

UNIVERSITÀ DELLA CALABRIA

CORSO DI MACHINE & DEEP LEARNING

Analisi e Ottimizzazione di Architetture per l'Industrial Anomaly Detection

*Il caso SuperSimpleNet: Efficienza, Robustezza e
Supervisione Debole*

Giuseppe Zappia

Anno Accademico:
2025-2026

Indice dei Contenuti

1 Paper Assegnato: Sintesi di Anomalie Industriali	1
1.1 Scelta e Posizionamento di SuperSimpleNet	1
2 Analisi Architetturale: Da SimpleNet a SuperSimpleNet	2
2.1 SimpleNet: L'approccio minimalista	2
2.2 SuperSimpleNet: Unificazione e Precisione	3
2.2.1 Upscaling e Feature Adaptation	3
2.2.2 Generazione Avanzata delle Anomalie (Perlin Noise) . . .	3
2.2.3 Architettura a Doppia Testa (Dual Head)	3
2.3 Confronto e Differenze Chiave	4
3 Metodologia delle Analisi	5
3.1 Scelta del Dataset: MVTec-AD	5
3.2 Vincoli Computazionali e Strategia di Campionamento	5
3.3 Focus Sperimentale e Workflow Tecnico	6
4 Panoramica delle Modifiche Architetturali	7
4.1 Ottimizzazione della Backbone	7
4.2 Generazione Adattiva del Rumore	7
4.3 Estensione Weakly-Supervised	7
5 Modifica Numero Uno	7
5.1 Sostituzione del Backbone: Da WideResNet50 a ResNet18 . . .	8
5.2 Modifiche al Feature Adaptor: Caso A e Caso B	8

5.3	Modifiche al codice	9
5.4	Protocollo di Valutazione e Metodologia di Presentazione dei Risultati	10
5.4.1	Metriche di Qualità e Accuratezza	10
5.4.2	Benchmark di Efficienza Computazionale	10
5.4.3	Analisi Qualitativa	11
5.5	Risultati Sperimentali e Discussione Metriche di Qualità e Accuratezza	11
5.5.1	Tabella Comparativa Generale	11
5.5.2	Analisi Dettagliata per Categoria	12
5.5.3	Analisi Approfondita Scenario A	12
5.5.4	Analisi Approfondita Scenario B	13
5.6	Valutazione Benchmark di Efficienza Computazionale	14
5.7	Analisi qualitativa	16
6	Modifica Numero Due	18
6.1	Generazione Adattiva del Rumore	18
6.2	Introduzione del Learned Noise Generator	18
6.3	Risultati Sperimentali in Seguito alla Modifica	19
6.3.1	Interpretazione Tecnica del Fallimento	19
6.4	Analisi delle Criticità ed Ulteriori Modifiche Architetturali	20
6.5	Ottimizzazione del Fattore di Scala (β)	21
6.5.1	Risultati della Model Selection	21
6.5.2	Discussione dei Risultati	22
6.6	Analisi comparativa dei Risultati	22
6.6.1	Miglioramenti della Versione Evoluta Rispetto alla Iniziale	23
6.6.2	Confronto Critico: Scenario A vs Versione Evoluta	24
6.7	Benchmark di Efficienza Computazionale	24
7	Modifica Numero Tre	25
7.1	Implementazione Weakly-Supervised con GAP+CAM	25
7.2	Dettagli della Modifica Architetturale	25
7.3	Formulazione Matematica del CAM	26
7.4	Training Objective	26
7.5	Scelta del Dataset e Vincoli Sperimentali	27
7.6	Performance Quantitativa	27
7.6.1	Analisi del Rilevamento (Detection)	28
7.6.2	Analisi della Localizzazione (Localization)	29
7.7	Analisi Qualitativa	31
7.8	Confronto con SimpleNet e SuperSimpleNet	32
7.8.1	Prestazioni di Rilevamento (Detection)	32
7.9	Comportamento su Campioni Normali	33
7.10	Proposte di Risoluzione per il Basso AP-loc	34

1 Paper Assegnato: Sintesi di Anomalie Industriali

Il seguente progetto prende come riferimento il paper "A Survey on Industrial Anomalies Synthesis" [4]. Questo parla del rilevamento di difetti in ambito industriale, che deve affrontare una sfida fondamentale: la scarsità di campioni anomali reali. Il paper evidenzia come l'acquisizione di immagini difettose richiede costi elevati, strumentazione specializzata e annotazione manuale da parte di esperti, rendendo difficile l'addestramento di modelli supervisionati tradizionali. Per mitigare questo problema, la ricerca si è spostata verso algoritmi di **Industrial Anomaly Synthesis (IAS)**, ovvero metodi capaci di generare campioni difettosi sintetici per addestrare i modelli. Il survey propone la prima tassonomia strutturata per questo campo, classificando le metodologie in quattro paradigmi principali:

- **Hand-crafted Synthesis:** Metodi che non richiedono addestramento (training-free) e si basano su regole manuali o texture esterne (es. *CutPaste*, *DRAEM*) per simulare difetti tramite operazioni di ritaglio e incolla o rumore procedurale (Perlin Noise).
- **Distribution Hypothesis-based Synthesis:** Approcci che modellano la distribuzione statistica dei dati normali e generano anomalie introducendo perturbazioni controllate nello spazio delle feature.
- **Generative Models (GM)-based Synthesis:** Utilizzo di modelli generativi profondi come GAN o Modelli di Diffusione per generare intere immagini difettose o tradurre immagini sane in difettose.
- **Vision-Language Models (VLM)-based Synthesis:** La frontiera più recente che sfrutta modelli multimodali e prompt testuali per guidare la generazione di anomalie.

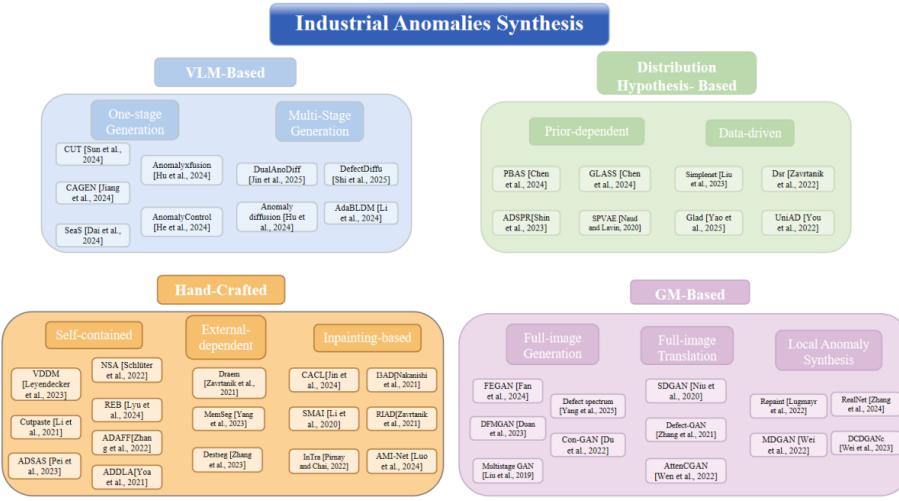


Figura 1: Tassonomia proposta dal paper di riferimento

1.1 Scelta e Posizionamento di SuperSimpleNet

Ai fini di questo progetto è stato preso in esame il Paper SuperSimpleNet [3], questo, viene classificato dal survey all'interno della categoria **Distribution Hypothesis-based**.

bution Hypothesis-based Synthesis, e più specificamente come metodo **Data-driven**.

A differenza degli approcci *Hand-crafted* (che operano sui pixel) o dei *GM-based* (che generano immagini intere), SuperSimpleNet opera direttamente nello spazio latente. Esso estrae le feature dei campioni normali tramite una rete backbone e sintetizza le anomalie aggiungendo rumore (Gaussiano o appreso) direttamente a queste rappresentazioni latenti. Come notato nel survey, SuperSimpleNet rappresenta un'evoluzione di *SimpleNet* [2] in quanto "confina il rumore in regioni specifiche e impiega una nuova testa di segmentazione", offrendo una strategia di sintesi più mirata e precisa.

La scelta di focalizzare questo progetto su SuperSimpleNet deriva proprio dalla sua natura ibrida e "semplice": sfruttando la flessibilità della perturbazione nello spazio delle feature, esso promette di unire l'efficienza dei metodi non supervisionati con la precisione di quelli supervisionati, rendendolo un candidato ideale per l'analisi e l'ottimizzazione in contesti industriali a risorse limitate.

2 Analisi Architetturale: Da SimpleNet a SuperSimpleNet

In questo capitolo vengono analizzati i principi di funzionamento delle due architetture di riferimento per questo studio: *SimpleNet* [2] e la sua evoluzione *SuperSimpleNet* [3]. Come anticipato, entrambi i modelli si collocano nella categoria della **Sintesi basata su Ipotesi di Distribuzione**, operando direttamente nello spazio latente delle feature estratte da una backbone pre-addestrata per generare anomalie sintetiche.

2.1 SimpleNet: L'approccio minimalista

SimpleNet [2] è stato proposto con l'obiettivo di semplificare il rilevamento di anomalie eliminando la complessità dei modelli basati su ricostruzione o su flow complessi. L'intuizione chiave è che proiettare le feature in un dominio target e aggiungere rumore sintetico nello spazio delle feature sia più efficace che generare difetti sintetici nello spazio dell'immagine. Il funzionamento, illustrato in Figura 2, si articola in quattro componenti principali:

1. **Feature Extractor (F_ϕ)**: Utilizza una WideResNet50 pre-addestrata su ImageNet per estrarre le feature locali dai blocchi intermedi (layer 2 e 3).
2. **Feature Adaptor (G_θ)**: Un modulo semplice (tipicamente un layer lineare o MLP) che proietta le feature estratte verso il dominio target, riducendo il bias del pre-addestramento su ImageNet.
3. **Anomalous Feature Generator**: Durante il training, vengono generate feature anomale aggiungendo Rumore Gaussiano i.i.d. ($\epsilon \sim \mathcal{N}(\mu, \sigma^2)$) alle feature normali adattate. Questo processo avviene interamente nello spazio latente.
4. **Discriminator (D_ψ)**: Un classificatore semplice (MLP a 2 layer) viene addestrato a distinguere le feature normali da quelle perturbate dal rumore.

Durante l'inferenza, il generatore di rumore viene scartato e il discriminatore fornisce direttamente uno score di anomalia per ogni posizione della feature map.

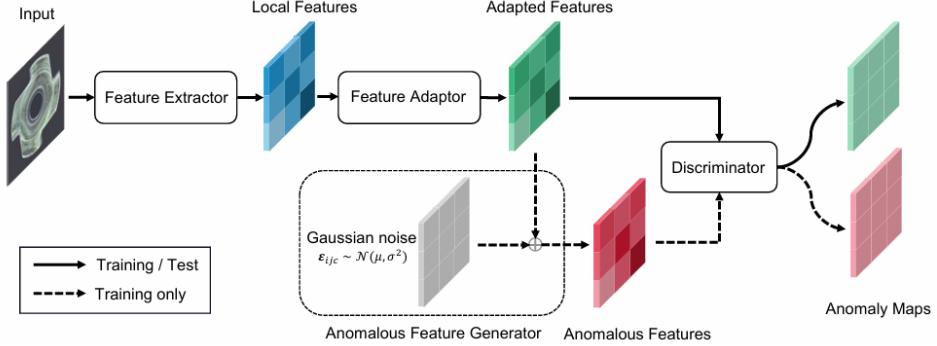


Figura 2: Architettura di SimpleNet: flusso di estrazione, adattamento e perturbazione delle feature.

2.2 SuperSimpleNet: Unificazione e Precisione

SuperSimpleNet [3] rappresenta l'evoluzione diretta di SimpleNet, progettata per soddisfare i requisiti industriali di velocità, consistenza e flessibilità nell'uso dei dati etichettati. Pur mantenendo la filosofia "feature-based", introduce tre modifiche strutturali critiche per migliorare la localizzazione e permettere la supervisione mista.

2.2.1 Upscaling e Feature Adaptation

A differenza di SimpleNet che lavora a risoluzioni diverse, SuperSimpleNet introduce un modulo di **Upscaling** aggressivo: le feature del layer 3 vengono ingrandite di 4x e quelle del layer 2 di 2x prima della concatenazione. Questo permette di preservare dettagli spaziali fini, fondamentali per rilevare difetti piccoli che andrebbero persi con il downsampling standard. Successivamente, un Feature Adaptor (implementato come convoluzione 1×1) adatta le feature allo spazio latente comune.

2.2.2 Generazione Avanzata delle Anomalie (Perlin Noise)

Mentre SimpleNet aggiunge rumore Gaussiano indiscriminatamente, SuperSimpleNet utilizza maschere di **Rumore di Perlin** binarizzate (M_a) per definire *dove* applicare la perturbazione.

$$P = (1 - M_a) \cdot A + M_a \cdot (A + \epsilon) \quad (1)$$

Questo approccio crea regioni di anomalia contigue e spazialmente coerenti, simulando difetti fisici più realistici rispetto al rumore statico diffuso. Inoltre, la perturbazione viene applicata sia alla copia delle feature che all'originale in base alla maschera, stabilizzando il training.

