Modelos de aprendizaje profundo

1.1. Introducción teórica

Las redes neuronales son modelos computacionales cuyo funcionamiento se inspira en el funcionamiento de las neurona cerebrales reales cuya función principal es recibir, procesar y transmitir información a través de señales químicas y eléctricas. En líneas generales las neuronas se especializan en la recepción de estímulos y generación de impulsos entre ellas mediante conexiones llamadas sinapsis. Cuando una neurona recibe un impulso eléctrico que la activa, esta dispara un impulso que a su vez estimula a otras neuronas a las que esta conectada. Algunas de estas neuronas a su vez se activan y se disparan activando a otras y así sucesivamente propagando los estímulos por toda la red. Entre más se practique una tarea, las conexiones que se generan entre las neuronas se hacen más robustas, esto es el proceso de aprendizaje.

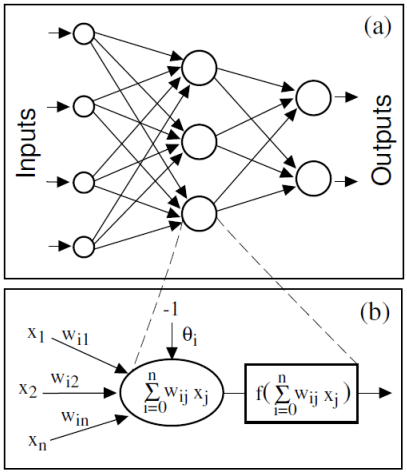


Figura 1.1: Representación gráfica de una neurona artificial (figura tomada de [1]).

Una neurona artificial (Fig. 1.1-b) funciona de una manera muy similar a las neuronas biológicas, ésta recibe datos de entrada que son combinados por la neurona asignando pesos a cada uno de ellos, a los cuales se aplica una función de activación que activa a la neurona si esta supera un umbral dado. La función de activación además de cumplir la función de activar a las neuronas, capta las características no lineales de los datos.

Una red neuronal consiste en un conjunto de neuronas conectados entre sí (Fig. 1.1-a), que al igual que en el caso de las neuronas reales son capaces de

ser entrenadas para aprender a realizar una tarea dada generando diferentes conexiones entre ellas.

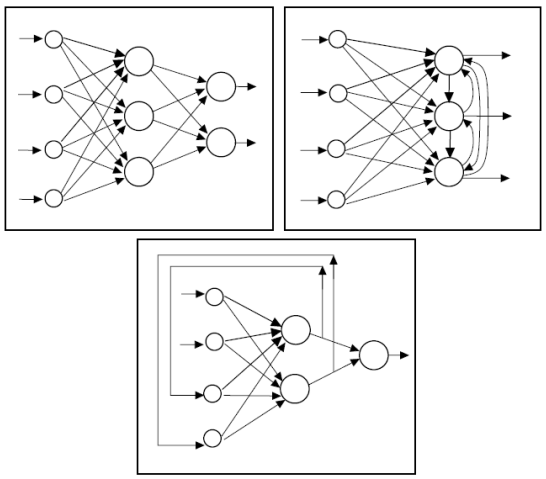


Figura 1.2: Tres topologías distintas de red neuronal: (a) multicapa, (b) competitivas, (c) recurrentes.

La estructura de la red puede tener distintas topologías, en la figura 1.2 se muestran tres modelos de red neuronales distintos: *re multicapa* que solo tiene conexiones entre neuronas de capas consecutivas, *red competitiva* que también posee conexiones entre de la última capa y *redes recurrentes* (RNN) que posee conexiones entre capas no consecutivas. En este trabajo se han utilizado tanto redes multicapa como redes recurrentes, que son apropiadas para problemas de aprendizaje supervisado en donde cada patron de entrenamiento (entrada de la red) $X_p = (x_{1p}, ...x_{mp})$ tiene asociado un correspondiente patrón de salida $Y_p = (y_{1p}, ...y_{np})$. El entrenamiento se basa en que la red sea capaz de reproducir los patrones de salida con el menor error posible.

El proceso de aprendizaje de la red consiste en la aplicación de métodos de optimización matemáticos para obtener los pesos w_{ij} que minimizan una cierta función de error o *loss*. Los algoritmos más populares se basan en minimizar el error cuadrático medio (RMSE):

$$E(w) = \sum_{j,p} (y_{jp} - \hat{y}_{jp})^2 \tag{1.1}$$

Uno de los algoritmos más simples es el de descenso de gradiente que en cada etapa intenta modificar los pesos de forma incremental de manera de minimizar la función de loss. El incremento de los pesos se obtiene en base al vector opuesto al gradiente del loss, que indica la dirección en la que la

función decrece más rápidamente:

$$\Delta w_i j = -\alpha \frac{\partial E(w)}{\partial w_i j} \tag{1.2}$$

donde α es la tasa de aprendizaje.

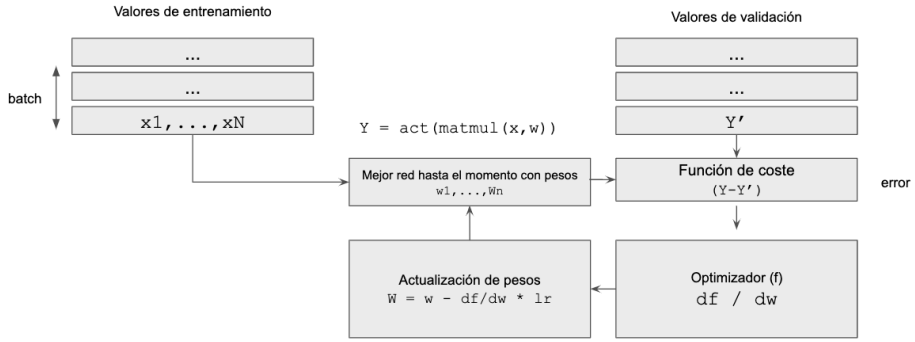


Figura 1.3: Back propagation

El algoritmo de retro-propagación consta de dos pasos, en primer lugar la entrada X_p se propaga hacia adelante, obteniendo el valor de todas las unidades ocultas y las salidas $\hat{y}_p$ y por lo tanto el error asociado a estas. Los valores obtenidos se utilizan para actualizar los pesos de la capa de salida, y estos se propagan hacia atrás para actualizar los pesos de las capas ocultas. Esto se repite para cada patrón de entrenamiento. En el esquema 1.3 se resume todo el proceso.

1.2. Redes recurrentes con memoria a largo plazo

Las redes neuronales recurrentes con memoria a largo plazo (LSTM) son redes RNN que poseen conexiones entre capas no consecutivas (Fig. 1.2-c) que además incluyen una conexión extra entre las neuronas dedicadas a almacenar información para largo períodos de tiempo. Esta "memoria a largo plazo"soluciona el problema del desvanecimiento que presentan las redes recurrentes tradicionales[3], [2].

En la figura 1.4 se puede observar la estructura de una celda de una red recurrente, las celdas amarillas son capas de neuronas y los círculos rosas son puntos en donde se realiza una cierta operación. La clave en estas redes es la línea horizontal superior que recorre la cadena casi sin sufrir modificaciones y permite que la información fluya a través de la red. Mediante las cuatro entradas (celdas amarillas en el esquema), estas redes poseen la habilidad de eliminar o agregar información al estado de la celda y decidir qué parte de la información proveniente de capas anteriores es relevante [2].

