

POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica



Tesi di Laurea Magistrale

Classificazione automatica di immagini di alimenti

Relatori

Prof. MAURIZIO MORISIO

Prof. ANDREA BOTTINO

Candidato

GIANMARCO BELLAVIA

Aprile 2025

Sommario

Il cibo è molto più di una semplice necessità biologica: è un pilastro fondamentale della nostra esistenza, intrecciato con la salute, le emozioni e la cultura. Attraverso il cibo, le persone celebrano tradizioni, esprimono la propria identità e costruiscono relazioni. Da sempre, ciò che mangiamo non solo soddisfa un bisogno primario, ma racconta anche chi siamo e da dove veniamo.

Oltre all'aspetto culturale, il cibo ha un ruolo centrale nella salute e nel benessere delle persone. Una dieta equilibrata e ricca di nutrienti è essenziale per prevenire malattie, migliorare l'energia e sostenere il funzionamento ottimale del corpo. Negli ultimi anni, è aumentata la consapevolezza sull'importanza di una corretta alimentazione, spingendo sempre più persone a scegliere alimenti salutari e a limitare il consumo di cibi trasformati e ricchi di zuccheri o grassi saturi.

Tuttavia, il tracciamento dei valori nutrizionali dei cibi consumati rappresenta una sfida sempre più rilevante nella società contemporanea, soprattutto per chi desidera mantenere una dieta equilibrata e uno stile di vita sano. Per semplificare questo processo, si propone lo sviluppo di un'applicazione innovativa in grado di riconoscere, attraverso una fotografia o un'immagine fornita dall'utente, il cibo rappresentato e stimarne il volume.

Questo lavoro di tesi esamina diversi approcci per la classificazione e la segmentazione di immagini di cibo.

In particolare vengono esaminati due diversi approcci:

- Il primo approccio utilizza il modello YOLOv8 [1] con l'obiettivo di integrare la classificazione delle immagini con la segmentazione degli alimenti;
- Il secondo approccio prevede due fasi distinte per la classificazione (utilizzando ResNet50 [2] o InceptionV3 [3]) e la segmentazione delle immagini (attraverso SAM [4]).

Inoltre viene presentata un'applicazione demo-interattiva che integra il lavoro. Tale demo permette:

- Caricamento e classificazione dell'immagine;
- Processo di segmentazione:

- Guidato tramite l’uso di SAM (Segment Anything Model) [4];
 - Manuale tramite il tracciamento del contorno.
- Stima del volume del cibo attraverso l’utilizzo di una mappa di profondità ottenuta con MiDaS [5].

Ringraziamenti

Desidero esprimere la mia sincera gratitudine ai Relatori, Professor Maurizio Morisio e Professor Andrea Bottino, per la loro preziosa guida e disponibilità durante la realizzazione della tesi.

Un ringraziamento alla mia famiglia, senza la quale questo giorno non sarebbe mai stato possibile. Il loro amore incondizionato, il sostegno costante e la fiducia nelle mie capacità mi hanno accompagnato lungo tutto questo percorso. Mi hanno insegnato il valore della determinazione e della resilienza; per questo non potrò mai ringraziarli abbastanza.

Un ringraziamento speciale va anche ai miei amici William, Valentina, Simone, Daniele, Miriam, Giuseppe, Tomaso e Michela, che sono stati una presenza fondamentale in questi anni. Grazie per aver condiviso con me momenti di gioia e di difficoltà, per avermi incoraggiato nei momenti più bui e per avermi ricordato, con una risata o una parola gentile, che non ero solo. Vi ringrazio per avermi supportato e sopportato lungo questo viaggio e per avermi mostrato quanto sia straordinario poter contare su veri amici. La vostra amicizia è stata una fonte inesauribile di forza e serenità e spero di poter ricambiare tutto il bene che mi avete dato.

A tutti voi, dal profondo del cuore: Grazie.

Indice

Elenco delle tabelle	VIII
Elenco delle figure	X
Acronyms	XV
1 Introduzione	1
1.1 Analisi del problema	1
1.2 Obiettivi della tesi	2
1.3 Contributo della tesi	3
2 Dataset	4
2.1 Food-101	4
2.1.1 Caratteristiche principali di Food-101	4
2.1.2 Composizione e dimensioni	4
2.1.3 Origine e pre-elaborazione	5
2.1.4 Varietà e complessità	5
2.1.5 Confronto con dataset simili	6
2.1.6 Applicazioni del Dataset	6
2.2 FoodSeg103	7
2.2.1 Caratteristiche Principali di FoodSeg103	7
2.2.2 Comparazione con dataset precedenti	8
2.2.3 Contributi principali	9
3 Modelli	11
3.1 Residual neural network	11
3.1.1 Limiti delle reti classiche	11
3.1.2 Soluzione offerta dalla ResNet	12
3.1.3 Struttura della ResNet	14
3.2 Inceptionv3	17
3.2.1 Caratteristiche principali di Inception-v3	18

3.2.2	Struttura dell'architettura	24
3.2.3	Prestazioni di Inception-v3	25
3.3	YoloV8	26
3.3.1	Evoluzioni delle Versioni YOLO	26
3.3.2	Architettura di YOLOv8	28
3.3.3	Nuova Funzione di Perdita (Loss Function)	30
3.3.4	Ottimizzazioni e prestazioni	30
3.3.5	Modelli di YOLOv8	32
3.3.6	Formato di annotazione	32
3.4	SAM	32
3.4.1	Task: Segmentazione Promptable	33
3.4.2	Architettura del Modello	33
3.4.3	Dataset SA-1B e Motore di Dati	37
3.4.4	Prestazioni e Applicazioni Zero-Shot	38
3.4.5	Conclusioni e limitazioni	40
3.5	MiDaS	40
3.5.1	Obiettivo del modello MiDaS	41
3.5.2	Architettura del modello	41
3.5.3	Prestazioni	46
4	Soluzione proposte	49
4.1	Approccio 1	49
4.1.1	Risultati	50
4.2	Approccio 2	58
4.2.1	Classificazione delle immagini	58
4.2.2	Training con ResNet50 su fastai	67
4.2.3	Preparazione del dataset	67
4.2.4	Training del modello	69
4.2.5	Test del modello	79
5	Demo	81
5.0.1	Caricamento dell'immagine	81
5.0.2	Segmentazione	81
5.0.3	Stima del volume	89
5.0.4	Risultati sul calcolo del volume	94
6	Conclusioni	95
A	Categorie Food-101	96
	Bibliografia	101

Elenco delle tabelle

3.1	Architetture per ImageNet[31]. I blocchi costitutivi sono mostrati tra parentesi (vedi anche la Fig 3.3), con il numero di blocchi impilati. Il downsampling viene eseguito da conv3_1, conv4_1 e conv5_1 con uno stride di 2.[2]	15
3.2	La struttura del modello Iception-v3. La dimensione dell'output di ogni modulo è la dimensione dell'input del successivo.[32]	25
3.3	Confronto delle performance metrics di YOLOv5 e YOLOv8 [37, 38].[1]	31
3.4	Performance metrics dei modelli YOLOv8.[1]	32
3.5	Generazione di proposte di oggetti su LVIS v1 [46]. SAM è applicato in modalità zero-shot, ovvero non è stato addestrato per la generazione di proposte di oggetti né ha avuto accesso alle immagini o alle annotazioni di LVIS.[4]	39
3.6	Lista dei film e il numero di frame estratti nel dataset 3D Movies.[5]	46
4.1	Risultati sulle 2135 immagini del validation set per l'intero set (all) e per ciascuna classe. In ogni riga è riportato il numero di istanze e le metriche sia per la box che per la maschera(riga sopra box, riga sotto maschera).	56
4.2	Distribuzione delle immagini usando uno slit del 75% del training set.	58
4.3	Miglior valore di accuracy su validation set e test accuracy Inception-v3 con: LR=0.0015, batch-size = 32 e usando data augmentation.	59
4.4	Miglior valore di accuracy su validation set e test accuracy ResNet50 con: LR=0.015, batch-size = 64 e usando data augmentation.	60
4.5	Distribuzione delle immagini usando uno slit del 80% del training set.	62
4.6	Miglior valore di accuracy su validation set e test accuracy Inception-v3 con: LR=0.0015, batch-size = 64 e data augmentation.	62
4.7	Miglior valore di accuracy su validation set e test accuracy ResNet50 con: LR=0.0015 e batch-size = 64 senza data augmentation.	63
4.8	Miglior valore di accuracy su validation set e test accuracy ResNet50 con: LR=0.0015, batch-size = 64 e data augmentation.	63

4.9	Andamento del modello nel primo segmento (prime 8 epoche). . . .	70
4.10	Andamento del modello nel secondo segmento (dalla nona alla sedicesima epoca).	71
4.11	Andamento del modello nel terzo segmento (dalla diciassettesima alla ventesima epoca).	73
4.12	Andamento del modello nel quarto segmento (prime 8 epoche). . . .	76
4.13	Andamento del modello nel quinto segmento (dalla quinta alla ottava epoca).	77
4.14	Andamento del modello nel sesto segmento (dalla nona alla dodicesi- ma epoca).	78
4.15	Valori di loss, accuracy e top 5 accuracy senza TTA.	79
4.16	Valori di accuracy e top 5 accuracy con TTA.	79
5.1	Errore medio sulla stima del volume sulle immagini raccolte.	94

Elenco delle figure

2.1	Un esempio per 100 delle 101 classi del dataset Food-101. Notare l'elevata varianza nel tipo di cibo, colore, esposizione e livello di dettaglio, ma anche tipi di cibo visivamente e semanticamente simili.[8]	5
2.2	Esempi in FoodSeg103.[10]	7
2.3	Esempi di FoodSeg103: a sinistra l'immagine sorgente a destra le annotazioni.[10]	8
2.4	Confronto tra diversi stili di annotazione per mascherare le immagini di cibo: immagini sorgenti (a sinistra), annotazioni ingredient-level (al centro) e annotazioni dish-level (a destra).[10]	9
2.5	Esempi qualitativi dei risultati di ReLeM-CCNet e CCNet su FoodSeg103. Si può notare come un modello più complesso quale ReLeM-CCNet sia in grado di fornire predizioni più accurate.[10]	10
3.1	Training error (sinistra) e test error (destra) su CIFAR-10 [23] con 20-layer e 56-layer su reti "plain". La rete più profonda ha un training error maggiore e pertanto un test error maggiore.[2]	12
3.2	Residual learning: un building block.[2]	13
3.3	Una deeper residual function F per ImageNet[31]. A sinistra: un building block (su feature con mappe 56×56) come nella Fig. 3.4 per la ResNet34. A destra: un "bottleneck" building block per ResNet-50/101/152.[2]	15
3.4	Esempio di architetture di rete per ImageNet[31]. A sinistra: il modello VGG-19 [20] (19,6 miliardi di FLOPs) come riferimento. Al centro: una rete plain con 34 strati parametrici (3,6 miliardi di FLOPs). A destra: una rete residual con 34 strati parametrici (3,6 miliardi di FLOPs). Le connessioni shortcut tratteggiate aumentano le dimensioni. La 3.1 mostra maggiori dettagli e altre varianti.[2]	16
3.5	Mini-network che sostituisce le convoluzioni 5×5 . [32]	18
3.6	A sinistra il modulo Inception originale come proposto [32] a destra il nuovo modulo Inception dove ciascuna convoluzione 5×5 è sostituita da due convoluzioni 3×3 . [32]	19

3.7	Inception modules dopo la fattorizzazione delle $n \times n$ convoluzioni.[32]	20
3.8	Mini-network che sostituisce le convoluzioni 3×3 . Il livello inferiore di questa rete consiste in una convoluzione 3×1 con 3 unità di output.[32]	21
3.9	Modulo Inception con output del filtro espanso.[32]	21
3.10	Modulo Inception con output del filtro espanso.[32]	22
3.11	Modulo Inception che riduce la dimensione della griglia mentre espande i banchi di filtri.[32]	23
3.12	Architettura Inception-v3.[32]	24
3.13	Timeline versioni di YOLO.[35]	26
3.14	Architettura YOLOv1.[33]	27
3.15	Architettura YOLOv8.[36]	29
3.16	Architettura di SAM.[4]	33
3.17	Encoder di prompt.[4]	34
3.18	Decoder di maschere (mask decoder).[4]	35
3.19	Ogni colonna mostra 3 maschere valide generate da SAM a partire da un singolo prompt ambiguo rappresentato da un punto (cerchio verde).[4]	36
3.20	Motore dei dati (in alto) e dataset (in basso).[4]	37
3.21	Proprietà delle maschere del dataset. La legenda fa riferimento al numero di immagini e maschere in ciascun dataset (LVIS v1 [46], COCO [47], ADE20K [48] e Open Images [45]). i noti che SA-1B contiene 11 volte più immagini e 400 volte più maschere rispetto al più grande dataset di segmentazione esistente, Open Images.[4]	38
3.22	Predizione dei contorni in modalità zero-shot su BSDS500 [49]. SAM non è stato addestrato per prevedere mappe dei contorni né ha avuto accesso alle immagini o alle annotazioni di BSDS durante l'addestramento.[4]	38
3.23	Segmentazione da testo a maschera in modalità zero-shot. SAM può funzionare con prompt testuali semplici o complessi. Quando SAM non riesce a fare una predizione corretta, un prompt aggiuntivo basato su un punto può essere di aiuto.[4]	39
3.24	Prestazioni relative delle diverse funzioni di perdita (valori più alti indicano prestazioni migliori) con la perdita migliore, $\mathcal{L}_{ssitrim} + \mathcal{L}_{reg}$, utilizzata come riferimento. Tutte le quattro funzioni di perdita proposte (area bianca) superano le attuali funzioni di perdita allo stato dell'arte (area grigia).[5]	43
3.25	Immagini campione dal dataset 3D Movies.[5]	45

