

UNIVERSITÀ DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e
Sistemistica



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
**INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA**

DIMES

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Progetto di Machine Deep Learning

Anomaly Detection

Mattia Presta
239051

Indice

1	Introduzione	2
2	Modelli Utilizzati	4
2.1	DRÆM	4
2.1.1	Vantaggi di DRÆM	6
2.1.2	Svantaggi di DRÆM	6
2.2	EfficientAD	7
2.2.1	Vantaggi di EfficientAD	9
2.2.2	Svantaggi di EfficientAD	9
3	Sviluppo	10
3.1	Datasets	10
3.2	Metriche	10
4	Risultati	11
4.1	DRÆM	11
4.1.1	MVTec AD	11
4.1.2	ViSa	14
4.1.3	WFDD	16
4.2	EfficientAD	17
4.2.1	MVTec AD	17
4.2.2	ViSa	20
4.2.3	WFDD	22
5	Confronto	23
6	Appendice	24
6.1	Undersampling Train Set	24
6.2	Nuovi parametri per il Train	25

1 Introduzione

L'anomaly detection (o rilevamento delle anomalie) nel contesto del machine learning e in particolare della visione artificiale è un compito fondamentale che mira a identificare e localizzare deviazioni significative dall'aspetto "normale" all'interno delle immagini. È un'area di ricerca in rapida crescita con diverse applicazioni pratiche.

Nel campo della visione artificiale, un'anomalia è una parte di un'immagine che si distacca notevolmente da ciò che è considerato tipico o atteso. Si distinguono principalmente due tipi di problemi di rilevamento delle anomalie:

- **Rilevamento di Anomalie a Livello di Immagine (General Anomaly Detection):** In questo caso, l'anomalia è l'intera immagine che differisce in modo significativo dall'insieme di immagini normali utilizzate per l'addestramento.
- **Rilevamento di Anomalie Superficiali (Surface Anomaly Detection):** Questo è un compito più specifico, dove le anomalie occupano solo una piccola frazione dei pixel di un'immagine e sono spesso molto simili alla distribuzione delle immagini di addestramento prive di anomalie. L'obiettivo principale è la localizzazione precisa di regioni specifiche dell'immagine che mostrano deviazioni dall'aspetto normale. Metodi come DRÆM si concentrano su questo tipo di anomalie

Le funzioni e gli scopi principali del rilevamento delle anomalie tramite machine learning includono:

- **Identificazione di Deviazioni Visive:** L'obiettivo primario è rilevare quelle regioni dell'immagine che deviano dall'aspetto previsto o "normale". Questo si traduce nell'individuazione di "difetti" o "irregolarità".
- **Localizzazione Precisa:** Spesso, non è sufficiente sapere che un'anomalia è presente, ma è essenziale identificarne la posizione esatta all'interno dell'immagine, spesso a livello di pixel. La capacità di localizzare le anomalie offre risultati di rilevamento più chiari e permette una post-elaborazione flessibile, come l'esclusione di difetti basati sulla loro dimensione.
- **Classificazione a Livello di Immagine:** Determinare se un'intera immagine contiene o meno un'anomalia, assegnando un punteggio complessivo all'immagine

Il rilevamento delle anomalie è una task fondamentale in vari settori industriali e scientifici, con un'enfasi crescente sull'efficienza computazionale per applicazioni in tempo reale:

- **Controllo Qualità Industriale e Ispezione Superficiale:** Questa è una delle applicazioni più comuni e critiche. Viene impiegato per rilevare difetti in oggetti manifatturati, come viti o superfici metalliche. Dataset standard come MVTec AD, VisA e WFDD sono benchmark per valutare i metodi in questo contesto.
- **Applicazioni in Tempo Reale:** Molte applicazioni richiedono che il rilevamento avvenga con latenze estremamente basse e throughput elevati. Questo è vitale in ambienti con alti tassi di produzione, dove un rilevamento tardivo può causare danni economici significativi o rischi per la salute. EfficientAD è un esempio di metodo progettato per operare con latenze di pochi millisecondi e un throughput di centinaia di immagini al secondo, rendendolo una soluzione economicamente valida per applicazioni reali.

I modelli **DRÆM**(Discriminatively Trained Reconstruction Anomaly Embedding Model) ed **EfficientAD** sono entrambi progettati per l'individuazione di anomalie visive di superficie, un compito cruciale nel campo della visione artificiale, in particolare per il controllo qualità industriale. Il loro obiettivo principale è identificare le regioni all'interno di un'immagine che deviano significativamente dall'aspetto normale o atteso.

DRÆM affronta questo problema trattandolo principalmente come un compito discriminativo. Il modello è composto da due sotto-reti: una sotto-rete ricostruttiva che impara a ricreare l'aspetto normale delle immagini e una sotto-rete discriminativa che apprende una rappresentazione congiunta dell'immagine originale e della sua ricostruzione priva di anomalie. DRÆM è addestrato esclusivamente su immagini prive di anomalie, utilizzando simulazioni semplici e generali di anomalie per apprendere un confine di decisione efficace. Questo approccio consente una localizzazione diretta e precisa delle anomalie senza la necessità di complessi passaggi di post-elaborazione.

D'altra parte, EfficientAD mira a un'individuazione delle anomalie estremamente rapida ed efficiente, raggiungendo latenze a livello di millisecondi. Utilizza un approccio studente-insegnante, in cui una rete "studente" impara a replicare le caratteristiche estratte da una rete "insegnante" pre-addestrata su immagini normali. La sua incapacità di replicare queste caratteristiche su immagini anomale consente l'individuazione. Inoltre, EfficientAD integra un autoencoder per rilevare le anomalie logiche, che riguardano violazioni di vincoli strutturali o combinazioni errate di oggetti normali. Questo duplice approccio lo rende efficace per diversi tipi di anomalie, mantenendo un'elevata efficienza computazionale per applicazioni in tempo reale.

2 Modelli Utilizzati

2.1 DRÆM

Il modello DRÆM (Discriminatively Trained Reconstruction Anomaly Embedding Model) è una soluzione avanzata per l'individuazione di anomalie visive di superficie. Il suo compito principale è identificare e localizzare con precisione le regioni all'interno di un'immagine che presentano deviazioni significative rispetto all'aspetto normale o atteso. DRÆM si distingue dagli approcci generativi precedenti trattando la rilevazione delle anomalie primariamente come un problema discriminativo.

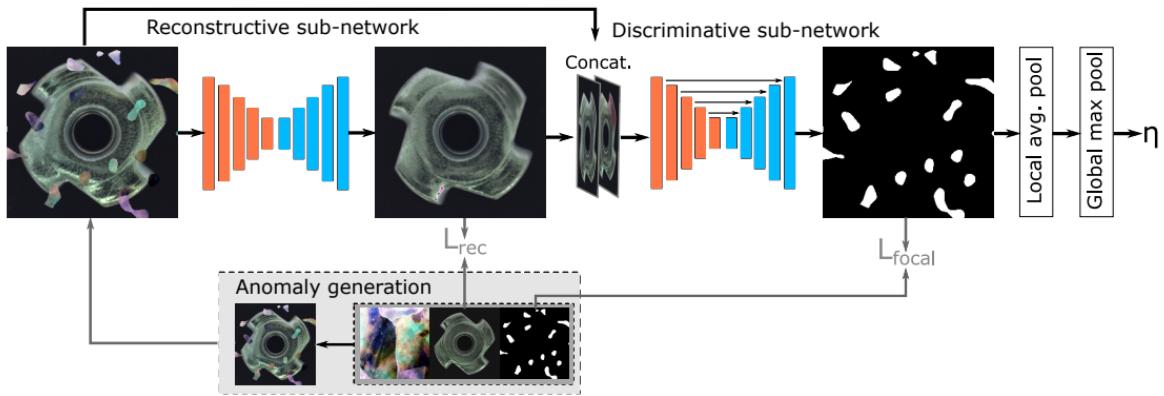


Figure 1: DRÆM Network

DRÆM è composto da due sotto-reti che lavorano in congiunzione, integrate in un'architettura addestrata end-to-end:

- **Sotto-rete Ricostruttiva (Reconstructive Sub-network):**

Questa sotto-rete è un'architettura encoder-decoder il cui scopo è imparare a rilevare implicitamente le anomalie con un contenuto semanticamente plausibile e privo di anomalie, mantenendo inalterate le regioni non anomale dell'immagine di input. In altre parole, trasforma i pattern locali di un'immagine di input in pattern più vicini alla distribuzione dei campioni normali.

- **Addestramento:** Viene addestrata per ricostruire l'immagine originale (I) da una versione artificialmente corrotta (I_a), ottenuta tramite un simulatore di anomalie
- **Funzione di perdita (Loss):** Utilizza una combinazione di perdita L2 e SSIM (Structural Similarity Index Measure) patch-based.

