# Disaggregazione dei consumi energetici con Tensorflow e Keras

Università degli Studi di Roma Tor Vergata - Corso di Machine Learning - A.A. 2019/2020 - Prova progettuale

Federico Viglietta
fe.viglietta@gmail.com

Tommaso Villa
tommaso.villa96@gmail.com

## I. Introduzione

Il presente documento illustra le tecniche e le metodologie impiegate nella progettazione e nell'addestramento di un modello di machine learning finalizzato alla disaggregazione dei consumi energetici. La disaggregazione dei consumi energetici, altrimenti nota come Non Intrusive Load Monitoring (NILM), è una tecnica che mira a stimare la potenza assorbita dai singoli elettrodomestici di un'abitazione a partire dal consumo elettrico complessivo. Ad oggi, le reti neurali deep rappresentano lo standard de facto in questo ambito di ricerca [1].

I primi a sperimentare l'impiego di reti deep nella disaggregazione dei consumi energetici sono stati Kelly e Knottenbelt [2]. I due ricercatori hanno affrontato il problema seguendo la metodologia nota in letteratura come *sequence-to-sequence learning*. Essa prevede di addestrare una rete neurale che mappa una sequenza di input - i consumi elettrici complessivi - in una sequenza di output - i consumi elettrici di un singolo apparecchio. Poiché sarebbe impraticabile processare l'intera sequenza di input in un colpo solo, è stato proposto un approccio *sliding window*: una finestra di ampiezza $W$ viene fatta scorrere sui dati con passo unitario e la rete viene addestrata per imparare a mappare $W$ consumi aggregati in $W$ consumi parziali. A causa della sovrapposizione delle finestre, la rete fornisce per ogni istante di tempo più di una stima del valore target. I diversi output disponibili per un dato input possono essere combinati tra loro semplicemente calcolando il valore medio o il valore mediano (come in [3]). Nel 2017 è stata proposta da Zhang et al. una variante del sequence-to-sequence detta *sequence-to-point learning* [4] nella quale, data una finestra di input $Y_{t:t+W-1}$, la rete neurale viene addestrata per restituire il valore $x_{i,\tau}$, che rappresenta la potenza assorbita dall'elettrodomestico $i$ all'istante $\tau = t + \lceil \frac{W}{2} \rceil$. In altre parole, la rete neurale stima il consumo dell'apparecchio $i$ considerando i consumi complessivi fatti registrare nei $\frac{W}{2}$ istanti precedenti e nei $\frac{W}{2}$ istanti successivi. Il modello *seq2point* offre in generale prestazioni migliori rispetto al *seq2seq*. Inoltre, è più semplice da implementare, in quanto produce per ogni valore di input un singolo valore di output.

Il progetto qui descritto si è basato sulla rete deep proposta da Zhang et al. adattandola alle esigenze specifiche del caso di studio in esame. La rete è stata implementata in Tensorflow con l'ausilio dell'interfaccia di alto livello Keras ed è stata addestrata su GPU ricorrendo al piano d'uso gratuito offerto dalla piattaforma Google Colaboratory. Il codice realizzato è disponibile su Github[1].

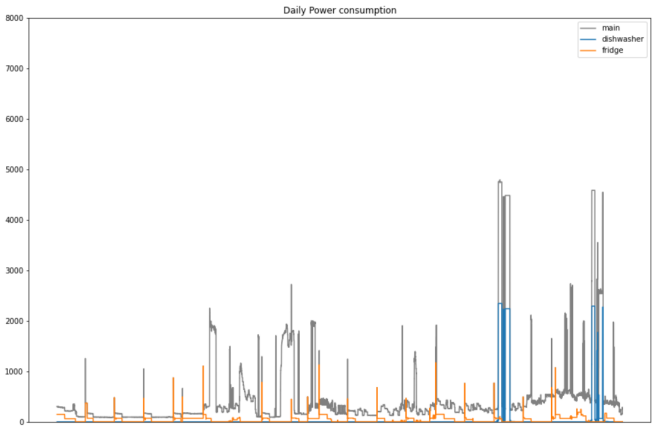


Figura 1. Andamento giornaliero della potenza

La relazione è organizzata nel modo seguente: la sezione II descrive il dataset utilizzato; la sezione III illustra il preprocessamento dei dati di training; nella sezione IV vengono presentate le architetture di rete utilizzate; l'addestramento dei modelli è descritto nella sezione V; chiude il documento una breve discussione sulle possibili integrazioni alla soluzione proposta (sezione VI).

