Implementación de un modelo LSTM para operaciones de control y fiscalización de la Armada Chilena

Pablo Jerez
pjerez2016@udec.cl

Gabriel Briceño
gbriceno2016@udec.cl

1 Introducción

La Armada de Chile realiza periódicamente operaciones de fiscalización y vigilancia oceánica sobre zonas económicas exclusivas del mar terrritorial chileno, con el objetivo de lidiar con la pesca no declarada no reglamentada y salvaguardar los ecosistemas marinos, garantizando la conservación de éstos en el largo plazo y el uso sostenido de los recursos pesqueros. Es importante señalar que este tipo de operaciones de control y fiscalización oceánica implican un gran esfuerzo institucional en cuanto a recursos materiales y de personal.

Por lo tanto, el presente artículo propone la implementación de un modelo LSTM para entregar regiones probables de localización de buques durante las operaciones de control y fiscalización de la Armada Chilena, buscando minimizar los costos de sus operaciones.

Para evaluar los resultados del modelo propuesto, se comparará la red LSTM con un modelo CNN.

2 Materiales y metodología

El data set adquirido está compuesto por las multiples trayectorias que las embarcaciones han cruzadoe el canal del Chacao. Cada instancia extraida contiene información relevente para la predicción de cada una de las embarcaciones, tales como su id, la coordenada en el eje de las abcisas y ordenadas, su velocidad actual y su ángulo de navegación. En síntesis, se extraen 166,669 instancias compuestas por matrices de 10x4.

2.1 Set de entrenamiento y set de testeo

Se crean el grupo de entrenamiento y el de prueba (80% y 20% respectivamente). Se crea una muestra representativa correspondiente al 5% del dataset original para entrenar y testear hiperparámetros de las arquitecturas a analizar. En la fase final se ocupa el 100% del dataset para evaluar resultados de las arquitecturas.

2.2 Definición de Hipermodelos e hiperparámetros

Se realizan pruebas para predefinir capas y estructuras a tunear en hipermodelos con Keras Tuner. Con esta prueba también se reduce el espacio de búsqueda de los hiperparámetros testeando modelos de prueba parametrizados.

Luego, se implementan los hipermodelos para las dos arquitecturas a comparar: LSTM y CNN. Se utiliza Keras Tuner para facilitar un espacio de búsqueda y encontrar los mejores valores de hiperparámetros. Se definen los algoritmos de optimización bayesiana, hiperbanda y búsqueda aleatoria. Los hiperparámetros a prueba son los siguientes:

Hiperparámetro	LSTM	CNN
Optimizador	{Adam, SGD}	
Learning rate	{ 1e-1, 1e-2, 1e-3}	
Batch size	{1024, 2048, 4096}	
Embeding	{64, 128, 256}	{ - }
RNN units	{124,256}	{ - }

Table 1: Hiperparámetros explorados para cada Arquitectura.

Ambos grupos de 80% y 20% se dividen en 5 pliegues para utilizarlos en 5-Fold cross validation en los posteriores modelos de entrenamiento y prueba.

Para evaluar el desempeño de las arquitecturas, se analiza las métricas de función de pérdida MSE y RMSE. Se utiliza MSE y RMSE para representar la distancia euclidiana entre los puntos predichos y reales.

2.3 Optimización de Hiperparámetros con validación cruzada

Para cada batch size explorado (1024, 2048, 4096) se probaron múltiples combinaciones de hiperparámetros para cada una de las arquitecturas. Luego de encontrar la mejor configuración de hiperparámetros para cada batch size, se reentrenó usando 5-Fold cross validation para obtener el mejor batch size en base al loss por cada una de las arquitecturas. Finalmente se itera usando 5-Fold cross validation en el mismo modelo, es decir, se iteró 5 veces hasta encontrar el modelo que encuentre la mejor semilla presentando el mejor Loss.

3 Experimentación y Resultados

3.1 Métricas en set de entrenamiento

Loss	LSTM	
	mean	std
MSE train	1.30e-4	±3.64e-5
MSE val	1.26e-4	±3.98e-5
Real MSE train	139,381,944	±572,343
Real MSE val	139,409,678	±1,737,603
Real RMSE train	11,805	±24.22
Real RMSE val	11,806	±73.83

Table 2: Resultados métricas de entrenamiento de loss para arquitectura LSTM en 5% del dataset.

Loss	CNN	
	mean	std
MSE	6.99e-4	±3.06e-4
MSE	7.01e-4	±2.99e-4
Real MSE train	139,491,844	±543,418
Real MSE val	139,311,200	±1,892,354
Real RMSE train	11,811	±22.98
Real RMSE val	11,803	±80.43

Table 3: Resultados métricas de entrenamiento de loss para arquitectura CNN en 5% del Dataset.

Finalmente los hiperparámetros escogidos para ambas arquitecturas se muestran en la siguiente tabla:

Hiperparámetro	LSTM	CNN
Optimizador	{Adam}	{Adam}
Learning rate	{1e-3}	{1e-2}
Batch size	{1024}	{1024}
Embeding	{256}	{ - }
Capas LSTM\ RNN units	{2, 256}	{ - }
Capas conv.\ neuronas	{ - }	{1, 10}
Capas densas \ neuronas	{1}	{ - }

Table 4: Configuración final para cada arquitectura a analizar.

3.2 Métricas en set de testeo

Loss	LSTM	CNN
MSE	1.15e-4	1.24e-3
Real MSE	45,995	693,525.64
Real RMSE	214.46	832.78

Table 5: Resultados métricas de testeo de loss para arquitectura LSTM y CNN en 5% del dataset.

