UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Scienze Economiche e Statistiche

Corso di Laurea Triennale in Statistica per i Big Data

Tesi in *Architetture per i Big Data*

Forecasting come ricostruzione di immagini: un approccio basato sull'architettura U-Net

Relatore:
Ch.mo Prof. Roberto Tagliaferri

Candidato:
Rosario Pio Gnazzo
matr. 0212801002

Anno Accademico 2023/2024

La statistica è la grammatica della scienza.
- K. Pearson

Indice

Elenco delle tabelle

Elenco delle figure

Abstract

L'obiettivo di questa tesi è proporre un nuovo approccio al forecasting, trasformando il problema in un compito di ricostruzione di immagini attraverso l'uso di tecniche di deep learning. In particolare, viene impiegata l'architettura U-Net, originariamente progettata per la segmentazione di immagini mediche, per analizzare e prevedere serie temporali convertite in matrici bidimensionali. Le serie storiche sono trasformate in immagini utilizzando tecniche matematiche come il Gramian Angular Summation Field (GASF) e il Gramian Angular Difference Field (GADF), rendendo possibile l'uso di modelli di deep learning specifici per la ricostruzione di immagini.

Sono stati implementati tre modelli U-Net ciascuno specifico per una delle diverse tecniche di trasformazione dati, con l'obiettivo poi di confrontarne le prestazioni e identificare quello più adatto per questa tipologia di dati. Successivamente, il modello migliore è stato confrontato con un modello statistico autoregressivo, tipicamente utilizzato per problemi di forecasting, per valutare vantaggi e limitazioni dell'approccio basato su deep learning rispetto a una soluzione più tradizionale.

I risultati dimostrano che il modello U-Net basato su GASF è il più performante tra i modelli U-Net in termini di errori assoluti (MSE e MAE). Tuttavia, nel confronto con il modello autoregressivo, il GASF si distingue per la qualità dell'immagine (PSNR e SSIM), mentre l'AR offre una maggiore precisione nella previsione dei valori numerici. Questo confronto evidenzia che la scelta del modello dipende dagli obiettivi: il GASF è preferibile per la qualità visiva, mentre l'AR è più efficace nel minimizzare gli errori di previsione. Dato l'obiettivo di ottenere previsioni accurate, con i dati e l'approccio utilizzati, è ancora preferibile l'uso di modelli statistici specifici per il forecasting.

Infine, la tesi propone possibili sviluppi futuri per migliorare l'accuratezza dei modelli di deep learning, come l'ampliamento del dataset, la riduzione della finestra di previsione e l'adozione di architetture più avanzate come diffusion models e transformers, per potenziare ulteriormente le prestazioni di previsione.

Introduzione

Con il termine forecasting, o previsione, si fa riferimento al processo di stima di valori futuri basandosi su dati storici. In ambito finanziario, il forecasting è molto utilizzato per prevedere l'andamento degli indici di mercato, o dei prezzi delle azioni o di altri strumenti finanziari.

Il task della previsione, soprattutto in contesti molto complessi, presenta diverse problematiche in quanto è caratterizzato da *dinamiche non lineari* e *comportamenti stocastici* che causano variazioni imprevedibili nei dati. Proprio per questi motivi, per poter essere efficace deve possedere alcune caratteristiche chiave:

1) *Accuratezza*, le previsioni devono essere il più vicino possibili al valore realmente osservato;

2) *Robustezza*, i modelli devono essere consistenti alle anomalie nei dati;

3) *Generalizzabilità*, è fondamentale che i modelli possano generalizzare bene su nuovi dati non visti durante la fase di addestramento.

Questa tesi si propone di affrontare il problema del forecasting adottando un approccio che più informatico, utilizzando tecniche di deep learning (*Conditional Time Series Forecasting with Convolutional Neural Networks, 2018*[1]). Verranno adottati dati che inizialmente sono rappresentati in formato di serie storiche, successivamente verranno trasformati in matrici bidimensionali mantenendo però intatta la relazione temporale fra essi (*Deep learning and time series-to-image encoding for financial forecasting, 2020*[2]). Questa trasformazione permette non solo di conservare la sequenza temporale fra i dati, ma anche di poter estrarre informazioni aggiuntive sulle relazioni tra i diversi punti della serie. Con questa tecnica, i dati possono essere rappresentati come immagini, rendendoli così adatti all'utilizzo con architetture software di deep learning specifiche per il processamento di immagini. Il modello di deep learning che utilizzeremo per l'analisi e la modellazione di questa tesi sarà la *U-Net* (*U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015*[3], *Deep learning for time series classification, 2019*[4], *U-Time: A Fully Convolutional Network for Time Series Segmentation Applied to Sleep Staging, 2019*[5]). Originariamente, questa architettura è stata sviluppata per compiti di segmentazione di immagini mediche, in questo progetto invece l'obiettivo è quello di utilizzare la capacità della U-Net di estrarre caratteristiche comuni da immagini, in modo tale da renderla potenzialmente efficace anche nell'analisi di dati temporali in formato matriciale. L'idea è quella che la U-Net sia capace

di *identificare pattern temporali significativi che possano essere utilizzati per generare previsioni successive a partire da un input dato*. Per poter raggiungere questo obiettivo, è stato generato un dataset utilizzando processi stocastici mirati a simulare l'andamento di serie storiche finanziarie. In particolare, ogni riga del dataset corrisponde ad una serie storica. Tali serie sono state successivamente suddivise in partizioni, potendo gestire la possibilità di avere o meno overlapping tra di esse. Su ciascuna partizione verranno applicate trasformazioni matematiche, prima di tutto sono state utilizzate le coordinate polari per trasformare i dati e poter generare matrici Gramian Angular Summation Field (GASF), Gramian Angular Difference Field (GADF) e matrici di transizione di Markov (*Imaging Time-Series to Improve Classification and Imputation, 2015*[6] *Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks, 2015*[7]). Queste trasformazioni permettono di convertire le serie storiche in immagini, rendendo possibile l'utilizzo della U-Net. Come metro di paragone, abbiamo sviluppato un modello statistico autoregressivo con lo stesso compito di forecasting del modello di deep learning. Questo ci permette di poter confrontare i risultati ottenuti dai modelli U-Net, con quelli di un approccio più classico per problemi di questa natura, così da poter discutere dei vantaggi e svantaggi del suo utilizzo.

Ricapitolando quindi, nella seguente tesi verranno presentate le tecniche di generazione del dataset, le trasformazioni applicate per convertire le serie storiche in immagini, l'approccio utlizzato e i risultati ottenuti dai modelli di U-Net confrontandoli con quelli ottenuti da un modello autoregressivo. Verranno quindi discussi i vantaggi e le limitazioni dell'approccio proposto, nonché le possibili direzioni future per ulteriori miglioramenti.

1 Strumenti formali utilizzati

Nel seguente capitolo verranno approfonditi gli strumenti statistici, matematici e informatici che sono stati fondamentali per lo sviluppo del progetto di tesi in questione. Questa parte introduttiva è importante per contestualizzare le metodologie e le scelte tecniche adottate.

1.1 Dataset

Iniziamo dall'analizzare il set di dati che è stato utilizzato per questo progetto, illustrando come è stato costruito e che tipologia di dati raggruppa.

Definizione 1: *Dataset per classificazione con time series*

$$D = \{(s_i, l_j), \quad i, j = 1, \cdots, n\} \tag{1.1}$$

Set di n righe, ciascuna identifica una serie storica s_i a cui è associata un etichetta l_j.

Definizione 2: Con il termine *serie storica* si fa riferimento all'insieme di osservazioni di un fenomeno in un dato intervallo di tempo. Supponiamo di avere un fenomeno di interesse X e che questo sia osservato da t_0 a t_N con una cadenza temporale di Δ_i. Tutte le osservazioni di X rilevate nell'intervallo $(t_0, t_N]$ avvengono rispettivamente ai tempi

$$t_1 = t_0 + \Delta_1, \quad t_2 = t_0 + \Delta_1 + \Delta_2 \quad \cdots \quad t_N = t_0 + \Delta_1 + \Delta_2 + \cdots + \Delta_N \tag{1.2}$$

che è equivalente ad

$$t_1 = t_0 + \Delta_1, \quad t_2 = t_1\Delta_2 \quad \cdots \quad t_N = t_{N-1} + \Delta_N \tag{1.3}$$

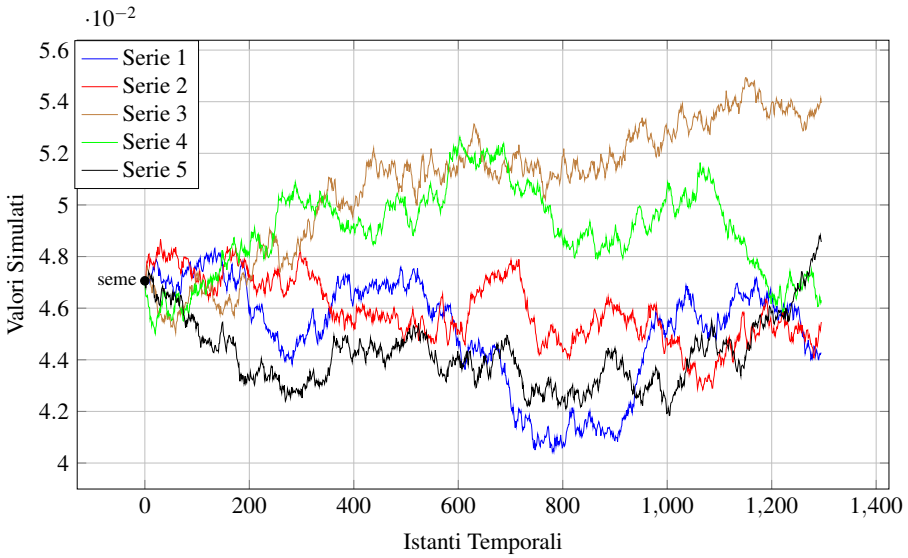
in modo che X assume nell'intervallo $(t_0, t_N]$ i seguenti valori

$$(x_{t_1}, x_{t_2}, \cdots, x_{t_N}) \tag{1.4}$$

tale vettore di osservazioni è definito serie storica di N elementi osservata sul fenomeno X nell'intervallo di tempo $(t_0, t_N]$.

1.1.1 Generazione delle time series

Figura 1.1: *Grafico delle prime 5 Time Series generate*



Per la generazione delle time series abbiamo adoperato come processo stocastico il modello di Vasicek (*An equilibrium characterization of the term structure, 1977*[8]), utilizzato per modellare i tassi di interesse. Per poter generare le simulazioni abbiamo utilizzato l'ultimo valore di una serie storica di un tasso di interesse reale, il titolo "CBOE Interest Rate 10 Year T No (TNX)", il quale si riferisce a un indice pubblicato dalla Chicago Board Options Exchange (CBOE) calcolato come una media ponderata dei 10-year Treasury note (Titoli che hanno tutti la scadenza iniziale a 10 anni). Presi i valori degli ultimi 10 anni, estratti da Yahoo Finance, al modello passiamo il valore finale in data 26 aprile 2024, con parametri non calibrati così da simulare serie storiche controllate. Dopo tale procedimento, otteniamo N time series che evolvono secondo un modello Vasicek partendo da valore iniziale reale r_0. Osserviamo analiticamente il processo.

Definizione 3: (*Processo di Vasiceck*)

$$\delta r_t = k(\theta - r_t)\delta t + \sigma \delta W_t \tag{1.5}$$

- k : velocità di ritorno di r_t verso il livello θ;
- θ : media di lungo termine di r_t;
- δt : è il passo temporale;
- σ : volatilità istantanea;
- δW_t : incremento infinitesimo del processo di Wiener.

Tramite discretizzazione di Eulero ha come soluzione

$$r_{t+\delta t} = r_t + k(\theta - r_t)\delta t + \sigma\sqrt{\delta t}\cdot\varepsilon \qquad \varepsilon \sim N(0,1) \tag{1.6}$$

Con questa soluzione possiamo generare una successione di dati simulati partendo da un *seme*, come possiamo vedere dalla figura [1.1.1].

1.1.2 Etichettamento delle time series

Per il processo di labeling delle serie storiche è stato scelta la seguente condizione

$$\begin{cases} (r_k - r_0) > 0, & 1 \\ (r_k - r_0) \leq 0, & 0 \end{cases} \tag{1.7}$$

l'etichetta sarà impostata ad 1 se l'ultimo valore simulato della serie storica è maggiore del valore reale usato come seme della simulazione. Viceversa, l'etichettà varrà 0.
Il dataset utilizzato nel progetto, costruito grazie alle procedure di generazione delle time series prima descritte, avrà dunque la seguente struttura

Serie	$\mathbf{r_0}$	$\mathbf{r_1}$	$\cdots$	$\mathbf{r_{k-1}}$	$\mathbf{r_k}$	**Label**
serie_1	dato reale	dato simulato$_{1,1}$	$\cdots$	dato simulato$_{1,k-1}$	dato simulato$_{1,k}$	0/1
serie_2	dato reale	dato simulato$_{2,1}$	$\cdots$	dato simulato$_{2,k-1}$	dato simulato$_{2,k}$	0/1
serie_3	dato reale	dato simulato$_{3,1}$	$\cdots$	dato simulato$_{3,k-1}$	dato simulato$_{3,k}$	0/1
serie_4	dato reale	dato simulato$_{4,1}$	$\cdots$	dato simulato$_{4,k-1}$	dato simulato$_{4,k}$	0/1
serie_5	dato reale	dato simulato$_{5,1}$	$\cdots$	dato simulato$_{5,k-1}$	dato simulato$_{5,k}$	0/1
serie_6	dato reale	dato simulato$_{6,1}$	$\cdots$	dato simulato$_{6,k-1}$	dato simulato$_{6,k}$	0/1
serie_7	dato reale	dato simulato$_{7,1}$	$\cdots$	dato simulato$_{7,k-1}$	dato simulato$_{7,k}$	0/1
serie_8	dato reale	dato simulato$_{8,1}$	$\cdots$	dato simulato$_{8,k-1}$	dato simulato$_{8,k}$	0/1
serie_9	dato reale	dato simulato$_{9,1}$	$\cdots$	dato simulato$_{9,k-1}$	dato simulato$_{9,k}$	0/1
serie_10	dato reale	dato simulato$_{10,1}$	$\cdots$	dato simulato$_{10,k-1}$	dato simulato$_{10,k}$	0/1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
serie_n	dato reale	dato simulato$_{n,1}$	$\cdots$	dato simulato$_{n,k-1}$	dato simulato$_{n,k}$	0/1

Tabella 1.1: *Struttura del dataset iniziale*

1.2 Tecniche di trasformazione dei dati

In questa sezione, analizzeremo le trasformazioni matematico-statistiche utilizzate per passare dal formato in serie storiche dei dati ad un formato matriciale visualizzabile come immagine. Questa panoramica, serve per fornire una base teorica che permetterà di comprendere meglio le tecniche implementate e i risultati ottenuti.

