**Planteamiento de la solución**

Para la solución de este problema se utilizará una red neuronal superficial con 4 neuronas de entrada para recibir las 4 características del dataset, una neurona de salida con función de activación lineal, debido a que los datos están comprendidos entre 0 y 7.1. Adicional a esto, se utilizará un número inicial de 10 neuronas en la capa oculta con función de activación sigmoidal, las cuales serán variadas con el objetivo de que la red aprenda la dinámica de la función. Al ser un problema no lineal, es idóneo utilizar esta técnica debido a que estas permiten resolver problemas de regresión suministrándole únicamente los datos de entrada y salida de la función a aprender.

Se entrenará la red con las librerías de Tensorflow donde se préndete variar el número de nodos en la capa oculta, el número de épocas y el optimizador junto a sus parámetros, para así, entrenar la red y comparar los resultados obtenidos con las diferentes variaciones nombradas previamente. Cabe resaltar, que la red será entrenada con el 70% del total de los datos del dataset y la validación será realizada con el 30% restante.

**Resultados**

Al desarrollar la aplicación propuesta en la sección anterior, se implementaron las diferentes variaciones nombradas obteniendo los resultados obtenidos en la tabla #. Cabe resaltar que, al hacer el entrenamiento de la red neuronal sin aumentar el número de capas ocultas, el grafo mostrado en la Fig. #, se mantiene constante, debido a que solo se varían parámetros de los optimizadores, los optimizadores y número de nodos en la capa oculta sin afectar directamente la visualización del grafo.

ECHE NO JODA AQUÍ VA LA TABLA

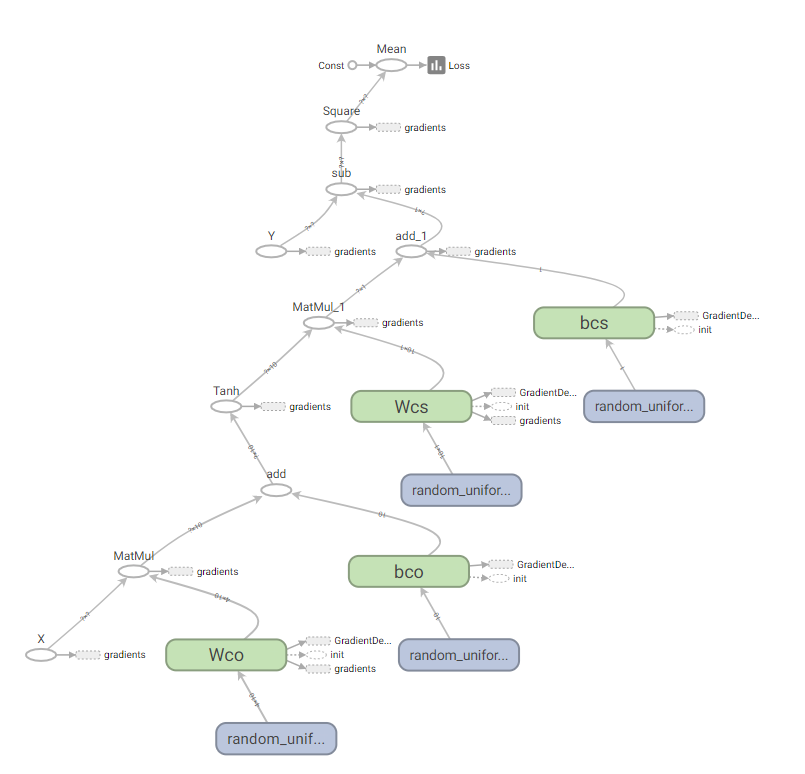


Fig. #. Grafo de la Red neuronal superfial

Se puede observar que los mejores resultados obtenidos fueron obtenidos en las pruebas 12, 13 y 14, debido a que al hacer una normalización los datos de entrada quedan comprendidos entre valores de -1 y 1 generando mejores resultados en la función de perdida. No obstante, en el momento de ver los anexos # (Agregar número anexo datos validación 12 13 y 14) se puede observar que la red no sigue de manera tan exacta los datos deseados como dice la función de perdida. Este comportamiento (Fig. 1.) es debido a la normalización, ya que al ser mas pequeños los valores de operación de la red se genera un valor de perdida más pequeño así los resultados desnormalizados de la red no sean los idóneos.

