UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Corso di: Digital Adaptive Circuit and Learning Systems

A.A. 2023-2024

Confronto tra varie tecniche di Deep Learning per il NILM

Studente: Luca Silvestrini

Sommario

1 INTRODUZIONE

Il machine learning (ML) è un sottoinsieme dell'intelligenza artificiale (AI) che si occupa di creare sistemi che apprendono o migliorano le performance in base ai dati che utilizzano. Il ML supporta una vasta gamma di casi d'uso che vanno dai servizi di finanza, all'ambito sanitario e energetico, ad esempio nella classificazione delle immagini si utilizza algoritmi di machine learning per assegnare etichette a un gruppo predefinito di categorie o a qualsiasi immagine di input. Consente alle aziende di creare modelli 3D e 2D con scopi nell'ambito videomaking e tutto ciò che riguarda la fotografia e nella progettazione e costruzione di edifici ma soprattutto semplifica la comunicazione diagnosi e molto altro. Si è vista la grandissima duttilità del machine learning, ad accompagnarlo ci sono dei metodi chiamati di deep learning che struttura gli algoritmi in modo da generare una rete neurale artificiale che raccoglie informazione dai dati e prende decisioni in autonomia nascono tecniche di apprendimento sempre più affidabili ed efficienti. Negli ultimi 20 anni c'è stato un rapido sviluppo tecnologico nel settore dell'energia elettrica che negli ultimi 20 anni ha portato a una crescita del fabbisogno energetico attraverso l'aumento di elettrodomestici e l'automazione delle attività. Allo stesso tempo gli obiettivi globali di protezione del clima, risparmio energetico ed efficienza energetica, gestione dell'interesse per la riduzione del consumo di energia. Questi requisiti hanno portato alla recente adozione di contatori intelligenti e di reti intelligenti denominate "smart grid" le quali sono in grado di gestire la domanda: servizi dedicati per il risparmio energetico in base alle abitudini del consumatore, risparmio energetico del consumatore finale, efficienza ambientale e, qualora ci fossero dei guasti, si è in grado di accorgersene molto velocemente. In particolare all'assunzione del monitoraggio del carico (LM) mediante disaggregazione dell'energia, denominato monitoraggio del carico non intrusivo (NILM) che consente il monitoraggio dell'energia dello specifico apparecchio osservando solo il consumo aggregato di energia domestica , in tempo reale le informazioni riguardo al livello di ogni singolo appliance può essere utilizzato per ottenere informazioni più approfondite all'origine del consumo di energia per effettuare ottimizzazione e pianificazione strategica del carico. Questa tecnica permette di evitare un costo maggiore perché implica una richiesta molteplice di smart meter (uno per ogni singolo appliance) all'interno di una singola abitazione. Il monitoraggio di cui si è parlato consente inoltre un risparmio in termini economici sia un controllo di possibili malfunzionamenti dei singoli apparati. Quando si parla di NILM si utilizzano tecniche sempre più affini proprio per evitare la complessità di calcolo e ridurre energia per questo motivo si utilizzano degli algoritmi con lo scopo di ottenere transitori on/off e di conseguenza al riconoscimento dello stato di funzionamento degli utilizzatori. Un modello statistico che rappresenta un sistema con stati nascosti, in cui non sono direttamente osservabili ma possono essere dedotti tramiti variabili osservabili è l'HMM (Hidden Markov Model), utilizzato per lo state detection ma anche per riconoscimento del parlato, sia processi del linguaggio naturale, la bioinformatica e il riconoscimento dei pattern. In questa relazione verrà illustrata una rete neurale per il NILM tramite Tensorflow, che permette l'uso di un hardware meno complesso con un conseguente risparmio economico. Per addestrare, valutare e testare è stato utilizzato il dataset pubblico "UK-Dale".

1.1 Introduzione alla prova

Nella seguente relazione verranno introdotti i concetti di RETE NEURALE ARTIFICIALE, in particolare delle reti CNN. Successivamente, verranno illustrate alcune tecniche per il NILM soffermandoci principalmente sulla rete SAMnet, in quanto oggetto di studio della prova. Quest'ultima verrà analizzata nel dettaglio con l'introduzione di un particolare tipo di rete neurale chiamata TCN. Inizialmente La prova da svolgere prevedeva l'addestramento della rete SAMnet per il monitoraggio del carico non intrusivo, ma a causa di problemi nell'implementazione dell'architettura e del data-preprocessing dei dati non si sono ottenuti valori accettabili per poter consentire una valutazione della rete stessa. La prova, quindi, è stata cambiata e sono state valutate e confrontate due architetture differenti: Convolution Neural Network e Temporal Convolution Network. I risultati illustrati nei successivi capitoli, quindi, sono riferiti a CNN e TCN, prendendo come parametri di confronto MAE, SAE e F1- SCORE.

2 LE RETI NEURALI ARTIFICIALI

Le reti neurali artificiali sono un sottoinsieme del machine learning e rappresentano l'elemento centrale degli algoritmi di deep learning. Le ANN (Artificial Neural network) sono modelli matematici che sono in grado di risolvere task come classificazione e regressione. Nella relazione andremo ad analizzare una rete con lavora in multi-task infatti i due task lavorano insieme e le rispettive loss dipendono tra loro. Le reti neurali sono molto potenti perché non sono caratterizzate da un unico processore ma sono composte da livelli di nodi che contengono un livello di input, uno o più livelli nascosti e un livello di output. Ciascun nodo o neurone artificiale, si connette ad un altro e ha un peso e una soglia associati. Questi neuroni lavorano in sintonia e se l'output di qualsiasi nodo è al di sopra del lavoro di soglia specificato, tale nodo viene attivato inviando i dati al successivo livello della rete. Proprio questa collaborazione rende le reti molto potenti e sono in grado di risolvere task complessi e in poco tempo.

Il modello di neurone più utilizzato è il percettrone ed è in grado di svolgere alcune operazioni come:

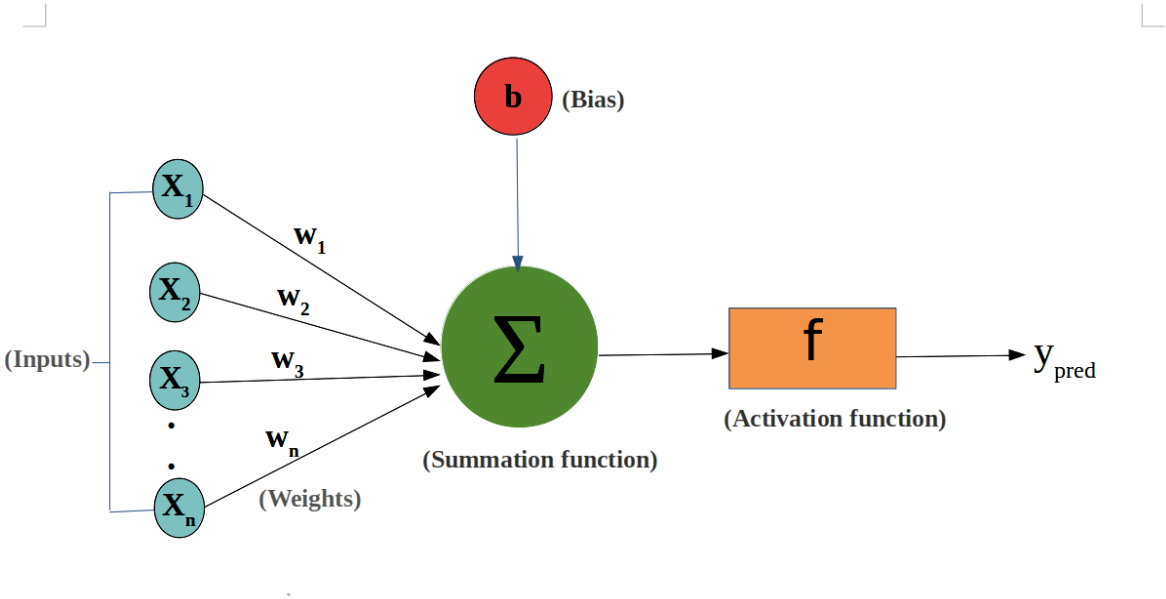


Figura 1 Percettrone

- Combinazione lineare degli ingressi attraverso opportuni pesi.
- Somma di un ulteriore valore detto "bias" al termine precedente;
- Applicazione di una funzione non lineare detta "funzione di attivazione".

Le operazioni svolte sono quindi semplici ma la forza della rete sta nei collegamenti presenti tra i vari neuroni.

2.1 CNN

Le reti CNN (Convolution Neural Network) sono reti neurali artificiali con architettura suddivisibile in due parti. La prima costituita da diversi strati convoluzioni (dinamici e lineari) combinati a strati di pooling che servono a comprimere la rete, la seconda parte è composta da strati costituiti da percettroni (dense layers). La loro idea è quella di utilizzare la parte convoluzionale e di pooling(compressione) per rappresentare il segnale di ingresso in un dominio in cui il segnale è più facilmente classificabile dalla rete MLP che costituisce la parte finale della CNN.

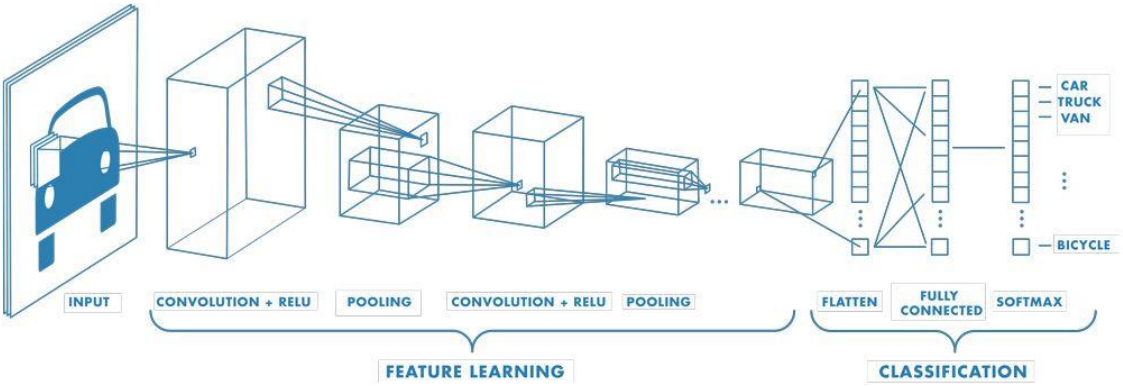


Figura 2 Esempio rete CNN

Le reti CNN sono utilizzate per lo più per il riconoscimento delle immagini seguendo il comportamento dell'apparato visivo umano. La convoluzione implementata solitamente è 2-D, In particolare vengono utilizzate per imaging medico (analizzare esami istologici per rilevare presenza di cellule tumorali), elaborazione audio e rilevamento oggetto.

Le reti neurali vengono addestrate tramite l'invio di input che contengono i features (Feature learning) di ciò che verrà addestrato, in particolare le proprietà individuali. Queste caratteristiche vengono selezionate, raggruppate o trasformate al fine di ottenere un pattern quindi un collegamento diretto con ciò che verrà inviato alla rete.

Viene inviata una matrice di ingresso che viene convoluta con la matrice filtrante (kernel), creando una features maps. Si possono effettuare più convoluzioni usando più di un kernel ognuno che produce una features map differente. Alcune tecniche come il padding permettono di estendere queste matrici e di mantenere invariate le dimensioni dell'output, ad esempio quando si lavora con il bordo delle immagini al fine di mantenere le informazioni e non avere dispersione di feature. Il pooling ha il compito di comprimere e può essere implementato con varie tecniche, la più famosa è la max-pooling che suddivide la matrice in sottomatrici ognuna delle quali vengono compresse mantenendo solo il valore massimo del contenuto. Il calcolo delle derivate delle funzioni per l'aggiornamento dei pesi avviene sempre in maniera backward tenendo conto dei filtraggi e delle compressioni avvenute dei vari strati.

La rete CNN utilizzata per la prova è una struttura tipica seq2point che verrà rappresentata di seguito:

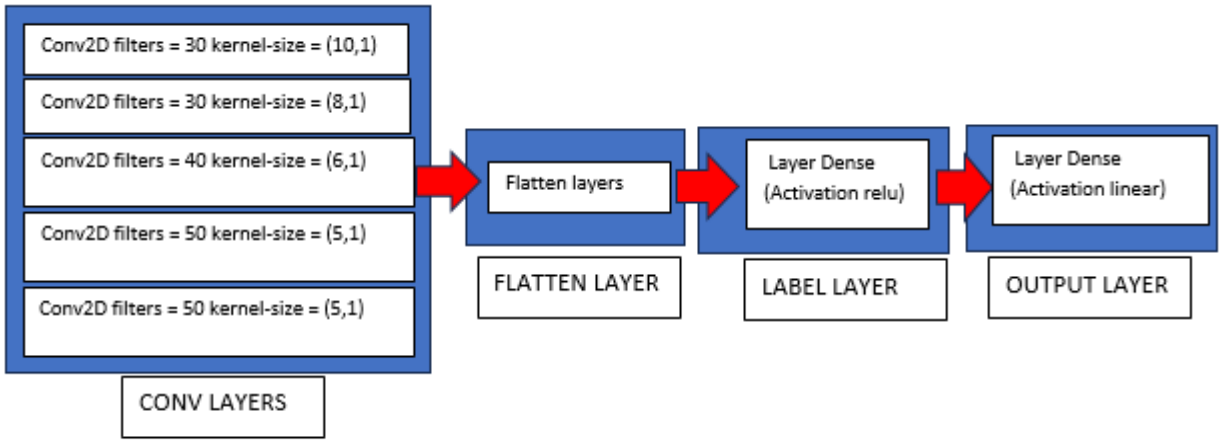


Figura 3 Modello Seq2point utilizzato per la prova

3 RETE SAMNET

La Scale and Attention Experts basata su Multi-Task Neural Network [1], ovvero la rete SAMNET, è una rete CNN frequentemente utilizzata per affrontare i problemi di disaggregazione dell'energia. Si sviluppa in due branch: uno relativo al rilevamento dello stato on/off e uno riferito alla disaggregazione dell'energia.

I tre aspetti principali che costituiscono la rete sono:

1. La scale and attention experts-based multi-task learning neural network è progettato per disaggregation energy e state detection. TCN (dilated temporal convulation) e Self Attention sono introdotti nella rete per l'apprendimento delle conoscenze. L'algoritmo proposto migliora l'accuratezza dello state detection e energy disaggregation.
2. Invece di un settaggio manuale di livelli di soglia (threshold) da dare per ottenere gli stati ON-OFF degli utilizzatori (appliance), introduciamo un algoritmo per l'apprendimento automatico dei valori di soglia, che permette al modello di ottenere grandi valori di accuratezza per il dissagregation energy con l'aiuto delle informazioni ottenute dal task di classificazione.
3. Latency free NILM può essere ottenuto con l'algoritmo proposto, il risultato ottenuto è migliore rispetto alle tecniche sviluppate precedentemente.

3.1 MULTITASK - LEARNING

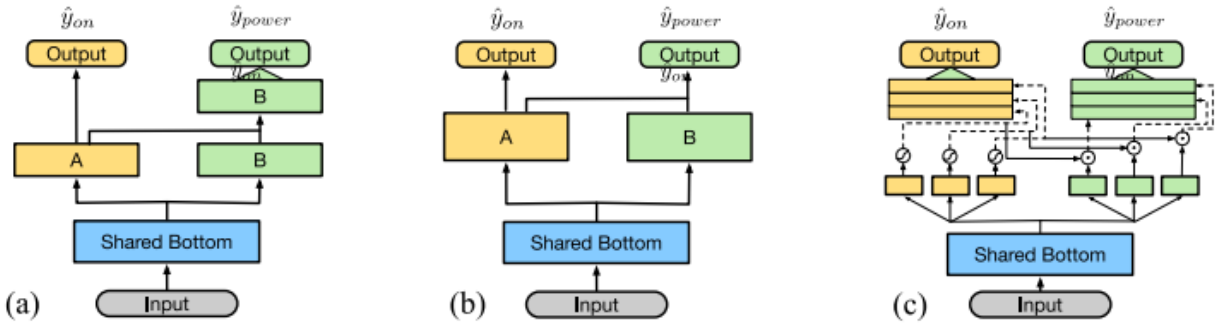


Figura 4 Diversi possibili architetture dei paradigmi multi-tasking

La tecnica multi-task learning rappresenta un paradigma di apprendimento in cui un modello impara a eseguire più compiti contemporaneamente. Il MTL permette di far interagire e creare una correlazione tra più task con lo scopo di ottenere migliori prestazioni in tempi e in qualità dei dati. La condivisione delle conoscenze apprese dal singolo task può contribuire a migliorare le prestazioni di un altro compito correlato. Un caso specifico è quello che sarà analizzato nella relazione dove i due task: "state detection" e "energy disaggregation" sono correlati tra loro, pertanto sarà

introdotto il paradigma MTL. Spesso viene introdotto in tutti quei casi in cui i dati per l'apprendimento sono limitati o costosi da ottenere.

