

Sistemas de Inteligencia Artificial

# Perceptrón Simple y Multicapa

ITBA 2024 - Grupo 02

# El equipo



Girod, Joaquín



Iijas, Christian



Magliotti, Gianfranco



Ferrutti, Francisco

A large, stylized blue neuron with a glowing blue nucleus and numerous branching dendrites and axons, some of which have small glowing nodes. The neuron is set against a dark blue background.

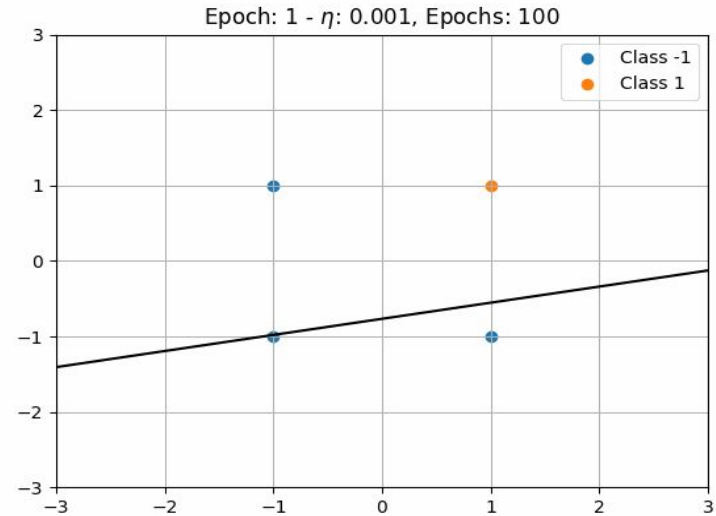
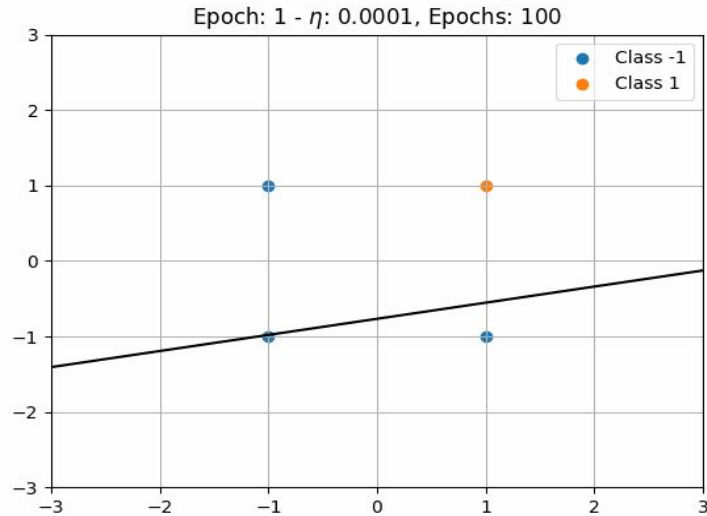
01

# Ejercicio 1

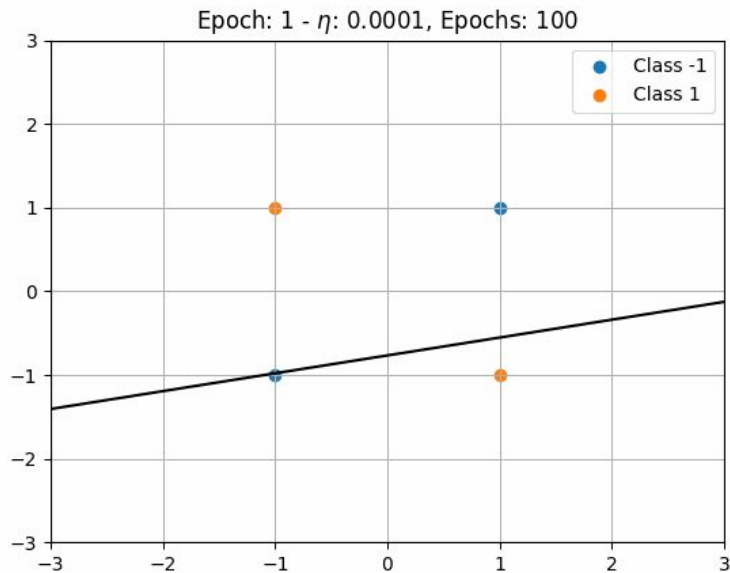
Perceptrón simple con  
activación escalón

# AND logico

El problema típico de clasificación con separación lineal



# Limitaciones del Perceptrón Escalón



Cuando el problema deja de ser linealmente separable los perceptrones simples empiezan a demostrar debilidades.

**Problema del XOR en 2D:** no existe ningún hiperplano que logre separar linealmente las posibles salidas.

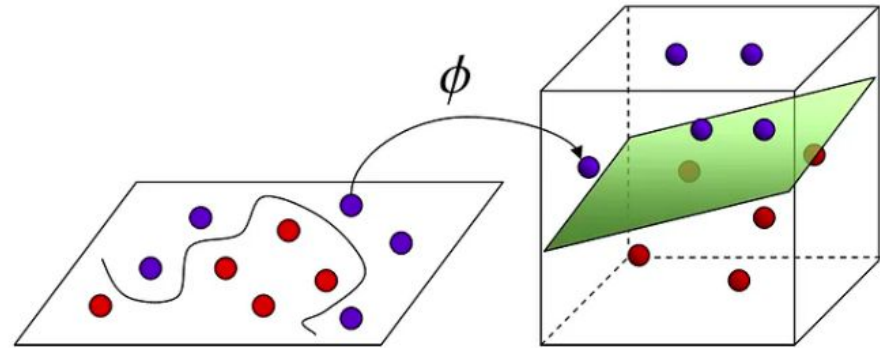
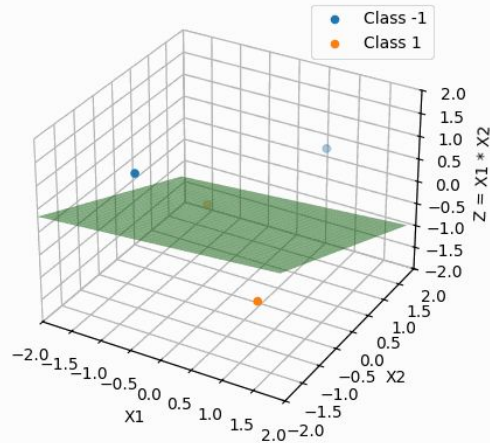
# Pseudo Kernel Trick

Posible solución para el XOR




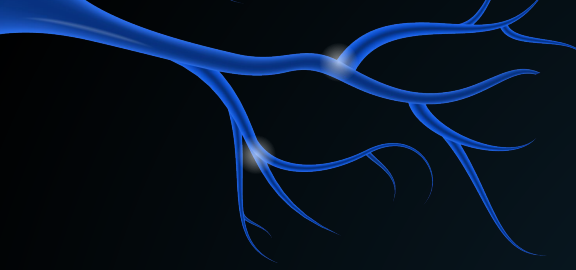
```
learning_rate = 0.001  
epochs = 100
```

