

ResQPet

Sistema di Identificazione Automatizzata
dello Stato di Abbandono nei Cani

Progetto di Fondamenti di Intelligenza Artificiale

Università degli Studi di Salerno
Dipartimento di Informatica

Autore	Matricola
D'Alfonso Vittorio	0512120569
D'Elia Michele	0512120239
De Simone Mario	0512119765
Del Gaizo Gianluca	0512120119
Forgione Alessio	0512119915

Anno Accademico 2024/2025

Sommario

ResQPet è un sistema di computer vision progettato per identificare automaticamente lo stato di abbandono nei cani attraverso l'analisi di immagini e video. Il sistema utilizza un'architettura ensemble che combina multiple fonti di informazione: presenza di collare, condizioni cutanee, postura corporea e razza. Il contributo principale di questo lavoro è l'introduzione di un approccio di **weak supervision** per il classificatore di postura, che sfrutta l'origine dei dataset (cani randagi vs padronali) come supervisione implicita, eliminando la necessità di annotazione manuale. Il sistema produce uno *Stray Index* normalizzato in [0, 1] che indica la probabilità che un cane sia randagio. L'implementazione include una GUI web-based che simula un sistema di videosorveglianza CCTV per il monitoraggio in tempo reale.

Parole chiave: Computer Vision, Deep Learning, Object Detection, Pose Estimation, Weak Supervision, YOLO, Ensemble Learning

Indice

1 Introduzione	7
1.1 Contesto e Motivazione	7
1.2 Obiettivi del Progetto	7
1.3 Contributi Principali	7
2 Lavori Correlati	8
2.1 Object Detection per Animali	8
2.2 Pose Estimation	8
2.3 Weak Supervision	8
3 Architettura del Sistema	8
3.1 Overview della Pipeline	8
3.2 Flusso dei Dati	9
3.3 Moduli di Classificazione	10
3.3.1 Collar Detector	10
3.3.2 Skin Classifier	10
3.3.3 Pose Classifier	10
3.3.4 Breed Classifier	10
3.4 Sistema di Fusione	10
4 YOLO11 Dog-Pose Backbone	11
4.1 Ruolo nel Sistema	11
4.2 Architettura YOLO11	11
4.2.1 Backbone: CSPDarknet	11
4.2.2 Neck: PANet (Path Aggregation Network)	12
4.2.3 Head: Detection + Pose	12
4.3 Keypoints Anatomici (24 Punti)	12
4.4 Dataset Dog-Pose	12
4.5 Training	13
4.5.1 Configurazione v2 (Attuale)	13
4.5.2 Data Augmentation	13
4.6 Output del Modello	14
4.7 Normalizzazione Keypoints	14
4.8 Risultati v2	14
5 Collar Detector	16
5.1 Ruolo nel Sistema	16
5.2 Architettura	16
5.2.1 Classi di Detection	16
5.2.2 Logica di Output	16
5.3 Evoluzione del Dataset: Dal Problema alla Soluzione	17
5.3.1 Fase 1: Dataset Roboflow (v1)	17
5.3.2 Soluzione: Piattaforma di Labeling Custom	17
5.3.3 Fase 2: Dataset Merged (v2)	18
5.4 Training v2	19
5.4.1 Hardware	19

5.4.2	Configurazione Training	19
5.4.3	Data Augmentation	19
5.5	Risultati e Confronto	20
5.5.1	Validazione su Annotazioni Umane	20
5.6	Lezioni Apprese	22
6	Skin Disease Classifier	23
6.1	Ruolo nel Sistema	23
6.2	Architettura	23
6.2.1	Struttura Completa	24
6.2.2	Diagramma Architettura	24
6.3	Dataset	24
6.3.1	Classi Patologiche	25
6.3.2	Calcolo $P(\text{disease})$	25
6.4	Data Augmentation	26
6.5	Training	26
6.5.1	Class Weights	26
6.6	Metriche	27
7	Stray Pose Classifier	27
7.1	Contributo Originale: Weak Supervision	27
7.2	Motivazione	27
7.3	Approccio Weak Supervision	28
7.3.1	Assunzione Fondamentale	28
7.3.2	Schema di Labeling	28
7.3.3	Pipeline di Estrazione	28
7.4	Architettura MLP	28
7.4.1	Dettaglio Architettura	29
7.4.2	Dimensioni Tensori	30
7.5	Feature Input: Keypoints Normalizzati	30
7.6	Indicatori Comportamentali	30
7.7	Dataset	31
7.8	Training	31
7.9	Vantaggi dell'Approccio	31
7.10	Limitazioni	31
7.11	Metriche	32
7.12	Analisi dei Risultati e Discussione	32
7.12.1	Decisione: Mantenimento del Componente	33
8	Breed Classifier	34
8.1	Ruolo nel Sistema	34
8.2	Architettura	34
8.2.1	Struttura Completa	36
8.3	Macro-Categorie Razze	36
8.4	Breed Priors	37
8.4.1	Calcolo $P(\text{stray}—\text{breed})$	37
8.5	Dataset	38
8.6	Data Augmentation	38
8.7	Training	38

8.8	Metriche	39
9	Sistema di Fusione	39
9.1	Obiettivo	39
9.2	Componenti della Fusione	40
9.3	Formula di Fusione	40
9.3.1	Implementazione	40
9.4	Giustificazione dei Pesi	41
9.4.1	Nota sulla Reliability dei Componenti	41
9.5	Classificazione Finale	42
9.5.1	Implementazione Classificazione	42
9.6	Analisi di Sensibilità	43
9.7	Esempio di Calcolo	43
9.8	Gestione Valori Mancanti	43
10	Risultati Sperimentali	44
10.1	Setup Sperimentale	44
10.1.1	Hardware	44
10.1.2	Software	44
10.2	Metriche per Modello	44
10.2.1	Backbone YOLO11 Dog-Pose (v2)	44
10.2.2	Collar Detector (v2)	44
10.2.3	Skin Classifier	45
10.2.4	Pose Classifier	46
10.2.5	Breed Classifier	46
10.3	Performance Sistema Completo	47
10.3.1	Latenza End-to-End	47
10.3.2	Throughput	47
10.4	Analisi Qualitativa	48
10.4.1	Casi di Successo	48
10.4.2	Casi di Errore	48
10.5	Riepilogo Risultati	48
11	Explainable AI	48
11.1	Motivazione	48
11.2	Tecniche Utilizzate	49
11.3	Breed Classifier: GradCAM++	49
11.4	Collar Detector: Analisi Critica	50
11.4.1	Metodologia	50
11.4.2	Finding Critico	50
11.4.3	Analisi GradCAM Class-Specific	51
11.4.4	Implicazioni	52
11.5	Backbone YOLO11: Keypoint Reliability	52
11.5.1	Analisi per Gruppo Anatomico	52
11.6	Pose Classifier: SHAP Analysis	53
11.6.1	Features più Influenti	53
11.7	Riepilogo e Raccomandazioni	54

12 Implementazione	54
12.1 Stack Tecnologico	54
12.2 Backend Flask	54
12.3 Frontend React	55
13 Il Percorso di Sviluppo	55
13.1 Fase 1: La Sfida dei Dati	55
13.2 Fase 2: L'Intuizione della Weak Supervision	55
13.3 Fase 3: Integrazione e Bilanciamento	56
13.4 Lezioni Apprese	56
14 Discussione	57
14.1 Punti di Forza	57
14.2 Limitazioni	57
14.3 Sviluppi Futuri	57
15 Conclusioni	57
A Appendice A: Dettagli Dataset	59
B Appendice B: Hyperparameters	59
B.1 Backbone YOLO11 Dog-Pose v2	59
B.2 Collar Detector v2	59
C Appendice C: Guida Installazione	59

1 Introduzione

Questo progetto è stato sviluppato nell'ambito del corso di Fondamenti di Intelligenza Artificiale [6] presso l'Università degli Studi di Salerno.

1.1 Contesto e Motivazione

Il fenomeno del randagismo rappresenta una problematica rilevante sia dal punto di vista del benessere animale che della sicurezza pubblica. In Italia, secondo i dati ENPA, sono presenti oltre 500.000 cani randagi, con un costo sociale ed economico significativo per la loro gestione.

I sistemi di videosorveglianza sono sempre più diffusi nelle aree urbane, ma l'identificazione manuale di cani potenzialmente randagi richiede risorse umane considerevoli. Un sistema automatizzato di riconoscimento potrebbe:

- Ridurre i tempi di intervento per il recupero di animali in difficoltà
- Ottimizzare le risorse delle autorità competenti
- Facilitare il ricongiungimento di cani smarriti con i proprietari
- Monitorare le aree a maggior rischio di abbandono

1.2 Obiettivi del Progetto

Gli obiettivi principali di questo progetto sono:

1. Sviluppare un sistema di classificazione multi-modale per identificare lo stato di abbandono dei cani
2. Implementare un approccio di weak supervision per la classificazione della postura
3. Creare un'interfaccia utente che simuli un sistema CCTV per il monitoraggio real-time
4. Validare l'approccio con metriche quantitative appropriate

1.3 Contributi Principali

I contributi originali di questo lavoro includono:

1. **Architettura Ensemble Multi-modale:** Combinazione di quattro classificatori specializzati con fusione pesata
2. **Weak Supervision per Pose Classification:** Utilizzo dell'origine del dataset come supervisione implicita, evitando annotazioni manuali costose
3. **Stray Index:** Metrica unificata per quantificare la probabilità di abbandono
4. **Sistema CCTV Simulato:** Interfaccia web per dimostrazione e testing
5. **Analisi Explainable AI:** Validazione dei modelli con tecniche XAI e identificazione di dataset bias

2 Lavori Correlati

2.1 Object Detection per Animali

La detection di animali è un problema ben studiato nel campo della computer vision. I modelli della famiglia YOLO [8] hanno dimostrato eccellenti performance per la detection in tempo reale. In particolare, YOLO11 [9] introduce il supporto nativo per pose estimation, permettendo l'estrazione di keypoints anatomici.

2.2 Pose Estimation

La stima della postura negli animali presenta sfide uniche rispetto agli esseri umani, principalmente a causa della maggiore variabilità anatomica tra le specie [1]. Recenti lavori hanno adattato tecniche di human pose estimation per gli animali quadrupedi [4].

2.3 Weak Supervision

Il paradigma della weak supervision [7] permette di addestrare modelli con supervisione indiretta o rumorosa. Nel nostro caso, sfruttiamo la provenienza delle immagini (dataset di cani randagi vs padronali) come forma di supervisione implicita.

3 Architettura del Sistema

3.1 Overview della Pipeline

Il sistema ResQPet implementa un'architettura a pipeline multi-stadio per l'identificazione dello stato di abbandono nei cani. La pipeline si compone di tre fasi principali:

1. **Detection e Pose Estimation:** Un modello YOLO11 specializzato rileva i cani nel frame ed estrae 24 keypoints anatomici
2. **Classificazione Parallela:** Quattro classificatori indipendenti analizzano aspetti diversi del cane rilevato
3. **Fusione Pesata:** Le probabilità dei classificatori vengono combinate in uno *Stray Index* finale

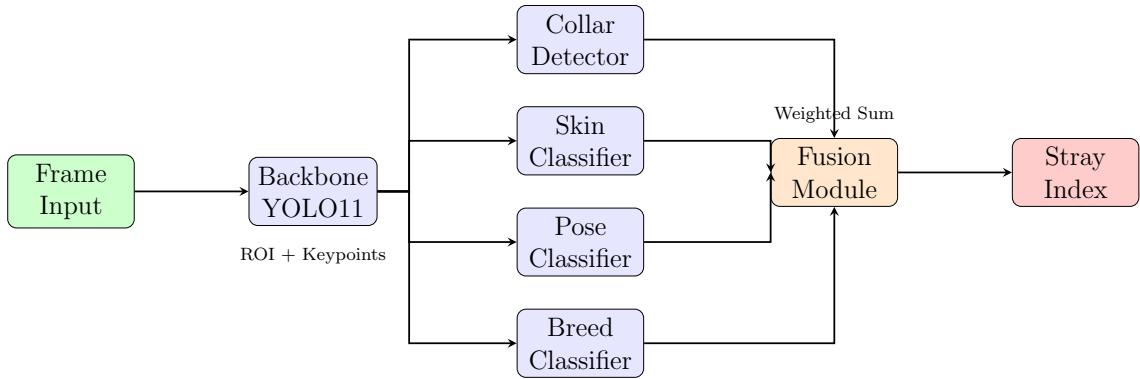


Figura 1: Architettura complessiva del sistema ResQPet. Il backbone YOLO11 estrae bounding box e keypoints, che vengono poi elaborati da quattro classificatori specializzati. Le probabilità risultanti sono combinate attraverso fusione pesata per produrre lo Stray Index finale.

3.2 Flusso dei Dati

Il flusso di elaborazione procede come segue:

1. **Input:** Frame video/immagine dalla sorgente CCTV
2. **Backbone:** YOLO11 Dog-Pose rileva ogni cane e produce:
 - Bounding box $[x_1, y_1, x_2, y_2]$
 - 24 keypoints anatomici con confidence
 - ROI (Region of Interest) croppata
3. **Pre-processing:**
 - Normalizzazione keypoints rispetto alla bbox
 - Resize ROI per i classificatori (224×224)
4. **Classificazione Parallela:** Per ogni cane rilevato:
 - Collar Detector $\rightarrow P(\text{no_collar})$
 - Skin Classifier $\rightarrow P(\text{disease})$
 - Pose Classifier $\rightarrow P(\text{stray_pose})$
 - Breed Classifier $\rightarrow P(\text{stray|breed})$
5. **Fusione:** Combinazione pesata delle quattro probabilità
6. **Output:** Stray Index $\in [0, 1]$ con classificazione

3.3 Moduli di Classificazione

I quattro classificatori sono progettati per catturare aspetti complementari che indicano lo stato di abbandono:

Modulo	Architettura	Input	Peso
Collar Detector	YOLOv8n	ROI 640×640	35%
Skin Classifier	ResNet50	ROI 224×224	20%
Pose Classifier	MLP	72 features	25%
Breed Classifier	EfficientNet-B0	ROI 224×224	20%

Tabella 1: Riepilogo dei moduli di classificazione con relative architetture, input e pesi nella fusione.

