Tarea 3
Martín Valderrama Illesca
MA7260 - Aprendizaje Profundo en Diagnóstico y Pronóstico de Fallas, Primavera 2020

1. Preprocesamiento de datos

1.1. MinMaxScaler

Se aplica un escalamiento de los datos para que todos tengan los mismos rangos y así sean comparables por la red neuronal. Para esto, sobre el conjunto de entrenamiento se definen los rangos mínimo y máximo de cada sensor, luego se confina cada sensor al rango -1 y 1. Estos mismos valores, obtenidos del conjunto de entrenamiento, se utilizan para ajustar los datos del conjunto de test. Esta forma de procesar los datos de entrenamiento y test consideran que los datos de test nunca han sido vistos por el modelo, por lo que tienen que ser procesados por conocimiento disponible en el conjunto de entrenamiento.

1.2. Separación en conjuntos de entrenamiento y test

Para el entrenamiento y evaluación de los modelos se realiza una separación de los datos de entrenamiento. Esto para obtener un conjunto de validación en donde se pueda evaluar el comportamiento durante el entrenamiento, y así, seleccionar el mejor modelo con la técnica de *early stopping*. Dicho modelo, luego se evalúa sobre dos conjuntos que la red nunca ha visto, el conjunto de test desde el entrenamiento: en este caso, se separa otro conjunto desde el entrenamiento original (independiente del que utiliza la red para entrenarse) para evaluar el rendimiento de la red; y el conjunto de test entregado por el cuerpo docente.

El procedimiento de crear un segundo conjunto de test, se realiza para tener una comparación con el conjunto de test entregado por el cuerpo docente, ya que el rendimiento obtenido en este último es similar entre todos los modelos. Pudiendo tener varias causas, las cuales se desconoce. El conjunto de test extraído de los datos de entrenamiento, se utiliza para comparar de mejor forma el comportamiento de las redes.

2. Metodología de entrenamiento

La metodología de entrenamiento consiste en cargar los datos, generar las respectivas datasets de entrenamiento, validación y test, además de cargar el dataset de testeo original. Luego se realizan las transformaciones de los datos y se entregan al modelo para su utilización.

En el entrenamiento, se tiene que el modelo es creado con las arquitecturas RNN, LSTM y RNN con convolución (especificadas en el notebook de la tarea), se genera un optimizador y se compila el modelo, para finalmente correr el entrenamiento por 100 épocas, a menos que la función de *early stopping* detenga el entrenamiento luego de no haber optimizado el valor de la loss (MSE) durante más de 5 épocas.

3. Arquitecturas

Las arquitecturas están basadas fuertemente en el workshop 05 del ramo. Muy pequeñas modificaciones son realizadas a las arquitecturas. Sin embargo, son con las que se obtienen mejores resultados sobre el conjunto de test.

3.1. RNN

```
model.add( RNN(64, activation='relu', return_sequences=True ) )
model.add( RNN(64, activation='relu', return_sequences=True ) )
model.add( Dropout(rate=0.1) )
model.add( RNN(32, activation='relu', return_sequences=True ) )
model.add( RNN(32, activation='relu', return_sequences=True ) )
model.add( Flatten() )
model.add( Dropout(rate=0.2) )
model.add( Dense(units=96, activation='relu') )
model.add( Dense(units=96, activation='relu') )
model.add( Dense(units=64, activation='relu') )
model.add( Dense(units=64, activation='relu') )
model.add( Dense(units=1, activation='linear') )
```

3.2. LSTM

```
model.add( LSTM(64, activation='relu', return_sequences=True ) )
model.add( LSTM(64, activation='relu', return_sequences=True ) )
model.add( Dropout(rate=0.1) )
model.add( LSTM(32, activation='relu', return_sequences=True ) )
model.add( LSTM(32, activation='relu', return_sequences=True ) )
model.add( Flatten() )
model.add( Dropout(rate=0.2) )
model.add( Dense(units=96, activation='relu') )
model.add( Dense(units=96, activation='relu') )
model.add( Dense(units=64, activation='relu') )
model.add( Dense(units=64, activation='relu') )
model.add( Dense(units=1, activation='linear') )
```

