Cuantificación de la incertidumbre en la predicción espacio-temporal con bayesian deep learning: Aplicación a El Niño

Trabajo Fin de Máster:
Máster Universitario en Ciencia de Datos
de la Universidad de Cantabria

Presentada por
Jose González Abad

bajo la dirección de
Jorge Baño Medina

Santander, 3 de Julio de 2019

Resumen

Los modelos de redes neuronales son capaces de modelizar complejas relaciones no lineales subyacentes en los datos. A medida que su campo de aplicación se ha extendido, se han diseñado nuevas arquitecturas enfocadas en la modelización de distintas dinámicas. En los últimos años estos modelos han conseguido alcanzar el estado del arte en multitud de disciplinas entre las que se encuentran el reconocimiento de objetos en imágenes o el procesamiento de lenguaje natural. En clima se han desarrollado algunas aplicaciones con deep learning, donde estos modelos han conseguido adaptarse a sus dinámicas espacio-temporales con éxito. Sin embargo estos modelos carecen de una característica fundamental en la modelización: la cuantificación de la incertidumbre. Recientemente ha surgido un marco teórico conocido como bayesian deep learning con el cual es posible cuantificar esta incertidumbre en los modelos de redes neuronales. En el presente trabajo se realiza un estudio comparativo de modelos de deep learning y otros métodos que representan el estado del arte en la predicción del fenómeno de El Niño. Además se utiliza el bayesian deep learning para la cuantificación de la incertidumbre. Debido a la limitación en la cantidad de datos de esta aplicación, las predicciones obtenidas superan a los modelos lineales pero no alcanzan a otros métodos de machine learning. Sin embargo las redes neuronales proveen de una mejor aproximación a los eventos extremos, por ejemplo el evento anómalo de Diciembre de 2015. Además, la introducción del dropout provee de una estimación de la incertidumbre sencilla y práctica.

Neural network models are able to model complex non-linear relationships underlying the data. These models and their different architectures achieved state-of-the-art in a variety of fields like NLP or visual object recognition. In climate some deep learning based applications have been developed and successfully adapted to its characteristic spatio-temporal dynamics. However, these models lack a fundamental characteristic: the quantification of uncertainty. In this work, deep learning models are compared with other traditional ones in terms of El Niño prediction and its uncertainty quantification. Uncertainty in these neural networks is quantified with a new theoretical framework recently developed known as Bayesian deep learning. Due to the limitation in the amount of data, the obtained predictions surpass the linear models but do not reach the other machine learning methods. However, neural networks provide a better approximation to extreme events (December 2015). In addition, the use of dropout provides a simple and practical uncertainty estimation.

Palabras clave: *Bayesian deep learning, El Niño, Forecasting.*

Índice general

CAPÍTULO 1

Introducción

1.1. Motivación

La complejidad de la realidad en la que vivimos implica que la mayoría de los procesos que ocurren a nuestro alrededor sean el resultado de complicadas interacciones entre los distintos elementos que la conforman. Un ejemplo de este dinamismo caótico puede observarse en la atmósfera, donde la evolución de las variables meteorológicas son consecuencia de mecanismos no lineales entre distintos componentes del sistema climático. La dinámica no lineal no es un problema particular de la meteorología, sino que puede apreciarse en otros ámbitos como por ejemplo en el estudio de la interacción humana o en la evolución de los procesos biológicos. A pesar de las inherentes dificultades, el estudio de los sistemas dinámicos, especialmente de los sistemas no lineales, es una tarea importante y complicada que se lleva abordando desde la Antigüedad.

Tradicionalmente se ha abordado el estudio mediante sistemas físico-matemáticos que modelaban el sistema en cuestión. Volviendo al ejemplo de la atmósfera, en [Charney et al. (1950)] se elaboró el primer modelo de predicción meteorológica que resolvía un sistema de ecuaciones basado en el movimiento de los fluidos. Sin embargo, no siempre es posible conocer las ecuaciones que rigen el comportamiento de un sistema e incluso conociéndolas puede haber cierta incertidumbre asociada. Entre las fuentes de esta incertidumbre se encuentran la imposibilidad de medir con exactitud el estado inicial de un sistema o las limitaciones computacionales en la integración de las ecuaciones diferenciales. La primera es de gran importancia y hace referencia a los sistemas caóticos ya que ligeros cambios en las condiciones iniciales conllevan evoluciones muy distintas en el tiempo. Por otro lado, la falta

de solución analítica en algunos sistemas de ecuaciones diferenciales, como en los usados en meteorología, requiere de métodos numéricos para su integración. El coste computacional asociado hace que la discretización, espacio-temporal en el caso de la meteorología, se realice sobre una malla grosera, contribuyendo a la pérdida de información y por tanto aumentando la incertidumbre del sistema.

Debido a las limitaciones, principalmente computacionales, de los modelos matemáticos, se empezaron a desarrollar modelos estadísticos, los cuales se basan en la inferencia de relaciones existentes de los datos. Las relaciones de algunos procesos tienen una naturaleza lineal por lo que su modelización a través de modelos estadísticos es relativamente sencilla. Los modelos capaces de trabajar con este tipo de relaciones se les conoce como modelos lineales. En estos tipos de modelos la variable dependiente no es más que una combinación lineal de ciertas variables independientes ponderadas por unos coeficentes a estimar. Algunos de los modelos lineales mas utilizados son la regresión lineal, modelos autorregresivos para datos temporales (AR, MA, ARIMA...) y modelos lineales generalizados.

Desgraciadamente la mayor parte de procesos no tienen una naturaleza lineal por lo que este tipo de modelos no conseguirían aprender correctamente los patrones entre las variables. Por ello es necesario recurrir a modelos no lineales como los support vector machines (SVM) o las redes neuronales (NN). La literatura en este caso es muy extensa y se pueden encontrar multitud de aplicaciones exitosas que van desde la clasificación de documentos de texto [Liang (2004)] hasta las aplicaciones en clima [Rhee and Im (2017)].

A pesar del diverso grado de éxito que han tenido los modelos no lineales tradicionales, en la actualidad se está prestando atención a un nuevo campo llamado deep learning (DL) debido a los éxitos que está obteniendo en tareas con una inherente estructura no lineal. En concreto, el deep learning es una evolución de las redes neuronales tradicionales, gracias a la serialización de las capas que permiten extraer patrones más complejos (no lineales) de los datos. Esto es posible gracias a la co-evolución de la infraestructura y de la tecnología sobre la que se apoya el deep learning. Problemas como la sobredimensionalidad de las redes o el desvanecimiento de gradiente [Bengio et al. (1994)] han sido solventados por la aparación de nuevos algoritmos de aprendizaje [LeCun et al. (1988)], nuevas funciones de activación que impiden el desvanecimiento del gradiente [Nair and Hinton (2010)], la introducción del aprendizaje estocástico como el stochastic gradient descent [Kiefer et al. (1952)] y nuevos métodos de regularización como el dropout [Srivastava et al. (2014)].

Además de las dificultades con las que se enfrentaban las redes neuronales tradicionales descritas arriba, uno de los problemas a los que se enfrentaban en tareas de predicción era la cuantificación de la incertidumbre. Uno de los enfoques era a través de la bayesianización de las redes neuronales que estimaban la *posterior predictive*

distribution y por tanto el resultado no era una estimación puntual sino una distribución. Aunque hubo algunos éxitos en los 90 a la hora de diseñar redes neuronales bayesianas, estas redes se limitaban a configuraciones muy simples (1 o 2 capas) y no fue hasta [Gal (2016)] cuando se desarrolló una teoría más completa para bayesianizar cualquier configuración de redes neuronales. El trabajo [Gal (2016)] consistía en añadir dropout [Gal and Ghahramani (2016a)] a las capas de una red neuronal, demostrando matemáticamente que el resultado era una red neuronal bayesiana, derivando en un campo que se conoce como el de bayesian deep learning. Esta técnica se ha aplicado con éxito en áreas como la conducción autónoma [Michelmore et al. (2018)] o scene understanding [Kendall et al. (2015)].

A pesar del éxito cosechado por el bayesian deep learning, aún hay un gran campo de aplicaciones como la meteorología, donde explorar la maquinaria del bayesian deep learning tanto como herramienta predictiva como para cuantificar la incertidumbre. En meteorología, se ha utilizado el deep learning para resolver problemas complejos de naturaleza no lineal como la identificación de problemas extremos en datasets climáticos [Liu et al. (2016)], la predicción de la trayectoria de huracanes [Giffard-Roisin et al. (2018)] o para aumentar la resolución de los modelos climáticos [Vandal et al. (2017)], entre otros.

Por tanto, a pesar de que el deep learning ya ha sobrepasado a métodos tradicionales no lineales en diversas aplicaciones de la meteorología, aún hay ciertas aplicaciones para las cuales la aplicabilidad del deep learning todavía no se ha investigado. En las siguientes secciones mostramos una aplicación inexplorada por el deep learning que consiste en la predicción del índice de El Niño y de la que va a tratar este trabajo.

1.2. El Niño

El Niño y La Niña son dos patrones opuestos de lo que se conoce como ENSO o El Niño-Oscilación del Sur. Unos de los primeros meteórologos en estudiar este fenómeno fue Richard T. Barber el cual lo resumió con la siguiente frase “The ocean is clearly driving the atmosphere”. El Niño se caracteriza por un aumento en las temperaturas de la superficies del Oceano Pacífico mientras que La Niña se corresponde con el efecto contrario de enfriamiento.

En condiciones normales los vientos Eliseos y Aliseos forman un sistema de circulación de aíre que desplaza las aguas templadas del Este hacia el Oeste. Este desplazamiento hace que en los países de América del Sur se de un clima seco y frío mientras que en países como Indonesia o Australia se dan periodos de lluvias. El Niño y La Niña interrumpen estos comportamientos. El Niño provoca que el sistema de circulación se detenga de forma que las aguas ya no se desplacen, haciendo que las aguas calientes terminen en América de Sur provocando periodos intensos de

precipitaciones y las frías vayan al Oeste originando intensos periodos de sequía. La Niña provoca que el comportamiento estandar se intensifique provocando precipitaciones intensas en el Oeste y sequía en el Este.

Debido a su influencia en el clima es de vital importancia conocer su evolución temporal ya que tienen un gran impacto medioambiental y económico. Entre los años 2015 y 2016 se dió con una extrema violencia el fenómeno del ENSO, alcanzando valores de temperatura en el océano anormalmente altos. Ninguno de los modelos predictivos utilizados por los centros meteorológicos fue capaz de estimar con precisión la magnitud de El Niño de ese año [van Oldenborgh et al. (2003)]. Por ello es un ejemplo claro donde no solo la precisión de la predicción es importante sino también la estimación de la incertidumbre asociada a dicha predicción, la cual hubiese permitido analizar la posibilidad de un Niño de tal magnitud para el año 2015-2016.

1.3. Predicción de El Niño

En particular, el problema de la predicción del índice de El Niño es un problema no lineal donde tanto los métodos deterministas como los modelos estadísticos muestran aún deficiencias [Wikle (2015)]. Recientemente, en [McDermott and Wikle (2019)] han evaluado un modelo no lineal recurrente basado en la infraestructura del reservoir computing mostrando mejoras tanto en términos de precisión como de la cuantificación de la incertidumbre con respecto a modelos de referencia.

En este estudio vamos a reproducir el estudio en [McDermott and Wikle (2019)] pero investigando el deep learning como herramienta para la predicción del índice de El Niño y utilizando la maquinaria del bayesian deep learning para construir una estimación sobre la incertidumbre de la predicción. Con el fin de aprovechar la infraestructura del deep learning, se van a probar tanto modelos convolucionales como recurrentes y así evaluar su capacidad para la extracción de patrones espacio-temporales en esta tarea en cuestión.

1.4. Objetivos

Los principales objetivos de este trabajo son:

- Construir modelos no lineales basados en redes neuronales para la predicción del índice de El Niño.
- Comparar los modelos neuronales con respecto a otros modelos de machine learning y con respecto a modelos dinámicos de referencia que suponen el actual estado del arte.
- Evaluar la utilidad del deep learning en la predicción del índice del Niño.

- Aplicar el Bayesian Deep learning por primera vez como método para cuantificar la incertidumbre en un caso de índole climática.
- Aplicar el conocimiento adquirido de deep learning en el máster y extenderlo con el bayesian deep learning

1.5. Estructura

Para lograr estos objetivos el trabajo ha sido estructurado de la siguiente forma. En la sección 2 se presentarán los datos sobre los que se va a trabajar así como su fuente de origen y las modificaciones realizadas sobre ellos. Después se hará un recorrido por la problemática del forecasting y los algoritmos desarrollados para su modelización y predicción dentro del marco teórico del deep learning. En la sección 3 se hará un recorrido por las redes neuronales densas, recurrentes y convolucionales y despúes se extenderán al deep learning. Después introduciremos el bayesian deep learning y veremos cómo y por qué es posible generar intervalos de confianza con el uso del dropout. Una vez introducida la parte teórica explicaremos los modelos ajustados en este trabajo y sus configuraciones así como los de referencia del artículo [McDermott and Wikle (2019)] en la sección 4. También expondremos la metodología seguida durante el entrenamiento de estos. Finalmente presentaremos los resultados del estudio en la sección 5 y extraeremos unas conclusiones.