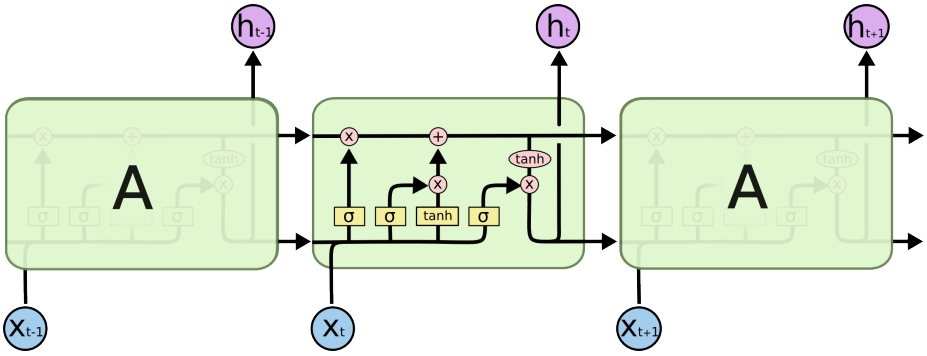


Figura 1.4: Red neuronal recurrente con memoria a largo y corto plazo (LSTM). Figura tomada de [2].

1.3. Topología de los modelos utilizados

Se han considerados 3 modelos que poseen una estructura básica que consta de una capa de entrada, dos o tres capas ocultas y una capa de salida:

1. **Modelo Denso**: En primer lugar se ha considerado un modelo que consta de dos capas densas, en la figura 1.5 podemos ver un resumen con las principales características de las mismas.

2. **Modelo LSTM1**: En segundo lugar se ha considerado un modelo que contiene una capa LSTM y una segunda capa densa (Fig. 1.6).

3. **Modelo LSTM2**: Por último se ha considerado un tercer modelo que consta de dos capas LSTM y una capa densa (Fig. 1.7) que se ha entrenado de manera secuencial.

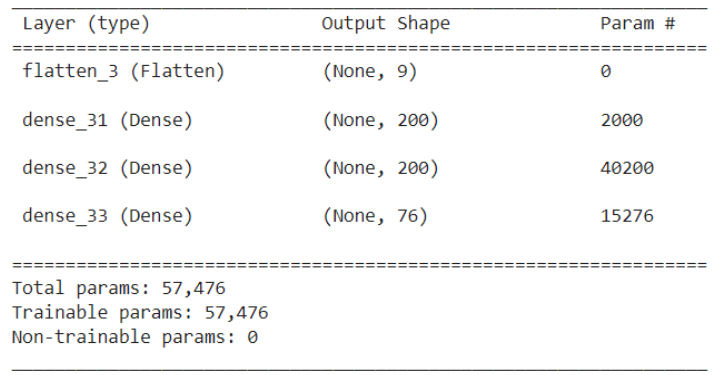
1.3.1. Regularización de la función de coste

Se han considerado dos tipos de regularizaciones, L1 y L2 que penalizan el loss añadiendo los siguientes términos para cada caso:

$$loss + \lambda \sum |\omega_i| \quad (L1) \tag{1.3}$$

$$loss + \lambda \sum \omega_i^2 \quad (L2) \tag{1.4}$$

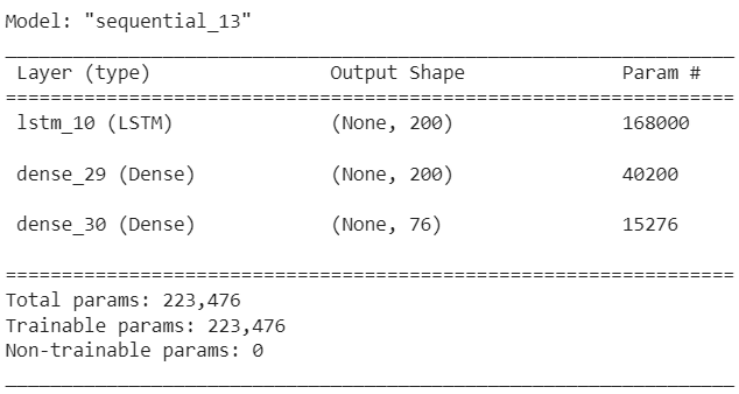
La función de estos términos extra es la de reducir el valor de los parámetros haciendo incluso que la influencia de algunas variables de entrada sean nula en la salida de la red, lo que da lugar a una selección de variables de forma natural y reduce el posible sobreajuste de los datos. El valor de λ varía entre 0 y 1 y controla la magnitud de la reducción de los parámetros.



Layer (type)	Output Shape	Param #
flatten_3 (Flatten)	(None, 9)	0
dense_31 (Dense)	(None, 200)	2000
dense_32 (Dense)	(None, 200)	40200
dense_33 (Dense)	(None, 76)	15276

Total params: 57,476
Trainable params: 57,476
Non-trainable params: 0

Figura 1.5: Modelo Denso.

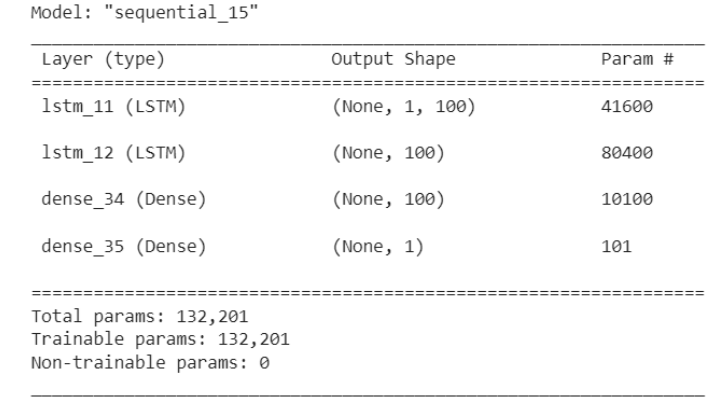


Model: "sequential_13"

Layer (type)	Output Shape	Param #
lstm_10 (LSTM)	(None, 200)	168000
dense_29 (Dense)	(None, 200)	40200
dense_30 (Dense)	(None, 76)	15276

Total params: 223,476
Trainable params: 223,476
Non-trainable params: 0

Figura 1.6: Modelo LSTM1.



Model: "sequential_15"

Layer (type)	Output Shape	Param #
lstm_11 (LSTM)	(None, 1, 100)	41600
lstm_12 (LSTM)	(None, 100)	80400
dense_34 (Dense)	(None, 100)	10100
dense_35 (Dense)	(None, 1)	101

Total params: 132,201
Trainable params: 132,201
Non-trainable params: 0

Figura 1.7: Modelo LSTM2.

1.3.2. Funciones de activación

En todas las capas se ha considerado la misma función de activación f, y se han entrenado los modelos considerando las opciones mostradas en 1.1.

$f(x) = \frac{1}{e^{-x}+1}$ (sigmoide)	$f(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{si } x > 0 \end{cases}$ $(relu)$
$f(x) = \frac{e^{2x}-1}{e^{2x}+1}$ $(tanh)$	$f(x) = x$ $(linear)$

Cuadro 1.1: Funciones de activación

1.3.3. Optimizadores

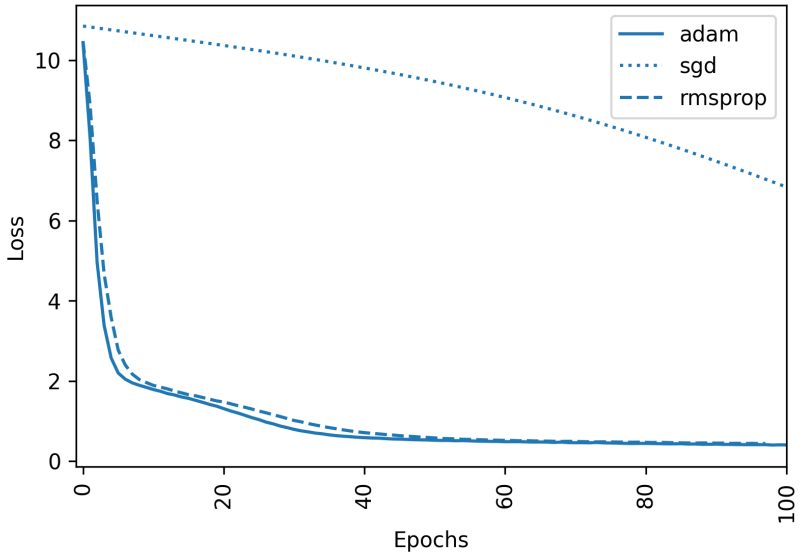


Figura 1.8: Evolución de la función de loss considerando diferentes optimizadores.