L'output di questa sotto-rete (I_r) viene concatenato con l'immagine di input originale (I) e fornito come input alla sotto-rete discriminativa successiva.

- **Sotto-rete Discriminativa (Discriminative Sub-network):**

Questa sotto-rete, che ha un'architettura simile a U-Net, impara una rappresentazione congiunta dell'immagine originale e della sua ricostruzione priva di anomalie. Il suo compito è produrre una mappa di rilevamento delle anomalie per-pixel ad alta fedeltà.

- **Input:** La concatenazione canale per canale dell'output della sotto-rete ricostruttiva (I_r) e dell'immagine di input originale (I)

- **Meccanismo di Rilevamento:** La chiave della sua efficacia risiede nel fatto che l’aspetto congiunto di I e I_r differisce significativamente in presenza di anomalie, fornendo le informazioni necessarie per la segmentazione. A differenza dei metodi basati sulla ricostruzione che spesso usano funzioni di similarità pre-definite come SSIM per confrontare l’originale e la ricostruzione, la sotto-rete discriminativa di DRÆM apprende automaticamente la misura di distanza appropriata.
- **Funzione di perdita (Loss):** Applica la Focal Loss (L_{seg}) sul suo output per aumentare la robustezza nella segmentazione accurata degli esempi difficili.
- **Total Loss:** L’addestramento di DRÆM si basa su una funzione di perdita totale che combina gli obiettivi di segmentazione e ricostruzione delle due sotto-reti:

$$L_{total} = L_{rec} + L_{seg}$$

L’output finale della sotto-rete discriminativa è una maschera di rilevamento delle anomalie a livello di pixel.

DRÆM è un modello robusto e altamente performante per l’individuazione di anomalie di superficie, che affronta la sfida della mancanza di dati anomali di training in modo innovativo e molto efficace, superando le limitazioni dei metodi puramente generativi.

2.1.1 Vantaggi di DRÆM

- **Localizzazione Diretta e Accurata:** DRÆM consente la localizzazione diretta e precisa delle anomalie senza la necessità di passaggi complicati di post-elaborazione dell'output della rete. Le mappe di anomalia ottenute sono molto dettagliate e vicine ai ground truth.
- **Addestramento su Dati Normali:** Il modello è addestrato esclusivamente su immagini prive di anomalie, un requisito cruciale nelle applicazioni industriali dove le anomalie sono rare o sconosciute.
- **Robustezza alle Anomalie Simulate:** Non richiede che l'aspetto delle anomalie simulate corrisponda strettamente alle anomalie reali, rendendolo versatile e generalizzabile
- **Apprendimento Discriminativo Esplicito:** A differenza dei metodi generativi che si basano solo sulla capacità di ricostruire dati normali, DRÆM apprende esplicitamente un confine di decisione tra esempi normali e anomali

2.1.2 Svantaggi di DRÆM

- **Anomalie Sottili:** Sebbene molto efficace, alcune anomalie particolarmente difficili, vicine alla distribuzione normale dell'immagine (ad esempio, l'assenza di una parte dell'oggetto, come un filo del transistor tagliato, dove le caratteristiche di sfondo sono comuni), possono essere difficili da rilevare per DRÆM
- **Latenza/Efficienza Computazionale:** ha una latenza di 17 ms, suggerendo che DRÆM potrebbe non essere il metodo più rapido in confronto a soluzioni focalizzate sull'efficienza computazionale come EfficientAD per applicazioni in tempo reale estremamente critiche.

2.2 EfficientAD

EfficientAD (Efficient Anomaly Detection) è un modello avanzato progettato per l' individuazione di anomalie visive di superficie, con un'enfasi particolare sull'efficienza computazionale e la velocità di inferenza. Il suo compito principale è identificare le regioni di un'immagine che si discostano dall'aspetto normale, non solo per anomalie strutturali come macchie o graffi, ma anche per anomalie logiche, che riguardano violazioni di vincoli strutturali o combinazioni errate di oggetti normali. EfficientAD si propone di stabilire nuovi standard sia per le prestazioni di rilevamento che per la latenza, consentendo un'elaborazione in tempo reale.

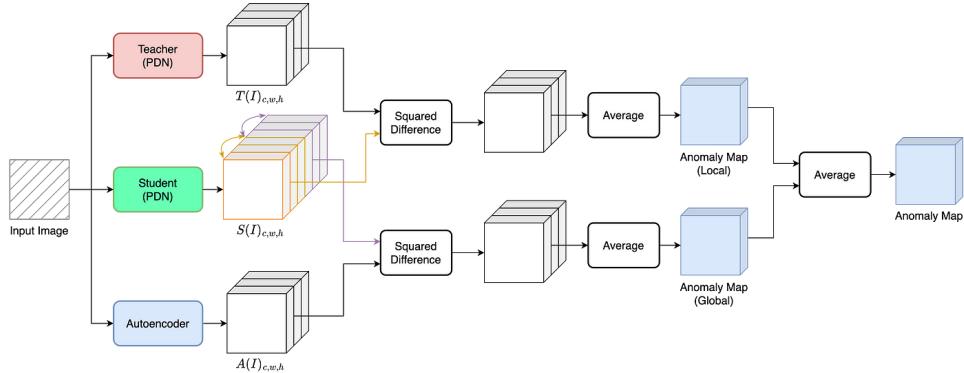


Figure 2: EfficientAD Network

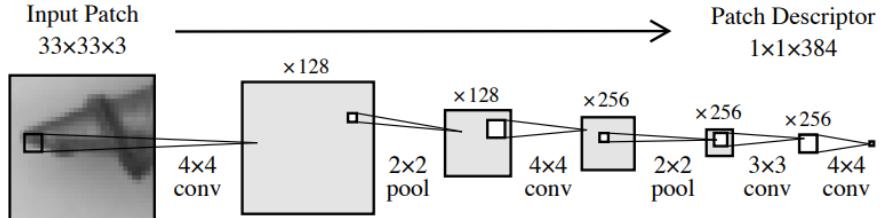


Figure 3: PDN

EfficientAD è costituito da un approccio ibrido che combina un modello studente-insegnante con un autoencoder, addestrati congiuntamente in modo efficiente.

- **Rete di Descrizione delle Patch (Patch Description Network - PDN):** La PDN è un'architettura di rete neurale leggera e profonda che funge da estrattore di caratteristiche efficiente (Features Extraction). È progettata per generare descrittori di patch espressivi con una latenza estremamente bassa, elaborando un'immagine in meno di un millisecondo su una GPU moderna
 - **Architettura:** Consiste in poche (ad esempio, quattro) strati convoluzionali e utilizza strati di downsampling (come l'average pooling con stride) in anticipo nella pipeline per ridurre la dimensione delle mappe di caratteristiche e ottimizzare il tempo di esecuzione e l'utilizzo della memoria. Ogni neurone di output del PDN ha un campo ricettivo ben definito (ad esempio, 33x33 pixel), garantendo che un'anomalia in una parte dell'immagine non innesci vettori di caratteristiche anomale in parti distanti, migliorando la localizzazione
 - **Addestramento (Distillazione):** Il PDN viene addestrato per distillare le caratteristiche da una rete di classificazione profonda pre-addestrata (ad esempio, WideResNet-101). Ciò significa che la PDN apprende a replicare l'output di un modello più grande e complesso su immagini normali (ad esempio, ImageNet), trasferendo la capacità di estrazione delle caratteristiche in un'architettura più leggera.

- **Approccio Studente-Insegnante (Student-Teacher - S-T):** Questo componente rileva le anomalie strutturali imparando a replicare le caratteristiche estratte da un’immagine normale da parte di una rete “insegnante” pre-addestrata. La rete “studente” ha la stessa architettura leggera del PDN, garantendo una bassa latenza complessiva.
 - **Meccanismo di Rilevamento:** Poiché la rete studente è addestrata solo su immagini prive di anomalie, generalmente fallisce nel mimare l’insegnante su immagini anomale. Una grande differenza tra gli output dell’insegnante e dello studente indica la presenza di un’anomalia.
 - **Asimmetria Indotta dalla Perdita:** A differenza di altri approcci S-T che usano asimmetrie architettoniche, EfficientAD introduce un’asimmetria indotta dalla funzione di perdita, che non influisce sui costi computazionali in fase di inferenza. Questo si ottiene tramite:
 - * **Hard Feature Loss:** Limita la perdita dello studente alle parti più ”difficili” dell’immagine, dove lo studente sta mimando meno l’insegnante. Questo impedisce allo studente di generalizzare la sua capacità di imitazione anche alle anomalie.
 - * **Penalità sulle Immagini di Pre-addestramento:** Una penalità viene aggiunta alla funzione di perdita per ostacolare lo studente dal generalizzare la sua imitazione dell’insegnante a immagini fuori distribuzione (ad esempio, provenienti dal dataset di pre-addestramento dell’insegnante).
- **Autoencoder per Anomalie Logiche:** Un autoencoder è integrato in EfficientAD per rilevare le anomalie logiche, che implicano violazioni di vincoli globali o strutturali, come oggetti mancanti, fuori posto o in quantità errata.
L’autoencoder è addestrato a imparare i vincoli logici delle immagini normali. A differenza del modello studente-insegnante basato su patch, l’autoencoder codifica e decodifica l’intera immagine attraverso un ”collo di bottiglia” latente. Quando si verificano anomalie logiche, l’autoencoder fallisce nel generare il codice latente corretto per ricostruire l’immagine nello spazio delle feature dell’insegnante, segnalando l’anomalia.
L’autoencoder è efficientemente integrato con il modello studente-insegnante. I suoi parametri sono influenzati sia dalla perdita di ricostruzione che da una perdita che confronta la sua ricostruzione con una parte dell’output dello studente.

