

"epsilon": 0.1

XOR



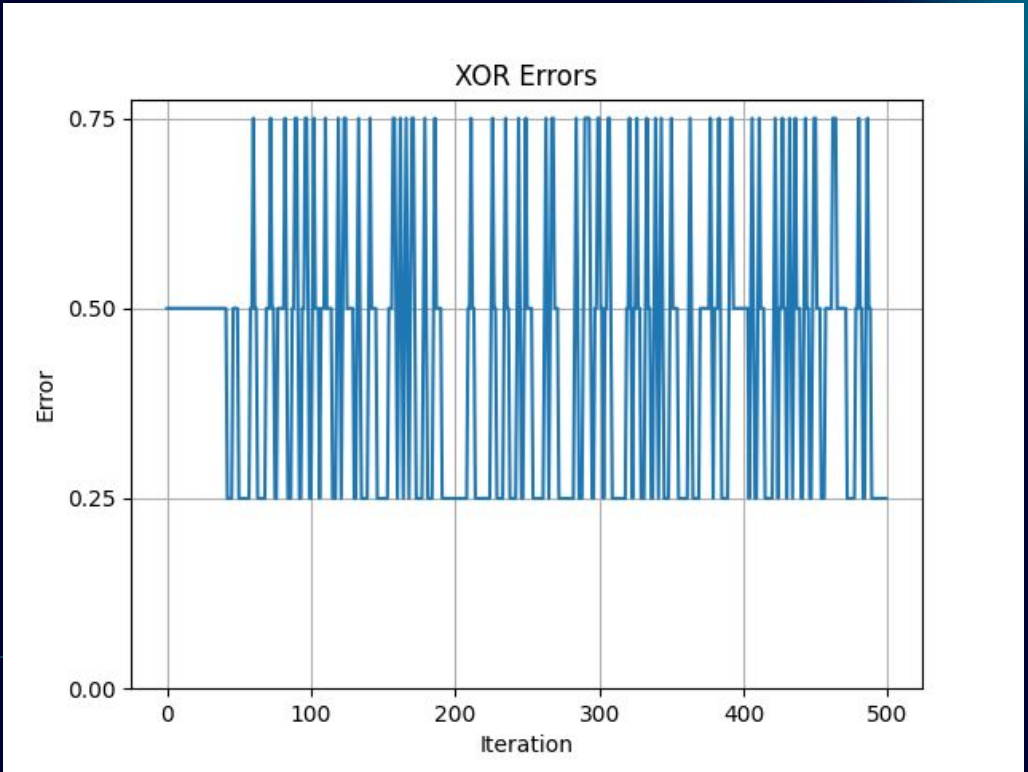
"limit": 1000,

"learning_rate": 0.02,

"bias": 0,

"epsilon": 0.1

XOR



"limit": 1000,

"learning_rate": 0.02,

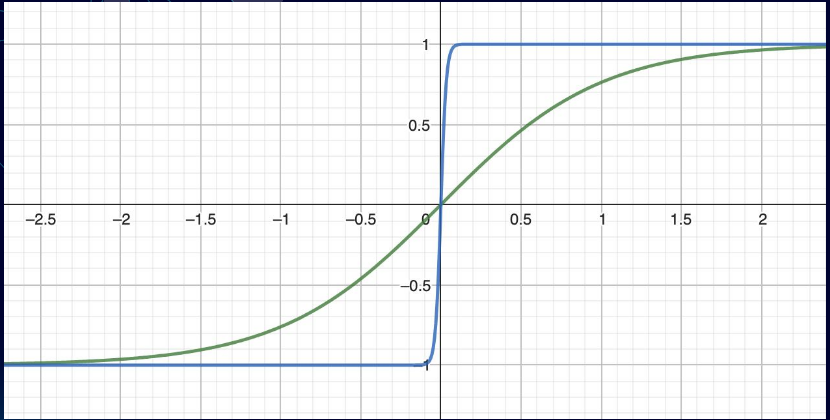
"bias": 0,

"epsilon": 0.1

2

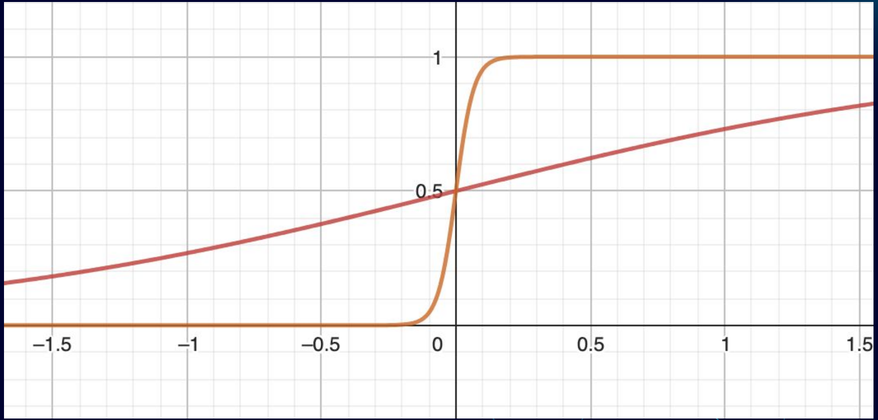
Perceptron Simple Lineal y No Lineal

Análisis de beta



$f(x) = \mathrm{tgh}(x)$

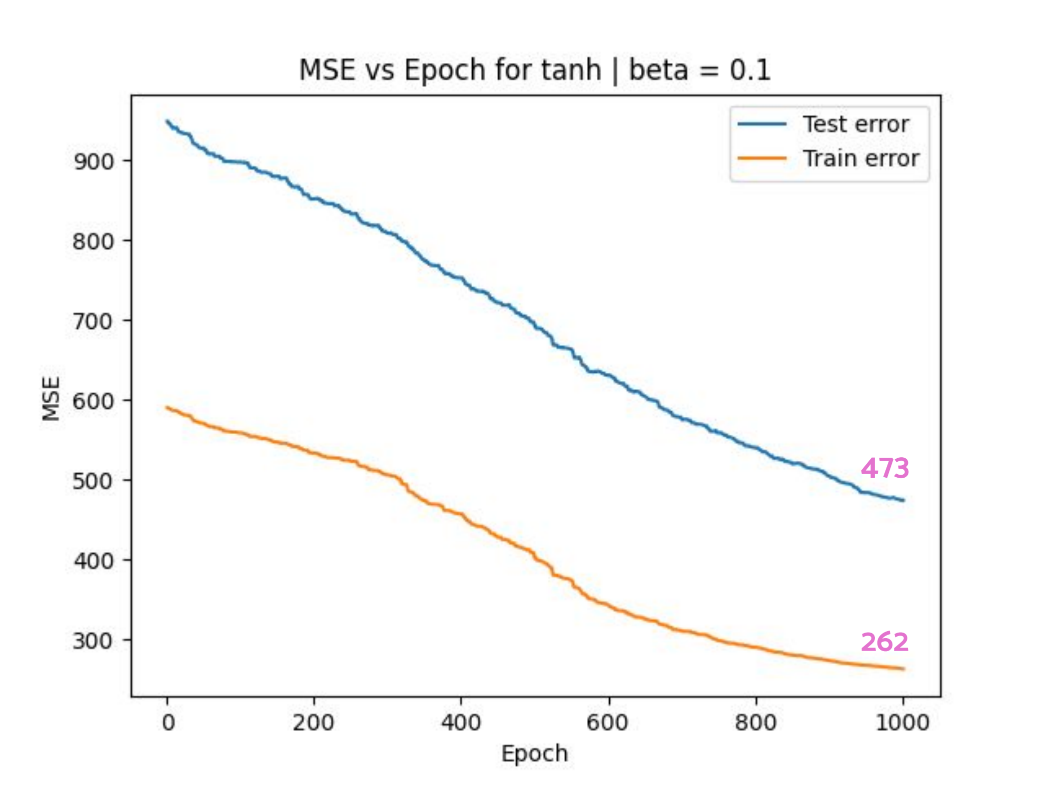
$g(x) = \mathrm{tgh}(30\,x)$



$$f(x) = \frac{1}{1 + e^{-x}}$$

$$g(x) = \frac{1}{1 + e^{-30x}}$$

Tanh

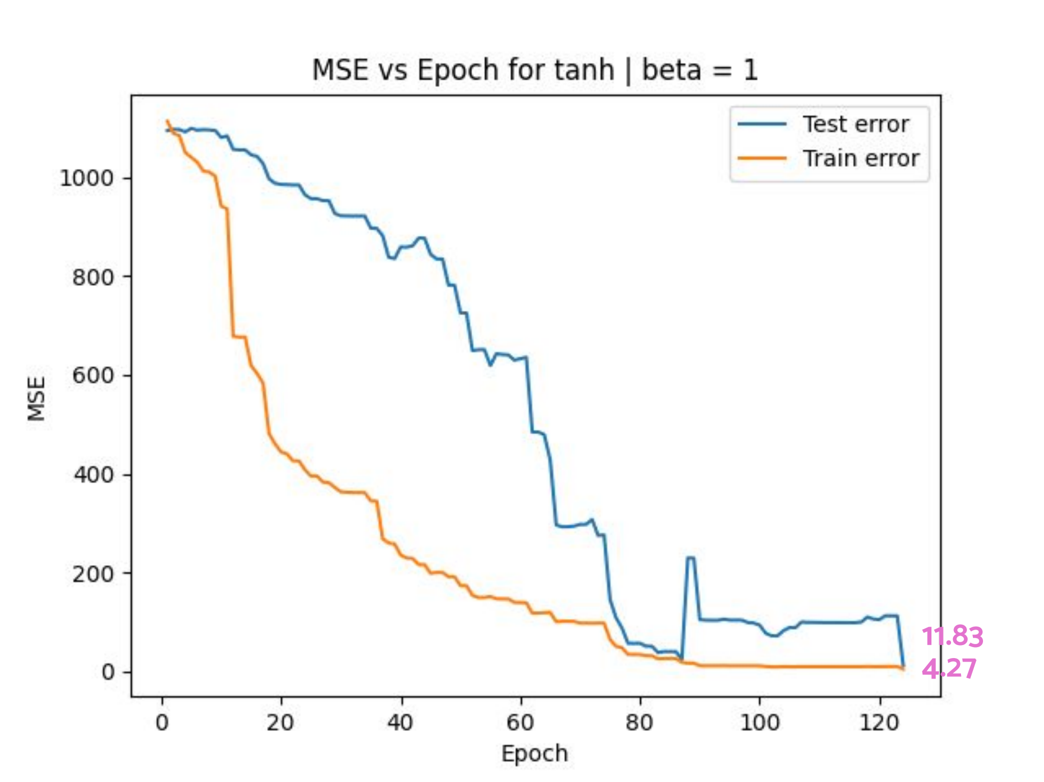


{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 6,
"beta": 0.1
}

#train = 24

#test = 4

Tanh

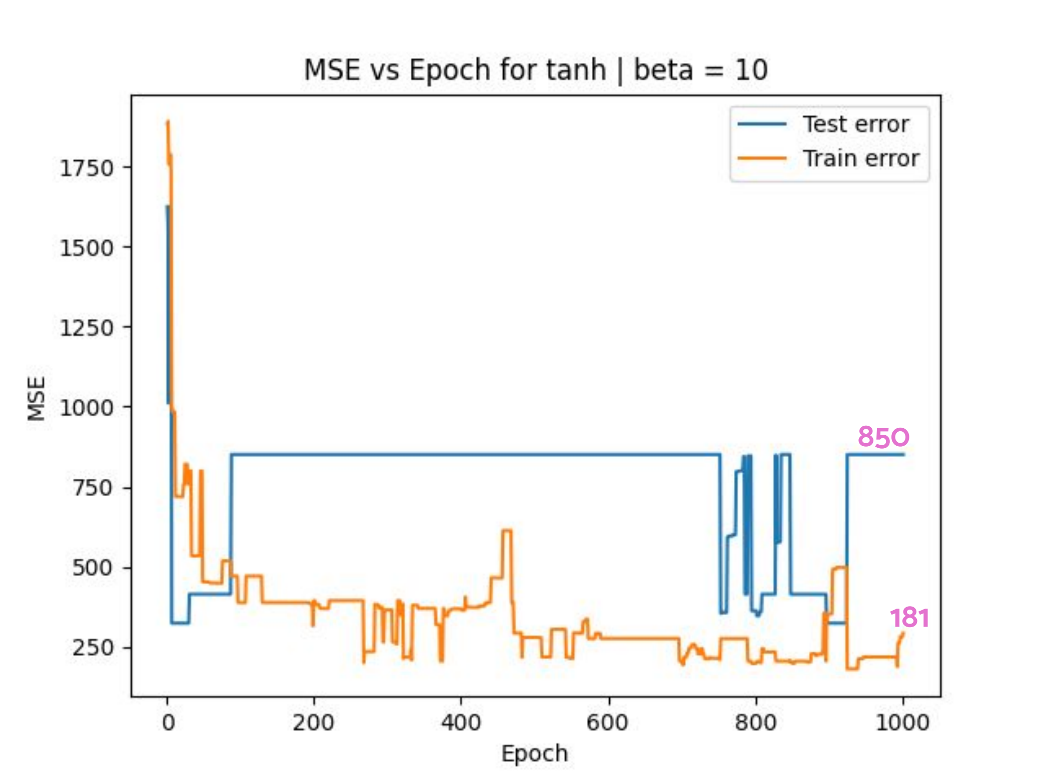


{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 6,
"beta": 1
}

#train = 24

#test = 4

Tanh

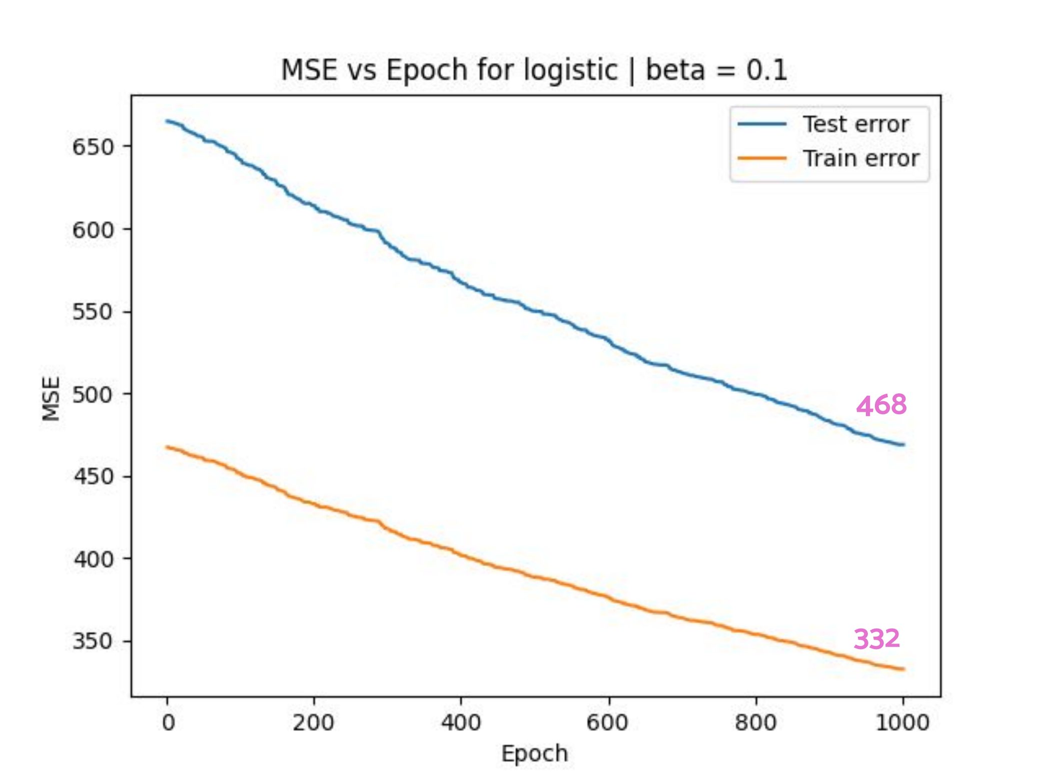


{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 6,
"beta": 10
}

#train = 24

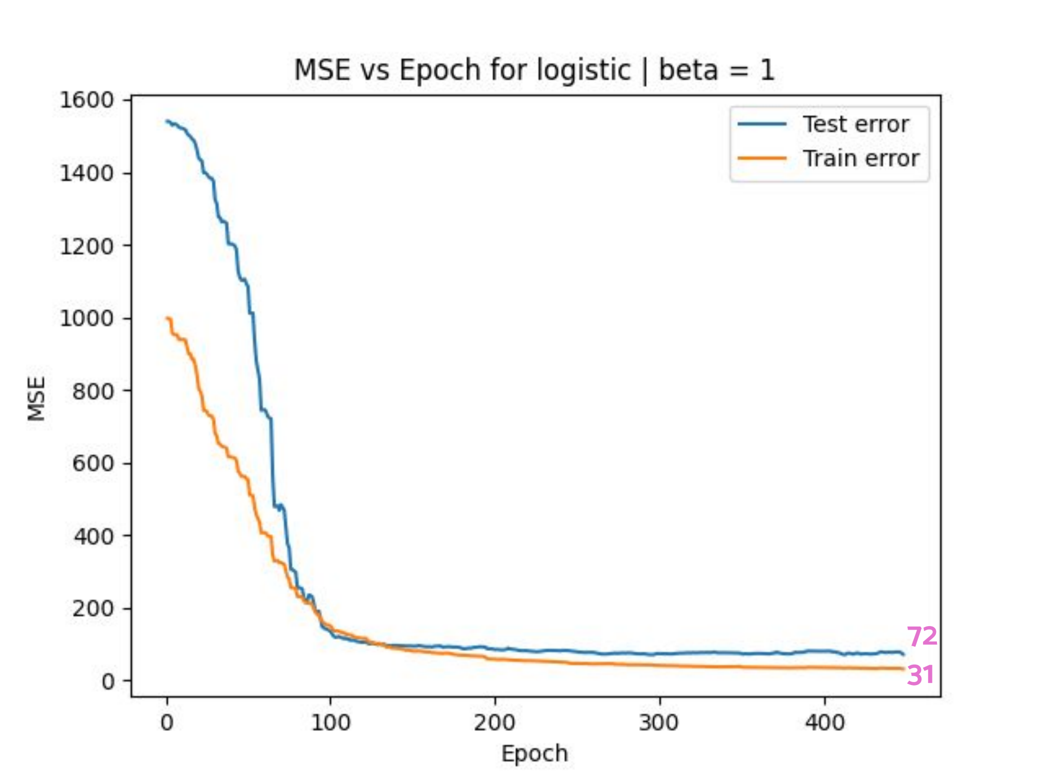
#test = 4

Logistic




```
{
  "limit": 1000,
  "learning_rate": 0.05,
  "bias": 0,
  "epsilon": 0.1,
  "k": 6,
  "beta": 0.1
}
#train = 24
#test = 4
```