Dado el gran número de diferentes pesos y observaciones, el cálculo de la derivada parcial de la función de coste respecto a cada uno de los pesos de la red para cada observación es inviable. Existen diferentes métodos que optimización este proceso y agilizan los cálculos. En este trabajo se ha hecho una exploración inicial manual para evaluar la performance de los

optimizadores "*Stochastic Gradient Descent*"(SGD) [4], "*Adaptive moment estimation*"(Adam) [5] y *Root Mean Square Propagation* (RMSprop) [4].

En la figura 1.8 se muestra a modo de ejemplo la evolución de la función de loss considerando cada uno de los optimizadores al entrenar el modelo LSTM1, se puede ver que la taza de aprendizaje con adam y rmspror es considerablemente mayor que con sgd, ya que en los primeros casos la función de loss alcanza su valor de equilibrio en aproximadamente 60 épocas, mientras que sgd necesita más de 200.

1.3.4. Calibración de los modelos

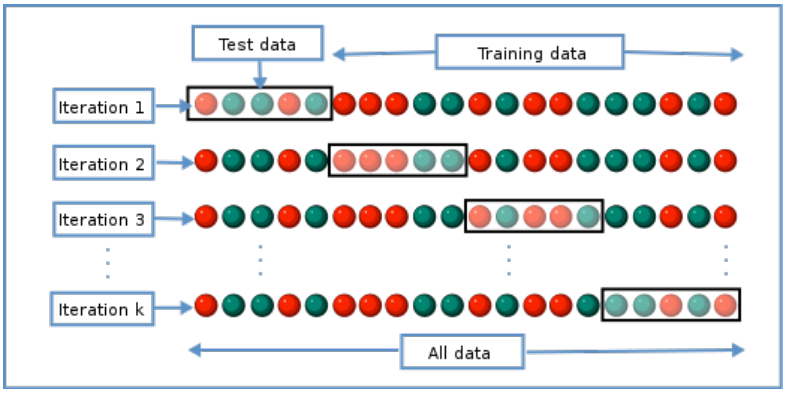


Figura 1.9: k-fold cross validation (imagen de Gufosowa - WikiMedia).

Los modelos se calibran realizando una búsqueda exhaustiva de hyper-parámetros utilizando el con el método "GridSearchCV"disponible en la librería scikit-learn [6]. Éste método crea una grilla con los hyper-parámetros del modelo y optimiza sus valores realizando una búsqueda con validación cruzada, en donde el conjunto de datos se divide en datos de entrenamiento y de validación (train/test en la figura 1.9). Cómo este conjunto se divide y cuantas veces se realiza el proceso, se controla mediante el número de pliegues o folds, en general denotado con la letra k. El modelo utiliza el primer fold en la primera iteración como conjunto de test y los siguientes k-1 como conjunto de train. Este proceso se repite k veces para cada uno de las posibles combinaciones de hyper-parámetros presentes en la grilla y arroja como resultado final un valor promedio del escore utilizado para evaluar la calidad del modelo.

En la tabla 1.2 se muestran los hyper-parámetros explorados para entrenar los diferentes modelos. En el caso del entrenamiento global realizado para los modelos denso y LSTM1 PCA, tres de estos parámetros (el número de neuronas, la función de activación y λ) han sido optimizados mediante el método GridSearchCV.

Modelo	Parámetros	valores
Denso	solver	rmsprop, adam, sgd
	Neuronas	10, 80, 200
	activación	sigmoid,relu,tanh,linear
	alpha	0.0001,0.0002,0.0003
	Regularización	l1,l2 $\lambda = 0{,}0001, 0{,}001, 0{,}1$
LSTM1 PCA	solver	rmsprop, adam, sgd
	Neuronas	10, 80, 200
	activación	sigmoid,relu,tanh,linear
	alpha	0.0001,0.0002,0.0003
	Regularización	l1,l2 $\lambda = 0{,}0001, 0{,}001, 0{,}1$
LSTM1 indv	solver	rmsprop, adam, sgd
	Neuronas	10, 80, 200
	activación	sigmoid,relu,tanh,linear
	alpha	0.0001,0.0002,0.0003
	Regularización	l1,l2 $\lambda = 0{,}0001$
LSTM2	solver	rmsprop, adam, sgd
	Neuronas	10, 80, 200
	activación	sigmoid,relu,tanh,linear
	alpha	
	Regularización	l1,l2 $\lambda = 0{,}0001$

Cuadro 1.2: Hyper parámetros explorados en los diferentes modelos.

1.4. Entrenamiento

Los modelos han sido entrenados durante un número de épocas o pasos igual a 200 y se ha utilizado el callback "*early stopping*çon paciencia 3, que interrumpe la ejecución cuando el valor del loss en el conjunto de validación aumenta durante tres iteraciones sucesivas. Se han considerado tres métodos diferentes de entrenamiento que se describen a continuación.

1.4.1. Entrenamiento global

Los modelos denso y LSTM1 han sido entrenados sobre una matriz que contiene las entradas o variables características (series temporales de precipitación, temperatura máxima y temperatura mínima) para todas las subcuencas que constituyen la cuenca hidrológica Chambo:

$$X_{comp} = \begin{bmatrix} p_{1,1} & T_{1,1}^{max} & T_{1,1}^{min} & p_{1,2} & T_{1,2}^{max} & T_{1,2}^{min} & ... & p_{1,nc} & T_{1,nc}^{max} & T_{1,nc}^{min} \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ p_{nt,1} & T_{nt,1}^{max} & T_{nt,1}^{min} & p_{nt,2} & T_{nt,2}^{max} & T_{nt,2}^{min} & ... & p_{nt,n} & T_{nt,n}^{max} & T_{nt,nc}^{min} \end{bmatrix}$$

Donde n_t es el número de pasos temporales y nc es el número de subcuencas. Las variables objetivo son los caudales naturales simulados con el modelo hidrológico LEM y se encuentran almacenados en una matriz Y_{comp} estructurada de la siguiente manera:

$$Y_{comp} = \begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,2}\cdots & q_{1,nc} \\ \cdots & \cdots & \cdots & \cdots \\ q_{nt,1} & q_{nt,2} & q_{nt,2}\cdots & q_{nt,nc} \end{bmatrix}$$

El paso de tiempo que se ha considerado es mensual y el rango temporal total es de 20 años (desde el año 2000 hasta el año 2020). Por otro lado la cuenca se encuentra compuesta por 76 subcuencas, entonces las dimensiones de X_{comp} e Y_{comp} son (229,228) y (229,76), respectivamente. Finalmente, los datos han sido divididos en conjuntos de train y test tomando los primeros 160 como conjunto de train y los últimos 69 como conjunto de test.

Para el caso de capas LSTM es necesario agregar una dimensión a las matrices de entrada que toma en cuenta la correlación temporal de las muestras. La estructura de las matrices de entrada toman la forma (*muestras*, *pasos de tiempo*, *características*). En el modelo LSTM1 se considera un paso de tiempo por cada muestra, por lo cual las matrices toman la dimensión (229,1,228) y (229,1,76), respectivamente.