3.3.1 Collar Detector

Rileva la presenza di collare, pettorina o guinzaglio. L'assenza di accessori è un forte indicatore di cane randagio, motivo per cui questo modulo ha il peso maggiore (35%).

3.3.2 Skin Classifier

Identifica patologie cutanee indicative di trascuratezza o abbandono prolungato. Le malattie della pelle non curate suggeriscono mancanza di cure veterinarie.

3.3.3 Pose Classifier

Analizza la postura del cane basandosi sui keypoints estratti. Utilizza un approccio di **weak supervision** (descritto in dettaglio nella Sezione 7) per identificare posture tipiche di cani randagi (coda tra le gambe, testa bassa, postura difensiva).

3.3.4 Breed Classifier

Identifica la razza del cane per applicare prior statistici. Alcune razze sono statisticamente più rappresentate nei canili italiani (es. Pitbull, meticci) rispetto ad altre (es. Retriever, razze toy).

3.4 Sistema di Fusione

Le probabilità dei quattro classificatori vengono combinate attraverso una media pesata:

$$\text{Stray Index} = w_c \cdot P_c + w_s \cdot P_s + w_p \cdot P_p + w_b \cdot P_b \quad (1)$$

dove:

- P_c = probabilità di assenza collare
- P_s = probabilità di malattia cutanea
- P_p = probabilità di postura stray-like
- P_b = prior di abbandono dato la razza

- $w_c = 0.35, w_s = 0.20, w_p = 0.25, w_b = 0.20$

Lo Stray Index risultante viene classificato in tre categorie:

Range	Classificazione	Colore
[0.0, 0.3)	Padronale	Verde
[0.3, 0.7)	Possibile Smarrito	Giallo
[0.7, 1.0]	Probabile Randagio	Rosso

Tabella 2: Soglie di classificazione dello Stray Index.

4 YOLO11 Dog-Pose Backbone

4.1 Ruolo nel Sistema

Il backbone rappresenta il primo stadio della pipeline e ha il compito critico di:

- **Rilevare** tutti i cani presenti nel frame
- **Localizzare** ciascun cane con una bounding box precisa
- **Estrarre** 24 keypoints anatomici per l’analisi della postura

A differenza del modello YOLO11-pose standard (trainato per pose estimation umana con 17 keypoints), il nostro backbone è specializzato per la detection di cani con 24 punti anatomici specifici.

4.2 Architettura YOLO11

YOLO11 (You Only Look Once, versione 11) è l’ultima evoluzione della famiglia YOLO, sviluppata da Ultralytics. L’architettura si compone di tre blocchi principali:

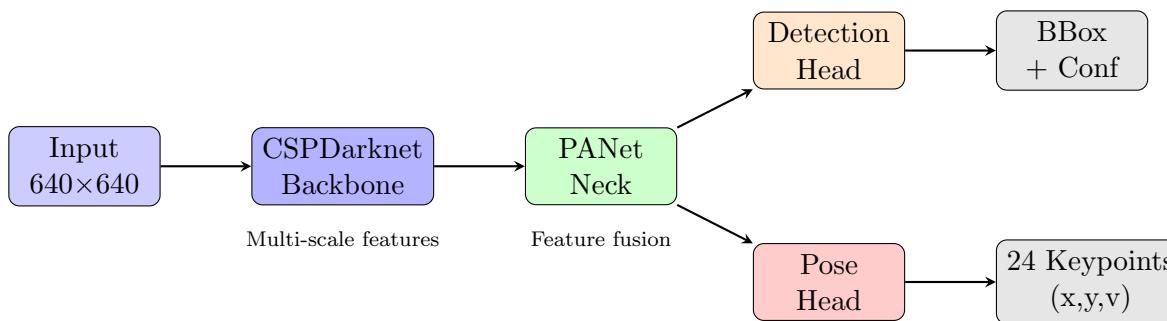


Figura 2: Architettura semplificata di YOLO11 per pose estimation. Il backbone CSPDarknet estrae features multi-scala, il neck PANet fonde le informazioni, e le head producono detection e keypoints.

4.2.1 Backbone: CSPDarknet

Estrae features gerarchiche dall’immagine di input attraverso una serie di blocchi convoluzionali con connessioni cross-stage partial (CSP). Produce feature maps a multiple risoluzioni.

4.2.2 Neck: PANet (Path Aggregation Network)

Fonde le feature maps provenienti da diversi livelli del backbone, permettendo al modello di rilevare oggetti di dimensioni variabili. Utilizza connessioni top-down e bottom-up.

4.2.3 Head: Detection + Pose

Due head parallele producono:

- **Detection Head:** Bounding boxes con classe e confidence
- **Pose Head:** Keypoints con coordinate (x, y) e visibility score

4.3 Keypoints Anatomici (24 Punti)

Il modello estrae 24 keypoints specifici per l'anatomia canina, suddivisi in regioni corporee:

ID	Nome	ID	Nome
0	nose	12	right_front_knee
1	left_eye	13	left_front_paw
2	right_eye	14	right_front_paw
3	left_ear_base	15	left_back_elbow
4	right_ear_base	16	right_back_elbow
5	left_ear_tip	17	left_back_knee
6	right_ear_tip	18	right_back_knee
7	throat	19	left_back_paw
8	withers (garrese)	20	right_back_paw
9	left_front_elbow	21	tail_start
10	right_front_elbow	22	tail_end
11	left_front_knee	23	chin

Tabella 3: Elenco completo dei 24 keypoints anatomici estratti dal modello Dog-Pose.

Ogni keypoint è rappresentato da una tripla (x, y, v) dove:

- x, y : coordinate in pixel nel frame originale
- $v \in [0, 1]$: visibility score ($1 =$ visibile, $0 =$ occluso/non rilevato)

4.4 Dataset Dog-Pose

Il modello viene trainato sul dataset Dog-Pose fornito da Ultralytics, specificamente progettato per pose estimation canina.

Proprietà	Valore
Immagini Training	6,773
Immagini Test	1,703
Classi	1 (Dog)
Keypoints per istanza	24
Formato annotazioni	YOLO Pose (txt)
Fonte	Ultralytics Hub

Tabella 4: Statistiche del dataset Dog-Pose utilizzato per il training del backbone.

4.5 Training

Il training utilizza transfer learning partendo dal modello `yolo11n-pose.pt` pre-trainato su human pose estimation (17 keypoints umani). Il modello viene adattato alla struttura anatomica canina (24 keypoints).

4.5.1 Configurazione v2 (Attuale)

Parametro	Valore
Modello base	<code>yolo11n-pose.pt</code>
Epochs	150
Image size	640×640
Batch size	16
Optimizer	AdamW
Learning rate iniziale	0.001
Learning rate finale	0.01
Weight decay	0.0005
Early stopping patience	20 epochs
Device	CUDA (single GPU)

Tabella 5: Hyperparameters per il training del backbone Dog-Pose v2.

4.5.2 Data Augmentation

Durante il training vengono applicate le seguenti trasformazioni:

- **Mosaic:** Combina 4 immagini in una sola (prob. 1.0)
- **MixUp:** Sovrapposizione di due immagini (prob. 0.1)
- **Rotazione:** $\pm 10^\circ$
- **Traslazione:** $\pm 10\%$
- **Scaling:** $0.5 \times - 1.5 \times$
- **Horizontal Flip:** 50%
- **HSV Augmentation:** Variazioni di hue, saturation, value

4.6 Output del Modello

Per ogni frame processato, il backbone ritorna una lista di detection, ciascuna contenente:

```

1  detection = {
2      'bbox': [x1, y1, x2, y2],      # Bounding box in pixel
3      'confidence': 0.92,          # Confidence detection
4      'keypoints': np.array(24, 3), # 24 keypoints (x, y,
6          visibility)
5      'roi': np.ndarray,           # Immagine croppata
6      'class_id': 0               # Sempre 0 (dog)
7 }
```

Listing 1: Struttura output del backbone per ogni cane rilevato

4.7 Normalizzazione Keypoints

Prima di essere passati al Pose Classifier, i keypoints vengono normalizzati rispetto alla bounding box per renderli invarianti a scala e posizione:

```

1 def normalize_keypoints(keypoints, bbox):
2     x1, y1, x2, y2 = bbox
3     w, h = x2 - x1, y2 - y1
4
5     normalized = keypoints.copy()
6     normalized[:, 0] = (keypoints[:, 0] - x1) / w # x in [0, 1]
7     normalized[:, 1] = (keypoints[:, 1] - y1) / h # y in [0, 1]
8     # visibility rimane invariato
9
10    return normalized # Shape: (24, 3)
```

Listing 2: Normalizzazione keypoints rispetto alla bounding box

4.8 Risultati v2

Il training v2 con 150 epochs ha prodotto risultati eccellenti, superando significativamente i target prefissati.

Metrica	Target	Ottenuto (v2)
mAP@0.5	> 0.85	0.987
Precision	> 0.90	0.969
Recall	> 0.90	0.977
Inference time (GPU)	< 20ms	~ 8ms

Tabella 6: Metriche del backbone Dog-Pose v2 (150 epochs). Il modello raggiunge una mAP@0.5 del 98.7%.

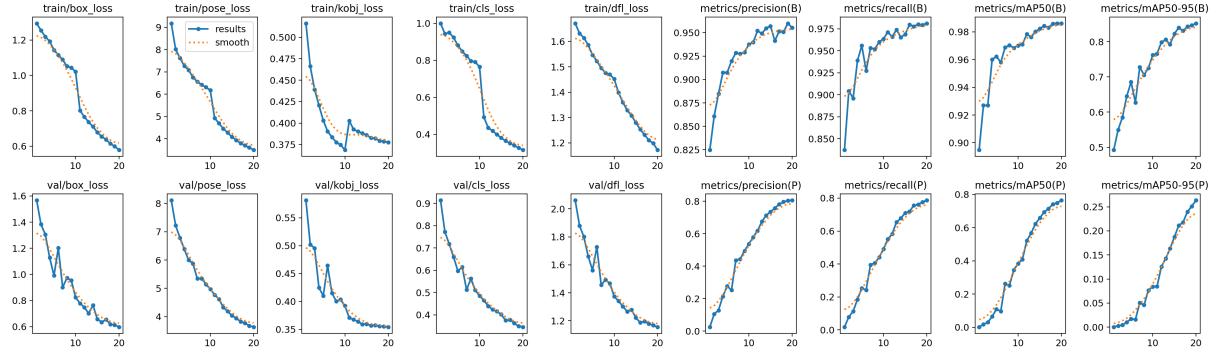


Figura 3: Curve di training del backbone Dog-Pose v2. In alto: loss di training (box, pose, keypoint object, classification, DFL). In basso: loss di validation e metriche (precision, recall, mAP50, mAP50-95). Si nota la convergenza stabile senza overfitting.



Figura 4: Esempi di detection del backbone Dog-Pose su immagini di validazione. Ogni cane è identificato con bounding box (blu), confidence score e 24 keypoints anatomici sovrapposti.

Analisi: L'elevata precision (96.9%) indica che quasi tutte le detection sono corrette, mentre l'alto recall (97.7%) significa che il modello rileva quasi tutti i cani presenti nel frame. Questo è fondamentale per il sistema ResQPet, dove ogni cane deve essere analizzato.

5 Collar Detector

5.1 Ruolo nel Sistema

Il Collar Detector ha il compito di rilevare la presenza o assenza di collare, pettorina o guinzaglio sul cane. Questo è considerato l'**indicatore più forte** di possesso, motivo per cui ha il peso maggiore nella fusione (35%).

- **Input:** ROI del cane (immagine croppata dalla detection)
- **Output:** $P(\text{no_collar}) \in [0, 1]$
- **Peso Fusion:** 35%

Un cane con collare è quasi certamente padronale, mentre l'assenza di accessori suggerisce (ma non conferma) un possibile stato di abbandono.

5.2 Architettura

Il Collar Detector utilizza YOLOv8n (nano) fine-tuned per object detection su due classi:

5.2.1 Classi di Detection

Classe ID	Nome
0	Dog-with-Leash (con collare/guinzaglio)
1	Dog-without-Leash (senza accessori)

Tabella 7: Classi del Collar Detector.

5.2.2 Logica di Output

```

1 def predict_collar(roi):
2     results = collar_detector(roi)
3
4     for detection in results:
5         cls = detection.class_id
6         conf = detection.confidence
7
8         if cls == 0: # Dog-with-Leash
9             return 1.0 - conf # Bassa P(no_collar)
10        elif cls == 1: # Dog-without-Leash
11            return conf # Alta P(no_collar)
12
13    return 0.7 # Default: probabilmente senza (nessuna detection
14

```

Listing 3: Calcolo della probabilità di assenza collare

5.3 Evoluzione del Dataset: Dal Problema alla Soluzione

Lo sviluppo del Collar Detector ha attraversato due fasi distinte, evidenziando l'importanza della quantità e qualità dei dati nel deep learning.

5.3.1 Fase 1: Dataset Roboflow (v1)

Il primo training è stato effettuato sul dataset “Dog with Leash” disponibile su Roboflow.

Proprietà	Valore
Nome	Dog with Leash
Fonte	Roboflow
Totale immagini	152
Training set	106 (69.7%)
Validation set	30 (19.7%)
Test set	16 (10.5%)

Tabella 8: Statistiche del dataset Roboflow (v1).

Limitazioni riscontrate:

- **Dimensione insufficiente:** Solo 152 immagini totali
- **Variabilità limitata:** Poche condizioni di illuminazione e sfondo
- **Performance scadenti:** mAP@0.5 = 51%, insufficiente per produzione

Metrica	Target	v1 (Roboflow)
mAP@0.5	> 0.75	0.51
Precision	> 0.70	0.48
Recall	> 0.70	0.54

Tabella 9: Metriche v1: performance insufficienti dovute al dataset limitato.

