2024

ALAD, studio e implementazione

PROGETTO MACHINE LEARNING, UNICAL / LAUREA MAGISTRALE IN INTELLIGENZA ARTIFICIALE

EDOARDO MISURELLI – 224 438

Indice

Abstract

Il seguente progetto riguarda lo studio e l'implementazione dell'architettura **ALAD** (*Adversarially Learned Anomaly Detection*);

ALAD, fornisce un metodo innovativo allo studio dei problemi di anomaly detection, con un approccio ibrido, basato su GANs bidirezionali e Autoencoder.

Il lavoro svolto si è concentrato innanzitutto, sulla conversione del codice, quest' ultimo è stato riprogettato utilizzando Tensorflow 2.x;
Si è deciso successivamente, di adattare la rete ad un nuovo tipo di dati in ingresso;

Sono stati presi in considerazione un tipo di dati specifico, le immagini;
Nel paper originale, il modello viene utilizzato con immagini di grandezza 32x32 pixels; dunque, il passaggio successivo ha richiesto l'adattamento della rete, per avere in ingresso immagini di dimensione 64x64 pixels.
Come ultimo step, si è rivolta l'attenzione sullo studio di un parametro molto importante per la rete, ovvero la dimensione latente, variando quest'ultimo, si è studiato il comportamento del modello, per riuscire ad evidenziare valori ottimali ed eventuali punti di forza e di debolezza dell'architettura ALAD.

Adversarially Learned Anomaly Detection

Il lavoro descritto dai ricercatori Francesi propone un approccio non solo efficace, ma anche efficiente in fase di test; i modelli GANs based, possono essere molto pesanti, se si parla di dati ad alta dimensionalità (come le immagini). Nello specifico, il metodo utilizza una classe di GANs che, durante il training, effettua il learning di una rete Encoder – Decoder, la quale rende successivamente più veloce la fase di inferenza.
È importante sottolineare come, la rete Decoder, in questo caso non sarà altro che, il Generatore della GAN;
A questo punto, come si può facilmente dedurre, la rilevazione o meno delle anomalie, verte sull' errore di ricostruzione dei sample, anche in questo caso il gruppo di ricerca propone nello studio, varie misure di ricostruzione che, descriverò più nel dettaglio in seguito.

Studiando la rete, si è deciso dunque di concentrarsi su un parametro fondamentale per l'architettura, questo è la dimensione latente;
La rete Encoder, destruttura le immagini e le descrive in una dimensione latente (una sorta di compressione), ovviamente comprimendo i dati, abbiamo una perdita di informazione sull'immagine originale, a questo punto, scegliere dunque un valore scorretto di z (dove con z indichiamo la dimensione dello spazio latente), porta il Decoder, ovvero la rete che, cerca di ricostruire il dato compresso, ad avere scarsi risultati. Qui entra in gioco la scelta essenziale e corretta di z e dunque il mio studio.
Inoltre cosa succede se la dimensione dei dati cresce? La rete è ancora in grado di rispondere adeguatamente?

Questi appena descritti, sono i punti cardine su cui ho condotto la mia ricerca, nei prossimi capitoli mostrerò più in dettaglio il problema, la struttura della rete e i risultati ottenuti.

GANs ed Anomaly Detection

Le **GAN**, o *Generative Adversarial Networks* (Reti Generative Avversarie), sono un tipo di rete neurale, utilizzata principalmente, per generare nuovi dati. Le GAN sono composte da due reti neurali principali che, competono tra loro: il generatore e il discriminatore.

Generatore: questa rete prende in input un rumore casuale e genera dati falsi che cercano di imitare i dati di addestramento. L'obiettivo del generatore è ingannare il discriminatore, facendo sì che i dati generati, sembrino il più possibile realistici. In particolare, G cerca di imparare la distribuzione reale dei dati.

Discriminatore: questa rete riceve in input sia dati reali (presi dal set di addestramento) che, dati falsi (generati dal generatore). Il compito del discriminatore è distinguere tra dati reali e dati generati.

Il processo di addestramento delle GAN è un gioco a somma zero, in cui il generatore e il discriminatore migliorano continuamente. Il generatore cerca di migliorare la qualità dei dati generati per ingannare il discriminatore, mentre il discriminatore diventa sempre più abile nel distinguere tra dati reali e falsi. Questo continuo miglioramento reciproco porta, idealmente, il generatore a produrre dati estremamente realistici.

$$V(D,G) = E_{x \sim p(x)}[logD(x)] + E_{z \sim p(z)}\left[\log\left(1 - \mathrm{D}(\mathrm{G}(\mathrm{z}))\right)\right]$$

Il problema definito come $\min\mathrm{G}\ maxD\ V(D,G)$, descrive come avviene formalmente il training delle GAN.

