

Apéndice G

Manual de usuario para clasificador de patrones de flujo

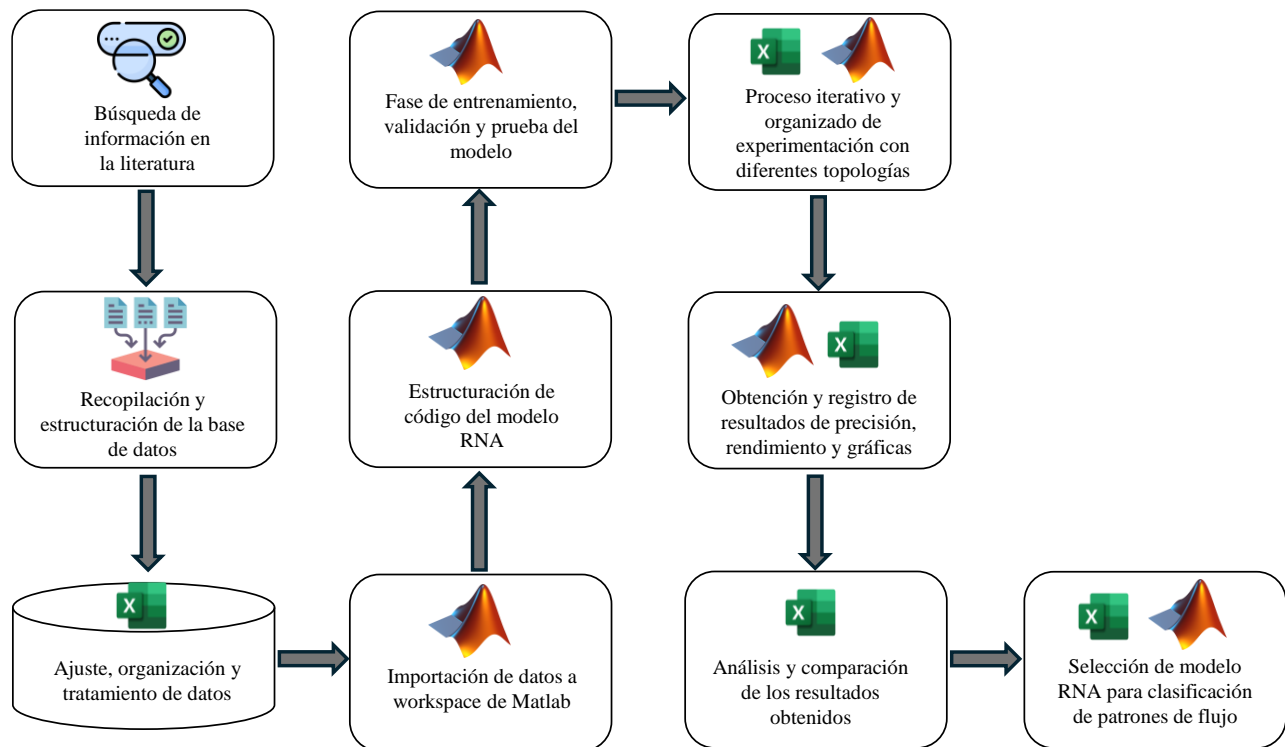
Manual de usuario: Clasificador de patrones de flujo bifásico aceite y agua en tubería horizontal.

Desarrollo del backend

A continuación, se muestra el código desarrollado como resultado de las diferentes estructuras implementadas a lo largo de la realización de este trabajo, cuyo objetivo era establecer un modelo de Red Neuronal Artificial (RNA) cuya configuración interna fue definida por el autor después de realizar los procesos de entrenamiento, validación y prueba, utilizando una base de datos estructurada a partir de la revisión y extracción de datos experimentales de flujo bifásico de aceite y agua en tubería horizontal obtenidos de diferentes autores de la literatura.

El backend generado se ha diseñado de manera autoexplicativa y adaptable a las necesidades específicas de los usuarios. Este incluye secciones con descripciones detalladas de las funciones y los procesos realizados. Asimismo, se ofrecen sugerencias para que el usuario pueda usar otras funciones de entrenamiento, métricas de rendimiento, algoritmos de aprendizaje, gráficas y la opción de modificar el número de capas ocultas y de neuronas en cada capa oculta.

