

# Reporte – Proyecto Mnist

Semana: X



**Nombre de los estudiantes:**

- Ian Román Beltrand Padilla

**Número de cuenta:**

- 22141085

**Sede de estudio:**

UNITEC SPS

**Docente:**

Lic. Iván Deras

**Sección:**

Redes Neuronales

**Fecha de entrega:**

Marzo 27, 2025

# ÍNDICE

<u>ÍNDICE</u>	<u>2</u>
<u>INTRODUCCIÓN</u>	<u>3</u>
<u>METODOLOGÍA</u>	<u>4</u>
<u>RESULTADOS</u>	<u>6</u>
<b>RESULTADOS: TRAINING</b>	<b>6</b>
<b>BATCHES</b>	<b>6</b>
<b>EPOCHS</b>	<b>10</b>
<b>RESULTADOS: TESTING</b>	<b>13</b>
<u>CHALLENGES</u>	<u>14</u>
<u>CONCLUSIÓN</u>	<u>15</u>
<u>ANEXOS</u>	<u>16</u>
<b>FASE DE ENTRENAMIENTO: ÉPOCAS</b>	<b>16</b>
<b>FASE DE ENTRENAMIENTO: BATCHES</b>	<b>21</b>

# INTRODUCCIÓN

La finalidad de este proyecto e informe es visualizar el desarrollo de diferentes redes neuronales o modelos que permitan discernir entre diferentes números escritos a mano. Se harán múltiples experimentos cambiando diferentes aspectos, añadiendo cada vez más, distintas capas de complejidad al modelo en búsqueda del mejor.

Se comenzará describiendo la estructura general del proyecto en el apartado de Metodología, repasando los distintos archivos y partes que lo conforman y entendiendo el porqué de cada uno de ellos. Pasando después a los resultados, los cuales buscan ser lo más breves posibles dentro de lo que cabe, e ir analizándolos en búsqueda de hallazgos interesantes. Posteriormente, se describirán los retos enfrentados a lo largo del proyecto y por último se brindarán diferentes conclusiones en forma de aprendizaje de lo visto en este reporte. Al final, se encontrarán los anexos, en donde se podrán visualizar de forma gráfica y específica los distintos intentos hechos para estos experimentos.

La sencillez del proyecto no demerita los intentos de profundizar en el análisis del mismo, los aprendizajes de algo básico serán las bases de proyectos más complejos.

# METODOLOGÍA

El proyecto cuenta con múltiples archivos de extensión “.py”, la idea fue modularizar los componentes lo más posible. Dentro de los distintos archivos tenemos a:

- **ActivationFunc.py:** Archivo que contiene todas las clases de funciones de activación con las que se busca contar en el entrenamiento y prueba de la red neuronal.
- **DenseLayer.py:** Archivo que contiene la clase principal que le da estructura a las capas de la red neuronal.
- **LossFunc.py:** Archivo que contiene todas las clases de funciones de pérdida con las que se busca contar en el entrenamiento de la red neuronal.
- **Optimizers.py:** Archivo que contiene todas las clases necesarias para hacer uso de diferentes optimizadores a la hora de entrenar a la red neuronal.
- **PlotFuncs.py:** Archivo que contiene todas las funciones que permiten las diferentes visualizaciones que se utilizarán posteriormente en el proyecto y muestra de resultados.
- **MnistDataset.py & Mnist.Loader.py:** Archivos que contienen la clase para el manejo de los datos de entrenamiento y prueba e indicaciones para su uso, respectivamente.

Todos estos archivos fueron planteados para su uso en diferentes experimentos que se encuentran en distintos archivos. Esta estructura permite la alteración individual de los archivos sin perjudicar los demás. Los experimentos realizados se encuentran en los siguientes archivos:

- **Basic\_NN.py:** Este experimento no hace uso de optimizadores ni de reguladores y cuenta con una sola capa oculta.
- **TwoL\_NN.py:** Este experimento no hace uso de optimizadores ni de reguladores y cuenta con dos capas ocultas, dándole más complejidad al modelo.
- **Adam\_NN.py:** Este experimento hace uso de optimizadores, más no de reguladores y cuenta con una sola capa oculta.
- **Adam\_L2\_NN.py:** Este experimento hace uso de optimizadores y de reguladores, sin embargo, hace uso de una sola capa oculta.
- **TwoLayerA+L2.py:** Este experimento hace uso de optimizadores y de reguladores, también hace uso de dos capas ocultas, siendo este por ende el más complejo de todos.

Cada experimento busca probar el rendimiento y forma de aprendizaje de diferentes modelos para un mismo objetivo. Al final de cada uno de ellos se guardan los pesos y sesgos de las diferentes capas en distintos archivos con extensión “.pkl” para su posterior carga en el archivo de testing (*Testing.py*). Este último archivo recoge todos los .pkl de todos los

experimentos y los carga en nuevas capas para usarlas en el dataset de testing y comprobar su accuracy final. Comparando el desempeño de cada modelo.

Por lo que, esencialmente hay tres bloques de archivos que interactúan entre sí. Los archivos de apoyo, que son los que fueron mencionados inicialmente, en los que se encuentran los códigos para ser llamados en el segundo bloque de archivos que sería el de experimentos, en los cuales se arman y se entrena las redes neuronales que al final guardarán dichos pesos y sesgos ya entrenados para hacer uso de ellos en la tercera fase, que sería la de testing.

Esta estructura permite seccionar bien incluso la sección de resultados, en la que se busca comparar desempeño en fase de entrenamiento y de testing.

# RESULTADOS

Los resultados se dividen en dos distintas categorías, los de entrenamiento y los de testing.

## Resultados: Training

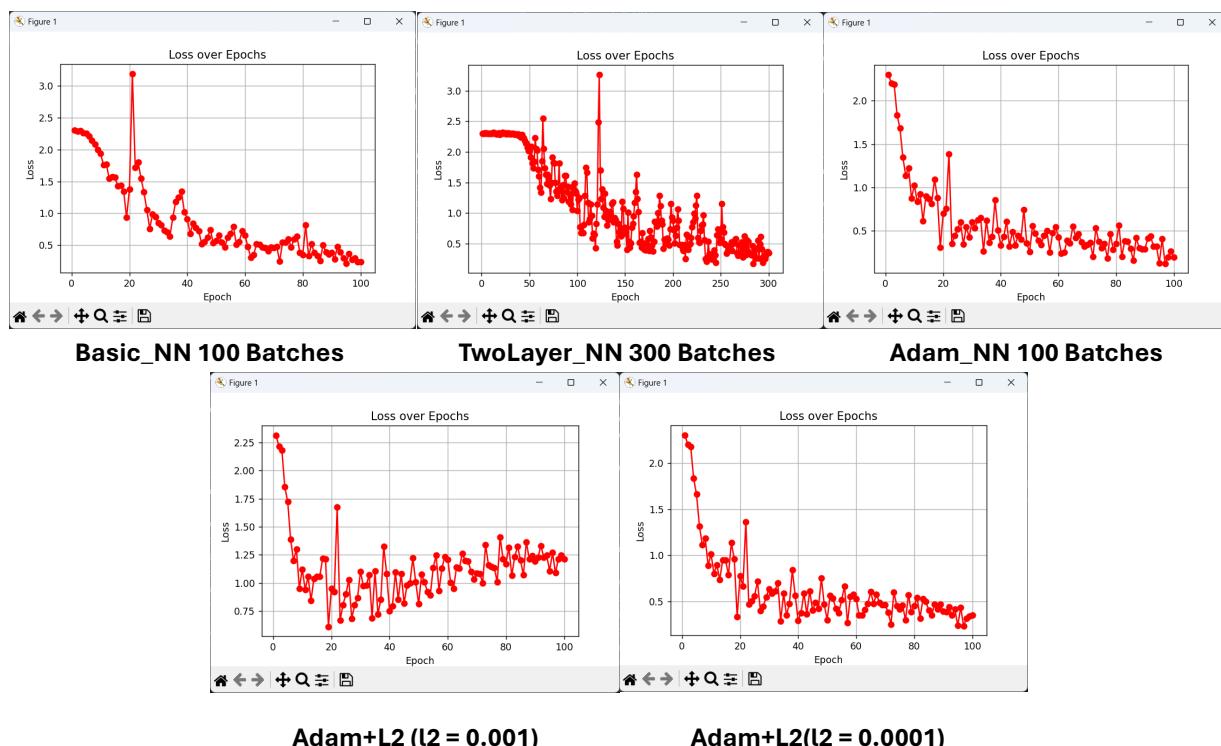
Dentro los resultados de Training se distinguen dos en mí metodología: Por Batches y por Épocas. En este apartado los registros del loss y accuracy se hacían con respecto a la cantidad de batches pasados más que épocas transcurridas, esto debido a que de esta forma se permite visualizar de una mejor forma el entrenamiento y su desarrollo.

Se tendrán en cuenta 4 disntintos aspectos relacionados al loss y al accuracy. El Loss vrs Épocas, Accuracy vrs Épocas, y dos de Loss vrs Accuracy en búsqueda de correlación.

Cada experimento cuenta con diferentes intentos solo para comprobar que realmente existía un patrón en cada uno de ellos, acá se comparará aspecto por aspecto y de sólo uno de los intentos, sin embargo, en búsqueda de más información y ordenada por experimento estará disponible la sección de anexos con cada uno de ellos.

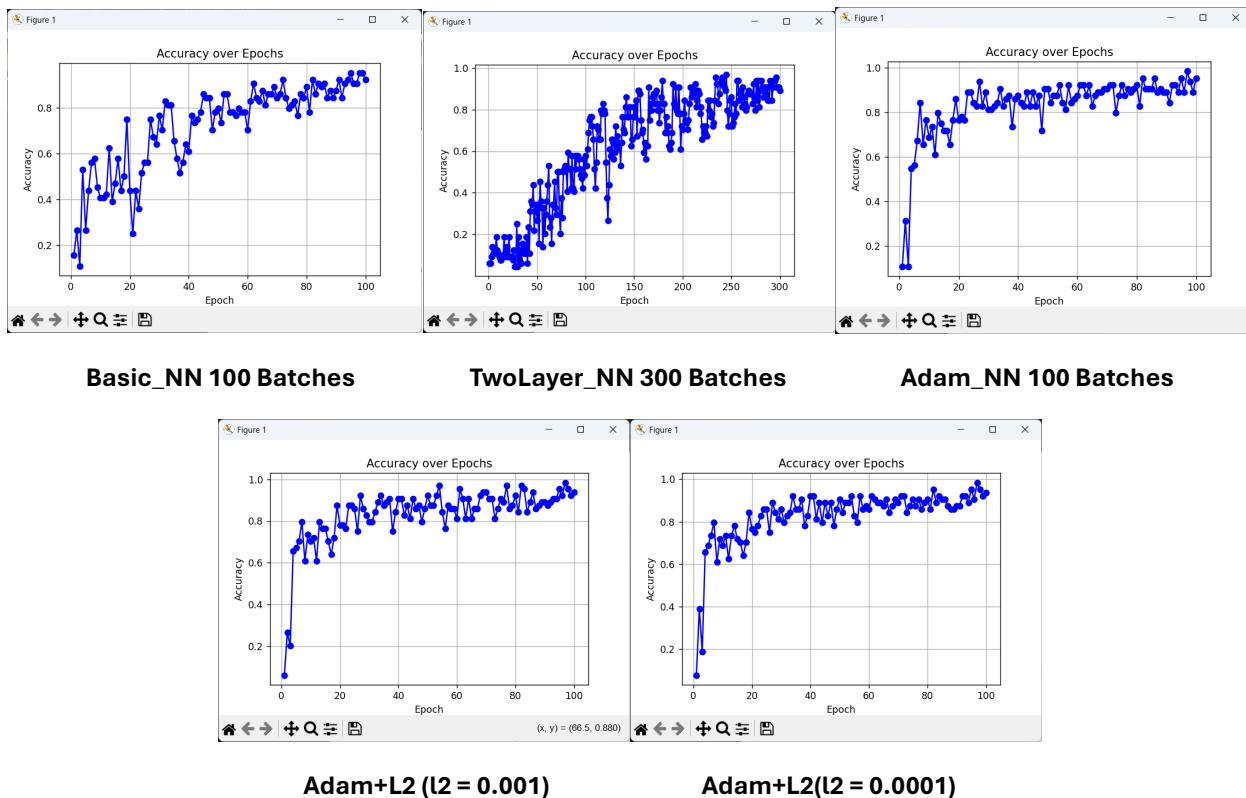
### Batches

#### Loss over Epochs



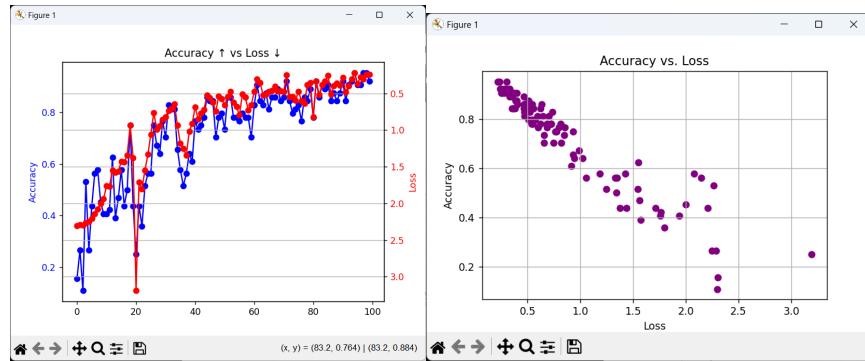
En este apartado lo que se puede observar es la comparación de cómo lo hizo cada experimento en la sección de entrenamiento. Lo primero a destacar son los 300 batches del TwoLayer\_NN y son debido a que cómo se puede apreciar, no comienza a decaer significativamente el los hasta que llega al N.50 y necesito de casi 300 para llegar al mismo nivel que los demás. Lo otro a destacar es que cuando se tuvo una lambda grande en el regulador el los empezaba a subir en un punto, por lo que se probó con uno más pequeño, resultando así más eficaz. Otro detalle que destacar es que una vez se implementó Adam, el loss se movía menos erráticamente y era más smooth. Lo que le da puntos por sobre de los anteriores modelos aunque estos hayan llegado a su mismo objetivo.

### Accuracy over Epochs

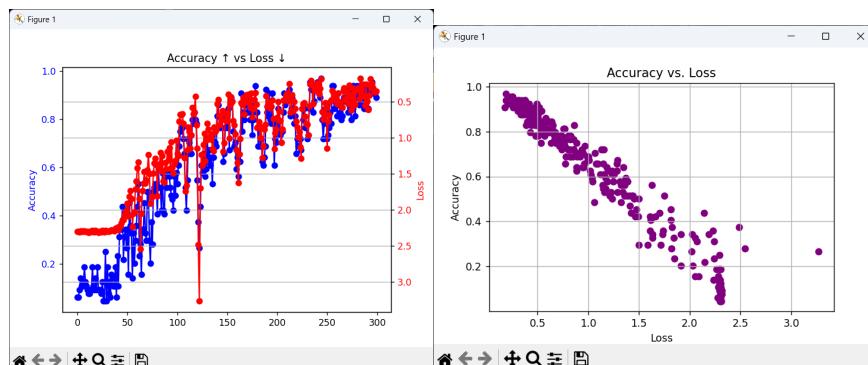


Las relaciones mencionadas anteriormente aplican aquí también, dos cosas son relevantes a destacar en esta ocasión, se ve la relevancia del optimizador cuando nos percatamos que llega a niveles de accuracy altos en la mitad de esfuerzo que los demás. También ahora sí notamos un impacto más determinante por parte de los reguladores, ya que estos permiten que el accuracy crezca de una forma menos errática.