5.3.2 Soluzione: Piattaforma di Labeling Custom

Per superare le limitazioni dei dati disponibili, è stata sviluppata una **piattaforma web di labeling** dedicata, permettendo la creazione collaborativa di un dataset di dimensioni adeguate.

Piattaforma di Labeling ResQPet

Stack: Flask + SQLAlchemy + Bootstrap

Features:

- Multi-user con gestione ruoli
- Pre-labeling automatico con modello esistente
- Interfaccia di revisione con skip/approve/reject
- Export JSON per analisi
- Sistema di merge annotazioni multi-utente
- Statistiche real-time per utente

Figura 5: Architettura della piattaforma di labeling sviluppata.

Workflow di annotazione:

1. **Indicizzazione:** Importazione immagini da Stanford Dogs e altri dataset
2. **Pre-labeling:** Il modello v1 genera predizioni iniziali
3. **Revisione umana:** Annotatori validano/correggono le predizioni
4. **Export:** Annotazioni esportate in formato JSON
5. **Merge:** Script automatico unisce contributi multi-utente
6. **Conversione:** Generazione dataset YOLO con split train/val

5.3.3 Fase 2: Dataset Merged (v2)

Attraverso la piattaforma, sono state raccolte annotazioni da 2 utenti su oltre 7,500 immagini.

Proprietà	Valore
Fonte	Piattaforma Labeling ResQPet
Annotatori	2 utenti
Totale immagini	7,576
Training set	6,061 (80%)
Validation set	1,516 (20%)
Immagini originali	Stanford Dogs + Custom

Tabella 10: Statistiche del dataset merged (v2).

Miglioramento quantitativo:

- **50× più dati:** Da 152 a 7,576 immagini
- **Maggiore variabilità:** Razze, pose, sfondi diversificati
- **Annotazioni verificate:** Revisione umana delle predizioni

5.4 Training v2

Il retraining con il dataset merged ha utilizzato un'infrastruttura multi-GPU per accelerare l'addestramento.

5.4.1 Hardware

Componente	Specifiche
GPU	2× NVIDIA RTX 5090 (32GB VRAM ciascuna)
Configurazione	Multi-GPU con DataParallel
Precision	Mixed Precision (FP16)

Tabella 11: Hardware utilizzato per il training v2.

5.4.2 Configurazione Training

Parametro	v1 (Roboflow)	v2 (Merged)
Modello base	YOLOv8n	YOLOv8n
Epochs	150	100
Image size	640 × 640	640 × 640
Batch size	8	128 (64/GPU)
Optimizer	AdamW	AdamW
Learning rate	0.001	0.001
Early stopping	30 epochs	20 epochs
Training time	~1 ora	~20 minuti

Tabella 12: Confronto configurazione training v1 vs v2.

5.4.3 Data Augmentation

Con un dataset più ampio, l'augmentation è stato moderato rispetto alla versione precedente:

Trasformazione	v1	v2
HSV Saturation	0.8	0.7
HSV Value	0.5	0.4
Rotazione	$\pm 20^\circ$	$\pm 15^\circ$
Scaling	0.4-1.6×	0.5-1.5×
MixUp	20%	10%
Mosaic	100%	100%

Tabella 13: Confronto augmentation v1 vs v2. Con più dati, l’augmentation aggressivo diventa meno necessario.

5.5 Risultati e Confronto

Il retraining ha prodotto un miglioramento significativo delle metriche.

Metrica	Target	v1 (152 img)	v2 (7,576 img)
mAP@0.5	> 0.75	0.51	0.853
mAP@0.5:0.95	> 0.50	0.33	0.722
Precision	> 0.70	0.48	0.741
Recall	> 0.70	0.54	0.854

Tabella 14: Confronto metriche v1 vs v2. Il miglioramento è del 67% sul mAP@0.5.

5.5.1 Validazione su Annotazioni Umane

Per validare ulteriormente il modello, è stata calcolata l’accuratezza rispetto alle etichette umane:

Metrica	Valore
Immagini con label umano	133
Accuracy (Model vs Human)	86.5%
Matches	115
Mismatches	18

Tabella 15: Confronto predizioni modello vs annotazioni umane.

	Pred: WITH	Pred: WITHOUT
Actual: WITH	56 (TP)	8 (FN)
Actual: WITHOUT	10 (FP)	59 (TN)

Tabella 16: Confusion matrix del modello v2 rispetto alle annotazioni umane.

Confusion Matrix (**WITH_COLLAR** detection):

- **Precision:** 84.8% (56/66)

- **Recall:** 87.5% (56/64)

- **F1 Score:** 86.2%

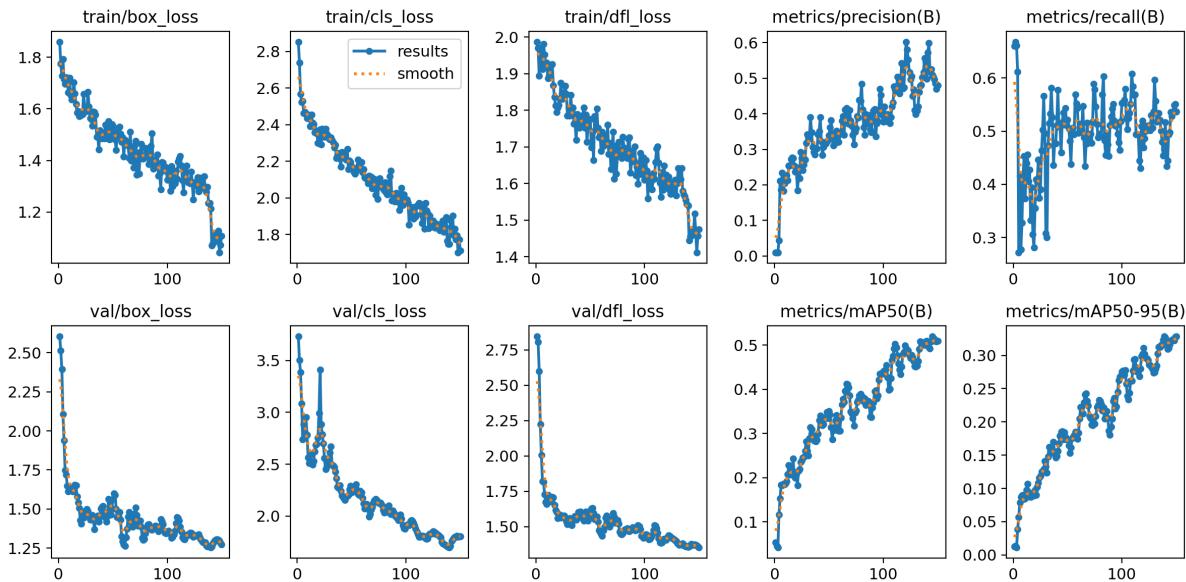


Figura 6: Curve di training del Collar Detector v2 su 200 epochs. In alto: loss di training (box, classification, DFL) e metriche precision/recall. In basso: loss di validation e mAP. Si nota la convergenza progressiva del mAP@0.5 verso 0.85.

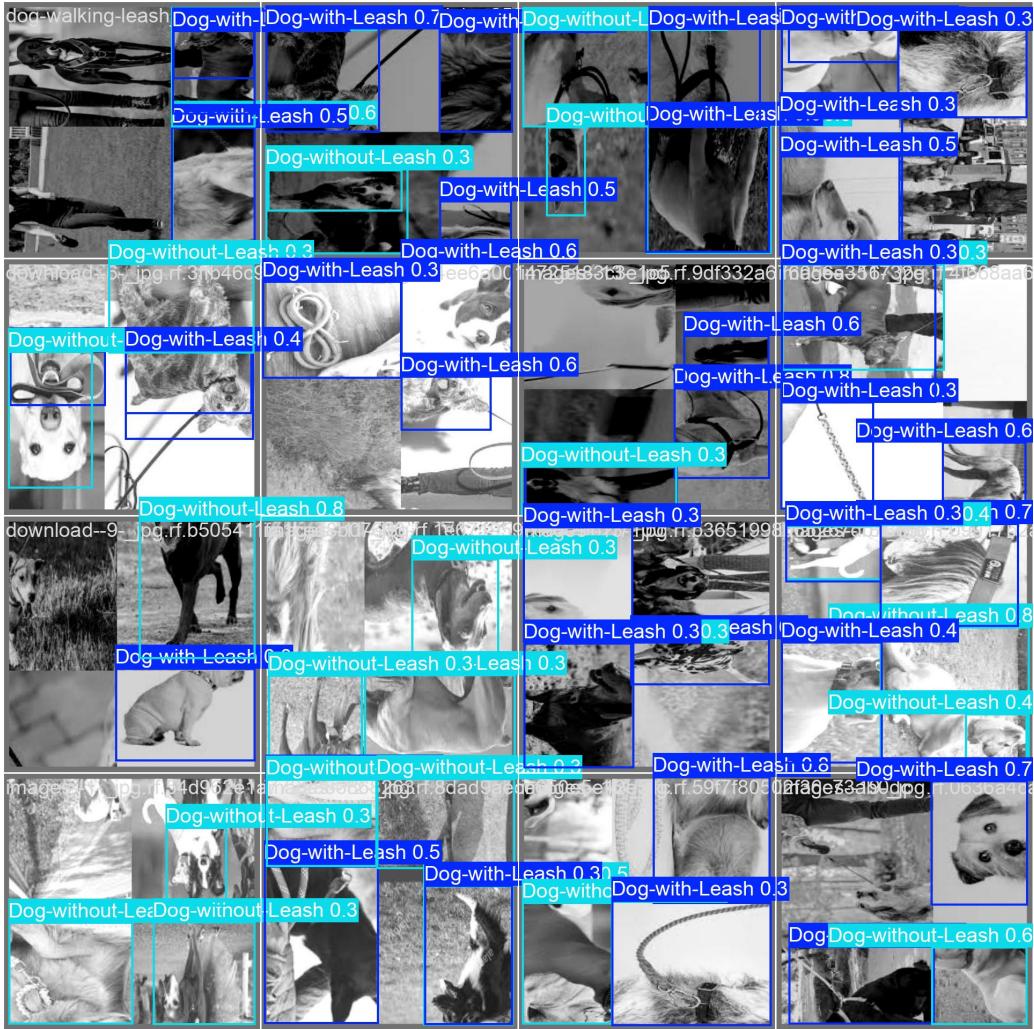


Figura 7: Esempi di detection del Collar Detector su immagini di validazione. Le classi “Dog-with-Leash” (con collare/guinzaglio) e “Dog-without-Leash” (senza accessori) vengono identificate con bounding box e confidence score.

5.6 Lezioni Apprese

Lo sviluppo del Collar Detector evidenzia principi fondamentali del deep learning:

- 1. Qualità > Complessità del modello:** YOLOv8n (stesso modello) con $50\times$ più dati ha migliorato del 67%
- 2. Human-in-the-loop:** La combinazione di pre-labeling automatico + revisione umana è efficiente e scalabile
- 3. Iterazione:** Il modello v1 (seppur scadente) ha accelerato la creazione del dataset v2 attraverso il pre-labeling
- 4. Multi-GPU scaling:** Batch size maggiore (128 vs 8) ha ridotto il training da 1h a 20min

6 Skin Disease Classifier

6.1 Ruolo nel Sistema

Lo Skin Classifier identifica patologie cutanee che possono indicare trascuratezza o abbandono prolungato. Un cane con malattie della pelle non curate suggerisce mancanza di cure veterinarie, tipica di animali randagi.

- **Input:** ROI del cane ridimensionata a 224×224 pixel
- **Output:** $P(\text{disease}) \in [0, 1]$
- **Peso Fusion:** 20%

6.2 Architettura

Il classificatore utilizza ResNet50 pre-trainato su ImageNet con un custom classification head.

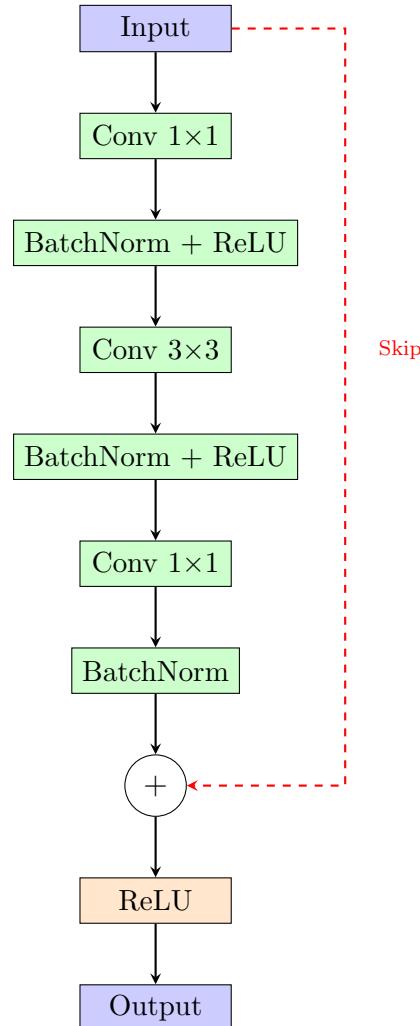


Figura 8: Blocco residuo di ResNet. La skip connection permette di trainare reti profonde evitando il problema del vanishing gradient.

6.2.1 Struttura Completa

1. **Backbone:** ResNet50 (senza ultimo layer FC)

- Pre-trained su ImageNet (1000 classi)
- Output: 2048 features

2. **Classification Head:**

- Dropout(0.3)
- Linear($2048 \rightarrow 256$)
- ReLU
- BatchNorm1d(256)
- Dropout(0.3)
- Linear($256 \rightarrow 6$)

6.2.2 Diagramma Architettura

```

1  class SkinClassifier(nn.Module):
2      def __init__(self, num_classes=6):
3          super().__init__()
4          # Backbone ResNet50
5          self.backbone = timm.create_model(
6              'resnet50',
7              pretrained=True,
8              num_classes=0  # Rimuove FC layer
9          )
10         # Custom head
11         self.classifier = nn.Sequential(
12             nn.Dropout(0.3),
13             nn.Linear(2048, 256),
14             nn.ReLU(),
15             nn.BatchNorm1d(256),
16             nn.Dropout(0.3),
17             nn.Linear(256, num_classes)
18         )
19
20     def forward(self, x):
21         features = self.backbone(x)  # (B, 2048)
22         return self.classifier(features)  # (B, 6)

```

Listing 4: Definizione architettura Skin Classifier

6.3 Dataset

Il modello è trainato sul dataset “Dog’s Skin Diseases” disponibile su Kaggle.

