Optimización del Subproceso de trefilado a través de Redes Neuronales en Matlab y TIA Portal.

Optimization of the Wire Drawing Subprocess through Neural Networks.

Jerick Robert Suarez Soza
Universidad Estatal Península de Santa Elena. Facultad de Sistemas y Telecomunicaciones. La Libertad. Ecuador.
ORCID: https://orcid.org/0009-0008-5319-3180
Correo: jerickah16@gmail.com

I. Resumen

Las redes neuronales son un tipo de inteligencia artificial que se inspira en el cerebro humano. Están formadas por capas de unidades interconectadas, llamadas neuronas artificiales, que pueden procesar información y aprender de los datos.

En el subproceso de trefilado de la fabricación de cables eléctricos, las redes neuronales se pueden utilizar para controlar el proceso de estirado del alambre de cobre. Esto se hace mediante el uso de sensores para medir la tensión y el diámetro del alambre, y luego utilizando estos datos para ajustar la velocidad del proceso de trefilado. La red neuronal se implementó en MATLAB para el control del proceso de trefilado con su debido entrenamiento. Las redes neuronales también se pueden utilizar para detectar defectos en el alambre de cobre. Esto se hace mediante el uso de cámaras para inspeccionar el alambre y luego utilizando algoritmos de aprendizaje automático para identificar cualquier defecto.

El uso de redes neuronales en el subproceso de trefilado de la fabricación de cables eléctricos puede ayudar a mejorar la calidad del producto final y a reducir los costes de producción.

Además, las redes neuronales también se pueden utilizar para optimizar el proceso de trefilado, reduciendo el consumo de energía y aumentando la eficiencia de la producción.

Palabras clave: Trefilado; optimizar; proceso; Matlab

Abstract

Neural networks are a type of artificial intelligence that is inspired by the human brain. They are made up of layers of interconnected units, called artificial neurons, that can process information and learn from data.

In the drawing subprocess of electrical cable manufacturing, neural networks can be used to control the drawing process of copper wire. This is done by using sensors to measure the tension and diameter of the wire, and then using this data to adjust the speed of the drawing process.

The neural network was implemented in MATLAB to control the drawing process with its proper training.

Neural networks can also be used to detect defects in copper wire. This is done by using cameras to inspect the wire and then using machine learning algorithms to identify any defects.

The use of neural networks in the drawing subprocess of electrical cable manufacturing can help improve the quality of the final product and reduce production costs.

In addition, neural networks can also be used to optimize the wire drawing process, reducing

E-mail: Ecosur@gopsapp.com

energy consumption and increasing production efficiency.

Keywords: Wire drawing; optimize; process; Matlab

II. Introducción

Las redes neuronales son técnicas no paramétricas muy utilizadas en diversos ámbitos de la ciencia e ingeniería porque permiten resolver problemas complejos, que muchas veces no son fáciles de resolver utilizando técnicas tradicionales como la regresión lineal o polinómica. [1]
Las redes neuronales artificiales tratan de emular las características y propiedades de las redes neuronales biológicas. En general, consisten en una serie de unidades denominadas neuronas, conectadas entre sí. Cada neurona recibe un valor de entrada, el cual transforma según una función específica denominada función de activación. Dicha señal transformada pasa a ser la salida de la neurona. [2] Las neuronas se conectan entre sí según una determinada arquitectura. Cada conexión tiene un determinado peso que pondera cada entrada a la neurona. De esta manera la entrada de cada neurona es la suma de las salidas de las neuronas conectadas a ella, multiplicadas por el peso de la respectiva conexión. [3]
Una red neuronal es un modelo de computación cuya estructura de capas se asemeja a la estructura interconectada de las neuronas en el cerebro, con capas de nodos conectados. [4]
Para estimar el modelo es necesario disponer de un conjunto de observaciones de las variables. Estas observaciones son usadas como patrones de entrenamiento para que la red aprenda y sea capaz de predecir una salida del modelo, ante nuevas observaciones. Por tanto, las capacidades de la red van a depender en gran medida de esta fase de entrenamiento. [5]
En la fase de entrenamiento es necesario controlar muchos parámetros y distintos algoritmos de optimización, por lo que el usuario de una red neuronal debe tener conocimiento suficiente de cuáles son estos parámetros y cómo funcionan. Por otro lado, una vez entrenada la red, es muy importante también evaluar la robustez del modelo creado, comprobando que es adecuado para nuevos datos. [6]
Una red neuronal convolucional (CNN o ConvNet) es una arquitectura de red para Deep Learning que aprende directamente a partir de datos. [7]
Son particularmente útiles para identificar patrones en imágenes con el fin de reconocer objetos, clases y categorías. Además, pueden ser muy eficaces para clasificar datos de audio, señales y series temporales.
Las redes neuronales convolucionales pueden tener decenas o cientos de capas, y cada una de ellas aprende a detectar diferentes características de una imagen. [8]
Los modelos de redes neuronales se estructuran en una serie de capas que reflejan la manera en que el cerebro procesa la información. Los modelos de redes neuronales de regresión disponibles en Statistics and Machine Learning Toolbox™ son redes neuronales predictivas interconectadas en las que puede ajustar el tamaño de las capas interconectadas y modificar las funciones de activación de estas. [9]