3.3. Convolutional RNN

```
model.add( layers.Conv2D(32,3, activation='relu') )
model.add( layers.Conv2D(32,3, activation='relu') )
model.add( layers.TimeDistributed( Flatten() ))
model.add( RNN(64, activation='relu', return_sequences=True ) )
model.add( RNN(64, activation='relu', return_sequences=True ) )
model.add( Dropout(rate=0.1) )
model.add( RNN(32, activation='relu', return_sequences=True ) )
model.add( RNN(32, activation='relu', return_sequences=True ) )
model.add( Flatten() )
model.add( Dropout(rate=0.2) )
model.add( Dense(units=96, activation='relu') )
model.add( Dense(units=96, activation='relu') )
model.add( Dense(units=64, activation='relu') )
model.add( Dense(units=64, activation='relu') )
model.add( Dense(units=1, activation='linear') )
```

4. Resultados

Del procesamiento de datos del conjunto FD0001 se obtienen los siguiente histogramas (previos al procesamiento) que dan cuenta de la distribución de los datos.

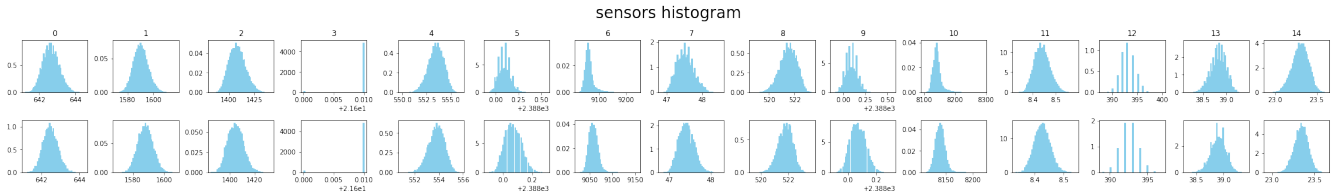

Figura 1: Distribución de los sensores del conjunto FD0001. Fila superior es conjunto de entrenamiento, fila inferior conjunto de test brindado por el cuerpo docente.

Se puede observar como algunas variables se concentran en algunos valores, mientras que otras abarcan más entre el -1 y 1. Otras toman solamente dos valores como el caso del sensor 3.

Estas distribuciones nos dicen que los datos de entrenamiento tienen tanta o más información que los del conjunto de testeo. Esto indicaría que los resultados obtenidos por una red entrenada deberían ser similares a los del conjunto de entrenamiento.

4.1. RNN

Los resultados del entrenamiento de la RNN se observan en la figura siguiente. El promedio del MSE de las 3 realizaciones del entrenamiento sobre el conjunto de test extraído del conjunto de entrenamiento original, es de 7.36. Mientras que sobre el conjunto de test original se logra un MSE de 75.42. Cabe en consideración que el MSE logrado sobre el conjunto de entrenamiento fue cercano a 5, por lo que se considera que el rendimiento sobre el conjunto de test extraído de los datos originales de entrenamiento con los cuales la red no es entrenada, son consistentes y una buena métrica de comparación con las otras redes.