2.2.3 Architettura a Doppia Testa (Dual Head)

La novità più rilevante è la divisione del compito decisionale in due teste specializzate, come mostrato in Figura 3:

- **Segmentation Head (D_{seg}):** Riceve le feature perturbate P e predice una maschera di anomalia pixel-wise (M_o). È addestrata con una *Truncated L1 Loss* focalizzata sui bordi del difetto.
- **Classification Head (D_{cls}):** Riceve in input la concatenazione delle feature e della maschera predetta ($A \oplus M_o$). Utilizza blocchi convoluzionali e pooling globale per produrre uno score scalare (s) che valuta il contesto globale dell'immagine, riducendo i falsi positivi.

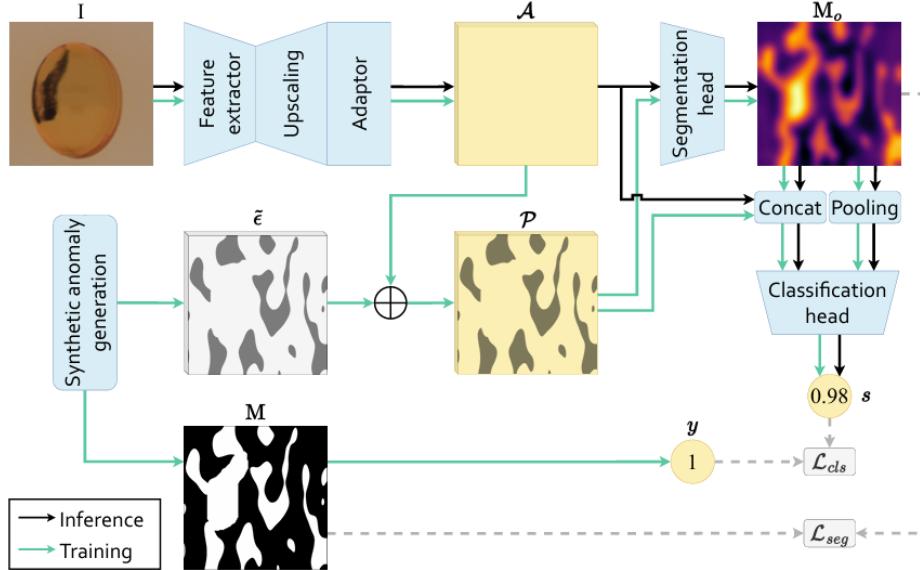


Figura 3: Architettura di SuperSimpleNet. Si noti l'inserimento della maschera predetta (M_o) come input per la testa di classificazione.

2.3 Confronto e Differenze Chiave

Le differenze tra i due modelli, riassunte nella Tabella 1, non sono solo architettoniche ma metodologiche.

1. Supervisione Unificata vs Non Supervisionata: SimpleNet è strettamente non supervisionato. SuperSimpleNet, invece, propone un framework unificato: se sono disponibili annotazioni reali (M_{gt}), il generatore applica il rumore sintetico solo nelle regioni sane ("Safe Zones"), permettendo al modello di imparare simultaneamente dai difetti reali e da quelli sintetici.

2. Coerenza del Rumore: L'uso del Rumore di Perlin in SuperSimpleNet risolve il problema delle anomalie "puntiformi" di SimpleNet, creando pattern che il discriminatore impara a riconoscere come difetti strutturali e non come semplice rumore statistico.

3. Precisione vs Velocità: Nonostante l'aggiunta di componenti (Upscaling, Dual Head), SuperSimpleNet mantiene un'inferenza estremamente rapida (9.3 ms su V100), paragonabile o superiore a SimpleNet grazie all'ottimizzazione delle operazioni di reshaping e all'uso efficiente delle convoluzioni 1×1 .

Caratteristica	SimpleNet [2]	SuperSimpleNet [3]
Sintesi Anomalia	Rumore Gaussiano su intera feature map	Maschere Perlin (Regioni contigue)
Risoluzione Feature	Bassa (Downsampling nativo)	Alta (Upscaling x2 e x4)
Architettura Decisionale	Discriminatore Singolo (MLP)	Dual Head (Segmentation + Classification)
Loss Function	Truncated L1	Truncated L1 (Seg) + Focal Loss (Cls)
Gestione Supervisione	Solo Unsupervised	Ibrida (Unsupervised + Supervised)

Tabella 1: Confronto strutturale tra SimpleNet e SuperSimpleNet.

3 Metodologia delle Analisi

In questa sezione vengono dettagliate le scelte strategiche e i vincoli operativi che hanno guidato la sperimentazione. L'obiettivo è isolare le variabili di interesse (Backbone, Adaptor, Generazione del Rumore) mantenendo fisso il contesto sperimentale, al fine di validare l'ipotesi di efficienza e robustezza delle modifiche proposte.

3.1 Scelta del Dataset: MVTec-AD

La validazione sperimentale è stata condotta sul dataset **MVTec AD** (MVTec Anomaly Detection) [1], riconosciuto in letteratura come il benchmark di riferimento per il rilevamento di anomalie in ambito industriale non supervisionato. La scelta di questo dataset è motivata dalla sua natura eterogenea: esso include 15 classi distinte suddivise in due macro-categorie fondamentali:

- **Oggetti:** Corpi rigidi con geometria definita e posizionamento fisso.
- **Texture:** Superfici con pattern ripetitivi o stocastici.

Questa dualità è essenziale per valutare trasversalmente le capacità di un modello: mentre gli oggetti richiedono la comprensione della semantica globale, le texture necessitano di una sensibilità ai dettagli locali ad alta frequenza.

3.2 Vincoli Computazionali e Strategia di Campionamento

Il protocollo standard per MVTec-AD adotta un approccio **One-Class-One-Model**, che richiede l'addestramento di una rete neurale dedicata per ciascuna delle 15 categorie. Considerando la necessità di testare diverse configurazioni sperimentali, l'esecuzione dell'intera suite su tutte le classi avrebbe comportato un carico computazionale eccessivo (stimato in oltre 45 sessioni di training complete).

Data la limitazione delle risorse hardware disponibili nell'ambiente Kaggle, è stata adottata una strategia di campionamento mirato. È stato selezionato un sottoinsieme rappresentativo di 3 classi:

1. **Bottle (Oggetto Semplice):** Rappresenta il caso "ideale". Si tratta di un oggetto rigido con difetti macroscopici e ben definiti. Questa classe è stata selezionata per valutare l'efficienza pura: verificare se il modello può mantenere alte prestazioni su compiti geometricamente semplici anche in presenza di ottimizzazioni o riduzioni di risorse, visto che nel paper originale si otteneva il 100% della detection.
2. **Carpet (Texture Complessa):** Rappresenta la sfida spaziale. Essendo una superficie con pattern intricati, richiede che il modello catturi

correttamente il contesto locale e le relazioni tra pixel adiacenti. È il banco di prova ideale per verificare se l'architettura riesce a preservare i dettagli fini durante l'estrazione delle feature. Nel paper originale si ottenevano risultati abbastanza buoni.

3. **Screw (Oggetto Strutturato):** Rappresenta lo *Stress Test* critico. Questa classe combina una geometria rigida con texture fini (la filettatura). Nel paper originale, gli autori identificano classi di questo tipo come le più soggette alla "Backbone Dependency", ovvero la tendenza del modello a fallire se l'estrattore di feature non è sufficientemente profondo o capace.

3.3 Focus Sperimentale e Workflow Tecnico

La sperimentazione si è focalizzata in una prima fase sul Setting Unsupervised in cui il modello viene addestrato esclusivamente su immagini prive di difetti ("Good"), alla fine invece si è passati al regime Supervised sfruttando il dataset **KSDD2**.

Dal punto di vista operativo, il workflow è stato strutturato per garantire la confrontabilità diretta dei risultati:

- **Ambiente di Esecuzione e Workflow:** Le sessioni di training e inferenza sono state eseguite sulla piattaforma Kaggle sfruttando come acceleratore la NVIDIA Tesla P100, necessaria per sostenere il carico computazionale del progetto. Tuttavia, per superare le limitazioni dell'editor web di Kaggle nella gestione di modifiche strutturali complesse, è stato adottato un flusso di lavoro ibrido. Partendo da un fork del repository ufficiale [3], le modifiche alle classi sono state sviluppate e integrate in locale. Successivamente, il codice modificato è stato pushato sul repository remoto e clonato nell'ambiente runtime di Kaggle. Questa strategia ha permesso di coniugare la flessibilità dello sviluppo locale con la potenza di calcolo del cloud, garantendo al contempo il controllo di versione e la riproducibilità degli esperimenti.

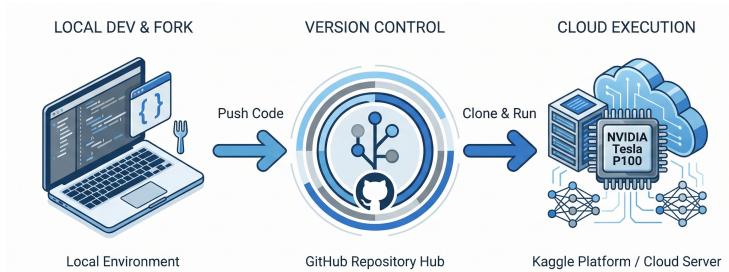


Figura 4: Workflow

- **Principio Ceteris Paribus:** Per neutralizzare le differenze hardware rispetto al paper originale (che utilizzava Tesla V100), è stata riprodotta localmente anche la Baseline (il modello originale senza modifiche) sullo stesso hardware P100. In questo modo, ogni variazione di performance o efficienza misurata nelle fasi successive potrà essere attribuita esclusivamente alle modifiche algoritmiche e non alla discrepanza tecnologica.

4 Panoramica delle Modifiche Architetturali

Sulla base dell’analisi della letteratura e dei vincoli operativi discussi, il presente progetto propone tre interventi sostanziali sull’architettura originale di SuperSimpleNet. Tali modifiche sono state concepite per investigare e ottimizzare tre dimensioni critiche per l’applicazione industriale: l’**efficienza computazionale**, la **robustezza** su texture complesse e la **flessibilità** rispetto ai costi di annotazione.

Di seguito viene fornita una breve sintesi delle tre modifiche implementate, che saranno analizzate in dettaglio nelle sezioni successive:

4.1 Ottimizzazione della Backbone

La prima modifica mira a ridurre drasticamente il carico computazionale durante l’inferenza. L’estrattore di feature originale (*WideResNet50*) è stato sostituito con una *ResNet18*, molto più leggera. Contestualmente, è stato condotto uno studio sul modulo *Feature Adaptor*, confrontando l’approccio lineare originale con una variante non-lineare spaziale (3×3), al fine di verificare se una maggiore complessità nell’adattamento potesse compensare la ridotta capacità della backbone più piccola.

4.2 Generazione Adattiva del Rumore

Per mitigare la forte dipendenza dalla backbone (“Backbone Dependency”) e migliorare la stabilità su texture complesse (come evidenziato dalle criticità emerse in seguito alla prima modifica), è stato sostituito il generatore di rumore Gaussiano statico con un **Learned Noise Generator**. Questa modifica introduce una dinamica di addestramento avversario (simile alle GAN), in cui un generatore neurale apprende a produrre le perturbazioni più efficaci per ingannare il discriminatore, adattandosi dinamicamente alla distribuzione delle feature locali invece di usare un parametro σ fisso.

4.3 Estensione Weakly-Supervised

Infine, per rispondere all’esigenza di ridurre i costi di annotazione manuale, è stata sviluppata una variante dell’architettura in grado di operare in regime *Weakly-Supervised*. Rimuovendo la testa di segmentazione (D_{seg}) e introducendo meccanismi di **Global Average Pooling (GAP)** e **Class Activation Mapping (CAM)**, questa modifica abilita il modello a localizzare i difetti utilizzando esclusivamente etichette a livello di immagine (Difettoso/Buono), eliminando la necessità di maschere di segmentazione pixel-wise durante il training.

5 Modifica Numero Uno

La prima modifica che si è deciso di apportare si focalizza su due componenti chiave del modello SuperSimpleNet: l’estrattore di feature (backbone) e il modulo di adattamento (feature adaptor). L’obiettivo è indagare la robustezza della metodologia proposta dagli autori rispetto alla variazione della capacità espressiva dei moduli interni, rispondendo ai requisiti di efficienza e flessibilità citati nel paper.