Complementando esta información se presenta un diagrama de flujo mostrando de manera general el paso a paso realizado para seleccionar el modelo de RNA para clasificación de patrones de flujo bifásico de aceite y agua en tubería horizontal.



El backend desarrollado se muestra a continuación:

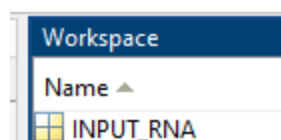
Los parámetros usados como entradas deben estar organizados en vectores fila, una para cada variable, y las salidas deben estar organizadas en vectores columna, cada una representando una categoría.

De esta forma, las matrices de entrada y salida son:

X = INPUT_RNA; Con INPUT_RNA siendo la matriz con las variables de entrada.

Vso [m/s]	0,0000000	0,0000000	0,0000994	0,0002016	0,0002016	0,0002538	0,0003604	0,0004149	0,0004149	0,0006408
Vsw [m/s]	0,0013239	0,1059950	0,0084530	0,0148286	0,0410636	0,0780868	0,0520026	0,0410636	0,0795997	0,0039677
Cw [-]	0,6056677	0,9828434	0,8028928	0,8627377	0,9440518	0,9719142	0,9540505	0,9405383	0,9710550	0,6781702
D [m]	0,4484127	0,4484127	0,4484127	0,4484127	0,4484127	0,4484127	0,4484127	0,4484127	0,4484127	0,4484127
po [kg/m^3]	0,7636364	0,7636364	0,7636364	0,7636364	0,7636364	0,7636364	0,7636364	0,7636364	0,7636364	0,7636364
μo [Pa.s]	0,0048088	0,0048088	0,0048088	0,0048088	0,0048088	0,0048088	0,0048088	0,0048088	0,0048088	0,0048088

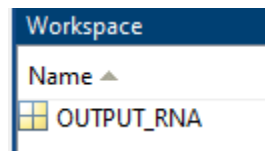
Al momento de importarse en el Workspace, la matriz numérica con las variables de entrada se visualiza así:



t = OUTPUT_RNA; Siendo la matriz con las variables de salida, en su respectiva codificación binaria para cada tipo de patrón de flujo.

ST	ST	ST	ST	ST	ST	ST	ST	ST	ST	ST
1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Al momento de importarse en el Workspace, la matriz numérica con las variables de salida se visualiza así:



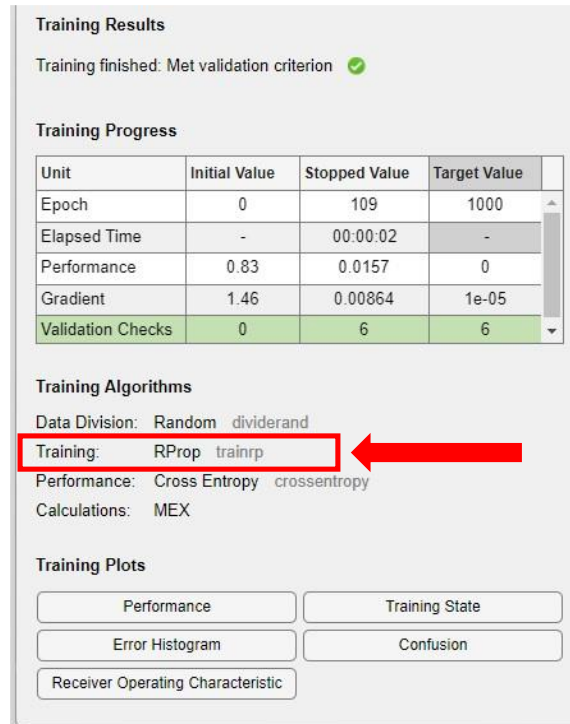
Se procede con la selección de la función de entrenamiento **trainFCN**, considerando las diferentes opciones presentadas junto con su respectivo algoritmo de entrenamiento, como se muestra en la Tabla G1.

Tabla G1. Funciones de entrenamiento a seleccionar

Siglas	Funciones de entrenamiento	Algoritmos de entrenamiento
SCG	TRAINSCG	Scaled conjugate gradient backpropagation
BFG	TRAINBFG	BFGS quasi-Newton backpropagation
RP	TRAINRP	Resilient Backpropagation
CGP	TRAINCGP	Conjugate gradient backpropagation with Polak-Ribiere updates
CGB	TRAINCGB	Conjugate gradient backpropagation with Powell-Beale restarts.