Logistic

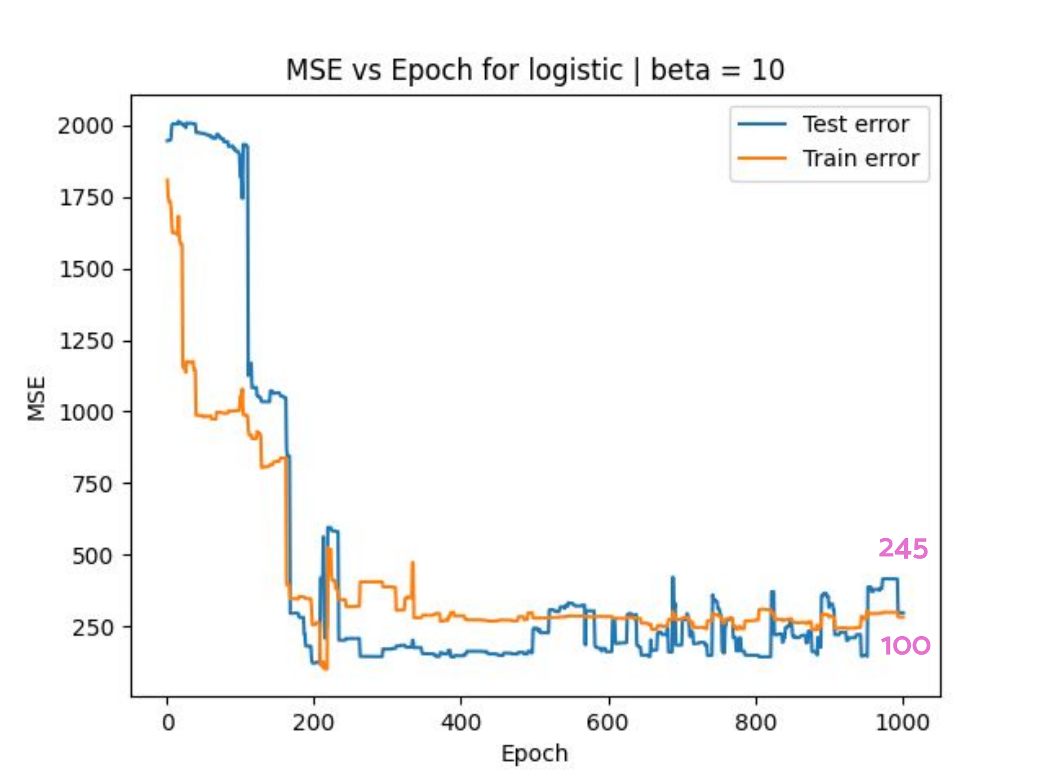


{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 6,
"beta": 1
}

#train = 24

#test = 4

Logistic



{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 6,
"beta": 10
}

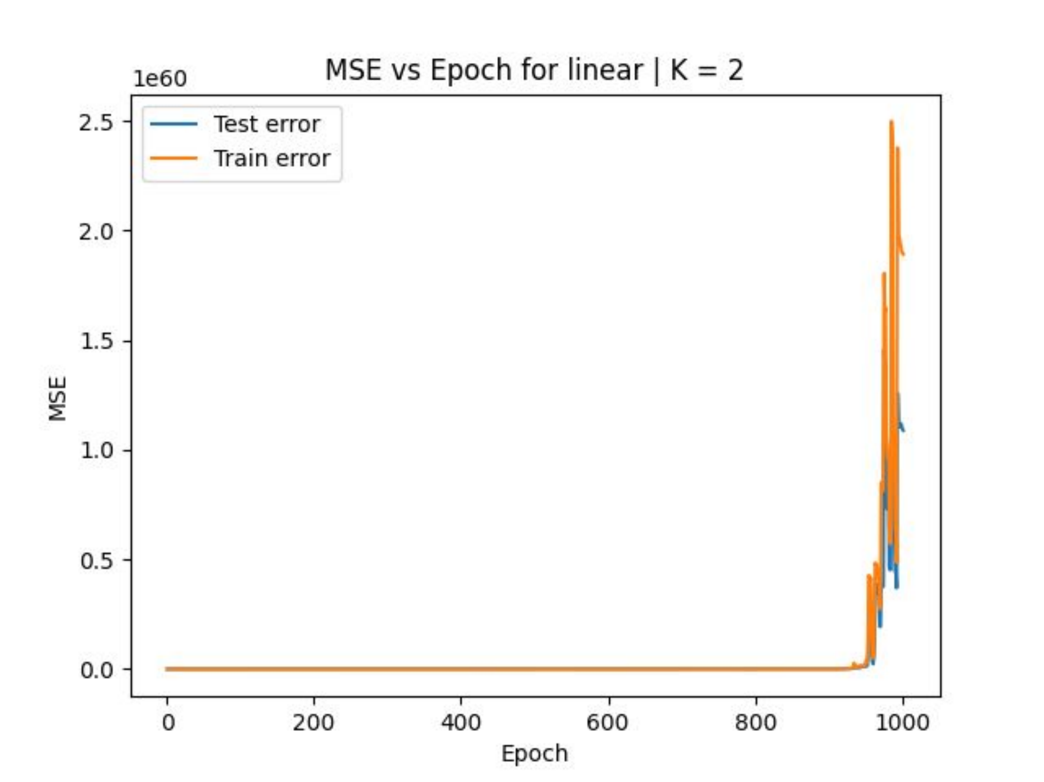
#train = 24

#test = 4

Análisis de k

Lineal

Lineal

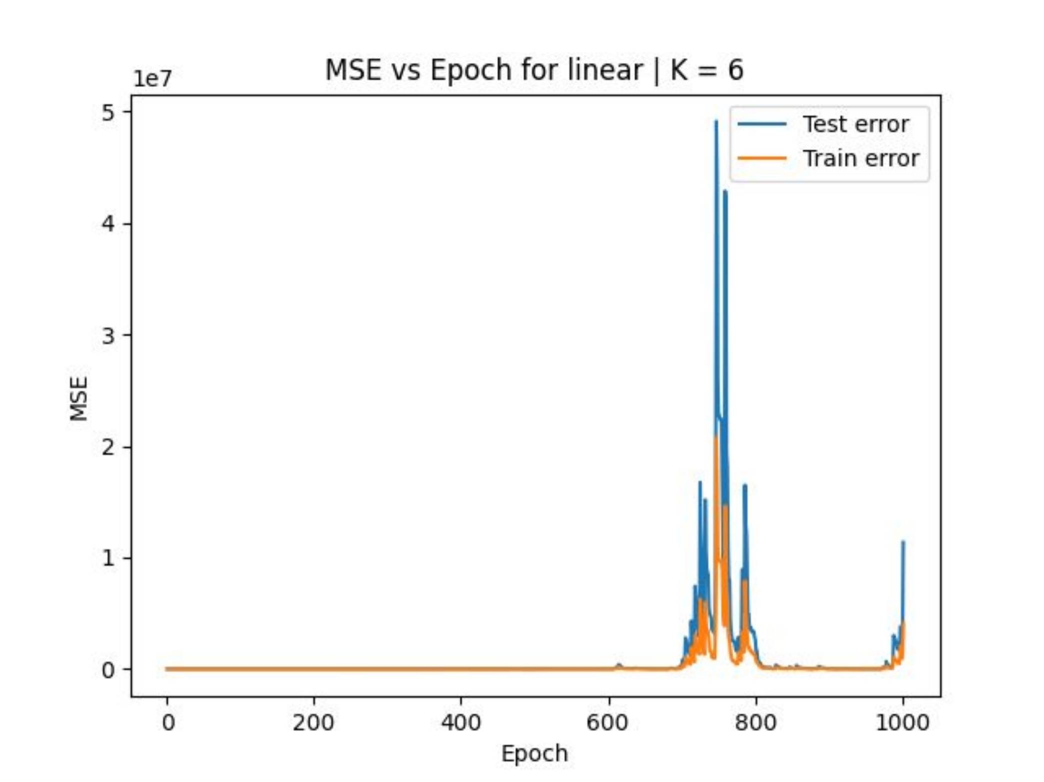


{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 2,
}

#train = 14

#test = 14

Lineal

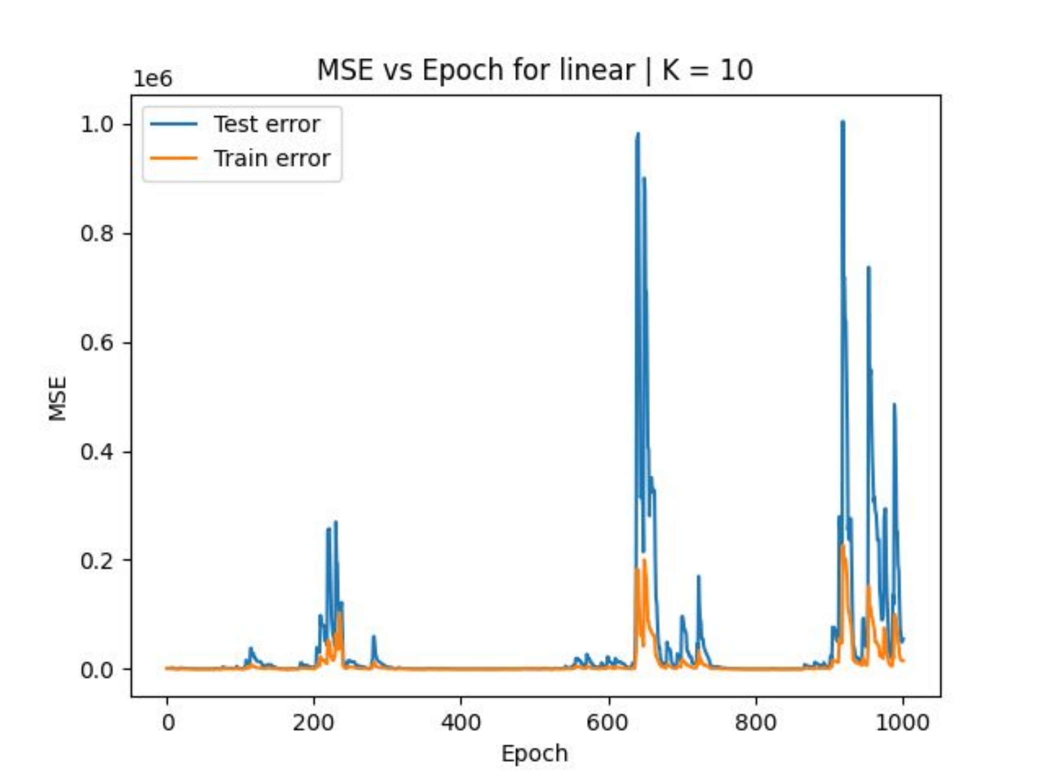


{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 6,
}

#train = 24

#test = 4

Lineal



{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 10,
}

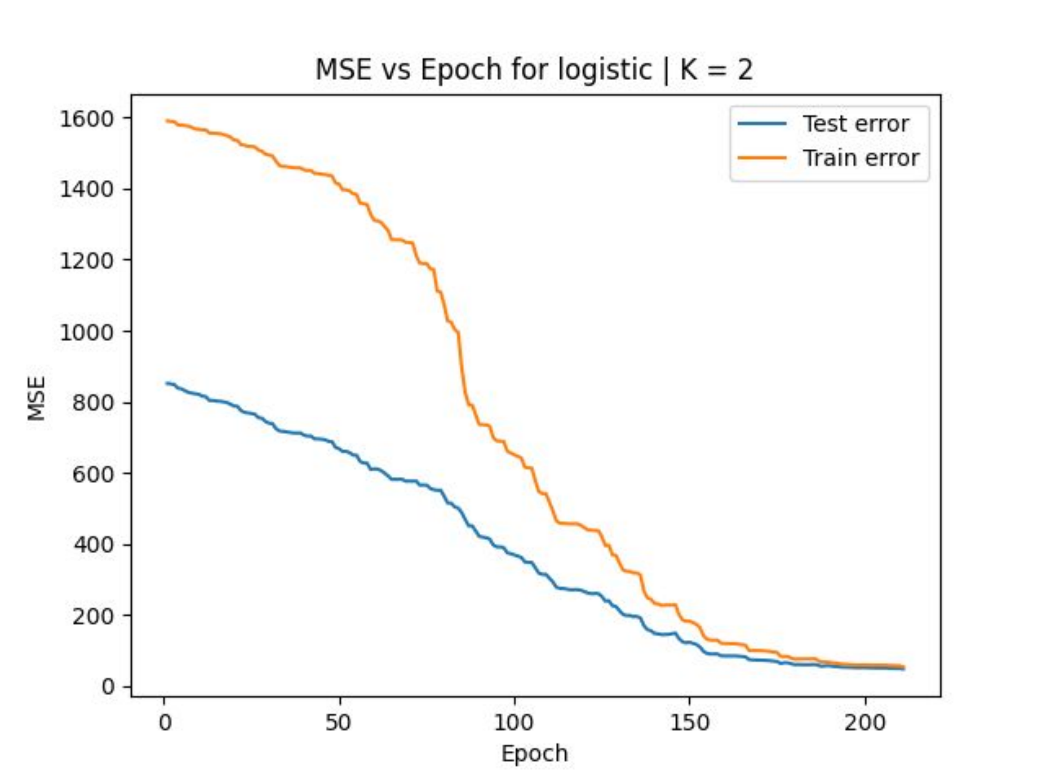
#train = 26

#test = 2

Logistic

ax.xaxis.set_major_locator(plt.MultipleLocator(100))

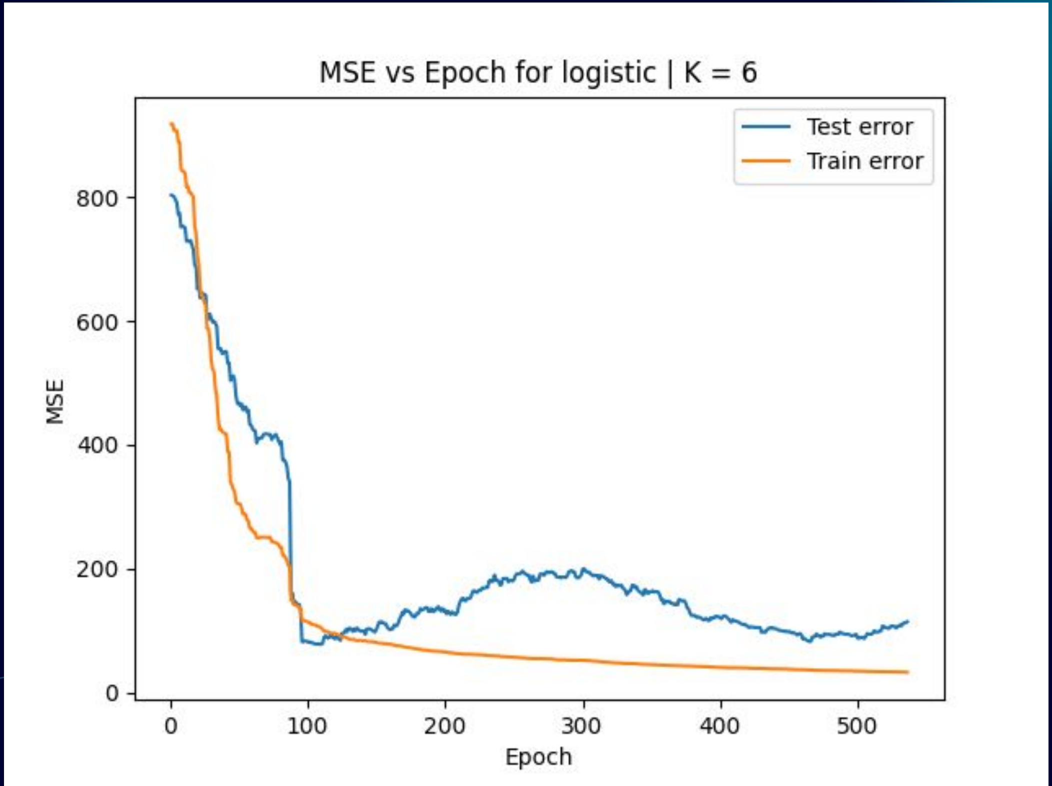
Logistic




```
{
  "limit": 1000,
  "learning_rate": 0.05,
  "bias": 0,
  "epsilon": 0.1,
  "k": 2,
  "beta": 1
}
#train = 14
#test = 14
```

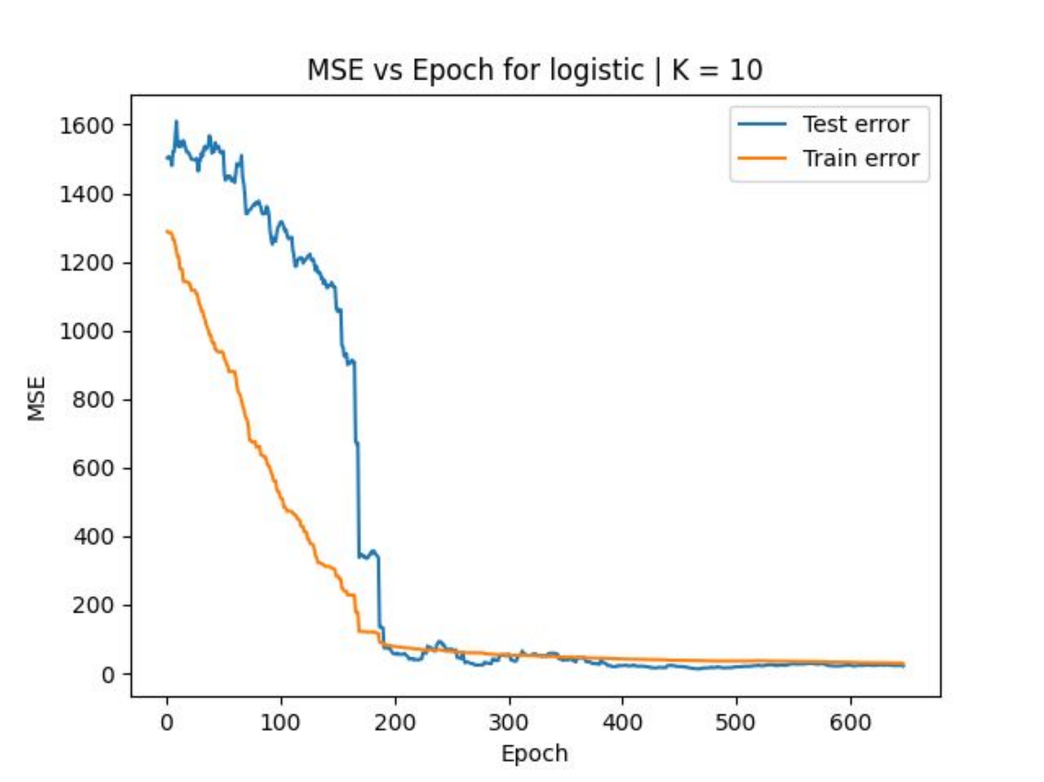
Error

Logistic




```
{
  "limit": 1000,
  "learning_rate": 0.05,
  "bias": 0,
  "epsilon": 0.1,
  "k": 6,
  "beta": 1
}
#train = 24
#test = 4
```

Logistic



{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 10,
"beta": 1
}

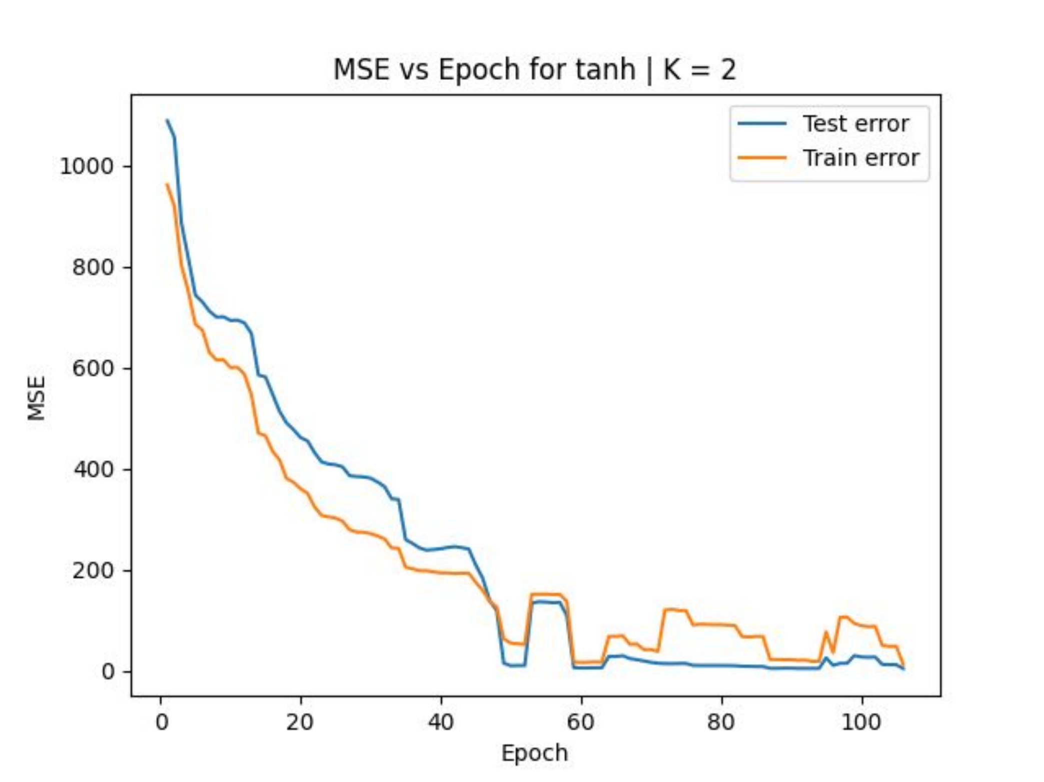
#train = 26

#test = 2

Tanh

Error
} Error

Tanh



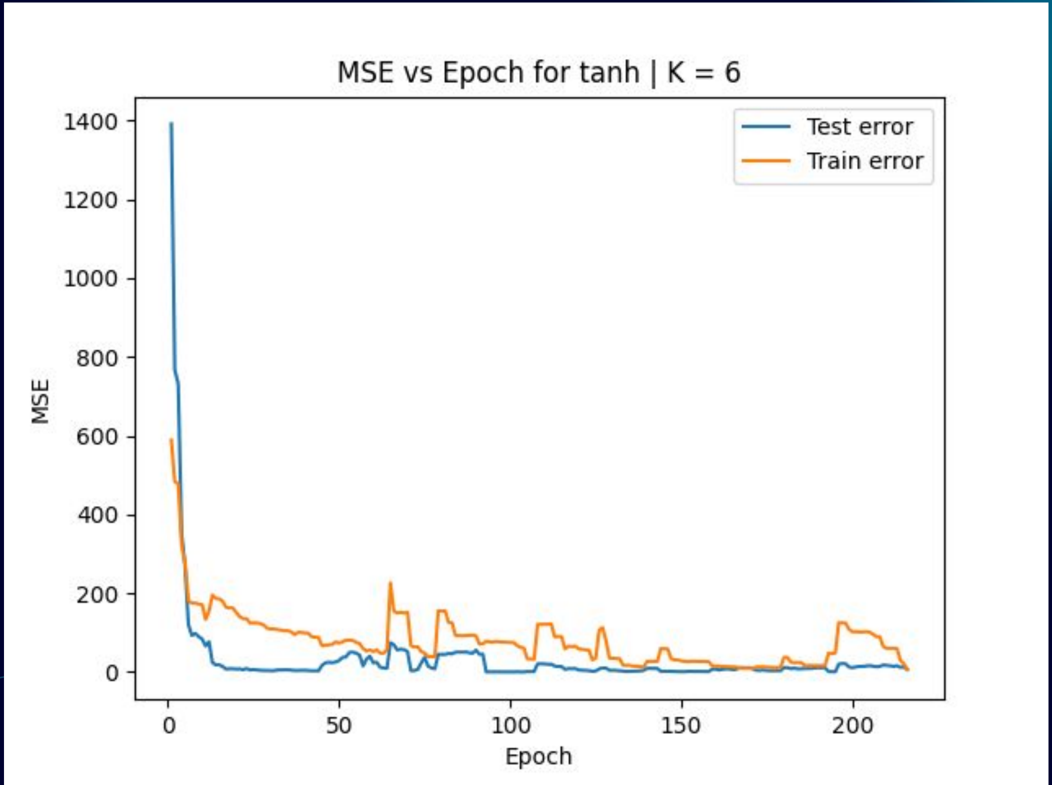
{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 2,
"beta": 1
}