CAPÍTULO 2

Datos

2.1. Fuente de Datos: NOAA

El problema de forecasting planteado en [McDermott and Wikle (2019)] sobre el que se apoya esta tesis, se basa en inferir modelos empíricos a través de relaciones existentes en los datos oceánicos para la predicción de la temperatura del mar (SST, por sus siglas en inglés). En particular, se utiliza la SST suministrada por la agencia de Administración Nacional Océanica y Atmosférica (NOAA). La NOAA guarda un registro de la temperatura de la superficie oceánica que abarca desde 1970 hasta 2016. Este dataset se conoce bajo el nombre de ERSST (Extendend reconstructed sea surface temperature) y contiene la información de la temperatura oceánica mensual sobre todo el globo a una resolución de 2º x 2º (ver Figura 2.1). Estos datos se pueden descargar de forma totalmente gratuita en el siguiente link: (http://iridl.ldeo.columbia.edu/SOURCES/.NOAA/.NCDC/.ERSST/.version5/). El dataset utilizado en [McDermott and Wikle (2019)], y que por tanto utilizamos en este trabajo, es un dataset derivado del anterior llamado *anomalías ERSST*, el cual puede ser descargado en el siguiente link: (http://iridl.ldeo.columbia.edu/SOURCES/.NOAA/.NCDC/.ERSST/.version5/.anom/). La diferencia radica en que en este último se ha extraído la media mensual (es decir, a todos los Eneros se le ha substraído la media de los Eneros y así sucesivamente). El período climatológico usado para computar la media es el comprendido entre 1981-2010. Las anomalías se expresan en grados centígrados y permiten apreciar con facilidad patrones de temperatura anómalos.

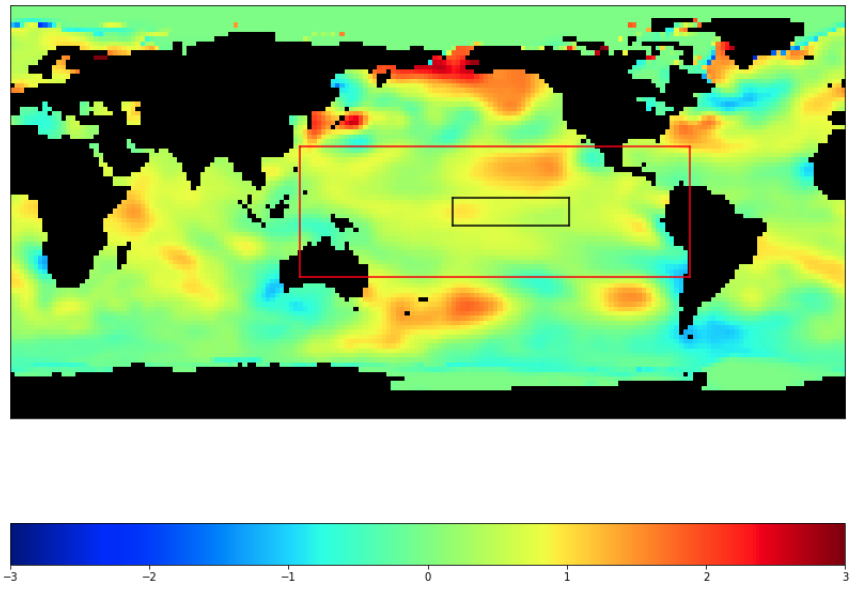

Figura 2.1: Anomalías ERSST correspondientes a Mayo de 2019

2.2. Región de estudio: El Niño 3.4

Se distinguen distintas zonas donde tradicionalmente se ha situado el área geográfica donde se estudia El Niño. En [McDermott and Wikle (2019)], utilizan la SST comprendida entre las latitudes 29S y 29N, y las longitudes 124E y 70W (ver región roja en Figura 2.2) para inferir los modelos (ver sección 4 para más detalles) y luego evalúan la predicción comparándolo con el índice de El Niño 3.4. Dicho índice es la media de la SST sobre la región de El Niño 3.4 que se sitúa entre las latitudes 5S y 5N, y las longitudes 170W y 120W (cuadrado negro de la Figura 2.1). En la Figura 2.2 se puede observar la variabilidad del índice de El Niño 3.4 y ejemplos de patrones espacio-temporales asociados asociados a situaciones de Niño (anomalía de SST positiva) o de Niña (anomalía de SST negativa).

En este estudio utilizamos las anomalías mensuales de temperatura del mar de la zona de El Niño 3.4 tanto como variable explicativa como variable de respuesta, donde el objetivo, al igual que en [McDermott and Wikle (2019)], será predecir la temperatura en la región de El Niño 3.4 con 6 meses de adelanto.

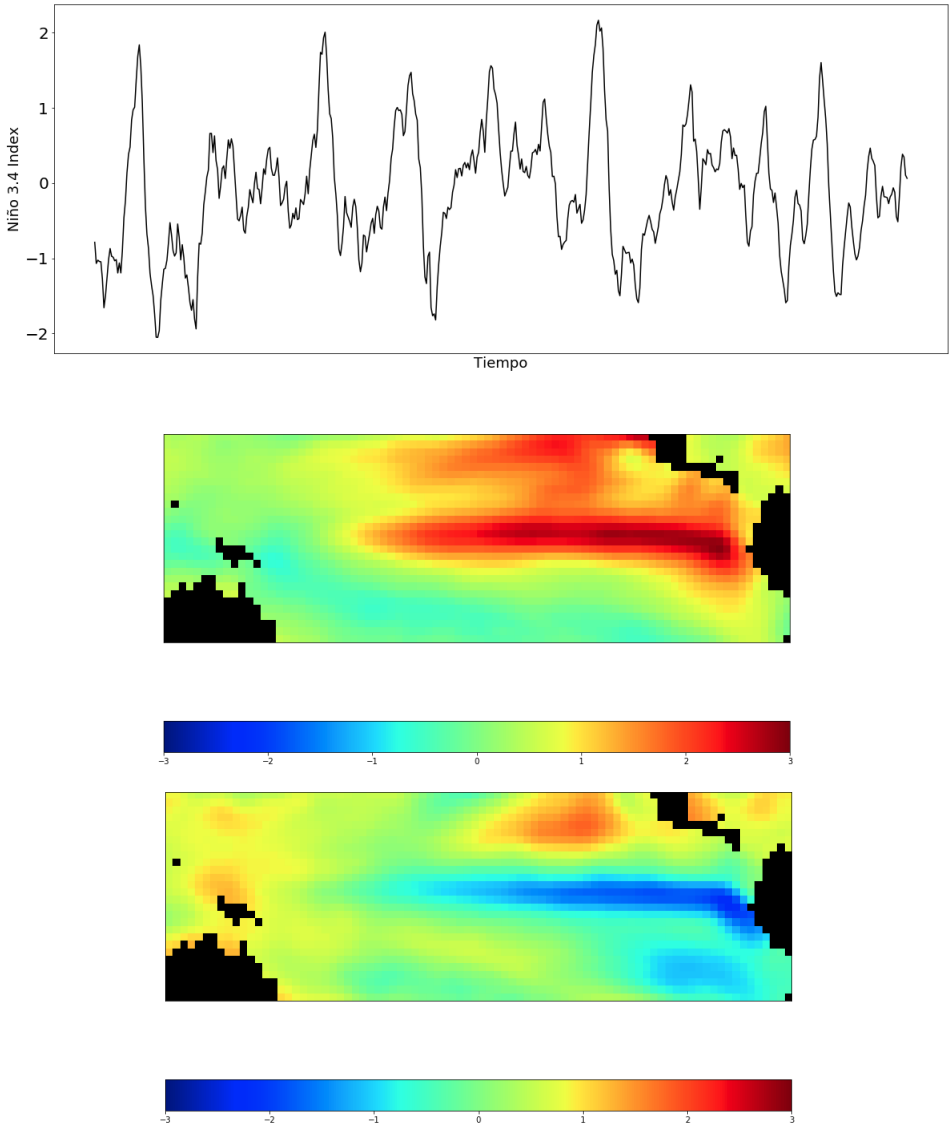

Figura 2.2: (De arriba a abajo) Serie temporal del índice de El Niño 3.4 para el periodo comprendido entre Enero de 1970 y Diciembre de 2016 y patrones espaciales para una situación de Niño y de Niña.

CAPÍTULO 3

Deep Learning aplicado al Forecasting

3.1. Datos temporales y Forecasting

Los datos de las anomalías de ERSST son datos numéricos en los que cada pixel representa la variación de la temperatura en grados centígrados respecto a su respectiva media mensual. Estamos interesandos en modelar las relaciones entre meses pasados con meses futuros a fin de ser capaces de adelantarnos a los posibles escenarios anómalos. Se puede ver entonces que estamos ante un problema de regresión con una fuerte componente temporal. La rama de la estadística dedicada a los problemas en los que se intenta conocer el futuro a través de datos pasados se la conoce como forecasting.

Muchas veces no queda clara la distinción entre un problema de regresión y uno de forecasting debido a sus múltiples similitudes. Forecasting se suele aplicar a datos temporales ya que realizamos predicciones del futuro en base al pasado (se podría hablar de una extrapolación como tal) sin embargo en una regresión calculamos la predicción centrándonos mucho más en la muestras disponibles y ya observadas (se podría ver más como un caso de interpolación). Existen muchos otros motivos teóricos que hacen imposible la aplicación de los modelos de regresión a problemas de forecasting (suposiciones respecto a la independencia de la observaciones, distribuciones de lo errores...).

Un tipo de problema clásico de forecasting es el de la predicción de series temporales. Una serie temporal está compuesta por una secuencia de variables aleatorias $Y_1, Y_2, Y_3...$ donde la variable Y_t denota el valor tomado por esta serie en el tiempo t. Generalmente estas variables aleatorias estarán correlacionadas temporalmente.

Desde el punto de vista estadsítico este tipo de series se han podido enfocar siempre desde dos puntos de vistas distintos: enfoque determinista y metodología Box-Jenkins.

El enfoque determinista supone que la señal original está compuesta por varias componentes deterministas (tendencia, estacionalidad y ciclo) además de un ruido inherente al propio proceso, de esta forma el proceso se puede expresar como una combinación de estas. Aunque parece una forma naive de abordar el problema en ocasiones existen series sencillas que encajan muy bien en este marco teórico. En la Figura 3.1 se muestra un ejemplo de descomposición de una serie temporal.

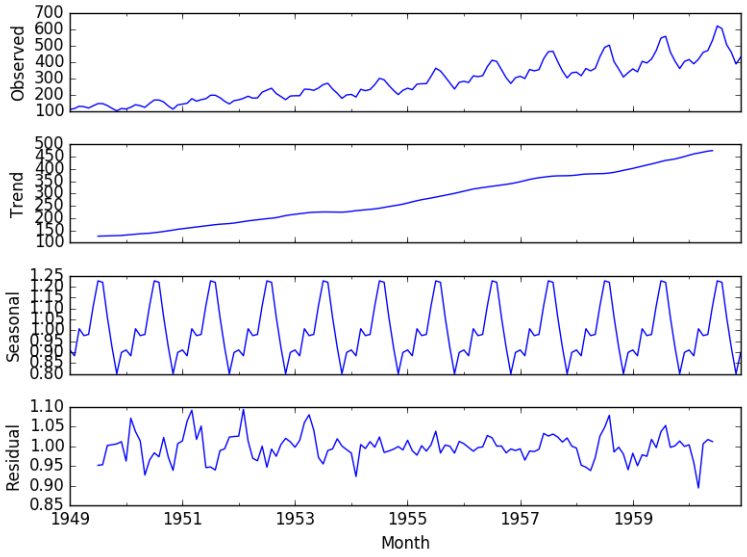

Figura 3.1: Descomposición de una serie temporal

Otro enfoque muy utilizado para el modelado de series temporales univariantes es la Metodología Box-Jenkins [Box et al. (2015)]. Supone que la serie temporal no es más que una realización de un proceso estocástico, por ello busca encontrar un modelo capaz de ajustarse a los datos de forma que genere unas predicciones fiables. Este enfoque se sustenta en los conocidos como modelos ARIMA. Las letras de su nombre hacen referencia a los elementos que los componen:

- AR: Parte autorregresiva que supone que el valor Y_t no es más que una combinación lineal de $Y_{t-1}...Y_{t-k}$

- I: Hace referencia al término integrado. Estos modelos necesitan trabajar bajo el supuesto de estacionariedad. En caso de no estacionariedad esta parte del modelo se encarga de transformar la serie para cumplir con este supuesto

- MA: Esta parte es similar a la autorregresiva solo que en este caso utiliza el ruido blanco del modelo para aproximar Y_t

Los modelos ARIMA, aunque lineales, tienen una gran ventaja y es que poseen la suficiente robustez estadística necesaria para generar unos intervalos de confianza de calidad. En la Figura 3.2 se muestra un ejemplo de predicción y generación de intervalos de confianza a partir de un modelo ARIMA.