3.26	In alto: immagini di input. Al centro: mappe di profondità inversa previste da MiDaS. In basso: corrispondenti point cloud renderizzati da un nuovo punto di vista, renderizzate tramite Open3D [66]. Immagini di input dal dataset Microsoft COCO [47], che non è stato visualizzato durante l'addestramento.[5]	47
3.27	Confronto qualitativo di MiDaS ai quattro migliori concorrenti sulle immagini dal dataset Microsoft COCO [47]. Dall'alto verso il basso: immagini di input, Xian et al. [56], Wang et al. [57], Li et al. [65], Li & Snavely [54] e MiDaS [5].[5]	48
4.1	Esempio di immagine (sinistra) e maschere (destra) in FoodSeg103.	50
4.2	Risultati sul batch 1 del validation set. In alto le labels ed in basso le prediction del modello allenato.	56
4.3	Confusion matrix normalizzata modello YOLOv8 allenato per 200 epoche con batch size 8.	57
4.4	Precisione (accuracy) e perdita (loss) training Inception-v3 con: LR=0.0015, batch-size = 32 e usando data augmentation.	59
4.5	Confusion matrix Inception-v3 con: LR=0.0015, batch-size = 32 e usando data augmentation.	60
4.6	Precisione (accuracy) e perdita (loss) training ResNet50 con: LR=0.0015, batch-size = 64 e usando data augmentation.	61
4.7	Confusion matrix ResNet50 con: LR=0.015, batch-size = 64 e usando data augmentation.	61
4.8	Precisione (accuracy) e perdita (loss) training Inception-v3 con: LR=0.0015, batch-size = 64 e data augmentation.	63
4.9	Confusion matrix Inception-v3 con: LR=0.0015, batch-size = 32 e data augmentation.	64
4.10	Precisione (accuracy) e perdita (loss) training ResNet50 con: LR=0.0015 e batch-size = 64 senza data augmentation.	64
4.11	Confusion matrix ResNet50 con: LR=0.0015 e batch-size = 64 senza data augmentation.	65
4.12	Precisione (accuracy) e perdita (loss) training ResNet50 con: LR=0.0015, batch-size = 64 e data augmentation.	65
4.13	Confusion matrix ResNet50 con: LR=0.0015, batch-size = 64 e data augmentation.	66
4.14	Batch da 64 immagini del training set con trasformazioni applicate.	68
4.15	Risultato della funzione <i>lr_find</i> per il primo segmento.	69
4.16	Andamento della loss nel primo segmento.	70
4.17	Risultato della funzione <i>lr_find</i> per il secondo segmento.	71
4.18	Andamento della loss nel secondo segmento.	72
4.19	Risultato della funzione <i>lr_find</i> per il terzo segmento.	73

4.20	Andamento della loss nel terzo segmento.	74
4.21	Risultato della funzione <i>lr_find</i> per il quarto segmento.	75
4.22	Andamento della loss nel quarto segmento.	76
4.23	Andamento della loss nel quinto segmento.	77
4.24	Andamento della loss nel sesto segmento.	78
4.25	Confusion matrix sul test set.	80
5.1	Schermata iniziale all'avvio della demo.	82
5.2	Schermata di caricamento dell'immagine dal pc.	83
5.3	Schermata dopo aver caricato l'immagine con le classi predette. . .	84
5.4	Schermata dopo aver selezionato la classe.	85
5.5	Maschera del piatto ottenuta usando SAM.	86
5.6	Maschera del cibo ottenuta usando SAM.	87
5.7	Esempio di segmentazione manuale.	88
5.8	Schermata dopo aver selezionato le maschere del piatto e del cibo. .	90
5.9	Menù che consente di specificare le caratteristiche del piatto. . . .	91
5.10	Esempio di maschere di piatto (a sinistra) e cibo (a destra) passate in ingresso alla funzione che calcola il volume.	91
5.11	Esempio di depth map ottenuta usando MiDaS.	92
5.12	Esempio di contorno del piatto.	93
5.13	Schermata di attesa (sinistra) e nuova schermata che mostra il volume calcolato.	94

Acronyms

AI

Artificial Intelligence (intelligenza artificiale)

BN

Batch Normalization (normalizzazione batch)

CNN

Convolutional Neural Network (rete neurale convoluzionale)

GUI

Graphical User Interface (interfaccia grafica)

LR

Learning Rate

SAM

Segment Anything Model

TTA

Test Time Augmentation

YOLO

You Only Look Once

Capitolo 1

Introduzione

1.1 Analisi del problema

Il cibo è molto più di una semplice necessità biologica: è un pilastro fondamentale della nostra esistenza, intrecciato con la salute, le emozioni e la cultura. Attraverso il cibo, le persone celebrano tradizioni, esprimono la propria identità e costruiscono relazioni. Da sempre, ciò che mangiamo non solo soddisfa un bisogno primario, ma racconta anche chi siamo e da dove veniamo.

A livello culturale, il cibo è un elemento di coesione e identità. Ogni cucina tradizionale riflette la storia, il clima e le risorse di una determinata regione, trasformandosi in un patrimonio collettivo che viene tramandato di generazione in generazione. Le ricette tradizionali non sono solo piatti da consumare, ma rappresentano un modo per mantenere vivo il legame con le proprie radici e per condividere con gli altri un pezzo della propria cultura.

Oltre all'aspetto culturale, il cibo ha un ruolo centrale nella salute e nel benessere delle persone. Una dieta equilibrata e ricca di nutrienti è essenziale per prevenire malattie, migliorare l'energia e sostenere il funzionamento ottimale del corpo. Negli ultimi anni, è aumentata la consapevolezza sull'importanza di una corretta alimentazione, spingendo sempre più persone a scegliere alimenti salutari e a limitare il consumo di cibi trasformati e ricchi di zuccheri o grassi saturi.

Questo cambiamento è evidente anche sui social media, che hanno contribuito alla diffusione di una cultura alimentare più consapevole. Milioni di post condividono ricette sane, stili di vita sostenibili e consigli su come migliorare le abitudini alimentari. Gli hashtag come #food e #healthyfood accumulano centinaia di milioni di contenuti, dimostrando quanto il cibo sia centrale nella vita quotidiana delle persone.

Non si può ignorare, inoltre, il crescente interesse verso l'impatto ambientale delle scelte alimentari. Una dieta sostenibile, basata su prodotti stagionali, biologici e

locali, può ridurre significativamente l'impatto sull'ambiente e promuovere pratiche agricole più etiche. Ridurre lo spreco alimentare, privilegiare i legumi e i vegetali rispetto alle proteine animali e scegliere alimenti a chilometro zero sono scelte che possono avere effetti positivi non solo sulla salute personale, ma anche sul pianeta.

Tuttavia, con l'aumento della vita frenetica e l'accesso diffuso a fast food e cibi pronti, mantenere un'alimentazione equilibrata è una sfida. La mancanza di informazioni sui valori nutrizionali di molti alimenti preparati fuori casa rende più difficile controllare ciò che consumiamo, da qui l'importanza di educare le persone su come leggere le etichette alimentari e fare scelte consapevoli.

In conclusione, il cibo è un elemento che trascende la semplice funzione di nutrimento: è cultura, connessione, benessere e responsabilità. Investire nella conoscenza e nella diffusione di buone abitudini alimentari non significa solo migliorare la qualità della vita individuale, ma anche contribuire a costruire una società più sana e sostenibile.

1.2 Obiettivi della tesi

Il tracciamento dei valori nutrizionali dei cibi consumati rappresenta una sfida sempre più rilevante nella società contemporanea, soprattutto per chi desidera mantenere una dieta equilibrata e uno stile di vita sano. Per semplificare questo processo, si propone lo sviluppo di un'applicazione innovativa in grado di riconoscere, attraverso una fotografia o un'immagine fornita dall'utente, il cibo rappresentato e stimarne le quantità. Questi dati consentirebbero il calcolo automatico dei valori nutrizionali associati, offrendo alle persone uno strumento pratico e veloce per monitorare la propria alimentazione.

L'idea prende ispirazione da applicazioni già esistenti, come FatSecret [6] o MyFitnessPal [7], che permettono agli utenti di ottenere statistiche nutrizionali sui piatti consumati tramite l'inserimento manuale di testo relativo al nome del piatto e alle sue quantità o tramite la scansione del codice a barre del prodotto.

Tuttavia, il progetto qui descritto mira a superare questa modalità manuale, offrendo un sistema più intuitivo e automatizzato basato sul riconoscimento visivo del cibo.

Questo approccio sfrutta algoritmi avanzati di intelligenza artificiale e machine learning per sviluppare un classificatore capace di stimare con buona precisione gli ingredienti presenti in un piatto. Integrando tecnologie come il riconoscimento delle immagini, l'applicazione sarebbe in grado di analizzare visivamente i pasti, identificandone gli elementi principali e calcolandone i valori nutrizionali (come calorie, proteine, carboidrati e grassi).

Il valore aggiunto di un'applicazione di questo tipo è duplice: da un lato, semplifica il monitoraggio nutrizionale per gli utenti, rendendo l'esperienza più

immediata e accessibile; dall'altro, promuove una maggiore consapevolezza riguardo alle proprie scelte alimentari, incentivando uno stile di vita più sano e bilanciato.

Questa soluzione risponderebbe non solo alle esigenze individuali di chi vuole controllare la propria dieta, ma anche a quelle collettive legate alla salute pubblica, poiché un maggiore accesso a strumenti di tracciamento potrebbe contribuire alla prevenzione di malattie legate all'alimentazione, come obesità, diabete e patologie cardiovascolari.

Il progetto della tesi si concentra specificamente sullo studio e sulla realizzazione di un classificatore di immagini di cibi e la successiva stima quantitativa del suo volume, con l'obiettivo di rendere il monitoraggio dei valori nutrizionali automatico e semplice.

Questa applicazione, una volta sviluppata, potrebbe rappresentare un importante passo avanti nel settore della nutrizione digitale, fornendo una risposta concreta alle sfide di una società sempre più attenta al proprio benessere e alla sostenibilità delle proprie scelte alimentari.

1.3 Contributo della tesi

Per il raggiungimento di tali obiettivi vengono esaminati due diversi approcci:

- Il primo approccio utilizza il modello YOLOv8 [1] con l'obiettivo di integrare la classificazione delle immagini con la segmentazione degli alimenti;
- Il secondo approccio prevede due fasi distinte per la classificazione (utilizzando ResNet50 [2] o InceptionV3 [3]) e la segmentazione delle immagini (attraverso SAM [4]).

Il seguente lavoro di tesi è strutturato nel seguente modo:

1. Capitolo 2: vengono descritti i dataset usati nei vari esperimenti;
2. Capitolo 3: vengono analizzati i modelli che sono stati utilizzati nei due approcci e il modello di calcolo della profondità (MiDaS [5]) che viene utilizzato per la stima del volume;
3. Capitolo 3: vengono descritti i due approcci e i relativi risultati; analizzando le performance e i limiti di ciascun approccio.
4. Capitolo 4: viene presentato il funzionamento di un'applicazione demo che integra il lavoro svolto, guidando l'utente nella classificazione dell'immagine e nella stima del volume del cibo in essa contenuto.

Capitolo 2

Dataset

2.1 Food-101

Il dataset Food-101, proposto da Lukas Bossard, Matthieu Guillaumin e Luc Van Gool [8], è una risorsa pionieristica nel campo del riconoscimento di immagini di cibo.

Presentato nella conferenza ECCV 2014, è stato progettato per rispondere alle sfide del riconoscimento visivo in contesti reali, proponendo un benchmark ampio e diversificato.

Il dataset non solo introduce un nuovo standard per i test, ma evidenzia anche metodologie innovative, come l'approccio Random Forest Discriminant Components (RFDC), e pone le basi per sviluppi futuri nel campo [8].

2.1.1 Caratteristiche principali di Food-101

2.1.2 Composizione e dimensioni

Il dataset include 101 categorie di cibo, rappresentative di piatti comuni e popolari in vari contesti gastronomici globali.

Di seguito sono elencate le principali caratteristiche del dataset [8]:

- Numero totale di immagini: 101.000 immagini;
- Distribuzione delle immagini:
 - 750 immagini per classe nel training set, che includono intenzionalmente rumore per simulare condizioni del mondo reale.
 - 250 immagini per classe nel test set, curate manualmente per garantire una valutazione accurata delle prestazioni dei modelli.

