POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Anomaly detection mediante Contrastive Learning: applicazioni su dati veicolari

Relatore

Prof. Francesco VACCARINO

Corelatore

Prof. Luca CAGLIERO

Candidato

Marianna DEL CORSO

Luglio 2022

Sommario

Le serie temporali giocano un ruolo sempre più importante in una moltitudine di ambiti, tra cui il monitoraggio e la manutenzione di macchine industriali. La diffusione dell'Internet of Things ha aumentato la facilità con cui è possibile reperire dati temporali e allo stesso tempo ha evidenziato l'importanza di sviluppare nuovi sistemi in grado di sfruttare l'enorme quantità di informazioni contenuta in tali dati.

Uno dei campi maggiormente interessanti nel Machine Learning è l'Anomaly Detection, ossia l'identificazione dell'inaspettato. L'anomaly detection viene applicata in molti contesti, come la sicurezza online, il controllo dei processi industriali o l'analisi del comportamento dei clienti. Nel caso specifico di questa tesi i dati analizzati sono metriche di servizi online e dati provenienti da dispositivi IoT installati su veicoli industriali.

Essere in grado di rilevare efficacemente le anomalie in modo automatico è un argomento di grande interesse economico per le aziende. Nel caso della manutenzione di macchine industriali, è estremamente importante essere in grado di individuare e riparare i guasti nel minor tempo possibile, maggiore è il ritardo e maggiore sarà la perdita economica. Le anomalie possono essere di diverso tipo e i comuni sistemi per il loro rilevamento non sono tipicamente in grado di gestire efficacemente tutte le tipologie.

Tra le tecniche di machine learning allo stato dell'arte spiccano quelle basate sull'apprendimento auto supervisionato. Quest'ultimo, più noto come self-supervised learning, consente di aumentare la quantità di dati a disposizione adottando strategie di data augmentation ed è in grado di apprendere una rappresentazione dai dati non etichettati sfruttando un compito ausiliario.

Questa tesi affronta il problema di anomaly detection sfruttando il Contrastive Learning, una tecnica di apprendimento auto supervisionato basata sul confronto. Il compito ausiliario del contrastive learning consiste nel creare copie alterate di ogni dato. Il modello impara a distinguere quali copie sono state ottenute a partire dallo stesso campione, raggruppandole tra loro e distinguendole da tutti gli altri dati. La risoluzione del task ausiliario forza il modello ad apprendere una rappresentazione migliore dei dati, utilizzabile facilmente per la risoluzione di compiti più complessi

come il rilevamento delle anomalie.

Il lavoro inizialmente si focalizza sul testing e ottimizzazione di un'architettura per il contrastive learning allo stato dell'arte, il framework TS2Vec, in presenza di anomalie puntuali e di contesto. Un ulteriore contributo del lavoro di tesi è l'analisi e la comparazione delle tecniche di anomaly detection allo stato dell'arte su dati acquisiti da veicoli industriali. L'obiettivo è analizzare i dati di tipo CAN Bus generati da dispositivi IoT installati a bordo dei veicoli per riconoscere in anticipo l'invio di un Diagnostic Message che segnala una possibile avaria o criticità di particolare rilevanza. Questo permette di passare dalla classica manutenzione preventiva a una manutenzione *predittiva*, molto più vantaggiosa in quanto i componenti vengono sostituiti quando è realmente necessario e non in base ai parametri dichiarati dai costruttori.

I test vengono condotti su dati provenienti da quattro dataset contenenti misurazioni normali e anomale etichettate.

I risultati ottenuti hanno evidenziato quanto sia difficile sviluppare un sistema di anomaly detection auto supervisionato in grado di funzionare in ambiti diversi e con anomalie di categorie diverse. La scelta dei componenti dell'architettura dipende fortemente dal tipo di criticità da analizzare. Le anomalie di contesto sono costituite da dati in apparenza normali ma che sono considerati anomali nel contesto in cui si verificano, riguardano un intervallo di tempo non breve e vanno gestite diversamente rispetto alle anomalie che coinvolgono solo pochi istanti. Un altro aspetto critico, molto comune nei dataset reali, è la presenza di rumore all'interno dei dati. Il rumore rappresenta un grande ostacolo per l'apprendimento auto supervisionato in quanto rende difficoltosa la distinzione delle reali anomalie dal resto dei dati.

Per alleviare le criticità riscontrate è necessario ottimizzare ulteriormente l'architettura proposta aggiungendo dei componenti che si concentrino maggiormente sul contesto.

A Mamma e Papà

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Le serie temporali sono costituite da dati ottenuti effettuando misurazioni ripetute a intervalli regolari e ultimamente giocano un ruolo sempre più importante in una moltitudine di ambiti, tra cui il monitoraggio e la manutenzioni di autoveicoli. La diffusione dell'Internet of Things ha reso i dati temporali facili da reperire e allo stesso tempo ha evidenziato l'importanza di sviluppare nuovi sistemi in grado di sfruttare l'enorme quantità di informazioni contenuta in tali dati.

Uno dei campi maggiormente interessanti nel Machine Learning è l'Anomaly Detection, ossia l'identificazione dell'inaspettato. Spesso i dati raccolti contengono misurazioni che differiscono dal comportamento normale ma che rappresentano un evento di particolare interesse proprio perché raro. L'anomaly detection viene applicata in molti contesti, come la sicurezza online, il controllo dei processi industriali o l'analisi del comportamento dei clienti. Nel caso specifico di questa tesi i dati analizzati sono metriche di servizi online e dati provenienti da dispositivi IoT installati su veicoli industriali.

Essere in grado di rilevare efficacemente le anomalie in modo automatico è un argomento di grande interesse economico. Nel caso della manutenzione di autoveicoli, è estremamente importante essere in grado di individuare e riparare i guasti nel minor tempo possibile, maggiore è il ritardo e maggiore sarà la perdita economica. L'approccio comunemente usato è la manutenzione preventiva, che consiste nel valutare la possibilità che si verifichi un guasto basandosi sulla frequenza di utilizzo del veicolo e su parametri dichiarati dai costruttori, come il tempo di vita medio dei componenti. Questa strategia non è molto efficace e spesso i veicoli si guastano prima di essere revisionati. Sfruttando una rete di dispositivi IoT montati

a bordo è possibile monitorare costantemente i parametri di funzionamento di tutte le componenti e, grazie a tecniche innovative di Machine Learning e Deep Learning, utilizzare le misurazioni raccolte per effettuare una manutenzione predittiva, cioè in grado di prevedere l'avvenimento dei guasti con un anticipo tale da permetterne la risoluzione nel minor tempo possibile.

Tra le tecniche di machine learning allo stato dell'arte spiccano quelle basate sull'apprendimento auto supervisionato. Quest'ultimo, più noto come self-supervised learning, consente di aumentare la quantità di dati a disposizione adottando strategie di *data augmentation* ed è in grado di apprendere una rappresentazione dai dati non etichettati sfruttando un compito ausiliario. Nel corso di questa tesi viene affrontato il problema del rilevamento delle anomalie sfruttando il *Contrastive Learning*, una tecnica di apprendimento auto supervisionato basata sul confronto. Il compito ausiliario del contrastive learning consiste nel creare copie alterate di ogni dato. Il modello impara a distinguere quali copie sono state ottenute a partire dallo stesso campione, raggruppandole tra loro e distinguendole da tutti gli altri dati. La risoluzione del task ausiliario forza il modello ad apprendere una rappresentazione migliore dei dati, utilizzabile facilmente per la risoluzione di compiti più complessi, tra cui l'anomaly detection.

1.1 Caso di studio

Il lavoro svolto si focalizza inizialmente sullo studio e ottimizzazione di una tecnica di anomaly detection allo stato dell'arte nel campo delle serie temporali. In particolare è stato scelto di integrare TS2Vec, un framework di recente sviluppo, con tre architetture note nell'ambito della classificazione delle serie temporali: FCN, MLP e ResNet.

Per testare le performance sono stati utilizzati dati sia reali sia sintetici (i.e. ottenuti tramite simulazione) ricavati da due contesti diversi: misurazioni di metriche online e dati di tipo CAN Bus.

Il lavoro prosegue testando il modello sviluppato su dati reali acquisiti da veicoli industriali. I dataset utilizzati vengono approfonditi nel Capitolo 5 e nel Capitolo 6.

La principale difficoltà incontrata consiste nel riconoscimento di anomalie di tipo diverso. I dataset riguardanti le metriche di servizi online contengono principalmente anomalie *puntuali*, mentre i dataset di CAN Bus contengono principalmente

anomalie *di contesto*. Queste ultime, approfondite nel Capitolo 4, sono le più interessanti da studiare ma anche le più complesse. Per riuscire a rilevarle correttamente è necessario un modello in grado di apprendere il *contesto* intrinseco nei dati, così da capire se l'andamento della serie in un certo intervallo è differente da quello che dovrebbe avere in condizioni di normalità.

Capitolo 2

Algoritmi di machine learning

L'Intelligenza Artificiale è un ramo dell'informatica nato attorno alla metà del secolo scorso grazie al quale è possibile progettare sistemi in grado di rendere le macchine capaci di imparare dagli errori e svolgere funzioni considerate esclusive dell'intelligenza umana.

Il Machine Learning è un'applicazione dell'intelligenza artificiale che si concentra sull'estrazione della conoscenza direttamente dai dati. Quando si parla di machine learning si intende la capacità di un sistema di imparare automaticamente dall'esperienza senza essere stato esplicitamente programmato. Come mostrato in Figura 2.1, il punto di vista del machine learning è diverso da quello della programmazione tradizionale. L'obiettivo di quest'ultima è creare manualmente un programma in grado di utilizzare i dati forniti in input per produrre un output in base alle regole definite dal programmatore. Nel machine learning invece i dati e l'output vengono forniti a un algoritmo con l'obiettivo di creare il programma, le regole non sono più definite da una persona ma vengono estrapolate dai dati stessi. Il programma prodotto può essere infine utilizzato per fare predizioni seguendo il modello della programmazione tradizionale.

Gli algoritmi di machine learning subiscono una fase di training in cui ricevono una grande quantità di dati da cui imparare. Tra le varie tipologie di training se ne distinguono principalmente quattro: *supervised*, *semi-supervised*, *self-supervised* e *unsupervised*. Per capire la differenza tra queste tipologie occorre prima notare che esistono due tipi di dati, i dati etichettati e i dati non etichettati. I dati non

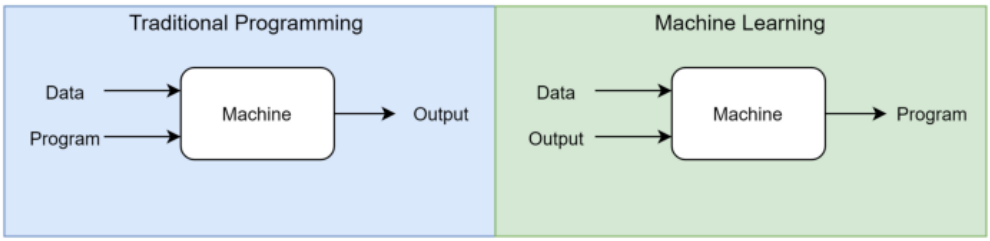


Figura 2.1: Programmazione tradizionale VS Machine Learning [1]

etichettati sono quelli più facili da ottenere, per esempio possono essere i campioni raccolti da un sensore o delle immagini. I dati etichettati sono ricavati dai dati grezzi non etichettati ed è necessario che una persona o uno strumento automatico usi la propria conoscenza per aggiungere delle informazioni. I dati vengono arricchiti con un'etichetta contenente un'informazione importante da sapere per quel dato, per esempio nel caso di immagini l'etichetta può consistere nel nome dell'oggetto presente in foto. Lavorare con dati non etichettati è molto conveniente perché non necessitano di alcuna elaborazione manuale e si possono ottenere con facilità.

2.1 Supervised learning

Il supervised learning è l'approccio più comune ed intuitivo. E' possibile ottenere ottimi risultati con modelli di questo tipo a patto di avere a disposizione una grande quantità di dati etichettati [2]. L'algoritmo riceve iterativamente in input coppie (X,y), dove X è il dato mentre y è il risultato atteso (l'etichetta), e impara una funzione f tale per cui y=f(X). L'algoritmo produce un certo risultato in base al dato in input e questo risultato viene confrontato con quello corretto atteso, nel caso in cui non coincidano l'algoritmo riceve una penalità. Quando si riesce a raggiungere una performance accettabile il training termina e il modello è in grado di generare un risultato corretto anche quando riceve in input nuovi dati. Questo tipo di apprendimento viene chiamato *supervised* perché la fase di training avviene in modo supervisionato, le risposte corrette sono note e l'algoritmo viene corretto ogni volta che commette un errore. Il supervised learning può essere ulteriormente suddiviso in due sottocategorie in base al tipo di output: classificazione e regressione [3]. La **classificazione** si ha quando l'output è categorico e quindi può assumere un numero finito di valori. Un esempio comune di classificazione è la differenziazione

delle email in ham e spam. La **regressione** si ha quando l'output è un valore reale e quindi può assumere infiniti valori. Un esempio di regressione è la previsione delle vendite in un determinato periodo dell'anno.

2.2 Unsupervised learning

Un approccio contrario al supervised learning è l'unsupervised learning, in quanto gli algoritmi di questo tipo hanno bisogno solamente di dati non etichettati. L'apprendimento avviene in modo non supervisionato perché le risposte corrette non sono note a priori come nel supervised learning. Il modello deve essere in grado di osservare la struttura dei dati ed estrarre informazioni significative. Due esempi tipici sono il clustering e la riduzione della dimensionalità dei dati (PCA) [4]. Negli ultimi anni l'unsupervised learning è stato ampiamente utilizzato anche per stimare la distribuzione di probabilità dei dati, sia esplicitamente, come nel caso dei Variational Autoencoders, che implicitamente, come nel caso delle GAN.

2.3 Semi-supervised learning

Come detto precedentemente, il supervised learning necessita di dati etichettati. Purtroppo questi non sono sempre disponibili oppure lo sono in quantità limitate. Il semi-supervised learning è adatto proprio alle situazioni in cui si hanno insiemi di dati misti, con pochi dati etichettati e molti non etichettati, e per questo viene considerato una via di mezzo tra il supervised learning e l'unsupervised learning [5]. Esistono diversi modi per sfruttare i dati, il più comune è fare un primo training con quelli non etichettati e poi fare un finetuning[1] del modello ottenuto sui dati etichettati. Purtroppo questo approccio non è sempre utilizzabile, è possibile ottenere ottimi risultati con modelli grandi mentre con modelli piccoli si va incontro al rischio di catastrophic forgetting [6], un fenomeno per cui il modello tende a dimenticare le caratteristiche dei pattern visti in passato e a ricordarsi solo ciò che ha visto di recente. Un altro approccio è di combinare un algoritmo di clustering con uno di classificazione (supervised). L'algoritmo di clustering viene applicato sui dati non etichettati per individuare quali sono i

[1]Fare un *finetuning* significa apportare piccolissime modifiche a qualcosa per farlo funzionare nel miglior modo possibile

campioni maggiormente rappresentativi per ogni classe. Questi campioni vengono etichettati (poiché si tratta di un sottoinsieme molto piccolo dei dati iniziali non si tratta di un processo costoso) e utilizzati per allenare il classificatore. Quest'ultimo viene allenato su una quantità di dati molto piccola ma, se i dati selezionati sono davvero rappresentativi dell'intera classe a cui appartengono, è possibile ottenere ottimi risultati. Sfortunatamente ci si trova spesso in situazioni più complesse dove non esistono dati rappresentativi dell'intera distribuzione, in questi casi il semi-supervised learning non può essere di aiuto.

2.4 Self-supervised Learning

Il self-supervised learning è una via di mezzo tra il supervised learning e l'unsupervised learning poiché prende i dati non etichettati e li arricchisce con etichette generate autonomamente, l'apprendimento non si basa sulla sola osservazione della struttura dei dati. I modelli supervised sono in grado di ottenere ottime prestazioni nei compiti per cui sono stati allenati ma non sono in grado di generalizzare abbastanza. Per acquisire le conoscenze necessarie per svolgere nuovi compiti è sempre necessaria una grande quantità di dati etichettati, spesso però questi dati non ci sono o sono troppo costosi da acquisire

Una persona a cui è stata mostrata la foto di un cane, è sicuramente in grado di riconoscere altri cani, anche di razze diverse da quello della foto o in posizioni differenti. Al contrario una macchina potrebbe fallire nella classificazione di cani in posizioni insolite nonostante sia stata allenata su migliaia di immagini di cani. Ciò avviene perché le persone fanno affidamento sulla loro conoscenza pregressa del mondo e sulla loro esperienza. Da questo problema è nato il bisogno di modelli in grado di comprendere meglio la realtà attraverso i dati di training. Il self-supervised learning è uno dei metodi più promettenti per riuscire ad approssimare tale conoscenza pregressa. E' in grado di imparare una rappresentazione dai dati non etichettati sfruttando un compito ausiliario, generalmente chiamato *pretext task*, e utilizzarla per lo svolgimento del compito principale, generalmente chiamato *secondary task*. Quindi gli algoritmi di self-supervised learning sono composti da due fasi: nella prima fase vengono estratte dai dati le caratteristiche più generali e indipendenti dal compito, mentre nella seconda fase si trasferisce la conoscenza acquisita nella prima per portare a termine il compito principale [7].

I metodi di self-supervised learning si suddividono in tre categorie:

1. Metodi generativi
2. Metodi *adversarial* (metodi generative-contrastive)
3. Metodi di constrasto

Tutte e tre le categorie, mostrate in Figura 2.2, sono caratterizzate da due elementi, il generatore e il discriminatore, e si differenziano per l'architettura interna di questi ultimi e per gli obiettivi finali.

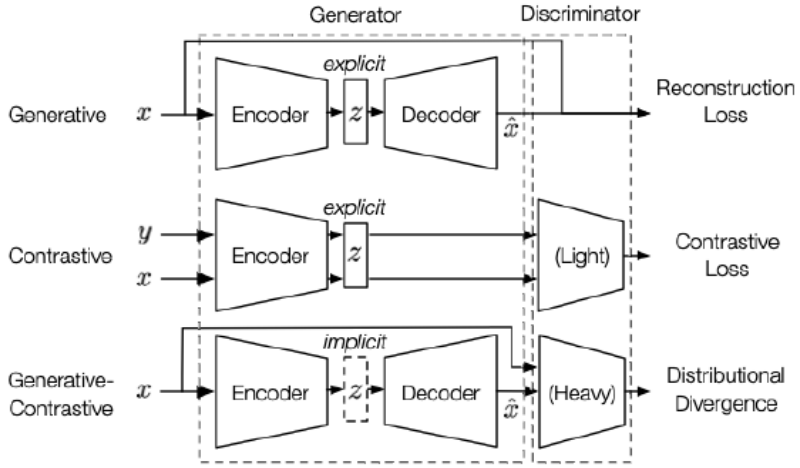


Figura 2.2: Confronto tra metodi Generative, Contrastive e Generative-Contrastive [8]

2.4.1 Metodi generativi

I modelli che rientrano in questa categoria sono gli *Autoencoder* e i *Variational Autoencoder*, vengono utilizzati come confronto per le analisi svolte nel corso della tesi e sono approfonditi nella sezione 4.3.

L'obiettivo dei metodi generativi è allenare un encoder a codificare l'input x in un vettore esplicito z e un decoder per ricostruire x a partire da z. A differenza dei metodi delle altre categorie, quelli generativi non hanno il discriminatore. L'apprendimento avviene cercando di minimizzare l'errore di ricostruzione.

Ricostruendo i dati originali i modelli estraggono una rappresentazione estremamente significativa, che può essere utilizzata per risolvere altri problemi come la classificazione. Il problema principale di questi modelli è che si concentrano molto sui dettagli dei dati di training, questo fa sì che la rappresentazione estratta non sia in grado di generalizzare abbastanza.