Analogamente, le GAN possono essere adattate ai task di *anomaly detection*. Gli approcci però sono computazionalmente molto pesanti e dunque difficilmente utilizzabili con dati ad alta dimensionalità.

ALAD si pone proprio l'obiettivo di migliorare l'efficienza delle GAN in questi task, nel prossimo capitolo descriverò brevemente l'architettura della rete.

Architettura ALAD

L'architettura del modello, si basa su una classe di GANs che incorpora una rete Encoder, la quale viene addestrata in parallelo a mappare i dati in ingresso (indicati con ***x***), nella dimensione latente (indicata con ***z***). Questo approccio proviene da una particolare rete, descritta nel modello **BiGAN**; in quest' ultimo tipo di GANs, il discriminatore non discrimina solo gli esempi generati artificialmente da quelli reali, ma anche le dimensioni latenti, ovvero, l'output dell'encoder e z in ingresso al generatore.

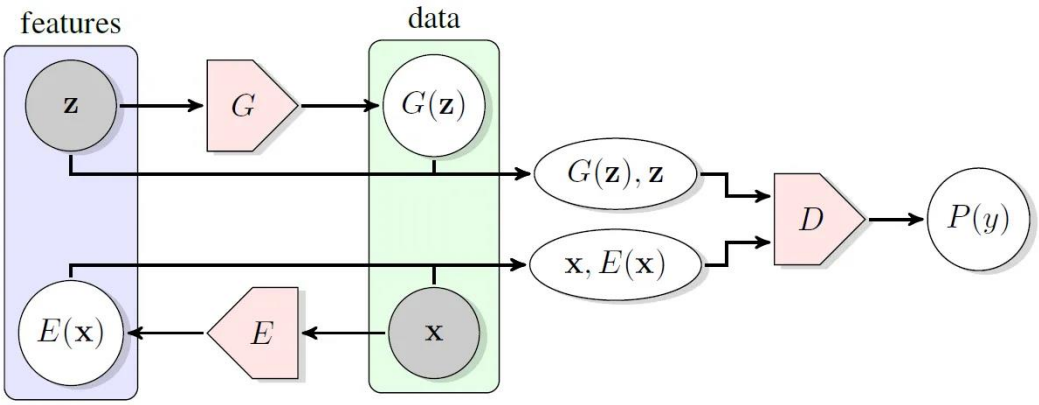

Figura 1: Architettura BiGAN

Dunque, il seguente discriminatore che nella rete ALAD viene indicato con ***Dxz*** , discriminerà sia nello spazio dei dati che in quello delle features. Questo, permette di poter utilizzare le GANs anche in contesti dove i dati in input sono immagini ad alta risoluzione, poiché solo l'encoder dovrà lavorare con input di grande dimensione, lasciando piccolo l'input di generatore e discriminatore. La seguente formula descrive il problema:

$$min_{G,E} max_D \, V(D,E,G)$$

$$V(D,E,G) = E_{x \sim p(x)}\big[\log\big(x, E(x)\big)\big] + E_{z \sim p(z)}[\log\big(1 - \mathrm{D}(\mathrm{G}(\mathrm{z}), \mathrm{z})\big)]$$

Nella pratica, la rete introdotta è molto difficile da allenare e da portare a convergenza, per questo nel lavoro viene introdotta una tecnica di regolarizzazione chiamata ***Conditional Entropy Regularization.*** Nella pratica però, al posto di introdurre il termine di regolarizzazione vero e proprio, viene aggiunto alla rete un ulteriore discriminatore, in grado di approssimare il valore.

Grazie a questa tecnica riusciamo a forzare la **Cycle Consistency**;

La Cycle Consistency può essere descritta brevemente dalla formula:

$$G\big(E(x)\big) \approx x$$

Ovvero gli esempi generati dal generatore a partire dalla rappresentazione latente di x, dovrebbero essere molto simili ad x. Questo non è sempre fattibile ma nel caso specifico si cerca di forzare questo comportamento introducendo, come introdotto in precedenza un nuovo discriminatore nella rete, inidcato con ***Dxx***.
Allo stesso modo viene introdotto un termine di regolarizzazione dello spazio latente, che fa fronte ad un ulteriore discriminatore, indicato stavolta con ***Dzz***.