1.2.1 Trasformazione in coordinate polari

La trasformazione in coordinate polari è il primo passo per convertire una serie temporale in una rappresentazione matriciale. In questa trasformazione, ogni punto della serie temporale viene convertito in coordinate angolari e radiali, convertendo ogni punto della serie temporale in un angolo (θ) e un raggio (r) dal punto centrale.

Definizione 3: (*Trasformazione in coordinate polari per serie storiche*)
Nel caso di una serie storica $X = \{x_1, x_2, \cdots, x_n\}$, prima di poter convertire da coordinate cartesiane (x, y) un valore osservato x_i in coordinate polari (r, θ), bisogna riscalare i dati fra $[-1, 1]$ oppure $[0, 1]$. Questo per via di come viene calcolato l'angolo per la trasformazione. Dunque, data una serie X questa viene riscalata in $\tilde{X}$ tramite

$$\tilde{x}_{-1} = \frac{(x_i - max(X)) + (x_i - min(X))}{max(X) - min(X)} \qquad \text{oppure} \qquad \tilde{x}_0 = \frac{x_i - min(X)}{max(X) - min(X)} \tag{1.8}$$

Con i nuovi dati riscalati $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n\}$ possiamo effettuare la trasformazione in coordinate polari attraverso le seguenti equazioni

$$r = \frac{t_i}{n}, \qquad t_i = 1, 2, \cdots, n \tag{1.9}$$

$$\theta = \arccos(\tilde{x}_i) \tag{1.10}$$

Questa normalizzazione rende i dati invariabili rispetto a cambiamenti di scala, permettendo di confrontare facilmente serie temporali con differenti range di valori, preservando le informazioni temporali. Una volta convertiti in coordinate polari, i dati angolari possono essere ulteriormente analizzati con tecniche specifiche per variabili angolari. Proprio per questo, tale trasformazione è posta prima dei metodi come la somma o la differenza degli angoli (GASF e GADF), così da creare immagini che rappresentano le relazioni temporali fra i dati.

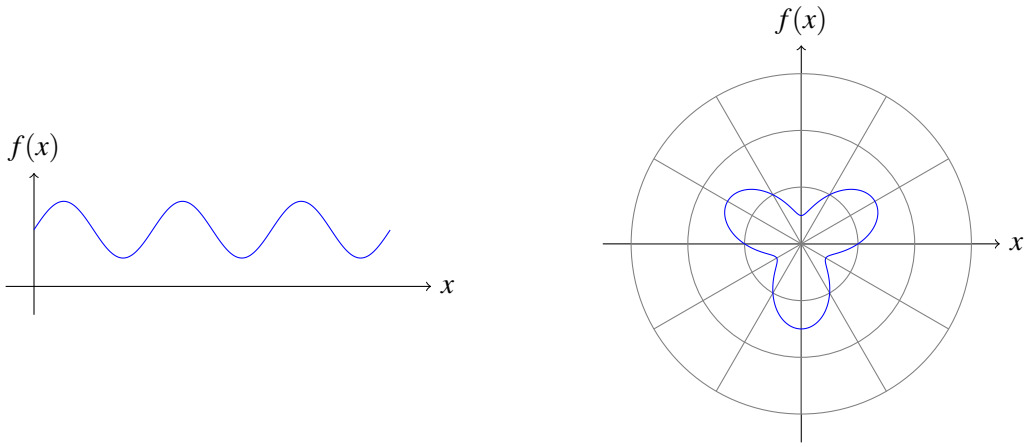


Figura 1.2: *Funzione* $f(x) = 1 + 0.5\sin(3x)$ *convertita in coordinate polari*

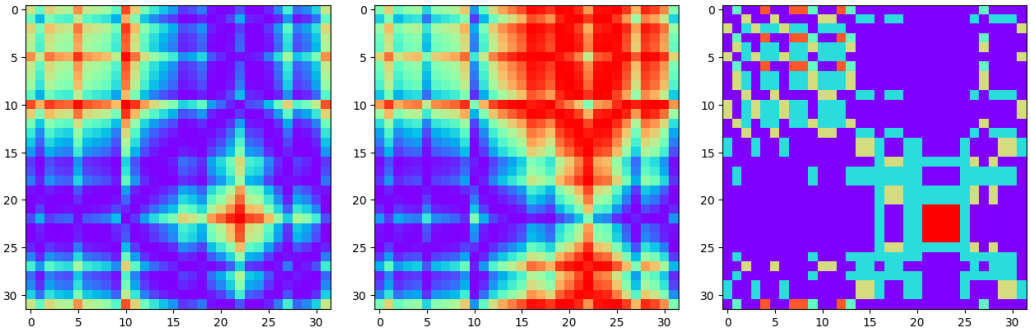


Figura 1.3: *Matrici GASF, GADF e Markov (stessa time series)*

1.2.2 Gramian Angular Summation Field (GASF)

Definizione 4: Il GASF è una tecnica di trasformazione dati che utilizza la somma degli angoli ottenuti dalla trasformazione in coordinate polari per rappresentare le relazioni temporali tra i punti della serie. Nello specifico, data una serie temporale riscalata $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n\}$ di n osservazioni, possiamo definirla come segue

$$GASF = [\cos(\theta_i + \theta_j)] \qquad \forall i, j = 1, 2, \cdots, n \tag{1.11}$$

$$= \begin{bmatrix} \cos(\theta_1+\theta_1) & \cos(\theta_1+\theta_2) & \dots & \cos(\theta_1+\theta_j) & \dots & \cos(\theta_1+\theta_n) \\ \cos(\theta_2+\theta_1) & \cos(\theta_2+\theta_2) & \dots & \cos(\theta_2+\theta_j) & \dots & \cos(\theta_2+\theta_n) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \cos(\theta_i+\theta_1) & \cos(\theta_i+\theta_2) & \dots & \cos(\theta_i+\theta_j) & \dots & \cos(\theta_i+\theta_n) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \cos(\theta_n+\theta_1) & \cos(\theta_n+\theta_2) & \dots & \cos(\theta_n+\theta_j) & \dots & \cos(\theta_n+\theta_n) \end{bmatrix} \tag{1.12}$$

Che corrisponde

$$GASF = \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} \tag{1.13}$$

$$= \begin{bmatrix} \langle \tilde{x}_1, \tilde{x}_1 \rangle & \langle \tilde{x}_1, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_1, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_1, \tilde{x}_n \rangle \\ \langle \tilde{x}_2, \tilde{x}_1 \rangle & \langle \tilde{x}_2, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_2, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_2, \tilde{x}_n \rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \langle \tilde{x}_i, \tilde{x}_1 \rangle & \langle \tilde{x}_i, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_i, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_i, \tilde{x}_n \rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \langle \tilde{x}_n, \tilde{x}_1 \rangle & \langle \tilde{x}_n, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_n, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_n, \tilde{x}_n \rangle \end{bmatrix} \tag{1.14}$$

definendo il prodotto interno come $\langle x_i, x_j \rangle = \tilde{x}_i \cdot \tilde{x}_j - \sqrt{1 - \tilde{x}_i^2} \cdot \sqrt{1 - \tilde{x}_j^2} \qquad \forall i, j = 1, 2, \cdots, n$

1.2.3 Gramian Angular Difference Field (GADF)

Definizione 5: Il GADF, simile al GASF, converte la serie temporale in una matrice di immagini, ma utilizza la differenza degli angoli ottenuti dalla trasformazione in coordinate polari. Questa tecnica evidenzia le variazioni locali e le differenze tra i punti della serie temporale. Data una serie temporale riscalata $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n\}$ di n osservazioni, possiamo definirla come segue

$$GADF = [\sin(\theta_i - \theta_j)] \qquad \forall i, j = 1, 2, \cdots, n \tag{1.15}$$

$$= \begin{bmatrix} \sin(\theta_1-\theta_1) & \sin(\theta_1-\theta_2) & \dots & \sin(\theta_1-\theta_j) & \dots & \sin(\theta_1-\theta_n) \\ \sin(\theta_2-\theta_1) & \sin(\theta_2-\theta_2) & \dots & \sin(\theta_2-\theta_j) & \dots & \sin(\theta_2-\theta_n) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \sin(\theta_i-\theta_1) & \sin(\theta_i-\theta_2) & \dots & \sin(\theta_i-\theta_j) & \dots & \sin(\theta_i-\theta_n) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \sin(\theta_n-\theta_1) & \sin(\theta_n-\theta_2) & \dots & \sin(\theta_n-\theta_j) & \dots & \sin(\theta_n-\theta_n) \end{bmatrix} \tag{1.16}$$

Che corrisponde

$$GADF = \sqrt{I - \tilde{X}^2}' \cdot \tilde{X} - \tilde{X}' \cdot \sqrt{I - \tilde{X}^2} \tag{1.17}$$

$$= \begin{bmatrix} \langle \tilde{x}_1, \tilde{x}_1 \rangle & \langle \tilde{x}_1, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_1, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_1, \tilde{x}_n \rangle \\ \langle \tilde{x}_2, \tilde{x}_1 \rangle & \langle \tilde{x}_2, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_2, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_2, \tilde{x}_n \rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \langle \tilde{x}_i, \tilde{x}_1 \rangle & \langle \tilde{x}_i, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_i, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_i, \tilde{x}_n \rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \langle \tilde{x}_n, \tilde{x}_1 \rangle & \langle \tilde{x}_n, \tilde{x}_2 \rangle & \cdots & \langle \tilde{x}_n, \tilde{x}_j \rangle & \cdots & \langle \tilde{x}_n, \tilde{x}_n \rangle \end{bmatrix} \tag{1.18}$$

definendo il prodotto interno come $\langle x_i, x_j \rangle = \sqrt{1 - \tilde{x}_i^2} \cdot \tilde{x}_j - \tilde{x}_i \cdot \sqrt{1 - \tilde{x}_j^2} \qquad \forall i, j = 1, 2, \cdots, n$

1.2.4 Matrici di Markov

Le matrici di transizione di Markov rappresentano le probabilità di transizione da uno stato all'altro nella serie temporale. Questa tecnica è utile per modellare le sequenze di eventi e preservare le informazioni sulle dinamiche temporali dei dati (*Markov Chains, 1997*[9]).

Definizione 6: (*Matrici di Markov per serie storica*)
Data una serie temporale X, identifichiamo i suoi Q quantili e assegniamo ogni x_i ai corrispondenti quantili q_j ($j \in [1, Q]$). Costruiamo una matrice di adiacenza pesata $Q \times Q = W$. I valori $w_{i,j}$ che la compongono, sono dati dalla frequenza con cui un punto nel quantile q_j è seguito da un punto nel quantile q_i, che corrisponde alla probabilità di passare da i a j in un intervallo di tempo, dunque possiamo scrivere $\Pr(j|i) = w_{i,j}$. La matrice di Markov W è

costruita utilizzando $P_{i,j}$ come elemento della i-esima riga e j-esima colonna.

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,j} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,j} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{i,1} & w_{i,2} & \dots & w_{i,j} & \dots & w_{i,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{\alpha,1} & w_{n,2} & \dots & w_{n,j} & \dots & w_{n,n} \end{bmatrix} \tag{1.19}$$

La somma delle probabilità di transizione da uno stato i a tutti gli altri stati deve essere 1

$$\sum_{j=1}^{n} w_{i,j} = 1 \tag{1.20}$$

1.2.5 Vantaggi

Il passaggio da time series ad immagini utilizzando varie trasformazioni come ad esempio in coordinate polari, Gramian Angular Summation Field (GASF), Gramian Angular Difference Field (GADF) e matrici di Markov offre numerosi vantaggi.

1) **Preservazione delle informazioni temporali**

– La trasformazione in coordinate polari, seguita da GASF e GADF, preserva le informazioni temporali della serie temporale poiché catturano le relazioni angolari e differenziali tra i punti della serie. Ciò permette di mantenere l'ordine temporale e le dipendenze tra i dati;

– Le matrici di transizione di Markov catturano le probabilità di transizione da uno stato all'altro, mantenendo così l'informazione sulle sequenze di eventi.

2) **Riduzione della dimensione e complessità dei dati**

Il passaggio da serie storica a matrice permette di poter applicare su quest'ultima tecniche di compressione dati che rende l'apprendimento automatico più efficiente sia in termini di tempo che di memoria.

3) **Rivelazione di pattern nascosti**

Queste trasformate possono evidenziare pattern e relazioni non evidenti nelle serie originali. Le matrici osservate come immagini potrebbero rilevare simmetrie, periodi e altre caratteristiche che possono essere cruciali.

4) **Robustezza agli errori e al rumore**

Le immagini prodotte possono essere meno sensibili agli errori puntuali o al rumore nella serie temporale originale, in quanto la trasformazione può attenuare le variazioni locali che non sono significative.

