Facultad de Ingeniería - Escuela de ingeniería

Tarea N°4

Sistemas Inteligentes

Profesor Billy Mark Peralta Marqués

Ayudante Javier Esteban Salazar Loyola

Matias Chiong

Cristobal Donoso

Jose Gonzalez

Alonso Nadeau

29 de noviembre de 2021.

Índice

Introducción

El en presente informe se aborda un problema específico de retención de clientes, esto usando redes neuronales, las redes neuronales provienen de la idea de lograr imitar el funcionamiento de las redes neuronales de los seres vivos logrando así representar hipótesis no lineales.

Para empezar la resolución del problema, iniciamos descargando la base de datos facilitada por el docente para luego empezar con la codificación de la resolución de los problemas planteados. Luego, en la codificación iniciamos estableciendo las capas de la red neuronal, la red neuronal se entrena usando un 75% de la base de datos para entrenar a la inteligencia y luego el otro 25% para realizar testeos, la codificación se realiza utilizando el lenguaje de programación Python.

Se utilizarán las librerías de numpy, matplotlib para el manejo numérico de variables, pandas para poder abrir y cargar el archivo csv, random para poder generar el número de la semilla de forma randomizada, keras para poder crear y ejecutar las capas ocultas y sklearn, la cual permite entrenar a la computadora.

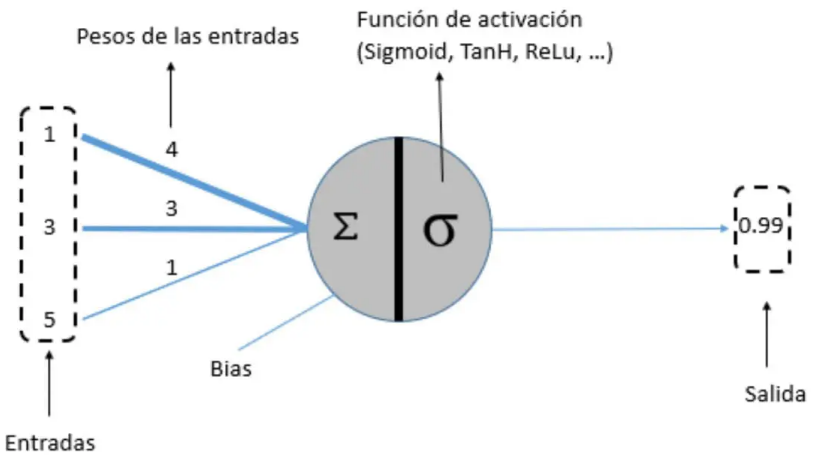
Problemática

El contexto del problema consiste en una base de datos que posee 10.000 clientes con 14 variables originales, donde la variable de salida es binaria, e indica si la persona se retuvo o no como cliente dentro de un plazo prefijado, se nos pide responder una serie de preguntas basadas en el comportamiento y resultados de las neuronas.

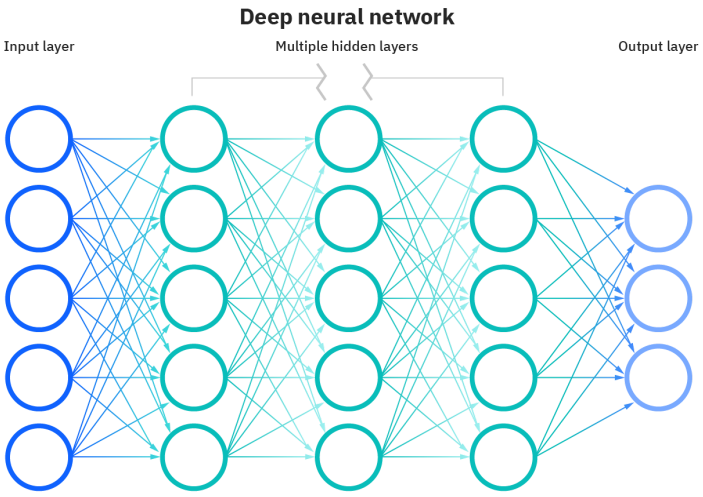
Redes Neuronales

Las redes neuronales, también conocidas como redes neuronales artificiales (ANN) o redes neuronales simuladas (SNN), son un subconjunto de Machine Learning y están en el corazón de los algoritmos de deep learning. Su nombre y estructura están inspirados por el cerebro humano, imitando la forma en que las neuronas biológicas se señalan entre sí.