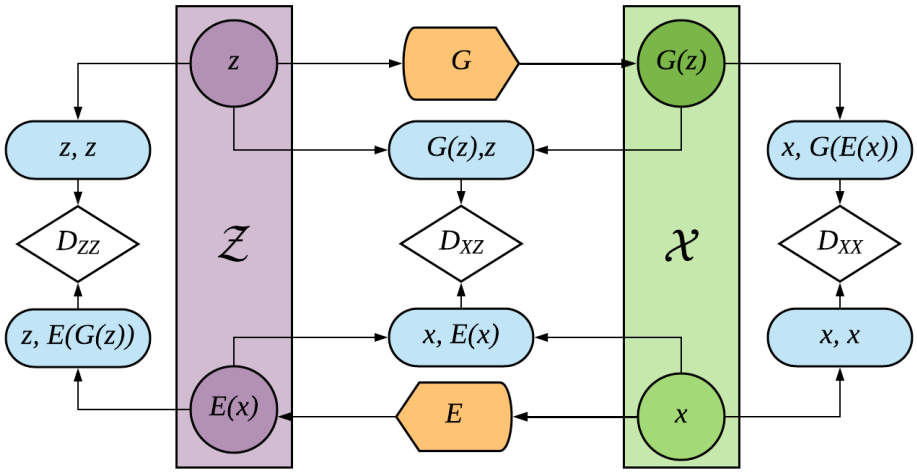

Figura 2: Architettura ALAD

Questa è una rappresentazione dell'architettura ALAD contenente tutti i moduli…di seguito una panoramica sulle reti che la compongono:

- ***Encoder* (*E*)**: si occupa di ridurre la dimensionalità dei dati, mappandoli in uno spazio latente. La rete encoder viene addestrata in parallelo alla rete GAN;

- ***Generator/Decoder (G)***: il generatore deve creare immagini simili alle originali, ricostruendole però dalla dimensione latente;

- ***Discriminatore xz*** (***Dxz***): particolare discriminatore che discrimina tra spazio dei dati e spazio delle features, questo particolare discriminatore viene introdotto nel modello **BiGAN**:

- ***Discriminatore xx*** e ***Discriminatore zz (Dxx*** , ***Dzz***): entrambi introdotti per stabilizzare il training del modello. In particolare, **Dxx** fa fronte alla cycle consistency, mentre **Dzz** lavora sulle features nello spazio latente.

Adattamento dell'architettura

L'obiettivo del progetto è stato studiare il comportamento del modello con sample di dimensione maggiore, in particolare si è scelto un dataset di immagini 64x64 pixel. L'adattamento della rete ha riguardato specifiche modifiche sui moduli, in modo di rendere in grado il modello di processare immagini con risoluzione maggiore.

I moduli interessati sono stati: **Encoder**, **Generatore**, **Discriminatore xz**. Per ognuno la scelta è stata aggiungere dei blocchi convoluzionali in particolare:

- ***Encoder Net***: aggiunta di un primo blocco Convoluzionale seguito da un layer di Batch Normalization ed un layer di attivazione (LeakyReLU);
- ***Decoder Net***: aggiunta di un ultimo blocco Convoluzionale seguito da un layer di Batch Normalization ed un layer di attivazione (ReLU);
- ***Discriminator xz Net***: aggiunta di un primo blocco Convoluzionale seguito da un layer di Batch Normalization ed un layer di attivazione (LeakyReLU);

A tutti i blocchi convoluzionali è stata applicata la *Spectral Normalization*, seguendo l'architettura orginale. Questa tecnica migliora la stabilità del training della GAN.

Individuare anomalie con ALAD

A questo punto, dopo aver descritto per sommi capi l'architettura del modello, come funziona la fase di inferenza? e dunque in che modo ALAD riconosce le anomalie nei dati? La tecnica di rilevamento delle anomalie è basata sulla ricostruzione, il modello valuta quanto un campione si discosta dalla sua ricostruzione. I campioni normali, dovrebbero essere ricostruiti accuratamente, mentre i campioni anomali probabilmente saranno ricostruiti in modo approssimativo. Negli approcci '***reconstruction based***' si utilizzano alcune misure o scores, in grado di misurare la distanza in termini di differenza tra due sample. Le più note distanze sono, la *norma l1* e la *norma l2*; nel paper però, si fa riferimento anche ad un altro tipo di score, quest'ultimo, calcola la distanza tra i sample nello spazio delle features di **Dxx** (Cycle Consistency Discriminator).

$$A(x) = \| f_{xx}(x, x) - f_{xx}(x, G(E(x)) \|_1$$

A(x) definisce la distanza, espressa come la norma l1 tra le distanze delle features. Più grande sarà il valore, più la ricostruzione sarà di scarsa qualità e dunque, molto probabilmente, saremo in presenza di un'anomalia. Nello studio sono state analizzate le distanze secondo varie metriche, per ricondurci a quella più adatta al task dell'anomaly detection.

Setup and Training

Per addestrare il modello, ho utilizzato la piattaforma ***Colab*** offerta da ***Google***, il linguaggio scelto per lo sviluppo del modello è stato ***Python***, mentre la libreria utilizzata ***Tensorflow*** versione 2.x;
La scelta di convertire il codice originale da tensorflow 1.x a tensorflow 2.x, è stata presa per una questione di semplicità di utilizzo della libreria.