In sintesi, tali tecniche ci aiutano a migliorare significativamente sia la capacità di analisi, che di visualizzazione e di modellazione dei dati.

1.3 Modelli di Deep Learning

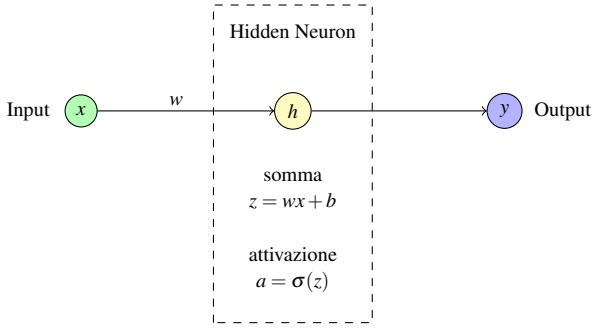
1.3.1 Cos'è il Deep Learning

Per **deep learning** - o apprendimento profondo - si fa riferimento ad una sotto-disciplina del machine learning - o apprendimento automatico - che si basa sull'uso di reti neurali artificiali con molteplici strati. Tali strati sono formati da tantissimi neuroni artificiali collegati fra loro, ognuno dei quali applica delle trasformazioni sui dati che riceve e li ritrasmette al neurone successivo.La sezione seguente, fornirà una panoramica introduttiva sul Deep Learning e sulle Reti Neurali, esplorando i concetti chiave che li strutturano (*I. Goodfellow: Deep Learning, 2016*[10]).

1.3.2 Struttura di una rete neurale

Gli strati di una rete neurale sono sostanzialmente di 3 tipologie: strati di input, strati nascosti e gli strati di output.

Figura 1.4: *Diagramma di un singolo neurone (hidden)*



Strati di Input

Gli *strati di input* sono il punto di partenza per l'elaborazione dei dati, in quanto sono utilizzati affinché i dati grezzi vengono introdotti nella rete. Infatti, ogni neurone in questo strato rappresenta una caratteristica del dato in ingresso, potrebbe essere un pixel di un'immagine o una misurazione in un dataset ecc. Per questo motivo, questi neuroni non eseguono trasformazioni complesse sui dati, ma hanno lo scopo di fornire le informazioni che saranno successivamente elaborate dagli strati successivi.

Strati Nascosti

Gli *strati nascosti*, o hidden layers, si trovano tra l'input e l'output, ed è il luogo dove avviene la gran parte del lavoro computazionale e di trasformazione dei dati. Ogni neurone in questo strato effettua sostanzialmente due trasformazioni sui dati, la somma pesata degli ingressi e l'applicazione di una funzione di attivazione sui dati.

Definizione 7: (*Somma pesata degli ingressi*)

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{1.21}$$

x_i : dati in ingresso;
w_i : pesi associati ai dati in ingresso x_i;
b : bias.

Il termine z rappresenta invece il risultato della somma pesata, dunque aggrega tutte le informazioni provenienti dagli ingressi, tenendo conto della loro importanza relativa determinata dai pesi e dell'aggiustamento fornito dal bias.

Definizione 8: (*Funzione di attivazione*)
La seconda operazione trasforma la somma pesata z in un'uscita a, ed è fondamentale per introdurre non linearità nel modello, permettendo alla rete di apprendere comportamenti complessi e variabili. La funzione di attivazione $\sigma(z)$ viene scelta in base al task da risolvere e può essere una funzione sigmoide, ReLU (Rectified Linear Unit), tanh, o altre funzioni non lineari, e viene applicata come segue:

$$a = \sigma(z) \tag{1.22}$$

Strati di Output

Infine, gli *strati di output* sono responsabili della produzione del risultato finale del modello che sia una previsione o una classificazione richiesta. La funzione degli strati di output è quindi quella di sintetizzare tutte le informazioni elaborate e fornire una risposta come classificare un'immagine o fare una previsione. Per capire come queste tre famiglie di neuroni interaggiscono tra di loro, osserviamo i legami che si formano tra un solo neurone nascosto in una rete neurale.

In una rete complessa esistono collegati insieme, milioni di neuroni come quelli in figura [1.3.2], disposti in layers successivi. Ad esempio, osserviamo la struttura di una semplice rete neurale con solo due layers di neuroni nascosti collegati tra loro.

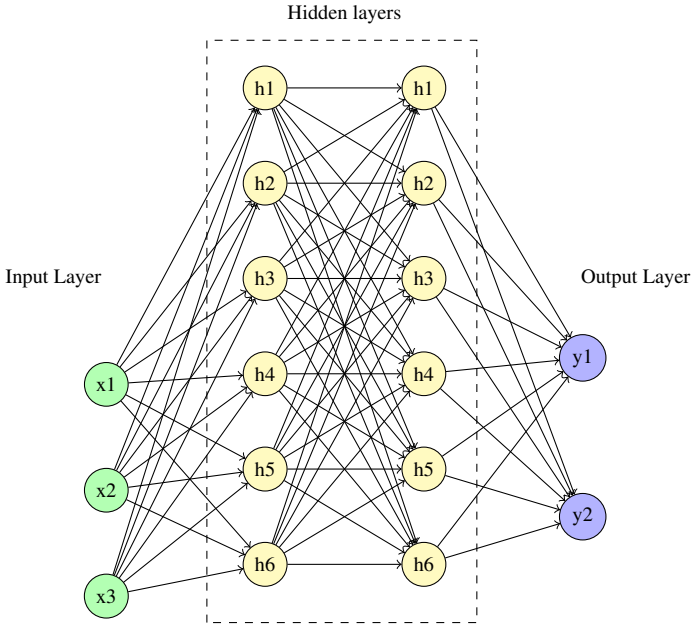


Figura 1.5: *Diagramma di una rete neurale con uno strato di input (3 neuroni), due strati nascosti (6 neuroni ciascuno) e due neuroni nello strato di output*

1.3.3 Funzionamento di una rete neurale

Abbiamo visto da cosa è formata una neural network, analizziamo ora nel dettaglio le fasi che caratterizzano il suo funzionamento, la **Forward Propagation** e la **Backpropagration** (*Learning representations by back-propagating errors, 1986*[11]).

Definizione 9: (*Forward Propagation*)
Durante questa fase, i dati di input vengono propagati attraverso la rete, strato dopo strato. Ogni neurone, come abbiamo visto nella sezione precedente, effettua una somma pesata dei suoi input e applica successivamente una funzione di attivazione per ottenere l'output. Formalmente, per un dato neurone j nel livello l, l'output $a_j^{(l)}$ è calcolato come:

$$z_j^{(l)} = \sum_i w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)} \tag{1.23}$$

$$a_j^{(l)} = \sigma(z_j^{(l)}) \tag{1.24}$$

dove:
$w_{ji}^{(l)}$ è il peso della connessione tra il neurone i nel livello $l-1$ e il neurone j nel livello l
$a_i^{(l-1)}$ è l'output del neurone i nel livello $l-1$
$b_j^{(l)}$ è il bias del neurone j nel livello l
σ è la funzione di attivazione

Questo processo continua strato dopo strato fino a raggiungere lo strato di output, dove

si ottiene la predizione finale.

Definizione 10: (*Backpropagation*)
Come fa la rete neurale a sapere quali pesi applicare ai dati?
Il cuore del processo di apprendimento di una rete neurale è la backpropagation. Questo algoritmo consente di aggiornare i pesi della rete in modo da minimizzando l'errore tra la predizione della rete e il valore reale. Il processo si svolge in tre fasi principali:

1) *Calcolo della Funzione di Perdita*
Dopo la fase di *forward propagation*, si calcola la funzione di perdita (o errore), che serve per misurare la differenza tra la predizione ottenuta e il valore atteso. Le funzioni di perdita comuni sono ad esempio l'errore quadratico medio (MSE) o l'errore mediano assoluto:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1.25}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{1.26}$$

2) *Propagazione dell'Errore*
L'errore calcolato usando la funzione di perdita viene propagato all'indietro attraverso la rete. Durante questa fase, si calcolano le derivate parziali dell'errore rispetto ai pesi della rete usando la regola della catena. Nello specifico, la derivata parziale dell'errore E rispetto ad un peso $w_{ji}^{(l)}$ è calcolata come:

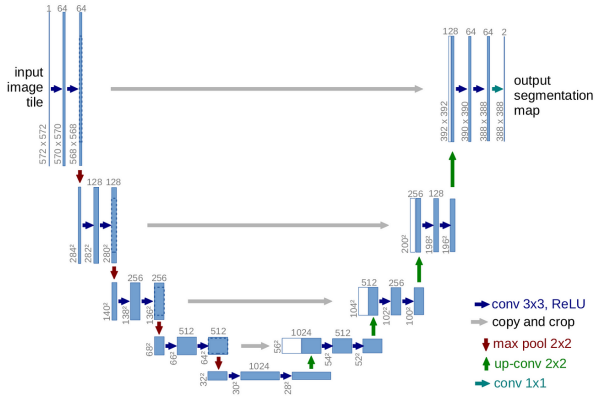
$$\frac{\partial E}{\partial w_{ji}^{(l)}} = \frac{\partial E}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}} \tag{1.27}$$

3) *Ottimizzazione*
Infine, i pesi vengono aggiornati usando un algoritmo di ottimizzazione, il più noto è il *Gradient Descent*, che lavora in modo tale che l'aggiornamento dei pesi sia proporzionale alla derivata parziale dell'errore rispetto a ciascun peso ma scalato da un fattore chiamato tasso di apprendimento (η).

$$w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} - \eta \frac{\partial E}{\partial w_{ji}^{(l)}} \tag{1.28}$$

Questa tecnica è fondamentale per minimizzare la funzione di perdita, infatti in questo modo, i pesi vengono aggiornati nella direzione opposta al gradiente della funzione di perdita. Ciò assicura che ad ogni passo, ci si muova verso un minimo della funzione di perdita.

Figura 1.6: *Architettura di una U-Net*



1.3.4 U-Net

La rete U-Net è una rete neurale convoluzionale sviluppata per la segmentazione di immagini biomediche. L'idea principale alla base della U-Net quando è stata sviluppata, era quella di trovare un modo per rendere la segmentazione più raffinata. Per fare ciò si è deciso di integrare ad una rete convoluzionale tradizionale con strati successivi, in cui le operazioni di pooling vengono sostituite da operazioni di upsampling, delle nuove connessioni chiamate **skip-connections**. Questi strati aumentano la risoluzione dell'output, permettendo ad uno strato convoluzionale successivo di apprendere a costruire un output più preciso grazie alle informazioni dello step corrispondente nella fase di Contrazione.

Struttura della U-Net

L'architettura della U-Net si compone principalmente di due parti: un percorso di contrazione (**encoder**) e un percorso di espansione (**decoder**).

Definizione 11: (*Percorso di Contrazione*)
Il percorso di encoder segue la classica architettura di una rete convoluzionale, composta da ripetute applicazioni di **convoluzioni** 3×3 che scorrendo l'immagine in input, applicano un filtro di dimensioni 3×3 producendo un'uscita che rappresenta le caratteristiche rilevate dal filtro. Ciascuna seguita da una funzione di attivazione e da un'operazione di **max pooling** 2×2 con passo 2 che riduce la dimensione spaziale dell'immagine di input prendendo il massimo valore in ogni finestra 2×2. Questo permette di ridurre la risoluzione spaziale dell'immagine preservando le caratteristiche più rilevanti. A ogni passaggio, il numero di canali di caratteristiche raddoppia, consentendo alla rete di apprendere rappresentazioni più complesse e di alto livello dei dati in input.

Definizione 12: (*Percorso di Espansione*)
Il percorso di decoder è costituito da una sequenza di **upsampling** dei dati, ossia aumenta la

dimensione spaziale dell'immagine di input. Esistono tanti metodi per effettuarlo, il più comune è l'interpolazione bilineare che per un'immagine di input X di dimensioni $H \times W \times C$, produce un'immagine di dimensioni $2H \times 2W \times C$. Dove ogni punto nell'output è una media ponderata dei punti circostanti nell'input. Queste fasi vengono seguite da convoluzioni 3×3 e da funzioni di attivazione.

In questo percorso, le informazioni ad alta risoluzione provenienti dal percorso di contrazione **vengono concatenate** con i dati upsampled attraverso le **skip-connections**, permettendo alla rete di ricostruire con precisione la segmentazione dell'immagine.

1.4 Modelli Autoregressivi

1.4.1 Cos'è un modello autoregressivo

Per *modello autoregressivo* ($AR(p)$) si intende un modello statistico utilizzato per descrivere un processo che cambia nel tempo. Nello specifico, i modelli autoregressivi sono un caso particolare del modello ARMA per serie storiche, ossia sono un modello lineare nel quale la variabile dipendente y, detta in "uscita", è funzione lineare dei valori delle uscite precedenti. In altre parole, tale modello stabilisce che il valore di una variabile in un dato istante temporale può essere rappresentata da una combinazione lineare dei suoi valori precedenti. I modelli autoregressivi vengono utilizzati principalmente per la modellazione di serie temporali in ambiti quali: finanza, economia, meteorologia, ingegneria e così via (*On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers, 1927*[12]).

Definizione 13: (*Modello autoregressivo di ordine p*)
A livello formale, i modelli autoregressivi sono descritti nel seguente modo

$$y_t = \beta_1 y_{t-1} + \beta_1 y_{t-2} + \cdots + \beta_1 y_{t-p} + \varepsilon_t = \sum_{i=1}^{p} \beta_i y_{t-i} + \varepsilon_t \tag{1.29}$$

dove:

– y_t è il valore della serie temporale all'istante t;

– il vettore dei parametri $\beta_1, \beta_2, \cdots, \beta_p$ rappresenta l'insieme dei coefficienti della regressione lineare;

– p è l'ordine del modello, ovvero il numero di istanti temporali presi in considerazione;

– ε_t è il termine di errore, rappresenta le fluttuazioni non spiegabili dal modello.