Después de un análisis exhaustivo a partir de un proceso organizado e iterativo de experimentación en el cual se probaron todas las funciones de entrenamiento mencionadas con diferentes topologías, se sugiere el algoritmo Resilient Backpropagation (RP), debido a sus altos porcentajes de precisión, óptimos resultados de rendimiento, y sus bajos requerimientos de memoria. La elección de dicho algoritmo garantiza una optimización efectiva y un mejor rendimiento para el modelo de Red Neuronal Artificial desarrollado en la investigación.

trainFcn = 'trainrp';



Para la estructuración de la red neuronal deseada se debe establecer la cantidad de capas ocultas y la cantidad de neuronas que tendrá cada capa oculta.

hiddenLayerSize = 50; Se define el número de neuronas en cada capa oculta.

net = pattern(hiddenLayerSize, trainFcn); Se crea la red neuronal con el número de neuronas especificado.

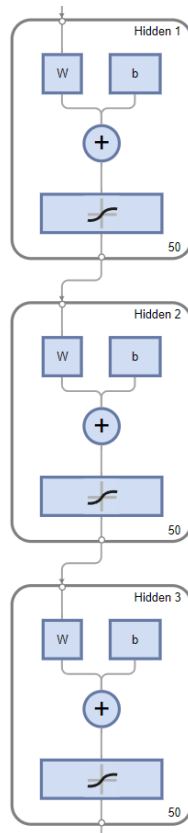
Si el usuario desea implementar más capas ocultas, debe agregar más líneas en el backend, definiendo la cantidad de capas ocultas y numerándolas como se muestra en el siguiente ejemplo, en donde hay tres capas ocultas y cada una de ellas está formada por 50 neuronas.

hiddenLayerSize1 = 50; Capa oculta 1

hiddenLayerSize2 = 50; Capa oculta 2

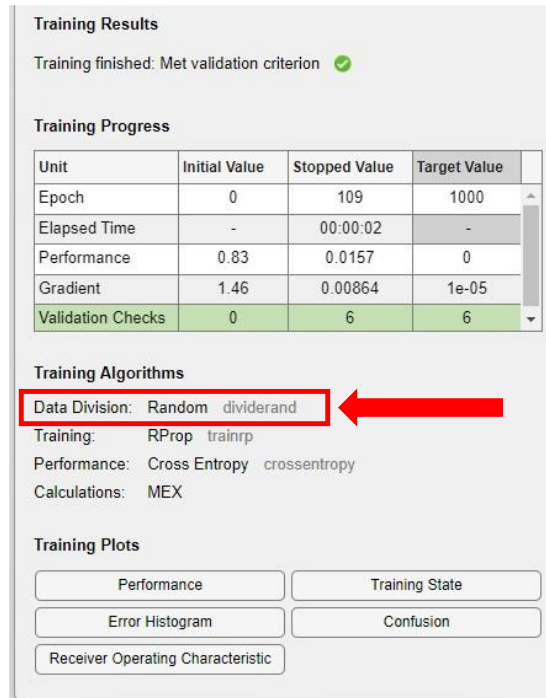
hiddenLayerSize3 = 50; Capa oculta 3

net = patternnet([hiddenLayer1Size, hiddenLayer2Size, hiddenLayer3Size],trainFcn); Red Neuronal Artificial con tres capas ocultas y cada capa oculta con 50 neuronas.



El siguiente paso es configurar la fracción de datos destinados para la etapa de entrenamiento, validación y prueba. La herramienta Neural Network Pattern Recognition, por defecto tiene implementada la función 'dividerand' para la división de los datos, esta división se realiza de forma aleatoria:

`%net.divideFcn = 'dividerand';` Se divide el conjunto de datos al azar.



