Una primera aproximación a la predicción de variables turísticas con Deep Learning

Daniel Trujillo, Antonio Jesús Rivera, Francisco Charte, María José del Jesus
Instituto Andaluz de Investigación en Ciencia de Datos e Inteligencia Computacional (DaSCI), Departamento de Informática
Universidad de Jaén
Jaén, España
{dtviedma,arivera,fcharte,mjjesus}@ujaen.es

***Resumen*—El turismo es una de las actividades económicas más importantes a nivel mundial, por lo que una correcta planificación de los recursos existentes en función de la demanda es fundamental. En este sentido, el trabajo desarrollado permite comparar la bondad de un nuevo modelo de *deep learning*, LSTM, frente a un modelo clásico ampliamente reconocido, ARIMA. Se ha llevado a cabo un proceso de entrenamiento para obtener los modelos LSTM y ARIMA que, posteriormente se han validado utilizando datos no disponibles durante el aprendizaje.**

Nuestros resultados muestran que los nuevos modelos LSTM obtienen una precisión mayor que el clásico ARIMA, tanto en la validación a priori como en la predicción posterior.

***Palabras clave*—LSTM; ARIMA; time series forecasting**

I. Introducción

La industria del turismo a nivel mundial se considera una de las actividades económicas más importantes, sólo superado por la industria del petróleo y productos derivados [2], según algunos autores. En el caso español, al sector turístico se vinculan un 12,1% del total de empleos en 2013, creciendo desde 9,8% en 2001 [3]. En una actividad tan importante es fundamental una correcta planificación, de manera que, por un lado, se pueda atender adecuadamente a la demanda, y por otro, se pueda realizar el mantenimiento del patrimonio que da lugar a la actividad turística.

Por su naturaleza, de este tipo de actividades emanan conjuntos de datos con una fuerte dependencia temporal, lo que también se conoce como **series temporales** [**?**]. En estos conjuntos de datos, las instancias aparecen ordenadas por un criterio cronológico, al tiempo que la variable de estudio presenta una fuerte dependencia cronológica. Ejemplos de este tipo de conjuntos de datos son la evolución del precio de un bien, del número de pasajeros de un aeropuerto, o del grado de ocupación hotelera en una provincia concreta.

Tradicionalmente, el método que se ha utilizado para analizar este tipo de conjuntos de datos ha sido ARIMA [4]. Este modelo integra análisis de medias móviles y análisis autoregresivo, conocido como modelo ARMA, y lo generaliza para series diferenciadas. Este modelo, junto con las estrategias heurísticas desarrolladas para encontrar sus mejores parámetros, tiene una muy importante fundamentación estadística y se le considera como el estado del arte en el ámbito de predicción de series temporales.

Por su parte, las técnicas de *deep learning* han supuesto una revolución importante en el campo del aprendizaje automático, gracias a evoluciones en los algoritmos de entrenamiento disponibles, así como el aumento de la potencia de cómputo de los ordenadores actuales. Esta combinación de factores posibilitan el uso de estos modelos capaces de reproducir procesos cognitivos complejos reservados a los humanos. Modelos de *deep learning* han conseguido resultados sorprendentes en tareas como el reconocimiento de escritura [5], lo cual los hace prometedores sucesores de los algoritmos tradicionales de aprendizaje automático.

Uno de estos modelos, LSTM (*Long Short Term Memory*), está especialmente diseñado para tareas de análisis de series temporal. Su estructura le permite almacenar conocimiento útil para razonar sobre una serie temporal con un contexto mayor que las técnicas tradicionales, estableciendo una nueva línea de estudio que quizás termine por suceder al actual estado del arte, el modelo ARIMA.

El objeto de este estudio es establecer una comparativa entre el algoritmo más comúnmente usado en predicción de series temporales, ARIMA, y las redes neuronales de tipo LSTM, en un problema real de predicción de datos turísticos de interés para la provincia de Jaén.

Este artículo se estructura de la siguiente forma: En la segunda sección se aportará una breve introducción de los métodos considerados en el estudio. A continuación, se detallará la experimentación llevada a cabo para establecer la comparativa, tras lo cual se mostrarán los resultados obtenidos para finalizar con una discusión de los mismos.