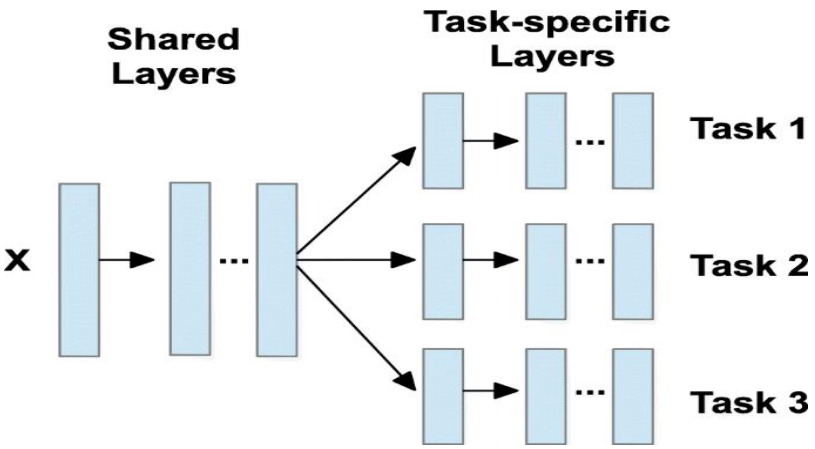


Figura 5 Schema del Multi tasking Learning

Lo schema appena mostrato rappresenta un tipo di struttura MTL:

- SHARED LAYERS: Un esempio potrebbe essere quello dove i primi strati di una rete neurale CNN sono condivisi tra due compiti di classificazione e regressione. Nei successivi livelli saranno poi applicati specifici accorgimenti per ogni singolo task.
- BRANCHING: l'architettura della rete neurale ha diversi rami ognuno dedicato ad un compito specifico. I primi livelli sono condivisi e raccolgono informazioni comuni poi successivamente si creano dei branch specifici ognuno per il compito da svolgere.

L'algoritmo su cui si basa la rete SAMnet è il Multi-gate Mixture of Experts (MmoE) che è un'architettura avanzata utilizzata principalmente nei modelli di apprendimento multi-task (Multi-Task Learning, MTL). Questo approccio è stato introdotto per affrontare la sfida della condivisione delle conoscenze tra diversi task correlati, ottimizzando contemporaneamente le prestazioni per ciascun task specifico. La parte "MoE" dell'architettura si riferisce al concetto di Mixture of Experts, che è un'architettura in cui esistono più "esperti" (cioè, reti neurali o moduli) che lavorano in parallelo. L'idea è quella di creare una gerarchia di livelli, dove ogni livello può apprendere rappresentazioni sempre più complesse e dettagliate dei dati con lo scopo di migliorare la capacità del modello. Per esempio, se stiamo affrontando un problema di classificazione delle immagini, potremmo avere esperti specializzati nel riconoscere forme, colori, texture, etc. Ognuno di questi esperti sarà responsabile di estrarre le caratteristiche rilevanti per il suo aspetto specifico del problema.

3.2 ARCHITETTURA DELLA RETE SAMnet

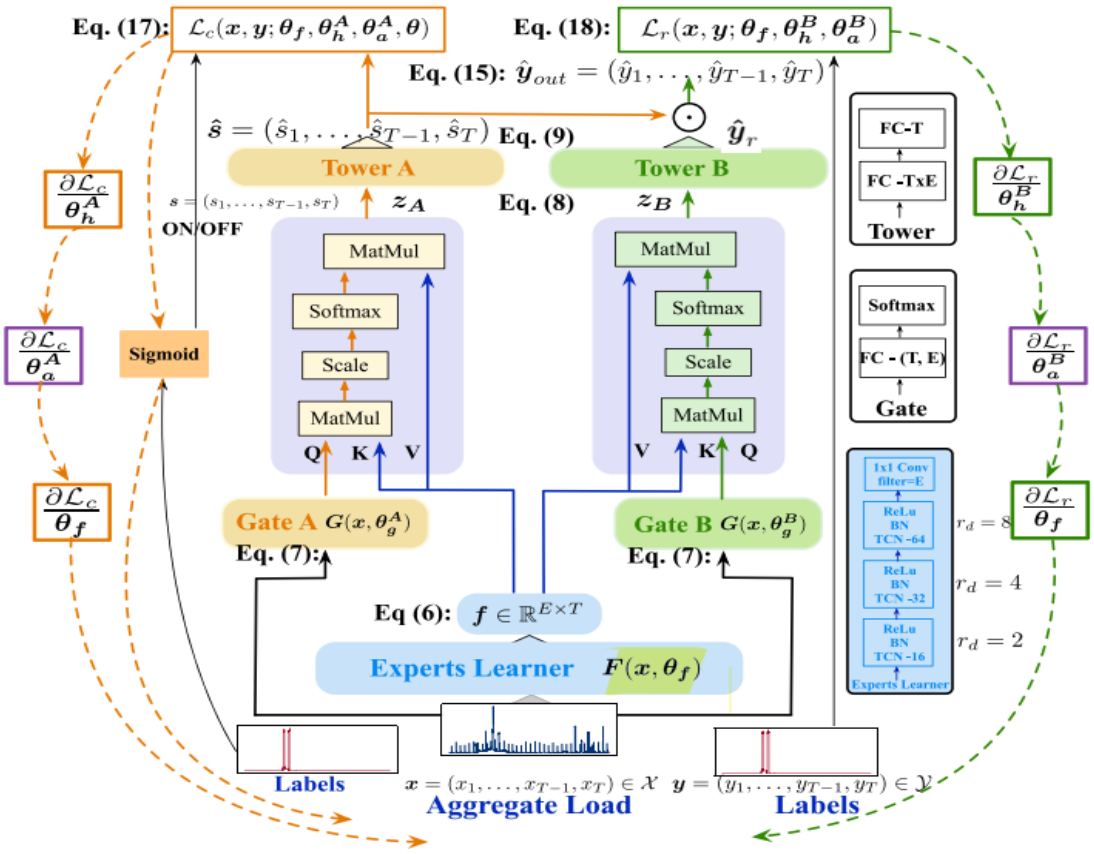


Figura 6 Architettura SAMnet

L'immagine sopra, mostra l'architettura della rete SAMnet, è una rete neurale robusta con latency-free per il NILM che sfrutta la relazione tra i due task: state dection e energy dissagregation.

L'architettura include quattro parti:

1. Expert learner $\mathrm{F}(\mathrm{x}, \theta\mathrm{f})$ Considerando la forte correlazione che esiste tra i due task, i layers condivisi sono i primi ad essere adottati. In questa parte della struttura viene applicata la TCN al fine di raccogliere tante informazioni e condividerle ai due rami che compongono la rete.
   $\mathrm{F}(\mathrm{x}, \theta_f)$ Funzione obiettivo $\mathrm{F} \in \mathrm{R}^{\mathrm{ExT}}$ dove gli esperti vengono condivisi con tutti i task ed E è il numero di esperti. Oltre all'applicazione della TCN vengono applicati layer Fully connected.
   Il FC è un layer in cui ogni unità neurale è connessa a tutte le unità dei layers precedenti.
   $$\mathrm{f} = \mathrm{F}(\mathrm{x}, \theta_f) \quad (1)$$

2. Gate: Il gate impara a selezionare informazioni per un task specifico raccogliendole dalle features condivise precedentemente. Il gate è composto da FC layer con softmax activation.
   $$\mathrm{g_J} = \mathrm{G}(\mathrm{x}, \theta_g) \in \mathrm{R}^{\mathrm{ExT}} \quad (2)$$

Anziché aggiungere semplicemente l'uscita del gate con gli esperti f verrà introdotto un meccanismo di auto-attenzione (self-attention) [4] che realizza una fusione dei pesi tra g_j e f. La struttura di self-attention utilizzata nella rete SAMnet è lo scaled-dot-product-attention essa permette di calcolare le relazioni tra diverse parti di una sequenza di input, consentendo al modello di focalizzarsi sulle parti più rilevanti dell'input per la generazione dell'output. Questo componente è molto più veloce di altre funzioni di attention e produce una matrice con alta ottimizzazione che misura la compatibilità tra i valori di query e di ogni chiave.

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = \text{softmax}\left(\frac{\text{Q}K^T}{\sqrt{d_k}}\right)\text{V} \quad (3)$$

Analizziamo da vicino la funzione di Attention:

- Input Embedding: gli embedding dell'input vengono trasformati in tre matrici: Query (Q), Key (K) e Value (V) tramite trasformazioni lineari.
- Calcolo delle Score: lo score rappresenta una misura di similarità tra la query e le key. Si calcola il prodotto scalare tra Q e K trasposto e lo si scala dividendo per d_k.

$$\text{scores} = \frac{\text{QK}^\text{T}}{\sqrt{\text{d}_\text{k}}} \quad (4)$$

- Applicazione della Softmax: la softmax viene applicata agli score per ottenere i pesi di attenzione.
- Ponderazione dei Valori: i pesi di attenzione vengono utilizzati per calcolare una combinazione ponderata dei valori V.

3. Tower

   Il risultato dell'assemblamento degli esperti è passato al singolo task per focalizzarsi sui rispettivi obiettivi. Due FC layers compongono la tower che permettono l'ottenimento della predizione del consumo energetico (power [W]) e il rilevamento dello stato on/off.

   I valori ottenuti sono compresi tra 1 e 0 dovuti all'applicazione della normalizzazione con funzione esponenziale.

4. Automatic learning of the on/off state

   Senza dover raccogliere i dati riferiti allo stato degli switch, definiamo le label on/off degli stati degli utilizzatori, impiegando i livelli di soglia opportunamente scelti per ogni dispositivo al segnale di ground truth s(t) al fine di identificare in maniera automatica gli stati on/off. Per poter ottenere la probabilità degli stati on/off la potenza consumata dall'appliance è passata attraverso la funzione Sigmoid.

3.2.1 Temporal Convolution Network

La Temporal Convolution Network [7] è una parte fondamentale presente nella struttura della rete SAMnet in particolare situata nel Gate. La TCN è una classe particolare di reti neurali progettate per elaborare sequenze temporali, esse sono una valida alternativa alle reti RNN come, ad esempio, LSTM (Long-short Term Memory).

Le caratteristiche principali sono:

1. La convoluzione causale assicura che l'output in un momento t dipenda esclusivamente dagli input fino a quel momento, evitando così il cosiddetto "leakage" di informazione, ossia la trasmissione di informazioni future ai dati passati. Questo preserva la sequenzialità temporale dei dati e garantisce che l'analisi rimanga coerente nel tempo.
2. Dilated Convolutation: Le convoluzioni dilatate permettono di gestire lunghe dipendenze temporali espandendo il campo ricettivo senza incrementare significativamente il numero di parametri o la complessità computazionale, inoltre inserisce spazi tra i valori dei vari kernel permettendo di coprire un ampio intervallo temporale senza aumento diretto delle dimensioni del kernel. Gli spazi tra i valori dei Kernel sono inizializzati tramite il dilation rate. Di solito viene vengono utilizzate dei Dilated Residual Blocks in cascata con dilation rate ciascuno che aumenta in maniera esponenziale (es. 2, 4, 8…) consentendo una computazione più efficiente rispetto all'aumento lineare infatti permettono di coprire lunghe sequenze senza bisogno di aumentare il numero di parametri o la complessità della rete.

3. Residual Connections: Le connessioni residuali sono implementate per facilitare l'addestramento di reti profonde, migliorando la propagazione del gradiente.

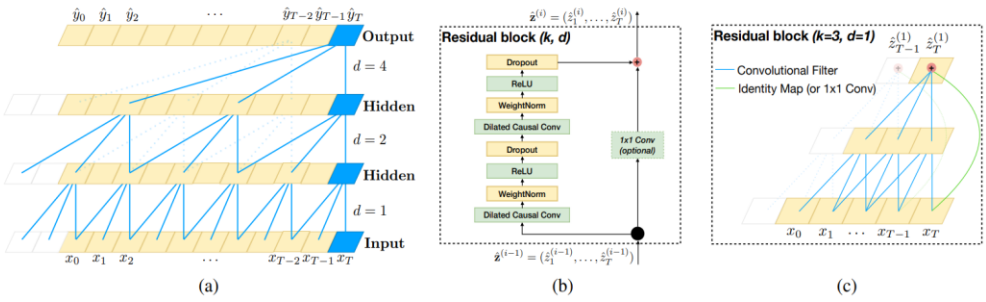


Figura 7. Elementi che compongono l'architettura della TCN. a) Convoluzione dilatata e causale con fattori di dilatazione d = 1,2,4 e filtro k = 3. b) Architettura TCN. c) esempio di connessione residuale nella TCN. Le linee blu sono i filtri nelle funzioni residuali mentre quelle in verde sono le mappature identitarie f(x) = x cioè una funzione che mappa un input su se stesso.

3.2.2 GATE

Il gate è composto dal blocco di self- attention.

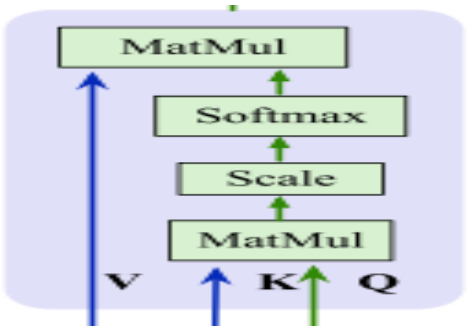


Figura 8 Blocco Self-attention

Il meccanismo di Self-Attention è ispirato al processo di percezione umano è utilizzato per guidare la rete a focalizzarsi sulle più importanti componenti informative del singolo task e in questo lavoro è adottato al fine di combinare le informazioni tra i differenti task. Il meccanismo si basa sul calcolo dei pesi di attenzione tra i token all'interno di una sequenza.

Questi pesi di attenzione riflettono la rilevanza di ciascun token rispetto agli altri, consentendo al modello di concentrarsi sui token più importanti per comprendere il contesto complessivo della sequenza.

- Calcolo dei punteggi di attention:
  In punteggi di attenzione sono calcolati attraverso il prodotto scalare tra i vettori di query e chiavi.

$$\text{Attention}(\text{Q}, \text{K}) = \sqrt{d}_k \text{Q}K^T \quad (5)$$

La formula appena illustrata è l'espressione Standard per la definizione dell'operazione di attenzione (Attention mechanism) in cui si normalizza il prodotto per evitare problemi di instabilità numerica e si applica una Softmax per ottenere i pesi di attenzione, dove Q rappresenta i vettori di query, K rappresenta i vettori di chiave.

I vettori di query è un vettore che rappresenta una trasformazione dell'input originale, viene utilizzato per calcolare i punteggi di attenzione nella self-attention. Viene calcolato moltiplicando l'input originale per la matrice di trasformazione appresa durante l'addestramento del modello.

$$\text{Qt} = \text{Xt} * \text{W} \quad (6)$$

Il vettore di query rappresenta l'informazione estratta dall'input originale in relazione all'informazione su un particolare aspetto della sequenza. Questo vettore viene quindi utilizzato insieme al vettore di chiave per calcolare i pesi di attenzione.