## II. Data Understanding

Il dataset contiene misurazioni al secondo del consumo elettrico complessivo e di due elettrodomestici: frigorifero e lavastoviglie. Nella fattispecie, il training set contiene la potenza assorbita, in Watt, nel periodo compreso tra il 2019-01-01 00:00:00 e il 2019-03-14 23:59:59, mentre il test set - non disponibile durante l'addestramento - contiene i dati relativi al periodo compreso tra il 2019-03-15 00:00:00 e il 2019-03-31 23:59:59. Il training set è costituito da tre file CSV: *main_train.csv*, *fridge_train.csv* e *dishwasher_train.csv*. L'importazione e l'analisi esplorativa dei dati sono state realizzate mediante le librerie Python *Pandas* e *matplotlib*. La

[1] https://github.com/redefik/ConvNILM

Tabella I
STATISTICHE SULLA POTENZA ASSORBITA.

| | **Lavastoviglie** | **Frigorifero** | **Aggregata** |
|---|---|---|---|
| **Numero campioni** | 6,082,508 | 6,082,508 | 6,082,508 |
| **Media (W)** | 25.8711 | 37.23711 | 370.9156 |
| **Dev.Std (W)** | 238.5802 | 46.9887 | 549.1881 |
| **Min (W)** | 0.0 | 2.2 | 73.481 |
| **Max (W)** | 2,570.6 | 1233.1 | 6048.7 |

tabella I mostra le statistiche calcolate sui dati di training. Si noti che la potenza minima assorbita dalla lavastoviglie è pari a $0$ Watt mentre la potenza assorbita dal frigorifero è almeno pari a $2.2$ Watt. Ciò mette in luce la diversa natura dei due apparecchi: il frigorifero è sempre funzionante, mentre la lavastoviglie è un dispositivo ON-OFF. Il training set è più piccolo rispetto a quelli utilizzati nei principali lavori di ricerca (e.g. lo UK-DALE utilizzato in [4]). Inoltre, le misurazioni sono campionate con una frequenza diversa da quella che caratterizza i dataset impiegati nel lavoro di Zhang et al. Dall'analisi esplorativa non sono emersi né valori mancanti né record duplicati, perciò non sono state necessarie operazioni di data cleaning. Il grafico di figura 1 mostra un esempio di andamento giornaliero della potenza. Dall'immagine si può osservare che lavastoviglie e frigorifero non sono gli unici apparecchi presenti nel fabbricato in esame.

## III. PRE-PROCESSAMENTO DEI DATI

### A. Data splitting

Il dataset è stato suddiviso in due parti: le prime settimane sono state impiegate per l'addestramento del modello, le ultime 2 sono state invece utilizzate come *held-out validation set* per configurare in maniera opportuna gli iperparametri (numero di layer, tipo di loss ecc.). Una volta scelti gli iperparametri, l'intero dataset è stato utilizzato per addestrare il modello finale.

### B. Zero-padding

Per costruzione, l'approccio sequence-to-point ignora i primi $\lceil \frac{W}{2} \rceil$ e gli ultimi $\lceil \frac{W}{2} \rceil$ valori della sequenza di input originaria. Per questo è stato necessario aggiungere $\lceil \frac{W}{2} \rceil$ zeri in testa e in coda al training set e al validation set.

### C. Rescaling

Nel lavoro di Zhang et al. sia i valori di input sia i valori di output sono stati standardizzati sottraendo la media e dividendo per la deviazione standard. In questo lavoro i dati usati per stimare il consumo elettrico del frigorifero sono stati ugualmente standardizzati, mentre i dati impiegati per inferire il consumo della lavastoviglie sono stati normalizzati sottraendo il valore minimo e dividendo per la differenza tra il valore massimo e il valore minimo. Questa scelta è stata fatta alla luce dei risultati ottenuti in cross-validation. Le statistiche utilizzate nel rescaling sono state calcolate sui dati di training prima di effettuare lo zero-padding.