II. Métodos

En esta sección se aporta una descripción a nivel introductorio de los métodos que utilizaremos para llevar a cabo la experimentación, LSTM y ARIMA. Al no ser objetivo de este estudio proporcionar una descripción suficientemente detallada, se aportan las referencias más relevantes en cada caso.

Universidad de Jaén.

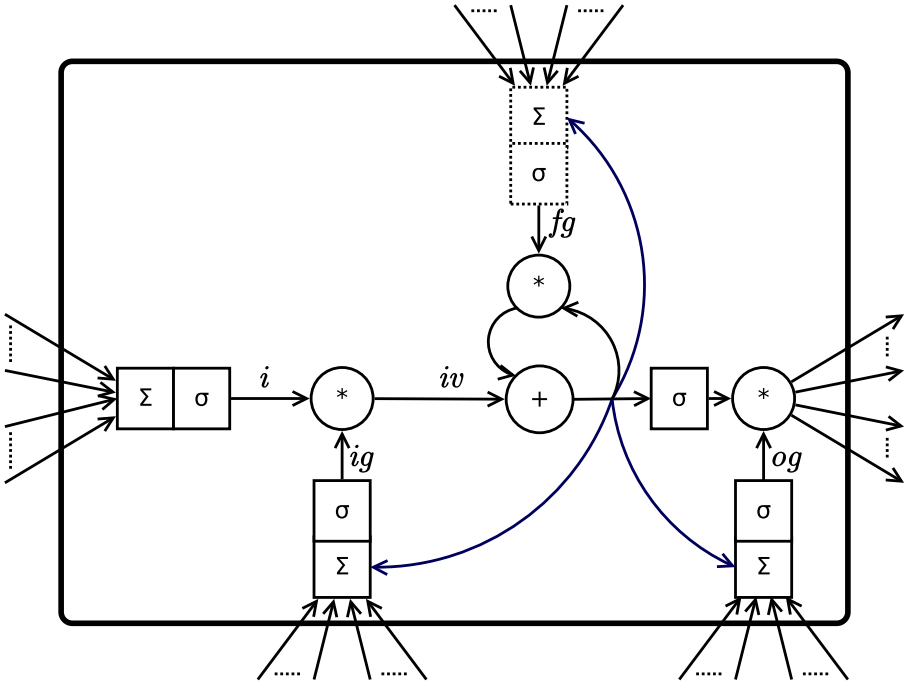

Fig. 1. Representación esquemática de una neurona tipo LSTM.

A. LSTM

Las redes neuronales de tipo LSTM surgen como respuesta a los diversos problemas en torno al flujo del gradiente que sufren los algoritmos de aprendizaje basados en gradiente de redes neuronales artificiales, los cuales se discuten ampliamente en [6], siendo el más recurrente en este tipo de redes el llamado ”vanishing gradient”. Este problema hace referencia al decreciente efecto de las variaciones de un parámetro en la salida final de la red, en la medida en la que el parámetro se aleja de la salida de la red. Esto afecta a las capas de entrada de redes prealimentadas con un número elevado de capas, pero también a redes neuronales recurrentes con amplias dependencias temporales en los datos de entrada.

Las redes neuronales de tipo LSTM han supuesto un importante impulso para las redes neuronales recurrentes, ya que, por los motivos expuestos, los algoritmos de entrenamiento de redes neuronales recurrentes más utilizados no ofrecían ninguna ventaja práctica frente a los mismos algoritmos en redes neuronales prealimentadas [6].

Long Short Term Memory es una arquitectura de neurona artificial cuya estructura le permite mantener un estado interno formado a partir de las evaluaciones que se han realizado, y que influye en las posteriores. Esta estructura le permite memorizar conocimiento que utilizará y actualizará en evaluaciones posteriores.

En su formulación original [7], esta célula define un flujo de información a través del cual, los valores de entrada se modifican mediante operaciones multiplicativas y funciones de activación para obtener su salida.