#train = 14

#test = 14

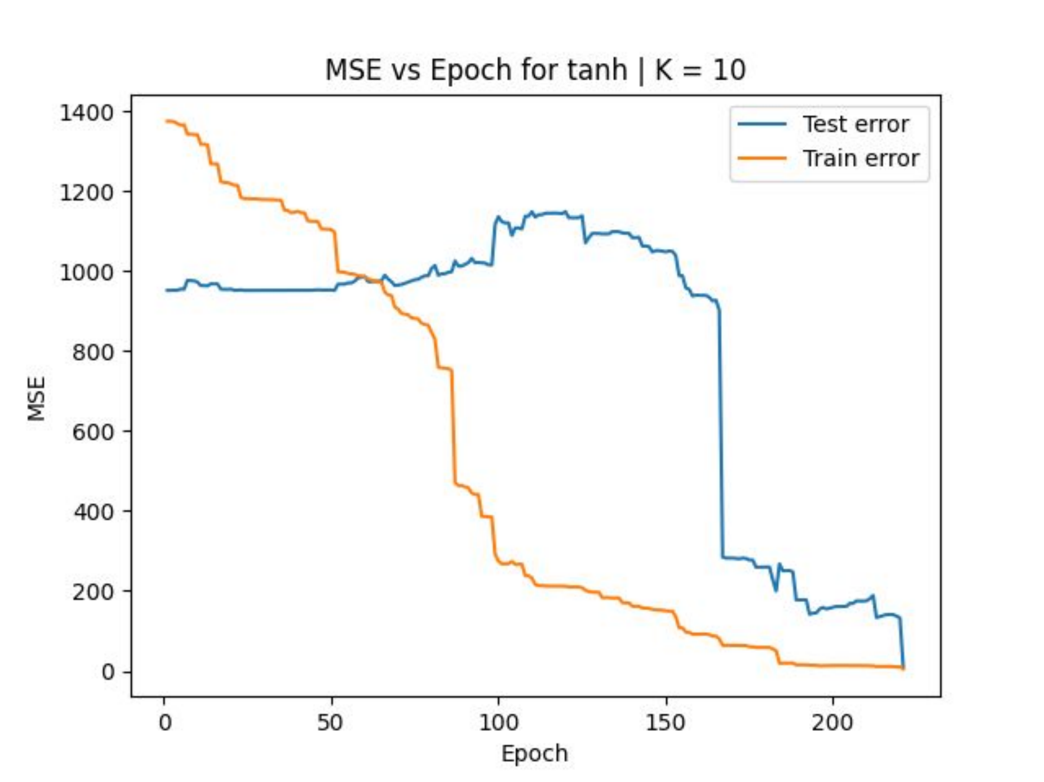
Error
} Error

Tanh



{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 6,
"beta": 1
}
#train = 24
#test = 4

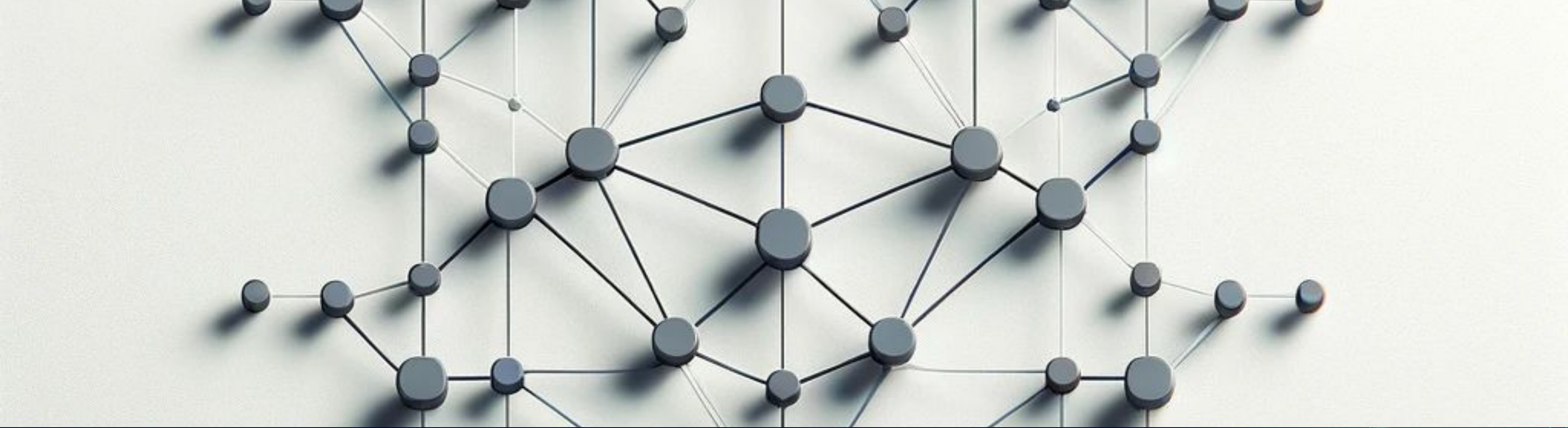
Tanh



{
"limit": 1000,
"learning_rate": 0.05,
"bias": 0,
"epsilon": 0.1,
"k": 10,
"beta": 1
}

#train = 26

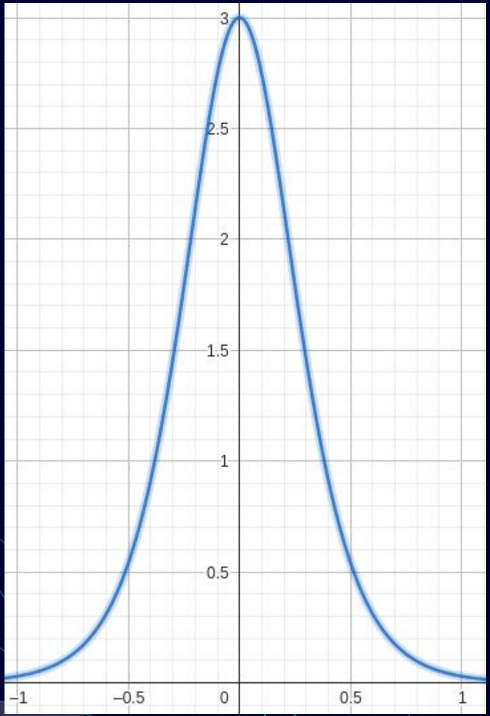
#test = 2



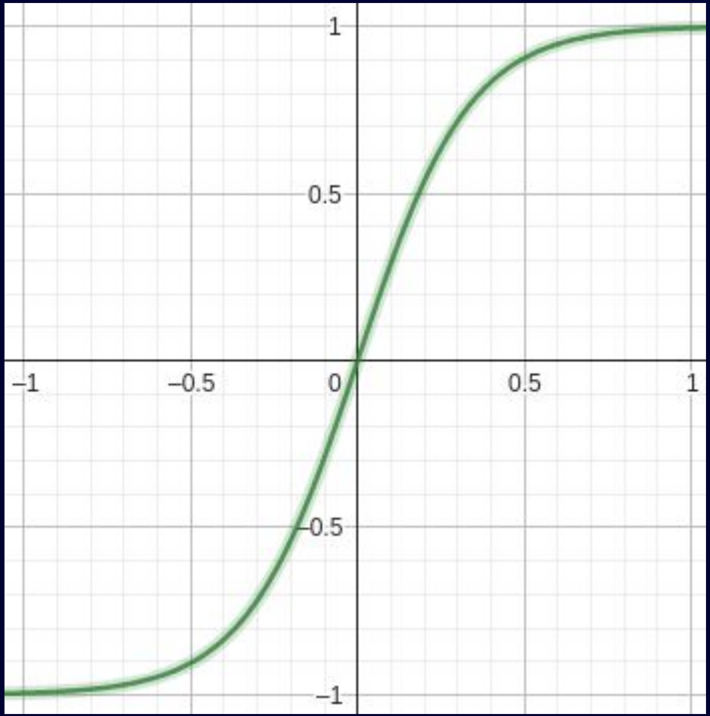
3 Perceptron Multicapa

XOR Multicapa

XOR con multicapa



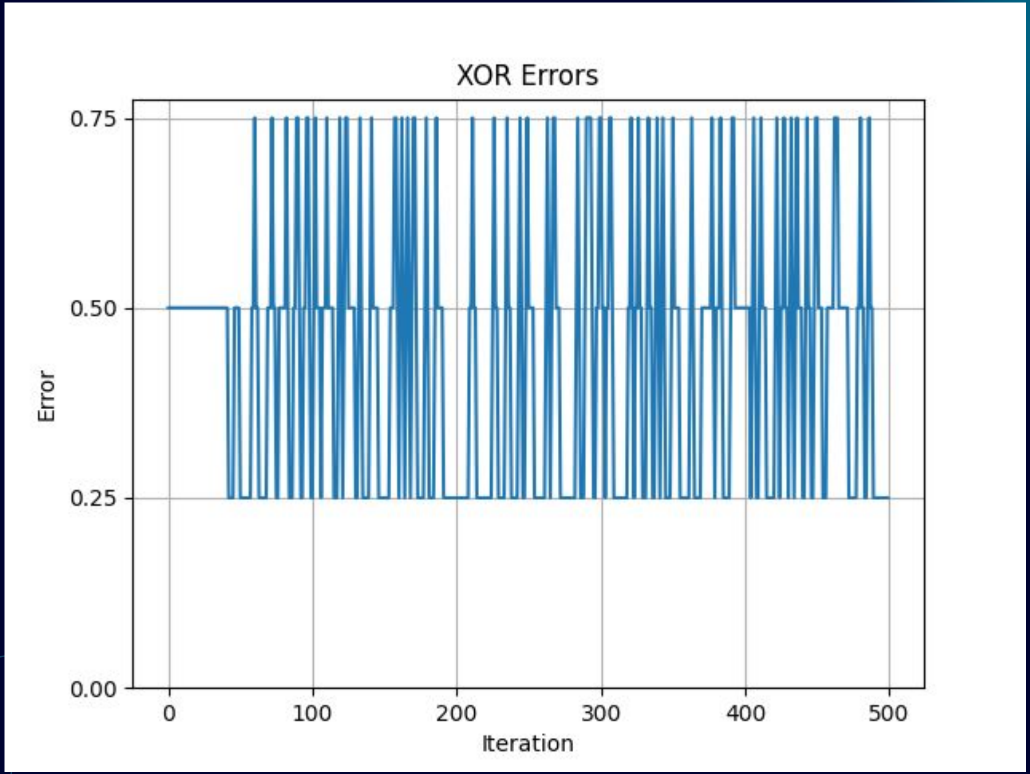
d(tanh(3x))/dx



tanh(3x)

Arquitectura [2,2,2,1]