Per utilizzare bene un modello AR, bisogna seguire delle fasi di analisi bene precise

1) *Identificare ordine p*: bisogna trovare il valore ottimale dell'ordine p del modello, utilizzando strumenti statistici come ACF, AIC, BIC ecc.

2) *Stima dei parametri*: i coefficienti $\beta_1, \beta_2, \cdots, \beta_p$ devono essere stimati utilizzando tecniche statistiche come massima verosimiglianza o minimi quadrati.

3) *Validazione del modello*: bisogna verificare che il modello sia adeguato mediante l'analisi dei residui, ossia di ε_t, questo si deve distribuire come un white noise (rumore bianco).

4) *Previsione*: alla fine si possono effettuare le previsioni su dati storici.

1.4.2 Vantaggi e svantaggi

I vantaggi di usare modelli autoregressivi nella previsione di serie temporali sono:

– *Semplicità*, essendo la struttura matematica dei modelli molto intuitiva;

– *Facilità di stima*, esistono metodi consolidati ed efficaci per stimare i parametri autoregressivi;

– *Previsione efficiente*, in molti casi i modelli AR sono sufficientemente potenti per produrre previsioni accurate con dati stazionari.

I svantaggi invece

– *Assunzione di stazionarietà*, tali modelli richiedono che la serie sia stazionaria;

– *Non adatto per relazioni non lineari*, in quanto i modelli riescono a catturare solo relazioni lineari tra i valori.

2 Approccio utilizzato

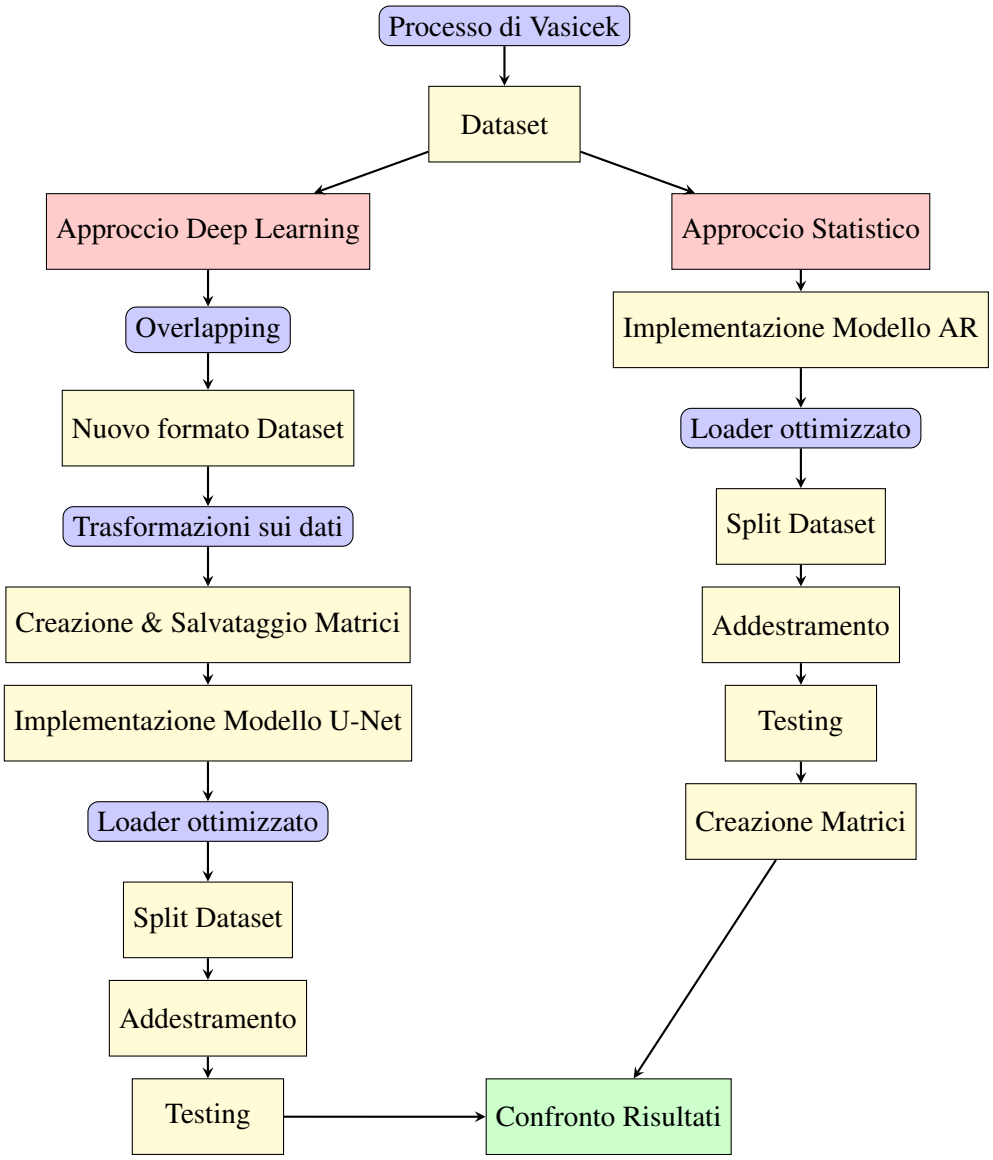


Figura 2.1: *Schema analisi effettuata.*

Nel presente capitolo, illustrerò nel dettaglio il procedimento adottato durante la fase di ricerca e verranno esaminati i risultati ottenuti. In primo luogo, facendo riferimento alle tecniche analizzate nel Capitolo 1, verrà mostrato come queste siano state applicate al contesto del progetto (*TensorFlow: A system for large-scale machine learning, 2016*[13]). Successivamente, verranno presentati i modelli scelti ed utilizzati, soffermandoci sulle loro caratteristiche. Per avere una panoramica generale del processo di analisi osserviamo la figura [2.1].

2.1 Manipolazione Dataset

Il dataset che ci è stato fornito inizialmente è descritto nella sezione (1.1); tuttavia, è stato necessario modificarlo leggermente per adattarlo meglio alle esigenze del task. In particolare, abbiamo introdotto l'overlapping nei dati per facilitare il modello ad apprendere dai dati. In tal modo la rete "osserva" nelle matrici di input e di output una parte generata da dati in comune, così facendo la si aiuta a comprendere che il compito da svolgere consiste non nel ricreare una nuova matrice successiva ad una data, ma nel continuare a ricostruirla partendo da una porzione che già conosce. Questo non solo alleggerisce il task, ma consente anche un maggior controllo sulla struttura dei dati che passiamo in input. Per implementare questo processo di overlapping nel dataset, abbiamo sviluppato una funzione che prende in input il dataset privo di overlapping e, per rendere più semplice la fase di debug del codice, restituisce un nuovo dataset dove ogni riga corrisponde ad un sottogruppo. In questo modo 1 riga del dataset iniziale diventa 80 nuove righe nel dataset restituito.

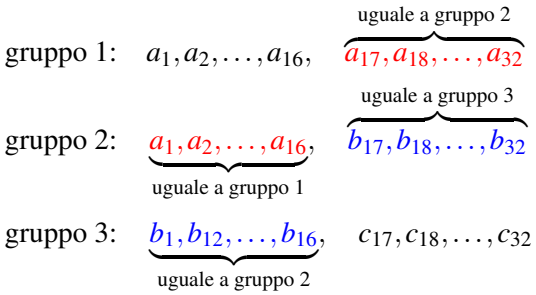
Algorithm 1 Creazione nuovo dataset con Overlap

```
procedure FUNZIONE(dataset: Array, group_length: int, overlap_percentage: float)
    step ← int(group_length × (1 − overlap_percentage))
    n_rows, n_cols ← shape(dataset)
    n_groups_per_row ← (n_cols − group_length) // step + 1
    groups ← preallocare array (n_rows × n_groups_per_row, group_length)
    for i ← 0 to (n_rows − 1) do
        row_groups ← operazione di stride()[1]
        groups[i × n_groups_per_row : (i + 1) × n_groups_per_row] ← row_groups
    end for
    return groups
end procedure
```

Il funzionamento è molto semplice, basandosi su una percentuale di sovrapposizione passata come parametro di input, la funzione itera su ogni riga del dataset dividendo in partizioni. Ciascuna coppia di partizione che si susseguono contengono una porzione di dati identici, facilitando così il processo di apprendimento del modello. Prendiamo come esempio i primi

[1] L'operazione di Stride crea una vista di un array (o matrice) esistente senza fare copie dei dati sottostanti, permettendo così di accedere ai dati con un determinato "passo" tra gli elementi. Nella funzione, lo stride è usato per creare gruppi di dati sovrapposti, sfruttando la memoria esistente del dataset per generare le sotto-sequenze in modo efficiente.

tre sotto-gruppi di una serie, per ogni *groups* verrà creata una matrice utilizzando le trasformazioni GASF, GADF e Markov, in tal modo le immagini che verranno prodotte avranno delle porzioni molto simili tra loro in corrispondenza delle parti uguali dei sottogruppi.



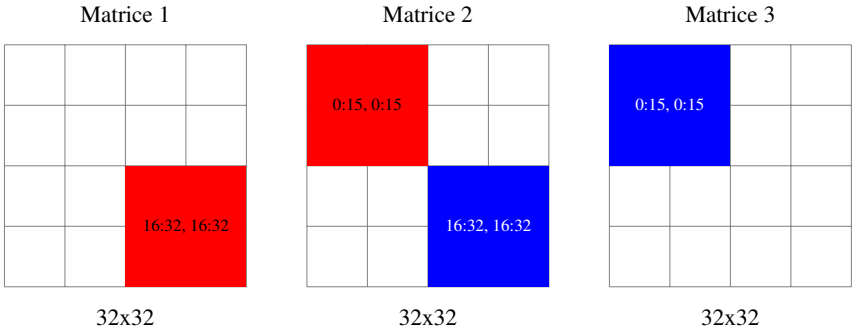


Figura 2.2: *Struttura delle matrici costruite a partire da tre sotto-gruppi successivi*

Ogni matrice rappresenta un intervallo di 16 giorni di transazioni, pertanto, da una prospettiva temporale, i modelli verranno addestrati a prevedere le transazioni di un periodo di 16 giorni utilizzando i dati dei 144 giorni precedenti. Nel nostro caso utilizzeremo come parametri della funzione *group lenght = 32* e *overlap percentage = 0.5*. Per quanto riguarda la lunghezza delle partizioni, la scelta è stata obbligata poiché le più piccole matrici che la U-Net accetta sono di dimensione 32×32. Dovevamo quindi trovare la giusta percentuale di overlap tra i sotto-gruppi, la scelta di utilizzare il 50% è stata fatta essenzialmente per aiutare il modello a ricostruire. Inoltre, da un punto di vista matematico per ottenere ci permetteva di avere un numero di sotto-gruppi per ogni serie che fosse un multiplo di 10 abbastanza grande. Infatti, dato un array di N elementi, il numero di gruppi n che può essere ottenuto dall'array può essere calcolato come

$$n = \left\lceil \frac{N-G}{G-O} \right\rceil + 1 \tag{2.1}$$

G : dimensione di ogni gruppo;
O : overlap tra gruppi

Con $N = 1296$, $G = 32$ e $O = 16$ (in quanto 50% di 32) si avrà

$$n = \left\lceil \frac{1296 - 32}{32 - 16} \right\rceil + 1 = \left\lceil \frac{1264}{16} \right\rceil + 1 = \lceil 79 \rceil + 1 = 80 \tag{2.2}$$

Dunque, utilizzando questa tecnica ogni riga del dataset iniziale verrà suddivisa in 80 sottogruppi da 32 dati ciascuno con overlap del 50% tra essi.
Il dataset finale avrà dunque la seguente struttura:

Serie	**Gruppo**	**dati simulati**						
1	1	dato reale	x_2	x_3	...	x_{15}	x_{16}	x_{17}
1	2	x_{18}	x_{19}	x_{20}	...	x_{30}	x_{31}	x_{32}
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	⋮
1	80	x_{66}	x_{67}	x_{68}	...	x_{78}	x_{79}	x_{80}
2	1	dato reale	x_{82}	x_{83}	...	x_{95}	x_{96}	x_{97}
2	2	x_{98}	x_{99}	x_{100}	...	x_{110}	x_{111}	x_{112}
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	⋮
2	80	x_{146}	x_{147}	x_{148}	...	x_{158}	x_{159}	x_{160}
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	⋮
n	1	dato reale	x_{J+2}	x_{J+3}	...	x_{J+15}	x_{J+16}	x_{J+17}
n	2	x_{J+18}	x_{J+19}	x_{J+20}	...	x_{J+30}	x_{J+31}	x_{J+32}
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	⋮
n	80	x_{J+66}	x_{J+67}	x_{J+68}	...	x_{J+78}	x_{J+79}	x_{J^2+80}

Tabella 2.1: *Struttura del dataset finale*

2.2 Creazione e Salvataggio delle Matrici

Una volta ottenuto il dataset, comincia la fase di creazione delle matrici che dovranno poi esser usate dai modelli. Il dataset iniziale 1.1 contiene 50000 simulazioni di serie storiche, quindi per riuscirci a lavorare in modo efficiente abbiamo optato per suddividerlo in 100 file *.csv* che contengono ognuno 1000 simulazioni del dataset iniziale. Ogni file viene processato come visto nella sezione precedente (2.1), il formato finale di ogni file *.csv* sarà come indicato dalla tabella [2.1] e su questa struttura dei dati si sviluppa il procedimento per creare le matrici. Si itera su ogni riga del dataset e per ognuna di esse vengono create tutte e tre le matrici GASF, GADF e Markov (*pyts: A Python Package for Time Series Classification, 2020*[14]) come visto nella sezione (1.2.2) e salvate in formato *.npy*.