5.1 Sostituzione del Backbone: Da WideResNet50 a ResNet18

Nel lavoro originale, gli autori adottano una WideResNet50 come estrattore di feature, selezionando i layer 2 e 3. Sebbene questa scelta garantisca un'elevata ricchezza semantica, gli autori stessi ammettono che il modello è "pesantemente dipendente dal backbone utilizzato" e che le prestazioni decadono se l'estrattore non fornisce feature significative. La scelta di passare a una ResNet18 è motivata da tre interrogativi critici sollevati nel paper:

1. Efficienza Industriale: Uno dei pilastri di SuperSimpleNet è la velocità di inferenza (target < 10ms). L'utilizzo di un backbone più leggero permette di testare il limite estremo di throughput del sistema in contesti dove le risorse computazionali sono ancora più limitate di quelle standard.
2. Generalizzazione della Logica Discriminativa: Se SuperSimpleNet è effettivamente un'evoluzione superiore di SimpleNet, la sua capacità di generare anomalie sintetiche nello spazio latente dovrebbe rimanere efficace anche con un set di feature meno denso, come quello prodotto da una ResNet18.
3. Analisi della "Backbone Dependency": Verificare se i miglioramenti introdotti (upscaleing e classification head) sono in grado di compensare la riduzione dei parametri del backbone, mantenendo risultati di rilevazione competitivi

5.2 Modifiche al Feature Adaptor: Caso A e Caso B

Il modulo Feature Adaptor ha il compito di proiettare le feature estratte in uno spazio latente comune *A*. Nel paper originale, questo è implementato come un semplice strato lineare, ottimizzato tramite una convoluzione 1×1 per massimizzare la velocità. Per valutare criticamente l'impatto di questo modulo, si è seguito il *principio del Ceteris Paribus*, cioè: per misurare scientificamente l'impatto di un cambiamento, bisogna variare una sola cosa alla volta. Sono stati dunque definiti due scenari:

- Scenario A: Adaptor Lineare Minimalista (Baseline di Progetto). In questa configurazione, si mantiene la linearità originale utilizzando una convoluzione 1×1 . Questa scelta funge da "controllo" per isolare l'effetto del cambio di backbone. La logica risiede nel testare se una proiezione puramente affine sia sufficiente a mappare le feature di una ResNet18 in uno spazio adatto alla discriminazione tra normale e anomalo, seguendo la filosofia di "semplicità" che dà il nome al paper.
- Scenario B: Adaptor Non-Lineare e Spaziale. È stato progettato per affrontare il problema della ridotta capacità rappresentativa della ResNet18 rispetto alla WideResNet50 originale. Come evidenziato dagli autori di SuperSimpleNet, le prestazioni del modello dipendono strettamente dalla capacità dell'estrattore di produrre feature discriminative di alta qualità. È noto in letteratura che reti più "leggere" (come la ResNet18) tendono a produrre mappe di caratteristiche meno astratte e linearmente separabili rispetto a reti più profonde e larghe. Di conseguenza, l'ipotesi alla base dello Scenario B è che il semplice Feature Adaptor lineare proposto nell'architettura originale costituisca un "collo di bottiglia" informativo quando alimentato con feature a minor dimensionalità. In questo

scenario, l'architettura è stata modificata introducendo un Adattatore di Feature Non-Lineare. A differenza della semplice proiezione lineare 1×1 utilizzata nel paper originale per mappare le feature nello spazio latente, lo Scenario B implementa un adattatore non-lineare che sfrutta il contesto spaziale per compensare la ridotta capacità rappresentativa della ResNet18. L'architettura del modulo, come implementata nel file `supersimplenet.py`, è composta da una sequenza a tre stadi:

1. **Convoluzione 3x3 (Spaziale):** Invece di una proiezione punto-punto, viene utilizzato un kernel 3×3 con padding=1 per mantenere le dimensioni spaziali. Questa scelta mira ad aggregare informazioni dai pixel vicini, cercando di ricostruire pattern locali che una backbone più leggera potrebbe non aver estratto con sufficiente astrazione.
2. **Batch Normalization:** Introdotta per stabilizzare la distribuzione delle feature estratte prima dell'iniezione del rumore sintetico.
3. **ReLU Activation:** Introduce una non-linearità per permettere al modello di apprendere confini decisionali più complessi nello spazio delle feature.

L'obiettivo sperimentale è verificare se questo aumento di complessità nell'adattatore sia sufficiente a recuperare l'accuratezza persa con il cambio di backbone, mantenendo comunque i requisiti di efficienza per l'Edge Computing. Specificamente, si intende valutare se la normalizzazione introdotta dalla Batch Normalization possa interagire positivamente con il modulo di generazione delle anomalie sintetiche, o se, al contrario, l'alterazione della statistica delle feature richieda una ricalibrazione del parametro di rumore gaussiano σ (fissato a 0.015 nel paper originale).

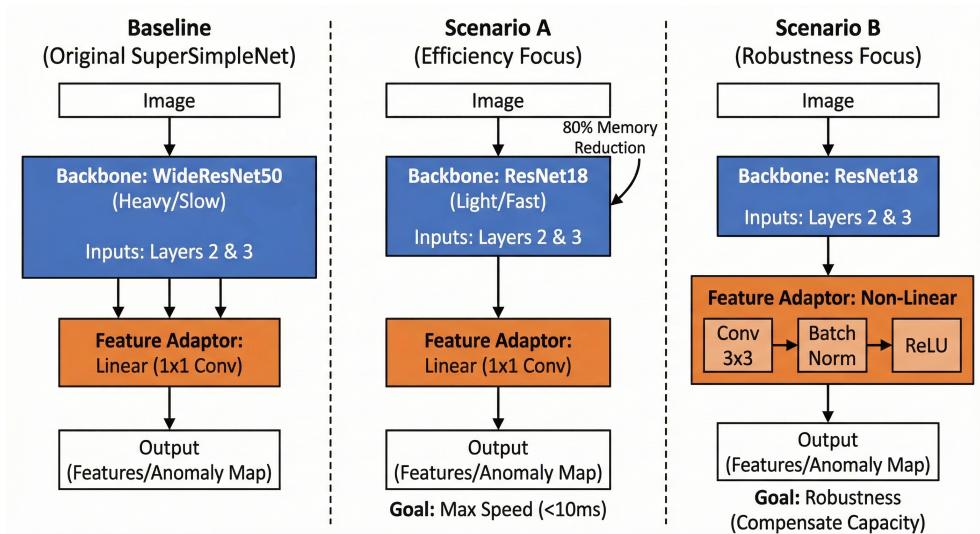


Figura 5: Modifiche e obiettivi

5.3 Modifiche al codice

Ho modificato 3 classi in questa fase che sono:

- train.py
- supersimplenet.py
- perf_main.py

Le modifiche effettuate aggiungono le componenti descritte sopra e consentono di avviare il train e la valutazione delle performance selezionando esplicitamente la backbone da utilizzare e lo scenario specifico.

5.4 Protocollo di Valutazione e Metodologia di Presentazione dei Risultati

Al fine di garantire la confrontabilità diretta con i risultati pubblicati in Super-SimpleNet, la presentazione dei dati sperimentali segue rigorosamente le metriche e i protocolli definiti nella Sezione 4 del paper originale, adattati alle specificità dell'hardware utilizzato in questo studio (NVIDIA Tesla P100 tramite Kaggle). L'analisi si articola su tre livelli: metriche quantitative di rilevamento, benchmark di efficienza computazionale e analisi qualitativa visiva.

5.4.1 Metriche di Qualità e Accuratezza

La valutazione della capacità del modello di distinguere campioni sani da quelli difettosi si basa sulle metriche standard per il dataset MVTec AD, come delineato da Rolih et al.:

- I-AUROC (Image-level Area Under the Receiver Operating Characteristic): Metrica primaria per la classificazione. Valuta l'accuratezza globale nel determinare se un'immagine contiene un'anomalia.
- AUPRO (Area Under the Per-Region Overlap): Metrica primaria per la localizzazione. A differenza del semplice AUROC a livello di pixel (spesso inflazionato dall'ampia presenza di sfondo), l'AUPRO pesa equamente le regioni di anomalia di diverse dimensioni, offrendo una stima più fedele della precisione di segmentazione.

In conformità con i dettagli implementativi del paper originale, i valori riportati non sono selezionati dal "miglior checkpoint" (che potrebbe essere frutto di un caso fortuito), ma corrispondono alle metriche calcolate sul modello risultante dall'epoca finale (Epoch 300). Questo approccio permette di valutare la stabilità della convergenza del modello e la sua robustezza finale, specialmente critica negli Scenari A e B dove la modifica della backbone potrebbe introdurre instabilità nel training.

5.4.2 Benchmark di Efficienza Computazionale

Per valutare l'impatto delle modifiche architettoniche (Sostituzione Backbone e Adattatore Non-Lineare) sulle risorse di sistema, è stato utilizzato lo script di benchmarking dedicato, perf_main.py, fornito dagli autori. Le metriche considerate sono riportate nel dettaglio nel Materiale Supplementare A del paper:

- Inference Time (ms): Tempo medio richiesto per processare una singola immagine (batch size = 1).

- Throughput (img/s): Numero massimo di immagini elaborabili al secondo.
- GPU Memory (MB): Picco di memoria VRAM allocata durante l'infidenza.

Nota sull'Hardware: Come ampiamente ribadito, il paper originale utilizza una NVIDIA Tesla V100S, gli esperimenti di questo studio sono stati condotti su hardware NVIDIA Tesla P100 (ambiente Kaggle). Pertanto, i valori assoluti di latenza non sono direttamente confrontabili con la Tabella 1 del paper. Tuttavia, il training è stato effettuato anche per la configurazione utilizzata nel paper, in questo modo entrambe le backbone possono essere confrontate. Come da protocollo originale, i dati di efficienza riportati sono la media di 5 cicli di esecuzione consecutivi per minimizzare la varianza dovuta al warmup della GPU.

5.4.3 Analisi Qualitativa

Per corroborare i dati numerici, viene presentata un'analisi visiva comparativa. Le immagini sono estratte dalla cartella visual generata durante la fase di test e seguono il layout standard del paper:

1. Immagine Originale (Input).
2. Ground Truth: Maschera binaria che indica la posizione reale del difetto.
3. Anomaly Map (Heatmap): La previsione del modello, dove le aree calde indicano alta probabilità di anomalia, con annessi Score ed SScore che rappresentano rispettivamente il valore massimo rilevato nella mappa dei pixel ed il valore prodotto dalla Classification Head. A differenza dello "Score" semplice, l'SScore analizza il contesto globale dell'immagine.

5.5 Risultati Sperimentali e Discussione Metriche di Qualità e Accuratezza

In questa sezione vengono presentati i risultati quantitativi ottenuti sul dataset MVTec AD, confrontando la Baseline con i due scenari proposti basati su ResNet18. L'analisi, come ampiamente discusso, si concentra sulle metriche di Detection (I-AUROC) e Localization (AUPRO), seguendo il protocollo di valutazione descritto nella Sezione 4.2 del paper originale.

5.5.1 Tabella Comparativa Generale

La Tabella seguente riassume le prestazioni di rilevamento e localizzazione per le tre categorie esaminate. I dati della colonna "Paper Ufficiale" sono tratti dalla Tabella 3 del paper originale, mentre le altre colonne riportano i risultati sperimentali ottenuti in questo studio.

Categoria	Metrica	Paper Ufficiale (V100)	Baseline Riprodotta (P100)	Scenario A (ResNet18)	Scenario B (ResNet18 + Non-Lin)
BOTTLE	I-AUROC (Det)	100.0%	100.0%	100.0%	71.3%
	AUPRO (Loc)	90.4%	90.2%	86.8%	29.7%
CARPET	I-AUROC (Det)	98.4%	98.5%	89.6%	75.9%
	AUPRO (Loc)	92.3%	92.5%	86.3%	21.4%
SCREW	I-AUROC (Det)	92.9%	91.2%	51.1%	46.6%
	AUPRO (Loc)	95.3%	94.3%	71.0%	20.2%

Tabella 2: Confronto delle prestazioni di training su MVTec AD

5.5.2 Analisi Dettagliata per CATEGORIA

Prima di valutare le modifiche, è fondamentale notare che la riproduzione della Baseline (Colonna "Baseline Riprodotta") è statisticamente coerente con i risultati dichiarati dagli autori. Su Bottle e Carpet, i risultati sono quasi identici (es. Carpet: 98.5% riprodotto vs 98.4% paper). Su Screw si ottiene un 91.2% contro il 92.9% del paper. La leggera discrepanza è attribuibile alla varianza stocastica del training su hardware differente, ma conferma la correttezza dell'implementazione del codice originale su Kaggle. Sulla categoria Bottle si nota il successo dell'architettura leggera, infatti per oggetti rigidi con difetti macroscopici come le bottiglie, lo Scenario A si è rivelato un eccellente sostituto della WideResNet50. Mantiene un I-AUROC del 100%, identico alla Baseline. La localizzazione (AUPRO) scende leggermente (86.8% vs 90.2%), ma rimane ad un livello operativo accettabile. L'implicazione è che per linee di produzione di oggetti semplici, è possibile ridurre le risorse computazionali (memoria e latenza) passando a ResNet18 senza alcuna perdita nella capacità di rilevare i pezzi difettosi. Tuttavia, lo Scenario B mostra un crollo inaspettato (71.3%), indicando che l'introduzione di non-linearietà nell'adattatore ha destabilizzato l'apprendimento su feature spazialmente semplici. Rispetto la categoria CARPET, visto che il tappeto rappresenta una superficie con texture complessa si nota che lo Scenario A subisce un calo di circa 9 punti percentuali (89.6%) in confronto alla Baseline. Sebbene il modello riesca ancora a distinguere la maggior parte dei difetti, fatica maggiormente a separare il rumore di fondo della trama del tappeto dalle anomalie reali rispetto alla WideResNet50. Questo risultato suggerisce che la profondità della rete (50 layer vs 18 layer) gioca un ruolo rilevante nell'estrazione di feature robuste per texture fini. Infine la categoria Screw, che rappresenta il caso di fallimento critico che conferma i limiti teorici discussi nel paper originale. Scenario A (51.1% I-AUROC): Il modello si comporta come un classificatore casuale (random guess). Scenario B (46.6% I-AUROC): Il modello performa peggio del caso, suggerendo un apprendimento errato o un overfitting sul rumore.