Recap: XOR con simple escalón
Error vs Epoch



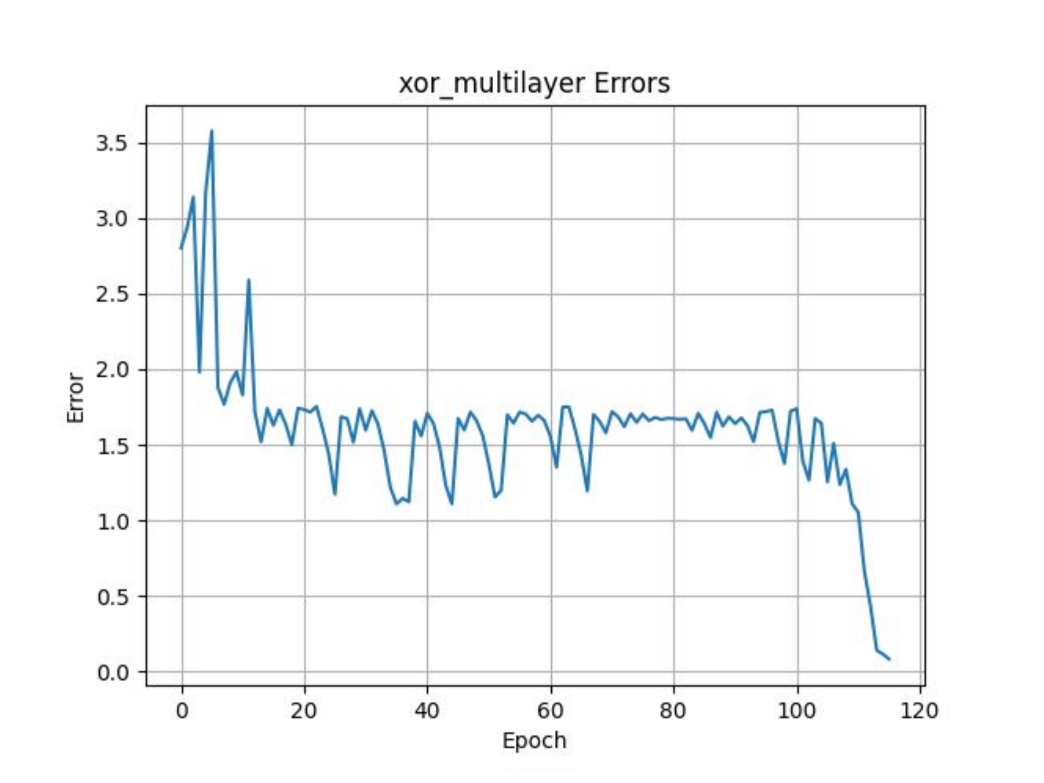
"limit": 1000,

"learning_rate": 0.02,

"bias": 0,

"epsilon": 0.1

XOR con multicapa
Error vs Epoch



Error es el
Mean Square Error

"limit": 500,

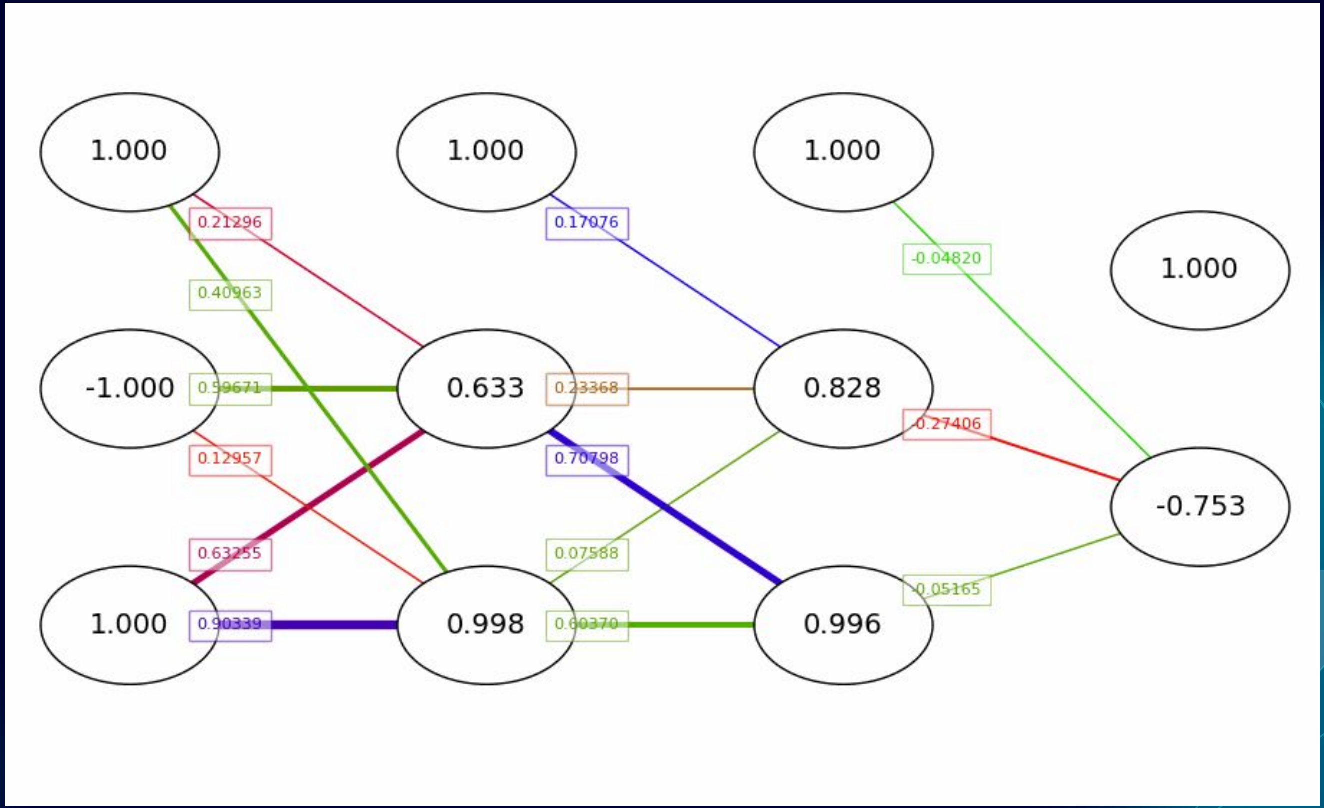
"learning_rate": 0.08,

"bias": 0,

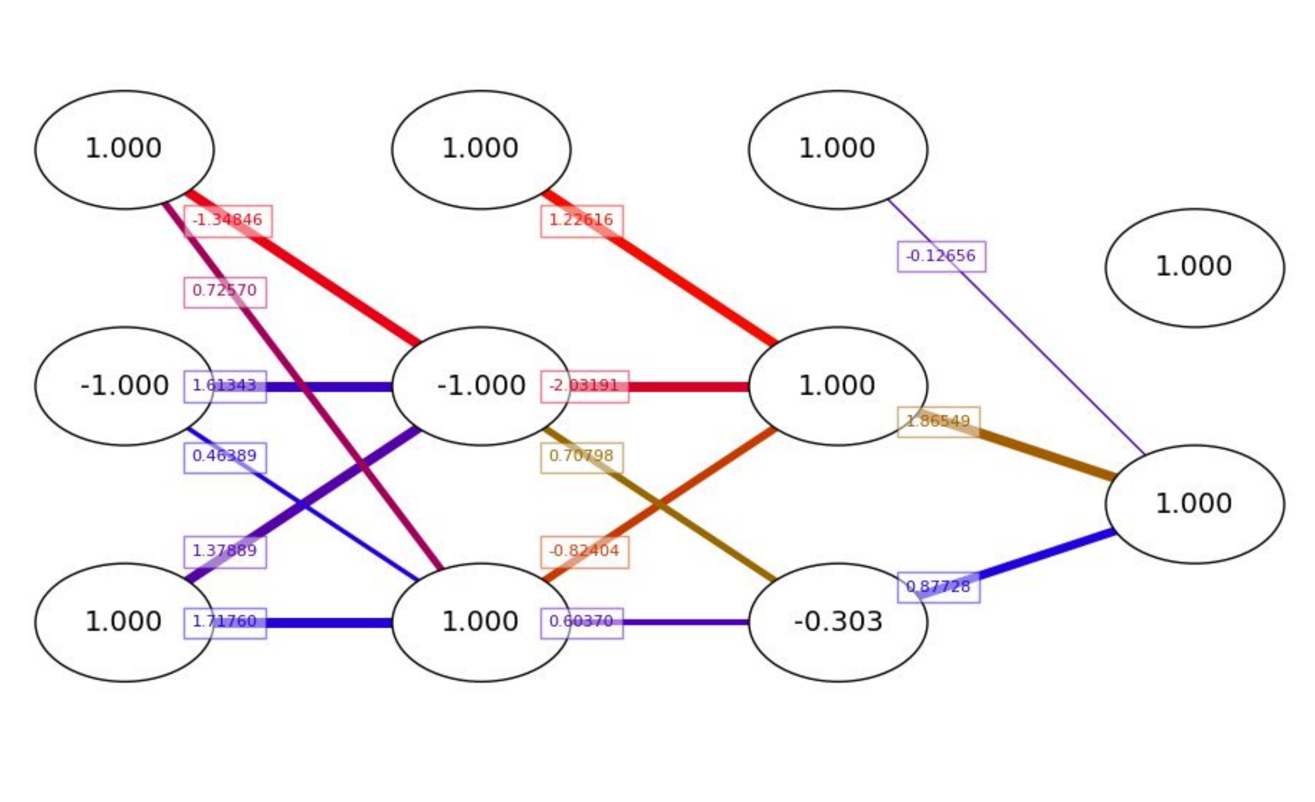
"epsilon": 0.1,

"beta": 3

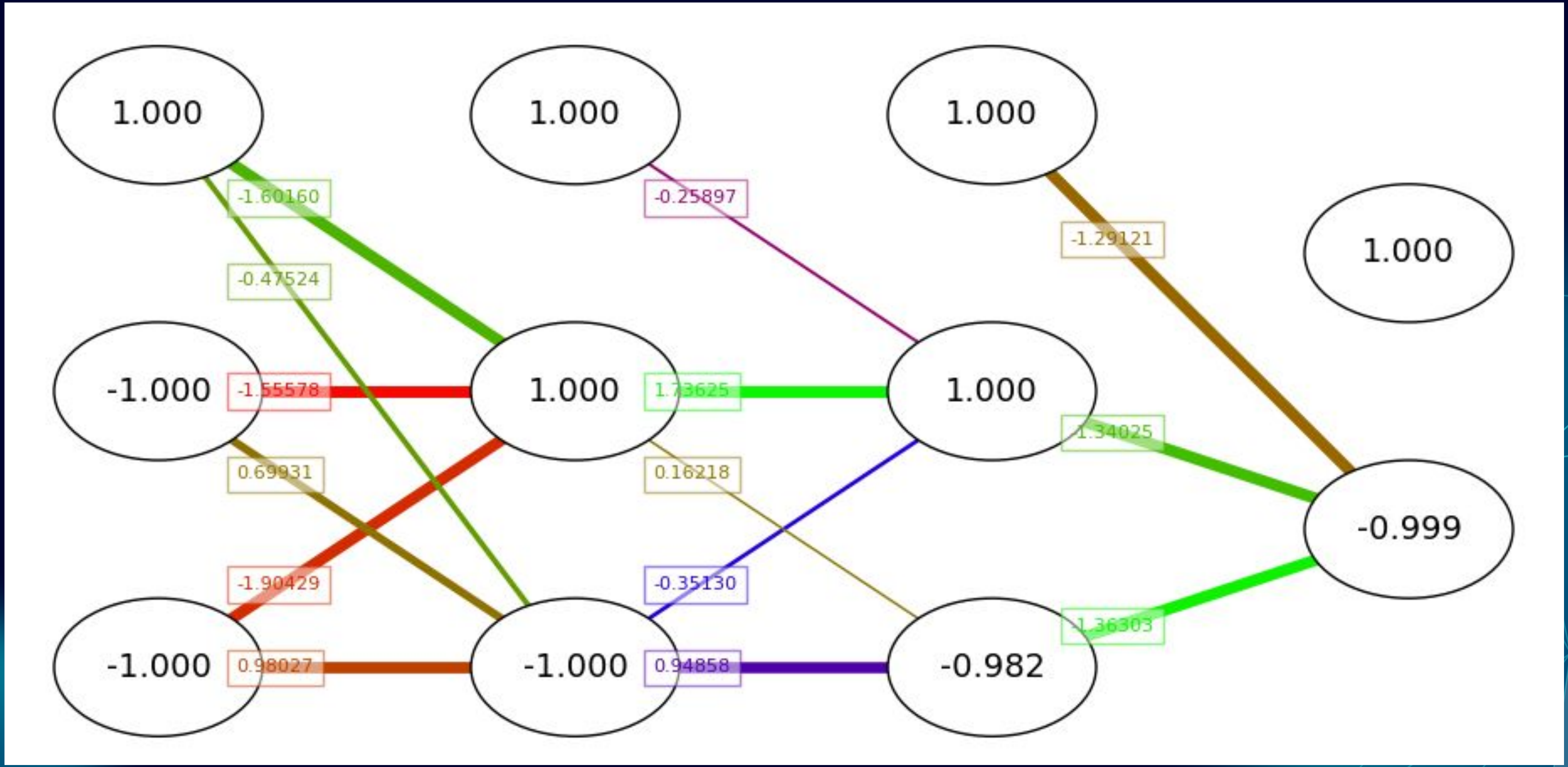
XOR - Funcionamiento de la red



XOR - Final

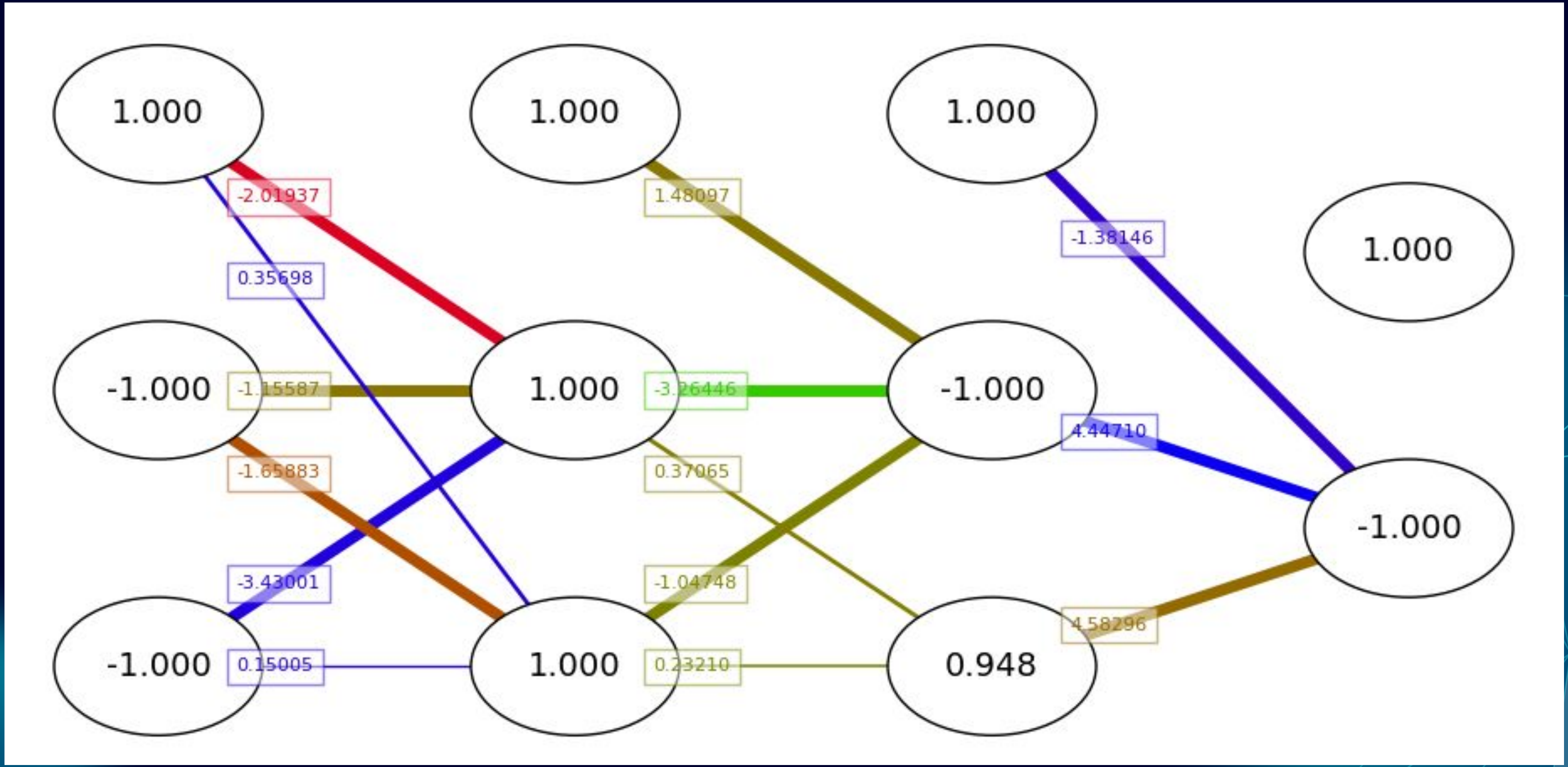

1.000
1.000
1.000
1.000
-1.000
-1.000
1.000
1.000
1.000
1.000
-0.303
-1.34846
0.72570
1.61343
0.46389
1.37889
1.71760
1.22616
-2.03191
0.70798
-0.82404
0.60370
-0.12656
1.86549
0.87728