[2] $J = 80(n-1)$ essendo che ogni serie ha 80 valori quindi la prima andrà da 1 ad 80, la seconda da 81 a 160, l'$n-esima$ andrà da 80(n-1)+1 ad 80n

Per il metodo di salvataggio abbiamo optato per dividere le matrici sulla base della natura dei loro dati, ossia come segue:

– Cartella che contiene tutte le immagini:

1) sotto-cartella **TRAIN**
   (a) sotto-cartella GASF: 0.npy, 1.npy, ⋯, k.npy
   (b) sotto-cartella GADF: 0.npy, 1.npy, ⋯, k.npy
   (c) sotto-cartella Markov: 0.npy, 1.npy, ⋯, **k**.npy
2) sotto-cartella **VALIDATION**
   (a) sotto-cartella GASF: (**k+1**).npy, (k+2).npy, ⋯, j.npy
   (b) sotto-cartella GADF: (k+1).npy, (k+2).npy, ⋯, j.npy
   (c) sotto-cartella Markov: (k+1).npy, (k+2).npy, ⋯, **j**.npy
3) sotto-cartella **TEST**
   (a) sotto-cartella GASF: (**j+1**).npy, (j+2).npy, ⋯, z.npy
   (b) sotto-cartella GADF: (j+1).npy, (j+2).npy, ⋯, z.npy
   (c) sotto-cartella Markov: (j+1).npy, (j+2).npy, ⋯, z.npy

Le matrici sono indicizzate in modo tale che tutte quelle appartenenti al training set abbiano lo stesso identificativo, denominato *indice.npy*, ma siano suddivise per tipologia. Le matrici appartenenti al validation set hanno un identificativo che continua la numerazione rispetto all'ultimo elemento del training set e lo stesso criterio è applicato per le matrici del test set. Questa metodo di indicizzazione è stato adottato per garantire che, durante la fase di caricamento, le matrici possano essere elaborate in maniera sequenziale, un aspetto fondamentale nel contesto del forecasting.

2.3 Loader Matrici

Una volta ottenute e salvate tutte le matrici necessarie per l'addestramento del mode llo, è fondamentale ottimizzare il processo di caricamento delle stesse. Caricare tutti i file contemporaneamente per poi processarli risulta inefficiente sia in termini di memoria (RAM) sia in termini di tempo. Dal punto di vista della memoria, caricare tutte le matrici insieme potrebbe saturare la RAM disponibile, impedendo al sistema di gestire altre operazioni necessarie e rallentando l'intero processo di training. Dal punto di vista del tempo, l'accesso simultaneo ad una grande quantità di dati può introdurre latenze, soprattutto quando si tratta di file di grandi dimensioni oppure, come nel nostro caso, di alta dimensionalità dei dati. Queste latenze possono rallentare il flusso di lavoro, ritardando l'addestramento del modello.

Per affrontare queste problematiche, abbiamo adottato una strategia più efficiente: invece di caricare immediatamente tutte le matrici, carichiamo solo i percorsi (paths) ai file delle

matrici in modo sequenziale, man mano che servono al modello. In questa fase, i paths vengono divisi in input e target in base alla struttura del modello. Successivamente, prendiamo coppie di paths (input e target) e creiamo degli oggetti *tf.data.Dataset* insieme con *map_fn* così da caricare i dati in batch, ottimizzando il processo utilizzando il parallelismo quando vengono passati uno alla volta al modello durante la fase di training. In questo modo, le matrici vengono caricate in memoria solo quando necessario, ottimizzando l'uso delle risorse di sistema. Questa tecnica dunque, sfruttando sia la programmazione funzionale che la gestione dinamica della memoria, ci permette di eseguire il training in modo più efficiente riducendo la complessità del processo.

Algorithm 2 Loader ottimizzato

```
1:  procedure FUNZIONE(path: str, method: str, lenght_input: int)
2:      array, paths ← Preprocess[3](path)
3:      inputs, targets ← list
4:      method_diz ← dict(sum : 0, dif : 1, mark : 2)[4]
5:      for J ← 0 to len(arrays) step lenght_input + 1 do
6:          if J + lenght_input ≤ len(arrays) then
7:              matx_paths ← list
8:              for i ← J to J + lenght_input do
9:                  matx_paths.append(Path Matrice Input)
10:             end for
11:             target_paths ← Path Matrice Target
12:             inputs.append(matx_paths)
13:             targets.append(target_paths)
14:         end if
15:     end for[5]
16:     return inputs, targets
17: end procedure
```

Algorithm 3 Create BatchDataset

```
1: procedure FUNZIONE(input_paths:list, target_paths:list, batch_size:int)
2:     Inizializzo tf.DataTensor(input_paths, target_paths)
3:     Applico map_fn(tf.DataTensor, batch_size) per parallelizzare processo
4:     return BatchDataset
5: end procedure
```

[3]La funzione *Preprocess()* prende in input la lista dei path delle matrici su cui applica una trasformazione di standardizzazione.

[4]Viene creato un dizionario per effettuare il mapping per gli indici dei path.

[5]Ciclo annidato che permette di scorrere la lista dei path con step di lunghezza pari al numero di matrici che verranno passati in input e concatenarli, successivamente salvare il path del target, così via.

2.4 Modelli implementati

Per il progetto abbiamo deciso di implementare quattro diversi modelli, tre di loro di deep learning e uno statistico, per poi effettuare un confronto fra di essi. I modelli di deep learning sono stati progettati per effettuare il forecasting utilizzando le informazioni contenute in un'unica tipologia di matrice, mentre il modello statistico sfrutta le informazioni dei p istanti di tempo precedenti al target di previsione.

I modelli di deep learning si basano su un'architettura chiamata U-Net, fornita dalla libreria *segmentation-models 1.0.1* (*Segmentation Models, 2019*[15]). Tale libreria implementa diverse architetture di deep learning specializzate nella segmentazione di immagini, tra cui proprio la U-Net. Per il modello autoregressivo invece è stata utilizzata la versione *AutoReg* implementata della libreria *statsmodels* (*statsmodels: Econometric and statistical modeling with python, 2010*[16]).

2.4.1 Modello $AR(p)$

Dato un modello autoregressivo classico, osserviamo come modificarlo per adattarsi al confronto con le metriche ottenute dal modello [2.4.2] di deep learning.
Presa una successione di serie storiche $\{Y_1, Y_2, \cdots, Y_k\}$ di lunghezza u, il modello autoregressivo $f(\cdot)$ lavora sulla singola serie nel seguente modo:

$$f(y_1, y_2, y_3, \cdots, y_{u-33}) \quad \rightarrow \quad \hat{y}_{u-32}, \hat{y}_{u-31}, \cdots, \hat{y}_u \tag{2.3}$$

– I valori che vanno dall'inizio della serie fino al valore antecedente gli ultimi 32, verranno usati per stimare i coefficienti del modelli;

– Gli ultimi 32 valori invece verranno usati per calcolare le metriche relative alle predizioni fatte dal modello.

Dunque, il modello $AR(p)$ [1.4] è stato implementato per eseguire forecast degli ultimi 32 valori utilizzando le informazioni contenute nei precedenti istanti temporali. Segue un esempio delle equazioni iterative del procedimento.

$$\hat{y}_{u-32} = \hat{\beta}_1 y_{u-33} + \hat{\beta}_2 y_{u-34} + \cdots + \hat{\beta}_k \, y_{u-(k+32)} \tag{2.4}$$

$$\hat{y}_{u-31} = \hat{\beta}_1 \hat{y}_{u-32} + \hat{\beta}_2 y_{u-33} + \cdots + \hat{\beta}_k \, y_{u-(k+31)} \tag{2.5}$$

$$\hat{y}_{u-30} = \hat{\beta}_1 \hat{y}_{u-31} + \hat{\beta}_2 \hat{y}_{u-32} + \cdots + \hat{\beta}_k \, y_{u-(k+30)} \tag{2.6}$$

e così via fino a raggiungere la 32-esima predizione.

$$\hat{y}_u = \hat{\beta}_1 \hat{y}_{u-1} + \hat{\beta}_2 \hat{y}_{u-2} + \cdots + \hat{\beta}_{32} \, \hat{y}_{u-32} + \cdots + \hat{\beta}_k \, y_{u-k} \tag{2.7}$$

Questo processo viene ripetuto per tutte le serie nel dataset e per ciascuna di esse vengono calcolate le metriche di accuratezza.[6] Le serie storiche utilizzate in questo modello sono generate da un processo di Vasicek, noto per non presentare stagionalità. Pertanto, per garantire che le serie siano stazionarie, è stata applicata la differenza prima ad ogni serie. La differenza prima è definita come:

$$\Delta Y_t = Y_t - Y_{t-1} \tag{2.8}$$

Questo passaggio consente di rimuovere eventuali trend lineari nelle serie temporali, rendendole più adatte per la modellizzazione tramite AR(p).

Durante la fase di stima dei parametri, è stato verificato che i residui del modello siano *white noise* utilizzando il test di *Breusch-Godfrey* che è simile al test di Ljung-Box, ma è progettato specificamente per i modelli di regressione e viene utilizzato per testare l'ipotesi congiunta che non vi sia autocorrelazione nei residui fino a un certo ordine. Quindi una volta calcolato il modello *AR* si estraggono i residui, $e_1, e_2, \cdots, e_k$, si stima la regressione ausiliaria:

$$e_t = \gamma_0 + \gamma_1\, x_{t,1} + \cdots + \gamma_k\, x_{t,k} + \rho_1\, e_{t1} + \cdots + \rho_p\, e_{tp} + u_t \tag{2.9}$$

Se per questo modello viene calcolato l'R^2, allora la seguente approssimazione asintotica può essere utilizzata per la distribuzione della statistica test:

$$(T-p)R^2 \sim \chi^2_p \tag{2.10}$$

Se p-value $\leq \alpha$, allora l'ipotesi nulla $H0 : \rho_1 = \rho_2 = \cdots = \rho_p = 0$ viene rifiutata e quindi almeno uno dei ρ_j è significativamente diverso da zero.
Dopo aver effettuato questa verifica sui residui del modello stimato, viene eseguito il forecast sugli ultimi 32 valori, di seguito è rappresentato lo pseudo-codice che mostra l'intero procedimento di analisi per ogni singola serie storica.

[6] Come si osserva dalla successione delle equazioni che caratterizzano il processo del modello autoregressivo, man mano che si va avanti il modello utilizza le predizioni fatte all'instante di tempo successivo come paramentro per effettuare la previsione. Per questo motivo ci aspetteremo che le predizioni tendino a divenatare sempre meno accurate a casua dell'accumularsi di distorsione.

Algorithm 4 Allenamento e Previsione su Singola Serie

```
1:  procedure FUNZIONE(Serie:array, n_forecast:int, lags:list)
2:      Split serie
3:      train_part ← serie[: −n_forecast])
4:      actual_values ← serie[−n_forecast :]
5:      Differenziazione della serie per renderle stazionarie
6:      diff_train, last_obs_train ← difference(train_part)
7:      diff_actual, last_obs_actual ← difference(actual_values)
8:      Inizializzazione parametri
9:      best_aicc ← ∞ e best_model ← None
10:
11:     for l in lags do
12:         Stima modello AR
13:         model ← AutoReg(diff_train, l)
14:         model_fit ← model.fit()
15:         Validazione dei residui - test di Breusch-Godfrey
16:         residuals ← model_fit.resid
17:         if il modello supera il test then
18:             Calcola l'AICc
19:             aic ← model_fit.aic
20:             if aicc < best_aicc then
21:                 Aggiorna il miglior modello
22:                 best_model ← model_fit
23:                 best_aicc ← aicc
24:             end if
25:         end if
26:     end for
27:
28:     if esiste un best_model then
29:         Prevedi gli ultimi n_forecast dati
30:         diff_pred ← best_model.predict()
31:         Riporto i dati alla scala originale
32:         pred ← inverse_diff(last_obs_train, diff_pred)
33:         actual_val ← inverse_diff(last_obs_actual, diff_actual)
34:         return pred e actual_val
35:     end if
36: end procedure
```

2.4.2 Modelli U-Net

```
model = sm.Unet('resnet34', input_shape=(None, None, 9),
    encoder_weights=None)
```

Per i modelli di deep learning che utilizzano le matrici GADF, GASF o Markov, abbiamo dovuto modificare la loro architettura in modo tale da accettare in input 9 canali (corrispondenti ai 9 istanti temporali), mantenendo invariato il singolo canale di output. Questo perché i modelli a singola matrice vengono addestrate utilizzando in input uno stack di 9 matrici consecutive, mentre il target è rappresentato dalla decima matrice. Questo processo viene ripetuto nel tempo, permettendo ai modelli di apprendere come ricostruire una matrice futura basandosi sulle informazioni contenute nelle 9 precedenti, tenendo presente che ogni matrice rappresenta un intervallo di 16 giorni di transazioni. Pertanto, da una prospettiva temporale, i modelli vengono addestrati a prevedere le transazioni di un periodo di 16 giorni utilizzando i dati dei 144 giorni precedenti, figura [2.3].