Epoch 1 - 3D Decision Boundary



Input Space

Feature Space



02

# Ejercicio 2

Perceptrón lineal y no lineal

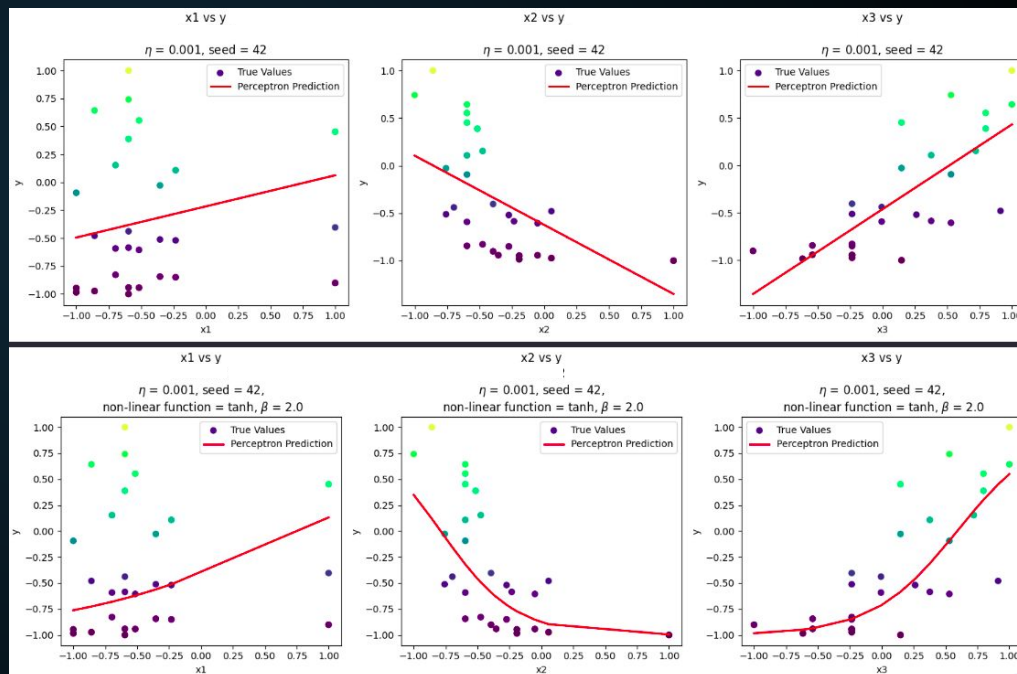
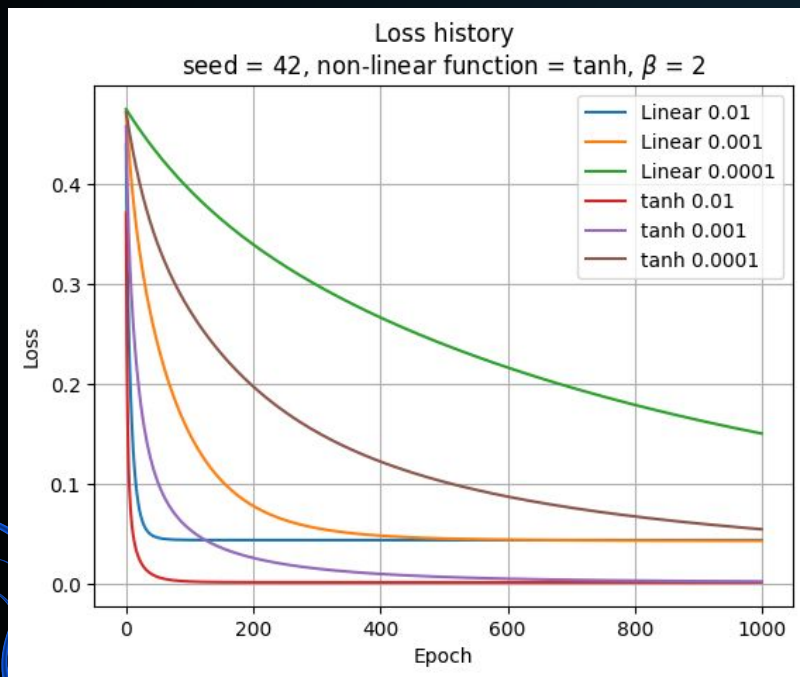


# Capacidad de Aprendizaje

```
dif_x_min = (x - min_valor)
dif_max_min = (max_valor - min_valor)
x_normalizado = dif_x_min / dif_max_min
```

Mirando el **learning rate** elegimos 0.001

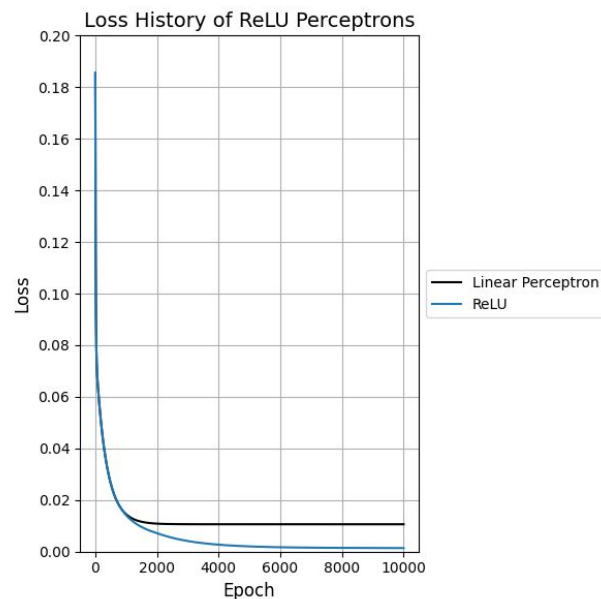
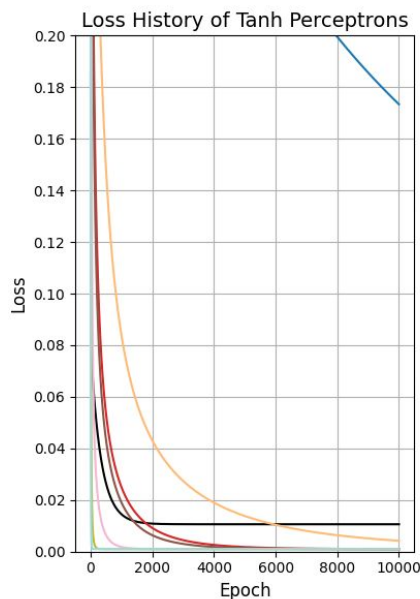
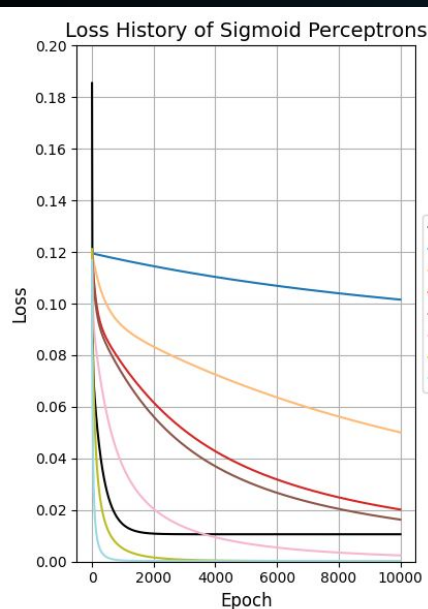
Se nota la diferencia de **capacidad de fitting**





