1.- Práctica 12: red neuronal

En esta práctica se demuestra el funcionamiento básico del aprendizaje de una máquina, vamos a reconocer dígitos de imágenes pequeñas en blanco y negro con una red neuronal. Dando por inicio una fase de entrenamiento y una fase de prueba para probar la efectividad del perpectron generado.

2.- Descripción de la tarea

- Paraleliza lo que se pueda sobre la red neuronal (ojo, no todo se podrá durante entrenamiento, ya que los pesos pueden cambiar en cada paso) y estudia el efecto de esto en su tiempo de ejecución con las pruebas estadísticas y las visualizaciones pertinentes.

3.- Tarea

Tomando en cuenta el código que nos proporciona la práctica, en esta tarea se realizó una versión paralela, donde se concluyó que la parte más factible que se encontró a realizar la paralización fue la sección de prueba, ya que ahí se evalúa de manera independiente la respuesta si es correcta o no, como en la descripción lo menciona no se recomienda paralelizar el apartado de aprendizaje o de entrenamiento, ya que contienen los cambios a los perceptrones, donde se observa si se obtiene una respuesta correcta. La librería que se utilizó fue *doParallel*, se creó una función *obj,* para realizar los cálculos y la evaluación de los resultados para ser guardados en dichas librerías y ser llamados más adelante. Se utilizó función *sys.time()* para guardar el tiempo que duración de ejecución de los códigos, donde se crearon *data.frame* para almacenar los resultados, por último se realizó un *for* para realizar las iteraciones de los códigos, los cuáles fueron añadidos al código con la función *source* y con el paquete *ggplot2* se generó los diagrama caja-bigotes que se muestran a continuación:

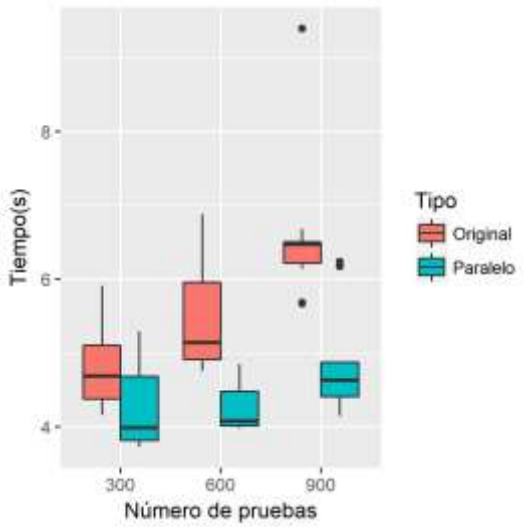

Figura 1.- Diagrama caja-bigotes de tiempo de ejecución de códigos original y paralelo.

En la figura 1 se muestra el diagrama caja-bigotes del tiempo de ejecución de comparación del código original y el código paralelo, en éste se puede observar la diferencia del tiempo de ejecución para distintos valores de *t1*, estos se usaron para distinguir cuando el cambio el tiempo de ejecución es significativo. Los cambio de tiempos se fueron separando una de la otra, al ir aumentando el número de pruebas realizadas, como se observa en 600 pruebas en adelante los rangos de tiempos se reducen en la sección paralelizada en la mitad del tiempo y mantienen un promedio mayor en sus medianas a comparación del código original que siguen aumentando.

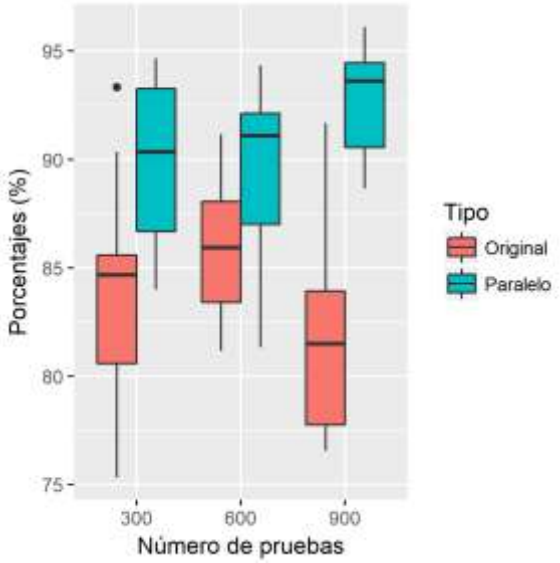

Figura 2.- Diagramas caja-bigotes de pruebas contra el porcentaje de resultados correctos.

En la figura 2 se muestran las diferentes pruebas en función del porcentaje del número de resultados correctos. Se puede observar un aumento en el porcentaje de aciertos, al incrementar la cantidad de pruebas realizadas. Esto se puede atribuir a que al tener mayor número de pruebas el código aprenda y como consecuencia le resulte un menor trabajo identificar los dígitos correctos.

4.- Conclusiones

Lo primero que se pudo concluir que si es posible poder paralizar el código base para así poder tener un mejor aprovechamiento en el tiempo, otro factor que se vio que al ir aumentando la cantidad de pruebas las probabilidades de tener un mayor número de aciertos aumenta significativamente eso se puede atribuir que el programa aprenda y le resulte más fácil el poder identificar los dígitos correctamente.

5.- Especificaciones de equipo

Modelo del sistema Inspiron 5420, fabricada por Dell Inc, procesador Intel(R) Core(TM) i5-3210MCPU @2.50Hz 2.50 Hz, memoria instalada (RAM) 8 GB (7.86 GB utilizable), tipo de sistema operativo de 64 bits procesador x64, edición de Windows 10 Pro.