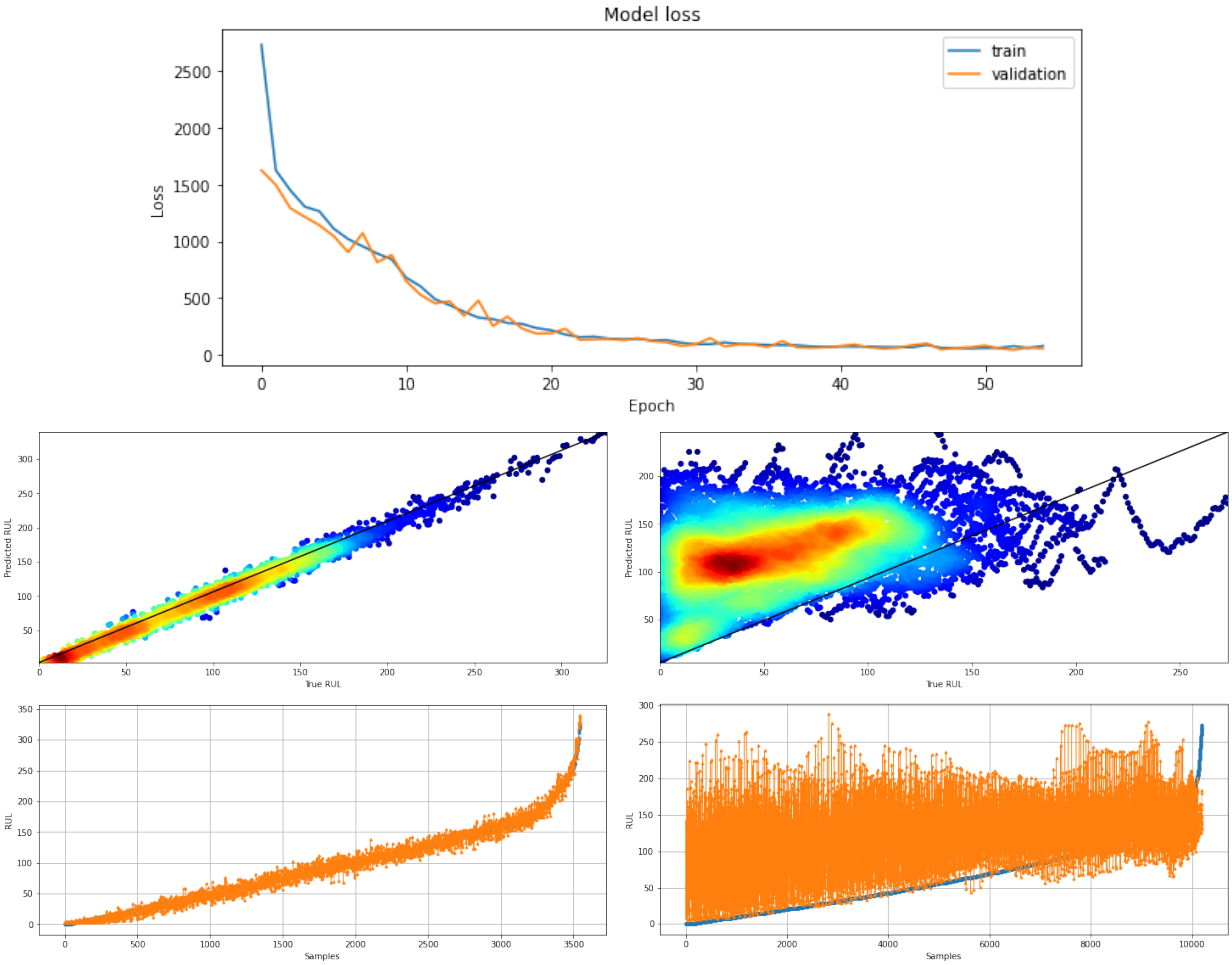

Figura 2: Resultado de entrenamiento de RNN sobre conjunto FD0001. El gráfico superior denota el transcurso del entrenamiento y su función de loss sobre el conjunto de entrenamiento y validación. Resultados de la RUL predicha y su contraparte ordenada, se ven en los siguientes gráficos, izquierda son las predicciones sobre el conjunto de test extraído de los datos de entrenamiento originales; en la derecha están los resultados de la RUL predicha sobre el conjunto de test entregado por el cuerpo docente.

4.2. LSTM

Resultados que se obtienen con la LSTM son de un MSE promedio de 14.49 sobre el conjunto de test extraído, y de 77.21. Siendo notablemente peor sobre el conjunto de test extraído y levemente peor sobre el conjunto de test original.