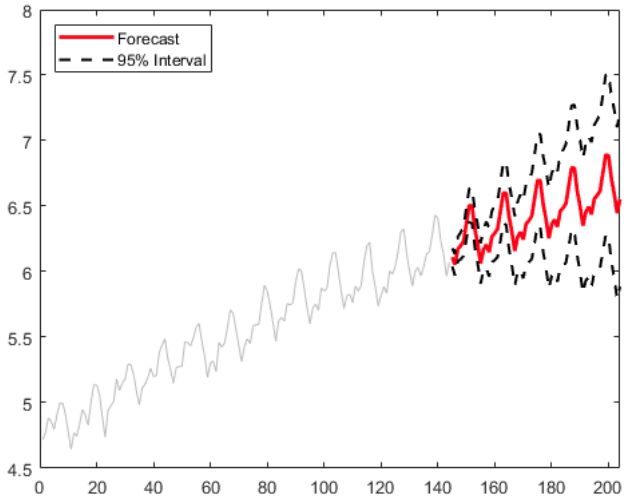

Figura 3.2: Predicciones e intervalos de confianza de un modelo ARIMA

Hasta hora hemos visto una forma de lidiar con datos temporales univariantes pero ¿qué ocurre si estamos ante un problema de predicción de series temporales multivariantes? Para estos casos existen una generalización de los modelos autorregresivos lineales conocida como VAR (modelos vectoriales autorregresivos). Estos modelos funcionan de una forma similar a los AR solo que permiten la interacción lineal de varias variables para el forecasting de series temporales.

Todos los modelos hasta ahora mencionados aunque permiten la generación robusta de un intervalo de confianza siguen siendo lineales, lo cual limita enormemente la tipología de problemas a englobar. En los últimos años se han comenzado a utilizar modelos basados en aprendizaje estadístico o machine learning para el modelado de una gran variedad de problemas, siendo uno de ellos el forecasting. Esto ha hecho que estos modelos se estableciesen como serios rivales de las técnicas más clásicas que se venían utilizando hasta la fecha.

Son numerosos los casos de éxito de la aplicación de este tipo de modelos en problemas de forecasting y van desde el campo financiero [Cao and Tay (2003)] hasta el biomédico [Mani et al. (2012)]. Máquinas de vectores de soporte, árboles de decisión y algoritmos de boosting son algunos de los muchos modelos que se empezaron a

aplicar en este tipo de problemas. Sin embargo muchas veces los investigadores se dejaron llevar por la moda del machine learning y empezaron a aplicar en forecasting modelos diseñados para problemas de regresión, esto junto a otras características propias de los datos manejados en problemas de forecasting llevó a los siguientes sesgos:

- Los dataset típicos de problemas de forecasting solían ser de un tamaño pequeño lo cual sesgaba el aprendizaje de algoritmos de machine learning muy dependientes de los datos.
- Muchos de los modelos de machine learning no están diseñados para extrapolar sino para hacer regresión lo cual, como se ha mencionado con anterioridad, podía sesgar los resultados
- Estos modelos carecen de un mecanismo robusto para la generación de intervalos de confianza.
- Los modelos de machine learning no poseen la interpretabilidad característica de modelos de forecasting como ARIMA

Es por estos motivos por los que grupos de investigación dedicados al forecasting comenzaron a estudiar la capacidad real de estos modelos. Son bien conocidas las competiciones M3 [Makridakis and Hibon (2000)] y M4 [Makridakis et al. (2018)] de forecasting. En estas competiciones podía participar cualquier persona y el objetivo era, dadas un número alto de series temporales, realizar predicciones a fin de ver cuales eran los métodos que mejores resultados generaban.

En la competición M3 el algoritmo que mejores resultado dió fue el theta. Este método no era más que que una combinación de regresión lineal y un suavizado exponencial con deriva. Otros algoritmos basados en modelos autorregresivos obtuvieron también buenos resultados. Esta competición se celebró en 1998 por lo que el campo del machine learning como tal aún no se había popularizado, sin embargo si que se había empezado a aplicar timidamente unos algoritmos conocidos como redes neuronales.

Las redes neuronales son unos algoritmos levemente bioinspirados, formados por nodos de cálculo conocidos como neuronas unidos a través de conexiones ponderadas. En estos algoritmos los datos entran por la capa de entrada y una combinación lineal de estos avanza a la primera capa oculta donde en cada neurona se aplica una transformación no lineal, esta operación continua dependiendo el número de capas ocultas hasta que llega a una capa final donde devuelve la correspondiente predicción para la entrada introducida. Los pesos de las conexiones son los que se ajustan intentado siempre minimizar cierta función de coste entre el valor original a predecir y la predicción generada por la red. En la Figura 3.3 se muestra un esquema típico

de red neuronal.

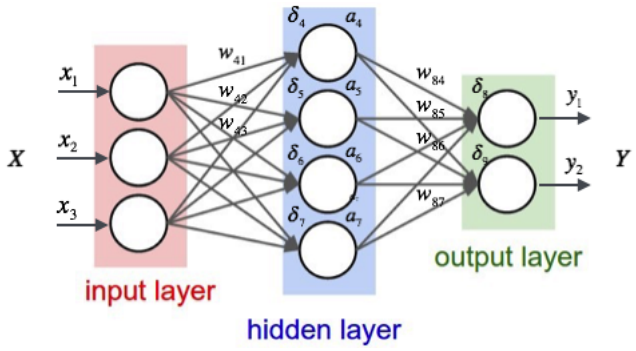

Figura 3.3: Esquema de una red neuronal simple

En la competición M3 únicamente una persona utilizó estos algoritmos para realizar predicciones con poco éxito. En el año 2018 se realizó la competición M4 con el propósito de enfrentar los ya desarrollados algoritmos de machine learning y redes neuronales con los clásicos de forecasting. En esos 20 años entre competiciones los campos del machine learning y el de la redes neuronales habían avanzado exponencialmente y formaban el estado del arte en casi todas las tipologías de problemas. En esta competición además de la calidad de la predicción se tenía en cuenta también la calidad de los correspondientes intervalos de confianza. Algunos de los resultados más sorprendentes fueron:

- De los 17 mejores resultados 12 estaban conformados únicamente por metodos estadísticos clásicos, el resto incluían además algún algoritmo sencillo de machine learning.
- Los métodos puramente basados en machine learning no conseguían unos resultados significativamente mejores que los del benchmark
- La generación de intervalos de confianza de los métodos de machine learning era también mala
- El resultado ganador, con una gran diferencia respecto al resto, era un híbrido entre redes neuronales y suavizados exponenciales.

Esta competición no estuvo exenta de crítica y la gente afín al campo del machine learning argumentaba que las series temporales a predecir no eran representativas del tipo de datos que se solían encontrar en el mundo real, siendo señales con pocos datos y poco ruido. Aún con todo esto parecía que el algoritmo que ofrecía mejores resultados era uno basado en un tipo de red neuronal adaptada a datos temporales conocida como red neuronal recurrente.

Este tipo de redes ha conseguido alcanzar el estado del arte en una multitud muy amplias de problemas desde procesamiento del lenguaje [Kumar and Rastogi (2019)] hasta la predicción de frames en vídeos [Fan et al. (2019)]. En los siguientes apartados haremos un recorrido más detallado por las distintas arquitecturas de redes neuronales útiles para el problema tratado en este trabajo. Además se introducirá también el bayesian deep learning y se mostrará como se relaciona con la cuantificación de la incertidumbre a través de intervalos de confianza.

3.2. Redes neuronales densas

Las redes neuronales densas fueron los primeros modelos de este tipo en lograr resultados destacables en multitud de campos. En este apartado describiremos su estructura, su funcionamiento, su capacidad de aprendizaje y demás conceptos que nos facilitarán la comprensión de futuros apartados.

Supongamos que tenemos una matriz $\mathbf{X}$ de dimension $n \times m$ siendo n el número de observaciones y m el número de variables y una matriz $\mathbf{Y}$ de dimension $n \times 1$ con n el número de observaciones. Supongamos para simplificar que tanto las m variables de $\mathbf{X}$ como $\mathbf{Y}$ son numéricas, esto nos lleva a un problema de regresión, en el que buscamos encontrar una relación del estilo $y = f(x)$.

Una red neuronal está formada por tres tipos distintos de capas de neuronas:

- Capa de entrada: Esta capa se encarga de introducir en la red el input, es decir, el vector $\mathbf{x}$ de dimensión $1 \times m$ extraído de la matriz $\mathbf{X}$. Este vector se corresponde con una observación.
- Capas intermedias: Estas capas reciben una combinación lineal del output de la capa anterior, aplican unas funciones no lineales y devuelven el resultado como input a la capa siguiente. Según la arquitectura elegida puede haber una o más capas de este tipo. Por ejemplo, el output de la capa oculta $\mathbf{h}$, es una combinación lineal del ouput de la capa anterior ($\mathbf{x}$) través de la matriz de parámetros $\mathbf{W}_1$ y $\mathbf{b}_1$ al que se le aplica una función no lineal $g(\cdot)$.

$$\mathbf{h} = g(x\mathbf{W}_1 + \mathbf{b}_1) \tag{3.1}$$

- Capa final: Esta capa se encarga de recoger los outputs de la última capa intermedia y realizar la transformación adecuada para devolver un resultado en el mismo rango que los valores de $\mathbf{Y}$.

Los parámetros de la red se optimizan minimizando una función de coste, generalmente el mean squared error. Para ello se recurre al método de descenso de gradiente [LeCun et al. (2015)] y al algoritmo de backpropagation [LeCun et al.

(1988)], el cual lleva a cabo un proceso numérico por el cual los parámetros se ajustan de tal manera que el error se disminuya.

Existe un teorema conocido como El Teorema del Aproximador Universal [Hornik (1991)] que demuestra que una red neuronal densa con una única capa intermedia compuesta por un número finito de neuronas es capaz de aproximar una gran parte de las funciones continuas posibles. Aunque esto es cierto en teoría en la práctica la tarea se complica ya que en ocasiones el aprendizaje de los parámetros que aproximan la función deseada no es trivial, llevando a problemas de convergencia o mínimos locales, entre otros.

Aunque estas redes supieron adaptarse muy bien a un espectro muy amplio de problemas sufrian una serie de carencias en su arquitectura que no las permitía tener éxito en algunos campos con su propia tipología de datos. Por ejemplo, las redes convolucionales o las redes recurrentes son más adecuadas para el aprendizaje de patrones espaciales o temporales, respectivamente. En nuestro estudio utilizaremos ambas arquitecturas y por tanto en la siguiente sección se hará una introducción a ambas tipologías.

3.3. Redes Neuronales Recurrentes

Las redes neuronales densas poseen una excelente capacidad de modelización no lineal de procesos multivariantes sin embargo son incapaces de tener en cuenta relaciones temporales en los datos. Esto se debe a que procesa los inputs de uno en uno de forma que no es le es posible tener en cuenta en el presente información de observaciones anteriores (pasado). A fin de resolver esta problemática surgieron las redes neuronales recurrentes [Hopfield (1982)]. Estas redes ya no solo tienen en cuenta la observación acutal para hacer la predicción si no que también almacenan información referente a observaciones anteriores en el tiempo a fin de incorporar esta posible estructura temporal al modelo

3.3.1. Red Neuronal Recurrente Simple

Supongamos que estamos ante un problema de forecasting en el que queremos conocer el valor de nuestra variable en el tiempo t a partir de sus valores rezagados hasta el tiempo t–k. Bajo estas condiciones nuestro vector de entrada x de la red necesitaría ser estructurado de la siguiente forma:

$$\mathbf{x} = (x_{t-1}, x_{t-2}, ..., x_{t-k}) \tag{3.2}$$

Este vector sería el que utilizariamos para predecir el valor y_t, es decir, estamos

utilizando los k valores pasados de la variable para realizar la predicción en el tiempo t. Para poder adaptar la arquitectura de la red a este tipo de datos necesitaríamos una retroalimentación entre estados de tiempo de forma que la red fuese capaz de recordar estados pasados a fin de realizar la predicción para el estado actual. Esto es posible a partir de una conexión como la de la Figura 3.4.

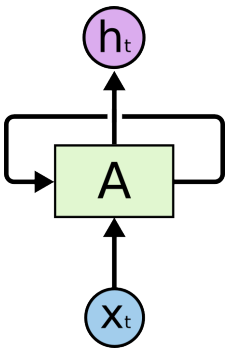

Figura 3.4: Ejemplo de recurrencia en una red neuronal

Se puede apreciar como la red se encarga de procesar los valores de la variable de entrada en tiempos pasados a fin de poder combinarla con el valor actual, de esta forma puede tomar decisiones teniendo en cuanta los valores más recientes pero sin olvidarse de la posible influencia de valores más alejados en el tiempo. Quizás se puede apreciar mejor esta característica en la Figura 3.5 donde se muestra la versión desenrollada del diagrama de la Figura 3.4.

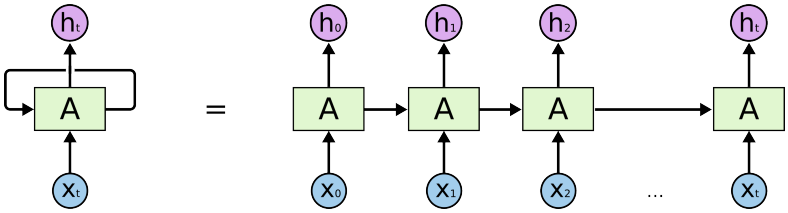

Figura 3.5: red neuronal recurrente desenrollada

Los bloques marcados con la letra A son, en cierta manera, equivalentes a las capas intermedias de las redes neuronales densas y son las encargados de realizar las transformaciones no lineales a los datos. Es fácil ver como ahora las capas intermedian procesan información tanto del input en el tiempo t como del input en tiempos anteriores, sin embargo no procesa directamente la entrada de tiempos anteriores si no una transformación de esta. A esta transformación se le conoce como estado oculto $\mathbf{h}_t$ y contiene la información seecuencial de estados anteriores que la red ha considerado que es útil mantener a fin de capturar la estructura temporal de los

datos.