Para entrenar un modelo de red neuronal de regresión, se utilizó la aplicación "Regression Learner". Para mayor flexibilidad, se entrena un modelo de red neuronal de regresión en la interfaz de línea de comandos. Tras el entrenamiento, puede predecir las respuestas con los nuevos datos pasando el modelo y los nuevos datos de los predictores a predict. [10]
Se explicarán las fases en las que se ha divido el proyecto; el trabajo realizado (el objeto de Matlab: Red Neuronal), centrándonos en cómo se han adaptado los algoritmos para su programación; y una aplicación práctica del trabajo, comprobando los resultados y analizando las ventajas e inconvenientes de los distintos métodos. [11]

E-mail: Ecosur@gopsapp.com

III. Materiales y métodos

Esta guía de investigación de carácter descriptivo y experimental. Dentro de esta búsqueda de retención de información empleamos las bases de datos, Tía Portal, Matlab, entre otras. Las palabras claves han sido seleccionadas refiriéndonos a comunicación entre Matlab y el proceso del trefilado la adquisiciones de datos fue desde Excel para que Matlab al momento de hacer el llamado de los datos tenga compatibilidad directa y sea más rápida y efectiva la adquisición de datos. [12]

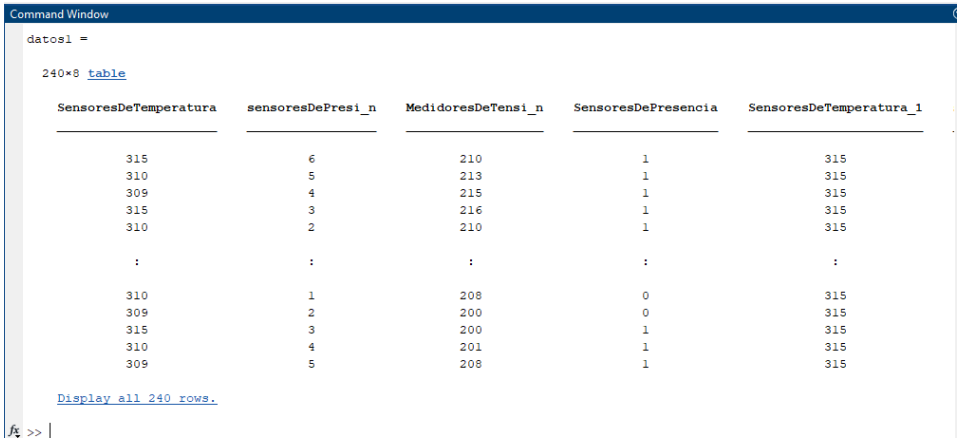
El aprendizaje supervisado es el tipo de algoritmo de Machine Learning más frecuente. Utiliza un conjunto de datos conocidos (denominado conjunto de datos de entrenamiento) para entrenar un algoritmo con un conjunto de datos de entrada conocidos (denominados características) y respuestas conocidas para realizar predicciones. El conjunto de datos de entrenamiento incluye datos de entrada etiquetados que se emparejan con los valores de salida o de respuesta deseados. A partir de él, el algoritmo de aprendizaje supervisado intenta crear un modelo estableciendo relaciones entre las características y los datos de salida para realizar predicciones acerca de los valores de respuesta para un nuevo conjunto de datos. [13]
Estos datos deberán ser tratados antes de que puedan ser usados por la red neuronal. Se estudiarán diferentes formas de suavizado para las curvas de potencia y se experimentará con las variables de entrada a la red neuronal esperando encontrar una solución óptima que permita predecir con la precisión requerida. [14]
Se estudia primero la teoría matemática que hay detrás, partiendo de los métodos clásicos de aproximación de funciones y evolucionando a su adaptación en las redes, mostrando especial atención al algoritmo de Backpropagation. Posteriormente se estudia su aplicación en Matlab, adaptándolos para sacar el máximo beneficio a las capacidades matriciales propias del programa y a la programación orientada a objeto, con que se desarrollan los contenidos de la red neuronal. [15]