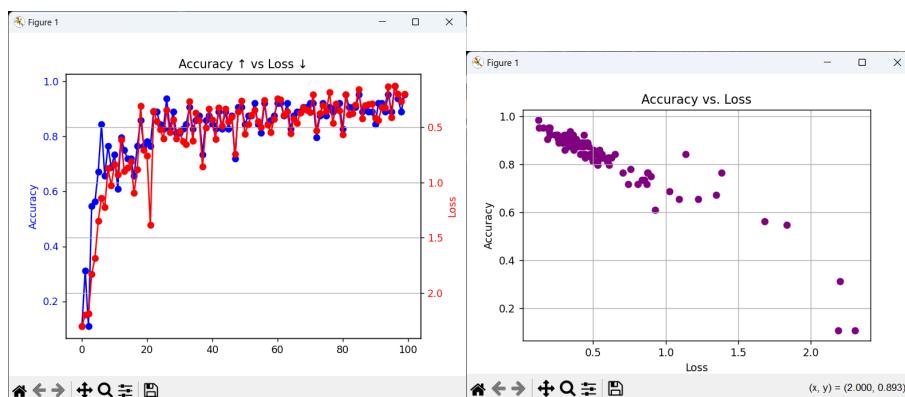
## Accuracy vs Loss



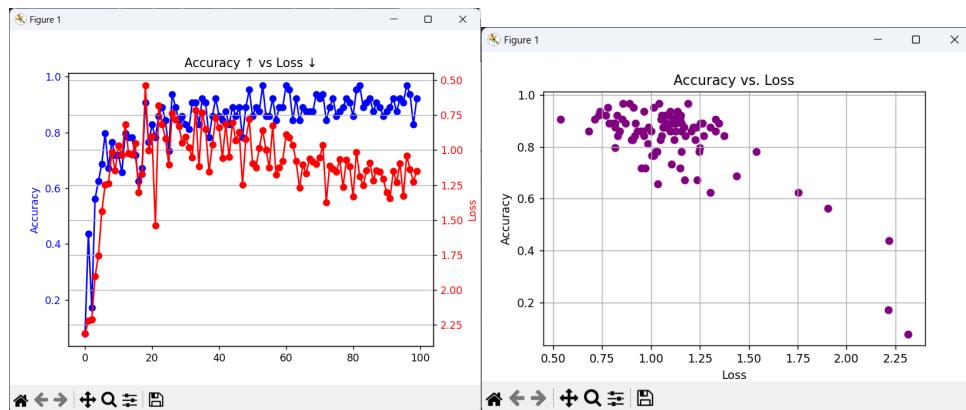
Basic\_NN - Correlation: -0.9387



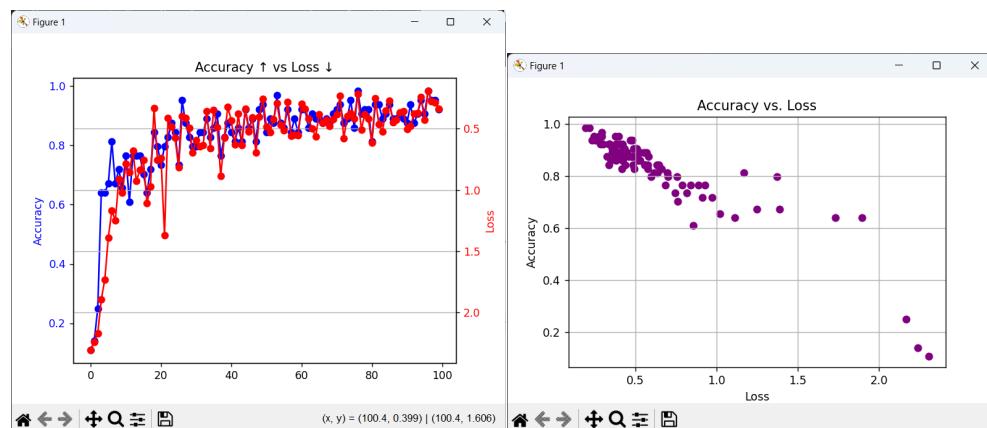
TwoLayer\_NN - Correlation: -0.9730



Adam\_NN - Correlation: -0.9405



Adam+L2 l2 = 0.001 - Correlation: -0.7437



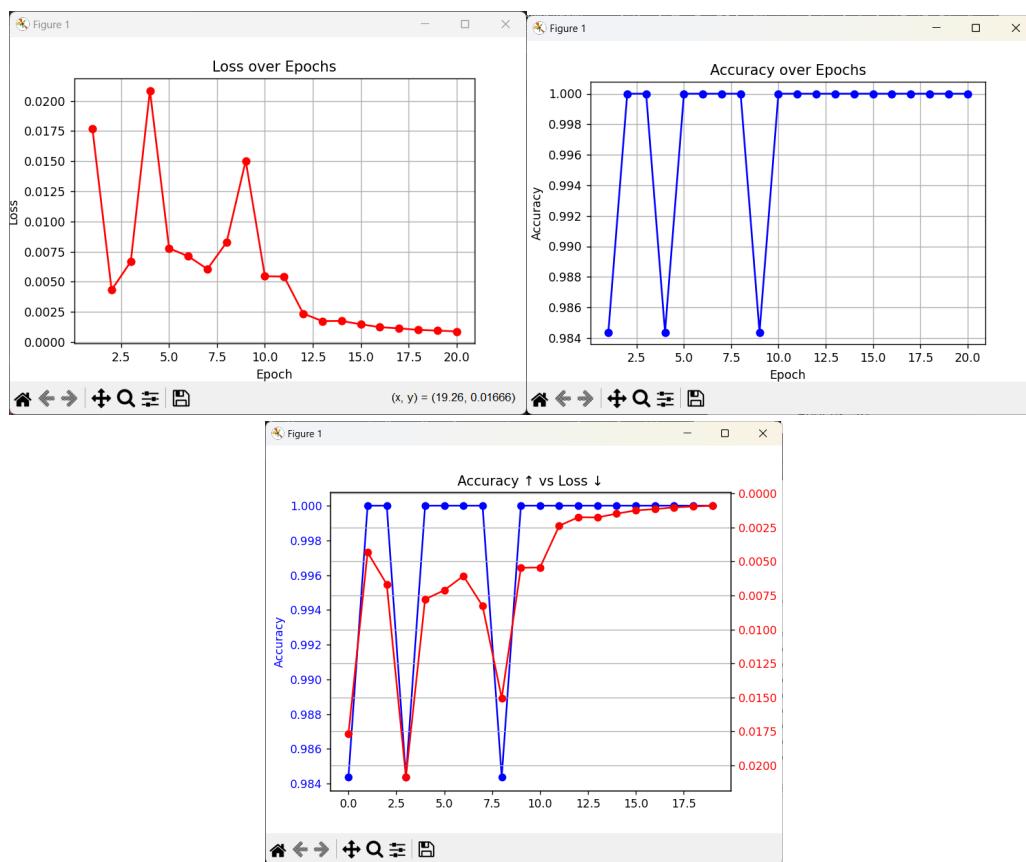
Adam+L2 l2 = 0.0001 Correlation: -0.9182

La pregunta por responder en este apartado es “¿Por qué es importante la correlación?”, primero que todo, la correlación entre dos fenómenos nos indica que tan relacionados están estos dos, en el caso del loss y la accuracy, su correlación para inexistente porque es sabido que mientras uno tiene que bajar el otro debería de subir, más, sin embargo, eso entonces nos indica una existente correlación, solamente que esta sería negativa y es precisamente lo que se buscaba. El primer chart tiene el eje del loss invertido, esto permite visualizar de una mejor forma lo relacionado que esta el accuracy con este indicador, el segundo chart nos muestra un scatter plot, cuya dirección nos afirma una correlación negativa y entre menos dispersos estén entre sí los puntos es una correlación más fuerte. Se puede observar, por el cálculo final de la correlación, que el único modelo cuya correlación no es tan alta es el de el regulador con una lambda más baja.

## Epochs

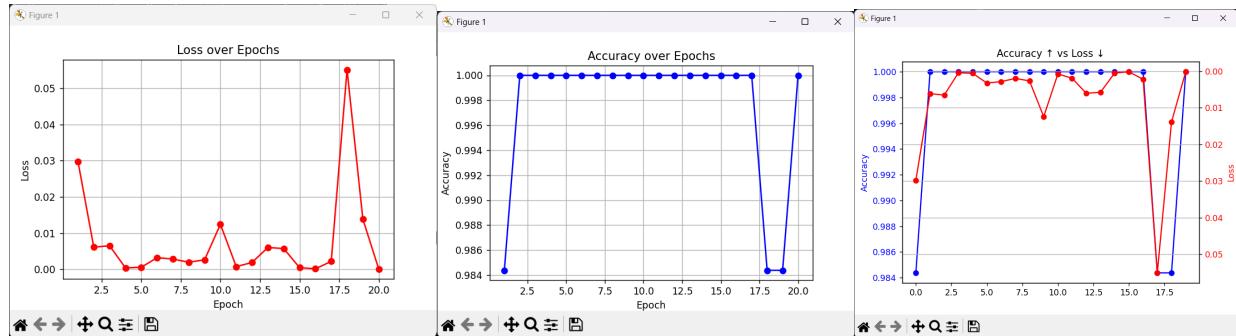
El anterior apartado de entrenamiento por batches nos mostraba un análisis más profundo del desempeño del modelo a la hora de estar siendo entrenado, ahora, en este apartado por épocas, se notarán saltos más altos ya que cada época conlleva haber traspasado el dataset entero. Ya que ya se realizó un análisis más profundo de cada modelo, en este apartado se presentarán individualmente y se añadirá uno más.

Data Registry: Basic Neural Network					
Batches size	64	Hidden Layers	1	Optimizer	NO
Epochs	20	Number of Neurons	128	Regulation	NO



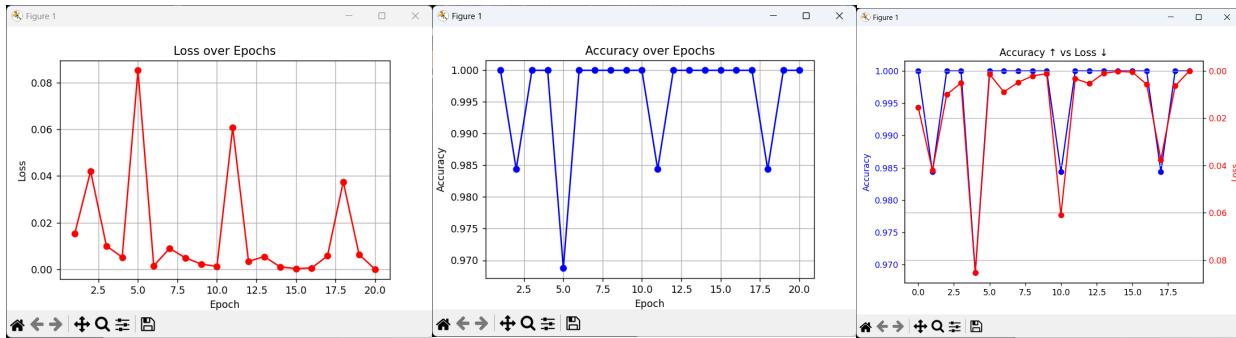
Correlation: -0.8877

Data Registry: Two Layer Neural Network					
Batches size	64	Hidden Layers	2	Optimizer	NO
Epochs	20	Number of Neurons	128	Regulation	NO



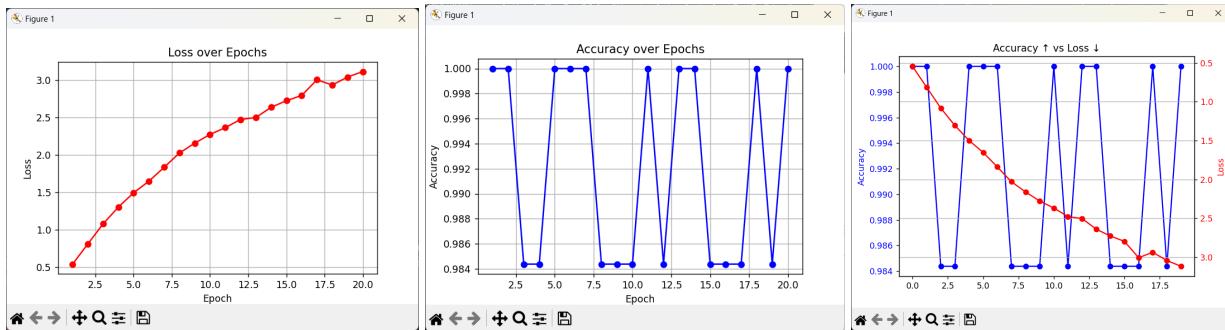
Correlation: -0.8278

Data Registry: Adam Neural Network					
Batches size	64	Hidden Layers	1	Optimizer	YES
Epochs	20	Number of Neurons	128	Regulation	NO



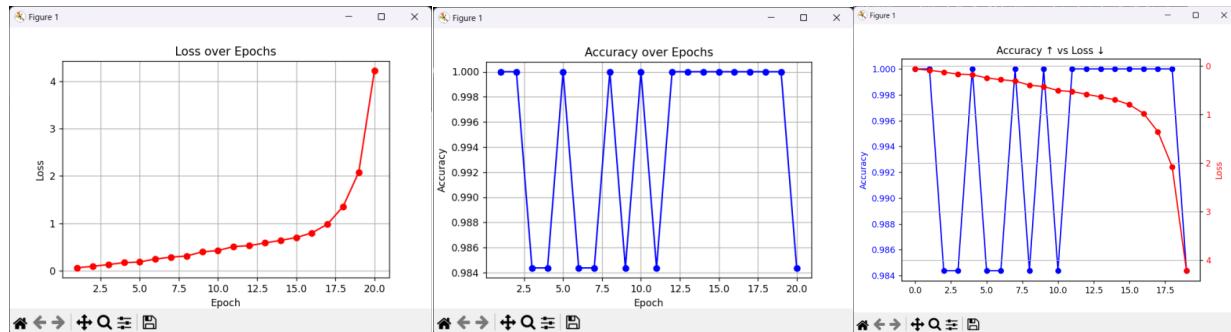
Correlation: -0.9715

Data Registry: Adam+L2 Neural Network					
Batches size	64	Hidden Layers	1	Optimizer	YES
Epochs	20	Number of Neurons	128	Regulation	YES



Correlation: -0.1987

Data Registry: Two Layer Adam+L2 Neural Network					
Batches size	64	Hidden Layers	2	Optimizer	YES
Epochs	20	Number of Neurons	128	Regulation	YES



Correlation: -0.0918

La naturaleza prevista gracias a lo visto en el análisis por batches prevalece, sin embargo, notamos que cuando se trata de épocas, en algún punto los reguladores comienzan a reportar cada vez más pérdida que sigue incrementando a medida ocurren las épocas, cosa que realmente no altera el accuracy, pero sí difumina la correlación que se apreciaba antes.

## Resultados: Testing

Una vez habiendo entrenado los diferentes modelos se pusieron a prueba con el dataset de prueba que contiene exactamente 10000 elementos. En esta fase lo único que se visualiza es el accuracy final, ya que el loss es un aspecto meramente del entrenamiento que ayuda a ajustar los pesos y sesgos (lo que sería el backward propagation), pero esa etapa no existe aquí, solamente el forward.

La competencia final resultó así:

```
Test Adam Accuracy: 96.43%
Test L2 Accuracy: 96.60%
Test Basic Accuracy: 97.57%
Test Two_Layer Adam + L2 Accuracy: 86.75%
```

A pesar de lo visto durante el training donde se podía visualizar que los optimizadores y reguladores mejoraban el desempeño de los modelos, al final, la red neuronal más básica resultó ser la más acertada en cada uno de los intentos, seguida de la que implementa tanto reguladores como optimizadores, siendo esta mejor que la que solo cuenta con optimizadores, lo que tiene sentido ya que los reguladores evitan el overfitting. Por último, podemos observar que a la hora de querer implementar una segunda layer el accuracy disminuye, lo que también va acorde a lo visto durante la fase de entrenamiento, ya que entrenar dos layers complica dicha fase y con el mismo nivel de entrenamiento no alcanza la certeza de los demás experimentos.

## CHALLENGES

El reto y bloqueo más grande fueron los reguladores, a diferencia de Adam, los reguladores son más difíciles de percibir en los resultados, y más complicados de integrar en el código, son las únicas funciones que tuvieron que ser integradas en la estructura original del DenseLayer, ya que crear una class y archivo dedicado a ellos complicaba el proyecto y su entendimiento. Posteriormente, también resultaron ser un problema en las visualizaciones ya que no contaban con el rendimiento ni comportamiento esperado, la solución en este caso fue disminuir la lambda.

# CONCLUSIÓN

Gracias a este proyecto pude sacar múltiples aprendizajes puntuales que serán expuestos a continuación:

- Comportamientos presentados en la fase de entrenamiento no siempre se ven representados en la fase de testeo. En este caso, por más que los optimizadores y reguladores parecían hacer un excelente trabajo en la parte de entrenamiento, la red neuronal más básica mostró mejores resultados al final.
- A la hora de tratar con tareas relativamente sencillas, el hecho de engrandecer el modelo no lo vuelve mejor, en este caso, agregar nuevas capaz volvía más difícil el proceso de entrenamiento sin ningún beneficio a la hora del testeo.
- La prueba y el error son relevantes. De haberme quedado con los primeros intentos el reporte sería considerablemente más pequeño, sin embargo, no habría notado que cada modelo sigue patrones característicos, ni que se podía mejorar el loss cambiando la lambda por un valor más pequeño.
- Resultó verdaderamente útil visualizar el proceso de entrenamiento en batches ya que permitía visualizar mejor la naturaleza de cada modelo. Cómo entrenamiento, no es necesario, pero ya que se busca aprender y representar cada modelo considerarlo fue un acierto.