XOR - Otras soluciones A



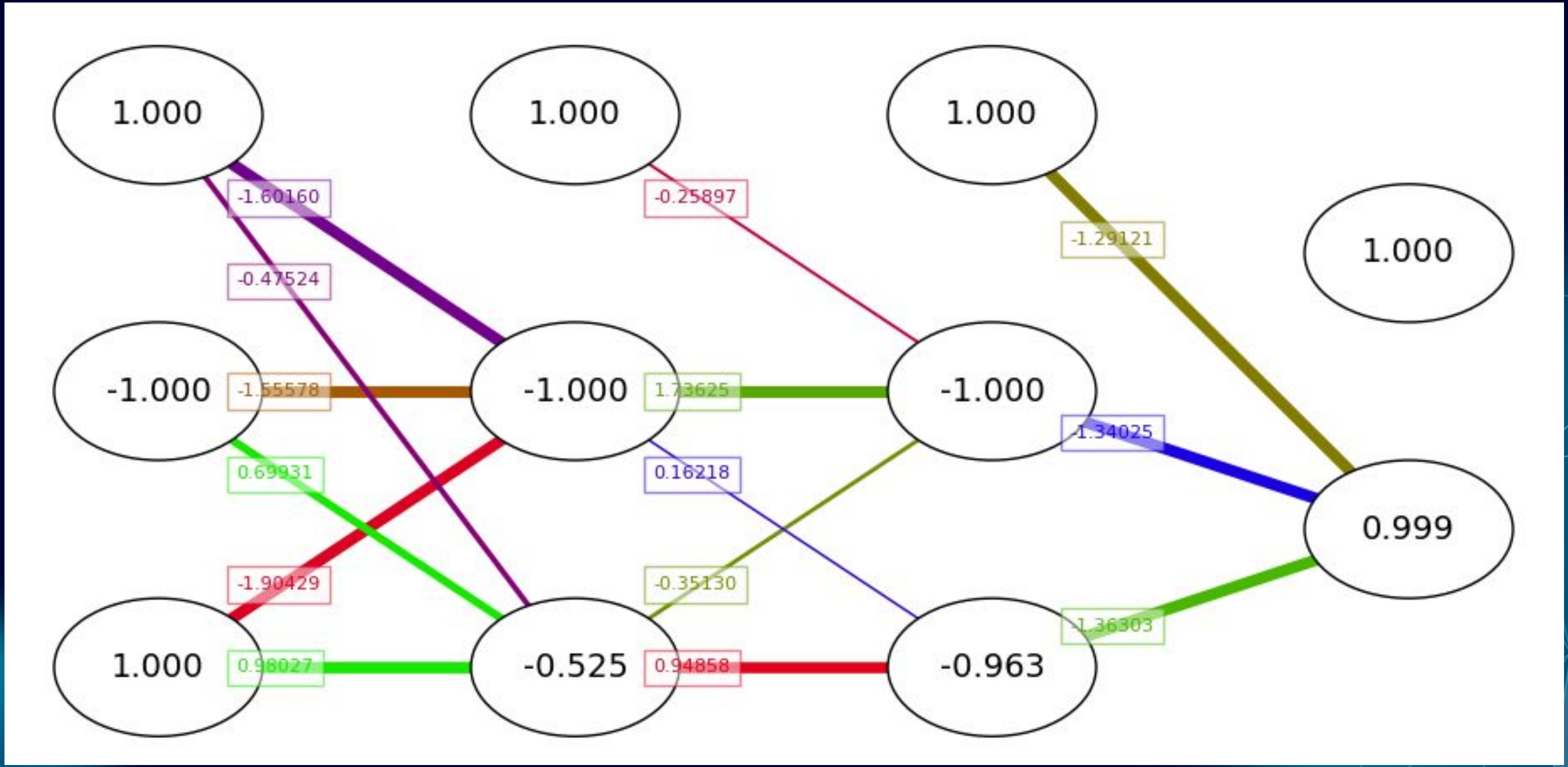
Entrada -1 -1

XOR - Otras soluciones B



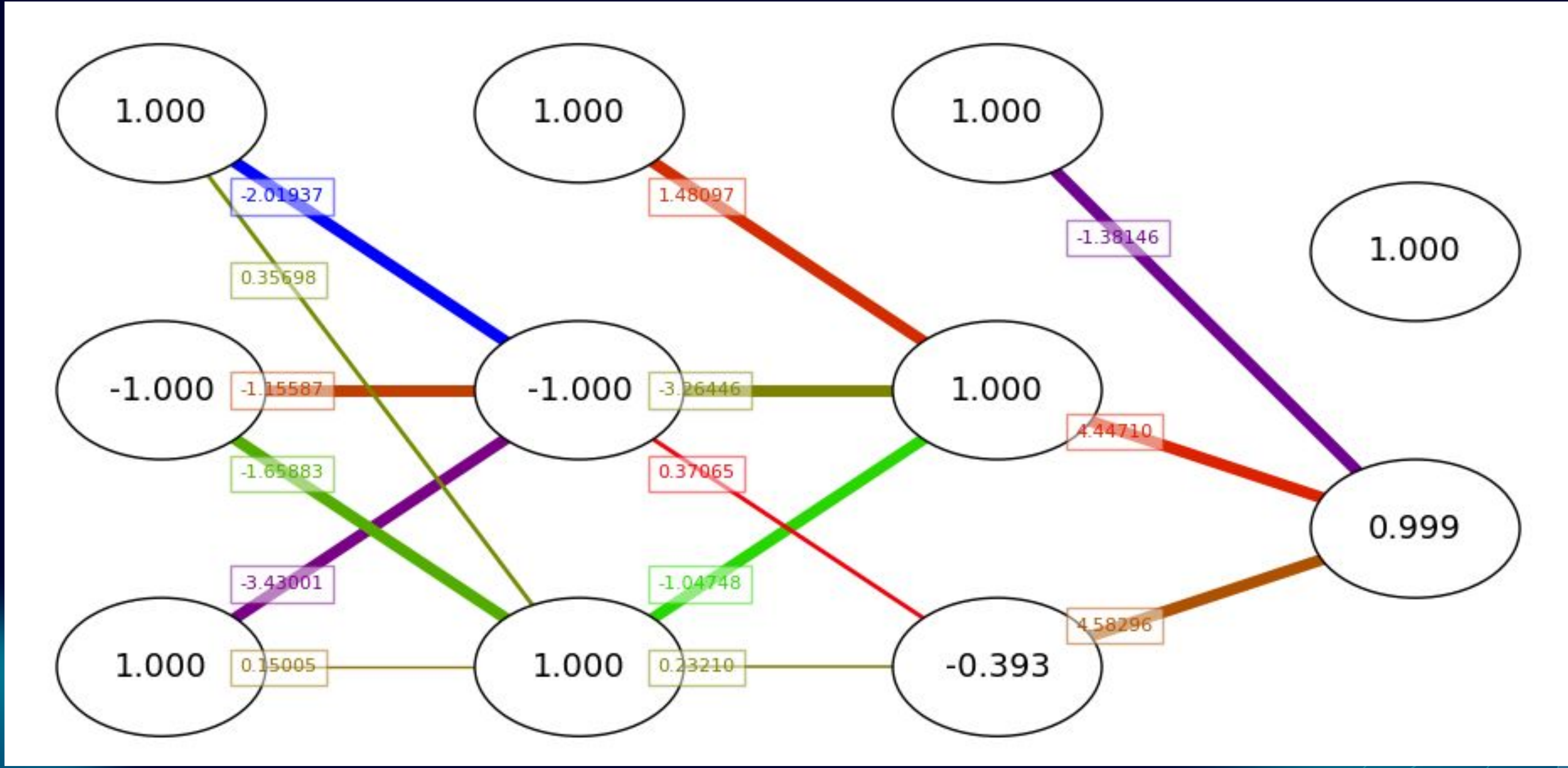
Entrada -1 -1

XOR - Otras soluciones A



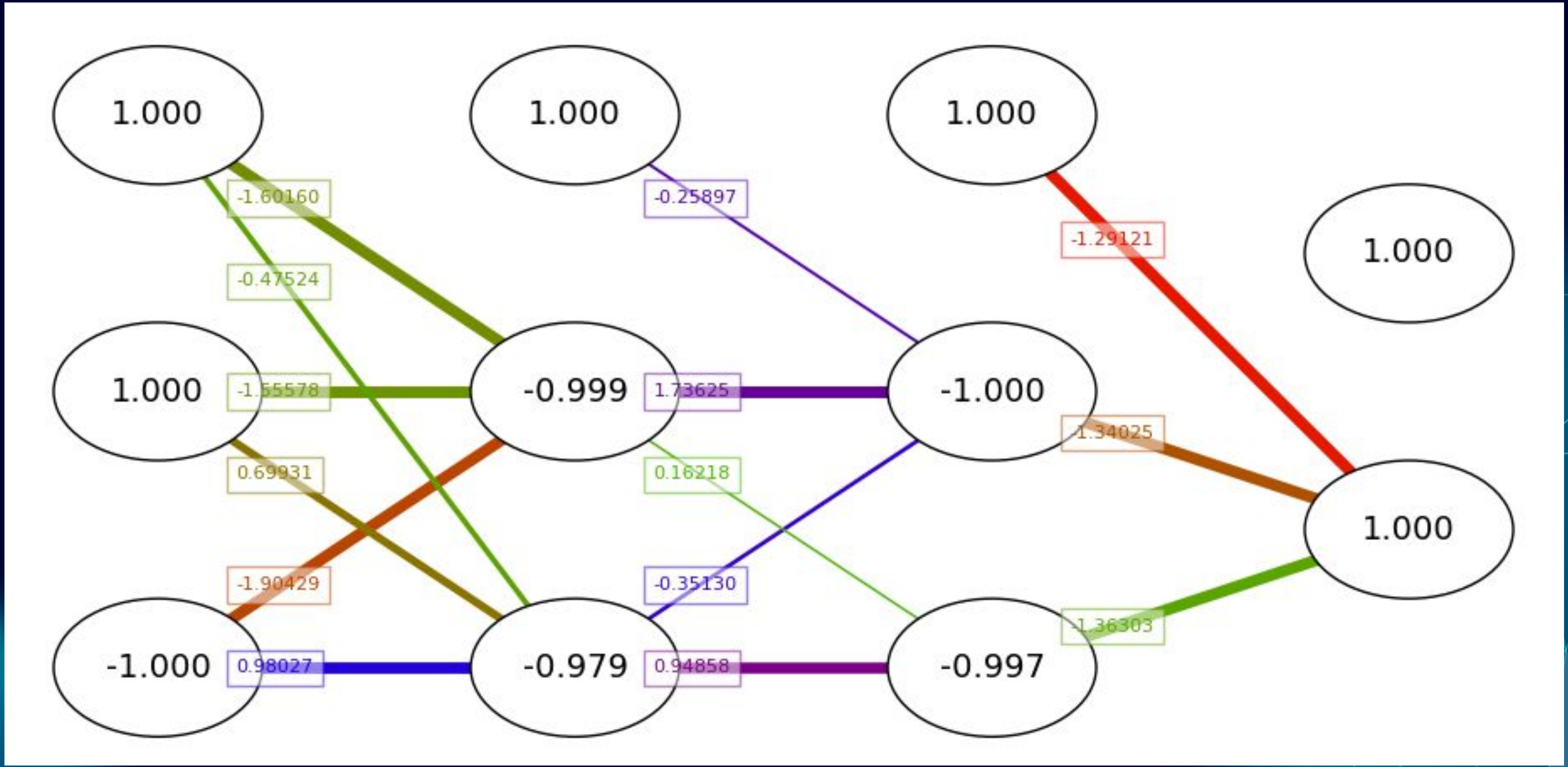
Entrada -1 1

XOR - Otras soluciones B



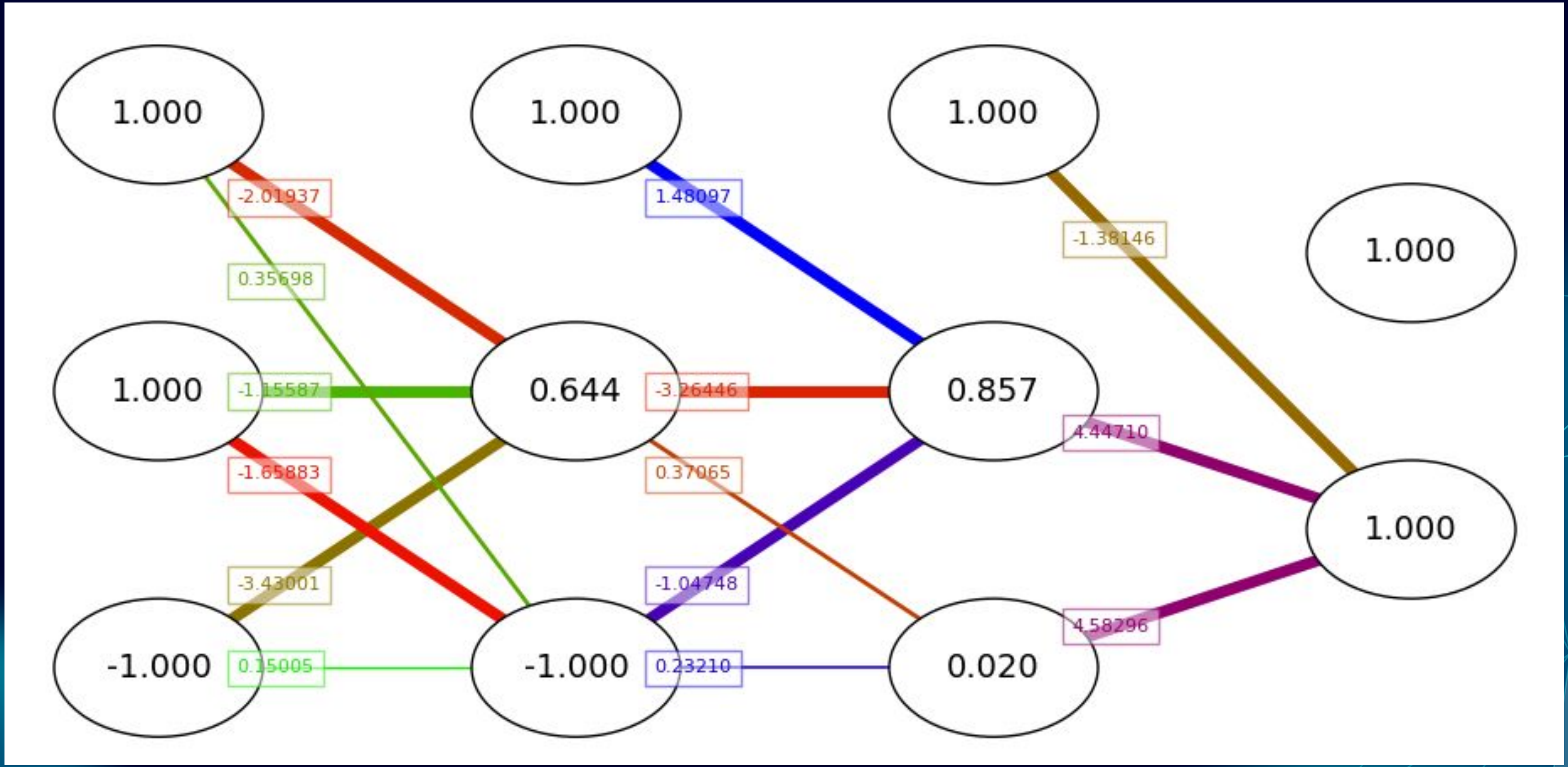
Entrada -1 1

XOR - Otras soluciones A



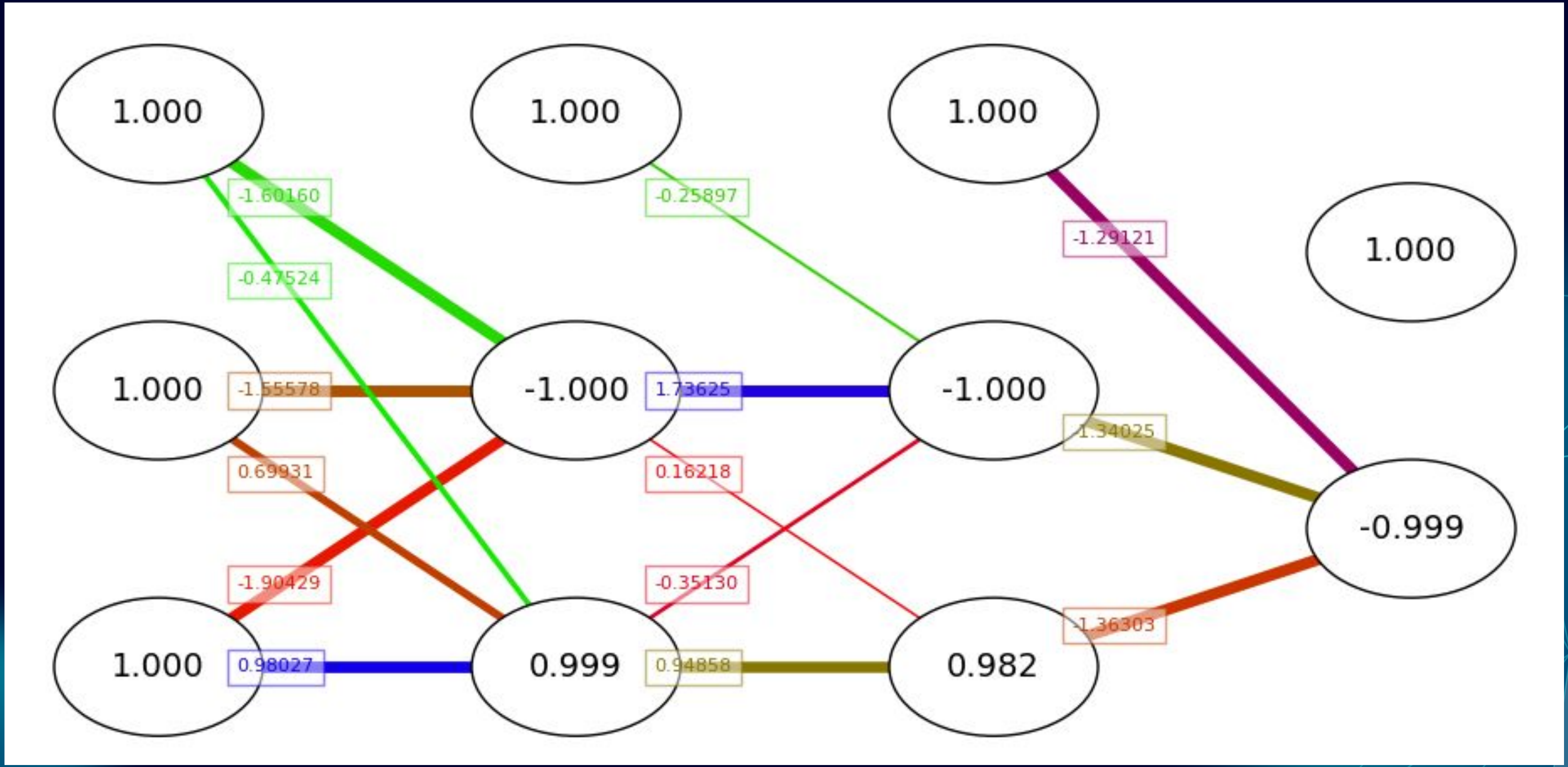
Entrada 1 -1

XOR - Otras soluciones B



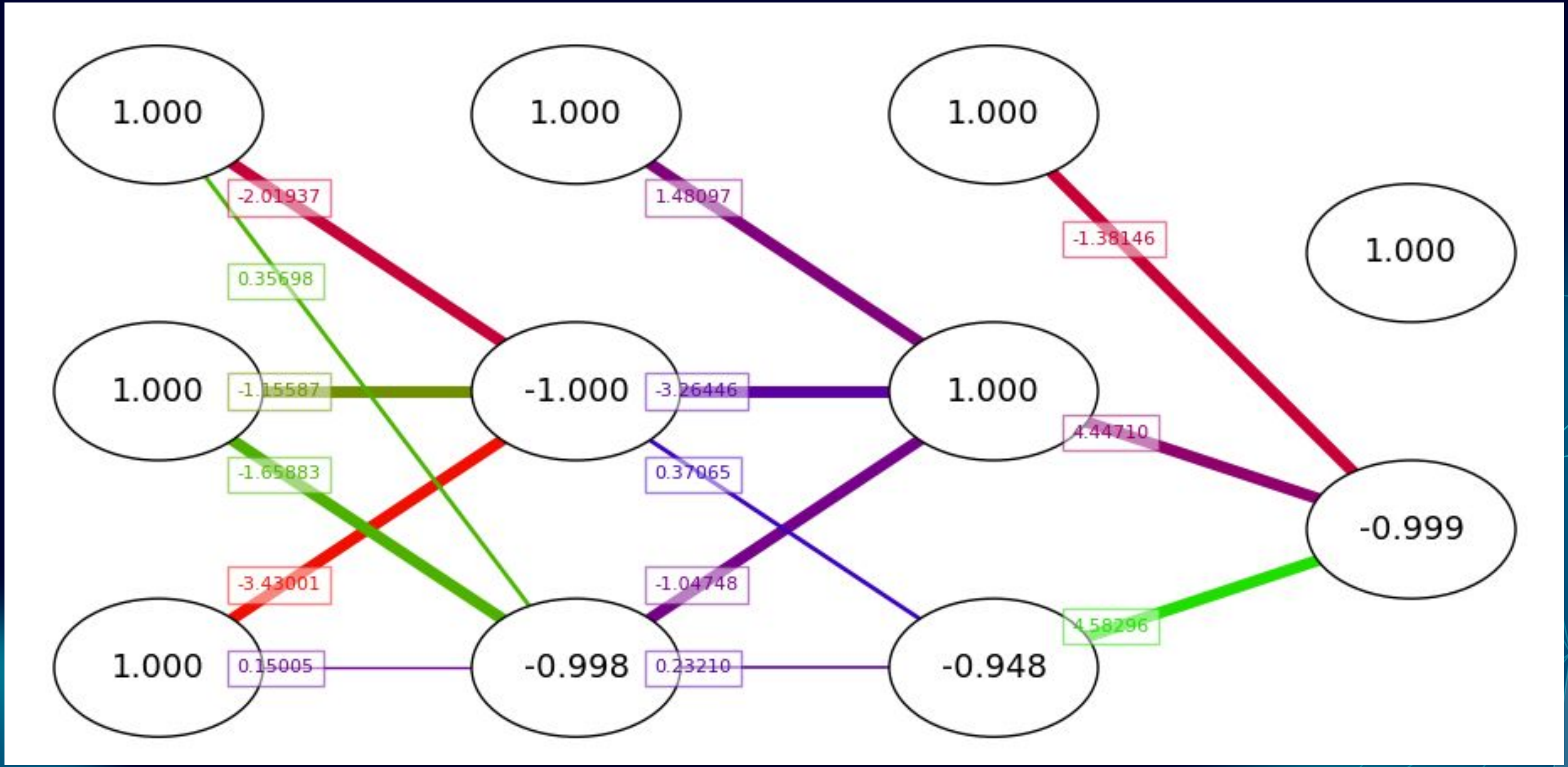
Entrada 1 -1

XOR - Otras soluciones A



Entrada 1 1

XOR - Otras soluciones B



Entrada 11

Paridad

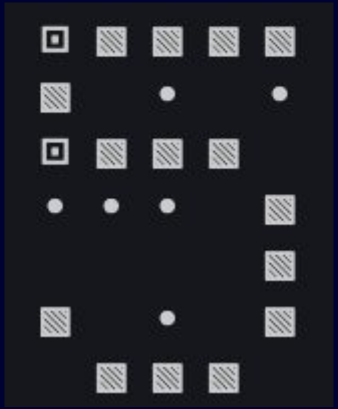
Arquitectura [35,10,2,1]

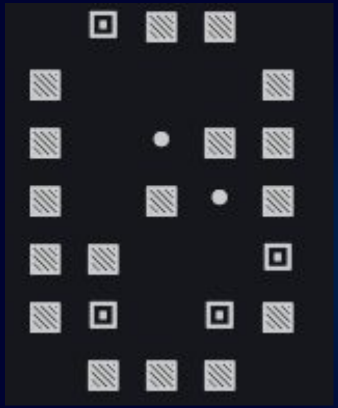
Acerca del ruido

Se utilizó ruido con distribución normal

Se armaron 10 archivos, cada uno con distintas versiones de los dígitos dependiendo del ruido aplicado

: 0 - 0.25
· : 0.25 - 0.5
⊡ : 0.50 - 0.75
▧ : 0.75 - 1





Extra Groups

Cantidad de copias extra por número

#total = 10 + 10 * #extra_groups
#test = floor(#total/k)
#train = #total - #test

Cómo evaluamos métricas

Métricas

Todos los valores se encuentran normalizados

	Par
Expected	0
Output	0

True Negative

Par
1
1

True Positive

Par
0
1

False Positive

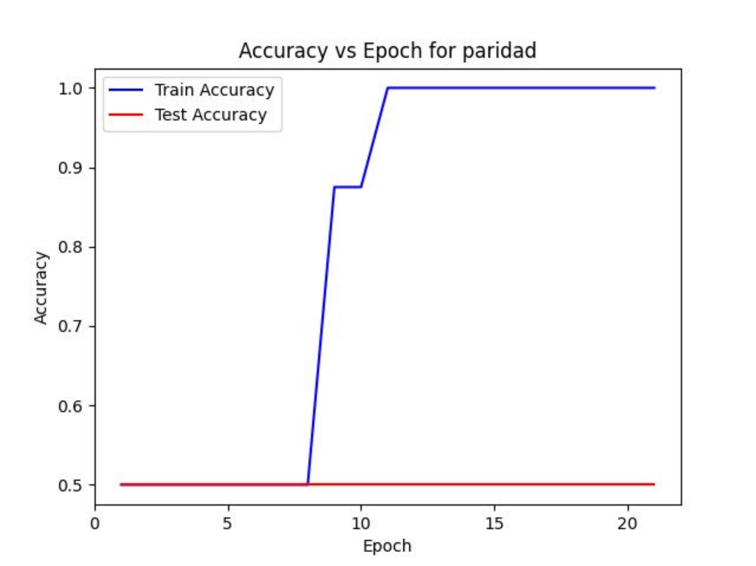
Par
1
0

False Negative

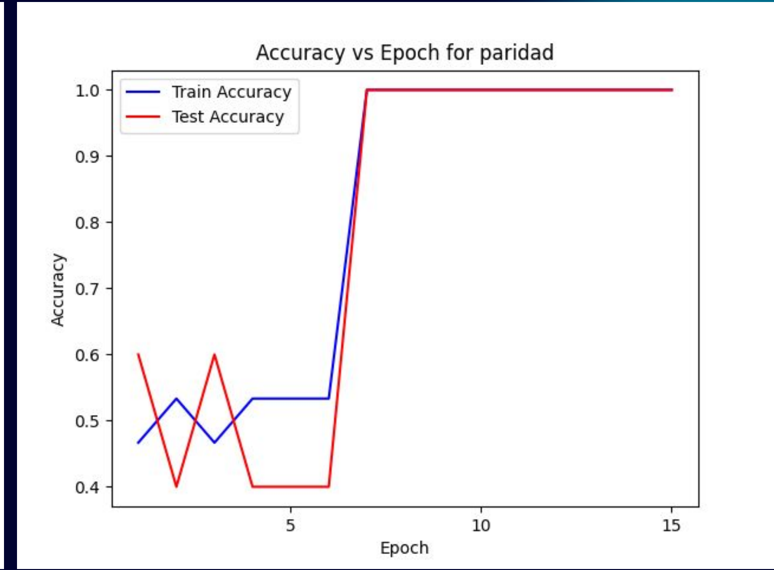
Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Sin extra group: #train = 8 #test = 2



1 extra group: #train 15 #test = 5

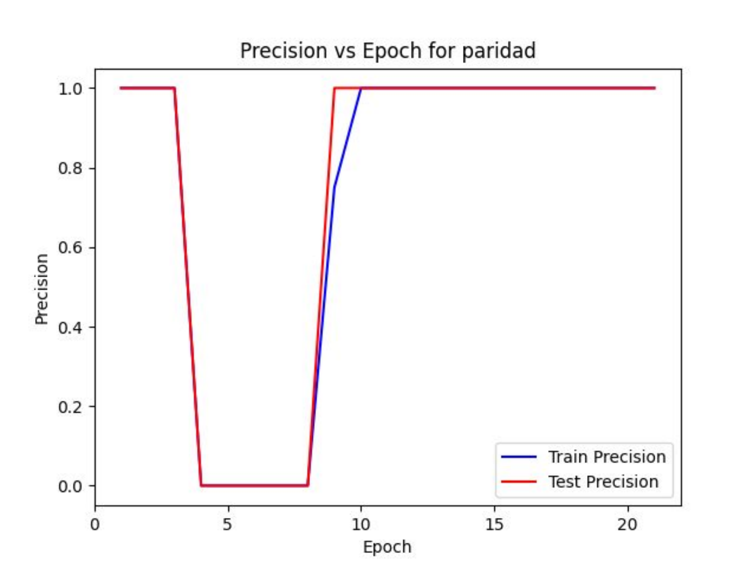


"limit": 500,
"random_start": true,
"learning_rate": 0.02,
"bias": 0,
"epsilon": 0.1,
"k": 4,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8

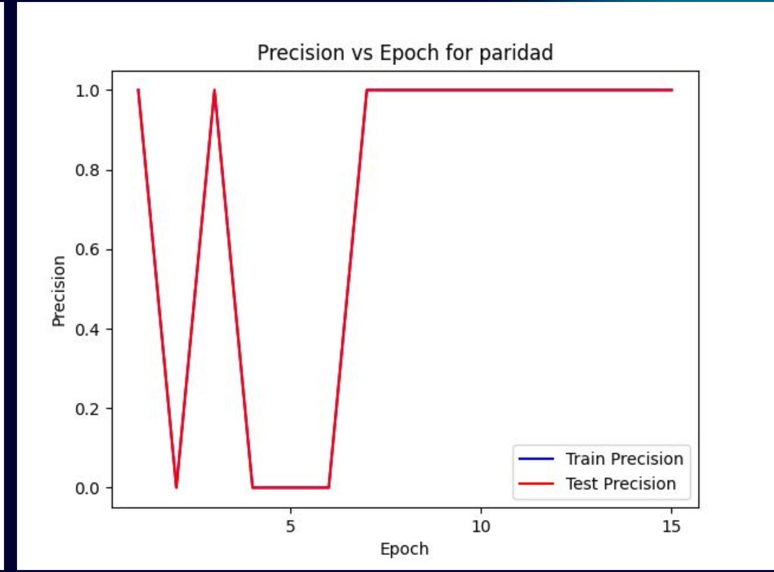
Precision

$$Precision = \frac{TP}{TP + FP}$$

Sin extra group: #train = 8 #test = 2



1 extra group: #train 15 #test = 5



”limit”: 500,

”random_start”: true,

”learning_rate”: 0.02,

”bias”: 0,

”epsilon”: 0.1,

”k”: 4,

”beta”: 1,

”optimizer”: ”adam”,

”b1”: 0.9331338848100159,

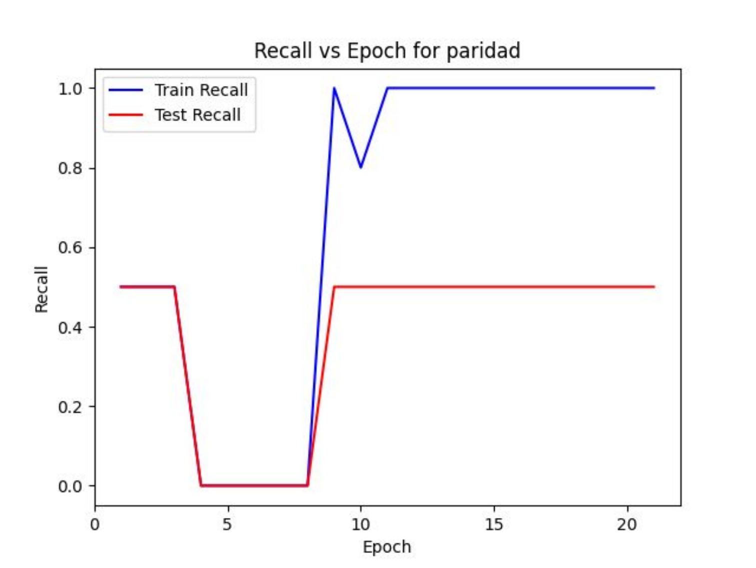
”b2”: 0.9658289155465659,

”e”: 1e-8

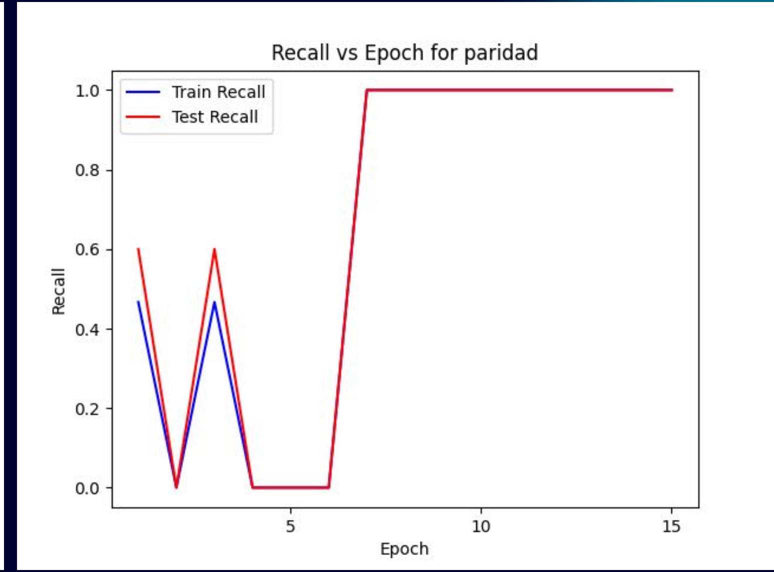
Recall

$$\frac{TP}{TP + FN}$$

Sin extra group: #train = 8 #test = 2



1 extra group: #train 15 #test = 5

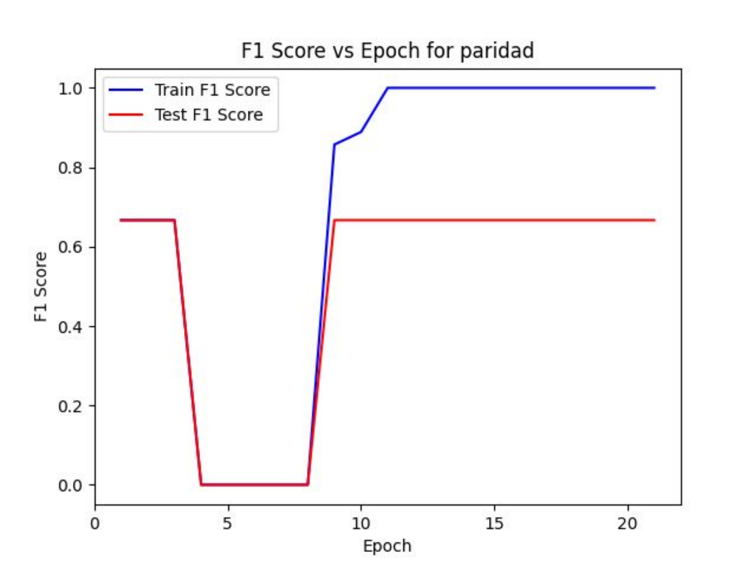


”limit”: 500,
”random_start”: true,
”learning_rate”: 0.02,
”bias”: 0,
”epsilon”: 0.1,
”k”: 4,
”beta”: 1,
”optimizer”: ”adam”,
”b1”: 0.9331338848100159,
”b2”: 0.9658289155465659,
”e”: 1e-8

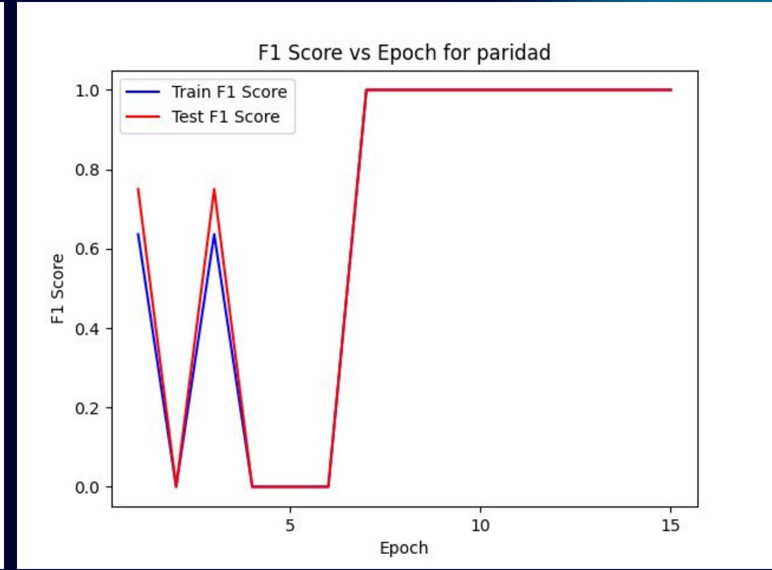
F1 Score

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Sin extra group: #train = 8 #test = 2



1 extra group: #train 15 #test = 5



”limit”: 500,

”random_start”: true,

”learning_rate”: 0.02,

”bias”: 0,

”epsilon”: 0.1,

”k”: 4,

”beta”: 1,

”optimizer”: ”adam”,

”b1”: 0.9331338848100159,

”b2”: 0.9658289155465659,

”e”: 1e-8

Dígitos

Arquitectura [35,10,10,10]

Métricas

	0	1	2	...
Expected	0	1	0	...
Output	0	1	1	...

