Universidad del Valle de Guatemala
Deep Learning y Sistemas Inteligentes
Sección 10

Laboratorio #4

FFNN vs RNN

21016 Javier Chavez
21085 Andres Quezada
21631 Mario Cristalitros

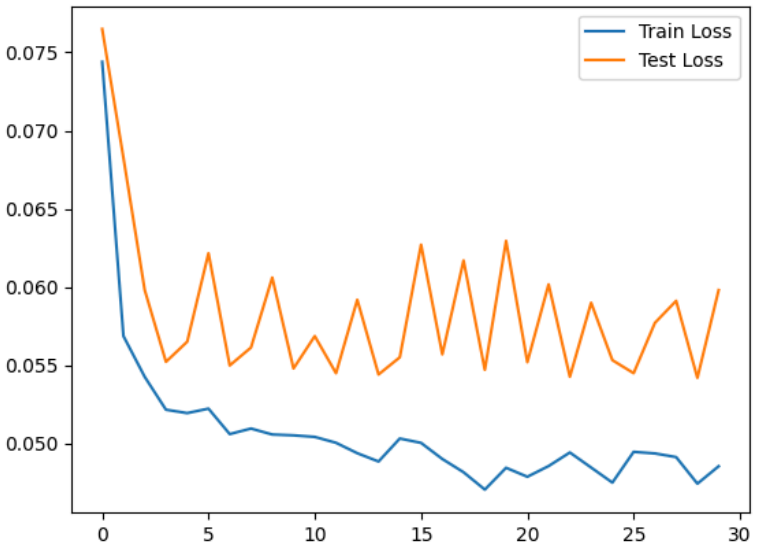
Resultados:

Red Simple FFNN:

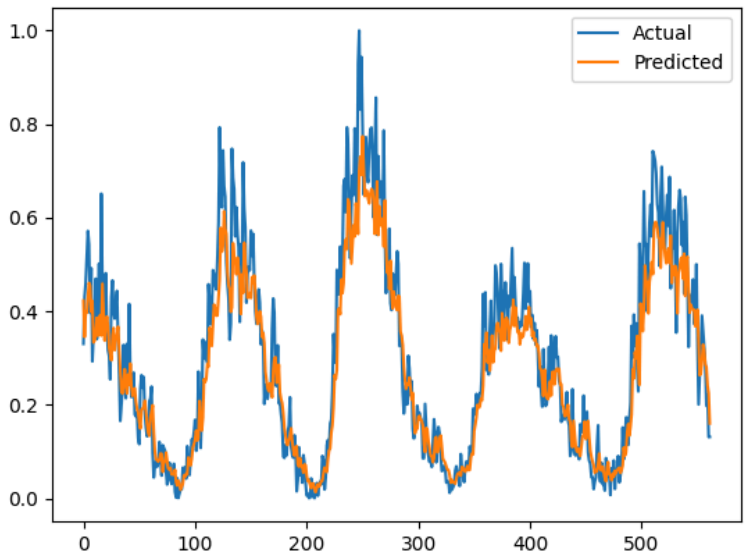
- Configuración:
  - Capas:
    - 1 Entrada Flatten
    - 2 Densas (100 y 50 unidades)
    - 1 Dropout
    - 1 Densa (salida)
  - Activación: Relu
  - Dropout: 50%
  - Épocas: 30
  - Porcentaje de entreno: 80-20

- Resultados:
  - Pérdida final: 5.98%

Pérdida durante entrenamiento vs durante validación:



Comparación valor actual vs predicción:

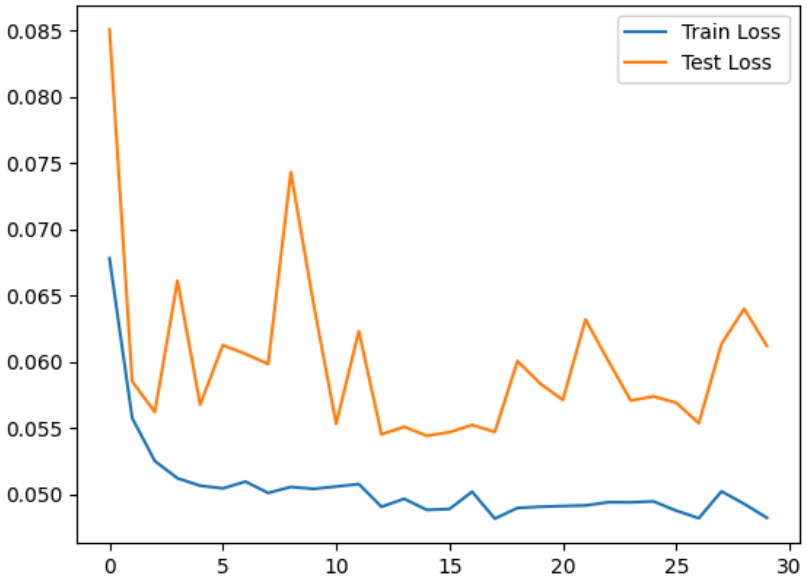


Red RNN:

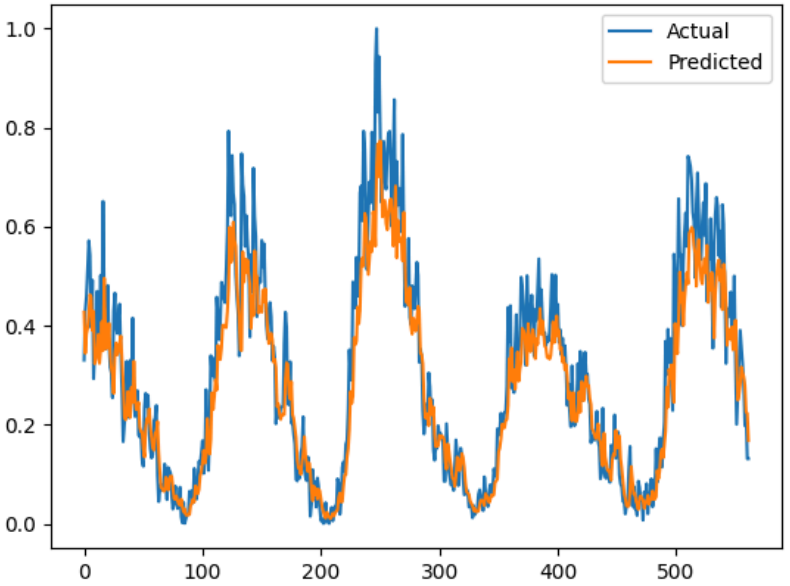
- Configuración:
  - Capas:
    - 2 Capas RNN
    - 1 Dropout
    - 1 Densa (salida)
  - Activación: Relu
  - Dropout: 50%
  - Épocas: 30
  - Porcentaje de entreno: 80-20

- Resultados:
  - Pérdida final: 6.12%

Pérdida durante entrenamiento vs durante validación:



Comparación valor actual vs predicción:



Discusión:

Redes Neuronales Feedforward (FFNN)

- Pros:
    - Simplicidad: son relativamente simples y rápidas de entrenar. Tienen una arquitectura directa con capas densas que no requieren procesamiento secuencial.

    - Computacionalmente eficientes: por su estructura estática, pueden ser más eficientes en términos de tiempo de entrenamiento y predicción.

    - Menor complejidad: no necesitan manejar estados internos o secuencias de datos, lo que simplifica su implementación y ajuste.

- Contras:

    - No captura dependencias temporales: como no tienen memoria a corto plazo para capturar las relaciones temporales en los datos. Esto es crucial en problemas de series temporales donde las dependencias a lo largo del tiempo son importantes.

    - Escalabilidad: para datos secuenciales extensos o con relaciones complejas a lo largo del tiempo, las FFNN pueden requerir arquitecturas más complejas o extensas para capturar adecuadamente la dinámica temporal.

Redes Neuronales Recurrentes (RNN)

- Pros:
    - Captura de dependencias temporales: están diseñadas para manejar datos secuenciales y capturar dependencias temporales a lo largo del tiempo. Esto es muy bueno para series temporales, donde los valores futuros dependen de los valores pasados.

    - Memoria a corto plazo: tienen la capacidad de mantener una memoria interna de los estados anteriores, lo que les permite recordar patrones temporales y dinámicas a corto plazo.

    - Adecuadas para series temporales: son naturalmente adecuadas para problemas de predicción en series temporales.

- Contras:

    - Complejidad computacional: suelen ser más complejas de entrenar y más lentas debido a la necesidad de procesar secuencias y mantener estados internos.

    - Problemas de desvanecimiento y explosión del gradiente: pueden enfrentar problemas con el desvanecimiento o explosión del gradiente, lo que puede dificultar el entrenamiento de redes profundas.

Elección para resolver el problema:

Luego de observar los resultados, a pesar de que los valores finales de pérdida son similares para ambos modelos, nuestra elección del tipo de red para resolver este problema es RNN por su capacidad superior para manejar datos secuenciales y series temporales. Al estar diseñadas para retener información a lo largo de secuencias pueden capturar patrones temporales y dependencias a corto plazo que son esenciales en la predicción de series temporales. Aunque las redes FFNN ofrecen un rendimiento comparable en términos de pérdida, la naturaleza de los datos temporales hace que las RNNs sean más adecuadas para capturar la dinámica y las correlaciones en el tiempo, proporcionando una ventaja en este tipo de problemas de series temporales.

Repositorio:

https://github.com/andresquez/DeLe-Lab4

Referencias:

Elsen, E. (2024) Optimizing RNN performance. Recuperado de: https://svail.github.io/rnn_perf/