2.4.2 Metodi adversarial

Anche chiamati metodi generative-contrastive, i modelli che rientrano in questa categoria sono le GAN (Generative Adversarial Network).

L'obiettivo dei metodi adversarial è allenare una coppia di encoder-decoder a generare dati sintetici, simili ai dati reali di training, e un discriminatore a distinguere tra i dati reali e sintetici. Il nome *adversarial* deriva proprio dal fatto che questi metodi sono composti da due elementi tra loro avversari: il generatore ha l'obiettivo di riuscire a generare nuovi dati sintetici il più possibile simili a quelli reali, mentre il discriminatore deve riuscire a distinguere con chiarezza i dati sintetici da quelli reali.

I metodi generativi imparano a ricostruire i dati, i metodi adversarial imparano a ricostruire la distribuzione originale dei dati minimizzando la divergenza di Kullback–Leibler [9] tra la distribuzione reale e quella generata.

2.4.3 Metodi di contrasto

L'obiettivo dei metodi contrastive è allenare un encoder a codificare l'input x in un vettore esplicito z, su cui vengono applicate delle misure di somiglianza. A differenza delle altre due categorie, non è presente il decoder. Infatti i metodi di contrasto non cercano di ricostruire i dati, estraggono una rappresentazione dai dati basandosi sulla minimizzazione di una contrastive loss calcolata sulle codifiche di coppie di dati. I metodi di contrasto vengono approfonditi nel capitolo successivo.

Capitolo 3

Contrastive Learning

Ultimamente i modelli self-supervised stanno riscuotendo molto successo. Sfruttando il Contrastive Learning è possibile ottenere risultati paragonabili o addirittura superiori rispetto a quelli dei modelli supervised nel campo delle immagini [10]. I modelli basati sul contrastive learning infatti imparano a distinguere quali coppie di dati contengono elementi simili e quali no, basandosi sulle features estratte. La potenza di questa strategia sta nel fatto che non servono dati etichettati per estrarre tali features.

Il contrastive learning è un approccio discriminativo (i.e. non generativo, non si basa sulla generazione di nuovi dati) che mira a ottenere una rappresentazione robusta dei dati, cioè con un'invarianza sufficiente da ignorare variazioni insignificanti senza scartare informazioni importanti [11]. Per raggiungere questo obiettivo i dati vengono mappati in uno spazio z, cercando, grazie a una metrica di somiglianza, di avvicinare le rappresentazioni dei dati simili e allontanare quelle dei dati diversi. Il modello estrae una rappresentazione tramite la risoluzione di un task ausiliario, spesso chiamato *pretext task*, e usa questa rappresentazione come punto di partenza per il task principale.

Come riportato in [8], i metodi di contrastive learning imparano a effettuare il confronto utilizzando la *Noise Contrastive Estimation Loss* (NCE) così definita:

$$\mathcal{L}_{NCE} = -\log\left(\frac{e^{sim(x,x^+)/\tau}}{e^{sim(x,x^+)/\tau} + e^{sim(x,x^-)/\tau}}\right) \tag{3.1}$$

dove x,x^+ e x^- sono rispettivamente il dato originale, il campione positivo e il campione negativo, mentre $sim()$ è la metrica di somiglianza scelta. Nel caso in

cui si voglia considerare più di un campione negativo allora si una una variante di 3.1 chiamata InfoNCE, così definita:

$$\mathcal{L}_{infoNCE} = -\log\left(\frac{e^{sim(x,x^+)/\tau}}{e^{sim(x,x^+)/\tau} + \sum_{i=0}^{N} e^{sim(x,x_i^-)/\tau}}\right) \tag{3.2}$$

dove x_i^- è l'insieme dei campioni negativi.

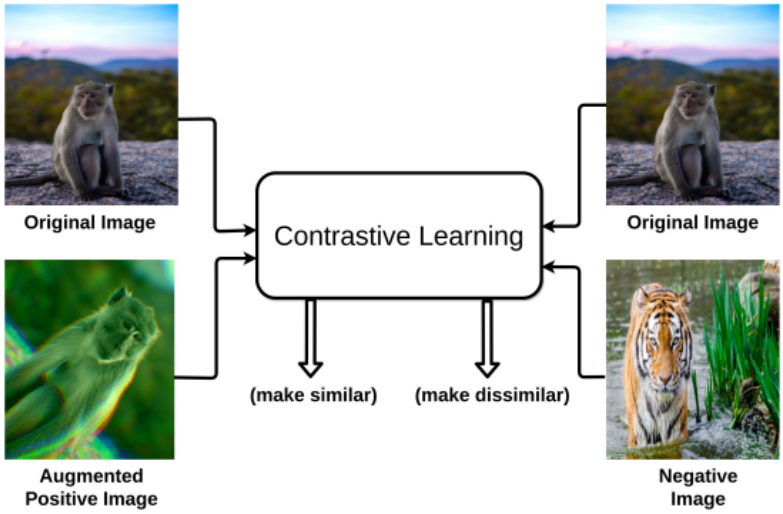


Figura 3.1: Esempio di coppia positiva e coppia negativa [12]

Poiché i metodi di contrastive learning considerano coppie di dati di input, è importante definire cosa si intende per *coppia positiva* e *coppia negativa*.

Coppia positiva: coppia di dati semanticamente identici o simili. Generalmente le coppie vengono costruite considerando due versioni aumentate dello stesso dato oppure vengono presi due campioni appartenenti alla stessa classe. Si assume che, data la somiglianza, le rappresentazioni estratte dall'encoder saranno molto simili. L'obiettivo del modello è proprio avvicinare queste due rappresentazioni nello spazio latente.

Coppia negativa: coppia di dati con semantica diversa. Generalmente le coppie vengono costruite considerando due campioni di classi diverse. E' anche possibile applicare delle trasformazioni pesanti che vanno a modificare la semantica del dato.

L'encoder dovrebbe estrarne rappresentazioni differenti e l'obiettivo del modello è allontanarle quanto più possibile nello spazio latente.

3.1 Categorie di contrastive learning

Esistono due categorie di contrastive learning, distinte in base a come vengono costruite le coppie positive e negative [8]:

Context-Instance: cerca di modellare la relazione tra le features locali di un dato e il suo contesto. Esistono due tipi di context-instance constrastile learning: *Predict Relative Position* e *Maximize Mutual Information*.

Il primo si focalizza sull'apprendimento delle posizioni relative delle caratteristiche locali del dato. Sfrutta il fatto che esistono tipi di dato in cui le varie parti sono spazialmente relazionate tra loro. Esistono vari algoritmi che utilizzano queste relazioni per la definizione di pretext task. Un esempio nel campo delle immagini è il jigsaw-puzzle, che consiste nel dividere l'immagine in blocchi, scambiarne la posizione e riuscire a ricomporre l'immagine originale.

Il secondo deriva dal concetto di informazione mutua. Modella la relazione tra due variabili e la massimizza. Un algoritmo che utilizza questo tipo di contrastive learning è AMDIM [13], il quale ottiene due campioni dall'immagine di partenza (tramite crop, cambio di colore, ...), ne usa uno per estrarre una rappresentazione del dato e l'altro per estrarre una rappresentazione del contesto.

Instance-Instance: non considera più il contesto, studia direttamente la relazione tra le rappresentazioni di dati diversi. In particolare le coppie positive vengono costruite tra versioni diverse dello stesso dato, e ogni altro dato viene considerato come negativo. E' importante che le trasformazioni applicate per ottenere le coppie positive non siano triviali: un esempio di trasformazione non adatta è la funzione identità. Se gli elementi della coppia positiva sono praticamente identici, la rete non imparerà una rappresentazione più significativa di quella che si avrebbe con comune un modello di classificazione.
Un algoritmo utilizza questo tipo di contrastive learning è SimCLR[14].

3.2 Pretext task

Un pretext task, o compito ausiliario, è un task self-supervised di fondamentale importanza per l'apprendimento di ottime rappresentazioni. Si basa sull'assunzione che la risoluzione di questo task sia benefica per la risoluzione del task principale. E' auto supervisionato poiché utilizza delle *pseudo labels*, cioè delle etichette generate automaticamente sulla base di determinati parametri ricavati dai dati stessi. Il concetto del pretext task è indipendente dal tipo di dato utilizzato e può essere applicato in qualsiasi dominio, dalle immagini alle serie temporali.

Le rappresentazioni ottenute tramite la risoluzione del task ausiliario possono essere usate per inizializzare un modello che può così essere usato per il problema principale (classificazione, anomaly detection, object detection, ...)

I pretext task producono delle versioni trasformate dei dati di input e le usano per la costruzione delle coppie positive e negative. Si suddividono in tre categorie principali [12]: trasformazioni di colore, trasformazioni geometriche, task basati sul contesto.

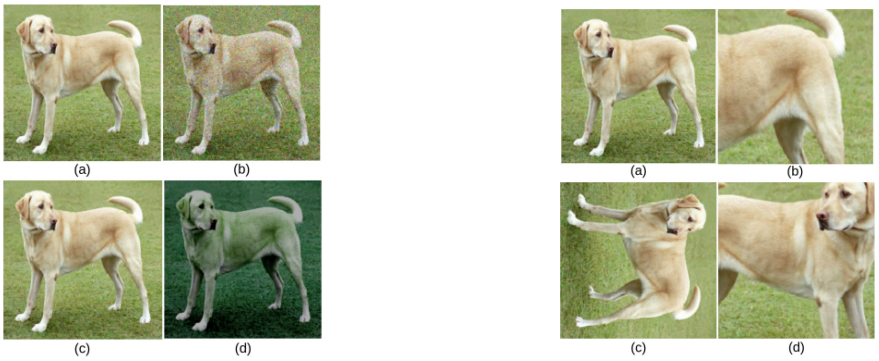


Figura 3.2: Sinistra: Trasformazioni di colore. a) originale, b) rumore gaussiano, c) sfocatura gaussiana, d) distorsione del colore
Destra: Trasformazioni geometriche. a) originale, b) ritaglio e ridimensionamento, c) rotazione, d) ritaglio, ridimensionamento, riflessione [12]

Trasformazioni di colore Sono adatte solo al dominio delle immagini. La trasformazione consiste in un'alterazione dei livelli di colore dell'immagine, introducendo sfocatura, distorsione, convertendo in scala di grigi, ecc. Questo tipo di task è indicato per modelli che devono essere invarianti ai cambiamenti di colore.

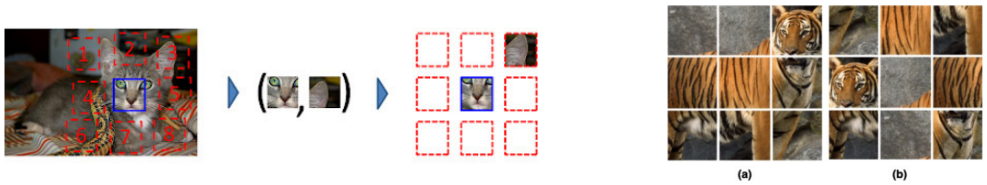


Figura 3.3: Destra: predizione della posizione relativa di due porzioni [15] dell'immagine.
Sinistra: jigsaw puzzle [12]

Trasformazioni geometriche Sono trasformazioni spaziali adatte a vari domini, purché abbiano senso in quel contesto. Nel caso delle immagini, la trasformazione non altera il valore dei pixel. L'immagine originale viene scalata, tagliata in modo casuale o riflessa. Questo tipo di task può essere indicato per modelli che devono essere invarianti all'orientamento, oppure è utile per modelli basati su context-instance contrastive learning. In quest'ultimo caso infatti si può considerare l'immagine originale come rappresentativa del contesto mentre la versione trasformata rappresenta l'istanza.

Task basati sul contesto Sono trasformazioni basate sul contesto dei dati su cui vanno applicate e sullo scopo del task secondario. Per esempio, per il problema di object detection è utile selezionare due porzioni dell'immagine e allenare il modello a predire la posizione di una porzione relativamente all'altra. Risolvendo questo task ausiliario il modello impara a osservare con attenzione diverse parti del soggetto e riuscirà probabilmente a estrarre una rappresentazione più significativa. Un altro task ausiliario simile è il *jigsaw puzzle*: il creatore di questo compito si è ispirato al gioco in cui bisogna ricomporre un immagine che è stata precedentemente suddivisa in piccoli pezzi. Infatti l'immagine originale viene suddivisa in un numero predefinito di pezzi, che vengono successivamente mescolati tra loro. Il modello impara a ricomporre l'immagine originale trovando la giusta permutazione e il giusto orientamento dei pezzi. Grazie a questo task è stato dimostrato che il modello apprende una rappresentazione molto più generale e per questo è spesso usato nei problemi di adattamento di dominio.

Tra i task ausiliari di questa categoria c'è anche quello maggiormente usato nell'ambito delle serie temporali e in generale per dati con informazioni temporali: il mascheramento. Consiste nell'omettere una porzione del segnale, il modello deve

riuscire a ricostruire la parte mancante.

La scelta del giusto pretext task non è banale e dipende dal problema da risolvere. La performance del modello dipende pesantemente da questa scelta quindi la selezione del problema ausiliario va fatta con molta attenzione. L'obiettivo principale dei task ausiliari è definire un insieme di trasformazioni alle quali si vuole che il modello sia resistente, rimanendo in grado di discriminare sulla base di altri parametri.
Utilizzando le trasformazioni sbagliate si introduce un bias dannoso per il task principale: per esempio utilizzare la rotazione per la classificazione di immagini aeree è benefico ma non lo è per la classificazione del giusto orientamento di una fotografia.

Anche la creazione di nuovi pretext task è critica perché bisogna sempre evitare che esista una soluzione triviale. Per esempio, nel task in Figura 3.3 è necessario che le diverse porzioni dell'immagine non siano contigue, deve esserci un piccolo margine tra le sezioni per evitare che il modello impari a risolvere il problema basandosi sui bordi di ogni sezione.

3.3 Metodi di contrastive learning

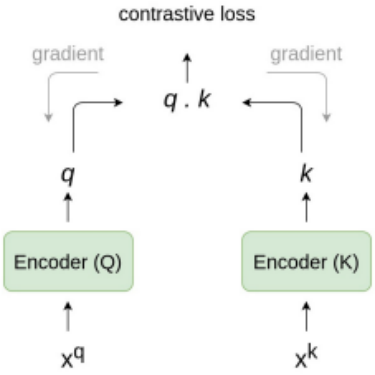


Figura 3.4: Architettura tipica per contrastive learning end-to-end [12]

In questa sezione vengono presentati tre noti metodi di contrastive learning nel campo delle immagini. Il primo, SimCLR, si occupa di classificazione, mentre gli altri si occupano di anomaly detection.

Tutti i metodi sfruttano alcuni tra i task ausiliari presentati precedentemente e fanno parte dei metodi di contrastive learning end-to-end.

I metodi end-to-end sono caratterizzati da architetture interamente differenziabili e generalmente beneficiano dall'uso di batch di grandi dimensioni in cui gli unici campioni positivi sono l'immagine originale e la sua versione trasformata, tutte le altre immagini nel batch vengono considerate come campioni negativi. L'architettura, come mostrato in Figura 3.4, ha due encoder paralleli Q e K, non necessariamente identici, nei quali vengono fatte passare le coppie positive e negative (uno dei due vede sempre le immagini originali mentre l'altro vede le versioni trasformate e i campioni negativi). L'obiettivo è che ciascun encoder estragga una rappresentazione diversa delle immagini e che alla fine grazie alla contrastive loss avvicini le rappresentazioni corrispondenti alla coppia positiva e allontani le rappresentazioni delle coppie negative.

La funzione di somiglianza utilizzata maggiormente è la *cosine similarity*, così definita:

$$cos\_sim(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^T \boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|} \tag{3.3}$$

3.3.1 SimCLR

E' un framework basato sul contrastive learning ed è stato il primo a ottenere risultati di classificazione paragonabili con lo stato dell'arte [14]. Ha dimostrato che la composizione di diverse trasformazioni ha un ruolo critico nella definizione del task ausiliario e che l'introduzione di un livello non lineare dopo i due encoder è in grado di migliorare significativamente la rappresentazione. Ottiene i risultati migliori considerando batch di grandi dimensioni e training più lunghi rispetto alla comune classificazione supervisionata.

L'architettura è composta da quattro moduli principali:

1. Un modulo che applica stocasticamente le trasformazioni alle immagini. Produce in output due versioni della stessa immagine, $\tilde{x_i}$ e $\tilde{x_j}$, che costituiscono la coppia positiva. In tutto vengono applicate tre trasformazioni consecutive: ritaglio casuale con ridimensionamento, distorsione casuale del colore e sfumatura gaussiana casuale. In particolare è stato dimostrato come la combinazione di ritaglio e distorsione del colore siano fondamentali per una buona performance.

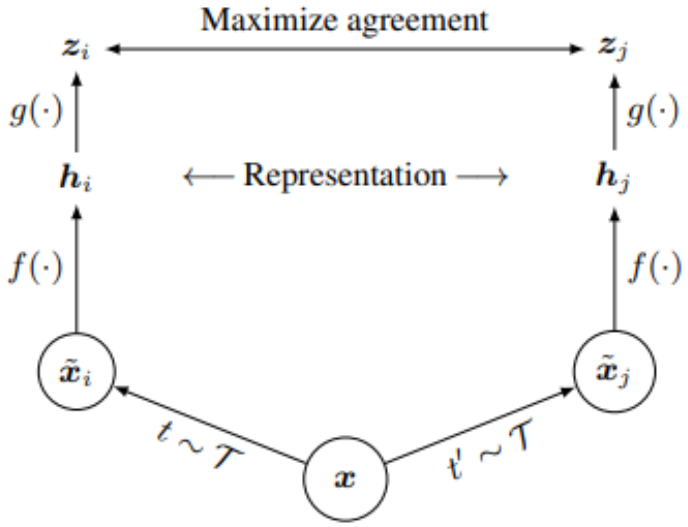


Figura 3.5: Struttura di SimCLR [14]

2. Un encoder $f()$ per estrarre la rappresentazione da $\tilde{x_i}$ e $\tilde{x_j}$. E' possibile scegliere tra diverse reti neurali, la scelta di default è ResNet.

3. Una rete neurale ridotta, $g()$, chiamata *projection head*. Mappa la rappresentazione estratta dall'encoder nello spazio in cui viene calcolata la contrastive loss. Generalmente $g()$ è un Multi Layer Perceptron con un solo livello nascosto. E' stato dimostrato che è più efficace calcolare la loss sul risultato del MLP piuttosto che sull'output dell'encoder.

4. Una contrastive loss. Viene utilizzata la infoNCE 3.2 con cosine similarity 3.3. La loss risultante viene chiamata NT-Xent Loss [16].

Per allenare il modello è necessario dividere i dati in batch di N campioni. Successivamente tutti gli elementi del batch vengono trasformati. Si ottiene così un batch finale di dimensione $2N$, che viene dato in input alla rete. I batch di grandi dimensioni migliorano le prestazioni, questo significa che i risultati sono fortemente dipendenti dalla quantità di campioni negativi. Il problema è riuscire a gestire tutti i dati senza avere problemi di memoria.

E' stata proposta una seconda versione dell'architettura in cui si affrontano alcuni problemi, tra cui appunto la quantità di memoria necessaria.