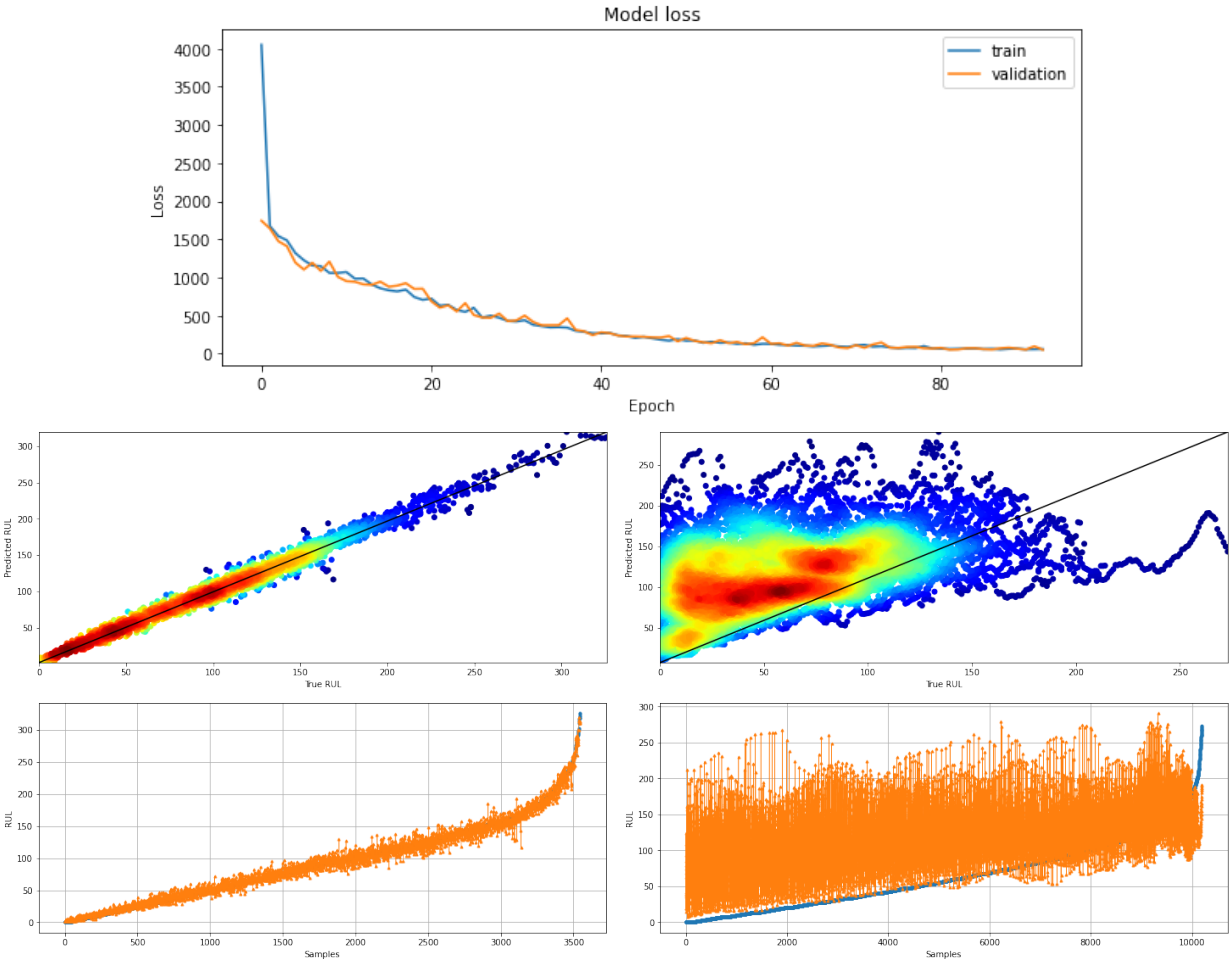

Figura 3: Resultado de entrenamiento de LSTM sobre conjunto FD0001.

4.3. RNN sobre FD0004

Con los resultados obtenidos anteriormente, se realiza el siguiente ejercicio de entrenar la red RNN sobre el conjunto de datos de FD0004.