# Capacidad de Aprendizaje

seed = 42  
learning\_rate = 0.001  
epochs = 10000

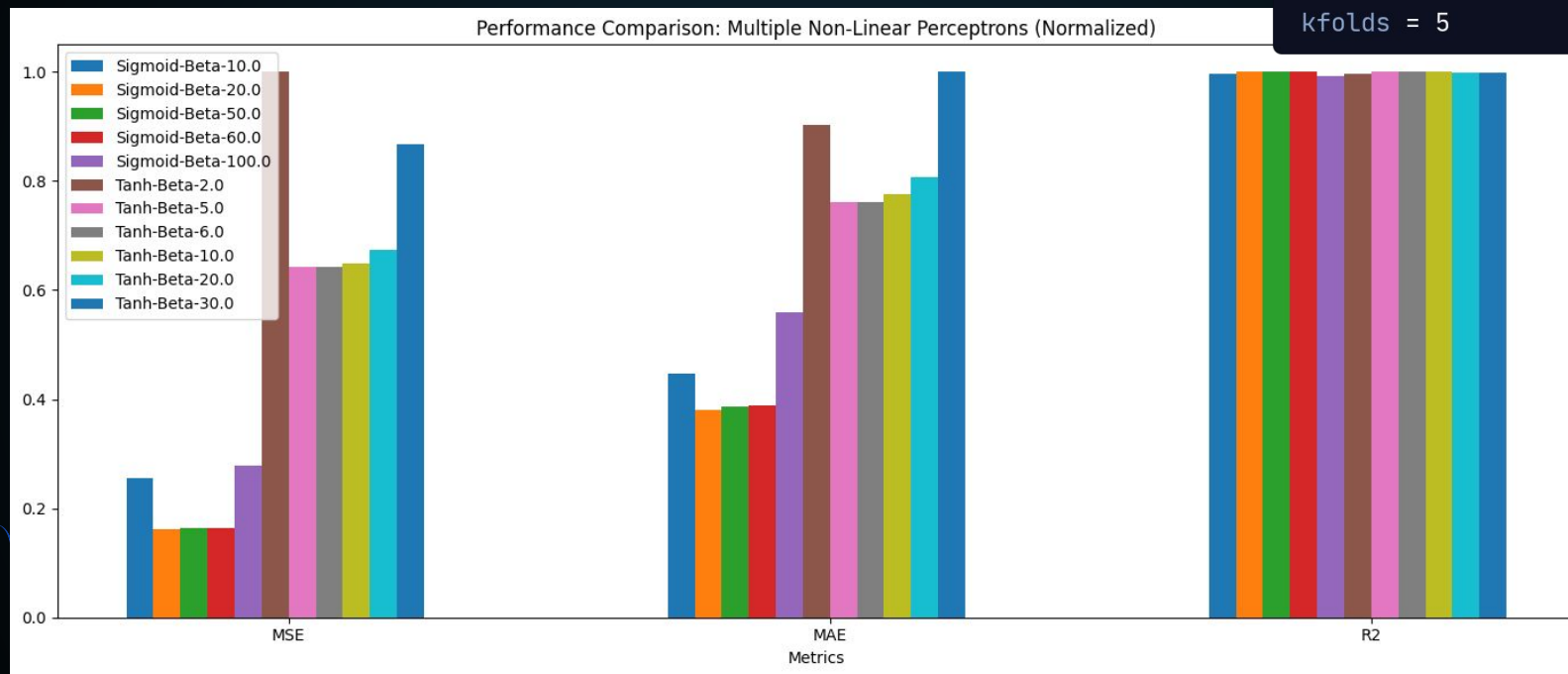


Se puede ver que la función tanh tiene mayor **velocidad de convergencia**.

# Capacidad de Generalización



```
seed = 42  
learning_rate = 0.001  
epochs = 1000  
kfolds = 5
```



Usando validación cruzada evaluamos la capacidad de generalización de los modelos y vemos que sigmoide se porta bien

# Elección

## Un buen conjunto de entrenamiento

Diversidad y representatividad

Tamaño del conjunto de entrenamiento

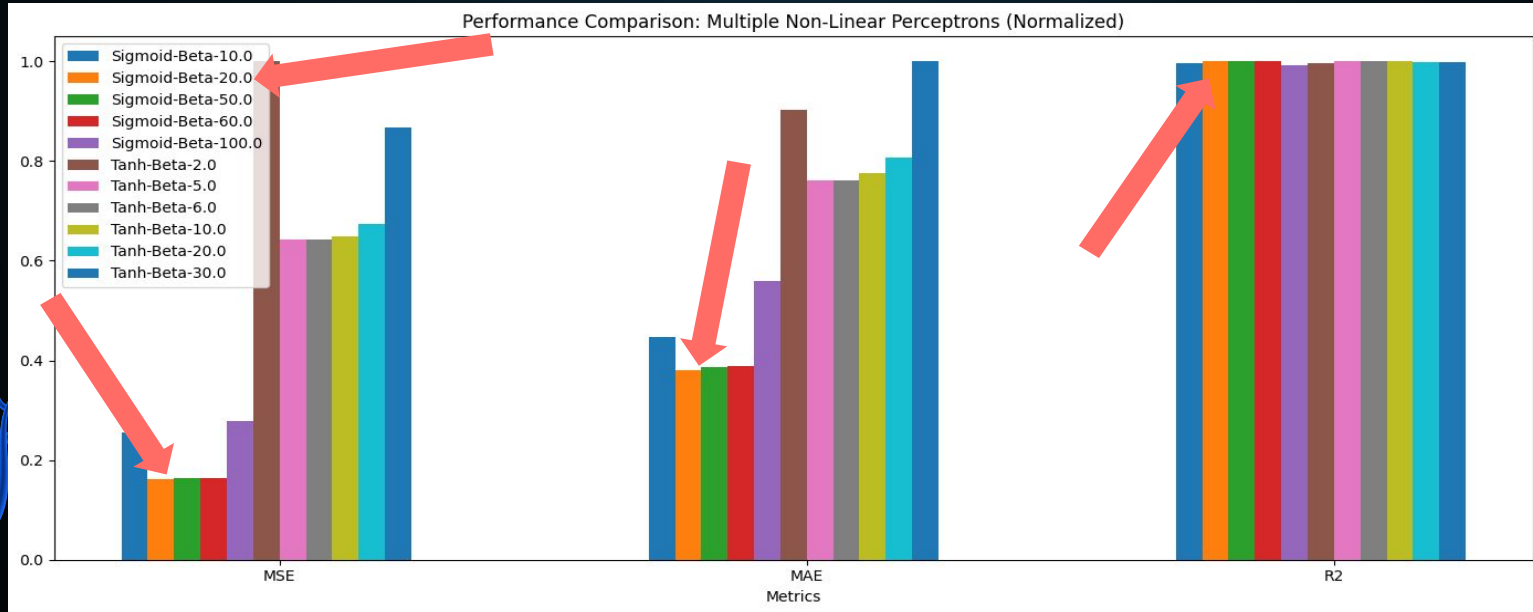
Buen desempeño en cross-validation

```
seed = 42
learning_rate = 0.001
epochs = 1000
kfold = 5
```