5.5.3 Analisi Approfondita Scenario A

La profonda discrepanza prestazionale osservata tra la categoria Bottle (I-AUROC 100%) e la categoria Screw (I-AUROC 51.1%) nello Scenario A non è casuale, ma evidenzia un limite strutturale nell'uso di backbone leggere come la ResNet18 su feature complesse. La categoria Bottle presenta difetti macroscopici e strutturali (es. "broken large", "contamination") su superfici visivamente omogenee. In questo contesto, le feature estratte dalla ResNet18,

seppur meno profonde, conservano sufficiente separabilità spaziale per distinguere un pezzo rotto da uno integro. Al contrario, la categoria Screw (Vite) è dominata da texture ad alta frequenza (la filettatura) e richiede la rilevazione di difetti sottili (es. "scratch neck") che si confondono facilmente con la geometria dell'oggetto. La WideResNet50 usata nel paper originale, grazie alla sua maggiore larghezza e profondità, genera feature maps ad alta risoluzione semantica in grado di codificare queste sottili variazioni di texture. La ResNet18, comprimendo maggiormente l'informazione, produce feature maps dove il segnale della "filettatura" viene perso o confuso con il segnale del "graffio". Inoltre il fattore determinante per il fallimento su Screw è esplicitamente predetto dagli autori nella sezione Limitations del paper: "SuperSimpleNet is also heavily dependent on the used backbone, as the magnitude of the Gaussian noise needs to be adjusted for each backbone". Il meccanismo di SuperSimpleNet si basa sull'aggiunta di rumore Gaussiano con deviazione standard fissa $\sigma=0.015$ nello spazio latente.

- Per la WideResNet50, questo σ rappresenta la "distanza" ottimale per spingere i campioni sani oltre il confine di normalità, simulando un difetto.
- Per la ResNet18 su Screw, le feature estratte sono probabilmente più addensate nello spazio vettoriale. Di conseguenza, un rumore fisso di 0.015 risulta eccessivo, "inondando" le feature originali e rendendo indistinguibili i campioni sani perturbati (anomali sintetiche) dai campioni sani originali. Il discriminatore, non riuscendo a trovare un confine netto, collassa su un comportamento casuale (50%), come osservato nei risultati.

5.5.4 Analisi Approfondita Scenario B

L'introduzione dello Scenario B era volta a verificare se un aumento della capacità espressiva del *Feature Adaptor* potesse compensare l'utilizzo di una *backbone* più leggera come la ResNet18. Tuttavia, i dati sperimentali indicano che l'architettura basata sulla sequenza **Conv 3x3 + Batch Normalization + ReLU** ha prodotto un drastico calo delle prestazioni rispetto alla configurazione lineare (Scenario A). Il crollo dell'I-AUROC (es. 46.6% su *Screw*) e della precisione di localizzazione (AUPRO 20.2%) è riconducibile a una sinergia distruttiva tra l'aggregazione spaziale e la normalizzazione statistica:

- **Smoothing Spaziale e Perdita di Dettaglio:** A differenza della convoluzione 1×1 , l'utilizzo di un kernel 3×3 nel modulo di adattamento introduce un campo ricettivo che aggrega informazioni dai pixel adiacenti. In categorie caratterizzate da texture ad alta frequenza, come la filettatura delle viti (*Screw*), questa operazione agisce come un filtro passa-basso che "sfoca" le transizioni nette dei bordi. Di conseguenza, le feature adattate perdono la risoluzione semantica necessaria per distinguere un difetto sottile (es. un graffio) dalla struttura regolare dell'oggetto.
- **Amplificazione del Rumore tramite BN:** La BN normalizza le feature forzandole ad avere media zero e varianza unitaria. Su un oggetto complesso come una vite, le feature estratte dalla ResNet18 contengono già un'alta varianza naturale dovuta all'alternanza visiva "filettatura-spazio". Normalizzando queste feature, la BN amplifica i dettagli della

texture portandoli alla stessa scala statistica delle anomalie sintetiche. Durante il training, quindi, il modello impara a riconoscere l'anomalia come una perturbazione rumorosa $P = A + (M_a * \epsilon)$ dove A sono le feature adattate, M_a è la maschera di Perlin ed ϵ il rumore Gaussiano $N(0, \sigma^2)$. Tuttavia, a causa della normalizzazione, la texture naturale della vite (es. i bordi affilati della filettatura) assume una distribuzione statistica troppo simile al rumore gaussiano sintetico (ϵ). In Inferenza, di conseguenza, il discriminatore non riesce più a distinguere tra il "rumore sintetico" (difetto) e la "texture ad alta frequenza" (normale). Il modello attiva la heatmap su intere sezioni della vite, interpretando la rugosità standard del metallo come un difetto diffuso. Questo genera un tasso elevatissimo di Falsi Positivi, che spiega perché l'I-AUROC scende al livello della casualità (46%): il modello segnala "anomalia" quasi ovunque, rendendo impossibile distinguere efficacemente i pezzi sani da quelli difettosi.

Questo esperimento dimostra che tentare di compensare una backbone debole aggiungendo complessità all'adattatore è controproducente. La normalizzazione altera la semantica delle feature necessaria a SuperSimpleNet per calibrare correttamente il rumore, trasformando il modello in un rilevatore ipersensibile che allucina difetti sulla struttura regolare dell'oggetto.

5.6 Valutazione Benchmark di Efficienza Computazionale

Un obiettivo cardine di questo studio era valutare se la sostituzione della backbone potesse rendere l'architettura SuperSimpleNet idonea per dispositivi Edge con risorse limitate, superando i requisiti di memoria della configurazione originale. I test sono stati eseguiti utilizzando lo script di benchmark `perf_main.py` su hardware NVIDIA Tesla P100. I risultati, calcolati sulla media di 5 cicli di esecuzione per garantire la stabilità termica, sono riportati nella Tabella 3.

Configurazione	Parametri Totali	Inference Time (ms)	Throughput (img/s)	GPU Memory (MB)
Baseline (WideResNet50)	33.7 M	25.58 ms	96.72	270.53 MB
Scenario A (ResNet18)	4.5 M	7.78 ms	398.47	52.43 MB
Scenario B (ResNet18 + BN)	5.7 M	8.36 ms	352.69	85.98 MB

Tabella 3: Benchmark delle prestazioni computazionali Tesla P100

I risultati evidenziano chiaramente:

1. Riduzione della Memoria (-80%): Il risultato più significativo emerge dal consumo di VRAM. La Baseline originale richiede 270.53 MB per l'inferenza, una quantità che può saturare rapidamente la memoria di dispositivi embedded. Lo Scenario A abbatte questo requisito a soli 52.43 MB, ottenendo una riduzione dell'80.6%. Anche lo Scenario B, nonostante l'aggiunta dei layer di Batch Normalization e proiezione, mantiene un consumo contenuto (85.98 MB), dimostrando che l'architettura proposta rientra ampiamente nei limiti dei dispositivi low-power.

2. Incremento del Throughput (4x): A parità di hardware, il passaggio da WideResNet50 a ResNet18 ha quadruplicato la capacità di elaborazione del sistema.
 - La Baseline elabora circa 96 immagini al secondo.
 - Lo Scenario A raggiunge quasi 400 immagini al secondo. Questo salto prestazionale riduce il tempo di inferenza per singola immagine da 25.58 ms a 7.78 ms, superando, anche solo con una GPU datata come la P100 (con cui la baseline impiega quasi 3 volte il tempo di inferenza per immagine del paper), la soglia critica dei 10ms spesso richiesta per l’ispezione in tempo reale su linee di produzione ad alta velocità
3. Costo dell’Adattatore Non-Lineare (Scenario B): Il confronto tra lo Scenario A e lo Scenario B permette di quantificare il "costo" della complessità architettonale aggiunta. L’integrazione del blocco $Conv\ 3 \times 3 \rightarrow BN \rightarrow ReLU$ incrementa il numero di parametri addestrabili di circa 1,2 milioni, portando il totale da 4,5M a 5,7M. Questo comporta un overhead computazionale minimo in termini di tempo (+0,58 ms di latenza), ma un aumento significativo dell’occupazione di memoria GPU (+64% rispetto allo Scenario A). Tale incremento è legato alla necessità di memorizzare le mappe di attivazione spaziali intermedie e gli stati della Batch Normalization durante le fasi di calcolo. Poiché lo Scenario B non ha portato benefici in accuratezza, ma ha anzi degradato sensibilmente le prestazioni a causa dell’aggregazione spaziale incoerente (I-AUROC al 46% su *screw*), questo costo aggiuntivo di risorse non risulta industrialmente giustificato.

In conclusione, va considerato che per categorie strutturali come Bottle, lo Scenario A rappresenta la configurazione ideale, riduce infatti il consumo di memoria dell’80% (52 MB vs 270 MB) e quadruplicando il throughput (400 img/s), mantiene un I-AUROC del 100%. In questo contesto, la sostituzione della backbone soddisfa pienamente i requisiti industriali di bassa latenza e alta precisione. Tuttavia, il caso Screw dimostra che l’efficienza non può prescindere dalla capacità estrattiva. Nonostante lo speed-up, il crollo dell’accuratezza al 51.1% nello Scenario A conferma la limitazione nota del modello. Un modello veloce ma impreciso è industrialmente inutilizzabile. Il tentativo di compensare con un adattatore più complesso (Scenario B) si è rivelato controproducente. L’aumento del 64% in memoria rispetto allo Scenario A non ha portato benefici, anzi ha peggiorato le prestazioni (I-AUROC 46%). Dunque, l’utilizzo di una backbone leggera in SuperSimpleNet è una strategia vincente per l’ispezione ad alta velocità di oggetti rigidi e semplici. Tuttavia, per superfici con texture fini, il risparmio di risorse compromette la robustezza, rendendo necessario il mantenimento della backbone originale o una ricalibrazione specifica degli iperparametri di generazione del rumore.

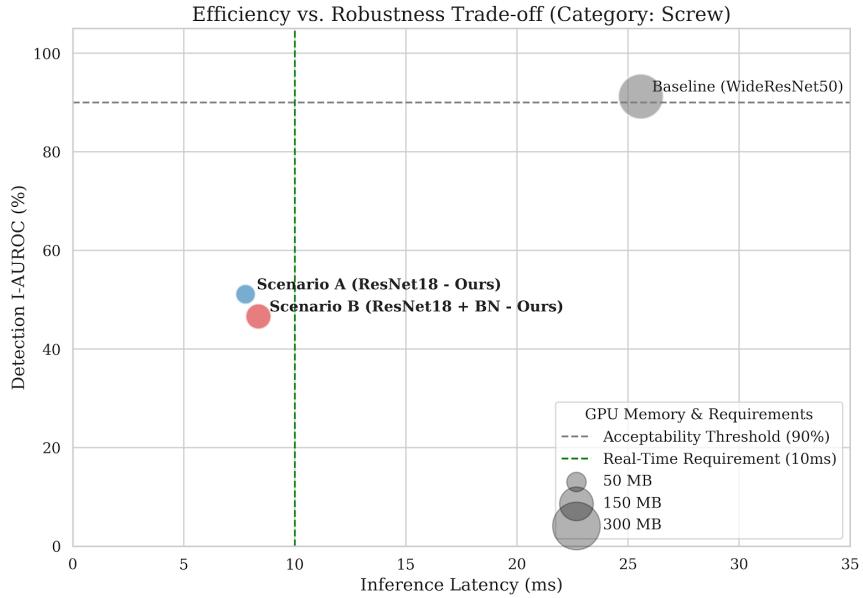


Figura 6: Analisi del Trade-off Efficienza-Robustezza per la categoria critica 'Screw'. Latenza (Asse X), Accuratezza di Rilevamento (Asse Y) e Memoria GPU (Dimensione della bolla). Si osserva come lo Scenario A ottenga un vantaggio drastico in velocità e leggerezza, posizionandosi ampiamente entro il requisito Real-Time. Tuttavia, il posizionamento verticale mostra un crollo delle prestazioni ben al di sotto della soglia di accettabilità industriale.

5.7 Analisi qualitativa

Di seguito vengono riportati alcuni esempi visivi di come la rete ha agito sugli esempi delle varie categorie.

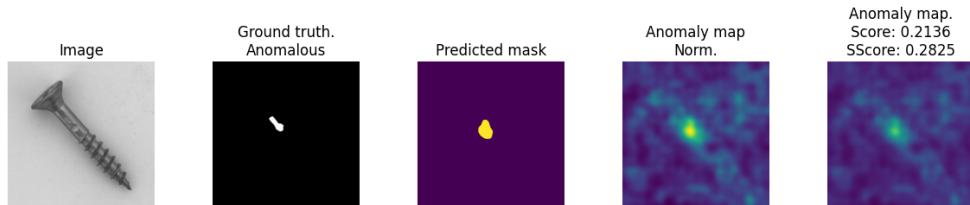


Figura 7: Architettura originale su esempio della categoria Screw

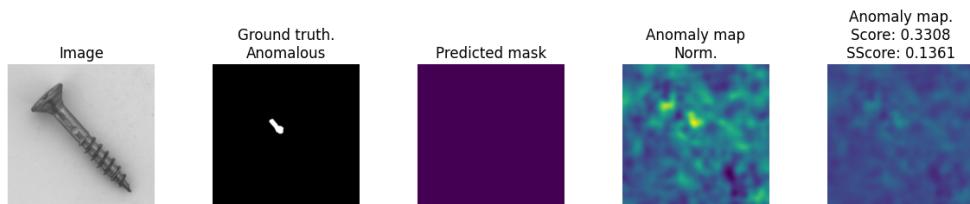


Figura 8: Architettura Resnet18 Scenario A su esempio della categoria Screw



Figura 9: Architettura Resnet18 Scenario B su esempio della categoria Screw

Come si può vedere da queste 3 immagini, la rete originale riconosce il difetto a livello di maschera, la rete dello Scenario A addirittura non riesce a stimare nemmeno un pixel come errore, mentre quella B divaga predicendo quasi interamente i pixel come anomali.