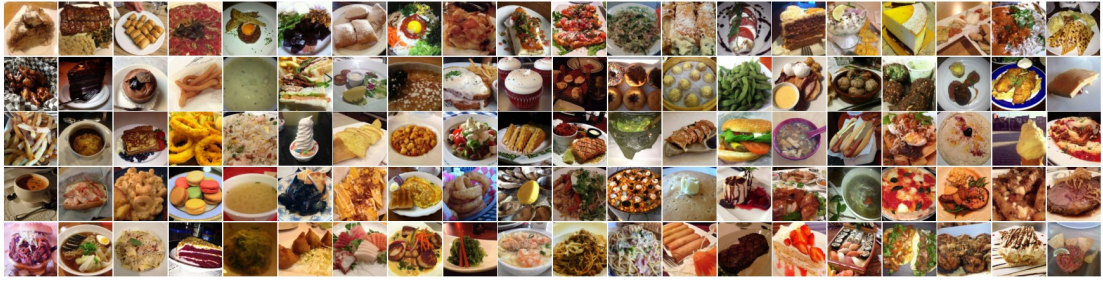


Figura 2.1: Un esempio per 100 delle 101 classi del dataset Food-101. Notare l'elevata varianza nel tipo di cibo, colore, esposizione e livello di dettaglio, ma anche tipi di cibo visivamente e semanticamente simili.[8]

Le immagini spaziano da piatti semplici come insalate e dessert a pietanze più elaborate come sushi e piatti a base di carne.

La vasta gamma di categorie include dessert (e.g., tiramisù, cheesecake), antipasti (e.g., seaweed salad, bruschetta), piatti principali (e.g., risotto, hamburger) e piatti etnici (e.g., bibimbap, pad thai) (Fig. 2.1).

L'elenco completo delle 101 classi è riportato nell'appendice A.

2.1.3 Origine e pre-elaborazione

Le immagini sono state raccolte da foodspotting.com (sito non più attivo), una piattaforma sociale in cui gli utenti condividono foto di cibo, annotate con il nome del piatto e il luogo.

Questa fonte garantisce una rappresentazione autentica, con immagini scattate in ambienti variabili, spesso non controllati.

La pre-elaborazione include:

- Ridimensionamento delle immagini con un lato massimo di 512 pixel.
- Esclusione delle immagini troppo piccole.
- Rumore intenzionale nei dati di training, con etichette talvolta errate o immagini sovraesposte.

2.1.4 Varietà e complessità

Uno dei punti di forza del Food-101 [8] è la sua diversità intra-classe:

- Illuminazione: le immagini presentano variazioni dovute a condizioni ambientali diverse;

- Presentazione: i piatti possono essere preparati e disposti in modi molto differenti;
- Prospettiva: gli angoli di scatto variano notevolmente.

Inoltre, alcune categorie condividono caratteristiche visive simili, come nel caso di onion rings e french fries, o tiramisù e carrot cake.

Questa complessità lo rende un benchmark ideale per valutare algoritmi di classificazione avanzati.

2.1.5 Confronto con dataset simili

Prima del rilascio di Food-101 PFID (Pittsburgh Fast-Food Image Dataset) [9] era l'unico dataset pubblicamente disponibile.

Il PFID è stato introdotto nel 2009 da Chen et al. è uno dei primi dataset specificamente progettati per il riconoscimento visivo di cibi.

Il PFID contiene immagini di cibi fast-food scattate in laboratorio e in ristorante.

Il dataset include 101 tipi di alimenti fast-food e sono stati raccolti tre esemplari di ciascun tipo di alimento, acquistati in date diverse dallo stesso ristorante o da filiali differenti della stessa catena di fast food.

Sebbene includa anch'esso 101 classi, il numero totale di immagini è di solo 4.545 rendendolo limitato rispetto al Food-101.

A questo si aggiunge il fatto che si concentra esclusivamente su alimenti fast-food, non rappresentando la diversità della cucina globale.

Inoltre, le immagini sono standardizzate, mentre il Food-101 presenta immagini realistiche e rumorose.

2.1.6 Applicazioni del Dataset

Il dataset si offre a diverse applicazioni come:

- Classificazione alimentare: il dataset può essere utilizzato per sviluppare app che riconoscono automaticamente cibi da foto (es. app di dieta o social network);
- Salute e nutrizione: il dataset può fungere da supporto per il monitoraggio calorico tramite analisi visiva;
- Ricerca accademica: il dataset viene ampiamente adottato come benchmark per valutare tecniche avanzate di visione artificiale.

2.2 FoodSeg103

FoodSeg103 [10] è un dataset di segmentazione delle immagini alimentari presentato nel 2021 come parte di uno sforzo accademico per colmare le lacune nei dataset esistenti relativi alla segmentazione delle immagini alimentari. Creato da ricercatori

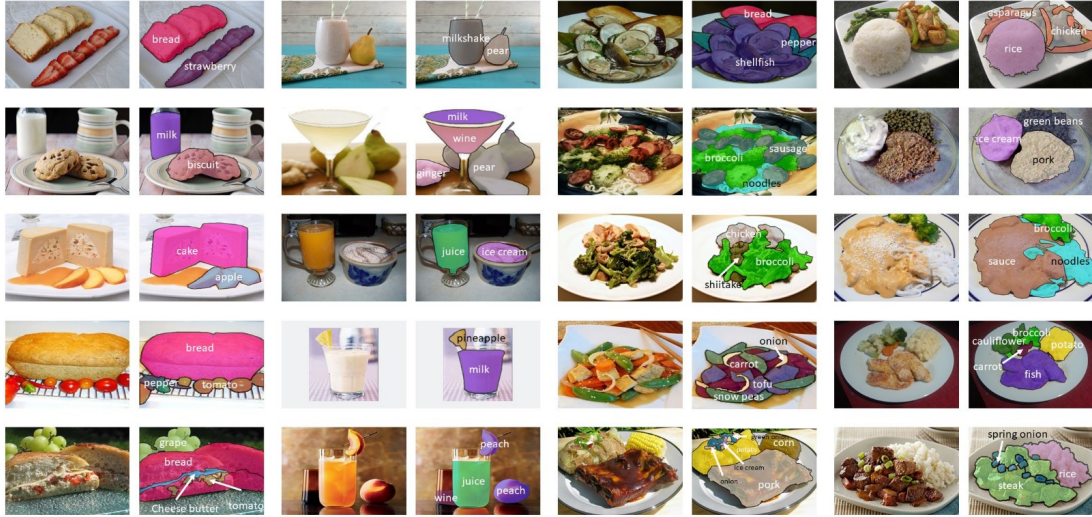


Figura 2.2: Esempi in FoodSeg103.[10]

di istituzioni come la Singapore Management University e il Salesforce Research Asia, il dataset è stato reso pubblico per promuovere ulteriori sviluppi nel campo della visione artificiale applicata agli alimenti. FoodSeg103 [10] è stato progettato per superare i limiti dei dataset esistenti, come UECFoodPix e UECFoodPixComplete [11, 12].

Esso introduce un livello di dettaglio senza precedenti nella segmentazione degli ingredienti alimentari, fornendo annotazioni a livello di ingrediente e maschere pixel-wise (Fig. 2.2).

Il dataset si concentra su una segmentazione fine, facilitando applicazioni come la stima nutrizionale e calorica degli alimenti.

2.2.1 Caratteristiche Principali di FoodSeg103

Di seguito sono elencate le caratteristiche principali di FoodSeg103 [10]:

- **Dimensioni e dettaglio:** comprende 7.118 immagini con 42.097 maschere e 103 categorie di ingredienti, derivati da Recipe1M [13]. La selezione garantisce visibilità chiara degli ingredienti e almeno due ingredienti per immagine.

La figura 2.3 presenta esempi di segmentazione mostrando la capacità del dataset di distinguere chiaramente gli ingredienti in scenari sia semplici che complessi, come il riconoscimento di pomodori sovrapposti o lattuga parzialmente visibile.

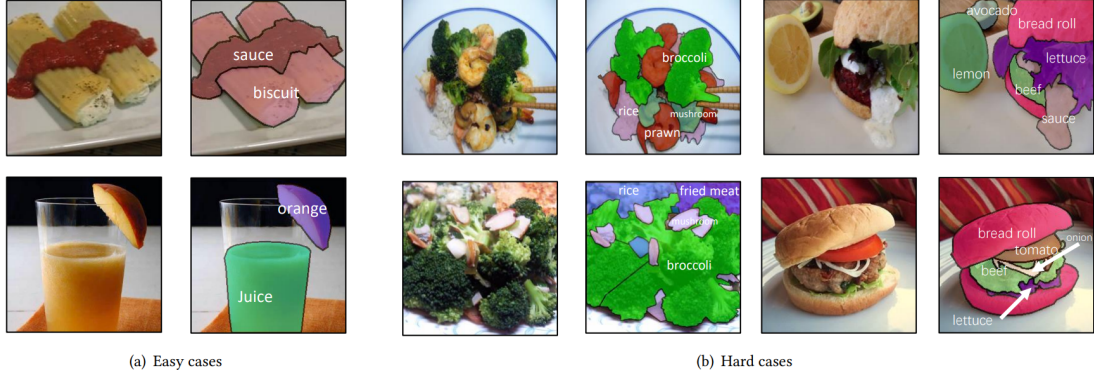


Figura 2.3: Esempi di FoodSeg103: a sinistra l’immagine sorgente a destra le annotazioni.[10]

- Annotazione Fine-Grained: le maschere sono a livello di ingrediente, a differenza di UECFoodPix o UECFoodPixComplete [11, 12], che fornisce maschere solo a livello di piatto (dish-level). Questo garantisce un’analisi più dettagliata e utile per la segmentazione (Fig. 2.4).
- Diversità degli ingredienti: include categorie comuni come pomodori, carote e lattuga, e ingredienti meno frequenti come edamame e funghi shiitake. Questa diversità migliora la capacità di testare modelli in scenari reali.

2.2.2 Comparazione con dataset precedenti

UECFoodPix e UECFoodPixComplete

UECFoodPix e UECFoodPixComplete [11, 12] offrono maschere dish-level per 10.000 immagini, ma non forniscono segmentazioni a livello di ingrediente. FoodSeg103 presenta più maschere (42.097 contro le 14.011 di UECFoodPix e le 16.069 di UECFoodPixComplete) e maggiore dettaglio grazie alle sue annotazioni fine-grained (Fig. 2.4).

Recipe1M

Sebbene Recipe1M [13] includa informazioni sugli ingredienti, non offre maschere di segmentazione. FoodSeg103 [10] arricchisce questi dati con annotazioni precise e

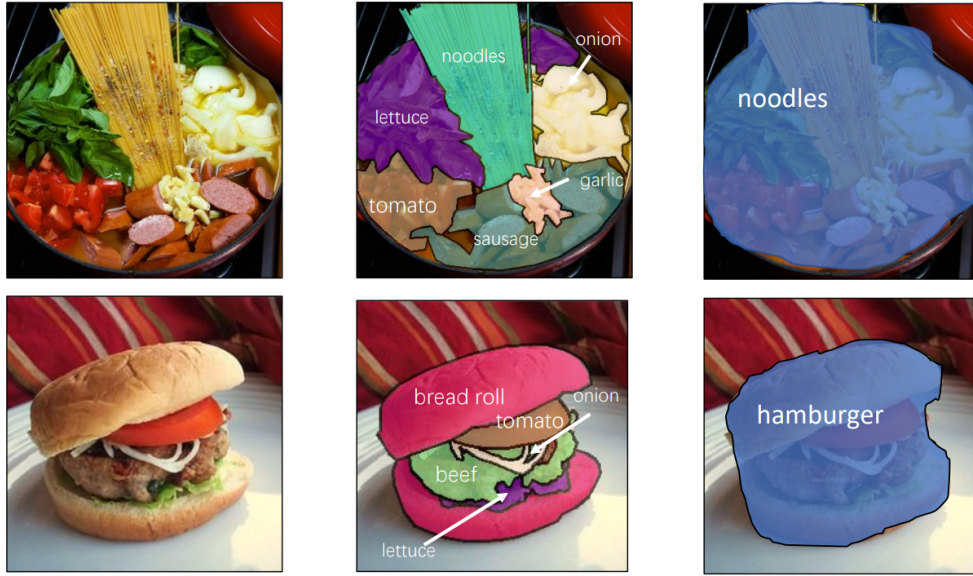


Figura 2.4: Confronto tra diversi stili di annotazione per mascherare le immagini di cibo: immagini sorgenti (a sinistra), annotazioni ingredient-level (al centro) e annotazioni dish-level (a destra).[10]

utili per la segmentazione.

2.2.3 Contributi principali

Dataset FoodSeg103 [10] fornisce un benchmark unico per la segmentazione degli alimenti con annotazioni dettagliate e una grande varietà di ingredienti.

I modelli di segmentazione semanticamente avanzati, come CCNet [14], SeTR [15] e FPN [16], dimostrano risultati significativamente inferiori su FoodSeg103 rispetto a dataset più semplici come Cityscapes [17], evidenziando la complessità delle immagini alimentari (Fig. 2.5).

In conclusione, FoodSeg103 rappresenta un passo avanti fondamentale nel campo della segmentazione delle immagini alimentari, superando i limiti dei dataset precedenti.