Están compuestas por capas de nodos, que contienen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo, o neurona artificial, se conecta a otro y tiene un peso y un umbral asociados. Si la salida de cualquier nodo individual está por encima del valor del umbral especificado se activa, enviando datos a la siguiente capa de la red. De lo contrario, no se pasan datos a la siguiente capa de la red



Las redes neuronales dependen de los datos de entrenamiento para aprender y mejorar su precisión con el tiempo. Sin embargo, una vez que estos algoritmos se ajustan a la precisión, son potentes herramientas en la ciencia de computación e inteligencia artificial, lo que nos permite clasificar y agrupar los datos a una alta velocidad.



Desarrollo: Puntos a resolver

Las preguntas a responder son las siguientes:

1. Considerando la eliminación de variables espúreas, divida la base de datos en los conjuntos de entrenamiento y testeo considerando 75% para entrenamiento. Use una semilla con números entre 1 y 100. Reporte los primeros 5 registros del set de entrenamiento y testeo.

```
[[651 0 0 ... 1 0 138113.71]
 [606 1 1 ... 0 1 96985.58]
 [522 0 0 ... 0 1 176780.39]
 ...
 [583 0 0 ... 1 0 12538.92]
 [679 0 1 ... 0 1 172501.38]
 [489 1 0 ... 1 0 37240.11]]
[[769 2 1 ... 0 0 104393.78]
 [534 0 0 ... 1 1 70179.0]
 [729 0 0 ... 0 1 105409.31]
 ...
 [528 2 0 ... 1 1 170309.19]
 [682 0 0 ... 1 1 63737.19]
 [748 0 1 ... 0 0 60416.76]]
(7500, 10)
(2500, 10)
<class 'numpy.ndarray'>
```

2. Utilizando una regresión logística indique el resultado en el set de testeo. reporte el resultado de precisión obtenida (accuracy en inglés). Recuerde que una regresión logística es equivalente a una red neuronal sin capas ocultas.

Regresión Logistica

Valor	Acurracy
1	0.7736
3	0.8065
5	0.8059
10	0.8087
15	0.8095
20	0.8085
25	0.8091
30	0.8080
35	0.8075

(Tabla 1. Resultados sin capa oculta)

Podemos observar que el valor de la precisión obtenida varía de forma randomizada entre las neuronas pero siempre está dentro del rango entre 0.8065 y 0.8095, algunas de ellas obteniendo valores que demuestran más presión y otras que no. **(Anexo C, Resultados Punto 2).**

Adicionalmente se realizó una predicción, en el cual se utilizó los datos de entrenamiento para poder generar una predicción de la regresión logística, resultando en la imagen siguiente, en donde cada par ordenado de la tabla si se suma da como resultado 1.

```
[[0.84157624 0.15842376]
 [0.9091981  0.0908019 ]
 [0.8886782  0.1113218 ]
 ...
 [0.72578148 0.27421852]
 [0.90718772 0.09281228]
 [0.33783415 0.66216585]]
```

3. Considere sólo una capa oculta. ¿Cómo evoluciona la precisión en el set de testeo si varía el número de neuronas en la capa oculta?. Considere los valores en lista [1,3,5,10,15,20,25,30,35]. Indique los valores en una tabla. ¿Qué puede concluir de los resultados?