Successivamente viene normalizzato attraverso la funzione di softmax, i valori normalizzati ottenuti vengono utilizzati per pesare i valori corrispondente ai token nella sequenza.

3.5.3 TOWER

I Tower presenti nell'architettura della rete costituiscono i task specifici per focalizzarsi sui rispettivi obiettivi. I Tower sono costituiti da 2 FC layers, la predizione di consumo di potenza $\hat{y}$ e degli stati on/off $\hat{s}$ può essere quindi ottenuta:

$$\hat{s} = Sigmoid\left(H_A(Z_A, \theta_h^A)\right) \quad (7)$$

$$\hat{y_r} = \left(H_B(Z_B, \theta_h^B)\right) \quad (8)$$

I valori di $\hat{s}$ sono limitati tra 0 e 1 con una normalizzazione esponenziale,

$$\text{Sigmoid } \delta(x) = \frac{1}{1+e^{-x}} \quad (9)$$

$\theta_h^*, \theta_f^*, \theta_a^*, \theta_g^*$ sono dei parametri ottimizzati

3.5.4 AUTOMATIC LEARNING DEGLI STATI ON/OFF

L'algoritmo di Automatic Learning degli stati on/off è utile al fine di non registrare i dati del cambiamento degli stati, nominando la label on/off degli stati degli apparati. Esiste proprio questo metodo che è in grado di decidere se un apparato è acceso o spento senza settare manualmente le singole soglie. Così l'etichettatura degli stati s(t) può essere usato come ground truth per la sotto rete che sarà inoltre il risultato nei differenti stati on/off, per stessi valori di potenza. La potenza è passata attraverso la funzione Sigmoid al fine di ottenere la probabilità degli stati on/off.

$$P(s(t) = on|y(t)) = \delta(y(t)) = \frac{1}{1 + e^{-y(t)}} \quad (10)$$

Per evitare gli effettui anche se piccoli dei rumori nelle misurazioni degli stati on s(t) , piccolo valore ζ è sottratto all'uscita y(t)

$$P(s(t) = on|y(t)) = \delta(y(t)) = \frac{1}{1 + e^{-(y(t)-\zeta)}} \quad (11)$$

3.3 CONFRONTO CON ALTRE RETI SIMILI

L'algoritmo appena illustrato rappresenta la rete SAMnet che verrà confrontato con altri multi-task learning-based NILM.

Una delle architetture con la quale verrà confrontata la rete proposta è la Baseline che in sostanza è una parte della rete SAMnet, questa parte della rete può anche essere utilizzata da sola, il risultato ottenuto è sicuramente inferiore rispetto alla rete SAMnet.

Le altre reti messe a confronto sono rappresentate nella figura 6 di seguito.

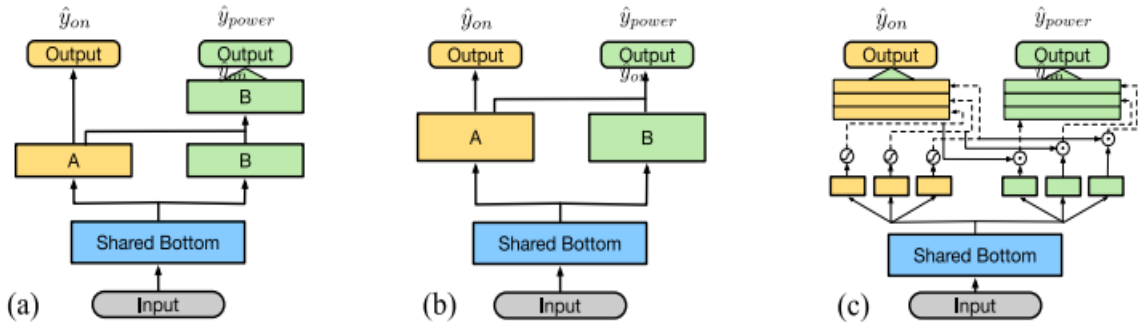


Figura 9 Reti a confronto a) GRUnet b) MGRU c) SCA

BASELINE è una parte della rete proposta senza la fusione dei due task. In particolare, è composto solamente da Expert Learner, il gate, self attention e tower riferiti al task di energy dissagregation.

GRUnet: La rete viene progettata per problemi di time-series e verrà comparata con la TCN-based proposta precedentemente.

MGRU**:** La rete verrà paragonata con la rete SAMnet entrambe contengono la fusione dei due task.

SGN: In questa architettura è stato settata la potenza di soglia pari a 10W per distinguere se un utilizzatore è on/off. Alla fine, verrà combinata la regressione e la classificazione delle due sottoreti per migliorare le performance della rete.

SCA: Propone un'architettura ancora più complessa, molto più complessa la fusione delle due subnetwork

4 SETUP SPERIMENTALE

4.1 DATATSET E DATAPREPROCESSING

Il dataset utilizzato per allenare e testare la rete è il dataset UK-DALE esso contiene la raccolta dati di consumo di aggregato e di ogni singolo apparato, in particolare l'aggregato è campionato a 1s mentre per i singoli apparati si ha un campionamento pari a 6s. I dati sono riferiti a 5 case residenziali locate in UK.

Le case 2 e 5 verranno utilizzate per il training mentre la casa 1 per il test. Le misure registrate sono riferite a questi utilizzatori: kettle, dishwasher, microwave, washing machine, fridge.

Il pre-processing dei dati è stato realizzato in questo modo: la prima azione da fare per poter ottenere dati coerenti tra loro è il sampling, infatti, deve essere di periodo 6s, sia per aggregato che ground-truth. Successivamente, avviene una rimozione delle righe con dati mancanti o nulli. In seguito, viene applicata una finestratura sulla sequenza di dati, con una finestra di lunghezza T = 600. Questa finestra scorre lungo tutta la sequenza con uno step pari a d, il cui valore varia a seconda dell'utilizzatore in prova. Per quanto riguarda lo state-detection, vengono presi campioni random con y(t) >= 10 watt: si rappresentano tutti i campioni con stato on, mentre gli altri con stato off, si rappresentano con lo scopo di ottenere un rapporto tra stati on e off pari 1:1. Infine, viene adottata una normalizzazione: $x = \frac{x}{xmax}$, $x = (x1, \ldots . . xt)$, y=(yk,..,yt-k), dove k rappresenta il ritardo di monitoraggio. Nel caso del dataset UK-DALE viene utilizzato K pari 200.

4.2 METRICHE

La funzione di perdita del nostro modello proposto è progettata come segue:

$$\mathcal{L} = \lambda \mathcal{L}c + \mathcal{L}r \quad (12)$$

dove λ e 1 sono i pesi usati per bilanciare le perdite dei due compiti. L'equazione 12 rappresenta una semplice combinazione lineare delle perdite per task. $\mathcal{L}c$ e $\mathcal{L}r$ rappresentano rispettivamente la funzione di perdita di entropia incrociata per il problema di classificazione e la funzione di perdita dell'errore quadratico medio (MSE) per il problema di regressione.

Nell'algoritmo presentato è stato implementato un aggiornamento della lambda ad ogni epoca in Tensorflow attraverso il parametro callback.

La valutazione del modello in fase di test viene svolta prendendo in considerazione principalmente due metriche: MAE e SAE.

Il MAE (Mean Absolut Error), esprime la differenza media presente tra il ground truth (x_t) e la curva predetta ($\hat{x}_t$) dalla rete neurale in ogni istante temporale. Questo parametro è definito come:

$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|x_t - \hat{x}_t| \qquad (13)$$

il SAE (Signal Aggregate Error) denota l'errore totale di consumo di energia in un periodo di tempo, può essere espresso come:

$$SAE = \frac{|r - \hat{r}|}{r} \qquad (14)$$

$$r = \sum_{t=1}^{T} x_t \qquad \hat{r} = \sum_{t=1}^{T} \hat{x}_t \qquad (15)$$

dove r indica l'energia totale consumata dall'appliance e $\hat{r}$ corrisponde all'energia totale predetta.

Si prendono in considerazione ulteriori tre parametri che si riferiscono alla sola attivazione dell'appliance:

Precision: fa il rapporto tra la somma di tutti i true positive (TP) diviso la somma dei true positive e false positive (FP):

$$PRECISION = \frac{\sum TP}{\sum (TP + FP)} \qquad (16)$$

Recall: rapporto tra la somma dei true positive e la somma tra i true positive e false negative (FN)

$$RECALL = \frac{\sum TP}{\sum (TP + FN)} \qquad (17)$$

F1 SCORE: è data dalla combinazione tra precision e recall e può essere espressa come:

$$F1\ SCORE = 2 * \frac{PRECISION * RECALL}{PRECISONI + RECALL} \qquad (18)$$

Per identificare gli eventi (true positive, true negative, false positive e false negative) si comparano il disaggregato predetto e il ground truth con la soglia di attivazione (on power threeshold) relativa ad ogni appliance.

4.3 IPERPARAMETRI DELLA RETE NEURALE

I parametri che caratterizzano l'architettura della rete su cui sono stati implementati gli algoritmi (noti come iperparametri), sono i seguenti:

1. Batch size: 128; indica il numero di campioni (finestre di lunghezza 600 del training dataset) che vengono propagati nella rete prima dell'aggiornamento dei pesi. Un valore elevato di batch richiederà una memoria elevata ma consentirà un addestramento più accurato e quindi una stima del gradiente meno rumorosa.
2. Crop: 3000000; indica il numero di righe del dataset da prendere contemporaneamente per estrarre i dati di training.
3. Epochs: 300; ogni epoca consiste nel passaggio alla rete dell'intero set di dati per l'addestramento della stessa; essa comprende sia la fase backward (per il calcolo delle derivate e l'aggiornamento dei pesi) che la fase forward (per la predizione dell'uscita). È possibile fissare un valore massimo raggiunto il quale il training termina. Il motivo di questa scelta è dettato dal fatto che si tenta di far terminare il training preventivamente dalla callback early stopping.

4. Input window legth: 600; essa indica il numero dei campioni del dataset in ingresso alla rete seq2sequence.
5. Validation frequency: 1; indica la frequenza di validazione dell'apprendimento della rete. Pari ad 1 significa che la fase di validazione avviene una sola volta in un'epoca, più precisamente alla fine di quest'ultima.
6. Ottimizzatore: ADAM; esso rappresenta la strategia utilizzata per l'aggiornamento dei pesi. Ad oggi è uno degli ottimizzatori più performanti in termini di rapidità di raggiungimento del minimo della funzione costo anche in presenza di punti di sella. Il suo modo di operare è quello di trattare il gradiente della funzione costo (CF) come fosse un rumore, stimando media m[n] e varianza v[n] in questo modo:

$$m[n] = \beta_1 m[n-1] + (1-\beta_1)\frac{\partial CF}{\partial W_{kj}^{(l)}}[n] \qquad (19)$$

$$v[n] = \beta_2 v[n-1] + (1-\beta_2)\left(\frac{\partial CF}{\partial W_{kj}^{(l)}}[n]\right)^2 \qquad (20)$$

   $W_{kj}^{(l)}[n]$ indica il peso del k-esimo neurone dell'l-esimo layers relativo al j-esimo ingresso all'n-esimo campione di ingresso, mentre β_1 e β_2 sono costanti arbitrarie minori e prossime ad 1.
   Per evitare la depolarizzazione si applicano alla media e varianza le seguenti formule di correzione:

$$\hat{m}[n] = \frac{m[n]}{(1-\beta_1^n)} \qquad \hat{v}[n] = \frac{v[n]}{(1-\beta_2^n)} \qquad (21)$$

   Tali correzioni diminuiscono sempre più, all'avanzare dei campioni del dataset, l'effetto della stima di media e varianza del gradiente sull'aggiornamento dei pesi.
   L'aggiornamento dei pesi è dettato dalla "ADAM update rule" così definita:

$$W_{kj}^{(l)}[n+1] = W_{kj}^{(l)}[n] - \frac{\mu}{\sqrt{\hat{v}[n]} + \epsilon}\hat{m}[n] \qquad (22)$$

   μ ed ϵ rappresentano rispettivamente il learning rate e il coefficiente per evitare la divisione per 0.

7. Il learning rate: 0.01 controlla la proporzionalità tra il gradiente della funzione costo e la variazione dei pesi del modello verso la discesa dello stesso. Un valore troppo piccolo può comportare un lungo processo di training, mentre un valore troppo grande può comportare un apprendimento eccessivamente "rumoroso".
8. Early Stopping: è un algoritmo fondamentale per interrompere il processo di training nel punto ottimale evitando under ed over fitting. Opera attraverso: il monitoraggio di una metrica selezionata, un valore di min_delta, un valore di pazienza opportunamente selezionato. Il min_delta indica la variazione minima della metrica osservata per considerare un miglioramento della stessa. La pazienza indica il numero di epoche per cui la metrica monitorata può non migliorare prima di interrompere il processo di training. Quando ciò accade il training si interrompe ripristinando la configurazione dei pesi che assicurava il miglior valore della metrica monitorata.
   Si è scelto di utilizzare la metrica di Mean Square Error in fase di validation, con una pazienza pari a 10 con 50 epoche per il training.

5 PROVE ESEGUITE E RISULTATI

Sono state eseguite diverse prove, con parametri differenti per ciascuna test, in particolare sono state eseguite prove su due diversi algoritmi: CNN e TCN. Per confrontare i due diversi algoritmi ed osservare le prestazioni ci si è basato sulle metriche considerate (MAE, SAE, precision, recall e F1- score). Nella prima prova è stata fatta lavorare la TCN con una parte dei crop presenti nel dataset mentre nella seconda prova sono stati fatti girare entrambi gli algoritmi (TCN, CNN) con tutti i crop disponibili. Nell'effettuare la seconda prova sono stati inoltre aumentati il numero di epoche e il numero di patience (parametro utilizzato nell'early stopping, al fine di evitare l'overfitting durante il processo di addestramento del modello. Se le prestazioni (loss o accuratezza) smettono di migliorare per un certo numero di epoche consecutive l'addestramento viene interrotto anticipatamente. Il parametro di patience determina il numero di epoche che l'algoritmo aspetterà prima di fermare l'addestramento se non viene rilevato alcun miglioramento.

Verrà mostrato di seguito in maniera tabellare sintetica i valori ottenuti di SAE, MAE, F1-SCORE per le prove svolte.

Caso Dishwasher

	MAE	SAE	F1-SCORE
TCN (500.000 crop)	55.07 W	0.01	0.32 (0.50/99.17 W)
TCN (1.051.073 crop)	46.52 W	0.06	0.43 (0.56/209.44 W)
CNN (1.051.073 crop)	36.36 W	0.42	0.59 (0.73/129.15 W)

Confronto tra i vari algoritmi

Metrics	Methods	UK-DALE Dataset					
		Kettle	W.M	D.W	Microwave	Fridge	Avg.
MAE (Watts)	Baseline	10.01	53.28	20.57	19.61	27.40	26.17
	GRUNet [17]	11.61	53.94	19.62	28.77	25.31	27.85
	SGN [18]	11.26	52.61	25.05	16.55	31.08	27.31
	SCA [19]	29.93	57.01	59.02	33.01	35.44	42.88
	MGRU [39]	13.25	58.83	19.86	20.33	39.42	30.34
	Ours	**9.33**	**42.36**	**19.44**	**10.18**	**24.57**	**21.18**
SAE (Watts)	Baseline	6.32	49.17	18.57	18.62	17.11	21.96
	GRUNet [17]	6.5	48.14	17.51	26.34	15.65	22.83
	SGN [18]	7.45	38.08	21.89	15.69	18.63	20.35
	SCA [19]	19.64	49.14	50.74	29.28	22.13	34.19
	MGRU [39]	10.45	51.37	**16.27**	17.02	22.43	23.51
	Ours	**4.49**	**35.61**	16.56	**8.96**	**13.39**	**15.80**
f_1 (%) ($\hat{\boldsymbol{y}}_{out}$)	Baseline	78.12	42.18	80.37	76.71	**74.42**	70.36
	GRUNet [17]	64.49	39.27	86.09	72.90	73.60	67.27
	SGN [18]	73.19	30.17	79.70	83.00	46.20	62.45
	SCA [19]	56.70	27.59	53.84	61.32	48.46	49.58
	MGRU [39]	67.48	32.88	86.47	81.05	61.72	65.92
	Ours	**94.69**	**67.48**	**93.02**	**90.81**	59.60	**81.12**
f_1 (%) ($\hat{\boldsymbol{s}}$)	GRUNet [17]	45.15	35.50	58.56	40.20	30.13	41.91
	SGN [18]	50.34	37.52	50.13	42.15	52.57	46.54
	SCA [19]	31.25	31.13	39.10	25.60	43.18	34.05
	MGRU [39]	46.54	63.89	70.15	48.60	42.06	54.25
	Ours	**66.96**	**88.75**	**87.92**	**54.01**	**81.18**	**75.76**

Prima di tutto è stato creato un piccolo programma per il pre-processamento del dataset come spiegato nel paragrafo 4, poi successivamente è stato implementato la rete e l'algoritmo in Tensorflow per i vari training, validation e test. Dopo l'implementazione si è analizzato i grafici di loss per regressione e classificazione e la combinazione di entrambi al variare delle epoche. Per la verifica delle loss si sono utilizzati valori di epochs piccoli al fine di velocizzare il processo.