Nel notebook realizzato, sono presenti due tipi di architetture, la classica a 32 e quella a 64 per riuscire a confrontare i risultati successivamente. Il dataset scelto, contiene immagini di grandezza 64x64 pixel, divise in 5 classi: ***bacche, uccelli, cani, fiori, altro***; ogni classe contenente più di mille sample.

Per addestrare la rete 32x32 ho utilizzato invece **CIFAR10**, come descritto anche nel paper originale.
Per la rete 64x64 sono state utilizzate le immagini del nuovo dataset.
La classe scelta per gli esperimenti è stata quella degli uccelli('***bird***'), anche perché presente in entrambi i dataset (la scelta di sperimentare su una sola label è stata imposta anche dai limiti computazionali della piattaforma scelta).
L' 80% del dataset è stato utilizzato per il training, mentre il restante 20% per il testing;
Il training set è stato ulteriormente diviso, sfruttando un 10% per un validation set (25% nell'architettura originale 32x32);
Il dataset utilizzato nel nuovo esperimento è stato sottoposto a ***data-augmentation***, tramite trasformazioni di rotazione, zoom e cambiamento della tonalità dei pixel, in modo da aumentare il numero di sample per classe; per ogni classe si è arrivati ad avere poco più di 3000 esempi.

Gli esempi di ogni classe sono stati rimappati ad appartenere ad una classe ***'0'*** rappresentante dei sample ***'normali'***, o ad una classe ***'1'*** degli esempi ***'anomali'***. Nell'esperimento originale questo è stato fatto per ogni label, creando così dieci dataset distinti.
La rete è stata addestrata solo con sample della classe considerata 'normale', scartando dunque da training set e validation set, gli esempi anomali, questo per avere una ricostruzione migliore solo dei sample della classe normale. Nella fase di testing, vengono ovviamente mantenuti sample normali e anomali.

La dimensione latente, utilizzata nell'esperimento originale, rimane invariata e fissata a priori a 100. Nel nuovo esperimento, si è deciso di variare questo parametro e verificare dunque, come la rete riesce a ricostruire i sample partendo da una dimensione latente più bassa, arrivando ad una molto più alta.
Le dimensioni scelte sono state ***100 – 300 – 500 - 800***;
Per ogni valore è stato addestrato un modello differente e successivamente si sono confrontati i risultati.

Exponential Moving Average (EMA)

Questa tecnica mantiene traccia degli aggiornamenti dei pesi della rete nel tempo, ovvero tra le epoche; la media mobile riesce a smussare e a stabilizzare i pesi della rete, con un peso, il quale da maggiore importanza alle ultime epoche rispetto alle precedenti.
Grazie a questa ottimizzazione, proposta nel paper originale, la rete riesce a convergere prima, ricostruendo meglio gli esempi.
L'iper-parametro utilizzato è detto ***ema-decay***, o decadimento;
Seguendo l'esperimento originale si è deciso di porre il valore di decadimento, pari a **0.999**.

Overview

Nella seguente tabelle vengono riportati, i parametri utilizzati per il training delle reti.

	Modello originale	Modello riadattato
Epoche	100	100
Dimensione Input	32 x 32	64 x 64
Learning Rate	0.0002	0.0002
Batch Size	32	64
Dimensione Training Set	4473 (CIFAR-10)	3011
Dimensione Test Set	10000 (CIFAR-10)	2000
Dimensione Validation Set	527 (CIFAR-10)	120
Ema decay	0.999	0.999
Dimensione Latente	100	100/300/500/800

Utilizzo del Validation Set

Il Validation set è stato utilizzato per monitorare l'errore di ricostruzione, durante il training della rete; Ad ogni checkpoint del ciclo di addestramento, infatti, si è deciso di calcolare un Mean Reconstruction Error sui sample appartenenti al Validation, se quest'ultimo diminuisce rispetto al miglior valore registrato durante il training, allora il valore viene aggiornato e il modello salvato.

Nell'esperimento proposto l'errore di ricostruzione sul validation è stato calcolato tramite norma l2, diversamente da quello originale, dove si è optato di utilizzare la norma l1 tra le distanze nel features space.