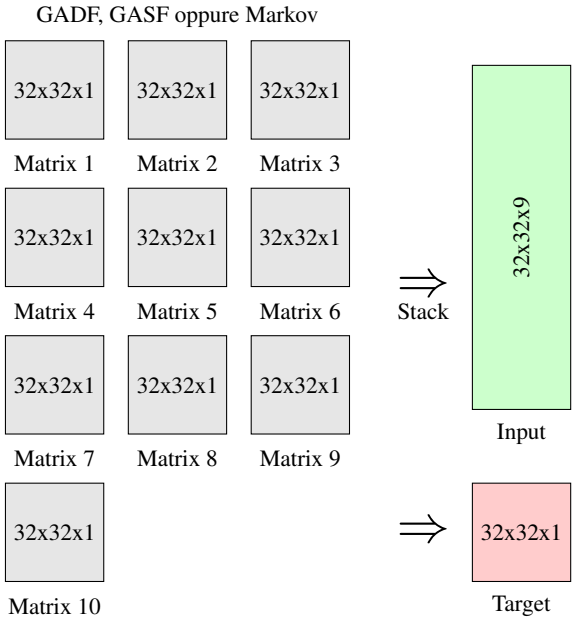


Figura 2.3: *Struttura Input-Target Modello U-Net*

Loss personalizzata

Abbiamo scelto di utilizzare il MSE (Mean Squared Error) come funzione di loss per il nostro modello. Inizialmente, abbiamo adottato la sua versione standard, ma successivamente, nel tentativo di migliorare le prestazioni del modello, abbiamo sviluppato una loss personalizzata basata proprio sul MSE. L'idea alla base di questa nuova loss era di penalizzare maggiormente le aree della matrice ricostruita che rappresentano le previsioni più vicine all'ultimo valore dell'input considerato, mentre le aree che corrispondono a previsioni più

lontane nel tempo dovevano essere penalizzate meno. Questa strategia di penalizzazione è stata dettata dalla natura dei nostri dati. Infatti, ricordando il paragrafo (2.1), per aiutare l'architettura ad imparare meglio, abbiamo costruito le matrici a partire da batch di dati con un overlapping del 50%. Ciò significa che il modello prenderà in input più volte le stesse informazioni, sempre corrispondenti alla stessa area, in matrici successive. In questo modo, il modello viene incentivato a migliorare la previsione di quelle aree specifiche, poiché le osserva ripetutamente. L'obiettivo di questa modifica alla funzione di loss è dunque quello di rendere il modello più preciso nelle previsioni a breve termine, che sono considerate più critiche, mentre si accetta una penalizzazione inferiore per errori nelle previsioni a lungo termine, per cui c'è meno informazione disponibile. Di seguito è rappresento l'algoritmo che implementa la funzione di Loss personalizzata.

Algorithm 5 Loss custom

procedure FUNZIONE(y_true: array, y_pred: array)
    $q1\_true,\ q1\_pred \leftarrow y\_true[:,0:16,0:16,:],\ y\_pred[:,0:16,0:16,:]$
    $q1\_true,\ q1\_pred \leftarrow y\_true[:,0:16,16:32,,:],\ y\_pred[:,0:16,16:32,:]$
    $q1\_true,\ q1\_pred \leftarrow y\_true[:,16:32,0:16,:],\ y\_pred[:,16:32,0:16,:]$
    $q1\_true,\ q1\_pred \leftarrow y\_true[:,16:32,16:32,:],\ y\_pred[:,16:32,16:32,:]$

    $mse\_q1 \leftarrow \text{tf.mse}(q1\_true - q1\_pred))$
    $mse\_q2 \leftarrow \text{tf.mse}(q2\_true - q2\_pred))$
    $mse\_q3 \leftarrow \text{tf.mse}(q3\_true - q3\_pred))$
    $mse\_q4 \leftarrow \text{tf.mse}(q4\_true - q4\_pred))$
    $\lambda_1,\ \lambda_2,\ \lambda_3,\ \lambda_4 \leftarrow \text{int}$

    $loss \leftarrow \text{tf.nn.sigmoid}(\lambda_1 * mse\_q1 + \lambda_2 * mse\_q2 + \lambda_3 * mse\_q3 + \lambda_4 * mse\_q4)$
    return *loss*
end procedure

La funzione di perdita personalizzata è descritta formalmente come segue

$$\mathscr{L}_{\text{custom}} = \frac{1}{N} \sum_{i,j} \lambda_{i,j} \cdot (y_{i,j} - \hat{y}_{i,j})^2 \tag{2.11}$$

dove $\lambda_{i,j}$ è il fattore di penalizzazione per la posizione (i, j) nella matrice, tale che:

$$\lambda_{i,j} = \begin{cases} \lambda_1 & \text{se } (i,j) \in \text{Quadrante 1 (previsioni a breve termine)} \\ \lambda_2 & \text{se } (i,j) \in \text{Quadrante 2} \\ \lambda_3 & \text{se } (i,j) \in \text{Quadrante 3} \\ \lambda_4 & \text{se } (i,j) \in \text{Quadrante 4 (previsioni a lungo termine)} \end{cases} \quad \text{Con } \lambda_1 > \lambda_2 \geq \lambda_3 > \lambda_4$$

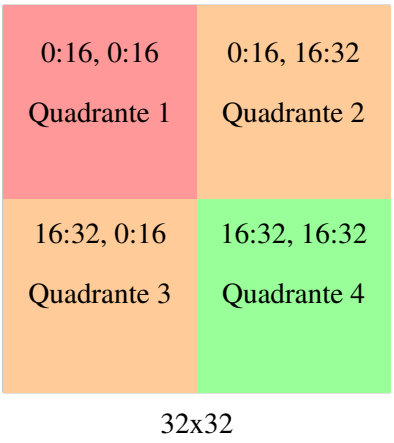


Figura 2.4: *Matrice 32x32 divisa in quattro quadranti distinti per colore in base alle penalizzazioni differenziate.*

2.5 Metriche di valutazione della ricostruzione e previsione

Per valutare l'accuratezza della ricostruzione della matrice e delle previsioni, abbiamo optato per usare quattro diverse metriche

- Structural Similarity Index Measure;
- Mean Squared Error;
- Mean Absolute Error;
- Peak Signal-to-Noise Ratio.

Structural Similarity Index Measure (SSIM)

Structural Similarity Index Measure o SSIM (*Image quality assessment: from error visibility to structural similarity, 2004*[17]). Questa metrica permette di valutare quanto due immagini siano simili tra di loro in termini di strutture visive percepite dall'occhio umano. Infatti, a differenza di altre metriche tradizionali come MSE o PSNR (Peak Signal-to-Noise Ratio) che valutano la differenza pixel per pixel senza considerare caratteristiche più elevate, SSIM cerca di misurare la similarità tenendo conto di tre fattori: *contrasto*, *luminosità* e la *struttura* delle immagini.

Matematicamente, la formula del SSIM per due immagini x e y con una finestra di confronto W è descritta dalla seguente equazione:

$$SSIM(x,y) = l(x,y) \cdot c(x,y) \cdot s(x,y) \tag{2.12}$$

Dove

– *Componente di luminanza*
$l(x,y)$ viene calcolata usando la media μ_x e μ_y delle due immagini:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2.13}$$

– *Componente di contrasto*
$c(x,y)$ utilizza la deviazione standard σ_x e σ_y delle due immagini

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{2.14}$$

– *Componente di struttura*
$s(x,y)$ valuta la correlazione tra le due immagini:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{2.15}$$

Con C_1, C_2, C_3 costanti che servono per evitare instabilità nel calcolo quando i valori del denominatore sono troppo piccoli. L'SSIM varia nell'intervallo -1 e 1, rappresenta una similarità perfetta tra le due matrici.

Mean Squared Error (MSE)

Il *Mean Squared Error* (MSE) viene calcolato come la media delle differenze quadrate tra i valori puntuali delle due matrici. In formula:

$$MSE(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{2.16}$$

Dove

– x e y sono le due matrici;

– N è il numero di elementi nelle due matrici, in questo caso $N = 32 \times 32 = 1024$;

– x_i e y_i sono i valori puntauli che formano le matrici.

L'MSE misura la differenza pixel-a-pixel tra le due matrici, penalizzando maggiormente gli errori più grandi visto il quadrato nella funzione. Questa metrica dunque fornisce una buona indicazione dell'accuratezza media della ricostruzione, trattando ogni errore in modo simmetrico.

Mean Absolute Error (MAE)

Il *Mean Absolute Error* (MAE) è simile all'MSE, ma in questo caso si considera la differenza assoluta tra i valori puntuali delle due matrici. È definito come:

$$MAE(x,y) = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i| \tag{2.17}$$

A differenza dell'MSE, è meno sensibile agli outlier in quanto non eleva al quadrato le differenze tra i valori. L'MAE fornisce un'indicazione chiara di quanto, in media, i valori della matrice ricostruita differiscano da quelli della matrice originale.

Peak Signal-to-Noise Ratio (PSNR)

Il *Peak Signal-to-Noise Ratio* (PSNR) è una metrica che confronta il massimo valore di una matrice (segnale) con il rumore, ovvero la differenza tra la matrice originale e quella ricostruita. È utilizzata per misurare la qualità della ricostruzione. Il PSNR è calcolato come:

$$PSNR(x,y) = 10 \cdot \log_{10}\left(\frac{MAX_x^2}{MSE(x,y)}\right) \tag{2.18}$$

Dove MAX_x^2 è il valore massimo possibile della matrice x. Il PSNR è espresso in decibel (dB), valori più alti indicano una maggiore qualità della ricostruzione. Infatti, un PSNR elevato ci dice che il rumore presente nella ricostruzione è più basso rispetto al segnale originale.

3 Risultati e Conclusioni

Analizzando i risultati delle metriche dei tre modelli di U-Net addestrati, emergono alcuni trend e conclusioni significative.

N. Serie	Modelli	MSE	MAE	SSIM	PSNR	Loss
10.000	GADF	5.961e-03	1.830e-01	0.45139	12.82328	0.79433
	GASF	4.106e-03	1.571e-01	0.20335	14.0642	0.75912
	MARK	5.438e-03	1.784e-01	0.18646	12.07703	0.85226
25.000	GADF	5.932e-03	1.845e-01	0.45971	12.82071	0.75422
	GASF	3.993e-03	1.544e-01	0.23985	14.30158	0.75201
	MARK	5.811e-03	1.674e-01	0.25760	12.4388	0.81001
50.000	GADF	5.896e-03	1.831e-01	0.48714	12.81515	0.75474
	GASF	3.876e-03	1.508e-01	0.25137	14.33880	0.74070
	MARK	6.624e-03	1.579e-01	0.27791	12.94890	0.74239

Tabella 3.1: *Tabella con i valori delle metriche dei modelli*

3.1 Confronto tra i modelli U-Net

– GADF (*Gramian Angular Difference Field*):
Le performance del modello rimangono relativamente costanti anche con l'aumentare del numero di serie storiche utilizzate. Ad esempio, l'MSE oscilla tra 5.961e-03 e 5.896e-03, mentre il MAE si mantiene costante intorno a 1.83e-01, con valori di PSNR sempre intorno a 12.82 mentre l'SSIM è l'unica metrica che migliora gradualmente con l'aumento del numero di serie, passando da 0.45139 a 0.48714. Nonostante ciò, il modello GADF ottiene i punteggi peggiori tra i tre in termini di MSE e MAE, ciò indica che non è il più preciso nella predizione puntuale. Tuttavia, ottiene il valore più alto di SSIM, suggerendoci che invece è il modello che preserva meglio la struttura generale delle immagini predette rispetto agli altri modelli;

– GASF (*Gramian Angular Summation Field*):
I modelli GASF mostrano un miglioramento piccolo ma continuo delle performance con l'aumentare del numero di serie storiche utilizzate. Ad esempio, l'MSE passa da 4.106e-03 con diecimila serie a 3.876e-03 con cinquantamila, mentre il MAE scende progressivamente da 1.571e-01 a 1.508e-01. Anche il SSIM migliora, seppur rimanga basso, indicando una miglioramento graduale nella preservazione delle informazioni strutturali nell'immagine predetta. Notiamo inoltre un miglioramento anche del PSNR, che cresce da 14.06 a 14.34, segno che le immagini ricostruite dal modello GASF divantino man mano meno rumorose;

– *Markov*:
I modelli basati sulla matrice Markov invece hanno prestazioni meno prevedibile rispetto agli altri due modelli. Infatti, con l'aumento delle serie storiche, i valori di MSE mostrano un comportamento altalenante (passando da 5.438e-03 a 6.624e-03), d'altra parte invece il MAE mostra un miglioramento, passando da 1.784e-01 a 1.579e-01. Tuttavia, Markov ottiene i valori più alti di SSIM nel secondo e terzo test rispetto ai modelli GASF, suggerendoci che, anche se meno preciso in termini di errori assoluti, è in grado di preservare meglio la struttura visiva dell'immagine rispetto ad essi. Il PSNR, rimane inferiore rispetto a GASF in linea con l'idea che nella previsioni puntuali sia meno preciso, tuttavia mostra un leggero miglioramento tra i vari test.

Dunque, l'aumento del numero di serie storiche utilizzate per l'analisi, sembra avere un impatto positivo sulle prestazioni dei modelli:

– Le metriche come MSE e MAE migliorano leggermente, suggerendoci che l'aumento del numero di time series fornisce più informazioni ai modelli che avendo più variabilità nei dati di training, riescono a performare meglio nel task;

– L'SSIM migliora costantemente per tutti i modelli con l'aumentare delle serie, il che implica che la qualità della struttura delle immagini ricostruite migliora all'aumentare dei dati di addestramento;

– Anche il PSNR tende a migliorare, in particolarmente per i modelli GASF, il che suggerisce che l'aumento dei dati aiuta i modelli a ridurre il rumore nelle previsioni.

Nel complesso, il modello GASF emerge come il più performante in termini di errori assoluti (MSE e MAE) e qualità dell'immagine (PSNR). Con l'aumentare del numero di serie, mantiene un trend positivo e costante (seppur piccolo) che porta a migliorare le sue performance metriche. Il modello GADF invece, è il più competitivo in termini di SSIM, ossia è il modello con la più alta capacità di preservare la struttura delle immagini.

3.2 Risultati modello AR

Consideranta l'analisi delle metriche della sezione precedente, possiamo affermare che il modello GASF con dataset di cinquantamila serie storiche sia il modello più performante in termini generali. Per poter valutare al meglio se utilizzarlo possa essere vantaggioso o meno, confrontiamolo con un modello statistico costruito appositamente per task di questa natura, ossia il modello auto-regressivo descritto nella sezione (2.4.1).
I risultati del modello autoregressivo (AR) evidenzia alcuni aspetti importanti riguardo alle performance metriche di entrambi.