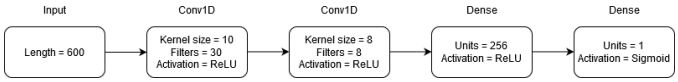


Figura 2. Rete neurale per la lavastoviglie

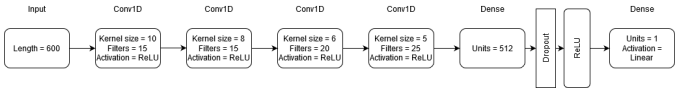


Figura 3. Rete neurale per il frigorifero

## IV. ARCHITETTURE DI RETE

La figura 2 mostra l'architettura della rete neurale addestrata per stimare il consumo elettrico della lavastoviglie (*dw_net*). La rete riceve in input finestre di ampiezza $W = 600$ e si compone di 3 strati nascosti: un primo layer convolutivo caratterizzato da 30 filtri di dimensione 10; un secondo layer convolutivo caratterizzato da 8 filtri di dimensione 8 e un layer fully connected con 256 neuroni. Lo strato di output è costituito da un singolo neurone con funzione di attivazione sigmoidale. La rete descritta differisce da quella proposta da Zhang et al. per: il minor numero di layer convolutivi (2 contro 5); il minor numero di neuroni dell'ultimo strato nascosto (256 contro 1024); la funzione di attivazione dello strato di output (la sigmoide rimpiazza la funzione di attivazione lineare). Nell'insieme la complessità della rete originale è stata ridotta con l'obiettivo di contrastare l'overfitting.

La figura 3 mostra invece la rete usata per stimare la potenza assorbita dal frigorifero (*fr_net*). In questo caso, sono state apportate ulteriori modifiche alla rete originale, ancora una volta allo scopo di ridurre l'overfitting. In particolare: è stato dimezzato il numero di filtri convolutivi; è stato aggiunto un layer di Dropout con rate pari a 0.05 ed è stata introdotta una l2-regularization (non visibile nel diagramma) con fattore di regolarizzazione pari a 0.0001.

In entrambe le reti i pesi dei layer contenenti la funzione di attivazione ReLU sono stati inizializzati mediante la *He Initialization*.

## V. ADDESTRAMENTO

### A. Data Ingestion

I dati di input delle due reti neurali sono fortemente ridondanti: l'approccio sliding window fa sì che il campione di training $i$ abbia molti elementi in comune con il campione $i + 1$. Allocare esplicitamente tutti i campioni all'inizio comporterebbe quindi uno spreco di risorse oltre che un potenziale memory overflow dell'ambiente di esecuzione. Per questo motivo, si è scelto di generare i campioni di training on-the-fly mediante un apposito oggetto Python di tipo *keras.utils.Sequence*. Ogni batch generato contiene 512 coppie $(Y_{t:t+W-1}, x_\tau)$. La dimensione del batch è stata impostata in modo tale da rendere i tempi di addestramento compatibili con le caratteristiche e i limiti d'uso della piattaforma di calcolo a disposizione.

### B. Algoritmo di addestramento

L'addestramento delle reti è stato realizzato mediante l'algoritmo adattativo Adam [5]. Per l'addestramento della *dw_net* i parametri di default dell'algoritmo sono stati lasciati invariati ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$), mentre per la *fridge_net* il learning rate è stato ridotto in cross-validation al valore $0.0001$.

### C. Funzione di loss

Quando si addestra una rete neurale per risolvere un problema di regressione, la funzione di loss comunemente utilizzata è l'MSE (errore quadratico medio). Nel caso di specie, i risultati ottenuti in cross-validation hanno tuttavia spinto ad utilizzare delle loss diverse. Per l'addestramento della *dw_net* è stata usata la binary crossentropy e corrispondentemente la sigmoide è stata impiegata come funzione di attivazione nello strato di output. Le migliori prestazioni offerte dalla binary crossentropy si possono forse attribuire al fatto che la predizione del consumo della lavastoviglie, essendo l'apparecchio ON-OFF, assomiglia ad un task di classificazione binaria in cui istante per istante bisogna determinare se l'apparecchio è spento o acceso. Per la *fridge_net* è stato invece impiegato l'errore medio assoluto (MAE). Si può ipotizzare che la maggiore efficacia del MAE sia legata alla sua minore sensibilità rispetto agli outlier.

### D. Metrica di riferimento

Le prestazioni dei modelli sono state valutate calcolando l'Energy-Based F1 score, una metrica già usata in altri lavori sulla disaggregazione energetica (si veda ad esempio [3]). Poiché la metrica non è disponibile in Keras, per poterla monitorare durante l'addestramento, è stato necessario implementarla da zero estendendo la classe *keras.metrics.Metric*. Nel lavoro di Zhang et al. sono state considerate altre metriche; con ciò si spiegano in parte le modifiche apportate in questo progetto al modello *seq2point* originale.

