



## **Laboratorio en el Uso de Redes Neuronales Artificiales**

### **Semillero de Redes Neuronales**

**Integrantes:**  
**Laurence Santamaría**  
**Maria Jose Sierra**

**Profesor:**  
**Albert Montenegro**

**Carrera:**  
**Ciencia de Datos**

**Facultad:**  
**Departamento de Matemáticas**

**Universidad Externado de Colombia**  
**Bogotá.D.C.**  
**2023**

# Informe de Laboratorio

## Asignación 1: Análisis del desempeño

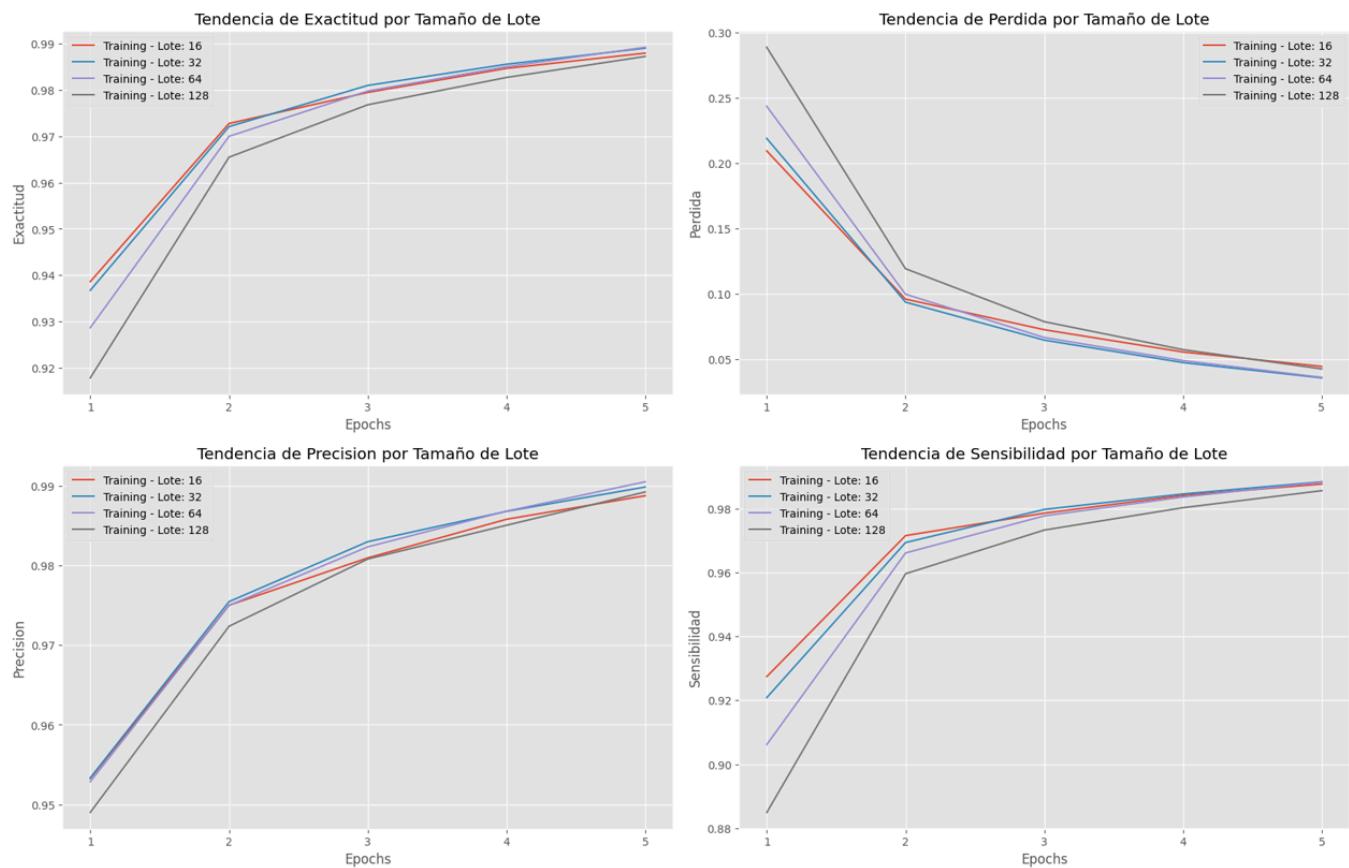
Consta de evaluar el rendimiento de su red para diferentes valores de hiper-parámetros. Manteniendo el resto de los valores constantes (e iguales a los valores predeterminados).

**A. Reporte y analice a profundidad las variaciones observadas para cada hiper-parámetro.**

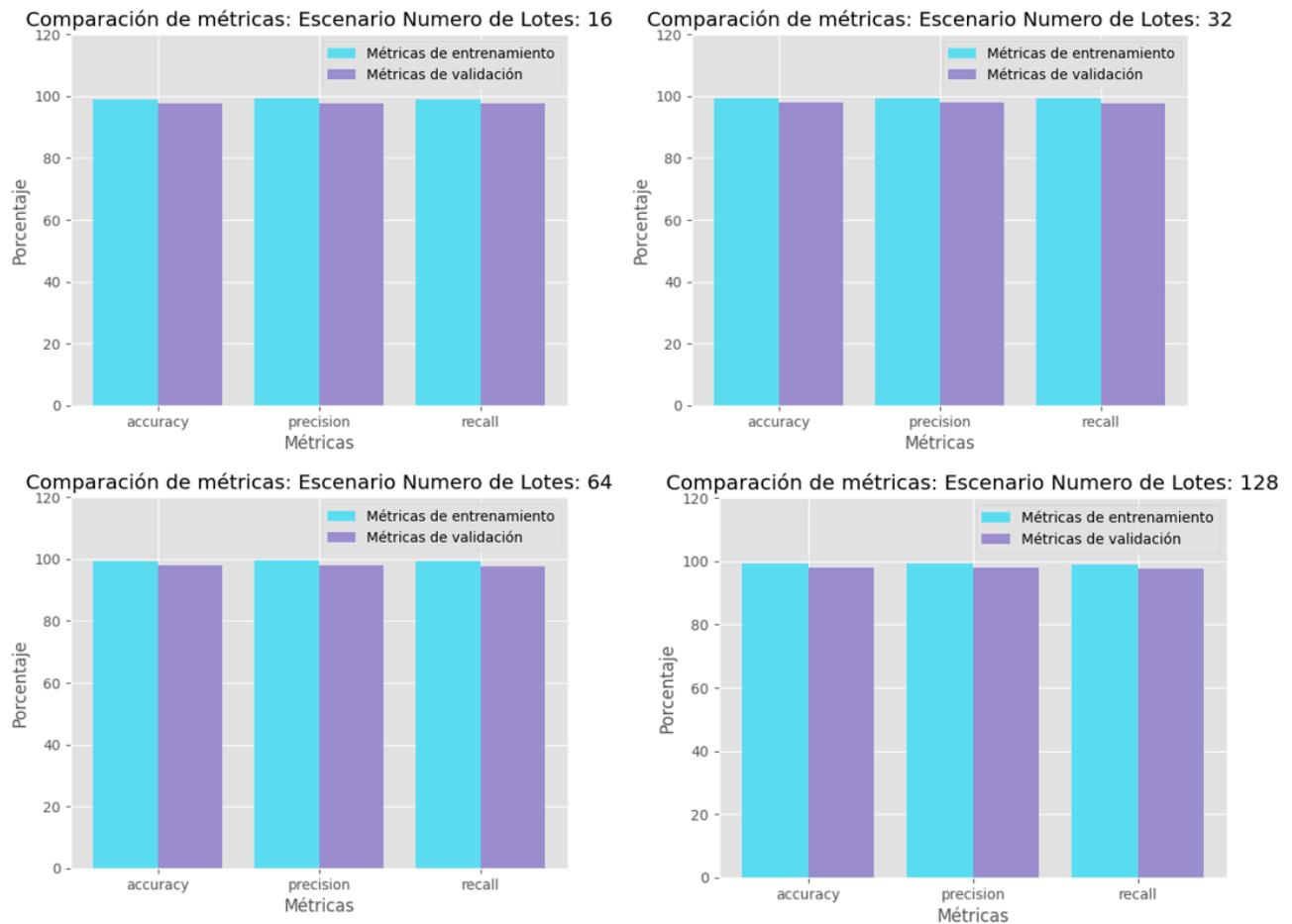
**Determine si cada uno de los modelos es sobre entrenado, sub ajustado o bien ajustado.**

### Tamaño de lote: 16, 32, 64 y 128

#### Gráfico de tendencias



## Comparación de métricas en entrenamiento y validación:



Los cuatro modelos están bien ajustados pues a pesar de tener métricas altas, tienen un desempeño similar tanto en entrenamiento y validación.

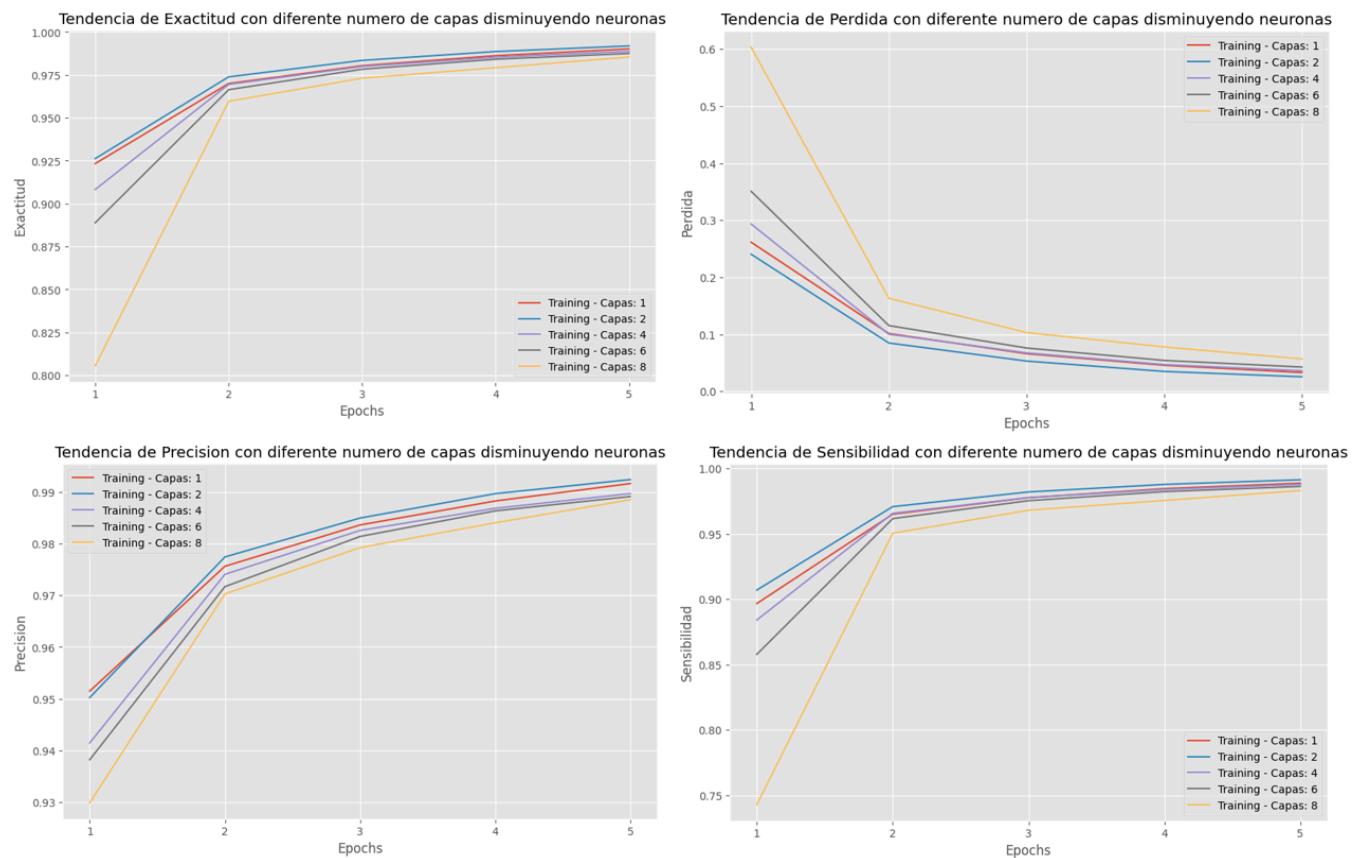
En general los modelos tienen un comportamiento similar a excepción del modelo de 128 lotes, el cual era el que se tenía originalmente en la estructura base pero terminó teniendo el peor desempeño. Esta poca variación puede deberse a que al tener 50.000 datos en el conjunto entrenamiento, tener lotes tan pequeños no genera cambios significativos en el desempeño de la red. En general, aunque por muy poca diferencia, el modelo con 32 lotes obtiene los mejores resultados en la red en todas las métricas.

## Número de capas ocultas: 1, 2, 4, 6, 8 disminuyendo progresivamente el número de neuronas por capa.

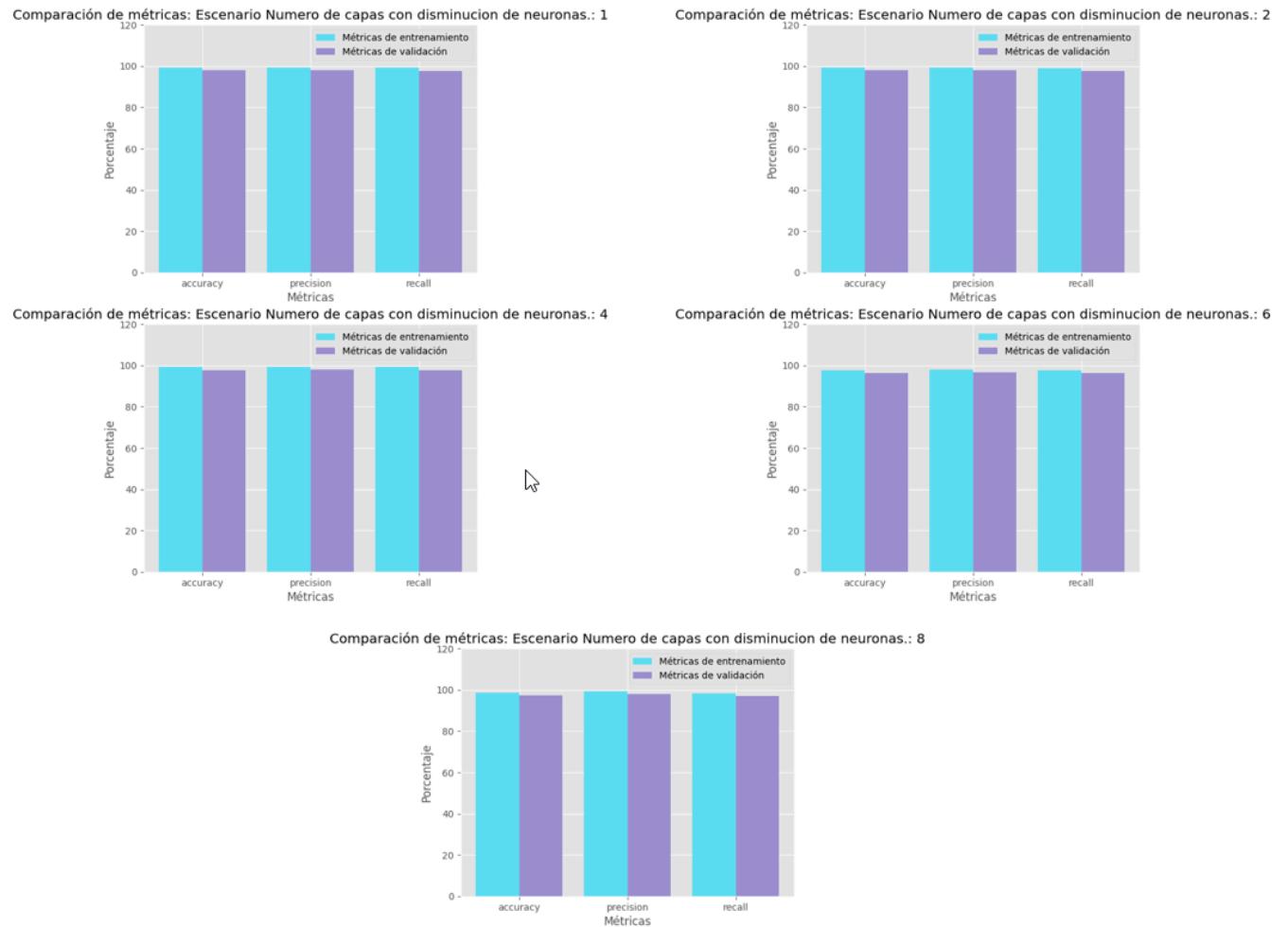
Número de neuronas por capa en cada modelo (incluyendo la de salida):

- **1 capa:** 1024/10. Total: 1034 neuronas
- **2 capas:** 1024/512/10. Total: 1546 neuronas
- **4 capas:** 1024/512/256/128/10. Total: 1930 neuronas
- **6 capas:** 1024/512/256/128/64/32/10. Total 2026 neuronas
- **8 capas:** 1024/512/256/128/64/32/16/8/10. Total 2050 neuronas

### Gráfico de tendencias



## Comparación de métricas en entrenamiento y validación:



Los cinco modelos están bien ajustados pues a pesar de tener métricas altas, tienen un desempeño similar tanto en entrenamiento y validación.

En estos modelos se quiso experimentar teniendo estructuras inicialmente bastante robustas para terminar teniendo muy pocas neuronas. Hay que destacar que el mejor rendimiento se tuvo con 2 capas con neuronas organizadas 1024/512/10 y el peor con 8 capas con la estructura 1024/512/256/128/64/32/16/8/10, a pesar de que no hubo una gran diferencia entre estos.