A. COMPONENTES

Para esta investigación, se emplean elementos físicos y lógicos. Entre ellos el uso de PLC, interruptores, potenciómetros que hacen de sensores analógicos, además TIA PORTAL para hacer la programación del proceso de trefilado y obtener los datos según el funcionamiento del sistema y el software Matlab.

B. Primeros pasos para crear la red neuronal

Recopilar datos: datos de entrada y salida.


Command Window

```
datos1 =

  240×8 table

    SensoresDeTemperatura    sensoresDePresi_n    MedidoresDeTensi_n    SensoresDePresencia    SensoresDeTemperatura_1
    _____________________    _________________    __________________    ___________________    _______________________

            315                      6                   210                    1                     315
            310                      5                   213                    1                     315
            309                      4                   215                    1                     315
            315                      3                   216                    1                     315
            310                      2                   210                    1                     315

             :                       :                    :                     :                      :

            310                      1                   208                    0                     315
            309                      2                   200                    0                     315
            315                      3                   200                    1                     315
            310                      4                   201                    1                     315
            309                      5                   208                    1                     315

    Display all 240 rows.

fx >>
```

Ilustración 1 Datos ya extraídos en Matlab

E-mail: Ecosur@gopsapp.com

C. CREAR LA RED

Se inicializan las dimensiones de las capas de entrada, salida y ocultas de la red neuronal. Primero, el tamaño de la capa de entrada (capaEntrada) según el número de características en los datos de entrenamiento (X_entrenamiento). Asimismo, se establece el tamaño de la capa de salida (capaSalida) en función de la cantidad de salidas deseadas, en los datos etiquetados (y_entrenamiento). Para la capa oculta (capaOculta), se elige tener 200 neuronas, lo que proporciona cierta complejidad para aprender patrones en los datos.

Este proceso de dimensionamiento es crucial para definir la estructura básica de la red. La capa de entrada se adapta a la naturaleza de los datos, la capa de salida coincide con la cantidad de salidas que se requiere predecir, y la capa oculta se elige con un número de neuronas que equilibra la capacidad de aprendizaje sin volverse demasiado complejo y propenso al sobreajuste.

```
capaEntrada = size(X_entrenamiento, 2);
capaSalida = size(y_entrenamiento, 2);
capaOculta = 200;
```

Ilustración 2 Creación de la red neuronal

D. CONFIGURAR LA RED NEURONAL

Se crea las capas de la red neuronal de acuerdo con la arquitectura diseñada para abordar la tarea específica. Se inicia con la capa de entrada, utilizando featureInputLayer, donde se ajusta su tamaño según el número de características en los datos y se normaliza los valores utilizando z-score para estandarizar el rango de entrada, se asignó el nombre 'entrada' para identificación.