La sua granularità e la varietà di ingredienti permettono di sviluppare modelli più robusti e applicabili in contesti reali, come la stima nutrizionale e il monitoraggio della dieta.

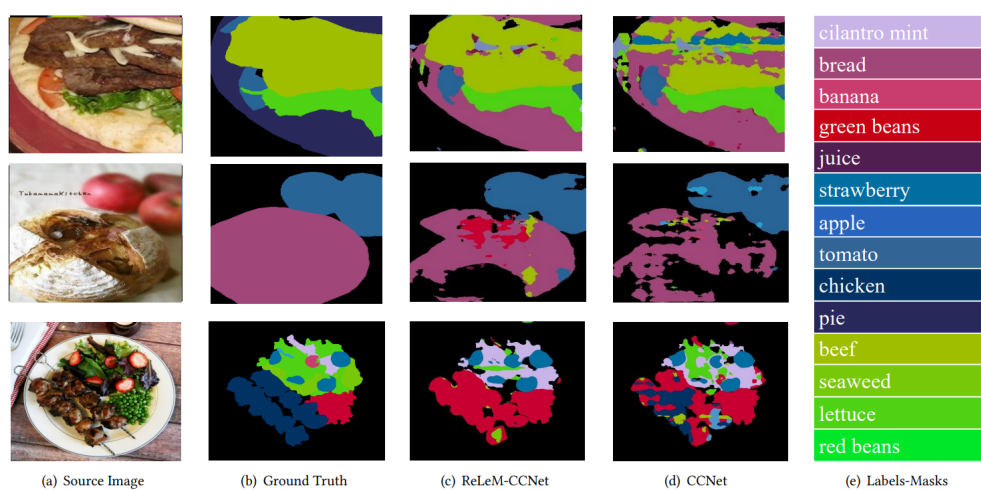


Figura 2.5: Esempi qualitativi dei risultati di ReLeM-CCNet e CCNet su FoodSeg103. Si può notare come un modello più complesso quale ReLeM-CCNet sia in grado di fornire predizioni più accurate.[10]

Capitolo 3

Modelli

3.1 Residual neural network

Una residual neural network [2] generalmente chiamata semplicemente residual network o ResNet è un'architettura di deep learning introdotta nel 2015 vincendo l'ImageNet Large Scale Visual Recognition Challenge [18] (ILSVRC) dello stesso anno.

Le ResNet rappresentano una delle innovazioni più significative nel campo delle reti neurali profonde, progettate per superare i limiti delle architetture tradizionali. Questi limiti emergono in particolare con l'aumento della profondità delle reti, elemento cruciale per ottenere rappresentazioni gerarchiche ricche, ma che introduce anche sfide notevoli.

3.1.1 Limiti delle reti classiche

Modelli come AlexNet [19] e VGGNet [20] hanno dimostrato l'efficacia delle reti profonde, ma con un costo computazionale elevato. Ad esempio, mentre la VGG-19 raggiunge buoni risultati, ha un costo computazionale di 19,6 miliardi di FLOPs, rendendola poco pratica per applicazioni più estese.

Problema della Degradazione

Uno dei problemi più significativi delle reti profonde è la degradazione delle prestazioni [21, 22]. Man mano che si aggiungono strati a una rete, ci si aspetterebbe un miglioramento dell'accuratezza grazie alla capacità di modellare funzioni più complesse.

Tuttavia:

- Le reti tradizionali (reti plain) mostrano un'accuratezza che si satura e in molti casi peggiora con l'aumento della profondità.

- Questo fenomeno non è dovuto a un overfitting, ma all'incapacità del modello di ottimizzare correttamente i parametri quando aumenta la profondità.

Tale andamento è evidenziato dai grafici sottostanti (Fig. 3.1).

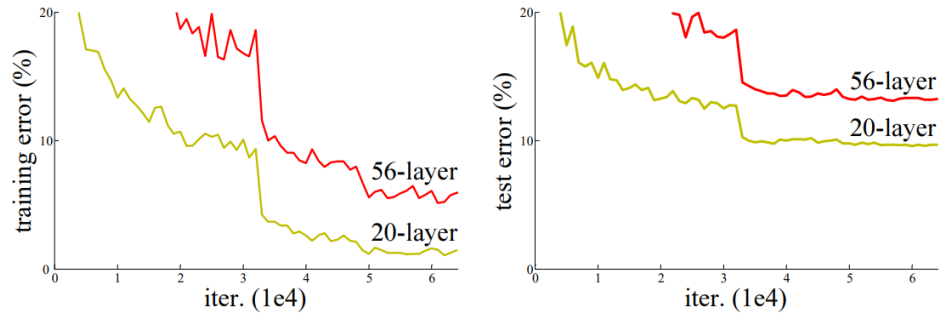


Figura 3.1: Training error (sinistra) e test error (destra) su CIFAR-10 [23] con 20-layer e 56-layer su reti “plain”. La rete più profonda ha un training error maggiore e pertanto un test error maggiore.[2]

Problemi di Ottimizzazione

Le reti tradizionali profonde soffrono di due problemi correlati alla propagazione del gradiente [24, 25, 26]:

- Gradiente che svanisce: nei livelli più bassi della rete, i gradienti possono diventare estremamente piccoli, impedendo un aggiornamento efficace dei pesi.
- Gradiente che esplode: nei livelli più alti, i gradienti possono diventare troppo grandi, causando instabilità numerica.

Sebbene tecniche come l’inizializzazione normalizzata [27, 26, 28, 29] e la normalizzazione batch (BN) [30] abbiano mitigato questi problemi, il fenomeno della degradazione rimane un ostacolo significativo per l’addestramento di reti molto profonde.

3.1.2 Soluzione offerta dalla ResNet

Per risolvere questi problemi, He et al. [2] hanno introdotto le ResNet, basate su due concetti chiave:

- Apprendimento residuo;
- Connessioni shortcut.

Apprendimento residuo

Le ResNet introducono l'idea di apprendere funzioni residue $F(x) = H(x) - x$, dove $H(x)$ è la mappatura target e x è l'input. Invece di apprendere $H(x)$ direttamente, il modello apprende la differenza tra $H(x)$ e x , ricostruendo il risultato come $F(x) + x$ (Fig. 3.2). Vantaggi:

- Semplifica l'apprendimento, poiché i livelli devono concentrarsi solo sulle deviazioni rispetto all'identità;
- Permette di modellare più facilmente funzioni complesse, migliorando la propagazione del gradiente;
- Anche se la mappatura ideale è l'identità, i solver possono semplicemente spingere i pesi verso lo zero per approssimarla, evitando il degrado.

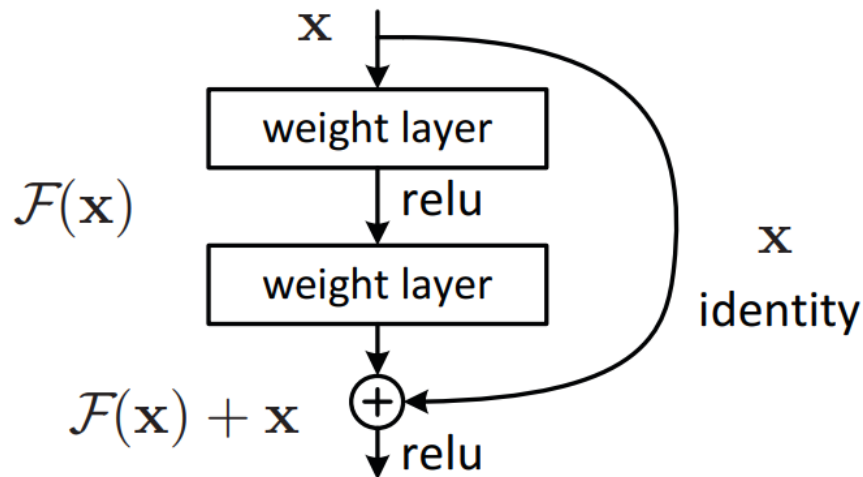


Figura 3.2: Residual learning: un building block.[2]

Connessioni shortcut

Le connessioni shortcut sono il fulcro delle ResNet e svolgono un ruolo fondamentale nella loro capacità di scalare a profondità estreme:

- Tipi di connessioni:
 - Identitarie: utilizzate quando input e output hanno le stesse dimensioni, aggiungono x all'output senza modifiche;

- Proiezione lineare: usano convoluzioni 1×1 per adattare le dimensioni quando queste differiscono.
- Impatto: le shortcut migliorano la propagazione del gradiente, riducendo il rischio di vanishing/exploding gradient.

L'elemento chiave di queste connessioni shortcut è che consentono il passaggio diretto delle informazioni e dei gradienti, migliorando l'ottimizzazione e la propagazione dei segnali anche in reti molto profonde.

3.1.3 Struttura della ResNet

Blocchi Residuali

L'architettura delle ResNet è composta da blocchi residuali, ciascuno dei quali integra convoluzioni, normalizzazione batch (BN) e attivazioni ReLU:

- Blocco Semplice: utilizzato in configurazioni come ResNet-18 e ResNet-34, è composto da due livelli convoluzionali 3×3 , seguiti da BN e ReLU. L'output del blocco viene sommato all'ingresso tramite una connessione shortcut.
- Blocco Bottle neck: utilizzato in configurazioni più profonde (es. ResNet-50, ResNet-101, ResNet-152), questo blocco riduce la complessità computazionale utilizzando:
 - Convoluzioni 1×1 per ridurre le dimensioni.
 - Convoluzioni 3×3 per elaborare le feature.
 - Convoluzioni 1×1 per ripristinare le dimensioni originali.

La differenza tra i due tipi di blocchi è sintetizzata nella Fig. 3.3.

Questi design, pur scalabili a profondità estreme (fino a oltre 1000 strati), mantengono una complessità computazionale contenuta.

Ad esempio, la ResNet-152 richiede 11,3 miliardi di FLOPs, significativamente meno della VGG-19 con 19,6 miliardi di FLOPs, ma garantisce una capacità rappresentativa molto maggiore [2, 20].

Operazioni di Downsampling

Il downsampling, necessario per ridurre le dimensioni delle feature map, viene implementato nei blocchi residuali attraverso convoluzioni con stride 2.

Nelle ResNet, il downsampling è posizionato strategicamente nei punti di transizione tra i livelli convoluzionali (ad esempio, tra conv2, conv3, conv4 e conv5), preservando così le informazioni chiave durante la riduzione delle dimensioni.

Nella Fig. 3.4 viene illustrata la struttura di una ResNet-34 confrontandola con il corrispondente modello "plain" e un VGG-19.

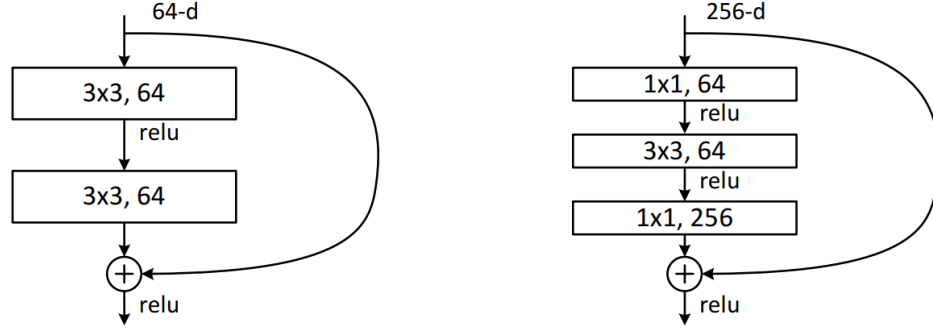


Figura 3.3: Una deeper residual function F per ImageNet[31]. A sinistra: un building block (su feature con mappe 56×56) come nella Fig. 3.4 per la ResNet34. A destra: un “bottleneck” building block per ResNet-50/101/152.[2]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112x112	7x7, 64, stride 2				
conv2_x	56x56	3x3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, & 64 \\ 3 \times 3, & 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, & 64 \\ 3 \times 3, & 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 256 \end{bmatrix} \times 3$
conv3_x	28x28	$\begin{bmatrix} 3 \times 3, & 128 \\ 3 \times 3, & 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, & 128 \\ 3 \times 3, & 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 512 \end{bmatrix} \times 8$
conv4_x	14x14	$\begin{bmatrix} 3 \times 3, & 256 \\ 3 \times 3, & 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, & 256 \\ 3 \times 3, & 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 1024 \end{bmatrix} \times 36$
conv5_x	7x7	$\begin{bmatrix} 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3, & 512 \\ 1 \times 1, & 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3, & 512 \\ 1 \times 1, & 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3, & 512 \\ 1 \times 1, & 2048 \end{bmatrix} \times 3$
	1x1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8	7.6×10^9	11.3×10^9

Tabella 3.1: Architetture per ImageNet[31]. I blocchi costitutivi sono mostrati tra parentesi (vedi anche la Fig 3.3), con il numero di blocchi impilati. Il downsampling viene eseguito da conv3_1, conv4_1 e conv5_1 con uno stride di 2.[2]