3.3.2 Contrasting Shifted Instances

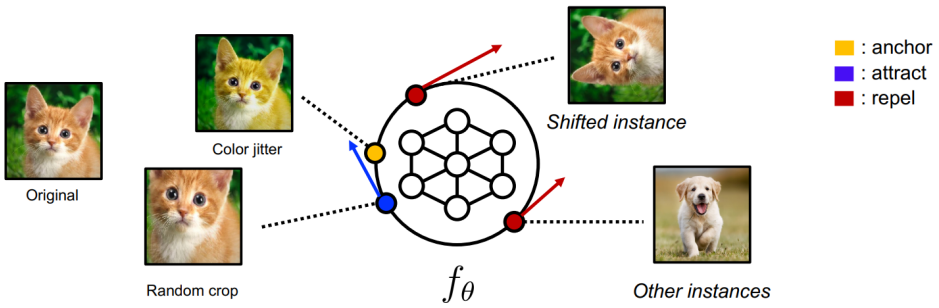


Figura 3.6: Esempio di trasformazioni adottate da CSI [17]

CSI un framework basato sul contrastive learning e affronta il problema dell'anomaly detection nel campo delle immagini [18]. E' costruito sulla base dell'architettura di SimCLR, con due modifiche:

1. Un nuovo metodo di training durante il quale ogni immagine x_i viene confrontata con delle versioni trasformate di se stessa. Alcune di queste trasformazioni sono le stesse utilizzate con SimCLR, altre invece sono trasformazioni diverse, considerate "dannose", che creano versioni dell'immagine che si discostano dalla distribuzione originale.

2. Una funzione di score per misurare il grado di anomalia delle immagini generate con le trasformazioni "dannose".

I creatori di SimCLR hanno condotto uno studio esaustivo su quali siano le trasformazioni che portano all'estrazione delle migliori rappresentazioni nel campo della classificazione delle immagini. Alcune trasformazioni sono state analizzate e scartate perché dannose per il task principale, un esempio di queste trasformazioni è la rotazione.

L'idea su cui si basa CSI è che queste trasformazioni possano essere utili per il task di anomaly detection, sfruttando le immagini a cui sono state applicate come elementi negativi con cui confrontare l'immagine originale. Viene proposto un insieme di trasformazioni $\mathcal{S}$, dette *shifting transformations*, che migliorano la rappresentazione estratta ai fini della rilevazione delle anomalie.

Per allenare il modello è necessario dividere i dati in batch di N campioni. Successivamente a tutti gli elementi del batch vengono applicate due trasformazioni: una dal nuovo insieme $\mathcal{S} = \{S_0 = \mathbb{1}, S_1, ..., S_{k-1}\}$ e una dall'insieme $\mathcal{T}$ di trasformazioni utilizzate da SimCLR. Le immagini ottenute con $S \neq \mathbb{1}$ vengono considerate anomale rispetto all'originale e sono quindi utilizzate per modellare la distribuzione diversa dalla normalità. La loss associata è la stessa NT-Xent Loss utilizzata in SimCLR

$$\mathcal{L}_{con\_CSI} = \mathcal{L}_{SimCLR}\left(\bigcup_{S\in\mathcal{S}} \mathcal{B}_S, \mathcal{T}\right), \mathcal{B}_S = \{S(x_i)\}_{i=1}^{N} \tag{3.4}$$

Viene inoltre proposto un ulteriore task ausiliario con il compito di classificare quale trasformazione $S \in \mathcal{S}$ è stata applicata. Per questa classificazione viene aggiunto un classificatore softmax ($p_{cls-CSI}(y^S|x)$) che lavora sulla rappresentazione z estratta dall'encoder $f()$. La loss associata al classificatore è

$$\mathcal{L}_{cls-CSI} = \frac{1}{2B}\frac{1}{K}\sum_{S\in\mathcal{S}}\sum_{x_S\in\mathcal{B}_S} -\log p_{cls-CSI}(y = S|x_S) \tag{3.5}$$

La loss totale è calcolata tramite la somma di 3.4 e 3.5

$$\mathcal{L}_{CSI} = \mathcal{L}_{con\_CSI} + \lambda\mathcal{L}_{cls-CSI} \tag{3.6}$$

dove λ è un iperparametro ma viene generalmente lasciato $\lambda = 1$.

Figura 3.7: Trasformazioni con score diversi [17]

Per misurare il grado di anomalia delle immagini ottenute con $S \neq \mathbb{1}$ è stata definita una funzione di score. Questa funzione lavora sulla rappresentazione z estratta dall'encoder e si basa sul calcolo della cosine similarity e della norma

$$s = \max_m sim(z(x_m), z(x))\|z(x)\| \tag{3.7}$$

dove x_m è l'immagine trasformata. Le trasformazioni con score maggiore, come la rotazione, sono le migliori per modellare il problema di anomaly detection.

Capitolo 4

Anomaly Detection

Esistono molte definizioni di cosa sia un'anomalia, quella di Grubbs sostiene che un'osservazione anomala, o outlier, è quella che sembra deviare notevolmente dagli altri membri del campione in cui si verifica [19]. Quindi i dati possono essere considerati anomali solo quando sono contestualizzati. Se ad esempio si considera il caso di una banca, dall'analisi dei dati temporali sono noti i tassi di prelievo da bancomat e l'importo medio per ogni categoria di cliente. Se in un certo momento un cliente volesse prelevare un importo dieci volte superiore alla sua media, dovrebbe poterlo fare in tranquillità dato che, supponendo di aver abbastanza credito, si tratta di un'operazione lecita. L'importo di per sé è un valore lecito ma lo scarto dal valore medio potrebbe indicare una frode. L'importo in questo caso viene considerato anomalo e dovrebbe far scattare una verifica preventiva da parte della banca per accertarsi della reale volontà del cliente, così da prevenire una truffa. Allo stesso modo, un aumento sensibile nei tassi di prelievo oppure una combinazione di questi due fattori potrebbe generare dei dati anomali che andrebbero monitorati continuamente per evitare danni all'azienda. Rilevare un'anomalia è un compito critico che serve a prevenire una perdita la cui natura dipende dal dominio di applicazione. Queste tecniche trovano un utilizzo intensivo in molti campi applicativi come rilevamento di frodi per carte di credito, applicazioni medicali, rilevamento di utenti non autorizzati in sicurezza informatica, rilevamento di guasti per la sicurezza dei sistemi industriali o sorveglianza militare per attività ostili [20, 21]. In [22] si implementa un sistema per rilevare diversi comportamenti anomali all'interno di una rete elettrica al fine di trovare pattern di consumo inusuali e capire se sono dovuti a un guasto nell'infrastruttura, un attacco

esterno o una frode di energia. Altri sistemi riguardanti i campi citati sono [23, 24, 25].

Il fatto che non esista una definizione univoca di cosa sia un outlier dipende anche dal fatto che ne esistono di vari tipi, con comportamenti diversi. Secondo la tassonomia presentata da [26] le anomalie presenti nelle serie temporali si dividono in tre categorie, mostrate in Figura 4.1: *anomalie puntuali*,*anomalie di contesto* e *anomalie collettive*.

Anomalie puntuali: sono le più facili da individuare in quanto si tratta di osservazioni puntuali (singoli dati o pochi dati consecutivi) che deviano notevolmente dal comportamento normale.

Anomalie di contesto: sono più complesse da individuare perché riguardano osservazioni considerate anormale rispetto al contesto in cui si verificano. Il singolo dato non è di per sé anomalo ma lo è in quel contesto, l'andamento dei dati in quel punto non è quello che dovrebbe essere in condizioni di normale funzionamento. Le anomalie di questo tipo dipendono dai valori circostanti quindi è necessario estrarre informazioni locali per riuscire ad individuarle.

Anomalie collettive: riguardano un sottoinsieme dei dati che presenta un comportamento anomalo rispetto al resto dei dati.

Le analisi svolte in questa tesi si concentrano sulle anomalie puntuali e di contesto.

4.1 Tecniche di rilevamento delle anomalie

Le tecniche di anomaly detection si suddividono in tre categorie in base al tipo dei dati in input [28]:

- **Supervised anomaly detection**, in cui si fa affidamento su dati già etichettati per costruire modelli predittivi. L'anomaly detection viene vista come un problema di classificazione binaria su un dataset sbilanciato in cui la classe predominante rappresenta i dati normali e l'altra quelli anomali.

- **Unsupervised anomaly detection**, in cui i dati a disposizione non sono etichettati. Viene fatta l'assunzione che solo una piccola percentuale dei dati sia composta da outlier, i dati tra loro simili vengono considerati normali mentre quelli poco frequenti e diversi sono le anomalie.

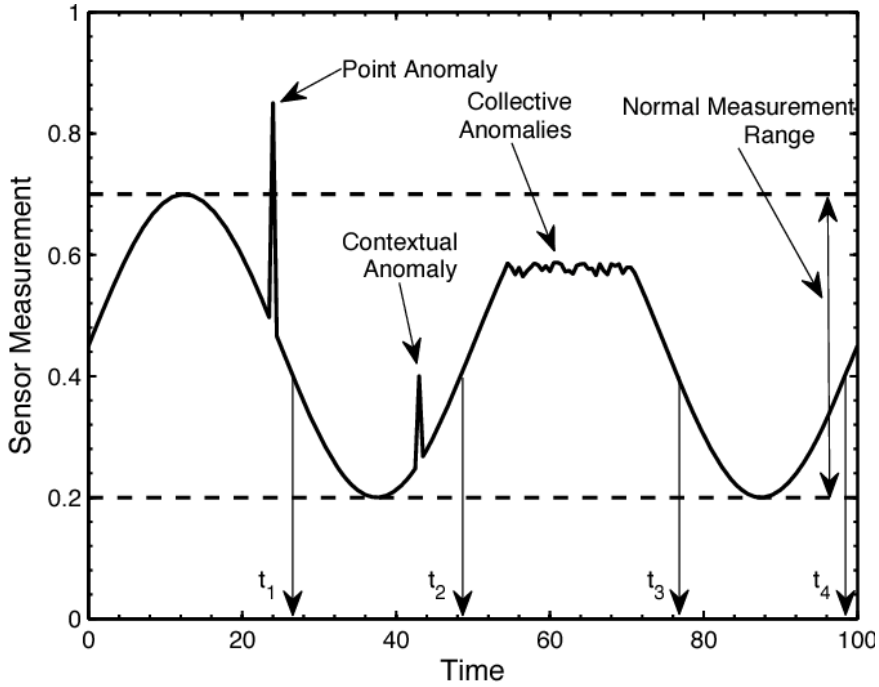


Figura 4.1: Anomalie puntuali, di contesto e collettive [27]

- **Semi-supervised anomaly detection**, in cui solo una porzione dei dati a disposizione è etichettata. Il problema di anomaly detection viene ricondotto ad un problema di classificazione con una sola classe, tipicamente rappresentante i valori normali [26].

I dataset etichettati sono in generale costosi da costruire, per questo motivo le tecniche non supervisionate sono attualmente oggetto di grande interesse e studio. Tra queste si distingue l'anomaly detection auto supervisionata (self-supervised). Come riportato nella sezione 2.4, gli algoritmi self-supervised imparano una buona rappresentazione dei dati grazie alla risoluzione di alcuni problemi ausiliari. Questi ultimi guidano il modello verso una rappresentazione più generica dei dati, che risulta utile per il problema principale di anomaly detection.

Indipendentemente dalla tecnica usata, gli algoritmi di anomaly detection individuano le anomalie in due modi:

1. Assegnazione di un punteggio: viene assegnato un punteggio a ogni dato, rappresentativo del suo grado di anormalità. Occorre scegliere una soglia

per discriminare i dati in base al punteggio. Questa soglia è tipicamente dipendente dal dominio a cui appartengono i dati e permette una gestione più flessibile del problema.

2. Classificazione binaria: a ogni dato viene assegnata un etichetta che lo individua come normale o anomalo.

4.2 Modelli di anomaly detection

Sono stati proposti numerosi modelli per il rilevamento delle anomalie, divisibili secondo [20] e [21] in almeno quattro categorie principali: modelli basati su statistica, modelli basati sulla distanza, modelli basati sul clustering, modelli di deep learning.

4.2.1 Modelli statistici

I modelli basati sulla statistica si distinguono principalmente in modelli parametrici e modelli non parametrici. Seguono il seguente principio: "Un'anomalia è un'osservazione che si sospetta essere parzialmente o totalmente irrilevante perché non generata dal modello stocastico assunto" [29]. Viene fatta l'assunzione che i dati normali[1] si trovino in regioni ad alta densità di probabilità, al contrario di quelli anomali. Le tecniche statistiche adattano un modello ai dati normali e lo usano per determinare se i nuovi dati siano outlier o meno. Un dato con una bassa probabilità di appartenere al modello viene considerano anomalo.

I modelli parametrici presuppongono di conoscere la distribuzione a cui appartengono i dati e stimano i parametri dai dati presenti. Per esempio, si assume comunemente che seguano una distribuzione Gaussiana e dai dati si stimano i valori di media e varianza tramite la Maximum Likelihood Estimate (MLE) [30]. Una volta stimate la media μ e la varianza σ^2 della distribuzione, un'osservazione viene considerata anomala se è distante più di 3σ dalla media, dove σ è la deviazione standard.

I modelli non parametrici invece non presuppongono alcuna conoscenza della distribuzione sottostante a priori, ma la stimano dai dati a disposizione.

[1]Con il termine "normale", se non altrimenti specificato, si intende descrivere un dato non anomalo.

Il problema principale dei modelli statistici è che spesso non si hanno informazioni sufficienti per poter fare le stime.

4.2.2 Modelli basati sulla distanza

Questi modelli si basano sul calcolo delle distanze tra i dati a disposizione [31]. Data una misura di distanza nello spazio di appartenenza dei dati, esistono diversi criteri per determinare se un campione sia normale o anomalo [32]:

1. gli outlier sono i dati per i quali esistono meno di p dati diversi distanti d.
2. gli outlier sono i primi n dati la cui distanza con i k dati più vicini è maggiore.
3. gli otlier sono i primi n dati la cui distanza media con i k dati più vicini è maggiore.
4. gli outlier sono i primi n dati per cui la somma delle distanze dai k dati più vicini è maggiore.

In pratica i modelli basati sulla distanza sono molto intuitivi ma trovano applicazioni limitate a causa di diversi problemi. I più evidenti sono la forte dipendenza dai parametri scelti per l'identificazione degli outlier e il fatto che il calcolo delle distanze tra tutti i punti sia molto costoso ($\mathcal{O}(N^2)$ senza ottimizzazioni). Inoltre, come riportato da [33], l'utilizzo della distanza non tiene conto di possibili differenze di densità all'interno dei dati.

4.2.3 Modelli basati sul clustering

Il clustering è una tecnica non supervisionata per raggruppare insieme dati tra loro simili. Le tecniche di anomaly detection basate su clustering si suddividono in tre categorie in base all'assunzione di partenza [20].

L'assunzione della prima categoria è che i dati normali appartengono ad un cluster mentre quelli anomali non appartengono a nessun cluster. Le tecniche basate su questa assunzione sfruttano algoritmi noti di clustering, come il DBSCAN [34], e dichiarano anomali tutti i dati che non rientrano in nessuno dei possibili cluster.

L'assunzione della seconda categoria è che i dati normali si trovano più vicini al centroide del cluster più vicino mentre le anomalie sono lontane da ogni centroide.

La distanza dal centroide più vicino viene considerata una sorta di punteggio in base a cui classificare i punti come normali o anomali. E' possibile che le anomalie stesse vengano raggruppate in un unico cluster, in questo caso nessuna delle tecniche finora discusse è in grado di discriminare i dati normali da quelli anomali.

L'assunzione della terza categoria è che i dati normali appartengono a cluster grandi e densi mentre le anomalie appartengono a cluster piccoli e sparsi. Con questa assunzione è possibile riuscire a scovare le anomalie anche nel caso appena citato. Queste tecniche si basano sulla dimensione e sulla misura della densità dei cluster, se i valori sono inferiori a quelli di soglia allora tutti gli elementi all'interno del cluster vengono classificati come anomali.

4.2.4 Modelli di deep learning

I modelli presentati finora non sono in grado di gestire le anomalie in tutte le loro sfumature. Ci sono ancora molti problemi che restano aperti, non ultimo quello dei dataset con un numero elevato di caratteristiche. In un caso del genere i metodi basati sul calcolo della distanza non sono più in grado di fornire un risultato attendibile a causa della cosiddetta *curse of dimensionality*. Inoltre molti modelli riescono a riconoscere le anomalie puntuali ma non di altri tipi. I modelli di deep learning cercano di alleviare questi problemi, e, come mostrati in Figura 4.2, si suddividono in due categorie [35]: modelli per l'estrazione delle caratteristiche e modelli per l'apprendimento di rappresentazioni della normalità.

I modelli per l'estrazione delle caratteristiche in pratica si limitano a ridurre la dimensione dei dati estraendone una rappresentazione che preserva le informazioni utili a discriminare i dati normali dai dati anomali. Sulla rappresentazione estratta viene applicata una funzione, indipendente dal passo precedente, che calcola l'anomaly score. Grazie a questo approccio è possibile avere dei modelli allo stato dell'arte pre-allenati e pronti per risolvere problemi di anomaly detection.

I modelli per l'apprendimento di rappresentazioni della normalità invece inglobano il modulo per il rilevamento delle anomalie. Tra questi modelli si distinguono gli Autoencoders (AE) e i Variational Autoencoders (VAE), verranno usati come confronto per i risultati delle analisi svolte nel corso di questa tesi e sono approfonditi nella sezione 4.3. Entrambi imparano una rappresentazione latente dei dati da cui tali dati possano essere ricostruiti. Questi modelli non nascono con lo scopo di rilevare le anomalie ma hanno trovato applicazione in questo campo grazie alla

seguente assunzione: i dati normali possono essere ricostruiti meglio a partire dalla rappresentazione estratta rispetto ai dati anomali. Quindi l'errore di ricostruzione può essere utilizzato come misura dell'anormalità dei dati.

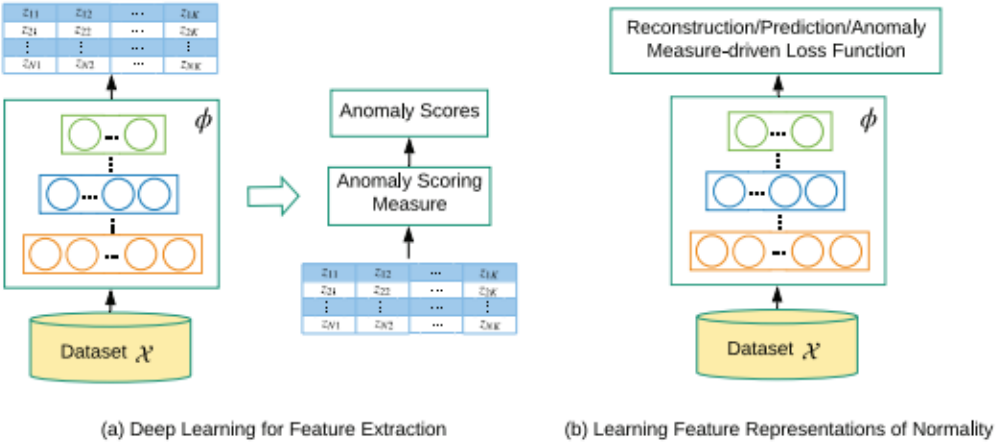


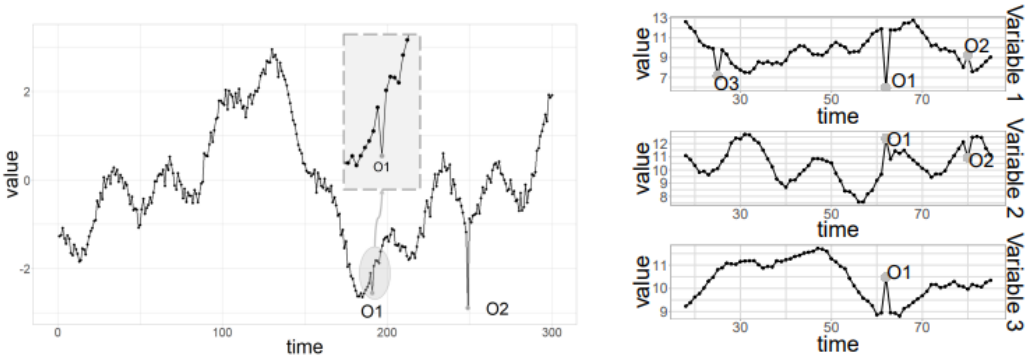
Figura 4.2: Le due categorie di modelli di deep learning [35]

4.3 Anomaly detection su serie temporali

4.3.1 Serie temporali

Le serie temporali sono composte da una sequenza di osservazioni ordinate nel tempo, ognuna associata a un timestamp. Le osservazioni non sono obbligatoriamente ottenute ad intervalli regolari e grazie al timestamp è possibile identificare l'intervallo di tempo intercorso tra due osservazioni consecutive. L'analisi delle serie temporali comporta l'estrazione di statistiche e caratteristiche significative dai dati, utili al fine di predire valori futuri a partire dai valori passati. I dati in generale possono essere di due tipi: dati strutturati e dati non strutturati. I primi sono dati che sono stati formattati in una struttura preimpostata, un esempio intuitivo sono i dati contenuti all'interno di un database relazionale. Gli ultimi invece sono dati che non sono stati elaborati e si trovano ancora nel loro stato nativo, un esempio sono le misurazioni raccolte tramite sensori IoT. Si tratta di dati che non possono essere

utilizzati direttamente, sono più flessibili, facilmente adattabili a scopi diversi in base a come vengono preparati. I dati non strutturati sono quelli più diffusi e le serie temporali rientrano in questa categoria. Quando i dati non strutturati vengono affiancati da metadati che ne facilitano l'ordinamento e la ricerca, si parla di dati semi-strutturati, un esempio sono le misurazioni salvate in un file CSV.