N. Serie	Modelli	MSE	MAE	SSIM	PSNR
50.000	Auto-Reg	5.872e-06	5.770e-03	0.10848	10.45736

Tabella 3.2: *Tabella valori metriche Modello AR*

– Osserviamo che il modello AR ha prestazioni nettamente superiori in termini di errori assoluti. Un MSE e un MAE significativamente più bassi indicano che il modello autoregressivo è molto più preciso nel predire i valori effettivi della serie temporale, commettendo errori minimi rispetto al GASF;

– Il modello GASF si dimostra più performante in termini di PSNR. Un valore più alto di PSNR significa che il modello GASF riesce a generare immagini più fedeli, con meno distorsione rispetto all'originale, rispetto al modello AR;

– Anche in termini di SSIM (Structural Similarity Index), il modello GASF supera di gran lunga il modello AR. Questo conferma che il GASF mantiene meglio la coerenza strutturale delle immagini rispetto al modello AR, nonostante gli errori numerici più elevati.

Il modello AR si dimostra dunque, molto efficace in termini di errori assoluti, quindi deve essere preferito quando l'obiettivo è minimizzare le differenze numeriche tra la previsione e i dati reali. Il modello GASF, d'altra parte, è più performante quando le qualità dell'immagine e la preservazione delle strutture visive sono gli obiettivi principali.

In conclusione, la scelta tra i due modelli dipenderà dalle nostre priorità: se si desidera minimizzare gli errori assoluti, il modello AR è la scelta migliore; se invece la qualità visiva è più importante, il modello GASF risulta preferibile.

3.3 Immagini a confronto

Osserviamo le migliori ricostruzioni di matrici effettuate dal modello GASF sulla base della metrica SSIM.

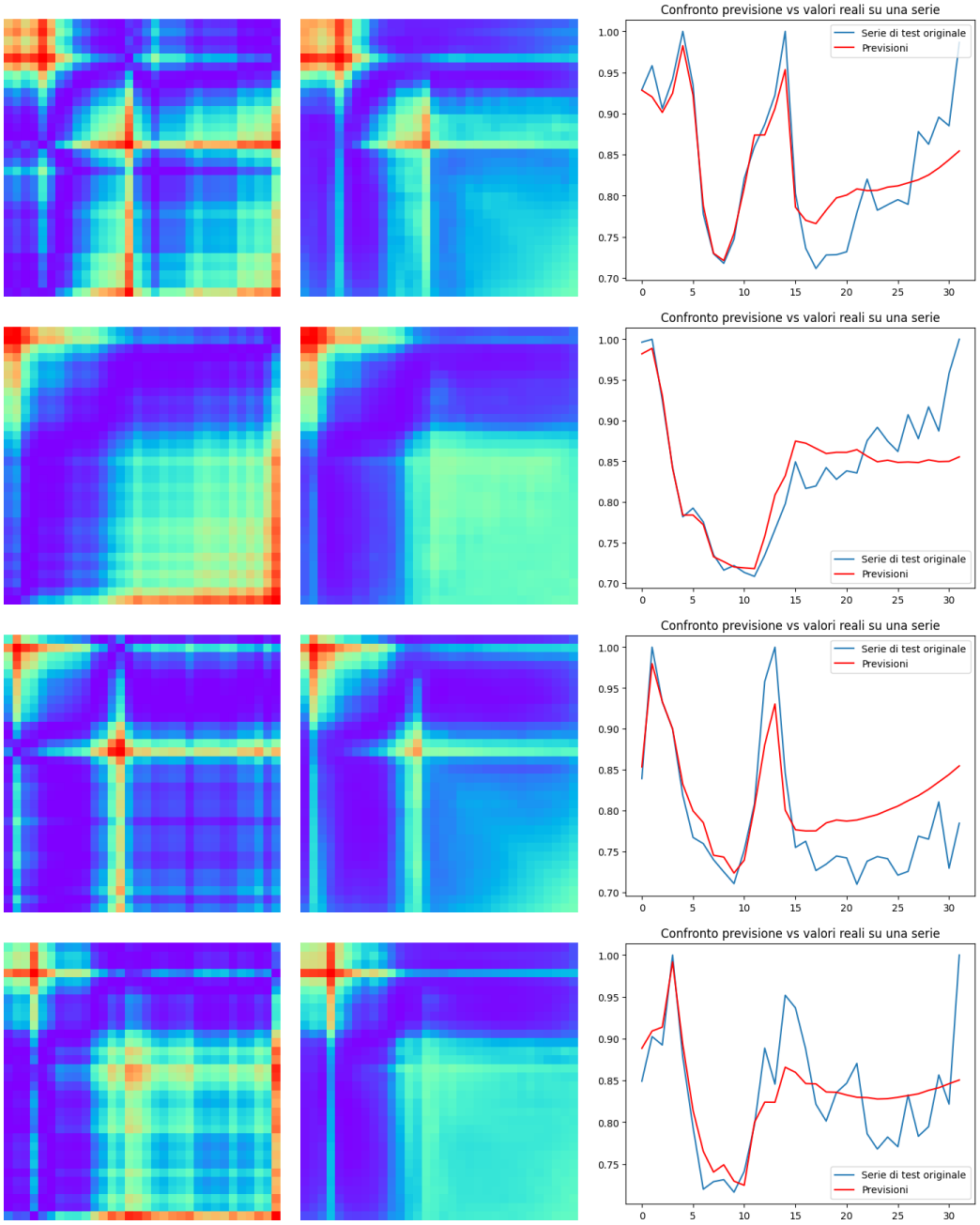


Figura 3.1: *Target-Output delle 4 migliori ricostruizioni GASF per SSIM. Dall'analisi di queste immagini possiamo affermare che il modello GASF è portato ad avere un'accuratezza maggiore nella prima metà dei dati, questo comportamento è figlio dell'approccio utilizzato, in quanto fa uso dell'overlapping durante la fase di addestramento (2.1) e della loss personalizzata che penalizza maggiormanete gli errori commessi nel primo quadrante della matrice (2.4.2).*

Analizziamo ora le predizioni effettuate dal modello AR e le relative matrici GASF prodotte da esse.

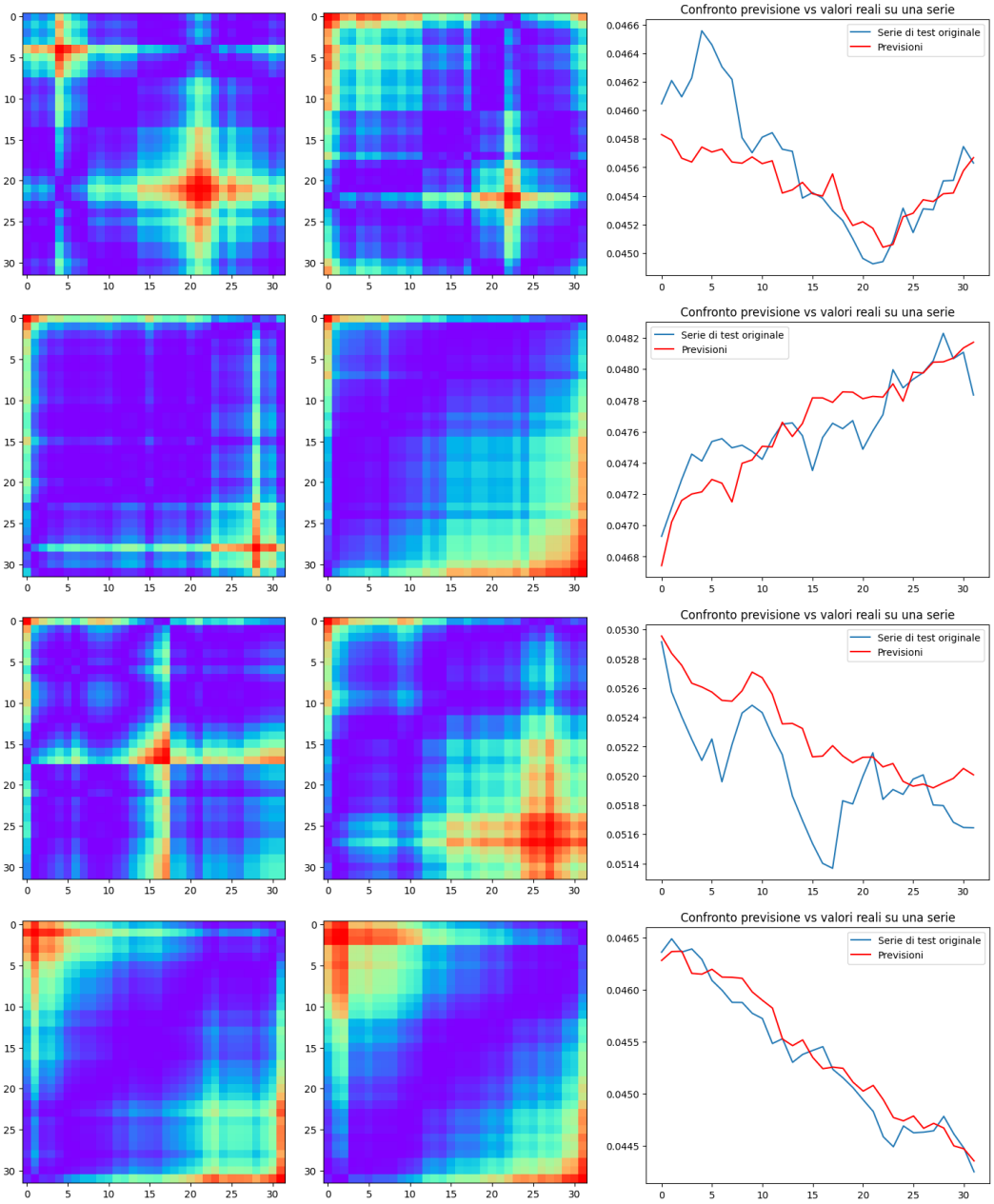


Figura 3.2: *Target-Output delle 4 migliori previsioni di AR valutate attraverso SSIM sulle matrici costruite. Dall'analisi di queste immagini si afferma, come visto anche dalle metriche, che il modello autoregressivo riesce ad avere un'accuratezza più generalizzata. Infatti, si nota come riesca sempre a cogliere l'andamento della serie in tutti e 32 gli instanti da predirre rispetto al modello GASF.*

3.4 Future work

I risultati ottenuti dimostrano che un task di forecasting, trasformato in un compito di ricostruzione di immagini attraverso l'uso del deep learning possa essere promettente. Tuttavia, ci sono diverse possibili direzioni attraverso le quali questa tipologia di approccio potrebbe essere ottimizzato per migliorare le performance. In primo luogo, abbiamo visto che l'aumentare delle time series aiuta i modelli, dunque è auspicabile che ampliando il numero di serie storiche utilizzate per l'addestramento anche l'accuratezza possa crescere. L'attuale quantità di dati dunque, potrebbe limitare la capacità del modello di catturare la variabilità presente nei dati. Utilizzando un insieme più ampio e soprattutto diversificato di serie storiche, il modello potrebbe generalizzare meglio, migliorando così la sua capacità di adattarsi a diverse tipologie di dati. Infatti, i modelli più robusti tendono ad emergere quando vengono esposti a dati eterogenei. Questo permetterebbe di poter allenare la U-Net a gestire situazioni più complesse migliorando la sua capacità di previsione. Un'altra modifica che potrebbe portare ad un miglioramento riguarda la riduzione della finestra di forecasting. Attualmente, i modelli tentano di predirre 32 istanti futuri poiché la U-Net richiede una dimensione minima della matrice di input pari a 32×32. Tuttavia, riducendo la finestra di previsione, il compito dovrebbe diventare più semplice in quanto consente al modello di concentrarsi sul ricostruire matrici più piccole, che corrispondono a previsioni di breve termine. Infine, il processo potrebbe beneficiare anche dall'uso di architetture di deep learning più sofisticate rispetto alla U-Net. La U-Net infatti, è una soluzione efficace per problemi di ricostruzione di immagini semplici, ma vi sono modelli più avanzati, come architetture transformer-based o diffusion models, che potrebbero riuscire ad avere una migliorare capacità di astrazione e generalizzazione del problema.

4 Ringraziamenti

Eccoci qui, al famigerato capitolo dei **ringraziamenti**. L'ho sognato la notte: incubi in cui, arrivati a questo punto della tesi (nessuno se la leggerà mai tutta, ma facciamo finta di sì dai), tutti si aspettano ringraziamenti strappalacrime e invece si ritrovano a leggere frasi dedicate all'Inter, a One Piece, a Dario Moccia ecc, rimanendo delusi. Però capisco che è un momento importante, quindi, con tutta la concentrazione e l'impegno possibili, sono stati scritti, ma a modo mio. Ovviamente, li ho scritti io di mio pugno, senza alcun aiuto da intelligenze artificiali varie, quindi non aspettatevi troppo.