Para determinar el porcentaje exacto de datos que se utilizarán en cada fase de entrenamiento, validación y prueba, el usuario debe establecer porcentualmente los valores que desea utilizar hasta garantizar que la suma de los tres sea de 100% de la información. Se sugiere que la fracción dedicada al entrenamiento sea mayor o igual al 70% de la información total disponible.

net.divideParam.trainRatio = 70 / 100; 70% de los datos totales para la fase de entrenamiento.

net.divideParam.valRatio = 15 / 100; 15% de los datos totales no utilizados para validar que la red está generalizando y detener el entrenamiento antes de sobreajustar.

net.divideParam.testRatio = 15 / 100; %15% de los datos totales no utilizados para probar de forma independiente la generalización de la red

A continuación, el usuario debe definir el parámetro estadístico que desea incluir en la estructuración de la red neuronal para evaluar el rendimiento del modelo a partir del error calculado en la fase de validación. Los parámetros disponibles para la evaluación del rendimiento de los modelos de aprendizaje automático están contenidos en la Tabla G2.

Tabla G2. Parámetros estadísticos

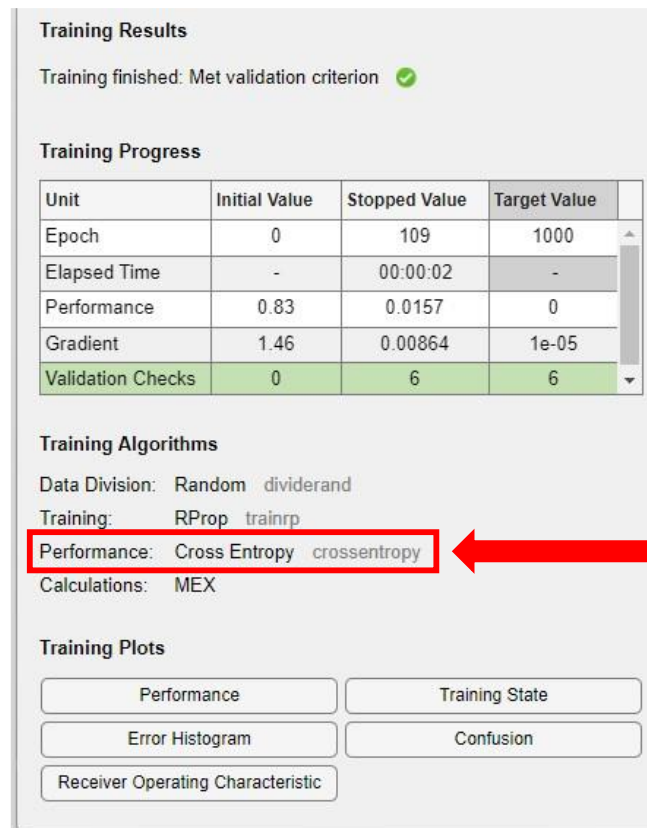
Parámetro estadístico	Nombre extendido
mae	Mean absolute error performance function
mse	Mean squared error performance function
sae	Sum absolute error performance function
sse	Sum squared error performance function

crossentropy	Cross-entropy performance
msespase	Mean squared error performance function with L2 weight and sparsity regularizers

Para obtener una lista de todos los parámetros de rendimiento, el usuario puede consultar ayuda escribiendo en el command window: **help nnperformance**.

Para los problemas de clasificación, la función de rendimiento establecida por defecto en la herramienta Neural Network Pattern Recognition es Cross-Entropy (Entropía cruzada).

%% net.performFcn = 'Crossentropy'; Por defecto se selecciona crossentropy como función de rendimiento para los problemas de clasificación.



Training Results

Training finished: Met validation criterion ✔

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	109	1000
Elapsed Time	-	00:00:02	-
Performance	0.83	0.0157	0
Gradient	1.46	0.00864	1e-05
Validation Checks	0	6	6

Training Algorithms

Data Division: Random dividerand

Training: RProp trainrp

Performance: Cross Entropy crossentropy ←

Calculations: MEX

Training Plots

Performance Training State

Error Histogram Confusion

Receiver Operating Characteristic

Continuando la secuencia de uso, el usuario debe activar el entrenamiento de la red neuronal artificial utilizando la siguiente función:

[net,tr] = train(net,x,t); Donde se usa la red creada, los datos de entrada establecidos y los datos de salida (patrones de flujo).