# ANEXOS

## Fase de Entrenamiento: Épocas

### Data Registry: Basic Neural Network

Batches size: 64

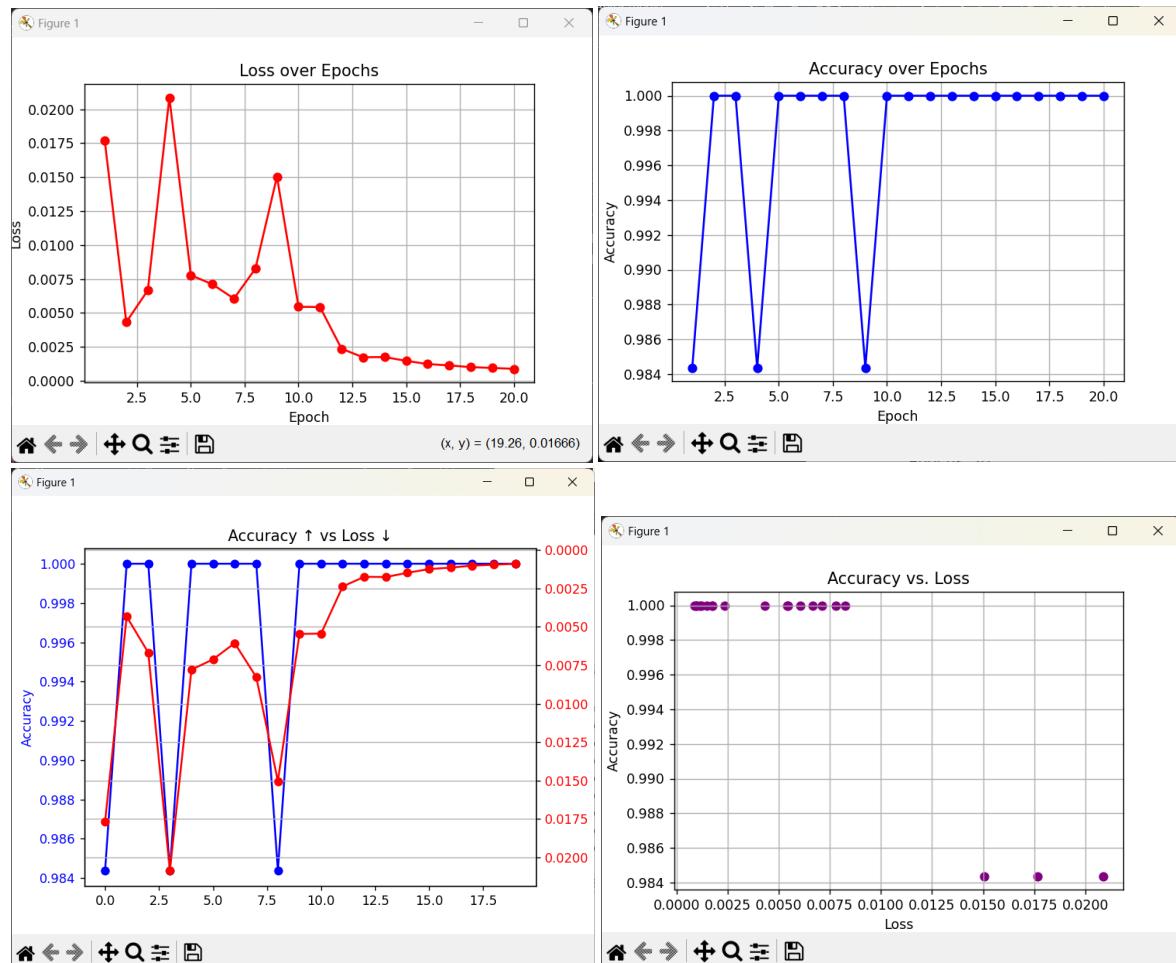
Epochs: 20

Hidden Layers: 1

Number of Neurons: 128

Optimizer: False

Regulation: False



**Correlation: -0.8877**

## Data Registry: Two Layer Neural Network

Batches size: 64

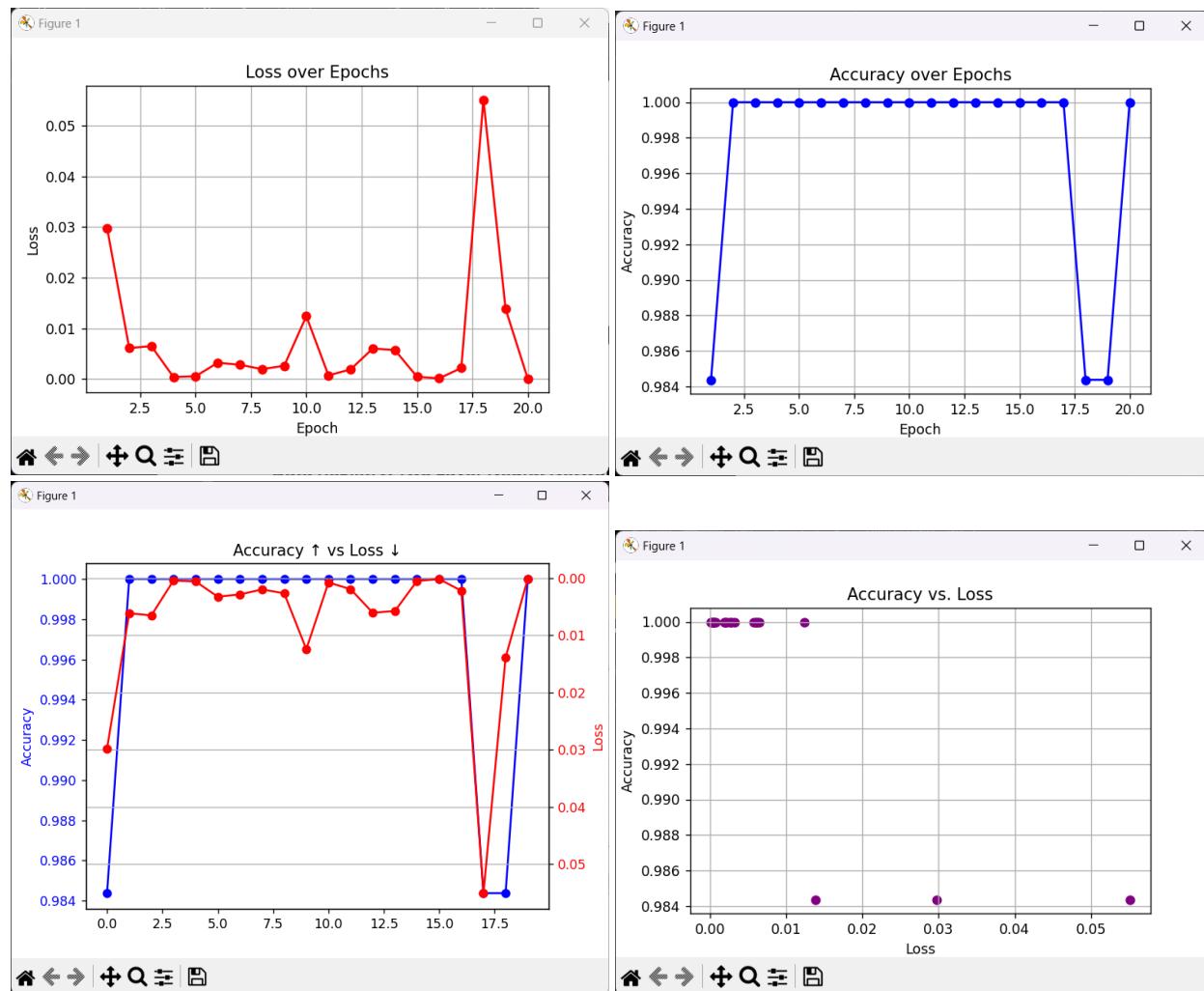
Epochs: 20

Hidden Layers: 2

Number of Neurons: 128

Optimizer: False

Regulation: False



**Correlation: -0.8278**

## Data Registry: Adam Neural Network

Batches size: 64

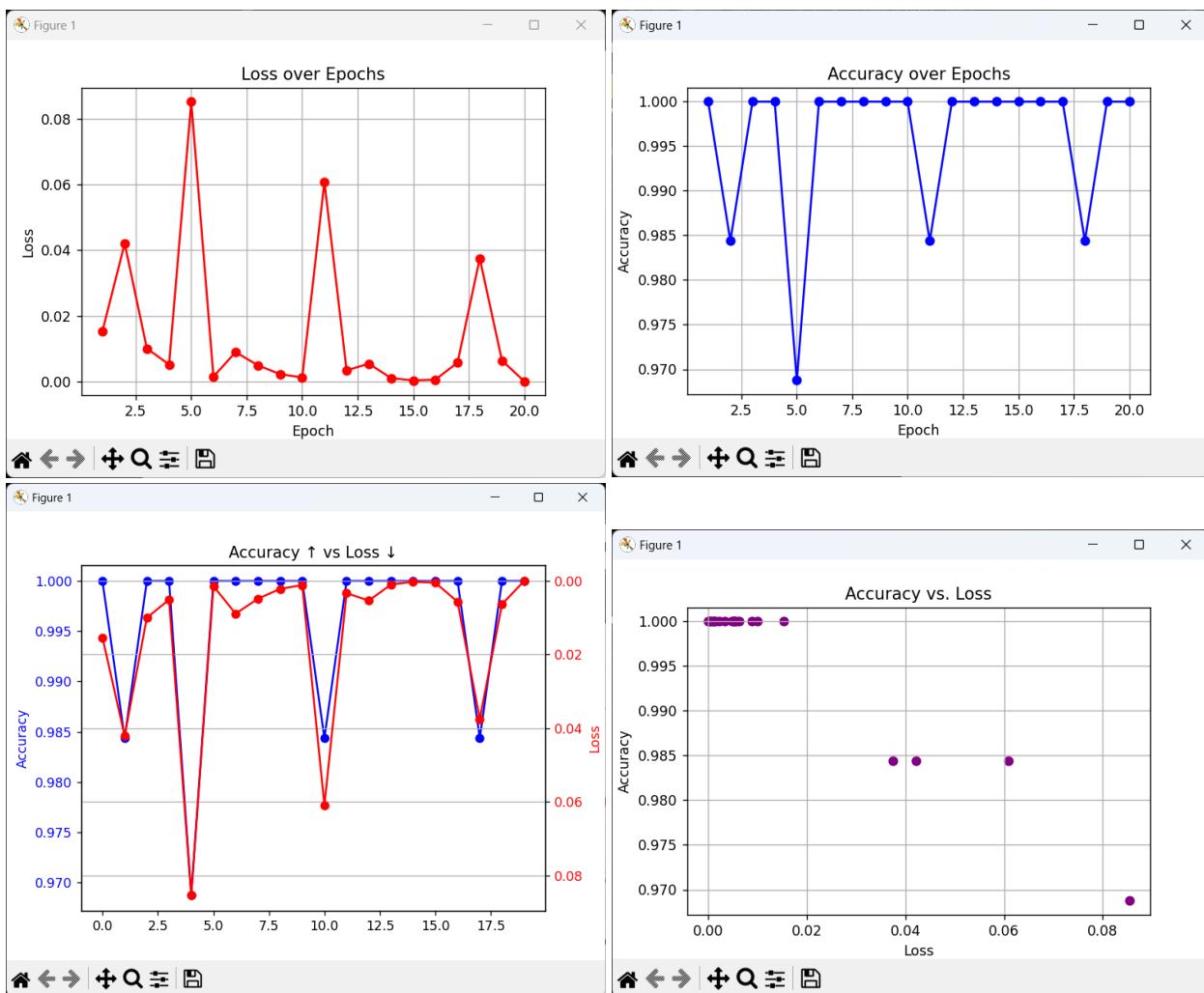
Epochs: 20

Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

Regulation: False



**Correlation: -0.9715**

## Data Registry: Adam + L2 Neural Network

Batches size: 64