Successivamente, allo stesso modo per i valori delle loss di training è stato fatto per la validazione con un dataset pari al 30% di quello di training.

5.1 RISULTATI A CONFRONTO DELLE 5 APPLIANCE

Di seguito verranno presentati i dati ottenuti dalle varie prove svolte.

Lo sviluppo è stato eseguito in ambiente COLAB, in particolare è stata utilizzata la CPU A100 GPU.

Successivamente visualizzeremo la rappresentazione delle varie epoche:

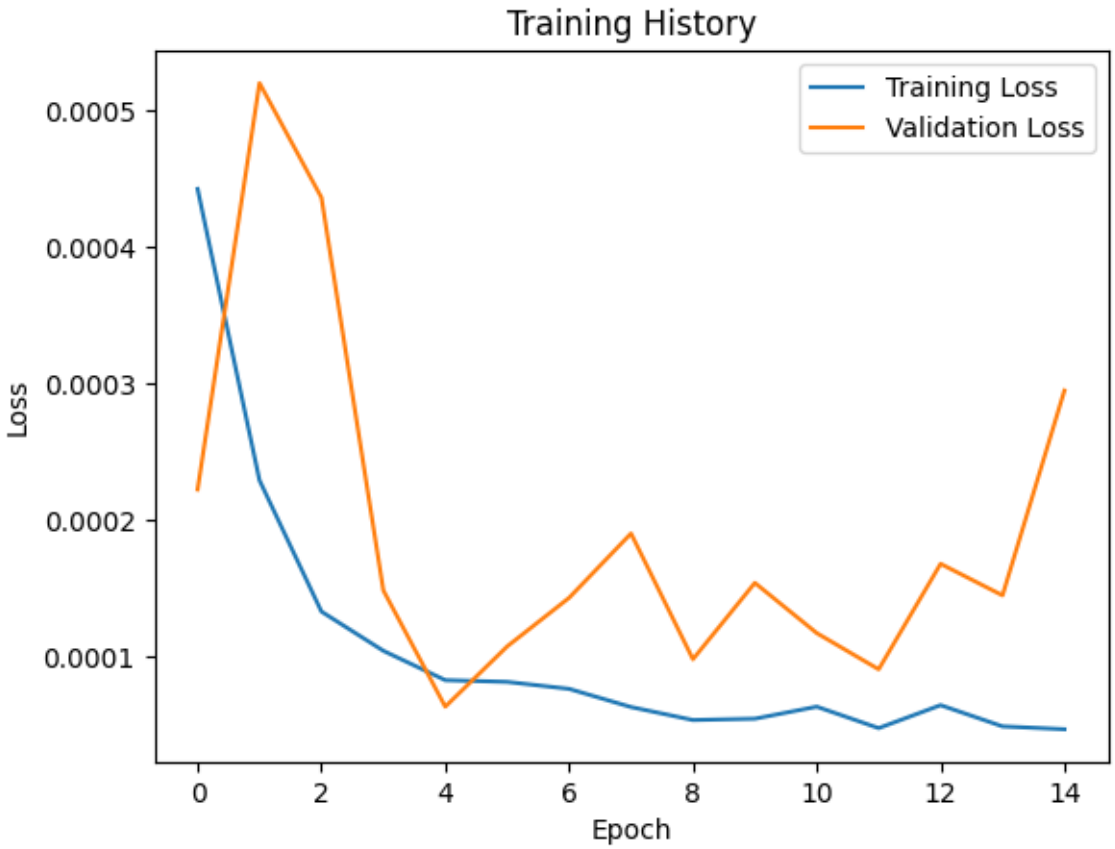


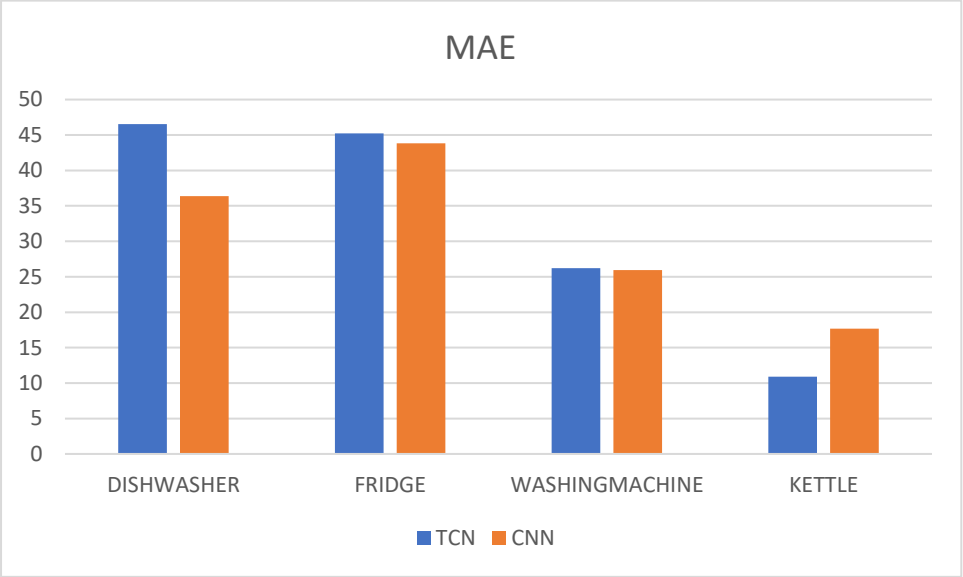
Figura 10 Andamento Training Loss, Validation Loss architettura CNN per Dishwasher

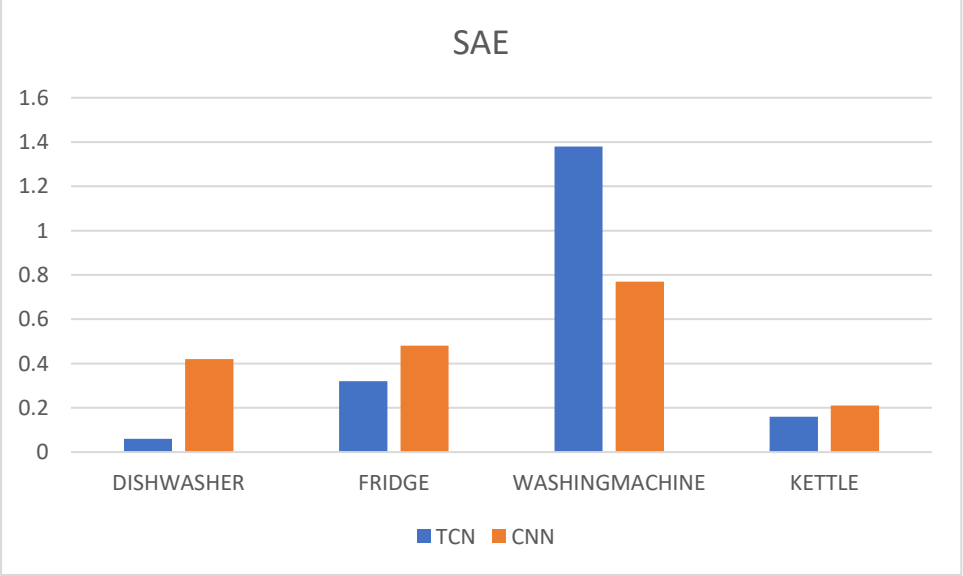
Il numero di epoche impostato in fase di training è pari a 50 ma grazie all'utilizzo dell'early stopping in fase di training ci si ferma alla -15-esima epoca proprio per il fatto che non vi è alcun miglioramento al passare delle epoche. Questa funzione è molto utile per migliorare l'efficienza computazionale e aumentare la velocità dei training. Dal grafico appena

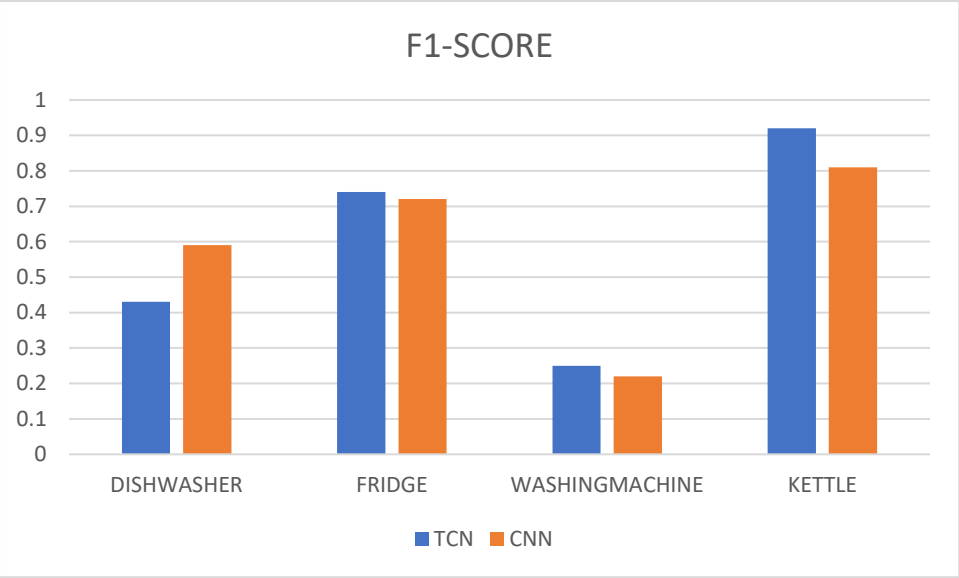
mostrato possiamo notare i valori di Training Loss e di Validation Loss entrambi sono riferiti al MSE (Mean Squared Error), soprattutto per i valori di Training-loss si osserva nettamente la diminuzione all'aumentare delle epoche. L'obiettivo è quello di minimizzare le Loss. Questo fatto è molto importante perché significa che il modello sta facendo delle previsioni più accurate.

Di seguito sono mostrati i risultati a confronto dei 4 elettrodomestici, il Microwave non è stato potuto confrontare perché i dati ottenuti con TCN non sono ammissibili e coerenti.

Ci soffermiamo ai 4 Elettrodomestici così da poter notare un risultato migliore con TCN rispetto a CNN soprattutto per quanto riguarda F1-SCORE.