A continuación, se configura una capa completamente conectada (fullyConnectedLayer) llamada 'oculta1' con 'capa Oculta' neuronas. Esta capa es fundamental para aprender patrones complejos en los datos de entrada y capturar relaciones no lineales. Se introduce no linealidades en la red aplicando la función tangente hiperbólica (tanh) con tanhLayer en la capa 'tansig'.

La capa de salida es otra capa completamente conectada llamada 'salida' (fullyConnectedLayer) con un número de neuronas igual a 'capa Salida', que se ajusta a la cantidad de salidas deseadas en el problema. La configuración finaliza con la capa de regresión (regressionLayer), etiquetada como 'mse' (Mean Squared Error), ya que la tarea es un problema de regresión y se buca minimizar el error cuadrático medio entre las predicciones y los valores reales.

```
layers = [
    featureInputLayer(capaEntrada, 'Normalization', 'zscore', 'Name', 'entrada')
    fullyConnectedLayer(capaOculta, 'Name', 'oculta1')
    tanhLayer('Name', 'tansig')
    fullyConnectedLayer(capaSalida, 'Name', 'salida')
    regressionLayer('Name', 'mse')];
```

Ilustración 3 configuración de la red neuronal

E-mail: Ecosur@gopsapp.com

E. CONFIGURACIÓN PARA EL ENTRENAMIENTO DE LA RED NEURONAL

En la configuración de las opciones de entrenamiento de la red neuronal, Se emplea el optimizador 'adam' para ajustar los pesos de la red durante el proceso de aprendizaje. Se personaliza estos ajustes para optimizar el rendimiento de la red en la tarea específica. Se establece un límite de 200 épocas para el entrenamiento, lo cual determina el número de veces que el modelo recorre todo el conjunto de datos de entrenamiento. La tasa de aprendizaje inicial se fija en 0.99, permitiendo un mayor impulso inicial para acelerar la convergencia de la red. Además, se implementa un programa de ajuste de la tasa de aprendizaje mediante la opción 'piecewise', que permite realizar cambios en la tasa de aprendizaje en momentos específicos del entrenamiento.
Si bien inicio con una tasa alta, se reduce en un 20% cada 5 épocas (ajustando 'LearnRateDropFactor' y 'LearnRateDropPeriod') para refinar los pesos de la red a medida que progresa el entrenamiento.

El tamaño del minibatch se establece en 32, determinando la cantidad de ejemplos de entrenamiento utilizados en cada iteración para ajustar los pesos. Se utiliza gráficos ('Plots', 'training-progress') para visualizar el progreso del entrenamiento y evaluar su rendimiento.

```
opciones_entrenamiento = trainingOptions('adam', ...
    'MaxEpochs', 200, ... % Ajustar el número de épocas
    'InitialLearnRate', 0.99, ... % Ajustar la tasa de aprendizaje inicial (mayor presión)
    'LearnRateSchedule', 'piecewise', ... % Programar cambios en la tasa de aprendizaje
    'LearnRateDropFactor', 0.2, ... % Factor de reducción de la tasa de aprendizaje
    'LearnRateDropPeriod', 5, ... % Período de reducción de la tasa de aprendizaje
    'MiniBatchSize', 32, ... % Tamaño del minibatch (ajustar según tu conjunto de datos)
    'Plots', 'training-progress', ... % Mostrar gráficos de progreso
    'ValidationFrequency', 10); % Frecuencia para mostrar información de validación
```