2.2.1 Vantaggi di EfficientAD

- **Efficienza Computazionale Straordinaria:** Raggiunge latenze a livello di milisecondi (ad esempio, 2.2 ms per EfficientAD-S su una NVIDIA RTX A6000 GPU) e un'elevata produttività (oltre 600 immagini al secondo), rendendolo ideale per applicazioni in tempo reale.

Questo è un miglioramento significativo rispetto ad altri metodi, con una riduzione della latenza fino a 24 volte.

- **Prestazioni All'Avanguardia:** Stabilisce nuovi standard per il rilevamento e la localizzazione delle anomalie, superando significativamente altri metodi su dataset industriali come MVtec AD, VisA. La sua efficacia è dimostrata sia per le anomalie strutturali che per quelle logiche.
- **Addestramento su Immagini Normali:** Il modello è addestrato esclusivamente su immagini prive di anomalie, un requisito fondamentale nelle applicazioni industriali dove i dati anomali sono scarsi o inesistenti.
- **Robustezza:** È robusto alla scelta del backbone di distillazione (es. WideResNet-101, ResNeXt-101, DenseNet-201)

2.2.2 Svantaggi di EfficientAD

- **Anomalie Logiche a Grana Fine:** Nonostante l'efficacia nel rilevare le anomalie logiche, alcune anomalie a grana estremamente fine (ad esempio, una vite di due millimetri troppo lunga) rimangono una sfida e potrebbero richiedere metodi di metrologia tradizionali.
- **Necessità di Addestramento:** A differenza di alcuni metodi basati su kNN che non richiedono un addestramento esplicito del modello, EfficientAD richiede un processo di addestramento (che dura circa venti minuti nella configurazione sperimentale) per l'autoencoder e il modello studente-insegnante

3 Sviluppo

Seguendo le indicazioni dei paper, ho replicato l’addestramento e l’inferenza dei due modelli tramite l’utilizzo della libreria **Anomalib** implementata nell’ambiente di sviluppo notebook fornito da **Kaggle** utilizzando una scheda GPU Nvidia P-100. Per aggevolare il confronto sono stati ulizzati i parametri di base e lo stesso numero di epoche di addestramento.

3.1 Datasets

Per entrambi i modelli sono stati eseguiti addestramento e test su tre dataset benckmark di anomaly detection:

- **MVTec AD**: set di dati per il benchmarking dei metodi di rilevamento delle anomalie con particolare attenzione all’ispezione industriale. Contiene oltre 5000 immagini ad alta risoluzione suddivise in quindici diverse categorie di oggetti e texture. Ogni categoria comprende una serie di immagini di formazione prive di difetti e una serie di immagini di test con vari tipi di difetti, nonché immagini senza difetti.
- **ViSa (Visual Anomaly Dataset)**: contiene 12 sottoinsiemi corrispondenti a 12 oggetti diversi. Ci sono 10.821 immagini con 9.621 campioni normali e 1.200 anomali. Quattro sottoinsiemi sono diversi tipi di circuiti stampati (PCB) con strutture relativamente complesse contenenti transistor, condensatori, chip, ecc.
- **WFDD (Woven Fabric Defect Detection)**: set di dati con particolare attenzione all’ispezione tessile. Include 4101 immagini di tessuto, suddivise in 4 categorie, ogni categoria contiene difetti a forma di blocco, puntiformi e lineari, con annotazioni a livello di pixel.

3.2 Metriche

Nel contesto dell’anomaly detection su immagini, specialmente quando si lavora con modelli come EfficientAD e DRAEM, è fondamentale utilizzare metriche che valutino in modo accurato sia la capacità di individuare la presenza di anomalie sia quella di localizzarle correttamente all’interno dell’immagine. Per questo motivo, si utilizzano comunemente quattro metriche: Image-AUROC, Image-F1, Pixel-AUROC e Pixel-F1. Ciascuna di queste offre un punto di vista diverso sulla performance del modello, e tutte insieme permettono un confronto più completo e significativo tra approcci diversi.

- **Image AUROC (Area Under the Receiver Operating Characteristic Curve a livello di immagine)** misura quanto bene un modello è in grado di distinguere tra immagini anomale e normali. In pratica, il modello assegna a ciascuna immagine un punteggio di anomalia, e questa metrica valuta la capacità di separare correttamente le classi.
- **Image F1**: tiene conto sia della precisione (cioè quante immagini classificate come anomale lo sono davvero) sia del recall (quante delle immagini effettivamente anomale sono state rilevate). Cioè la media armonica tra precisione e recall nel classificare un’immagine come anomala o normale.
- **Pixel AUROC**: valuta la capacità del modello di distinguere tra pixel normali e pixel anomali all’interno dell’immagine. Questa metrica è cruciale quando si analizzano modelli che, oltre a individuare se un’immagine è anomala, devono anche localizzare con precisione l’area in cui si trova l’anomalia.
- **Pixel F1**: misura la qualità della segmentazione a livello di pixel, confrontando le aree anomale rilevate dal modello con le annotazioni reali. Essendo una media armonica tra precisione e richiamo dei pixel classificati come anomali, questa metrica è altamente indicativa della capacità del modello di identificare correttamente la forma e la dimensione dell’anomalia.

4 Risultati

Di seguito vengono riportati i risultati ottenuti dai test effettuati sui dataset sopra citati, riguardanti le metriche discusse e la velocità di inferenza dei modelli implementati.

4.1 DRÆM

4.1.1 MVTec AD

Table 1: Risultati MVTec AD DRÆM

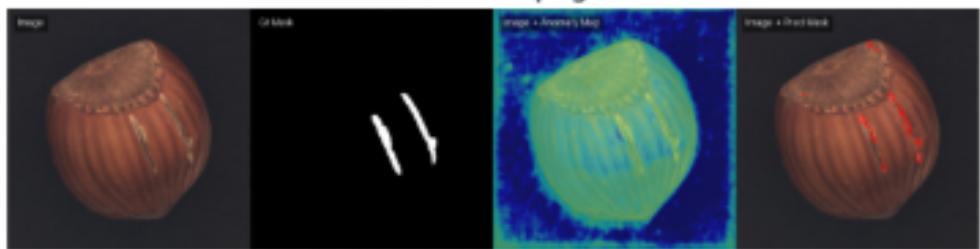
Category	Image_AUROC	Image_F1	Pixel_AUROC	Pixel_F1
bottle	0.9056	0.9037	0.8150	0.4626
cable	0.6362	0.7660	0.4679	0.0971
capsule	0.6562	0.9030	0.7937	0.1520
carpet	0.5317	0.8614	0.5934	0.1338
grid	0.3885	0.8358	0.4831	0.0137
hazelnut	0.8218	0.8323	0.8712	0.3478
leather	0.4918	0.8505	0.4969	0.2184
metal_nut	0.6224	0.8889	0.7680	0.3706
pill	0.5584	0.9121	0.8702	0.3803
screw	0.7393	0.8642	0.9007	0.0943
tile	0.8701	0.8729	0.7825	0.4296
toothbrush	0.6500	0.8169	0.8768	0.1527
transistor	0.7025	0.6275	0.5783	0.1759
wood	0.9175	0.9280	0.7937	0.0000
zipper	0.7831	0.9048	0.6030	0.1400

Il modello mostra prestazioni accettabili a livello di immagine (Image_AUROC 0.38-0.92) ma risulta poco preciso nella localizzazione delle anomalie (Pixel_F1 <0.5 in tutte le categorie), con particolare difficoltà su texture complesse come grid e wood. Le performance variano significativamente tra categorie, evidenziando una dipendenza critica dalle caratteristiche specifiche degli oggetti analizzati.