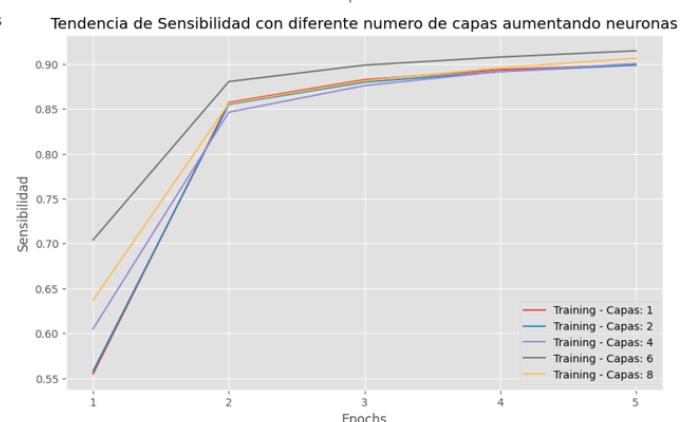
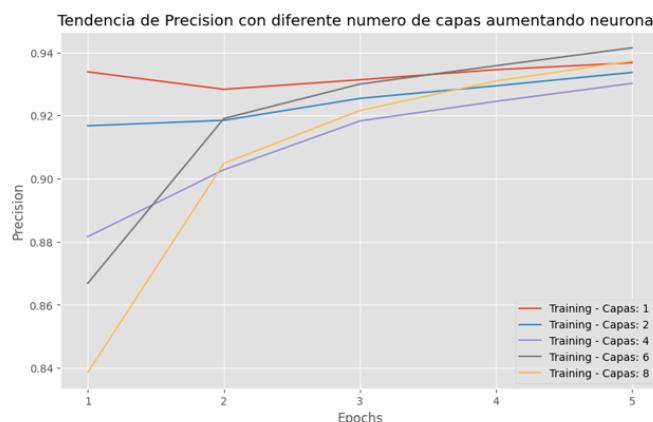
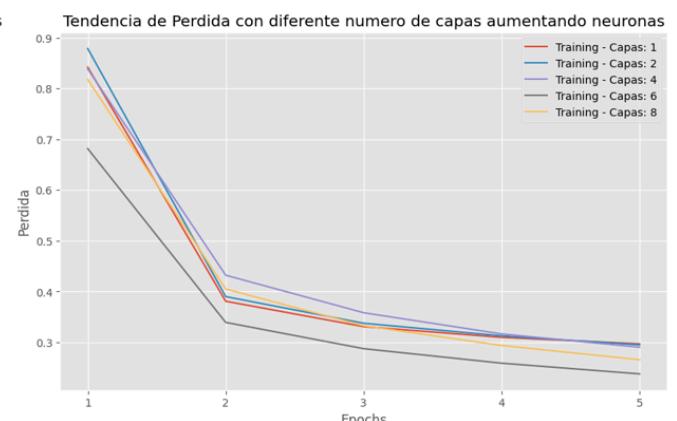
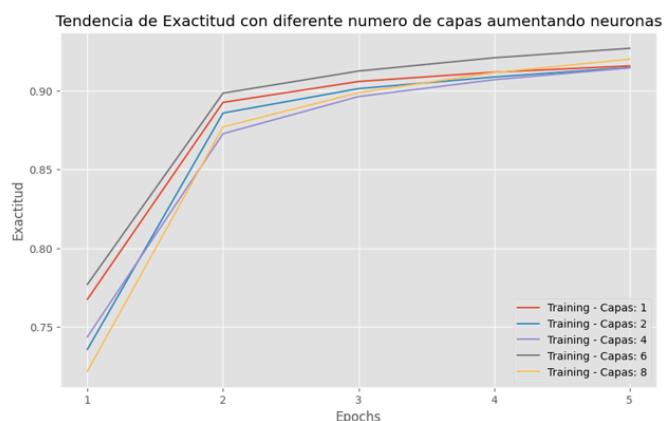
Esto lo que nos arroja es que al ir disminuyendo el número de neuronas, es preferible usar un menor número de capas, principalmente debido a que se pueden ir perdiendo características importantes de las imágenes y que ocurra un desvanecimiento del gradiente, evitando que se puedan ajustar correctamente los pesos.

## Número de capas ocultas: 1, 2, 4, 6, 8 aumentando progresivamente el número de neuronas por capa.

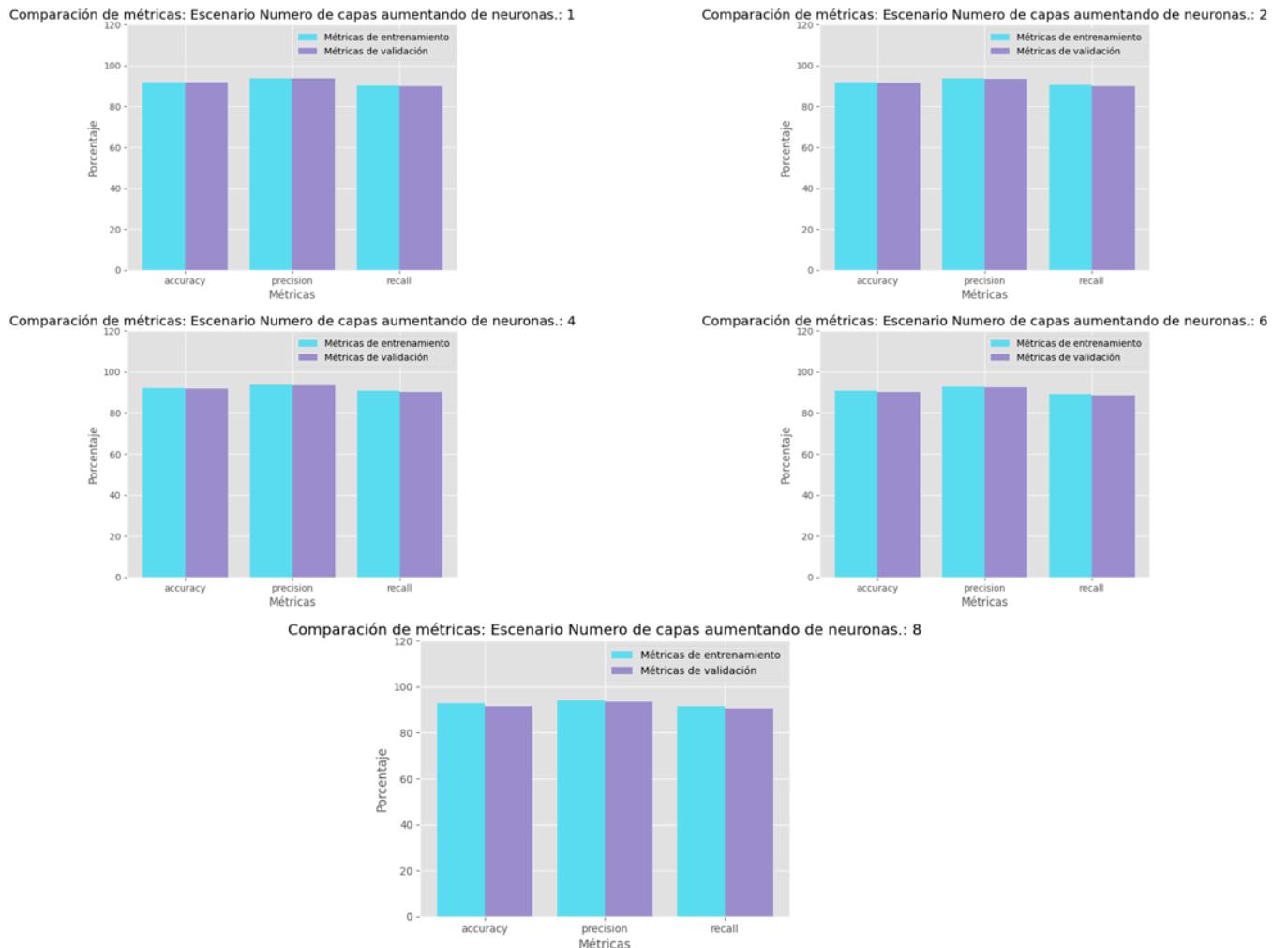
Número de neuronas por capa en cada modelo (incluyendo la de salida):

- **1 capa:** 8/10. Total: 18 neuronas
- **2 capas:** 8/16/10. Total: 34 neuronas
- **4 capas:** 8/16/32/64/10. Total: 130 neuronas
- **6 capas:** 8/16/32/64/128/256/10. Total 514 neuronas
- **8 capas:** 8/16/32/64/128/256/512/1024/10. Total 2050 neuronas

### Gráfico de tendencias



## Comparación de métricas en entrenamiento y validación:



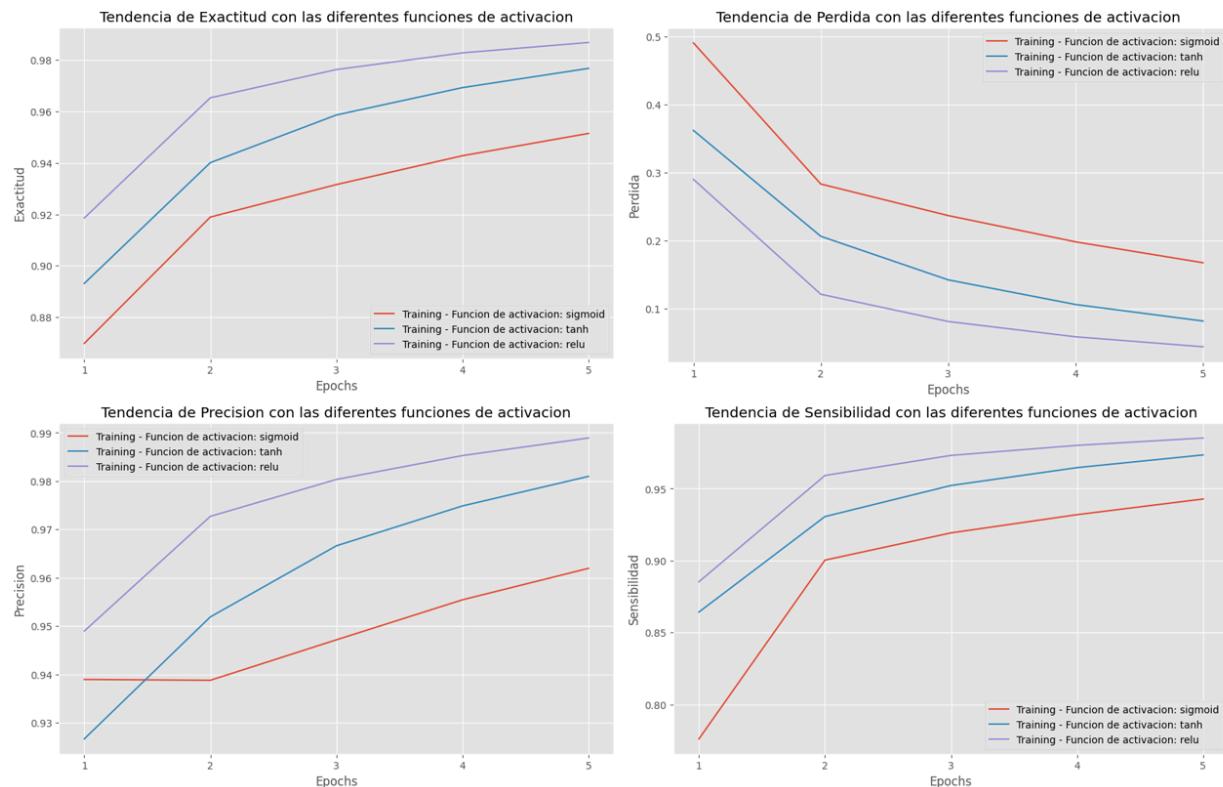
Los cinco modelos están bien ajustados pues a pesar de tener métricas altas, tienen un desempeño similar tanto en entrenamiento y validación.

En este escenario se utilizó una estrategia similar que al disminuir el número de neuronas por capa, pero esta vez empezando con muy pocas neuronas y terminando con una estructura densa. De esto se evidencia que con 6 capas y una arquitectura de 8/16/32/64/128/256/10 se consiguió el mejor desempeño en todas las métricas pero en este escenario no existe una distinción clara entre qué modelo fue el peor a menos que se priorice alguna métrica.

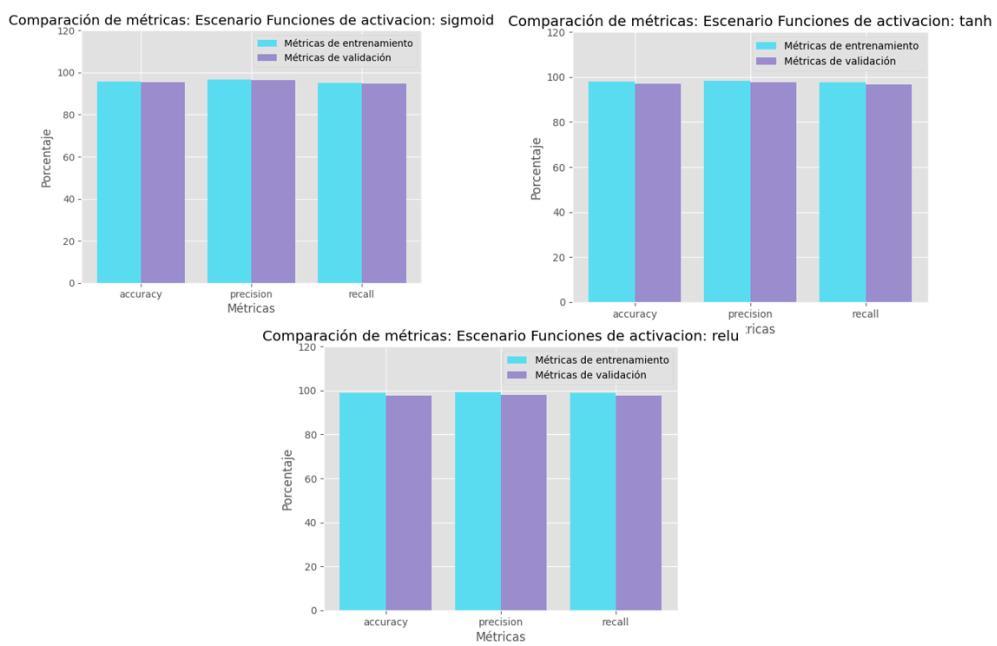
En general, este escenario dio los peores resultados, principalmente debido a que al haber una cantidad tan baja de neuronas al principio, no se extraen correctamente los atributos de cada número. Además el conjunto de datos de MNIST no requiere de estructuras tan complejas para poder predecir acertadamente.

## Función de activación: “sigmoid”, “tanh” y “relu”.

### Gráfico de tendencias



### Comparación de métricas en entrenamiento y validación:



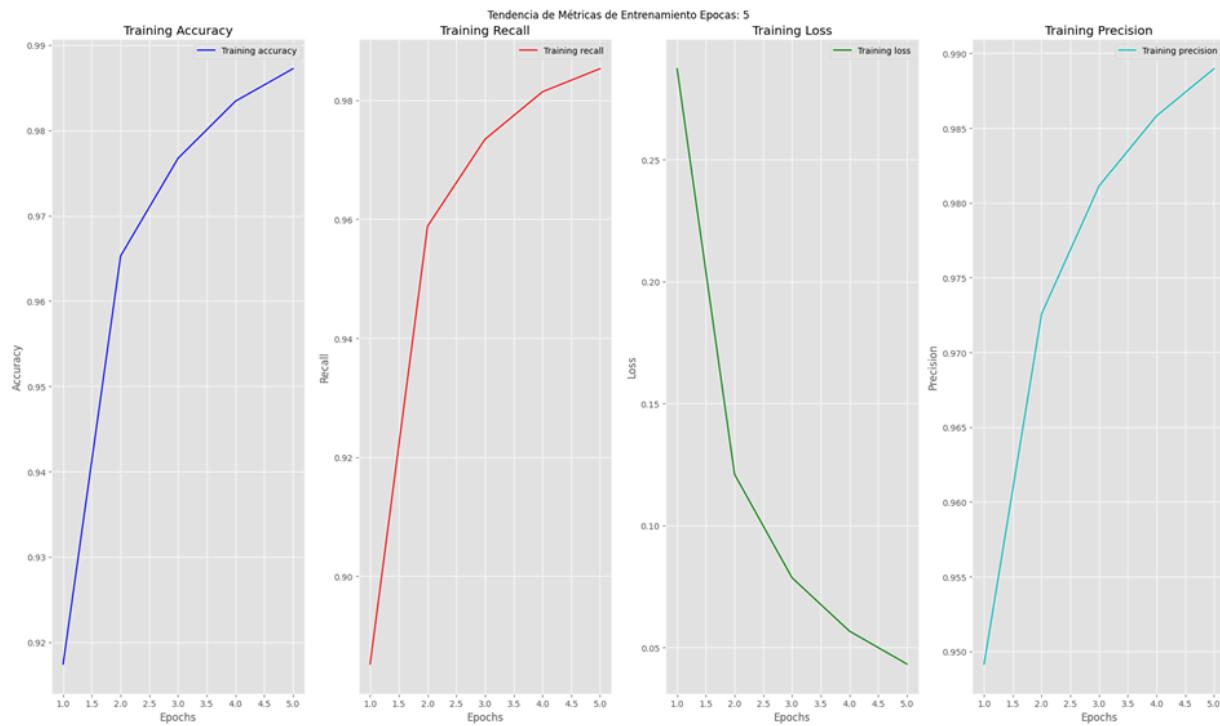
Los tres modelos están bien ajustados pues a pesar de tener métricas altas, tienen un desempeño similar tanto en entrenamiento y validación.

En este escenario encontramos la mayor variación entre modelos. El mejor sin duda fue en el que se usó Relu, esto probablemente gracias a que su rango es de  $[0, \infty)$ , por lo que puede mandar a través de la red valores realmente grandes pero nunca valores negativos, a diferencia de funciones con un rango acotado como Sigmoid (entre 0 y 1) o tanh (entre -1 y 1). Además, estas dos últimas funciones son generalmente más útiles para reconocer patrones en bases de datos más complejas, pero en el caso de MNIST terminan siendo contraproducentes.