True Negative

False Positive

True Positive

	...	8	...
Expected	...	1	...
Output	...	0	...

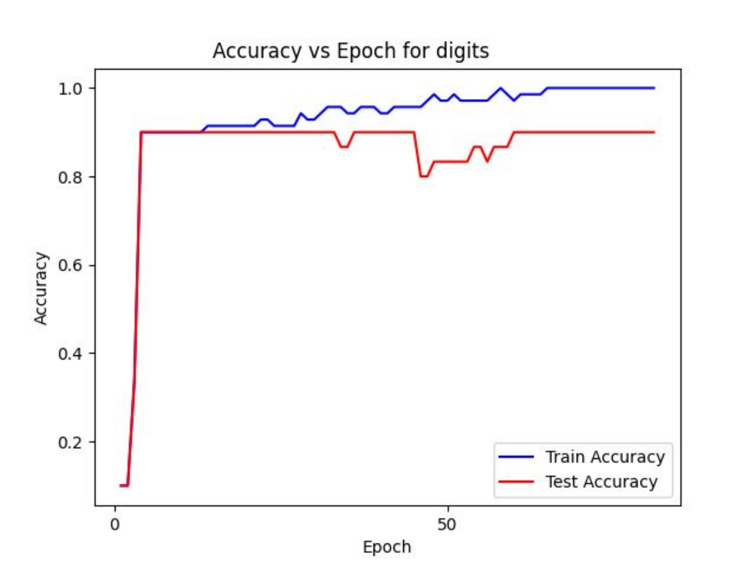
False Negative

constantemente

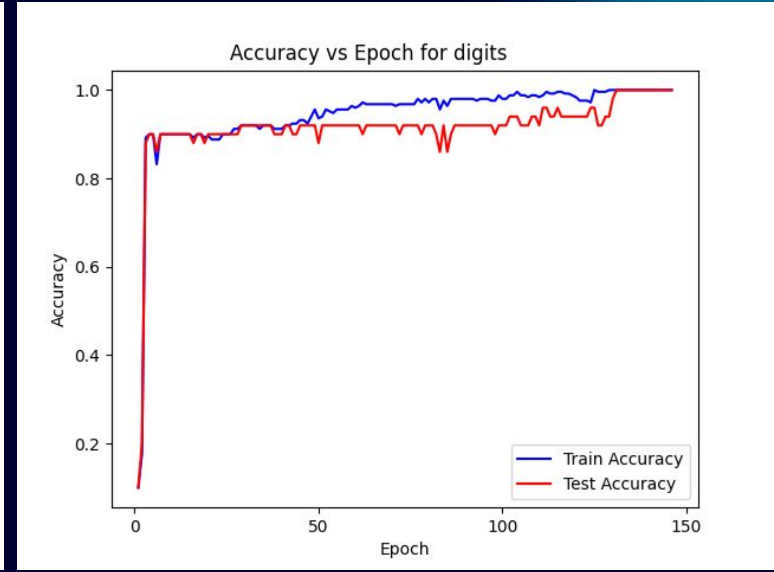
Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Sin extra group



2 extra group

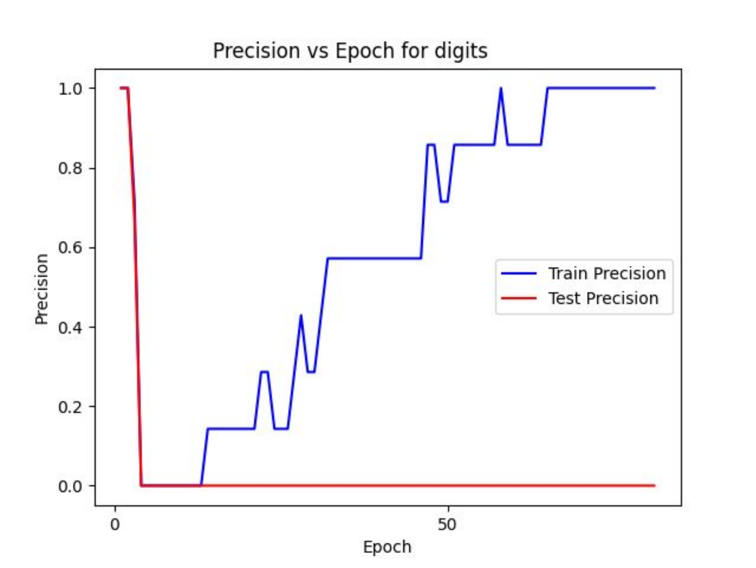


"limit": 1000,
"random_start": true,
"learning_rate": 0.02,
"bias": 0,
"epsilon": 0.1,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
"batch_size": 10

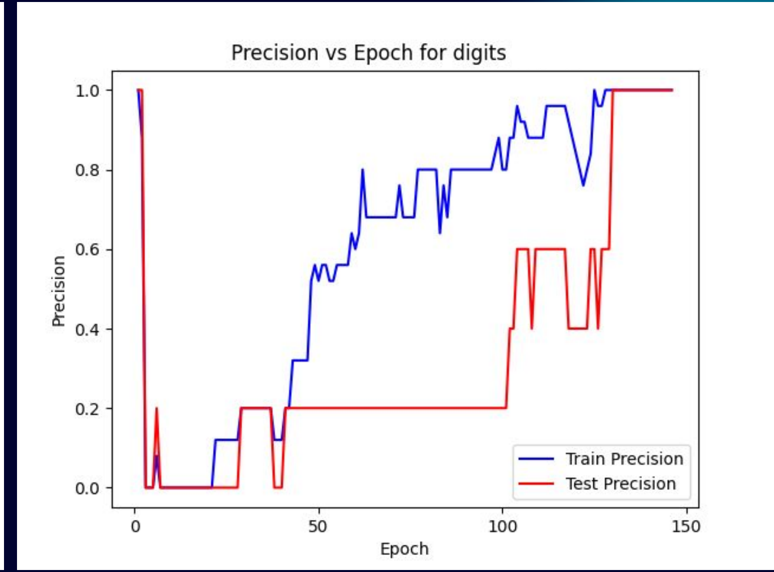
Precision

$$Precision = \frac{TP}{TP + FP}$$

Sin extra group



2 extra group

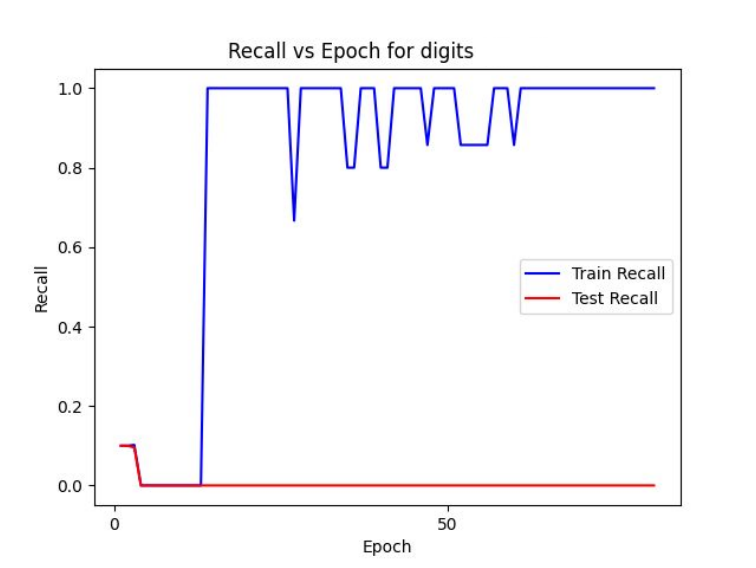


"limit": 1000,
"random_start": true,
"learning_rate": 0.02,
"bias": 0,
"epsilon": 0.1,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
"batch_size": 10

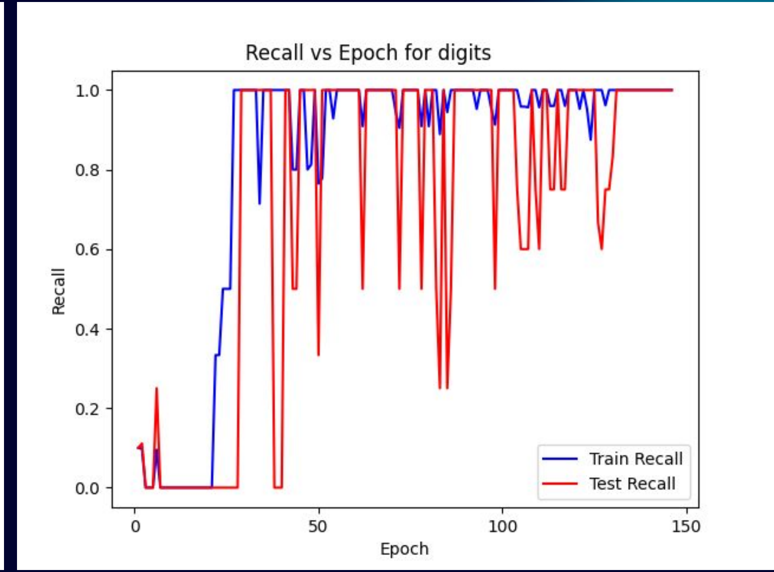
Recall

$$Recall = \frac{TP}{TP + FN}$$

Sin extra group



2 extra group

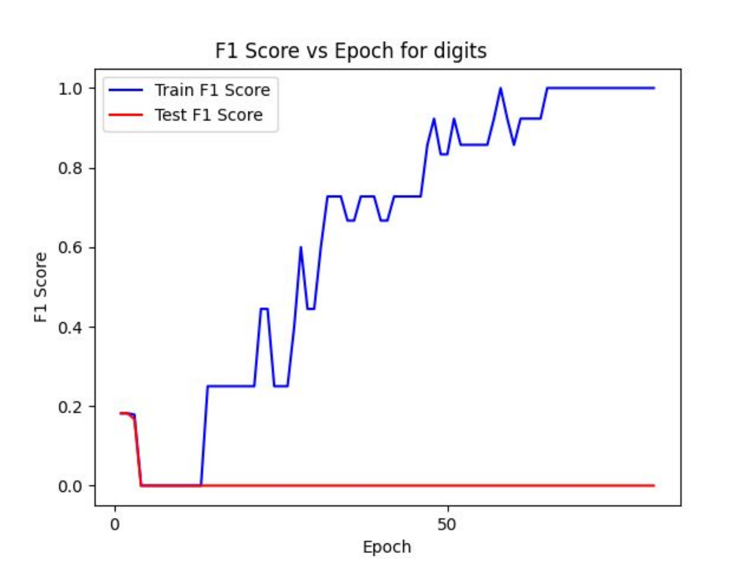


"limit": 1000,
"random_start": true,
"learning_rate": 0.02,
"bias": 0,
"epsilon": 0.1,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
"batch_size": 10

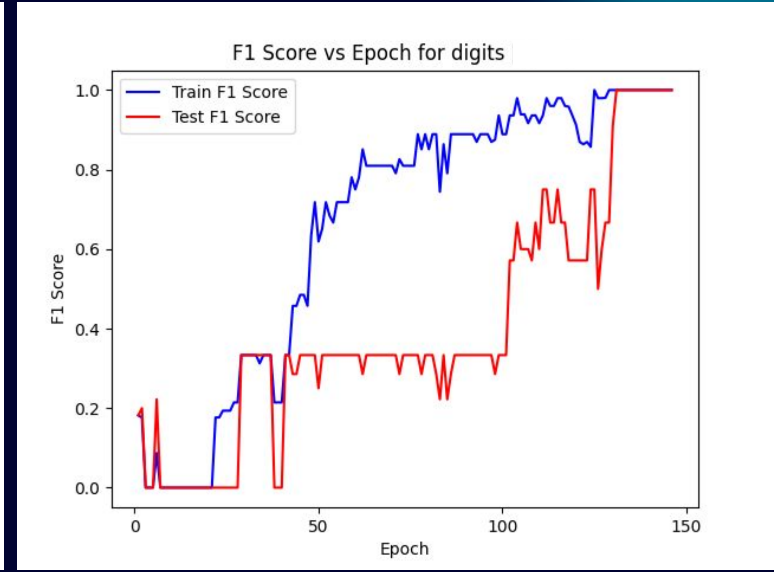
F1 Score

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Sin extra group



2 extra group



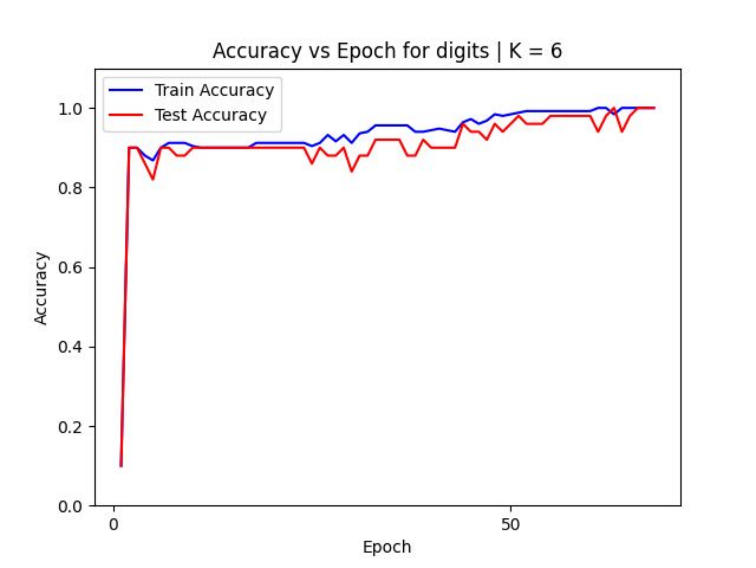
"limit": 1000,
"random_start": true,
"learning_rate": 0.02,
"bias": 0,
"epsilon": 0.1,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
"batch_size": 10

ADAM vs GDS

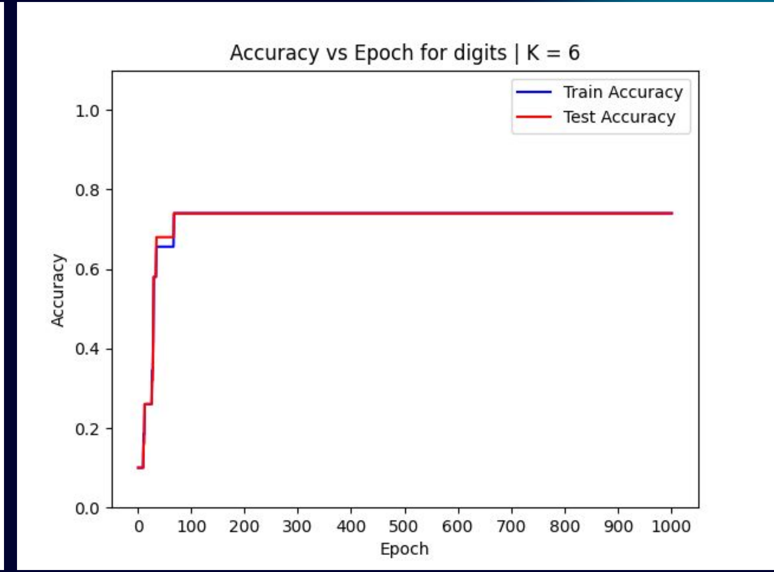
Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