Sin embargo, posteriores modificaciones a la estructura original se han incorporado de facto en esta formulación, dando lugar a la estructura LSTM más utilizada en la literatura actual, que consta de los siguientes elementos:

- *Input*, *input gate*, *forget gate* y *output gate*: unidades simples, que primeramente aplican pesos a los valores de entrada, a los que se suman el valor de salida de *CEC* resultado de la última evaluación del *CEC* (*peephole connections*), y posteriormente aplica una función de activación.
- CEC (*Constant Error Carrousel*): situado en la parte central de la neurona, está formado por una autoconexión recurrente. Esta autoconexión constituye la memoria de la neurona, cuyo valor se suma a la entrada del *CEC* y se actualiza en cada evaluación de la neurona a su salida.

En la fig. 1 se detalla el interior de una neurona LSTM. En ella se puede apreciar muy facilmente el flujo de informacion dentro de la neurona, de izquierda a derecha, mientras es transformada por las 3 válvulas de información. Asimismo, se puede ver el CEC, justo en el centro de la neurona.

A continuación se proporciona una descripción textual de la neurona LSTM:

1) Los valores de entrada se suministran a las unidades *input*, *input gate* y *forget gate*, obteniendo los valores *i*, *ig* y *fg*, respectivamente
2) *i* e *ig* se combinan multiplicando sus valores elemento a elemento, obtenido *iv*.
3) *fg* se multiplica, elemento a elemento, con el valor de salida del *CEC* en el instante de tiempo anterior, obteniendo el nuevo valor de entrada de *CEC*.
4) Este valor de entrada de *CEC* se suma con *iv*, elemento a elemento, obteniendo la salida de *CEC*. Ésta, además de continuar por el flujo de información, se copia a

las *input*, *forget* y *output gate* a través de las *peephole connections*.

5) Se calcula la salida del *output gate*, *og*, dados los valores originales de entrada y el nuevo valor de salida del *CEC* a través de las peephole connection.
6) Por último, la salida de *CEC* se suministra a una función de activación, cuyo resultado se multiplica elemento a elemento con *og* para obtener la salida final de toda la neurona LSTM.

Esta estructura interna está diseñada para trabajar con series de datos en intervalos mucho más amplios, ya que se asegura, junto con un entrenamiento adecuado, generalmente mediante backpropagation, un flujo de error constante en el *CEC*, mientras que la *input gate* protege el valor recordado en el *CEC* de entradas irrelevantes, y la *output gate* protege a otras células de valores irrelevantes en el *CEC*.

Esta descripción de la estructura estándar es producto, como se ha mencionado antes, de modificaciones sobre la estructura original que se define en [7]. Dichas modificaciones incluyen:

- *Forget gate*: Previamente a su introducción en [8], el valor de salida del *CEC* simplemente se copia a la entrada del *CEC*, mientras que este valor, agregado a los datos de entrada, se suministran a una función de activación. Tras su introducción, se elimina la función de activación, y la salida del *CEC* pasa a ser simplemente la agregación del valor anterior del *CEC* a los datos de entrada.
- *Peephole connections*: Son conexiones con peso desde la salida del *CEC* hasta las *input gate* y *forget gate*, donde se usarán en la siguiente evaluación; y hasta la *output gate*, donde se usará inmediatamente para calcular el valor *og* y la salida final de la LSTM.

B. ARIMA

ARIMA [4] es un modelo ampliamente utilizado en la literatura, por su especial bondad en tareas de predicción de series temporales. El funcionamiento general de este modelo consiste en aproximar la serie temporal mediante una función matemática que pueda ser evaluada en las condiciones que se desean predecir. Esto significa que ARIMA no sólo es útil en tareas predictivas, sino que el ajuste que realiza de la serie temporal puede ser analizado desde un punto de vista descriptivo.

ARIMA se construye a partir de otro modelo, llamado ARMA, generalizándolo para series temporales que han sido diferenciadas. Esto es, una serie temporal obtenida como las diferencias entre cada elemento y el anterior de la serie temporal inicial. El nombre de ARMA proviene de las iniciales *AutoRegressive and Moving Averages*, y consiste en encontrar, primeramente un modelo autoregresivo sobre las diferencias de los valores a predecir, y posteriormente modelar el error de la anterior regresión a través de un modelo de medias móviles.