Estos estados ocultos se calculan en los bloques A a partir del siguiente cálculo:

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1}) \tag{3.3}$$

siendo $g(\cdot)$ una función no lineal. Se puede ver fácilmente como ahora la red no solo realiza la predicción con el estado actual sino que además utiliza un estado oculto que contiene información acerca de los estados pasados de la variable. Las matrices de parámetros $\mathbf{W}$ y $\mathbf{U}$ cumplen funciones distintas, la primera se centra en los datos de entrada de cada estado mientras que la $\mathbf{U}$ define los estados ocultos. Estos bloques propagan los estados ocultos hacia delante en el tiempo y retornan el estado oculto en ese momento.

Los parámetros de esta red se actualizan de forma que se minimize el error de la predicción devuelta por la red y el valor real al igual que en la red neuronal densa sin embargo ahora en vez del método de backpropagation se utiliza el backpropagation through time. A efectos prácticos ambos funcionan de la misma forma sin embargo el segundo antes de calcular y propagar el gradiente debe desenrollar la red de forma que no haya ningún bucle de retroalimentación que impida el flujo del gradiente desde la salida hasta la entrada de la red.

3.3.2. Long Short Term Memory

A medida que el uso de este tipo de redes empezó a extenderse se vió que presentaban problemas de aprendizaje. Parecía que cuando los estados ocultos de la red necesitaban aprender y mantener información de estados muy alejados en el tiempo su rendimiento empeoraba. Los investigadores trabajaron en ello y llegaron a la conclusión de que el problema se relacionaba con la propagación del gradiente [Pascanu et al. (2013)].

Durante el backpropagation through time la red se desenrrolla siendo el número de capas proporcional al numero de valores N que se uilizan para hacer la predicción. Esto lleva a que las redes recurrentes a optimizar contengan un número alto de capas. El gradiente en este tipo de modelos no es más que el producto de los gradientes de las capas posteriores por lo que si se incrementa el número de capas la convergencia se vuelve inestable.

Otras de las razones era al propio funcionamiento de las redes recurrentes. La matriz de parámetros $\mathbf{W}$ que multiplica a $\mathbf{x}_t$ es la misma para todo t, esto hacía que $\mathbf{W}$ apareciese muchas veces en el gradiente provocando también un comportamiento inestable en la convergencia de la red.

Las redes propuestas para solventar esta problemática fueron las LSTM (Long Short Term Memory) [Hochreiter and Schmidhuber (1997)]. Se diseñaron con el objetivo de no presentar problemas en el aprendizaje de secuencias con una dependencia a largo plazo. Su estructura era similar a la de las redes recurrentes clásica sin embargo ahora en su capa intermedia (bloque A) se realizaban más operaciones. En la Figura 3.6 se muestra una representación gráfica de esta arquitectura.

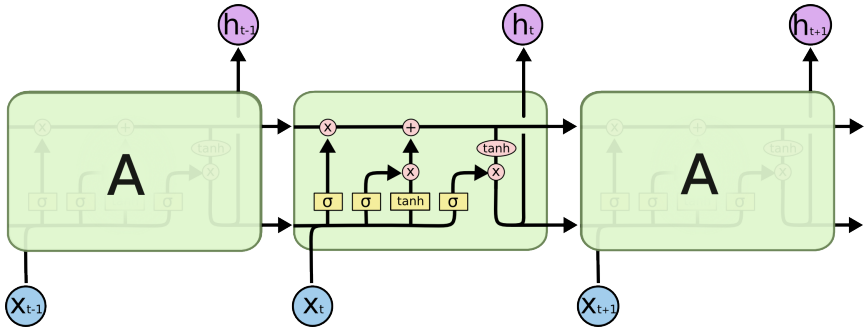

Figura 3.6: Arquitectura de una LSTM

Una de las diferencias fundamentales de las LSTM respecto a las redes recurrentes estándar es que ahora en vez de solo comunicarse temporalmente a través del estado oculto $\mathbf{h}_t$ lo hacen también a través de un nuevo término conocido como estado de la celda $\mathbf{c}_t$. Este nuevo término añade capacidades selectivas a $\mathbf{h}_t$ permitiendo que mantenga información importante de estados lejanos.

Aunque existen muchas variaciones de redes LSTM la más común se caracteriza por componerse de tres puertas o gates. La primera de ellas se conoce como forget gate o puerta de olvido y se define como:

$$f_t = \sigma(\mathbf{W}_j[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \tag{3.4}$$

Se puede interpretar como una ponderación de los elementos del estado de la celda $\mathbf{c}_t$ relativo a aspectos a olvidar o recordar de estados anteriores teniendo en cuenta la entrada actual (el rango de la función sigmoidal está entre 0 y 1). La siguiente puerta se conoce con el nombre de puerta de entrada o input gate y tiene la forma:

$$\begin{aligned} \mathbf{i}_t &:= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \\ \mathbf{c}'_t &:= tanh(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \end{aligned} \tag{3.5}$$

Aunque está formada por dos capas se puede apreciar como el resultado es una actualización del estado de la celda $\mathbf{c}_t$ a $\mathbf{c}'_t$. A diferencia de la forget gate donde se reescalaban los elementos de $\mathbf{c}_t$ aquí se actualizan en su totalidad generando por lo

tanto una nueva $\mathbf{c}_t$. En último lugar se encuentra la output gate o puerta de salida:

$$\begin{aligned} \mathbf{o}_t &:= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \\ \mathbf{h}_t &:= \mathbf{o}_t tanh(\mathbf{c}_t) \end{aligned} \tag{3.6}$$

donde finalmente teniendo en cuenta la información generada en esa capa se devuelve el estado oculto $\mathbf{h}_t$ y se introduce como input en la siguiente capa.

Este tipo de redes son un factor clave en la consecución del estado del arte en multitud de problemas con un marcado componente temporal como pueden ser el procesamiento del lenguaje natural, descripción automática de imágenes, reconocimiento de audio, etc.

3.4. Redes Neuronales Convolucionales

Hasta ahora hemos visto como las redes neuronales densas fueron capaces de aprender funciones no lineales y las recurrentes funciones no lineales marcadas por un alto componente temporal pero ¿qué ocurre con las estructuras o relaciones espaciales?

Imaginemos que queremos distinguir si en una imagen formada por un número de pixeles determinados (alto y ancho) y 3 capas de color (el estándar RGB) hay un gato o no. Una posible forma de abordar el problema sería utilizar redes neuronales densas en las que la primera capa esté formada por un número de neuronas igual a los pixeles presentes en la imagen. Rápidamente surgen dos problemas a considerar si se utiliza este enfoque:

- Debido a la cantidad de pixeles en las imágenes el número de neuronas presentes en la capa de entrada sería muy alto (también lo serían las de las capas intermedias debido a la arquitectura en forma de embudo que suelen tener este tipo de modelos)
- Las capas densas hacen a la red invariante a translaciones. Si la red aprende a localizar a gatos centrados en la imagen no va a ser capaz de identificar a un gato en un extremo debido a la naturaleza estática de sus neuronas. Otros factores como la vectorización de las imágenes añadirían dificultad en la extracción de la representación espacial de los datos.

A fin de solventar estos problemas surgieron unas redes especializadas en relaciones espaciales conocidas como redes neuronales convolucionales [LeCun et al. (1999)]. La principal innovación que dio a estas arquitecturas la capacidad de aprendizaje espacial fue la inclusión de filtros. Estos filtros se apoyan en la idea de que una imagen está compuesta de elementos más pequeños y que los pixeles cercanos tienen

mucha más influencia que los lejanos. En la Figura 3.7 se muestra la aplicación de un filtro a una imagen:

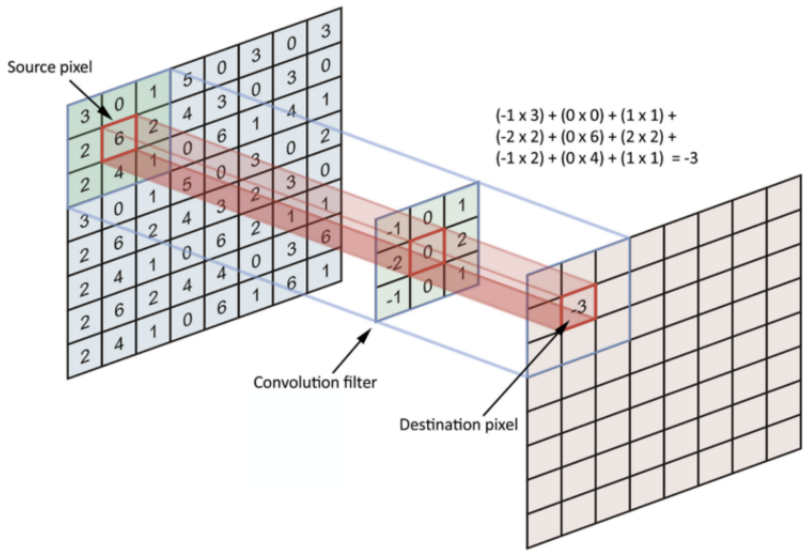

Figura 3.7: Ejemplo de aplicación de un filtro convolucional

Estos filtros son el equivalente a las capas de las redes neuronales densas. Están formados por parámetros que se buscarán optimizar para minimizar el error de la red. Estos filtros se deslizan por toda la imagen de entrada de forma que siempre se aplican a conjuntos de píxeles espacialmente cercanos. En cada pasada el filtro calcula la suma de la combinación lineal de los pixeles que engloba y le aplica una función no lineal. El resultado, que tendrá una dimensión inferior al de entrada será el input de la siguiente capa. Esta siguiente capa será un filtro distinto que aprenderá a identificar otro aspectos útiles de los datos de entrada. Mientras que la entrada de la red es la propia imagen las transformaciones realizadas por los filtros son representacion específicas de la red útiles para minimzar el error de la predicción.

El objetivo final de este tipo de redes es el de construir una representación de la imagen a través de la aplicación de distintos filtros que contenga la información necesaria para la consecución de cierta tarea. Volvamos al ejemplo de la identificación del gato, en este caso los distintos filtros aprenderán a localizar orejas, ojos y características propias de los gatos a fin de relacionar las imágenes de gatos con el concepto de gato.

Una red neuronal convolucional aunque basada en filtros contiene también otro tipo de capas conocida como max pooling que ayudan a acelerar el aprendizaje. Su objetivo es reducir la dimensionalidad de los espacios de entrada a fin de reducir el número de parámetros necesarios en los filtros y aumentar la cantidad de invarianza a la translación tan característica de las redes convolucionales. Su aplicación es sen-

cilla (max filter) y no necesita del aprendizaje de ningún parámetro. En la Figura 3.8 se muestra un esquema de este tipo de capa.

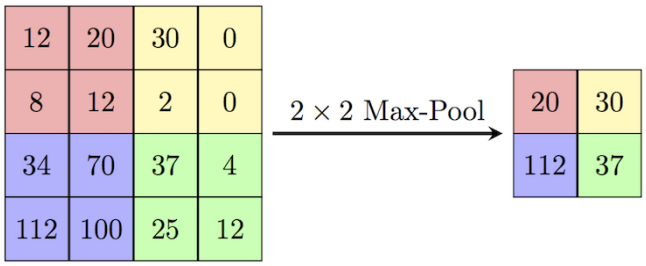

Figura 3.8: Esquema del Max Pooling

Los parámetros de estos modelos se aprenden a través del backpropagation. Al tratarse de una arquitectura compuesta por capas muy distintas a las redes densas y recurrentes su aplicación difiere de la de estas pero los principios tras su funcionamiento son los mismos.

Las redes convolucionales enfocadas a resolver tareas complejas están compuestas por la combinación de varios filtros, capas de max pooling y, en ocasiones, algunas capas más complejas. En la Figura 3.9 se muestra la arquitectura completa de una red convolucional.

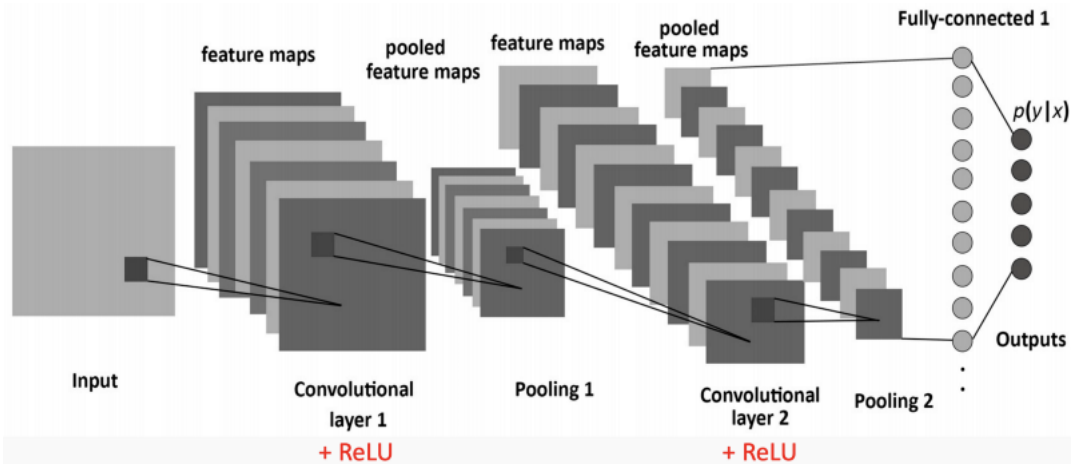

Figura 3.9: Ejemplo de red neuronal convolucional

Este tipo de redes supusieron un antes y un después en la modelización de tareas con un fuerte componente espacial como en clasificación de imágenes [Krizhevsky et al. (2012)], detección de objetos [Redmon et al. (2016)] y super resolución en

imágenes [Dong et al. (2015)].

