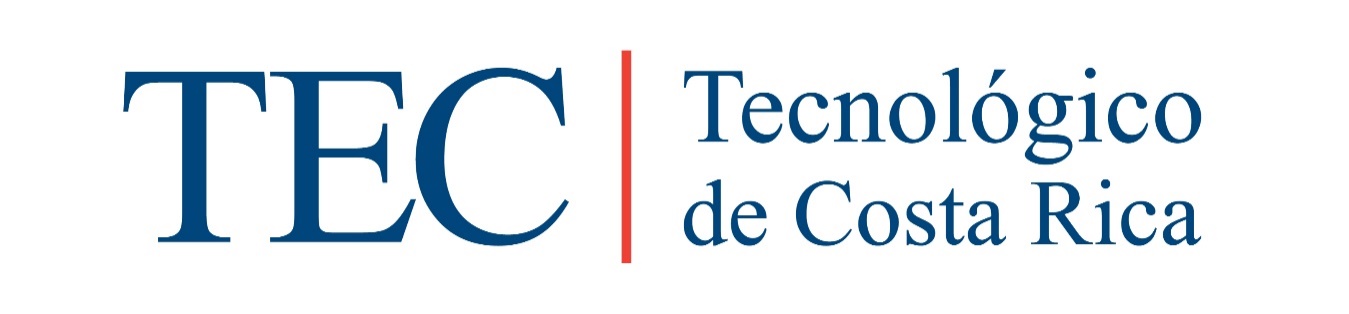
[](https://www.google.co.cr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjL1sn7ksLOAhXDpx4KHb1SB4gQjRwIBw&url=https://iesummerschool.wordpress.com/&psig=AFQjCNGejWKQ95Q2d0-TQxzw_wDuI820tQ&ust=1471306760343223)

Instituto Tecnológico de Costa Rica

Curso: Inteligencia Artificial

Profesor: Juan Luis Crespo Mariño

Tarea 1

Conexionistas

Estudiante:

Marco Rodríguez Morales 2019163031

Sofía Valverde Gutiérrez 2019390095

I Semestre

2023

Fecha de Entrega: 28 de abril del 2023

**Tarea Sistemas conexionistas. Parte 1.**

**Objetivo:** demostrar los conocimientos y habilidades adquiridos en el uso de redes neuronales “clásicas”, en un problema de naturaleza técnica, generando conocimiento que permite tomar decisiones ingenieriles avanzadas

**Problema**

Construir un modelo neuronal de regresión que permita la estimación de la corriente de excitación a partir del resto de datos

Caso

Un motor sincrónico es un motor de CA, cuyo giro se da gracias a la alternación de la corriente en las bobinas/electroimanes, tal que se crea un campo magnético giratorio. El rotor, con su campo magnético CD (y por tanto constante), buscará alinearse con los polos giratorios del estator, produciendo movimiento. Eventualmente, logra sincronizarse, tal que se tiene la velocidad máxima y constante del motor sincronizado con la frecuencia de la corriente (es decir, la velocidad síncrona).

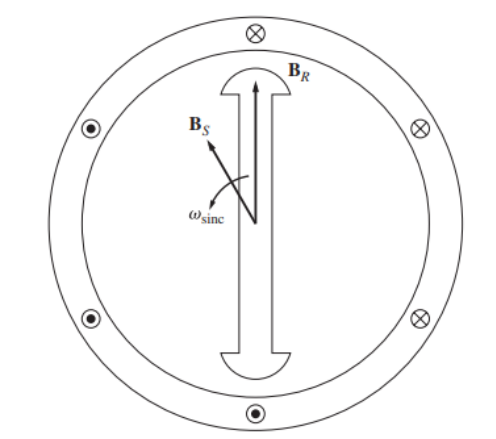


Figura 1Diagrama de un motor sincrónico de dos polos

Fuente: Máquinas Eléctricas, 5a edición (J. Chapman)

Debido a lo simples que son de controlar, los motores sincrónicos se utilizan en la industria como una carga reactiva para controlar el factor de potencia (y, por tanto, la eficiencia) de otros dispositivos de potencia. Debido a que, al conectar un motor a una red ya existente, lo más fácilmente controlable es la corriente de excitación del motor. Por tanto, se requiere un modelo que pueda predecir el amperaje necesario para lograr las condiciones deseadas, tomando como base datos tomados de motores sincrónicos reales y sus corrientes de excitación bajo condiciones específicas de operación.

En esta tarea se va a utilizar el conjunto de datos referenciado en: <https://archive.ics.uci.edu/ml/datasets/Synchronous+Machine+Data+Set>

1. Construcción de un modelo neuronal de regresión que permite la estimación de la corriente de excitación a partir del resto de datos.

Se adjunta la programación del modelo en la sección de presentación de resultados. Dicho modelo fue realizado tomando partes de las programaciones de [1] y de [2].