1.4.2. Entrenamiento local

Por otro lado el modelo LSTM1 también se ha entrenado individualmente cuenca por cuenca, para lo cual se han utilizado matrices X_{ind} e Y_{ind} con las siguientes formas:

$$X_{indv,id} = \begin{bmatrix} p_{1,id} & T^{max}_{1,id} & T^{min}_{1,id} \\ \cdots & \cdots & \cdots \\ p_{nt,id} & T^{max}_{nt,id} & T^{min}_{nt,id} \end{bmatrix}$$

$$Y_{indv,id} = \begin{bmatrix} q_{1,id} \\ \cdots \\ q_{nt,id} \end{bmatrix}$$

Donde id es el número de la subcuenca.

En este último caso, las matrices poseen las dimensiones (229,1,3) y (229,1,1), respectivamente. La división entre los conjuntos de entrenamiento y test se ha hecho de la misma manera que en el apartado anterior.

1.4.3. Entrenamiento secuencial

Finalmente, el modelo LSTM2 se ha entrenado de manera secuencial utilizando únicamente los valores simulados de los caudales en la matriz $Y_{indv,id}$. En este enfoque, la red utiliza los valores de los caudales en tiempos anteriores $q_{t-1}, q_{t-2}, q_{t-n}$ para predecir el valor actual q_t. La variable n también llamada "look back"se puede elegir arbitrariamente y se puede ajustar haciendo una busca con el método de cross-validation.

1.4.4. Análisis de componentes principales

El espacio de variables predictoras del conjunto de entrenamiento que contiene la información de las 76 subcuencas posee una dimensión (160,228). Sin embargo, existen correlaciones entre las variables que provocan que muchas de estas aporten información poco relevante o redundante [7]. Estas correlaciones pueden distorsionar el proceso de aprendizaje de los modelos [1] y es por eso que se ha optado por reducir el número de variables mediante el análisis de componentes principales[8].

Este método realiza una transformación del espacio predictor a un espacio vectorial cuya base son los auto-vectores de la matriz de covariancia. En este espacio, los valores en la diagonal de dicha matriz son las varianzas en la dirección de cada uno de los vectores de la base o componentes principales. La reducción de dimensiones se realiza escogiendo las direcciones que captan la mayor variación de los datos originales, es decir las PCA con las mayores varianzas.

Luego de realizar este análisis se ha encontrado que se puede explicar el 96 % de la varianza del conjunto de datos original considerando solo las primeras 9 componentes. Por lo tanto la reducción del espacio predictor es considerable, ya que ha pasado de ser (160,228) a (160,9).

1.4.5. Coeficiente de Nash-Sutcliffe

Calidad del ajuste	NSE
Excelente	$0{,}75 < NSE \leq 1{,}00$
Bueno	$0{,}65 < NSE \leq 0{,}75$
Aceptable	$0{,}5 < NSE \leq 0{,}65$
No aceptable	$NSE \leq 0{,}5$

Cuadro 1.3: Calidad de los ajustes en función del coeficiente NSE [9].

Además de utilizar la función de loss para validar los modelos durante el entrenamiento de los mismos, también se ha utilizado el coeficiente Nash-

Sutcliffe (NSE) [9] en el conjunto de test. Este coeficiente es uno de los más utilizados en hidrología y se define de la siguiente manera:

$$NSE = 1 - \frac{\sum_i^n (y_i - \hat{y}_i)^2}{\sum_i^n (\bar{y} - \hat{y}_i)^2} \tag{1.5}$$

Los valores de NSE varían entre $-\infty$ y 1, siendo este último valor el correspondiente a un ajuste perfecto. En la tabla 1.3 se indica la relación entre los valores de NSE y la calidad de los ajustes.

Resultados

En este capítulo se muestran los resultados obtenidos al entrenar los modelos previamente descritos, el objetivo es dado un evento de precipitación, predecir los caudales de descarga naturales en cada una de las sub-cuencas que conforman Chambo. La performance predictiva de los modelos es evaluada en el conjunto de test, en la tabla 2.1 se muestran los hyper-parámetros utilizados para cada modelo obtenidos tras realizar la exploración descrita en 1.3.4.

2.1. Validación de los modelos

Como se ha mencionado en la sección 1.4.5, se ha utilizado el coeficiente de Nash-Sutcliffe para validar los modelos y hacer un análisis un poco más profundo sobre cómo es su performance en los diferentes puntos de la cuenca. En la figura 2.1 se muestran los valores obtenidos de NSE para todas las subcuencas de Chambo considerando el modelo denso y el modelo LSTM1 entrenado de manera global en el espacio de las componentes principales. Las líneas horizontales muestran los umbrales correspondientes a ajustes excelentes y aceptables.

El modelo LSTM1 posee una mejor performance general a lo largo de toda la cuenca, en este caso se puede considerar que el ajuste para el 68 % de las subcuencas es excelente, mientras que el 11 % de los ajustes son buenos y el 16 % aceptable. Por otro lado, si bien la performance general del modelo denso es buena (el 63 % de los ajustes es excelente), el porcentaje de fallos asciende al 13 %. Se puede observar que en las subcuencas con ids 5, 26, 31 y 66 el coeficiente NSE adquiere valores negativos, mientras que el modelo LSTM1 realiza muy buenos ajustes. La mejor performance del modelo LSTM1 se debe a que las celdas de memoria permiten una interpretación a lo largo del eje del tiempo de los diferentes procesos que ocurren en cada una de las cuencas, lo que permite captar más información sobre la relación entre eventos de precipitación y descarga.

Modelo	optimizador	Neuronas	Activación	alpha	reg
Denso	rmsprop	200	relu	0.0002	l2, 0.0001
LSTM1 PCA	rmsprop	200	linear	0.0002	0.0001
LSTM1 loc	adam	200	relu	0.0001	l1, 0.0001
LSTM2	adam	200	relu	0.001	l2, 0.0000001

Cuadro 2.1: Hyper parámetros utilizados en los diferentes modelos.

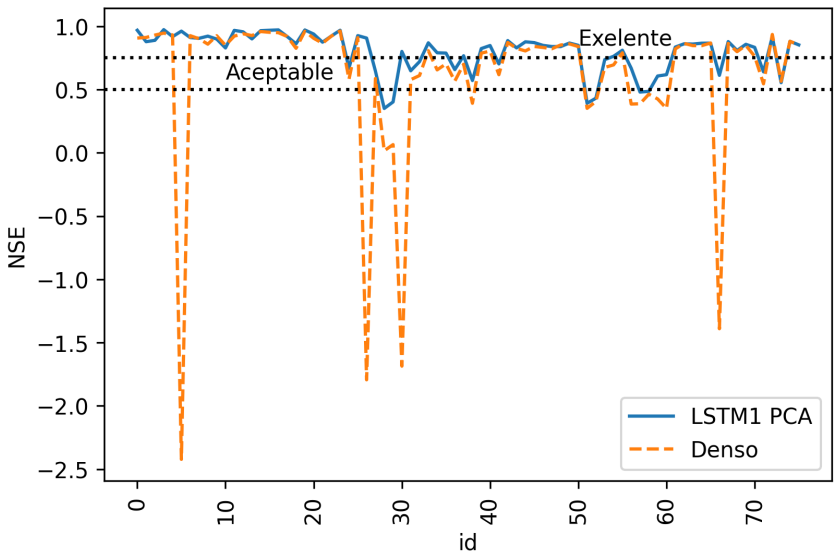


Figura 2.1: Valores obtenidos para el coeficiente NSE a lo largo de toda la cuenca con los modelos denso y LSTM1 entrenado con PCAs.