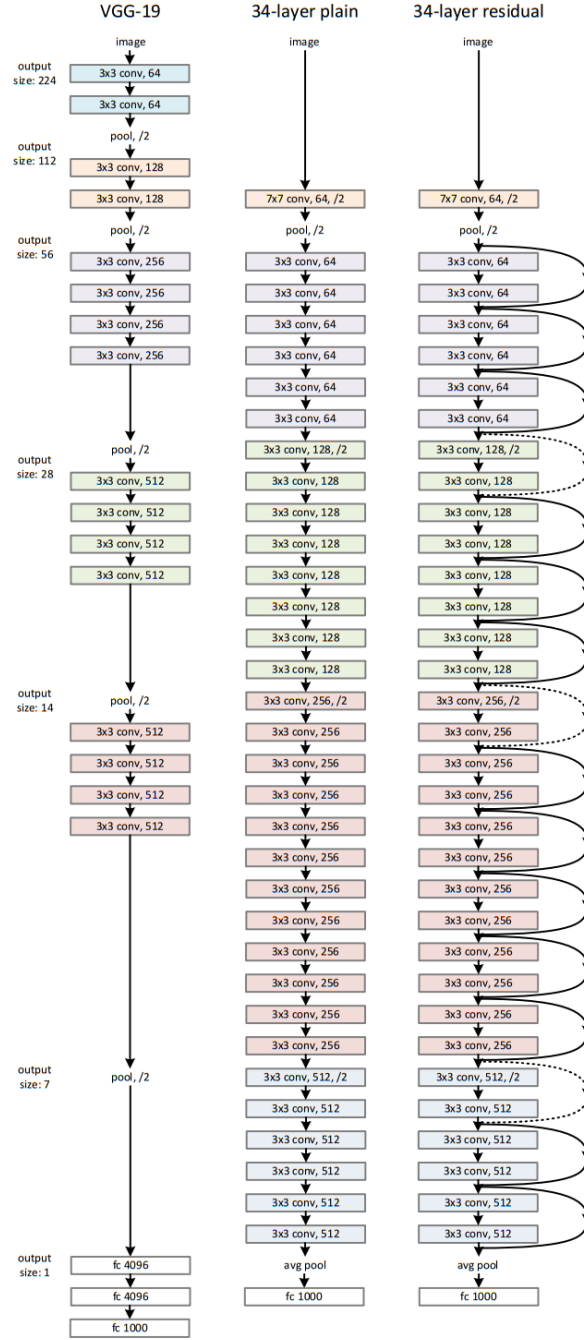


Figura 3.4: Esempio di architetture di rete per ImageNet[31]. A sinistra: il modello VGG-19 [20] (19,6 miliardi di FLOPs) come riferimento. Al centro: una rete plain con 34 strati parametrici (3,6 miliardi di FLOPs). A destra: una rete residual con 34 strati parametrici (3,6 miliardi di FLOPs). Le connessioni shortcut tratteggiate aumentano le dimensioni. La 3.1 mostra maggiori dettagli e altre varianti.[2]

3.2 Inceptionv3

L'architettura Inception è stata introdotta con il modello GoogLeNet durante la competizione ILSVRC 2014 [32], segnando un'importante svolta nel design delle reti neurali profonde per la visione artificiale.

Il focus dell'architettura Inception è sempre stato il bilanciamento tra efficienza computazionale e capacità rappresentativa, mantenendo un numero relativamente basso di parametri rispetto ad altri modelli dell'epoca come VGGNet[20] o AlexNet[19].

Di seguito sono riportate le varie versioni di Inception sino alla v3.

- **Inception-v1/GoogLeNet**

- Anno di introduzione: 2014 [32]
- Caratteristiche principali:
 - * Il modello combinava convoluzioni di dimensioni diverse (1×1 , 3×3 , 5×5) in un unico modulo, insieme al pooling. Questo approccio consentiva di catturare informazioni a diverse scale spaziali.
 - * Con soli 5 milioni di parametri, era significativamente più efficiente di AlexNet (60 milioni di parametri) e VGGNet (138 milioni di parametri), pur raggiungendo un'accuratezza superiore.

- **Inception-v2**

- Anno di introduzione: 2015 [3]
- Evoluzione dell'architettura originale:
 - * Introduzione della fattorizzazione delle convoluzioni per ridurre i costi computazionali.
 - * Utilizzo della normalizzazione batch (batch normalization) per migliorare la stabilità dell'addestramento e accelerare la convergenza.
 - * Riduzioni della griglia ottimizzate per preservare le informazioni durante la transizione tra i livelli.
 - * Il modello risultante era più profondo e performante rispetto a GoogLeNet, mantenendo l'efficienza.

- **Inception-v3**

- Anno di introduzione: 2015 [3], è un'ulteriore evoluzione che integra le innovazioni di Inception-v2 con ulteriori ottimizzazioni strutturali e tecniche di regolarizzazione.
- Obiettivo:

- * Migliorare ulteriormente la capacità rappresentativa senza incrementare significativamente i costi computazionali, spingendo al limite il design efficiente dei moduli Inception.

3.2.1 Caratteristiche principali di Inception-v3

L'architettura Inception-v3 introduce una serie di miglioramenti progettati per aumentare l'efficienza e le prestazioni rispetto alle versioni precedenti.

Di seguito, i principali avanzamenti:

1. Fattorizzazione in convoluzioni più piccole;
2. Fattorizzazione Spaziale in Convoluzioni Asimmetriche;
3. Utilizzo dei Classificatori Ausiliari;
4. Riduzione Efficiente della Dimensione della Griglia;
5. Regolarizzazione del modello tramite Label Smoothing.

Fattorizzazione in convoluzioni più piccole

Uno dei principali punti di forza del modello Inception-v1 era la generosa riduzione dimensionale.

Per migliorarlo ulteriormente, le convoluzioni più grandi nel modello sono state fattorizzate in convoluzioni più piccole.

L'obiettivo della fattorizzazione delle convoluzioni è ridurre il numero di connessioni/parametri senza diminuire l'efficienza della rete.

Due sequenze di convoluzioni 3×3 sostituiscono una convoluzione 5×5 (Fig. 3.5).

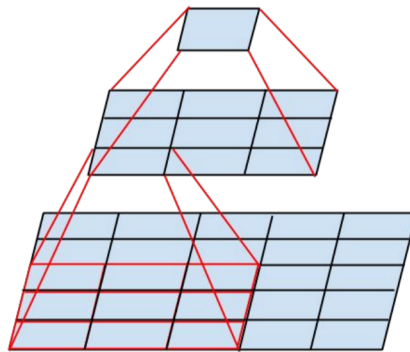


Figura 3.5: Mini-network che sostituisce le convoluzioni 5×5 . [32]

Usando 1 layer di filtri 5×5 il numero di parametri è 25, mentre usando due layer di filtri 3×3 il numero di parametri scende a 18 ($9+9$) con una conseguente riduzione dei parametri del 28%.

Il nuovo modulo Inception viene illustrato nella Fig.3.7.

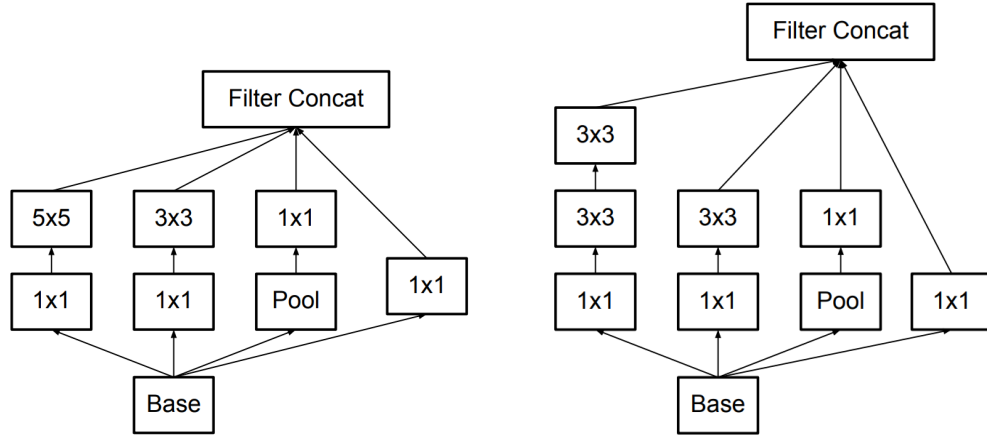


Figura 3.6: A sinistra il modulo Inception originale come proposto [32] a destra il nuovo modulo Inception dove ciascuna convoluzione 5×5 è sostituita da due convoluzioni 3×3 . [32]

Fattorizzazione Spaziale in Convoluzioni Asimmetriche

Le convoluzioni $n \times n$ vengono scomposte in due convoluzioni successive $1 \times n$ e $n \times 1$ (Fig. 3.7).

Questo riduce significativamente i costi computazionali senza perdita di capacità rappresentativa.

Per esempio usando una convoluzione 3×1 seguita da una 1×3 è equivalente a far scorrere una rete a due strati con lo stesso campo recettivo di una convoluzione 3×3 (Fig. 3.8). Usando 1 layer di filtri 3×3 il numero di parametri è 9, mentre usando un layer di filtri 3×1 e 1×3 il numero di parametri scende a 6 con una conseguente riduzione dei parametri del 33%.

Gli autori non hanno scelto due filtri 2×2 perchè il numero di parametri sarebbe stato 8 con una riduzione del solo 11%.

Il nuovo modulo Inception ottenuto usando questa tecnica è raffigurato nella Fig.3.10.

Questa fattorizzazione non funziona bene sui layer iniziali, ma dà buoni risultati su griglie di dimensioni medie (su feature map $m \times m$ dove m ha un valore compreso tra 12 e 20).

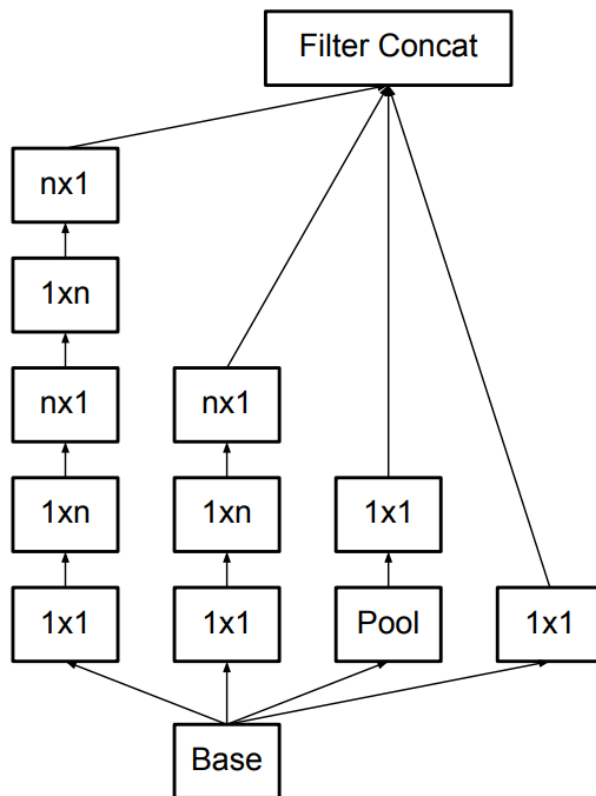


Figura 3.7: Inception modules dopo la fattorizzazione delle $n \times n$ convoluzioni.[32]

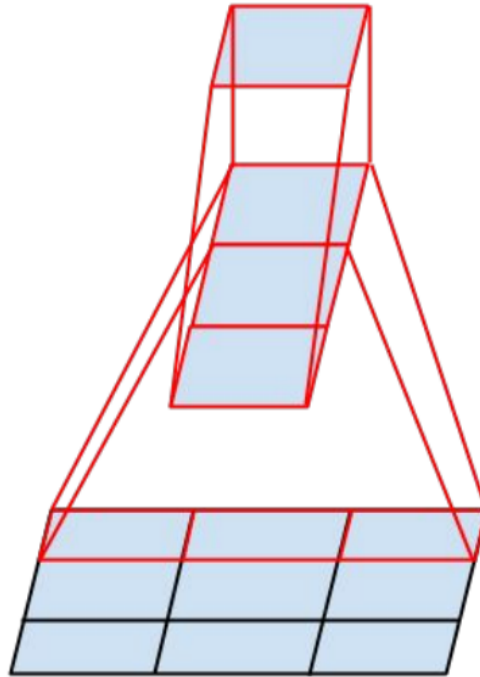


Figura 3.8: Mini-network che sostituisce le convoluzioni 3x3. Il livello inferiore di questa rete consiste in una convoluzione 3x1 con 3 unità di output.[32]

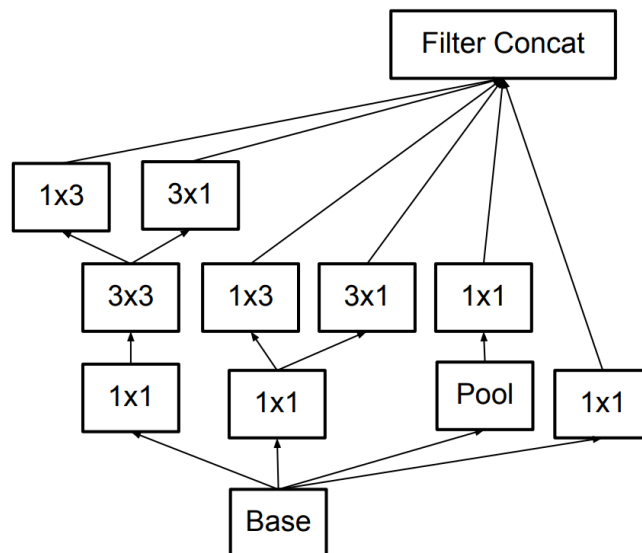


Figura 3.9: Modulo Inception con output del filtro espanso.[32]

Classificatori Ausiliari

I classificatori ausiliari erano già stati suggeriti in GoogLeNet/Inception-v1 [32]. Ci sono alcune modifiche in Inception-v3.