Neuronas	2
Valor	Accuracy
1	0.7952
3	0.7952
5	0.7952
10	0.8287
15	0.8317
20	0.8343
25	0.8352
30	0.8335
35	0.8376

Neuronas	6
Valor	Accuracy
1	0.794
3	0.7952
5	0.8171
10	0.8323
15	0.8368
20	0.8361
25	0.8363
30	0.8369
35	0.8369

Neuronas	10
Valor	Accuracy
1	0.7952
3	0.7952
5	0.8232
10	0.8329
15	0.8352
20	0.8368
25	0.8357
30	0.8357
35	0.8369

(Tabla 2. Resultados con una capa oculta)

A medida que el valor va aumentando, la precisión aumenta con el, por lo que se puede apreciar una relación lineal directa entre los valores de la tabla. Comparando

con las otras tablas los valores se asemejan, por lo que decimos que los datos tienden a ser homogéneos **(Anexo C, Resultados Punto 3).**

4. Considere 2 capas. Considere la combinación de neuronas considerando el set [5,10,15] para el número de neuronas para ambas capas. En este caso hay 9 combinaciones. Indique los valores de precisión en sets de testeo en una tabla. ¿Qué puede concluir de los resultados?

Combinaciones

testeo	1	2	3	4	5	6	7	8	9
capa 1	5	5	10	10	10	15	15	5	15
capa 2	5	10	5	10	15	10	15	15	5

(Tabla 3. combinaciones)

Testeo	Accuracy Epoch 5	Accuracy Epoch 10	Accuracy Epoch 15
1	0.8259	0.8347	0.8349
2	0.8251	0.8337	0.8337
3	0.8267	0.8331	0.8359
4	0.8279	0.8345	0.8345
5	0.8347	0.8372	0.8383
6	0.8347	0.8380	0.8404
7	0.8349	0.8401	0.8383
8	0.8284	0.8345	0.8373
9	0.8319	0.8369	0.8369

(Tabla 4. Resultados con dos capas)

Se puede observar un comportamiento homogéneo atribuible a la anterior normalización de datos, donde se percibe una pequeña diferencia entre los resultados de precisión de Epoch, la cual tiene un aumento en los testeos que en su primera capa poseen una mayor cantidad de neuronas con respecto a su segunda. No obstante, los testeos que contienen menor cantidad de neuronas en la primera capa tienen una menor precisión en los Epoch con respecto a su contraparte.

.

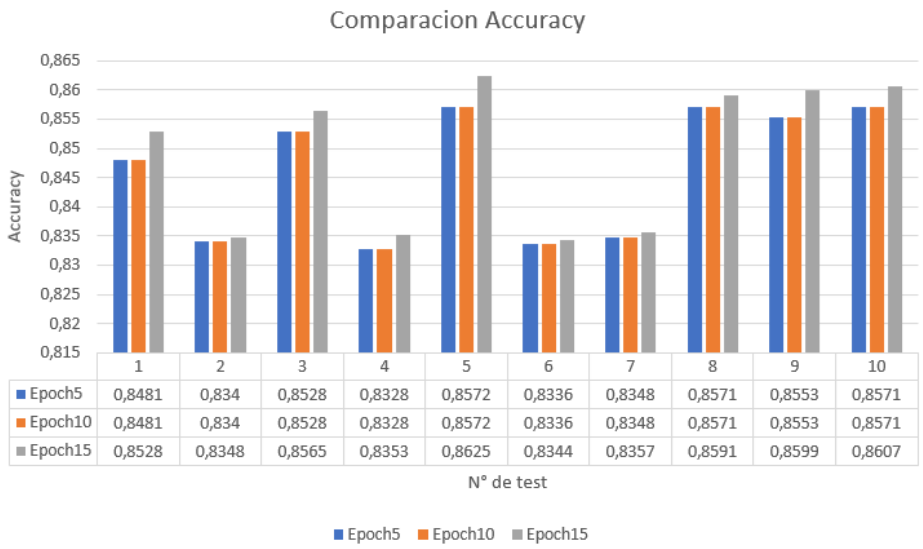
5. Considere 3 capas. En este caso elija 10 combinaciones considerando todo lo que ha analizado hasta el momento. Considerando las mismas preguntas del punto anterior.