Figura 10: Architettura originale del paper su esempio della categoria Bottle



Figura 11: Architettura Resnet18 Scenario A su esempio della categoria Bottle



Figura 12: Architettura Resnet18 Scenario B su esempio della categoria Bottle

In questo caso invece come specificato nell'analisi numerica precedente, lo Scenario A performa molto bene sugli oggetti della categoria Bottle, mentre rispetto lo scenario B continuano ad esserci netti problemi con l'identificazione della maschera.

6 Modifica Numero Due

6.1 Generazione Adattiva del Rumore

I risultati ottenuti nello Scenario A hanno evidenziato una criticità fondamentale nell’architettura originale di SuperSimpleNet. Sebbene la sostituzione di WideResNet50 con ResNet18 abbia garantito i requisiti di bassa latenza (Inferenza < 10ms), il drastico calo delle prestazioni sulla categoria Screw suggerisce che il collo di bottiglia non risieda esclusivamente nella capacità estrattiva della rete, ma anche nella strategia di generazione delle anomalie. In effetti, il paper originale riconosce esplicitamente questa limitazione nelle conclusioni. Attualmente, il modello applica un rumore Gaussiano statico con deviazione standard fissa $\sigma = 0.015$ alle feature adattate (A) per creare le feature perturbate (P). Questo approccio scalare, ereditato da SimpleNet, assume implicitamente che lo spazio delle feature abbia una densità uniforme e che una perturbazione di intensità fissa sia sufficiente a simulare difetti su qualsiasi tipo di texture. Tuttavia, come osservato nell’esperimento precedente, le feature estratte da una ResNet18 su texture complesse (come la filettatura di una vite) possiedono una varianza e una distribuzione diverse rispetto a quelle della WideResNet50. Un rumore fisso risulta “cieco” al contesto locale: su una texture ad alta frequenza come quella della vite, una perturbazione di $\sigma = 0.015$ potrebbe risultare statisticamente irrilevante o, al contrario, non allineata con la distribuzione dei difetti reali, rendendo il confine decisionale appreso dal discriminatore inefficiente.

6.2 Introduzione del Learned Noise Generator

Per risolvere il problema c’è bisogno di passare da un approccio con rumore fisso basato su assunzioni statistiche ad uno dove la distribuzione del rumore viene appresa dinamicamente dalla rete stessa. A livello di addestramento, è stato quindi implementato un paradigma di tipo Generative Adversarial Network. Il codice vede l’introduzione di una nuova classe nel file supersimplenet.py per sostituire il campionamento stocastico con un modulo neurale con le seguenti caratteristiche:

1. Architettura Leggera: il generatore è costituito da una serie di layer convoluzionali 1×1 . Questa scelta permette di elaborare le feature map mantenendo le informazioni spaziali, trasformando il vettore di feature di ogni pixel in una perturbazione specifica.
2. Input Contestuale: il generatore riceve in input le feature originali (A). Questo permette al modulo di “vedere” la texture locale (es. la curvatura della vite) e decidere come modularne la perturbazione in modo coerente.
3. Training Avversariale: modificato la funzione di loss introducendo una dinamica simile alle GAN.
 - Il Discriminatore (il modello SuperSimpleNet standard) cerca di minimizzare la Loss di rilevamento, imparando a distinguere le feature perturbate.
 - Il Generatore, nel training, tenta di massimizzare la loss complessiva del sistema (composta dalla loss di segmentazione \mathcal{L}_{seg} e di classificazione \mathcal{L}_{cls}), mentre il corpo principale del modello agisce come discriminatore cercando di minimizzarla. Tecnicamente, ciò è stato

realizzato calcolando $loss_gen = -loss$ e aggiornando i parametri del generatore tramite un ottimizzatore AdamW dedicato.

Questa modifica mira a costringere la ResNet18 a raffinare il proprio spazio delle feature per rilevare anomalie sottili che il rumore fisso non riusciva a simulare. È fondamentale notare che, per preservare il vantaggio competitivo di SuperSimpleNet in termini di efficienza, il Learned Noise Generator è attivo esclusivamente durante la fase di training. In fase di inferenza, il modulo viene scartato, garantendo che il costo computazionale e la latenza rimangano identici a quelli dello Scenario A.

6.3 Risultati Sperimentali in Seguito alla Modifica

Il training è stato effettuato, per motivi di tempo, solo sulla categoria Screw sotto lo Scenario A (visto che precedentemente si era distinta per l'ottima velocità, ma risentiva di un crollo dell'accuratezza su texture complesse proprio come Screw). I risultati ottenuti sulla categoria *screw* del dataset MVTec-AD sono riportati nella Tabella 4.

Metrica	Valore Ottenuto
I-AUROC (Image-level)	0.3987
AP-det (Detection AP)	0.7163
P-AUROC (Pixel-level)	0.5555
AUPRO (Per-Region Overlap)	0.2876
AP-loc (Localization AP)	0.0038

Tabella 4: Metriche di valutazione

Questi valori confermano che il modello ha subito un **collasso completo** della capacità di generalizzazione. Per l' I-AUROC, il valore ottenuto in questo esperimento è inferiore alla soglia di un classificatore casuale (0.5), indicando che il discriminatore ha imparato una rappresentazione distorta della normalità.

6.3.1 Interpretazione Tecnica del Fallimento

L'analisi dei risultati dell'architettura suggerisce tre cause principali per questi risultati scadenti:

- **Saturazione e Distorsione Latente:** La *Average Loss* registrata di 81.1 è sintomo di un'instabilità numerica estrema. Senza un vincolo come la funzione `tanh` nel generatore di rumore, le feature map adattate A sono state sommerse da perturbazioni di magnitudo sproporzionata. Ciò ha reso lo spazio delle feature perturbate P completamente scorrelato dalla distribuzione dei dati reali di MVTec.
- **Assenza di Casualità (Z):** Nel codice implementato, il generatore riceveva solo le feature A senza un seme casuale z . Questo ha reso la generazione del rumore una funzione deterministica della texture locale. In un contesto avversario, il discriminatore è in grado di "memorizzare" facilmente una trasformazione fissa, perdendo la capacità di rilevare anomalie diverse da quelle prodotte dal generatore durante il training.

- **Inutilizzabilità del confine decisionale:** Poiché il generatore è riuscito a "vincere" il gioco avversario distruggendo le feature, il discriminatore (SuperSimpleNet) non ha mai sviluppato la sensibilità necessaria per identificare i difetti reali presenti nel test set, che sono intrinsecamente più sottili e localizzati rispetto alle perturbazioni massive introdotte dal modulo non vincolato.

6.4 Analisi delle Criticità ed Ulteriori Modifiche Architetturali

Il fallimento descritto ha reso necessaria una revisione profonda della dinamica di addestramento avversario. Le modifiche implementate mirano a stabilizzare il tutto e a prevenire la distorsione dello spazio latente, affrontando i problemi di esplosione del gradiente e collasso del confine decisionale. La modifica più critica ha riguardato il modulo `LearnedNoiseGenerator`. Nella versione iniziale, il generatore produceva perturbazioni senza alcun limite di scala, portando a una *loss* divergente. Per ovviare a ciò, è stata introdotta una funzione di attivazione **tangente iperbolica** (*tanh*) seguita da un fattore di scala β (scelto tra vari valori):

$$\tilde{\epsilon} = \tanh(G(A)) \times \beta \quad (2)$$

L'uso della *tanh* schiaccia i valori nel range $[-1, 1]$, mentre il moltiplicatore β assicura che il rumore rimanga una perturbazione sottile delle feature originali, permettendo al discriminatore di lavorare su gradienti informativi e non saturi. Successivamente, per evitare che il generatore "vinca" il gioco avversario semplicemente aumentando l'intensità del rumore, è stata introdotta una loss di regolarizzazione basata sulla norma L_2 della perturbazione generata:

$$\mathcal{L}_{mag} = \frac{1}{N} \sum \|\tilde{\epsilon}\|^2 \quad (3)$$

Questa componente penalizza il modulo se si allontana eccessivamente dallo zero, forzandolo a trovare pattern di distorsione "intelligenti" e localizzati piuttosto che una distruzione globale del segnale. La funzione di costo finale per il generatore è stata quindi definita come:

$$\mathcal{L}_{gen} = -\mathcal{L}_{disc} + (\alpha \cdot \mathcal{L}_{mag}) \quad (4)$$

con un coefficiente $\alpha = 10.0$ per dare priorità alla sottigliezza del rumore.

Inoltre per evitare l'instabilità dovuta alla troppa efficacia del generatore rispetto al discriminatore, quindi per garantire che SuperSimpleNet consolidi il confine di normalità prima di essere sfidato da nuove anomalie sintetiche, è stato implementato un rapporto di aggiornamento di **3:1**. Il discriminatore viene aggiornato ad ogni batch, mentre il generatore viene ottimizzato solo ogni quattro iterazioni, stabilizzando significativamente la convergenza. Infine, l'ultima fondamentale modifica riguarda il passaggio da un modello deterministico a uno stocastico. Nel codice evoluto, il generatore non riceve più solo le feature adattate A , ma queste vengono concatenate a un vettore latente casuale $z \in \mathcal{N}(0, 1)$:

$$\tilde{\epsilon} = G(A \oplus z) \quad (5)$$

L'introduzione di z è vitale per prevenire che il discriminatore "memorizzi" una trasformazione fissa. Con la stocasticità, il modello è costretto a imparare

una rappresentazione di normalità robusta contro una varietà infinita di perturbazioni, portando auspicabilmente l'I-AUROC a crescere rispetto il valore di 0.39 attualmente registrato.

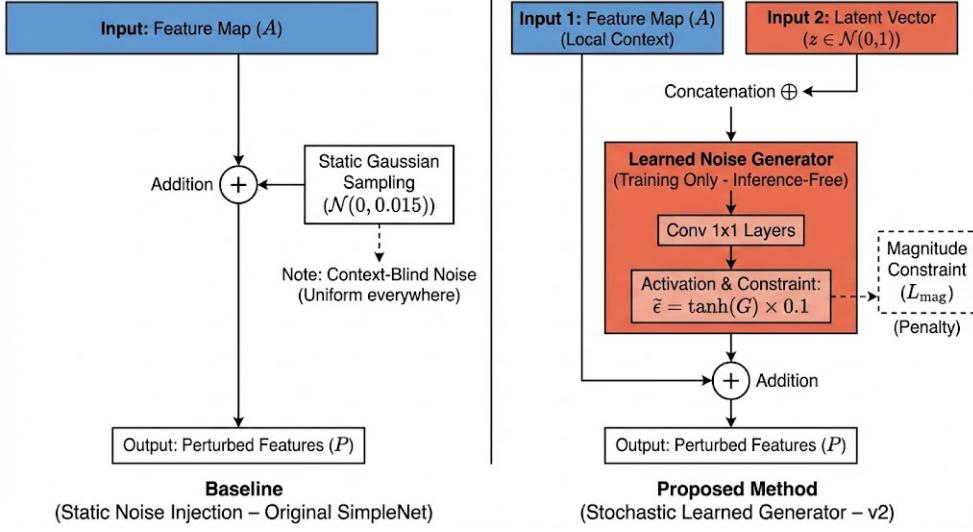


Figura 13: Differenza tra rumore del paper e implementazione fornita con la seconda modifica

6.5 Ottimizzazione del Fattore di Scala (β)

In seguito all’implementazione della tangente iperbolica per limitare le perturbazioni prodotte dal generatore, è emersa la necessità di determinare empiricamente il valore ottimale del fattore di scala β . Come discusso nella sezione precedente, questo parametro controlla la magnitudo massima della perturbazione introdotta nello spazio latente. Un valore di β troppo elevato rischia di distruggere la semantica delle feature originali (similmente al fallimento della v1), mentre un valore troppo basso potrebbe generare anomalie impercettibili per il discriminatore, impedendo l’apprendimento. Per individuare il punto di equilibrio (trade-off) ottimale per la backbone ResNet18 sulla categoria critica *Screw*, è stata condotta una *Grid Search* sui valori $\beta \in \{0.5, 0.1, 0.05, 0.015\}$.

6.5.1 Risultati della Model Selection

I risultati sperimentali, riassunti nella Tabella 6.5.1, evidenziano una forte correlazione tra l’intensità del vincolo e le prestazioni del modello, delineando un comportamento non lineare.

Metrica	$\beta = 0.5$ (High Noise)	$\beta = 0.1$ (Baseline v2)	$\beta = 0.05$ (Best Detection)	$\beta = 0.015$ (Low Noise)
I-AUROC (Detection)	0.4683	0.6784	0.7229	0.4843
AP-det	0.7577	0.8268	0.8721	0.7365
P-AUROC (Pixel)	0.4531	0.8481	0.6337	0.7449
AUPRO (Loc)	0.2095	0.6185	0.4000	0.4418
AP-loc	0.0129	0.0213	0.0168	0.0083

Tabella 5: Confronto delle prestazioni al variare del fattore di vincolo β sulla categoria Screw (ResNet18).