Aprender patrones generales

Evitar subajuste

Evitar sobreajuste



A stylized illustration of a neuron in shades of blue. The neuron has a large, glowing spherical cell body (soma) at the bottom right, with numerous branching dendrites extending upwards and outwards. Some of the dendrites have small, bright white dots at their tips, suggesting synaptic connections. The background is a dark blue gradient.

03

# Ejercicio 3

Perceptrón multicapa

# Ejercicio 3.2

Perceptrón multicapa

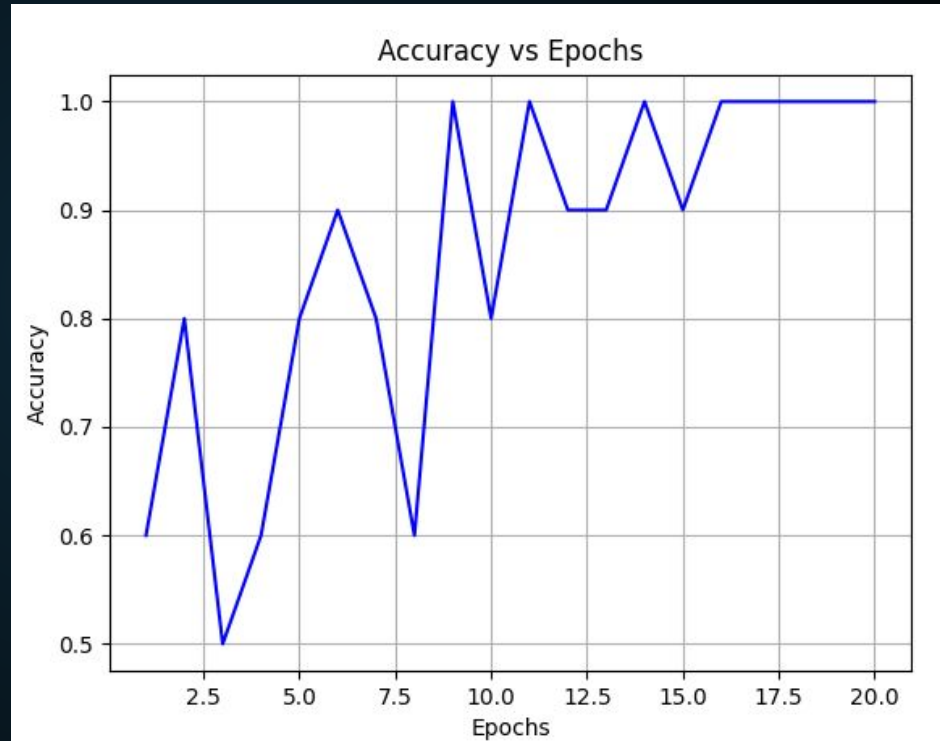
# Discriminación de Paridad

## Resultados

- Se logra una precision del 100% rapidamente



```
topology = [35, 10, 1]
activation_function = sigmoid
sigmoid_beta = 1
optimizer = gradient_descent
mini_batch_size = 4
learning_rate = 8
seed = 23
```



# Ejercicio 3.3

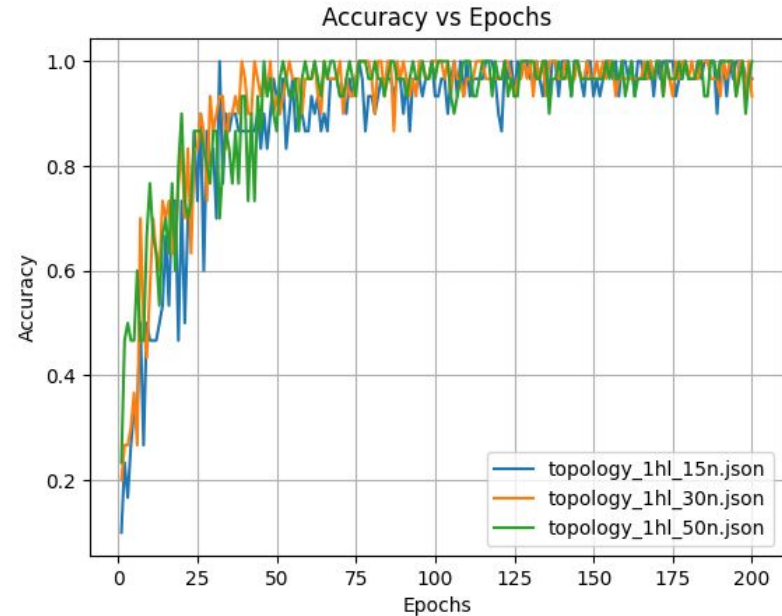
Perceptrón multicapa



# Discriminación de dígito

- Dígitos con ruido Gaussiano de media 0 y desvío estándar 0.75
- Pesos y bias iniciales aleatorios
- Conjunto de 300 elementos con *K-fold* cross-validation,  $K = 10$

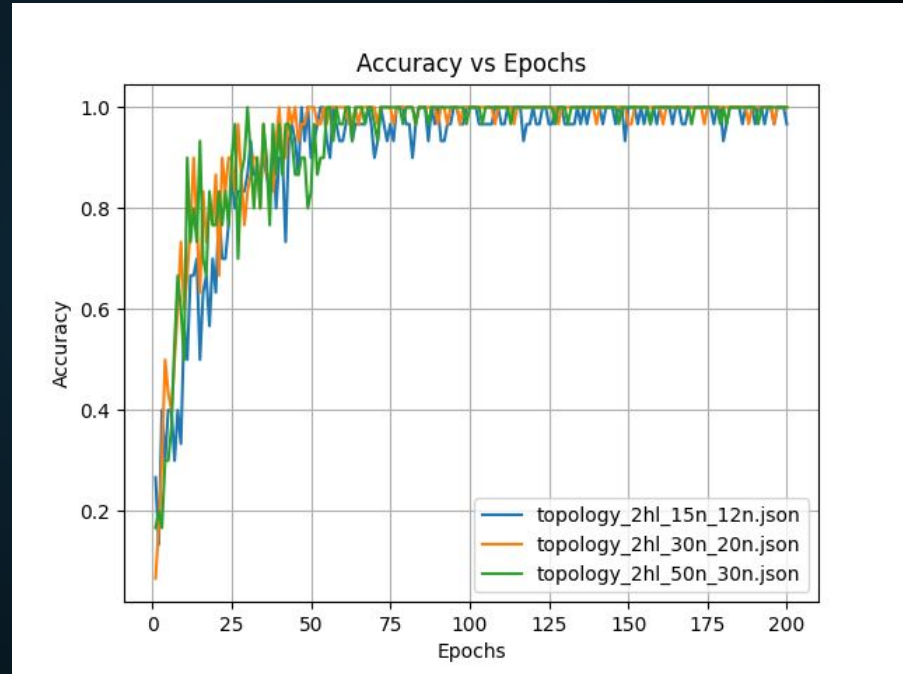
```
activation_function = sigmoid  
sigmoid_beta = 1  
optimizer = gradient_descent  
mini_batch_size = 5  
learning_rate = 3
```