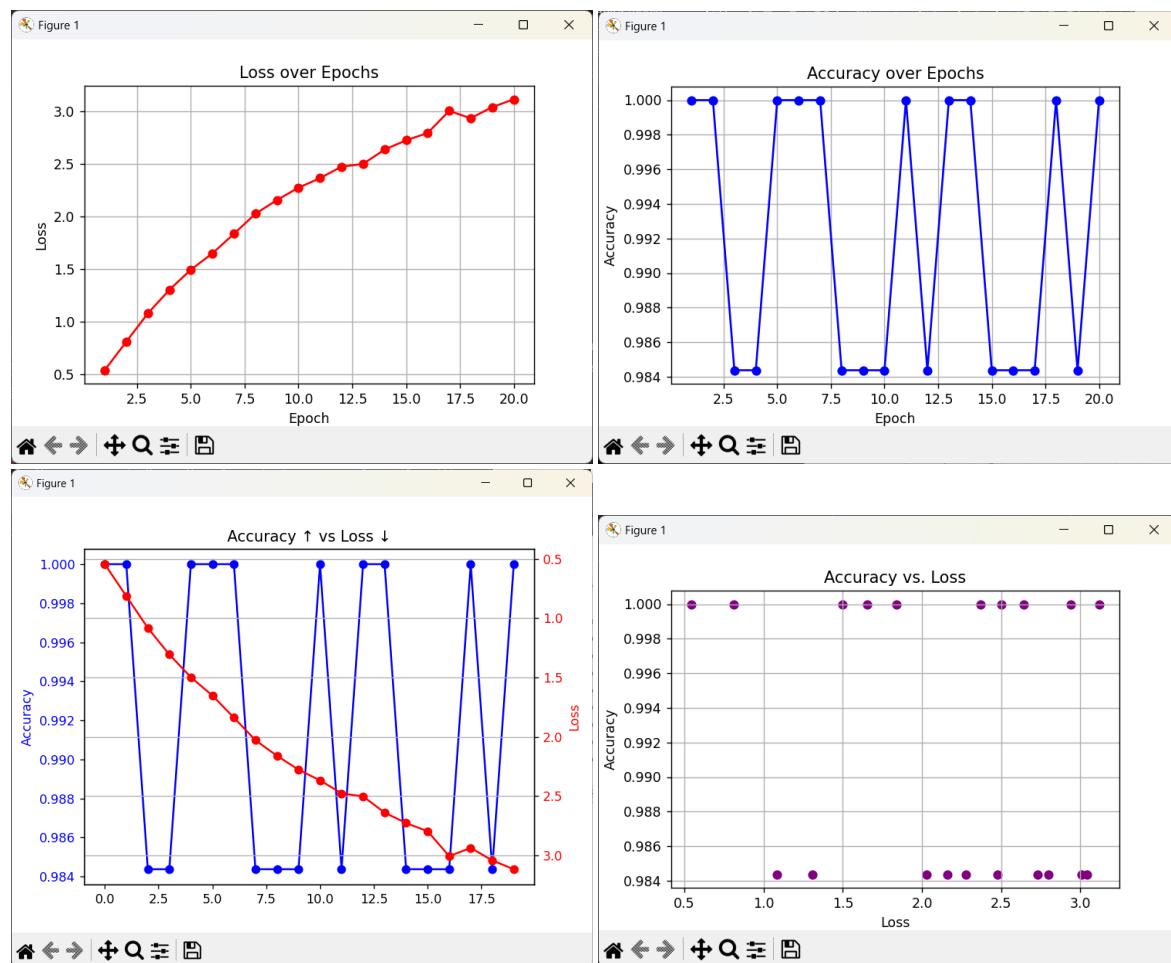
Epochs: 20

Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

Regulation: True



Correlation: -0.1987

## Data Registry: Adam + L2 Neural Network

Batches size: 64

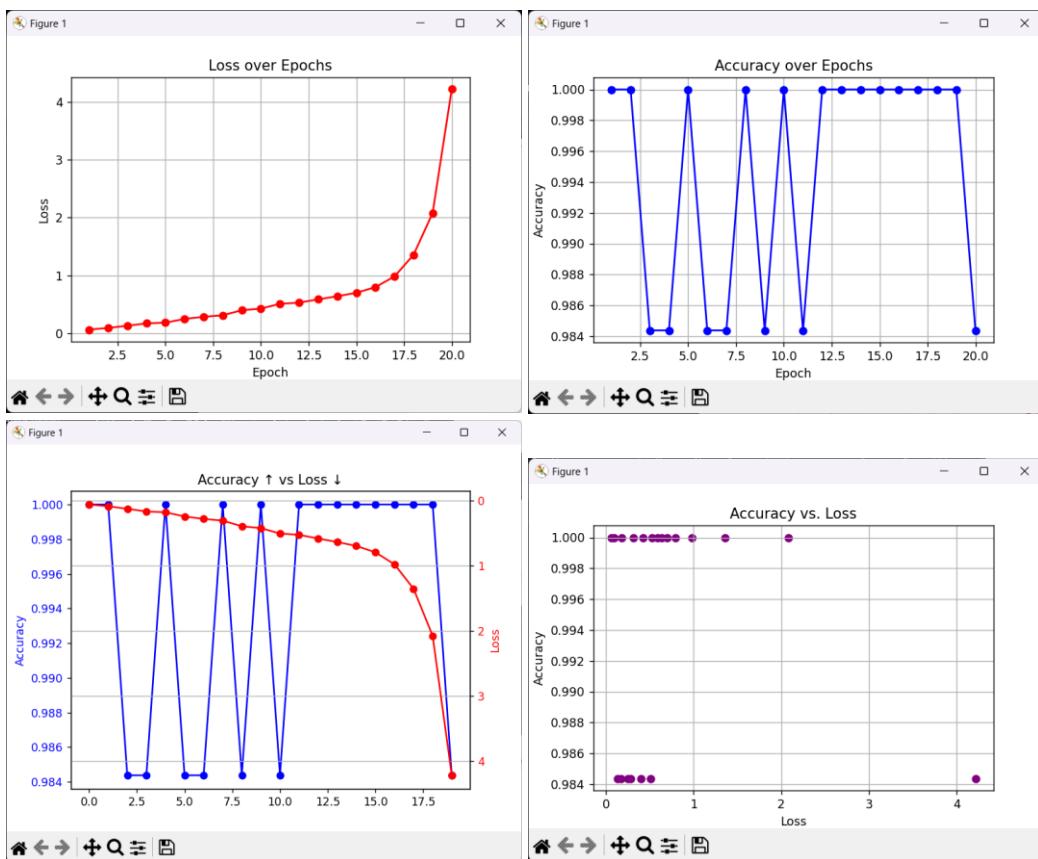
Epochs: 20

Hidden Layers: 2

Number of Neurons: 128

Optimizer: True

Regulation: True



Correlation: -0.0918

## Fase de Entrenamiento: Batches

### Data Registry: Basic Neural Network

Batches size: 64

Batches: 20

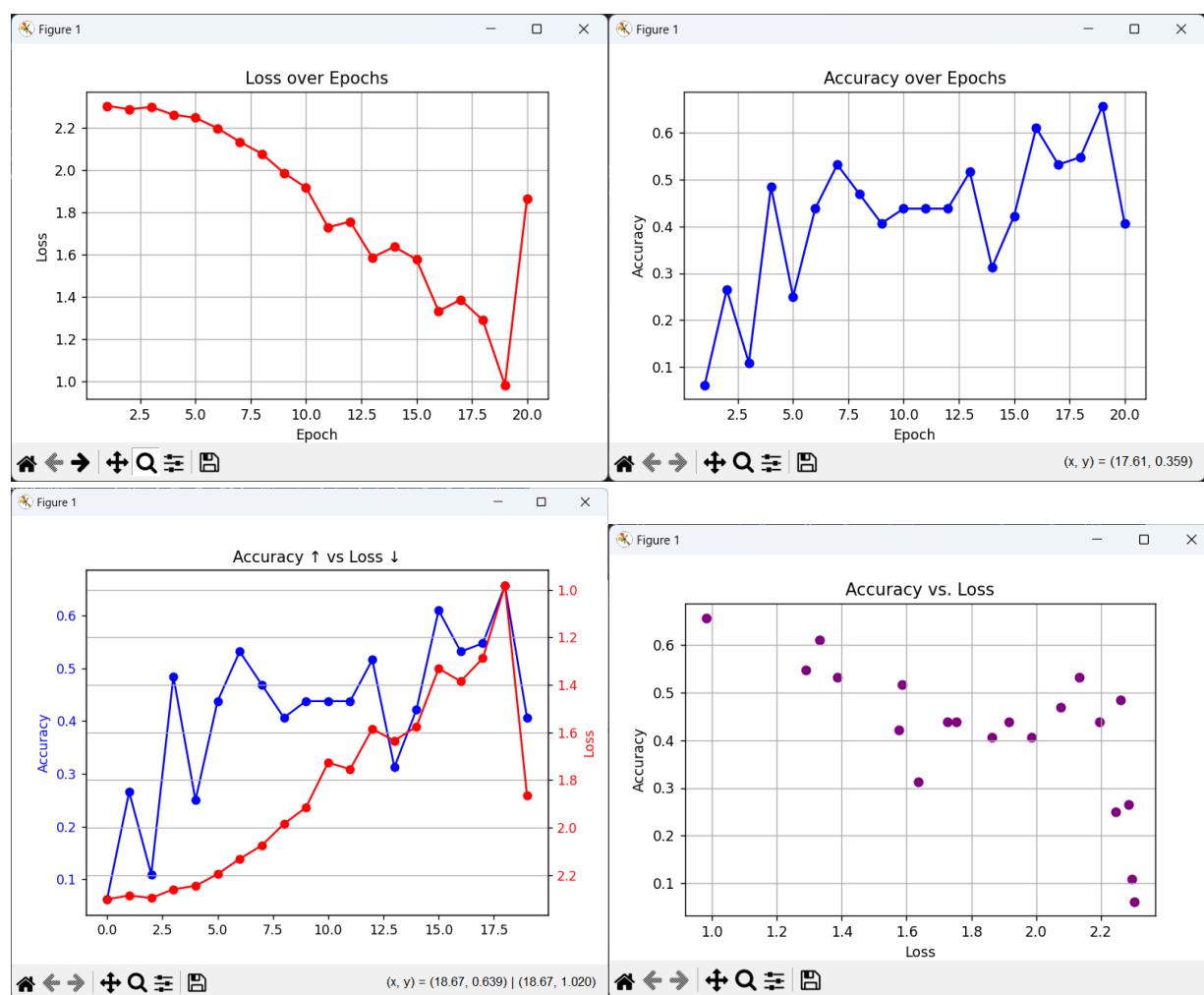
Hidden Layers: 1

Number of Neurons: 128

Optimizer: False

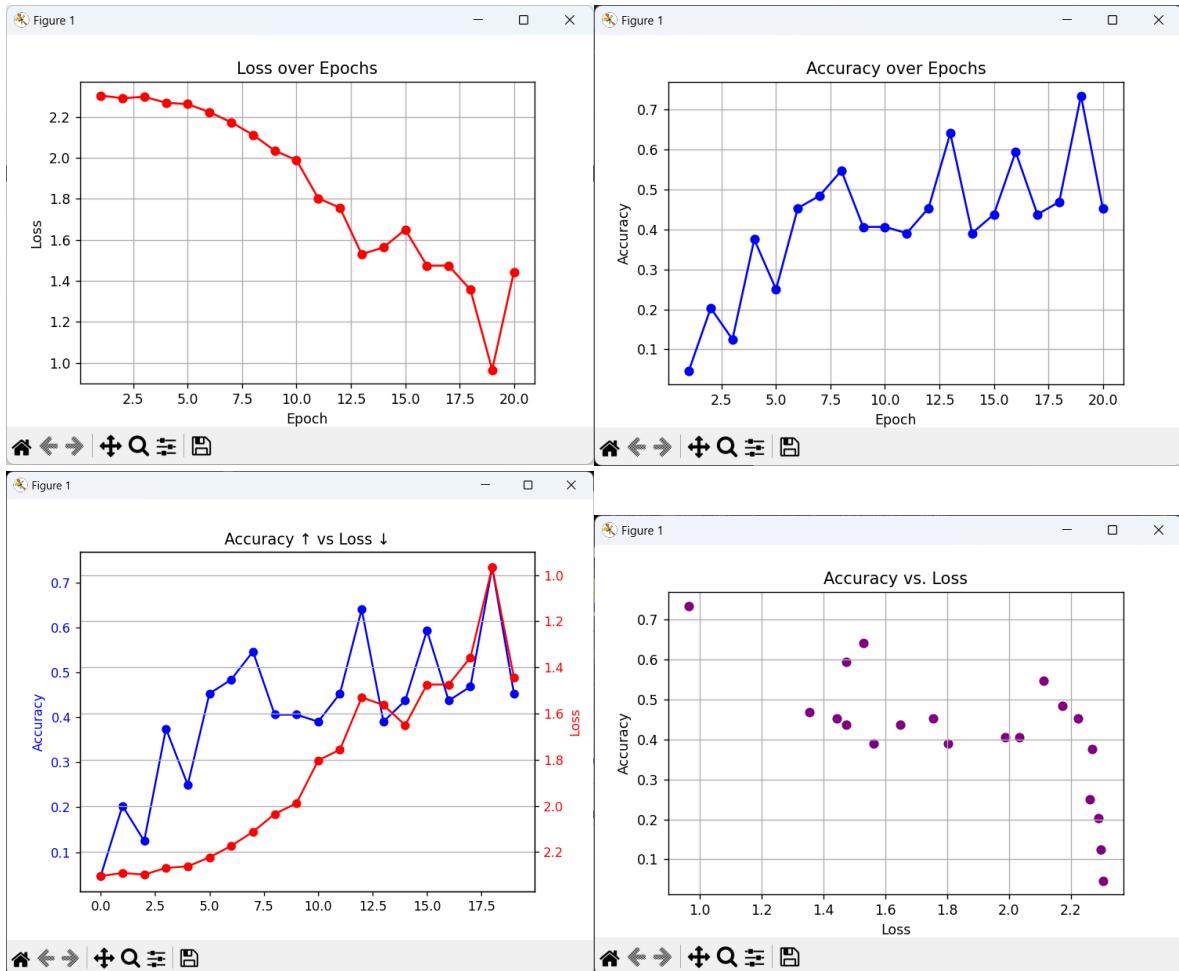
Regulation: False

**Try: 1**



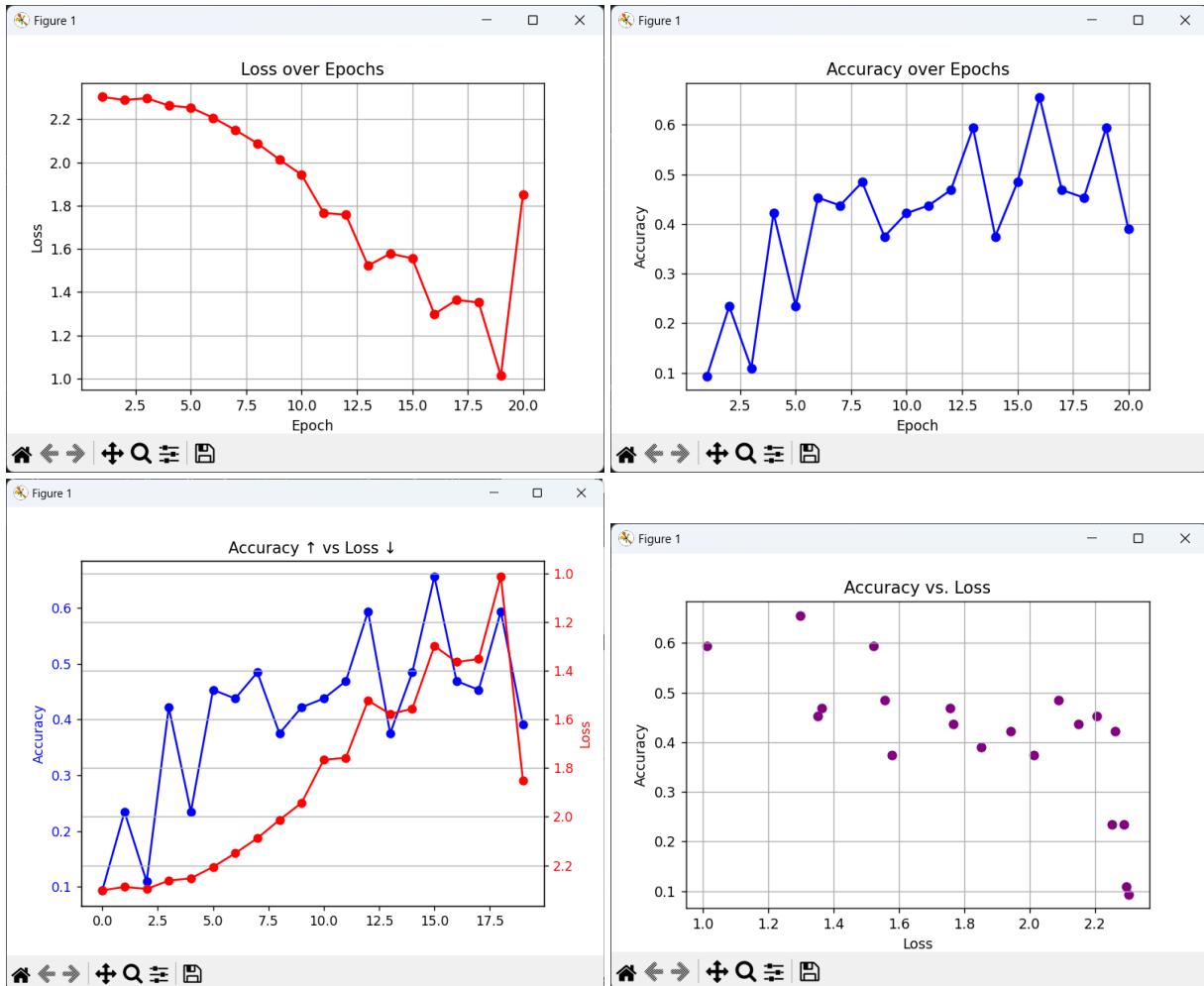