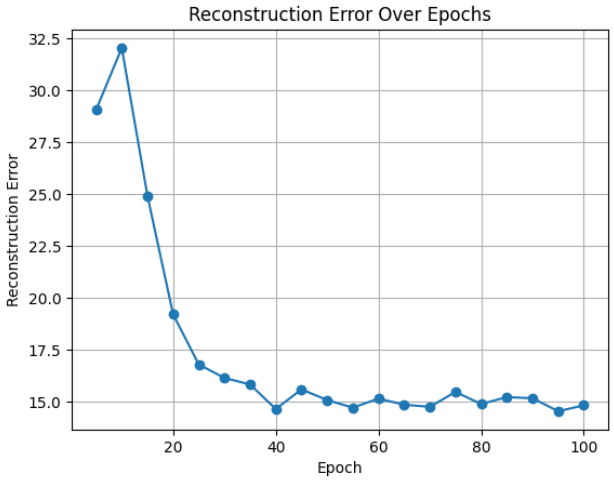

Figura 3: Architettura 32x32 errore di ricostruzione sul Validation set

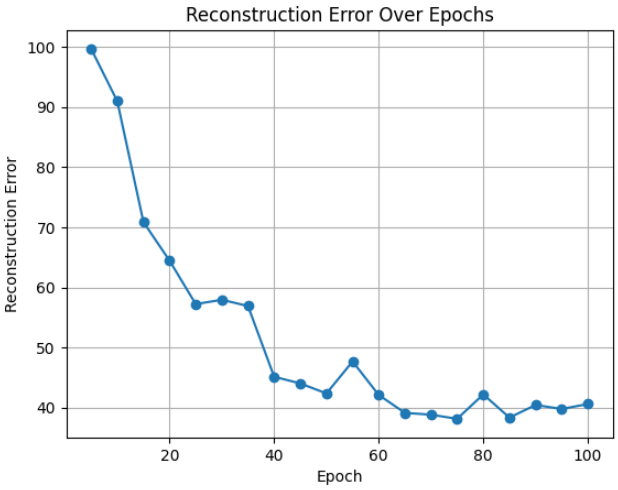

Figura 4: Architettura 64x64 errore di ricostruzione sul Validation set

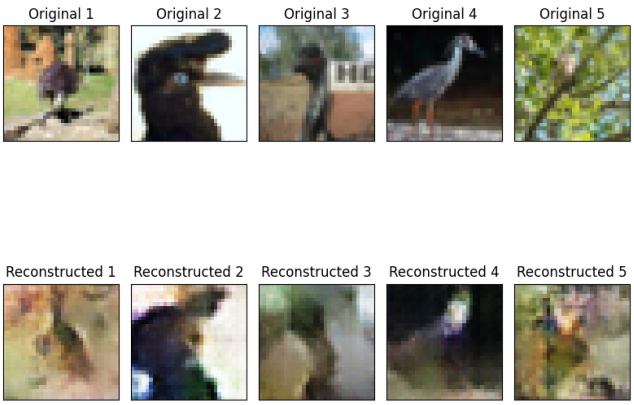

Figura 5: Esempio di ricostruzione architettura 32x32

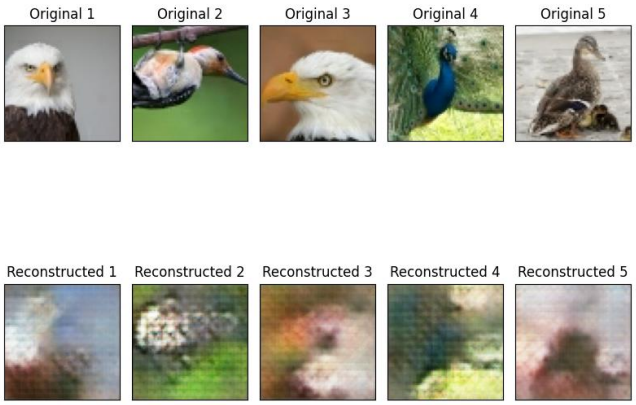

Figura 6: Esempio di ricostruzione architettura 64x64

Testing

Una volta ultimata la fase di training, si è passato alla fase di test, per verificare le prestazioni dei modelli allenati;
Importante è stato confrontare i risultati tra modello originale e modello riadattato per i nuovi dati. Per ogni sample, appartenente al test set, si è deciso di calcolare varie metriche/distanze, tra l'immagine originale e la sua ricostruzione. Le metriche utilizzate sono:

- ***Score l1***: $A(x) = \|x - x'\|_1$
- ***Score l2***: $A(x) = \|x - x'\|_2$
- ***Score ch***: $A(x) = \log(D_{xx}(x, x'))$
- ***Score fm***: $A(x) = \|f_{xx}(x, x') - f_{xx}(x, x')\|_1$

Nel nuovo esperimento, basato sulle diverse dimensioni latenti, le due metriche comparate sono state: la norma l2, indicata con ***score l2*** e la norma l1 sullo spazio delle features, chiamata ***score fm***;

Gli score vengono in seguito utilizzati, per tracciare le curve ***ROC-AUC***, valutando così le reali capacità dei modelli sul task di rilevazione delle anomalie.

Risultati ottenuti

Differenze tra modelli

La nuova architettura è riuscita ad avere risultati soddisfacenti anche per sample di grandezza maggiore?