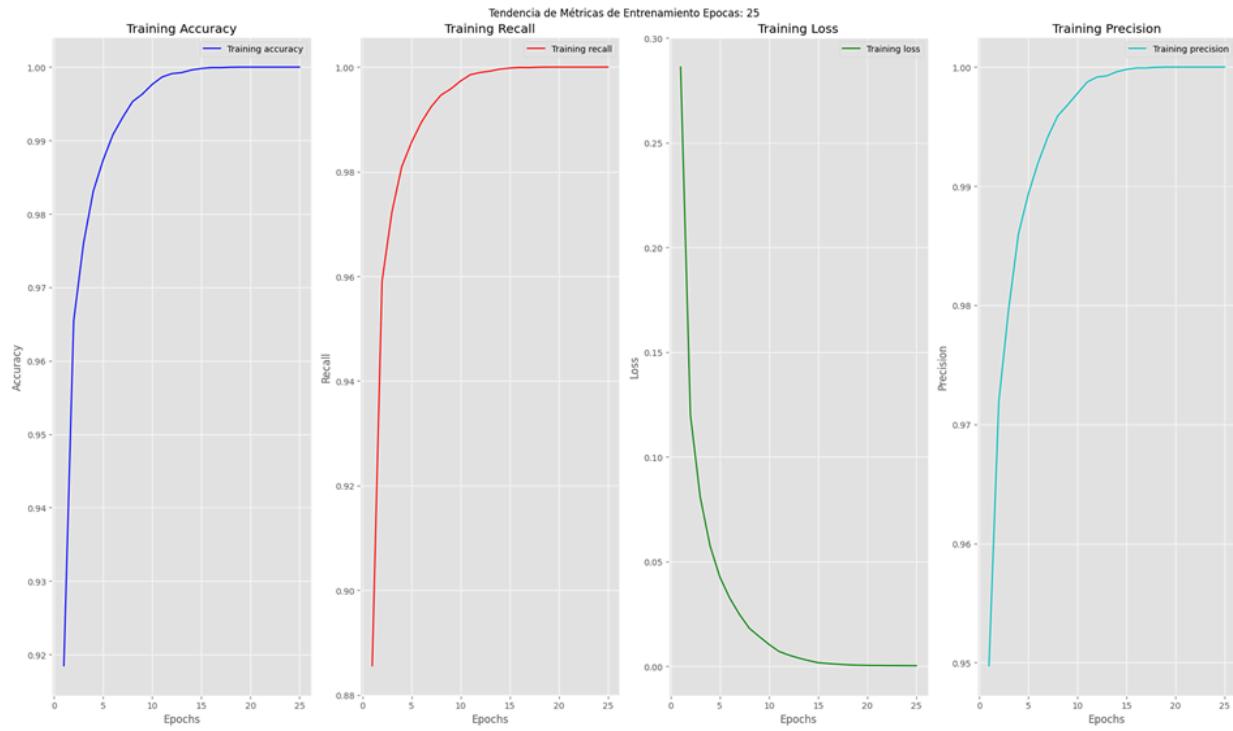
## Número de épocas: 5, 25, 50 y 100.

### Gráficos de tendencias

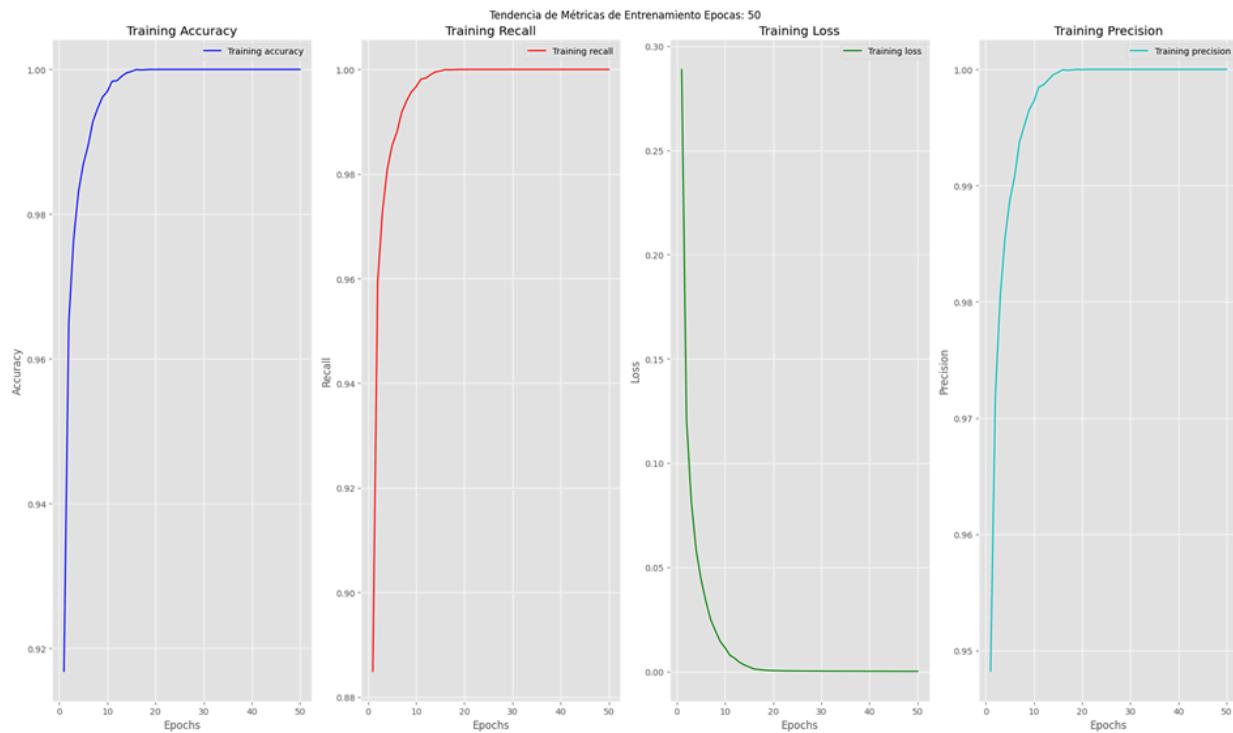
5 épocas:



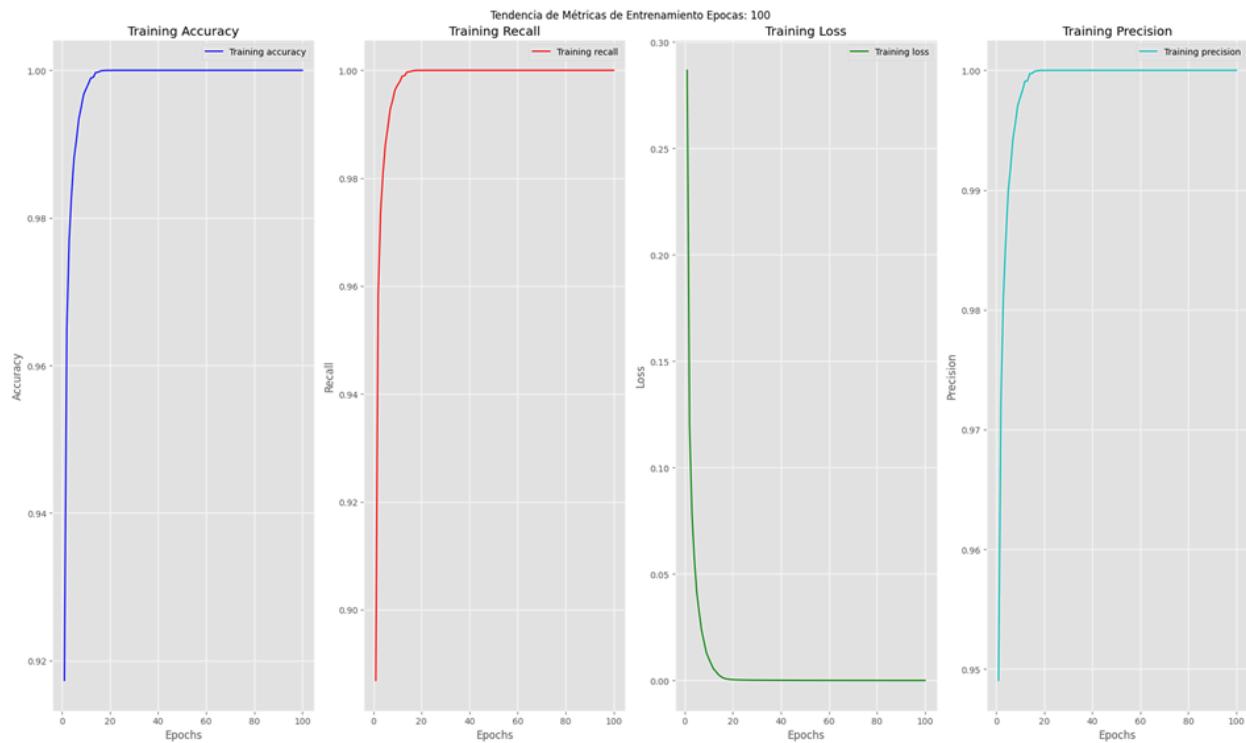
## 25 épocas:



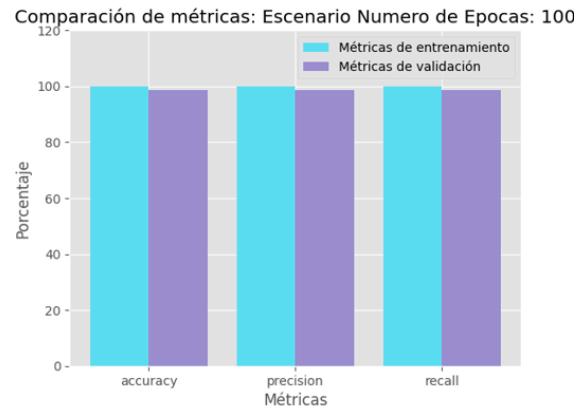
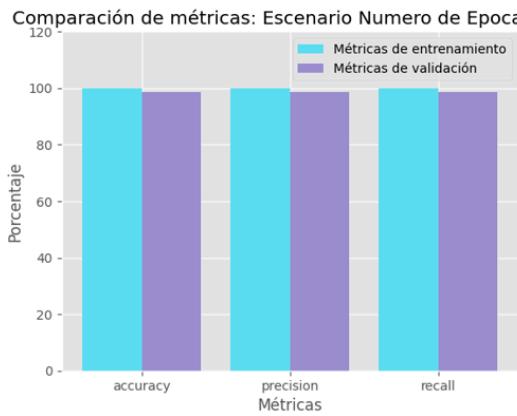
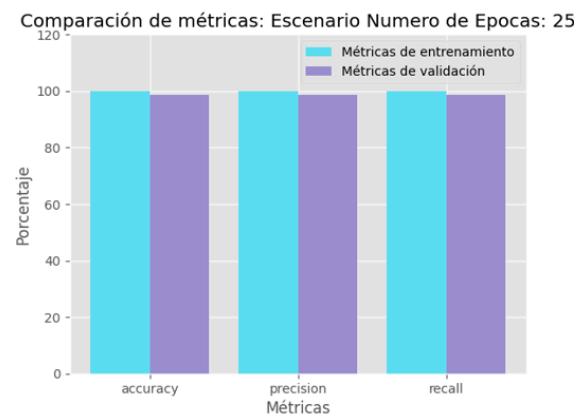
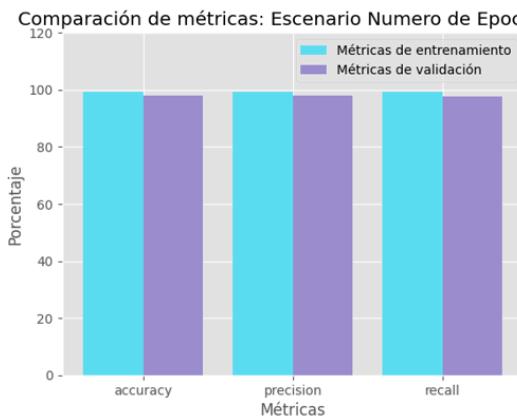
## 50 épocas:



## 100 épocas



## Comparación de métricas en entrenamiento y validación:



Con 5 épocas está bien ajustado y el resto están sobre justados pues llegan al 100% en las métricas, es decir que se aprendieron los datos.

De este escenario podemos deducir que, como la base de datos MNIST es relativamente sencilla de entrenar, utilizar demasiadas épocas solo hace que el modelo termine aprendiéndose los datos una y otra vez, lo cual se evidencia en las gráficas, pues todos llegan a un 100% de desempeño y 0 pérdida alrededor de la época 15; a pesar de que no se pierde la capacidad de predecir (como se ve en los resultados del conjunto de validación), se gasta una mayor capacidad computacional sin ser realmente necesario. Asimismo, teniendo en cuenta que la arquitectura base es bastante simple, no se requiere un número tan alto de épocas para poder entrenar adecuadamente los modelos.

#### **B. ¿Debería la exactitud por sí sola ser el criterio para decidir el ajuste de un hiper-parámetro? ¿Cuáles podrían ser otras consideraciones en la práctica?**

Desde nuestra perspectiva, para modificar como tal un hiper-parámetro si se podría ver solo la exactitud por sí sola, ya que a primera vista nos podría estar diciendo que tan bien está clasificando el modelo y si esta métrica es muy baja, ya podríamos decidirnos por modificar algún hiper-parámetro. Por otro lado, para decisiones finales de los hiper-parámetros, escogencia de modelos, etc. Si tendríamos que tener en cuenta más métricas como la sensibilidad o la precisión.

Otras consideraciones que podríamos tomar es tener en cuenta el pre-procesamiento de los datos y la distribución por cada clase, ya que esto nos ayudaría a eliminar sesgos desde el principio y a que los modelos no tengan un favoritismo hacia alguna clase en específico. Por ejemplo, en el conjunto de datos que estamos manejando se hizo un correcto pre-procesamiento y en el conjunto de entrenamiento se encuentran estas distribuciones, que para nuestro criterio están bien niveladas y lo mismo para el conjunto de validación y prueba.

Clase (Número)	Cantidad entrenamiento	Cantidad validación	Cantidad prueba
0	4939	984	980
1	5649	1093	1135
2	4964	994	1032
3	5131	1000	1010
4	4862	980	982
5	4502	919	892
6	4937	981	958
7	5205	1060	1028
8	4872	979	974
9	4939	1010	1009

**Distribución por clase  
en cada conjunto.**

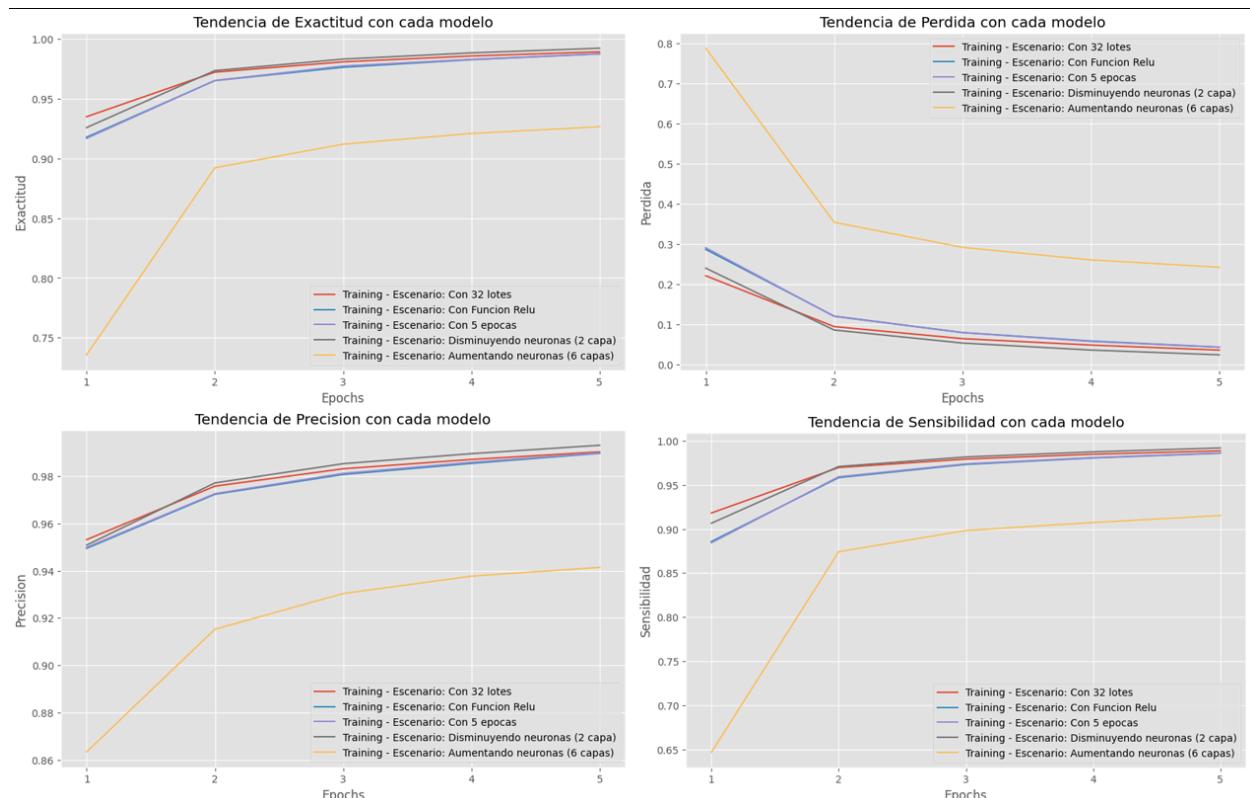
En este mismo sentido, en esta tabla podemos apreciar la distribución de las clases en cada uno de los conjuntos. Al haber mayor cantidad de imágenes del número 1 y una menor cantidad del 5 en el de entrenamiento, en los modelos podría haber cierto sesgo por lo que se debe tener en cuenta a la hora de ajustar los hiper-parámetros.