ADAM



GDS



"optimizer": "adam",

"b1": 0.9331338848100159,

"b2": 0.9658289155465659,

"e": 1e-8,

"limit": 1000,

"random_start": true,

"learning_rate": 0.08,

"bias": 0,

"epsilon": 0.1,

"extra_groups": 2,

"k": 6,

"beta": 1,

"batch_size": 10

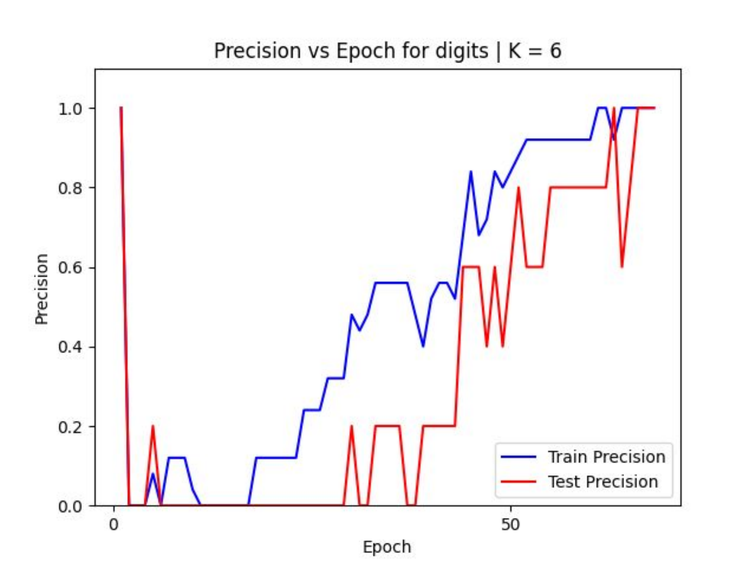
#train = 25

#test = 5

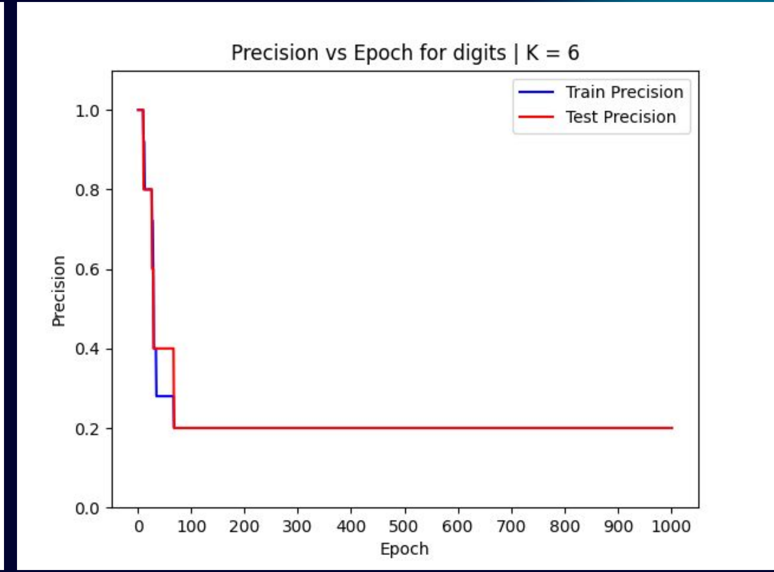
Precision

$$Precision = \frac{TP}{TP + FP}$$

ADAM



GDS



”optimizer”: ”adam”,

”b1”: 0.9331338848100159,

”b2”: 0.9658289155465659,

”e”: 1e-8,

”limit”: 1000,

”random_start”: true,

”learning_rate”: 0.08,

”bias”: 0,

”epsilon”: 0.1,

”extra_groups”: 2,

”k”: 6,

”beta”: 1,

”batch_size”: 10

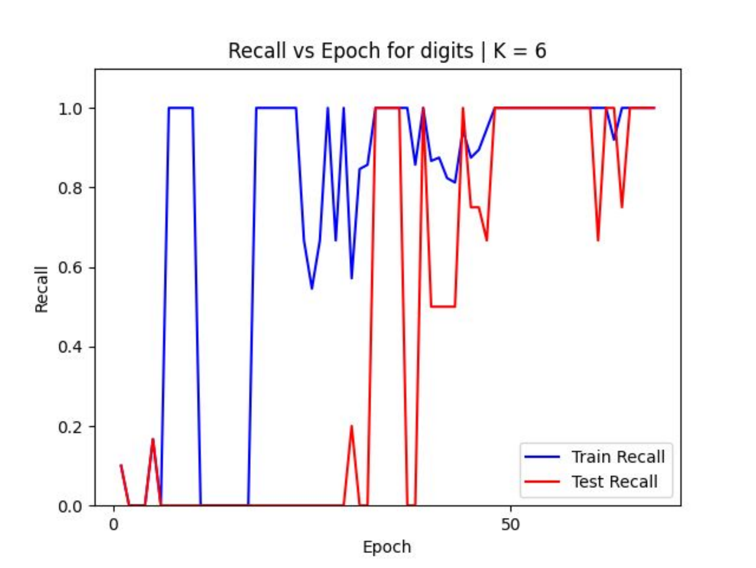
#train = 25

#test = 5

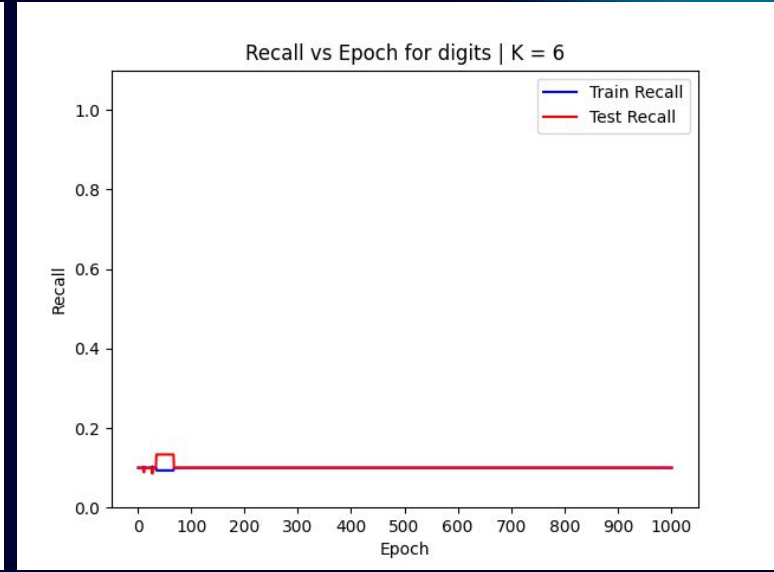
Recall

$$\text{Recall}$$
$$\frac{TP}{TP + FN}$$

ADAM



GDS



”optimizer”: ”adam”,

”b1”: 0.9331338848100159,

”b2”: 0.9658289155465659,

”e”: 1e-8,

”limit”: 1000,

”random_start”: true,

”learning_rate”: 0.08,

”bias”: 0,

”epsilon”: 0.1,

”extra_groups”: 2,

”k”: 6,

”beta”: 1,

”batch_size”: 10

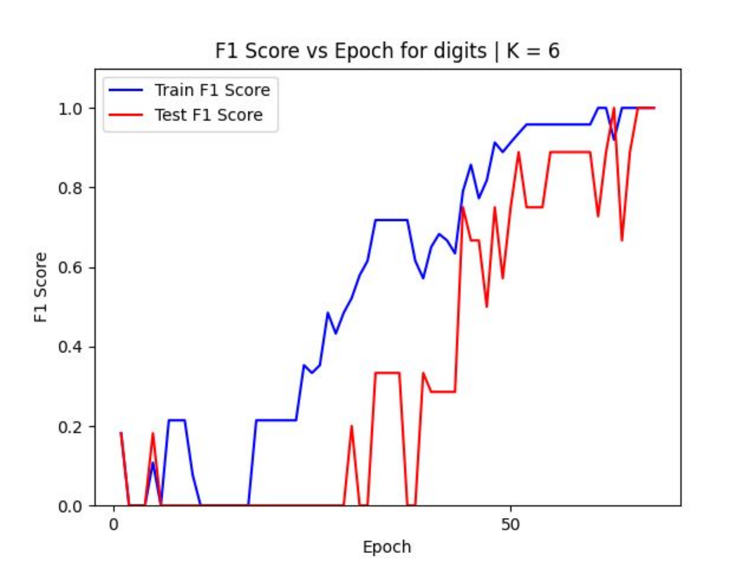
#train = 25

#test = 5

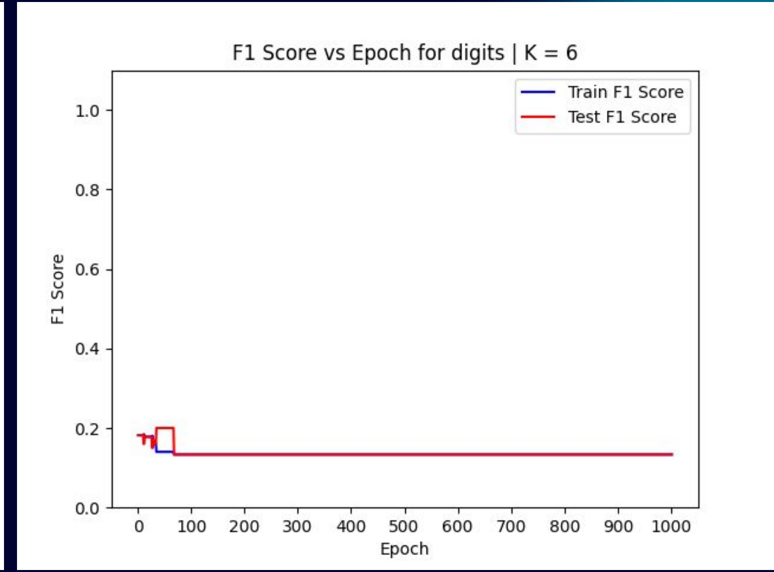
F1 Score

$$F1 - Score$$
$$\frac{2 * Precision * Recall}{Precision + Recall}$$

ADAM



GDS



”optimizer”: ”adam”,

”b1”: 0.9331338848100159,

”b2”: 0.9658289155465659,

”e”: 1e-8,

”limit”: 1000,

”random_start”: true,

”learning_rate”: 0.08,

”bias”: 0,

”epsilon”: 0.1,

”extra_groups”: 2,

”k”: 6,

”beta”: 1,

”batch_size”: 10

#train = 25

#test = 5

ONLINE vs BATCH

ONLINE vs BATCH: complejidad

#train = floor((10 + 10 * #extra_groups) * (k-1)/k)

En online:
#propagaciones = #epochs

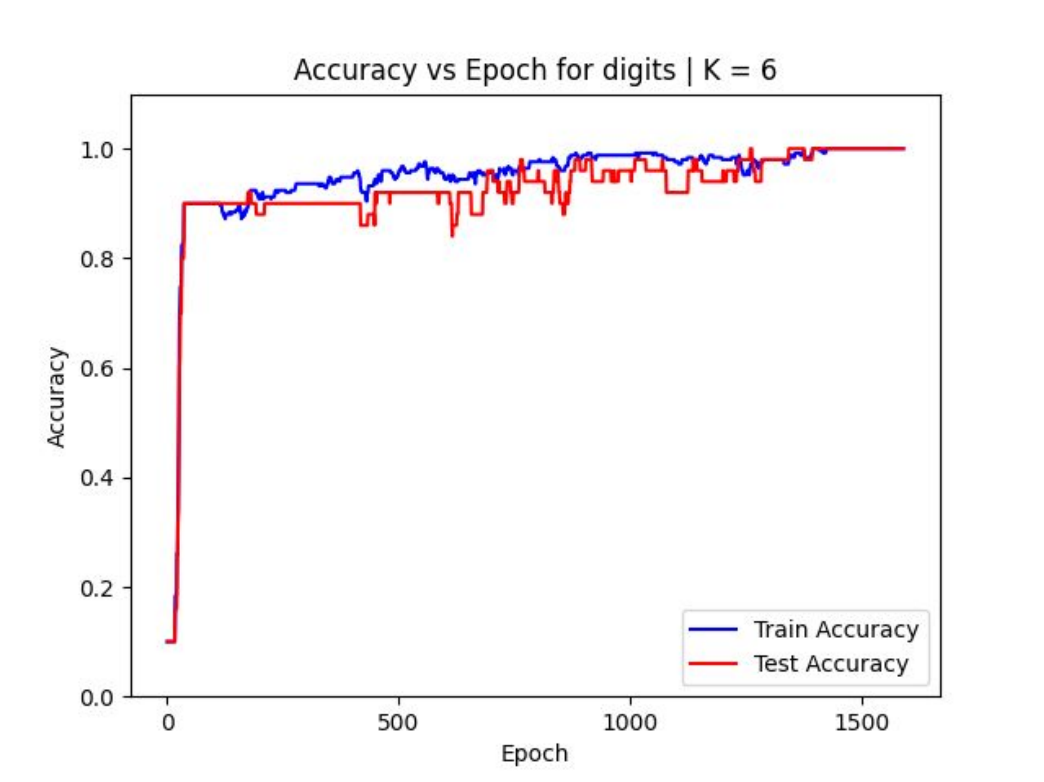
En batch
#propagaciones = #epochs * #train

En general
#calculo_costo = #epochs

Accuracy

Online

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$



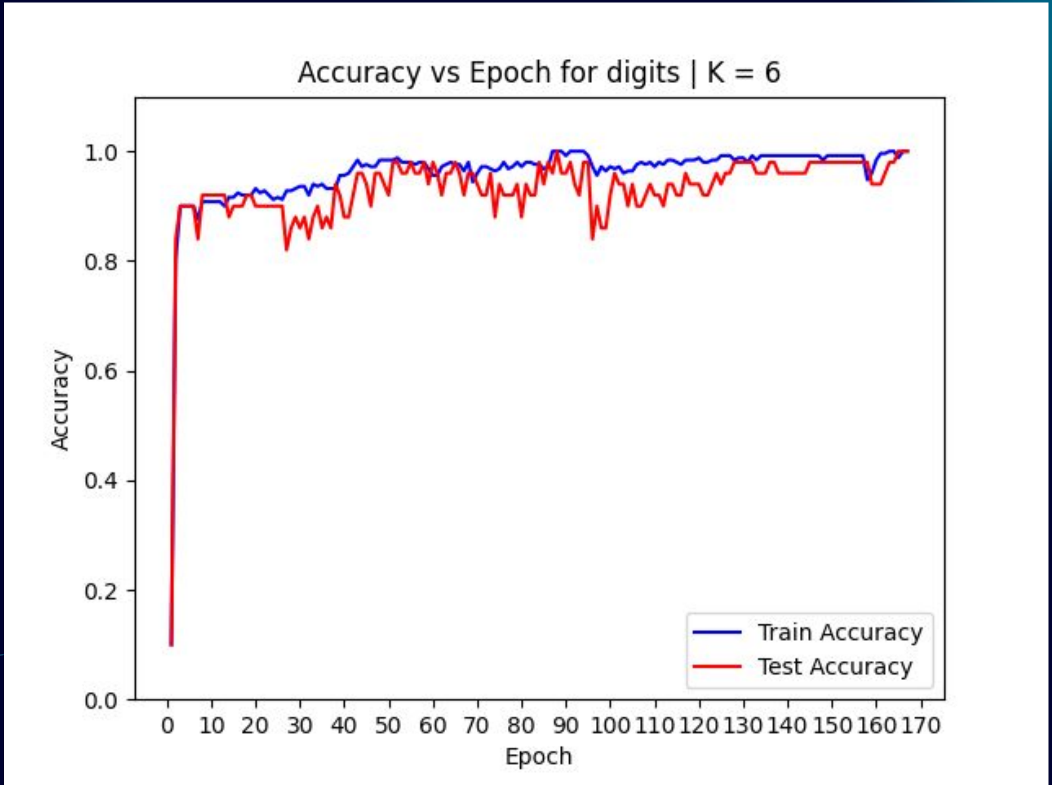
"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,

#train = 25
#test = 5

Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Batch

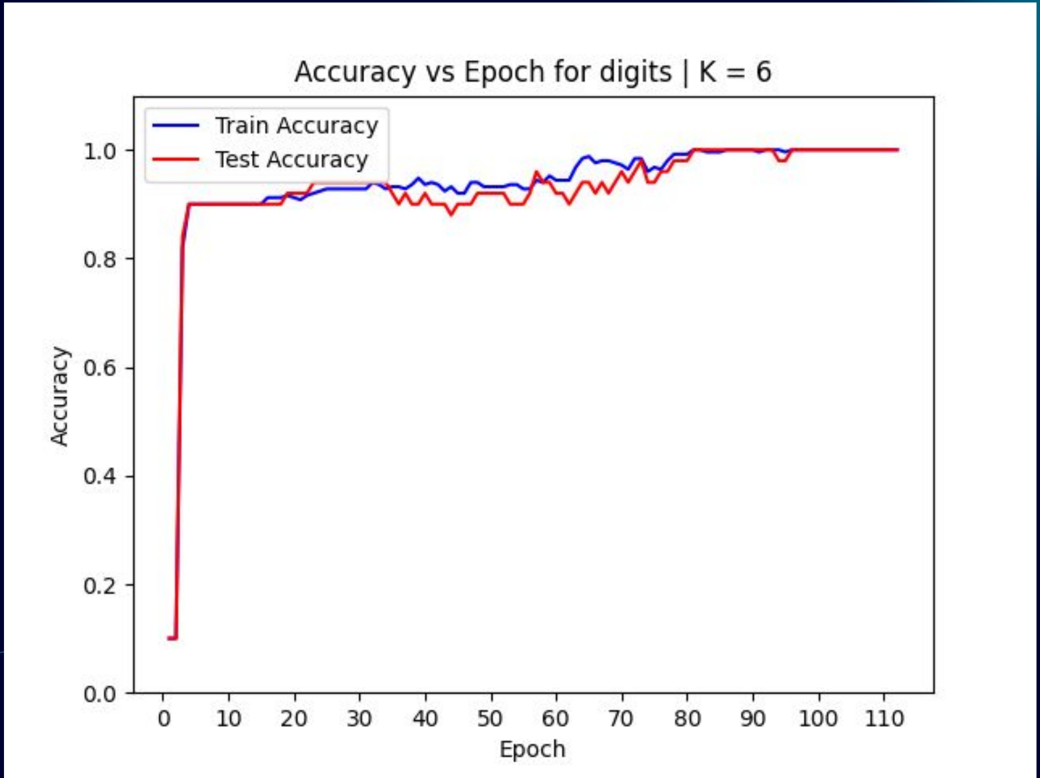


”limit”: 2000,
”random_start”: true,
”learning_rate”: 0.08,
”bias”: 0,
”epsilon”: 0.1,
”extra_groups”: 2,
”k”: 6,
”beta”: 1,
”optimizer”: ”adam”,
”b1”: 0.9331338848100159,
”b2”: 0.9658289155465659,
”e”: 1e-8,
#train = 25
#test = 5

Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Semi-Batch (10)

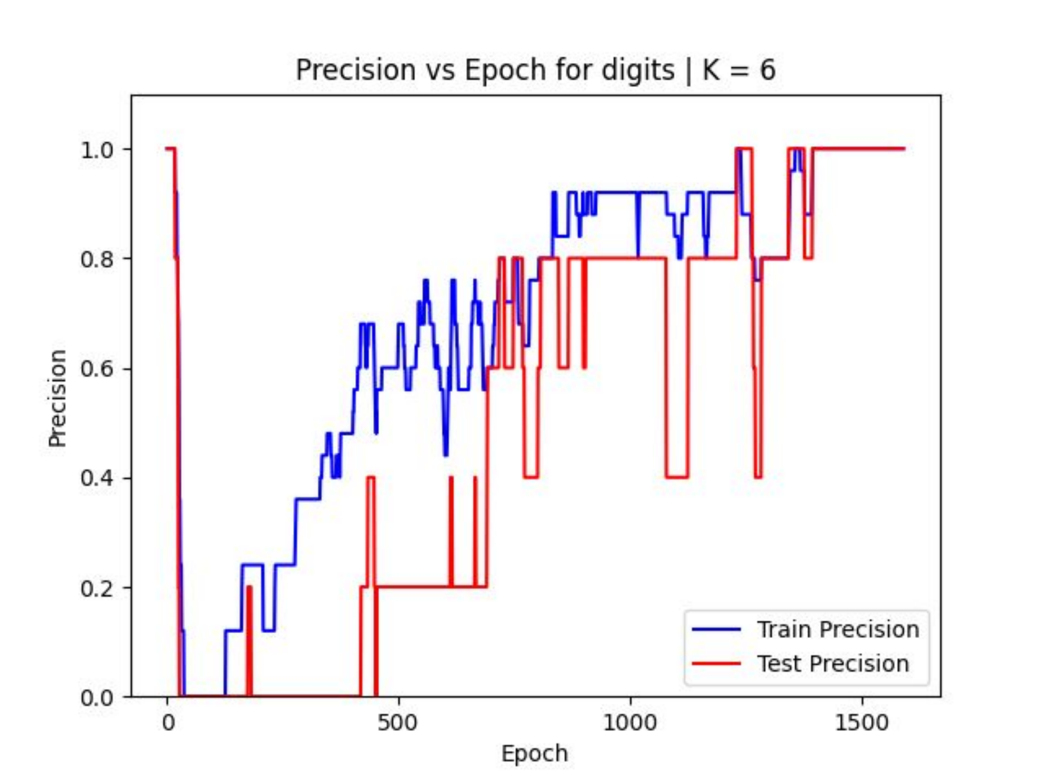


”limit”: 2000,
”random_start”: true,
”learning_rate”: 0.08,
”bias”: 0,
”epsilon”: 0.1,
”extra_groups”: 2,
”k”: 6,
”beta”: 1,
”optimizer”: ”adam”,
”b1”: 0.9331338848100159,
”b2”: 0.9658289155465659,
”e”: 1e-8,
#train = 25
#test = 5

Precision

Online

$$Precision = \frac{TP}{TP + FP}$$



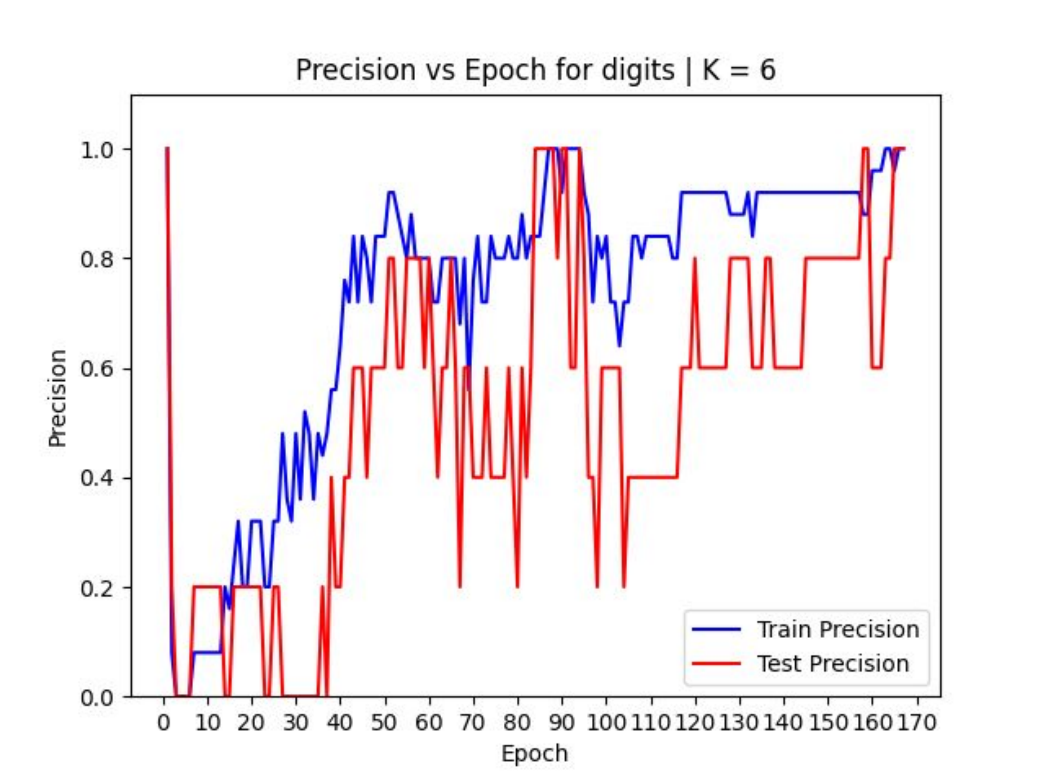
"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