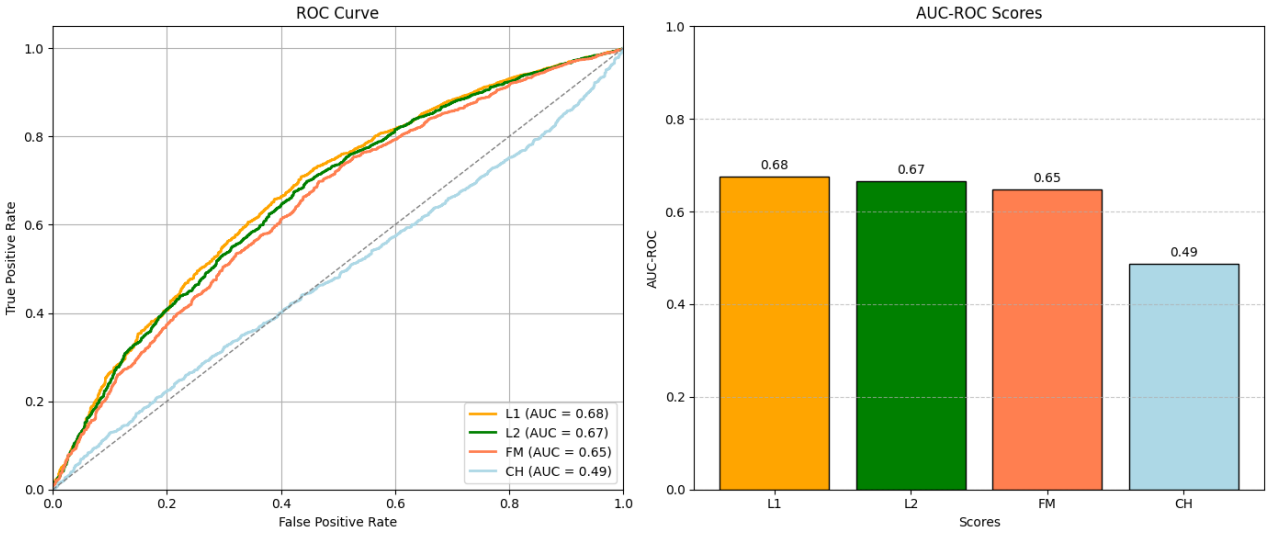

Figura 7: Curva ROC architettura 32x32; classe normale 'Bird'

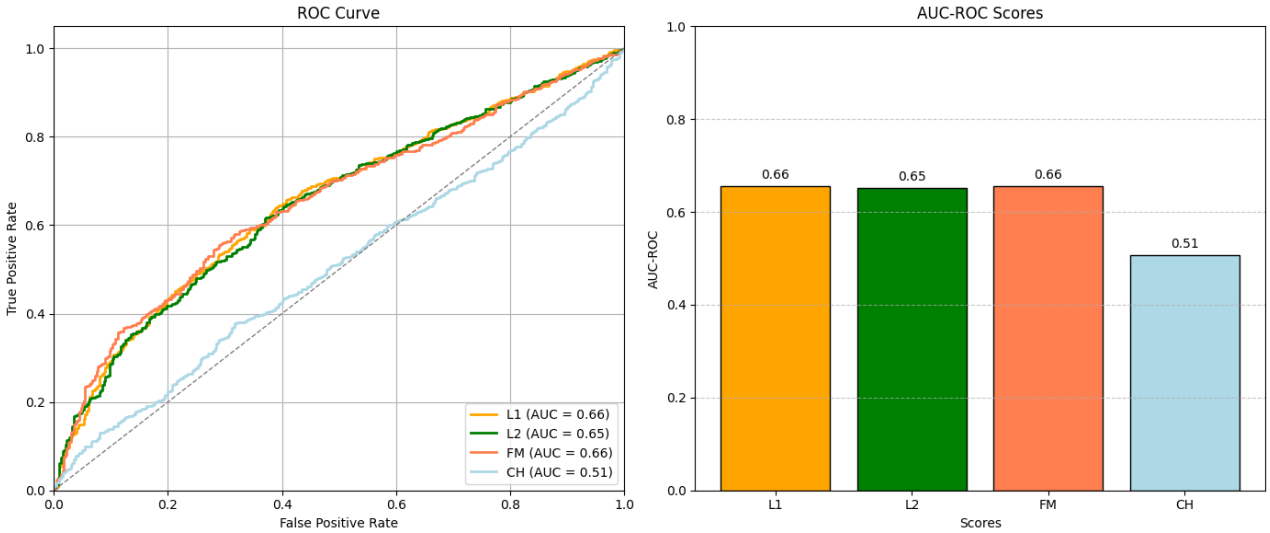

Figura 8Curva ROC architettura 64x64; classe normale 'Bird'

I grafici rappresentano come varia la curva ROC-AUC in base alla metrica utilizzata.
Come si nota facilmente, il modello riadattato riesce ad essere competitivo con quello originale, pur trattando immagini di risoluzione maggiore;
I grafici mostrano, come gli score più competitivi sono sicuramente ***L2*** e ***FM***, ma anche lo score ***L1***, di solito non utilizzata in ambiti reconstruction based.