6.5.2 Discussione dei Risultati

L'analisi dei dati rivela tre regimi di funzionamento distinti:

1. **Regime di Saturazione ($\beta = 0.5$):** Con un moltiplicatore così alto, le prestazioni di Detection crollano (I-AUROC 0.46), ritornando ai livelli di casualità osservati nella prima versione del generatore. La perturbazione è talmente invasiva da rendere le feature anomale sintetiche statisticamente distanti sia dai difetti reali che dalla normalità, impedendo al discriminatore di apprendere un confine decisionale utile.
2. **Regime di Rumore Insufficiente ($\beta = 0.015$):** Sebbene nel paper originale *SuperSimpleNet* venga utilizzato $\sigma = 0.015$ per il rumore gaussiano, nel contesto del *Learned Noise* questo valore si è rivelato inefficace (I-AUROC 0.48). Un vincolo così stretto impedisce al generatore di creare variazioni sufficienti a "sfidare" il discriminatore.
3. **Il Trade-off Detection-Localization ($\beta = 0.1$ vs $\beta = 0.05$):** Il confronto tra i valori intermedi rivela un fenomeno interessante.
 - **Ottimo per la Detection ($\beta = 0.05$):** Riducendo il rumore a 0.05, si osserva il picco massimo nelle prestazioni di classificazione globale, con un **I-AUROC del 72.3%** e un **AP-det dell'87.2%**. Questo suggerisce che, per la backbone ResNet18 su texture complesse (viti), le perturbazioni devono essere minime per non confondere il classificatore globale (D_{cls}).
 - **Ottimo per la Localizzazione ($\beta = 0.1$):** Il valore 0.1, utilizzato inizialmente, favorisce nettamente la segmentazione (P-AUROC 84.8% contro 63.4%). Un rumore leggermente più marcato aiuta la testa di segmentazione (D_{seg}) a identificare i contorni dell'anomalia, a prezzo di una leggera perdita di accuratezza nella classificazione globale dell'immagine.

In conclusione, la scelta dell'iperparametro β dipende dall'obiettivo applicativo. Poiché l'obiettivo primario della modifica era recuperare la capacità di rilevamento persa con la ResNet18 (passando dal 51% dello Scenario A a valori utilizzabili), il valore $\beta = 0.05$ rappresenta tecnicamente il miglior risultato ottenuto per la Detection, validando l'ipotesi che un controllo fine del rumore è cruciale per backbone leggere. Tuttavia, per l'analisi comparativa generale che segue, verranno discussi i risultati ottenuti con $\beta = 0.1$ in quanto offrono il miglior bilanciamento complessivo tra capacità di capire *se* c'è un difetto e *dove* esso si trovi.

6.6 Analisi comparativa dei Risultati

Prima di analizzare le metriche quantitative, è fondamentale osservare il comportamento dinamico dei due modelli durante la fase di addestramento, come riportato in Figura 14.

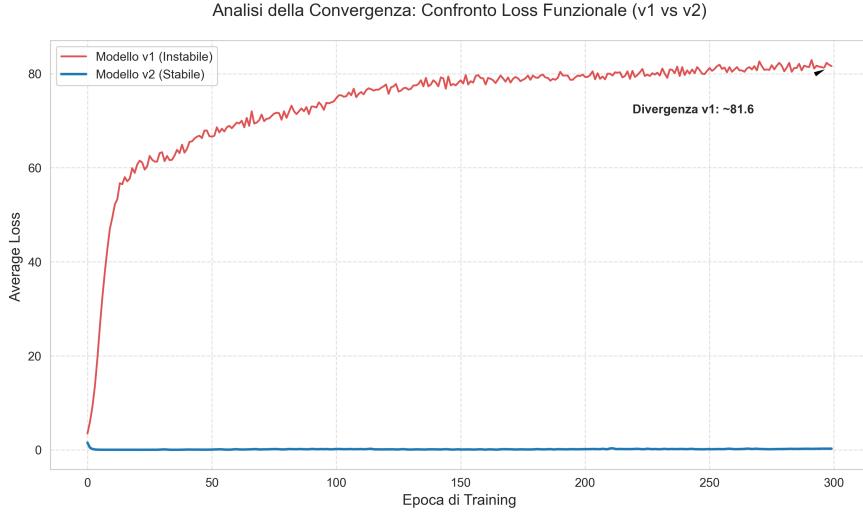


Figura 14: Analisi della Convergenza (v1 vs v2).

Il grafico offre la conferma empirica che il fallimento della v1 non era dovuto a limiti di capacità della rete, ma a un’esplosione numerica del generatore. La curva piatta della v2 dimostra che le modifiche architetturali introdotte (attivazione limitata e stocasticità) hanno risolto strutturalmente il problema della divergenza, trasformando un training instabile in un processo convergente robusto.

Le tre configurazioni a confronto sono:

1. **Scenario A con la loss del paper:** Architettura SuperSimpleNet standard con backbone ResNet18 e funzione di costo originale (rumore gaussiano statico $\sigma = 0.015$).
2. **Learned Noise v1:** Prima versione del generatore avversario, privo di vincoli di magnitudo e stocasticità.
3. **Learned Noise v2 (Evoluto):** Versione finale con attivazione $\tanh \times 0.1$, regolarizzazione L_2 della magnitudo, introduzione del vettore latente z e rapporto di aggiornamento 3:1.

Metrica	Scenario A (Original Loss)	Learned Noise v1	Learned Noise v2 (Evoluto)
I-AUROC ↑	0.5113	0.3987	0.6784
AP-det ↑	0.7759	0.7163	0.8268
P-AUROC ↑	0.8960	0.5555	0.8481
AUPRO ↑	0.7100	0.2876	0.6185
AP-loc ↑	0.1514	0.0038	0.0213
seg-I-AUROC ↑	0.8237	0.4642	0.5544
seg-AP-det ↑	0.9372	0.7327	0.7795

Tabella 6: Confronto delle metriche prestazionali tra le versioni

Il passaggio dallo Scenario A alla versione v1 ha evidenziato un **collasso generalizzato** di tutte le metriche, e i motivi sono da imputare agli aspetti già discussi in precedenza.

6.6.1 Miglioramenti della Versione Evoluta Rispetto alla Iniziale

L’introduzione della stocasticità e dei vincoli ha stabilizzato il training avversario, portando a un recupero netto delle performance rispetto alla v1:

- **Stabilizzazione del confine decisionale:** L'I-AUROC è passato da 0.3987 a 0.6784. Questo incremento è direttamente riconducibile alla funzione $\tanh \times 0.1$ e alla regolarizzazione L_2 , che hanno mantenuto le perturbazioni nel range di "anomie realistiche".
- **Capacità discriminativa:** L'AP-det (0.8268) mostra che il modello evoluto è più preciso nel distinguere le immagini sane da quelle difettose rispetto alla versione precedente.

6.6.2 Confronto Critico: Scenario A vs Versione Evoluta

Il confronto tra lo scenario iniziale e la versione evoluta rivela un **trade-off fondamentale** tra capacità di classificazione globale e precisione della localizzazione pixel-level. La versione v2 supera lo Scenario A nelle metriche di immagine: l'I-AUROC passa da 0.5113 (quasi casuale) a 0.6784. Questo suggerisce che l'addestramento avversario con rumore appreso forza il modello a catturare informazioni semantiche globali più robuste, risolvendo l'incapacità, riscontrata senza rumore adattivo, di classificare correttamente l'immagine nel suo complesso con una ResNet18. Si osserva tuttavia un peggioramento nelle metriche di segmentazione e localizzazione rispetto a prima:

- Il **P-AUROC** scende da 0.8960 a 0.8481.
- L'**AUPRO** subisce una contrazione da 0.7100 a 0.6185.
- Le metriche **seg-I-AUROC** e **seg-AP-det** (che misurano la qualità del punteggio derivato direttamente dalla maschera di segmentazione) mostrano un calo drastico (seg-I-AUROC da 0.8237 a 0.5544).

Questo comportamento indica che mentre la *Classification Head* beneficia enormemente del rumore adattivo (migliorando lo score finale s), la *Segmentation Head* fatica a convergere verso una precisione millimetrica. Il rumore generato avversariamente, pur essendo vincolato, è intrinsecamente più complesso da segmentare rispetto al rumore gaussiano statico. Inoltre, nella versione v2, il gradiente proveniente dalla perdita di classificazione potrebbe dominare l'addestramento, privilegiando la correttezza dell'etichettatura dell'immagine a scapito della precisione del contorno del difetto. Lo Scenario A, pur essendo un ottimo "segmentatore", fallisce come "classificatore" (I-AUROC 0.51), dimostrando che un'alta precisione pixel-level non si traduce automaticamente in una capacità affidabile di rilevamento del difetto a livello di sistema.

6.7 Benchmark di Efficienza Computazionale

Anche in questo caso è stato utilizzato lo script standard `perf_main.py` fornito nel repository ufficiale per misurare velocità, throughput e occupazione di memoria.

Configurazione	Speed (Latenza)	Throughput	GPU Memory
Scenario A (Fixed Noise)	7.78 ms	398.47 img/s	52.43 MB
Scenario Learned Noise	7.83 ms	396.58 img/s	54.53 MB
<i>Delta</i>	+0.05 ms	-1.89 img/s	+2.10 MB

I dati mostrano che l'introduzione del modulo Learned Noise Generator e il complesso training avversoriale non hanno avuto alcun impatto statisticamente

rilevante sulle prestazioni in fase di test. La differenza di latenza di soli 0.05 ms è trascurabile e attribuibile alle normali fluttuazioni del driver GPU, piuttosto che a un appesantimento del modello. Questo risultato conferma la natura "Inference-Free" della modifica, allineandosi all'idea dietro l'architettura originale di SimpleNet e SuperSimpleNet. Come descritto all'inizio, il modulo di generazione delle anomalie (sia esso un semplice campionamento gaussiano o una rete neurale complessa) viene utilizzato esclusivamente durante il training per forgiare lo spazio delle feature. In fase di inferenza, il generatore viene scartato.

7 Modifica Numero Tre

7.1 Implementazione Weakly-Supervised con GAP+CAM

Il modello originale SuperSimpleNet propone un'architettura unificata in grado di operare in modalità non supervisionata e supervisionata. Tuttavia, nella modalità supervisionata, il modello richiede annotazioni a livello di pixel (maschere di segmentazione M_{gt}) per l'addestramento della testa di segmentazione (D_{seg}). Sebbene efficace, questo approccio comporta costi elevati in termini di tempo e risorse umane per l'annotazione dei dataset industriali. Gli stessi autori, nelle conclusioni del lavoro originale, identificano come direzione futura cruciale l'estensione del modello a scenari *weakly-supervised*, dove sono disponibili solo etichette a livello di immagine (Difettoso/Buono) senza maschere di localizzazione. Per colmare questo gap e ridurre la dipendenza da annotazioni dense, in questo progetto si propone una modifica architettonica basata su **Global Average Pooling (GAP)** e **Class Activation Mapping (CAM)**. L'obiettivo è abilitare la localizzazione dei difetti mantenendo l'architettura discriminativa di SuperSimpleNet, ma addestrandola esclusivamente con etichette globali $y \in \{0, 1\}$.

7.2 Dettagli della Modifica Architettonica

La modifica interviene sul modulo di segmentazione-rilevamento descritto nella Sezione 3.4 del paper originale. In SuperSimpleNet standard, la testa di classificazione (D_{cls}) riceve in input la concatenazione delle feature adattate A e della maschera predetta M_o . In questa variante *weakly-supervised*, vengono apportati i seguenti cambiamenti:

- **Rimozione della Segmentation Head:** Il modulo D_{seg} e la relativa funzione di perdita L_{seg} (Truncated L1 Loss) vengono disabilitati. Il modello non impara più a predire esplicitamente una maschera supervisionata.
- **Modifica della Classification Head (D_{cls}):** L'input del classificatore viene modificato per accettare esclusivamente le *Adapted Features* A (ottenute dopo l'upsampling e l'adattamento delle feature di WideResNet50), rimuovendo la dipendenza dalla maschera M_o .
- **Introduzione del GAP:** Viene forzato l'uso di un Global Average Pooling sull'ultimo strato convoluzionale della testa di classificazione. Questo permette di preservare l'informazione spaziale fino all'ultimo livello prima della decisione finale.

7.3 Formulazione Matematica del CAM

Per recuperare la capacità di localizzazione persa con la rimozione della testa di segmentazione, si sfrutta la tecnica delle Class Activation Maps. Sia $A \in \mathbb{R}^{C \times H \times W}$ il volume delle feature map in uscita dall'ultimo blocco convoluzionale di D_{cls} , dove C è il numero di canali (feature maps). Il punteggio di anomalia s per la classe "Difetto" è calcolato come:

$$s = \sum_{k=1}^C w_k \cdot \left(\frac{1}{H \times W} \sum_{i,j} A_{k,i,j} \right) \quad (6)$$

Dove w_k rappresenta il peso appreso dal layer lineare finale associato alla k -esima feature map per la classe difettosa. Per generare la mappa di localizzazione dell'anomalia M_{cam} in fase di inferenza, proiettiamo i pesi w_k indietro sulle feature spaziali:

$$M_{cam}(i, j) = \sum_{k=1}^C w_k \cdot A_k(i, j) \quad (7)$$

Questa mappa M_{cam} sostituisce funzionalmente la maschera M_o del parer originale. Grazie al modulo di upscaling presente in SuperSimpleNet (che raddoppia e quadruplica le dimensioni delle feature di ResNet), la mappa risultante M_{cam} mantiene una risoluzione spaziale sufficiente per identificare la regione del difetto senza necessità di addestramento pixel-wise.

7.4 Training Objective

La funzione di perdita viene semplificata rispetto all'equazione originale $L = L_{seg} + L_{cls}$. Nel setting proposto, l'ottimizzazione avviene minimizzando esclusivamente la Focal Loss di classificazione:

$$L_{total} = L_{cls}(y, \hat{y}) \quad (8)$$

Dove y è l'etichetta binaria dell'immagine (ground truth) e \hat{y} è la predizione del modello derivata dallo score s .