Ilustración 4 CONFIGURACIÓN DE ENTRENAMIENTO DE LA RED NEURONAL

F. VALIDAR LA RED NEURONAL

Se implementa una validación cruzada de 5-fold para evaluar y entrenar de la red. Primero, se crea una partición de datos utilizando 'cvpartition' para dividir mi conjunto de entrenamiento en 5 folds. Esto es crucial para evaluar la capacidad de generalización del modelo en diferentes subconjuntos de datos.
Dentro del bucle for, se itera a través de cada fold y se selecciona los índices correspondientes a los conjuntos de entrenamiento (train_idx) y de validación (val_idx). Luego, se extrae los datos de entrada (X_train y X_val) y las etiquetas (y_train y y_val) según estos índices.
En cada iteración del bucle, se crea un nuevo objeto layerGraph (lgraph) basado en las capas previamente definidas (layers). Este paso es esencial porque cada fold de la validación cruzada representa una instancia única de entrenamiento y validación, y es fundamental tener un nuevo grafo de capas para garantizar que la red neuronal se entrene desde cero en cada iteración, evitando la influencia de entrenamientos anteriores
La validación cruzada de 5-fold me permite evaluar el rendimiento del modelo en diferentes conjuntos de datos y reducir el riesgo de sobreajuste. Durante cada iteración, entreno la red neuronal con los datos de entrenamiento específicos del fold actual y evaluo su rendimiento en los datos de validación.

E-mail: Ecosur@gopsapp.com

```
num_folds = 5;
c = cvpartition(length(y_entrenamiento), 'KFold', num_folds);

for fold = 1:num_folds
    train_idx = training(c, fold);
    val_idx = test(c, fold);

    X_train = X_entrenamiento(train_idx, :);
    y_train = y_entrenamiento(train_idx, :);
    X_val = X_entrenamiento(val_idx, :);
    y_val = y_entrenamiento(val_idx, :);

    % Crear el objeto layerGraph
    lgraph = layerGraph(layers);
```

Ilustración 5 VALIDAR LA RED NEURONAL

IV. Resultados y discusión

Del proceso de trefilado salen los datos de los sensores tanto de presencia, diámetro, temperatura entre otros, los datos que genera el proceso de la máquina de trefilado son adquiridos gracias al uso de PLC S7-Siemens 1200 y exportados a Matlab para su debido tratamiento.

Según la metodología de la red neuronal se usan las entradas que constan en el proceso de trefilado osea la información adquirida por los sensores. En el contexto del trefilado de cables eléctricos, estas características podrían incluir parámetros del proceso de fabricación, como la temperatura, la velocidad, la fuerza de tracción, etc. Esta red cuenta con 4 entradas y una arquitectura como se muestra en la siguiente figura.

E-mail: Ecosur@gopsapp.com

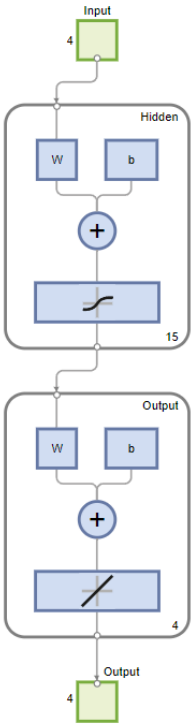


Figura 1 Estructura de la RNA

La figura 2 muestra un histograma de error de una red neuronal. representa la distribución de los errores en la red neuronal. Cada barra del histograma representa la cantidad de errores que caen dentro de un rango específico. Esto puede ayudar a entender la variabilidad y la tendencia central de los errores.

Colores de las Barras: Los diferentes colores en las barras representan diferentes conjuntos de datos. El rojo representa los datos de entrenamiento, el verde los datos de validación, y el azul los datos de prueba. Esto permite comparar cómo se desempeña la red en diferentes conjuntos de datos.

Ejes: El eje Y muestra el número de instancias y el eje X los valores del error. Esto permite ver cuántas instancias caen dentro de cada rango de error, se muestra como la altura de la barra. Por ejemplo, si una barra tiene una altura de 60, significa que hay 60 instancias que tienen un error dentro del rango de esa barra.

Líneas Zero error: Las líneas negras indican "Zero Error", lo que significa que en esos puntos, la red neuronal hizo una predicción perfecta sin error.

Overfitting: si el error en los datos de validación y prueba es significativamente mayor que en los datos de entrenamiento, podría ser una señal de sobreajuste. Esto significa que la red podría estar memorizando los datos de entrenamiento en lugar de aprender a generalizar a partir de ellos.