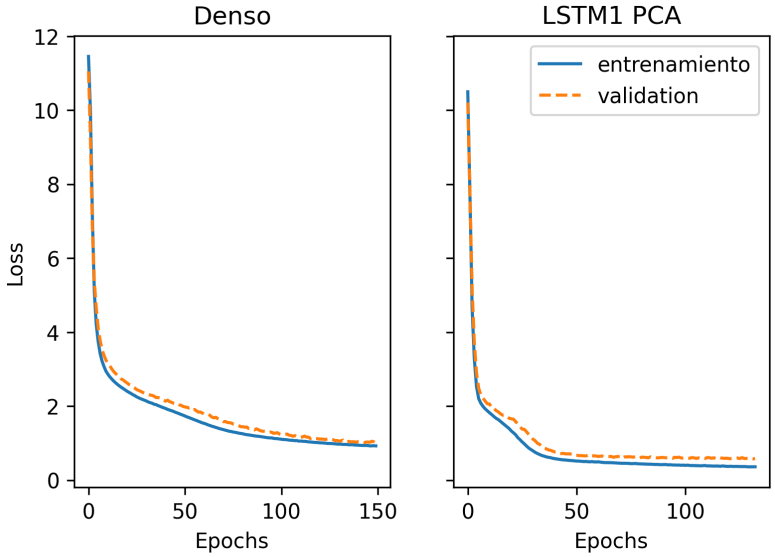


Figura 2.2: Evolución de la función de loss en los conjuntos de entrenamiento y validación para los modelos denso y LSTM1 con PCAs.

En la figura 2.2 se muestran los valores de la función de loss obtenidos en cada época o iteración durante el entrenamiento de los modelos denso y

LSTM1 PCA. Durante el entrenamiento se ha dividido el conjunto de datos de manera que el 85 % de los mismos han sido utilizados para entrenar el modelo y el 15 % restante para validarlo y así evitar el sobre ajuste monitorizando la evolución del loss. Además se ha implementado el callback *.earlystopping"* que interrumpe la ejecución cuando el loss en el conjunto de validación comienza a aumentar.

Si se comparan ambos modelos, podemos observar que en el modelo LSTM1 PCA aprende más rápido ya que en solo 50 épocas el valor de loss alcanza su valor de equilibrio. En el caso del modelo denso, esto ocurre luego de 150 épocas. Por otro lado, y en concordancia con los resultados arrojados por el coeficiente NSE, el valor final del loss en el modelo LSTM1 PCA (entrenamiento: 0.3587 , validación: 0.5787) es menor que en el modelo denso (entrenamiento: 0.9245 validación: 1.0060) los que indica un mejor ajuste global.

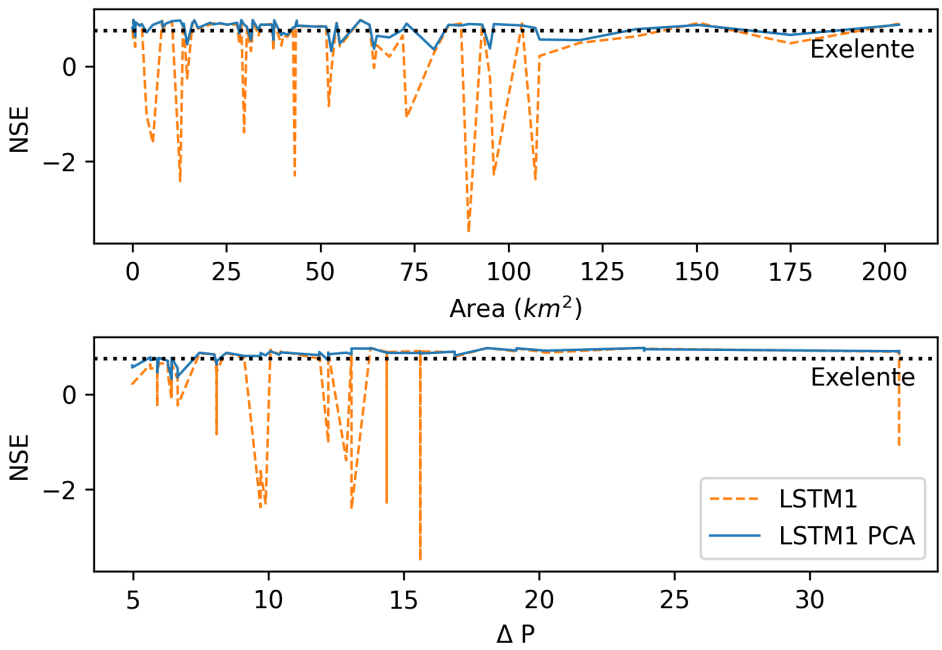


Figura 2.3: Evolución del coeficiente NSE en función del área (panel superior) y la variación de precipitación (panel inferior) para los modelos LSTM1 con y sin PCAs.

En la figura 2.3, se muestra la comparación entre los valores de NSE obtenidos para el modelo LSTM1 con PCAs (curva continua) y sin PCAs (curva a rayas) en función del área de la cuenca (panel superior) y de la variación de precipitación (panel inferior). Los resultados obtenidos cuando consideramos las 288 características de entrada muestran una gran variación

de los valores de NSE y peores predicciones. Una razón por la que las componentes principales funcionan mejor puede deberse a que éstas agregan la información proveniente de todo el dominio del espacio predictor [7]. En línea con este argumento, en estas figuras también se puede observar que el área de las sub-cuencas y la variación de precipitación son factores relevantes que condicionan la calidad de los ajustes. El modelo que no incluye componentes principales tiende a fallar más en las cuencas pequeñas ($Area < 120\ km^2$) y áridas ($\Delta P < 20\ mm/d$). Esto puede deberse a que en estos casos los valores de la precipitación y caudales son más pequeños y el loss es en general menor que el loss para una sub-cuenca con una descarga grande. Esto último provoca que el modelo deje de aprender demasiado pronto y se genere un un sobre peso para las sub-cuencas más grandes y húmedas mientras que la performance en las más pequeñas y áridas disminuye [3].

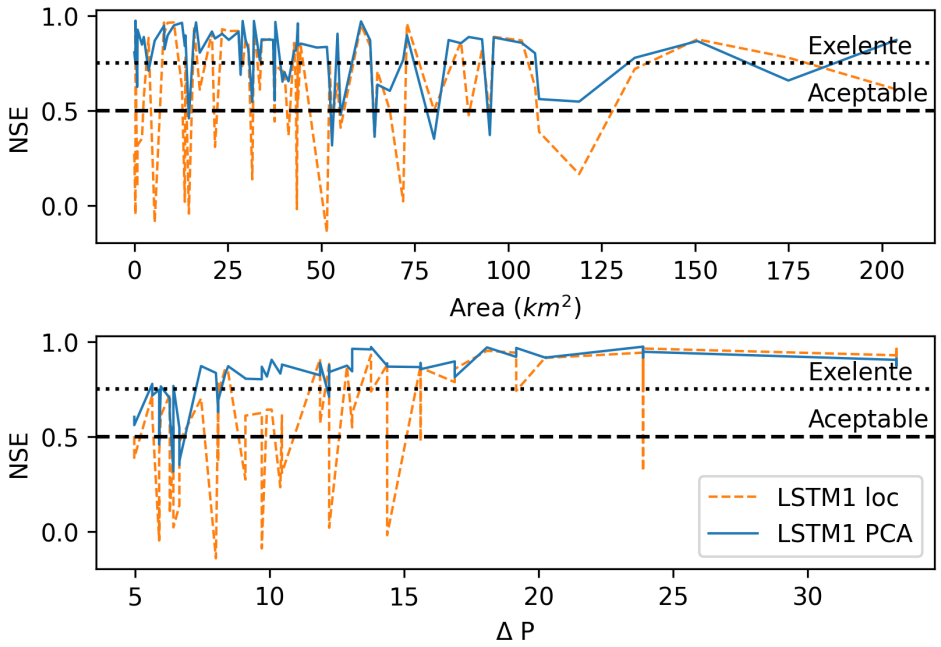


Figura 2.4: Evolución del coeficiente NSE en función del área (panel superior) y la variación de precipitación (panel inferior) para los modelos LSTM1 con PCAs y LSTM1 entrenado localmente.