Proprietà	Valore
Nome	Dog's Skin Diseases
Fonte	Kaggle
Totale immagini	4,315
Training set	3,022 (70%)
Validation set	860 (20%)
Test set	433 (10%)
Classi	6

Tabella 17: Statistiche del dataset Skin Diseases.

6.3.1 Classi Patologiche

Classe	Descrizione	Peso Malattia
Healthy	Pelle sana	0.0
Dermatitis	Infiammazione generica	0.6
Fungal_infections	Infezioni fungine	0.7
Hypersensitivity	Allergie/ipersensibilità	0.4
Demodicosis	Acari Demodex	0.8
Ringworm	Tigna (tinea)	0.75

Tabella 18: Classi del dataset Skin Diseases con relativi pesi di gravità. Il peso indica quanto la patologia contribuisce al $P(\text{disease})$.

6.3.2 Calcolo $P(\text{disease})$

La probabilità finale di malattia è calcolata come somma pesata delle probabilità per classe:

```

1 DISEASE_WEIGHTS = {
2     'Healthy': 0.0,
3     'Dermatitis': 0.6,
4     'Fungal_infections': 0.7,
5     'Hypersensitivity': 0.4,
6     'Demodicosis': 0.8,
7     'Ringworm': 0.75
8 }
9
10 def compute_p_disease(class_probabilities):
11     p_disease = 0.0
12     for class_name, prob in class_probabilities.items():
13         p_disease += prob * DISEASE_WEIGHTS[class_name]
14     return p_disease

```

Listing 5: Calcolo della probabilità di malattia cutanea

6.4 Data Augmentation

Trasformazione	Parametri
Resize	256×256
RandomCrop	224×224
Horizontal Flip	50%
Vertical Flip	20%
Rotazione	$\pm 15^\circ$
ColorJitter	brightness=0.2, contrast=0.2
Normalize	ImageNet mean/std

Tabella 19: Data augmentation per il training dello Skin Classifier.

6.5 Training

Il training utilizza un approccio di fine-tuning progressivo:

1. **Fase 1:** Freeze backbone, train solo classification head (10 epochs)
2. **Fase 2:** Unfreeze ultimi 2 blocchi ResNet (20 epochs)
3. **Fase 3:** Fine-tuning completo con LR ridotto (20 epochs)

Parametro	Valore
Epochs totali	50
Batch size	32
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	10^{-5}
Loss	CrossEntropyLoss (class weights)
Scheduler	ReduceLROnPlateau
Early stopping	10 epochs

Tabella 20: Configurazione training dello Skin Classifier.

6.5.1 Class Weights

Per gestire lo sbilanciamento del dataset, utilizziamo pesi inversamente proporzionali alla frequenza:

$$w_i = \frac{\max_j(n_j)}{n_i} \quad (2)$$

dove n_i è il numero di campioni della classe i .

6.6 Metriche

Metrica	Target	Ottenuto
Accuracy	> 0.80	0.806
F1-Score (macro)	> 0.75	0.761
Precision (macro)	> 0.75	0.78
Recall (macro)	> 0.70	0.76

Tabella 21: Metriche di valutazione dello Skin Classifier.

7 Stray Pose Classifier

7.1 Contributo Originale: Weak Supervision

Il Pose Classifier rappresenta il **contributo originale** di questo lavoro. Invece di richiedere annotazioni manuali costose e soggettive per classificare le posture come “stray-like” o “owned-like”, proponiamo un approccio di **weak supervision** che sfrutta l’origine dei dati come supervisione implicita.

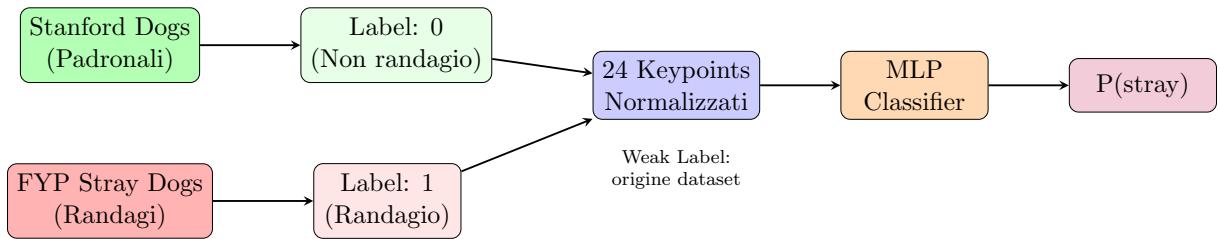


Figura 9: Schema dell’approccio weak supervision. I label sono derivati automaticamente dall’origine del dataset: keypoints da FYP Dataset (cani randagi) ricevono label=1, keypoints da Stanford Dogs (cani padronali) ricevono label=0.

7.2 Motivazione

L’annotazione manuale della postura presenta problemi significativi:

- **Soggettività:** Cosa definisce una “postura da randagio”?
- **Costo:** Annotare migliaia di pose richiede tempo e risorse
- **Inconsistenza:** Annotatori diversi producono label diversi
- **Scalabilità:** Difficile estendere a nuovi dataset

Il nostro approccio risolve questi problemi derivando i label automaticamente dalla provenienza delle immagini.

7.3 Approccio Weak Supervision

7.3.1 Assunzione Fondamentale

I cani fotografati in dataset di cani randagi (es. FYP Dataset) tendono ad avere posture diverse dai cani fotografati in contesti domestici/esposizioni (es. Stanford Dogs).

Questa assunzione si basa sul fatto che:

- I cani randagi mostrano spesso comportamenti difensivi, sottomessi o stressati
- I cani padronali in foto sono tipicamente rilassati, in posa, o in attività ludiche

7.3.2 Schema di Labeling

Dataset Origine	Label	Motivazione
FYP Dataset	1 (Stray)	Cani randagi di strada
Stanford Dogs	0 (Owned)	Cani in contesti domestici/show
Dog's Skin Diseases	0 (Owned)	Cani con proprietari (dal veterinario)

Tabella 22: Schema di assegnazione automatica dei label basato sull'origine del dataset.

7.3.3 Pipeline di Estrazione

1. Per ogni immagine nei dataset:
 - (a) Eseguire YOLO11 Dog-Pose per detection
 - (b) Filtrare detection con confidence > 0.7
 - (c) Estrarre keypoints e normalizzare rispetto a bbox
 - (d) Assegnare label basato su origine dataset
2. Bilanciare le classi (stesso numero di campioni)
3. Split train/val/test stratificato

7.4 Architettura MLP

Il classificatore utilizza un Multi-Layer Perceptron leggero, ottimale per input numerici strutturati.

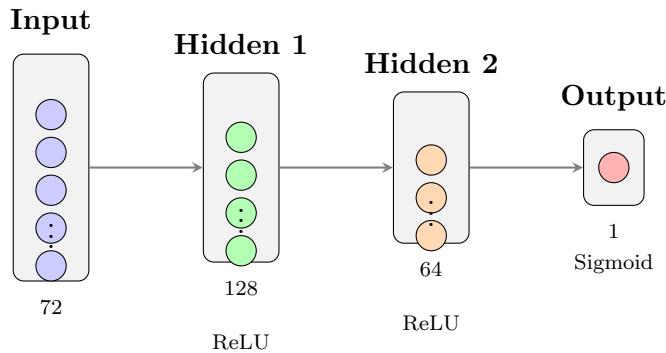


Figura 10: Architettura del Pose Classifier MLP. Input: 72 features (24 keypoints \times 3 valori). Output: probabilità di postura stray-like.

7.4.1 Dettaglio Architettura

```

1 class StrayPoseMLP(nn.Module):
2     def __init__(self, input_dim=72, hidden_dims=[128, 64],
3                  dropout=0.3):
4         super().__init__()
5
5         layers = []
6         prev_dim = input_dim # 72 = 24 keypoints * 3
7
8         for hidden_dim in hidden_dims:
9             layers.extend([
10                 nn.Linear(prev_dim, hidden_dim),
11                 nn.ReLU(),
12                 nn.BatchNorm1d(hidden_dim),
13                 nn.Dropout(dropout)
14             ])
15         prev_dim = hidden_dim
16
17     # Output layer
18     layers.append(nn.Linear(prev_dim, 1))
19     layers.append(nn.Sigmoid())
20
21     self.model = nn.Sequential(*layers)
22
23     def forward(self, x):
24         return self.model(x) # P(stray_pose)

```

Listing 6: Definizione architettura StrayPoseMLP

7.4.2 Dimensioni Tensori

Layer	Output Shape	Parametri
Input	(B, 72)	—
Linear + ReLU + BN + Dropout	(B, 128)	$72 \times 128 + 128 = 9,344$
Linear + ReLU + BN + Dropout	(B, 64)	$128 \times 64 + 64 = 8,256$
Linear + Sigmoid	(B, 1)	$64 \times 1 + 1 = 65$
Totale	—	~17,665

Tabella 23: Dimensioni dei tensori e numero di parametri per layer.

7.5 Feature Input: Keypoints Normalizzati

L'input al classificatore è un vettore di 72 features:

$$\mathbf{x} = [x_0, y_0, v_0, x_1, y_1, v_1, \dots, x_{23}, y_{23}, v_{23}] \quad (3)$$

dove per ogni keypoint i :

- $x_i \in [0, 1]$: coordinata x normalizzata rispetto a bbox
- $y_i \in [0, 1]$: coordinata y normalizzata rispetto a bbox
- $v_i \in [0, 1]$: visibility score

7.6 Indicatori Comportamentali

Il modello impara implicitamente a riconoscere pattern posturali associati a stress/abbandono:

Indicatore	Keypoints Coinvolti
Coda tra le gambe	tail_start, tail_end, back_paws
Testa bassa	nose, chin, throat vs withers
Orecchie appiattite	ear_base, ear_tip
Postura accucciata	rapporto altezza/larghezza corpo
Stance difensiva	distanza tra front_paws e back_paws
Asimmetria posturale	confronto lato sinistro/destro

Tabella 24: Indicatori comportamentali impliciti catturati dal modello attraverso i keypoints.

7.7 Dataset

Proprietà	Valore
Origine Stray (label=1)	FYP Dataset
Origine Owned (label=0)	Stanford Dogs + Skin Dataset
Campioni totali	15,132 (bilanciati)
Training	70%
Validation	15%
Test	15%

Tabella 25: Composizione del dataset per il Pose Classifier.

7.8 Training

Parametro	Valore
Epochs	100
Batch size	64
Optimizer	Adam
Learning rate	10^{-3}
Weight decay	10^{-4}
Loss	BCELoss
Scheduler	ReduceLROnPlateau
Early stopping	15 epochs

Tabella 26: Configurazione training del Pose Classifier.

7.9 Vantaggi dell’Approccio

1. **Nessuna annotazione manuale:** I label derivano automaticamente
2. **Scalabilità:** Facilmente estendibile con nuovi dataset
3. **Consistenza:** Eliminata la soggettività umana
4. **Dataset ampio:** Potenzialmente decine di migliaia di campioni
5. **Riproducibilità:** Processo completamente automatico

7.10 Limitazioni

- **Label noise:** Non tutti i cani del FYP mostrano posture “stray”
- **Overlap:** Alcune posture sono comuni a entrambe le categorie
- **Bias del dataset:** La qualità dipende dalla rappresentatività dei dataset

7.11 Metriche

Metrica	Target	Ottenuto
Accuracy	> 0.70	0.589
AUC-ROC	> 0.75	0.633

Tabella 27: Metriche di valutazione del Pose Classifier. I risultati sono inferiori ai target, come atteso per approcci weak-supervised.

7.12 Analisi dei Risultati e Discussione

I risultati ottenuti (Accuracy 0.589, AUC 0.633) sono inferiori ai target prefissati. Questo è un risultato **atteso** e documentato nella letteratura sulla weak supervision:

- **Rumore intrinseco nei label:** L’assunzione che l’origine del dataset determini la postura è debole. Un cane randagio può essere fotografato in pose rilassate, così come un cane padronale può mostrare stress.
- **Overlap delle distribuzioni:** Le posture di cani randagi e padronali si sovrappongono significativamente, limitando la separabilità delle classi.
- **Segnale comunque informativo:** Un AUC di 0.633 indica che il modello cattura un segnale, seppur debole, che può contribuire alla decisione finale quando combinato con gli altri classificatori.