Correlation: -0.6977

## Try: 2



Correlation: -0.7080

### Try: 3



Correlation: -0.7265

## Data Registry: Basic Neural Network

Batches size: 64

Epochs: 100

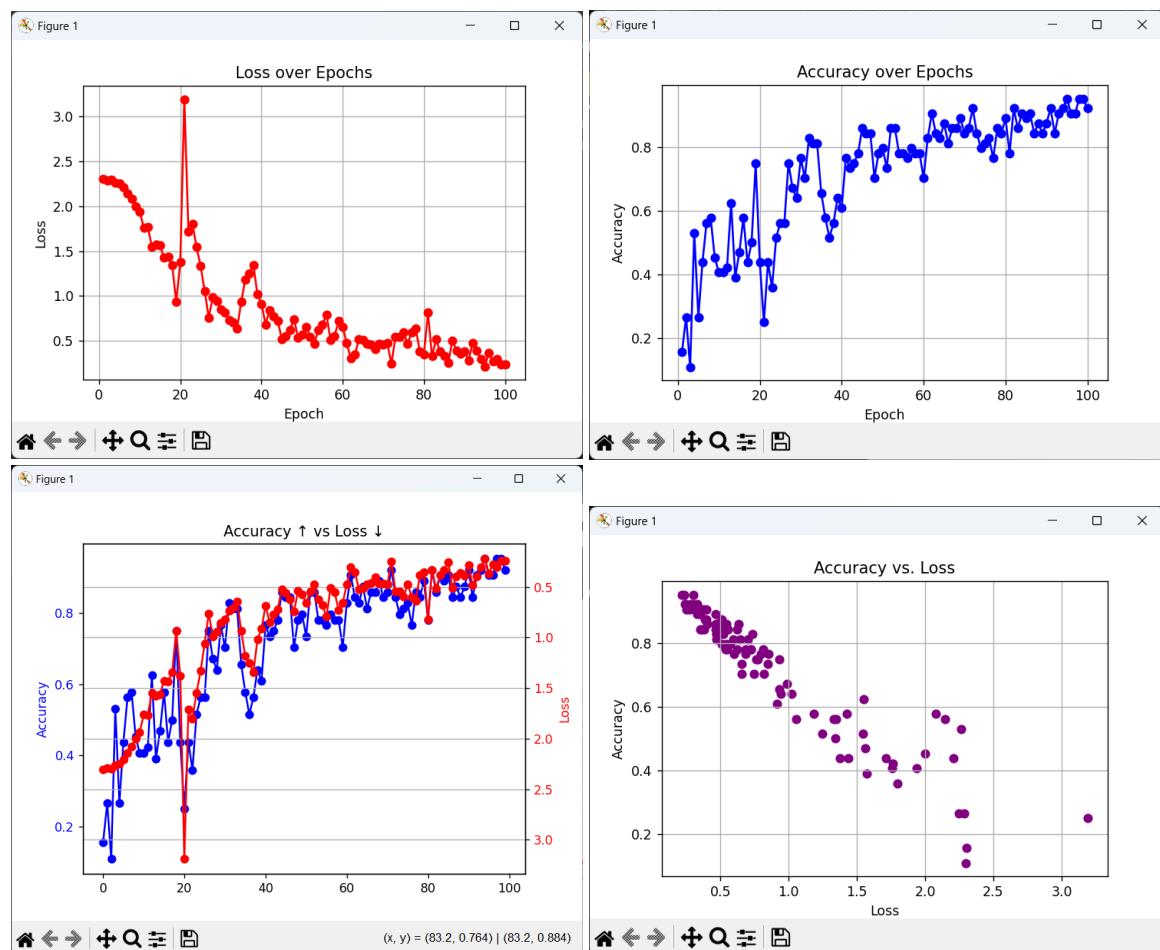
Hidden Layers: 1

Number of Neurons: 128

Optimizer: False

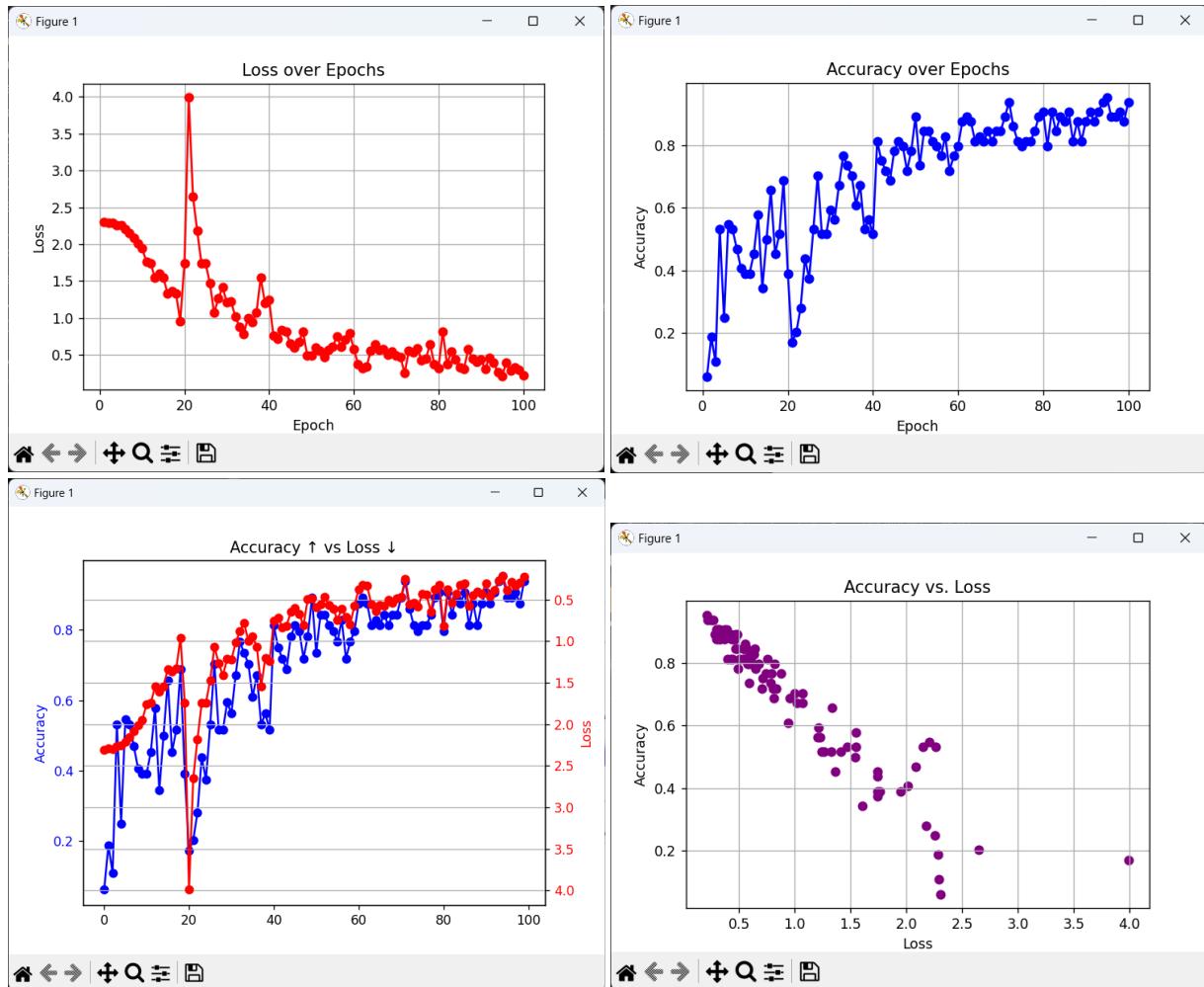
Regulation: False

Try: 1

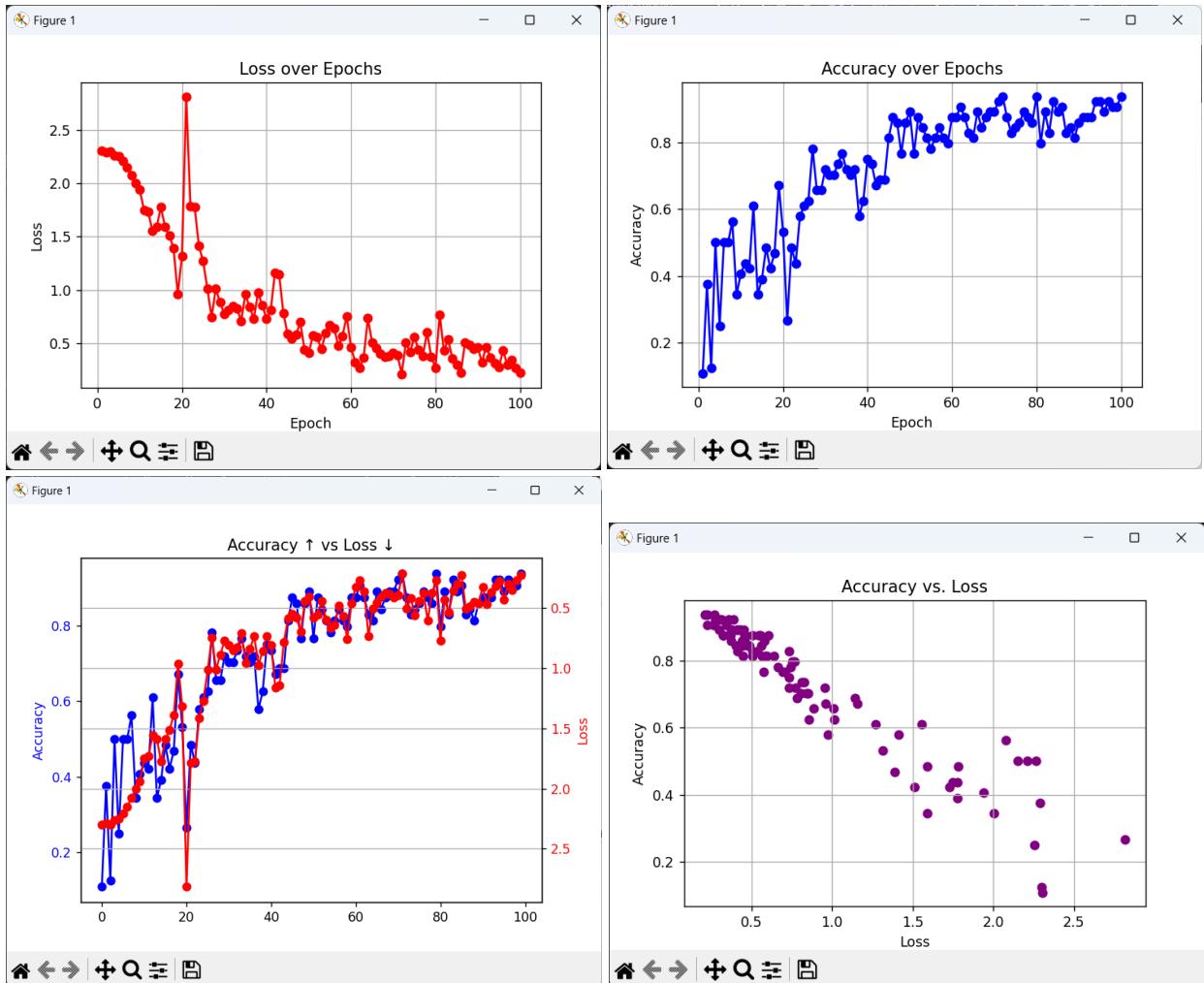


Correlation: -0.9387

Try: 2



Try 3:



Correlation: -0.9484

## Data Registry: Two-Layer Neural Network

Batches size: 64

Epochs: 20

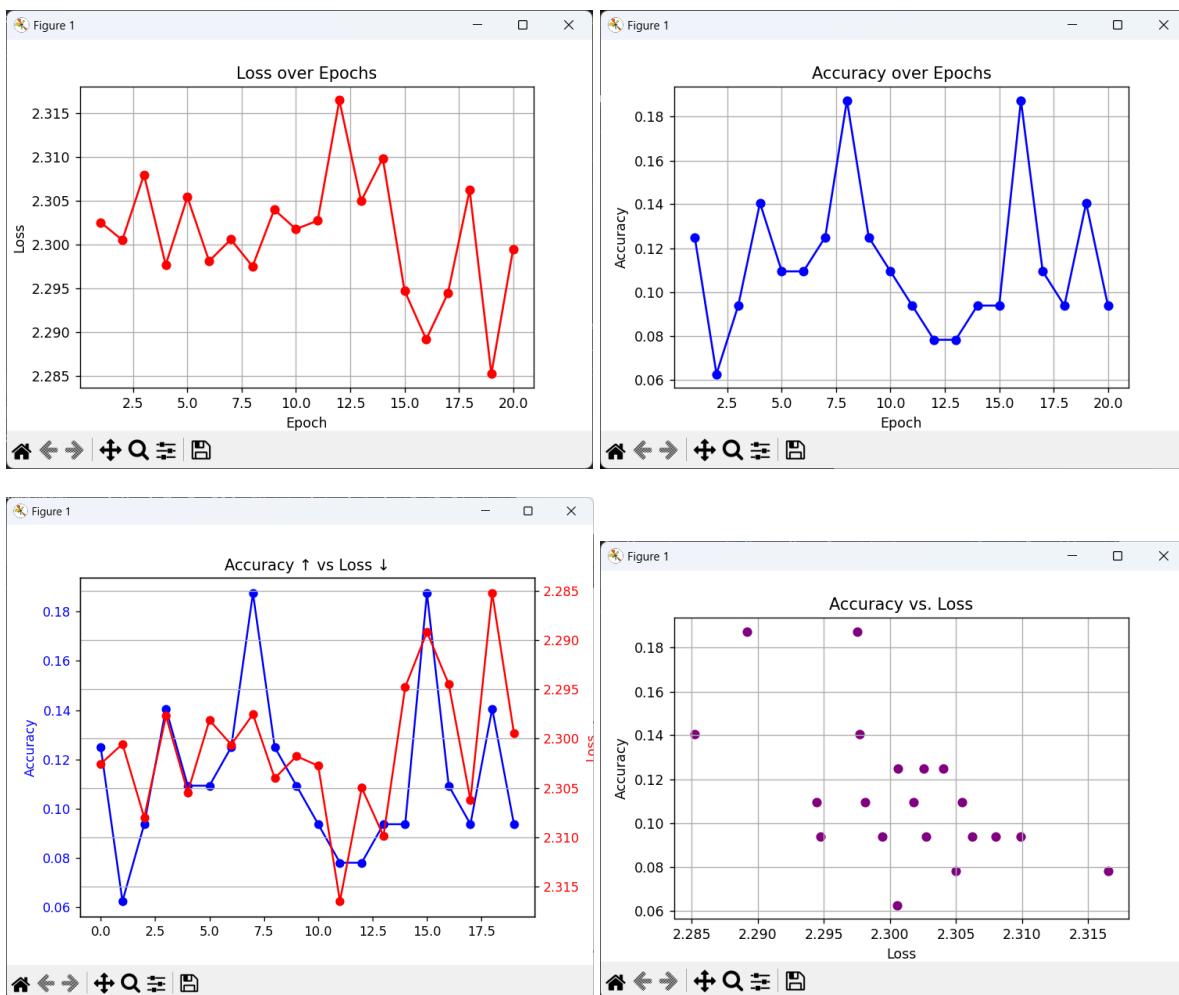
Hidden Layers: 2

Number of Neurons: 128

Optimizer: False

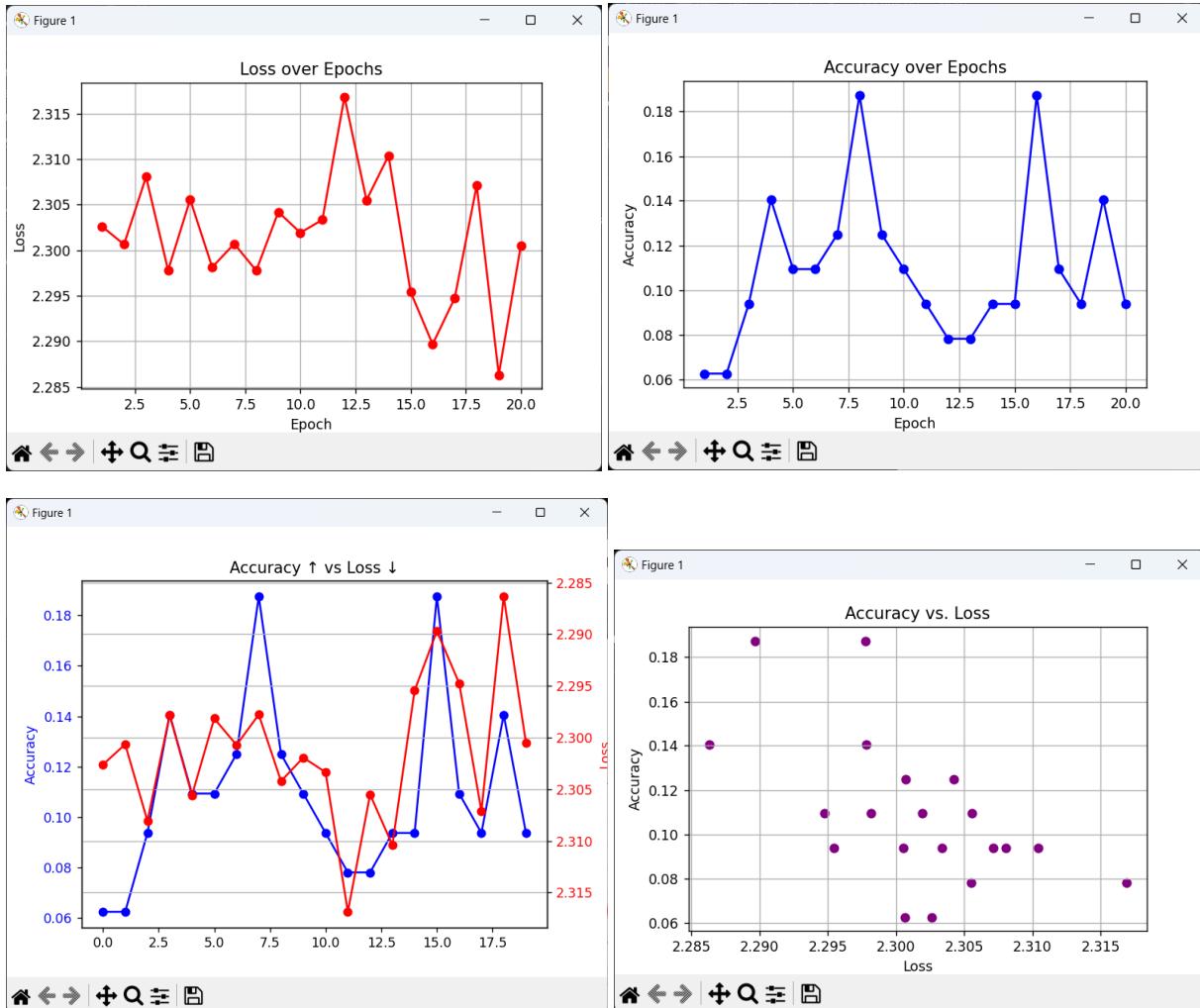
Regulation: False

Try 1:



Correlation: -0.5798

Try 2:



Correlation: -0.5781

## Data Registry: Two-Layer Neural Network

Batches size: 64

Epochs: 300

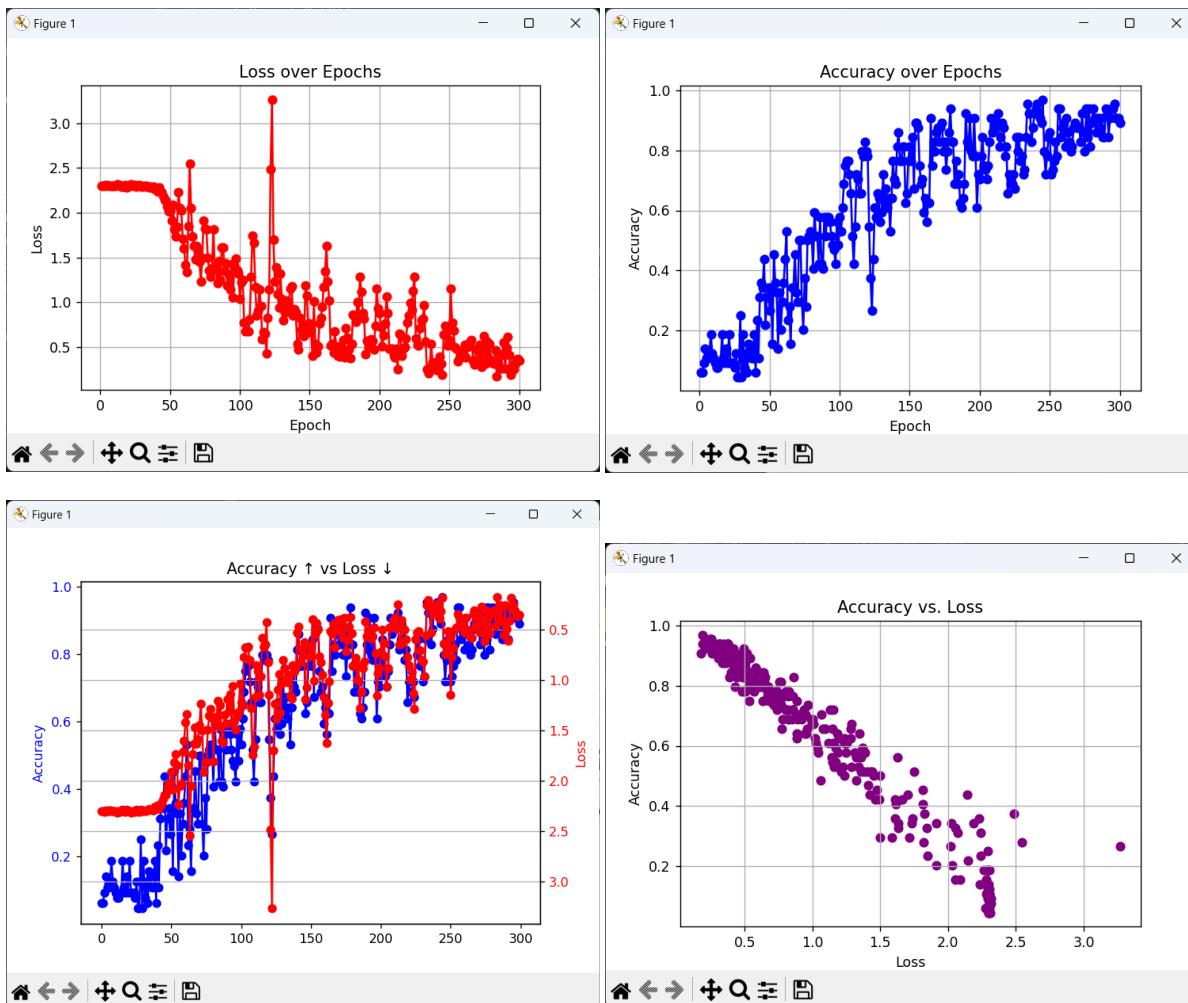
Hidden Layers: 2

Number of Neurons: 128

Optimizer: False

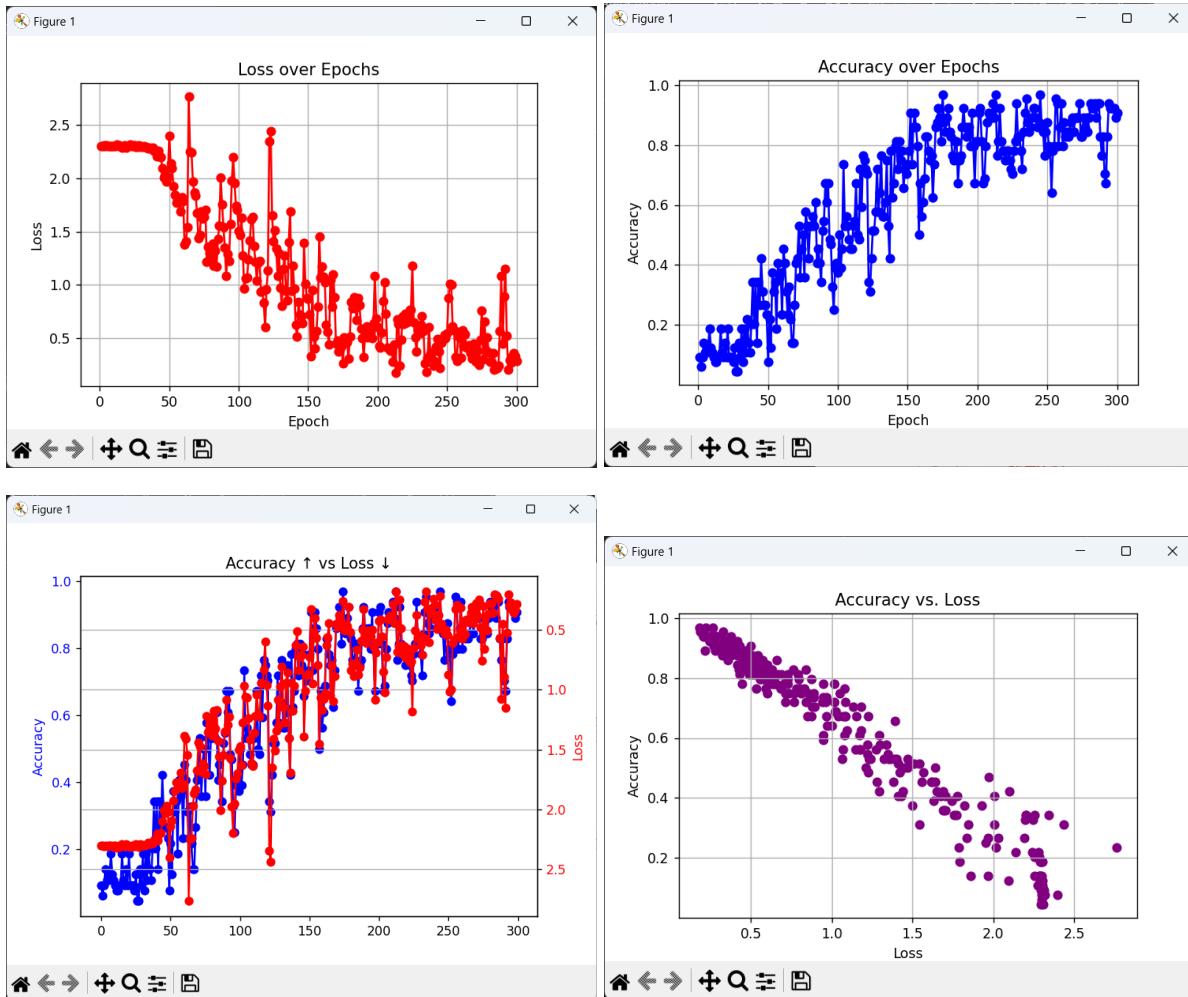
Regulation: False

Try: 1



Correlation: -0.9730

Try: 2



Correlation: -0.9765

## Data Registry: Adam Neural Network

Batches size: 64

Epochs: 20

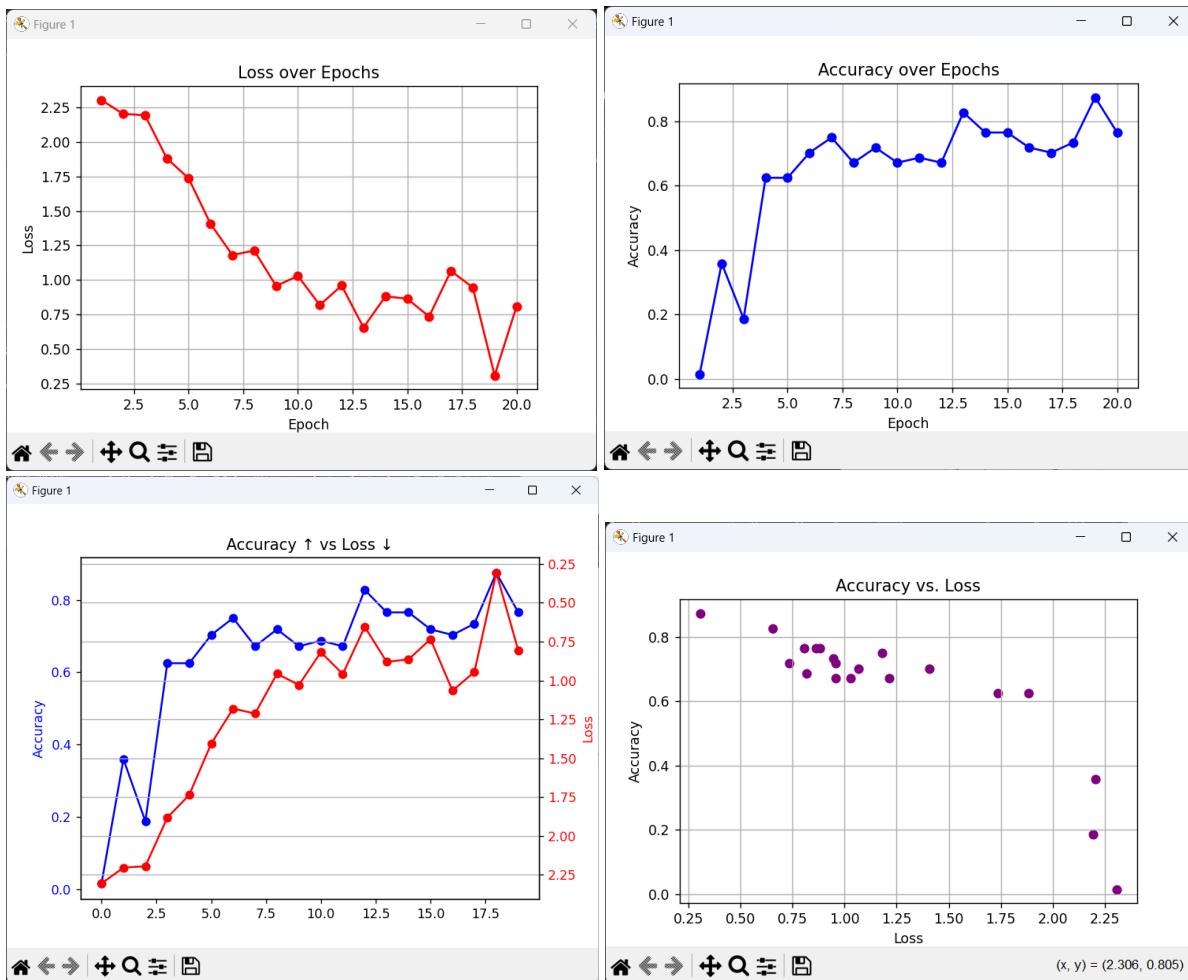
Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

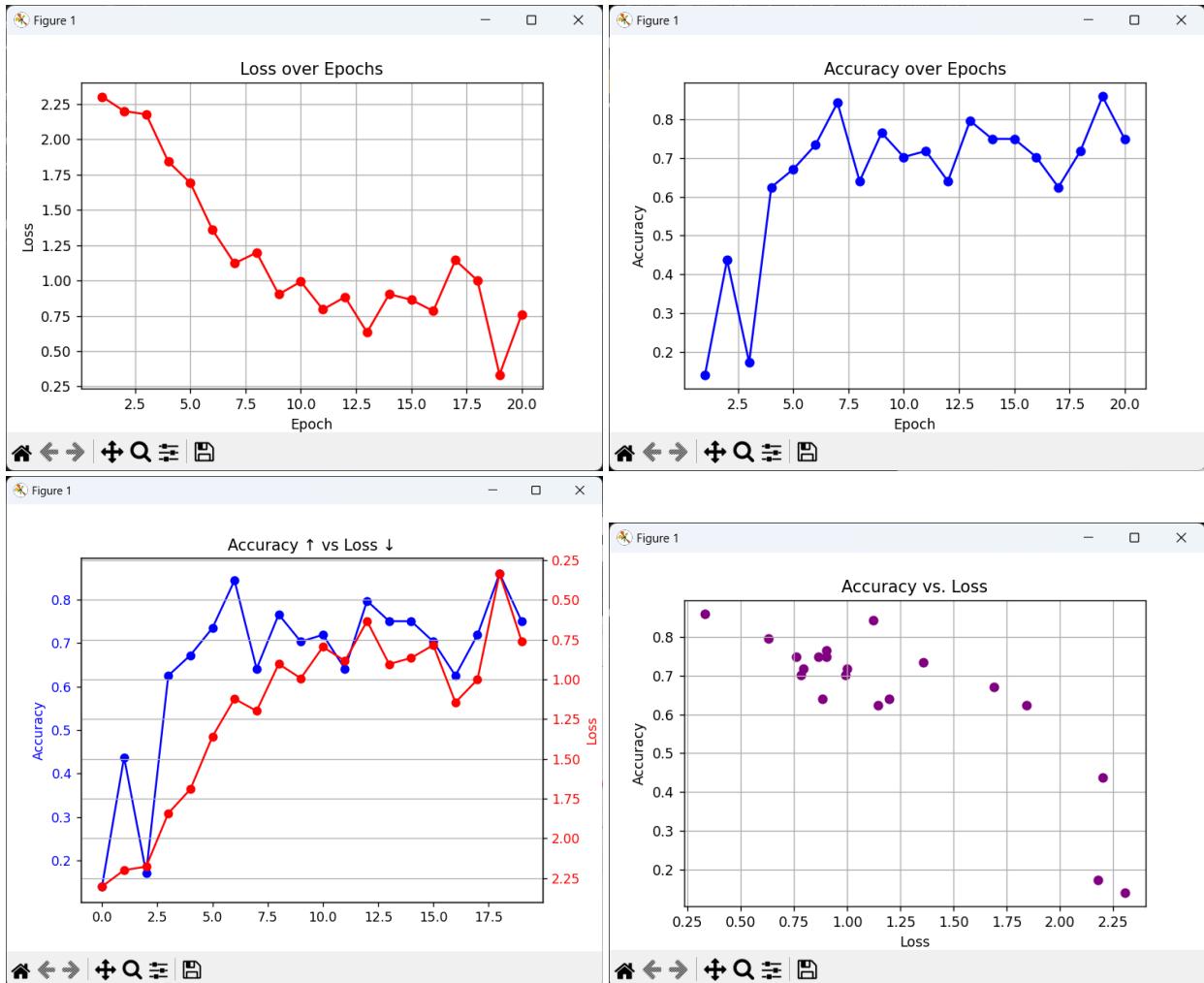
Regulation: False

Try: 1



Correlation: -0.8731

Try: 2



Correlation: -0.8443

## Data Registry: Adam Neural Network

Batches size: 64

Epochs: 100

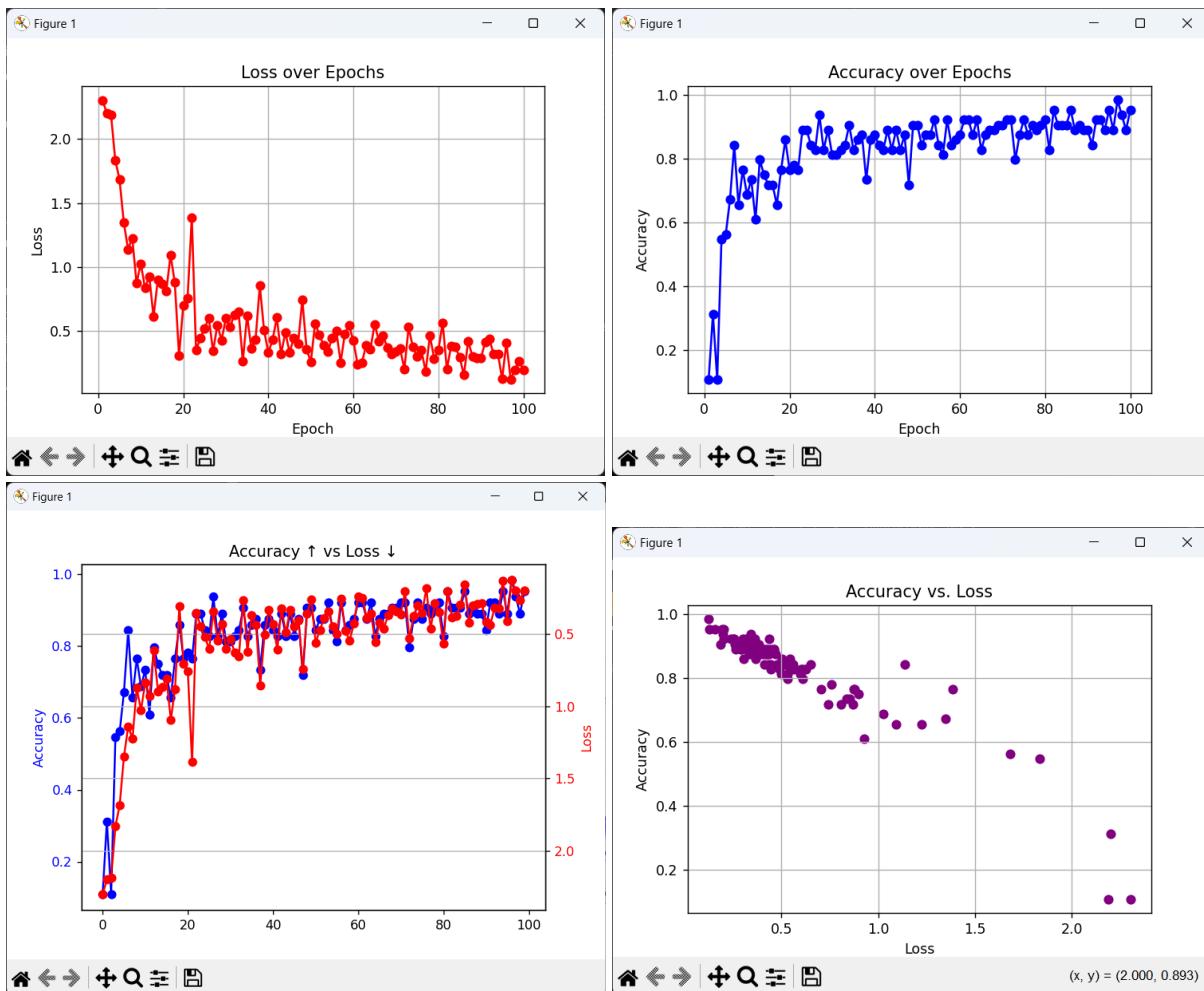
Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