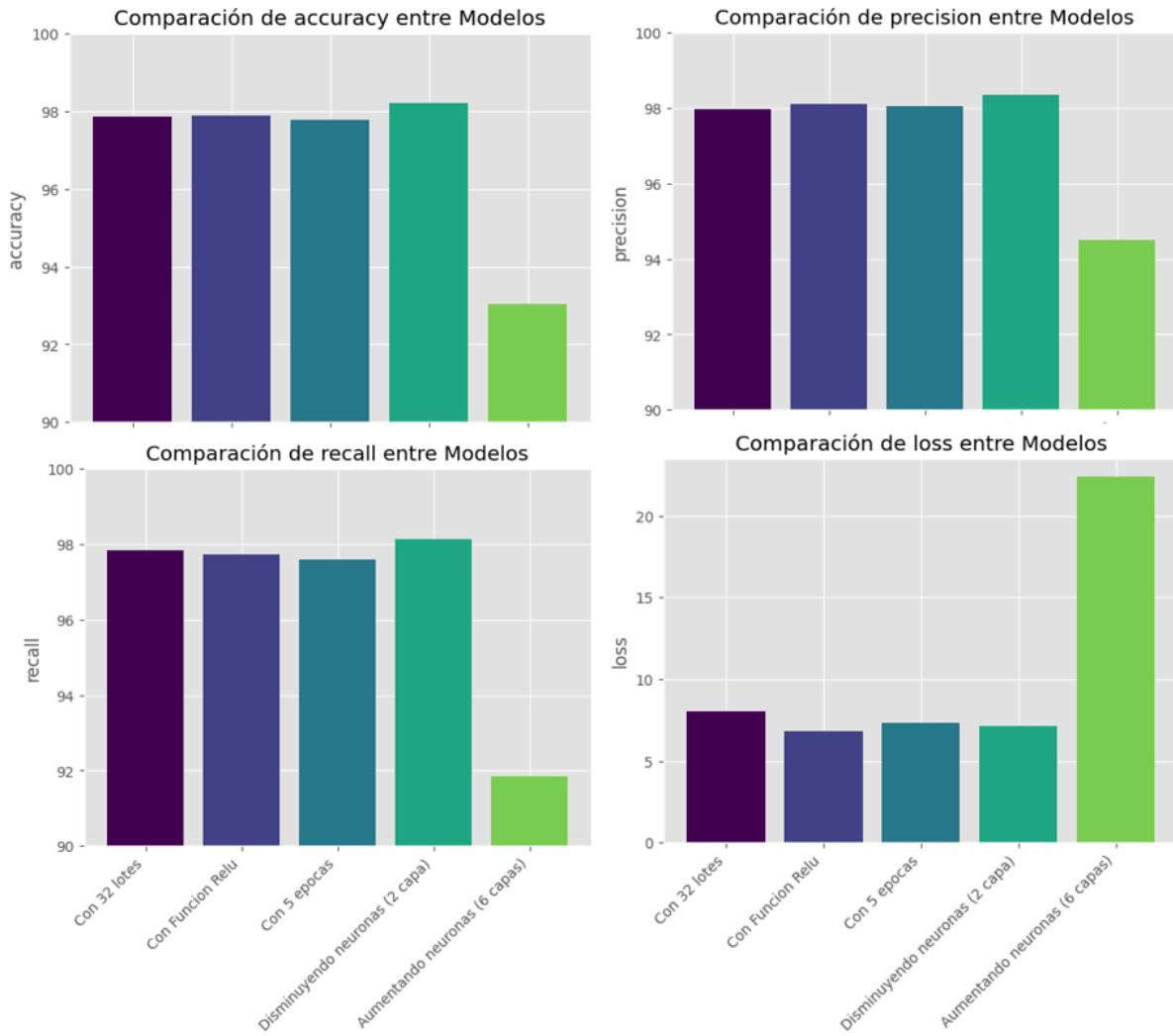
Asimismo, cuando nos enfrentemos a estas situaciones en la práctica, necesitamos entender el contexto en el que se encuentra el conjunto de datos, esto debido a que se le puede dar mayor prioridad a una métrica que a otra; por ejemplo en un caso médico, deberíamos de darle mayor importancia a la sensibilidad para poder modificar los hiper-parámetros. En este caso, al ser un proyecto académico, no es necesario tener en cuenta esa distinción.

### C. ¿Cuál es el “mejor” modelo en cada escenario?

- **Tamaño del lote:** El modelo con 32 lotes. Se destacó en todas las métricas.
- **Funciones de activación:** El modelo con Relu. Se destacó en todas las métricas.
- **Número de épocas:** El modelo con 5 épocas al ser el único bien ajustado
- **Disminuyendo neuronas:** 2 capas y una estructura de neuronas 1024/512/10. Se destacó en todas las métricas.
- **Aumentando neuronas:** 6 capas y una estructura de neuronas 8/16/32/64/128/256/10. Se destacó en todas las métricas.

### ¿Cuál es el mejor modelo entre los mejores?



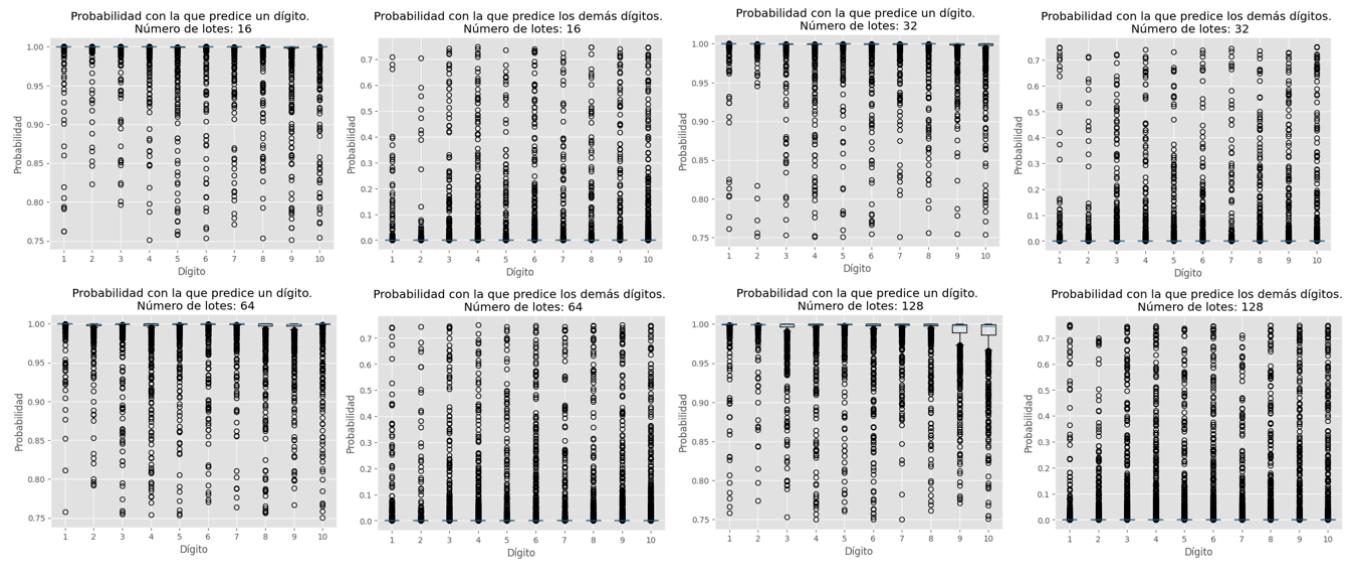


Realizamos un gráfico de tendencia y una comparación de las métricas en el conjunto de validación entre los modelos con los mejores desempeños. En cuanto al conjunto de validación, los modelos que usa la función que activación Relu y una estructura que va disminuyendo neuronas con 2 capas tienen resultados similares, pero finalmente obtuvimos que el mejor modelo es el de 2 capas y con la estructura 1024/512/10, pues su gráfico de tendencia nos arroja mejores resultados.

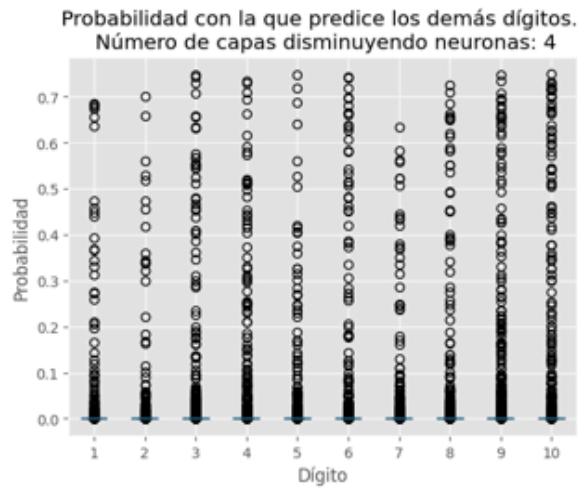
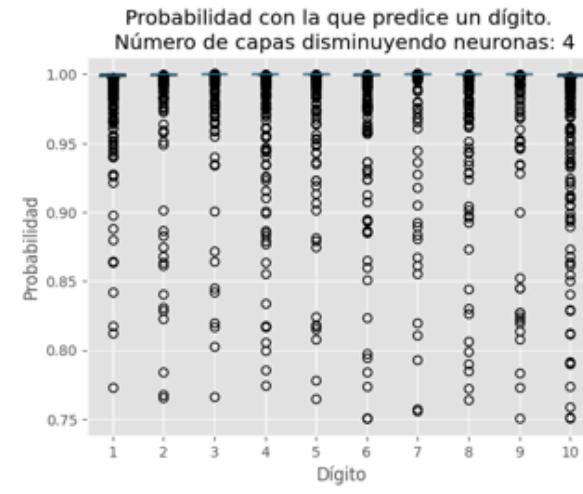
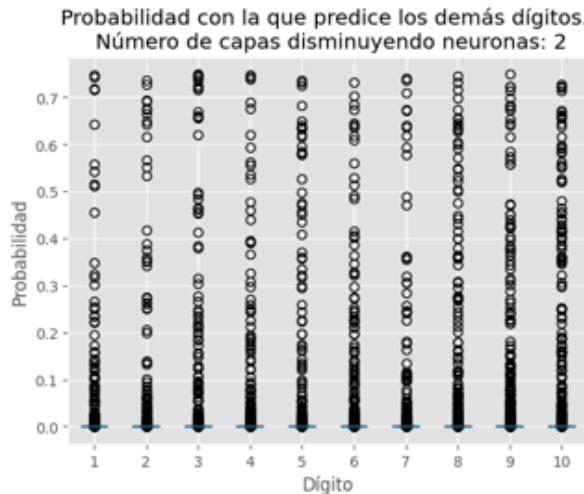
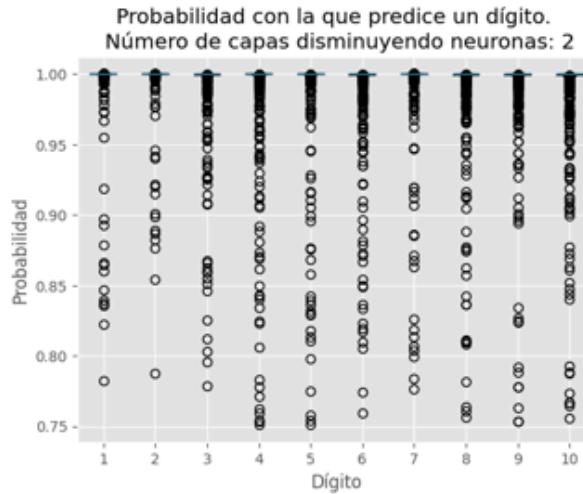
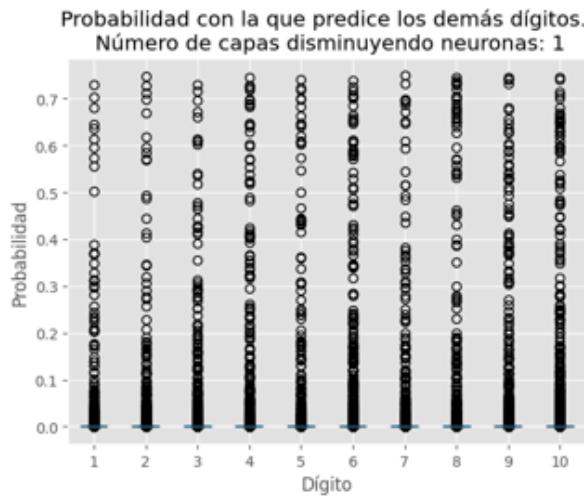
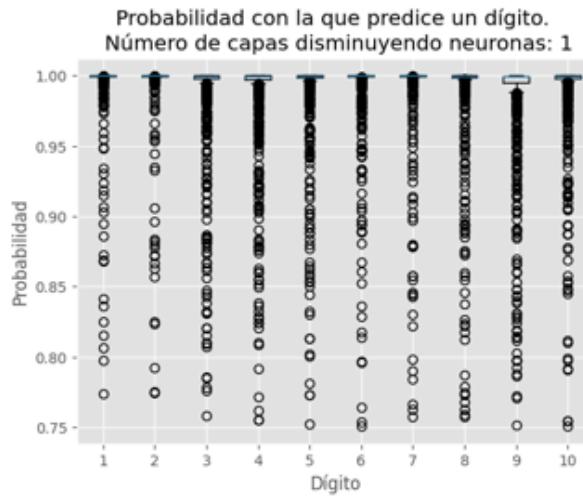
## Asignación 2: Análisis de certeza de predicción

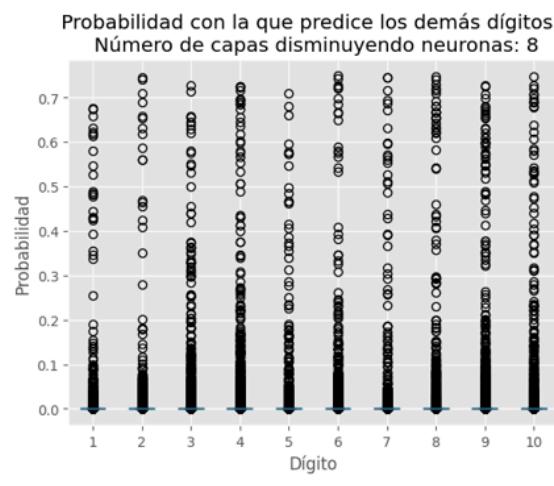
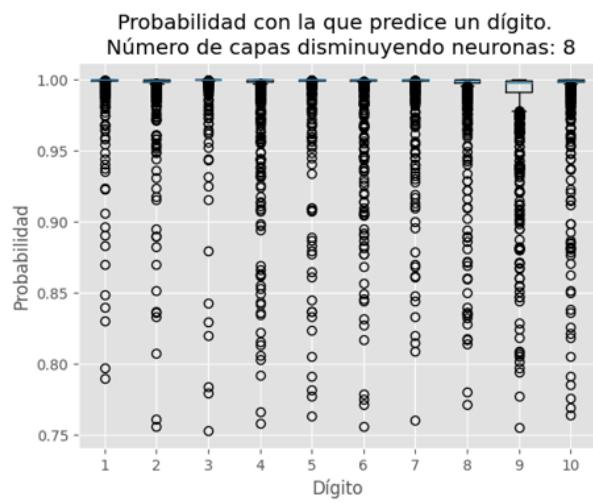
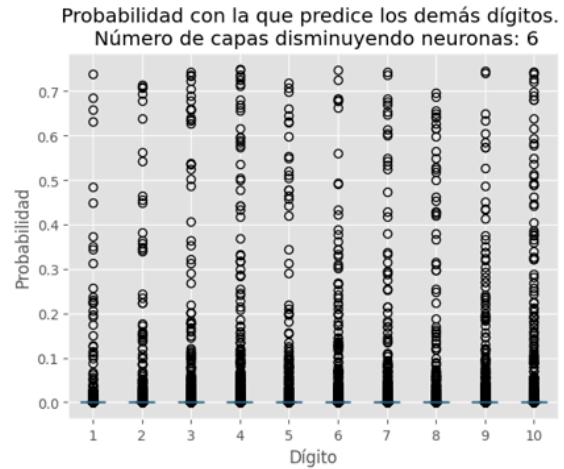
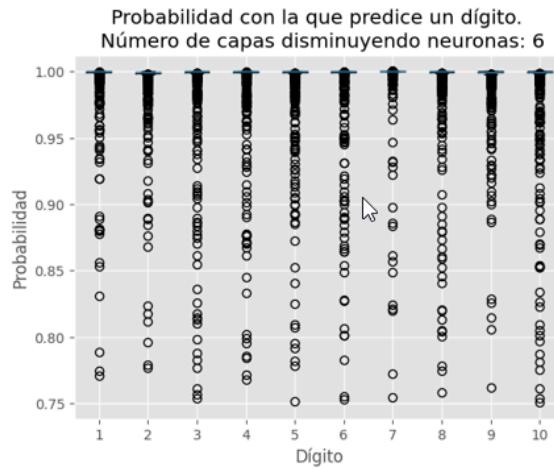
**A. Grafique las probabilidades de predicción de los modelos establecidos en la asignación 1. Utilice en cada caso boxplots, para ilustrar la distribución de dichas probabilidades.**