Studio sulla dimensione latente

Intuitivamente, aumentando le dimensioni dello spazio latente, allora la ricostruzione dovrebbe migliorare a discapito ovviamente dell'accuratezza del modello, che ricostruendo troppo bene, si ritroverà ad essere troppo 'bravo' e dunque, a ricostruire bene sia sample normali che anomali. Contrariamente i risultati mostrano un aspetto diverso; infatti, aumentando la dimensione dello spazio latente, notiamo come in realtà l'errore di ricostruzione cresca drasticamente.

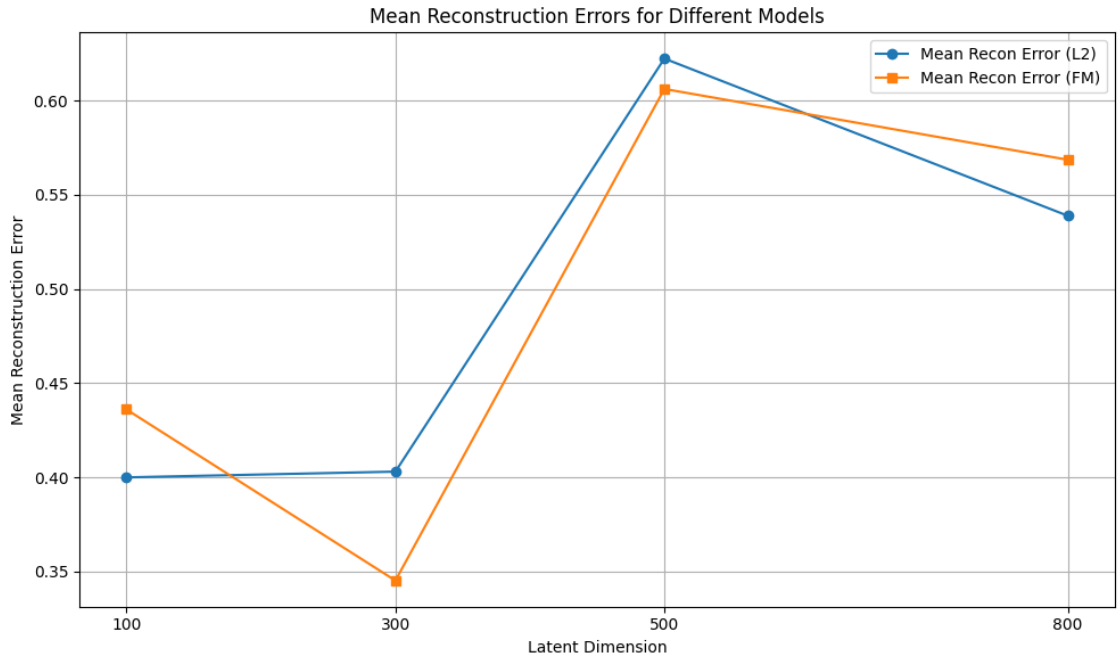

Figura 9: Errore di ricostruzione medio per diversi valori di dimensione latente

Si nota che sia l'errore calcolato secondo lo score l2, che quello secondo lo score fm, crescano all' aumentare della dimensione dello spazio latente.