Noemi:
Mi è sempre stato detto che bisogna partire dalle cose più belle, quindi ho deciso di iniziare da te. Da quando, ormai più di quattro anni fa, ci siamo incontrati, sei entrata nella mia vita e le hai dato una direzione inaspettata e meravigliosa. Sei sicuramente *la cosa più bella che mi sia successa in questi ultimi anni*... dopo la vittoria dello scudetto dell'Inter... e il computer nuovo... beh, sicuramente nelle prime posizioni. A parte gli scherzi, anche se non sono molto bravo a esprimermi a parole, sai benissimo cosa penso e provo per te. Spero di essere riuscito a dimostrartelo nel tempo con i miei comportamenti. In questi anni siamo cresciuti e abbiamo affrontato tante nuove esperienze, ma lo abbiamo sempre fatto insieme, fianco a fianco, sostenendoci nei momenti difficili e godendoci quelli belli. La nostra relazione mi ha insegnato che due persone che si amano veramente non vivono in simbiosi, come se fossero una sola persona, ma devono saper accompagnarsi: né un passo indietro, né uno avanti, ma sempre l'uno accanto all'altro. Questo ci ha reso complementari.
Grazie per essermi stata vicino, tra un pisolino pomeridiano di 3 ore e l'altro, durante tutto il periodo di scrittura della tesi.
Grazie per essere stata la mia prima sostenitrice, insieme a mia madre, nell'organizzare la festa. Non sei stata per nulla stressante... no, no...
Grazie per tutti i tuoi *cinque minuti e scendo*, che poi diventavano 30 o 40 minuti. Mi hanno aiutato a sviluppare la tolleranza necessaria per affrontare i problemi universitari.
Ma soprattutto, grazie per avermi sopportato e supportato nei giorni in cui la mia voglia di vivere era pari alla voglia che hai di prendere la patente :)
Averti accanto mi rende la persona più felice del mondo, e spero che possa rimanere così per taaaaanto tempo. Mi raccomando, non cambiare mai.
Sei speciale. **Ti amo.**

Equitalia:
Si dice che, per capire com'è una persona, basta osservare chi sono i suoi amici, poiché le persone che ti circondano rappresentano un po' lo specchio di quello che sei. Beh, se è così, allora sono messo benissimo. Come si potrebbe desiderare di meglio?
In sequenza casuale abbiamo:

– Un ludopatico con la passione per le minorenni (si scherza)

– Un ex militare con tendenze omosessuali (si scherza)

– Un simpatizzante di estrama destra (si scherza)

– Un ex nuotatore che ora massaggia ragazzini (si scherza)

– Un marito con relazione a distanza che per sfogarsi va a disoneste (si scherza)

– Un futuro premio Nobel che intrattiene relazioni illegali con professoresse universitarie (si scherza)

– Un ragazzo che percepisce 700 euro di stipendio al mese da 15 anni, ne ha 22 (si scherza)

– Un ragazzo con un evidente stato di ADHD avanzato e con la passione per il cioccolato brasiliano (non si scherza)

Un gruppo semplicemente **perfetto**.
Con Gianni e Francesco ho iniziato, vissuto e terminato le superiori. Tutti i giorni in quella classe, io e Francesco all'ultimo banco in fondo a destra mentre Gianni nel banco centrale in mezzo all'aula. Anni passati tra fare i versi alla Moschillo, registrare gli scleri di Sorgente e litigare con la Vairo. Ogni mattinata passata a scroccare a Gianni qualsiasi cosa avesse da mangiare, un vizio che non mi è passato visto che continuo a scroccarti i passaggi in università la mattina. Ormai sono passati anni eppure eccoci quà, tu con qualche capello in meno (Gianni), tu con qualche muscolo in meno (Francesco) io invece sempre più bello. Detto questo, Francesco, se stai leggendo questi ringraziamenti, mi devi ancora 10 euro. Muoviti, grazie...
Voi ragazzi siete stati una bellissima scoperta. Quando ci siamo conosciuti per la prima volta durante la gita in Sicilia, eravamo tutti al primo anno di superiori (tranne Kevin, che aveva già i figli), e la mia prima reazione fu: *Ma da dove sono usciti questi?* Eravate l'opposto di quello che vivevo tutti i giorni in classe, ma in realtà riflettevate una parte di me che non conoscevo ancora bene. Stare con voi infatti mi ha aiutato tantissimo, mi ha reso molto più estroverso e capace di relazionarmi con chiunque. Grazie, perché, nonostante all'inizio non fossi parte del vostro gruppo, mi avete accolto e, più tempo passavo con voi, più capivo quante cose in comune avevamo e abbiamo tuttora, fino a diventare migliori amici. Ormai dopo quasi 5 anni possiamo dire di essere il miglior gruppo di amici della penisola italica? Per forza, peffò, oaezzá. A parte tutti gli scherzi, con il passare del tempo l'amicizia che

ci lega è diventata una parte fondamentale della mia vita. Non abbiamo bisogno di molte parole per capirci, sappiamo perfettamente che ci siamo sempre l'uno per l'altro, ed è proprio questa consapevolezza che rende il nostro rapporto d'amicizia così forte. Concludendo, si sente spesso dire che gli amici sono la vera famiglia che ti scegli nel corso degli anni. Dunque: Davide, Gianni, Kevin, Giovanni, Mast, Alfonso, Paolo e Francesco; posso dire di essere fiero della famiglia che mi sono scelto. Prima di chiudere, non posso non menzionare le ragazze del gruppo: Alessia, Sara, Camilla, Giovanna e Giorgia. Siete e sarete una parte fondante di questo gruppo, scusate se non vi ho citato prima ma ahimè qui vige il patriarcato quindi questo è il massimo che si può fare. Vi voglio bene :)

UniGuys:
Un ringraziamento speciale va anche a voi: Neve, Morena, Lucia, Alessandro e Tod. Abbiamo trascorso tre anni fantastici insieme, tra studio, caffè e il far finta di credere ai racconti di Coretto. Ora ci aspettano altri due anni di fuoco, ma purtroppo ci siamo separati: voi a sopportare i tic di Ciccio Orciuoli, io tra i tizi strani di informatica. Spero davvero di rivederci fra due anni, magari a festeggiare di nuovo, questa volta con un bel contratto di lavoro in mano.

Ma un pensiero speciale va soprattutto a te, Tod. *Ma come ci è venuto in mente quel giorno di chiedere la tesi a Tagliaferri?* Lo so, è stata un'idea mia, ma tu sei il più saggio dovevi fermarmi! Solo noi sappiamo quante volte abbiamo rinnegato quella scelta, e se fosse stato possibile, avremmo venduto l'anima pur di tornare indietro. Purtroppo, non si può fare, ci siamo rimboccati le maniche, armati di pazienza in una mano e del calendario dei santi nell'altra, e alla fine siamo riusciti a portarla a casa. Non so nemmeno come, ma ce l'abbiamo fatta.

C'è solo una cosa che mi resta da dirti, e da dirmi: per la tesi magistrale, si sceglie solo la via più **semplice**.

La mia famiglia:
Per concludere, ho voluto lasciare i ringraziamenti più importanti alla fine. Come ho detto all'inizio di questo capitolo, sapete bene che non sono esattamente il tipo più adatto in queste circostanze, ma tenterò di fare del mio meglio.

Sono fortemente convinto che una persona non formi il proprio carattere in maniera autonoma, ma seguendo esempi, riconoscendo gesti e azioni delle persone che le stanno accanto. Ed è per questo che, ogni volta che ricevo un complimento per la persona che sono, non posso fare a meno di pensare a voi e ringraziarvi.
Grazie papà, per avermi insegnato l'arte della tolleranza, perché a volte nella vita è importante saper far scivolare le cose addosso.
Grazie mamma, per l'educazione che mi hai trasmesso, per avermi insegnato l'eleganza del comportamento.
Grazie papà, per avermi mostrato che la vita è fatta di momenti alti e bassi, e che l'importante è non scoraggiarsi mai.

Grazie mamma, per avermi fatto apprezzare le cose semplici, ma anche per avermi insegnato a non accontentarmi mai ed essere sempre competitivo.
Grazie papà, per quei piccoli momenti di tenerezza che nascondevano tanto amore, perché lo so: sono esattamente come te.
Grazie mamma, per quei minuti interminabili in cui mi riempivi di baci, anche se cercavo di sfuggire: in realtà ero il bambino più felice del mondo.
Grazie a entrambi, perché mi siete sempre stati accanto in ogni scelta, supportandola o contestandola, ma senza mai privarmi della mia libertà.
Grazie per essere esattamente come siete.

Vi ho sempre ammirato, vi ho sempre considerato figure inarrivabili, capaci di affrontare difficoltà che se le raccontassi pochi ci crederebbero, senza mai farci pesare nulla, né a me né a Ludo. Col tempo, ho iniziato a vedervi sempre più come persone comuni e ho pensato: *Forse potrei diventare come loro*. Ma, guardando indietro a quello che avete superato, mi rendo conto che non potrei mai raggiungervi. Per me siete e resterete sempre i miei eroi.

Voglio spendere due parole in più per te, mamma. Quest'anno è stato tosto, eh? Papà ha ben pensato di prendersi la scena proprio l'anno della mia laurea. Che egocentrico. In questo periodo ho capito ancora di più quanto tu sia una donna speciale: hai preso sulle spalle una famiglia e l'hai portata avanti da sola in un momento in cui il 99% delle persone avrebbe ceduto. Tranne te. Se non fosse stato per te, devo essere onesto, non so nemmeno se oggi sarei qui a festeggiare la mia laurea. Anzi, probabilmente no. Sei sempre stata il motore che mi ha spinto nello studio, la persona che mi ha seguito nei compiti delle elementari, che mi ascoltava ripetere ore e ore alle medie, e che mi dava consigli alle superiori. Questo traguardo è più tuo che mio. So quanto ci hai tenuto e i sacrifici che hai fatto. Ti voglio tanto tanto bene.

Infine, ci sei tu, Lulù. Lo so, non ti piace quando ti chiamano così, ma lasciamelo fare, mi ricorda quando eri piccola. Anche se ormai sei una piccola donna. Sei bellissima, e più cresci, più lo diventi. Questa è una condanna per me, perché vuol dire che si avvicina il giorno in cui dovrò chiamare Davide per farmi dare una mano a picchiare qualche povero ragazzo. Ma vabbè. Dal giorno in cui sei nata, hai dato un twist alla mia vita: hai portato quello che io chiamo il caos buono. All'inizio pensavo che il nostro rapporto sarebbe rimasto quello classico tra fratello maggiore e sorella minore, con me che dovevo farti da esempio. Ma negli ultimi anni, crescendo, siamo diventati amici ma con la potenza di essere fratelli. Capirci con uno sguardo, ridere delle stesse scemenze, confidarsi, essere complici alle spalle di mamma e papà... Sei la persona più simile a me che esista. Spero che, crescendo, questo rapporto non cambi, o se lo farà, che cambi in meglio. Voglio arrivare a 40 anni e dire ai miei figli: *Aspettate un attimo, papà deve andare a fare la lotta con zia.* Ti auguro il meglio per la tua vita, anche se so già che raggiungerai tutti i tuoi obiettivi, perché hai l'intelligenza e la determinazione per farlo. Ricorda che, da fratello maggiore, sarò sempre al tuo fianco.

Un ringraziamento va anche e soprattutto ai miei nonni, Nonno Rosario e Nonna Michela, per avermi accompagnato fino a questo momento con il loro amore. Grazie per essere stati

una presenza costante e preziosa nella mia vita. Un ringraziamento speciale va soprattutto a te, Nonna. Grazie per esserci sempre stata, anche quando avresti potuto non farlo. Invece eri lì. Grazie.

Dedicata alle donne più importanti della mia vita: Mamma e Lulù.

Bibliografia

[1] Anastasia Borovykh, Sander Bohte e Cornelis W. Oosterlee. *Conditional Time Series Forecasting with Convolutional Neural Networks, 2018*. 2018. arXiv: 1703.04691 [stat.ML]. URL: https://arxiv.org/abs/1703.04691.

[2] Silvio Barra et al. *Deep learning and time series-to-image encoding for financial forecasting, 2020*. 2020. DOI: 10.1109/JAS.2020.1003132.

[3] P. Fischer e T. Brox:2015 O. Ronneberger. *U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015*. 2015. arXiv: 1505.04597 [cs.CV]. URL: %5Curl%7Bhttps://arxiv.org/abs/1505.04597%7D.

[4] Hassan Ismail Fawaz et al. *Deep learning for time series classification, 2019*. Mar. 2019. DOI: 10.1007/s10618-019-00619-1. URL: http://dx.doi.org/10.1007/s10618-019-00619-1.

[5] Mathias Perslev et al. *U-Time: A Fully Convolutional Network for Time Series Segmentation Applied to Sleep Staging, 2019*. 2019. arXiv: 1910.11162 [cs.LG]. URL: https://arxiv.org/abs/1910.11162.

[6] Zhiguang Wang e Tim Oates. *Imaging Time-Series to Improve Classification and Imputation, 2015*. 2015. arXiv: 1506.00327 [cs.LG]. URL: https://arxiv.org/abs/1506.00327.

[7] Zhiguang Wang e Tim Oates. *Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks, 2015*. Gen. 2015.

[8] Oldrich Vasicek. *An equilibrium characterization of the term structure, 1977*. 1977. DOI: https://doi.org/10.1016/0304-405X(77)90016-2. URL: https://www.sciencedirect.com/science/article/pii/0304405X77900162.

[9] J. R. Norris. *Markov Chains, 1997*. 1997.

[10] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *I. Goodfellow: Deep Learning, 2016*. http://www.deeplearningbook.org. 2016.

[11] David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams. *Learning representations by back-propagating errors, 1986*. 1986. URL: https://api.semanticscholar.org/CorpusID:205001834.

[12] George Udny Yule. *On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers, 1927*. 1927. URL: https://api.semanticscholar.org/CorpusID:122033085.

[13] Martín Abadi et al. *TensorFlow: A system for large-scale machine learning, 2016*. 2016. arXiv: 1605.08695 [cs.DC]. URL: https://arxiv.org/abs/1605.08695.

[14] Johann Faouzi e Hicham Janati. *pyts: A Python Package for Time Series Classification, 2020*. 2020. URL: http://jmlr.org/papers/v21/19-763.html.

[15] Pavel Iakubovskii. *Segmentation Models, 2019*. https://github.com/qubvel/segmentation_models. 2019.

[16] Seabold Skipper e Perktold Josef. *statsmodels: Econometric and statistical modeling with python, 2010*. 2010.

[17] Zhou Wang et al. *Image quality assessment: from error visibility to structural similarity, 2004*. 2004. DOI: 10.1109/TIP.2003.819861.