### Tamaño de lote: 16, 32, 64 y 128

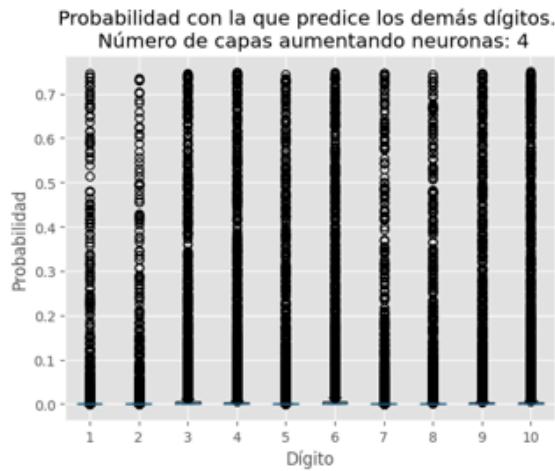
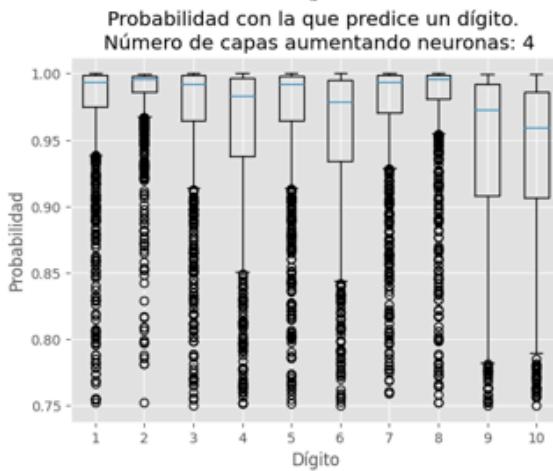
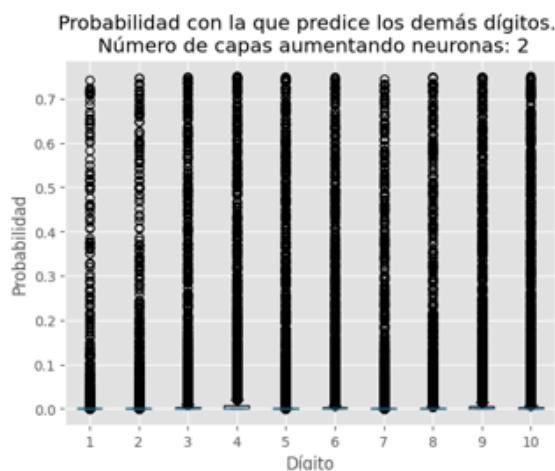
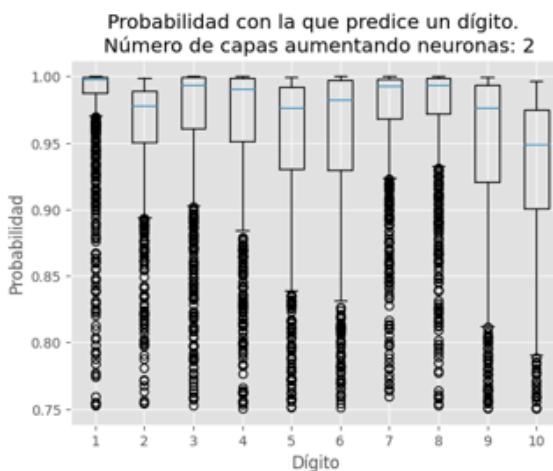
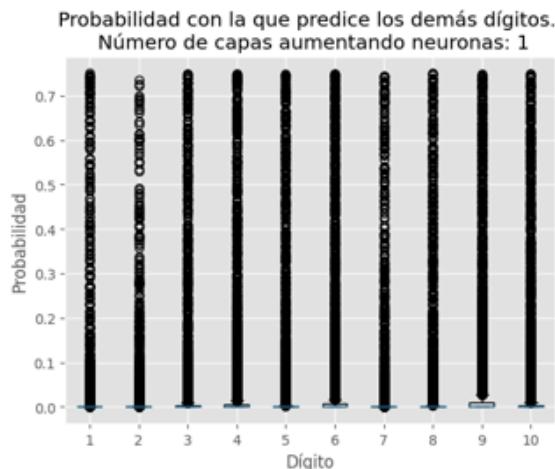
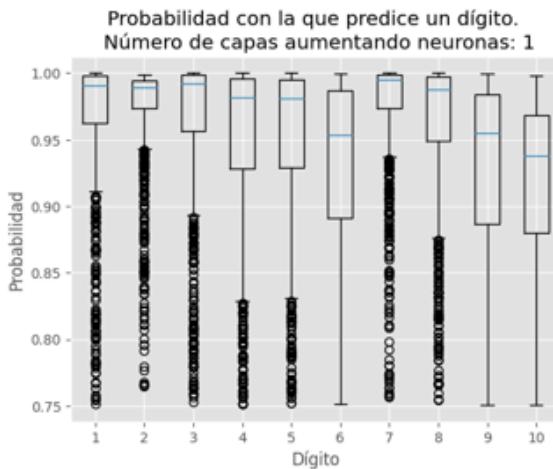


## Número de capas ocultas: 1, 2, 4, 6, 8 disminuyendo progresivamente el número de neuronas por capa.

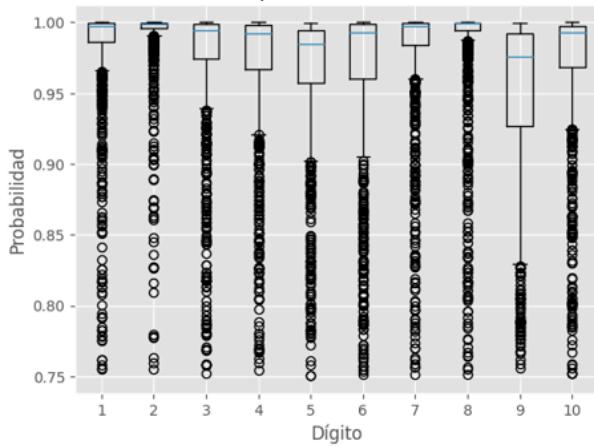




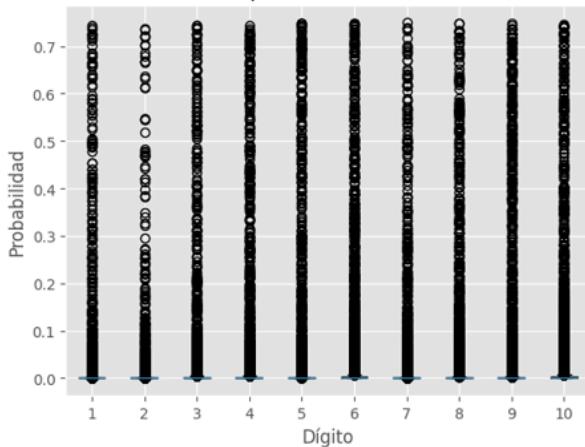
## Número de capas ocultas: 1, 2, 4, 6, 8 aumentando progresivamente el número de neuronas por capa.



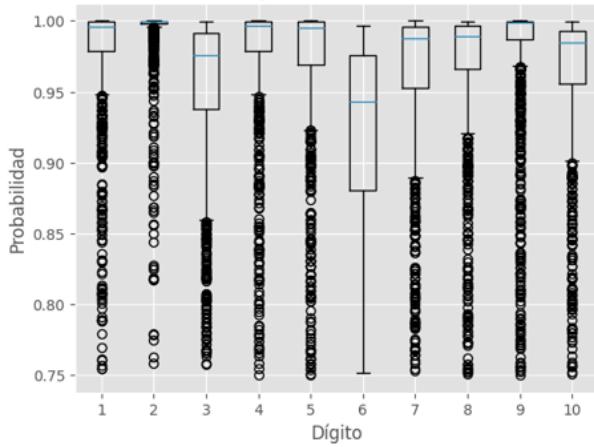
Probabilidad con la que predice un dígito.  
Número de capas aumentando neuronas: 6



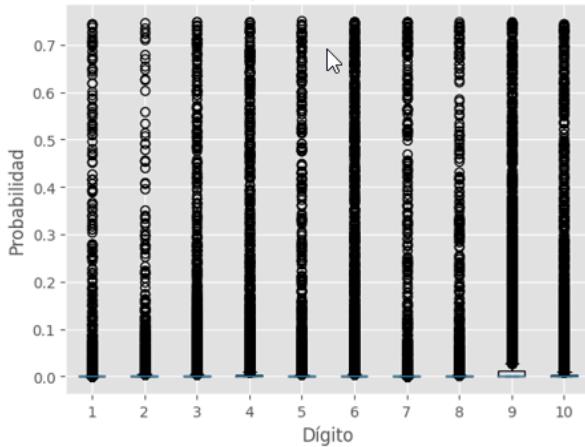
Probabilidad con la que predice los demás dígitos.  
Número de capas aumentando neuronas: 6



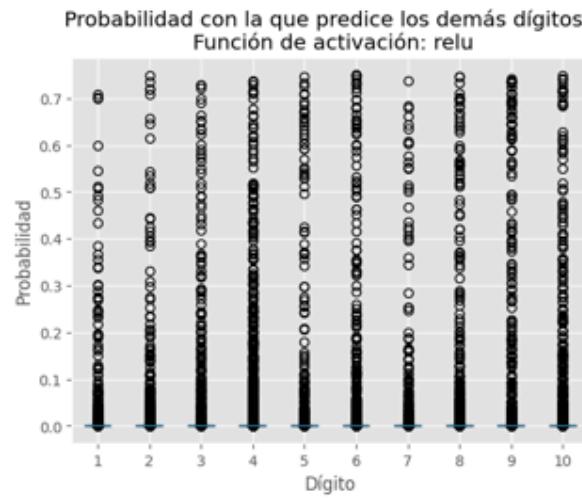
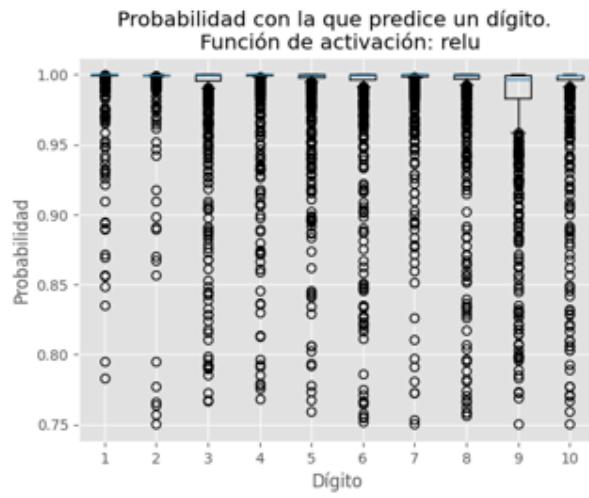
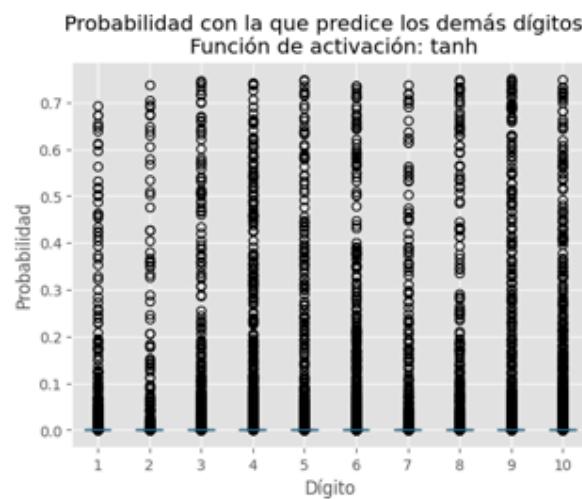
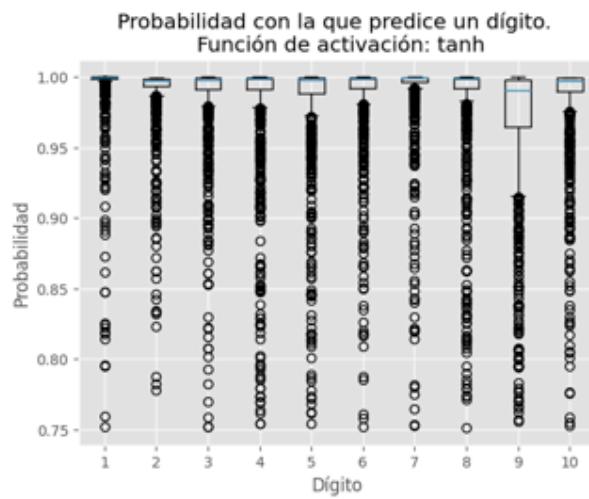
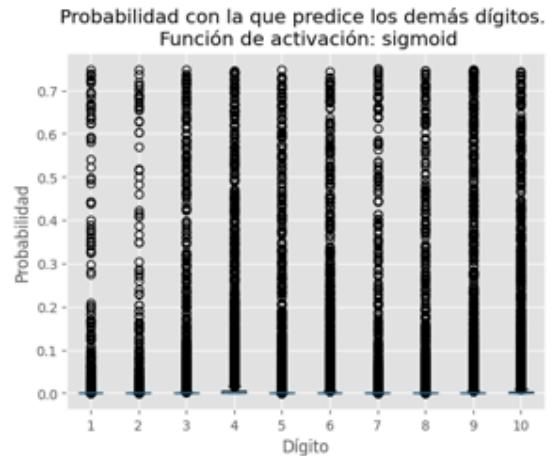
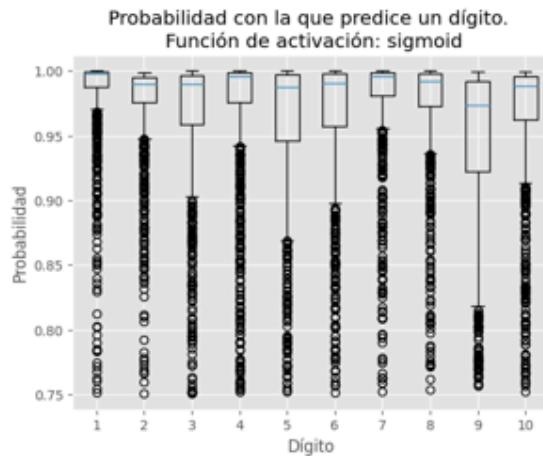
Probabilidad con la que predice un dígito.  
Número de capas aumentando neuronas: 8



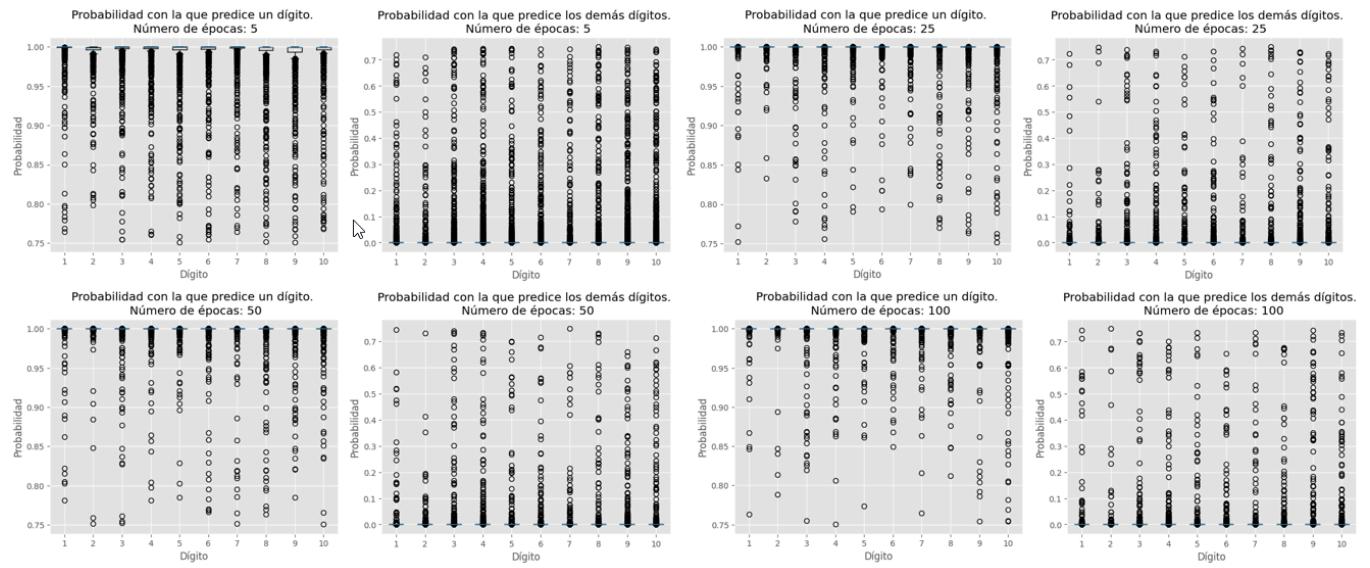
Probabilidad con la que predice los demás dígitos.  
Número de capas aumentando neuronas: 8



## Función de activación: “sigmoid”, “tanh” y “relu”.



## Número de épocas: 5, 25, 50 y 100.



En primera instancia cabe aclarar que hay 2 gráficas por cada modelo, ya que una gráfica nos muestra las probabilidades con las que los modelos predicen el dígito correcto, pero en la otra gráfica agregamos las probabilidades restantes que se dejan para los demás dígitos después de cada predicción. Un problema de estas gráficas es que es muy difícil su interpretación, visualización y de la misma manera ver sus estadísticas con tantos outliers.

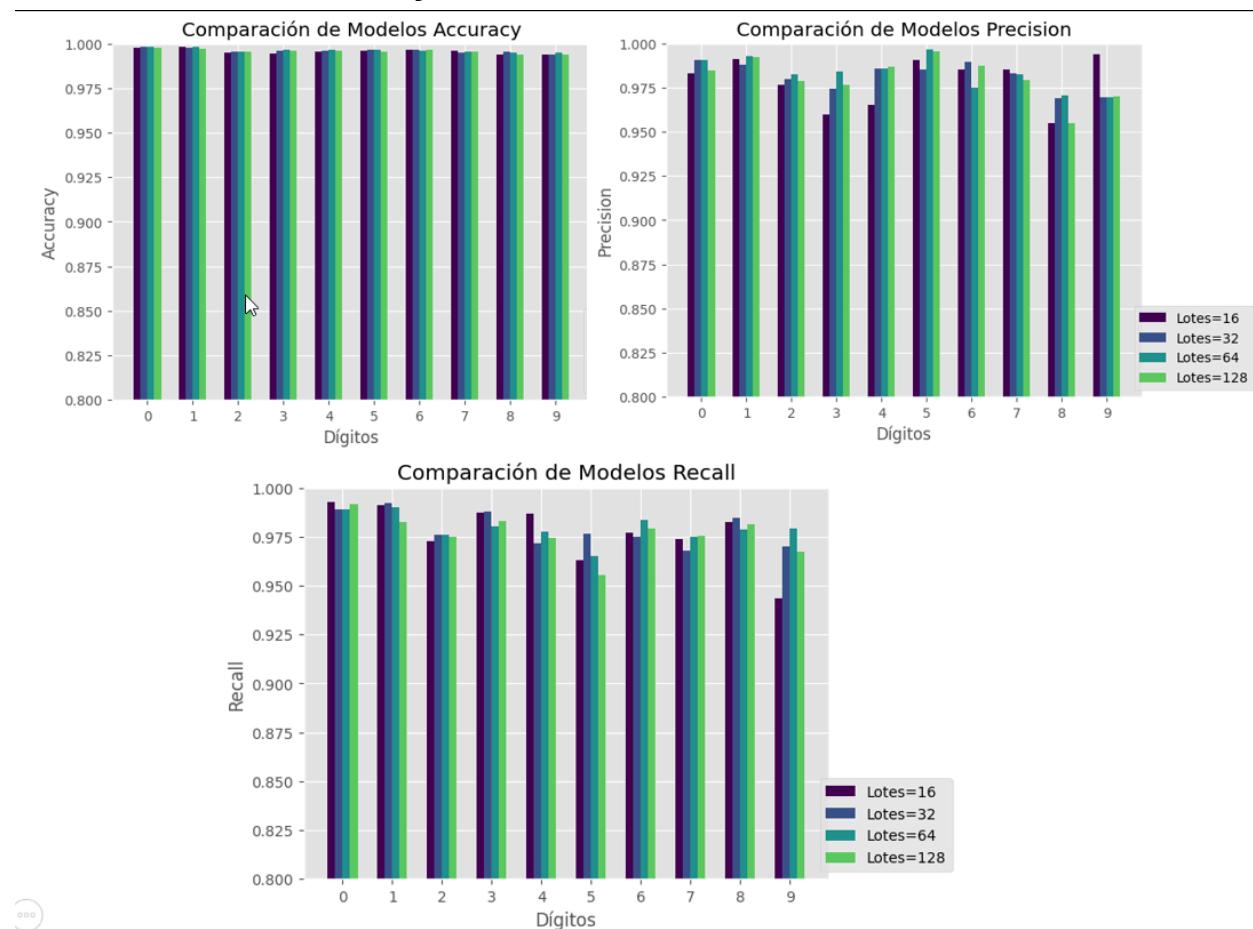
No obstante, se evidencia que en la mayoría de los casos, las cajas de los boxplots parecieran ser solo una línea. Esto quiere decir que existe una variación muy baja entre los datos que se encuentran entre los cuartiles 1 y 3, donde se encuentran el 50% de los datos. Lo demás que se evidencia es que hay una gran cantidad de outliers, pero a pesar de esto, las probabilidades encontradas son generalmente uniformes.