E-mail: Ecosur@gopsapp.com

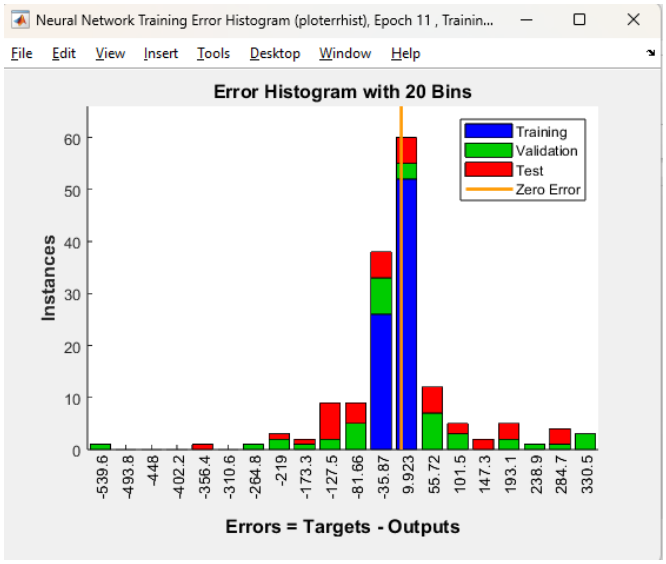


Figura 2 Histograma de error

Rendimiento del modelo de la RNA

La figura 2 muestra un gráfico de error cuadrático medio (MSE) en función de las épocas durante el entrenamiento y la validación de un modelo. Aquí están los detalles:

Ejes: El eje Y representa el error cuadrático medio (MSE) en escala logarítmica, y el eje X muestra las épocas, numeradas del 0 al 9.

Curva de Error: Hay tres curvas que representan el entrenamiento (verde), la validación (rojo) y la prueba (azul).

Entrenamiento: La curva verde muestra que el error disminuye rápidamente durante las primeras épocas y luego se estabiliza.

Validación: La curva roja indica que el error disminuye inicialmente pero comienza a aumentar después de alcanzar su punto más bajo, lo cual puede ser una señal de sobreajuste.

Prueba: Se indica con una línea punteada vertical en la época 3, donde el MSE es 8578.6442. Esto sugiere que este podría ser un buen punto para detener el entrenamiento para evitar el sobreajuste.

Análisis de Overfitting: El aumento del error de validación después del mínimo sugiere sobreajuste; el modelo está aprendiendo características específicas del conjunto de datos de entrenamiento.

E-mail: Ecosur@gopsapp.com

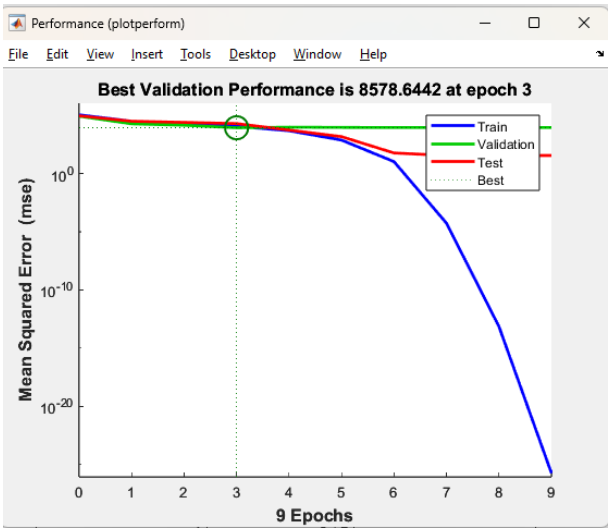


Figura 3 Rendimiento del modelo de la RNA

La figura 4 muestra cuatro gráficos de dispersión que representan los resultados de un modelo de machine learning en las fases de entrenamiento, validación, prueba Los gráficos muestran la relación entre los datos objetivo y los datos ajustados por el modelo.

Validación (R=0.997): En la fase de validación, el coeficiente de correlación es 0.997. Esto indica que el modelo ha aprendido adecuadamente.

Prueba (R=0.9887): En la fase de prueba, el coeficiente de correlación es 0.9887. Esto indica que el modelo ha aprendido adecuadamente.