3.4.1. LSTM Convolucional

Hasta ahora hemos visto un tipo de red enfocado a modelar relaciones temporales y otro para relaciones espaciales pero ¿qué ocurre si en nuestros datos nos encontramos con ambas estructuras? Esta situación se suele dar en el campo de la meteorología dónde además de la relación temporal típica se da una relación espacial ya que las condiciones meterológicas de un punto en concreto dependen de las condiciones cercanas en el espacio.

En un principio la modelización de este tipo de procesos se llevó a cabo mediante la inclusión de capas convolucionales en las entradas de las redes recurrentes de forma que la red ya recibía un vector con información espacial de los datos originales de entrada y lo único que tenía que hacer era modelizar su parte recurrente a través de una LSTM. En la Figura 3.10 se muestra un arquetipo de este tipo de estructuras:

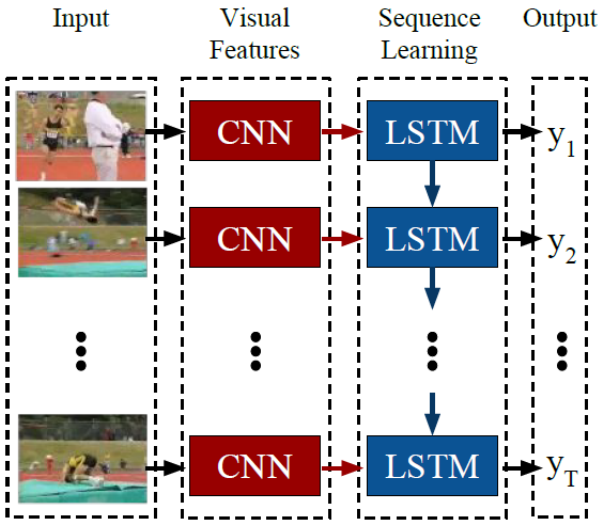

Figura 3.10: Arquitectura LSTM + CONV

Aunque parecía un enfoque adecuado no tenía en cuenta un pequeño detalle y es que las conexiones recurrentes de la LSTM no eran convolucionales. Por ello surgió un nuevo hibrido compuesto por la combinación de estas dos estructuras conocido como LSTM convolucional [Xingjian et al. (2015)]. Este modelo se caracteriza por implementar operaciones convolucionales tanto en la entrada de la LSTM como en las conexiones recurrentes entre capas intermedias. Esto hace que los estados ocultos $\mathbf{h}_t$ y los estados de las celdas $\mathbf{c}_t$ se calculen teniendo en cuenta también la información espacial, de forma que si un pixel de la imagen es importante para la

modelización de la estructura temporal es probable que los píxeles cercanos también lo sean.

Esta arquitectura aún siendo relativamente nueva ya ha sido aplicada con éxito a problemas regidos por una dinámica espacio-temporal como el de la predicción de patrones de tráfico [Lin et al. (2018)].

3.5. Deep Learning

Aunque en la primera década del siglo XXI se dieron avances significativos en el campo de las redes neuronales no fue hasta aproximadamente el año 2012 [Krizhevsky et al. (2012)] cuando este tipo de algoritmos comenzó a establecerse en un campo propio conocido como deep learning. Los modelos neuronales a partir de este momento comenzaron a ganar en complejidad con todo lo que ello conlleva (más parámetros que aprender y por la tanto un mayor coste computacional). La explosión de este campo se debió principalmente a dos factores claves:

- La generación de datos aumentó de forma exponencial debido principalmente al uso masivo de Internet. Esto llevo a que los modelos tuviesen mucho más material del que aprender, potenciando sus resultados y sus posibles aplicaciones.

- Los avances en el hardware trajeron consigo la drástica reducción de los tiempos de entrenamiento además de la posibilidad de desarrollar modelos neuronales mucho más profundos.

Es por esto por lo que hoy en día tenemos modelos compuestos por un numero elevadísimo de capas [He et al. (2016)], modelos especializados en problemas muy concretos a través de complejas arquitecturas [Ronneberger et al. (2015)] o incluso redes capaces de aprender la representación latente de un proceso y generar nuevos elementos en base a unas distribuciones de probabilidad aprendidas [Goodfellow et al. (2014)]. Sin embargo, el deep learning no es solo la introducción de grandes cantidades de capas sino también la tecnología asociada que ha hecho posible el entrenamiento de estos modelos. Cabe mencionar la introducción de nuevos algoritmos de aprendizaje, como el Adam [Kingma and Ba (2014)], o nuevas funciones de activación, como la ReLU, además de frameworks como Tensorflow o Keras que hacen posible la construcción de estos modelos de manera sencilla.

3.5.1. Métodos de regularización

El entrenamiento de una red neuronal no es tarea sencilla y en ocasiones surgen problemas que nos impiden llegar a soluciones óptimas en nuestros datos. Uno de

estos problemas es el conocido como overfitting o sobreajuste. Esto se da cuando nuestra red en vez de aprender memoriza y por lo tanto no es capaz de generalizar para datos no vistos durante la fase de entrenamiento. En la Figura 3.11 se aprecia como la línea verde aunque discrimina perfectamente los datos no generaliza a nuevos datos ya que su frontera posee una varianza muy alta. La línea negra sin embargo aunque tiene un peor ajuste tiene una capacidad de generalización mayor.

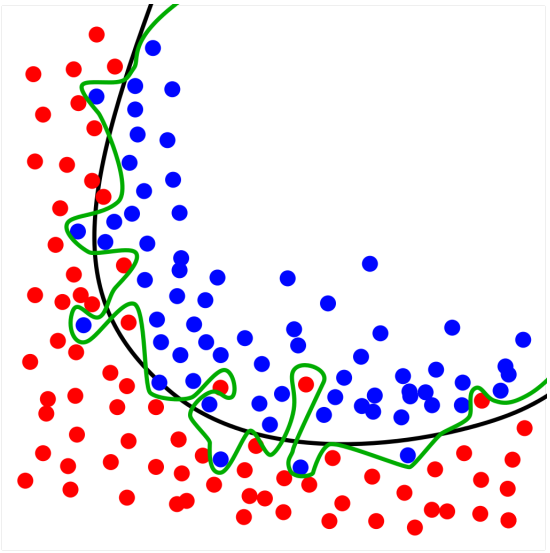

Figura 3.11: Ejemplo de overfitting

Se han desarrollado métodos para evitar el sobreajuste conocidos como métodos de regularización. Uno de los más conocidos y usados en deep learning es la penalización de los parámetros mediante la norma $L1$ o $L2$ [Schmidhuber (2015)]. Es de especial interés en este trabajo la técnica de regularización conocida como dropout ya que está directamente asociado con la bayesianización de las redes neuronales.

Dropout

Una técnica de regularización específica de las redes neuronales es la conocida como dropout. Aunque es sencilla se utiliza mucho ya que suele presentar buenos resultados. Consiste en, de forma aleatoria durante el entrenamiento, eliminar neuronas cortando sus conexiones con el resto. Se muestra un ejemplo de esto en la Figura 3.12.

El número de neuronas a eliminar en cada capa por cada pasada de forward propagation y backpropagation se controla a través del hiperparámetro p que no es más que la probabilidad de que una neurona pase a desaparecer en la actual pasada.

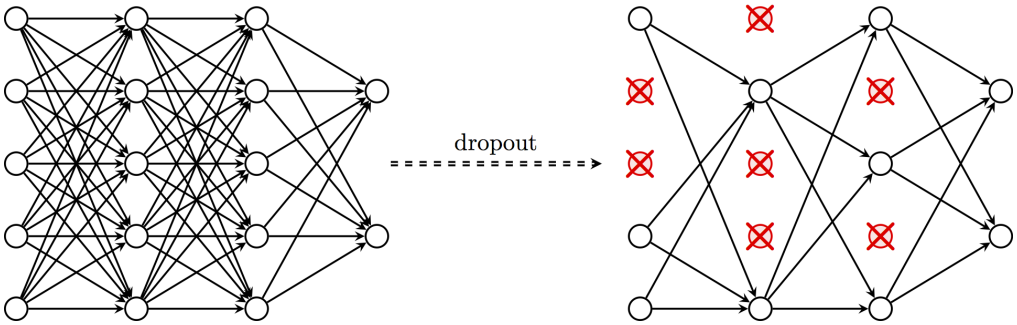

Figura 3.12: Visualización del dropout

Esta técnica consigue regularizar la red ya que evita que distintos grupos de neuronas se centren únicamente en una cierta tipología de casos y que haya neuronas que dependan enormemente de la salida generada por otras neuronas específicas. Esto hace que las neuronas estén obligadas a adaptarse a una amplia gama de patrones y, por lo tanto, generalicen con más facilidad.

3.6. Bayesian Deep Learning

Hemos visto como las redes neuronales, tras un entrenamiento en el que se ajustan los valores de ciertos parámetros, son capaces de aproximar funciones no lineales que actuan como mapeo entre las variables independientes y la dependiente. Aunque estos modelos ofrecen unos resultados muy buenos carecen de una característica fundamental: la cuantificación de la incertidumbre.

Los modelos de redes neuronales poseen aspectos negativos, siendo uno de ellos la no interpretabilidad. Este tipo de algoritmos actuan como una black box o caja negra donde introducimos unos datos y recibimos una salida, si todo ha ido bien el modelo predice correctamente por lo que suponemos que ha conseguido aprender correctamente la función no lineal que ha generado nuestros datos, sin embargo esto no tiene por qué ser así. ¿Podemos saber qué ha aprendido realmente? ¿podemos saber como de seguro está de sus predicciones? Una correcta cuantificación de la incertidumbre nos puede aportar información acerca de los mecánismos internos de la red facilitando la interpretabilidad y ayudando en la toma de decisiones.

Una de las primeras aproximaciones utilizadas para cuantificar la incertidumbre en este tipo de modelos fueron las redes neuronales bayesianas [MacKay (1992)]. Estas redes utilizan la teoría estadística bayesiana para modelizar los parámetros de la red a través de distribuciones de probabilidad, de esta forma mientras la aproximación tradicional asigna valores puntuales a los parámetros estas redes les asignan un función de distribución de forma que se dispone de mucha más información so-

bre el posible valor real del parámetro. Se muestra una comparación de estas dos aproximaciones para el valor de un parámetro y un término de bias en la Figura 3.13.

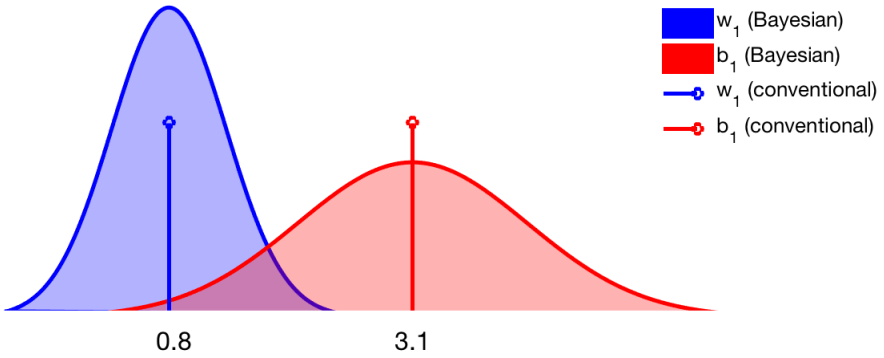

Figura 3.13: Diferencias entre la estimación de una red clásica y una bayesiana

Tener distribuciones en vez de valores puntuales nos permite, para una determinada entrada de la red, samplear los parámetros pudiendo obtener así muchas predicciones distintas para la misma entrada. Si las predicciones son consistentes se puede afirmar que la red está segura de su predicción, en caso contrario no. El marco bayesiano equipa a este tipo de redes con un método robusto de cuantificación de la incertidumbre.