1. Estudio de hiperparámetros.

Se realizó un estudio de hiperparámetros variando el optimizador, se utilizó Adam, Adamax y Adagram ya que Adam y Adagram son los optimizadores que más se mencionaron en la clase teórica del curso. Por otro lado, se utilizó Adamax ya que es una variante del optimizador Adam .

Luego se varió la taza de aprendizaje utilizando 0.01, 0.06 y 0.001 para obtener el tamaño de paso adecuado.

Se usó un número de capas ocultas de 2 y 4 capas y un número de neuronas de 4 y 6.

Se pueden observar las gráficas resultantes del entrenamiento en las Figuras de la 1 a la 54.

Optimizador: Adam

Taza de aprendizaje: 0.01

Capas ocultas: 2

Número de neuronas: 4

Gráfico

Descripción generada automáticamente

Figura 22 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 33 Error cuadrático medio .

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.01

Capas ocultas: 2

Número de neuronas: 4

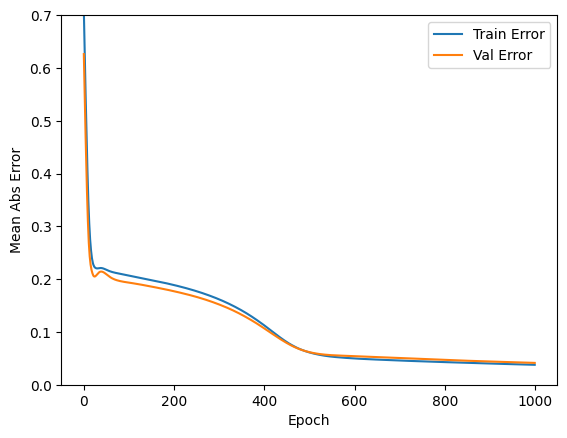


Figura 4 4 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 5 5 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.01

Capas ocultas: 2

Número de neuronas: 4

Gráfico

Descripción generada automáticamente

Figura 6 6Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 7 7 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.06

Capas ocultas: 2

Número de neuronas: 4

Gráfico

Descripción generada automáticamente

Figura 8 8 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 9 9 0ror Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.06

Capas ocultas: 2

Número de neuronas: 4

Gráfico, Histograma

Descripción generada automáticamente

Figura 10 10 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 11 11 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.06

Capas ocultas: 2

Número de neuronas: 4

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 12 12Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 13 13 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.001

Capas ocultas: 2

Número de neuronas: 4

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura14 14 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 15 15 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.001

Capas ocultas: 2

Número de neuronas: 4

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 16 16 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 17 17 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.01

Capas ocultas: 4

Número de neuronas: 4

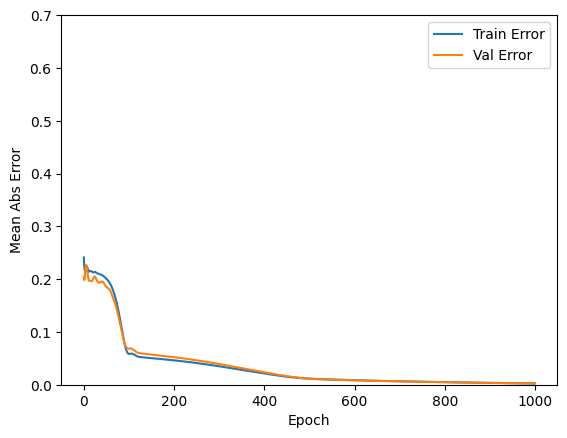


Figura 18 18 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 19 19 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.01

Capas ocultas: 4

Número de neuronas: 4

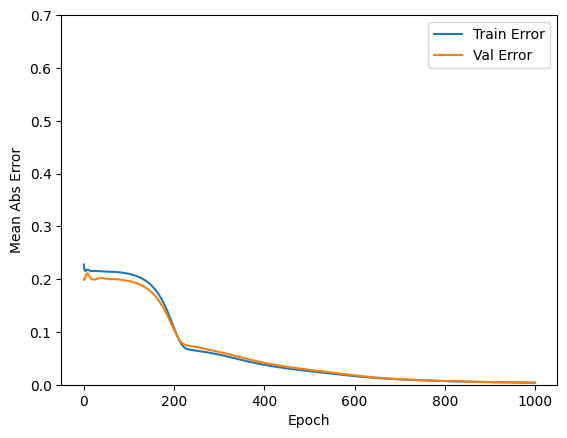


Figura 20 20Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 2121 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.01

Capas ocultas: 4

Número de neuronas: 4

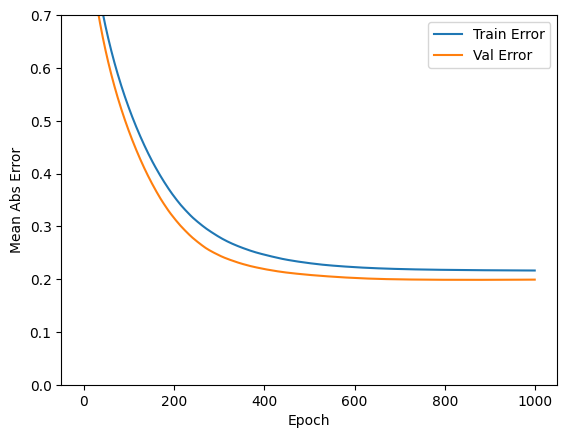


Figura 22 22 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente con confianza baja