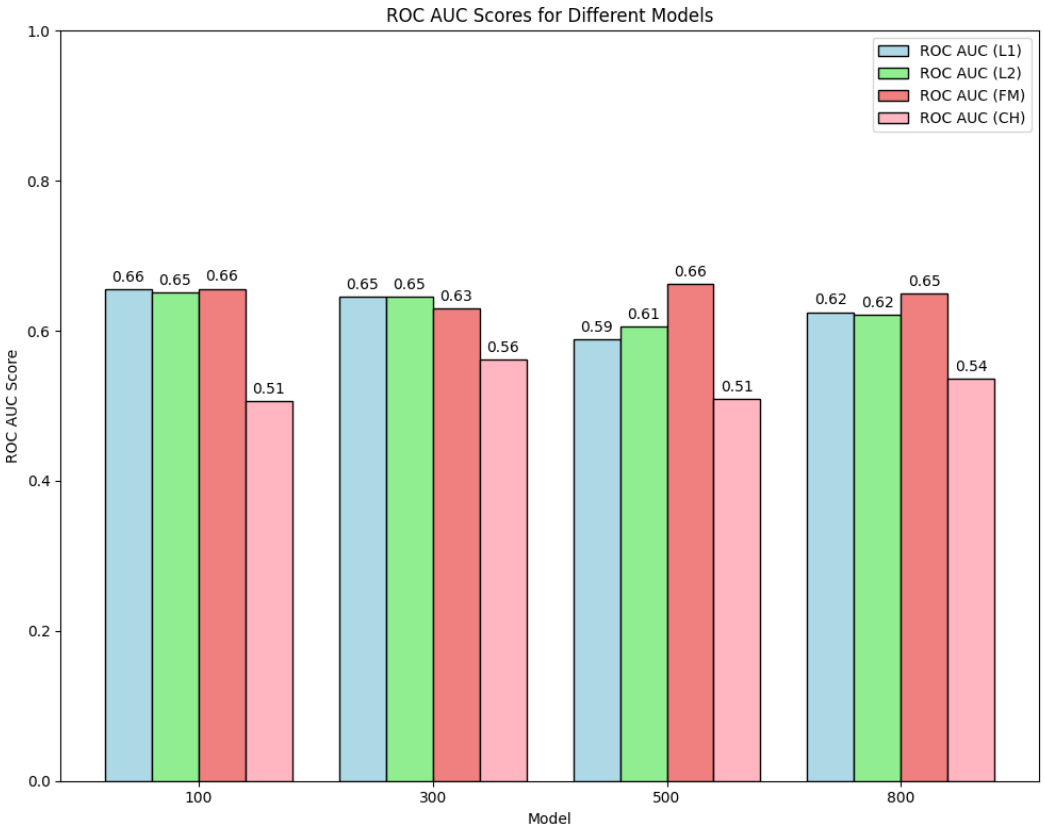

Figura 10: ROC score per dimensione latente e diverse misure di distanza.

Quest'ultimo grafico riassume i ROC-AUC score divisi per distanza utilizzata e dimensione latente; Da notare che, come effettivamente suggerito anche nel paper, la misura migliore che oscilla di meno tra i modelli è sicuramente lo score FM.

Esempi di ricostruzione

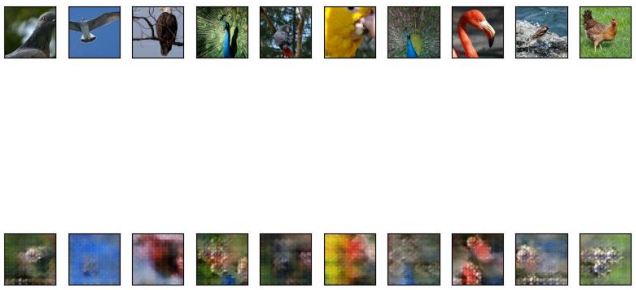

Figura 12: Ricostruzione esempi normali, architettura 64x64.

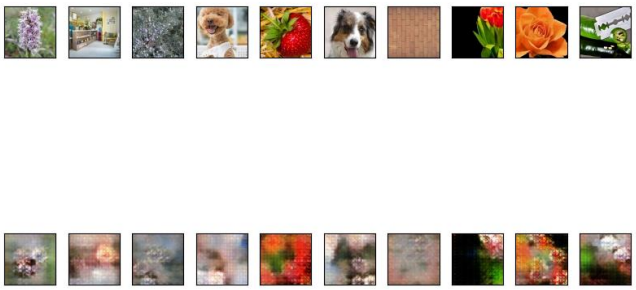

Figura 11: Ricostruzione esempi anomali, architettura 64x64.

Gli esempi mostrano come in realtà il modello faccia molta fatica a ricostruire esempi sia della classe anomala (destra) che quelli della classe normale(sinistra).
Nonostante ciò gli esempi della classe normale (‘***bird***’), sembrano essere comunque ricostruiti meglio, come si nota dalle tonalità di colore e dai contorni delle forme.
Gli esempi anomali ricostruiti meglio sono invece quelli delle classi ‘semplici’, come quella dei fiori.

Conclusioni

L'esperimento ha mostrato come il modello si comporti comunque bene, pur aumentando la dimensione dei sample in input alla rete, confermando dunque, la scelta di utilizzare come errore di ricostruzione lo score ***FM***;
Utilizzare come score di valutazione sul *Validation set*, lo score ***L2***, non sembra aver comunque cambiato le prestazioni finali del modello.
Sicuramente, lo studio sulla dimensione latente, ha portato invece alla luce un aspetto controintuitivo, ovvero aumentando la dimensione dello spazio ***z***, l'errore di ricostruzione medio non è migliorato anzi…
Questo può essere dovuto sicuramente a diversi fattori, uno di questi potrebbe essere il numero ridotto di epoche utilizzate, infatti, una dimensione latente maggiore, potrebbe portare il modello a convergere più lentamente e dunque ad avere un alto errore di ricostruzione, dopo poche epoche.
Lo score utilizzato ***FM***, sembra comunque essere una buona soluzione anche a questo problema, mantenendo curve ROC-AUC simili, nonostante le differenze di dimensione latente tra i modelli.