Estos datos son procesados de igual forma que el FD0001, y se obtienen los siguientes resultados relativos al MSE promedio de las tres realizaciones: sobre el conjunto de test extraído se obtiene 19.51, y el conjunto de test original obtiene 92.34. Con estos resultados, se puede ver que en las figuras obtenidas no se encuentra una RUL predicha de forma tan satisfactoria como anteriormente se había logrado sobre el conjunto de train extraído.

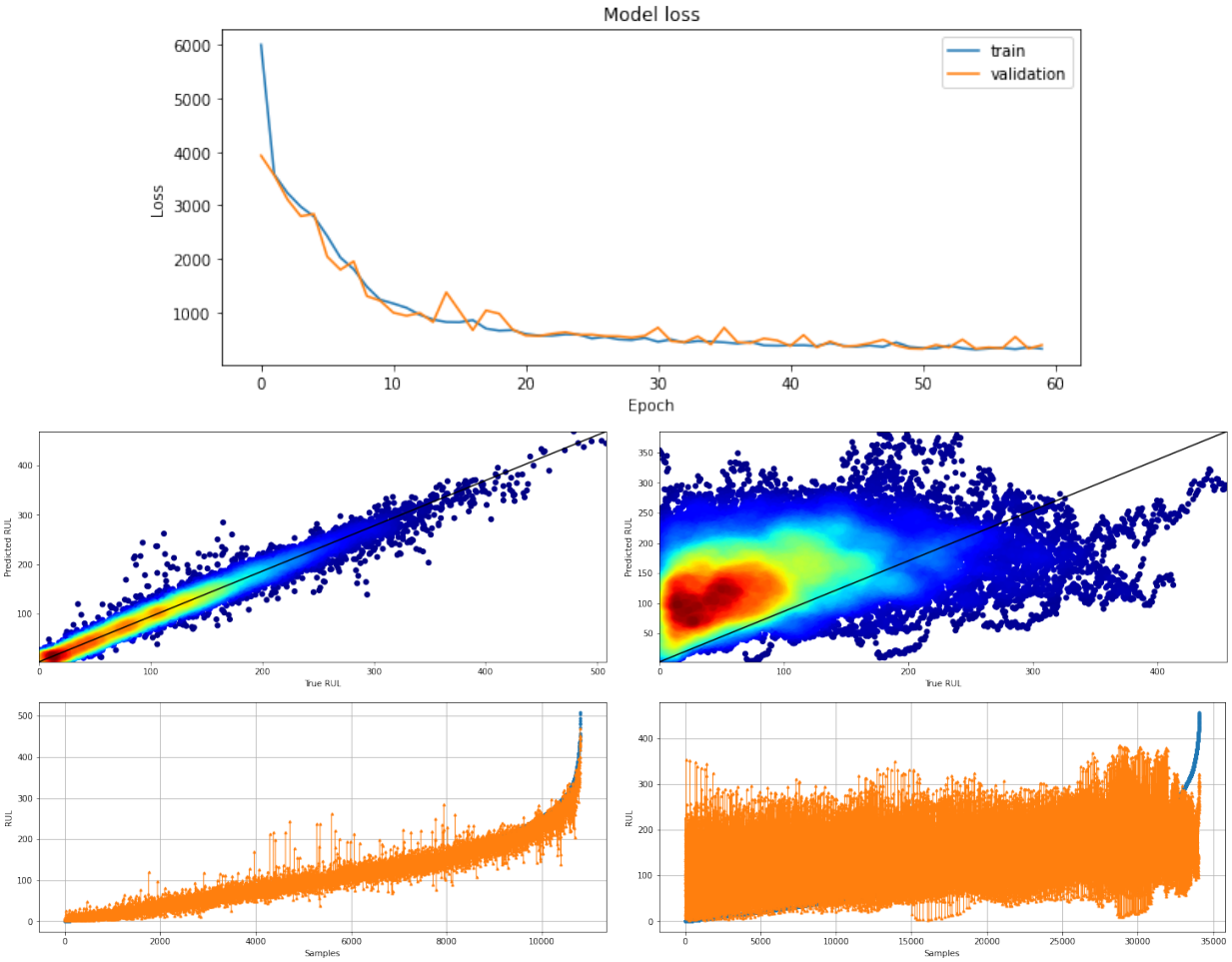

Figura 4: Resultado de entrenamiento de RNN sobre conjunto FD0004.

