Machine Learning A.A. 2019/2020: Relazione Progetto NILM

Marco Balletti, Francesco Marino

22 luglio 2020

1 Introduzione

La presente relazione tratta dei modelli basati su reti neurali proposti per la risoluzione del *Non-Intrusive Load Monitoring (NILM)*. Lo scopo è quello di realizzare un'architettura di reti neurali in grado di rilevare, a partire da un valore (espresso in Watt) di consumo energetico totale di un'abitazione, il consumo relativo delle diverse apparecchiature, in particolare, in questo caso, di un frigorifero e di una lavastoviglie. Per la realizzazione del progetto è stato utilizzato *Python*, come linguaggio di programmazione, affiancato alle librerie *Numpy* (per la gestione efficiente di *array* di dati), *Pandas* (per il reperimento dei dati), *Tensorflow* e *Keras* (per la realizzazione e l'addestramento dell'architettura). La piattaforma di sviluppo utilizzata è *Google Colab* che mette a disposizione degli utenti la possibilità di addestrare delle reti neurali utilizzando anche delle *GPU* remote.

2 I dati

Per la realizzazione del progetto sono stati forniti tre *data set*: il primo contiene i consumi dell'intera abitazione presa in esame registrati, con una granularità pari ad un secondo, il secondo ed il terzo contengono, invece, i consumi relativi al frigorifero e alla lavastoviglie registrati con la stessa granularità e nello stesso intervallo temporale del primo *data set*. Visto l'ampio quantitativo di dati e la loro dimensione, si è ritenuto conveniente eseguirne il caricamento su *GitHub* in formato compresso (`.zip`), tale scelta è

motivata anche dalla semplicità con cui, utilizzando *Python*, *Keras* e *Pandas*, è possibile trasformare un *data set* remoto compresso in un *dataframe*.

2.1 *Training*, *Validation* e *Test set*

Per garantire la correttezza dell'addestramento del modello e poterne valutare le prestazioni, è stato realizzato lo *split* del *dataframe*, ricavato precedentemente, in *training set*, *validation set* e *test set*. Il primo è stato utilizzato per eseguire l'addestramento dei modelli, il secondo per monitorarne l'*overfitting* e l'ultimo per valutare le metriche richieste (l'*energy precision*, l'*energy recall* e l'*energy F1*) sul modello e poter confrontare le differenti possibili soluzioni tra loro. Durante la creazione di questi *set* e la conversione da *dataframe* ad *array Numpy*, si pone particolare attenzione al consumo della memoria RAM, vengono, infatti, eliminati tutti i riferimenti alle strutture dati non più necessarie dopo la conversione in modo da permettere al *garbage collector* di eliminarle e recuperare memoria.

Poiché i dati in questione sono di tipo serie temporali, si è deciso di realizzare il *validation set* ed il *test set* con misurazioni che fossero temporalmente successive a quelle presenti all'interno del *training set*, tale relazione di ordine si mantiene anche tra *validation* e *test set*.

2.2 Normalizzazione

I dati così suddivisi vengono, quindi, sottoposti a normalizzazione sottraendo loro il valore medio e dividendoli per la deviazione standard, tale procedimento si è rivelato essere una strategia particolarmente efficace per la riduzione dei tempi di addestramento dei modelli. I valori necessari per effettuare questa trasformazione sui dati sono stati calcolati utilizzando il solo *training set*: si è proceduto, in particolare, calcolando la media e la deviazione standard dei consumi dell'abitazione, del frigorifero e della lavastoviglie su questo *set* di dati, il *training set* ed il *validation set* normalizzati sono stati ottenuti, quindi, andando a sottrarre la media relativa al tipo di dato (totale, frigorifero o lavastoviglie) e dividendo per la corrispondente deviazione standard precedentemente calcolate. Per quanto concerne il *test set*, l'operazione di normalizzazione è stata svolta solo sui consumi totali (sempre utilizzando i valori ricavati dal *training set*), non è stato necessario eseguirla sui corrispondenti valori di frigorifero e lavastoviglie poiché i dati ottenuti dalla predizione sul *set* in questione vengono denormalizzati (moltiplicando per la deviazione standard e sommando la media del *training set*) prima di eseguire la valutazione delle prestazioni.