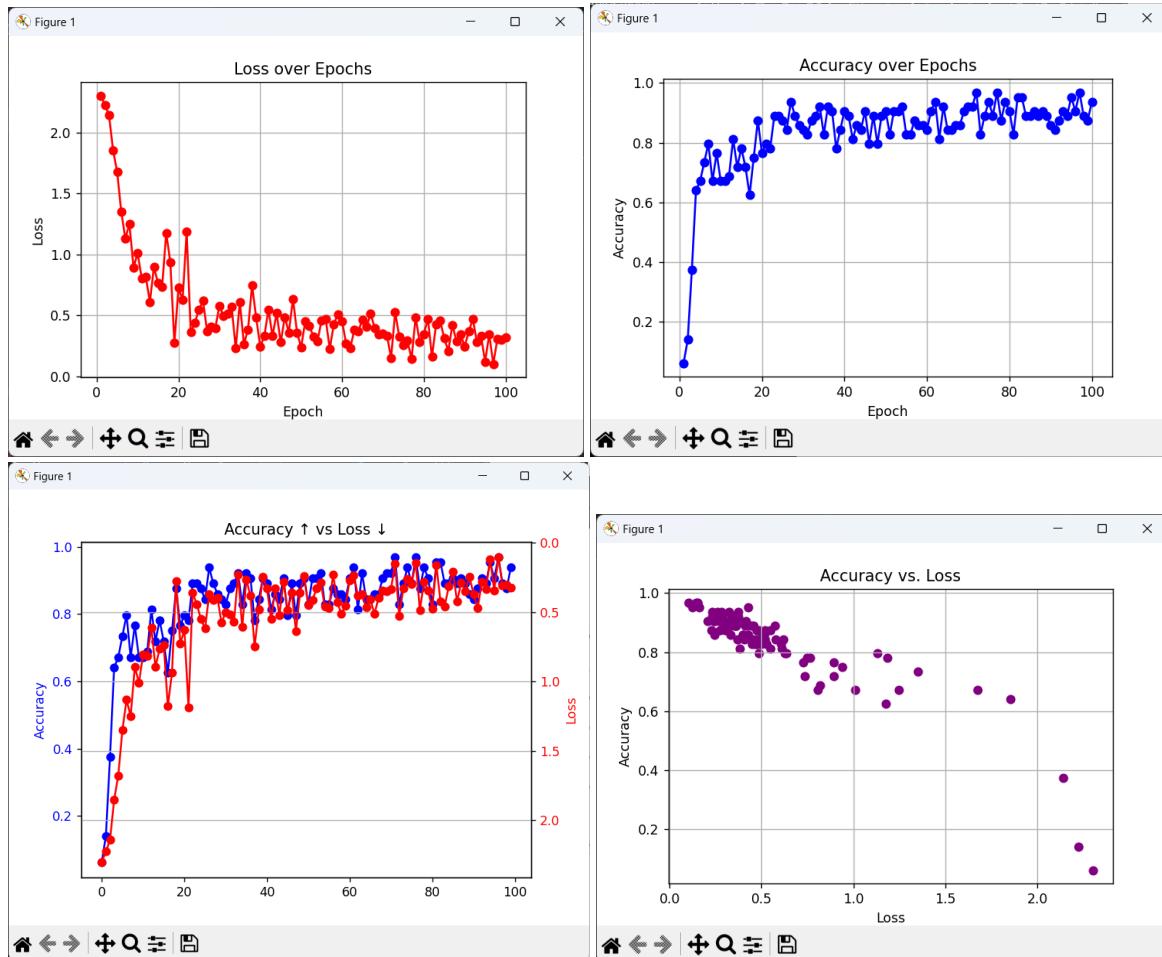
Regulation: False

Try: 1



Correlation: -0.9405

Try: 2



Correlation: -0.9153

## Data Registry: Adam+L2 Neural Network

Batches size: 64

Epochs: 20

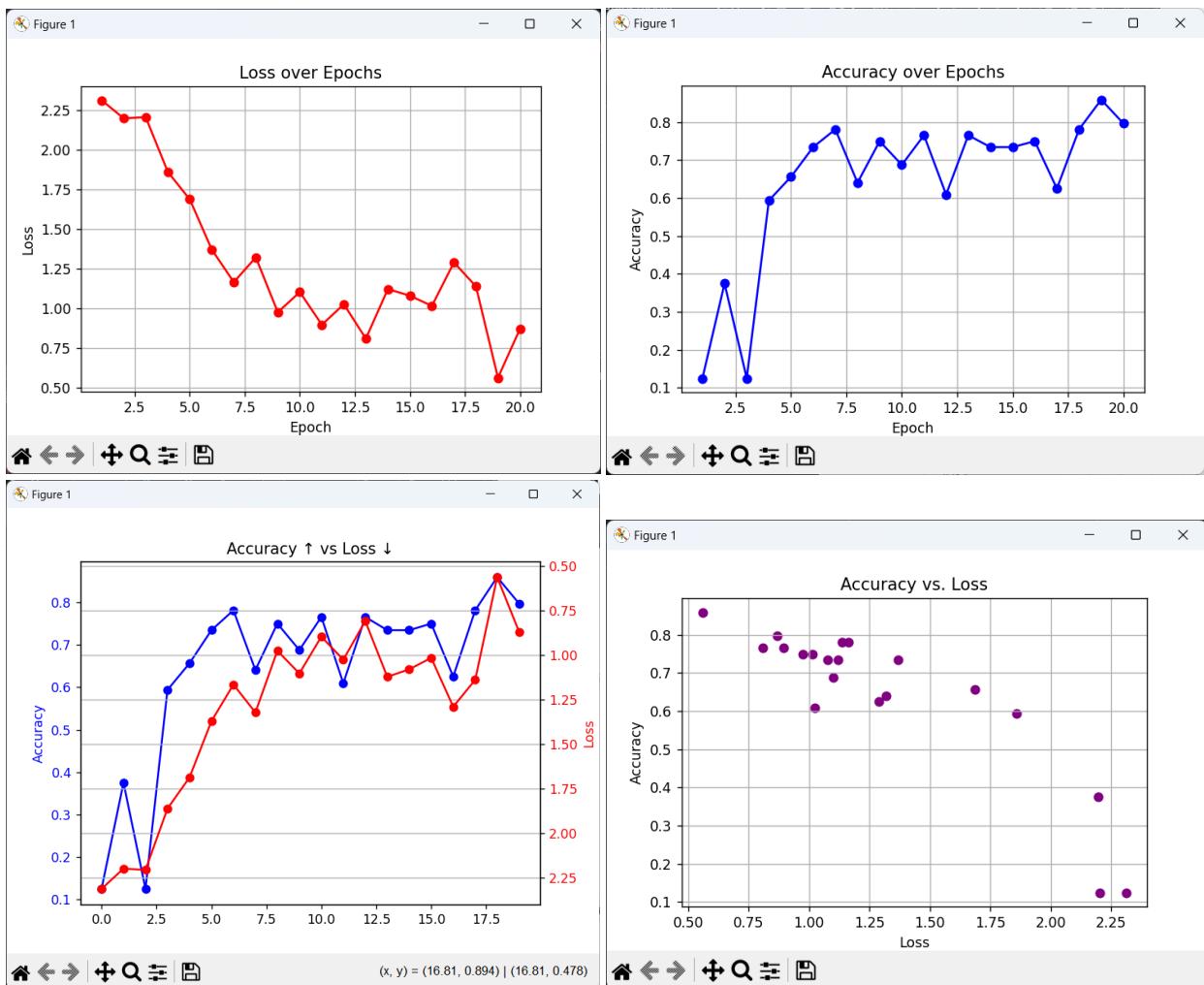
Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

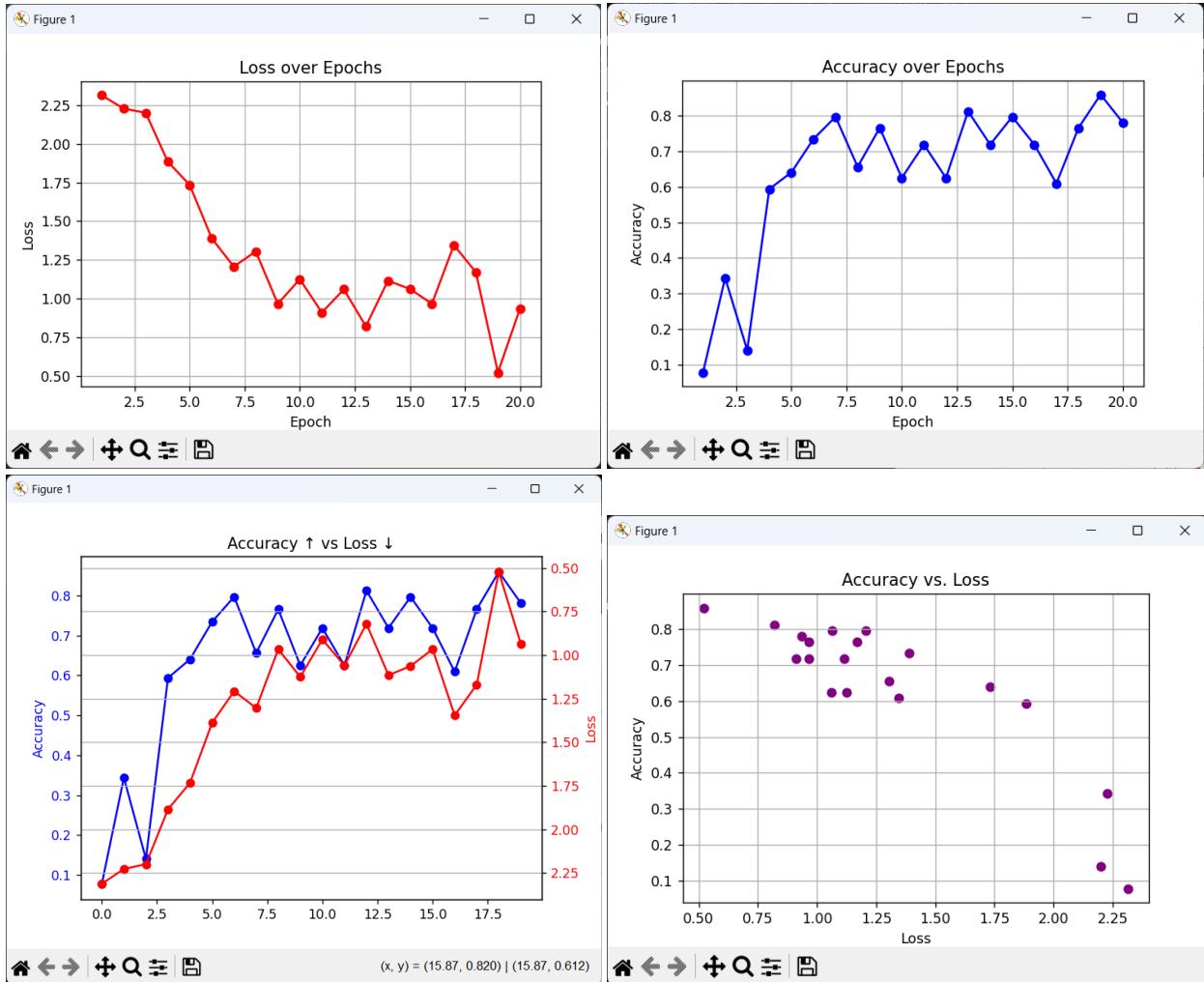
Regulation: True l2(0.001)

Try: 1



Correlation: -0.8922

Try: 2



Correlation: -0.8830

## Data Registry: Adam+L2 Neural Network

Batches size: 64

Epochs: 100

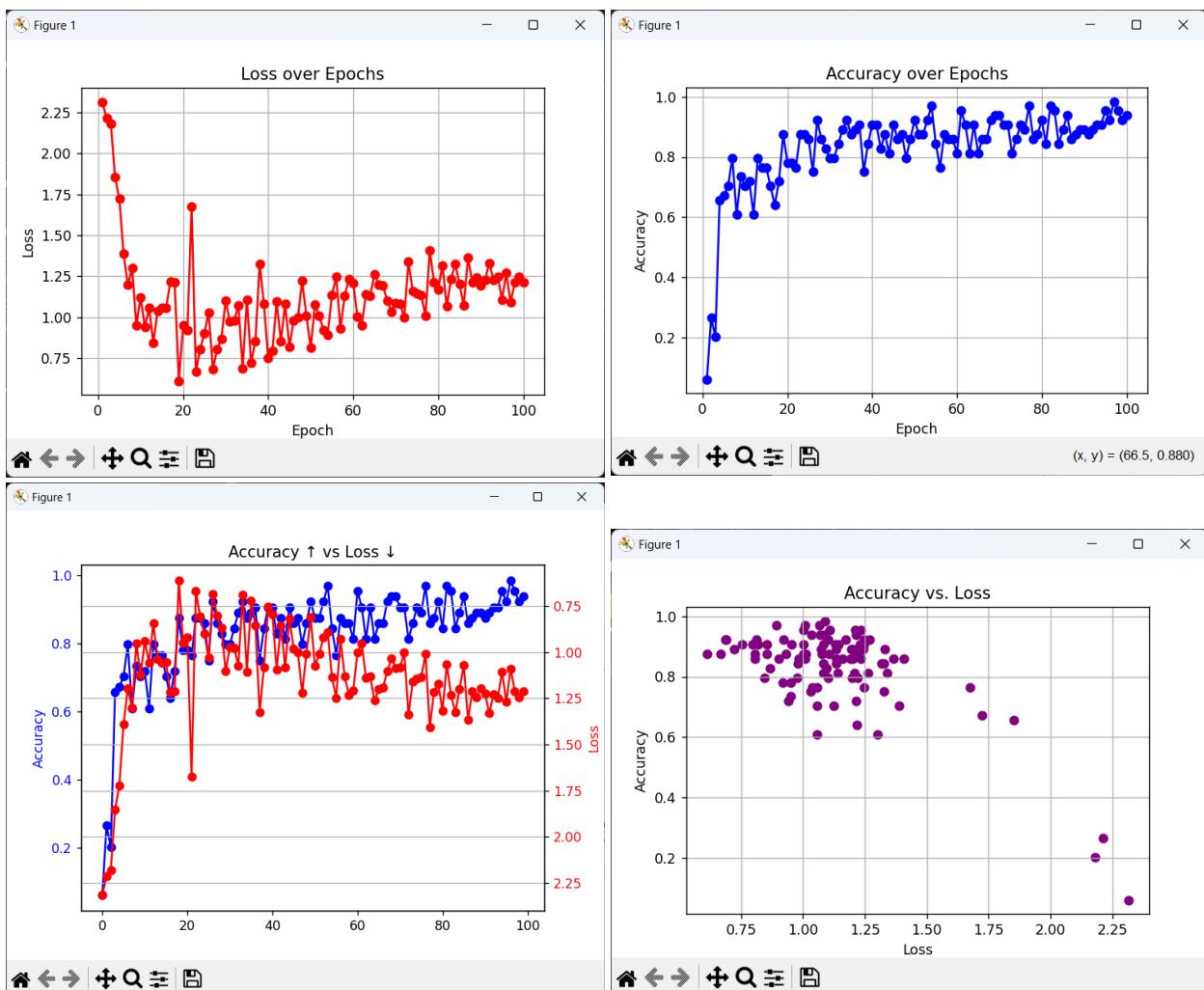
Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

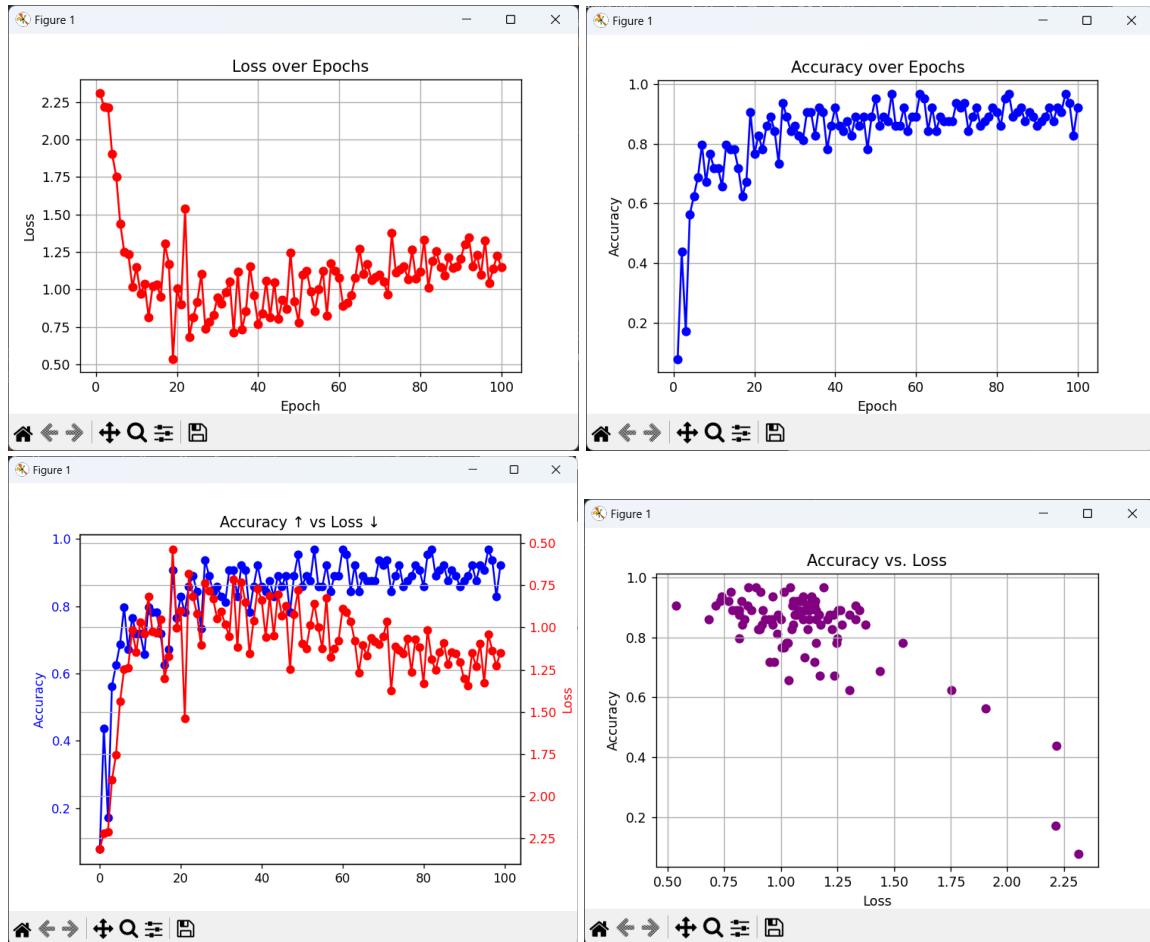
Regulation: True l2(0.001)