III. Experimentación

Para conocer el desempeño de los algoritmos se han realizado ejecuciones de ambos sobre un conjunto de datos real,

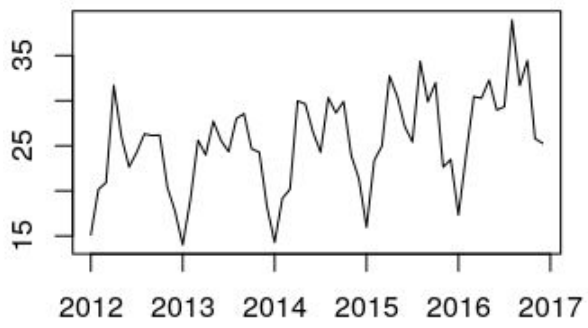

Fig. 2. Visualización de la serie temporal.

obteniendo para cada una de las ejecuciones unas métricas de error que serán objeto de comparación más adelante.

Con el objeto de hacer la comparativa lo más justa posible, el criterio que se ha utilizado a la hora de seleccionar los mejores parámetros del modelo LSTM es la precisión de predicción en el conjunto de entrenamiento. Tras seleccionar los mejores conjuntos de parámetros según este criterio, se ha procedido a reevaluar los modelos en el conjunto de test. De este modo el conjunto de test no interviene en el entrenamiento, como es natural, pero tampoco interviene en la selección del modelo, simulando con mayor precisión un escenario real cuyo objetivo sea el de predecir valores futuros.

A. Métricas de error

En la elaboración de esta experimentación se han considerado 2 métricas de error distintas: La raíz del error cuadrático medio (*rmse*); y el error porcentual absoluto medio (*mape*).

Las métricas de error cuantifican el desvío de una muestra (en este caso, representa la predicción realizada por nuestros modelos) con respecto de otra (los valores reales para la serie temporal).

Siendo E el vector de valores esperados y P el vector de valores predichos, teniendo E y P el mismo número de valores, ambas métricas se definen como sigue:

1) RMSE:

$$RMSE(E,P) = \sqrt{MSE(E,P)} \tag{1}$$

$$MSE(E,P) = \frac{\sum_{i=1}^{n}(E_i - P_i)^2}{n} \tag{2}$$

2) MAPE:

$$MAPE(E,P) = \frac{100}{n}\sum_{i=1}^{n}\frac{E_i - P_i}{E_i} \tag{3}$$

B. Conjunto de datos

Los datos empleados se han obtenido del repositorio público de Instituto Nacional de Estadística. En concreto, se han utilizado los relativos a la Encuesta de Ocupación Hotelera, serie correspondiente al grado de ocupación por plazas de la provincia de Jaén. Estos datos han sido agrupados por la fuente en intervalos de tiempo de un mes, siendo este intervalo la resolución de la que disponemos. En la fig. 2 se puede ver una representación de los datos obtenidos.

Como cabe esperar de una serie temporal relacionada con el sector turístico, se puede apreciar a simple vista una periodicidad de 12 meses en los valores, además de picos de ocupación en los meses de marzo-abril y julio-agosto, correspondiente con los períodos vacacionales de semana santa y verano.

De todos los datos disponibles, se han utilizado los comprendidos entre enero de 2012 y diciembre de 2016. Posteriormente, se han construido 2 conjuntos de datos: El primero de ellos es el conjunto de entrenamiento, y está formado por los datos comprendidos entre enero de 2013 y diciembre de 2015. El resto de datos, los comprendidos entre enero y diciembre de 2016 compondrán el conjunto de datos de test.

A las series temporales extraídas se les han añadido regresores externos adicionales para aumentar la eficiencia de los métodos. Estas columnas adicionales se han decidido con cuidado de no introducir incertidumbre en el conjunto de datos, considerando sólo aquellos regresores que se puedan obtener de forma precisa de antemano. En concreto, se ha añadido información sobre:

- Año
- Mes del año
- Semana santa: Indica si el mes en cuestión contiene parte de la semana santa de ese año.
- Puentes largos: Número de festivos en días martes o jueves en el mes dado.
- Puentes cortos: Número de festivos en días lunes o viernes en el mes dado.

Si bien esta es la estructura general de los conjuntos de datos de partida, por exigencias de cada implementación la estructura se ha adaptado en cada caso:

LSTM: Cada instancia suministrada a la red durante el entrenamiento tiene la forma $(V_n, V_{n-1}, V_{n-11}, V_{n-12}, SS, PC, PL)$, mientras que para la predicción se elimina el valor V_n por motivos obvios.