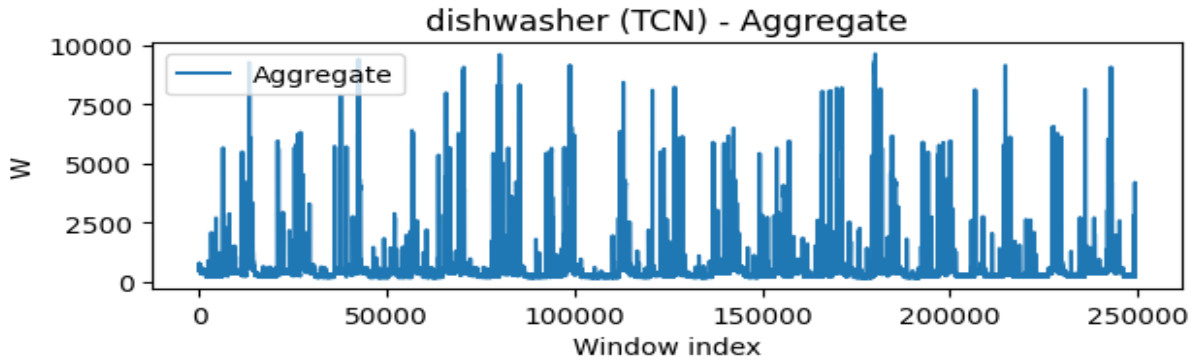


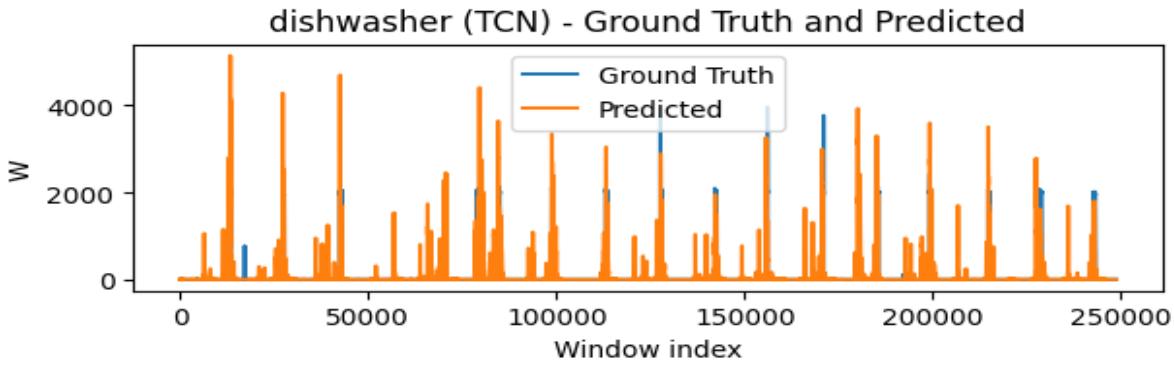


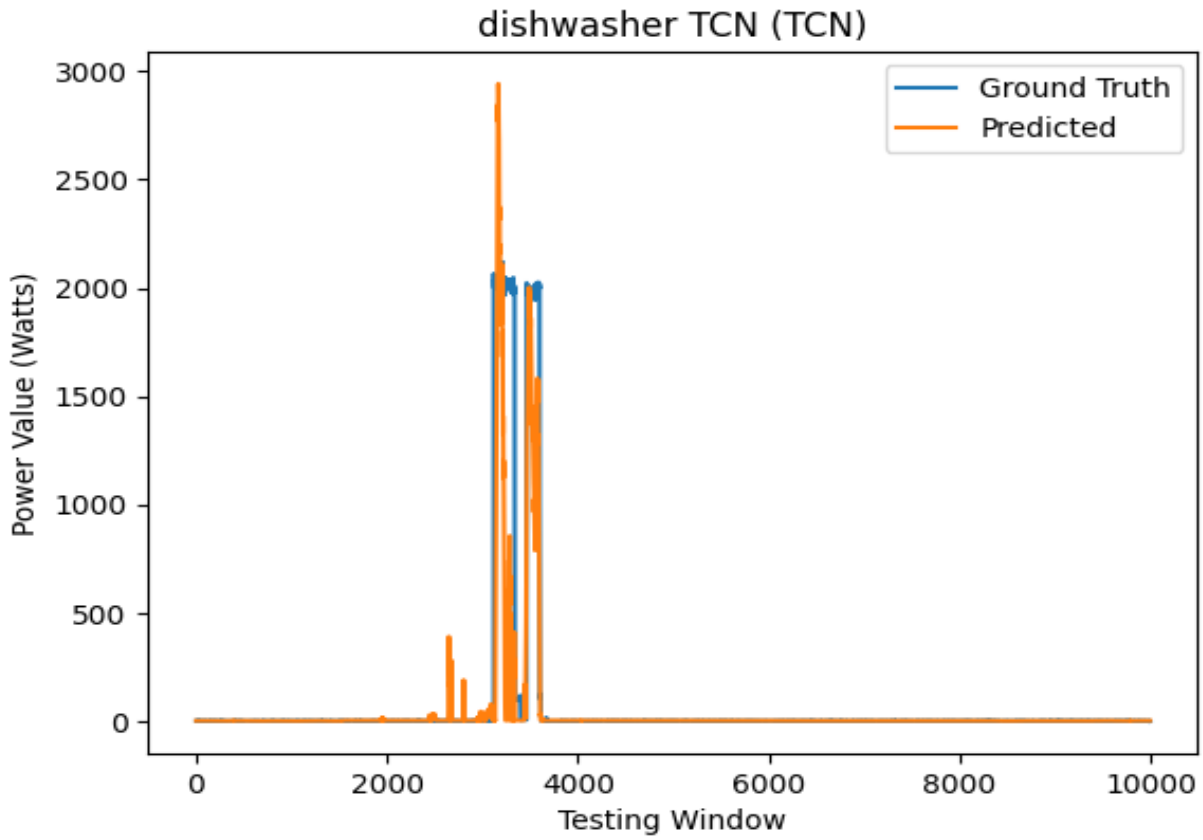

F1-SCORE
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
DISHWASHER
FRIDGE
WASHINGMACHINE
KETTLE
TCN
CNN

5.2 RISULTATI TCN

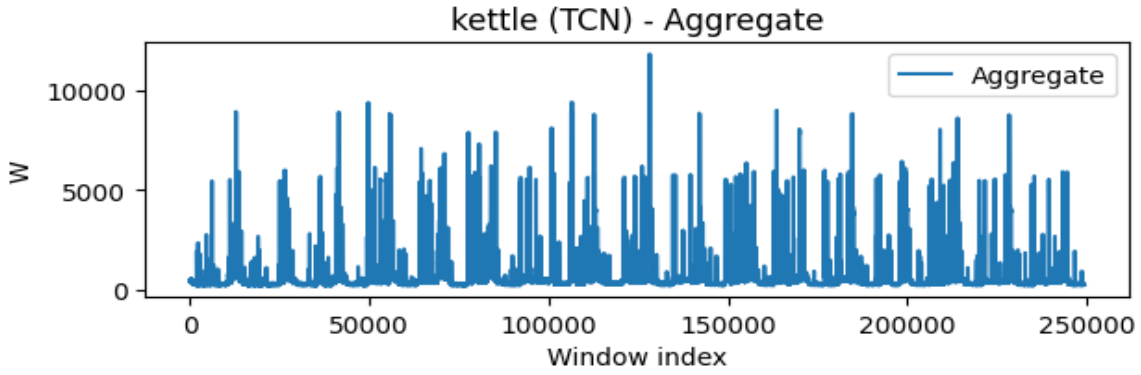
5.2.1 RISULTATI TCN Dishwasher

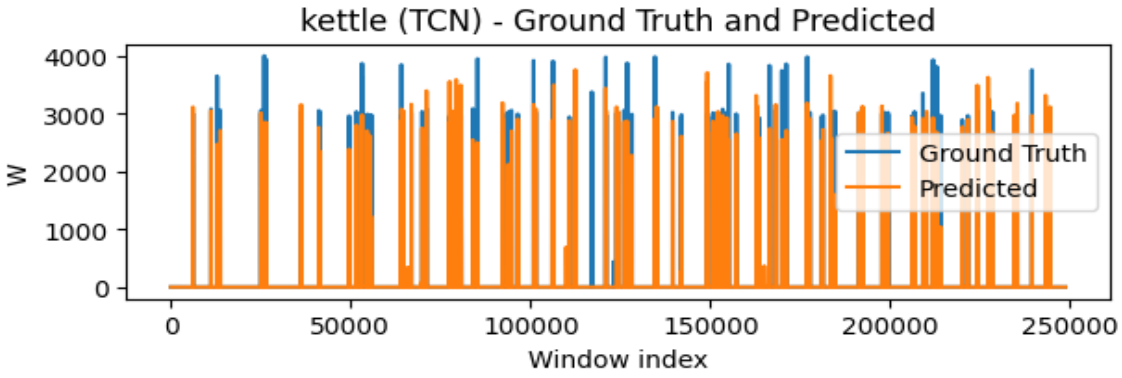


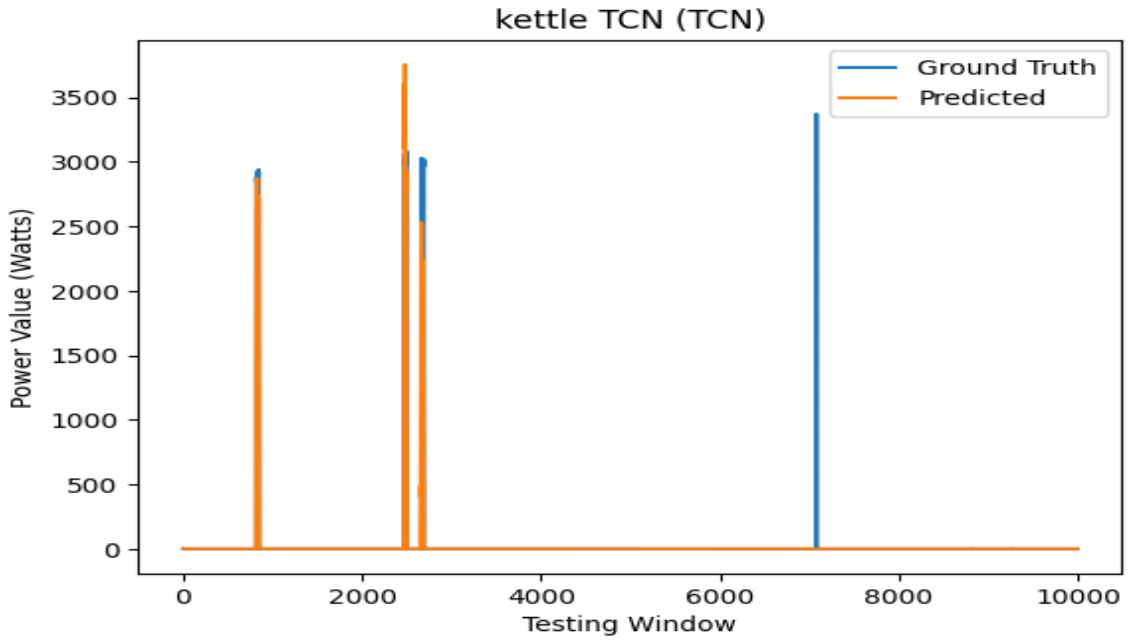




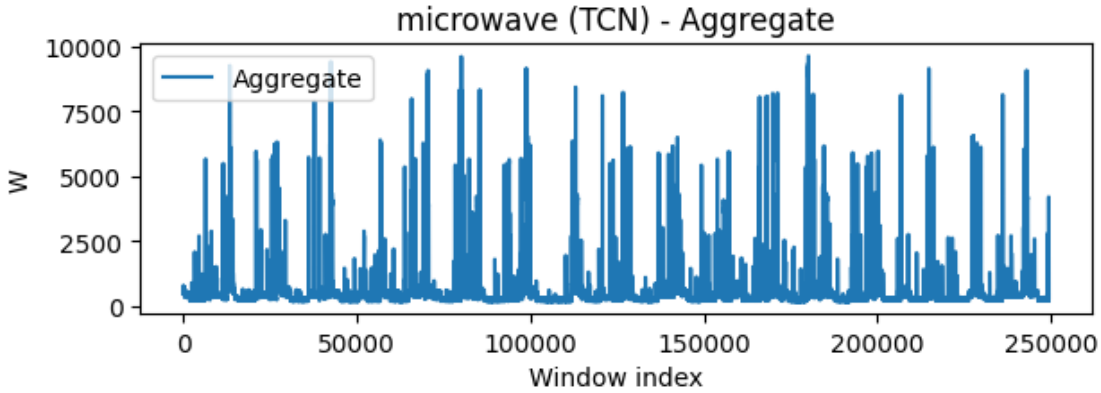
5.2.2 RISULTATI TCN Kettle

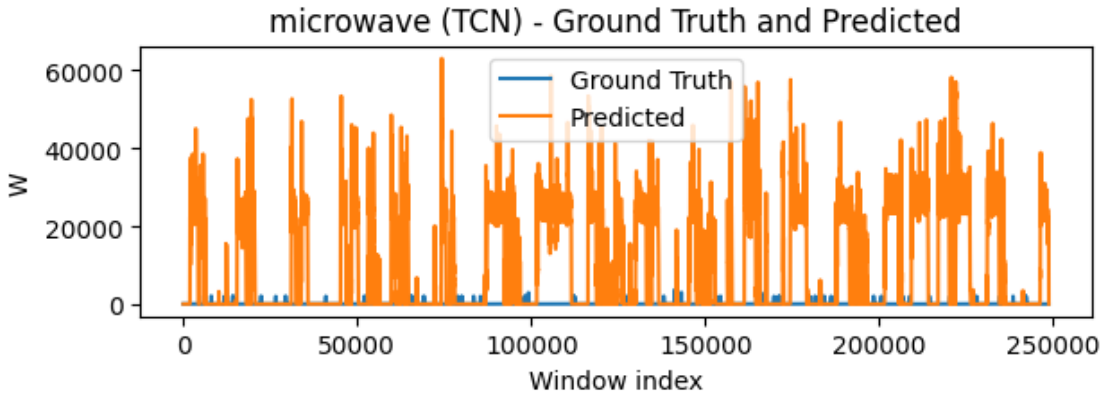


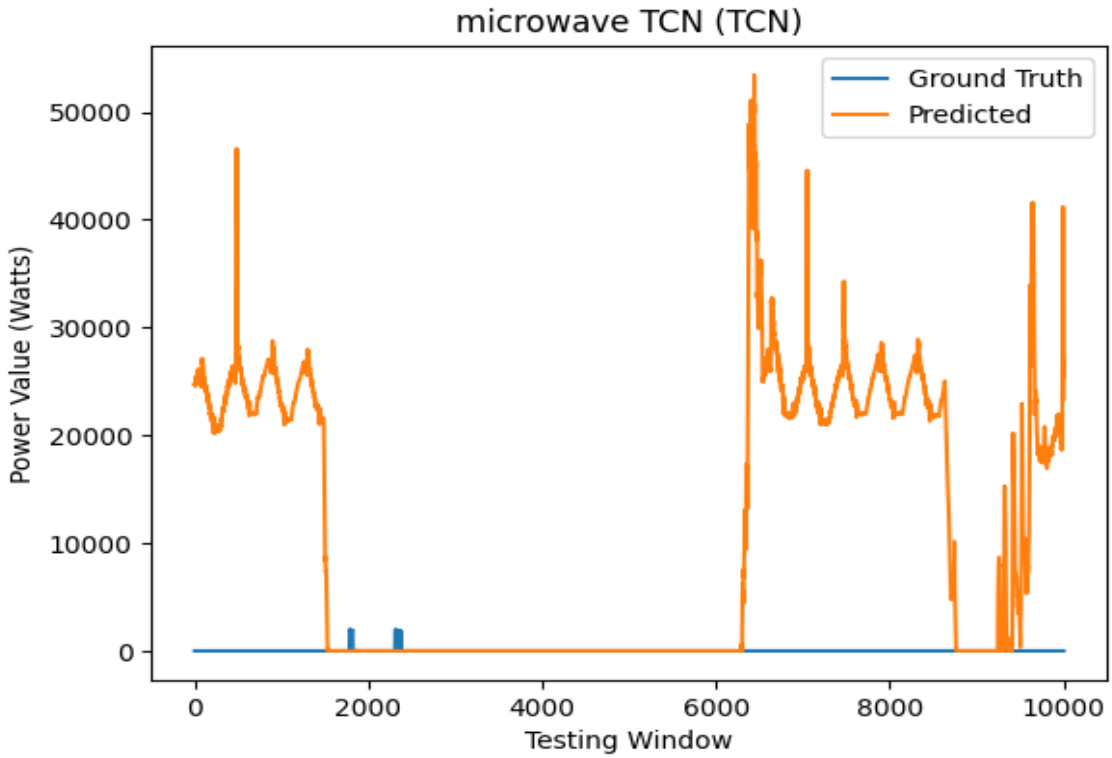




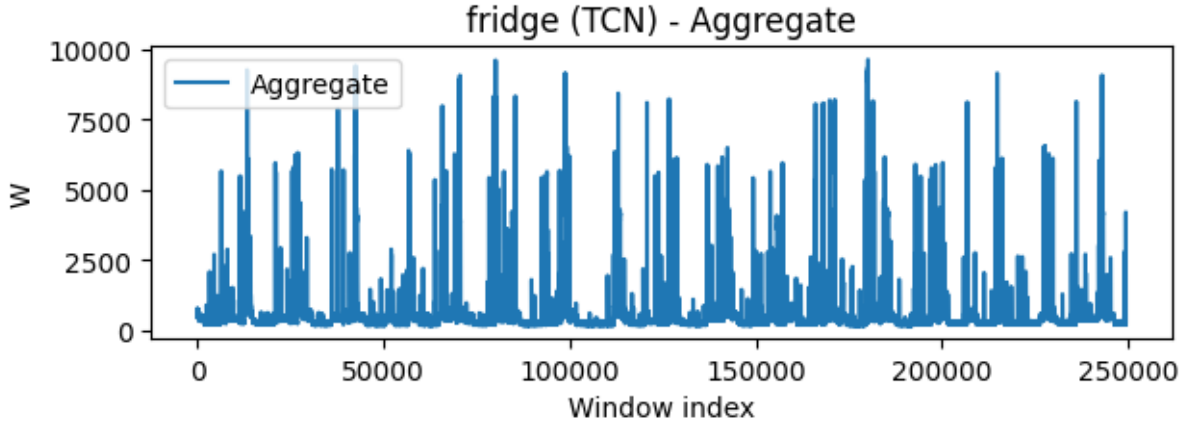
5.2.3 RISULTATI TCN Microwave

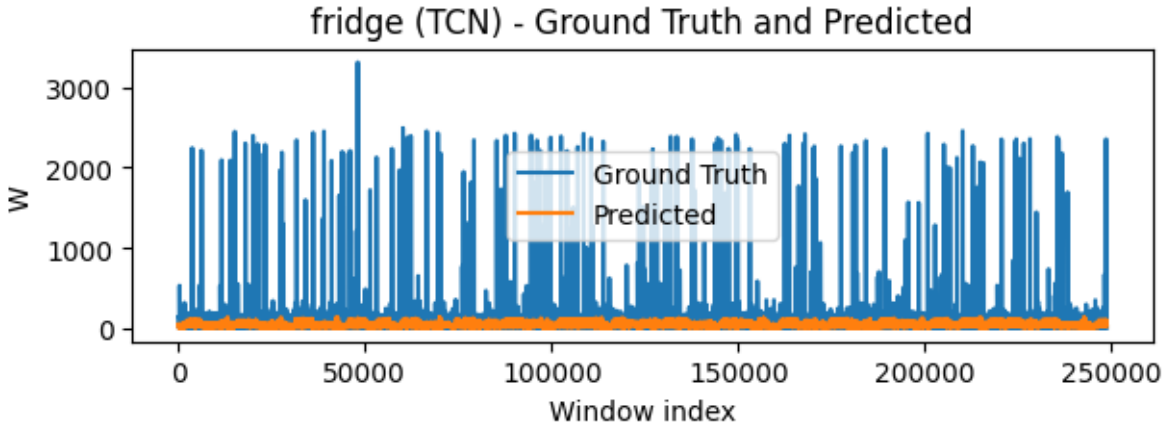


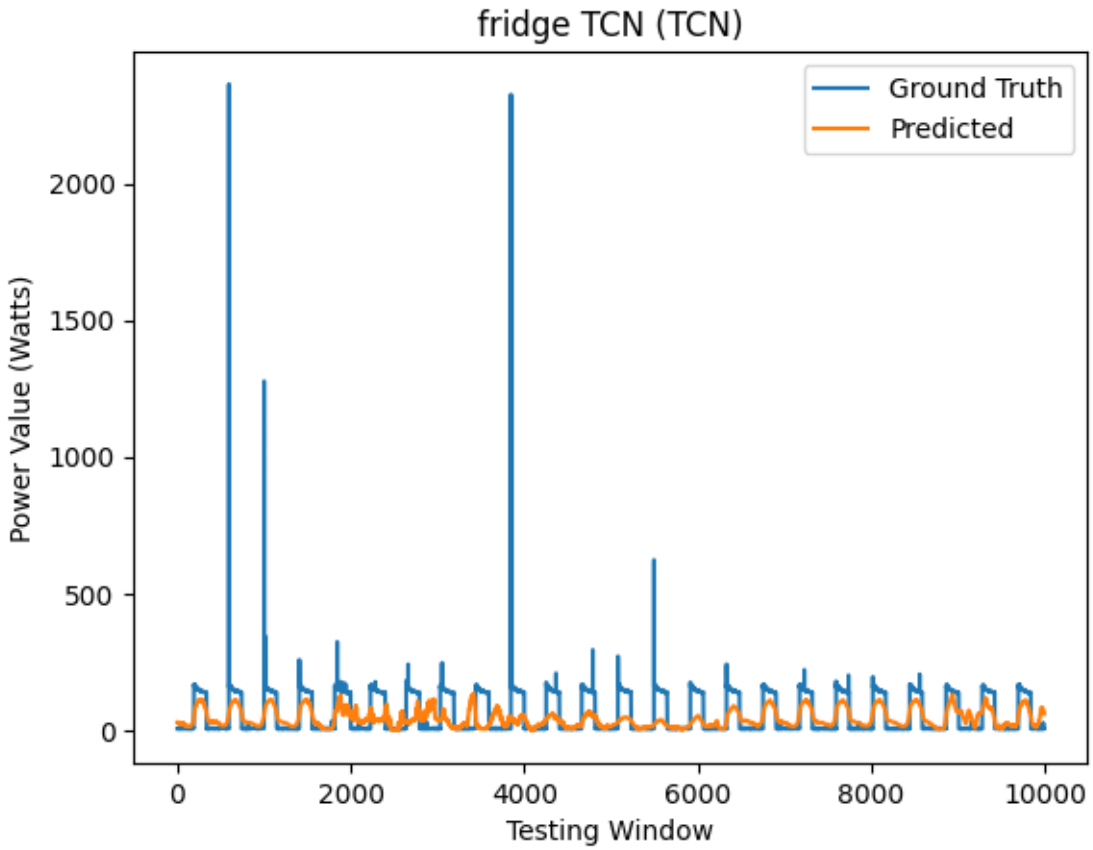




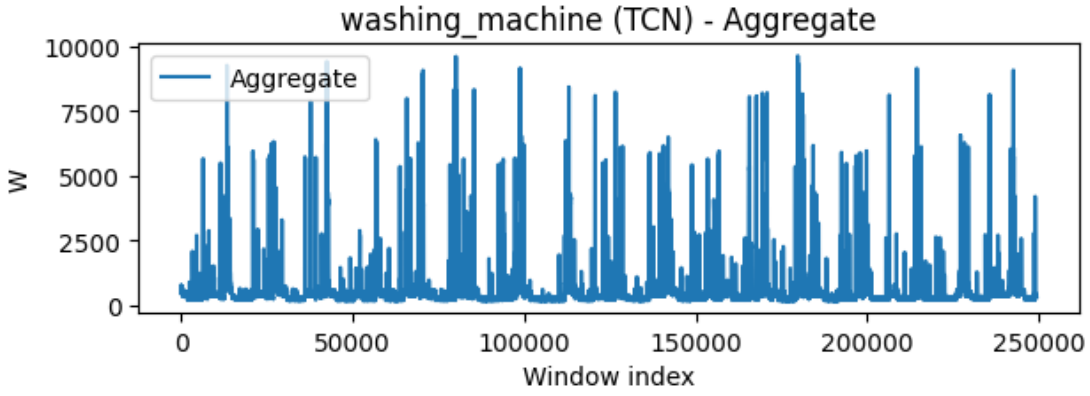
5.2.4 RISULTATI TCN Fridge

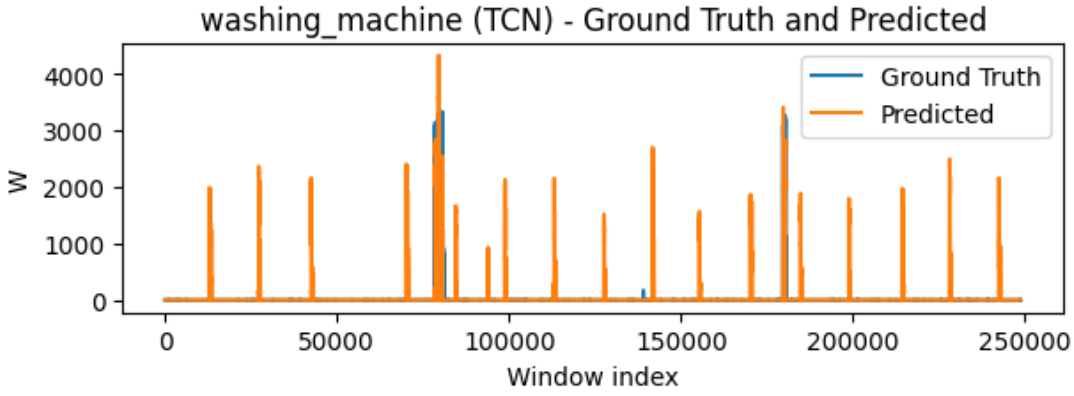


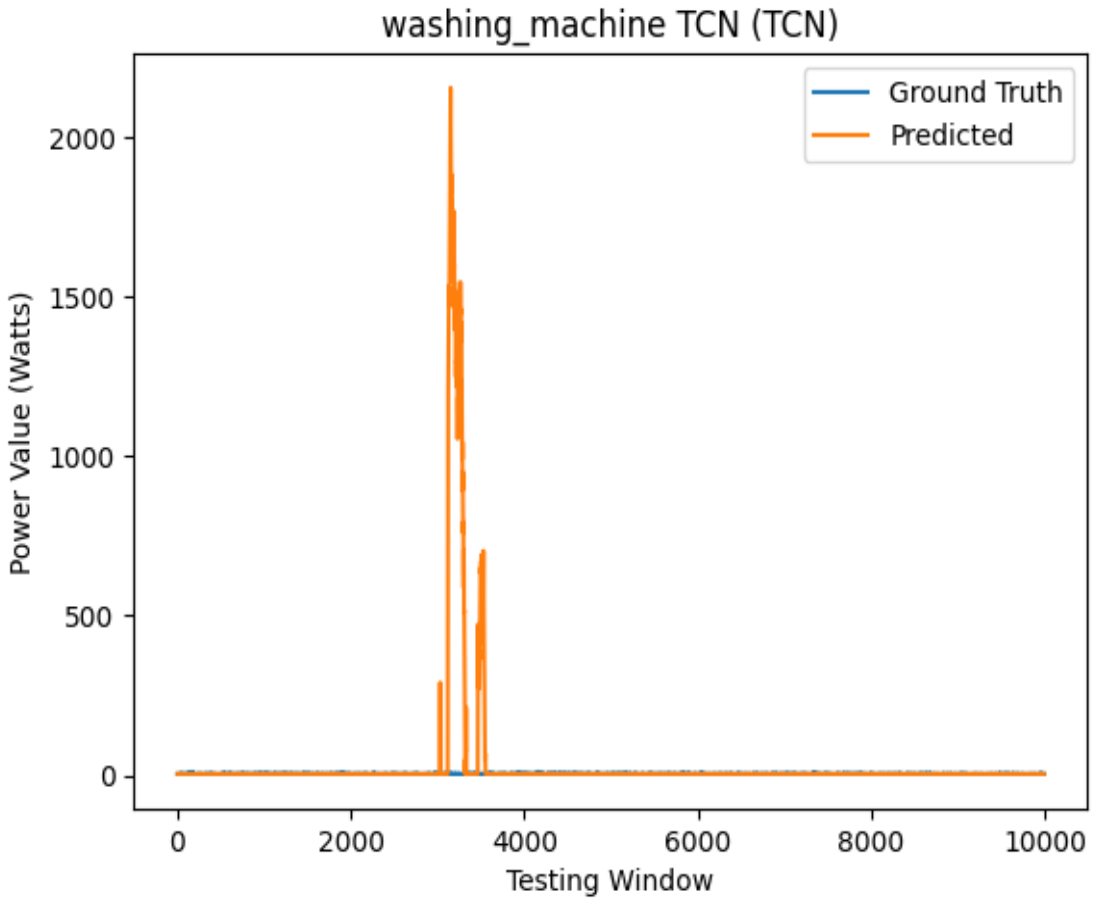




5.2.5 RISULTATI TCN WashingMachine

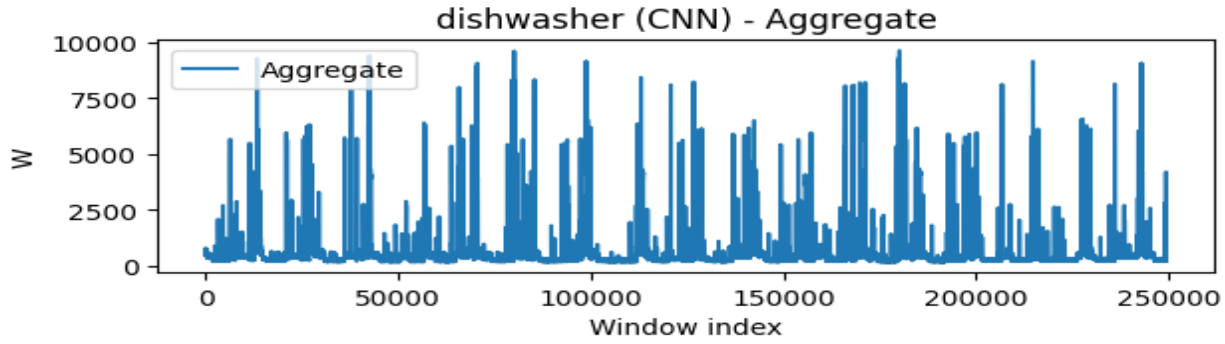


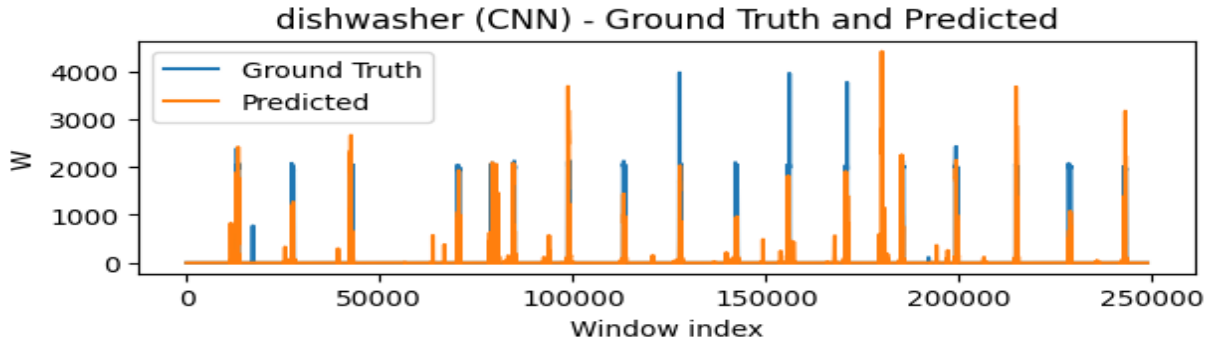


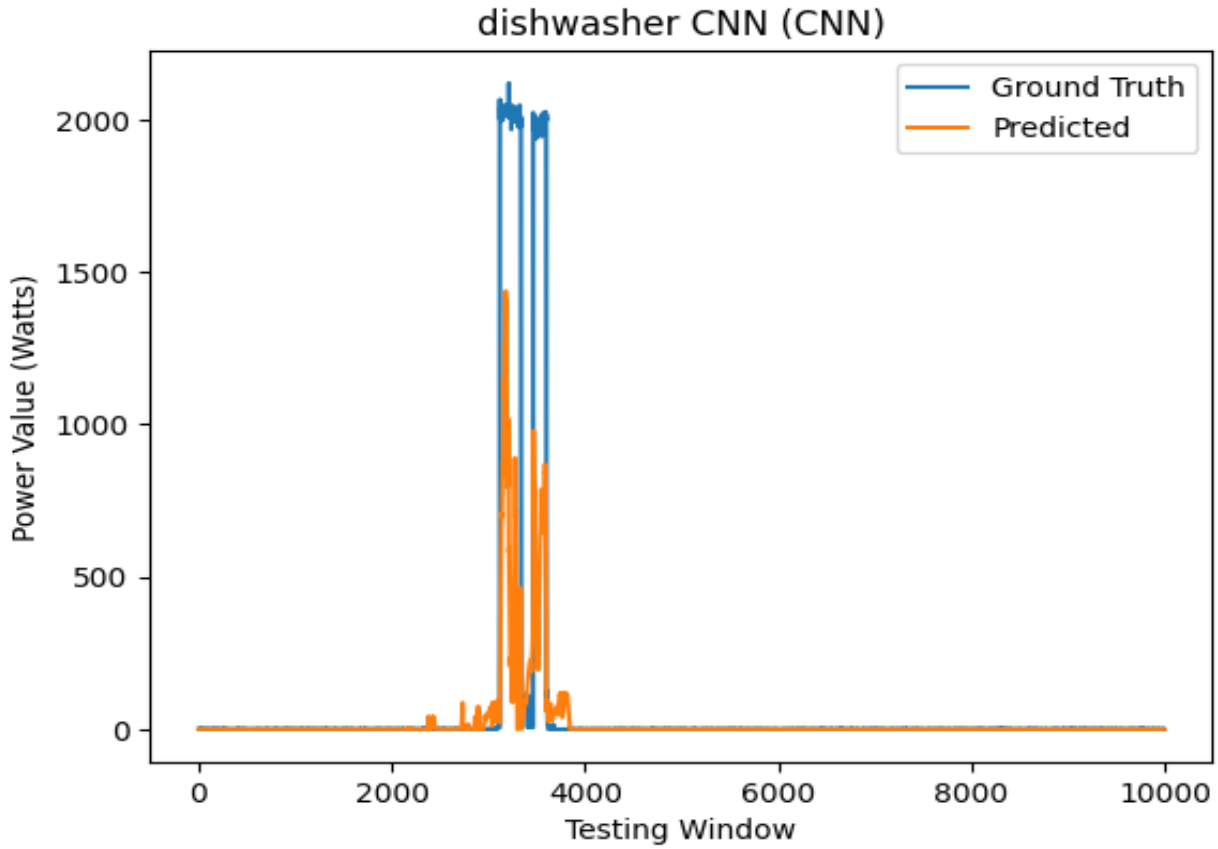


5.3 RISULTATI CNN

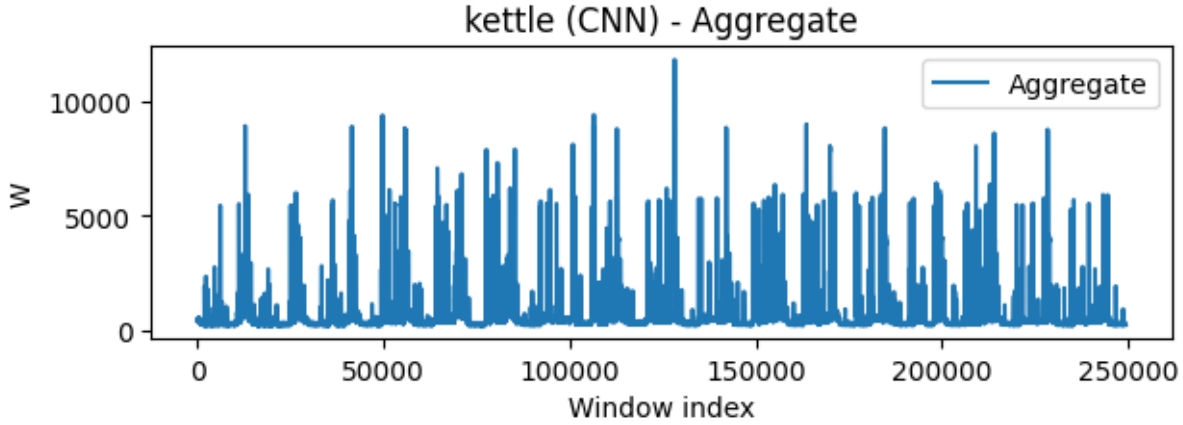
5.3.1 RISULTATI CNN Dishwasher

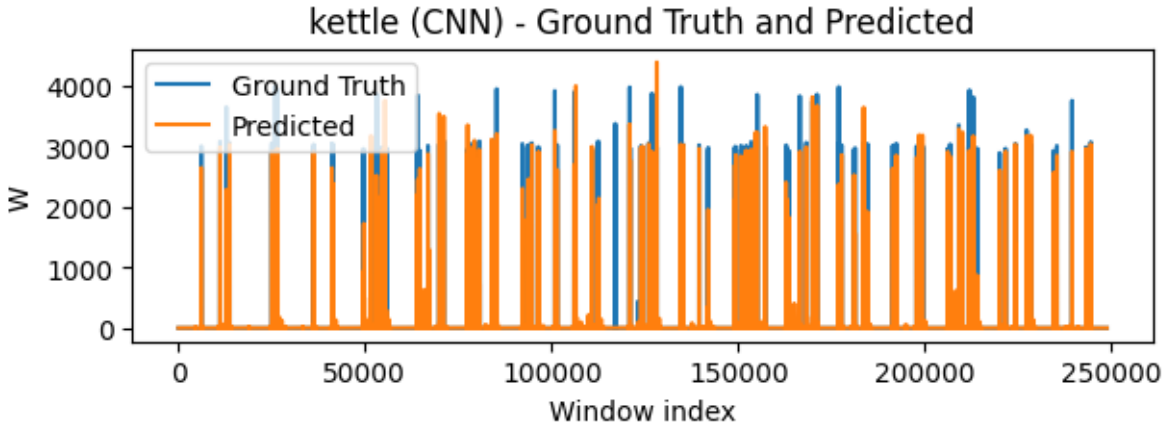


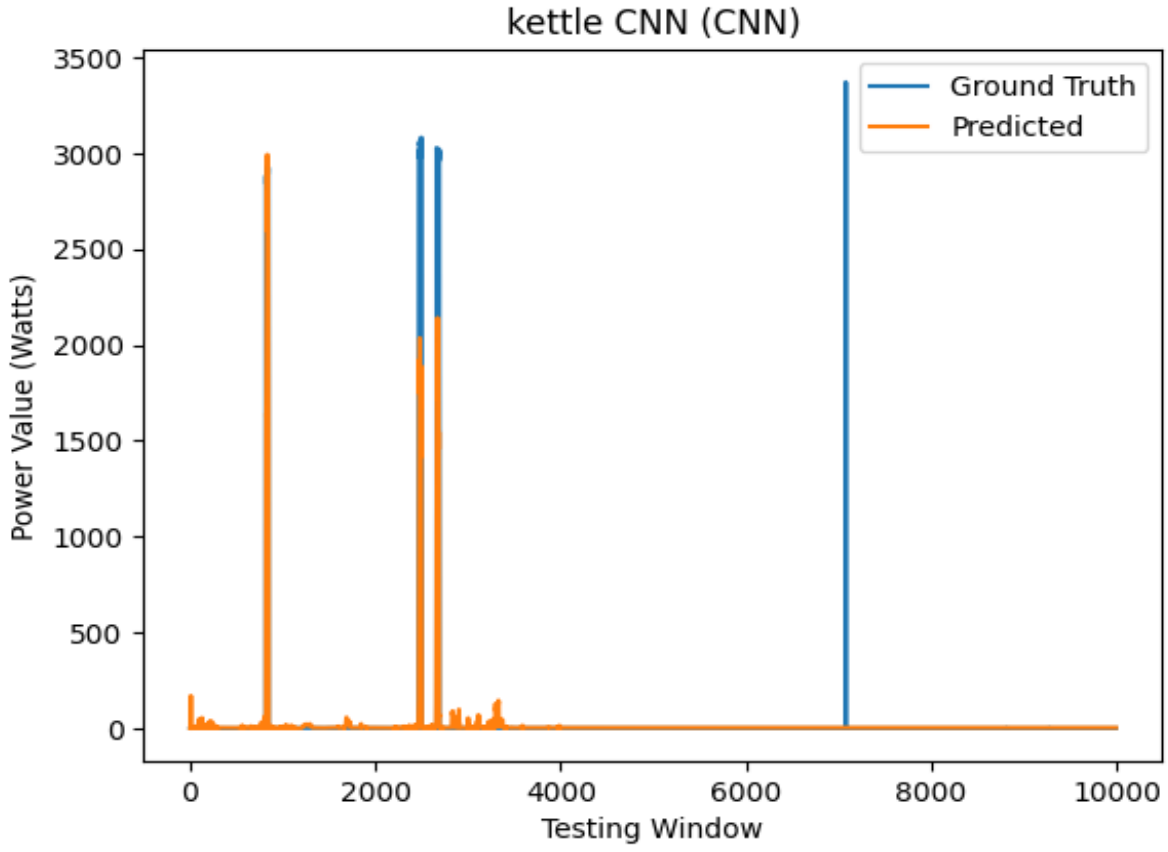




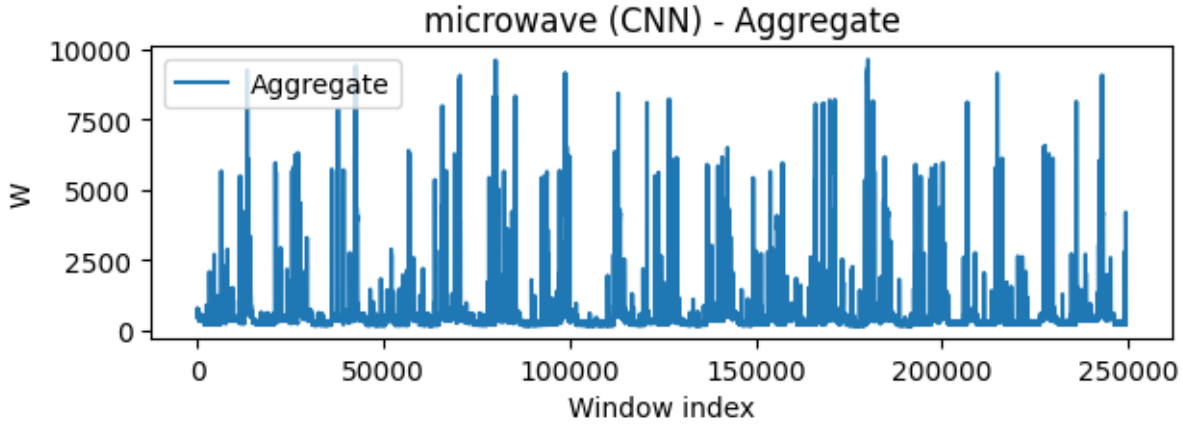
5.3.2 RISULTATI CNN Kettle

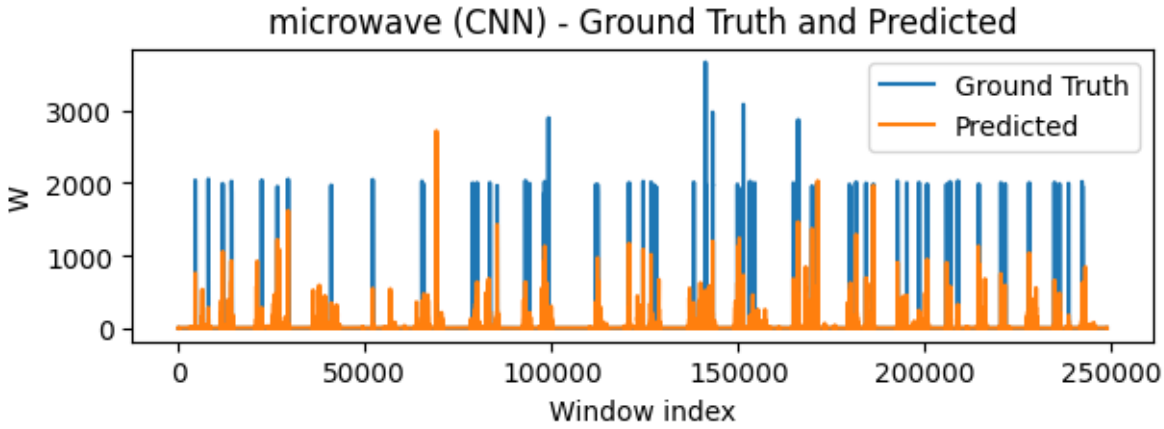


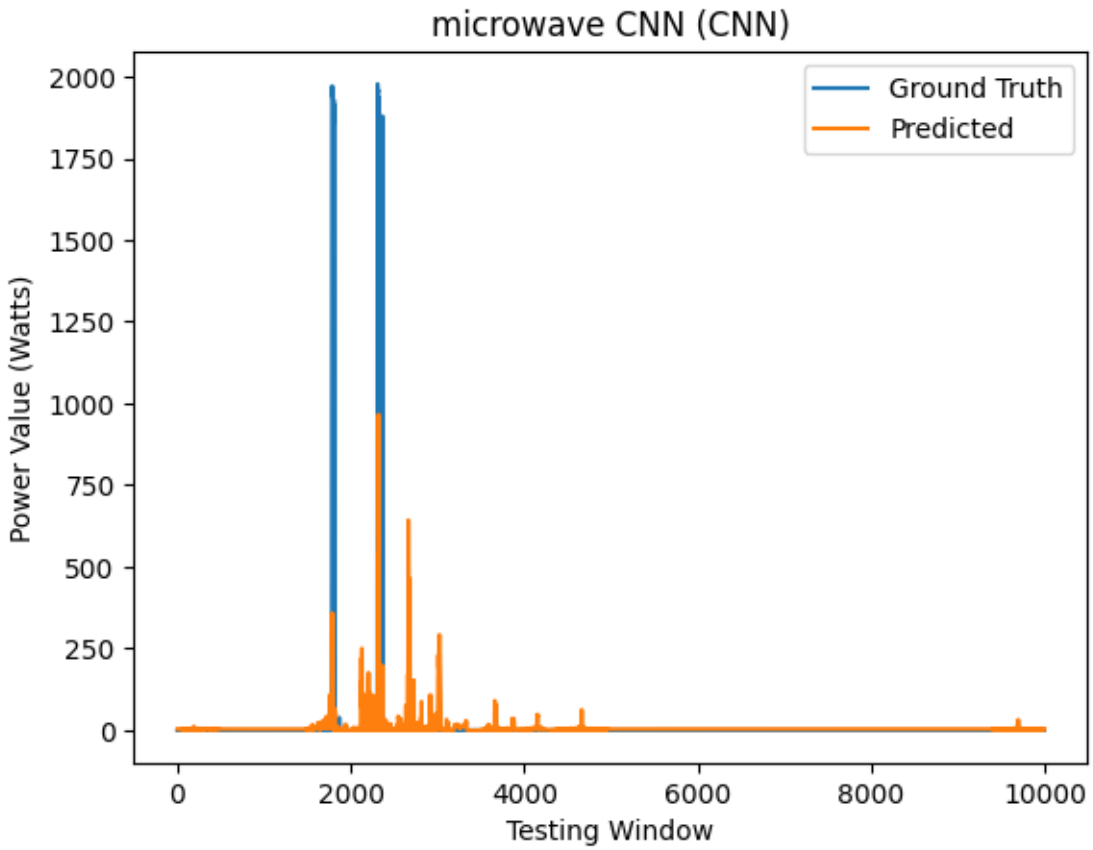




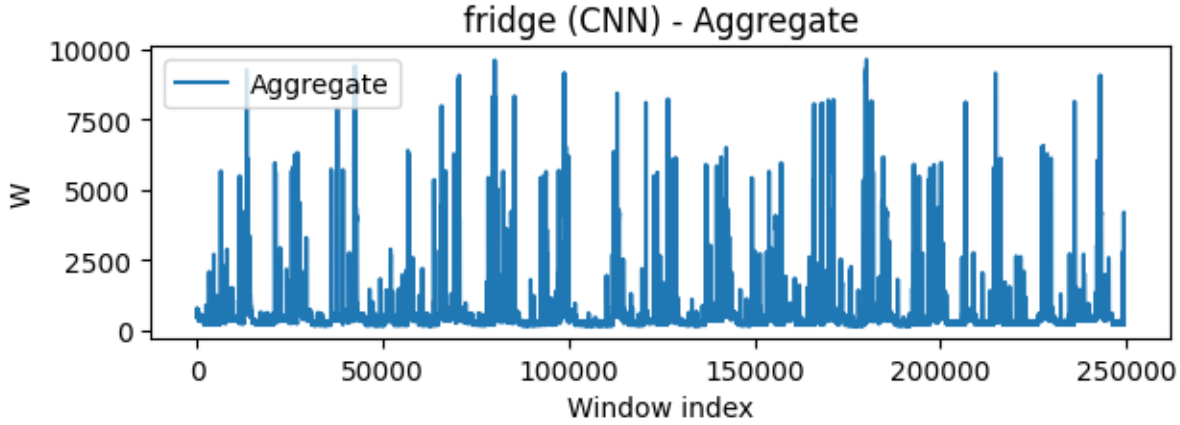
5.3.3 RISULTATI CNN Microwave

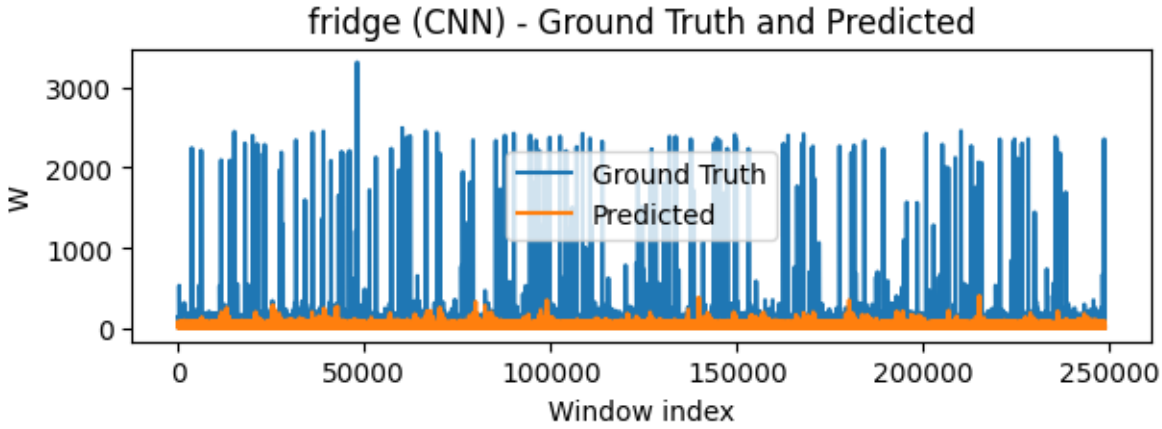


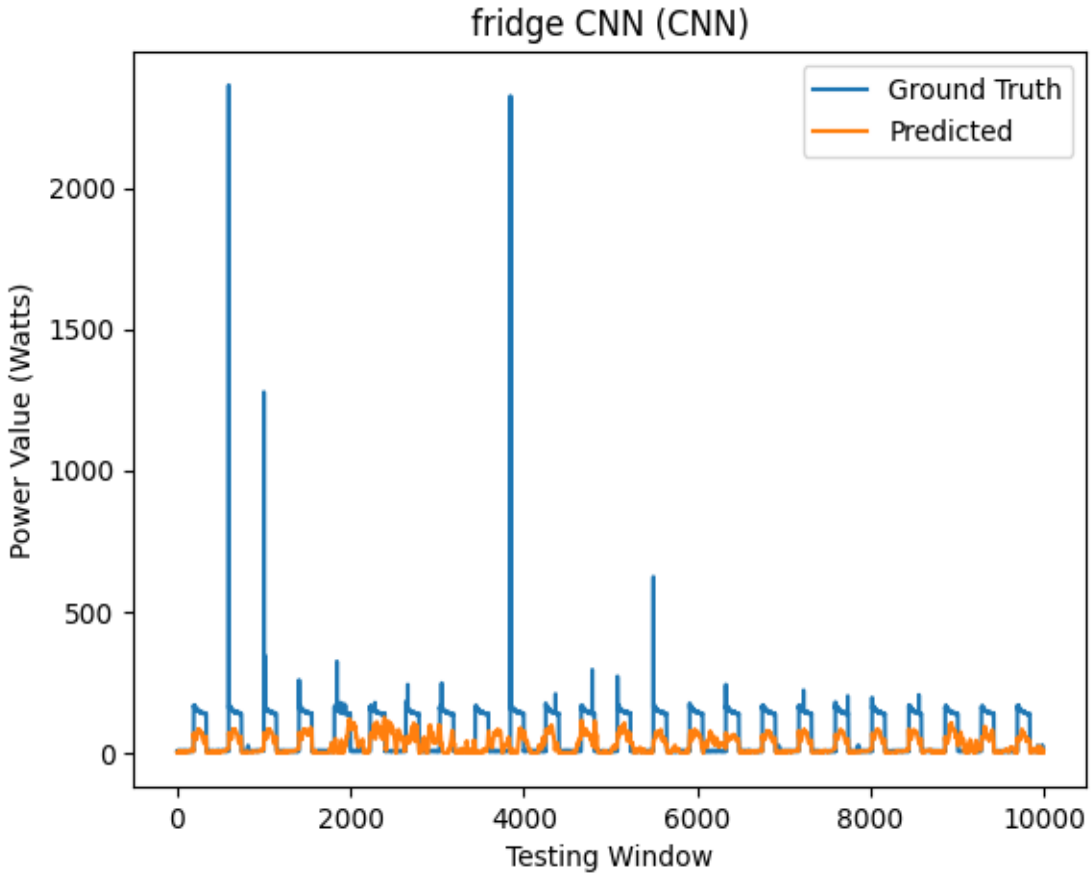




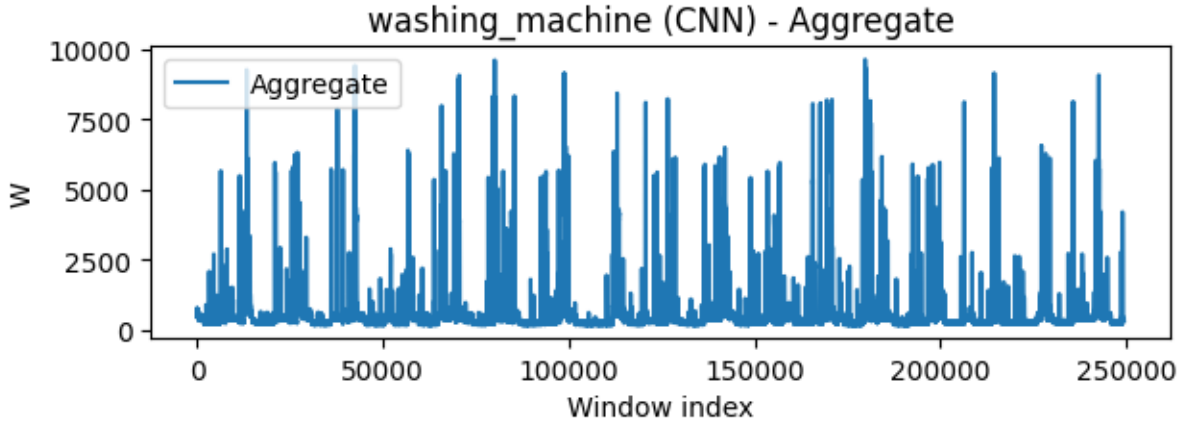
5.4.4 RISULTATI CNN Fridge

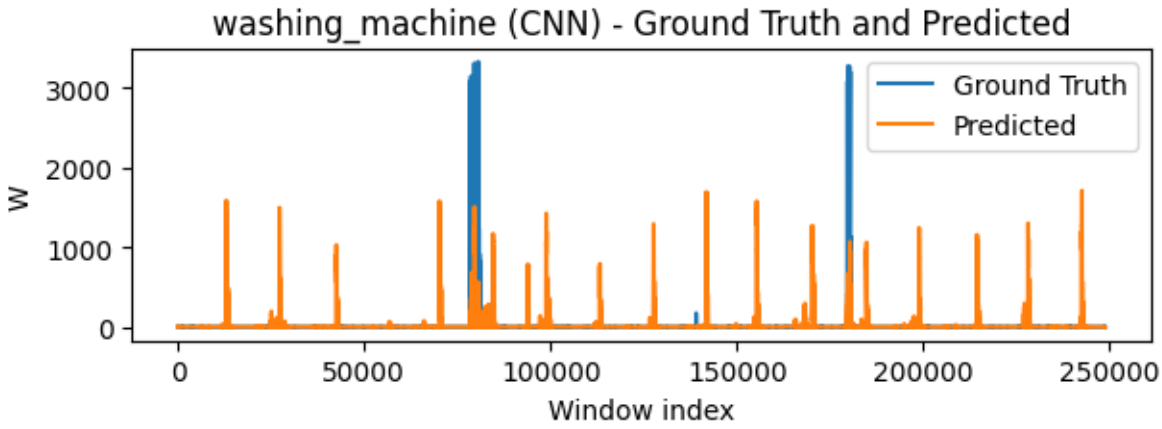


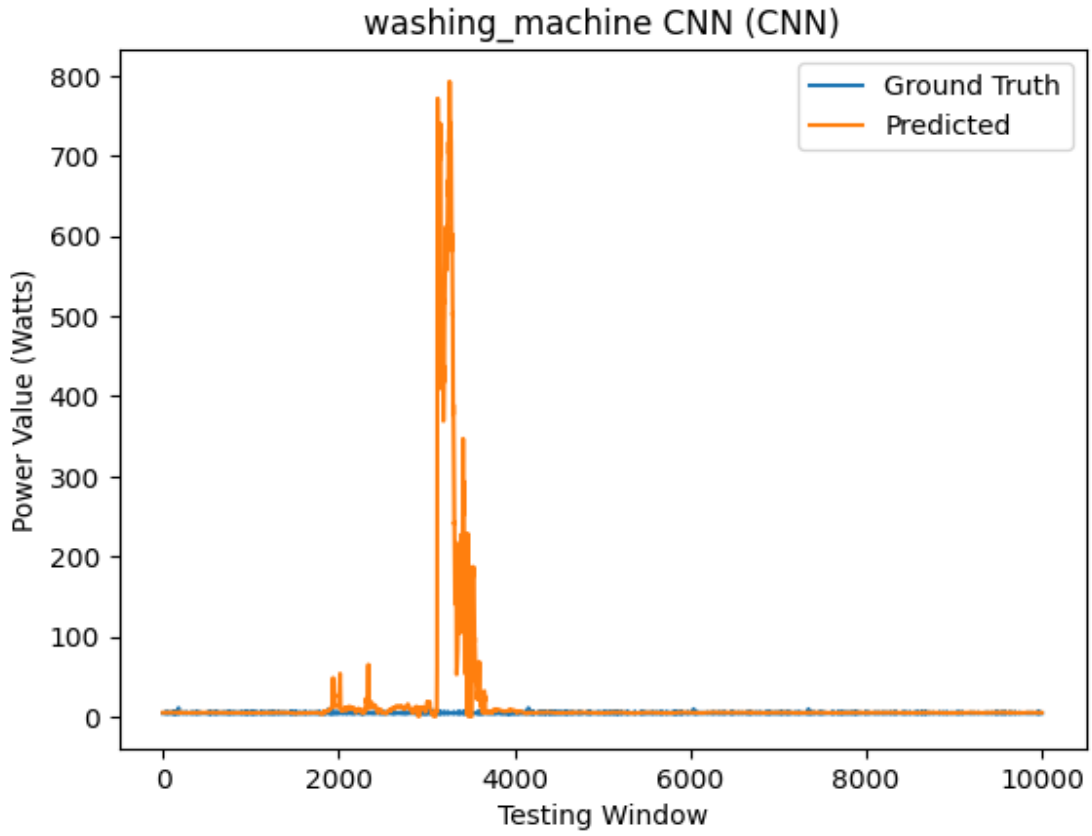




5.4.5 RISULTATI CNN WashingMachine







6 DISCUSSIONE

6.1 RIFLESSIONI E COMMENTO DEI RISULTATI