Asimismo, se debe llevar a cabo la prueba de la Red Neuronal Artificial.

y = net(x); Se obtienen las salidas de la red neuronal para los datos de entrada.

e = gsubtract(t, y); Se calcula el error de predicción comparando los valores de salidas reales y los de las salidas predichas por la red.


performance = perform(net, t, y); Se calcula el rendimiento de la red neuronal definido anteriormente por la entropía cruzada (Crossentropy)

También se calcula la fracción de observaciones mal clasificadas

tind = vec2ind(t);

yind = vec2ind(y);

percentErrors = sum(tind ~= yind)/numel(tind);

Training Results
Training finished: Met validation criterion 

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	109	1000
Elapsed Time	-	00:00:02	-
Performance	0.83	0.0157	0
Gradient	1.46	0.00864	1e-05
Validation Checks	0	6	6

Training Algorithms
Data Division: Random dividerand
Training: RProp trainrp
Performance: Cross Entropy crossentropy
Calculations: MEX

Training Plots

Performance

Training State

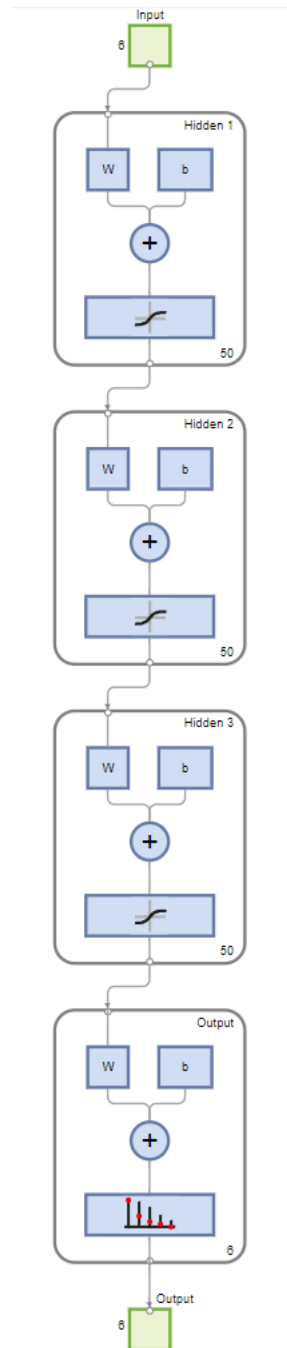
Error Histogram

Confusion

Receiver Operating Characteristic

Luego, se genera un esquema correspondiente a la estructura general de la RNA donde se presentan el número de entradas a la red, el número de capas ocultas, el número de neuronas que contiene cada capa oculta, y la capa de salida con sus respectivas neuronas. Para generar el esquema se utiliza la función:

view(net); Visualización de la estructura de la red neuronal.



Finalmente, se permite al usuario generar graficas complementarias a partir de los resultados obtenidos. De forma interactiva se puede seleccionar la gráfica deseada en el panel de resultados del entrenamiento:

Training Results
 Training finished: Met validation criterion ✔

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	109	1000
Elapsed Time	-	00:00:02	-
Performance	0.83	0.0157	0
Gradient	1.46	0.00864	1e-05
Validation Checks	0	6	6

Training Algorithms
 Data Division: Random dividerand
 Training: RProp trainrp
 Performance: Cross Entropy crossentropy
 Calculations: MEX