(a) Univariate time series. (b) Multivariate time series.

Figura 4.3: Anomalie in serie temporali univariate e multivariate [36]

Le serie temporali si suddividono in due categorie [36]: univariate e multivariate (Figura 4.3).

Serie temporale univariata: $\boldsymbol{X} = \{x_t\}_{t \in T}$ è un set ordinato di valori reali, dove ogni osservazione è registrata al tempo specifico $t \in T \subseteq \mathbb{Z}^+$.

Serie temporale multivariata: $\boldsymbol{X} = \{\boldsymbol{x}_t\}_{t \in T}$ è un set ordinato di vettori k-dimensionali, ognuno registrato al tempo specifico $t \in T \subseteq \mathbb{Z}^+$ e composto da k osservazioni reali $\boldsymbol{x}_t = (x_{1t}, ..., x_{kt})$.

Spesso si fa riferimento alle variabili con i termini *dimensioni*, *features* o *caratteristiche*. I modelli di anomaly detection che lavorano su serie temporali univariate considerano una sola dimensione dipendente dal tempo, mentre quelli che lavorano su serie multivariate sono in grado di considerare simultaneamente tutte le k dimensioni. E' possibile utilizzare un modello adatto alle serie univariate anche su serie multivariate lavorando individualmente su ognuna delle variabili. Un'analisi di questo tipo è comunque sconsigliata perché prendendo singolarmente le features

vengono trascurate le relazioni che intercorrono tra le diverse dimensioni.

4.3.2 Stato dell'arte

In questa sezione vengono descritti i modelli allo stato dell'arte nel rilevamento delle anomalie su serie temporali univariate e multivariate. Entrambe le architetture si basano sui modelli di deep learning introdotti nella sottosezione 4.2.4. Per comodità è stato usato il toolkit TODS [37], un sistema di machine learning full-stack automatizzato per l'anomaly detection su serie temporali univariate e multivariate. Il modulo dedicato alle anomalie contiene una moltitudine di modelli per risolvere problemi di anomaly detection, tra questi sono stati scelti l'Autoencoder e il Variational Autoencoder.

TODS Autoencoder

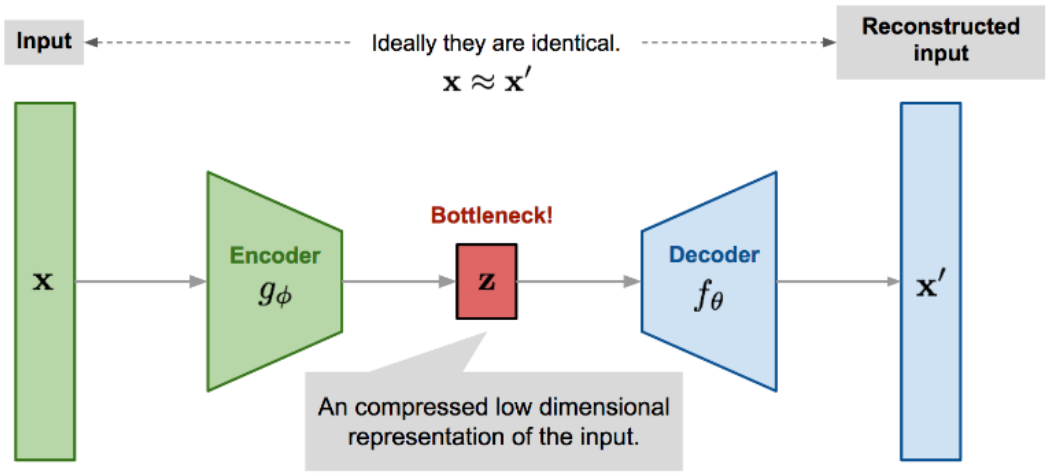


Figura 4.4: Struttura di un Autoencoder [38]

Un Autoencoder è un particolare tipo di rete neurale capace di codificare l'input in una rappresentazione più compatta e altamente significativa, e ricostruire i dati da questa rappresentazione in modo che siano il più possibile simili a quelli originali [39, 40].

Come mostrato in Figura 4.4, è composto da due elementi principali:

1. Encoder: comprime l'input ottenendone la rappresentazione compatta $\boldsymbol{z}$, spesso chiamata *bottleneck*.

2. Decoder: ricostruisce il dato a partire da $\boldsymbol{z}$.

Entrambi gli elementi sono composti da livelli convolutivi.

La funzione di compressione appresa deve essere in grado di preservare le informazioni essenziali per la ricostruzione corretta dei dati.

Per formalizzare, l'obiettivo è trovare le funzioni $g_\phi : \mathbb{R}^n \to \mathbb{R}^p$ (encoder) e $f_\theta : \mathbb{R}^p \to \mathbb{R}^n$ (decoder) che minimizzino la funzione di costo associata al problema

$$\mathbb{L}_{AE}(\phi, \theta) = \frac{1}{n} \sum_{i=1}^{n} (x_i - f_\theta(g_\phi(x_i)))^2 \tag{4.1}$$

E' importante notare che le funzioni g_ϕ e f_θ sono specifiche per la distribuzione dei dati forniti durante il training e non sono in grado di comprimere/decomprimere correttamente nuovi dati appartenenti a una distribuzione diversa. Questa proprietà si rivela particolarmente utile per il problema di anomaly detection.

Per utilizzare gli AE nel campo dell'anomaly detection occorre prima modellare il comportamento normale e successivamente generare un anomaly score per ogni nuovo dato. Il modello viene allenato esclusivamente su dati normali. Durante il test invece gli vengono proposti dati sia normali sia anomali: avendo imparato a codificare solo dati normali, l'errore di ricostruzione delle anomalie sarà grande. Confrontando l'errore con un valore di soglia è possibile distinguere efficacemente i dati anomali da quelli normali.

Il processo descritto dà ottimi risultati nell'anomaly detection ma parte dal presupposto che ci sia un dataset di training "pulito", cioè contenente solo dati normali. Dato che le anomalie si verificano in modo del tutto inaspettato, spesso non si può avere questa garanzia e si ottengono risultati al di sotto delle aspettative.

TODS Variational Autoencoder

Il concetto del Variational Autoencoder è simile a quello dell'AE ma aggiunge il concetto di probabilità. Infatti l'AE si limita a ricostruire i dati originali mentre i VAE sono nati con lo scopo di generare nuovi dati mai visti prima e simili ai dati visti durante il training.

Invece di mappare l'input in un vettore, il VAE cerca, tramite l'encoder, di mapparlo in una distribuzione latente q_ϕ. Per generare un nuovo dato appartenente alla stessa distribuzione occorre estrarre un campione $\boldsymbol{z}$ da q_ϕ e ricostruirlo grazie al decoder.

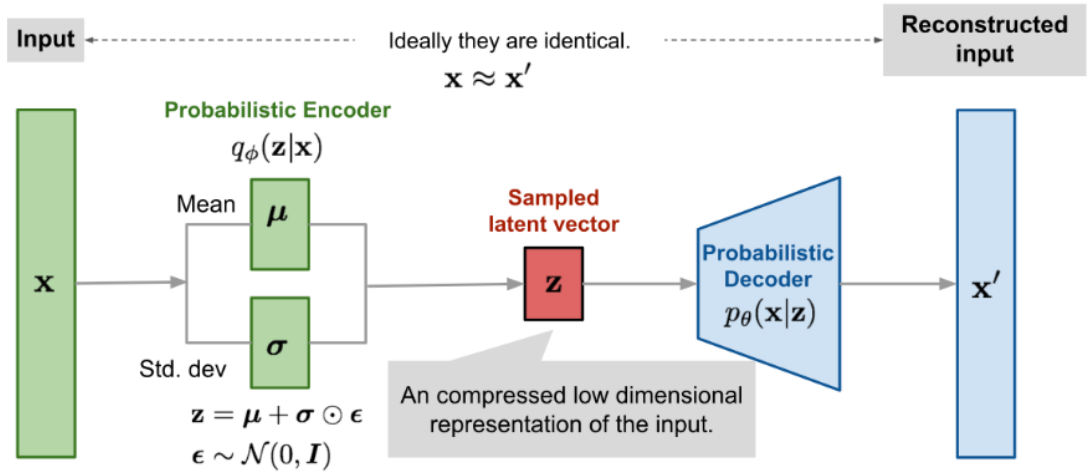


Figura 4.5: Struttura di un Variational Autoencoder [38]

L'encoder apprende i parametri della distribuzione latente q_ϕ da cui viene campionato il vettore $\boldsymbol{z}$. Il decoder invece apprende una distribuzione p_θ in grado di ricostruire il dato da $\boldsymbol{z}$ in modo che sia molto simile a quello originale.

L'intero modello viene allenato cercando di minimizzare la differenza tra la distribuzione q_ϕ e la reale distribuzione dei dati. Questa differenza viene calcolata grazie alla divergenza di Kullback-Leibler, che quantifica la distanza tra due distribuzioni misurando quanta informazione viene perduta quando una distribuzione viene usata per approssimare l'altra.

I VAE sono stati spesso usati nel campo dell'anomaly detection [41, 42, 43], per stabilire se un dato sia anomalo è possibile procedere in due modi:

1. basandosi sull'errore di ricostruzione, esattamente come avviene con gli AE.

2. calcolando la probabilità che il dato generato appartenga alla distribuzione dei dati normali. Se il dato si trova in una zona a bassa densità di probabilità, al di sotto di una certa soglia, allora viene identificato come anomalo.

I VAE sono generalmente migliori degli AE nell'identificazione delle anomalie in quanto cercano di modellare la distribuzione di probabilità della normalità mentre gli AE si limitano a codificare e decodificare staticamente i dati.

4.4 Anomaly detection basata su contrastive learning

L'anomaly detection basata su contrastive learning è attualmente oggetto di grande ricerca e la maggior parte dei modelli allo stato dell'arte, tra cui [14, 44, 18], riguarda il campo delle immagini. Per quanto riguarda le serie temporali, esistono al momento pochissimi progetti che applicano l'approccio contrastive al rilevamento delle anomalie. Per lo svolgimento di questa tesi è stato scelto il framework TS2Vec [45].

Il contrastive learning si è rivelato molto utile per distinguere le anomalie. Allenando il modello su problemi secondari, come descritto nel Capitolo 3, si ottiene una rappresentazione più generale in grado di modellare bene il comportamento dei dati normali.

4.4.1 Il framework TS2Vec

E' un framework di recente sviluppo per l'estrazione di una rappresentazione esaustiva e universale dalle serie temporali univariate e multivariate. Sono stati proposti numerosi lavori riguardanti questo problema, spesso focalizzati sull'apprendimento di rappresentazioni a livello di istanza, riassuntive dell'intero dato. Rappresentazioni di questo tipo hanno delle limitazioni poiché non sono adatte per i task che hanno bisogno di rappresentazioni "a grana fine" come l'anomaly detection. Con il termine "a grana fine" si intende il bisogno di riuscire ad analizzare i dati nel dettaglio, fino al singolo timestamp. Un altro problema degli approcci comunemente utilizzati è che si ispirano a lavori nel campo delle immagini e dell'elaborazione del linguaggio naturale. Questi domini sono basati su bias induttivi che non sono validi per le serie temporali. Un esempio immediato è l'invarianza al cropping: se il cropping viene applicato a un'immagine il soggetto non cambia ma se viene applicato a una serie temporale ne può modificare la distribuzione.

TS2Vec propone un approccio basato sul contrastive learning, capace di estrarre informazioni dai dati temporali a diversi livelli semantici. Per la costruzione delle coppie positive viene utilizzata una strategia chiamata *contextual consistency*.

Il modello TS2Vec è un modello di contrasto e la sua struttura è conforme a quanto descritto nella sezione 2.4. Come mostrato in Figura 4.6 la rete riceve in input una serie temporale e, campionandola in modo casuale, ne produce

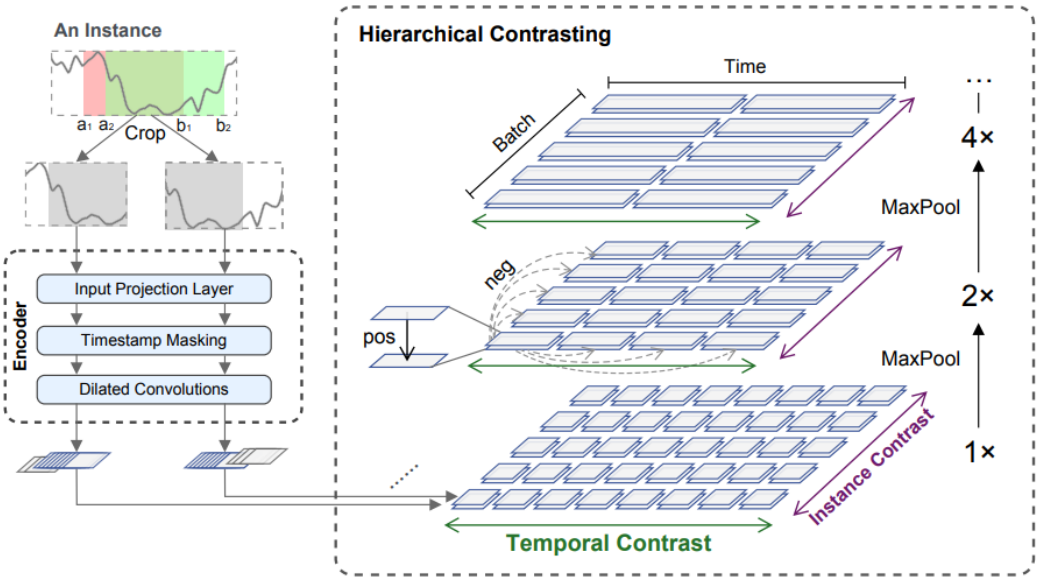


Figura 4.6: Architettura di TS2Vec [45]

due sottoserie parzialmente sovrapposte. Queste ultime vengono fornite in input all'encoder f_θ. L'encoder è composto da tre elementi:

1. Un livello di proiezione, che serve a mappare l'input in uno spazio latente con dimensionalità maggiore su cui applicare il mascheramento.

2. Un livello di mascheramento , che crea la coppia di dati per l'approccio contrastive. Ognuna delle due sottoserie riceve un maschermento su alcuni timestamp random. Come mostrato in Figura 4.7 le coppie positive sono costituite dai singoli timestamp, questo è ciò che costituisce la contextual consistency. La selezione delle giuste coppie è essenziale per il buon funzionamento del contrastive learning, altri lavori riguardanti le serie temporali hanno considerato criteri diversi di selezione che però sono basati su assunzioni troppo forti per le serie temporali e non adatti per l'anomaly detection. Ad esempio in [46] viene utilizzata la *subseries consistency*, in cui si assume che la serie non presenti mai variazioni di livello, mentre in [47] vine utilizzata la *temporal consistency*, in cui si assume che non siano presenti anomalie puntuali (Figura 4.8).

3. Un livello convolutivo, che ricava la rappresentazione latente dai dati aumentati. E' costituito da dieci blocchi, ognuno composto da due livelli convolutivi 1-D

con dilazione crescente.

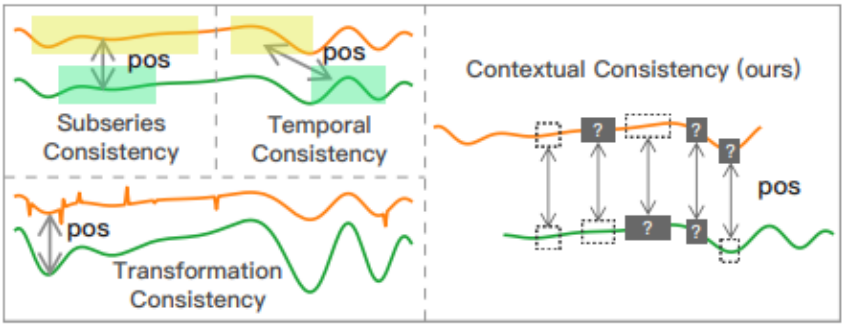


Figura 4.7: Confronto tra contextual consistency (destra) e altre tecniche contrastive (sinistra) [45]

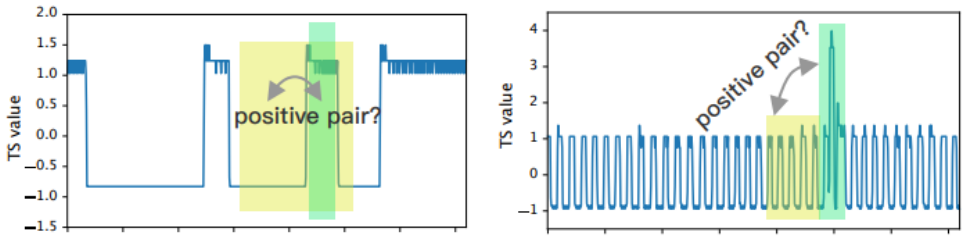


Figura 4.8: Esempi sbagliati di coppie positive per subseries consistency (sinistra) e temporal consistency (destra) [45]

Il punto di forza di TS2Vec è l'apprendimento di rappresentazioni a diversi livelli di semantica. Questo rende il modello flessibile e in grado di ottenere ottimi risultati in problemi diversi, tra cui la classificazione e l'anomaly detection. Tutto questo è possibile grazie alla definizione di una nuova contrastive loss *gerarchica* $\mathcal{L}_{hier}$, calcolata ricorsivamente sull'output dell'encoder. La rappresentazione estratta dalla convoluzione viene analizzata a diversi livelli di dettaglio, dal particolare al generale, applicando il max pooling. La loss complessiva è la media delle loss $\mathcal{L}_{hier}$ calcolate ad ogni passo.

La loss $\mathcal{L}_{hier}$ viene calcolata a partire da due contrastive loss, la loss temporale ℓ_{temp} e la loss di istanza ℓ_{inst}.