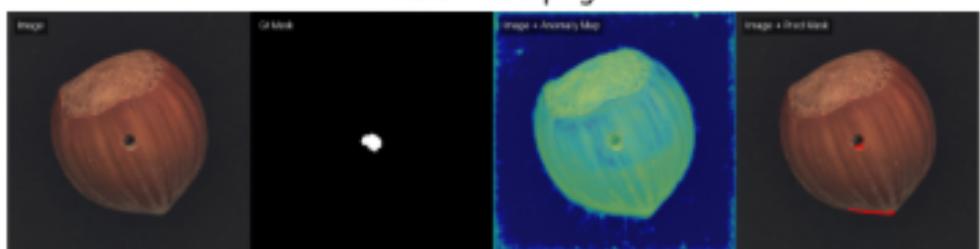
Table 2: Velocità di inferenza MVTec AD DRÆM

Categoria	Immagini	Tempo Totale (s)	Tempo/Immagine (s)	Throughput (img/s)
bottle	83	13.91	0.1675	5.97
cable	150	28.62	0.1908	5.24
capsule	132	28.21	0.2137	4.68
carpet	117	20.39	0.1743	5.74
grid	78	12.51	0.1604	6.24
hazelnut	110	22.70	0.2064	4.85
leather	124	21.75	0.1754	5.70
metal_nut	111	18.59	0.1676	6.18
pill	167	29.32	0.1756	5.70
screw	160	22.96	0.1435	6.97
tile	117	18.55	0.1586	6.31
toothbrush	42	10.39	0.2473	4.04
transistor	100	21.20	0.2120	4.72
wood	79	17.31	0.2192	4.56
zipper	151	21.29	0.1410	7.09

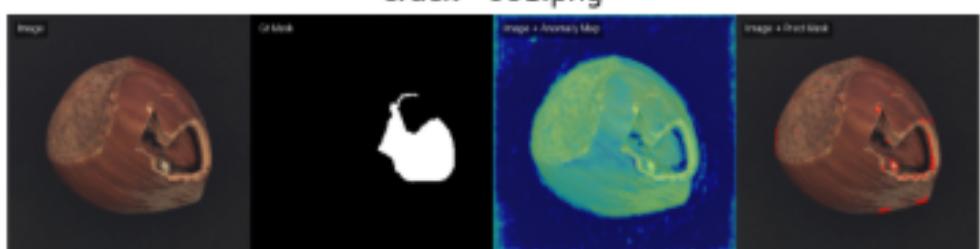
cut - 001.png



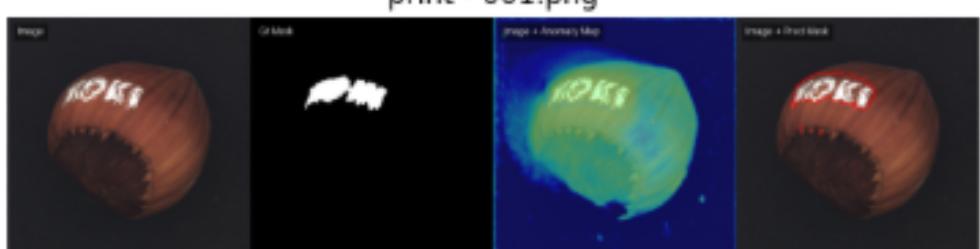
hole - 001.png



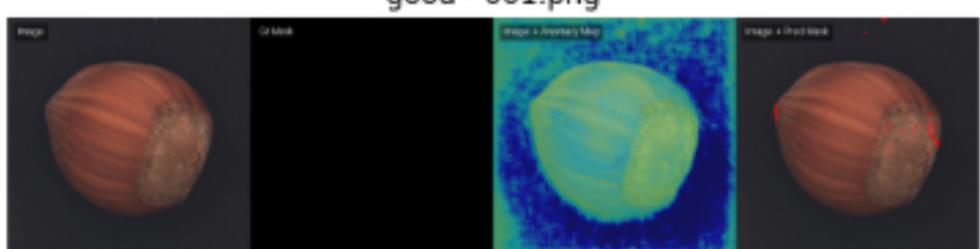
crack - 001.png



print - 001.png

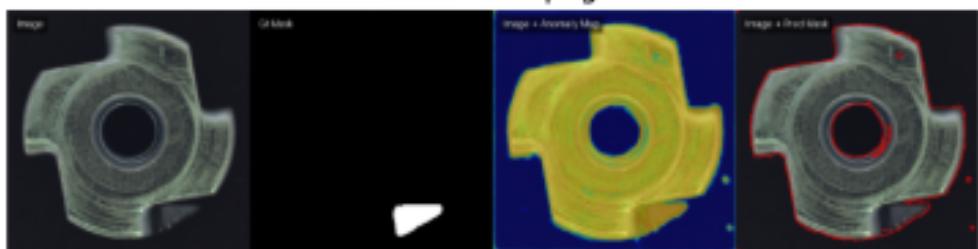


good - 001.png

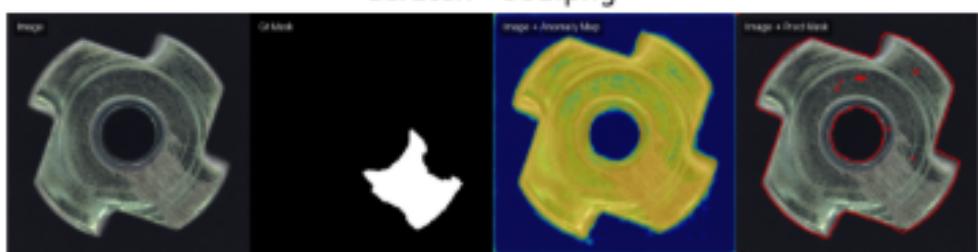


12
Figure 4: Hazelnut DR&EM

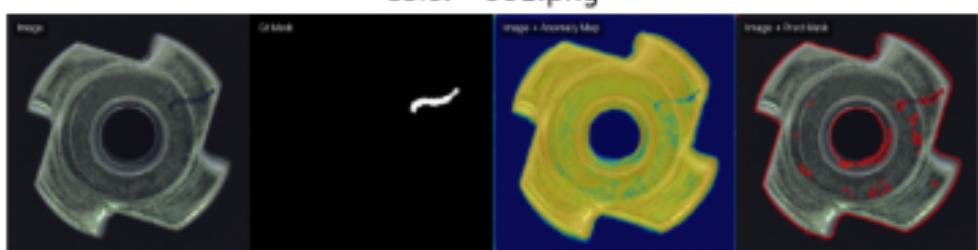
bent - 001.png



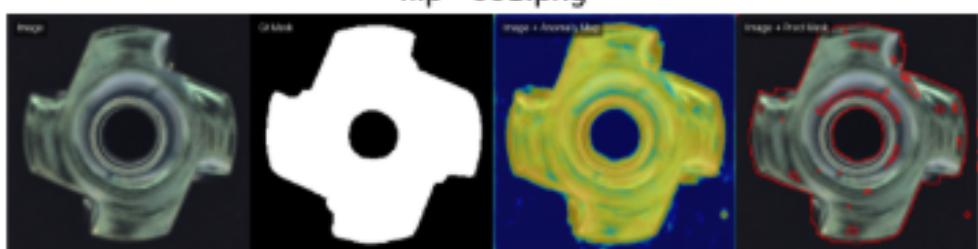
scratch - 001.png



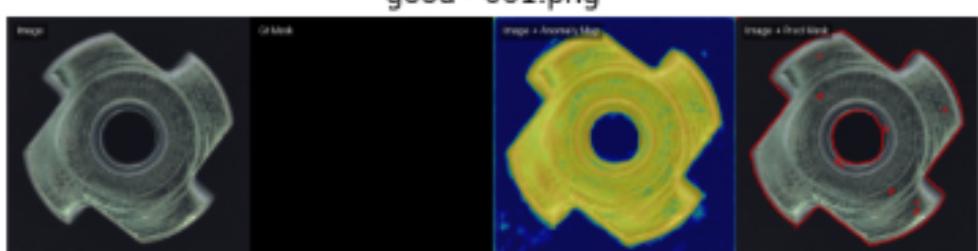
color - 001.png



flip - 001.png



good - 001.png



4.1.2 ViSa

Table 3: Risultati ViSa DRÆM

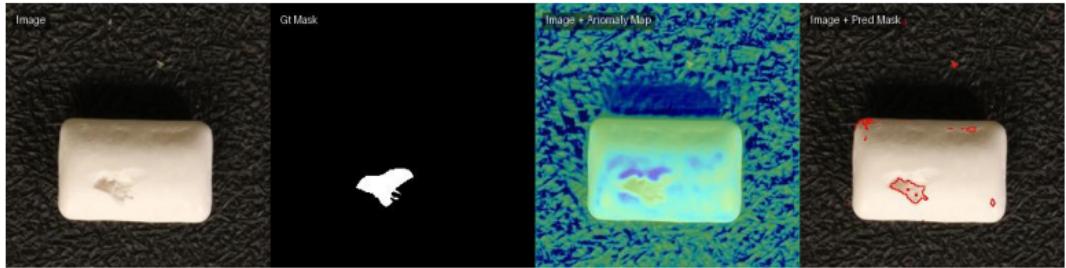
Category	Image_AUROC	Image_F1	Pixel_AUROC	Pixel_F1
candle	0.8406	0.7759	0.7797	0.1863
capsules	0.4890	0.7597	0.6072	0.0488
cashew	0.9152	0.9346	0.6798	0.0121
chewinggum	0.9032	0.8936	0.8636	0.2648
fryum	0.7864	0.7925	0.7410	0.1119
macaroni1	0.7658	0.7158	0.9458	0.1321
macaroni2	0.4948	0.6269	0.9104	0.0312
pcb1	0.8292	0.7216	0.4947	0.0313
pcb2	0.7044	0.6667	0.6675	0.0214
pcb3	0.7388	0.7273	0.8827	0.1715
pcb4	0.9180	0.8807	0.8856	0.1784
pipe_fryum	0.5508	0.7903	0.9463	0.0906