Este efecto es también visible cuando entrenamos el modelo LSTM1 de manera local, como puede verse en la figura 2.4 en la mayoría de los casos la performance del modelo global es mejor que la del modelo local. Este resultado se encuentra alineado con estudios anteriores que demuestran que los modelos LSTM entrenados en un gran número de sub-cuencas son capaces de vincular Las características de las mismas para aprender un modelo global

que a su vez es capaz de reflejar explícitamente las similitudes y diferencias de las cuencas a nivel local [3]. Más aun, estos estudios han demostrado que las redes LSTM son incluso capaces de modelar la presencia de nieve y almacenar esta información en las celdas específicas de memoria sin siquiera haber sido entrenadas directamente en ningún tipo de observación relacionada con nieve más allá de la precipitación total y la temperatura [10].

2.2. Caudales naturales

En las figuras 2.5 se muestran los resultados obtenidos para los caudales en la sub-cuenca con id 20 y los grados de ajustes correspondientes obtenidos con los diferentes modelos. Las curvas continuas son los valores simulados con el modelo LEM y las curvas a rayas son los valores predichos con los modelos denso, LSTM1 PCA y LSTM1 loc en el conjunto de test. En este caso la performance de los modelos LSTM1 PCA y denso es excelente, con valores de NSE iguales a 0.97 y 0.96, respectivamente, mientras que el modelo entrenado localmente es un poco más baja. Como se ha explicado en sesiones anteriores, los modelos entrenados de manera global con componentes principales, aprenden en un espacio que contiene información agregada proveniente de todo el espacio predictor. El ajuste del modelo LSTM1 PCA es aún mejor porque la presencia de neuronas recurrentes permiten contemplar la correlación de los datos en la dimensión temporal.

En la figura 2.6 se muestra a modo de ejemplo un caso en el que el modelo LSTM1 loc falla al predecir los valores en el conjunto de test, mientras que los modelos Denso y LSTM1 PCA arrojan un ajuste aceptable ($NSE > 0{,}6$). En este caso, el modelo LSTM1 loc ha sido capaz de reflejar cierta tendencia de los datos pero aún así la calidad del ajuste es baja ($NSE < 0{,}5$). El modelo LSTM1 PCA, que posee ambas ventajas, la de considerar componentes principales y redes recurrentes, es capaz de predecir los valores de los caudales en el conjunto de test con una calidad aceptable ($NSE = 0{,}7$).

En la figura 2.7 se muestran los resultados obtenidos para el modelo LSTM2 entrenado secuencialmente. en el panel superior se muestran lo valores del coeficiente NSE a lo largo de toda la cuenca y en el panel inferior los resultados obtenidos para el mejor ajuste. Se puede observar que la performance del modelo es en general bastante pobre, sólo unos pocos puntos sobrepasan en umbral de calidad aceptable. Los mayores problemas que presenta esta aproximación es que por un lado los errores cometidos en cada una de las predicciones es propagado y acumulado a lo largo del tiempo y por otro lado se pierde completamente la información otorgada por las series hidro-climáticas de entrada por lo cual el modelo no es capaz de aprender ningún patrón relacionado con la respuesta que las diferentes sub-cuencas

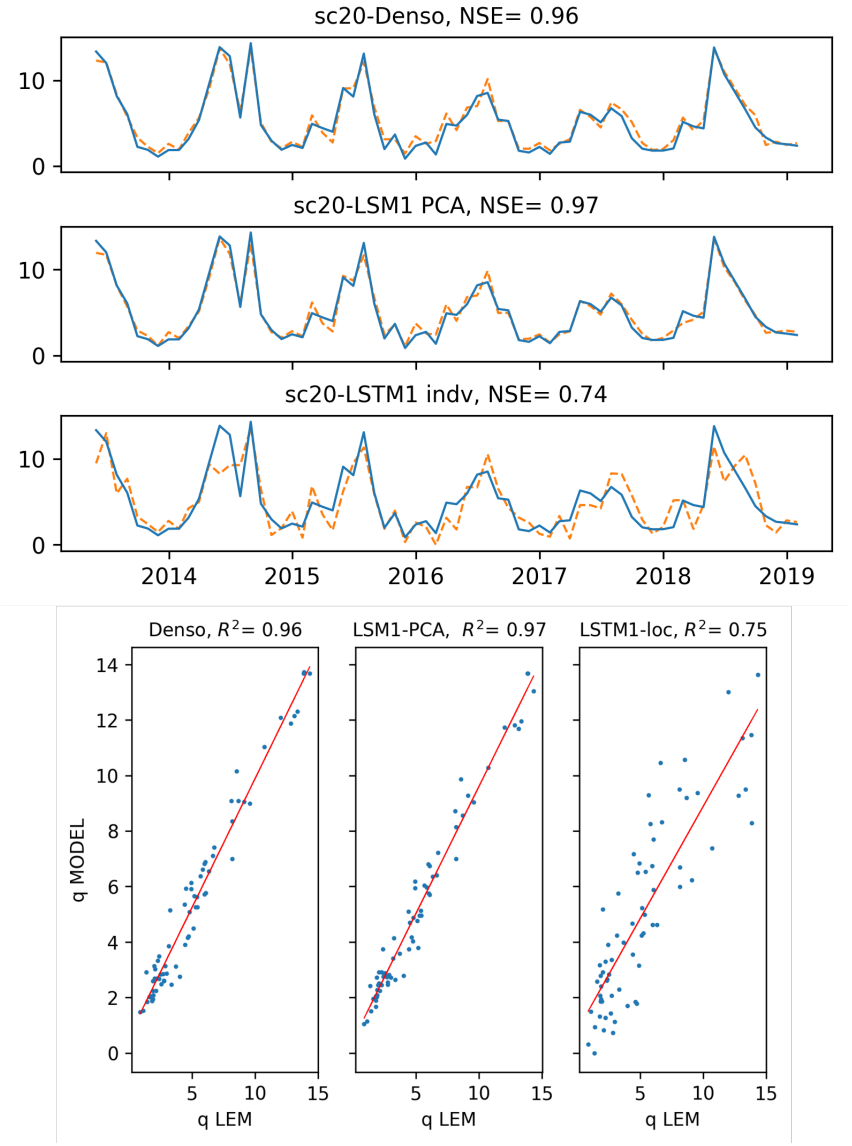


Figura 2.5: Predicciones del caudal de descarga obtenidos con los modelos Denso, LSTM1 con PCAs y LSTM1 local en la subcuenca con id 20. En los paneles inferiores se muestran los grados de ajuste correspondientes.

tienen frente a las series de precipitación.