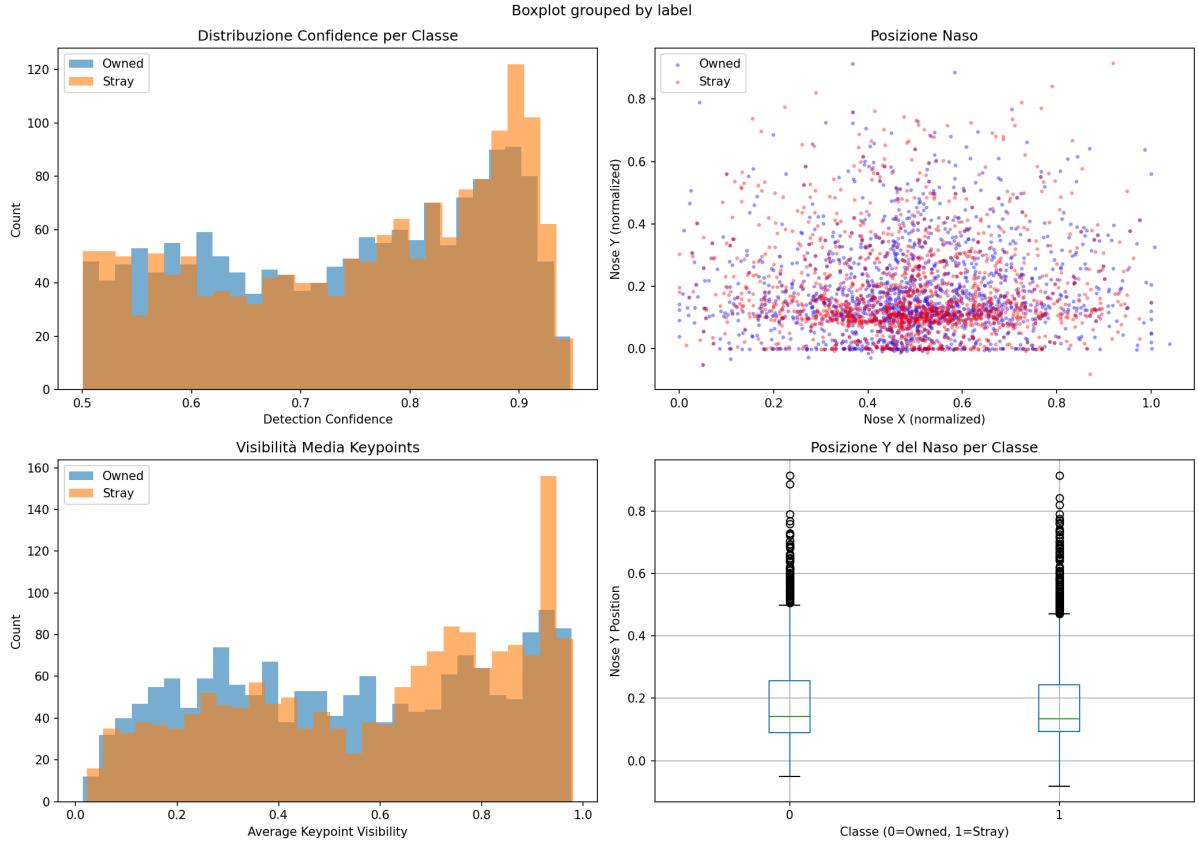


Figura 11: Analisi delle distribuzioni dei keypoints per classe (Owned vs Stray). In alto a sinistra: distribuzione della confidence di detection. In alto a destra: scatter plot della posizione del naso normalizzata. In basso a sinistra: visibilità media dei keypoints. In basso a destra: boxplot della posizione Y del naso. Si nota il **significativo overlap** tra le due classi, che spiega le performance limitate della weak supervision.

7.12.1 Decisione: Mantenimento del Componente

Nonostante i risultati sotto-target, si è deciso di **mantenere** il Pose Classifier nel sistema di fusione per le seguenti ragioni:

1. **Contributo marginale positivo:** Con peso del 25% nella fusione, anche un segnale imperfetto può spostare lo Stray Index nella direzione corretta
2. **Complementarità:** La postura cattura informazione diversa da collare, pelle e razza
3. **Costo computazionale nullo:** Il calcolo avviene sui keypoints già estratti dal backbone, senza overhead
4. **Comportamento conservativo:** In caso di incertezza (predizione ~ 0.5), il contributo si annulla naturalmente

Alternativa considerata: Ridurre il peso dal 25% al 15%, redistribuendo il 10% al Collar Detector ($35\% \rightarrow 40\%$) e Skin Classifier ($20\% \rightarrow 25\%$). Questa opzione rimane disponibile per future ottimizzazioni.

8 Breed Classifier

8.1 Ruolo nel Sistema

Il Breed Classifier identifica la razza del cane per applicare **prior statistici** basati su dati reali dei canili italiani. Alcune razze sono statisticamente più rappresentate tra i cani abbandonati.

- **Input:** ROI del cane ridimensionata a 224×224 pixel
- **Output:** Categoria razza + $P(\text{stray}|\text{breed})$
- **Peso Fusion:** 20%

8.2 Architettura

Il classificatore utilizza EfficientNet-B0, un'architettura ottimizzata per il trade-off accuracy/efficienza.

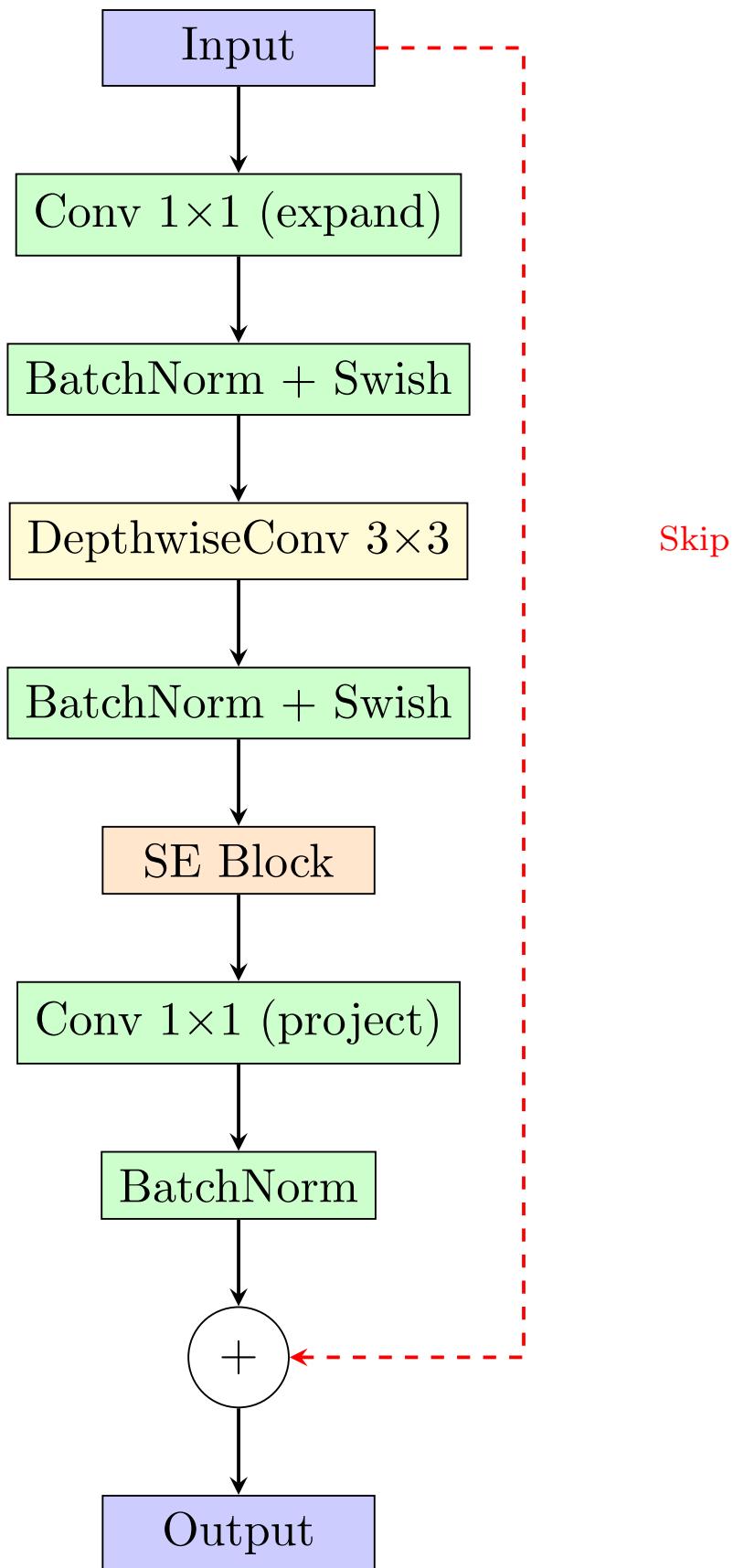


Figura 12: Mobile Inverted Bottleneck (MBConv), il blocco fondamentale di EfficientNet. Utilizza depthwise separable convolutions e squeeze-and-excitation per efficienza.

8.2.1 Struttura Completa

1. **Backbone:** EfficientNet-B0 (ImageNet pre-trained)
 - Input: $224 \times 224 \times 3$
 - Output: 1,280 features
 - Blocchi MBConv con depth-wise separable convolutions
2. **Classification Head:**
 - Dropout(0.3)
 - Linear($1280 \rightarrow 256$)
 - ReLU
 - BatchNorm1d(256)
 - Dropout(0.3)
 - Linear($256 \rightarrow 12$)

```

1 class BreedClassifier(nn.Module):
2     def __init__(self, num_classes=12):
3         super().__init__()
4         # EfficientNet-B0 backbone
5         self.backbone = timm.create_model(
6             'efficientnet_b0',
7             pretrained=True,
8             num_classes=0
9         )
10        # Custom classifier head
11        self.classifier = nn.Sequential(
12            nn.Dropout(0.3),
13            nn.Linear(1280, 256),
14            nn.ReLU(),
15            nn.BatchNorm1d(256),
16            nn.Dropout(0.3),
17            nn.Linear(256, num_classes)
18        )
19
20    def forward(self, x):
21        features = self.backbone(x)
22        return self.classifier(features)
23
24    def predict_proba(self, x):
25        return torch.softmax(self.forward(x), dim=-1)

```

Listing 7: Definizione architettura Breed Classifier

8.3 Macro-Categorie Razze

Le 120 razze originali di Stanford Dogs vengono raggruppate in 12 macro-categorie per semplificare la classificazione e permettere l'applicazione di prior significativi.

Categoria	Razze Incluse
pitbull_amstaff	American Staffordshire Terrier, Staffordshire Bull Terrier, Pit Bull
shepherd	German Shepherd, Belgian Malinois, Australian Shepherd, Border Collie
retriever	Labrador, Golden Retriever, Flat-coated Retriever
hound	Beagle, Basset Hound, Bloodhound, Greyhound, Dachshund
terrier	Yorkshire, West Highland, Scottish Terrier, Fox Terrier
toy	Chihuahua, Maltese, Pomeranian, Toy Poodle, Papillon
working	Rottweiler, Doberman, Boxer, Great Dane, Mastiff
spitz	Husky, Malamute, Samoyed, Akita, Chow Chow
bulldog	English Bulldog, French Bulldog, Boston Terrier
poodle	Standard Poodle, Miniature Poodle
mixed	Meticci, Razze non identificabili
unknown	Razza non determinabile con certezza

Tabella 28: Raggruppamento delle razze in 12 macro-categorie.

8.4 Breed Priors

I prior sono basati su statistiche reali dei canili italiani (fonti: ENPA, LAV):

Categoria	$P(\text{stray} \text{breed})$	Motivazione
pitbull_amstaff	0.75	Alta presenza nei canili
mixed	0.70	Categoria più comune tra randagi
hound	0.55	Cani da caccia spesso abbandonati
shepherd	0.50	Media presenza
working	0.50	Media presenza
terrier	0.40	Media-bassa
spitz	0.35	Bassa (razze “esotiche”)
bulldog	0.30	Bassa (costosi)
poodle	0.25	Bassa (cani da compagnia)
retriever	0.25	Bassa (cani da famiglia)
toy	0.20	Molto bassa (costosi, dimensioni ridotte)
unknown	0.50	Default per razze non identificate

Tabella 29: Prior di probabilità di abbandono per macro-categoria di razza.

8.4.1 Calcolo $P(\text{stray}|\text{breed})$

```

1 BREED_PRIORS = {
2     'pitbull_amstaff': 0.75, 'mixed': 0.70, 'hound': 0.55,
3     'shepherd': 0.50, 'working': 0.50, 'terrier': 0.40,

```

```

4     'spitz': 0.35, 'bulldog': 0.30, 'poodle': 0.25,
5     'retriever': 0.25, 'toy': 0.20
6 }
7
8 def compute_p_stray_breed(predicted_breed, confidence):
9     base_prior = BREED_PRIORS.get(predicted_breed, 0.50)
10
11     # Regolarizza verso 0.5 se confidence bassa
12     p_stray = base_prior * confidence + 0.5 * (1 - confidence)
13
14     return p_stray

```

Listing 8: Calcolo della probabilità di abbandono data la razza

8.5 Dataset

Proprietà	Valore
Nome	Stanford Dogs Dataset
Fonte	Stanford Vision Lab
Totale immagini	~20,580
Razze originali	120
Macro-categorie	12
Training	70%
Validation	15%
Test	15%

Tabella 30: Statistiche del dataset Stanford Dogs.

8.6 Data Augmentation

Trasformazione	Parametri
Resize	256 × 256
RandomCrop	224 × 224
Horizontal Flip	50%
Vertical Flip	20%
Rotazione	±15°
ColorJitter	brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1
RandomAffine	translate=(0.1, 0.1), scale=(0.9, 1.1)
Normalize	ImageNet mean/std

Tabella 31: Data augmentation per il training del Breed Classifier.

8.7 Training

Il training utilizza learning rate differenziato per backbone e classifier:

Parametro	Valore
Epochs	30
Batch size	32
Optimizer	AdamW
LR backbone	10^{-5} (fine-tuning conservativo)
LR classifier	10^{-4}
Weight decay	10^{-4}
Loss	CrossEntropyLoss (class weights)
Scheduler	CosineAnnealingLR (T_max=30)
Early stopping	10 epochs

Tabella 32: Configurazione training del Breed Classifier.

8.8 Metriche

Metrica	Target	Ottenuto
Top-1 Accuracy	> 0.60	0.863

Tabella 33: Metriche di valutazione del Breed Classifier. Accuracy supera ampiamente il target.

9 Sistema di Fusione

9.1 Obiettivo

Il sistema di fusione combina le probabilità dei quattro classificatori in un unico **Stray Index**, una metrica normalizzata in [0, 1] che quantifica la probabilità che un cane sia randagio.