## **B. ¿Cuál es el modelo más “seguro” en cada escenario?**

En las siguientes gráficas se muestra el accuracy, la precisión y sensibilidad (recall) de los modelos de cada escenario cuando estos predicen los datos del conjunto de prueba y los separamos en qué tan bien predicen cada clase.

Generalmente el accuracy tiene resultados similares en todos los modelos, por lo que en este caso le dimos más relevancia a la sensibilidad, pues esta se calcula respecto a los verdaderos positivos y falsos negativos, es decir frente a los datos verdaderos. En caso de que no sea claro qué modelo es más seguro, es decir, cuál predice correctamente más dígitos, revisamos también la precisión.

## Tamaño de lote: 16, 32, 64 y 128



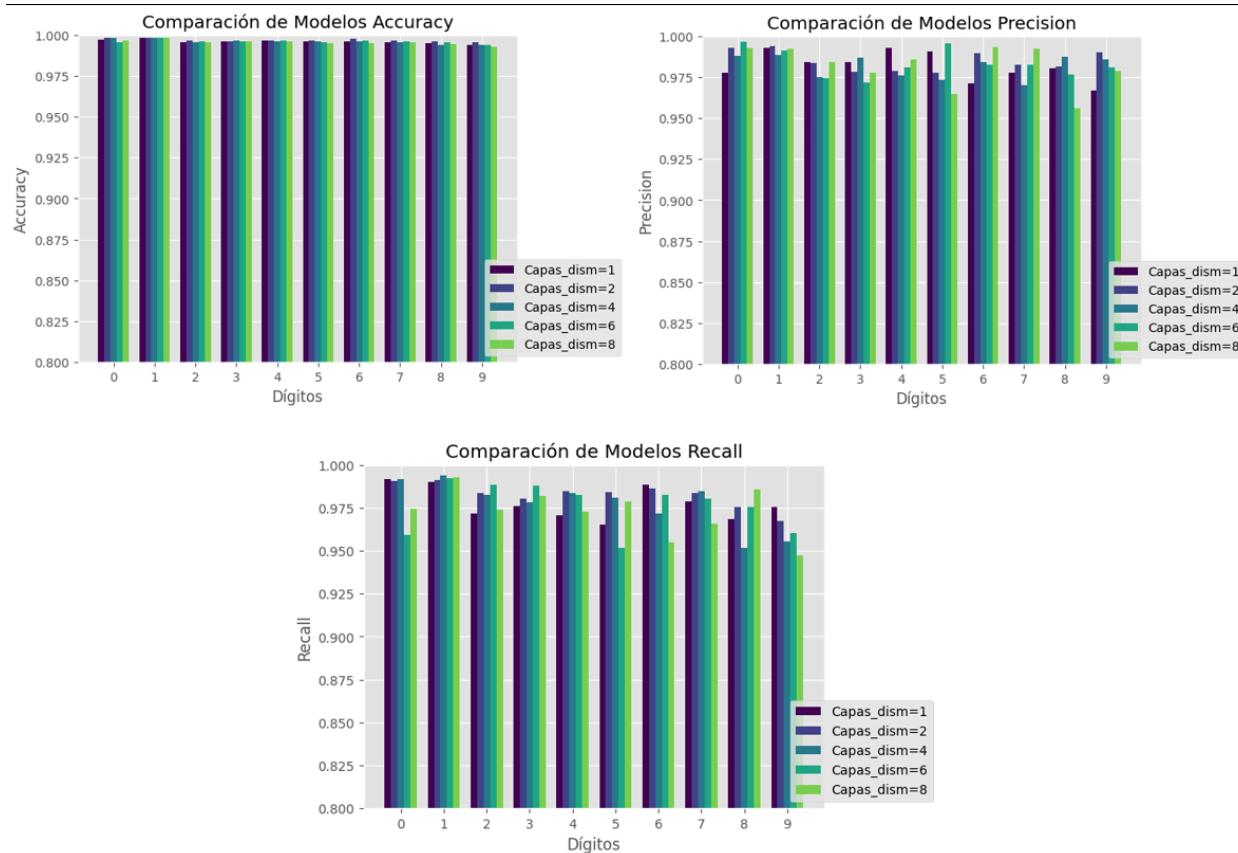
¿Qué número clasifica mejor cada modelo?

Modelos	Sensibilidad	Precisión
16 lotes	0 - 4	7 - 9
32 lotes	1 - 2* - 3 - 5 - 8	0* - 2
64 lotes	2* - 6 - 9	0* - 1 - 2 - 3 - 5 - 8
128 lotes	7	4

\* Varios modelos obtuvieron el mismo resultado al clasificar dicho número.

A pesar que el modelo de 64 lotes obtuvo muy buenos resultados en precisión, nosotros al querer centrarnos en la sensibilidad consideramos que el modelo más seguro es el de 32 lotes al tener el mejor desempeño en esta métrica.

## Número de capas ocultas: 1, 2, 4, 6, 8 disminuyendo progresivamente el número de neuronas por capa.



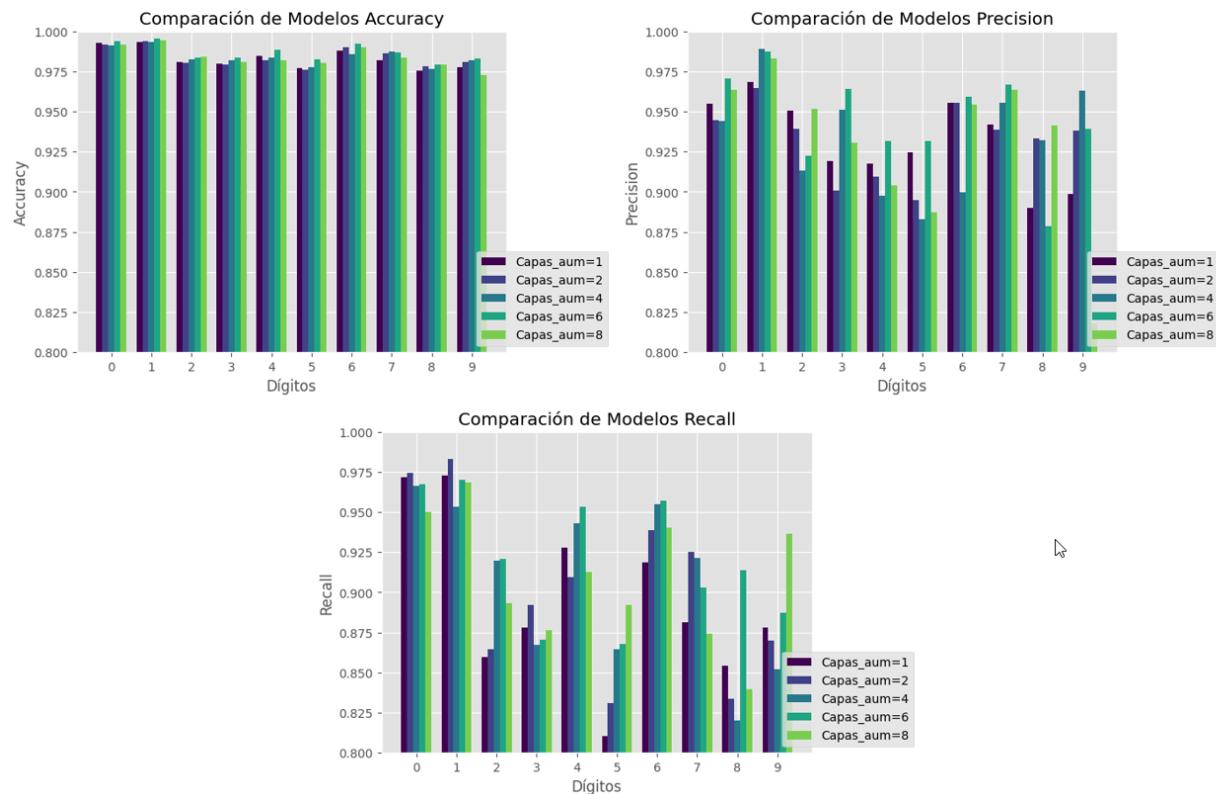
¿Qué número clasifica mejor cada modelo?

Modelos	Sensibilidad	Precisión
1 capa	0* - 6 - 9	2 - 4
2 capas	4 - 5	1 - 9
4 capas	0* - 1 - 7	3 - 8
6 capas	2 - 3	0 - 5
8 capas	8	6 - 7

\* Varios modelos obtuvieron el mismo resultado al clasificar dicho número.

Los modelos con 1 y 4 capas obtuvieron exactamente los mismos resultados en sensibilidad y precisión por lo que para escoger el más seguro, decidimos tener en cuenta la capacidad computacional y cuál modelo utiliza menos recursos. En este sentido, tomaremos como el más seguro al de 1 capa, pues el dataset de MNIST, como hemos mencionado antes, no requiere de modelos tan robustos.

**Número de capas ocultas: 1, 2, 4, 6, 8 aumentando progresivamente el número de neuronas por capa.**

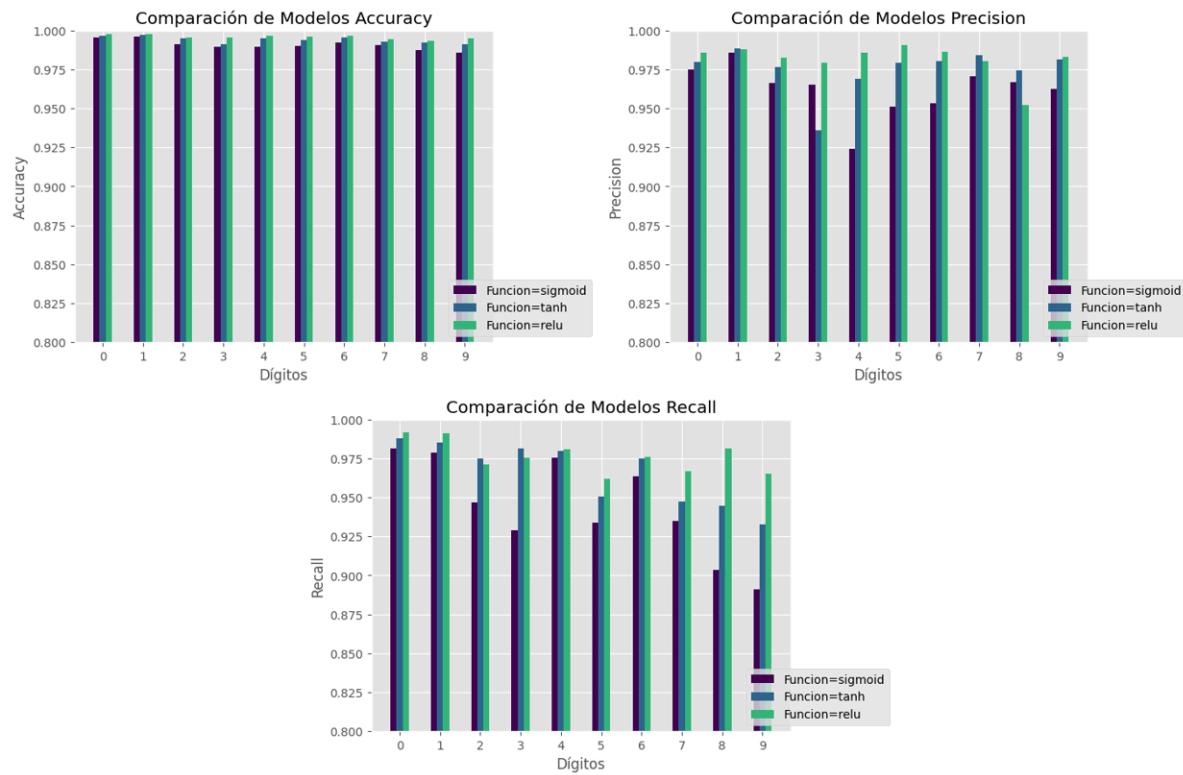


¿Qué número clasifica mejor cada modelo?

Modelos	Sensibilidad	Precisión
1 capa	-	-
2 capas	0 - 1 - 3 - 7	-
4 capas	-	1 - 9
6 capas	2 - 4 - 6 - 8	0 - 3 - 4 - 5 - 6 - 7
8 capas	5 - 9	2 - 8

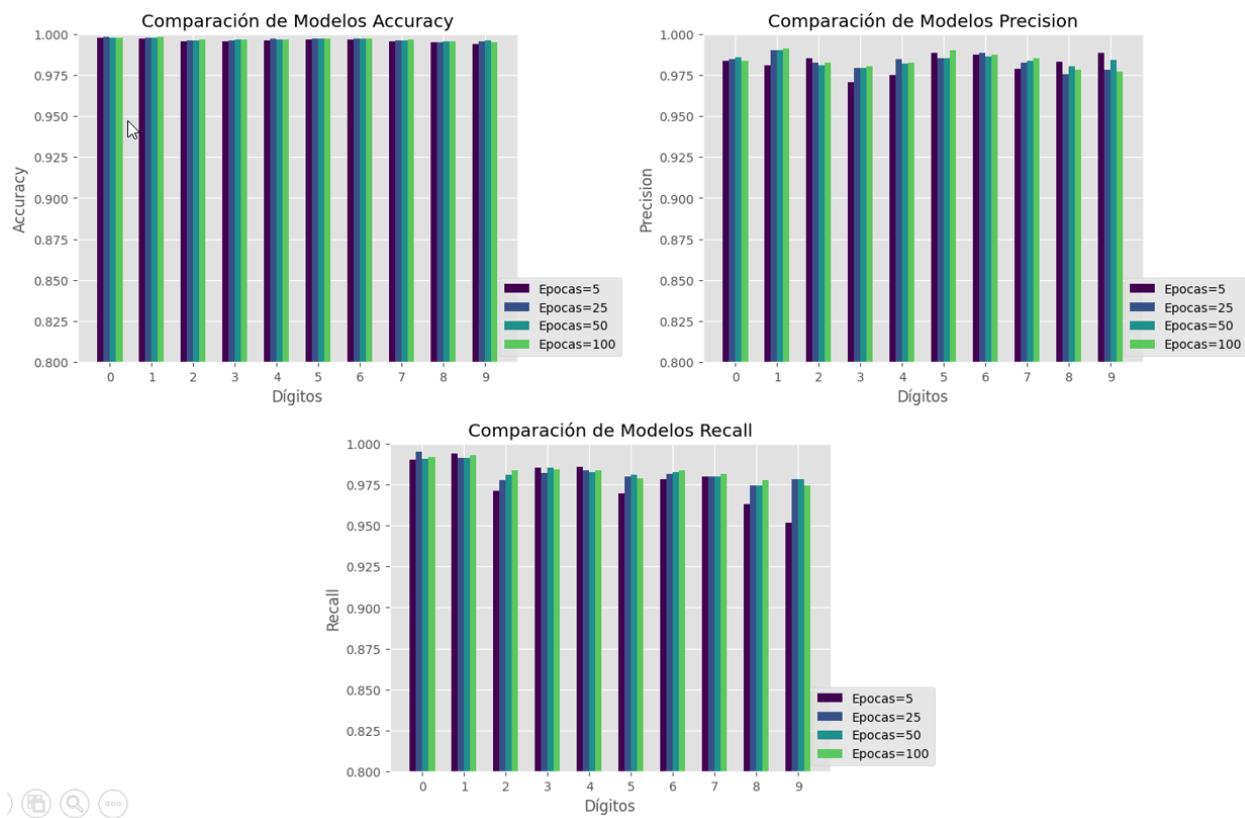
En general, el modelo con 6 capas y neuronas 8/16/32/64/128/256/10 es el mejor de este escenario, pero debemos tener en cuenta que usando esta estructura obtenemos los peores modelos, cosa que se evidencia en la gráfica, pues hay incluso algunos modelos con métricas en 80%, que para el conjunto de datos MNIST son resultados pésimos.

## Función de activación: “sigmoid”, “tanh” y “relu”.



En todas las métricas, el modelo que usó Relu se destacó al predecir con mejores métricas la gran mayoría de los números, a excepción del 2, 3 y 4 en sensibilidad y el 7 y 8 en precisión, pero por poca diferencia, por lo que es el más seguro del escenario.