I risultati mostrano prestazioni disomogenee tra le categorie, con Image_AUROC che varia significativamente (da 0.489 a 0.918) e Pixel_F1 generalmente basso. Questo suggerisce che il modello DRÆM su ViSa ha una capacità accettabile di rilevamento anomalie a livello di immagine ma mostra limitazioni nella localizzazione precisa a livello pixel, con particolari difficoltà nelle categorie più complesse come capsules e cashew.

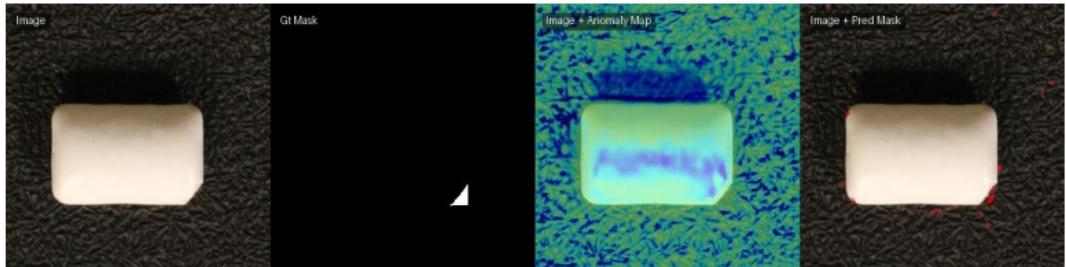
Table 4: Velocità di inferenza ViSa DRÆM

Categoria	Immagini	Tempo Totale (s)	Tempo/Immagine (s)	Throughput (img/s)
candle	100	10.38	0.1038	9.63
capsules	80	8.97	0.1121	8.92
cashew	75	8.79	0.1173	0.0121
chewinggum	75	8.43	0.1124	8.90
fryum	75	8.41	0.1122	8.91
macaroni1	100	10.38	0.1038	9.63
macaroni2	100	10.06	0.1006	9.94
pcb1	100	10.22	0.1022	9.79
pcb2	100	10.26	0.1026	9.75
pcb3	101	10.21	0.1011	9.89
pcb4	101	10.58	0.1048	9.54
pipe_fryum	75	8.28	0.1104	9.06

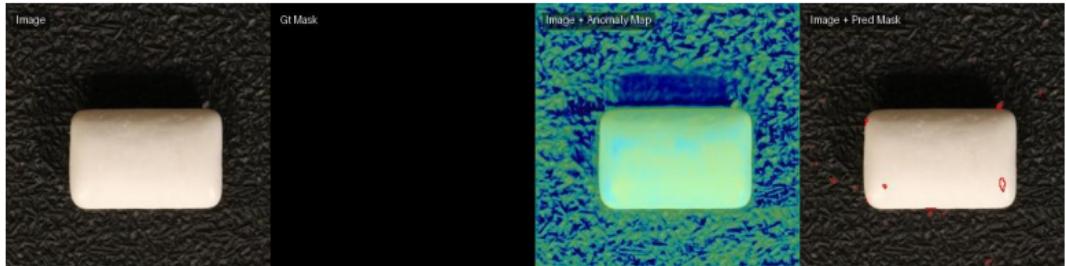
017



025



487



405

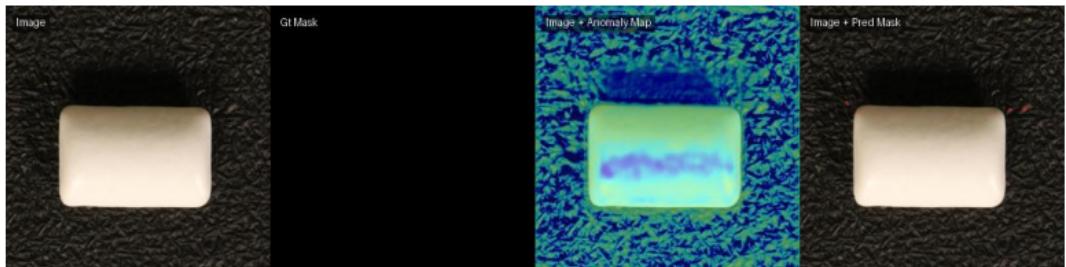


Figure 6: Chewinggum DRÆM

4.1.3 WFDD

Table 5: Risultati WFDD DRÆM

Category	Image_AUROC	Image_F1	Pixel_AUROC	Pixel_F1
grid_cloth	0.8801	0.8571	0.7872	0.1361
pink_flower	0.7441	0.6857	0.4588	0.0483
yellow_cloth	0.9735	0.8932	0.5272	0.0423

Il modello DRÆM su WFDD mostra buone prestazioni di rilevamento a livello di immagine (Image_AUROC fino a 0.97), ma una limitata capacità di localizzazione precisa delle anomalie (Pixel_F1 < 0.14 in tutte le categorie). In particolare, la categoria pink_flower presenta le performance più deboli sia a livello di immagine che pixel, suggerendo difficoltà con texture floreali complesse.

Table 6: Velocità di inferenza WFDD DRÆM

Categoria	Immagini	Tempo Totale (s)	Tempo/Immagine (s)	Throughput (img/s)
grid_cloth	56	7.71	0.1376	7.27
pink_flower	32	5.12	0.1600	6.25
yellow_cloth	96	11.80	0.1229	8.14