Loss	LSTM	CNN
MSE	2.92e-5	9.39e-5
Real MSE	8912.15	39,195.27
Real RMSE	94.04	197.97

Table 6: Resultados métricas de testeo de loss para arquitectura LSTM y CNN en 100% del dataset.

De los resultados obtenidos, se evidencia un mayor desempeño por parte de la arquitectura LSTM con un RMSE de 94.04 a diferencia del RMSE de 197.97 de la arquitectura convolucional.

4 Discusión

Una vez reescalado los valores normalizados de la prediccón, muchas instancias revelaron una diferencia significativa entre la ubicación predicha de las embarcaciones y la real. A su vez, hay algunas instancias que no tenían una trayectoria clara y presentaban una aparente mala predicción, presentando un valor loss de predicción menor que aquellas instancias que si presentan trayectorias claras con una aparente buena predicción, pero un valor loss mayor que el de los otros casos. Asimismo se puede dar el caso contrario, y es por eso que debe tomarse en cuenta el contexto y el entorno al cual están sujetos los puntos analizados ya que los puntos recorridos pueden estar acumulados en un área concreta o pueden describir una trayectoria clara, lo que variará en la futura predicción del modelo. Estos casos se pueden apreciar mejor en el Anexo.

Estos resultados reflejan que mientras más clara sea la trayectoria de la embarcación, es más problable que el modelo prediga mejor. Por el contrario, si la trayectoria se encuentra en un espacio reducido con una tendencia poco clara, es muy problable que la predicción se encuentre dentro de esa área pero de forma aleatoria.

5 Conclusión

La red LSTM presentó mejor rendimiento que la red CNN, evidenciando un RSME de 94.04 menor que el RSME de 197.97 obtenido por CNN. Esto prueba la eficacia de las arquitecturas de redes neuronales recurrentes para tratar este tipo de problemas, ya que almacenan información a medida que estas se van desenvolviendo en el tiempo. Cabe destacar que por la naturaleza secuencial de la arquitectura LSTM y por la simplicidad de la arquitectura CNN implementada, el tiempo de entrenamiento de la LSTM es significativamente mayor con respecto al de la arquitectura convolucional.

Si bien el loss es pequeño, una vez vuelto a la normalidad la dimensión de los datos, hay instancias que presentan grandes unidades de distancias entre el valor predicho y el real, por lo que aún existe espacio de mejora en la presición de la predicción.

6 Anexos

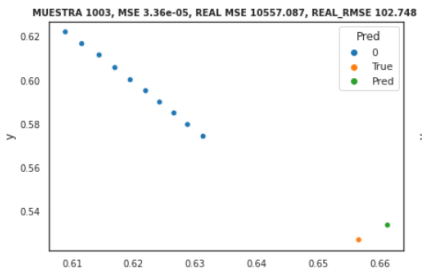

Figure 1: Predicción instancia 1003 LSTM
Fuente: Elaboración propia

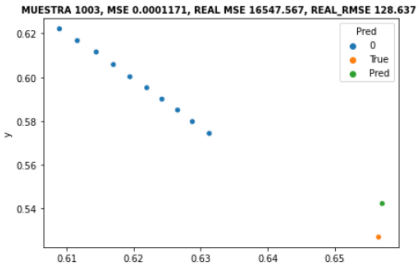

Figure 2: Predicción instancia 1003 CNN
Fuente: Elaboración propia

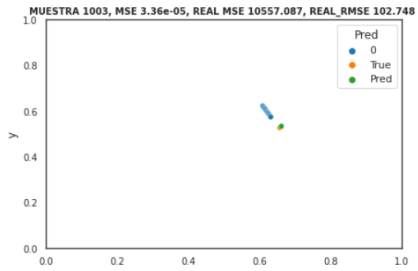

Figure 3: Contexto instancia 1003 RNN
Fuente: Elaboración propia

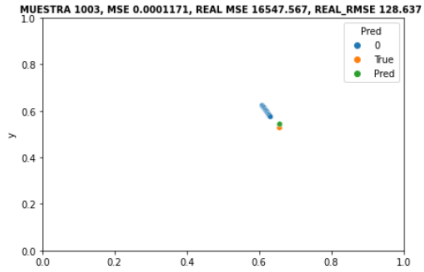

Figure 4: Contexto instancia 1003 CNN
Fuente: Elaboración propia

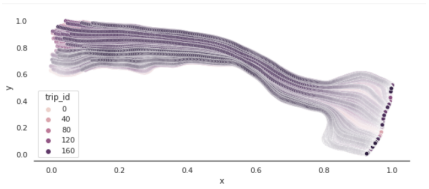

Figure 5: Trayectorias completas de embarcaciones
Fuente: Elaboración propia

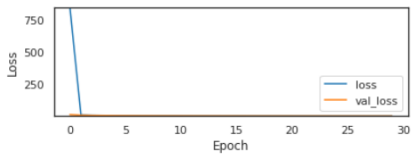

Figure 6: Resultado entrenamiento RNN aumentado x10,000
Fuente: Elaboración propia

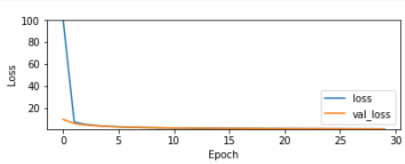

Figure 7: Resultado entrenamiento CNN aumentado x10,000
Fuente: Elaboración propia