Este tipo de redes sin embargo sufre de un problema y es su alto coste computacional. Las distribuciones de los parámetros se inicializan con unas distribuciones a priori determinadas, después con cada pasada de los datos estas se actualizan de acuerdo al teorema de Bayes y se obtienen las distribuciones a posteriori:

$$P(\mathbf{W}|\mathbf{X},\mathbf{Y}) = \frac{P(\mathbf{X}|\mathbf{Y},\mathbf{W})P(\mathbf{W})}{\int P(\mathbf{X}|\mathbf{Y},\mathbf{W})P(\mathbf{W})d\mathbf{W}} \tag{3.7}$$

El cuello de botella de 3.7 se encuentra en la integral del denominador ya que en la mayoría de casos su solución analítica es desconocida y la dimensión de $\mathbf{W}$ muy alta. Para aproximar esta integral se utilizan métodos numéricos que se limitan a aproximar el resultado, dos de los más extendidos son:

- MCMC: Las cadenas de Markov de Monte Carlo son métodos numéricos de simulación cuyo objetivo es aproximar una distribución de probabilidad determinada. La idea clave de este algoritmo es la de simular una cadena de Markov cuya distribución estacionaria se aproxime a la distribución a posteriori del modelo, de esta forma somos capaces de obtener una aproximación de esa distribución.

- Variational Inference: Estos algoritmos aproximan la distribución a posteriori a través de funciones de distribucion más sencillas, de esta forma transforma el problema a uno de optimización donde buscamos minimizar la diferencia entre la función a posteriori y la aproximación. Esta diferencia se mide a través de la divergencia de Kullback-Leibler:

$$D_{KL}(P \mid\mid Q) = \int_{-\infty}^{+\infty} p(x) ln \frac{p(x)}{q(x)} dx \tag{3.8}$$

  donde P y Q son distribuciones de probabilidad.

Aunque estos modelos se sustentan sobre un fuerte marco teórico necesitan de mucha capacidad de computo para su correcta aplicación lo que lleva a la limitación de la complejidad de las redes en las que se puede aplicar. Esto llevó a un grupo de investigadores a desarrollar una intepretación bayesiana de la técnica de regularización dropout [Gal and Ghahramani (2016a)], de esta forma habilitaban la cuantificación de la incertidumbre en redes neuronales sin necesidad de recurrir a técnicas bayesianas complejas. El artículo de referencia es complejo a nivel matemático, por ello aquí presentamos la conclusión que concierne a este estudio.

Estos investigadores llegan a la conclusión de que si se define la función de distribución de aproximación Q del variational inference de una red neuronal bayesiana a partir de una Bernoulli su optimización es equivalente a la realizada por una red neuronal regularizada con Dropout. Este resultado, en teoría, nos permite cuantificar la incertidumbre en una red neuronal no bayesiana a partir de la aplicación del dropout en la fase de predicción. El resultado es una distribución de probabilidad generada mediante las disintas simulaciones, en vez de una estimación puntual, de forma que si para cierta entrada todas las predicciones son similares la red está segura de su resultado. En la Figura 3.14 se muestra un ejemplo de uso de esta técnica para la cuantificación de la incertidumbre en la predicción de una serie temporal.

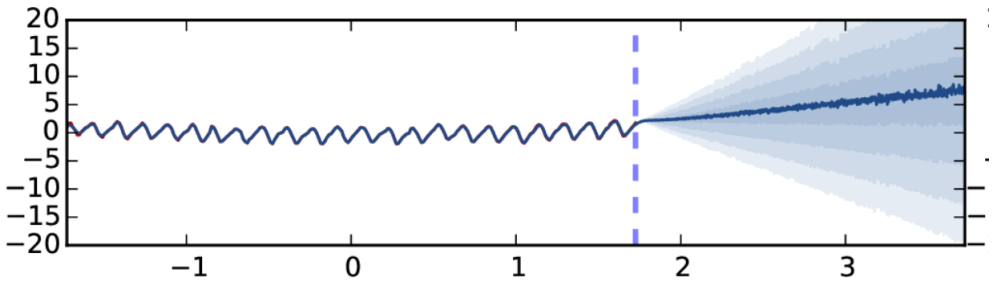

Figura 3.14: Cuantificación de la incertidumbre en datos temporales a través de dropout

Estudios posteriores profundizaron en la aplicación óptima del dropout en redes neuronales para la cuantificación de la incertidumbre. En [Gal and Ghahramani

(2016b)] se propone una metodología de aplicación para redes recurrentes en la que las, en cada pasada, se eliminan neuronas de acuerdo al tipo de información que manejan. Esta metodología se muestra en la Figura 3.15.

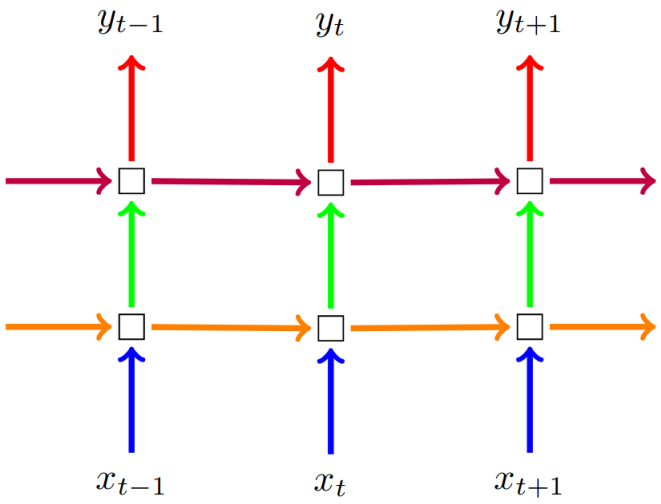

Figura 3.15: Estructura de Dropout para redes recurrentes

Con distintos colores se muestran las distintas máscaras de Dropout aplicadas. Se puede ver como hay mascaras distintas para las entradas de la primera y segunda capa así como para las distintas comunicaciones de recurrencia de la red. También se aplica una máscara distinta para la salida final de la red.

Aunque esta solución aun no está extendida en el campo del deep learning parece ofrecer momentáneamente una solución al problema de la cuantificación de la incertidumbre en los modelos de deep learning.

CAPÍTULO 4

Métodos

En esta sección se describirán los modelos desarrollados en este trabajo para predecir el fenómeno de El Niño 3.4 así como los utilizados en [McDermott and Wikle (2019)]. Estos ultimos modelos serán utililizados a modo de benchmark con el objetivo de comparar la nueva aproximación con la ya publicada. Con el fin de aprovechar la batería de técnicas que ofrece deep learning para datasets espacio-temporales, se han probado tanto redes neuronales tradicionales como redes recurrentes, LSTM o convolucionales. Debido a la gran variedad de modelos desarrollados los nombres con los que se han codificado para facilitar la explicación en la sección 5 vienen en el título de cada subsección.

Antes de explicar las particularidades de cada método conviene destacar algunos procedimientos metodológicos comunes a todos los modelos. Cabe mencionar que el problema en cuestión es la predicción a 6 meses del valor mensual de la SST en la región del Niño 3.4. Por tanto, las capas de salida y de entrada de las redes neuronales están formadas por los puntos de grid localizados en dicha región con un desfase temporal o lead time de 6 meses. A fin de comprobar el efecto del dropout en la cuantificación de la incertidumbre se han probado valores de p iguales a 0.05, 0.1, 0.2, 0.5 y 0.8. Finalmente, todos los modelos se han entrenado con el algoritmo Adam aplicando un criterio de parada o early-stopping cuando el error, en un 10 % de los datos de entrenamiento elegidos al azar (dataset de validación), dejaba de disminuir.

4.1. Modelo lineal: LIN

Se ha realizado un modelo de regresión lineal a fin de servir de referencia contra los modelos no lineales de deep learning. Se ha implementado en Keras para poder introducir early-stopping, regularizándolo.

4.2. Modelo denso: NN

Uno de los modelos desarrollados es una red neural tradicional formada por una única capa densa al estilo de las redes anteriores a la aparición del deep learning. La arquitectura óptima encontrada está compuesta por una capa oculta de 4 neuronas donde se ha usado una función de activación ReLU. Se han probado variaciones de esta arquitectura tratando de introducir más neuronas o más capas intermedias pero no se han conseguido mejorar los resultados, por esto debido al principio de parsimonia hemos decidido quedarnos con la arquitecturas más sencilla.

Para su entrenamiento se han utilizado batches de tamaño 64 con un learning rate de 0.001. El dropout se ha introducido en la capa de entrada de forma que en cada pasada se oculta a la red cierta información del mapa de origen. Se intentó introducir el dropout en la capa intermedia pero debido a las pocas neuronas la convergencia se dificultaba. Para el entrenamiento de las versiones con dropout el learning rate se ha reducido a 0.0001.

4.3. Modelos recurrentes: RNN y LSTM

Para intentar capturar la dinámica temporal de los datos se han utilizado dos modelos de redes neuronales recurrentes: RNN y LSTM. Estos modelos toman como entrada los datos rezagados en el tiempo que, en nuestro caso, se corresponden con los mapas de meses previos al que queremos predecir. Para ambos tipos de modelos se ha probado a utilizar como entrada los 2, 12 y 24 últimos meses observados.

Ambos modelos comparten la misma arquitectura, donde se ha encontrado una configuración de 20 neuronas en la capa oculta como mejor resultado. Si embargo, a diferencia de la LSTM, en la RNN ha sido necesaria la introducción de un regularizador $L2$. En cuanto al entrenamiento, el tamaño de batch usado ha sido de 64 con un learning rate de 0.01. En el caso de RNN el modelo que ha dado mejores resultados ha sido aquel que utilizaba los últimos 12 meses para realizar las predicciones, mientras que en el LSTM era aquel que únicamente tenía una recurrencia de 2 meses. Por tanto, estos modelos han sido los elegidos para ser reentrenados con distintas configuraciones de dropout. Siguiendo el ejemplo de [Gal and Ghahramani (2016b)] hemos añadido dropout tanto en las capas de entrada como en las de recurrencia y la de salida.

4.4. Autoencoders convolucionales: Conv AE

Con el fin de aprovechar la estructura espacial se han probado redes convolucionales. En particular es una configuración con una estructura similar a un autoencoder, donde la capa de entrada se comprime (encoder) mediante convoluciones para luego volver a reconstruirla hasta su resolución original (encoder) mediante deconvoluciones (ver 4.1). La diferencia con los autoencoders radica en que en nuestro caso, la capa de salida y de entrada difieren en que la muestra tiene un lead time de 6 meses.

En concreto, la red está formada por 5 capas ocultas (3 pertenecen al encoder y otras 2 al decoder) donde se han utilizado filtros de tamaño 3x3 y con un stride de 1. El número de filtros en el encoder es de 12, 24 y 48 y en el decoder 24, 12 y 1 siendo este último necesario para poder obtener un resultado con las mismas dimensiones que el mapa a predecir. En todas las capas se ha utilizado la función de actrivación ReLU salvo en la última deconvolución que se aplicado una función lineal. Se ha probado a introducir el tiempo como un mapa de entrada más a la red y de esta manera convolucionar no solo la componente espacial sino también información temporal. Por tanto, se han entrenado distintos modelos añadiendo 1, 2, 12 y 24 meses a la capa de entrada.

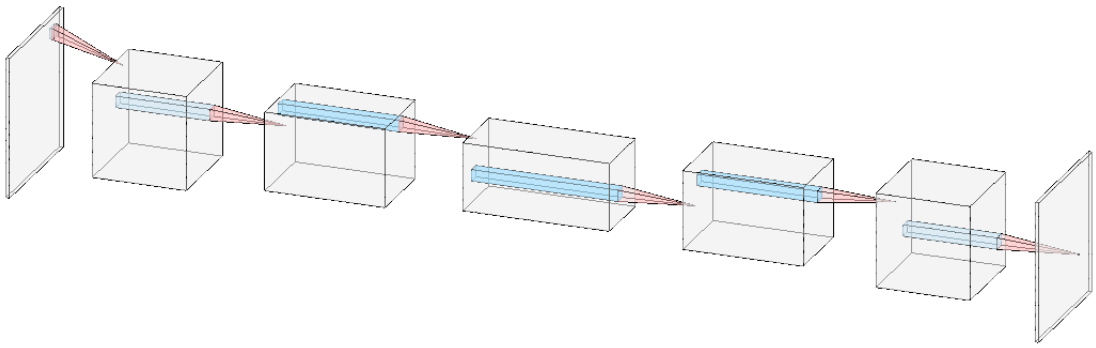

Figura 4.1: Arquitectura del Autoencoder Convolucional

En cuanto al entrenamiento, los batches tenían tamaño 64 y se ha utilizado una learning rate de 0.01. No se ha podido aplicar dropout a ningún modelo de este estilo debido a la ausencia de una implementación en Keras.

4.5. LSTM Convolucionales: Conv LSTM

Hemos visto como nuestros datos poseen estructuras tanto temporales como espaciales. Esto nos lleva a probar la aplicación de LSTM convolucionales, modelos capaces de aprender la dinámica espaciotemporal de los datos. Este modelo ha sido

diseñado con una capa oculta de convolución a la que se le ha aplicado recurrencia mediante una LSTM. Esta capa está a su vez conectada con la capa de salida mediante otra convolución LSTM. La primera capa está compuesta por 12 filtros y la segunda solo por 1. En ambas tenemos un tamaño de filtro 3 x 3 y un stride de 1. Se han entrenado tres modelos con distinto número de meses en la entrada (2, 12 y 24). Se han utilizado batches de tamaño 64 con un learning rate de 0.001. El modelo que mejores resultados ha dado ha sido el que utilizaba como entrada los últimos 12 meses. Por tanto a este modelo se le ha aplicado la misma metodología de dropout que al LSTM simple.