Loss temporale: date le due sottoserie ottenute dalla serie i, considera le rappresentazioni r e r' dello stesso timestamp t come positive e le rappresentazioni

di tutti gli altri timestamp $t' \neq t$ come negative (considerando l'insieme Ω dei timestamp in cui le due sottoserie sono sovrapposte).

$$\ell_{temp}^{(i,t)} = -\log \frac{\exp(r_{i,t} r'_{i,t})}{\sum_{t' \in \Omega} (\exp(r_{i,t} r'_{i,t'}) + \mathbb{1}_{t \neq t'} \exp(r_{i,t} r_{i,t'}))} \tag{4.2}$$

Loss di istanza: dato un batch B di coppie di sottoserie, considera le rappresentazioni dello stesso timestamp t degli altri elementi di B come negative.

$$\ell_{inst}^{(i,t)} = -\log \frac{\exp(r_{i,t} r'_{i,t})}{\sum_{j=1}^{B} (\exp(r_{i,t} r'_{j,t}) + \mathbb{1}_{i \neq j} \exp(r_{i,t} r_{j,t}))} \tag{4.3}$$

La somma di (4.2) e (4.3) è la loss $\mathcal{L}_{dual,h}$ per un certo livello gerarchico,

$$\mathcal{L}_{dual,h} = \frac{1}{NT} \sum_i \sum_t (\ell_{temp}^{(i,t)} + \ell_{inst}^{(i,t)}) \tag{4.4}$$

mentre la loss gerarchica totale è data dalla media delle $\mathcal{L}_{dual,h}$ su tutti i possibili livelli gerarchici.

$$\mathcal{L}_{hier} = \frac{1}{H} \sum_h \mathcal{L}_{dual,h} \tag{4.5}$$

Per risolvere problemi di anomaly detection, il modello considera delle sottoserie $(x_1, ..., x_t)$ e determina se l'ultimo punto x_t sia anomalo o meno. Se è diverso dal normale la sua rappresentazione dovrebbe mostrare una chiara differenza rispetto ai dati normali. La serie viene fatta passare nella rete due volte: la prima volta il dato in esame x_t viene mascherato, la seconda volta non viene applicato alcun mascheramento. Se la differenza tra la ricostruzione di x_t e il valore vero è superiore a una certa soglia α allora x_t viene etichettato come anomalo. Il parametro α viene calcolato considerando la media e la deviazione standard degli errori di ricostruzione.

4.4.2 Altri modelli

Come evidenziato nel Capitolo 6, TS2Vec mostra di avere difficoltà nel rilevamento delle anomalie di contesto. Per cercare di mitigare il problema sono state apportate

alcune modifiche alla rete. In particolare il primo livello di proiezione viene sostituito con una di tre architetture note nel campo della classificazione delle serie temporali [48].

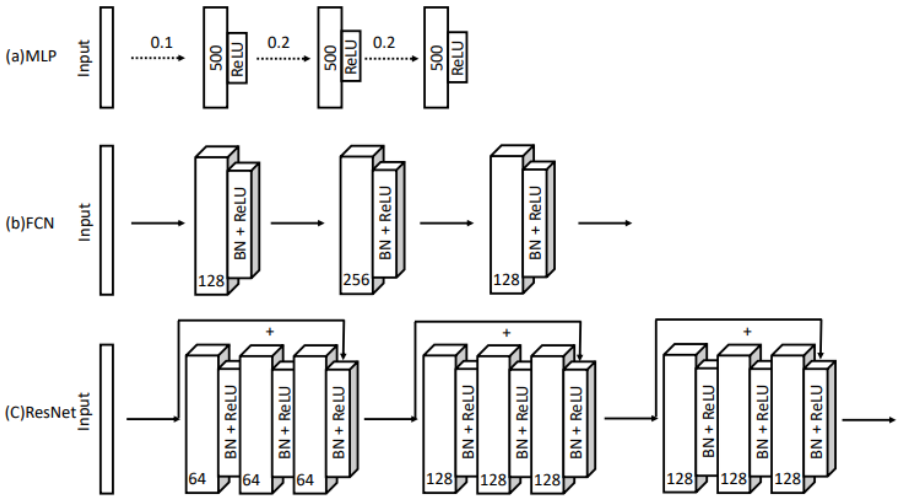


Figura 4.9: Le tre architetture utilizzate per sostituire il livello di proiezione di TS2Vec [48]

Il livello di proiezione originale è un livello fortemente connesso con il solo scopo di mappare i dati in uno spazio latente a maggiore dimensionalità. Le modifiche apportate conservano lo scopo originale ma, aumentando la profondità della rete, estraggono una rappresentazione più significativa dai dati.

Due di queste architetture utilizzano una serie di livelli convolutivi 1-D. I livelli convolutivi sono molto noti grazie ai livelli 2-D, ampiamente usati nell'ambito delle immagini perché in grado di sfruttare le informazioni spaziali in esse contenute. Al contrario, i livelli 1-D sono meno conosciuti ma permettono di sfruttare i vantaggi della convoluzione anche con dati diversi dalle immagini, come le serie temporali. Invece di estrarre le informazioni spaziali, estraggono informazioni lungo l'asse del tempo. Le informazioni temporali vengono generalmente estratte tramite reti LSTM (Long Short Term Memory) o RNN (Recursive Neural Network) ma entrambe sono soluzione onerose in termini di parametri e durata dell'allenamento. In confronto i livelli convolutivi 1-D sono molto più leggeri e veloci. Lo svantaggio è che riescono a modellare intervalli di tempo molto meno ampi rispetto alle reti citate prima.

Per questo sono oggetto di costante ricerca.

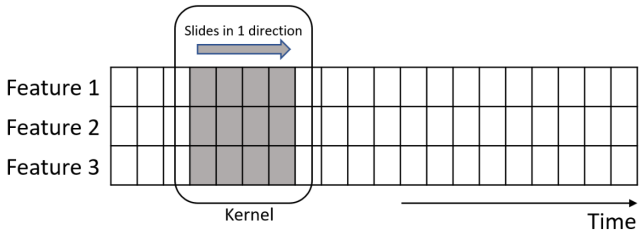


Figura 4.10: Esempio di convoluzione 1-D su una serie temporale caratterizzata da 3 features [49]

Le architetture, mostrate in Figura 4.9, sono:

MLP: è un Multilayer Perceptron composto da tre livelli fortemente connessi alternati da una funzione non lineare ReLu. Ad ogni livello viene applicato il dropout per aumentare la capacità di generalizzazione e diminuire il rischio di overfitting, la probabilità è rappresentata dai numeri associati agli archi tratteggiati.

FCN: è una rete Fully Convolutional costituita da tre blocchi. Ogni blocco è composto da un filtro convolutivo 1-D, un livello di normalizzazione e una funzione non lineare ReLu.

ResNet: è una rete composta tra tre blocchi FCN. Come la rete ResNet originale, possiede tre scorciatoie che permettono di saltare i blocchi.

Tutte e tre hanno dimostrato di essere in grado di ottenere ottimi risultati nella classificazione delle serie temporali. Le rappresentazioni estratte sono generali e robuste.

Capitolo 5

Contesto applicativo

Il corretto funzionamento dei sistemi industriali implica la disponibilità di macchine affidabili e sicure. Per questo essere in grado di prevedere e prevenire i guasti è di fondamentale importanza [50]. L'approccio classico è quello della manutenzione preventiva, secondo cui le macchine vengono revisionate periodicamente e i componenti vengono sostituiti sulla base di parametri dichiarati dai costruttori, come il tempo di vita medio. Gli svantaggi sono evidenti e spesso i guasti si verificano prima della revisione. Utilizzando invece dei sensori per monitorare i parametri delle macchine si può fare una manutenzione predittiva. Grazie alle misurazioni raccolte è possibile prevedere se nel futuro prossimo si verificherà un determinato guasto e quindi agire di conseguenza.

Le tecniche di anomaly detection presentate nel capitolo precedente vengono testate su dati CAN bus acquisiti da veicoli industriali con l'obiettivo di riconoscere segnali che indichino una possibile criticità o avaria.

5.1 Dati CAN Bus

I veicoli, dalle macchine industriali fino alle autovetture, sono dotati di molti sistemi interni ed ECUs (Electronic Control Units).

Il protocollo CAN (Controller Area Network) è un protocollo standard basato su messaggi che permette alle ECUs di comunicare tra loro senza il bisogno di un computer centrale [51]. Ogni messaggio è caratterizzato da un codice univoco SPN (Suspect Parameter Number) che identifica un certo parametro. I messaggi vengono inviati circa ogni 10 minuti ma l'intervallo può essere variabile.

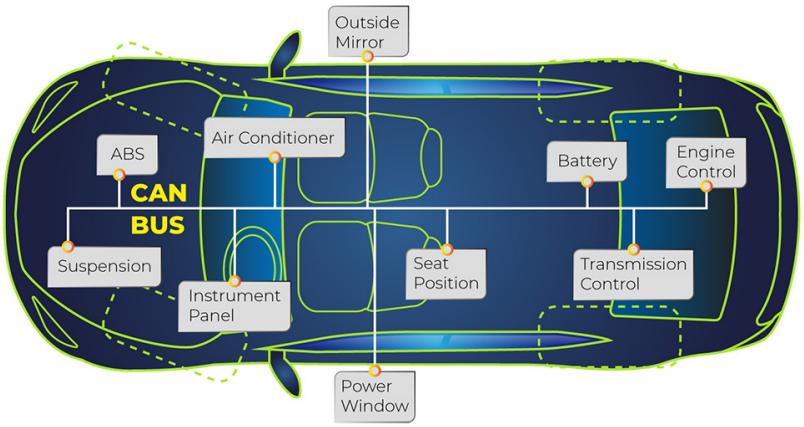


Figura 5.1: Esempio di veicolo con ECUs comunicanti tramite protocollo CAN [52]

Un aspetto molto importante del protocollo è che può diagnosticare eventuali guasti o anomalie e notificarlo tramite messaggio. Un messaggio di questo tipo è un messaggio di diagnostica, identificato con l'acronimo DM1. Ogni DM1 è caratterizzato dal parametro DTC (Diagnostic Trouble Code), il quale è a sua volta composto da due parametri: SPN e FMI (Failure Mode Identifier). Il primo serve a identificare da quale parametro è stato rilevato il guasto, il secondo indica perché il parametro identificato da SPN è considerato anomalo (voltaggio troppo alto, voltaggio troppo basso, ...). Quindi in base alla coppia (SPN, FMI) è possibile capire l'entità del guasto rilevato e se ci sia bisogno di un intervento.

Ogni volta che viene rilevato un guasto, il relativo DM1 viene inviato a un server remoto che aggiunge un record in un dataset. Il record contiene il timestamp, la coppia (SPN, FMI) e altre informazioni tecniche. Il timestamp indica l'inizio del guasto ed è paragonabile all'accensione di una spia sul cruscotto del veicolo. Quando il guasto viene risolto viene aggiunto un record nel database con il timestamp di fine e (SPN, FMI) uguali a 0.

5.2 Dataset Tierra

Il dataset, di proprietà dell'azienda Tierra S.p.A. e analizzato in [53], contiene i messaggi CAN generati da 22 veicoli industriali, tutti esemplari dello stesso modello. Il numero di messaggi generati per ogni unità è molto diverso, si va da 90 messaggi

per il veicolo meno usato a 80.000 messaggi per quello più utilizzato.

I messaggi costituiscono una serie temporale multivariata, dove ogni dimensione corrisponde a un parametro SPN. Il dataset è composto da:

- 1 colonna contenente l'identificativo univoco del veicolo che ha generato il messaggio;
- 3 colonne consecutive contenenti i timestamp di accensione e spegnimento del veicolo e il timestamp di invio del messaggio CAN;
- 1 colonna contenente un valore booleano: 'True' indica che la riga corrente corrisponde a un messaggio DM1;
- 1 colonna contenente il timestamp del prossimo DM1. E' utile per sapere tra quanto tempo avverrà il prossimo guasto;
- 22 colonne consecutive contenenti valori di parametri diversi, ognuno identificato da un SPN;
- 16 colonne contenenti le etichette (*Normal, Faulty*). Ogni colonna indica con quanto anticipo si vuole essere in grado di prevedere il guasto: gli intervalli sono multipli di 3 e vanno da 3 ore a 48 ore.

Il dataset è stato costruito con l'obiettivo di prevedere in anticipo l'invio di un messaggio di diagnostica DM1. E' stato ipotizzato che, prima che il DM1 venga inviato, si verifichi un cambiamento nei parametri CAN. Questa mutazione può avvenire in un momento imprecisato durante le ore che precedono il guasto. La creazione di diverse colonne di target serve a capire con quanto anticipo è possibile predire l'avaria. Per esempio la colonna *'30.0h'* indica che i messaggi CAN inviati nelle 30 ore precedenti a un guasto sono da considerare tutti anomali, saranno quindi etichettati come *Faulty*. Questo procedimento genera degli intervalli di anomalie consecutive, di ampiezza massima pari al numero di ore con cui si vuole riuscire a fare la previsione. L'ampiezza degli intervalli non è costante perché i messaggi CAN non sono regolari.

Poiché il numero di dimensioni del dataset è molto elevato, in [53] è stato ritenuto opportuno ridurre la dimensionalità del problema applicando tecniche di selezione delle features, tra cui SVM-RFE ed ERT.

Tutte le tecniche applicate hanno mostrato che l'unico parametro lievemente correlato con l'invio dei messaggi di diagnostica è SPN100. In particolare, anche dalla distribuzione dei valori dei vari parametri mostrata in Figura 5.2, emerge che solo SPN100 ha un comportamento che si differenzia in base alla classe. Per questo motivo è stato scelto di semplificare il problema mantenendo come unica dimensione quella relativa a SPN100.

Il dataset è molto rumoroso, per alleviare il problema durante le analisi viene considerata anche la media mobile del parametro SPN100.

In conclusione le analisi vengono svolte su un dataset multivariato con due dimensioni: SPN100, l'unico parametro selezionato tra i 22 disponibili, e la sua media mobile.

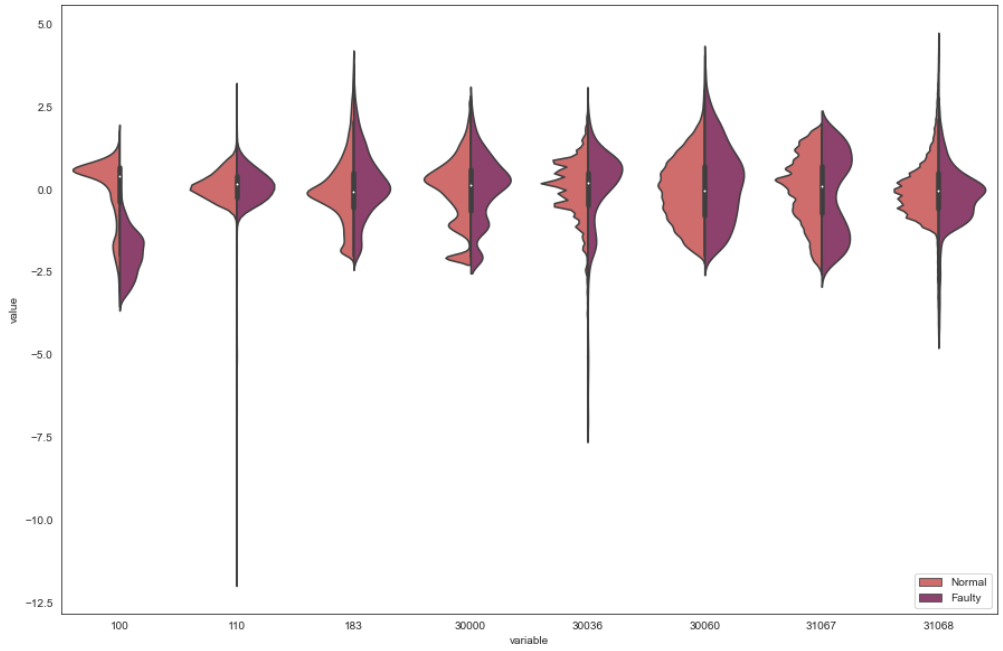


Figura 5.2: Diagramma a violino di alcuni SPN per le classi (*Normal, Faulty*) considerando un anticipo di 30 ore

Anomalie nel dataset

Per le analisi svolte nel corso di questa tesi è stata scelta la colonna di target *'30.0h'* (come consigliato in [53]), quindi l'obiettivo è riuscire a rilevare i guasti con

un anticipo massimo di 30 ore. Il dataset contiene in tutto 1391 anomalie, che corrispondono a 1,8% dei dati totali. Come ogni problema di anomaly detection è molto sbilanciato, la Figura 5.3 mostra l'andamento di SPN100 e le anomalie su un sottoinsieme dei dati. E' evidente che non si tratta di anomalie puntuali ma di anomalie di contesto: non ci sono picchi evidenti nel grafico, i valori assunti dal parametro SPN100 sono sempre leciti. Questa è una differenza fondamentale rispetto agli altri dataset utilizzati ed è probabilmente la causa dei risultati ottenuti.

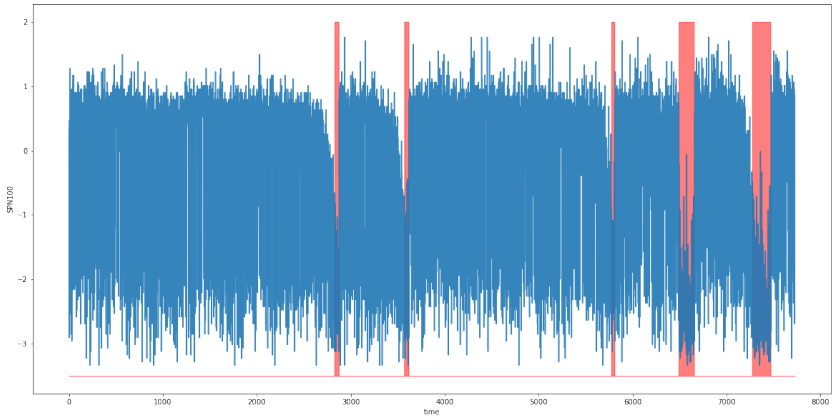


Figura 5.3: Anomalie nel dataset Tierra rispetto a SPN100

Capitolo 6

Risultati sperimentali

6.1 Metriche

In questa sezione vengono approfondite le metriche utilizzate per misurare la performance dei modelli e le metriche utilizzate per analizzare le serie temporali, sia univariate che multivariate. Prima di iniziare il training dei modelli, è stato ritenuto opportuno analizzare le caratteristiche dei dataset per comprendere meglio quali sono le relazioni tra le variabili. Tutte le misure utilizzate per questo scopo sono state selezionate tra quelle disponibili nella libreria tsfresh [54] e sono riportate nella Tabella 6.7.

6.1.1 F-score, precisione e richiamo

Per misurare la performance dei modelli è stata utilizzata la misura *F-score*. E' la scelta indicata in tutti i casi in cui le classi presenti nel dataset non sono bilanciate. Questo significa che c'è almeno una classe che prevale sulle altre in termini di numerosità. Tutti i problemi di anomaly detection sono fortemente sbilanciati, infatti le anomalie sono eventi rari e compaiono in una percentuale molto ridotta del dataset. Esistono diverse tecniche, tra cui SMOTE, Oversampling e Undersampling [55, 56], che mitigano lo sbilanciamento di un dataset aumentando il numero di campioni della classe minoritaria o diminuendo quello della classe maggioritaria. Per il rilevamento delle anomalie non è necessario applicare queste tecniche.