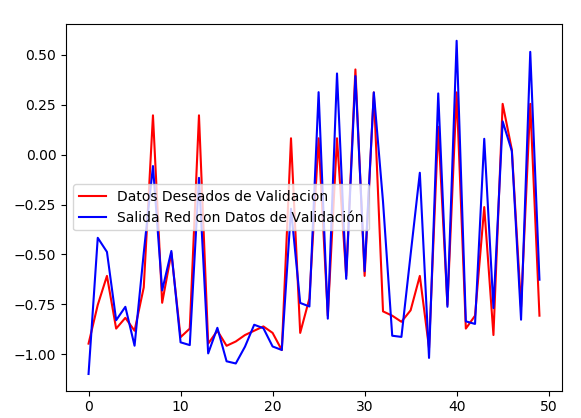


Fig 1. Salida de la red con optimizador RMS y los datos normalizados

Por otra parte, al realizar las pruebas de la red con los diferentes optimizadores, pero sin normalizar los datos de entrada, se puede observar que las funciones de perdida dan valores muy grandes de error, como es el caso de la prueba número 7 con el optimizador de Adagrad, que tiene un error en el entrenamiento del 105.2%, este comportamiento se puede observar en la Fig. 2, la cual fue obtenida por medio de tensorboard.

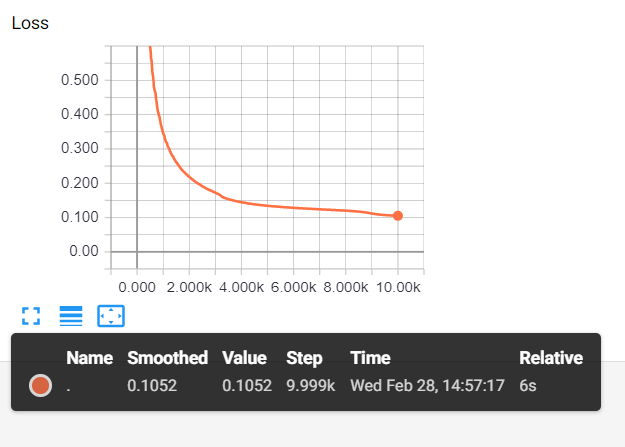


Fig. 2. Función de perdida con el optimizador Adagrad en la prueba número 7

Sin embargo, estas pruebas arrojaron los mejores resultados de la regresión en términos de afinidad con los datos deseados. Finalizada las pruebas, se decide trabajar con la prueba número 11, la cual es una red entrenada con el optimizador de momentum y arrojo una pérdida del 5.17% en la fase de entrenamiento y del 13.98% en la validación de la red. Se decide utilizar esta red debido a su alta afinidad con los datos de validación como se puede observar en la Fig. 3, superando la afinidad de la red entrenada con RMS y normalizada nombrada en la Fig. 1.

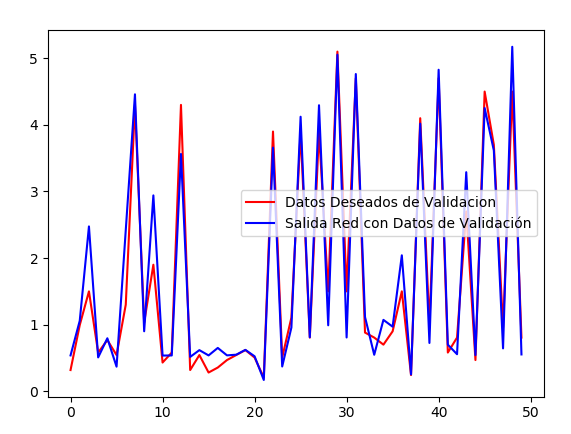


Fig. 3. Salida de la red con optimizador por Momentum, 10 nodos en la capa oculta

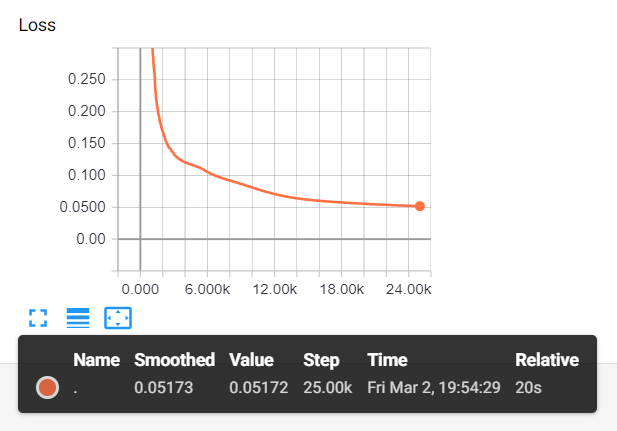


Fig. 4 Función de perdida de la red en la fase de entrenamiento.