Vediamo come le architetture e le caratteristiche differenti della TCN e CNN ci conducono alle conclusioni che verranno presentate in seguito. Analizziamo le differenze sostanziali tra le due architetture: la prima fondamentale è la progettazione, infatti, la CNN sono progettare principalmente per elaborazioni spaziali (es. immagini) e non per elaborazioni per catturare elaborazioni al lungo termine in serie temporali grazie all'utilizzo di convoluzioni casuali e dilatate che consentono di lavorare con grandi range di dati evitando problemi di computazione. Le CNN al contrario sono più limitare per serie a lungo raggio e all'uso del pooling. Un'altra caratteristica da non sottovalutare è in particolare l'efficienza nel parallelizzare i dati di immagini e segmenti dati ma tendono a richiedere tecniche aggiuntive per gestire la dipendenza temporale al contrario le TCN hanno una forte stabilità temporale grazie alla loro struttura convoluzione. I risultati migliori sono stati ottenuti in TCN rispetto a CNN questo potrebbe essere dovuta al fatto che la TCN è più adatta per elaborazione di dati sequenziali proprio come quelli che sono stati utilizzati (potenza al varia del tempo), inoltre quando è necessario catturare .I risultati migliori sono stati ottenuti in TCN rispetto a CNN questo potrebbe essere dovuta al fatto che la TCN è più adatta per elaborazione di dati sequenziali proprio come quelli che sono stati utilizzati (potenza al varia del tempo), inoltre quando è necessario catturare dipendenze a lungo raggio nei dati temporali le TCN sono più efficaci grazie alla loro capacità di modellare queste relazioni senza perdere il contesto storico. Le TCN rispetto alle RNN tendono a ridurre l'overfitting e spesso sono in grado di superare i termini di prestazioni le CNN quando il compito da svolgere richiede una comprensione dettagliata delle relazioni temporali.

6.2PROBLEMI RISCONTRATI

La prova eseguita non è stata proprio quella di verificare e processare la rete SAMnet, infatti, non è stata usata quell'architettura ma una rete CNN e una rete TCN distinte. La rete SAMnet è stata implementata e fatta lavorare sia in training che in test ma senza ottenere risultati accettabili, la rete combina le due strutture in un'unica architettura