F-score permette di avere una valutazione della performance del modello per ogni classe, a differenza dell'accuratezza che invece considera l'insieme di tutte le

classi, ed è definito così

$$Fscore = 2 * \frac{precisione * richiamo}{precisione + richiamo} \tag{6.1}$$

Precisione e richiamo si basano sulle seguenti misure:

- TP: campioni per cui la predizione è positiva e la classe reale è positiva.
- FP: campioni per cui la predizione è positiva ma la classe reale è negativa.
- TN: campioni per cui la predizione è negativa e la classe reale è negativa.
- FN: campioni per cui la predizione è negativa ma la classe reale è positiva.

e sono definite così

$$precisione = \frac{TP}{TP + FP} \quad , \quad richiamo = \frac{TP}{TP + FN} \tag{6.2}$$

6.1.2 Analisi delle serie temporali

Tsfresh è una collezione di pacchetti compatibili con il linguaggio Python e con Pandas. Forniscono una serie di strumenti utili per eseguire l'ingegnerizzazione delle caratteristiche su dati sequenziali o serie temporali. Questi dati hanno in comune il fatto di essere ordinati secondo una variabile indipendente, (per le serie temporali è il tempo). Attraverso un processo automatico le caratteristiche estratte possono essere usate a scopo descrittivo o per algoritmi di machine lerning che lavorano su serie temporali.

Valore medio Calcola il valore medio della serie.

$$M = \frac{1}{n-1} \sum_{i=1,..,n-1} |x_{i+1} - x_i| \tag{6.3}$$

Valore massimo e minimo Calcola il valore massimo e minimo della serie.

Kurtosis Calcola l'indice di curtosi della serie. La curtosi indica un allontanamento dalla densità distributiva, dà una misura dello spessore delle code della funzione di densità. Un indice con valore > 0 indica una densità più "appuntita" rispetto alla normale, un valore < 0 indica invece una densità più "piatta" di una normale, un valore $= 0$ indica una densità simile a una normale.

Numero di picchi Stima il numero di picchi della serie tramite la Ricker wavelet con ampiezza da 1 a n.

Energia Calcola l'energia della serie.

$$E = \sum_{i=1,..,n} x_i^2 \tag{6.4}$$

Statistica C3 Calcola la non linearità della serie tramite la statistica C3 come riportato in [57].

$$E = \frac{1}{n - 2lag} \sum_{i=1}^{n-2lag} x_{i+2lag} \cdot x_{i+lag} \cdot x_i \tag{6.5}$$

Autocorrelazione media e varianza Applica una funzione di aggregazione sul risultato dell'autocorrelazione in base ad un certo lag. Come funzioni di aggregazione sono state utilizzate la media e la varianza.

$$R(l) = \frac{1}{(n-l)\sigma^2} \sum_{t=1}^{n-l} (X_t - \mu)(X_{t+l} - \mu) \tag{6.6}$$

$R(l)$ è un vettore contenente l'autocorrelazine per vari valori di lag l. Il risultato viene ottenuto applicandole una funzione di aggregazione $f_{agg}(R(1), .., R(m))$ con $m = max(n, maxlag)$, dove $maxlag$ è un parametro passato alla funzione.

Welch Density Trasforma la serie nel dominio della frequenza e successivamente calcola la potenza spettrale a diverse frequenze.

Pendenza regressione La serie temporale viene suddivisa in sequenze, per ognuna calcola una regressione lineare. E' possibile selezionare alcuni parametri della regressione, tra cui la pendenza.

CID CE Stima la complessità della serie temporale attraverso la seguente formula come descritto in [58] (una serie temporale più complessa ha molti picchi, avvallamenti, ecc).

$$\sqrt{\sum_{i=1}^{n-1} (x_i - x_{i-1})^2} \tag{6.7}$$

Entropia Calcola un'approssimazione dell'entropia della funzione di densità spettrale.

6.2 Analisi del dataset Tierra

Il dataset viene inizialmente suddiviso in due parti, una utilizzata per il training e una per il test. La porzione di training viene a sua volta suddivisa in una parte di training e una di validazione. Il dataset di validazione, nonostante riduca il numero di dati disponibili per il training, è utile per la ricerca dei parametri migliori della rete e permette di lasciare intoccato il dataset di test. Per mitigare la perdita di dati causata dal set di validazione, una volta trovata miglior combinazione di iperparametri il modello viene nuovamente allenato sull'insieme di campioni dati dall'unione dei set di training e validazione.

La Tabella 6.1 mostra le suddivisioni utilizzate rispettivamente per il training, per la validazione e per il test.

Dataset		#samples	#anomalies	%anomalies
Training	training_set	25753	747	2.9%
	validation_set	12877	76	0.6%
Test	test_set	38630	568	1.5%

Tabella 6.1: Divisione del dataset Tierra in *training_set*, *validation_set*, *test_set*. In seguito alla grid search il modello viene allenato sull'intero set di training, dato dall'unione di *training_set* e *validation_set*

TS2Vec

La rete TS2Vec non ha molti parametri di cui fare il tuning. Il parametro fondamentale, tipico di ogni rete neurale, è il *learning rate*. Quest'ultimo determina la "velocità" dell'apprendimento: la sua regolazione è critica per l'ottenimento di buoni risultati. Un learning rate troppo basso fa sì che il modello converga troppo lentamente e rimanga probabilmente bloccato in un minimo locale della funzione di loss. Al contrario un learning rate alto velocizza la convergenza ma può far sì che il modello faccia ogni volta passi troppo ampi e superi il minimo della funzione di loss. La soluzione generalmente utilizzata implica una riduzione graduale del

learning rate durante il training, partendo da un valore inizialmente più alto e riducendolo man mano che ci si avvicina alla soluzione.

Oltre al learning rate, due parametri importanti per TS2Vec sono: la dimensione dello spazio latente prodotto dal primo livello di proiezione e la dimensione della rappresentazione prodotta dall'encoder. E' interessante provare diversi valori crescenti. Infatti tali valori portano alla generazione di uno spazio a dimensione sempre maggiore, che favorisce la sparsità della rappresentazione e la scorrelazione delle caratteristiche.

Per trovare la combinazione ottimale degli iperparametri sono stati selezionati degli insiemi di valori plausibili, mostrati in Tabella 6.2, e su questi è stata eseguita una *grid search*. La grid search è una tecnica di ricerca esaustiva grazie alla quale il modello viene allenato con ogni combinazione dei valori scelti, al termine del training il modello viene testato sul dataset di validazione. Al termine viene scelta la combinazione di parametri che ha prodotto il risultato migliore durante la validazione.

Tabella 6.2: Parametri per Grid Search

Parametri	Valori
learning_rate	[0.1, 0.05, 0.01, 0.007, 0.005, 0.001]
output_proiezione	[32, 64, 128]
output_encoder	[220, 320, 420]

In Tabella 6.3 sono riportati i risultati della grid search e la relativa performance, la Figura 6.1 e la Figura 6.2 mostrano la matrice di confusione e le anomalie predette.

Modifiche a TS2Vec

Il risultato ottenuto con la rete originale presenta un ampio margine di miglioramento. Il framework è stato progettato per problemi di anomaly detection con anomalie puntuali, è quindi plausibile che con anomalie di tipo diverso i risultati siano inferiori.

Seguono tre modifiche alla rete originale in cui il primo livello di proiezione viene sostituito con una delle reti presentate nella sottosezione 4.4.2. Anche in questo caso i parametri della rete vengono scelti tramite una grid search. Per avere

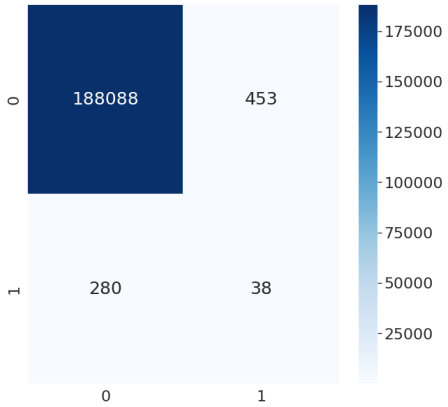


Figura 6.1: Matrice di confusione per TS2Vec originale

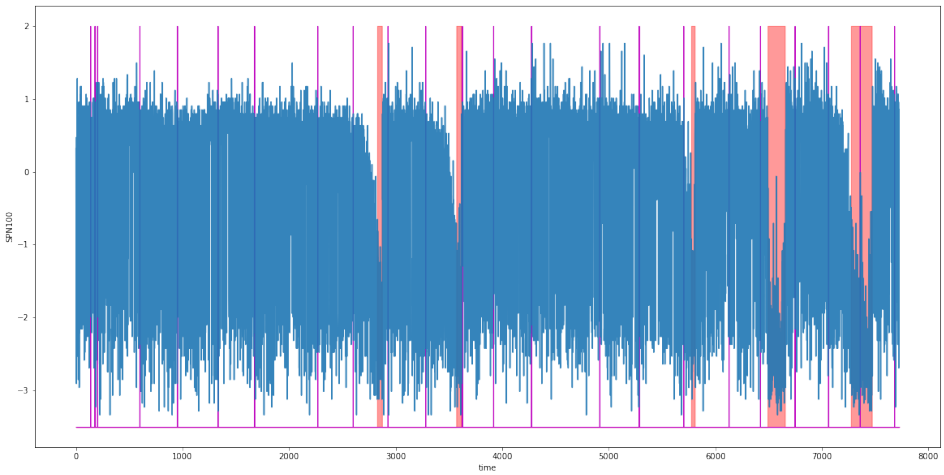


Figura 6.2: Anomalie predette da TS2Vec originale (magenta) rispetto alla predizione corretta

risultati confrontabili, la ricerca è stata fatta considerando gli stessi insiemi di valori utilizzati nel paragrafo precedente. In Tabella 6.3 sono riportati i risultati della grid search e la relativa performance in termini di F-score, precisione e richiamo.

Confronto con altri modelli

I risultati vengono confrontati con quelli ottenuti dai modelli allo stato dell'arte nel campo delle serie temporali discussi nella sezione 4.3: TODS Autoencoder e TODS Variational Autoencoder.

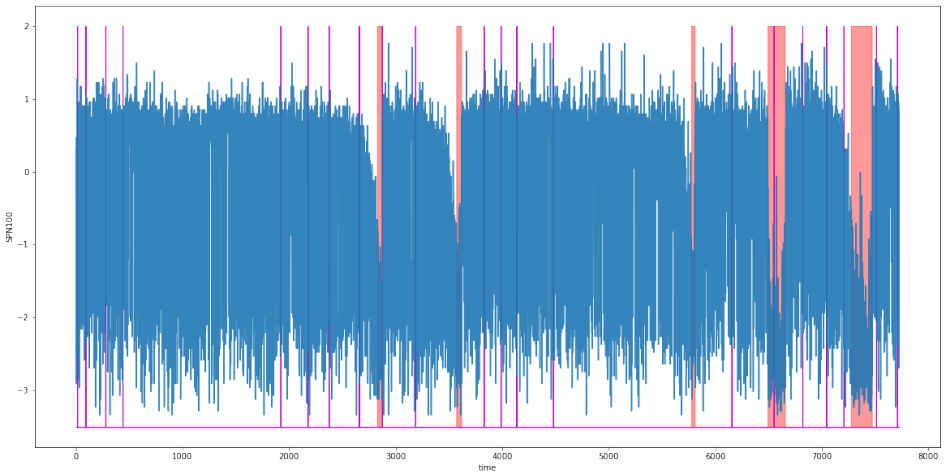


Figura 6.3: Anomalie predette dalla rete migliore TS2Vec+FCN (magenta) rispetto alla predizione corretta

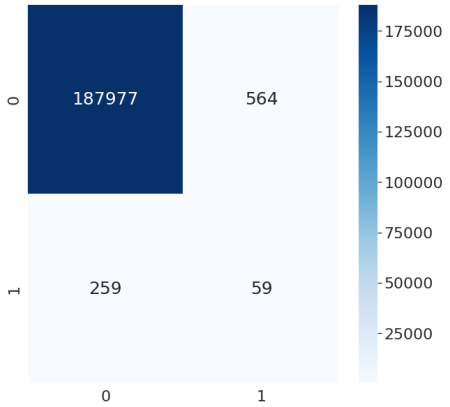


Figura 6.4: Matrice di confusione per TS2Vec+FCN

Con questi modelli si ottengono i risultati migliori, anche se molto al di sotto delle aspettative. Occorre tenere presente gli Autoencoder sono molto sensibili al rumore. Poiché il dataset Tierra è molto rumoroso, è possibile che l'uso della moving average non sia sufficiente a risolvere il problema e questa potrebbe essere una delle cause dei risultati.

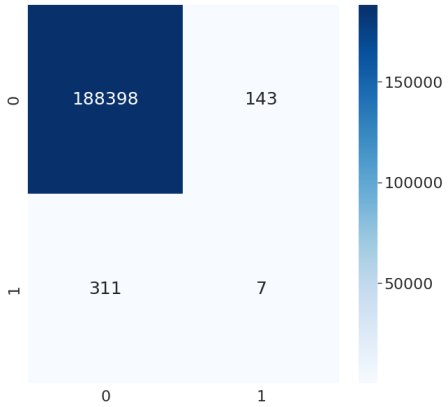


Figura 6.5: Matrice di confusione per TS2Vec+MLP

Tabella 6.3: Risultati su dataset Tierra

Parametri	TS2Vec	FCN	ResNet	MLP	TODS AE	TODS VAE
learning_rate	0.001	0.1	0.01	0.001	-	-
output_proiezione	64	128	128	64	-	-
output_encoder	320	420	220	320	-	-
F-score	0.09	**0.12**	0.09	0.03	**0.13**	0.12
Precisione	0.07	0.09	0.07	0.05	0.08	0.07
Richiamo	0.12	0.18	0.12	0.02	0.35	0.42

6.3 Analisi dei dataset di benchmark

La rete è stata testata anche su tre dataset di benchmark. La divisione in training e test per questi dataset viene già fornita: vengono semplicemente divisi a metà, la prima parte viene usata per il training e la seconda per il test. La Tabella 6.4 riassume le caratteristiche, i risultati sono nella Tabella 6.5.

6.3.1 Yahoo

E' una raccolta di serie temporali univariate rilasciata da Yahoo [59]. Una parte dei dati è sintetica, cioè ottenuta tramite simulazione, mentre la restante parte è stata ricavata dal traffico reale dei servizi Yahoo. Tutti i dati, reali e sintetici, sono etichettati. I dati simulati contengono serie temporali generate algoritmicamente con trend e rumore variabili.

Dataset	training_set	%anomalie_train	#features
Yahoo	286098	0.47%	1
KPI	2961439	1.68%	1
SynCAN	111385 (avg)	17.69% (avg)	[1, 2, 3, 4]

Tabella 6.4: Caratteristiche dei dataset Yahoo, KPI e SynCAN. I valori riportati per SynCAN sono ottenuti calcolando la media tra i 50 dataset disponibili

	Yahoo			KPI			SynCAN		
Rete	F-score	Prec.	Rich.	F-score	Prec.	Rich.	F-score (avg)	Prec. (avg)	Rich. (avg)
TS2Vec	**0.74**	0.73	0.76	0.67	0.93	0.53	0.05	0.34	0.03
FCN	0.5	0.62	0.42	0.67	0.85	0.55	0.04	0.4	0.02
ResNet	0.49	0.59	0.42	**0.74**	0.85	0.65	0.04	0.27	0.02
MLP	0.26	0.06	0.17	0.06	0.24	0.04	0.04	0.36	0.02
TODS AE	0.71	0.73	0.69	0.65	0.64	0.66	**0.09**	0.35	0.05
TODS VAE	0.72	0.71	0.73	0.64	0.64	0.64	0.1	0.4	0.06

Tabella 6.5: Risultati su dataset di benchmark

Le anomalie presenti in questo dataset sono di tipo puntuale. Come mostrato in Figura 6.6, i dati anomali si distinguono facilmente da quelli normali e corrispondono a un evento molto limitato nel tempo, ad esempio un sovraccarico momentaneo su un certo servizio.

6.3.2 KPI

E' una raccolta di serie temporali univariate rilasciata da AIOPS [60]. Le KPI, Key Performance Indicators, sono metriche che misurano le prestazioni nel tempo di una azienda. Il dataset consiste di numerose curve KPI con anomalie etichettate. I dati provengono da diverse Internet Companies tra cui eBay.

Come mostrato in Figura 6.7, anche questo dataset presenta solamente anomalie puntuali, facilmente distinguibili dal resto.

6.3.3 SynCAN

E' una raccolta di dataset contenenti serie temporali sintetiche univariate e multivariate [51]. E' un dataset di benchmark utilizzato per sistemi di Intrusion Detection su dati di tipo CAN Bus. I dataset di training forniti non contengono anomalie. Poiché per l'allenamento della rete TS2Vec è necessario un dataset con anomalie

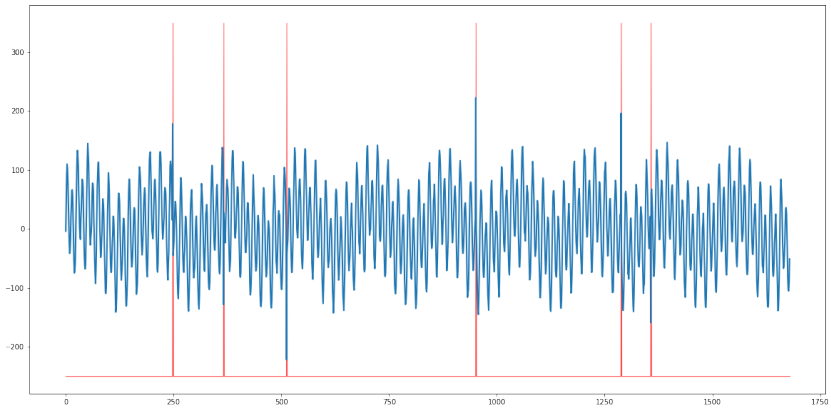


Figura 6.6: Anomalie nel dataset Yahoo

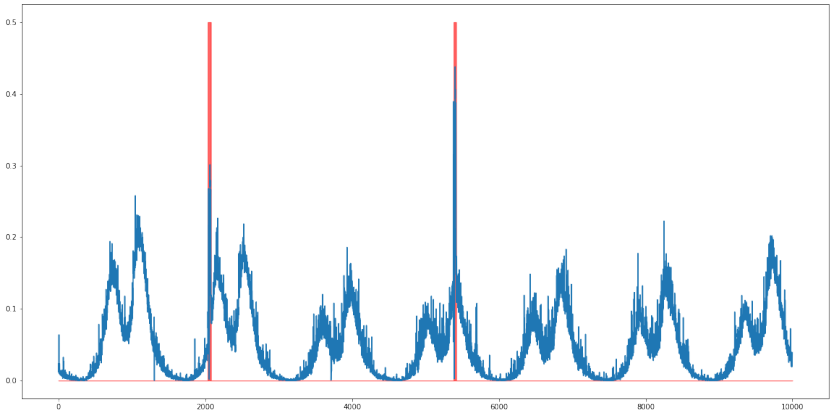


Figura 6.7: Anomalie nel dataset KPI

etichettate, è stato deciso di lavorare solo sui dataset di test. Questi ultimi sono sufficientemente grandi da essere divisi a metà in training set e test set.

Sono presenti cinque tipologie diverse di anomalie etichettate automaticamente.

Tra i dataset di benchmark, questo è il più simile al caso di studio dato dal dataset Tierra. Presenta cinque tipi di anomalie corrispondenti ad altrettanti tipi di attacchi informatici (Figura 6.8):

1. *Plateau Attack*: un intervallo di durata variabile viene sovrascritto con un valore costante.

2. *Continuous Change Attack*: un intervallo di durata variabile viene sovrascritto in modo che assuma valori linearmente crescenti nel tempo. Il segnale originale viene modificato poco alla volta.

3. *Playback Attack*: un intervallo di durata variabile viene sostituito con dei valori passati.

4. *Flooding Attack*: un intervallo del segnale presenta una frequenza molto più alta.

5. *Suppress Attack*: una delle caratteristiche del segnale viene soppressa per un intervallo variabile di tempo.

La maggior parte di queste anomalie rientra nella categoria delle anomalie di contesto, i segnali non contengono valori anormali ma presentano vari intervalli in cui il comportamento è differente da quello atteso.