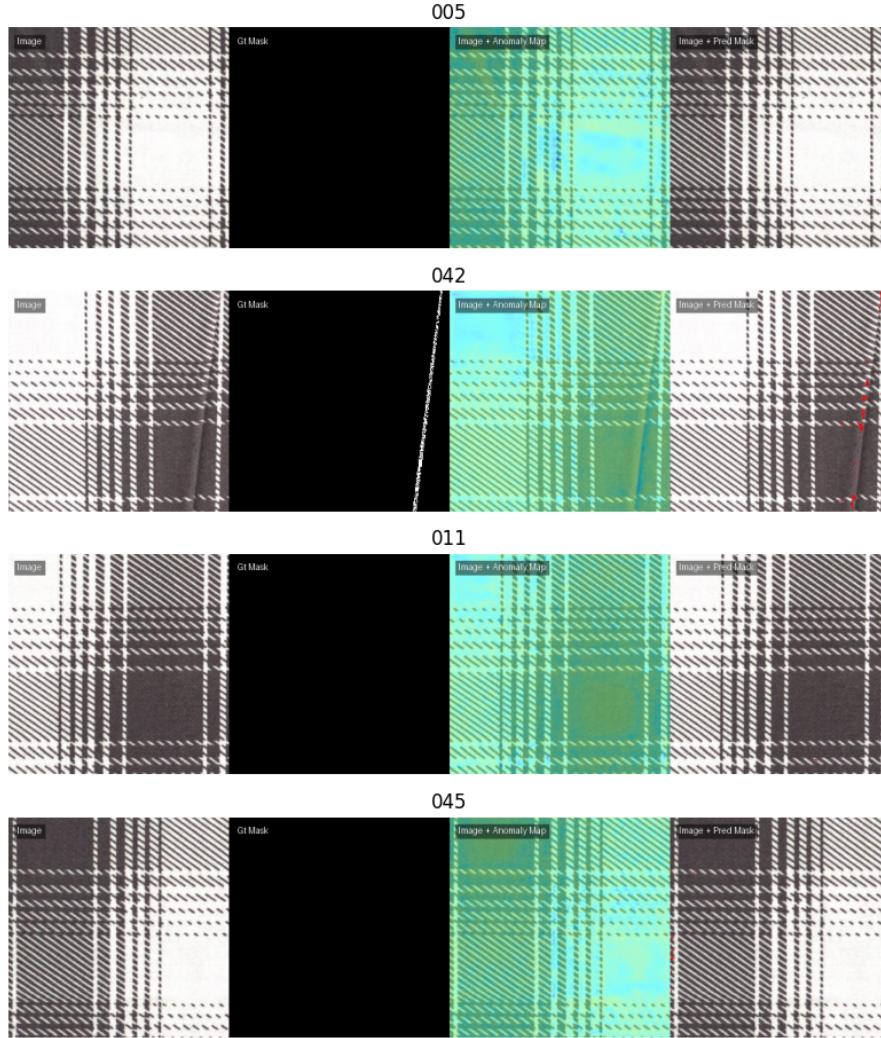


Figure 7: Grid Cloth DRÆM

4.2 EfficientAD

4.2.1 MVTec AD

Table 7: Risultati MVTec AD EfficientAD

Category	Image_AUROC	Image_F1	Pixel_AUROC	Pixel_F1
bottle	1.0000	0.9920	0.9778	0.7765
cable	0.9421	0.8852	0.9748	0.6534
capsule	0.7327	0.9138	0.8898	0.4018
carpet	0.9920	0.9714	0.9560	0.6711
grid	0.9983	0.9821	0.9332	0.5200
hazelnut	0.9154	0.8714	0.9308	0.5971
leather	0.9898	0.9730	0.9740	0.5616
metal_nut	0.9741	0.9524	0.9577	0.7905
pill	0.9686	0.9716	0.9842	0.6899
screw	0.8981	0.9008	0.9808	0.4765
tile	0.9993	0.9881	0.8937	0.7180
toothbrush	0.9000	0.8788	0.9387	0.3755
transistor	0.7900	0.6733	0.9062	0.5634
wood	0.9579	0.9431	0.8653	0.5417
zipper	0.9790	0.9750	0.9562	0.6934

EfficientAD dimostra prestazioni eccellenti sul dataset MVTec AD, raggiungendo $\text{Image_AUROC} > 0.99$ in 7 categorie e $\text{Pixel_F1} > 0.7$ in 5 categorie, con una superiorità particolarmente marcata per oggetti strutturati come bottle e metal_nut. Il modello mostra comunque qualche difficoltà con categorie complesse come transistor e capsule, dove le metriche pixel rimangono sotto lo 0.6, indicando che le sfide nella segmentazione precisa permangono per alcuni tipi di anomalie.

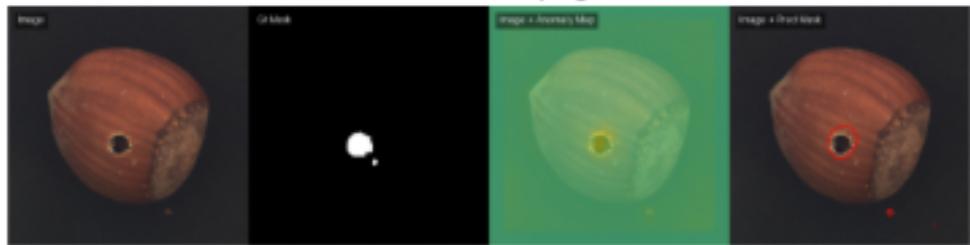
Table 8: Velocità di inferenza MVTec AD EfficientAD

Categoria	Immagini	Tempo Totale (s)	Tempo/Immagine (s)	Throughput (img/s)
bottle	83	10.32	0.1244	8.04
cable	150	25.35	0.1690	5.92
capsule	132	24.75	0.1875	5.33
carpet	117	17.04	0.1457	6.87
grid	78	10.51	0.1347	7.42
hazelnut	110	19.46	0.1769	5.65
leather	124	19.24	0.1552	6.44
metal_nut	111	15.26	0.1327	7.53
pill	167	24.58	0.1472	6.79
screw	160	18.78	0.1174	8.52
tile	117	16.52	0.1412	7.08
toothbrush	42	8.39	0.1997	5.01
transistor	100	18.59	0.1859	5.38
wood	79	15.25	0.1930	5.18
zipper	151	16.91	0.1120	8.93