In Inception-v3, viene utilizzato un solo classificatore ausiliario sopra l'ultimo livello 17×17 , invece di utilizzare 2 classificatori ausiliari.

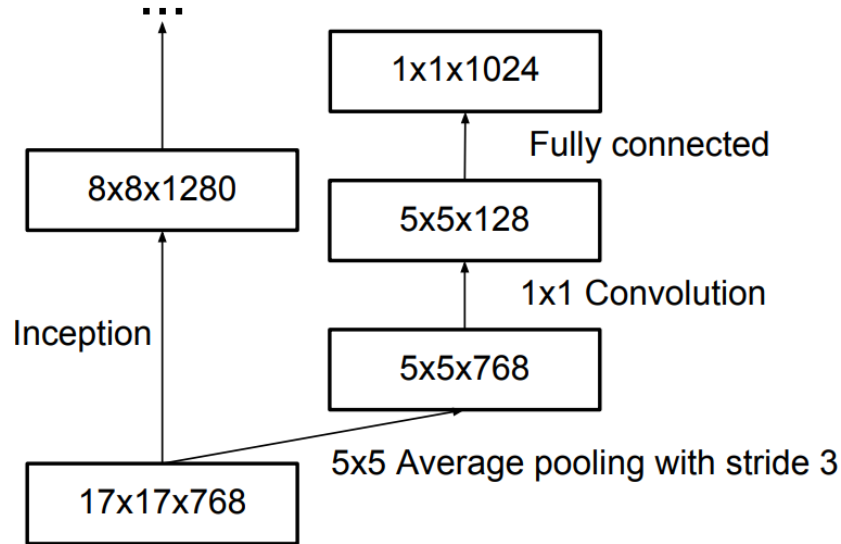


Figura 3.10: Modulo Inception con output del filtro espanso.[32]

Lo scopo è anche diverso: in GoogLeNet/ Inception-v1 [32], i classificatori ausiliari venivano utilizzati per avere una rete più profonda in risposta al problema della scomparsa del gradiente.

I classificatori ausiliari non hanno portato a miglioramenti nelle fasi iniziali dell'addestramento. Tuttavia, verso la fine, la rete con classificatori ausiliari ha mostrato una maggiore precisione rispetto alla rete senza classificatori ausiliari.

Pertanto, in Inception-v3, il classificatore ausiliario viene utilizzato come regolarizzatore.

La batch normalization [30], suggerita in Inception-v2, è anche utilizzata nel classificatore ausiliario.

Riduzione Efficiente della Dimensione della Griglia

Tradizionalmente, come in AlexNet[19] e VGGNet[20], la riduzione della mappa delle caratteristiche viene effettuata tramite max pooling.

Tuttavia, lo svantaggio è che, se seguito da un layer convoluzionale, il max pooling può essere troppo "avidio", oppure diventa troppo costoso se preceduto da un layer convoluzionale.

Qui, viene proposta una riduzione efficiente della dimensione della griglia come illustrato nella figura sottostante (Fig.3.11).

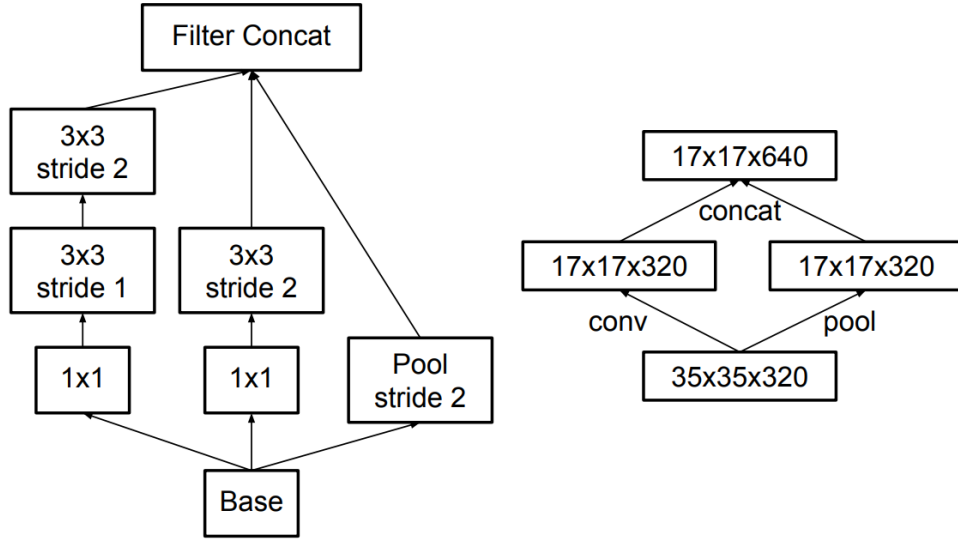


Figura 3.11: Modulo Inception che riduce la dimensione della griglia mentre espande i banchi di filtri.[32]

Con la riduzione efficiente della dimensione della griglia, 320 mappe delle caratteristiche vengono ottenute tramite una convoluzione con stride 2. Altre 320 mappe delle caratteristiche vengono ottenute tramite max pooling. Questi 2 set di mappe delle caratteristiche vengono concatenate per formare 640 mappe delle caratteristiche, che poi passano al livello successivo del modulo Inception.

Una rete meno costosa e comunque efficiente viene raggiunta grazie a questa riduzione efficiente della dimensione della griglia.

Regolarizzazione del modello tramite Label Smoothing

Il label smoothing è una tecnica di regolarizzazione impiegata durante l'addestramento di reti neurali, specificamente per ridurre l'overfitting e migliorare la generalizzazione del modello.

Funziona modificando la distribuzione delle etichette nel calcolo della funzione di perdita.

Sia $q(k)$ la distribuzione delle etichette per un esempio di addestramento x , dove k è l'indice dell'etichetta. Normalmente, $q(k)$ è un vettore one-hot, cioè $q(k) = 1$

per la classe corretta e $q(k) = 0$ per tutte le altre classi. La cross-entropy loss è definita come:

$$l = - \sum_{k=1}^K \log(p(k)q(k)) \quad (3.1)$$

dove $p(k)$ è la probabilità predetta per la classe k , e K è il numero totale di classi.

Nel label smoothing, la distribuzione delle etichette viene modificata sostituendo $q(k)$ con una distribuzione "smussata" $q'(k)$, che è una combinazione tra la distribuzione originale ground-truth e una distribuzione uniforme con pesi $1 - \epsilon$ e ϵ rispettivamente:

$$q'(k) = (1 - \epsilon)\delta_{k,y} + \epsilon/K \quad (3.2)$$

dove:

- $\delta_{k,y}$ è la funzione delta di Dirac, che è 1 per la classe corretta y e 0 per le altre,
- ϵ è il parametro di "smussante" (tipicamente un piccolo valore come 0.1),
- K è il numero totale di classi.

Il principale vantaggio del label smoothing è che impedisce al modello di diventare troppo sicuro delle sue previsioni ed aiuta a mantenere un comportamento di previsione più equilibrato, riducendo l'overfitting.

3.2.2 Struttura dell'architettura

Il modello finale ha una struttura simile a quella mostrata in Fig.3.12.

Con 42 layer di profondità, il costo di calcolo è solo circa 2,5 volte superiore a quello di GoogLeNet[32], e molto più efficiente di quello di VGGNet[20]. Le

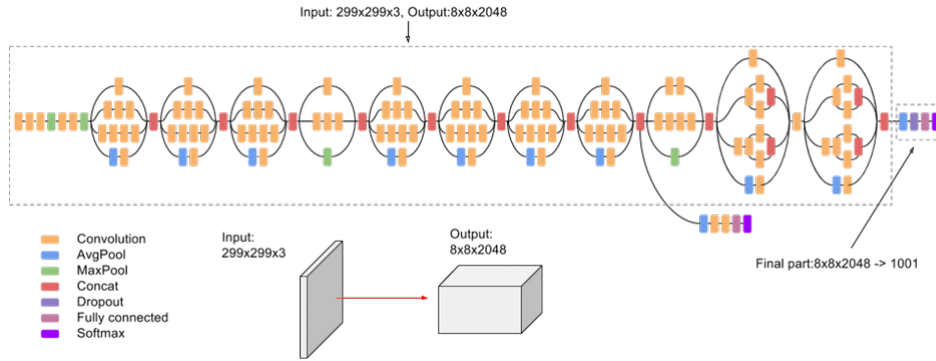


Figura 3.12: Architettura Inception-v3.[32]

componenti del modello Inception-v3 sono visibili in dettaglio nella tabella 3.2.

type	patch size/stride or remarks	input size
conv	3x3/2	299x299x3
conv	3x3/1	149x149x32
conv padded	3x3/1	147x147x32
pool	3x3/2	147x147x64
conv	3x3/1	73x73x64
conv	3x3/2	71x71x80
conv	3x3/1	35x35x192
3xInception	Come in Fig.3.6	35x35x288
5xInception	Come in Fig.3.7	17x17x768
2xInception	Come in Fig.3.10	8x8x1280
pool	8x8	8x8x2048
linear	logits	1x1x2048
softmax	classifier	1x1x1000

Tabella 3.2: La struttura del modello Inception-v3. La dimensione dell'output di ogni modulo è la dimensione dell'input del successivo.[32]

3.2.3 Prestazioni di Inception-v3

Il modello presentato si impose nel 2015 stabilendo un nuovo stato dell'arte, con una complessità computazionale contenuta rispetto ai modelli concorrenti.

Di seguito sono riportati i risultati su ImageNet[31]:

- Errore Top-1: 21,2% (single crop).
- Errore Top-5: 5,6% (single crop), 3,5% (multi-crop).

3.3 YoloV8

L'object detection è una delle discipline centrali della visione artificiale, consentendo ai computer di identificare e localizzare oggetti all'interno di immagini o video. Questo compito è fondamentale in numerose applicazioni, tra cui la guida autonoma, la videosorveglianza, il monitoraggio industriale e l'assistenza/diagnosi medica.

L'efficacia di un sistema di rilevamento oggetti si basa sulla sua capacità di bilanciare precisione, velocità e semplicità di implementazione. YOLO (You Only Look Once) ha rivoluzionato questo campo introducendo un approccio unificato che coniuga velocità e accuratezza.

YOLO, proposto per la prima volta nel 2015 da Joseph Redmon et al. [33], introduce un approccio innovativo che ridefinisce il paradigma dell'object detection.

Prima della sua comparsa, i metodi dominanti si basavano su architetture come R-CNN (Region-Based Convolutional Neural Network)[34], che dividevano il compito in più fasi: selezione delle regioni rilevanti (proposals), estrazione delle caratteristiche e classificazione. Questi approcci, pur accurati, erano computazionalmente intensivi e inadatti per applicazioni in tempo reale.

YOLO si distingue per la sua semplicità ed efficienza. Esso tratta il problema come un'unica regressione end-to-end: il modello esamina l'immagine una sola volta (one pass) per prevedere simultaneamente le coordinate dei bounding box, la classe di ogni oggetto e la probabilità associata.

Questo approccio consente una velocità di elaborazione significativamente maggiore, aprendo la strada ad applicazioni in tempo reale come la videosorveglianza, i veicoli autonomi e la realtà aumentata.

3.3.1 Evoluzioni delle Versioni YOLO

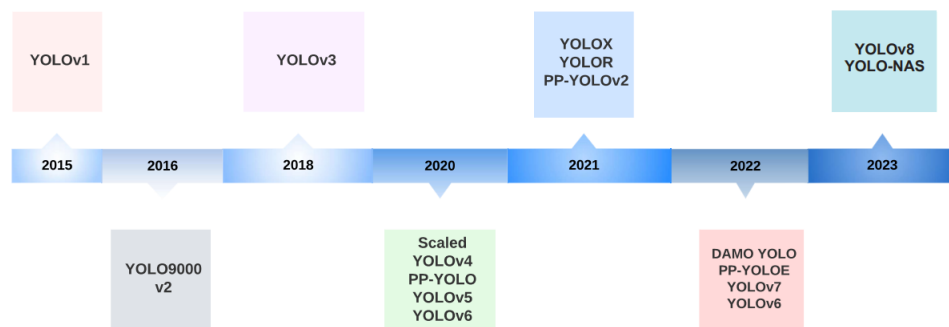


Figura 3.13: Timeline versioni di YOLO.[35]

YOLOv1 (2015)

YOLOv1 fu il primo tentativo di creare un modello unificato per la detection. L'architettura era composta da 24 strati convoluzionali seguiti da due strati completamente connessi (fully connected) come illustrato nella Fig. 3.14. Pur essendo rivoluzionario, presentava alcune limitazioni:

- Scarsa precisione per oggetti piccoli: la divisione dell'immagine in una griglia uniforme penalizzava oggetti di dimensioni ridotte.
- Problemi di localizzazione: la semplificazione della funzione di perdita portava a predizioni meno accurate delle coordinate delle bounding box.