capa 1	5	5	5	5	10	10	10	15	15	15
capa 2	5	5	10	10	10	10	10	15	15	15
capa 3	5	15	10	15	5	10	15	5	10	15

(Tabla 5.combinaciones)

Testeo	Accuracy Epoch 5	Accuracy Epoch 10	Accuracy Epoch 15
1	0.8299	0.8481	0.8528
2	0.8273	0.8340	0.8348
3	0.8327	0.8528	0.8565
4	0.8251	0.8328	0.8353
5	0.8444	0.8572	0.8625
6	0.8316	0.8336	0.8344
7	0.8297	0.8348	0.8357
8	0.8508	0.8571	0.8591
9	0.8395	0.8553	0.8599
10	0.8485	0.8571	0.8607

(Tabla 6. Resultados con 3 capas)



(Gráfico 1. Comparación de Accuracy entre las capas)

Con los datos obtenidos podemos ver que a diferencia de cuando teníamos dos capas el tener una capa mas, aun asi, estemos usando la misma cantidad de neuronas en las tres capas, obtenemos que el tener 3 capas es más preciso que tener dos capas con 5 neuronas, por ejemplo, esto lo podemos ver más claramente comparando el test 1 de la tabla 6 y 4 en el Epoch 5,10 y 15.

Se puede observar que el tener más neuronas en la tercera capa en comparación a las demás produce un efecto negativo, a causa que la precisión se reduce, esto se puede ver reflejado en el test 2 y 7. Por el contrario, al tener más neuronas en la capa 1 y 2 es más efectivo, debido a que la precisión mejora. Al existir más combinaciones la precisión aumenta, ya que se obtiene un valor mucho más cercano al uno.

6. Considerando todos los experimentos, ¿ que puede concluir de forma general? Indique 4 maneras realistas en las que usted cree que podría mejorar el resultado.

Como grupo podemos concluir que en la implementación de capas ocultas la precisión en los set es más exacta a medida que aumentan la cantidad de neuronas en los experimentos (véase la tabla 2), no obstante, no ocurre con la regresión logística debido a que la precisión que se obtiene es mucho menor que cuando se aplican las capas (véase las tablas 1 y 2).

Con respecto a lo anterior se puede observar que tener 3 capas ocultas mejora la precisión, se puede ver reflejado al comparar el Test 1 de las tablas 4 y 6, donde las capas 1 y 2 utilizan la misma cantidad de neuronas en los dos test y la tercera capa agregada con la misma cantidad de neuronas se logra obtener mejores resultados. Debido a esto existen formas que se pueden aplicar para mejorar el rendimiento y junto con eso el resultado de los experimentos, donde podemos mencionar 4 formas en las cuales se pueden llevar a cabo:

Forma 1: Expandir el porcentaje de entrenamiento otorgando más espacio para el entrenamiento.

Forma 2: Agregar capas para mejorar la precisión como se habló anteriormente, buscando un balance entre resultados precisos, buen rendimiento y estabilidad tanto del programa, como de la máquina.

Forma 3: Al agregar más capas estas tienen que estar preferiblemente en orden descendente de neuronas, por ejemplo la capa 1 puede tener 10 neuronas y la capa dos preferiblemente tener menos o las mismas, asimismo, con la capa 3.

Forma 4: Ampliar la base de datos a trabajar, mientras más grande sea la db, más espacio va a tener el programa para poder entrenarse y poder analizar, Los resultados serían más precisos. Además, el porcentaje de entrenamiento podría ser el mismo.