Curvas en Líneas Diagonales: En las fases de entrenamiento y validación, las líneas están casi superpuestas con Y=T, indicando un buen ajuste del modelo. En "All", hay una separación notable.

Falta de Separación Entre las Clases: No es completamente evidente debido a la naturaleza cuantitativa continua del gráfico ni de los datos que se proporcionaron.

Implicaciones para el Rendimiento del Modelo: Aunque el modelo tiene un excelente desempeño en entrenamiento y validación.

Análisis detallado: El alto valor R cuadrado durante las fases de entrenamiento y validación podría sugerir un posible sobreajuste del modelo ya que se adapta extremadamente bien a estos conjuntos de datos pero muestra una eficacia reducida cuando se consideran todos los datos juntos.

E-mail: Ecosur@gopsapp.com

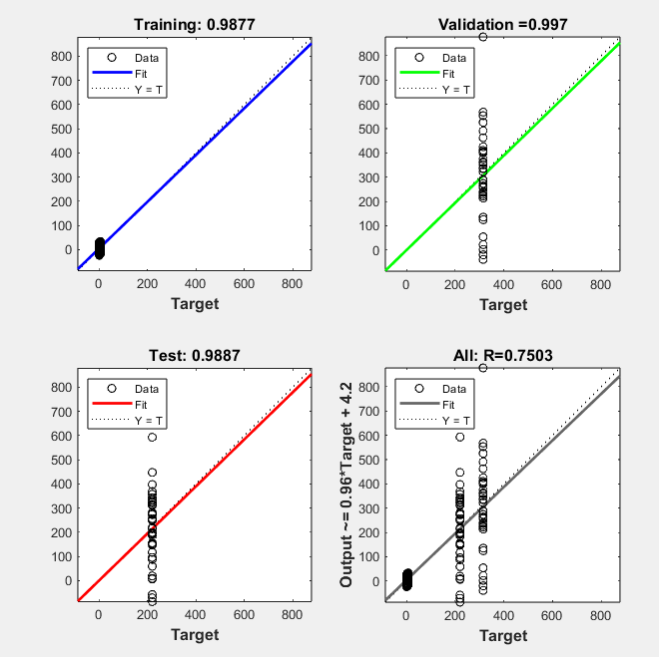


Figura 4 Correlación entre los valores esperados

La figura 5 contiene tres gráficos que están relacionados con un proceso de optimización o aprendizaje automático.

Gráfico del Gradiente: El primer gráfico en la parte superior muestra una curva descendente etiquetada como “gradiente”. Comienza en un valor alto y disminuye exponencialmente. Está anotado con “Gradiente = 4.8627e-11, en la época 11”.

Gráfico de Mu: El segundo gráfico en el medio también muestra una curva descendente etiquetada como “mu”. Similar al primer gráfico, comienza en un valor alto y disminuye exponencialmente. Está anotado con “Mu = 1e-09, en la época 11”.

Gráfico de Validación: El tercer gráfico en la parte inferior está etiquetado como “Validación” y muestra un aumento lineal en las comprobaciones de validación a lo largo del tiempo. Utiliza diamantes rojos para marcar cada punto de datos. Está anotado con “Comprobaciones de Validación = 6, en la época 11”.

Todos los gráficos están trazados contra un eje x que representa las épocas numeradas del 0 al 11. Esto sugiere que los valores del gradiente y mu disminuyen a lo largo del tiempo, mientras que las comprobaciones de validación aumentan, lo que podría indicar la convergencia de un algoritmo de aprendizaje automático.

Basado en los gráficos el algoritmo de aprendizaje automático está convergiendo, ya que los valores del gradiente y mu están disminuyendo, mientras que las comprobaciones de validación están aumentando. Esto generalmente es una señal positiva de que la red está funcionando correctamente.