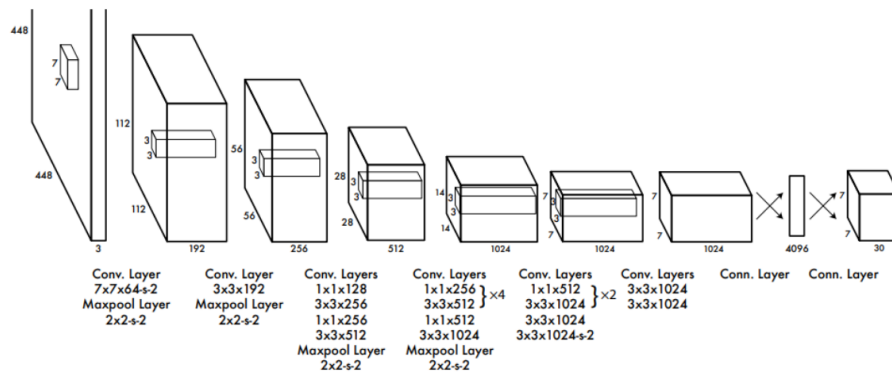


Figura 3.14: Architettura YOLOv1.[33]

YOLOv2 e YOLOv3 (2016-2018)

Queste versioni hanno apportato miglioramenti fondamentali:

- Introduzione degli anchor boxes, che definiscono a priori diverse dimensioni di oggetti.
- Adozione di un backbone più potente (Darknet-19 per YOLOv2 e Darknet-53 per YOLOv3), con l'inclusione di connessioni residuali per migliorare la propagazione del gradiente.
- Supporto multi-scala per rilevare oggetti di diverse dimensioni utilizzando feature maps da più livelli della rete.

YOLOv4 e YOLOv5 (2020)

Con YOLOv4 e YOLOv5, il focus si è spostato su:

- Efficienza computazionale: riduzione della complessità senza sacrificare precisione, con l'adozione di tecniche come CSPNet (Cross Stage Partial Network);
- Aggiunta di moduli come Spatial Pyramid Pooling (SPP) e Path Aggregation Network (PAN), migliorando la generalizzazione multi-scala e la rappresentazione semantica.

YOLOv8 (2023)

YOLOv8 [1] perfeziona ulteriormente il modello combinando le migliori idee delle versioni precedenti con nuove innovazioni per risolvere le sfide attuali del rilevamento in tempo reale.

L'obiettivo del nuovo modello è quello di massimizzare la precisione mantenendo velocità e semplicità.

Di seguito sono elencate le principali nuove caratteristiche:

1. Approccio anchor-free, che semplifica il training e migliora la flessibilità.
2. Backbone CSPNet avanzato per estrarre caratteristiche più ricche.
3. Neck FPN+PAN per una rappresentazione multi-scala più efficace.
4. Ottimizzazioni per il training, come il mixed precision training e tecniche avanzate di data augmentation.

3.3.2 Architettura di YOLOv8

YOLOv8 rappresenta un significativo passo avanti rispetto alle versioni precedenti, grazie a una serie di innovazioni architettoniche che migliorano la precisione, la scalabilità e l'efficienza. L'architettura di YOLOv8 è suddivisa in tre parti principali (Fig. 3.15):

1. Backbone
2. Neck
3. Head

Backbone

YOLOv8 utilizza una versione aggiornata di CSPDarknet53, arricchita con il modulo C2f (Cross Stage Partial with Feature Fusion). Questo modulo introduce connessioni parziali a diversi stadi della rete e fusioni delle caratteristiche, garantendo un flusso più efficiente delle informazioni e riducendo la ridondanza computazionale [35]. Novità rispetto a YOLOv5: C2f ottimizza il riutilizzo delle caratteristiche, migliorando l'efficienza senza sacrificare la precisione.

Neck

Collega il backbone alla testa del modello, aggregando le informazioni dalle diverse scale dell'immagine.

YOLOv8 utilizza una combinazione di Feature Pyramid Network (FPN) e Path Aggregation Network (PAN) [1]. Questa configurazione migliora la capacità del modello di rilevare oggetti di dimensioni diverse combinando informazioni dalle scale più basse e più alte. Questa combinazione rappresenta un miglioramento significativo rispetto al solo PAN utilizzato in YOLOv5, rendendo YOLOv8 più efficace nel rilevare oggetti di dimensioni variabili.

Head

La testa del modello genera le previsioni finali, ovvero le coordinate dei bounding box, le probabilità di classe e i punteggi di confidenza.

YOLOv8 abbandona i tradizionali anchor boxes, introducendo un meccanismo di rilevazione anchor-free [1]. Invece di prevedere un offset rispetto a box predefiniti, il modello identifica direttamente il centro dell'oggetto e ne calcola le dimensioni. Questo approccio ha numerosi vantaggi:

- Semplificazione del training;
- Riduzione del numero di iperparametri;
- Migliore adattabilità a oggetti con proporzioni variabili.

3.3.3 Nuova Funzione di Perdita (Loss Function)

YOLOv8 introduce una funzione di perdita migliorata, composta da tre componenti principali [1]:

- Focal Loss: migliora la classificazione assegnando più peso agli esempi difficili;
- IoU Loss: raffina la localizzazione dei bounding box;
- Objectness Loss: concentra l'attenzione del modello sulle aree contenenti oggetti.

3.3.4 Ottimizzazioni e prestazioni

YOLOv8 include ottimizzazioni per accelerare il training e migliorare la generalizzazione [1]:

- Data Augmentation Avanzato: Mosaic e Mixup per esporre il modello a scenari complessi;

- Mixed Precision Training: utilizza calcoli a precisione ridotta per accelerare il training, ridurre l'uso di memoria e consentire batch più grandi.

Le metriche utilizzate per valutare le prestazioni di YOLOv8 includono [1]:

1. Mean Average Precision (mAP): misura l'accuratezza del rilevamento degli oggetti su tutte le classi, con valori più alti che indicano una migliore performance;
2. Inference Time: valuta la velocità del modello nel processare le immagini;
3. Training Time: misura l'efficienza del processo di addestramento, evidenziando la rapidità con cui il modello può essere addestrato per raggiungere prestazioni ottimali;
4. Dimensioni del modello: indica le risorse di calcolo richieste per l'implementazione, con dimensioni del modello più piccole che risultano vantaggiose per l'implementazione su dispositivi con memoria e potenza di elaborazione limitate.

Nella tabella 3.3 vi è riportata una comparazione tra le performance metrics di YOLOv5 e YOLOv8.

Metrics	Yolov5	Yolov8
mAP@0.5	50.5%	55.2%
Inference Time	30 ms/image	25 ms/image
Training Time	12 hours	10 hours
Model Size	14 MB	12 MB

Tabella 3.3: Confronto delle performance metrics di YOLOv5 e YOLOv8 [37, 38].[1]

Queste metriche sono fondamentali per valutare i vantaggi pratici di YOLOv8 rispetto alle versioni precedenti come YOLOv5.

Valori mAP più elevati e tempi di inferenza più bassi si traducono direttamente in un rilevamento degli oggetti più accurato e veloce, rendendo YOLOv8 particolarmente adatto per applicazioni in cui l'elaborazione in tempo reale e la precisione sono essenziali [1].

Inoltre, tempi di formazione ridotti e dimensioni del modello suggeriscono che YOLOv8 è più efficiente da distribuire e mantenere, il che è vantaggioso sia per le applicazioni di ricerca che per quelle industriali.

3.3.5 Modelli di YOLOv8

YOLOv8 offre cinque varianti per adattarsi a diversi scenari:

La seguente tabella 3.4 presenta una panoramica completa delle varianti del modello YOLOv8, tra cui il numero di parametri, l'accuratezza in termini di precisione media (mAP@0,5) e tempi di inferenza su entrambe le piattaforme CPU e GPU per una dimensione di immagine standard di 640 pixel.

Model	Params (Million)	Accuracy (mAP@0.5)	CPU Time (ms)	GPU Time (ms)
YOLOv8n	2.0	47.2	42	5.8
YOLOv8s	9.0	58.5	90	6.0
YOLOv8m	25.0	66.3	210	7.8
YOLOv8l	55.0	69.8	400	9.8
YOLOv8x	90.0	71.5	720	11.5

Tabella 3.4: Performance metrics dei modelli YOLOv8.[1]

3.3.6 Formato di annotazione

Un ultimo aspetto da analizzare è il sistema di annotazioni usato da YOLOv8 che utilizza un formato basato su TXT PyTorch[39] di YOLOv5.

Le annotazioni sono archiviate in un file di testo in cui ogni riga corrisponde a un oggetto nell'immagine.

Ogni riga contiene l'etichetta di classe seguita dalle coordinate normalizzate del riquadro di delimitazione (center_x, center_y, width, height) relative alle dimensioni dell'immagine.

Il formato è il seguente:

<class> <center_x> <center_y> <width> <height>

Ad esempio, un'annotazione potrebbe apparire così:

0 0,492 0,403 0,212 0,315

Questo formato è accompagnato da un file di configurazione YAML, che specifica l'architettura del modello e le etichette di classe.

Per una gestione e annotazione efficiente dei dati, Ultralytics, gli sviluppatori di YOLOv8, consigliano di utilizzare Roboflow[40] come strumento di etichettatura compatibile.

3.4 SAM

Il progetto Segment Anything (SA) [4] introduce un approccio rivoluzionario per la segmentazione delle immagini, proponendo un nuovo task, un modello e un

dataset progettati per migliorare le capacità di generalizzazione zero-shot nella segmentazione.

L'obiettivo principale è costruire un modello fondamentale per la segmentazione che possa essere adattato a nuovi task e distribuzioni di dati attraverso il semplice utilizzo di prompt. Questo approccio si ispira ai successi dei modelli fondamentali nel NLP, come evidenziato in [10, 21].

3.4.1 Task: Segmentazione Promptable

Il cuore del progetto è il task di segmentazione "promptable", in cui il modello genera una maschera di segmentazione valida in risposta a un input flessibile chiamato prompt.

I prompt possono includere informazioni spaziali come punti o box, maschere predefinite o persino testo libero. Questo approccio garantisce che, anche in caso di ambiguità (ad esempio, un punto su una maglietta potrebbe riferirsi alla maglietta stessa o alla persona che la indossa), il modello produca una maschera ragionevole per almeno uno degli oggetti rilevanti.

3.4.2 Architettura del Modello

L'architettura di Segment Anything Model (SAM) [4], progettata per bilanciare flessibilità, efficienza e capacità di gestione dell'ambiguità, è composta da tre moduli principali (Fig. 3.16):

1. Encoder di immagine (image encoder);
2. Encoder di prompt (prompt encoder);
3. Decoder di maschere (mask decoder).

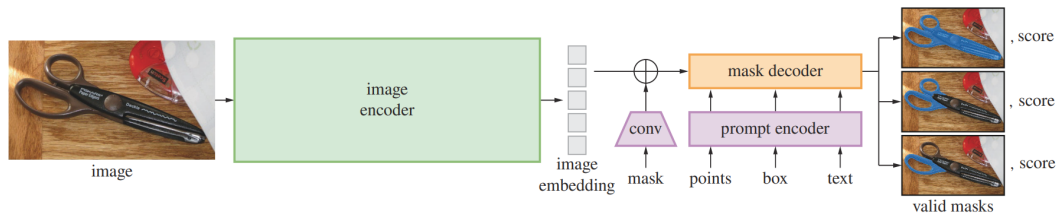


Figura 3.16: Architettura di SAM.[4]

Encoder di immagine

L'encoder di immagine (image encoder) è un componente fondamentale che trasforma un'immagine in una rappresentazione ricca e scalabile.

L'encoder utilizza un Vision Transformer (ViT) [41] pre-addestrato con la tecnica Masked Autoencoder (MAE) [42]. Questo approccio garantisce che l'encoder produca embedding di alta qualità, ottimizzati per una varietà di distribuzioni di dati. Caratteristiche Chiave:

- Supporta input ad alta risoluzione;
- Calcola l'embedding una sola volta per immagine, rendendo l'elaborazione successiva più efficiente (amortizzazione dei costi computazionali).