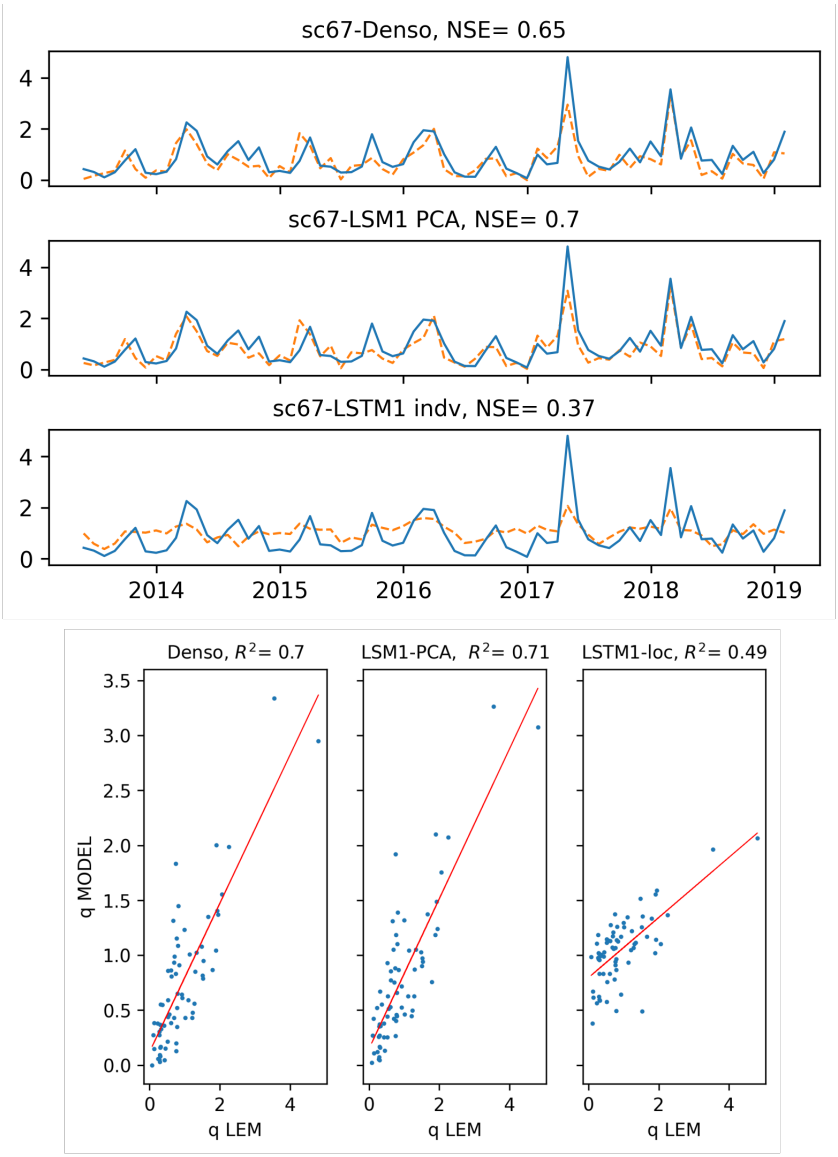


Figura 2.6: Predicciones del caudal de descarga obtenidos con los modelos Denso, LSTM1 con PCAs y LSTM1 local en la subcuenca con id 67. En los paneles inferiores se muestran los grados de ajuste correspondientes.

2.3. Resultados con Modzim

Con el fin de determinar si los resultados obtenidos con redes neuronales pueden ser utilizados para simular operaciones en la cuenca Chambo, se ha determinado el balance hidrológico utilizando el software Modsim con los caudales simulados por el modelo hidrológico LEM y las predicciones

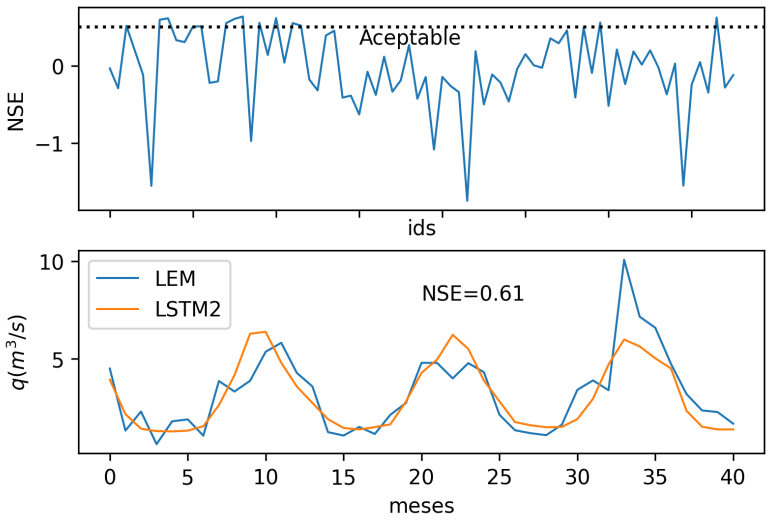


Figura 2.7: Resultados del Modelo LSTM2, en el panel superior se muestran los valores de NSE a lo alrgo de toda la cuenca y En el panel inferior, los valores predichos en la sub-cuenca con id 15.

de los modelos de redes neuronales en el conjunto de test. Este software optimiza la distribución del agua en las redes del cauce teniendo en cuenta la complejidad hidrológica de la cuenca e incluyendo los derechos del agua.

En el panel superior de la figura 2.8 se muestra el residuo (R), es decir la diferencia entre los valores de los caudales predichos respecto a los caudales de referencia (en este caso simulados con el modelo LEM), para diferentes sub-cuencas distribuidas a lo largo de toda la cuenca hidrográfica Chambo. En los paneles inferiores se muestran los valores de los caudales arrojados por Modsim para la sub-cuenca que se encuentra a la salida de la cuenca hidrográfica (aguas abajo) y para una de las sub-cuencas que se encuentra a mayor altitud (aguas arriba). Las curvas continuas corresponden al resultado obtenido con los caudales simulados con el modelo LEM, y la curva a rayas es el resultado obtenido cuando consideramos los valores predichos por el modelo LSTM1 PCA. Se puede apreciar que los resultados arrojados por Modsim son los mismos para ambos casos, es decir, las pequeñas variaciones mostradas en el panel superior no influyen en el resultado final.