2.3 I generatori

L'addestramento e la valutazione delle prestazioni di modelli basati su dati di tipo serie temporali richiede l'utilizzo di finestre, queste strutture di dati risultano essere necessarie sopratutto se ci si trova nei casi di predittori *sequence-to-sequence* o *sequence-to-point*. Realizzare una finestra temporale nel modo classico (`for` *loop*) si rivela essere particolarmente sconveniente sia in termini di prestazioni che in termini di consumo di memoria RAM, quest'ultimo aspetto, in particolare, risulta essere maggiormente chiaro se si considerano finestre temporali parzialmente sovrapposte per cui una stessa misurazione, ricadendo in più finestre temporali, viene memorizzata molteplici volte.

Per ovviare a questa problematica ed evitare anche di incorrere in *crash* dell'*environment Google Colab*, si è proceduto realizzando dei generatori di dati, particolari strutture in grado di creare dinamicamente (all'accesso dell'elemento) *batch* di dati. L'idea alla base della realizzazione di queste strutture è quella di eseguire un addestramento basato su *batch* di finestre temporali, quando ne viene richiesto uno nuovo, il generatore determina gli indici degli elementi che faranno parte di ogni singola finestra appartenente al *batch* (le dimensioni di questo e della finestra sono parametri della struttura, lo scorrimento delle finestre è pari a un elemento), genera le finestre richieste e restituisce l'insieme di dati. Il risparmio di memoria è dovuto al fatto che questo approccio permette di istanziare le sole finestre necessarie in ogni momento, quelle appartenenti a *batch* precedenti, invece, perdendo ogni riferimento, possono essere deallocate dal *garbage collector* mantenendo l'utilizzo della RAM costante.

3 Il modello finale proposto

Per la predizione dei valori di consumo del frigorifero e della lavastoviglie è stata utilizzata una modifica del modello *sequence-to-point* proposto da Zhang. Il modello applica alla finestra in input cinque layer Convoluzionali (ognuno, a parte l'ultimo, seguito da un layer di *Max Pooling*), esegue il *Dropout* ed il *Flatten* dei risultati per poi sottoporli a due *layer Dense* intervellati da un *Dropout*.

Nel caso del frigorifero è stata utilizzata una finestra temporale di input contenente 2400 misurazioni (40 minuti), nel caso della lavastoviglie, invece, la finestra ha ampiezza di 900 elementi (15 minuti).

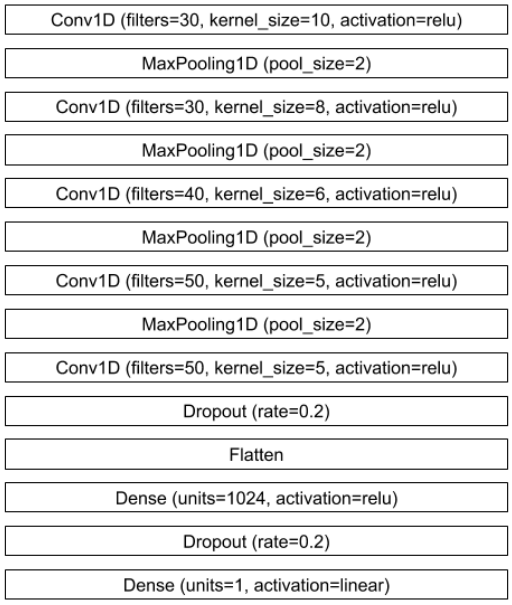

Figura 1: Rappresentazione grafica delle rete neurale utilizzata per le due apparecchiature

3.1 *Cross Validation*

Per determinare le lunghezze delle finestre da utilizzare per ognuna delle apparecchiature, è stata scelta la tecnica della *cross validation*; per confrontare i risultati, gli addestramenti sono stati effettuati con *early stopping* (*patience* = 5) su 15 epoche (quantitativo inferiore rispetto a quello scelto per l'addestramento finale). Per il frigorifero sono state testate finestre di 2000, 2200, 2400 e 2600 elementi (i valori scelti dipendono anche dalla struttura del *data set*), mentre, per la lavastoviglie, finestre di dimensione 600, 700, 800, 900 e 1000 elementi.