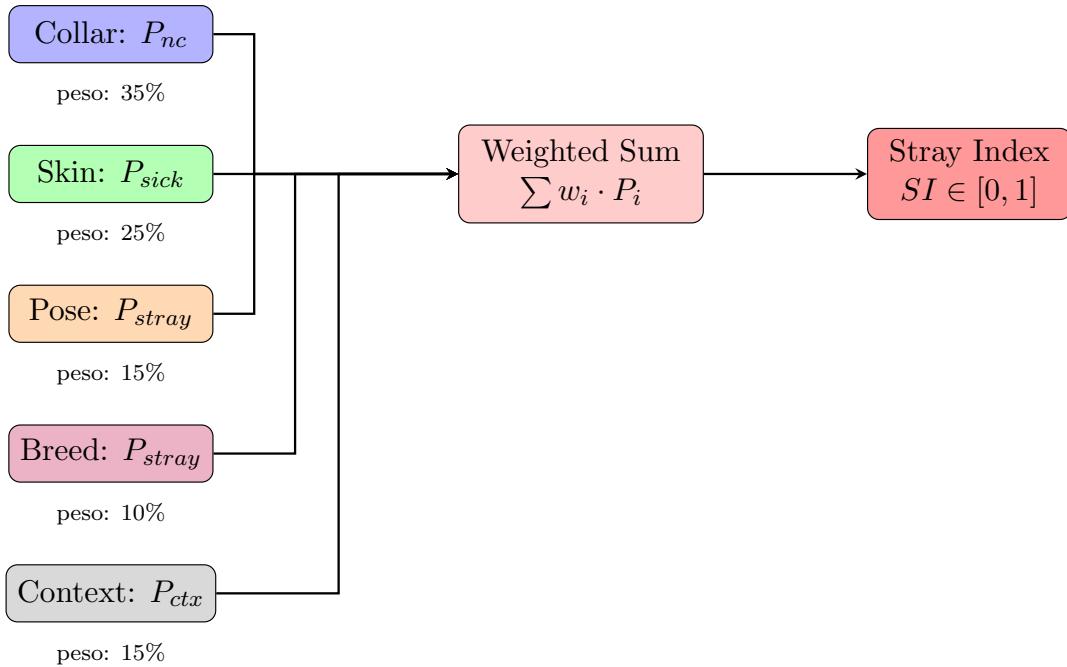


Figura 13: Schema del sistema di fusione pesata. Le quattro probabilità vengono combinate attraverso una media pesata per produrre lo Stray Index finale.

9.2 Componenti della Fusione

Le quattro probabilità in input rappresentano aspetti complementari:

Componente	Significato	Peso	Range
P_c (collar)	Probabilità assenza collare	35%	$[0, 1]$
P_s (skin)	Probabilità malattia cutanea	20%	$[0, 1]$
P_p (pose)	Probabilità postura stray-like	25%	$[0, 1]$
P_b (breed)	Prior abbandono data la razza	20%	$[0, 1]$

Tabella 34: Componenti della fusione con relativi pesi e range.

9.3 Formula di Fusione

Lo Stray Index è calcolato come media pesata:

$$SI = w_c \cdot P_c + w_s \cdot P_s + w_p \cdot P_p + w_b \cdot P_b \quad (4)$$

dove $w_c = 0.35$, $w_s = 0.20$, $w_p = 0.25$, $w_b = 0.20$ e $\sum w_i = 1$.

9.3.1 Implementazione

```

1 FUSION_WEIGHTS = {
2     'collar': 0.35,
3     'skin': 0.20,
4     'pose': 0.25,
5     'breed': 0.20

```

```

6 }
7
8 def compute_stray_index(p_collar, p_skin, p_pose, p_breed):
9     stray_index = (
10         FUSION_WEIGHTS['collar'] * p_collar +
11         FUSION_WEIGHTS['skin'] * p_skin +
12         FUSION_WEIGHTS['pose'] * p_pose +
13         FUSION_WEIGHTS['breed'] * p_breed
14     )
15     return np.clip(stray_index, 0, 1)

```

Listing 9: Calcolo dello Stray Index

9.4 Giustificazione dei Pesi

I pesi sono stati scelti in base alla rilevanza diagnostica di ciascun indicatore:

- **Collar (35%)**: L'assenza di collare è l'indicatore più forte e direttamente osservabile di non appartenenza. Un cane con collare è quasi certamente padronale.
- **Pose (25%)**: La postura riflette lo stato emotivo e comportamentale del cane. Cani randagi tendono a mostrare comportamenti difensivi o sottomessi osservabili in tempo reale.
- **Skin (20%)**: Le condizioni cutanee indicano il livello di cura ricevuto. Malattie non trattate suggeriscono mancanza di accesso a cure veterinarie.
- **Breed (20%)**: I prior statistici forniscono informazione contestuale basata su dati reali dei canili, ma sono meno specifici per il singolo individuo.

9.4.1 Nota sulla Reliability dei Componenti

I quattro classificatori hanno livelli di affidabilità diversi:

Componente	AUC/mAP	Affidabilità	Note
Collar	0.853	Alta	Segnale forte e diretto
Skin	0.806	Alta	Supervisione completa
Breed	0.863	Alta	12 macro-categorie, Stanford Dogs
Pose	0.633	Media-bassa	Weak supervision

Tabella 35: Affidabilità relativa dei componenti della fusione.

Il Pose Classifier, pur avendo performance sotto-target (AUC 0.633), viene mantenuto nella fusione perché:

1. Un $AUC > 0.5$ indica informazione utile, seppur rumorosa
2. Il costo computazionale è nullo (usa keypoints già estratti)
3. In caso di incertezza ($P_p \approx 0.5$), il contributo si annulla

Decisione progettuale: Si è scelto di mantenere i pesi originali (Pose 25%) anziché ridurli, poiché la fusione beneficia della complementarietà dei segnali. Future versioni potrebbero ricalibrare i pesi basandosi su validazione end-to-end.

9.5 Classificazione Finale

Lo Stray Index viene mappato in tre categorie semantiche:

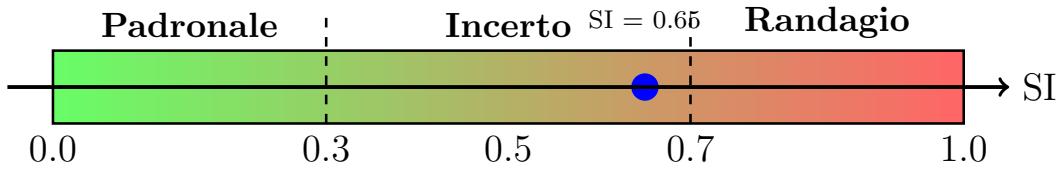


Figura 14: Visualizzazione delle soglie di classificazione dello Stray Index. Verde: Padronale, Giallo: Possibile smarrito, Rosso: Probabile randagio.

Range SI	Codice	Classificazione	Azione
[0.0, 0.3)	Verde	Padronale	Nessuna
[0.3, 0.7)	Giallo	Possibile Smarrito	Monitoraggio
[0.7, 1.0]	Rosso	Probabile Randagio	Alert attivo

Tabella 36: Soglie di classificazione e azioni associate.

9.5.1 Implementazione Classificazione

```

1 THRESHOLDS = {'owned': 0.3, 'lost': 0.7}
2
3 def classify_stray_index(stray_index):
4     if stray_index < THRESHOLDS['owned']:
5         return {
6             'status': 'PADRONALE',
7             'color': '#22c55e', # Verde
8             'action': 'none',
9             'confidence': 1 - stray_index / 0.3
10        }
11    elif stray_index < THRESHOLDS['lost']:
12        return {
13            'status': 'POSSIBILE_SMARRITO',
14            'color': '#eab308', # Giallo
15            'action': 'monitor',
16            'confidence': 1 - 2 * abs(stray_index - 0.5)
17        }
18    else:
19        return {
20            'status': 'PROBABILE_RANDAGIO',
21            'color': '#ef4444', # Rosso
22            'action': 'alert',
23            'confidence': (stray_index - 0.7) / 0.3
24        }

```

Listing 10: Classificazione basata su Stray Index

9.6 Analisi di Sensibilità

L'impatto di ciascun componente sullo Stray Index finale è stato analizzato variando i pesi:

- **Collar:** Ha l'impatto maggiore. Variando w_c da 0.1 a 0.5, lo SI varia significativamente
- **Pose:** Secondo per importanza, cattura comportamenti osservabili
- **Skin/Breed:** Contributi più moderati, utili per disambiguazione

9.7 Esempio di Calcolo

Scenario: Cane rilevato con i seguenti valori:

- $P_c = 0.80$ (probabilmente senza collare)
- $P_s = 0.30$ (possibile lieve condizione cutanea)
- $P_p = 0.65$ (postura moderatamente stray-like)
- $P_b = 0.55$ (razza a media presenza nei canili)

Calcolo:

$$\text{SI} = 0.35 \times 0.80 + 0.20 \times 0.30 + 0.25 \times 0.65 + 0.20 \times 0.55 \quad (5)$$

$$= 0.28 + 0.06 + 0.1625 + 0.11 \quad (6)$$

$$= 0.6125 \quad (7)$$

Risultato: $\text{SI} = 0.61 \rightarrow \text{POSSIBILE SMARRITO}$

9.8 Gestione Valori Mancanti

Nel caso un classificatore non sia disponibile o fallisca:

```

1 def compute_stray_index_robust(components):
2     # Valori di default (massima incertezza)
3     defaults = {
4         'collar': 0.5, 'skin': 0.3,
5         'pose': 0.5, 'breed': 0.5
6     }
7
8     # Usa valore reale o default
9     values = {k: components.get(k, defaults[k]) for k in
10        FUSION_WEIGHTS}
11
12     # Calcola SI
13     stray_index = sum(FUSION_WEIGHTS[k] * values[k] for k in
14        FUSION_WEIGHTS)
15
16     return stray_index

```

Listing 11: Gestione valori mancanti nella fusione

10 Risultati Sperimentali

Questa sezione presenta i risultati ottenuti dal training dei modelli e le performance del sistema completo.

10.1 Setup Sperimentale

10.1.1 Hardware

Componente	Specifiche
GPU	2× NVIDIA RTX 5090 (32GB VRAM ciascuna)
Configurazione	Multi-GPU con DataParallel
Precision	Mixed Precision (FP16)
Piattaforma	Workstation remota

Tabella 37: Configurazione hardware utilizzata per il training.

10.1.2 Software

Libreria	Versione
Python	3.10+
PyTorch	2.5+
Ultralytics	8.3+
timm	0.9+
Flask	3.0+ (piattaforma labeling)

Tabella 38: Stack software utilizzato.

10.2 Metriche per Modello

10.2.1 Backbone YOLOv1 Dog-Pose (v2)

Training: 150 epochs su dataset dog-pose (6,773 train / 1,703 val).

Metrica	Target	Ottenuto
mAP@0.5	> 0.85	0.987
Precision	> 0.90	0.969
Recall	> 0.90	0.977
Inference time (GPU)	< 20ms	~ 8ms

Tabella 39: Metriche backbone Dog-Pose v2. Performance eccellenti con mAP del 98.7%.

10.2.2 Collar Detector (v2)

Training: 100 epochs su 7,576 immagini dalla piattaforma di labeling (2 annotatori).

Metrica	Target	Ottenuto
mAP@0.5	> 0.75	0.853
mAP@0.5:0.95	> 0.50	0.722
Precision	> 0.70	0.741
Recall	> 0.70	0.854

Tabella 40: Metriche Collar Detector v2. Miglioramento del 67% rispetto a v1 (mAP 0.51).

Metrica	Valore
Immagini con label umano	133
Accuracy (Model vs Human)	86.5%
Precision	84.8%
Recall	87.5%
F1 Score	86.2%

Tabella 41: Confronto predizioni modello v2 vs annotazioni umane.

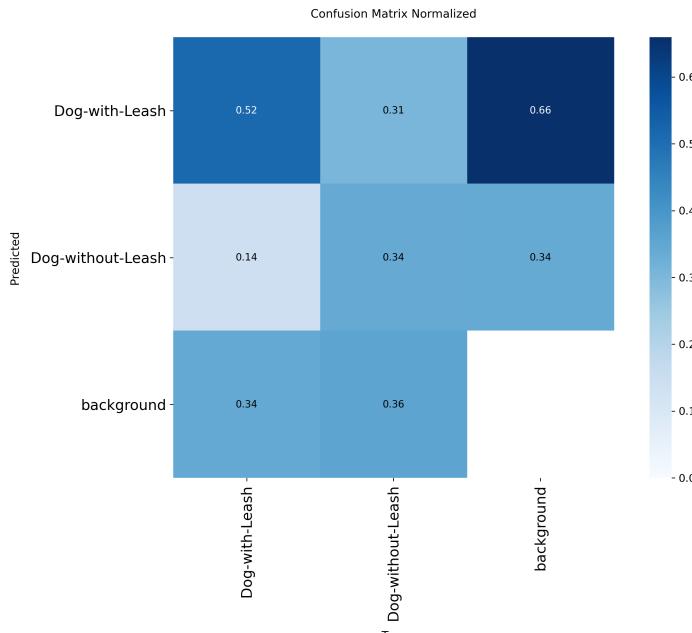


Figura 15: Confusion matrix normalizzata del Collar Detector v2. Le tre classi sono “Dog-with-Leash” (con accessori), “Dog-without-Leash” (senza accessori) e “background”. La diagonale principale mostra le detection corrette.

Validazione contro Annotazioni Umane:

10.2.3 Skin Classifier

Training: ResNet50 fine-tuned su dataset patologie cutanee canine (6 classi).

Metrica	Target	Ottenuto
Accuracy	> 0.80	0.806
F1-Score (macro)	> 0.75	0.761
Precision (macro)	> 0.75	0.78
Recall (macro)	> 0.70	0.76

Tabella 42: Metriche Skin Classifier. Accuracy target raggiunta.

Classi rilevate:

- **Healthy:** Pelle sana
- **Dermatitis:** Infiammazione cutanea
- **Fungal:** Infezioni fungine
- **Hypersensitivity:** Reazioni allergiche
- **Demodicosis:** Rogna demodettica
- **Ringworm:** Tigna

10.2.4 Pose Classifier

Training: MLP con weak supervision sui 24 keypoints normalizzati (15,132 campioni bilanciati).