# Discriminación de dígito

Luego optamos por verificar con 2 capas ocultas, nuevamente variando la cantidad de neuronas en las mismas.

```
● ● ●  
  
activation_function = sigmoid  
sigmoid_beta = 1  
optimizer = gradient_descent  
mini_batch_size = 5  
learning_rate = 3
```

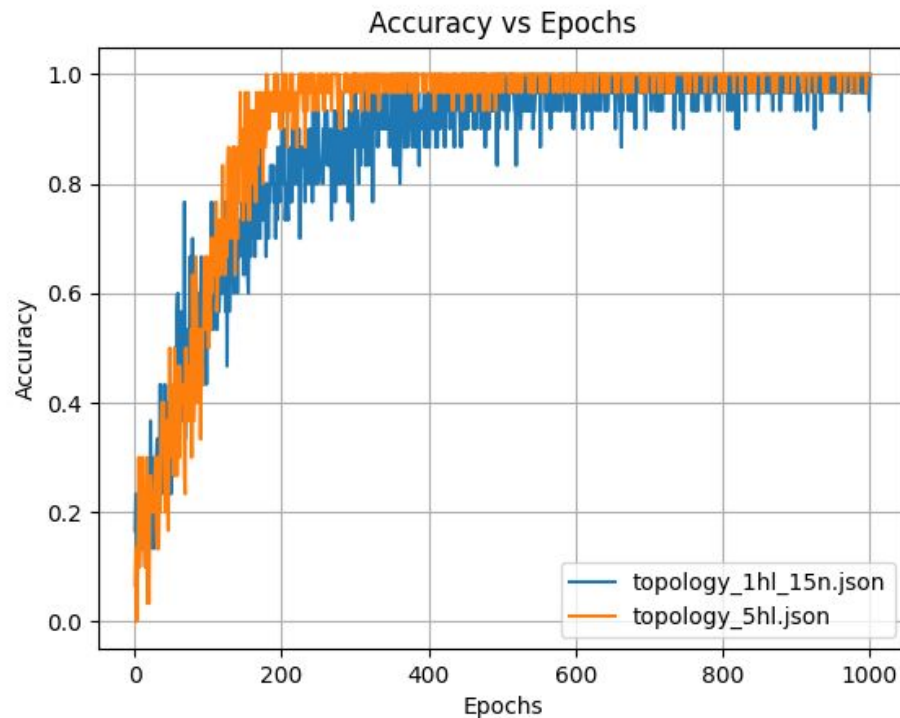


# Discriminación de dígito

Finalmente, simplemente por curiosidad, optamos por poner a prueba una arquitectura 35-100-70-40-25-15-10, de 5 capas ocultas.

Aumentamos las epochs a 1000 para ver comportamiento, el resto de hiperparametros se mantiene.

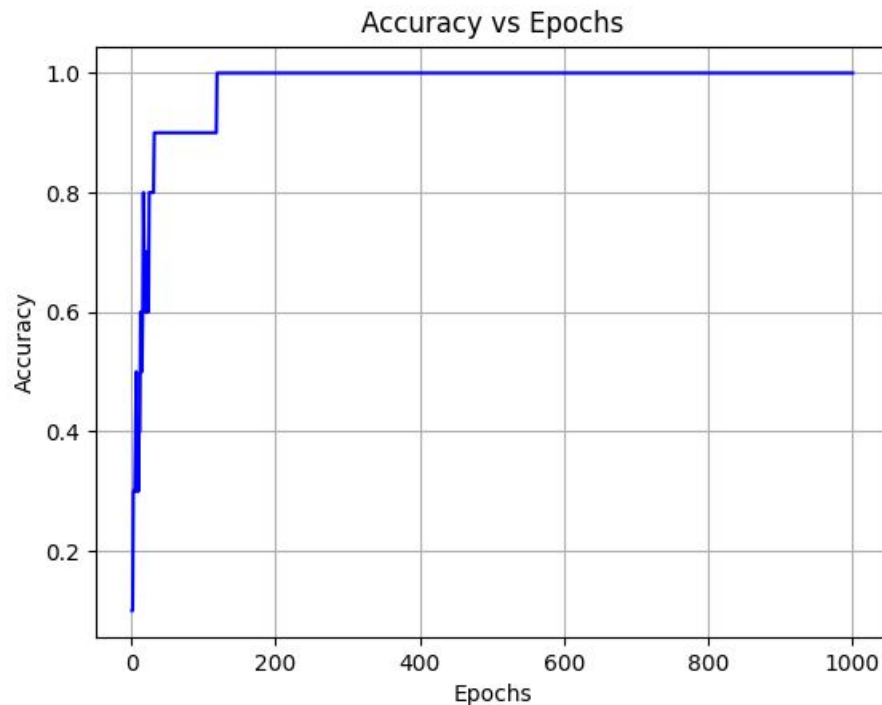
```
● ● ●  
activation_function = sigmoid  
sigmoid_beta = 1  
optimizer = gradient_descent  
mini_batch_size = 5  
learning_rate = 3
```



# Discriminación de dígito

- Dígitos sin ruido
- Pesos y bias iniciales random
- Training set igual al testing set (10 elementos)
- Topologia de [35, 20, 10]

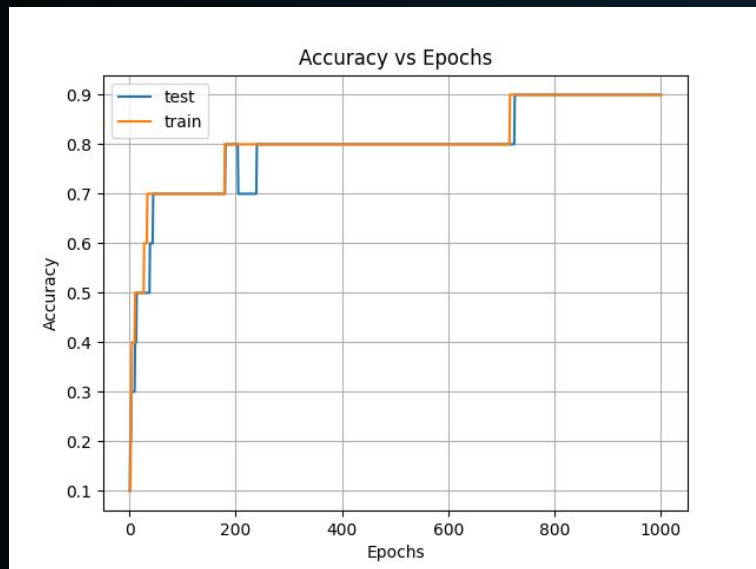
```
activation_function = sigmoid  
sigmoid_beta = 1  
optimizer = gradient_descent  
mini_batch_size = 5  
learning_rate = 8
```