Bibliografía

N. (2019, 16 mayo). *Regresión Logística en Python*. Aprende Machine Learning. https://www.aprendemachinelearning.com/regresion-logistica-con-python-paso-a-paso/

sklearn.linear_model.LogisticRegression. (2021, 25 octubre). Scikit-Learn. https://scikit--learn-org.translate.goog/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic+regression&_x_tr_sl=auto&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=nui#sklearn.linear_model.LogisticRegression

Heras, J. M. (2020, 2 septiembre). *¿Cómo usar Regresión Logística en Python?* IArtificial.net. https://www.iartificial.net/como-usar-regresion-logistica-en-python/

sau. (4 de 5 de 2016). it-swarm-es. Obtenido de it-swarm-es: https://www.it-swarm-es.com/es/deep-learning/como-aumentar-la-precision-de-validacion-con-una-red-neuronal-profunda/824747610/

Anexos

Anexo C, Resultados Punto 2:

utilizamos 6 neuronas

```
Epoch 1/100
750/750 [==============================] - 2s 2ms/step - loss: 0.5920 - accuracy: 0.7736

Epoch 3/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4543 - accuracy: 0.8065

                                                    -                         -
Epoch 5/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4389 - accuracy: 0.8059

                                                    -                         -
Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4367 - accuracy: 0.8087

Epoch 15/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4367 - accuracy: 0.8095
```

```
Epoch 20/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4367 - accuracy: 0.8085

Epoch 25/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4368 - accuracy: 0.8091

Epoch 30/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4367 - accuracy: 0.8080

Epoch 35/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4367 - accuracy: 0.8075
```

Anexo C, Resultados Punto 3:

utilizamos 6 neuronas

```
Epoch 1/100
750/750 [==============================] - 1s 1ms/step - loss: 0.5113 - accuracy: 0.7949

Epoch 3/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4268 - accuracy: 0.7952

Epoch 5/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4215 - accuracy: 0.8171

Epoch 10/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4137 - accuracy: 0.8323

Epoch 15/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4092 - accuracy: 0.8352

Epoch 20/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4066 - accuracy: 0.8368

Epoch 25/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4051 - accuracy: 0.8361
Epoch 30/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4037 - accuracy: 0.8363

Epoch 35/100
750/750 [==============================] - 1s 1ms/step - loss: 0.4030 - accuracy: 0.8369
```

utilizamos dos neuronas :

```
Epoch 1/100
750/750 [==============================] - 2s 1ms/step - loss: 0.5317 - accuracy: 0.7952
```

```
Epoch 3/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4336 - accuracy: 0.7952

Epoch 5/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4286 - accuracy: 0.7952

Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4196 - accuracy: 0.8287

Epoch 15/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4154 - accuracy: 0.8317

Epoch 20/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4130 - accuracy: 0.8343

Epoch 25/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4119 - accuracy: 0.8352

Epoch 30/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4107 - accuracy: 0.8335

Epoch 35/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4100 - accuracy: 0.8367
```

consideramos 10 neuronas:

```
Epoch 1/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4886 - accuracy: 0.7952

Epoch 3/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4261 - accuracy: 0.7952

Epoch 5/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4209 - accuracy: 0.8232

Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4134 - accuracy: 0.8329

Epoch 15/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4085 - accuracy: 0.8352

Epoch 20/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4057 - accuracy: 0.8368

Epoch 25/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4038 - accuracy: 0.8357

Epoch 30/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4024 - accuracy: 0.8357

Epoch 35/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4016 - accuracy: 0.8369
```

pregunta 4

Capa1:5 capa 2:5

```
Epoch 5/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4214 - accuracy: 0.8259

Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4140 - accuracy: 0.8347

Epoch 15/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4097 - accuracy: 0.8349
```

capa1:5 capa 2:10

```
Epoch 5/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4203 - accuracy: 0.8251

Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4110 - accuracy: 0.8337

Epoch 15/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4070 - accuracy: 0.8337
```