Figura 23 23 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.001

Capas ocultas: 2

Número de neuronas: 4

Gráfico

Descripción generada automáticamente

Figura 24 24 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 25 25 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.001

Capas ocultas: 4

Número de neuronas: 4

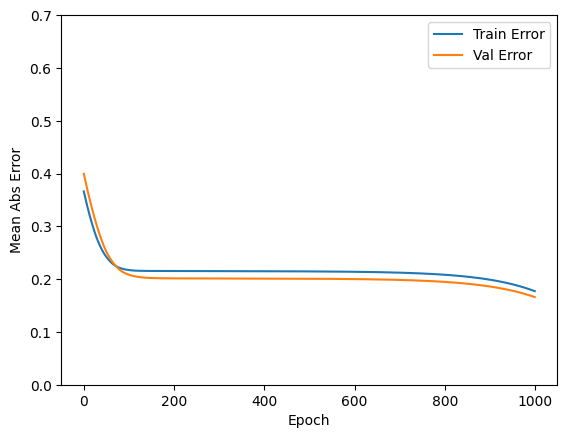


Figura 2626 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 2727 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.001

Capas ocultas: 4

Número de neuronas: 4

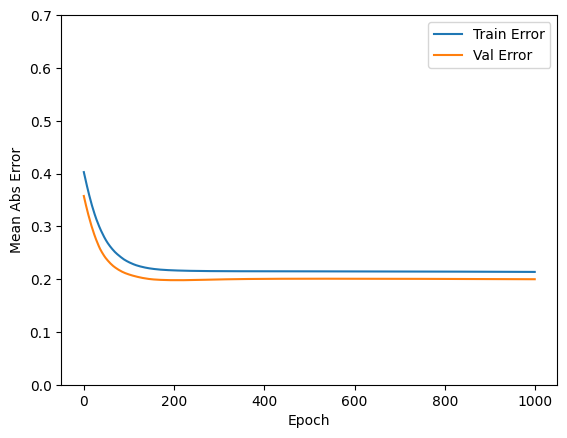


Figura 28. 28Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 29.29 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.001

Capas ocultas: 4

Número de neuronas: 4

Gráfico

Descripción generada automáticamente

Figura 3030.Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 31.31 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.06

Capas ocultas: 4

Número de neuronas: 4

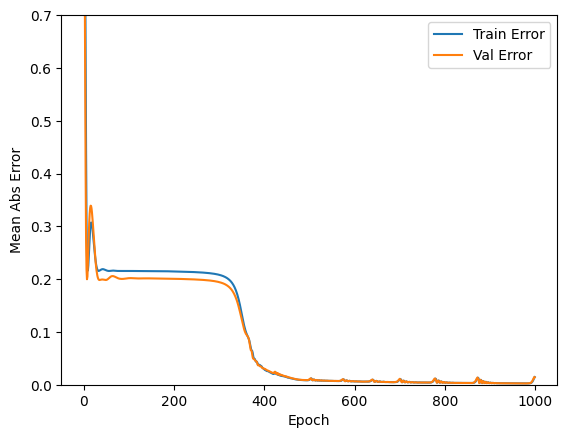


Figura 3231.Error Medio Absoluto

Fuente: Elaboración propia

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 33 32 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.06

Capas ocultas: 4

Número de neuronas: 4

Gráfico

Descripción generada automáticamente

Figura 34 33 .Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 35 34Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.06

Capas ocultas: 4

Número de neuronas: 4

Gráfico

Descripción generada automáticamente

Figura 35.36 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 3736 .Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.01

Capas ocultas: 2

Número de neuronas: 6

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 37 38 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 39 38 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.01

Capas ocultas: 2

Número de neuronas: 6

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 39 40 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 4041 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.01

Capas ocultas: 2

Número de neuronas: 6

Gráfico

Descripción generada automáticamente

Figura 42 41 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 4243 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.06

Capas ocultas: 2

Número de neuronas: 6

Gráfico

Descripción generada automáticamente

Figura 44 43 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 45 44 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.06