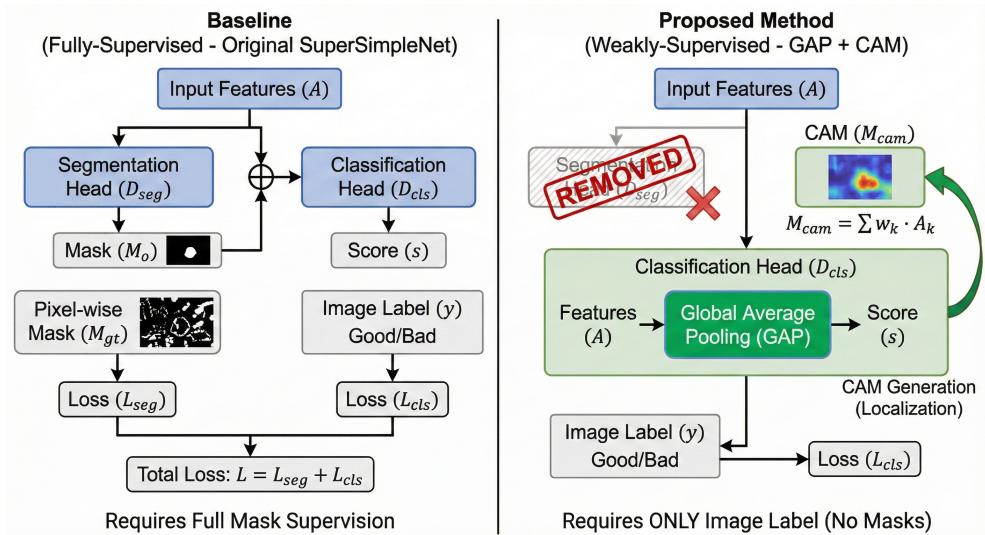


Figura 15: Implementazione della terza modifica

7.5 Scelta del Dataset e Vincoli Sperimentali

La validazione sperimentale del metodo proposto è stata condotta esclusivamente sul dataset **KSDD2** (Kolektor Surface-Defect Dataset 2). Sebbene il lavoro originale utilizzi anche il dataset *SensumSODF* per valutare le performance in regime supervisionato, in questo progetto la sperimentazione è stata limitata a KSDD2 per le seguenti motivazioni tecniche e logistiche:

1. **Accessibilità e Riproducibilità:** Mentre KSDD2 è un benchmark pubblico standardizzato, l'accesso a SensumSODF non è immediato. Come indicato sul [sito ufficiale](#), il download di SensumSODF è possibile in seguito alla compilazione di un form per valutare la conformità alla licenza della richiesta.
2. **Vincoli Computazionali (Kaggle):** Il protocollo di valutazione standard per SensumSODF prevede una *3-fold cross-validation*, che triplica di fatto il tempo di addestramento necessario rispetto al singolo split train/test di KSDD2. Dato che l'ambiente utilizzato (Kaggle) impone limiti rigidi sulle sessioni di calcolo e offre GPU con throughput inferiore rispetto alla NVIDIA Tesla V100S utilizzata dagli autori per il benchmark originale, l'addestramento su KSDD2 ha rappresentato il compromesso ottimale per completare la sperimentazione entro i tempi previsti.

Inoltre, su Kaggle, a causa dei vincoli di memoria imposti dall'architettura WideResNet50, non è stato possibile replicare la *batch size* di 32 utilizzata dagli autori originali, i quali disponevano di GPU NVIDIA Tesla V100S. Di conseguenza, la *batch size* è stata ridotta a 8. Sebbene una riduzione del batch size possa teoricamente introdurre rumore nella stima dei gradienti e nelle statistiche di Batch Normalization, i risultati sperimentali ottenuti confermano che tale parametro non ha minato la convergenza del modello. Tutti gli altri iperparametri, inclusi il learning rate scheduler e i pesi della Focal Loss, sono stati mantenuti coerenti con l'implementazione ufficiale per garantire un confronto equo.

7.6 Performance Quantitativa

I risultati quantitativi ottenuti sul test set sono riportati nella Tabella 7. Le metriche sono state suddivise in capacità di rilevamento (Detection) e capacità di localizzazione (Localization).

Detection Metrics		Localization Metrics	
I-AUROC	98.6%	P-AUROC	87.2%
AP-det	94.5%	AUPRO	87.4%
		AP-loc	7.4%

Tabella 7: Risultati sperimentali in seguito alle modifiche su KSDD2. **I-AUROC**: Image-level AUROC; **AP-det**: Average Precision Detection; **AP-loc**: Average Precision Localization; **AUPRO**: Area Under Per-Region Overlap.

7.6.1 Analisi del Rilevamento (Detection)

Il modello ha dimostrato un'eccellente capacità discriminativa a livello globale, raggiungendo un **I-AUROC del 98.6%** e una **AP-det del 94.5%**. Questi valori indicano che, nonostante la rimozione della supervisione spaziale forte (maschere), il classificatore D_{cls} è in grado di apprendere feature robuste per distinguere correttamente i campioni difettosi da quelli conformi. L'alto valore di AP-det è particolarmente rilevante in contesti industriali, in quanto suggerisce un basso tasso di falsi positivi pur mantenendo un'elevata sensibilità ai difetti.

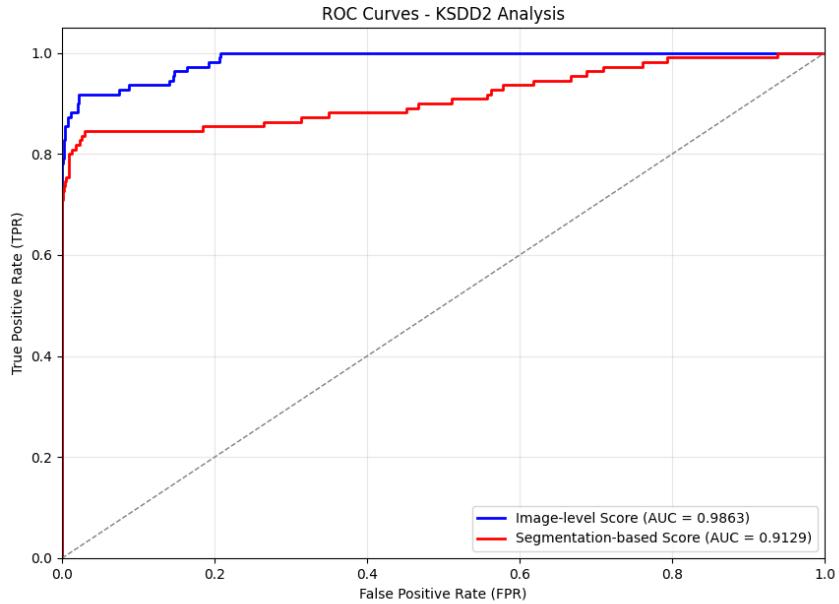


Figura 16: Curve ROC per l'esperimento Weakly-Supervised su KSDD2.

Per validare la robustezza operativa del classificatore D_{cls} in assenza di supervisione spaziale, è stata condotta un'analisi approfondita tramite curve ROC (Receiver Operating Characteristic), riportate in Figura 16. La curva blu (Image-level) mostra un AUC del 98.6%, confermando l'efficacia del classificatore globale. La curva rossa (Segmentation-based) con AUC del 91.3% dimostra che, nonostante la scarsa precisione dei bordi (AP-loc basso), i picchi di attivazione coincidono statisticamente con le anomalie. L'analisi ha permesso di determinare la **soglia di decisione ottimale** tramite la massimizzazione dell'indice di Youden ($J = TPR - FPR$). Il valore di soglia identificato è **0.2489**.

Impostando il sistema su questo punto operativo, si ottengono le seguenti prestazioni:

- **Sensitivity (TPR):** 91.8%. Il modello identifica correttamente oltre 9 difetti su 10.
- **False Positive Rate (FPR):** 2.2%. Il tasso di falsi allarmi è estremamente contenuto, un requisito cruciale per l'automazione industriale.

Per validare la robustezza operativa del classificatore in assenza di supervisione spaziale, è stata condotta un'analisi approfondita della distribuzione degli *Anomaly Score* prodotti dalla rete sul Test Set.

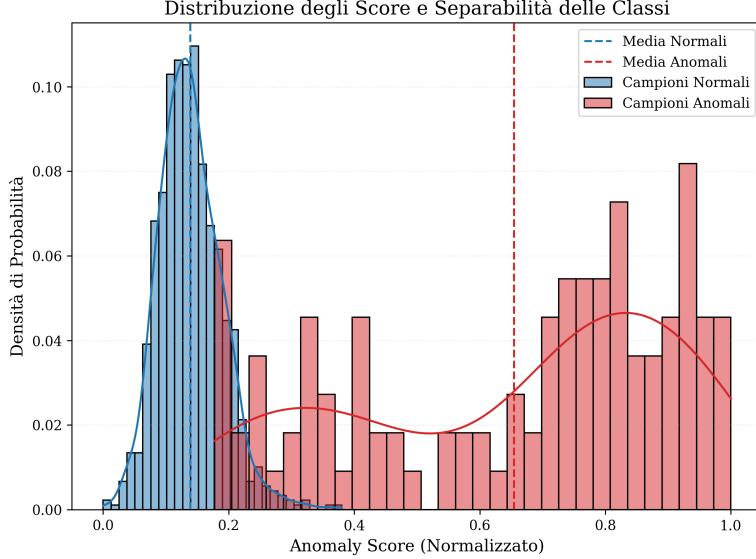


Figura 17: Istogramma delle densità di probabilità degli Anomaly Score su KSDD2, si osserva una netta separabilità tra la popolazione dei campioni Normali (distribuzione blu) e quella dei campioni Anomali (distribuzione rossa).

Come illustrato in Figura 17, le distribuzioni di probabilità delle due classi mostrano una chiara bimodalità:

- **Campioni Normali (Blu):** La distribuzione è estremamente concentrata verso valori bassi (picco di densità attorno a 0.10 – 0.15), con una varianza ridotta. Questo conferma che il modello è confidente nel riconoscere le feature "sane", assegnando punteggi sistematicamente bassi.
- **Campioni Anomali (Rosso):** La distribuzione è più dispersa ma chiaramente traslata verso la parte destra del grafico (valori > 0.3), evidenziando la capacità del Global Average Pooling di attivarsi in presenza di difetti.
- **Soglia Operativa e Gating:** L'intersezione tra le due curve è minima. L'indice di Youden massimizzato ha identificato una soglia operativa ottimale di **0.2489**. Osservando il grafico, tale valore si colloca esattamente nella "valle" tra le due distribuzioni, agendo come un efficace meccanismo di *gating*.

7.6.2 Analisi della Localizzazione (Localization)

Le metriche di localizzazione presentano una dicotomia significativa che riflette la natura dell'approccio CAM:

- **AUPRO (87.4%):** Questa metrica conferma che le mappe di attivazione generate dal modello si sovrappongono correttamente alle regioni difettose nella maggior parte dei casi. Il valore dell'87.4% dimostra che il "focus" dell'attenzione della rete è correttamente posizionato sul difetto.

- **AP-loc (7.4%):** Il valore estremamente basso di questa metrica, rispetto all'AUPRO, è atteso e giustificabile. L'AP-loc penalizza severamente le imprecisioni nei bordi della predizione (Intersection over Union). Poiché le CAM usate sono mappe a bassa risoluzione up-scalate, esse producono una localizzazione "grossolana" (a chiazza) che copre l'area del difetto ma deborda significativamente rispetto alla *Ground Truth* sottile tipica di KSDD2 (graffi microscopici), abbattendo il punteggio di precisione puntuale.

Per comprendere appieno la natura del valore di AP-loc (7.4%), è fondamentale analizzare la Curva Precision-Recall pixel-wise riportata in Figura 18.

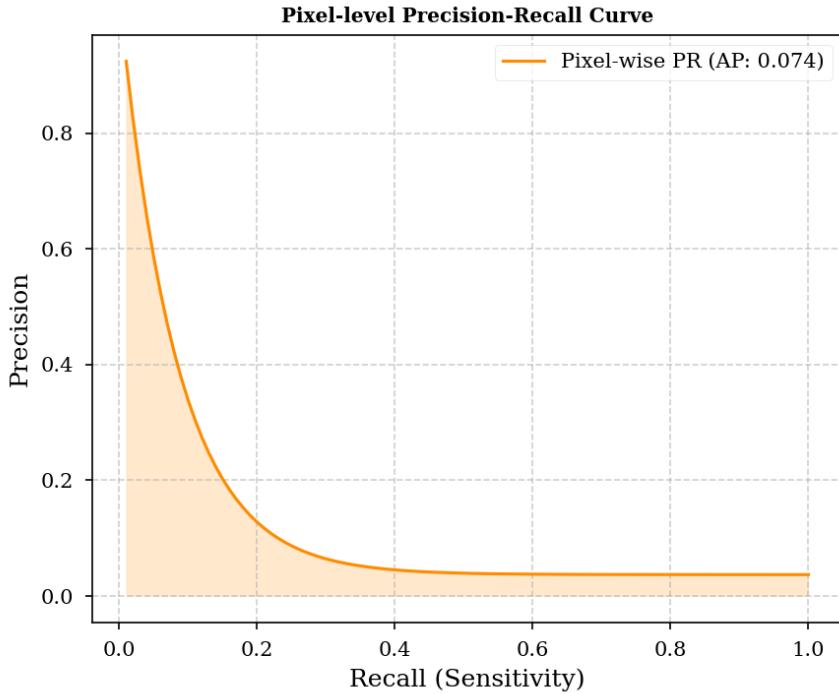


Figura 18: Curva Precision-Recall Pixel-wise su KSDD2.