molto potente in termini computazionali. Le cause dei valori non pertinenti possono derivare da un cattivo preprocessing dei dati e/o ad una non corretta implementazione dalla rete. In particolare, nel preprocessing alcune particolari indicazioni presenti nell'articolo SAMnet non sono state prese a considerazione per non chiarezza e ciò può far venire meno prestazioni ottimali della rete. Inoltre, nel pre-processing sono state verificate:

1. La stessa lunghezza e campionatura dei segnali di input e di output (ground-dtruth e aggregate) per verificare ciò si sono elaborati diversi grafici nei quali vi sono rappresentate le attivazioni dei vari appliance mettendo a confronto i due segnali.

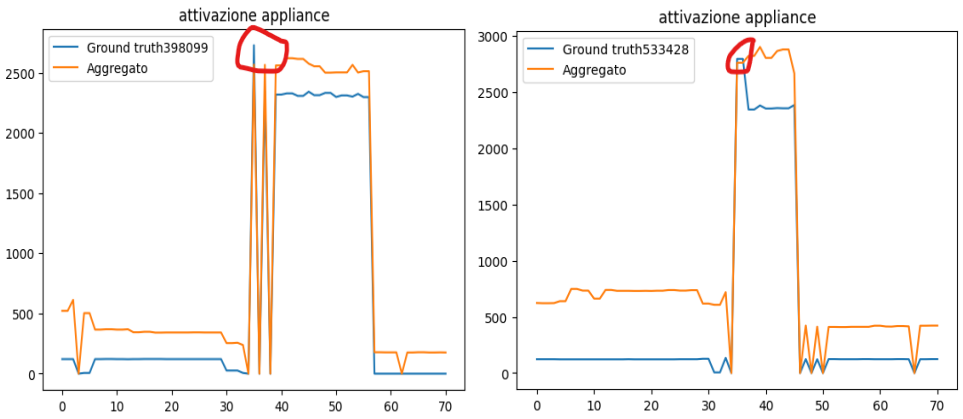


Figura 11 Ground truth e aggregate a confronto nel caso del dishwasher

2. La normalizzazione, si sono effettuate diverse prove modificando il valore di normalizzazione prima utilizzando il $\max \chi$ (massimo di aggregate) e successivamente il $max\Upsilon$ (massimo di ground truth). Il primo caso è risultato migliore.

Riguardo la non corretta implementazione dell'architettura le cause possono sempre essere dovute alla non chiarezza del paper consultato e quindi informazioni non attendibili nelle grandezze dei filtri presenti nei vari singoli layers e nell'implementazione dei singoli blocchi. Nonostante le varie modifiche e i numerosi accorgimenti nel realizzare l'architettura non sono stati ottenuti dei test ragionevoli per poter effettivamente fare un commento sulle prestazioni della rete stessa, infatti, i valori ottenuti sono randomici e l'uscita del test non ha un aspetto minimamente simile a quello predetto (ground truth). Considerando che, vista la non accettabilità dei dati e lavorando con una rete seqtoseq non si è potuto effettuare un ricombinamento dell'uscita al fine di ottenere il singolo campione che rappresenta realmente l'uscita. In questo caso i campioni in ingresso sono 600 e muovendosi con uno step piccolo rispetto alla lunghezza della finestra è opportuno effettuare un ricombinamento del risultato, in modo tale da considerare la ricombinazione delle parti sovrapposte delle finestre come dato da prendere.