ARIMA: En el caso de ARIMA, cada instancia tiene la forma $(V_n, V_{n-12}, SS, PC, PL)$

Siendo:

n El mes correspondiente a la instancia en cuestión.
V_i El valor de la serie temporal correspondiente al mes i-ésimo.
SS Indica si en ese mes tiene lugar la Semana Santa.
PC El número de puentes cortos del mes correspondiente.
PL El número de puentes largos del mes correspondiente.

En el caso de la LSTM, se ha optado por añadir el valor de la serie temporal de mes anterior, del año anterior, y del undécimo mes anterior. Esta selección de retardos obedece a los resultados de un análisis preliminar de autocorrelación de la serie temporal, fig. 3, que muestra que estos valores son los más significativos.

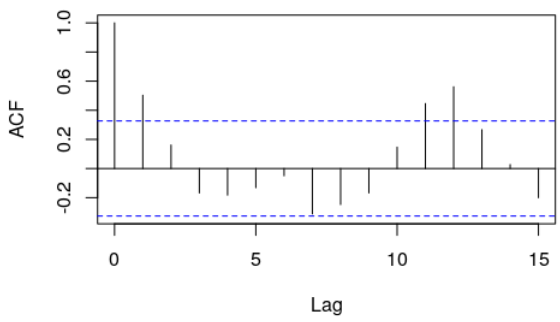

Fig. 3. Función de autocorrelación.

C. Métodos

La experimentación con las redes neuronales recurrentes de tipo LSTM se ha implementado en *Python*, utilizando la librería *Keras* con el backend *TensorFlow*, configurado por defecto. Asimismo, se ha utilizado la librería *Pandas* para la gestión de los datos, *Numpy* para el cálculo numérico, *Matplotlib* para la visualización de los resultados y *Scikit-learn* para el cálculo de métricas de error, así como la normalización de los datos de entrada.

A pesar de que las redes neuronales recurrentes LSTM son un modelo de tipo *deep learning*, el tiempo de entrenamiento por modelo en esta experimentación no ha sido excesivo.

Una vez realizadas ejecuciones preliminares explorando distintas zonas del espacio de configuraciones de parámetros, a la vista de las métricas de error obtenidas, se ha procedido a seleccionar la región del espacio donde encontraremos los modelos que calculen las mejores predicciones:

- Número de LSTM: 25, 50 y 100
- Iteraciones: 100 y 200
- *batch_size*: 1 y 2

Dado que el algoritmo utilizado para el entrenamiento de las redes LSTM, BackPropagation Through Time, parte de un estado inicial aleatorio, para reducir el efecto del azar de los elementos del estado inicial de la red, se han realizado 20 ejecuciones con cada una de las combinaciones de parámetros del espacio de búsqueda reducido definido anteriormente. De esta forma el error medio obtenido está muy escasamente influenciado por algunas ejecuciones cuyos resultados pueden verse desviados por motivos del azar.

Para el entrenamiento del modelo ARIMA se ha utilizado el entorno R de computación estadística, en lugar del lenguaje Python, como se ha hecho anteriormente. Esto es debido a que la implementación de referencia del modelo ARIMA está desarrollada en este lenguaje. En concreto, se ha utilizado la función *auto.arima* del paquete *forecast* [4].

El programa de entrenamiento del modelo ARIMA es visiblemente más simple que en el caso de la red LSTM. Esto es debido a que todo el análisis de parámetros y entrenamiento del modelo se realiza de forma automática por la implementación elegida, utilizando técnicas como las descritas en los trabajos [10], [11]. La existencia de distintas heurísticas

TABLA I
Resultados ejecuciones LSTM - RMSE Entrenamiento

Num LSTM	Iterac.	batch size	RMSE	
			Promedio	Desv. Est.
25	200	1	2,17954	0,01534
50	200	1	2,18941	0,01802
25	200	2	2,19675	0,01309
50	200	2	2,20690	0,00849
100	200	1	2,21396	0,03308
100	200	2	2,22174	0,01000
25	100	1	2,22621	0,01257
50	100	1	2,23581	0,01208
25	100	2	2,23769	0,01105
100	100	1	2,24942	0,02027
50	100	2	2,24990	0,01107
100	100	2	2,27130	0,01539

TABLA II
Resultado ejecución ARIMA

Entrenamiento	Test	
RMSE	RMSE	MAPE
2,40561	1,89010	5,01111

hace posible que este proceso se pueda automatizar teniendo en cuenta métricas comparativas como *Akaike information criterion* [12]

Además, dado que ni ARIMA ni las heurísticas que buscan los mejores parámetros son probabilísticos, no es necesario realizar varias ejecuciones del algoritmo, ya que todas arrojarían exactamente los mismos resultados.