Los malos resultados (en comparación con los obtenidos por la RNN sobre FD0001) de la red se pueden deber a la morfología de los datos, ya que los histogramas obtenidos (distribución) de los datos, difiere mucho de la de la base de datos FD0001.

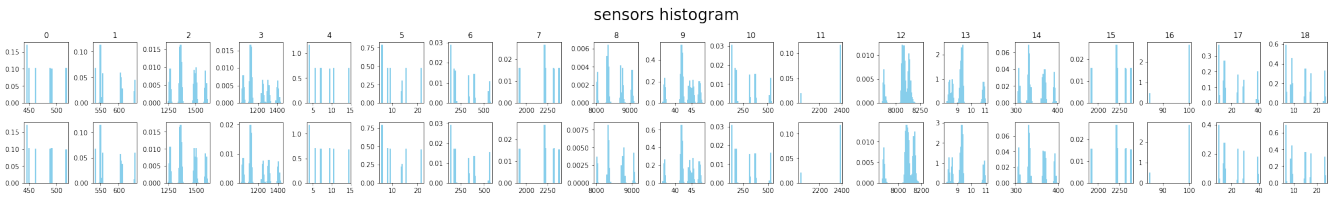

Figura 5: Distribución de los sensores del conjunto FD0004. Fila superior es conjunto de entrenamiento, fila inferior conjunto de test brindado por el cuerpo docente.

Las distribuciones de datos no son tan gaussianas como las del conjunto FD0001, por lo que tienen menor regularidad, lo que podría perturbar el entrenamiento de las redes sobre este conjunto.

4.4. ConvolutionalRNN

El entrenamiento de la red convolucional sobre el conjunto FD0001, da resultados sobre el conjunto de test extraído de 8.32, y sobre el conjunto de test original 72.77. Muy similar al obtenido por la RNN sobre el conjunto FD0001. Mientras que el entrenamiento demora ligeramente más tiempo, ya que se considera un proceso de convoluciones previo a las RNN. No se observa una mejora respecto a la RNN sin convolución, sin embargo, sus resultados son un poco más regulares, donde la RNN simple obtiene resultados desde 6 a 9, agregando convoluciones se obtienen siempre entre 7.