Capas ocultas: 2

Número de neuronas: 6

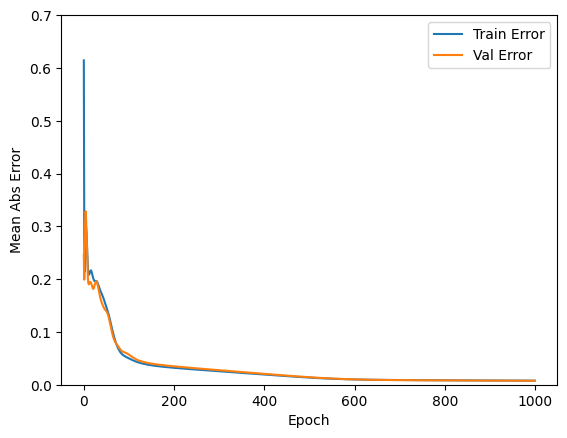


Figura 46 45 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 4647 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.06

Capas ocultas: 2

Número de neuronas: 6

Gráfico

Descripción generada automáticamente

Figura 4748 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 49 48 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adam

Taza de aprendizaje: 0.001

Capas ocultas: 2

Número de neuronas: 6

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 49.50 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 50 51 Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adamax

Taza de aprendizaje: 0.001

Capas ocultas: 2

Número de neuronas: 6

Gráfico

Descripción generada automáticamente

Figura 52 51 6ror Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 53Error Medio Cuadrático

Fuente: Elaboración propia

Optimizador: Adagrad

Taza de aprendizaje: 0.001

Capas ocultas: 2

Número de neuronas: 6

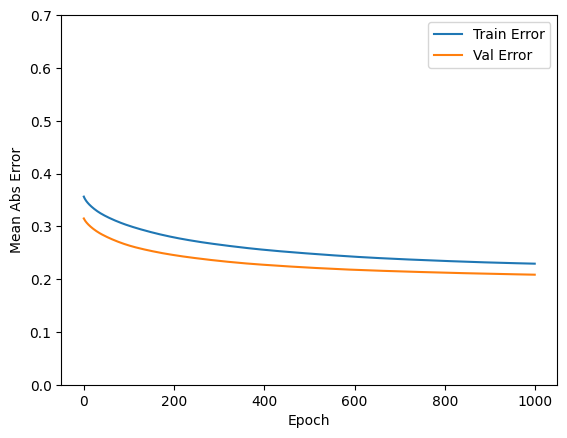


Figura 52 54 Error Medio Absoluto

Fuente: Elaboración propia

Gráfico

Descripción generada automáticamente

Figura 55 53 Error Medio Cuadrático

Fuente: Elaboración propia

Como se puede observar en las gráficas obtenidas, hay varias combinaciones que dan muy buenos resultados, sin embargo, se pueden resaltas algunos parámetros que dan paso a resultados que sobresalen del resto.

Primero, las gráficas que presentan el mejor aprendizaje de la red son la que corresponden a las que tienen el algoritmo Adam como optimizador. Por otro lado, las iteraciones realizadas con una taza de aprendizaje del 0.06 presentan un mejor aprendizaje en menos iteraciones que el resto. Con estos dos parámetros definidos se reduce la lista de decisión a las Figuras 7 y 8, 31 y 32, 43 y 44. Tomando en cuenta el error absoluto medio, ya que es el que presenta mayor variación, la combinación óptima se presenta utilizando el optimizador Adam, una taza de aprendizaje del 0.06, 2 capas ocultas y 4 neuronas por capa, datos correspondientes a las Figuras 7 y 8.

Cabe resaltar que el modelo se entrenó por 1000 iteraciones, usando un tamaño de lote de 512

Para lo que resta de esta parte A se utilizará el modelo entrenado con los hiperparámetros escogidos como los mejores (optimizador Adam, taza de aprendizaje del 0.06, 2 capas ocultas y 4 neuronas por capa).

**Resultados**

**Implementación del modelo neuronal obtenido**

1. Cálculo de la salida para todas las posibles combinaciones de la Tabla 1

Table

Description automatically generated

Tabla 1 Valores para probar la red neuronal

Al realizar todas las combinaciones posibles con la tabla anterior, so obtiene un total de 81 combinaciones, las cuales se utilizan para alimentar la red ya entrenada, de forma que se obtiene un valor de corriente de salida para cada una. Estos datos se pueden observar siguiendo el siguiente enlace: <https://github.com/Makako08/Tarea1_IA/blob/main/ParteA/Datos_predichos.csv>

1. Clasificación las variables de entrada en función de su sensibilidad en el resultado final mediante un estudio de tipo “ceteris paribus”.

Como resultado del estudio, se obtiene la tabla de la Figura 54.

Interfaz de usuario gráfica, Calendario

Descripción generada automáticamente con confianza media

Figura 54. Tabla de resultados de estudio Ceteris Paribus

Fuente: Elaboración propia 1

De ella se obtiene que el parámetro que genera una mayor variación en la salida es “dIf”, luego le sigue “Pf”, “e” y por último “Iy”

1. En función de los resultados del apartado anterior, se redacta un conjunto de directrices orientado a la automatización del proceso (control del valor de corriente).