## Número de épocas: 5, 25, 50 y 100.



### ¿Qué número clasifica mejor cada modelo?

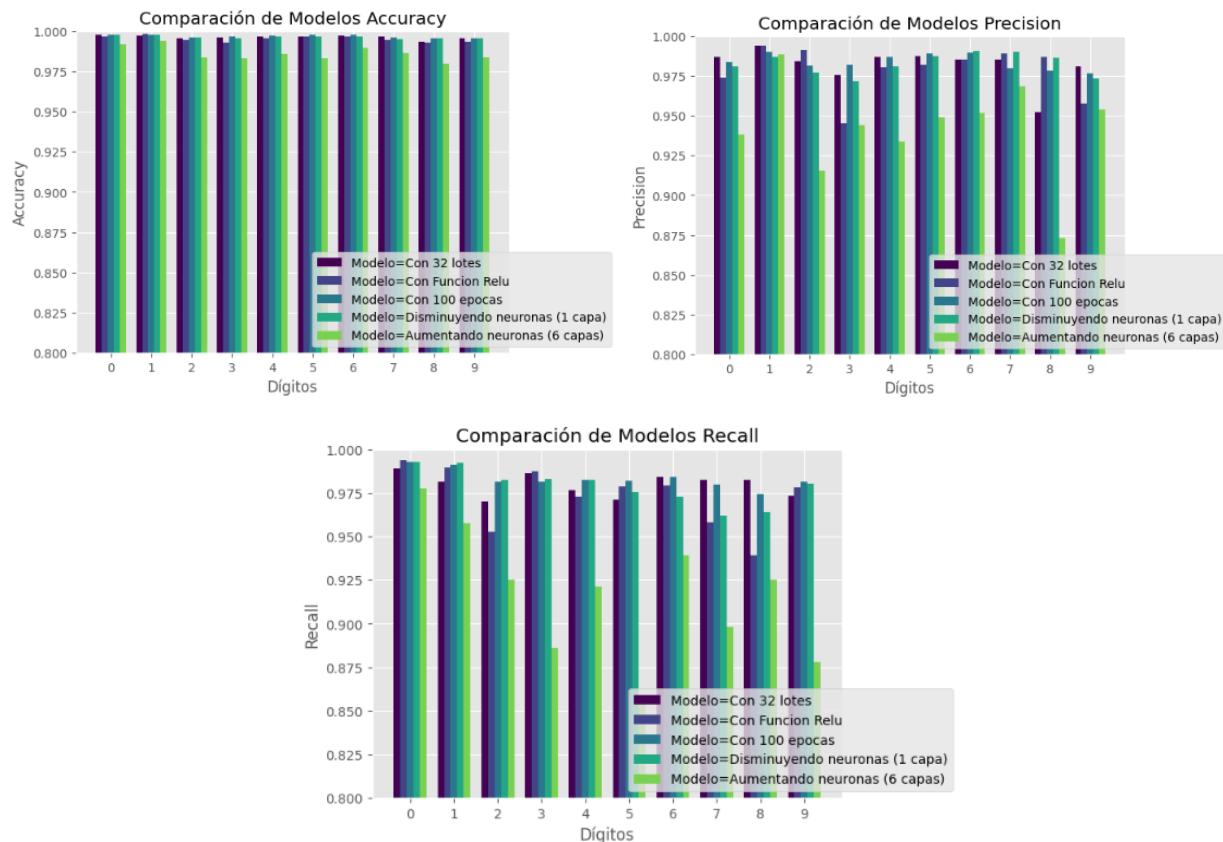
Modelos	Sensibilidad	Precisión
5 épocas	1 - 3* - 4	2 - 8 - 9
25 épocas	0 - 9*	4 - 6
50 épocas	3* - 5 - 9*	0
100 épocas	2 - 6 - 7 - 8	1 - 3 - 5 - 7

\* Varios modelos obtuvieron el mismo resultado al clasificar dicho número.

Los modelos con 5 y 100 épocas obtuvieron excelentes resultados en sensibilidad, prediciendo adecuadamente casi la mitad de los dígitos, pero al revisar la precisión nos damos cuenta que el modelo con 100 épocas nos da mayor seguridad en este escenario.

## ¿Cuál es el modelo más “seguro” entre los mejores?

Entre los mejores modelos realizamos la siguiente comparación:



## ¿Qué número clasifica mejor cada modelo?

Modelos	Sensibilidad	Precisión
32 lotes	6* - 7 - 8	0 - 1* - 4* - 9
Relu	0 - 3	1* - 2 - 8
100 épocas	4* - 6* - 9	3 - 4* - 5
1 capa dismin. neuronas	1 - 2 - 4* - 5	6 - 7
6 capas aumen. neuronas	-	-

\* Varios modelos obtuvieron el mismo resultado al clasificar dicho número.

En sensibilidad, el modelo de 1 capa disminuyendo neuronas tuvo muy buenos resultados al predecir con mejores resultados 4 dígitos. Cabe aclarar que a excepción del modelo de 6 capas, no hubo una diferencia significativa entre los modelos escogidos, solo que el de 1 capa se destacó un poco más. Aún así, consideramos que en cuanto a las predicciones en el conjunto de prueba este es el modelo más seguro.

**C. Analice las diferencias y similitudes de estos resultados con los obtenidos en la asignación 1.**

Escenario	Ganador asignación 1	Ganador asignación 2
Tamaño de lote: 16, 32, 64 y 128	32 lotes	32 lotes
Número de capas ocultas: 1, 2, 4, 6, 8 disminuyendo progresivamente el número de neuronas por capa.	2 capas ocultas con neuronas 1024/512/10	1 capa oculta con neuronas 1024/10
Número de capas ocultas: 1, 2, 4, 6, 8 aumentando progresivamente el número de neuronas por capa.	6 capas ocultas con neuronas 8/16/32/64/128/256/10	6 capas ocultas con neuronas 8/16/32/64/128/256/10
Función de activación: "sigmoid", "tanh" y "relu".	Relu	Relu
Número de épocas: 5, 25, 50 y 100.	5 épocas	100 épocas

**Ganador final de la asignación 1:** 2 capas ocultas con neuronas 1024/512/10

**Ganador final de la asignación 2:** 1 capa oculta con 1024/10 neuronas

En la asignación 1 revisamos cómo se comportan las redes a un nivel más general; en este caso las métricas nos indican a grandes rasgos qué modelo es mejor que otro y nos dan indicios de cómo les podría ir en el conjunto de prueba. Por otro lado, en la asignación 2 nos enfocamos en su comportamiento al predecir cada uno de los números, haciendo que se tuviera que revisar a un nivel más específico cada red para escoger la mejor.

En este caso, se entiende que si necesitamos un único modelo para la tarea de predecir números, deberíamos utilizar el de 2 capas ocultas con neuronas 1024/512/10, pues sus métricas en entrenamiento y validación nos dan buenos resultados y a nivel general haría bien la labor de predecir todos los números, pero en caso de necesitar algo más robusto, podríamos mezclar varios de los modelos que resultaron ser los más seguros en su escenario (en la asignación 2) y utilizar cada uno para poder predecir con mayor eficacia cierto grupo de dígitos.

### **Asignación 3: Visualización de una red neuronal**

En esta parte, deberá jugar con redes neuronales en una versión interactiva: Tensorflow Playground, y encontrar la arquitectura que clasifique los conjuntos de datos de "espiral" y "batalla de bastardos" usando tan pocas capas (y la cantidad de neuronas en esas capas) como sea posible.

Experimente con varias topologías para obtener una clasificación (casi) perfecta de los puntos de datos, utilizando una arquitectura mínima. Debe adjuntar capturas de pantalla del modelo que completa la clasificación.

Reporte todos sus experimentos y analícelos a profundidad.

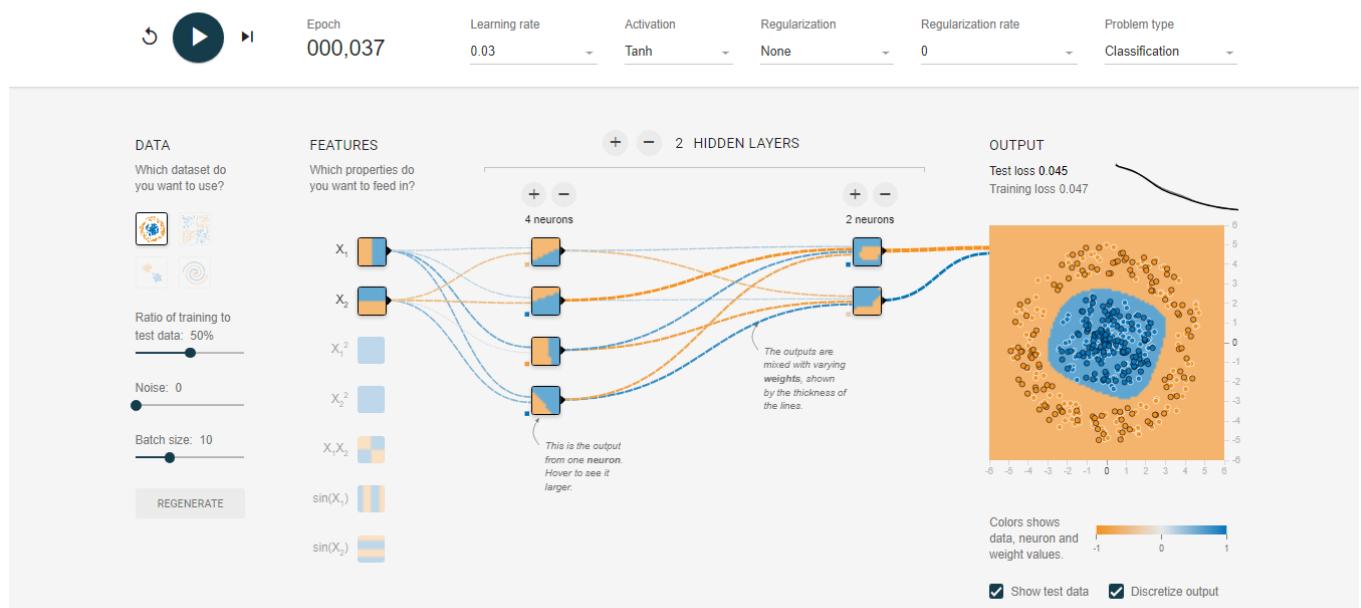
Para esta parte se usó un enfoque similar al de la primera asignación, a partir de una estructura base se modificaron uno a uno los hiper parámetros. Para decidir el mejor modelo, se escogerá el que use el menor número de épocas y el que tenga menor pérdida. El mejor modelo final es el que use la arquitectura más mínima junto a los hiper parámetros con mejores resultados.

### **Batalla de bastardos**

Estructura base (predeterminada en Tensorflow Playground):

- **Capas ocultas y neuronas:** 2 capas con estructura 4/2
- **Datos de entrenamiento:** 50% del total de datos
- **Tamaño de lotes:** 10
- **Tasa de aprendizaje:** 0.03
- **Función de activación:** Tanh
- **Épocas:** 37 (a partir de la cual ya se clasifica correctamente la información)

Resultados con la estructura base:



### **Modificando uno por uno los hiper parámetros:**

En este caso, si se llega a 500 épocas, se entiende que el modelo no pudo clasificar correctamente la información a pesar de la gran cantidad de épocas.

#### Función de activación:

Función de activación	Épocas	¿Cómo lo clasifica?
Tanh (Estructura base)	37	<p>OUTPUT Test loss 0.045 Training loss 0.047</p>
Relu	35	<p>OUTPUT Test loss 0.045 Training loss 0.047</p>
Sigmoide	370	<p>OUTPUT Test loss 0.052 Training loss 0.055</p>
Linear	+500	<p>OUTPUT Test loss 0.505 Training loss 0.493</p>

A excepción de la función linear, todas las funciones de activación clasifican bastante bien, pero se usará Relu al utilizar la menor cantidad de épocas y tener poca variación.

### Tasa de aprendizaje:

Tasa de aprendizaje	Épocas	Clasificación
0.0001	+500	<p>OUTPUT</p> <p>Test loss 0.513 Training loss 0.498</p>
0.01	152	<p>OUTPUT</p> <p>Test loss 0.065 Training loss 0.074</p>
0.03 (Estructura base)	37	<p>OUTPUT</p> <p>Test loss 0.045 Training loss 0.047</p>
0.1	25	<p>OUTPUT</p> <p>Test loss 0.024 Training loss 0.031</p>

0.3	209	<p>OUTPUT Test loss 0.026 Training loss 0.000</p>
-----	-----	---

Usando una tasa de aprendizaje de 0.3 en adelante, los modelos resultan ser bastante irregulares como se evidencia en la pérdida. La tasa de 0.1 a pesar de ser un poco irregular, se estabiliza a las 25 épocas, el mejor resultado.

#### Tamaño de los lotes:

Tamaño de los lotes	Épocas	Clasificación
5	44	<p>OUTPUT Test loss 0.020 Training loss 0.021</p>
10 (Estructura base)	37	<p>OUTPUT Test loss 0.045 Training loss 0.047</p>

15	82	<p>OUTPUT</p> <p>Test loss 0.042 Training loss 0.046</p>
20	96	<p>OUTPUT</p> <p>Test loss 0.054 Training loss 0.060</p>
25	142	<p>OUTPUT</p> <p>Test loss 0.076 Training loss 0.076</p>
30	449	<p>OUTPUT</p> <p>Test loss 0.028 Training loss 0.029</p>

A medida que aumentan los lotes, los resultados empeoran, probablemente por la poca cantidad de datos. En este caso nos mantenemos con lotes de 10 datos.

Cantidad de datos de entrenamiento:

Cantidad de datos de entrenamiento	Épocas	Clasificación
20%	236	<p>OUTPUT</p> <p>Test loss 0.047 Training loss 0.009</p>
40%	94	<p>OUTPUT</p> <p>Test loss 0.048 Training loss 0.031</p>
50% (Estructura base)	37	<p>OUTPUT</p> <p>Test loss 0.045 Training loss 0.047</p>
60%	66	<p>OUTPUT</p> <p>Test loss 0.029 Training loss 0.019</p>

80%	51	<p>OUTPUT Test loss 0.015 Training loss 0.022</p>
-----	----	---

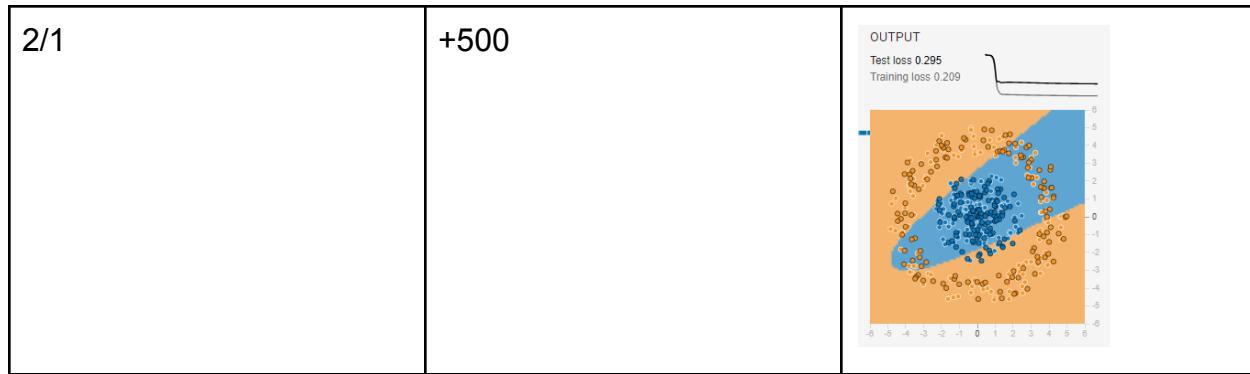
En este caso, es útil tener la misma cantidad de datos de entrenamiento y de test. Tener menos en algún grupo genera que los modelos sean irregulares y necesiten más épocas para clasificar correctamente.

### Capas y neuronas

Resultados con 2 capas

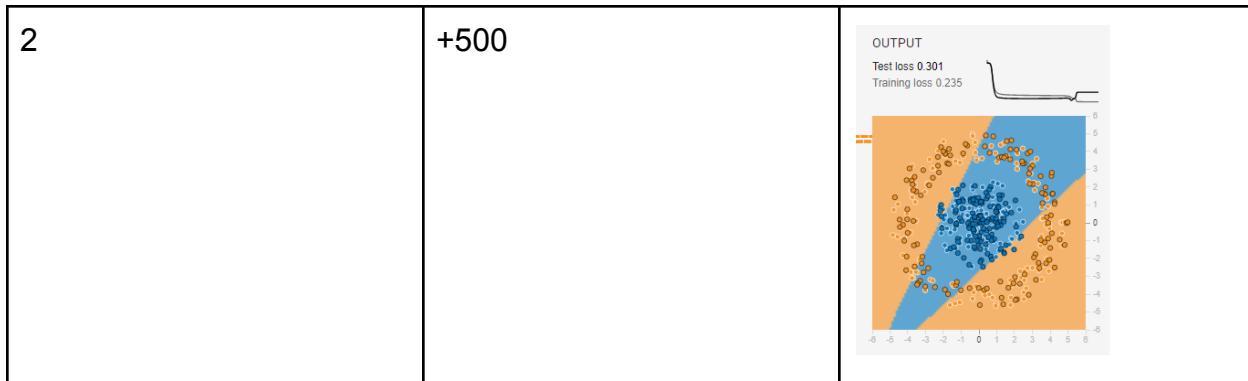
Neuronas por capa	Épocas	Clasificación
5/2	58	<p>Test loss 0.023 Training loss 0.025</p>
4/2 (Estructura base)	37	<p>OUTPUT Test loss 0.045 Training loss 0.047</p>

3/2	70	<p>OUTPUT Test loss 0.022 Training loss 0.022</p>
2/2	+500	<p>OUTPUT Test loss 0.290 Training loss 0.216</p>
4/1	58	<p>OUTPUT Test loss 0.039 Training loss 0.039</p>
3/1	70	<p>OUTPUT Test loss 0.031 Training loss 0.033</p>



Resultados con 1 capa

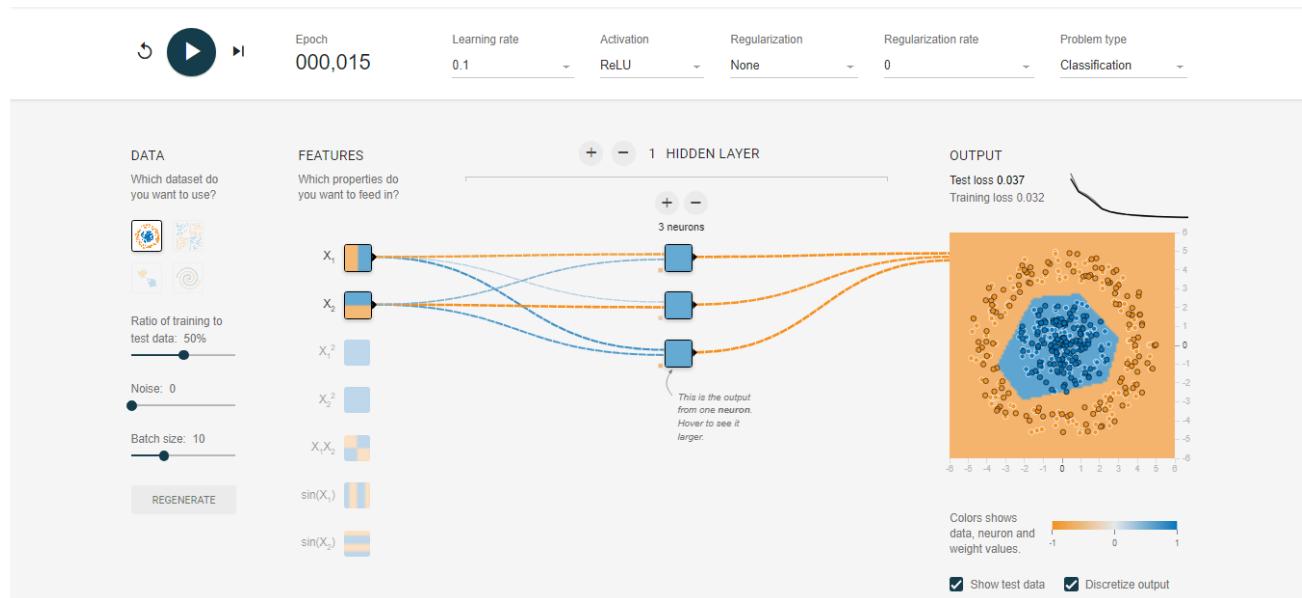
Neuronas	Épocas	Clasificación
5	145	<p>OUTPUT Test loss 0.027 Training loss 0.023</p>
4	108	<p>OUTPUT Test loss 0.045 Training loss 0.043</p>
3	58	<p>OUTPUT Test loss 0.091 Training loss 0.092</p>