### E. Early stopping

Allo scopo di contenere l'overfitting, entrambe le reti sono state addestrate applicando una procedura di early stopping basata sull'F1 score del validation set con *patience* pari a 4 per la *dw_net* e pari a 10 per la *fridge_net*. Per l'addestramento finale - sull'intero training set - è stato utilizzato come numero di epoche il valore medio ottenuto ripetendo più volte l'early stopping.

Le figure 4 e 5 mostrano due esempi di disaggregazione effettuata dalla rete sui dati di training.

## VI. Possibili miglioramenti

Compatibilmente con le risorse messe a disposizione dal piano gratuito di Google Colaboratory, non è stato possibile regolare gli iperparametri dei modelli tramite una *Grid Search* esaustiva ma si è dovuto ripiegare su una *Random Search* manuale. É quindi probabile che la configurazione ottenuta sia sub-ottima.

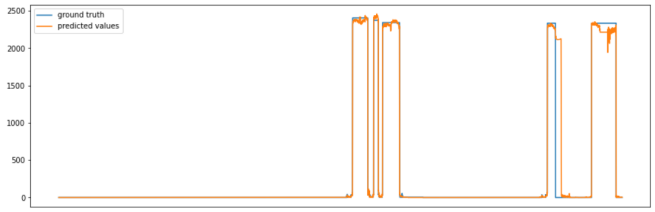


Figura 4. Esempio di disaggregazione per la lavastoviglie

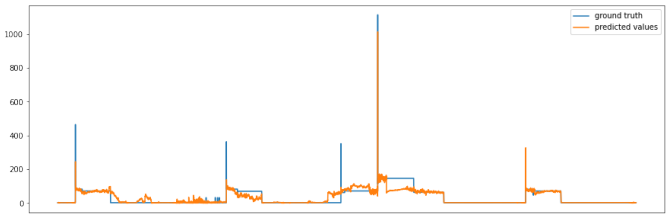


Figura 5. Esempio di disaggregazione per il frigorifero

In [2] le prestazioni delle reti neurali sono state incrementate attraverso un meccanismo di data augmentation, che però in questa sede non è stato possibile attuare in mancanza di dati relativi ad altri elettrodomestici del fabbricato.

Con maggiori risorse a disposizione, sarebbe stato inoltre interessante provare ad addestrare su uno dei dataset di grandi dimensioni disponibili online (eventualmente ricampionato) la rete seq2point originale e successivamente riaddestrarla sui dati di interesse (*fine-tuning*). Il *transfer learning* ha infatti dato prova di poter fornire buoni risultati anche nel campo della disaggregazione dei consumi energetici [6].

## Riferimenti bibliografici


[1] Faustine, A., N. H. Mvungi, S. Kaijage and Michael Kisangiri. "A Survey on Non-Intrusive Load Monitoring Methodies and Techniques for Energy Disaggregation Problem." ArXiv abs/1703.00785 (2017)

[2] Jack Kelly and William Knottenbelt. 2015. Neural NILM: Deep Neural Networks Applied to Energy Disaggregation. In Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments (BuildSys '15). Association for Computing Machinery, New York, NY, USA, 55–64. DOI:https://doi.org/10.1145/2821650.2821672

[3] Roberto Bonfigli, Andrea Felicetti, Emanuele Principi, Marco Fagiani, Stefano Squartini, Francesco Piazza, Denoising autoencoders for Non-Intrusive Load Monitoring: Improvements and comparative evaluation, Energy and Buildings, Volume 158, 2018, Pages 1461-1474, ISSN 0378-7788, https://doi.org/10.1016/j.enbuild.2017.11.054.

[4] Zhang, Chaoyun & Zhong, Mingjun & Wang, Zongzuo & Goddard, Nigel & Sutton, Charles. (2016). Sequence-to-point learning with neural networks for nonintrusive load monitoring.

[5] Kingma, Diederik P. and Jimmy Ba. "Adam: A Method for Stochastic Optimization." CoRR abs/1412.6980 (2015)

[6] D'Incecco, Michele & Squartini, Stefano & Zhong, Mingjun. (2019). Transfer Learning for Non-Intrusive Load Monitoring. IEEE Transactions on Smart Grid. PP. 1-1. 10.1109/TSG.2019.2938068.