Se debe de realizar un estudio para obtener la forma más eficaz de variar el parámetro “dIf” ya que es el que más afecta al sistema. Luego se debe de conectar un medidor de corriente a la salida que esté tomando valores de corriente constantemente, estos valores se le deben de pasar a un controlador que tenga la capacidad de variar la corriente “dIf” de forma que el proceso de variación de la corriente final se hará de forma automática.

**Tarea Sistemas conexionistas. Parte 2.**

OBJETIVO: desarrollar de manera organizada y metodológicamente adecuada un modelo de red neuronal convolucional para clasificar datos de interés ingenieril, demostrando posteriormente la utilidad de la herramienta

**Problema**

Desarrollar un modelo neuronal convolucional que permita la clasificación de las condiciones (normal ó fallos).

Caso: se usa el conjunto de datos etiquetado como lp.5 dentro de los provistos y descritos en el siguiente enlace:

<https://archive.ics.uci.edu/ml/datasets/Robot+Execution+Failures>

El conjunto consiste en una serie de valores de fuerzas y torques medidas en un sistema robótico operando con un efector final tipo herramienta, y clasificados en forma de operación normal o bien un tipo de error de herramienta o pieza (las condiciones normal o defectuosa están etiquetadas como categorías textuales).

**Planteo de estrategia**

Se utiliza un modelo de convolución que permite realizar una clasificación multiclase, esto debido a las características del problema, las cuales indican que se debe clasificar los casos ingresados en condiciones normal o fallos.

Se selecciona el modelo convolucional que utiliza la biblioteca Keras , la cual funciona a nivel de modelo proporcionando bloques modulares sobre los que se pueden desarrollar modelos complejos de aprendizaje profundo [3].

* Citado y referenciado adecuadamente

El modelo utilizado fue tomado de [4] .El código usado se presenta en el siguiente link <https://colab.research.google.com/drive/1g-cJZlxlnDqmEPcWUnjOyk7EowwzhqMU?usp=sharing>

* Explicar todos los elementos de ese modelo

Primero se preparan los datos, estos deben normalizarse antes de ingresar a la red y además del archivo se toman las 5 categorías y se crea una matriz que asigna un valor a cada una. Con ello, del conjunto de datos dado en el archivo lp.5, se toman los valores de entrada y las categorías esperadas.

Después, se dividen lo datos de entrada en un conjunto de entrenamiento y otro de test, además se asigna un porcentaje de datos para validación y se crea una matriz que posee un error de +/- 5% en cada dato para una segunda validación.

Posterior a ello, se procede a crear por medio de la librería keras lo necesario para el modelo. Primero las capas de convolución y pooling, en este caso se decide utilizar 3 convoluciones y 2 pooling, además de dos capas dense y una de Flatten para finalizar con el proceso. Esto va a permitir pasar las imágenes por una descomposición jerárquica de la información de entrada, luego un submuestreo de los mapas de activación al aplicar el pooling y para hacer la tarea de clasificación se utiliza las capas densas, en este caso se usa una softmax al final al ser una clasificación multi clase.

El modelo se define con model.complite en el que se determina la función de perdida a utilizar, la función de optimización y la metrics. Luego con el model.fit se solicita el entrenamiento del modelo. Mostrando en pantalla el valor de la perdida y accuracy.

**Soluciones**

* Descripción el estudio de hiperparámetros realizado

Se determino utilizar la siguiente

Numero de capas: 3 capas de convolución

Numero de filtros 16, 32 y 32 en las capas de convolución

Tamaño de Kernel: 3

Numero de neuronas 8 y 5 en capas densas

strides= 2x1

Padding = “same”

Se determino este número de capas ocultas ya que al ingresar una o dos capas se obtienen valores muy bajos de accuracy y mayor perdida obtuvo un valor mayor de perdida. Como se aprecia en las imágenes primer resultado 2 capas y ultimo 3 capas, se aclara que al tratar de incrementar el número de capas el pooling genera un error esto podria relacionarse al tamaño de los datos el cual es muy pequeño por ende se utilizan 3 capas de convolución.



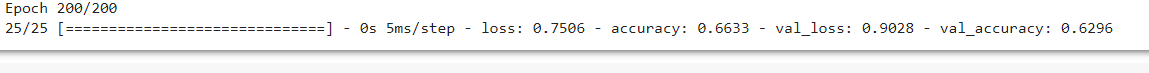


Figura 55. Valores obtenidos al variar número de capas ocultas, con 2 y 3 capas respectivamente. Fuente: elaboración propia

Para el tamaño del Kernel se define utilizar uno 3x3, debido a que fue el valor que permitió mejores resultados de entrenamiento sin caer un generar un error en el pooling. En las imágenes inferiores se muestra dos evaluaciones realizadas





Figura 56. Valores obtenidos al variar tamaño de kernel, con 1x1 y 3x3 respectivamente. Fuente: elaboración propia

Para el valor de stride considerando el tamaño de los datos se seleccionaron dimensiones de prueba pequeñas, por ejemplo, 2x2, 1x1, 2x1, 2x1. A partir del comportamiento observado se determina que se utiliza un stride de 1x1 ya que esto permite ir abarcando la mayoría de los datos en la matriz. En la imagen siguiente se observa el comportamiento del modelo con 2x1 y 1x1 notando como mejoran los resultados al usar un stride de 1x1.