Precision

Batch

$$\frac{TP}{TP + FP}$$

Precision

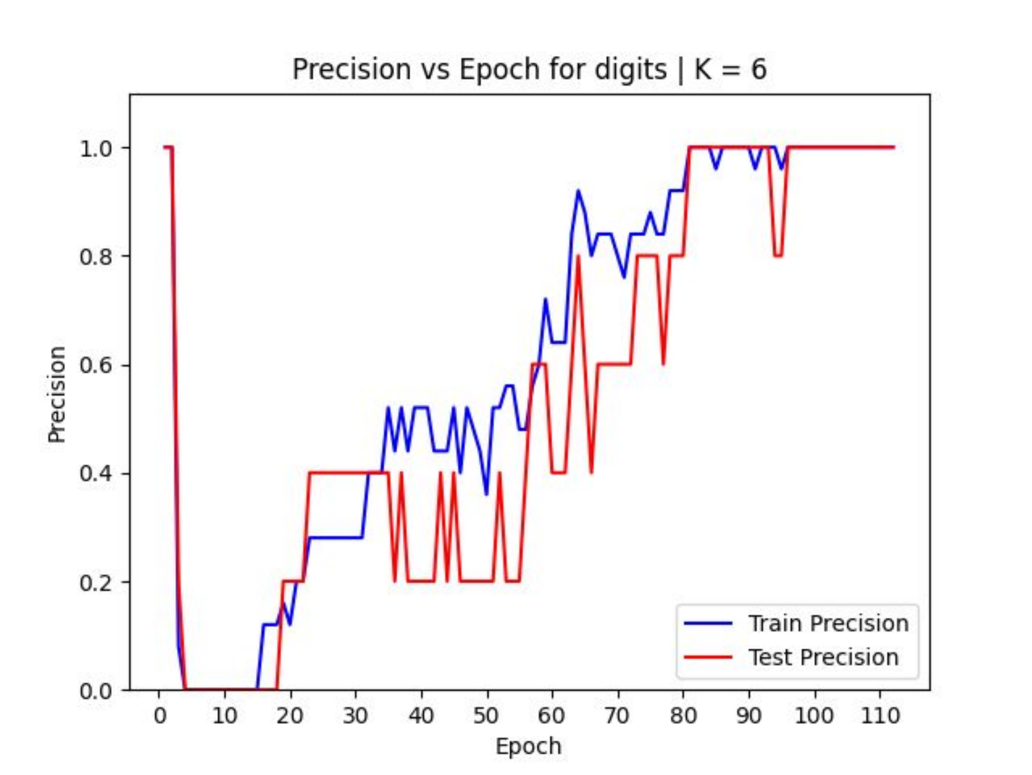


"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

Precision

$$\text{Precision} \quad \frac{TP}{TP + FP}$$

Semi-Batch (10)

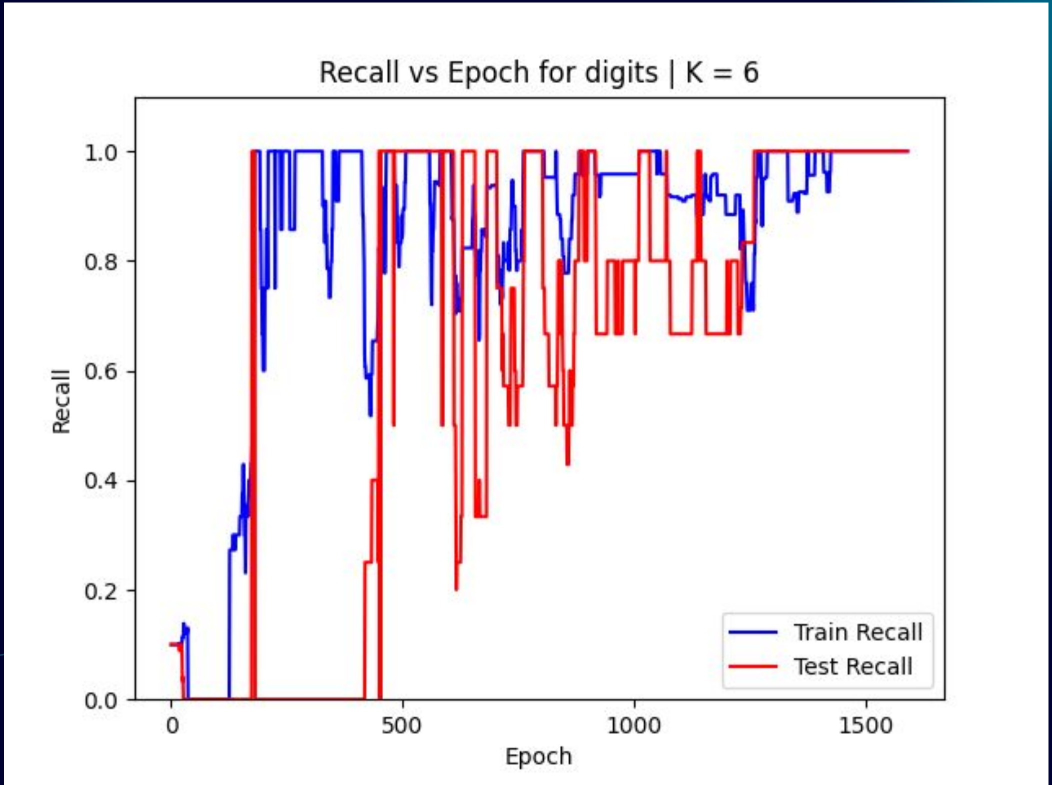


"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

Recall

Online

$$Recall = \frac{TP}{TP + FN}$$

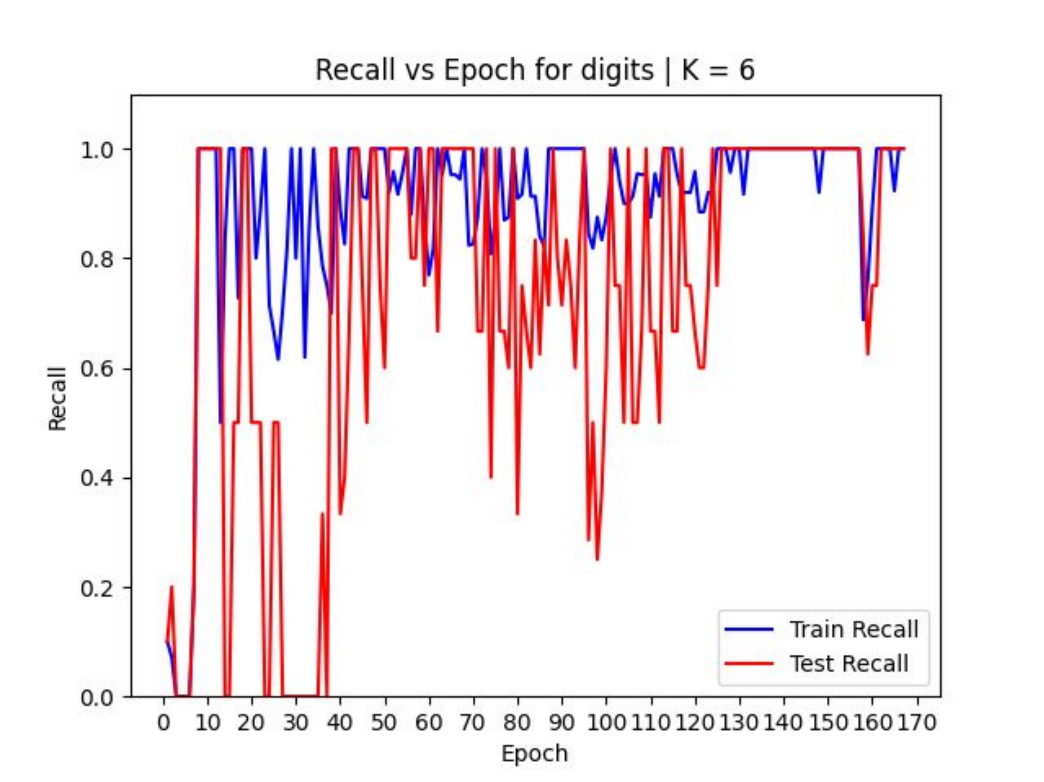


"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,

#train = 25
#test = 5

Recall

Batch



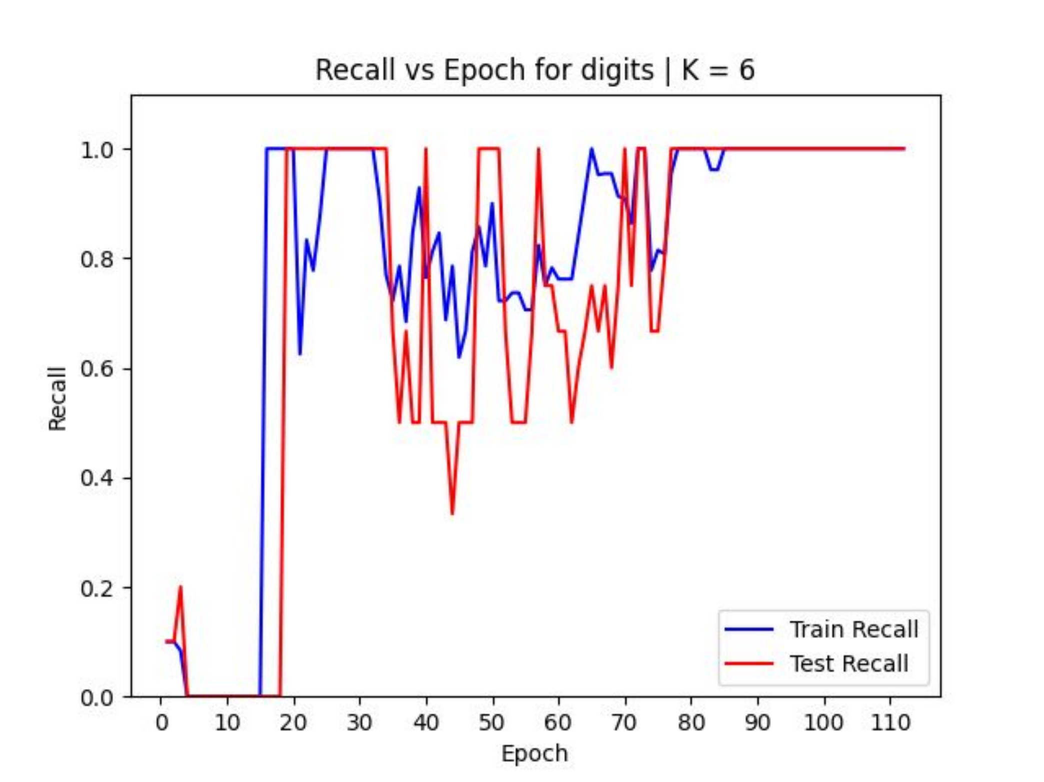
$$Recall = \frac{TP}{TP + FN}$$

"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

Recall

$$Recall$$
$$\frac{TP}{TP + FN}$$

Semi-Batch (10)

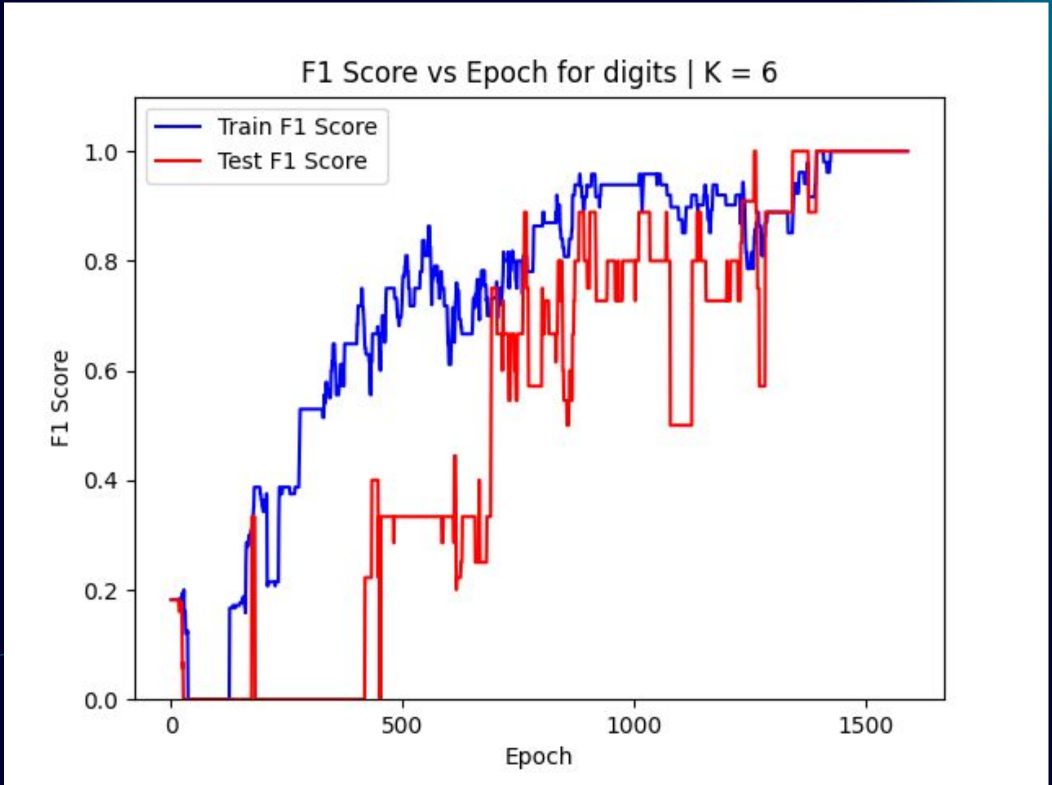


"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

F1 Score

Online

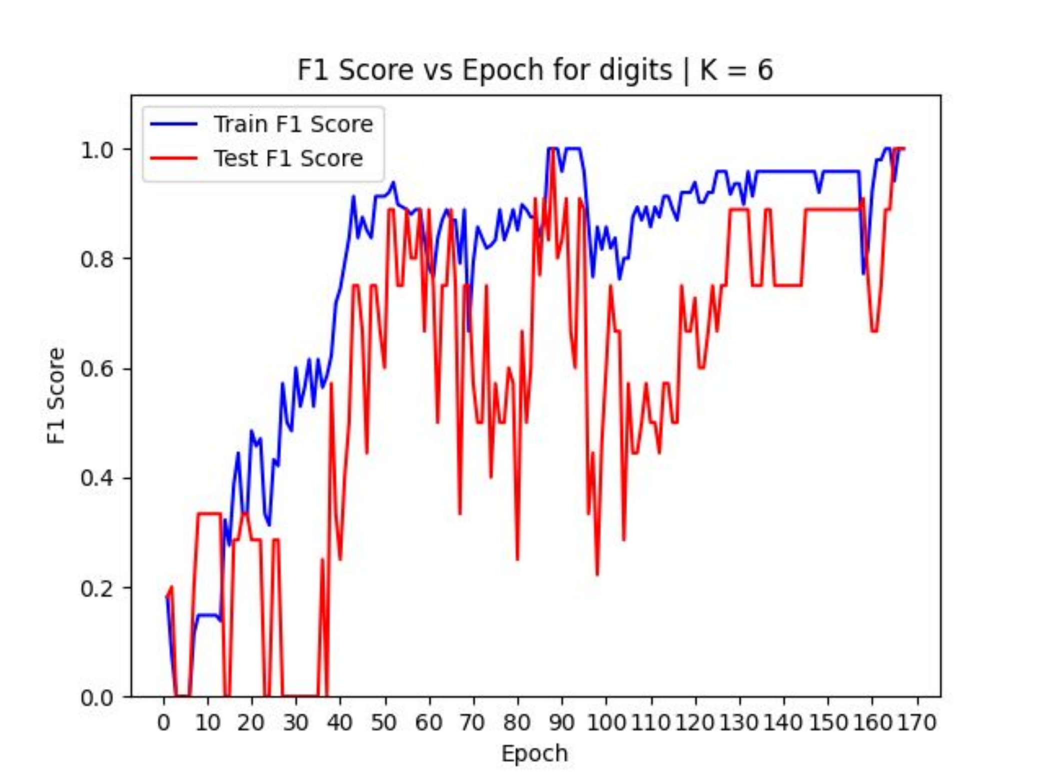
$$F1-Score = \frac{2 * Precision * Recall}{Precision + Recall}$$



"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

F1 Score

Batch



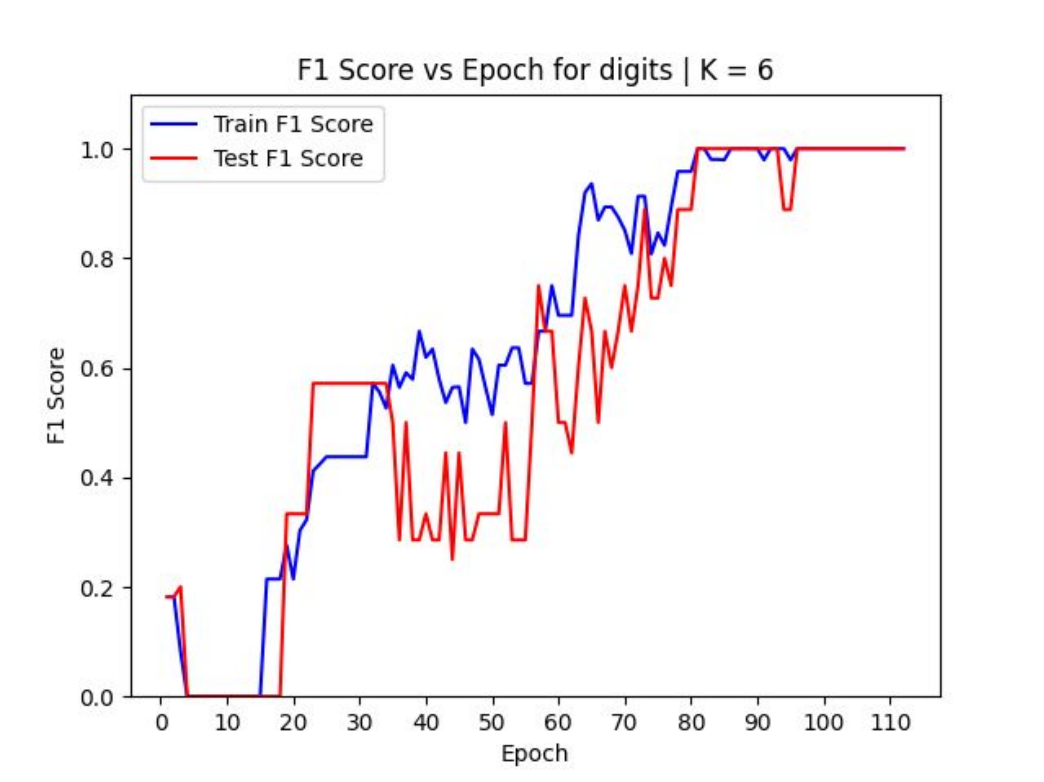
$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

F1 Score

$$F1 - Score$$
$$\frac{2 * Precision * Recall}{Precision + Recall}$$

Semi-Batch (10)



"limit": 2000,
"random_start": true,
"learning_rate": 0.08,
"bias": 0,
"epsilon": 0.1,
"extra_groups": 2,
"k": 6,
"beta": 1,
"optimizer": "adam",
"b1": 0.9331338848100159,
"b2": 0.9658289155465659,
"e": 1e-8,
#train = 25
#test = 5

Conclusiones

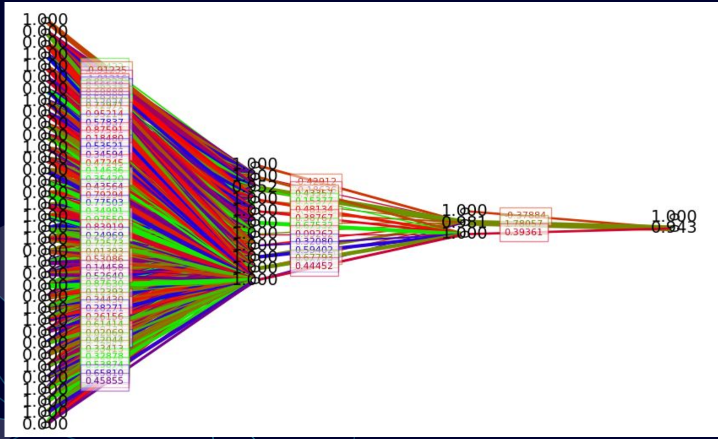
- Existen problemas no linealmente separables que no se pueden resolver con el perceptrón simple escalón, pero sí aproximar con un perceptrón multicapa (Teorema de Aproximación Universal)

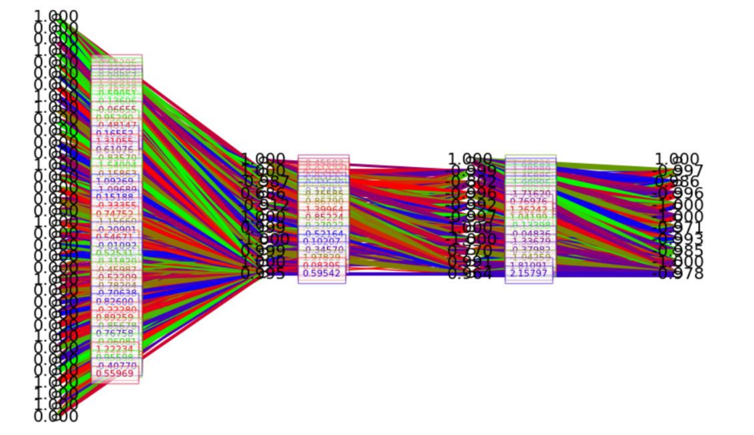
- Utilizar k-folding con distintos valores de k es una buena manera de obtener pesos que se ajusten bien al problema

- Es fundamental la incorporación de elementos adicionales (por ejemplo con ruido) en los data sets para evitar overfitting y lograr la generalización

Conclusiones

- Si bien se le puede dar un significado semántico a cada valor en cada neurona, interpretarlo se puede complejizar mucho, sobre todo considerando que hay varias soluciones posibles





Gracias!

Preguntas?