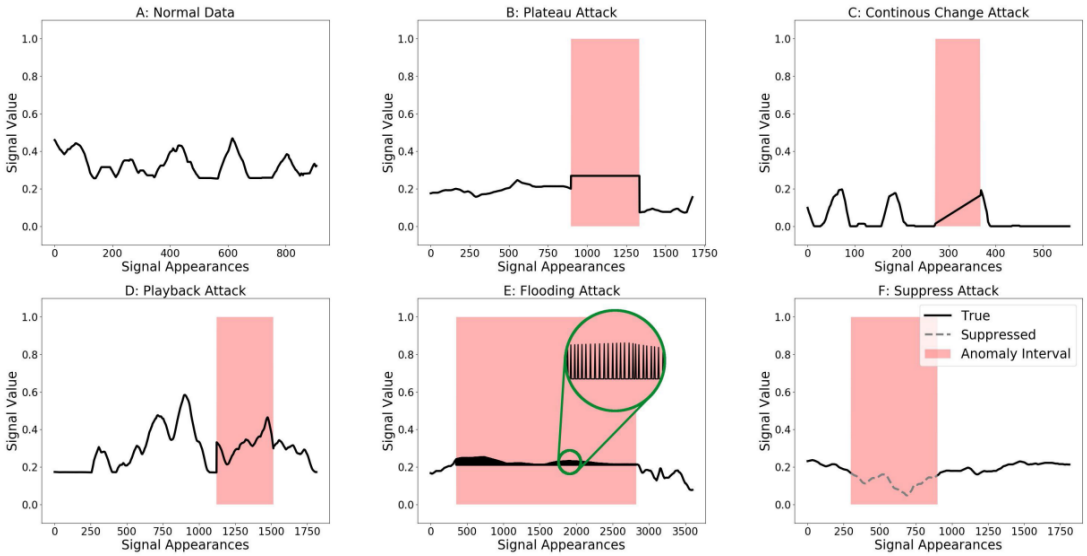


Figura 6.8: Anomalie nel dataset SynCAN [51]

6.4 Risultati complessivi

Nella Tabella 6.6 sono contenuti i risultati F-score di ogni modello, lo skew tra le varianti proposte e il modello base (la differenza tra la performance del modello proposto e quella di TS2Vec) e le caratteristiche di ogni dataset. La Tabella 6.7 prosegue con ulteriori caratteristiche per ogni dataset, calcolate con le metriche citate nella sottosezione 6.1.2.

La rete TS2Vec dimostra di essere in grado di rilevare efficacemente le anomalie di tipo puntuale ma ha difficoltà su dataset contenenti principalmente anomalie di contesto (Tierra e SynCAN). Nonostante l'architettura utilizzi una loss gerarchica, non è in grado di cogliere le caratteristiche più rappresentative del contesto.

La modifica al livello iniziale di proiezione è stata fatta con l'obiettivo di estrarre una rappresentazione migliore dai dati originali, non ancora mascherati. La rete che ha prodotto il risultato migliore è FCN, composta da una serie di livelli convolutivi 1-D. Aumentando la profondità della rete il modello diventa in grado di cogliere più informazioni rilevanti riguardo al contesto, guadagnando 3 punti percentuali rispetto all'architettura base.

Come citato precedentemente, i livelli convolutivi 1-D sono indicati per serie temporali e in generale per dati sequenziali perché costituiscono una valida alternativa alle reti LSTM. Nonostante questo, non possono avere la capacità di memoria di una LSTM e i risultati lo dimostrano.

La rete ResNet aumenta ulteriormente la profondità della rete ma non apporta nessun beneficio in termini di F-score, precisione e richiamo. Questo dimostra che aumentare semplicemente la profondità della rete non è la soluzione corretta per modellare meglio il contesto al fine di riconoscere le anomalie.

La rete MLP deteriora le performance. Nonostante abbia dimostrato di essere in grado di ottenere risultati comparabili con lo stato dell'arte nella classificazione delle serie temporali, risulta inadatta per l'anomaly detection. Non è un risultato inaspettato in quanto, non possedendo alcun livello convolutivo, non si tratta di una rete adatta all'apprendimento di un contesto.

Le analisi sui dataset di benchmark Yahoo e KPI hanno avuto esiti discordanti: per il primo il risultato migliore si ha con l'architettura base, mentre per il secondo si verifica un miglioramento con la rete ResNet. Per comprendere meglio i risultati e cercare di giustificarli, ogni dataset è stato analizzato a fondo, applicando le misure descritte all'inizio del capitolo. Infine è stata misurata la correlazione dei parametri calcolati (media, autocorrelazione, entropia, numero di picchi nella serie, ...) con lo scarto tra le performance delle reti proposte (TS2Vec+FCN, TS2Vec+ResNet, TS2Vec+MLP) e la rete originale. Il risultato è riportato nella Figura 6.9, non sono presenti forti correlazioni che portino a una chiara giustificazione dei risultati ottenuti.

L'approccio contrastive, nonostante si sia dimostrato efficace nella classificazione delle serie temporali multivariate, produce risultati inferiori alle attese nel riconoscimento di anomalie in contesti reali. In particolare, le anomalie riconosciute correttamente sono principalmente dovute alla presenza di valori fuori scala, facilmente riconoscibili, causati probabilmente da errori di trasmissione o di registrazione. Le anomalie causate da variazioni di periodicità o forme d'onda, che risultano essere maggioritarie nel contesto reale analizzato, non vengono riconosciute correttamente.

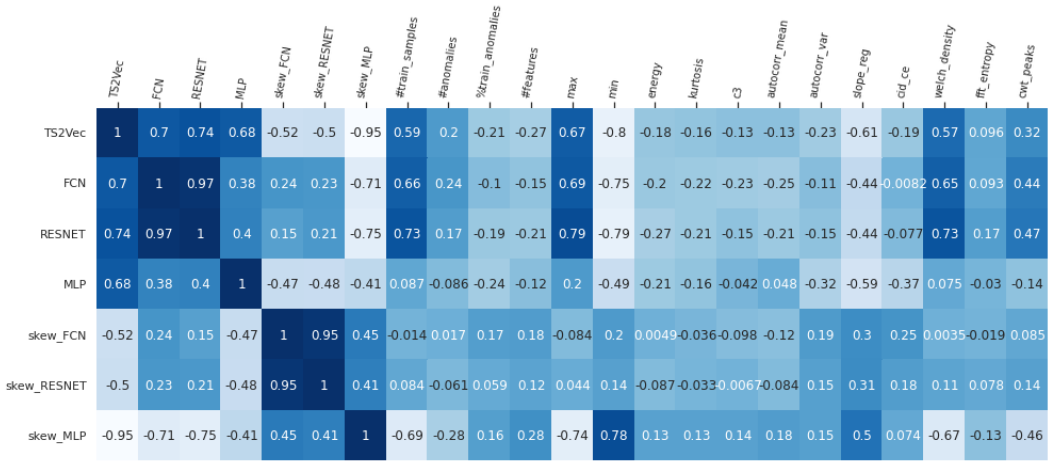


Figura 6.9: Tabella delle correlazioni tra lo scarto tra le performance della rete originale e le reti proposte (skew) e caratteristiche dei dataset

Tabella 6.6: Risultati per ogni dataset

Name	TS2Vec	FCN	RESNET	MLP	skew_FCN	skew_RESNET	skew_MLP	#train_samples	#anomalies	%train_anomalies	#features
yahoo	0.74	0.5	0.49	0.26	-24.0	-25.0	-48.0	286098	1338	0.47	1
kpi	0.67	0.67	0.74	0.06	0.0	7.0	-61.0	2961439	49779	1.68	1
tierra_oil_30	0.09	0.12	0.09	0.03	3	0	-6	38630	823	2.13	2
SynCAN_1_flooding	0.0	0.0	0.0	0.0	0.0	0.0	0.0	171413	47429	27.67	2
SynCAN_2_flooding	0.0	0.42	0.42	0.04	42.0	42.0	4.0	81407	19432	23.87	3
SynCAN_3_flooding	0.0	0.13	0.0	0.0	13.0	0.0	0.0	194962	76758	39.37	2
SynCAN_4_flooding	0.01	0.0	0.0	0.0	-1.0	-1.0	-1.0	63520	19684	30.99	1
SynCAN_5_flooding	0.38	0.11	0.11	0.11	-27.0	-27.0	-27.0	193457	72331	37.39	2
SynCAN_6_flooding	0.0	0.39	0.26	0.0	39.0	26.0	0.0	90413	30936	34.22	2
SynCAN_7_flooding	0.0	0.01	0.0	0.0	1.0	0.0	0.0	171113	42044	24.57	2
SynCAN_8_flooding	0.21	0.11	0.11	0.13	-10.0	-10.0	-8.0	169592	45654	26.92	1
SynCAN_9_flooding	0.21	0.0	0.0	0.0	-21.0	-21.0	-21.0	95561	35571	37.22	1
SynCAN_10_flooding	0.0	0.0	0.0	0.0	0.0	0.0	0.0	56460	11104	19.67	4
SynCAN_1_plateau	0.0	0.04	0.0	0.04	4.0	0.0	4.0	149998	24474	16.32	2
SynCAN_2_plateau	0.11	0.09	0.03	0.06	-2.0	-8.0	-5.0	75008	12241	16.32	3
SynCAN_3_plateau	0.0	0.03	0.0	0.06	3.0	0.0	6.0	149999	24477	16.32	2
SynCAN_4_plateau	0.08	0.06	0.06	0.03	-2.0	-2.0	-5.0	50005	8158	16.31	1
SynCAN_5_plateau	0.27	0.09	0.06	0.13	-18.0	-21.0	-14.0	149997	24476	16.32	2
SynCAN_6_plateau	0.06	0.0	0.0	0.09	-6.0	-6.0	3.0	75008	12240	16.32	2
SynCAN_7_plateau	0.0	0.1	0.0	0.0	10.0	0.0	0.0	149998	24476	16.32	2
SynCAN_8_plateau	0.2	0.18	0.15	0.02	-2.0	-5.0	-18.0	149999	24475	16.32	1
SynCAN_9_plateau	0.06	0.0	0.04	0.03	-6.0	-2.0	-3.0	75007	12242	16.32	1
SynCAN_10_plateau	0.07	0.03	0.0	0.03	-4.0	-7.0	-4.0	50005	8157	16.31	4
SynCAN_1_continuous	0.0	0.0	0.0	0.0	0.0	0.0	0.0	149999	20017	13.34	2
SynCAN_2_continuous	0.04	0.0	0.0	0.0	-4.0	-4.0	-4.0	75008	10006	13.34	3
SynCAN_3_continuous	0.0	0.0	0.0	0.0	0.0	0.0	0.0	149998	20013	13.34	2
SynCAN_4_continuous	0.0	0.0	0.0	0.0	0.0	0.0	0.0	50005	6671	13.34	1
SynCAN_5_continuous	0.0	0.0	0.0	0.0	0.0	0.0	0.0	149996	20015	13.34	2
SynCAN_6_continuous	0.0	0.03	0.0	0.0	3.0	0.0	0.0	75008	10006	13.34	2
SynCAN_7_continuous	0.0	0.0	0.0	0.0	0.0	0.0	0.0	149998	20013	13.34	2
SynCAN_8_continuous	0.0	0.0	0.0	0.0	0.0	0.0	0.0	149998	20014	13.34	1
SynCAN_9_continuous	0.04	0.0	0.05	0.05	-4.0	1.0	1.0	75008	10007	13.34	1
SynCAN_10_continuous	0.0	0.0	0.0	0.11	0.0	0.0	11.0	50005	6673	13.34	4
SynCAN_1_playback	0.0	0.0	0.0	0.0	0.0	0.0	0.0	149997	18580	12.39	2
SynCAN_2_playback	0.2	0.02	0.0	0.08	-18.0	-20.0	-12.0	75007	9290	12.39	3
SynCAN_3_playback	0.0	0.03	0.0	0.0	3.0	0.0	0.0	149998	18583	12.39	2
SynCAN_4_playback	0.08	0.04	0.04	0.11	-4.0	-4.0	3.0	50005	6193	12.38	1
SynCAN_5_playback	0.25	0.1	0.05	0.1	-15.0	-20.0	-15.0	149999	18581	12.39	2
SynCAN_6_playback	0.0	0.0	0.0	0.05	0.0	0.0	5.0	75007	9291	12.39	2
SynCAN_7_playback	0.06	0.0	0.0	0.0	-6.0	-6.0	-6.0	149998	18581	12.39	2
SynCAN_8_playback	0.15	0.08	0.07	0.15	-7.0	-8.0	0.0	149999	18578	12.39	1
SynCAN_9_playback	0.11	0.05	0.05	0.11	-6.0	-6.0	0.0	75008	9290	12.39	1
SynCAN_10_playback	0.0	0.0	0.0	0.04	0.0	0.0	4.0	50005	6193	12.38	4
SynCAN_1_suppress	0.0	0.0	0.0	0.0	0.0	0.0	0.0	147969	24135	16.31	2
SynCAN_2_suppress	0.0	0.0	0.0	0.0	0.0	0.0	0.0	73496	11580	15.76	3
SynCAN_3_suppress	0.0	0.04	0.0	0.0	4.0	0.0	0.0	147160	23334	15.86	2
SynCAN_4_suppress	0.0	0.0	0.03	0.0	0.0	3.0	0.0	49388	8059	16.32	1
SynCAN_5_suppress	0.0	0.04	0.0	0.06	4.0	0.0	6.0	148138	25191	17.01	2
SynCAN_6_suppress	0.0	0.02	0.0	0.0	2.0	0.0	0.0	73771	12489	16.93	2
SynCAN_7_suppress	0.0	0.0	0.0	0.0	0.0	0.0	0.0	147036	24998	17.0	2
SynCAN_8_suppress	0.0	0.0	0.04	0.0	0.0	4.0	0.0	146923	23501	16.0	1
SynCAN_9_suppress	0.0	0.0	0.0	0.04	0.0	0.0	4.0	73775	11389	15.44	1
SynCAN_10_suppress	0.0	0.0	0.0	0.04	0.0	0.0	4.0	48665	7608	15.63	4

Tabella 6.7: Caratteristiche dei dataset

Name	max	min	energy	kurtosis	c3	autocorr_mean	autocorr_var	slope_reg	cid_ce	welch_density	fft_entropy	cwt_peaks
yahoo	25.3594	-18.772	779.5586	35.37	-0.0162	-0.0276	0.0349	-0.0036	46.1165	1.1857	0.2445	16011.9482
kpi	99.5275	-15.787	51059.2931	9.221	0.0643	0.2896	0.0573	-0.0	214.305	27.6528	0.3739	98278.7931
tierra_oil_36	1.9663	-3.7761	38630.0	13.0596	-0.2805	0.4926	0.0039	0.0003	147.3232	2.1297	0.5477	5003.5
SynCAN_1_flooding	1.2785	-2.2513	171413.0	17.2646	0.1215	0.4903	0.0804	-0.0	314.2224	0.0139	0.2136	32036.0
SynCAN_2_flooding	2.0457	-1.0037	81407.0	4.4311	0.5361	0.9889	0.0	-0.0	34.5429	0.0013	0.159	2095.0
SynCAN_3_flooding	0.7382	-2.1246	194962.0	14.9274	-0.0326	0.5848	0.0438	-0.0	335.7775	0.0164	0.1812	14273.5
SynCAN_4_flooding	6.3526	-0.8725	63520.0	4.0066	0.9568	0.943	0.001	0.0	49.375	0.5652	0.4852	607.0
SynCAN_5_flooding	1.8044	-1.5355	193457.0	7.4149	0.6811	0.9574	0.0	0.0	126.6999	0.0044	0.1684	4750.5
SynCAN_6_flooding	1.2445	-1.9297	90413.0	11.0376	0.2352	0.5933	0.0292	0.0001	168.4436	0.0217	0.191	7657.5
SynCAN_7_flooding	2.1957	-0.8558	171113.0	18.3124	-0.1314	0.4989	0.0768	0.0	316.5759	0.0107	0.1812	35695.0
SynCAN_8_flooding	6.9135	-2.4436	169592.0	4.2466	-0.6456	0.9848	0.0	0.0	70.8415	0.007	0.2264	1697.0
SynCAN_9_flooding	5.0154	-1.2556	95561.0	3.7306	0.9285	0.9561	0.0002	-0.0	72.2882	0.0913	0.2264	946.0
SynCAN_10_flooding	2.1688	-1.9221	56460.0	11.3769	0.6092	0.7282	0.0386	0.0	88.3928	0.0126	0.2011	3596.5
SynCAN_1_plateau	1.5356	-2.1012	149998.0	14.535	0.1567	0.4495	0.1029	0.0	273.9406	0.0015	0.1812	13280.0
SynCAN_2_plateau	2.4243	-1.0519	75008.0	11.1939	0.5465	0.9943	0.0	0.0	9.523	0.0077	0.1626	1898.0
SynCAN_3_plateau	0.8079	-2.122	149999.0	10.0539	0.3903	0.4617	0.1082	0.0	259.9516	0.0037	0.1758	15417.0
SynCAN_4_plateau	8.6865	-0.7914	50005.0	17.0237	1.4614	0.9588	0.001	0.0	14.172	0.494	0.4512	323.0
SynCAN_5_plateau	1.5599	-1.483	149997.0	15.8987	0.4837	0.997	0.0	0.0	7.7855	0.0018	0.1306	3266.0
SynCAN_6_plateau	1.1812	-2.0187	75008.0	9.7929	0.4247	0.4521	0.061	-0.0	144.5659	0.0063	0.191	7291.0
SynCAN_7_plateau	2.1239	-0.8953	149998.0	21.0632	-0.2012	0.4522	0.1014	0.0	277.1099	0.003	0.1812	28062.0
SynCAN_8_plateau	6.9419	-2.3661	149999.0	18.6204	-0.5336	0.9983	0.0	0.0	6.4348	0.0071	0.2264	418.0
SynCAN_9_plateau	7.46	-1.3743	75007.0	8.0598	3.069	0.9758	0.0002	-0.0	21.0128	0.1772	0.3359	405.0
SynCAN_10_plateau	1.9562	-1.7944	50005.0	73.6657	0.6593	0.7256	0.0407	0.0	75.3339	0.0094	0.1984	2095.25
SynCAN_1_continuous	1.3264	-2.0793	149999.0	53.0132	0.2223	0.4444	0.1049	0.0001	274.3001	0.0	0.1812	26806.5
SynCAN_2_continuous	1.9748	-1.0242	75008.0	28.1344	0.6464	0.9979	0.0	0.0001	3.1451	0.0002	0.1475	1893.0
SynCAN_3_continuous	0.8386	-2.1338	149998.0	22.5511	0.4473	0.4563	0.111	-0.0	258.1322	0.0011	0.1812	14774.5
SynCAN_4_continuous	8.6971	-0.7851	50005.0	10.3806	1.4228	0.9487	0.0014	0.0	18.8334	0.6171	0.4512	354.0
SynCAN_5_continuous	1.6354	-1.4611	149996.0	28.3865	0.5737	0.9966	0.0	0.0	7.0267	0.0008	0.1306	3150.5
SynCAN_6_continuous	1.2277	-1.9081	75008.0	18.7633	0.3088	0.444	0.0628	-0.0	145.5971	0.0047	0.191	8092.5
SynCAN_7_continuous	2.1228	-0.9226	149998.0	56.767	-0.2403	0.4441	0.1048	0.0	276.5996	0.0006	0.1758	31150.0
SynCAN_8_continuous	8.4289	-1.9753	149998.0	7.7095	0.6718	0.9946	0.0	-0.0	17.071	0.0767	0.3507	415.0
SynCAN_9_continuous	9.673	-1.8203	75008.0	11.2816	1.3008	0.9756	0.0003	0.0	16.4425	0.0861	0.2264	413.0
SynCAN_10_continuous	1.989	-1.7921	50005.0	34.8657	0.5724	0.7283	0.0407	0.0	74.6157	0.0101	0.2011	3150.75
SynCAN_1_playback	1.3463	-2.1861	149997.0	37.3181	0.2087	0.4443	0.1049	0.0	276.0744	0.0002	0.1812	28343.5
SynCAN_2_playback	1.7633	-1.0142	75007.0	15.5412	0.5353	0.997	0.0	-0.0	6.4103	0.001	0.159	1916.0
SynCAN_3_playback	0.818	-2.1456	149998.0	30.9441	0.4421	0.4568	0.111	-0.0	255.8679	0.0002	0.1758	15314.0
SynCAN_4_playback	8.7744	-0.7661	50005.0	19.029	1.3009	0.9505	0.0015	-0.0	14.828	0.549	0.4404	334.0
SynCAN_5_playback	1.6063	-1.218	149999.0	20.4979	0.4334	0.9967	0.0	0.0	7.2305	0.0016	0.1306	3314.0
SynCAN_6_playback	1.2218	-2.1017	75007.0	12.4591	0.4617	0.4436	0.0628	0.0	146.4354	0.0044	0.191	8847.0
SynCAN_7_playback	2.1311	-0.8966	149998.0	64.2763	-0.2348	0.4441	0.1048	-0.0	276.1871	0.0003	0.1758	29890.5
SynCAN_8_playback	4.2385	-1.8995	149999.0	13.9009	-0.3909	0.9982	0.0	-0.0	7.3748	0.0065	0.2264	422.0
SynCAN_9_playback	6.6876	-2.5063	75008.0	23.2132	0.9602	0.9777	0.0003	0.0	11.9337	0.0436	0.2264	447.0
SynCAN_10_playback	2.021	-1.9251	50005.0	27.9738	0.6094	0.7241	0.0408	0.0	75.8987	0.012	0.1957	2138.0
SynCAN_1_suppress	1.3047	-2.1512	147969.0	59.408	0.2357	0.4444	0.1049	0.0	272.4004	0.0002	0.1812	29615.0
SynCAN_2_suppress	1.9327	-1.0198	73496.0	35.4575	0.5713	0.9973	0.0	0.0	4.4165	0.0002	0.159	1866.0
SynCAN_3_suppress	0.8016	-2.14	147160.0	32.0889	0.4321	0.4568	0.1109	-0.0	252.849	0.0001	0.1758	15168.5
SynCAN_4_suppress	9.1733	-0.79	49388.0	31.5742	1.3396	0.9522	0.0014	0.0	13.5846	0.5906	0.4404	332.0
SynCAN_5_suppress	1.4961	-1.5509	148138.0	53.2033	0.3931	0.9973	0.0	0.0	4.3267	0.0001	0.1306	3246.0
SynCAN_6_suppress	1.1421	-2.0544	73771.0	18.3183	0.392	0.4441	0.0628	0.0	144.0215	0.0018	0.191	8628.5
SynCAN_7_suppress	2.1298	-0.8904	147036.0	72.3876	-0.2256	0.4442	0.1048	0.0	272.9232	0.0003	0.1812	29352.0
SynCAN_8_suppress	4.1753	-2.3906	146923.0	17.0392	-0.6827	0.9981	0.0	0.0	6.7557	0.0038	0.2264	397.0
SynCAN_9_suppress	8.3315	-2.5105	73775.0	23.4329	1.2635	0.9786	0.0003	0.0	11.6868	0.0457	0.2264	424.0
SynCAN_10_suppress	1.8992	-1.8614	48665.0	30.8589	0.6104	0.7263	0.0406	0.0	74.2634	0.0109	0.1957	1817.25