capa1:10 capa 2: 5

```
Epoch 5/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4208 - accuracy: 0.8267

Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4130 - accuracy: 0.8331

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4082 - accuracy: 0.8359
```

capa1:10 capa 2: 10

```
Epoch 5/100
750/750 [==============================] - 2s 3ms/step - loss: 0.4198 - accuracy: 0.8279

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4105 - accuracy: 0.8345

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4048 - accuracy: 0.8357
```

capa1:10 capa 2: 15

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4077 - accuracy: 0.8347

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3984 - accuracy: 0.8372

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3959 - accuracy: 0.8383
```

capa1:15 capa 2: 10

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4112 - accuracy: 0.8347

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3984 - accuracy: 0.8380

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3905 - accuracy: 0.8404
```

capa1:15 capa 2: 15

```
Epoch 5/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4065 - accuracy: 0.8349

Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.3981 - accuracy: 0.8401

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3946 - accuracy: 0.8383
```

capa1:5 capa 2: 15

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4215 - accuracy: 0.8284

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4108 - accuracy: 0.8345

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4034 - accuracy: 0.8373
```

capa1:15 capa 2: 5

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4154 - accuracy: 0.8319

Epoch 10/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4026 - accuracy: 0.8369

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3990 - accuracy: 0.8372
```

pregunta 5

capa 1:5 capa 2: 5 capa 3: 5

```
Epoch 5/100
750/750 [==============================] - 1s 2ms/step - loss: 0.3969 - accuracy: 0.8299

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3735 - accuracy: 0.8481

Epoch 15/100
750/750 [==============================] - 1s 2ms/step - loss: 0.3665 - accuracy: 0.8528
```

capa 1:5 capa 2: 5 capa 3: 15

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4197 - accuracy: 0.8273
```

```
Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4120 - accuracy: 0.8340
Epoch 15/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4072 - accuracy: 0.8348
```

capa 1:5 capa 2: 10 capa 3: 10

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3851 - accuracy: 0.8327

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3593 - accuracy: 0.8528

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3503 - accuracy: 0.8565
```

capa 1:5 capa 2: 10 capa 3: 15

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4220 - accuracy: 0.8251

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4140 - accuracy: 0.8328

Epoch 15/100
750/750 [==============================] - 2s 3ms/step - loss: 0.4090 - accuracy: 0.8353
```

capa 1:10 capa 2: 10 capa 3: 5

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3764 - accuracy: 0.8444
Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3540 - accuracy: 0.8572

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3473 - accuracy: 0.8625
```

capa 1:10 capa 2: 10 capa 3: 10

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4181 - accuracy: 0.8316

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4115 - accuracy: 0.8336
Epoch 15/100
750/750 [==============================] - 1s 2ms/step - loss: 0.4077 - accuracy: 0.8344
```

capa 1:10 capa 2: 10 capa 3: 15

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4183 - accuracy: 0.8297
Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4068 - accuracy: 0.8348
```

```
Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.4013 - accuracy: 0.8357
```

capa 1:15 capa 2: 15 capa 3: 5

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3696 - accuracy: 0.8508

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3490 - accuracy: 0.8571

Epoch 15/100
750/750 [==============================] - 2s 3ms/step - loss: 0.3428 - accuracy: 0.8591
```

capa 1:15 capa 2: 15 capa 3: 10

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3885 - accuracy: 0.8395

Epoch 10/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3600 - accuracy: 0.8553

Epoch 15/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3535 - accuracy: 0.8599
```

capa 1:15 capa 2: 15 capa 3: 15

```
Epoch 5/100
750/750 [==============================] - 2s 2ms/step - loss: 0.3646 - accuracy: 0.8485

Epoch 10/100
750/750 [==============================] - 2s 3ms/step - loss: 0.3478 - accuracy: 0.8571

Epoch 15/100
750/750 [==============================] - 2s 3ms/step - loss: 0.3443 - accuracy: 0.8607
```