cut - 003.png



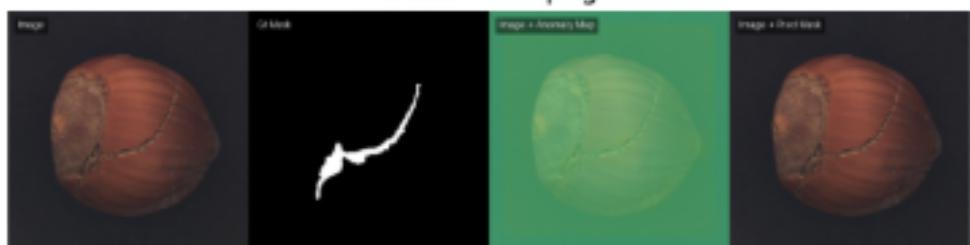
hole - 003.png



print - 003.png



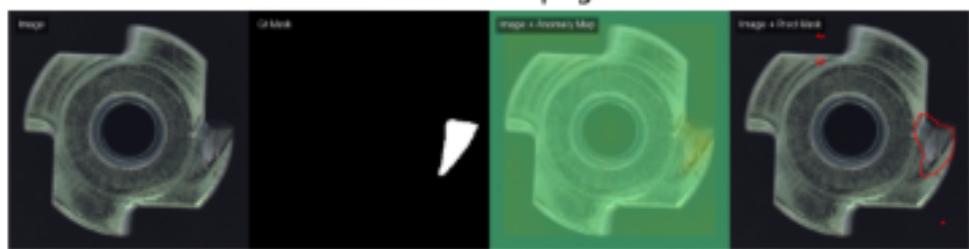
crack - 003.png



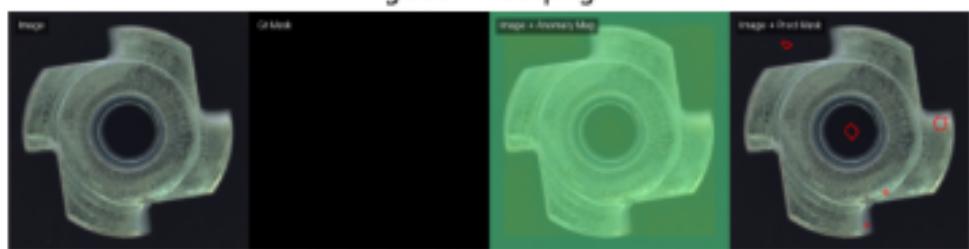
good - 020.png



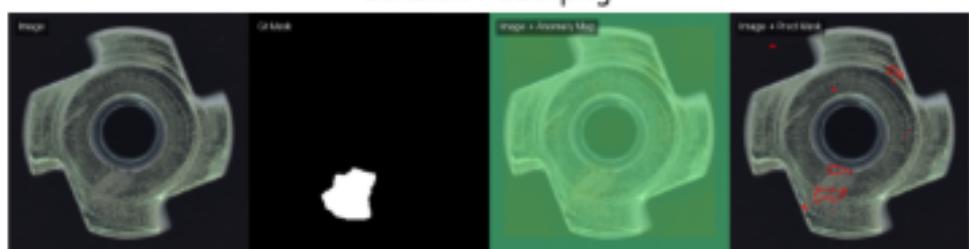
bent - 020.png



good - 020.png



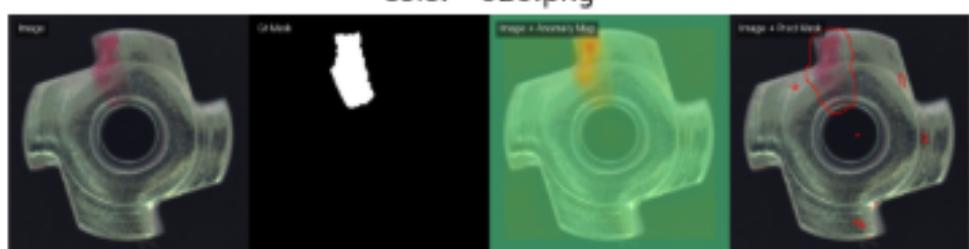
scratch - 020.png



flip - 020.png



color - 020.png



4.2.2 ViSa

Table 9: Risultati ViSa EfficientAD

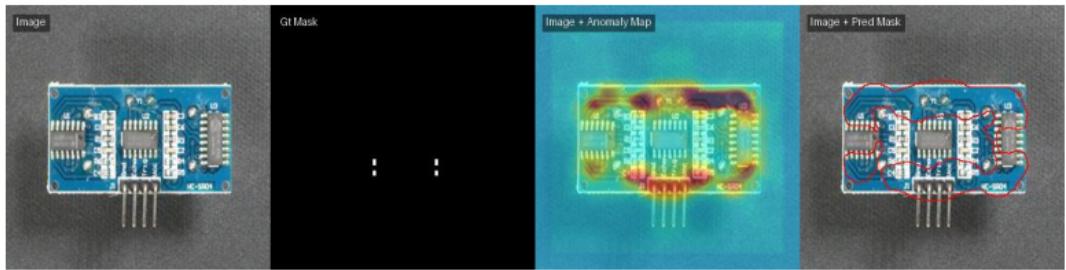
Category	Image_AUROC	Image_F1	Pixel_AUROC	Pixel_F1
candle	0.7792	0.6737	0.9279	0.0893
capsules	0.6527	0.7258	0.9682	0.3029
cashew	0.9280	0.8673	0.9209	0.6720
chewinggum	0.9632	0.9592	0.9813	0.3778
fryum	0.8856	0.8421	0.9297	0.4119
macaroni1	0.9472	0.8824	0.9938	0.1837
macaroni2	0.7760	0.6917	0.9870	0.1501
pcb1	0.9596	0.8542	0.9889	0.5167
pcb2	0.9456	0.9126	0.9799	0.2475
pcb3	0.9898	0.9412	0.9927	0.1493
pcb4	0.9706	0.9346	0.9653	0.4732
pipe_fryum	0.9272	0.8785	0.9941	0.7261

Il modello EfficientAD su ViSa mostra ottime capacità di rilevamento a livello di immagine (Image_AUROC fino a 0.9898) e localizzazione (Pixel_AUROC superiore a 0.92 in 11/12 categorie), ma presenta prestazioni disomogenee nella segmentazione fine, con Pixel_F1 che varia significativamente da 0.0893 (candle) a 0.7261 (pipe_fryum). Le performance più deboli si osservano nelle categorie con anomalie sottili (come candle e macaroni) dove il Pixel_F1 risulta inferiore a 0.2, suggerendo difficoltà nel segmentare piccoli difetti nonostante l'accurato rilevamento a livello globale.

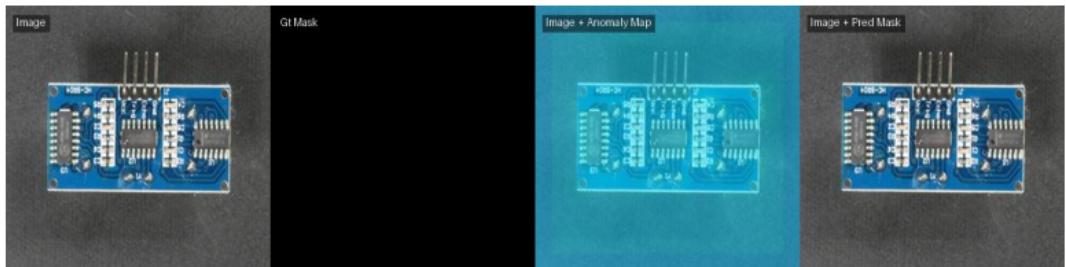
Table 10: Velocità di inferenza ViSa EfficientAD

Categoria	Immagini	Tempo Totale (s)	Tempo/Immagine (s)	Throughput (img/s)
candle	100	6.93	0.0693	14.43
capsules	80	6.01	0.0751	13.32
cashew	75	6.00	0.0800	12.50
chewinggum	75	5.75	0.0767	13.03
fryum	75	5.72	0.0762	13.11
macaroni1	100	7.37	0.0737	13.57
macaroni2	100	7.20	0.0720	13.89
pcb1	100	7.18	0.0718	13.93
pcb2	100	7.17	0.0717	13.94
pcb3	101	7.36	0.0729	13.72
pcb4	101	7.09	0.0702	14.25
pipe_fryum	75	5.69	0.0758	13.19

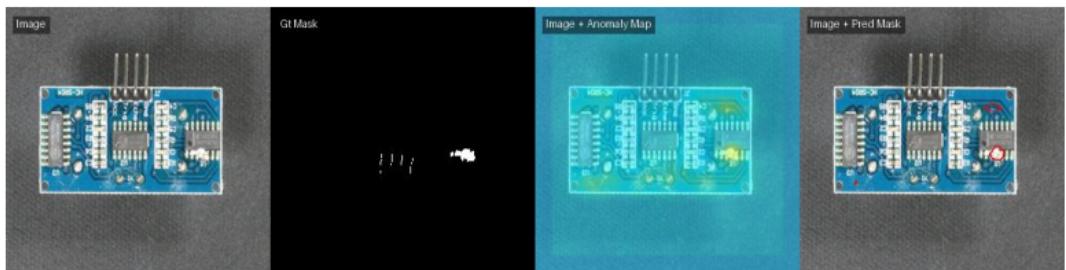
047



952



037



615

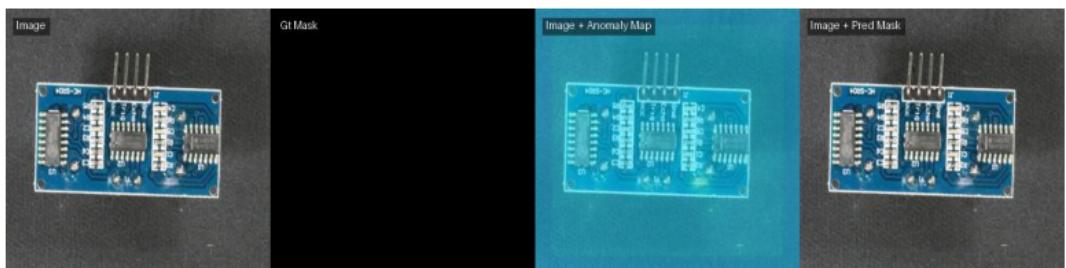


Figure 10: pcb2 EfficientAD

4.2.3 WFDD

Table 11: Risultati WFDD EfficientAD

Category	Image_AUROC	Image_F1	Pixel_AUROC	Pixel_F1
grid_cloth	0.8852	0.7857	0.8061	0.2003
pink_flower	0.6484	0.7111	0.8835	0.4811
yellow_cloth	0.9974	0.9691	0.8403	0.4060

EfficientAD sul dataset WFDD mostra un'ottima capacità di rilevamento per yellow_cloth (Image_AUROC 0.9974), ma prestazioni inferiori per pink_flower (0.6484), evidenziando una sensibilità alle caratteristiche specifiche delle categorie. Nonostante i buoni valori di Pixel_AUROC (tutti superiori a 0.8), i bassi risultati di Pixel_F1 (inferiori a 0.5) rivelano difficoltà nella segmentazione precisa delle anomalie, specialmente per grid_cloth (0.2003), suggerendo che il modello identifica correttamente le regioni anomale ma con limitata accuratezza nei confini.

Table 12: Velocità di inferenza WFDD EfficientAD

Categoria	Immagini	Tempo Totale (s)	Tempo/Immagine (s)	Throughput (img/s)
grid_cloth	56	6.99	0.1248	8.02
pink_flower	32	4.89	0.1527	6.55
yellow_cloth	96	10.61	0.1105	9.05

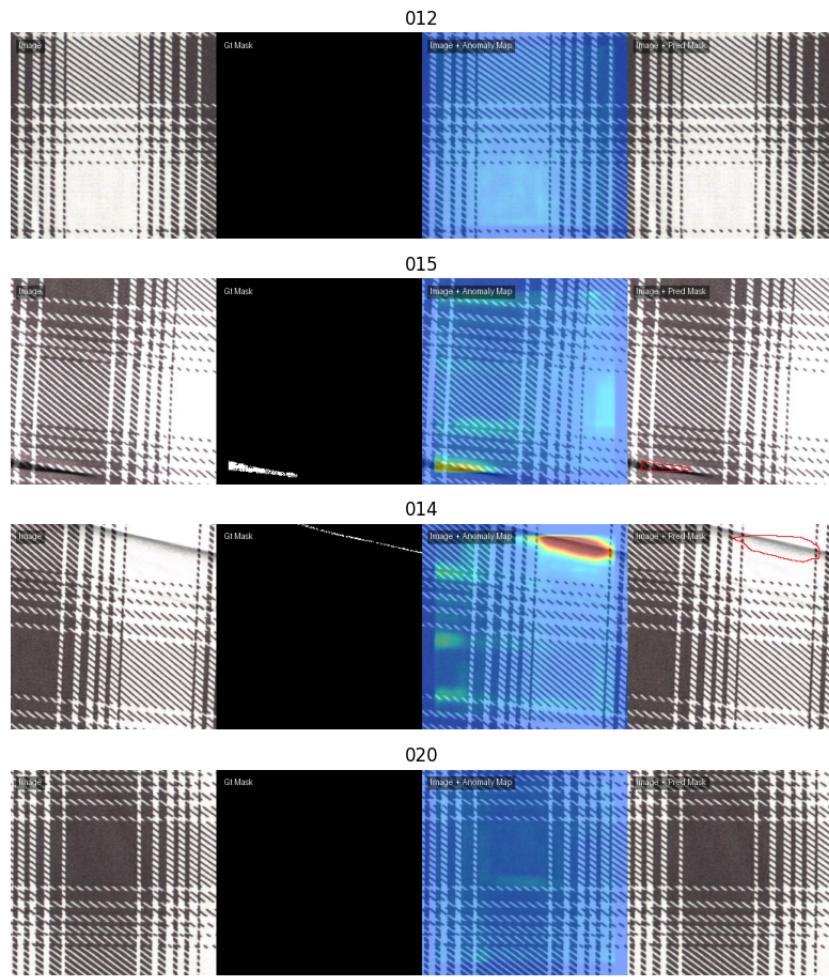


Figure 11: Grid Cloth EfficientAD

5 Confronto

In questa sezione si mettono a paragone i risultati dei rispettivi modelli, evidenziando i migliori risultati riguardanti le metriche e la velocità di inferenza ottenuta.