4.6. Modelos de referencia: linear DSTM, GQN, E-QESN y BAST-RNN

En el artículo de referencia [McDermott and Wikle (2019)] se ajustan cuatro modelos distintos a los datos: linear DSTM, GQN, E-QESN y BAST-RNN.

El linear DSTM es un modelo de dinámica espacio-temporal en el que la predicción para un punto es una combinación lineal de los valores de tiempos pasados cercanos en el espacio a ese punto más un término de ruido distribuido según una Normal. Su formulación es la siguiente:

$$Y_{t,i} = \sum_{j=1}^{n_y} a_{i,j} Y_{t-1,j} + \zeta_{t,i}^{(l)} \tag{4.1}$$

El GQN es una versión del linear DSTM a la que se la ha añadido un término de interacción entre valores cercanos en el espacio:

$$Y_{t,i} = \sum_{j=1}^{n_y} a_{i,j} Y_{t-1,j} + \sum_{k=1}^{n_y} \sum_{k=1}^{n_y} b_{i,k,l} Y_{t-1,k} Y_{t-1,l} + \zeta_{t,i}^{(q)} \tag{4.2}$$

El E-QESN [McDermott and Wikle (2017)] es un modelo espacio-temporal basado en Echo State Networks. Estas son unas redes recurrentes pertencientes al campo del Reservoir Computing que se caracterizan por entrenar únicamente los coeficientes de sus capas finales dejando el resto con valores aleatorios. Los defensores de este tipo de redes argumentan que no merece la pena el coste computacional y las posibles divergencias asociadas al entrenamiento de estas capas intermedias recurrentes ya que lo aprendido rara vez supera a la inicialización aleatoria. El modelo E-QESN se ha aplicado con anterioridad a problemas similares al actual y cuantifica bien la incertidumbre, es por esto por lo que deciden incluirlo en el artículo.

Finalmente llegamos al BAST-RNN que es el modelo objeto de estudio en el artículo. Es una red neuronal recurrente con la capa intermedia definida de la siguiente forma:

$$\mathbf{h}_t = f(\frac{\delta}{\mid \lambda_w \mid}\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t) \tag{4.3}$$

y donde la salida se calcula con la siguiente expresión:

$$Y_t = \boldsymbol{\mu} + \mathbf{V}_1\mathbf{h}_t + \mathbf{V}_1^2\mathbf{h}_t^2 + \epsilon_t, \quad \epsilon_t \sim Gau(0, \mathbf{R}_t) \tag{4.4}$$

En (4.3) la matriz de parámetros $\mathbf{W}$ se reescala con un coeficiente calculado a partir de ciertas propiedades espectrales de $\mathbf{W}$ a fin de evitar comportamientos inestables en el modelo. La salida es la suma de un vector $\boldsymbol{\mu}$ que actua como bias, el estado oculto multplicado por una matriz de parámetros, el estado oculto al cuadrado multiplicado por otra matriz de parámetros y un ruido que sigue una distribución normal. Con la inclusión de los estados ocultos al cuadrado se busca encontrar un mayor número de relaciones no lineales entre los datos.

Sobre estas matrices de parámetros se imponen unas distribuciones a priori para, una vez observados los datos, calcular sus respectivas distribuciones a posteriori. El cálculo de está distribución se hace a través del algoritmo MCMC. Esto hace que la red neuronal recurrente presentada en el artículo sea una red neuronal recurrente bayesiana y, por lo tanto, tenga sentido enfrentarla (en terminos de cuantificación de la incertidumbre) a nuestras redes neuronales recurrentes con dropout.

4.7. Metodología

4.7.1. Zona de predicción

En el artículo de referencia [McDermott and Wikle (2019)] se evaluan los modelos sobre dos zonas distintas. La primera cubre una zona amplia del Pacífico mientras que la segunda cubre únicamente la zona de El Niño 3.4. Como bien hemos comentado en apartados anteriores nosotros únicamente trabajaremos sobre la segunda zona por lo que los resultados presentados en los próximos apartados serán aplicables únicamente a la zona de El Niño 3.4.

4.7.2. Generación de intervalos de confianza

La generación de los intervalos de confianza se ha llevado a cabo mediante la aplicación del dropout en la fase de predicción. A continuación se describe brevemente el proceso de generación:

- Para cada dato de entrada se realizan 100 predicciones activando el dropout en la fase de predicción
- Sobre estas predicciones se calculan los percentiles 5, 50 y 95
- Con estos percentiles se construyen los intervalos de confianza

4.7.3. Estructuración de los datos

A fin de generar unas comparaciones justas utilizaremos la misma partición de los datos en train y test que las del artículo de referencia [McDermott and Wikle (2019)]. Para entrenar el modelo (train) se utilizarán los meses comprendidos entre las fechas de Enero de 1970 y Agosto de 2014, mientras que las predicciones (test) se realizarán sobre los meses incluidos en el periodo de Febrero de 2015 a Diciembre de 2016. En este último conjunto de datos los meses se seleccionan de forma alterna (de dos en dos). Estos periodos no se han seleccionado de forma aleatoria si no que se buscaba que el conjunto de test incluyera el valor extremo del ENSO ocurrido entre los años 2015 y 2016. Este suceso no se consiguió predecir con los modelos vigentes en la fecha por lo que su modelización es de gran interés.

Es importante destacar el lead time presente en la estructuración de los datos. Las predicciones siempre se realizan con un lead time de 6 meses, es decir, que para por ejemplo hacer la predicción de Julio de 2016 estamos obligados a utilizar meses anteriores a Enero de 2016.

4.7.4. Evaluación de los modelos

A fin de catalogar las predicciones de los distintos modelos como buenas o malas necesitamos medidas de evaluación acordes a nuestro problema. Utilizaremos las medidas de evaluación usadas en el artículo de referencia: MSPE y CRPS.

MSPE

Estas siglas hacen referencia al término Mean Squared Prediction Error que, aunque tiene distinto nombre, es equivalente al MSE (Mean Squared Error).

Dado un conjunto de predicciones $\hat{y}_1, \hat{y}_2, ..., \hat{y}_N$ y los respectivos valores a estimar $y_1, y_2, ..., y_N$ el MSE se define como:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{4.5}$$

Esta métrica se interpreta como el promedio de los errores al cuadrado. En el caso específico de nuestro problema se interpretaría como el promedio de los errores entre la predicción de nuestros modelos y la anomalía real existente en el mapa.

CRPS

El MSPE resulta útil para evaluar la diferencia entre el valor real y nuestra predicción puntual sin embargo no nos vale para evaluar como de correcto es nuestro intervalo de confianza con respecto al valor real. La medida de evaluación utilizada en estos casos es la CRPS (continous ranked probability score) [Matheson and Winkler (1976)].

Siendo F la función de distribución de nuestro intervalo de confianza y h el valor real a estimar el CRPS se define de la siguiente manera:

$$CRPS(F,h) = \int_R (F(r) - 1\{r \geq h\})^2 dr \tag{4.6}$$

donde 1 es la función escalón de Heaviside. El CRPS se expresa en la mismas unidades que las variables sobre las que se calcula y si el intervalo estuviese compuesto de un solo valor se reduciría al MSE.

Esta medida de evaluación es capaz de cuantificar tanto la precisión como la distribución del intervalo entorno al valor real, esto la convierte en una métrica idónea para nuestro objetivo de cuantificar la calidad de los intervalos de confianza generados.

CAPÍTULO 5

Resultados

5.1. Intercomparación

Comenzaremos la exposición de los resultados mostrando la Tabla 5.1 en la que se comparan en términos de MSPE y CRPS nuestros modelos y los ajustados en el artículo de referencia. Los modelos resaltados en negrita se corresponden con los del artículo de referencia.

Los modelos se encuentran ordenados respecto al MSPE. EL CRPS solo se muestra para aquellos modelos con dropout ya que, como hemos visto, este es necesario para la generación de los intervalos de confianza. Para cada tipo de modelo se muestra únicamente un modelo con dropout que se corresponde al $p = 5\,\%$. Se ha decidido utilizar este valor ya que en la mayoría de casos es el que mejores resultados ofrecía.

Ninguno de nuestros modelos ha conseguido superar al BAST-RNN pero sí al E-QESN. La red neuronal densa, uno de los modelos más sencillos, ha sido el que más se ha acercado al BAST-RNN. En la Tabla 5.1 se ve como los convolucionales LSTM son los que consiguen unos peores resultados, también se puede apreciar que las versiones regularizadas con dropout siempre consiguen unos peores resultados que los modelos en los que se basan.

Las predicciones para el índice de El Niño pueden verse con más detalle en la figura 5.1, donde se muestran tanto la observación como la predicción para los distintos métodos. Mientras que algunos métodos no han sido capaces de aprender ninguna relación, como los LSTM convolucionales, si cabe destacar la capacidad predictora del resto de modelos. Por ejemplo, al contrario que los métodos existentes, la red

Modelo	Niño 3.4 MSPE	Niño 3.4 CRPS
BAST-RNN	**0.193**	**0.290**
NN	0.241	-
LSTM t=2	0.259	-
E-QESN	**0.261**	**0.354**
RNN t=12	0.291	-
LIN	0.357	-
RNN t=24	0.4	-
Conv AE t=2	0.401	-
RNN t=2	0.427	-
LSTM t=24	0.438	-
Conv AE t=1	0.44	-
LSTM t=2 w/ DP=5 %	0.491	0.534
RNN t=12 w/ DP=5 %	0.503	0.469
GQN	**0.619**	**0.538**
Conv AE t=12	0.757	-
NN w/ DP=5 %	0.766	0.689
Lin. DSTM	**0.785**	**0.699**
LSTM t=12	0.839	-
Conv LSTM t=12	0.874	-
Conv AE t=24	0.936	-
Conv LSTM t=2	1.015	-
Conv LSTM t=12 w/ DP=5 %	1.408	0.94
Conv LSTM t=24	1.801	-

Tabla 5.1: Comparación en términos de MSPE y CRPS de los distintos modelos ajustados. En negrita los de referencia. La t hace referencia al número de meses utilizados como recurrencia en la capa de entrada. El DP es el valor del hiperparámetro p del dropout.

neuronal densa es capaz de predecir el valor anormalmente alto que tuvo El Niño de Febrero del 2016. El mismo comportamiento se observa en las redes recurrentes y en las LSTM donde el valor se aproxima al valor observado para ese mes. Destaca como la red neuronal densa, el LSTM y el Autoencoder Convolucional producen unas predicciones que subestiman el valor original en el mismo intervalo de tiempo (mediados de 2015).

Podemos observar con más detalle espacial la predicción del modelo denso para el mes de Diciembre de 2015 en la Figura 5.2. Tanto la observación como la predicción muestran similares patrones espaciales, con una anomalía cálida generalizada en toda la región. Parece que este modelo ha conseguido aprender de forma precisa la distribución de las anomalías en la zona.

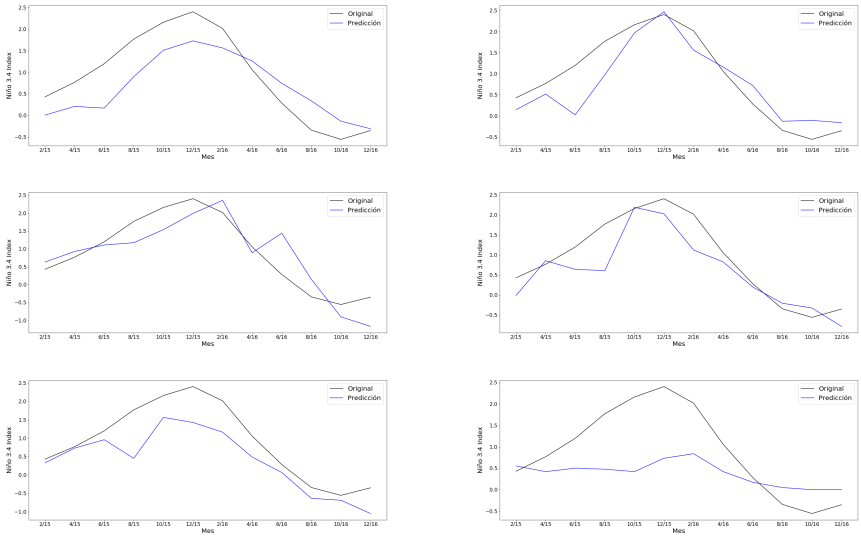

Figura 5.1: Predicciones respecto a los valores originales. En la columna de la izquierda de arriba a abajo se muestran las correspondientes a la red lineal, red recurrente y autoencoder convolucional. En la columna de la derecha de arriba a abajo se muestran las de la red densa, LSTM y LSTM convolucional.

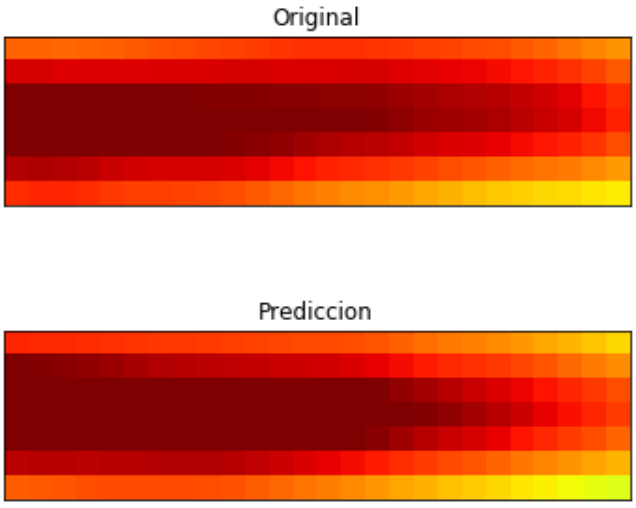

Figura 5.2: Temperatura en la superficie del mar observada (arriba) y predicha por el modelo denso (abajo), en la región del Niño 3.4 (Diciembre 2015).