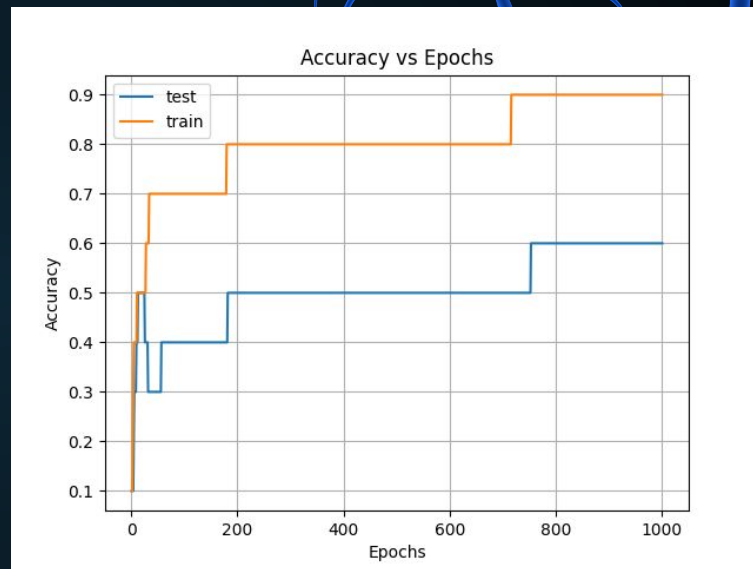
# Discriminación de dígito

Aumentamos el ruido para el testing set...

```
activation_function = sigmoid  
sigmoid_beta = 1  
optimizer = gradient_descent  
mini_batch_size = 5  
learning_rate = 8
```



Ruido gaussiano con media 0 y desvío estándar 0.4 para un testing set con ruido de 100 elementos



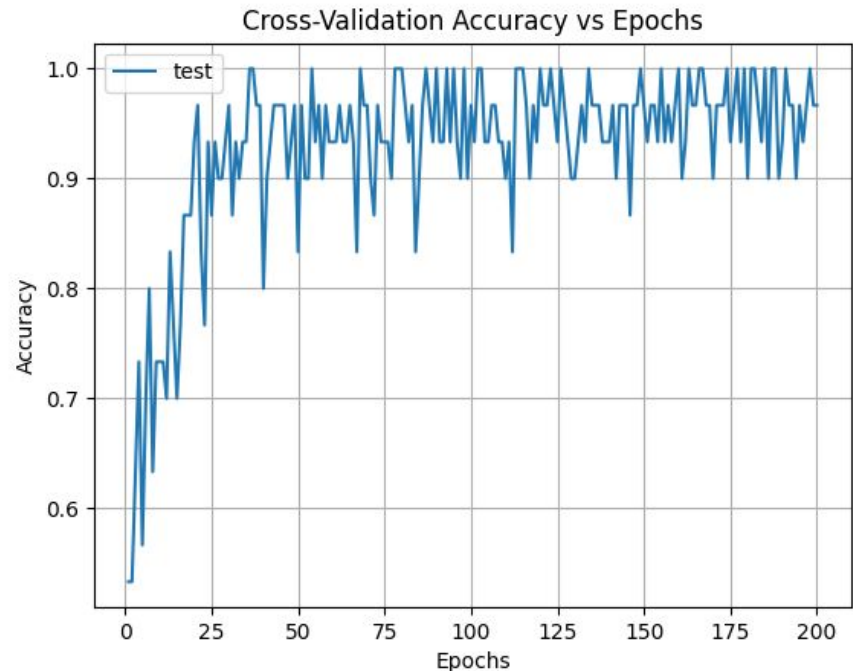
Ruido gaussiano con media 0 y desvío estándar 0.75 para un testing set con ruido de 100 elementos

# Discriminación de dígito

Entrenamos al perceptrón con ruido...

- Weights y biases iniciales obtenidos del entrenamiento sin ruido
- *K-Fold* Cross-Validation con  $K = 10$
- Tamaño del training set: 1000 dígitos
- Agregamos al training set ruido gaussiano con media 0 y desvío estándar 0.4

```
activation_function = sigmoid
sigmoid_beta = 1
optimizer = gradient_descent
mini_batch_size = 50
learning_rate = 16
```

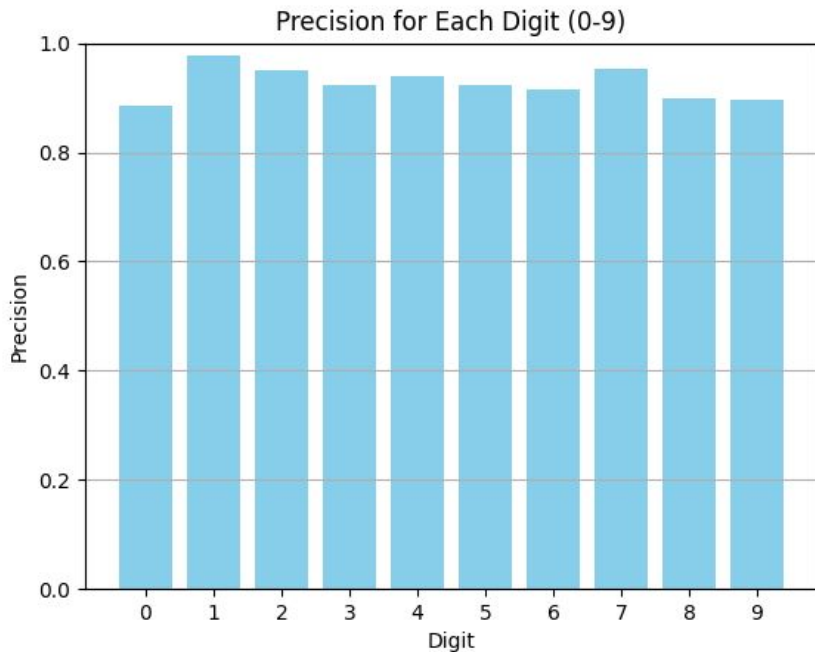


# Discriminación de dígito

Evaluemos cómo fue el desempeño para cada dígito

- Evaluamos los resultados del entrenamiento anterior para un conjunto de 300 números con más ruido
- Ruido gaussiano con media 0 y desviación estándar 0.75

```
activation_function = sigmoid
sigmoid_beta = 1
optimizer = gradient_descent
mini_batch_size = 50
learning_rate = 16
epochs = 100
```