Training Plots

Performance

Training State

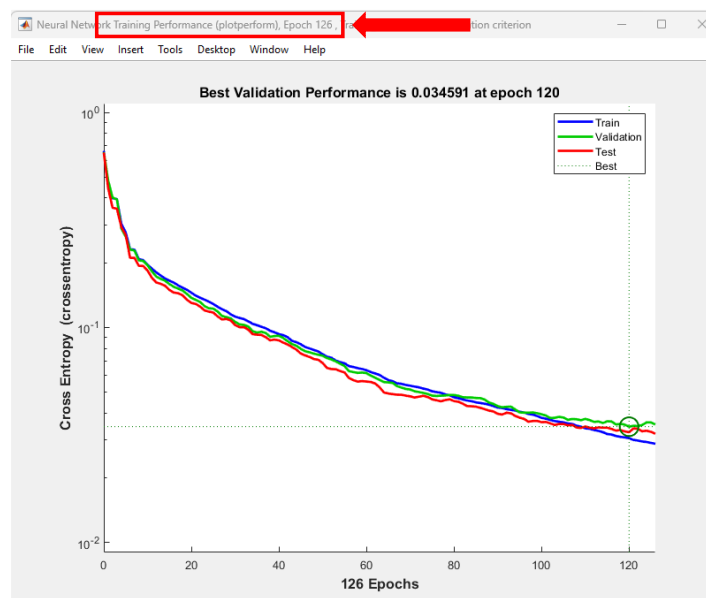
Error Histogram

Confusion

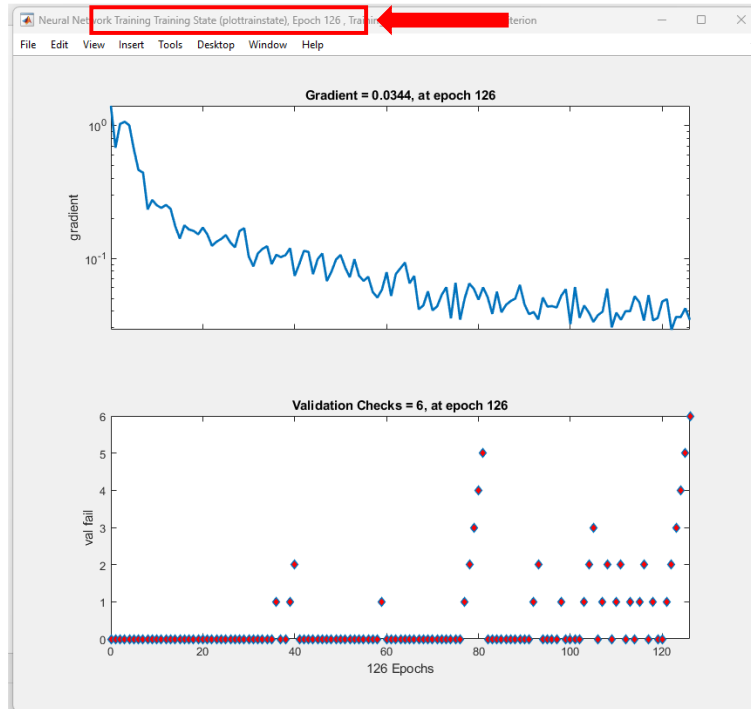
Receiver Operating Characteristic

En cuanto al código, se presentan las funciones respectivas que generaran de manera automática las gráficas de interés:

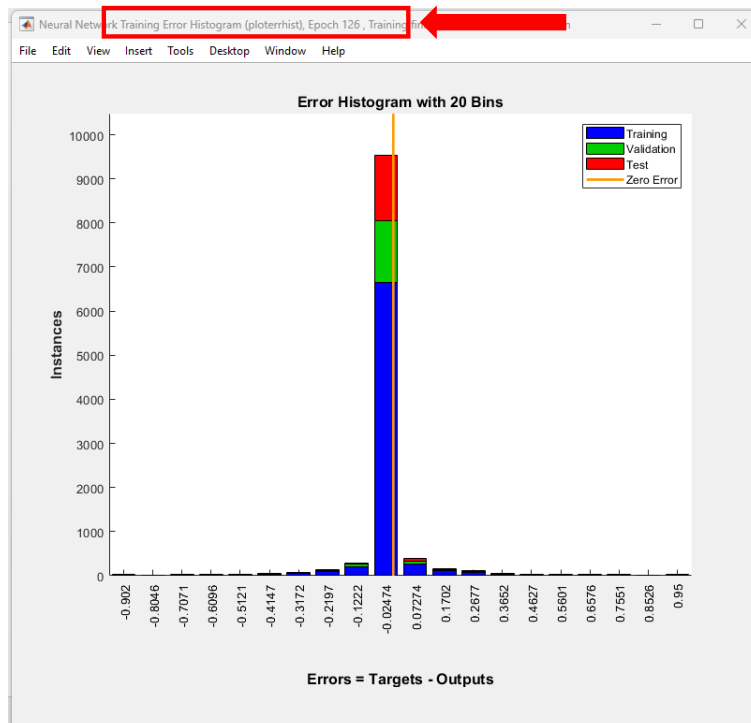
figure, plotperform(tr); Gráfica del rendimiento de la red durante el entrenamiento.



figure, plottrainstate(tr); Gráfica del estado del entrenamiento.



figure, plotterrhist(e); Histograma de los errores de predicción.



figure, plotconfusion(t,y): Matriz de confusión de la clasificación.



figure, plotroc(t,y) ; Gráfica ROC (Receiver operating characteristic) de la clasificación