E-mail: Ecosur@gopsapp.com

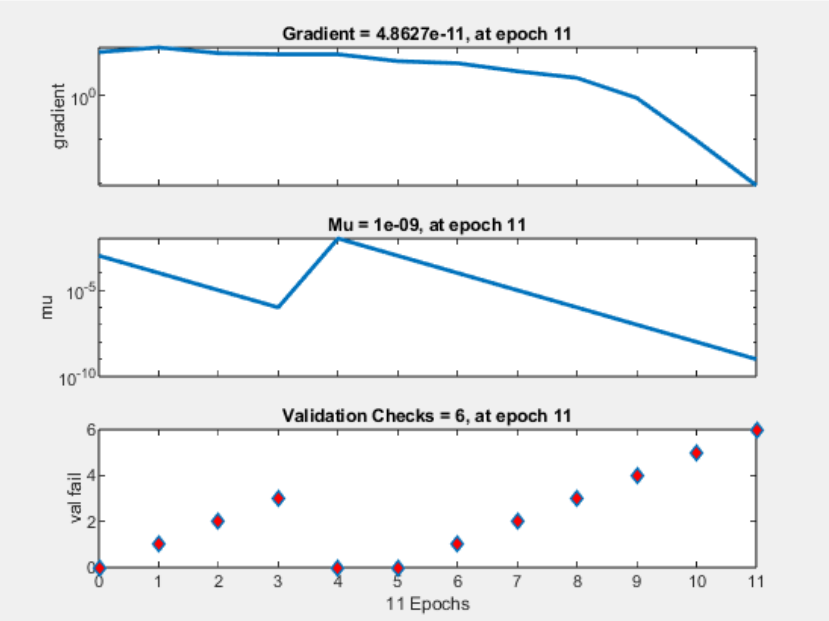


Figura 5 Estado de entrenamiento de la RNA

V. Conclusiones

La integración de redes neuronales en el proceso de trefilado ejemplifica el impacto sustancial de la IA y el aprendizaje automático en aplicaciones industriales. El estudio muestra el potencial de las redes neuronales para interpretar y aprovechar datos complejos, mejorando la toma de decisiones y las capacidades predictivas. A medida que estas tecnologías continúan evolucionando, su poder transformador en diversos campos promete desplegarse aún más, afirmando el papel fundamental de las RNA en el avance de la inteligencia computacional y las aplicaciones prácticas.

Referencias

[1] R. A. MORENO, 1 7 2009. [En línea]. Available: https://e-archivo.uc3m.es/bitstream/handle/10016/8488/Proyecto%20Redes%20neuronales%20GUI.pdf.

[2] «Deep Learning,» [En línea]. Available: https://la.mathworks.com/discovery/neural-network.html.

[3] Mathworks, «REDES NEURONALES».

[4] «Mathworks,» [En línea]. Available: https://la.mathworks.com/discovery/neural-network.htmlm.

[5] «redes neuronales».

[6] «entrenamiento de redes neuonales».

[7] «The MathWorks, Inc.,» [En línea]. Available: https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[8] «mathworks,» [En línea]. Available: https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[9] «Cargar redes neuronales preentrenadas».

E-mail: Ecosur@gopsapp.com

[10] «centro de ayuda,» [En línea]. Available: https://la.mathworks.com/help/stats/neural-networks-for-regression.html?s_tid=CRUX_lftnav.

[11] I. Zumárraga Eguidazu, 17 Junio 2019. [En línea]. Available: https://addi.ehu.es/bitstream/handle/10810/36584/Redes_neuronales_en_matlab.pdf?sequence=1.

[12] «Cargar datos secuenciales,» [En línea]. Available: https://la.mathworks.com/help/deeplearning/ug/sequence-classification-using-1-d-convolutions.html.

[13] «PROYECTO DE REDES NEURONALES GUI,» [En línea]. Available: https://e-archivo.uc3m.es/bitstream/handle/10016/8488/Proyecto%20Redes%20neuronales%20GUI.pdf.

[14] J. San Miguel Salas. [En línea]. Available: https://uvadoc.uva.es/handle/10324/18120.

[15] I. Zumarraga Eguidazu. [En línea]. Available: https://addi.ehu.es/handle/10810/36584.