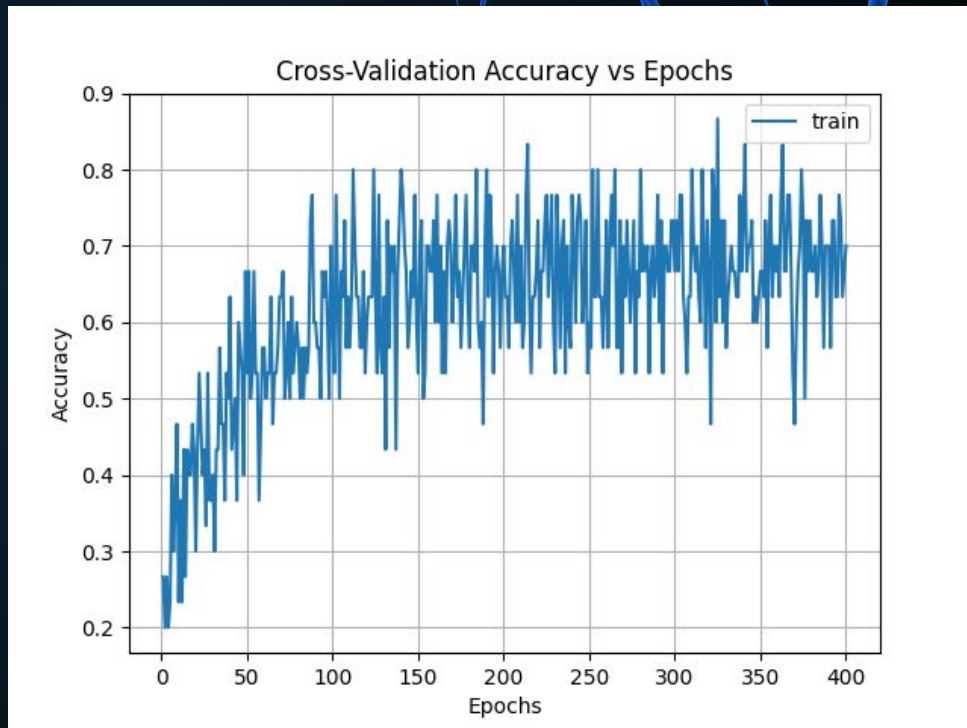


# Discriminación de dígito

¿Qué pasa si entrenamos ahora con salt and pepper?

- Seguimos aprendiendo
  - Usamos los pesos y bias del entrenamiento anterior
- Salt and pepper
  - Salt prob 0.4
  - Pepper prob 0.4
- Mayor error en el entrenamiento
- Probable overfitting

```
activation_function = sigmoid
sigmoid_beta = 1
optimizer = gradient_descent
mini_batch_size = 50
learning_rate = 16
epochs = 200
```



A large, stylized blue neuron is positioned on the right side of the slide. It features a central cell body (soma) and numerous branching dendrites and axons that spread across the frame. The neuron is rendered in a glowing blue color against a dark blue background. The number '04' is displayed in a large, light blue font in the upper right corner.

04

# Ejercicio 4

Perceptrón multicapa

# Comparando Optimizadores



```
topology = [784,30,10]
actiation = sigmoid
beta = 1
seed = 42
epochs = 20
mini_batch_size = 16
learning_rate = 0.1
```



```
gradient_descent
```

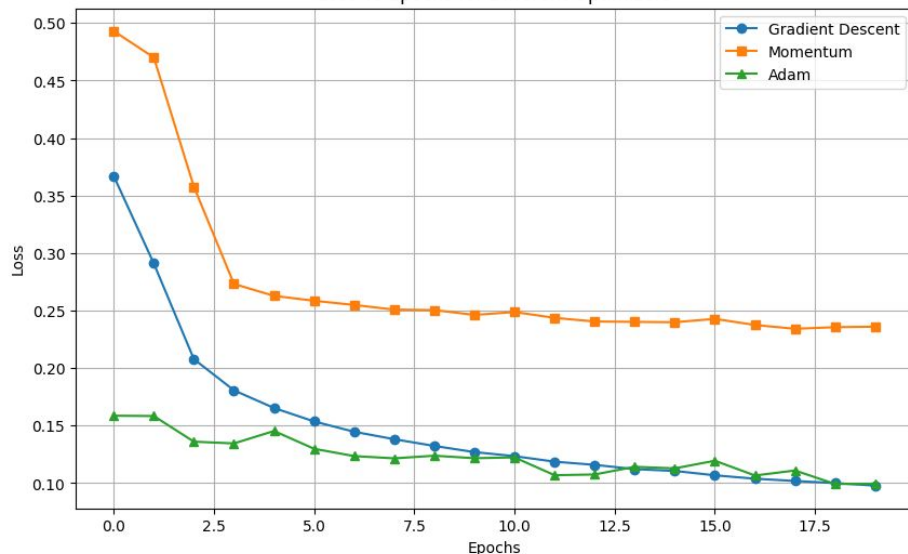
```
adam
```

```
beta_1 = 0.9
beta_2 = 0.999
epsilon = 1e-8
```

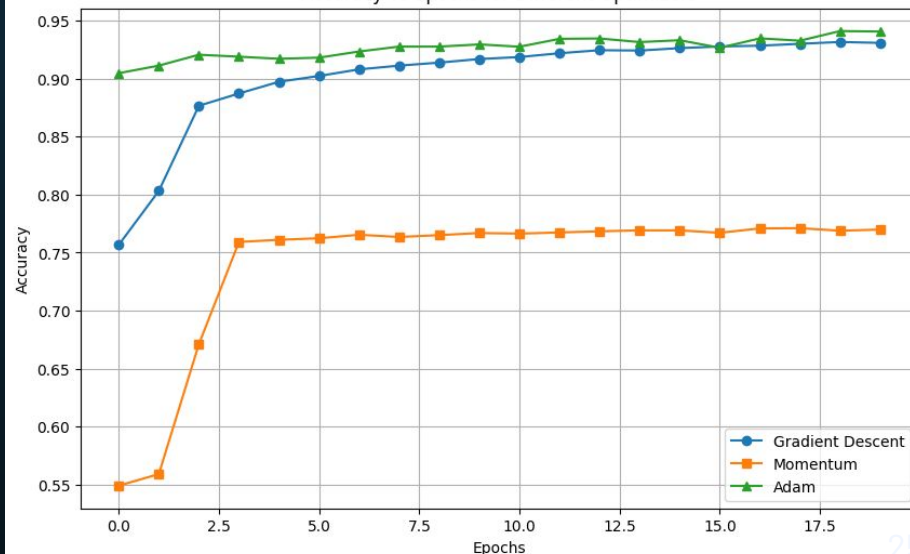
```
momentum
```

```
alpha = 0.9
```

Loss vs Epochs for Different Optimizers



Accuracy vs Epochs for Different Optimizers



# Discriminación de dígito manuscrito

Nuestros mejores resultados

94.92%

```
topology = [784, 30, 10]
activation_function = sigmoid
sigmoid_beta = 1
optimizer = adam
beta_1 = 0.9
beta_2 = 0.999
adam_epsilon = 1e-8
seed = 42
epochs = 30
mini_batch_size = 16
learning_rate = 0.1
epsilon = 0.01
```