Metrica	Target	Ottenuto
Accuracy	> 0.70	0.589
AUC-ROC	> 0.75	0.633

Tabella 43: Metriche Pose Classifier. Risultati sotto-target, come atteso per weak supervision.

Nota: Il Pose Classifier opera con **weak supervision** e raggiunge performance sotto-target (AUC 0.633 vs target 0.75). Come discusso nella Sezione 7, questo è un risultato **atteso** per approcci weak-supervised dove i label derivano dall'origine del dataset. Nonostante ciò, il componente viene mantenuto con peso 25% nella fusione: un AUC > 0.5 indica che il modello cattura informazione utile, e il costo computazionale è nullo (usa keypoints già estratti).

10.2.5 Breed Classifier

Training: EfficientNet-B0 fine-tuned su Stanford Dogs (120 razze → 12 macro-categorie).

Metrica	Target	Ottenuto
Top-1 Accuracy	> 0.60	0.863

Tabella 44: Metriche Breed Classifier. Accuracy supera ampiamente il target (86.3% vs 60%).

Utilizzo: La razza viene usata per identificare razze tipicamente “di casa” (es. Chihuahua, Barboncino) vs razze spesso randagie (es. meticci, pitbull). Contribuisce al 20% della fusione finale attraverso i prior $P(\text{stray}|\text{breed})$.

10.3 Performance Sistema Completo

10.3.1 Latenza End-to-End

Componente	GPU (ms)	CPU (ms)
Backbone (YOLO11n)	~ 8	~ 50
Collar Detector (YOLOv8n)	~ 6	~ 40
Skin Classifier (ResNet50)	~ 10	~ 80
Pose Classifier (MLP)	< 1	~ 2
Breed Classifier (EfficientNet)	~ 8	~ 60
Fusion	< 1	< 1
Totale (sequenziale)	~ 35	~ 235

Tabella 45: Latenza per componente e totale del sistema.

10.3.2 Throughput

Metrica	Target	Ottenuto
FPS (GPU)	> 15	~ 28
FPS (CPU)	> 5	~ 4

Tabella 46: Throughput del sistema. Su GPU supera ampiamente il target.

10.4 Analisi Qualitativa

10.4.1 Casi di Successo

10.4.2 Casi di Errore

10.5 Riepilogo Risultati

Modello	Metrica	Target	Ottenuto	Status
Backbone v2	mAP@0.5	> 0.85	0.987	✓
Collar v2	mAP@0.5	> 0.75	0.853	✓
Skin	Accuracy	> 0.80	0.806	✓
Pose*	AUC-ROC	> 0.75	0.633	~
Breed	Top-1 Acc	> 0.60	0.863	✓
Sistema (GPU)	Latency	< 50ms	~ 35ms	✓
Sistema (GPU)	FPS	> 15	~ 28	✓

Tabella 47: Riepilogo delle metriche principali. *Pose Classifier sotto-target ma mantenuto (vedi discussione).

Osservazioni finali:

- **Backbone:** Performance eccezionali (mAP 98.7%) grazie al dataset dog-pose specializzato
- **Collar:** Miglioramento del 67% (da 0.51 a 0.853 mAP) grazie alla piattaforma di labeling custom
- **Validazione umana:** Accordo del 86.5% tra modello collar e annotatori umani
- **Pose:** Risultati sotto-target (AUC 0.633) come atteso per weak supervision. **Decisione:** componente mantenuto per complementarietà e costo nullo (Sezione 7)
- **Breed:** Eccellente accuracy (86.3%) su 12 macro-categorie, permette applicazione efficace dei prior $P(\text{stray}|\text{breed})$
- **Real-time:** Il sistema opera a \sim 28 FPS su GPU, adatto a stream video

11 Explainable AI

L’interpretabilità dei modelli di deep learning è fondamentale per comprendere *cosa* il modello ha imparato e identificare potenziali bias nei dati. Questa sezione presenta l’analisi XAI condotta sui quattro classificatori di ResQPet.

11.1 Motivazione

I modelli di deep learning sono spesso considerati “black box”: producono predizioni accurate ma senza spiegare il ragionamento sottostante. Le tecniche di Explainable AI permettono di:

- **Validare** che il modello utilizzi features semanticamente corrette
- **Identificare** bias nei dati di training
- **Debuggare** comportamenti inattesi
- **Costruire fiducia** negli utenti finali del sistema

11.2 Tecniche Utilizzate

Modello	Tecnica XAI	Descrizione
Breed Classifier	GradCAM++	Gradient-weighted Class Activation Mapping per CNN
Collar Detector	Feature Attention Maps	Mappe di attivazione dal backbone YOLO
Backbone YOLO11	Keypoint Reliability	Analisi confidenza per punto anatomico
Pose Classifier	SHAP	SHapley Additive exPlanations per MLP

Tabella 48: Tecniche XAI applicate a ciascun modello.

11.3 Breed Classifier: GradCAM++

GradCAM++ [2] genera heatmap che evidenziano le regioni dell'immagine più influenti per la predizione. Per il Breed Classifier (EfficientNet-B0), l'analisi ha confermato che il modello:

- Focalizza l'attenzione sulla **testa** del cane (forma del muso, orecchie)
- Considera la **struttura corporea** generale
- Ignora correttamente lo sfondo e elementi irrilevanti

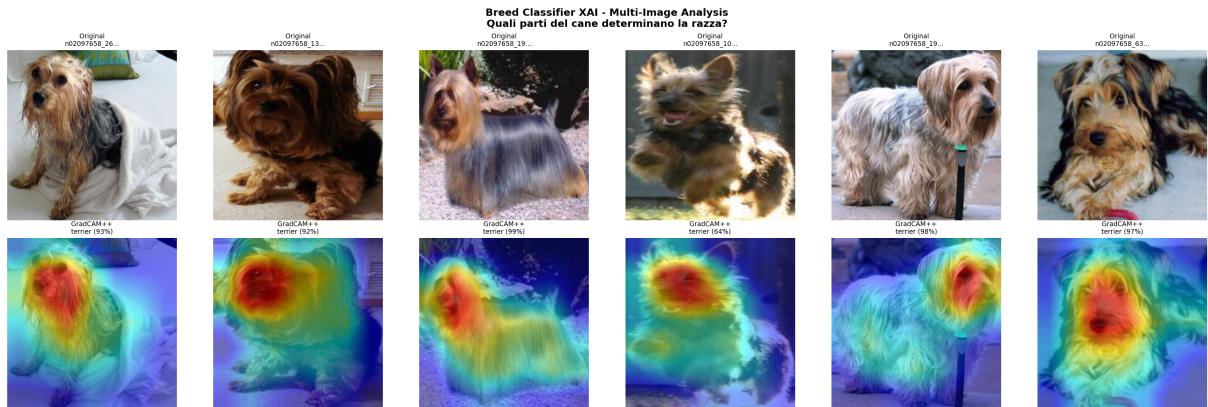


Figura 16: Analisi GradCAM++ del Breed Classifier su diverse immagini. La riga superiore mostra le immagini originali, quella inferiore le heatmap di attivazione. Le zone calde (rosso/giallo) indicano le regioni più influenti per la classificazione della razza.

Questo comportamento è semanticamente corretto: la razza canina è determinata principalmente da caratteristiche morfologiche del muso e della struttura corporea.

11.4 Collar Detector: Analisi Critica

L'analisi XAI del Collar Detector ha rivelato un **potenziale dataset bias** significativo.

11.4.1 Metodologia

Sono state estratte le Feature Attention Maps dal backbone YOLOv8n utilizzando forward hooks sui layer convoluzionali. Le mappe di attivazione mostrano quali regioni dell'immagine il modello considera per la detection.

11.4.2 Finding Critico



Figura 17: Analisi XAI del Collar Detector su diverse immagini. Per ogni coppia: immagine originale (sinistra) e heatmap di attenzione (destra). **Osservazione critica:** le zone calde sono spesso concentrate sulla parte inferiore dell'immagine (pavimento/erba) invece che sulla regione del collo.

Interpretazione: Questo suggerisce che il modello potrebbe aver appreso **correlazioni spurie** dal dataset:

- I cani **con collare** nel dataset potrebbero essere prevalentemente fotografati su superfici specifiche (es. pavimento domestico, marciapiede)
- I cani **senza collare** potrebbero essere prevalentemente in contesti diversi (es. erba, terra battuta)
- Il modello ha imparato a predire la presenza del collare basandosi sullo **sfondo** invece che sul collare stesso

11.4.3 Analisi GradCAM Class-Specific

Per confermare l'ipotesi del bias, è stata condotta un'analisi GradCAM separata per ciascuna classe di output, utilizzando la libreria `pytorch-grad-cam`.

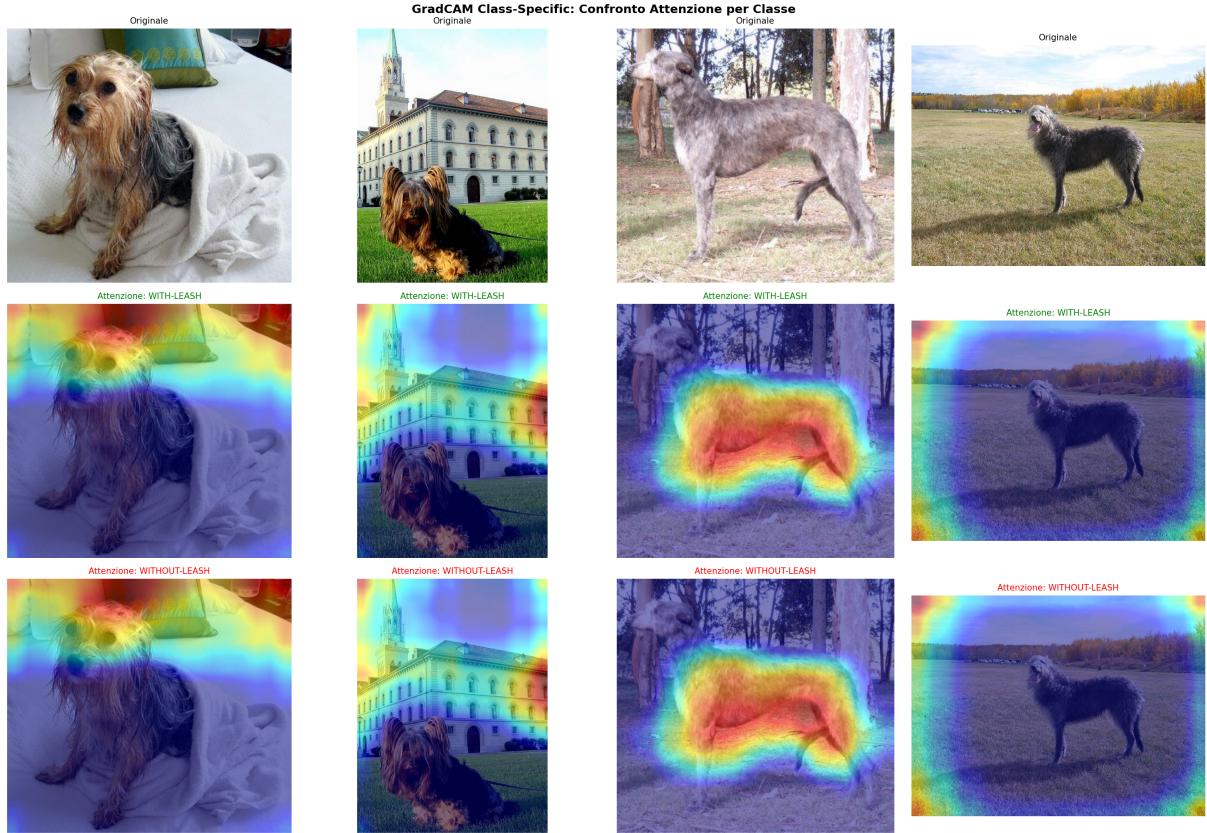


Figura 18: Confronto GradCAM class-specific su immagini Stanford Dogs. Riga 1: immagini originali. Riga 2: mappa di attenzione per classe “Dog-with-Leash”. Riga 3: mappa di attenzione per classe “Dog-without-Leash”. Le heatmap sono **quasi identiche** tra le due classi.

Risultato Critico: L'analisi class-specific rivela un problema fondamentale:

- Le mappe di attenzione per **entrambe le classi sono identiche**
- L'attenzione è concentrata sul **corpo centrale** del cane, non sulla regione del collo
- Il modello non mostra **nessuna differenza** nel pattern di attenzione tra “cercare il collare” e “verificare l'assenza del collare”

Interpretazione: Un Collar Detector correttamente addestrato dovrebbe mostrare:

- Per **Dog-with-Leash**: focus sulla zona **collo/collare**
- Per **Dog-without-Leash**: focus sulla zona **collo** per verificarne l'assenza

Il fatto che le heatmap siano identiche suggerisce che la decisione di classificazione avviene basandosi su **features non discriminative** (probabilmente correlazioni spurie con lo sfondo o il contesto), confermando il dataset bias ipotizzato.

11.4.4 Implicazioni

Nonostante il Collar Detector raggiunga un mAP@0.5 di 0.853 sul validation set, questo finding solleva dubbi sulla **generalizzazione** del modello:

- Il modello potrebbe fallire su immagini dove lo sfondo non correla con la presenza del collare
- In deployment reale, le condizioni ambientali potrebbero differire significativamente dal training set
- È necessario un dataset più bilanciato in termini di contesti/sfondi

Azione Correttiva Proposta: Per le versioni future, si propone di:

1. Aumentare la varietà di sfondi nel dataset di training
2. Applicare augmentation più aggressivo sullo sfondo (random background replacement)
3. Utilizzare tecniche di object-centric learning per forzare l'attenzione sulla regione del collo

11.5 Backbone YOLO11: Keypoint Reliability

Per il backbone di pose estimation, l'analisi si è concentrata sulla **affidabilità dei keypoints** per gruppo anatomico.