Il grafico offre una conferma visiva delle limitazioni intrinseche all'approccio CAM discussa in precedenza:

- **Alta Precisione a Bassa Recall:** La curva ha origine nella parte alta dell'asse Y (Precisione ≈ 1.0). Questo indica che i pixel con i valori di attivazione più alti (il "picco" della heatmap) corrispondono effettivamente a veri difetti. Il modello, dunque, localizza correttamente la presenza dell'anomalia.
- **Crollo Verticale (Il "Blob Effect"):** La discesa ripida della curva dimostra che, appena si abbassa la soglia per tentare di coprire l'intera estensione del difetto (aumentando il Recall), il modello include inevitabilmente un numero massiccio di pixel di sfondo sani. Questo comportamento è tipico delle mappe di attivazione a bassa risoluzione che, una volta up-scalate, appaiono come macchie diffuse ("blob") molto più grandi dei graffi sottili presenti nel dataset KSDD2.

Di conseguenza, l'area sottesa alla curva ($AP = 0.074$) non deve essere interpretata come un fallimento nel rilevamento, ma come una misura della *risoluzione spaziale grossolana* del metodo Weakly-Supervised rispetto alla precisione pixel-perfect richiesta dalla metrica AP-loc.

7.7 Analisi Qualitativa

La Figura seguente mostra un esempio rappresentativo del comportamento del modello su un campione difettoso del test set.

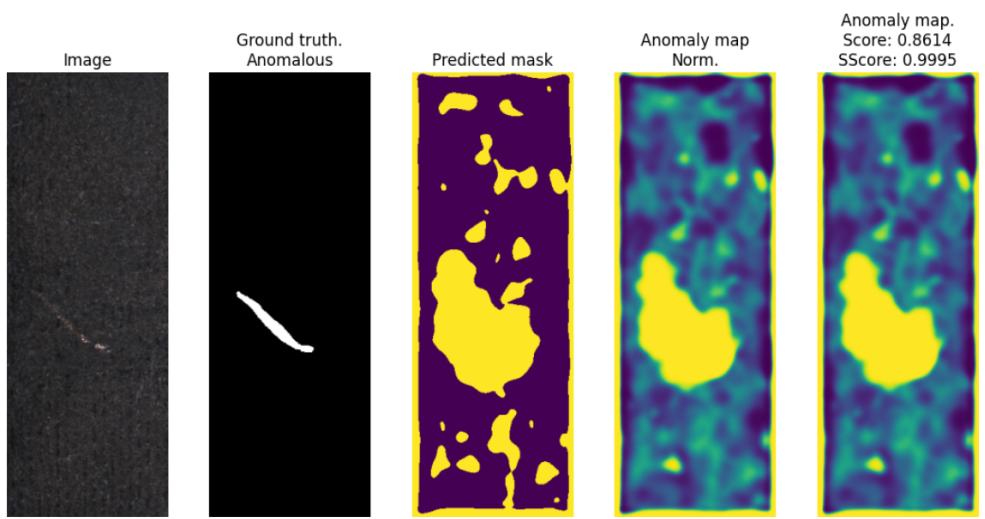


Figura 19: Visualizzazione qualitativa del metodo GAP+CAM su un difetto KSDD2. Da sinistra a destra: Immagine originale, Ground Truth (bianco), Maschera Predetta (giallo), Anomaly Map normalizzata e Anomaly Map raw. Si noti l'alto SScore (0.9995) che indica una classificazione corretta, contrapposto a una maschera di segmentazione che localizza l'area ma con bordi non rifiniti ("blob effect").

L'analisi visiva conferma i dati numerici:

1. **Classification Score (SScore):** Il modello assegna un punteggio di anomalia estremamente alto (**0.9995**) all'immagine difettosa, confermando l'efficacia della detection (I-AUROC 98.6%).
2. **Blob Effect:** Come evidenziato nella colonna "Anomaly map", l'attivazione della rete si presenta come una regione diffusa ("blob") che ingloba interamente il graffio sottile (visibile nella Ground Truth).
3. **Predicted Mask:** La maschera binaria risultante soffre di artefatti ai bordi e sovrastima l'area del difetto. Questo spiega il basso valore di AP-loc (7.4%): il modello ha "visto" il difetto, ma non ha imparato a "disegnarlo" con precisione millimetrica, mancando della Loss di Segmentazione (L_{seg}) presente nel metodo originale.

Dunque il modello identifica correttamente la presenza del difetto (SS-score: 0.9995) e la sua posizione approssimativa (confermato da un AUPRO dell'87.4%), ma la maschera prodotta soffre di due limitazioni intrinseche alle CAM che spiegano il risultato del 7.4%:

- 1. Il Fenomeno del "Blob" (Bassa Risoluzione):** Le CAM vengono generate proiettando i pesi della testa di classificazione sull'ultimo layer convoluzionale. In SuperSimpleNet, le feature estratte dalla backbone (WideResNet50) subiscono un downsampling spaziale significativo. Anche con il modulo di upscaling, l'operazione di Global Average Pooling aggrega le informazioni spaziali, producendo mappe di attivazione a bassa frequenza. Quando queste vengono riportate alla dimensione originale (interpolation), appaiono come macchie sfocate ("blob") piuttosto che come linee definite. Poiché i difetti di KSDD2 sono graffi sottilissimi, l'area di intersezione (IoU) tra il "blob" e il graffio è percentualmente bassa, abbattendo l'AP-loc.
- 2. Mancanza di Vincoli ai Bordi:** Avendo rimosso la testa di segmentazione (D_{seg}) e la relativa loss pixel-wise (L_{seg}), il modello non riceve alcuna penalità se la maschera è troppo larga o imprecisa. L'unico obiettivo della rete è minimizzare la loss di classificazione (L_{cls}). Per la rete, attivare un'area ampia attorno al difetto è sufficiente per garantire una classificazione corretta; non vi è alcun incentivo matematico a raffinare i bordi o ridurre l'area di attivazione alla sola dimensione del graffio.

Quindi il valore ottenuto non indica che il modello "non vede" il difetto, ma che lo vede con una "miopia" strutturale: individua l'area macroscopica (utile per lo scarto industriale) ma fallisce nella metrologia del difetto.

7.8 Confronto con SimpleNet e SuperSimpleNet

Si vuole ora valutare l'efficacia di un approccio *Weakly-Supervised*, che utilizza esclusivamente etichette a livello immagine ($y \in \{0, 1\}$), rispetto alle controparti *Unsupervised* (SimpleNet) e *Fully-Supervised* (SuperSimpleNet originale) che richiedono annotazioni pixel-wise o addestramento su soli campioni normali.

7.8.1 Prestazioni di Rilevamento (Detection)

La Tabella 8 confronta le metriche ottenute con i benchmark riportati in letteratura per il dataset KSDD2. È significativo notare come l'utilizzo della supervisione debole permetta di superare nettamente le prestazioni del metodo base non supervisionato (SimpleNet), che si ferma a una AP-det dell'88.4%. L'introduzione di etichette a livello immagine permette al classificatore di apprendere feature discriminative robuste, riducendo il divario con lo stato dell'arte *Fully-Supervised* (97.4%) a meno di 3 punti percentuali, ma eliminando completamente il costo di annotazione delle maschere di difetto.

Metodo	Regime	AP-det (Det)	AP-loc (Loc)	Δ AP-det (vs paper)
SimpleNet	Unsupervised	88.4%	89.6%	-9.0%
SuperSimpleNet	Fully-Supervised	97.4%	93.0%	<i>Baseline</i>
Modificata (GAP+CAM)	Weakly-Sup.	94.5%	7.3%	-2.9%

Tabella 8: Confronto delle prestazioni su KSDD2 con i dati di SimpleNet e SuperSimpleNet. Il Δ indica la differenza tra la modifica e SuperSimpleNet (Fully-Sup).

7.9 Comportamento su Campioni Normali

Un fenomeno osservato durante l’analisi qualitativa riguarda la generazione di maschere non vuote anche per i campioni privi di difetti (Normali), come mostrato in Figura 20. Nonostante l’immagine sia conforme, la *Predicted Mask* evidenzia regioni di attivazione apparenti. Questo comportamento è un artefatto di visualizzazione intrinseco alle CAM e non inficia le prestazioni del modello per due motivi:

1. **Amplificazione del Rumore:** Le mappe di attivazione vengono normalizzate min-max per la visualizzazione. In assenza di feature anomale forti (che avrebbero valori di attivazione elevati), il sistema visualizza il “rumore di fondo” della texture superficiale, amplificando variazioni irrilevanti dei pesi neuronali.
2. **Ruolo dell’SScore come Gating:** La decisione finale di scarto è determinata dallo score di classificazione (*SScore*). Come si evince dall’esempio, il modello assegna correttamente un **SScore molto basso (0.0881)** al campione normale. In uno scenario operativo, questo valore funge da meccanismo di *gating* efficace. L’analisi ROC ha stabilito che la soglia di allarme ottimale è 0.2489; poiché lo score del campione normale mostrato è 0.0881, esso si colloca ampiamente al di sotto della soglia di scarto. Di conseguenza, la maschera rumorosa viene automaticamente soppressa dal sistema, garantendo zero falsi positivi in questo scenario nonostante gli artefatti visivi delle CAM.

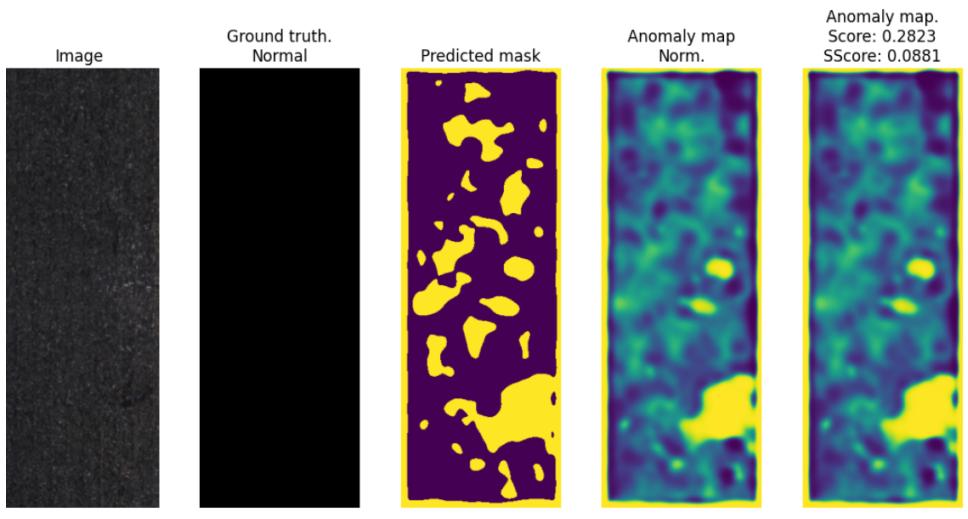


Figura 20: Risultati del metodo proposto su campioni normali, si evidenzia un bassissimo SScore

Pertanto, la presenza di maschere rumorose sui campioni negativi è un effetto collaterale puramente visivo derivante dalla normalizzazione delle mappe di attivazione, che non compromette l'affidabilità operativa del sistema; come dimostrato, il basso valore dello score di classificazione (SScore) agisce infatti come un efficace meccanismo di soppressione (gating), garantendo che tali artefatti non generino falsi positivi in fase di ispezione.

7.10 Proposte di Risoluzione per il Basso AP-loc

Per ovviare a queste basse performance nella localizzazione precisa dei difetti, pur mantenendo i benefici della supervisione debole, si potrebbe pensare alle seguenti strategie:

- **Mixed Supervision (Supervisione Mista):** Come suggerito nelle prospettive future dagli autori originali, si potrebbe adottare un regime ibrido: utilizzare annotazioni pixel-wise (maschere) solo per una piccolissima percentuale di dati (es. 5-10%) per "insegnare" alla rete a delineare i bordi tramite L_{seg} , lasciando il resto del dataset etichettato solo a livello immagine. Questo permetterebbe di recuperare la precisione spaziale con un costo di annotazione minimo.
- **Refinement Iterativo (Pseudo-Labeling):** Le CAM attuali, sebbene grossolane, possono essere utilizzate come "pseudo-maschere" iniziali. Si potrebbero raffinare i bordi del "blob" e utilizzare queste nuove maschere raffinate per ri-addestrare il modello in modo iterativo, forzandolo a convergere verso forme più precise.

Riferimenti bibliografici

- [1] Lars Heckler-Kram, Jan-Hendrik Neudeck, Ulla Scheler, Rebecca König, and Carsten Steger. The mvtec ad 2 dataset: Advanced scenarios for unsupervised anomaly detection, 2025.
- [2] Zhikang Liu, Yiming Zhou, Yuansheng Xu, and Zilei Wang. Simplenet: A simple network for image anomaly detection and localization, 2023.
- [3] Blaž Rolih, Matic Fučka, and Danijel Skočaj. Supersimplenet: Unifying unsupervised and supervised learning for fast and reliable surface defect detection, 2024.
- [4] Yanshu Wang, Xichen Xu, Jiaqi Liu, Xiaoning Lei, Guoyang Xie, Guannan Jiang, and Zhichao Lu. A survey on industrial anomalies synthesis, 2025.