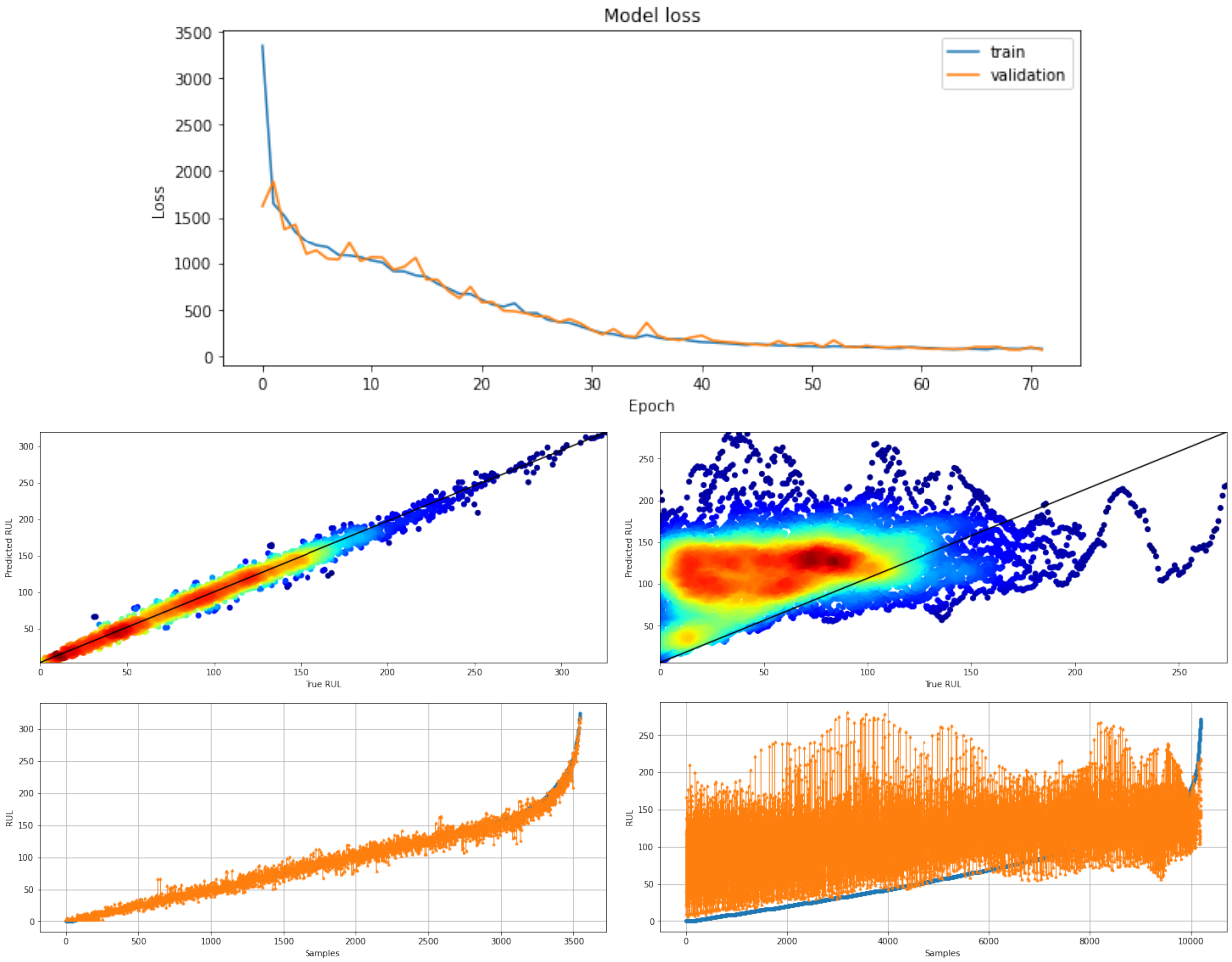

Figura 6: Resultado de entrenamiento de RNN con convoluciones sobre conjunto FD0001.

5. Conclusión

Observando los resultados obtenidos por las distintas arquitecturas de redes sobre el conjunto de test extraído y el conjunto de test original. Se puede aprender que redes de alta complejidad no pueden dar muy buenos resultados si es que no se tiene un conocimiento acabado de los conjuntos de datos a trabajar. Esto ya que sobre el conjunto de test original se obtienen resultados paupérrimos en comparación a los datos que son extraídos del conjunto de entrenamiento para test (que la red no ve durante la fase de entrenamiento).

Esto se puede deber a ciertos factores de anotación de las etiquetas de la RUL, a diferencias en las correlaciones de los datos entre el conjunto de entrenamiento y test. Estas diferencias en la correlación pueden cambiar significativamente los resultados, ya que las redes neuronales no pueden inferir relaciones que no existen en los datos de entrenamiento. De mejor forma explicado, es que las redes neuronales explotan las correlaciones entre los datos de entrada y la salida. De forma que si se pierden las relaciones aprendidas en los nuevos datos vistos, cualquier cosa que salga de la red sobre este nuevo conjunto, no tendrá sentido. Pudo haber pasado que los datos de test fueron sacados desde otra máquina, desde otro estado de la máquina, equivocaciones en la anotación, etc.

El punto anterior tiene sentido, ya que se realizan experimentos sobre un conjunto de datos que tampoco ve la red, y se obtienen buenos resultados. Podría deberse a que son extraídos desde el mismo set de datos que con el que se entrena, sin embargo, el test extraído solamente se utiliza para test, no para entrenar la red.

Sin embargo, y considerando todo lo anterior, muchas veces una red neuronal no será la respuesta a un problema de estimación. Existiendo soluciones mucho más simples que puedan resolver el problema de forma efectiva.