94.84%

```
topology = [784, 30, 10]
activation_function = sigmoid
sigmoid_beta = 1
optimizer = momentum
alpha = 0.9
seed = 42
epochs = 30
mini_batch_size = 16
learning_rate = 0.1
epsilon = 0.01
```

90.05%

```
topology = [784, 30, 10]
activation_function = sigmoid
sigmoid_beta = 1
optimizer = gradient_descent
seed = 42
epochs = 30
mini_batch_size = 16
learning_rate = 0.1
epsilon = 0.01
```

# Pero cambiando la topología

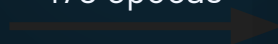
Nuestros mejores resultados

97.16%



```
topology = [784, 128, 10]
activation_function = sigmoid
sigmoid_beta = 1
optimizer = adam
beta_1 = 0.9
beta_2 = 0.999
adam_epsilon = 1e-8
epochs = 30
mini_batch_size = 16
learning_rate = 0.1
epsilon = 0.01
```

+170 épocas



97.56%



```
topology = [784, 128, 10]
activation_function = sigmoid
sigmoid_beta = 1
optimizer = adam
beta_1 = 0.9
beta_2 = 0.999
adam_epsilon = 1e-8
epochs = 200
mini_batch_size = 16
learning_rate = 0.1
epsilon = 0.01
```



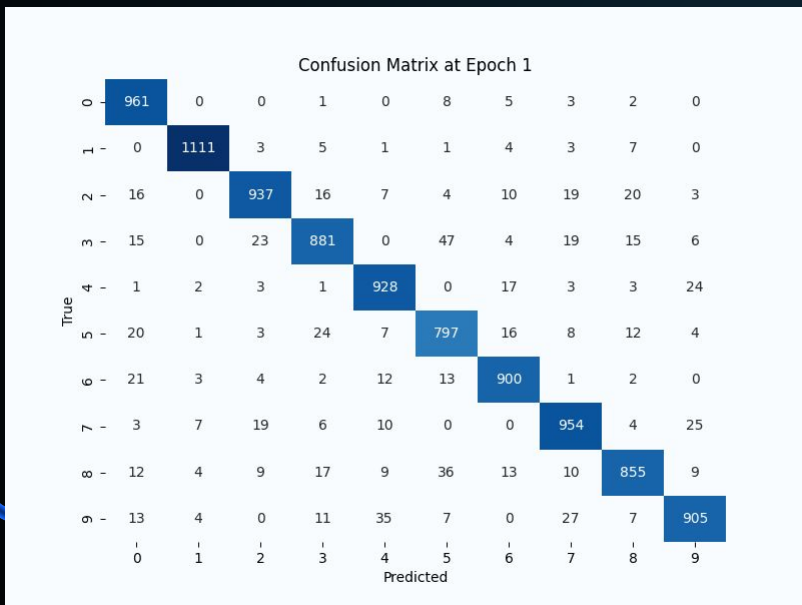
Demo Time!



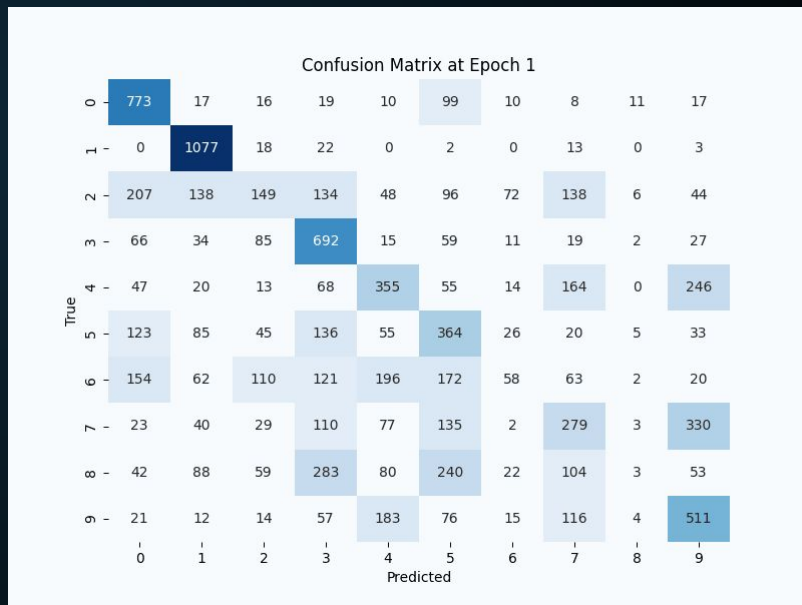
# Matrices de Confusión

Accuracy

Adam



Gradient Descent



```
topology = [794, 30, 10]
activation_function = sigmoid
sigmoid_beta = 1
seed = 42
mini_batch_size = 16
learning_rate = 0.1
epsilon = 0.01
```



# Conclusiones Generales

- 1) Los modelos simples reducen la probabilidad de overfitting, ajustar el modelo al problema no el problema al modelo
- 2) Importancia de la función de activación no lineal para flexibilizar la red
- 3) La elección de la topología es un arte
- 4) Potencial de las MLP para problemas con dominio de entrada no explorable
- 5) Capacidad de representar cualquier función a cambio de poder de cómputo
- 6) Se aproxima asintóticamente al máximo de accuracy
- 7) Peligro del overfitting y relevancia de la validación junto con el noise
- 8) Importancia del dataset, preparar para el mundo real, no el ideal
- 9) Black Box?

Original Cat Image



Initial Classification (Cat Image):  
Class Name: tabby, Probability: 0.7971  
Class Name: tiger cat, Probability: 0.1733  
Class Name: Egyptian cat, Probability: 0.0222  
Class Name: carton, Probability: 0.0007  
Class Name: Persian cat, Probability: 0.0006

Modified Cat Image with Tennis Added



Modified Classification (Cat + Tennis Image):  
Class Name: tennis ball, Probability: 0.9961  
Class Name: spider web, Probability: 0.0028  
Class Name: chainlink fence, Probability: 0.0009  
Class Name: hare, Probability: 0.0002  
Class Name: puffer, Probability: 0.0000



# Futuros Posibles

- 1) Sería interesante analizar el ejercicio 3 o 4 agregando la opción de clasificar el número en una categoría “desconocido”
- 2) Data sets desbalanceados
- 3) Usar CUDA para aprovechar la GPU, actualmente la red es CPU-intensive. O en su defecto usar una librería que aproveche la GPU (cómo Torch).
- 4) Agrupar biases y weights en la misma matriz
- 5) Sparse Neural Networks
- 6) Múltiples funciones de activaciones para las distintas capas

A stylized illustration of a neuron with a large, glowing blue nucleus and several branching dendrites, set against a dark blue background. The neuron is positioned in the bottom-left corner, with its branches extending towards the center and right. The overall aesthetic is scientific and modern.

**¡Gracias!**