Una vez se escogió el optimizador y parámetros a utilizar, se realiza la extracción de los pesos sinápticos para realizar la validación de la red en Arduino con ayuda de la librería CSV de Python, que permite hacer la exportación de diferentes datos a un archivo .csv previamente creado. Los archivos creados para esta tarea son MLP\_Pesos\_CO y MLP\_Pesos\_CS que contienen los pesos sinápticos que conectan la capa de entrada con la capa oculta y los pesos que conectan la capa oculta con la de salida con sus respectivos bias.

Al realizar la implementación de la red neuronal en Arduino, se puede observa que la red genera una respuesta similar a la obtenida en la Fig. 3 por medio de Tensorflow, obteniendo una dinámica muy similar como se puede observar en la Fig. 5, la cual es una gráfica obtenida por medio del IDE de Arduino y la herramienta Serial Plotter.

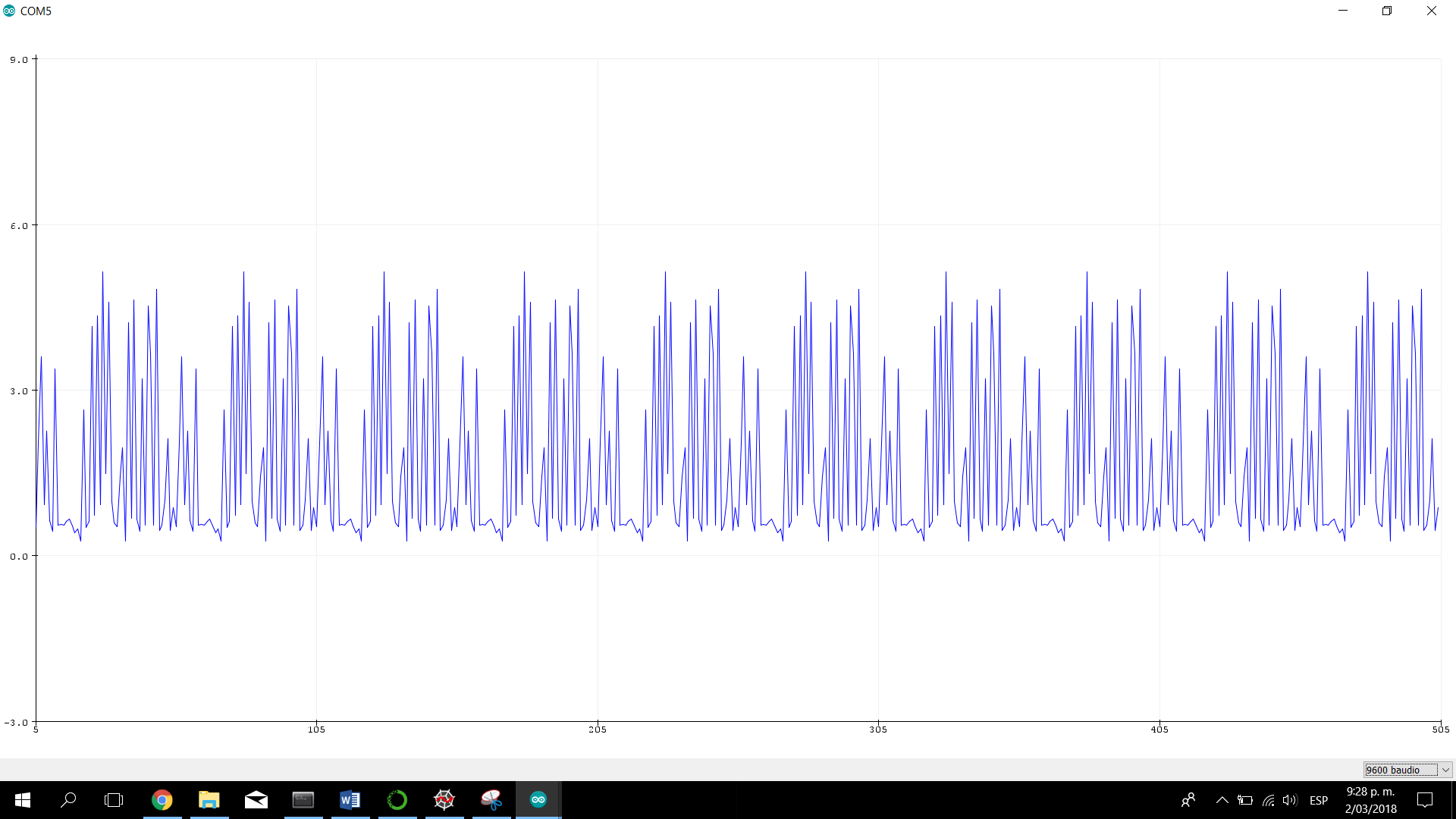


Fig. 5. Respuesta de la red neuronal superficial implementada en Arduino.