7 CONCLUSIONI

In conclusione, la prova, anche se differente rispetto a quella prevista, ha ottenuto, a livello teorico, un risultato considerevole; infatti, si può dire che ogni rete ha il suo comportamento e la sua adattabilità a seconda dello scopo da raggiungere. È importante conoscere l'ambiente dove si sta lavorando ed il tipo di dato da gestire in modo tale da poter scegliere la rete più opportuna per avere meno costi computazionali, e quindi, anche economici, e ottenere le migliori prestazioni.

BIBLIOGRAFIA

[1] *Yinyan Liu, Jing Qiu: SAMNet: Toward Latency-Free Non-Intrusive Load Monitoring via Multi-Task Deep Learning IEEE TRANSACTIONS ON SMART GRID, VOL. 13, NO. 3, MAY 2022*

[2] *Schirmer PA, Mporas I. Non-Intrusive Load Monitoring: A Review. IEEE Transactions on Smart Grids. 2023 Jan 30;14(1):769-784. Epub 2022 Jul 8. doi: 10.1109/TSG.2022.3189598*

[3] *Heaton, Jeffrey. (2017). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618. Genetic Programming and Evolvable Machines. 19. doi: 10.1007/s10710-017-9314-z.*

[4] *Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia (2017). "Attention is All you Need". Advances in Neural Information Processing Systems. 30. Curran Associates, Inc.*

[5] *Shaojie Bai and J. Zico Kolter and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. 2018, arXiv, cs.LG; https://doi.org/10.48550/arXiv.1803.0127.*

[6] *Aaron van den Oord and Sander Dieleman and Heiga Zen and Karen Simonyan and Oriol Vinyals and Alex Graves and Nal Kalchbrenner and Andrew Senior and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. 2016, arXiv, cs.SD. https://doi.org/10.48550/arXiv.1609.03499*

[7] *C. Lea, M. D. Flynn, R. Vidal, A. Reiter and G. D. Hager, "Temporal Convolutional Networks for Action Segmentation and Detection", 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1003-1012, doi: 10.1109/CVPR.2017.113.*