Table 13: Confronto modelli per dataset

	Img-AUROC	Img-F1	Pixel-AUROC	Pixel-F1
MVTec				
DRÆM	0.69	0.86	0.69	0.21
EfficientAD	0.94	0.92	0.94	0.60
ViSa				
DRÆM	0.74	0.77	0.78	0.11
EfficientAD	0.89	0.84	0.97	0.37
WFDD				
DRÆM	0.87	0.81	0.59	0.08
EfficientAD	0.84	0.82	0.84	0.36

Dal confronto emergono chiare differenze nelle prestazioni dei modelli.

EfficientAD domina su MVTec e ViSa, mostrando una superiorità sistematica in tutte le metriche, specialmente nell'AUROC a livello di pixel (0.94 e 0.97 rispettivamente), suggerendo una migliore capacità di localizzazione delle anomalie.

Su WFDD invece i risultati sono più bilanciati: DRÆM mantiene un leggero vantaggio nell'AUROC immagine (0.87 vs 0.84), mentre EfficientAD eccelle nelle metriche pixel, particolarmente nel Pixel-F1 (0.36 vs 0.08), indicando che, nonostante prestazioni simili a livello di immagine, EfficientAD localizza meglio le anomalie. La discrepanza nei risultati di WFDD potrebbe dipendere dalla natura specifica del dataset, dove DRÆM sembra adattarsi meglio alla classificazione globale, mentre EfficientAD preserva la sua forza nella segmentazione.

Questi risultati suggeriscono che la scelta del modello dovrebbe considerare sia il tipo di dataset che il trade-off tra rilevazione e localizzazione delle anomalie.

Table 14: Confronto velocità media di inferenza per dataset e modello

	Tempo/Immagine (s) ↓	Throughput (img/s) ↑	Tempo Totale (s) ↓	Immagini
MVTec				
DRÆM	0.18	5.58	308.30	1,721
EfficientAD	0.1522	6.57	261.95	1,721
ViSa				
DRÆM	0.1072	9.33	115.96	1,082
EfficientAD	0.0734	13.62	79.47	1,082
WFDD				
DRÆM	0.1338	7.47	24.63	184
EfficientAD	0.1222	8.18	22.49	184

Dal confronto emergono chiare tendenze prestazionali tra i modelli e i dataset. EfficientAD dimostra una superiorità sistematica, riducendo i tempi di elaborazione del 15-30% rispetto a DRÆM, con un miglior throughput particolarmente marcato su ViSa (+45%). La differenza prestazionale si attenua nel dataset WFDD (+9.5%), suggerendo che i vantaggi di EfficientAD siano più evidenti su dataset più complessi e diversificati.

I risultati confermano che EfficientAD rappresenta un progresso significativo, sebbene permangano margini di ottimizzazione per specifiche categorie problematiche. Questi dati guidano verso una selezione del modello basata sul trade-off tra precisione e velocità richiesto dall'applicazione specifica.

6 Appendice

6.1 Undersampling Train Set

Per valutare l'efficienza computazionale e la robustezza dei modelli in condizioni di dati limitati, abbiamo condotto test riducendo del 50% il training set tramite undersampling (rapporto 0.5). Questo approccio permette di analizzare la capacità dei modelli di mantenere buone prestazioni con meno dati di addestramento, e l'eventuale riduzione del tempo di training, particolarmente rilevante per applicazioni industriali dove la disponibilità di dati annotati è spesso limitata. I risultati seguenti dimostrano come i modelli si comportano in questo scenario ottimizzato, bilanciando efficienza e accuratezza.

Table 15: Confronto modelli per dataset

	Img-AUROC	Img-F1	Pixel-AUROC	Pixel-F1
MVTec				
DRÆM	0.69	0.86	0.69	0.21
DRÆM Under	0.75	0.86	0.69	0.20
EfficientAD	0.94	0.92	0.94	0.60
EfficientAD Under	0.90	0.91	0.93	0.55
ViSa				
DRÆM	0.74	0.77	0.78	0.11
DRÆM Under	0.73	0.75	0.78	0.11
EfficientAD	0.89	0.84	0.97	0.37
EfficientAD Under	0.86	0.83	0.95	0.33
WFDD				
DRÆM	0.87	0.81	0.59	0.08
DRÆM Under	0.86	0.78	0.60	0.14
EfficientAD	0.84	0.82	0.84	0.36
EfficientAD Under	0.85	0.80	0.85	0.36

Dall'analisi emergono due evidenze chiave: l'undersampling influisce marginalmente sulle performance (cali contenuti entro il 5% per EfficientAD), dimostrando la robustezza dei modelli anche con metà dati di training. Tuttavia, DRÆM mostra maggiore sensibilità alla riduzione del dataset, mentre EfficientAD mantiene una netta superiorità in quasi tutte le metriche, confermandosi più stabile ed efficiente anche in condizioni di dati limitati.

6.2 Nuovi parametri per il Train

Dopo aver identificato il modello EfficientAD come migliore performer ma con margini di miglioramento sul dataset WFDD (dove mostrava le performance relativamente più basse), è stato implementato un test intensivo con:

- **Aumento delle epoches (30→100)**: per sfruttare appieno la capacità di apprendimento del modello su un dataset complesso
- **Modifiche strategiche agli iperparametri**:
 - Learning rate 5x maggiore ($0.0001 \rightarrow 0.0005$) per accelerare la convergenza
 - Aumento del weight decay ($0.00001 \rightarrow 0.0001$) per contrastare l'overfitting in training prolungato
 - Ottimizzazione del padding (padding=True, pad_maps=False) per migliorare l'elaborazione delle feature spaziali

Queste modifiche mirano a bilanciare capacità di generalizzazione ed efficienza computazionale, testando l'ipotesi che un training più aggressivo possa compensare le criticità emerse sul dataset WFDD.

```
model = EfficientAd(
    imangenet_dir=os.path.join(imagenette2_path, 'imagenette2'),
    model_size=EfficientAdModelSize.S,
    lr=0.0001,
    teacher_out_channels=384,
    weight_decay=0.00001,
    padding=False,
    pad_maps=True,
    pre_processor=True,
    post_processor=True,
    evaluator=True,
    visualizer=True,
)
```

Codice EfficientAD

```
model = EfficientAd(
    imangenet_dir=os.path.join(imagenette2_path, 'imagenette2'),
    model_size=EfficientAdModelSize.S,
    lr=0.0005, # Aumentato
    teacher_out_channels=384,
    weight_decay=0.0001, # Aumentato
    padding=True,
    pad_maps=False, # Disabilitato
    pre_processor=True,
    post_processor=True,
    evaluator=True,
    visualizer=True
)
```

Codice EfficientAD nuovi parametri

Table 16: Confronto modelli per dataset WFDD

	Img-AUROC	Img-F1	Pixel-AUROC	Pixel-F1
WFDD				
EfficientAD	0.84	0.82	0.84	0.36
EfficientAD new	0.83	0.81	0.98	0.43

L'ottimizzazione iperparametrica ha prodotto un significativo miglioramento a livello di segmentazione (+19% Pixel-AUROC e +19% Pixel-F1), dimostrando che il tuning degli iperparametri è particolarmente efficace per le metriche pixel-level. Nonostante un lieve calo nelle metriche a livello di immagine (-1% Image-AUROC e -1% Image-F1), il nuovo setup mostra un netto vantaggio complessivo, specialmente nella localizzazione precisa delle anomalie, confermando l'importanza di un training più lungo e aggressivo per compiti di segmentazione fine.