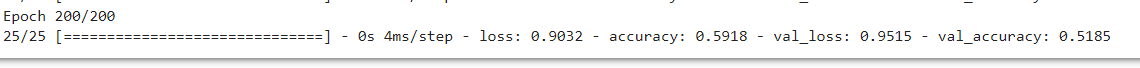


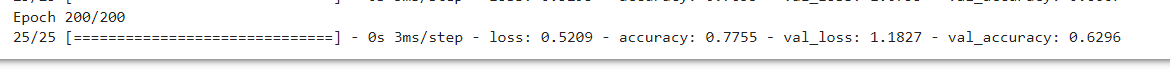


Figura 57. Valores obtenidos al variar stride, con 2x1 y 1x1 respectivamente. Fuente: elaboración propia

El padding puede asumir 2 valores; "valid" o "same", en este caso se selecciona same debido a que da como resultado un relleno con ceros uniformemente a la izquierda/derecha o arriba/abajo de la entrada. Mientras que el valid no añade ceros por lo cual se podría perder información y en este caso la cantidad de datos no es tan grande para permitirse perder datos.

Para el número de filtros de la capa de convolución, donde cada filtro buscará la extracción de características específicas, se determinan considerando la cantidad de datos y la cantidad de capas convolucionales. Un valor de 16 en una la primera capa convolucional, 32 en la segunda capa convolucional y 32 en la tercera capa convolucional, con el fin de lograr extraer las características que permiten lograr la clasificación. Se define un valor menor en la primera capa ya que está considerando que las capas más cercanas a la entrada aprenderán información de bajo nivel y las capas más profundas pueden aprender información de alto nivel. Además, estos valores presentaron la mejor puntuación al entrenar en comparación a las combinaciones de 4,8,8 y 32,64,64 como se puede apreciar en las siguientes imágenes.





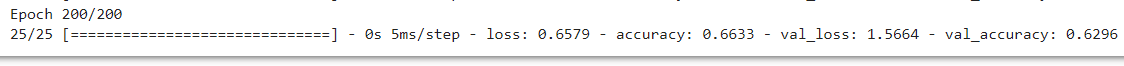


Figura 58. Valores obtenidos al variar número filtros en capas convolucionales (4,8,8 filtros), (16,32,32 filtros) y (32,64,64 filtros) respectivamente. Fuente: elaboración propia

Se seleccionan solo 2 capas de pooling con el fin de no perder información al ser un conjunto de datos pequeño, ya que el pooling mejora el desempeño, pero reduce la información espacial. Para el caso de las capas de las capas densas se seleccionan 16 neuronas y una final con 5 neuronas ya que la salida del sistema son 5 categorías.





Figura 59. Valores obtenidos al variar número neuronas de la capa densa, se presenta el resultado a 8 y 16 respectivamente. Elaboración propia

**Resultados**

* Evidencia cuantitativa del rendimiento

A –Usando para la validación un subconjunto de los datos disponibles

Se presentan los resultados del model predict con el subconjunto de validación



Figura 60. Resultado de la predicción con los valores de validacion

B – usando un conjunto de datos sintéticos generado por medio de un ruido aleatorio de máximo +- 5 % de cada dato, y comparando el resultado de la clasificación obtenida con el del dato de partid**a**

Se presentan los resultados del model predict con el subconjunto creado de datos sintéticos



Figura 61. Resultado de la predicción con los valores de sintéticos

Se aclara no se puedo proyectar las matrices de confusión, pero si se generaron las predicciones y programaciones que debían generarlas, sucedió un error que no se pudo corregir para el momento de redacción.

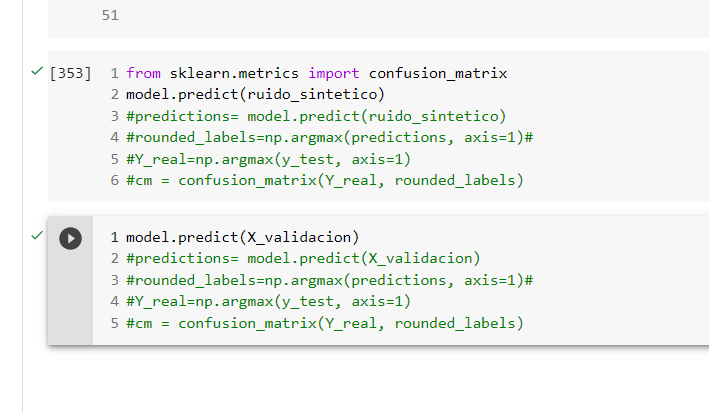


Figura 62. Código para matrices de convolución

**Anexos**

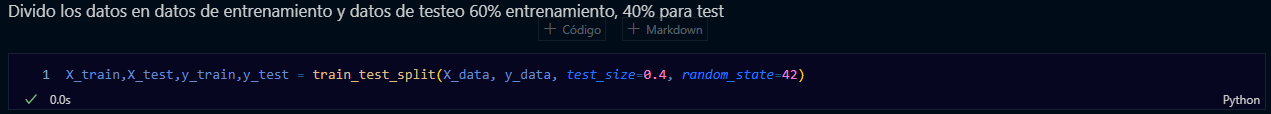
Programación parte II.