Capitolo 7

Conclusioni e futuri sviluppi

Dal lavoro svolto è emerso quanto sia difficile sviluppare un sistema per il rilevamento delle anomalie che sia in grado di funzionare in ambiti diversi e con anomalie di tipo diverso. Il contesto applicativo costituisce un aspetto fondamentale nella scelta degli strumenti di analisi. Sebbene la combinazione di TS2Vec e FCN si sia dimostrata efficace nella classificazione delle serie temporali multivariate, per lo studio di dati di tipo CAN Bus la scelta dell'architettura di base si è rivelata infelice. TS2Vec ha dimostrato di non essere in grado di gestire anomalie contestuali, le modifiche apportate danno qualche beneficio ma lasciano ancora un ampio margine di miglioramento.

Per un ulteriore sviluppo futuro, sarebbe interessante combinare l'approccio di contrastive learning con un'architettura di base più indicata all'apprendimento di un contesto. Esistono già svariati progetti mirati all'anomaly detection che combinano diverse architetture allo stato dell'arte nell'analisi delle serie temporali. Un esempio è [43], in cui il modello è la combinazione di un Variational Autoencoder con una rete LSTM.

Bibliografia

[1] Chao De-Yu. *What is the difference between Traditional Programming and Machine Learning*. 2021. URL: https://medium.com/mlearning-ai/what-is-the-difference-between-traditional-programming-and-machine-learning-f6128ed4f595 (cit. a p. 5).

[2] Qiong Liu e Ying Wu. «Supervised Learning». In: (gen. 2012). DOI: 10.1007/978-1-4419-1428-6_451 (cit. a p. 5).

[3] Salim Dridi. *Supervised Learning - A Systematic Literature Review*. Set. 2021. DOI: 10.13140/RG.2.2.34445.67049 (cit. a p. 5).

[4] Salim Dridi. *Unsupervised Learning - A Systematic Literature Review*. Dic. 2021. DOI: 10.13140/RG.2.2.16963.12323 (cit. a p. 6).

[5] Xiangli Yang, Zixing Song, Irwin King e Zenglin Xu. *A Survey on Deep Semi-supervised Learning*. 2021. DOI: 10.48550/ARXIV.2103.00550. URL: https://arxiv.org/abs/2103.00550 (cit. a p. 6).

[6] Michael McCloskey e Neal J. Cohen. «Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem». In: a cura di Gordon H. Bower. Vol. 24. Psychology of Learning and Motivation. Academic Press, 1989, pp. 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8. URL: https://www.sciencedirect.com/science/article/pii/S0079742108605368 (cit. a p. 6).

[7] Chuanxing GENG Songcan CHEN. «A comprehensive perspective of contrastive self-supervised learning». In: *Frontiers of Computer Science* 15.4, 154332 (2021), p. 154332. DOI: 10.1007/s11704-021-1900-9. URL: https://journal.hep.com.cn/fcs/EN/abstract/article_30092.shtml (cit. a p. 7).

[8] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang e Jie Tang. «Self-supervised Learning: Generative or Contrastive». In: *IEEE Transactions on Knowledge and Data Engineering* (2021), pp. 1–1. DOI: 10.1109/TKDE.2021.3090866 (cit. alle pp. 8, 10, 12).

[9] S. Kullback e R. A. Leibler. «On Information and Sufficiency». In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694 (cit. a p. 9).

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie e Ross Girshick. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2020. arXiv: 1911.05722 [cs.CV] (cit. a p. 10).

[11] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid e Phillip Isola. «What Makes for Good Views for Contrastive Learning?» In: *Advances in Neural Information Processing Systems*. A cura di H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan e H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 6827–6839. URL: https://proceedings.neurips.cc/paper/2020/file/4c2e5eaae9152079b9e95845750bb9ab-Paper.pdf (cit. a p. 10).

[12] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee e Fillia Makedon. *A Survey on Contrastive Self-supervised Learning*. 2020. DOI: 10.48550/ARXIV.2011.00362. URL: https://arxiv.org/abs/2011.00362 (cit. alle pp. 11, 13–15).

[13] Philip Bachman, R Devon Hjelm e William Buchwalter. *Learning Representations by Maximizing Mutual Information Across Views*. 2019. DOI: 10.48550/ARXIV.1906.00910. URL: https://arxiv.org/abs/1906.00910 (cit. a p. 12).

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi e Geoffrey Hinton. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. DOI: 10.48550/ARXIV.2002.05709. URL: https://arxiv.org/abs/2002.05709 (cit. alle pp. 12, 16, 17, 31).

[15] Carl Doersch, Abhinav Gupta e Alexei A. Efros. «Unsupervised Visual Representation Learning by Context Prediction». In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1422–1430. DOI: 10.1109/ICCV.2015.167 (cit. a p. 14).

[16] Kihyuk Sohn. «Improved Deep Metric Learning with Multi-class N-pair Loss Objective». In: *Advances in Neural Information Processing Systems*. A cura di D. Lee, M. Sugiyama, U. Luxburg, I. Guyon e R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf (cit. a p. 17).

[17] Jinwoo Shin. *Contrastive Learning for Novelty Detection*. URL: https://alinlab.kaist.ac.kr/resource/csi_v0.pdf (cit. alle pp. 18, 19).

[18] Jihoon Tack, Sangwoo Mo, Jongheon Jeong e Jinwoo Shin. «CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances». In: *Advances in Neural Information Processing Systems*. A cura di H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan e H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 11839–11852. URL: https://proceedings.neurips.cc/paper/2020/file/8965f76632d7672e7d3cf29c87ecaa0c-Paper.pdf (cit. alle pp. 18, 31).

[19] Grubbs. «Procedures for Detecting Outlying Observations in Samples». In: *Technometrics* 11 (1969), pp. 1–21. DOI: http://dx.doi.org/10.1080/00401706.1969.10490657 (cit. a p. 20).

[20] Varun Chandola, Arindam Banerjee e Vipin Kumar. «Anomaly Detection: A Survey». In: *ACM Comput. Surv.* 41.3 (lug. 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882 (cit. alle pp. 20, 23, 24).

[21] Jim Austin Victoria Hodge. «A Survey of Outlier Detection Methodologies». In: *Artificial Intelligence Review* 22 (2004), pp. 85–126. DOI: 10.1023/B:AIRE.0000045502.10941.a9 (cit. alle pp. 20, 23).

[22] Jiuqi Elise Zhang, Di Wu e Benoit Boulet. «Time Series Anomaly Detection for Smart Grids: A Survey». In: *2021 IEEE Electrical Power and Energy Conference (EPEC)*. 2021, pp. 125–130. DOI: 10.1109/EPEC52095.2021.9621752 (cit. a p. 20).

[23] Ang Zhang, Xiaoyong Zhao e Lei Wang. «CNN and LSTM based Encoder-Decoder for Anomaly Detection in Multivariate Time Series». In: *2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)*. Vol. 5. 2021, pp. 571–575. DOI: 10.1109/ITNEC52019.2021.9587207 (cit. a p. 21).

[24] Jingye Yee, Cheng Yee Low, Natiara Mohamad Hashim, Fazah Akhtar Hanapiah, Ching Theng Koh, Noor Ayuni Che Zakaria, Khairunnisa Johar e Nurul Atiqah Othman. «Systematic Development of Machine for Abnormal Muscle Activity Detection». In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. 2021, pp. 1376–1381. DOI: 10.1109/CASE49439.2021.9551525 (cit. a p. 21).

[25] Qunke Wang, Lanting Fang, Zhenchao Zhu e Jie Huang. «Detection Algorithm of the Mimicry Attack based on Variational Auto-Encoder». In: *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 2021, pp. 114–120. DOI: 10.1109/DSN-W52860.2021.00029 (cit. a p. 21).

[26] Kamran Shaukat Dar, Talha Mahboob Alam, Suhuai Luo, Shakir Shabbir, Ibrahim Hameed, Jiaming Li, Syed Abbas e Umair Javed. «A Review of Time-Series Anomaly Detection Techniques: A Step to Future Perspectives». In: apr. 2021. ISBN: 978-3-030-73099-4. DOI: 10.1007/978-3-030-73100-7_60 (cit. alle pp. 21, 22).

[27] Colin O'Reilly, Alexander D. Gluhak, Muhammad Ali Imran e Sutharshan Rajasegarar. «Anomaly Detection in Wireless Sensor Networks in a Non-Stationary Environment». In: *IEEE Communications Surveys & Tutorials* 16 (2014), pp. 1413–1432 (cit. a p. 22).

[28] Hadi Hojjati, Thi Kieu Khanh Ho e Narges Armanfard. *Self-Supervised Anomaly Detection: A Survey and Outlook*. 2022. DOI: 10.48550/ARXIV.2205.05173. URL: https://arxiv.org/abs/2205.05173 (cit. a p. 21).

[29] Francis John Anscombe. «Rejection of Outliers». In: *Technometrics* 2 (1960), pp. 123–146 (cit. a p. 23).

[30] S. Golden e B. Friedlander. «Maximum likelihood estimation, analysis, and applications of exponential polynomial signals». In: *IEEE Transactions on Signal Processing* 47.6 (1999), pp. 1493–1501. DOI: 10.1109/78.765111 (cit. a p. 23).

[31] Edwin M. Knorr e Raymond T. Ng. «A Unified Notion of Outliers: Properties and Computation». In: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. KDD'97. Newport Beach, CA: AAAI Press, 1997, pp. 219–222 (cit. a p. 24).

[32] Jingyu Sun Zhixian Niu Shuping Shi e Xiu He. «A Survey of Outlier Detection Methodologies and Their Applications». In: *Artificial Intelligence and Computational Intelligence* 7002 (2011). DOI: https://doi-org.ezproxy.biblio.polito.it/10.1007/978-3-642-23881-9_50 (cit. a p. 24).

[33] Markus Breunig, Hans-Peter Kriegel, Raymond Ng e Joerg Sander. «LOF: Identifying Density-Based Local Outliers.» In: vol. 29. Giu. 2000, pp. 93–104. DOI: 10.1145/342009.335388 (cit. a p. 24).

[34] Martin Ester, Hans-Peter Kriegel, Jörg Sander e Xiaowei Xu. «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise». In: KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cit. a p. 24).

[35] Guansong Pang, Chunhua Shen, Longbing Cao e Anton Van Den Hengel. «Deep Learning for Anomaly Detection: A Review». In: *ACM Comput. Surv.* 54.2 (mar. 2021). ISSN: 0360-0300. DOI: 10.1145/3439950. URL: https://doi.org/10.1145/3439950 (cit. alle pp. 25, 26).

[36] Ane Blázquez-García, Angel Conde, Usue Mori e Jose A. Lozano. *A review on outlier/anomaly detection in time series data*. 2020. DOI: 10.48550/ARXIV.2002.04236. URL: https://arxiv.org/abs/2002.04236 (cit. a p. 27).

[37] Kwei-Herng Lai et al. *TODS: An Automated Time Series Outlier Detection System*. 2020. DOI: 10.48550/ARXIV.2009.09822. URL: https://arxiv.org/abs/2009.09822 (cit. a p. 28).

[38] Lilian Weng. *From Autoencoder to Beta-VAE*. Ago. 2018. URL: https://lilianweng.github.io/posts/2018-08-12-vae/ (cit. alle pp. 28, 30).

[39] Dor Bank, Noam Koenigstein e Raja Giryes. *Autoencoders*. Mar. 2020 (cit. a p. 28).

[40] David E. Rumelhart e James L. McClelland. «Learning Internal Representations by Error Propagation». In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362 (cit. a p. 28).

[41] Yuta Kawachi, Yuma Koizumi e Noboru Harada. «Complementary Set Variational Autoencoder for Supervised Anomaly Detection». In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 2366–2370. DOI: 10.1109/ICASSP.2018.8462181 (cit. a p. 30).

[42] Jinwon An e Sungzoon Cho. «Variational Autoencoder based Anomaly Detection using Reconstruction Probability». In: 2015 (cit. a p. 30).

[43] Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni e Stephen Roberts. «Anomaly Detection for Time Series Using VAE-LSTM Hybrid Model». In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 4322–4326. DOI: 10.1109/ICASSP40776.2020.9053558 (cit. alle pp. 30, 57).

[44] Jinhua Zhu, Yingce Xia, Lijun Wu, Jiajun Deng, Wengang Zhou, Tao Qin, Tie-Yan Liu e Houqiang Li. «Masked Contrastive Representation Learning for Reinforcement Learning». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–1. DOI: 10.1109/TPAMI.2022.3176413 (cit. a p. 31).

[45] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong e Bixiong Xu. *TS2Vec: Towards Universal Representation of Time Series*. 2021. DOI: 10.48550/ARXIV.2106.10466. URL: https://arxiv.org/abs/2106.10466 (cit. alle pp. 31–33).

[46] Jean-Yves Franceschi, Aymeric Dieuleveut e Martin Jaggi. «Unsupervised Scalable Representation Learning for Multivariate Time Series». In: (2019). DOI: 10.48550/ARXIV.1901.10738. URL: https://arxiv.org/abs/1901.10738 (cit. a p. 32).

[47] Sana Tonekaboni, Danny Eytan e Anna Goldenberg. «Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding». In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=8qDwejCuCN (cit. a p. 32).

[48] Zhiguang Wang, Weizhong Yan e Tim Oates. *Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline*. 2016. DOI: 10.48550/ARXIV.1611.06455. URL: https://arxiv.org/abs/1611.06455 (cit. a p. 35).

[49] *Convolutional Neural Network (CNN) for Time Series Classification*. 2020. URL: https://www.macnica.co.jp/business/ai_iot/columns/135112/ (cit. a p. 36).

[50] Oliver Ammann, Gabriel Michau e Olga Fink. «Anomaly Detection And Classification In Time Series With Kervolutional Neural Networks». In: *ArXiv* abs/2005.07078 (2020) (cit. a p. 37).

[51] Markus Hanselmann, Thilo Strauss, Katharina Dormann e Holger Ulmer. «CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data». In: *IEEE Access* 8 (2020), pp. 58194–58205. DOI: 10.1109/ACCESS.2020.2982544 (cit. alle pp. 37, 50, 52).

[52] *Automotive CAN Bus Explained*. URL: https://ocsaly.com/can-bus-explained-controller-area-network-detailed/ (cit. a p. 38).

[53] Lorenzo Perini. *Predictive Maintenance for off-road vehicles based on Hidden Markov Models and Autoencoders for trend Anomaly Detection*. Tesi di laurea magistrale. 2018/2019 (cit. alle pp. 38–40).

[54] *tsfresh*. URL: https://tsfresh.readthedocs.io/en/latest/index.html (cit. a p. 42).

[55] N. V. Chawla, K. W. Bowyer, L. O. Hall e W. P. Kegelmeyer. «SMOTE: Synthetic Minority Over-sampling Technique». In: *Journal of Artificial Intelligence Research* 16 (giu. 2002), pp. 321–357. DOI: 10.1613/jair.953. URL: https://doi.org/10.1613%2Fjair.953 (cit. a p. 42).

[56] Haibo He e Edwardo A. Garcia. «Learning from Imbalanced Data». In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239 (cit. a p. 42).

[57] Thomas Schreiber e Andreas Schmitz. «Discrimination power of measures for nonlinearity in a time series». In: *Phys. Rev. E* 55 (5 mag. 1997), pp. 5443–5447. DOI: 10.1103/PhysRevE.55.5443. URL: https://link.aps.org/doi/10.1103/PhysRevE.55.5443 (cit. a p. 44).

[58] Gustavo Batista, Eamonn Keogh, Oben Tataw e Vinícius Alves de Souza. «CID: An efficient complexity-invariant distance for time series». In: *Data Mining and Knowledge Discovery* 28 (apr. 2013). DOI: 10.1007/s10618-013-0312-3 (cit. a p. 44).

[59] Nikolay Laptev, Saeed Amizadeh e Youssef Billawala. *A Benchmark Dataset for Time Series Anomaly Detection*. 2015. URL: https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly (cit. a p. 49).

[60] Hansheng Ren et al. «Time-Series Anomaly Detection Service at Microsoft». In: *CoRR* abs/1906.03821 (2019). arXiv: 1906.03821. URL: http://arxiv.org/abs/1906.03821 (cit. a p. 50).