5.2. Dropout: Estimación de la incertidumbre

En esta sección se inspeccionarán los intervalos de confianza de los distintos modelos ajustados. El dropout se ha aplicado al mejor modelo de cada tipo de acuerdo al valor de t. En la columna de la izquierda de la Figura 5.3 se puede apreciar el intervalo generado por $p = 5\,\%$ y en la de la derecha los intervalos generados por distintos valores de p.

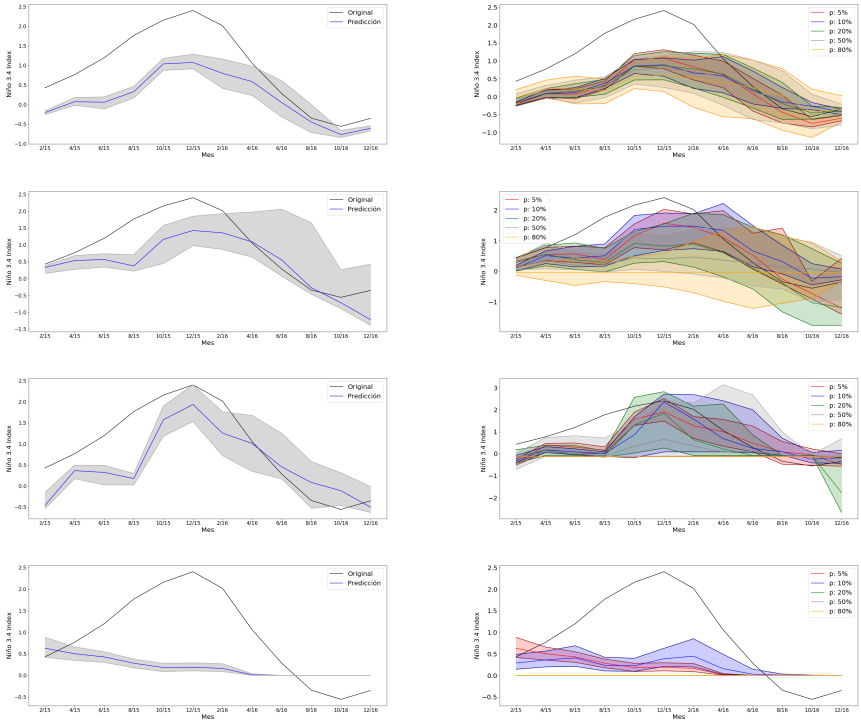

Figura 5.3: En la columna de la izquierda de arriba a abajo se muestran los intervalos de confianza con $p = 5\,\%$ para la red densa, red recurrente, LSTM y LSTM convolucional. En la columna de la derecha los intervalos de confianza para distintos valores de p

La predicción y el intervalo de la LSTM convolucional indican que, bajo dropout, esta red no es capaz de aprender nada de los datos. El intervalo de confianza de la red densa, red recurrente y LSTM parece mostrar un ensanchamiento en la misma zona (últimos meses del 2016), casualmente en el caso de la red recurrente este ensanche guarda relación con la sobrestimación de la predicción puntual en ese mismo periodo. El intervalo de confianza de la LSTM consigue capturar la anomalía de El Niño de Diciembre de 2015. Todos los modelos pierden capacidad predictiva con la inclusión del dropout.

Los distintos intervalos de confianza de acuerdo a p de la red neuronal densa parecen mostrar un ensanchamiento proporcional al valor de este hiperparámetro. Se observa algo parecido para la red recurrente con la diferencia de que para valores altos el intervalo no ajusta bien a los datos originales. La LSTM solo muestra un intervalo de confianza con una forma similar a la curva original para $p = 5\,\%$, el resto no ajustan bien. La LSTM convolucional no consigue generar ningún intervalo de confianza válido para ningún valor de p.

5.3. Efecto del dropout en la capacidad predictiva

Finalmente en la Figura 5.4 se ve como afecta el valor del hiperparámetro p al valor del MSPE (MSE) y CRPS.

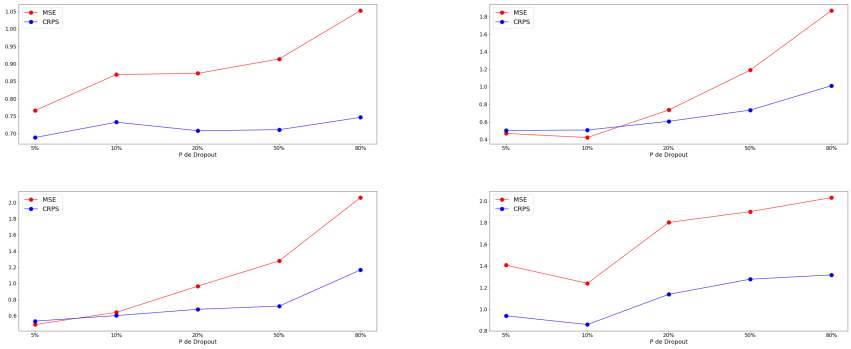

Figura 5.4: Evolución del MSE Y CRPS en función de la p del dropout. En la columna izquierda de arriba a abajo la correspondiente a la red densa y LSTM. En la columna derecha la de la red recurrente y LSTM convolucional

La tendencia general es del aumento de ambas métricas de acuerdo al valor de p además de valores más altos del MSPE respecto al CRPS. La red densa es la que menos replica este comportamiento llegando a mantener unos valores estables de las métricas para algunos valores de p.

Los modelos que recogen dependecias temporales (la red recurrente y la LSTM) presentan un comportamiento muy parecido. El MSPE y CRPS en estos dos modelos presentan valores similares para p bajos mientras que para p altos la diferencia aumenta de forma significativa. La LSTM convolucional presenta un crecimiento constanste de ambas métricas con una bajada repentina para $p = 10\,\%$.

CAPÍTULO 6

Conclusiones y Discusión

Uno de los principales aspectos a destacar de los distintos modelos entrenados es que los más sencillos son los que obtienen unos mejores resultados. La red neuronal densa, no diseñada de forma explicita para capturar dinámicas espacio-temporales, es la que reporta unos mejores resultados seguida de cerca por una LSTM con tan solo 2 meses de datos de entrada. Este comportamiento probablemente se deba a la limitación en la cantidad de datos disponibles.

Con la red neuronal densa se ha logrado predecir de forma casi perfecta la anomalía del ENSO de Diciembre de 2015. Según el artículo de referencia no se había conseguido con anterioridad capturar este evento con ningún modelo por lo que el resultado alcanzado en el presente trabajo se puede considerar como pionero.

Los modelos recurrentes consiguen, por norma general, capturar la estructura temporal de los datos ya que sus resultados están incluidos entre los mejores del actual trabajo (una red LSTM es la segunda mejor y una RNN la tercera).

Los modelos convolucionales no consiguen aprender la estructura espacial de los datos. El ajuste de estos modelos ha sido complicado y ha necesitado de muchos experimentos distintos. Quizás sean estos tipos de modelos los que más afectados se han visto por la poca cantidad de datos disponibles.

La generación de intervalos de confianza parece estar muy condicionada al valor p del Dropout. Valores muy altos ensanchan demasiado los intervalos y hacen que la cuantificación de la incertidumbre sea menos precisa. Esta problemática puede

que sea exclusiva del problema aquí tratado y que guarde relación con la poca cantidad de datos disponibles para el entrenamiento. Los intervalos de confianza de los modelos recurrentes y LSTM son los que mejor han conseguido mantener la forma respecto a la curva original que buscaban ajustar.

Efectivamente los intervalos nos han ayudado a identificar rangos en los que el modelo está más seguro de sus predicciones lo que nos puede aportar algunas ventajas interesantes:

- Nos arroja luz acerca del funcionamiento interno de la red, identificando rangos de datos en los que el modelo no ha terminado de aprender la relación que los subyace.
- Nos puede ayudar en la óptima toma de datos. Si un modelo está seguro de sus predicciones en cierto rango de datos no será necesaria la recogida de más observaciones lo que nos permitiría centrarnos en la recogida en rangos en los que el modelo presente una menor seguridad.

En conclusión, en este trabajo hemos estudiado la capacidad del bayesian deep learning para cuantificar la incertidumbre en el problema de forecasting de El Niño. Ningún modelo de deep learning ha alcanzado al benchmark en términos de error, a pesar de haberse situado muy cerca. Esto probablemente se deba la falta de datos, haciendo más eficientes en esta aplicación métodos de machine learning menos parametrizados. Sin embargo, El Niño de diciembre de 2015 ha sido mejor predicho por las redes neuronales. Con el dropout se ha conseguido dar una estimación de la incertidumbre, sin embargo el efecto de añadir dropout a la red penaliza su capacidad predictiva, de nuevo por la limitación del dataset.

Bibliografía

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. 5(2):157–166.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Cao, L.-J. and Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. 14(6):1506–1518.

Charney, J. G., Fjörtoft, R., and Neumann, J. v. (1950). Numerical integration of the barotropic vorticity equation. 2(4):237–254.

Dong, C., Loy, C. C., He, K., and Tang, X. (2015). Image super-resolution using deep convolutional networks. 38(2):295–307.

Fan, H., Zhu, L., and Yang, Y. (2019). Cubic lstms for video prediction.

Gal, Y. (2016). *Uncertainty in deep learning*. Ph.D. thesis, PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016a). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059.

Gal, Y. and Ghahramani, Z. (2016b). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027.

Giffard-Roisin, S., Yang, M., Charpiat, G., Kégl, B., and Monteleoni, C. (2018). Deep learning for hurricane track forecasting from aligned spatio-temporal climate datasets.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. 9(8):1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. 79(8):2554–2558.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. 4(2):251–257.

Kendall, A., Badrinarayanan, V., and Cipolla, R. (2015). Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding.

Kiefer, J., Wolfowitz, J., et al. (1952). Stochastic estimation of the maximum of a regression function. 23(3):462–466.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105.

Kumar, A. and Rastogi, R. (2019). Attentional recurrent neural networks for sentence classification. In *Innovations in Infrastructure*, pp. 549–559. Springer.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. 521(7553):436.

LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pp. 319–345. Springer.

LeCun, Y., Touresky, D., Hinton, G., and Sejnowski, T. (1988). A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, vol. 1, pp. 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann.

Liang, J.-Z. (2004). Svm multi-classifier and web document classification. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826)*, vol. 3, pp. 1347–1351. IEEE.

Lin, Y., Sarvi, M., Khoshelham, K., and Haghani, M. (2018). Predicting traffic congestion maps using convolutional longshort-term memory. In *International Symposium of Transport Simulation*, pp. 15–25.

Liu, Y., Racah, E., Correa, J., Khosrowshahi, A., Lavers, D., Kunkel, K., Wehner, M., Collins, W., et al. (2016). Application of deep convolutional neural networks for detecting extreme weather in climate datasets.

MacKay, D. J. (1992). A practical bayesian framework for backpropagation networks. 4(3):448–472.

Makridakis, S. and Hibon, M. (2000). The m3-competition: results, conclusions and implications. 16(4):451–476.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). The m4 competition: Results, findings, conclusion and way forward. 34(4):802–808.

Mani, S., Chen, Y., Elasy, T., Clayton, W., and Denny, J. (2012). Type 2 diabetes risk forecasting from emr data using machine learning. In *AMIA annual symposium proceedings*, vol. 2012, p. 606. American Medical Informatics Association.

Matheson, J. E. and Winkler, R. L. (1976). Scoring rules for continuous probability distributions. 22(10):1087–1096.

McDermott, P. L. and Wikle, C. K. (2017). An ensemble quadratic echo state network for non-linear spatio-temporal forecasting. 6(1):315–330.

McDermott, P. L. and Wikle, C. K. (2019). Bayesian recurrent neural network models for forecasting and quantifying uncertainty in spatial-temporal data. 21(2):184.

Michelmore, R., Kwiatkowska, M., and Gal, Y. (2018). Evaluating uncertainty quantification in end-to-end autonomous driving control.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Rhee, J. and Im, J. (2017). Meteorological drought forecasting for ungauged areas based on machine learning: Using long-range climate forecast and remote sensing data. 237:105–122.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. 61:85–117.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. 15(1):1929–1958.

van Oldenborgh, G. J., Balmaseda, M. A., Ferranti, L., Stockdale, T. N., and Anderson, D. L. (2003). Did the ecmwf seasonal forecast model outperform a statistical model over the last 15 years.

Vandal, T., Kodra, E., Ganguly, S., Michaelis, A., Nemani, R., and Ganguly, A. R. (2017). Deepsd: Generating high resolution climate change projections through single image super-resolution. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1663–1672. ACM.

Wikle, C. K. (2015). Modern perspectives on statistics for spatio-temporal data. 7(1):86–98.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pp. 802–810.