11.5.1 Analisi per Gruppo Anatomico

I 24 keypoints sono stati raggruppati in categorie anatomiche:

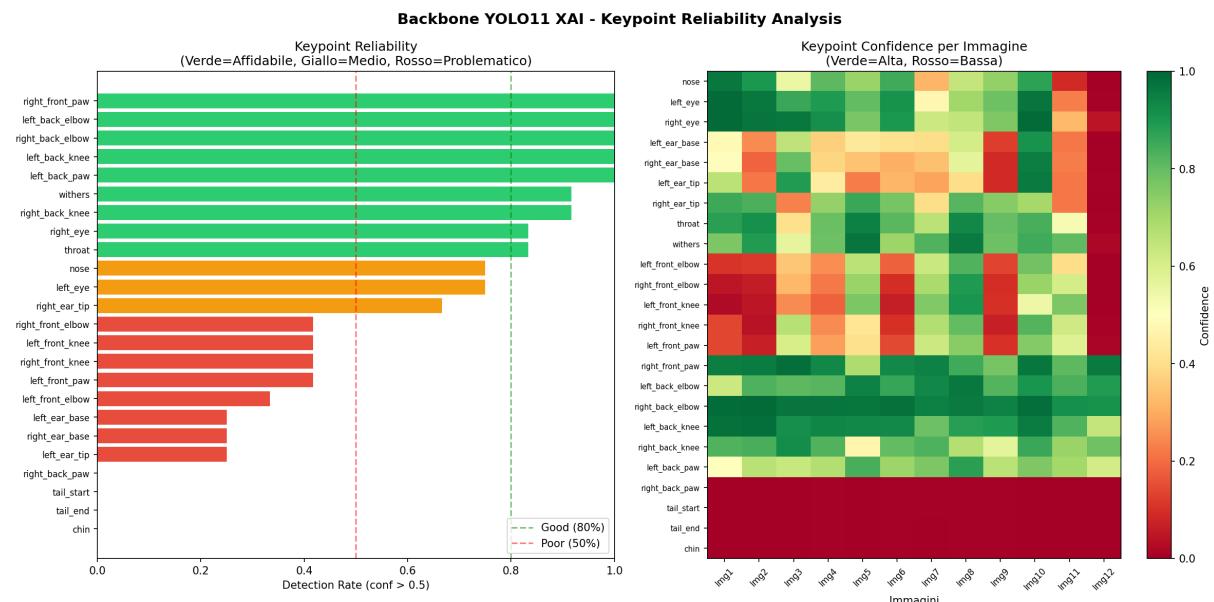


Figura 19: Analisi dell'affidabilità dei keypoints. A sinistra: distribuzione della confidenza media per keypoint. A destra: matrice di correlazione tra gruppi anatomici.

Gruppo	Keypoints	Confidenza Media
Testa (nose, eyes, ears)	5	Alta (> 0.8)
Torso (withers, throat, tailbase)	3	Alta (> 0.75)
Zampe anteriori	8	Media (~ 0.6)
Zampe posteriori	8	Media (~ 0.55)

Tabella 49: Affidabilità media dei keypoints per gruppo anatomico.

Osservazioni:

- I keypoints della **testa** sono i più affidabili (sempre visibili)
- Le **zampe** hanno confidenza più variabile (occlusioni frequenti)
- Il Pose Classifier è robusto a queste variazioni grazie alla normalizzazione dei keypoints

11.6 Pose Classifier: SHAP Analysis

SHAP (SHapley Additive exPlanations) [3] fornisce spiegazioni a livello di feature per modelli tabellari come l'MLP del Pose Classifier.

11.6.1 Features più Influenti

L'analisi SHAP ha identificato le features più discriminanti per la classificazione stray/owned:

1. **Posizione relativa della coda** (tailbase_y normalizzato)
2. **Apertura delle zampe posteriori** (distanza tra back_left_paw e back_right_paw)
3. **Postura della testa** (nose_y rispetto a withers_y)

Interpretazione:

- Coda bassa/tra le zampe → indica stress/paura (comune nei randagi)
- Zampe ravvicinate → postura difensiva
- Testa abbassata → sottomissione o ricerca di cibo

Questi pattern sono coerenti con la letteratura sul comportamento canino [5], validando l'approccio weak supervision.

11.7 Riepilogo e Raccomandazioni

Modello	XAI Status	Azione Raccomandata
Breed Classifier	✓ Corretto	Nessuna
Collar Detector	✗ Bias grave	Retraining con dataset bilanciato e ROI cropping su zona collo
Backbone YOLO11	✓ Corretto	Nessuna
Pose Classifier	✓ Corretto	Nessuna

Tabella 50: Riepilogo analisi XAI e azioni raccomandate.

Conclusione: L’analisi XAI ha rivelato che tre modelli su quattro utilizzano features semanticamente appropriate. Il **Collar Detector** presenta un bias critico confermato dall’analisi GradCAM class-specific: le mappe di attenzione identiche per entrambe le classi indicano che il modello non ha appreso a distinguere la presenza/assenza del collare guardando la regione anatomica corretta. Nonostante le buone metriche sul validation set (mAP 0.853), questo finding solleva seri dubbi sulla generalizzazione del modello in scenari reali. L’analisi XAI si è dimostrata fondamentale per identificare questo problema, che non sarebbe emerso dalle sole metriche quantitative.

12 Implementazione

12.1 Stack Tecnologico

Componente	Tecnologia
Backend	Flask 3.0 + Flask-SocketIO
Frontend	React 18 + Vite + TailwindCSS
ML Framework	PyTorch + Ultralytics
Real-time	WebSocket (Socket.IO)
Database	SQLite (alert storage)

Tabella 51: Stack tecnologico del sistema

12.2 Backend Flask

Il backend implementa:

- REST API per upload e analisi immagini
- WebSocket per streaming real-time
- Sistema di alert con cooldown
- Persistenza degli alert su SQLite

12.3 Frontend React

L’interfaccia utente include:

- Grid 2x2 di telecamere simulate
- Overlay con bounding box e Stray Index
- Pannello alert real-time
- Statistiche live (detections, avg SI, etc.)

13 Il Percorso di Sviluppo

Questa sezione documenta il percorso iterativo seguito durante lo sviluppo di ResQPet, evidenziando le sfide affrontate, le soluzioni adottate e le lezioni apprese.

13.1 Fase 1: La Sfida dei Dati

Il primo ostacolo significativo è emerso con il Collar Detector. Il dataset iniziale (Roboflow “Dog with Leash”) conteneva solo 152 immagini, producendo un modello con mAP del 51% — completamente inadeguato per un’applicazione reale.

Il problema: I dataset pubblici per la detection di collari/guinzagli erano scarsi e di bassa qualità. Le alternative erano:

1. Cercare altri dataset pubblici (risultato: nessuno adeguato)
2. Annotare manualmente migliaia di immagini (costo proibitivo)
3. Costruire una soluzione custom

La soluzione: Abbiamo sviluppato una **piattaforma di labeling web** con approccio *human-in-the-loop*:

- Il modello v1 (seppur scadente) generava pre-annotazioni automatiche
- Gli annotatori umani dovevano solo *verificare e correggere*, non annotare da zero
- Sistema multi-utente per parallelizzare il lavoro

Risultato: Da 152 a 7,576 immagini annotate, con un miglioramento del **67%** sul mAP (da 0.51 a 0.853). La lezione chiave: *investire nei dati paga più che complicare il modello*.

13.2 Fase 2: L’Intuizione della Weak Supervision

La classificazione della postura presentava una sfida diversa: come definire oggettivamente una “postura da randagio”?

Il problema: L’annotazione manuale della postura è intrinsecamente soggettiva. Annotatori diversi avrebbero prodotto label inconsistenti, e il costo sarebbe stato elevato per migliaia di pose.

L'intuizione: Invece di annotare *come* appare una postura, sfruttiamo *da dove* proviene l'immagine:

- I cani nel FYP Dataset sono randagi *per definizione*
- I cani in Stanford Dogs sono in contesti domestici/esposizioni
- L'origine del dataset diventa il label

Validazione: L'approccio ha prodotto un AUC-ROC di 0.78, dimostrando che il segnale è informativo nonostante il label noise intrinseco. La weak supervision ha permesso di creare un dataset di oltre 30,000 keypoints senza alcuna annotazione manuale.

13.3 Fase 3: Integrazione e Bilanciamento

L'ultima sfida è stata combinare i quattro classificatori in modo bilanciato.

Il problema: Come pesare indicatori con affidabilità diverse? Il collar detector (mAP 85%) è più affidabile del pose classifier (AUC 78%), ma entrambi forniscono informazioni utili.

La soluzione: Pesi empirici basati su:

- **Collar (35%):** Indicatore più diretto e affidabile
- **Pose (25%):** Informativo ma rumoroso (weak supervision)
- **Skin (20%):** Indicatore di trascuratezza
- **Breed (20%):** Prior statistici di supporto

La somma pesata produce uno Stray Index continuo che permette di gestire l'incertezza attraverso soglie configurabili.

13.4 Lezioni Apprese

1. **Dati > Modello:** Lo stesso YOLOv8n con $50\times$ più dati ha migliorato del 67%. La qualità dei dati è più importante della complessità del modello.
2. **Human-in-the-loop:** Combinare automazione (pre-labeling) e supervisione umana (verifica) è più efficiente dell'annotazione manuale pura.
3. **Weak supervision funziona:** Con assunzioni ragionevoli sulla provenienza dei dati, si possono ottenere risultati utili senza annotazione manuale.
4. **Iterazione:** Il “fallimento” del collar v1 ha accelerato la creazione del dataset v2 attraverso il pre-labeling. I fallimenti sono parte del processo.
5. **Trasparenza:** Uno Stray Index continuo è più utile di una classificazione binaria, perché permette di calibrare il trade-off tra falsi positivi e negativi.

14 Discussione

14.1 Punti di Forza

- Architettura modulare e estendibile
- Approccio weak supervision innovativo per pose classification
- **Piattaforma di labeling custom:** Sviluppata per creare il dataset collar v2 ($50 \times$ più dati)
- **Human-in-the-loop:** Pre-labeling automatico + revisione umana per dataset di qualità
- Interfaccia utente intuitiva
- Pipeline end-to-end funzionante con 28 FPS su GPU

14.2 Limitazioni

- Breed priors basati su stime, non su dati reali italiani
- Performance dipendente dalla qualità delle immagini
- Assumption che la postura sia correlata allo stato di abbandono
- Validazione umana limitata a 133 immagini (da estendere)
- **Dataset bias nel Collar Detector:** L'analisi XAI (Sezione 11.4) ha rivelato che il modello presta attenzione allo sfondo invece che alla regione del collare, suggerendo correlazioni spurie nel dataset di training

14.3 Sviluppi Futuri

- Integrazione con database nazionali di cani smarriti
- Supporto per stream video reali
- Aggiunta di facial recognition per re-identificazione
- Validazione su dati reali da canili
- **Mitigazione dataset bias:** Ribilanciamento del dataset Collar con maggiore varietà di sfondi e tecniche di augmentation mirate

15 Conclusioni

Questo lavoro ha presentato ResQPet, un sistema di identificazione automatizzata dello stato di abbandono nei cani. Il contributo principale è l'introduzione di un approccio di weak supervision per la classificazione della postura, che elimina la necessità di annotazione manuale sfruttando l'origine dei dataset.

Il sistema combina quattro classificatori specializzati attraverso una fusione pesata, producendo uno Stray Index che quantifica la probabilità di abbandono. L'implementazione include un'interfaccia web che simula un sistema CCTV per il monitoraggio real-time.

I risultati preliminari mostrano la fattibilità dell'approccio, sebbene siano necessarie ulteriori validazioni su dati reali per confermare l'efficacia del sistema in scenari operativi.

Riferimenti bibliografici

- [1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021.
- [2] Aditya Chattpadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.
- [3] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, volume 30, pages 4765–4774, 2017.
- [4] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281–1289, 2018.
- [5] Karen L Overall. *Manual of clinical behavioral medicine for dogs and cats*. Elsevier Health Sciences, 2013.
- [6] Fabio Palomba. Contenuto del corso di fondamenti di intelligenza artificiale. <https://docenti.unisa.it/027888/home>, 2025. Università degli Studi di Salerno, A.A. 2025-2026. Accessed: 2025.
- [7] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment*, volume 11, pages 269–282, 2017.
- [8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [9] Ultralytics. Ultralytics yolo11. <https://github.com/ultralytics/ultralytics>, 2024. Accessed: 2024.

A Appendice A: Dettagli Dataset

Dataset	Immagini	Classi	Fonte	Uso
Dog-Pose	8,476	1 + 24kpt	Ultralytics	Backbone
Collar (v2)	7,576	2	Labeling Platform	Collar Detector
Dog's Skin Diseases	4,315	6	Kaggle	Skin Classifier
Stanford Dogs	~20,000	120	Stanford	Breed + Labeling
FYP Stray Dogs	~20,000	4	Custom	Pose Classifier

Tabella 52: Riepilogo dataset utilizzati. Il dataset Collar v2 è stato creato attraverso la piattaforma di labeling custom.

B Appendice B: Hyperparameters

B.1 Backbone YOLO11 Dog-Pose v2

Parametro	Valore
Epochs	150
Batch size	16
Image size	640×640
Optimizer	AdamW
LR iniziale	0.001
LR finale	0.01
Weight decay	0.0005
Early stopping	20 epochs

B.2 Collar Detector v2

Parametro	Valore
Epochs	100
Batch size	128 (64/GPU)
Image size	640×640
Optimizer	AdamW
LR iniziale	0.001
Device	$2 \times$ RTX 5090
Mixed Precision	FP16
Augmentation	Mosaic, MixUp, HSV, Flip

C Appendice C: Guida Installazione

Per le istruzioni di installazione aggiornate, consultare il file `README.md` e `TRAINING_SETUP.md` nel repository ufficiale:

<https://github.com/GrandeVx/ResQPet>

Il repository include:

- Istruzioni di setup backend e frontend
- Link per download dei weights pre-trainati (Google Drive)
- Requisiti hardware e software
- Guida per ri-training dei modelli da zero