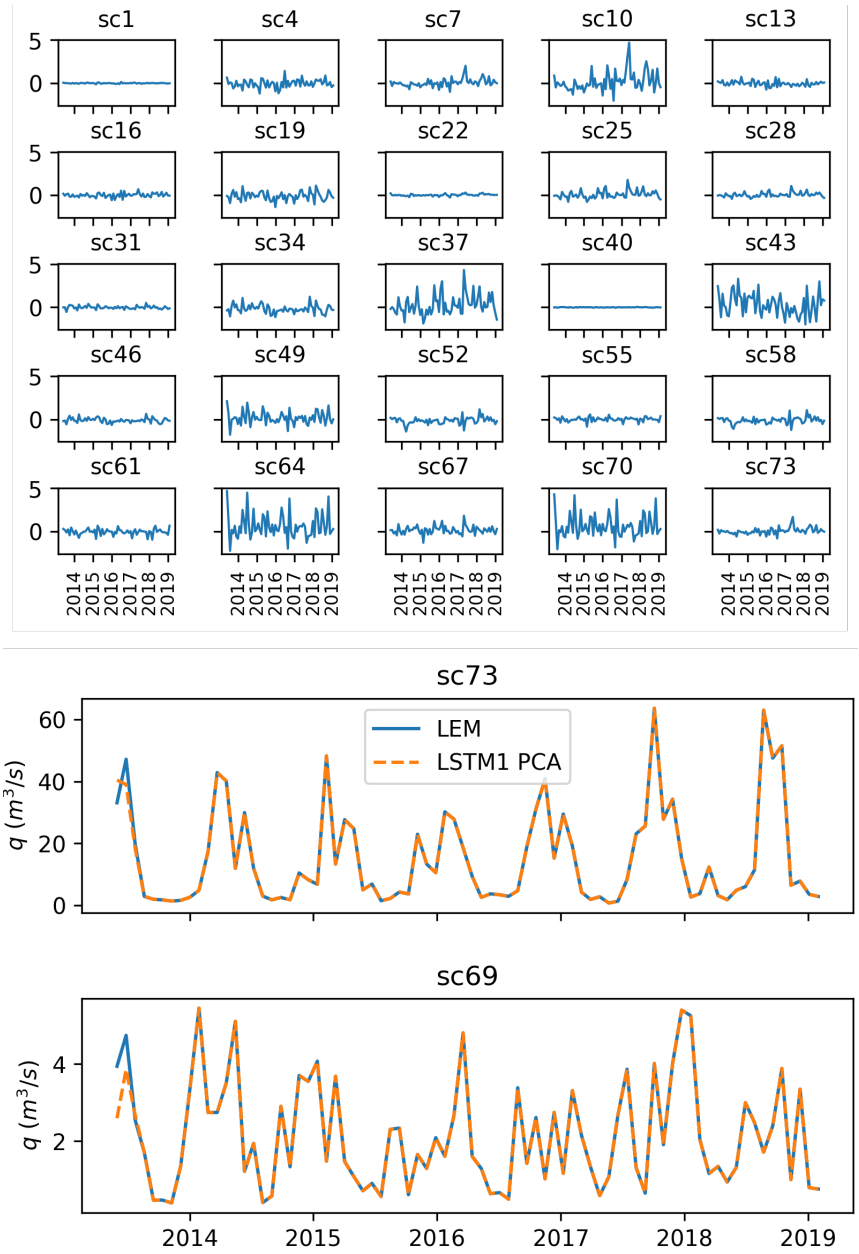


Figura 2.8: Paneles superiores: residuos obtenidos a lo largo de diferentes sub-cuencas. Paneles inferiores: resultados finales obtenidos utilizando el software de gestión de recursos Modsim, teniendo en cuenta los usos del agua

Discusiones y Conclusiones

Los resultados descritos en la sección anterior demuestran que el modelo que realiza los mejores ajustes es el LSTM1 cuando es entrenado en el espacio de las componentes principales. Aproximadamente un 70 % de los ajustes obtenidos con el mismo son excelentes mientras que esta cifra desciende al 60 % para los modelos denso y LSTM1 entrenado localmente. Si bien los últimos modelos poseen una buena performance en la mayoría de las sub-cuencas, presentan mucha variación en el coeficiente de NSE, alcanzando incluso valores negativos en algunos puntos para los cuales el modelo LSTM1 PCA arroja ajustes que son desde aceptables hasta excelentes. Por otro lado el experimento realizado con el modelo LSTM2, que considera un entrenamiento secuencial utilizando solamente los valores de la serie de caudales de descarga de las sub-cuencas, ha demostrado tener una mala performance en la mayoría de los puntos.

Los resultados antes mencionados nos llevan a concluir que la inclusión de las celdas de memoria LSTM y la presencia de componentes principales mejoran notablemente la performance de los modelos en cuencas hidrográficas. Por un lado las celdas LSTM permiten almacenar información de eje temporal sobre los diferentes procesos que ocurren en las cuencas, y así captar mas información sobre la relación entre eventos de precipitación y descarga. Por otro lado, las componentes principales reducen considerablemente la dimensión del espacio predictor y combinan la información proveniente de todo el dominio del mismo, lo que soluciona problemas de sobre ajustes y el problema des desvanecimiento del loss presente en las cuencas más pequeñas y áridas. Por último, se ha demostrado que los resultados obtenidos con redes neuronales pueden ser utilizados para simular operaciones en la cuenca Chambo, ya que se ha comprobado que el resultado del balance hidrológico obtenido con Modsim son los mismo que los obtenidos con los valores simulados por el modelo LEM.

Estos resultados concuerdan con estudios anteriores en los cuales se concluye que las redes neuronales generalmente requieren una gran cantidad de datos de entrenamiento y que los ajustes que se obtienen al entrenar modelos de aprendizaje profundo en una sola sub-cuenca no suelen ser fiables [3]. Esto supone una gran diferencia con el modelado y calibrado hidrológico tradicional que normalmente obtiene los mejores resultados cuando los modelos se calibran de forma independiente para cada sub-cuenca. Esta propiedad de los modelos clásicos presenta problemas, ya que se ha observado que los parámetros obtenidos por extrapolaciones basadas en valores calibrados en cuencas de referencia pueden dar lugar a espacios de parámetros poco realistas[11]. Los modelos LSTM en cambio, demuestran tener la capacidad

de aprender simultáneamente relaciones de series temporales y espaciales en el mismo marco predictivo, lo que evita muchos problemas que actualmente se encuentran asociados con la estimación y transferencia de parámetros de modelos hidrológicos tradicionales [3], [12].

Un conclusión importante es entonces que las celdas LSTM son capaces de generar un modelo único a partir de grandes conjuntos de datos de que sea capaz de reflejar los comportamientos hidrológicos regionales específicos de cada sub-cuenca ya que estos modelos son capaces de vincular las características locales de las sub-cuencas y aprender un modelo general a partir de los datos combinados de todas ellas. Más aún, existe un estudio donde se han utilizado datos provenientes de varios cientos de cuencas continentales en los Estados Unidos a lo largo de 30 años, que demuestra que los modelos de aprendizaje profundo arrojan mejores predicciones para los caudales de descarga en cuencas no aforadas que los obtenidos por modelos tradicionales calibrados con registros de datos en cuencas aforadas[12][3]. Es por esto que concluimos que la principal virtud de las redes neuronales no es simplemente el hecho de que ajusten bien sino su capacidad de aprendizaje y flexibilidad para ser utilizadas en una variedad de lugares y condiciones diferentes.

Bibliografía

[1] Gutierrez, J. M., R. Cano, A. S. Cofino y C. M. Sordo: *Redes Probabilísticas y Neuronales en la Ciencias Atmosféricas*. 2004.

[2] Olah, C.: *Understanding LSTM networks*. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[3] Kratzert, F., D. Klotz, G. Shalev, G. Klambauer, S. Hochreiter y G. Nearing: *Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets*. Hydrology and Earth System Sciences, 23(12):5089–5110, 2019. `https://hess.copernicus.org/articles/23/5089/2019/`.

[4] Ruder, Sebastian: *An overview of gradient descent optimization algorithms*, 2016. `https://arxiv.org/abs/1609.04747`.

[5] Kingma, Diederik P. y Jimmy Ba: *Adam: A Method for Stochastic Optimization*, 2014.

[6] al., Pedregosa et: *Scikit-learn: Machine Learning in Python*. 2012. `https://arxiv.org/abs/1201.0490`.

[7] Javier Diez-Sierra, Manuel del Jesus: *Long-term rainfall prediction using atmospheric synoptic patterns in semiarid climates with statistical and machine learning methods*. 2020.

[8] Abdi, Hervé y Lynne J. Williams: *Principal component analysis*. WIREs Computational Statistics, 2(4):433–459, 2010. `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.101`.

[9] al., D. N. Moriasi et: *Model Evaluation Guidelines for Systematic Quantification of Accuracy in Watershed Simulations*. 2007.

[10] Kratzert, Frederik, Daniel Klotz, Mathew Herrnegger y Sepp Hochreiter: *A glimpse into the Unobserved: Runoff simulation for ungauged catchments with LSTMs*, Diciembre 2018.

[11] Mizukami, Naoki, Martyn Clark, A. Newman, A. Wood, Ethan Gutmann, Bart Nijssen, Oldrich Rakovec y Luis Samaniego: *Toward seamless large domain parameter estimation for hydrologic models*. Water Resources Research, 53, Agosto 2017.

[12] Nearing, G. S., F. Kratzert, A. K. Sampson, C. S. Pelissier, D. Klotz, J. M. Frame, C. Prieto y H. V. Gupta: *What Role Does Hydrological Science Play in the Age of Machine Learning?* Water Resources Research, 57(3):e2020WR028091, 2021.