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente



**Texto

Descripción generada automáticamente**

Texto

Descripción generada automáticamente

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

Forma

Descripción generada automáticamente con confianza media

**Texto

Descripción generada automáticamente**

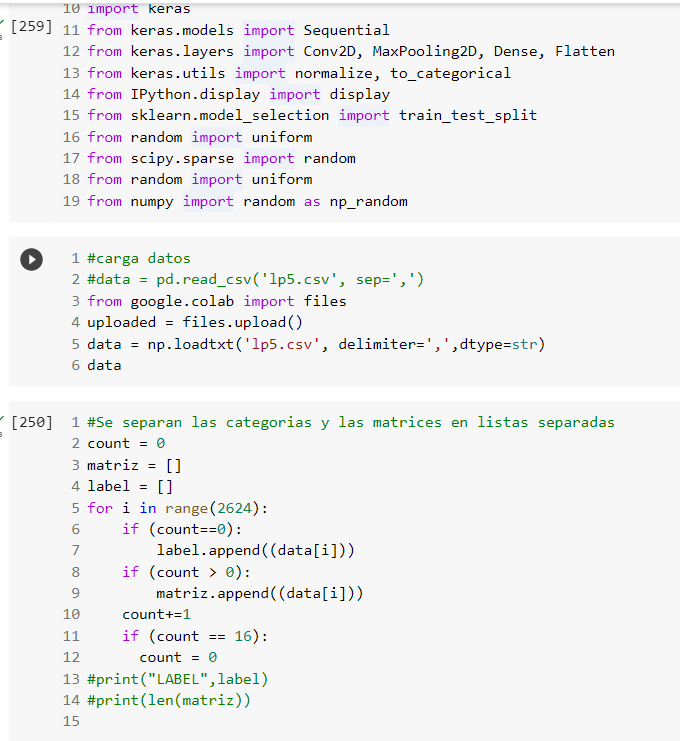
Texto

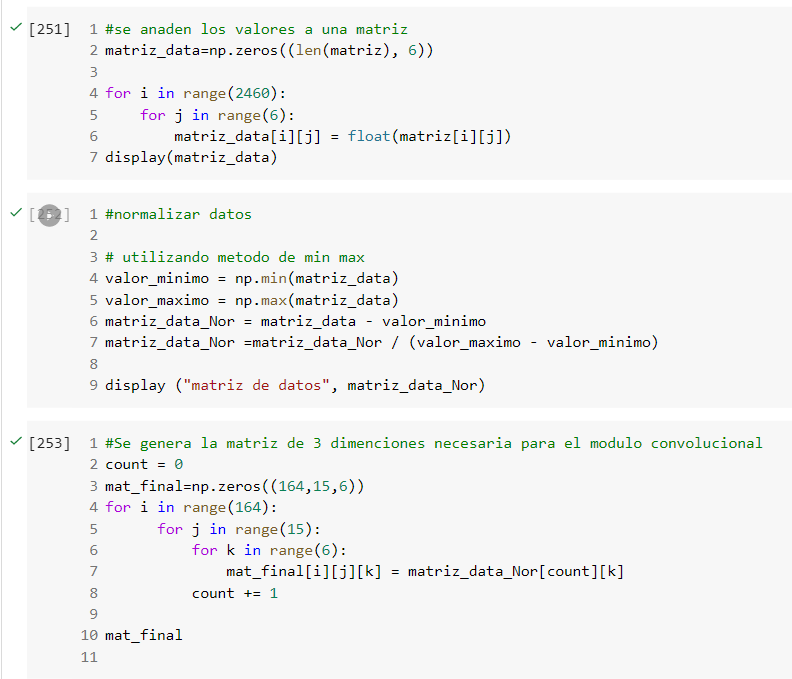
Descripción generada automáticamente

**Texto

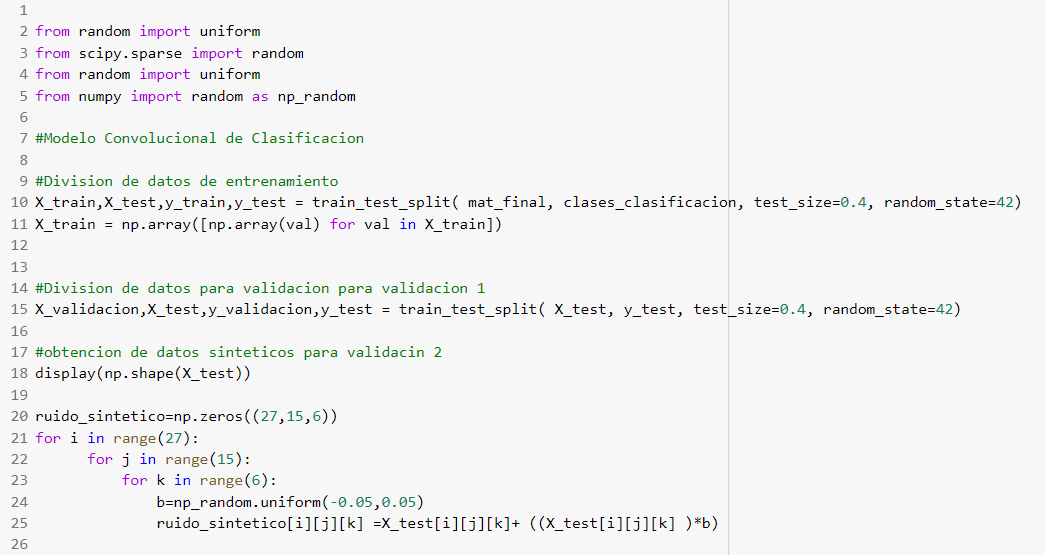
Descripción generada automáticamente**

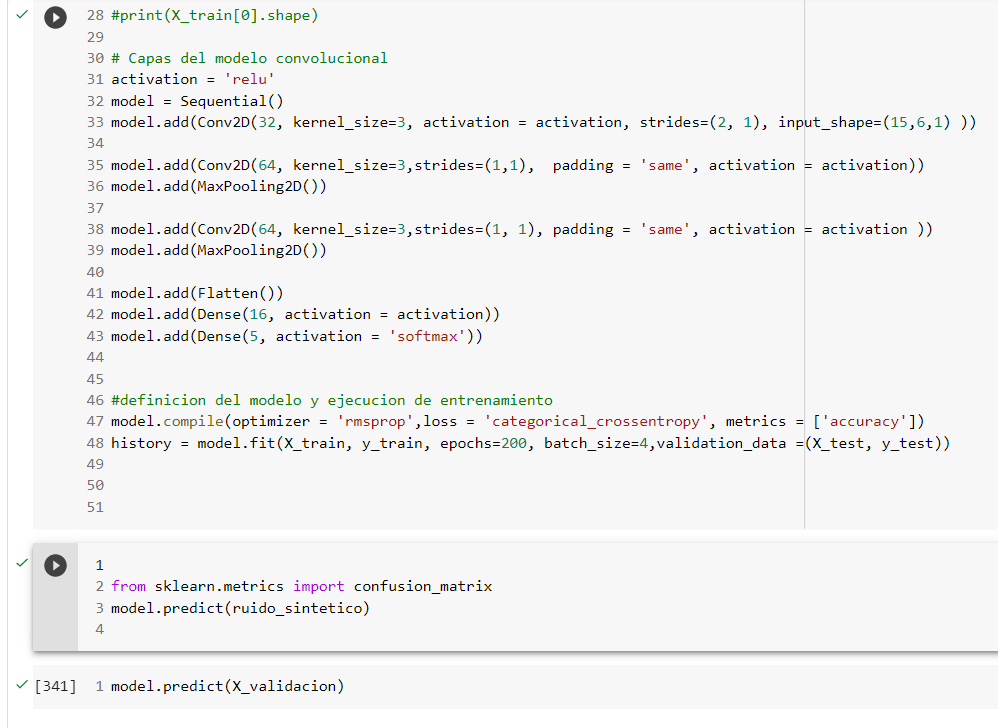
Programación parte II. Este también se puede observar en el siguiente link <https://colab.research.google.com/drive/1g-cJZlxlnDqmEPcWUnjOyk7EowwzhqMU?usp=sharing>











**Referencias**

|  |  |
| --- | --- |
| [1] | cctmexico, «Youtube,» cctmexico, 28 junio 2020. [En línea]. Available: https://www.youtube.com/watch?v=RU--D9BWIT4. [Último acceso: 28 abril 2023]. |
| [2] | K. Team, «Keras.io,» 2014. [En línea]. Available: https://keras.io/api/optimizers/adamax/. [Último acceso: 28 abril 2023]. |

[3]IONOS, ¿Qué es Keras? “Introducción a la biblioteca de redes neuronales”, 8.10.2020 Jul. 21, 2020. https://www.youtube.com/@DigitalSreeni/featured (accessed Apr. 28, 2023).

[4]DigitalSreeni, “143 - Clasificación multiclase usando Keras,” www.youtube.com, Jul. 21, 2020. https://www.youtube.com/@DigitalSreeni/featured (accessed Apr. 28, 2023).