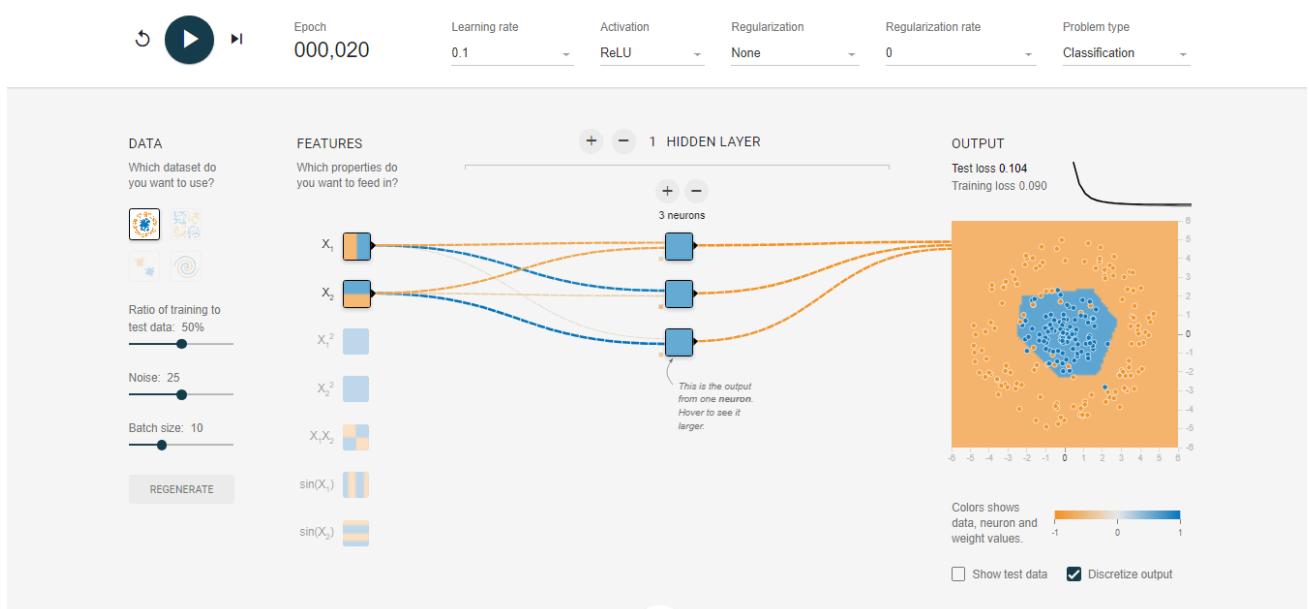
Al usar 2 capas, la estructura base (4/2) obtiene buenos resultados con 37 épocas, y utilizando 1 capa, usando 3 neuronas también tiene un buen desempeño, usando 58 épocas. En este proyecto, al estar buscando una arquitectura mínima, decidimos usar solo 1 capa mezclada con los demás hiper - parámetros.

#### **Mejor estructura y con la arquitectura mínima que encontramos:**

- **Capas ocultas y neuronas:** 1 capa y 3 neuronas
- **Datos de entrenamiento:** 50% del total de datos
- **Tamaño de lotes:** 10
- **Tasa de aprendizaje:** 0.1
- **Función de activación:** Relu
- **Épocas:** 15



Asimismo, usando 25 de ruido y usando 20 épocas también se obtienen buenos resultados en la clasificación. Para más ruido, como los datos están tan dispersos se necesitan más de 500 épocas, lo cual se sale de nuestros límites iniciales.

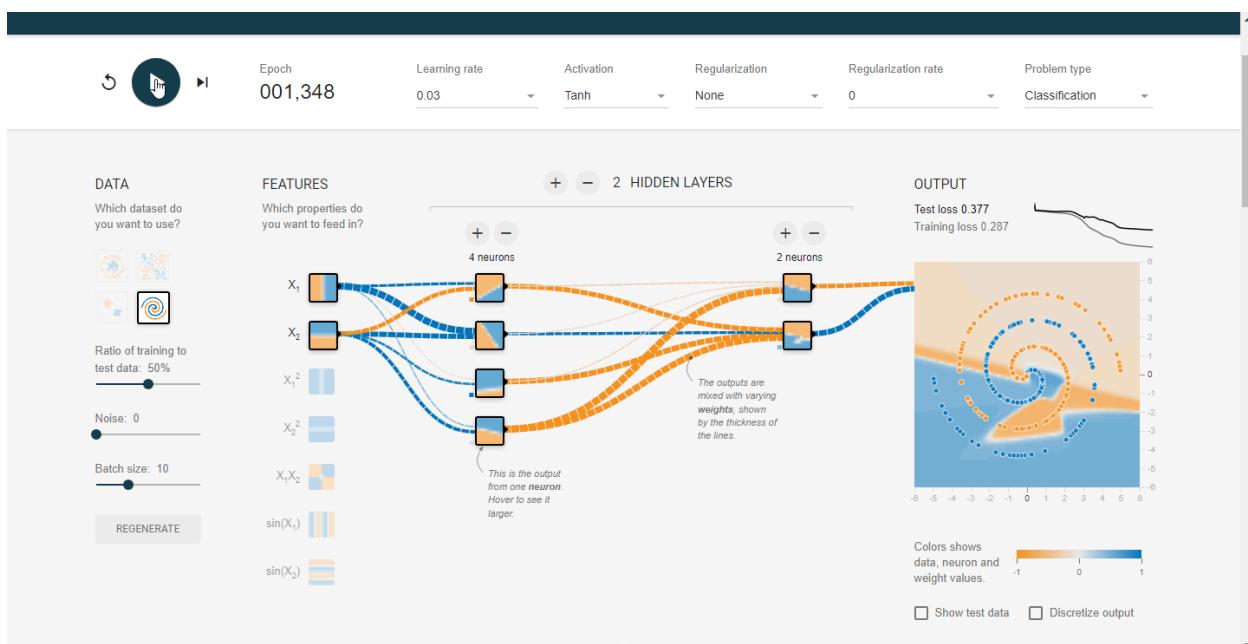


## Espiral

Estructura base (predeterminada en Tensorflow Playground):

- **Capas ocultas y neuronas:** 2 capas con estructura 4/2
- **Datos de entrenamiento:** 50% del total de datos
- **Tamaño de lotes:** 10
- **Tasa de aprendizaje:** 0.03
- **Función de activación:** Tanh
- **Épocas:** Con esta estructura base no logra clasificar bien la espiral (+1000)

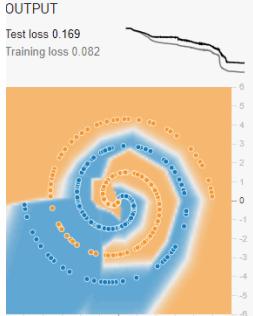
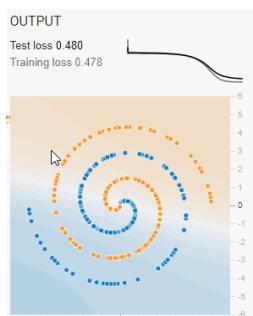
Resultados con la estructura base:

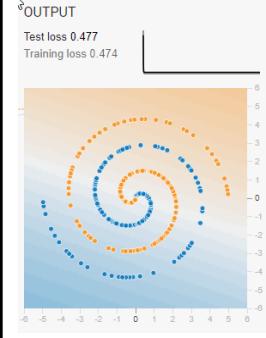


### **Modificando uno por uno los hiper parámetros:**

En el caso de la espiral las estructuras pequeñas demoran más en clasificar los datos, entonces si se llega a 800 épocas, se entiende que el modelo no pudo clasificar correctamente la información a pesar de la gran cantidad de épocas. También hacemos más grande la estructura base (3 capas y neuronas 6/5/2), ya que para estos datos es necesario por su complejidad. Asimismo, en este caso al escoger un ajuste para un hiper parámetro, lo utilizábamos para realizar los siguientes experimentos y así tener una mayor facilidad encontrando el ajuste correcto.

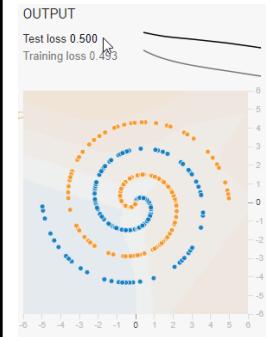
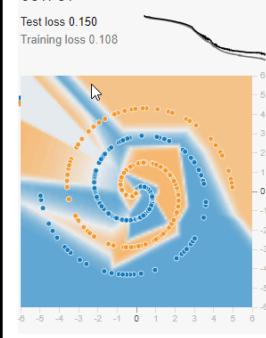
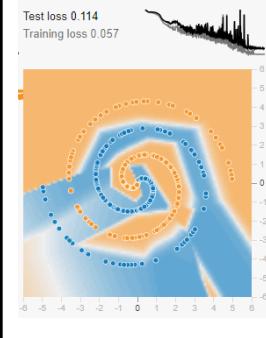
#### Función de activación:

Función de activación	Épocas	¿Cómo lo clasifica?
Tanh (Estructura base)	+800	<p>OUTPUT</p> <p>Test loss 0.419 Training loss 0.400</p> 
Relu	+800	<p>OUTPUT</p> <p>Test loss 0.169 Training loss 0.082</p> 
Sigmoide	+800	<p>OUTPUT</p> <p>Test loss 0.480 Training loss 0.478</p> 

Linear	+800	
--------	------	---

Ninguno clasifica correctamente los datos, pero podemos ver indicios de que función de activación es mejor, para los siguientes cambios nos quedaremos con la función Relu.

Tasa de aprendizaje:

Tasa de aprendizaje	Épocas	Clasificación
0.0001	+800	
0.01	+800	
0.03 (Estructura base)	+800	

0.1	+800	<p>OUTPUT</p> <p>Test loss 0.252 Training loss 0.193</p>
0.3	+800	<p>OUTPUT</p> <p>Test loss 0.486 Training loss 0.430</p>

Para estas modificaciones, usando el ratio de aprendizaje 0.03, el modelo reconoce mejor las curvas y no es tan irregular, por lo que es el que se usará en el modelo final.

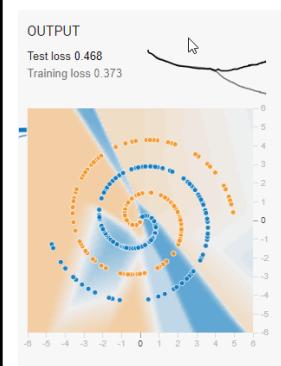
#### Tamaño de los lotes:

Tamaño de los lotes	Épocas	Clasificación
5	+800	<p>OUTPUT</p> <p>Test loss 0.138 Training loss 0.160</p>

10 (Estructura base)	+800	<p><b>OUTPUT</b></p> <p>Test loss 0.250 Training loss 0.179</p>
15	+800	<p><b>OUTPUT</b></p> <p>Test loss 0.232 Training loss 0.156</p>
20	+800	<p><b>OUTPUT</b></p> <p>Test loss 0.189 Training loss 0.124</p>
25	+800	<p><b>OUTPUT</b></p> <p>Test loss 0.276 Training loss 0.208</p>

30

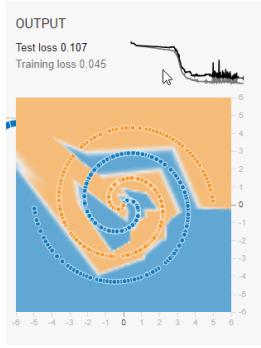
449



En este caso para la cantidad de datos que hay, lo más equilibrado y para que no caiga en un sesgo con datos muy atípicos, consideramos que el tamaño de lote 15 es el que mejor funciona.

Cantidad de datos de entrenamiento:

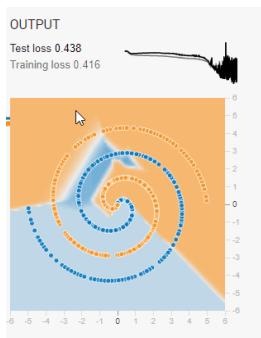
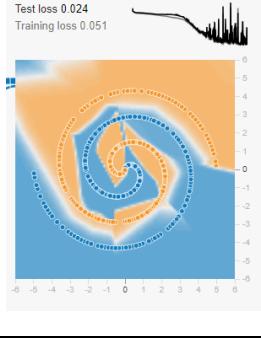
Cantidad de datos de entrenamiento	Épocas	Clasificación
50% (Estructura base)	+800	
70%	+800	

80%	+800	
-----	------	---

Todos los modelos son irregulares, pero teniendo una mayor cantidad de datos de entrenamiento el modelo empieza a clasificar mejor, así que para los siguientes modelos escogere arbitrariamente entre 70% y 80% de los datos para entrenar, en mi opinión es un número equilibrado para este hiper parámetro.

### Capas y neuronas

Resultados con 3 capas

Neuronas por capa	Épocas	Clasificación
6/5/2 (Estructura base)	+800	
7/6/2	624	

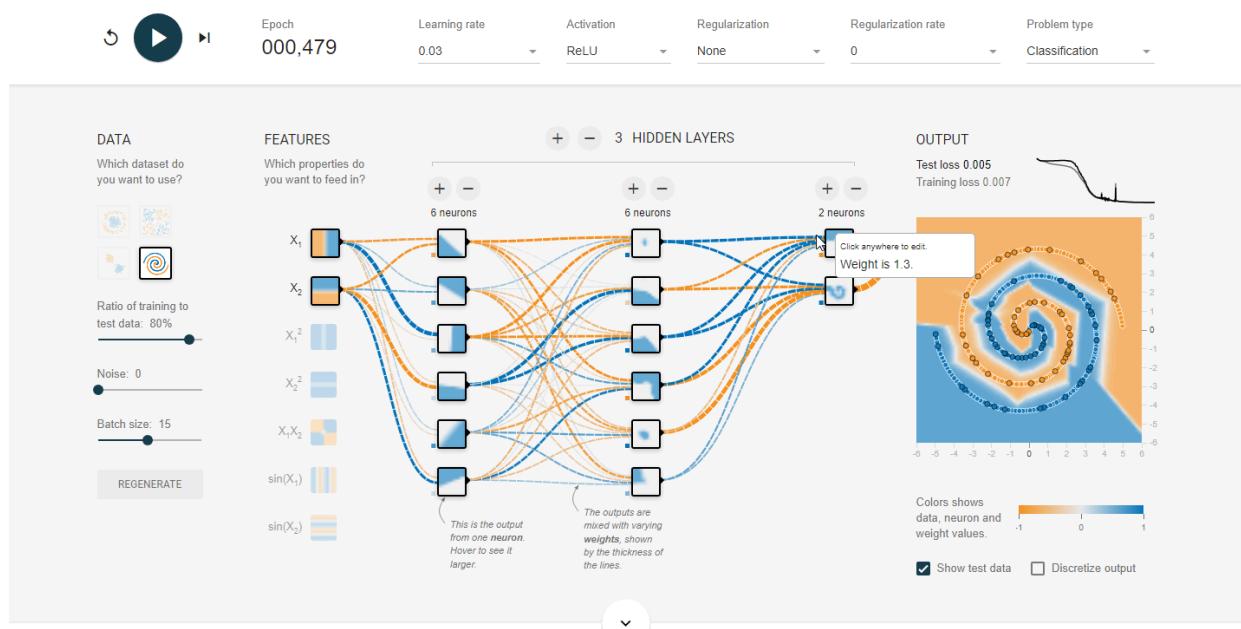
8/7/2	400	<p>OUTPUT Test loss 0.023 Training loss 0.015</p>
7/7/3	688	<p>OUTPUT Test loss 0.030 Training loss 0.008</p>
5/3/3	+800	<p>OUTPUT Test loss 0.258 Training loss 0.300</p>
3/2/2	+800	<p>OUTPUT Test loss 0.478 Training loss 0.467</p>

Para este caso escogimos una estructura de 3 capas y 6/6/2 neuronas, ya que después de revisar diferentes variaciones cambiando de a pocas neuronas, este modelo es el que ha tenido un aprendizaje suave (sin muchas irregularidades), sin tantas épocas y un error en los 2 conjuntos de datos (80%entrenamiento y 20% test) muy bajo. Por otro lado, estructuras de 2 y 1 capa, no encontramos que sirvieran para este conjunto de datos.

## Mejor estructura y con la arquitectura mínima que encontramos:

- **Capas ocultas y neuronas:** 3 capas y 6/6/2 neuronas
- **Datos de entrenamiento:** 80% del total de datos
- **Tamaño de lotes:** 15
- **Tasa de aprendizaje:** 0.03
- **Función de activación:** Relu
- **Épocas:** 479

Clasificación de los datos. Se puede apreciar incluso en la gráfica de error, que esta estructura genera una clasificación muy suave, o sea en su proceso no hay una gran variación.



Después de varias pruebas, esta estructura incluso clasifica bien con un 20 en el apartado de ruido con 814 épocas, para más ruido se necesitan alrededor de más de 1000 épocas, lo que ya está fuera de nuestras configuraciones iniciales.