3.2 Addestramento

Una volta scelte le lunghezze delle finestre, il modello è stato addestrato in 100 epoche e con *early stopping* (*patience*=5), l'ottimizzatore scelto è *Adam* con funzione di *loss* la *mean squared error*. Come anticipato nei paragrafi precedenti, i dati utilizzati per l'addestramento e la valutazione del modello sono stati prodotti utilizzando dei generatori a partire dal *data set* iniziale, questi, trovandoci nel caso di *sequence-to-point* con predizione la mediana della finestra, sono stati configurati per porre all'inizio e alla fine del *set* di dati dei *dummy values* (elementi pari a 0) in modo tale da creare tante

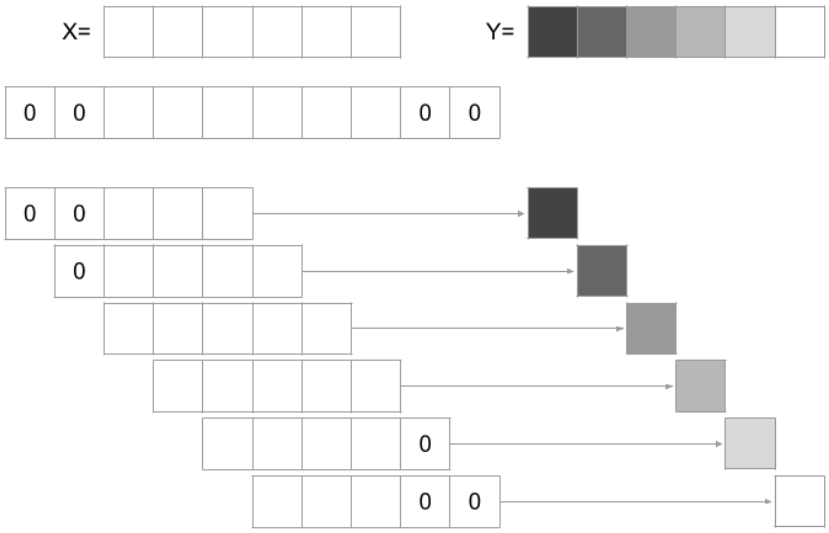

Figura 2: Rappresentazione grafica della generazione delle finestre sfruttando i *dummy values*

finestre quanti sono gli effettivi valori da predire e facendo sì che al punto mediano di ognuna corrispondesse il corretto valore da ottenere.

Per garantire una maggiore velocità di convergenza dell'algoritmo, è stato eseguito un *reshuffling* dell'ordine dei *batch* all'inizio di ogni epoca di addestramento.

Sono state salvate due versioni del modello, una addestrata sui dati del frigorifero ed una addestrata sui dati della lavastoviglie, e valutate utilizzando la metrica di *energy F1*, i risultati ottenuti sono:

energy F1 per il frigorifero valutata sul *test set*: `0.7745927`
energy F1 per la lavastoviglie valutata sul *test set*: `0.9533379`

3.3 Alternative testate

3.3.1 Modelli alternativi

Prima di giungere alla soluzione finale proposta, sono stati eseguiti confronti tra differenti architetture (evitando di effettuare *tuning* dei parametri per ognuna), in particolare, è stato testato l'utilizzo di *Denoising Autoencoder (dAE)*, del modello *sequence-to-point* "puro" proposto da Zhang e del modello proposto dall'organizzazione CSEM (in particolare, la soluzione finale prende spunto dagli ultimi due *paper* riportati).

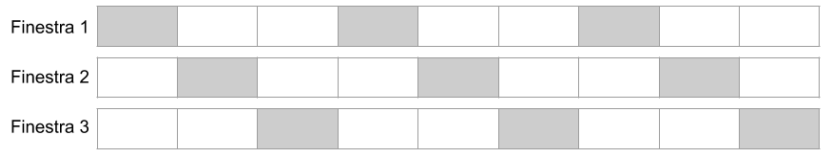

Figura 3: Rappresentazione grafica della generazione di finestre di dati con *downsampling*

3.3.2 *Downsampling*

Il confronto del *data set* fornito per l'esecuzione del progetto con i *set* di dati utilizzati in letteratura ha evidenziato una forte differenza sulla granularità dei dati: il primo si compone di misurazioni prese ogni secondo, i secondi, tipicamente, di misure intervallate di sei secondi. Alla luce di questa differenza, si è provato a realizzare un *downsampling* all'interno delle finestre di dati restituite dai generatori in modo tale che, mantenendo fissa la dimensione, l'intervallo temporale coperto dalla finestra fosse maggiore. Questo approccio consente di evitare di dover rinunciare a priori ad una grande quantità di dati, è stato solamente necessario modificare la logica di costruzione dei batch di finestre affinché ognuna prendesse un elemento ogni sei; si noti che finestre consecutive, in questo modo, però, non avranno dati sovrapposti.

Questo tipo di approccio si è rivelato non essere l'ideale per la rete scelta come soluzione finale e, quindi, si è rinunciato ad applicarlo.