IV. Resultados

En la tabla I se pueden ver los resultados de las mejores ejecuciones conseguidas con redes neuronales de tipo LSTM. En ella se muestra, para la métrica de error RMSE en la partición de entrenamiento del conjunto de datos, el promedio y la desviación típica de 20 repeticiones completas del experimento. Los distintos modelos se muetran en la tabla ordenados según el error cometido en la experimentación. A la vista de estos resultados, seleccionamos el modelo de menor error para la comparativa con ARIMA.

En la tabla II se muestra el resultado de la ejecución realizada con el modelo ARIMA. Como se ha introducido antes, dado que este modelo no es probabilístico, carece de sentido realizar varias repeticiones, ya que todas predicen exactamente igual, por lo que las métricas obtenidas serán idénticas.

La tabla comparativa III se muestran los resultados obtenidos en la partición de test, tanto para el modelo ARIMA como para el mejor modelo LSTM, el cual se ha seleccionado siguiendo el criterio del menor error en la partición de entrenamiento (tabla I). Es fácil observar de esta forma que LSTM obtiene mejores resultados que ARIMA en ambas métricas de error consideradas.

V. Conclusión

En este artículo se ha realizado una comparativa entre un nuevo modelo de red neuronal recurrente, LSTM, y el algoritmo considerado estado del arte en predicción de series temporales, ARIMA, sobre un problema de predicción de series temporales en turismo, justificado por la importancia del sector turístico en la economía mundial.

TABLA III
Comparativa ARIMA - LSTM

	RMSE	MAPE
LSTM	**1,70039**	**4,61407**
ARIMA	1,89010	5,01111

Los resultados de la experimentación llevada a cabo, resumidos en la tabla III, indican que un modelo de red neuronal de tipo LSTM, seleccionado basándonos exclusivamente en datos de entrenamiento, obtiene una mayor precisión en la predicción de los datos de test que el modelo ARIMA clásico, ampliamente utilizado en el campo de series temporales.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de España de Ciencia y Tecnología bajo el proyecto TIN2015-68454-R.

Referencias

[1] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Advances in neural information processing systems*, 1997, pp. 473–479.
[2] C. Altés, "Marketing y turismo," *Madrid: Editorial Síntesis*, 1993.
[3] J. R. Cuadrado Roura, J. M. López Morales *et al.*, "El turismo, motor del crecimiento y de la recuperación de la economía española," 2015.
[4] R. J. Hyndman, Y. Khandakar *et al.*, *Automatic time series for forecasting: the forecast package for R*. Monash University, Department of Econometrics and Business Statistics, 2007, no. 6/07.
[5] V. V. Romanjuk, "Training data expansion and boosting of convolutional neural networks for reducing the mnist dataset error rate," *Naukovi Visti NTUU KPI*, no. 6, pp. 29–34, 2016.
[6] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.
[7] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.
[8] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.
[9] J. Brownlee, "Multivariate time series forecasting with lstms in keras-machine learning mastery," *Machine Learning Mastery*, 2017.
[10] E. J. Hannan and J. Rissanen, "Recursive estimation of mixed autoregressive-moving average order," *Biometrika*, vol. 69, no. 1, pp. 81–94, 1982.
[11] V. Gómez, *Automatic model identification in the presence of missing observations and outliers*. Ministerio de Economía y Hacienda, Dirección General de Análisis y Programación Presupuestaria, 1998.
[12] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected Papers of Hirotugu Akaike*. Springer, 1998, pp. 199–213.
[13] J. E. Ball, D. T. Anderson, and C. S. Chan, "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community," *Journal of Applied Remote Sensing*, vol. 11, no. 4, p. 042609, 2017.