Try: 1



Correlation: -0.7009

## Try: 2



Correlation: -0.7437

## Data Registry: Adam+L2 Neural Network

Batches size: 64

Epochs: 20

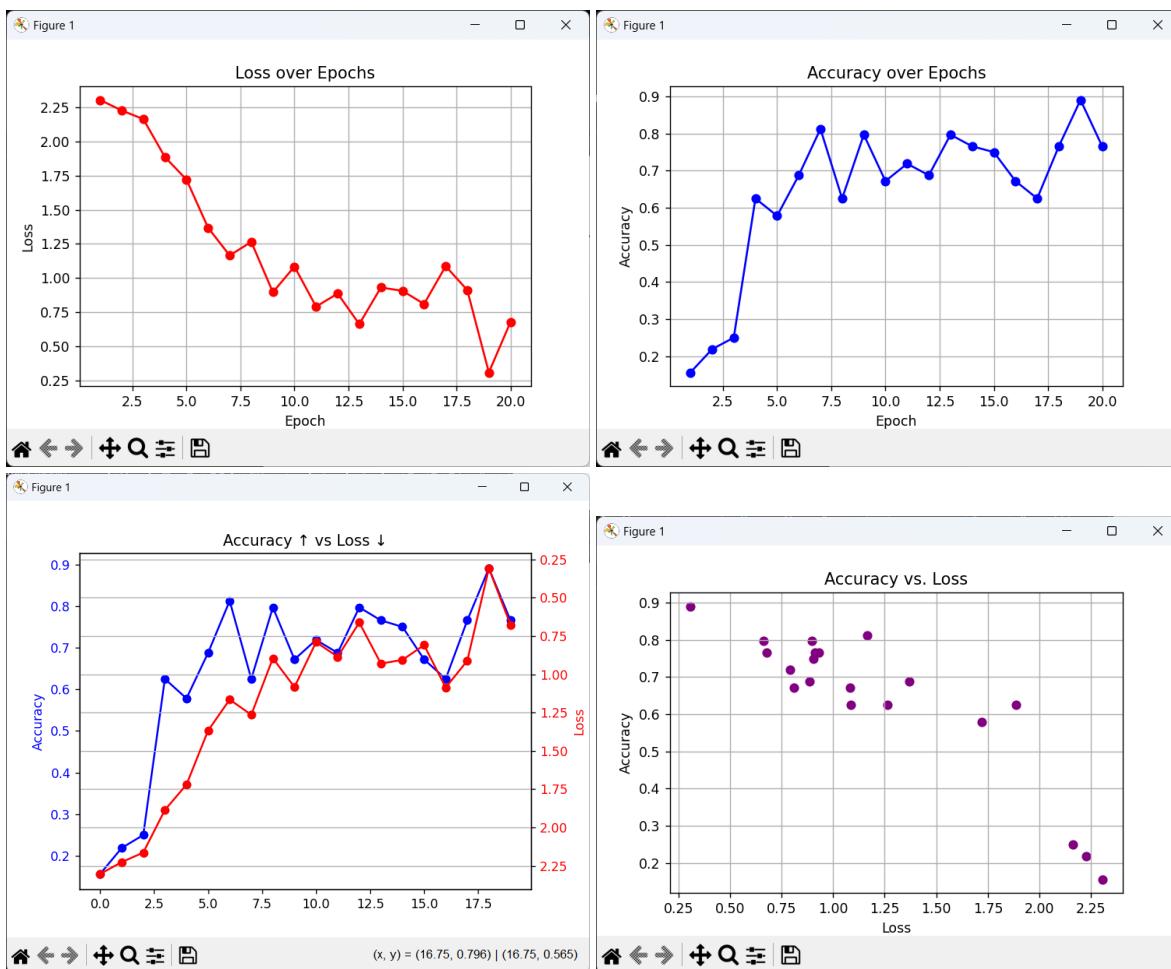
Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

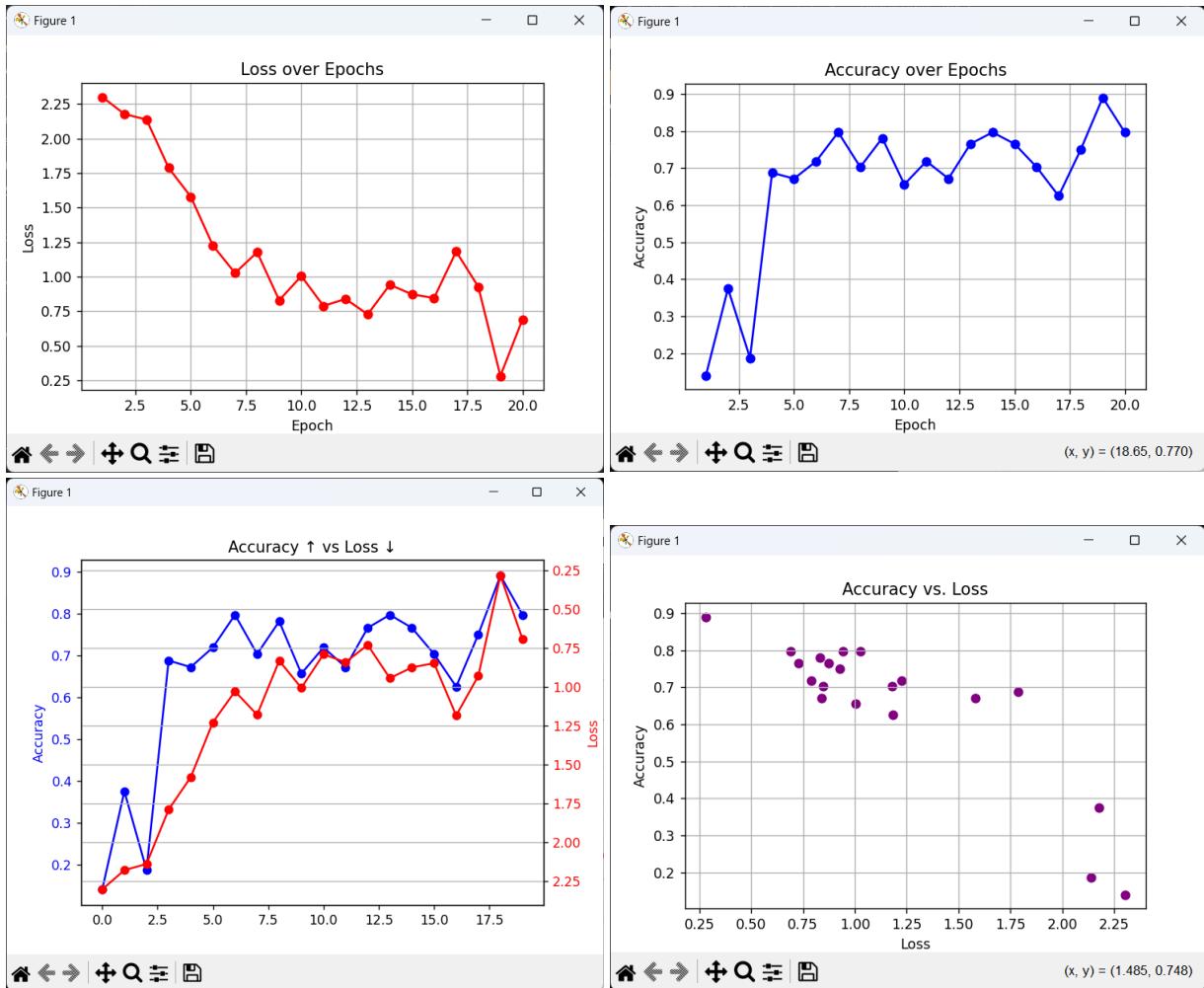
Regulation: True l2(0.0001)

Try: 1



Correlation: -0.9007

## Try: 2



Correlation: -0.8782

## Data Registry: Adam+L2 Neural Network

Batches size: 64

Epochs: 100

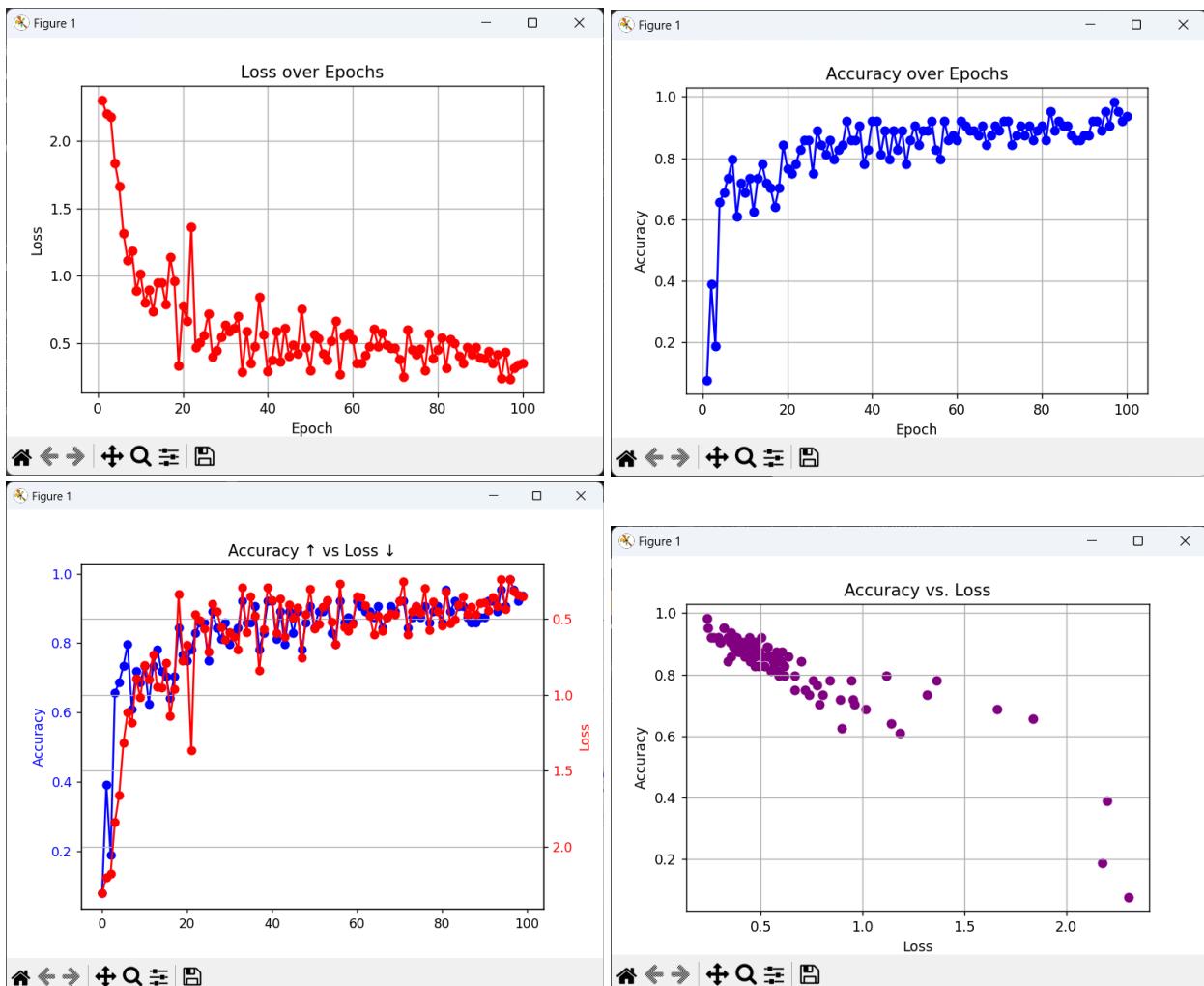
Hidden Layers: 1

Number of Neurons: 128

Optimizer: True

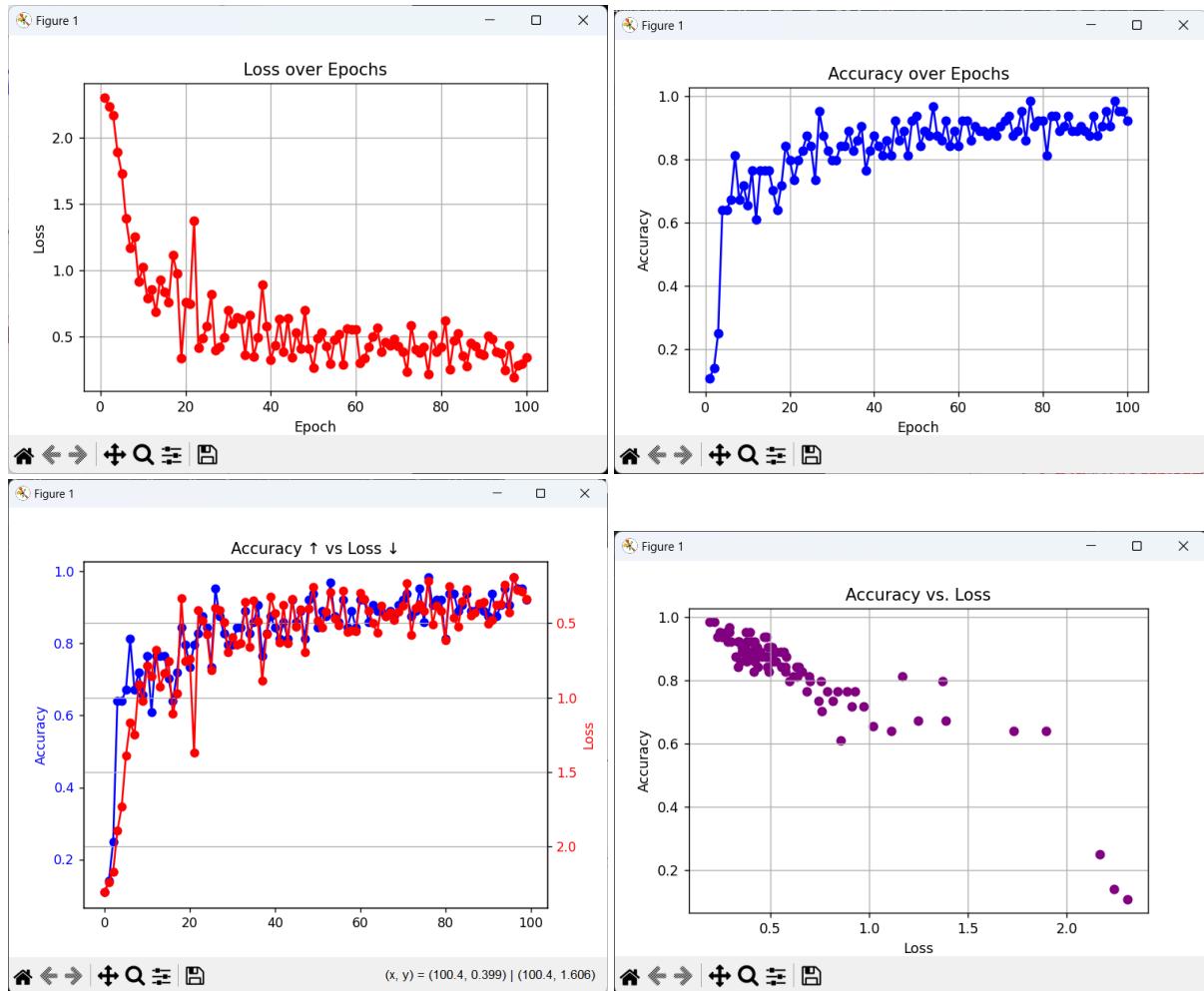
Regulation: True l2(0.0001)

Try: 1



Correlation: -0.9115

Try: 2



Correlation: -0.9182