Encoder di prompt

L'encoder di prompt gestisce (Fig. 3.17) input di tipo sparso e denso, traducendo le istruzioni fornite dall'utente o dal sistema in embedding compatibili con l'encoder di immagini:

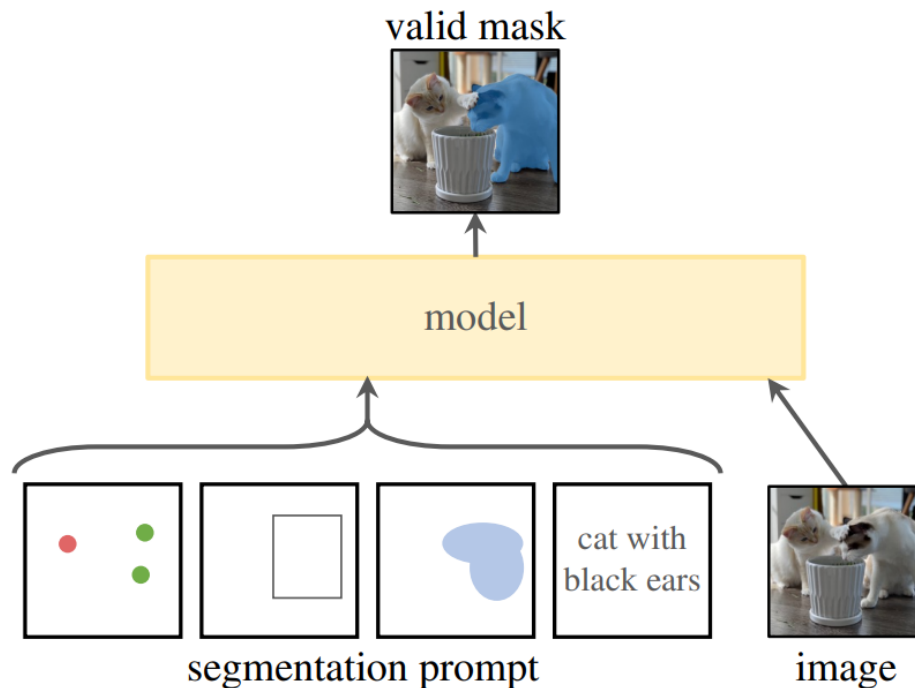


Figura 3.17: Encoder di prompt.[4]

- Prompt Sparsi: include punti (es. clic su un oggetto), box (rettangoli delimitanti) e testo. Questi vengono codificati utilizzando:
 - Encodings Posizionali: per punti e box;
 - Encoder di Testo CLIP [43]: per prompt testuali.
- Prompt Densi: come maschere già definite. Questi sono codificati tramite convoluzioni e combinati element-wise con l'embedding dell'immagine.

Decoder di maschere

Il decoder di maschere combina gli embedding generati dall'encoder di immagini e dal prompt encoder per produrre le maschere di segmentazione.

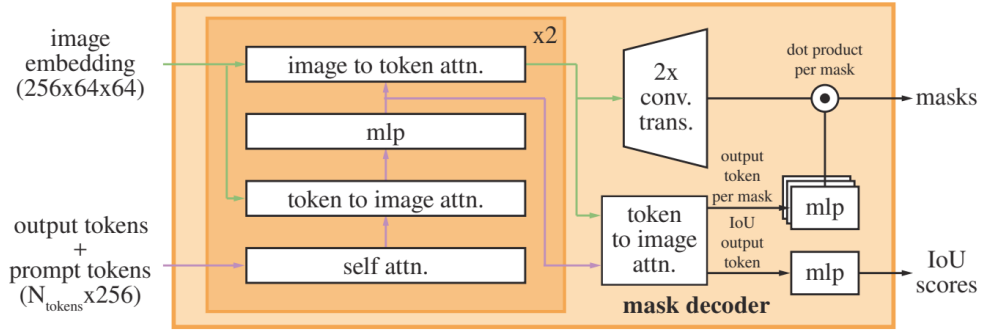


Figura 3.18: Decoder di maschere (mask decoder).[4]

Il decoder si basa su una variante del Transformer Decoder Block [44] (Fig. 3.18), che utilizza:

- Self-Attention dei Prompt per affinare la rappresentazione del prompt;
- Cross-Attention Immagine-Prompt per combinare le informazioni dell'immagine e del prompt.

L'output finale è calcolato con un classificatore dinamico lineare che genera una probabilità di appartenenza al foreground per ogni posizione nell'immagine.

Per gestire l'ambiguità, il decoder può generare più maschere (di default 3), ciascuna con un punteggio di confidenza.

L'intero processo, escluso l'encoder di immagini, è ottimizzato per girare su un browser web in ~50ms.

Gestione dell'Ambiguità

SAM adotta una strategia unica per affrontare prompt ambigui, come nel caso di un punto che potrebbe riferirsi a più oggetti [4].

Invece di restituire una sola maschera, il modello produce fino a 3 maschere valide (Fig. 3.19).



Figura 3.19: Ogni colonna mostra 3 maschere valide generate da SAM a partire da un singolo prompt ambiguo rappresentato da un punto (cerchio verde).[4]

Durante l'addestramento, si utilizza la minima perdita tra queste maschere rispetto al ground truth, permettendo al modello di apprendere rappresentazioni

utili per più contesti.

3.4.3 Dataset SA-1B e Motore di Dati

Per addestrare SAM, è stato creato il dataset SA-1B [4], che include oltre 1 miliardo di maschere generate automaticamente da 11 milioni di immagini ad alta risoluzione. La raccolta è avvenuta in tre fasi (Fig. 3.20):

- Annotazione assistita manuale: annotatori esperti, supportati da SAM, hanno creato maschere interattive.
- Fase semi-automatica: SAM ha generato maschere automatiche per oggetti prominenti, consentendo agli annotatori di concentrarsi su oggetti più complessi.
- Annotazione completamente automatica: SAM ha generato maschere per tutte le immagini utilizzando una griglia regolare di punti di foreground e selezionando le maschere più stabili.

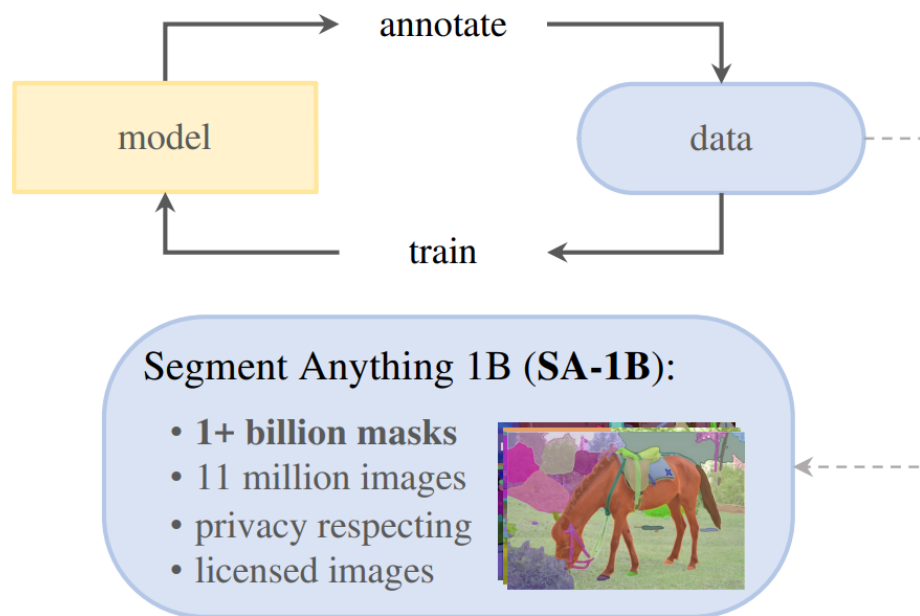


Figura 3.20: Motore dei dati (in alto) e dataset (in basso).[4]

Il dataset risultante (SA-1B) contiene 11 volte più immagini e 400 volte più maschere rispetto al secondo dataset più grande, Open Images [45], garantendo una diversità e una copertura senza precedenti (Fig. 3.21).

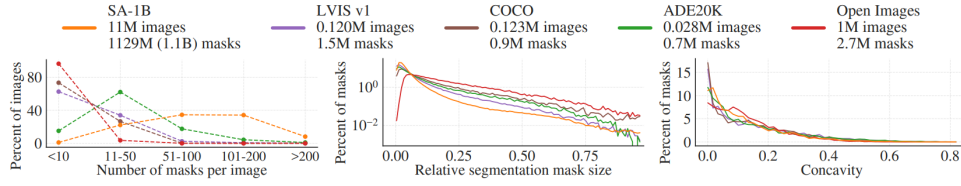


Figura 3.21: Proprietà delle maschere del dataset. La legenda fa riferimento al numero di immagini e maschere in ciascun dataset (LVIS v1 [46], COCO [47], ADE20K [48] e Open Images [45]). i noti che SA-1B contiene 11 volte più immagini e 400 volte più maschere rispetto al più grande dataset di segmentazione esistente, Open Images.[4]

3.4.4 Prestazioni e Applicazioni Zero-Shot

SAM è stato testato su un'ampia gamma di compiti e dataset mai visti durante l'addestramento, dimostrando capacità notevoli in scenari di zero-shot transfer:

- Segmentazione da un punto: SAM ha generato maschere valide da un singolo punto con prestazioni competitive rispetto ai metodi interattivi esistenti.
- Rilevamento di contorni: utilizzando un grid di punti e filtri Sobel, SAM ha prodotto mappe dei contorni accurate senza essere specificamente addestrato per questo compito (Fig. 3.22 e Tabella 3).



Figura 3.22: Predizione dei contorni in modalità zero-shot su BSDS500 [49]. SAM non è stato addestrato per prevedere mappe dei contorni né ha avuto accesso alle immagini o alle annotazioni di BSDS durante l'addestramento.[4]

- Generazione di Proposte di Oggetti: SAM ha generato maschere per oggetti di medie e grandi dimensioni con prestazioni comparabili a modelli supervisionati come ViTDet-H [50] (tabella 3.5).

method	all	small	med.	large	freq.	com.	rare
ViTDet-H [50]	63.0	51.7	80.8	87.0	63.1	63.3	58.3
<i>zero-shot transfer methods:</i>							
SAM-single out.	54.9	42.8	76.7	74.4	54.7	59.8	62.0
SAM	59.3	45.5	81.6	86.9	59.1	63.9	65.8

Tabella 3.5: Generazione di proposte di oggetti su LVIS v1 [46]. SAM è applicato in modalità zero-shot, ovvero non è stato addestrato per la generazione di proposte di oggetti né ha avuto accesso alle immagini o alle annotazioni di LVIS.[4]

- Segmentazione da testo: come proof-of-concept, SAM ha dimostrato la capacità di segmentare oggetti basandosi su prompt testuali, combinati con punti di riferimento aggiuntivi in caso di fallimento (Fig. 3.23).

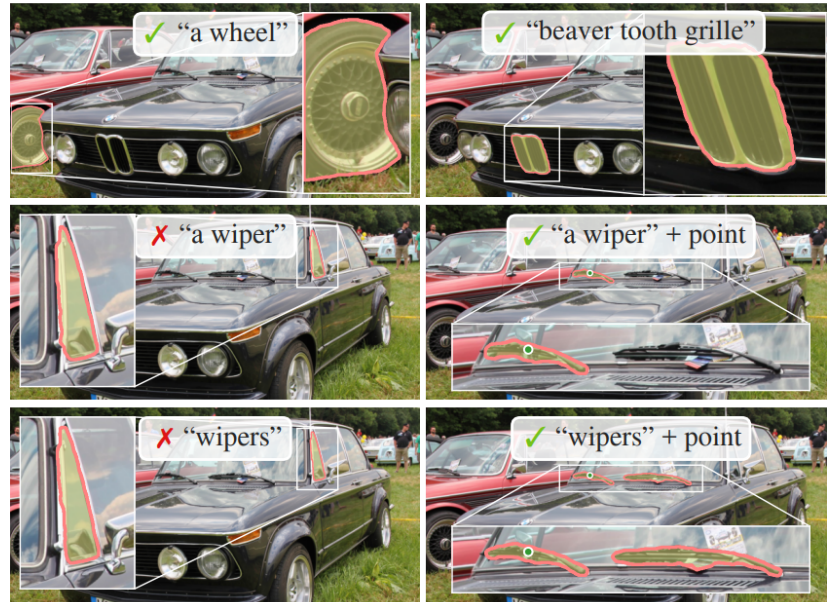


Figura 3.23: Segmentazione da testo a maschera in modalità zero-shot. SAM può funzionare con prompt testuali semplici o complessi. Quando SAM non riesce a fare una predizione corretta, un prompt aggiuntivo basato su un punto può essere di aiuto.[4]

3.4.5 Conclusioni e limitazioni

Il Segment Anything Model rappresenta un passo avanti verso modelli fondamentali per la visione artificiale, offrendo un sistema altamente generalizzabile e flessibile. Tuttavia, presenta alcune limitazioni:

- Le maschere possono mancare di dettagli fini o mostrare componenti non connessi;
- Non è ottimizzato per task come la segmentazione semantica o panottica;
- L'efficienza in tempo reale è limitata dalla complessità dell'encoder di immagini.

Il rilascio pubblico di SAM e del dataset SA-1B (sotto licenza Apache 2.0 su <https://segment-anything.com>) segna un contributo significativo alla ricerca sui modelli fondamentali per la visione artificiale.

3.5 MiDaS

Il modello MiDaS [5](Multiple Depth Estimation Accuracy with Single Network) rappresenta un significativo passo avanti nel campo della stima della profondità monoculare, un problema centrale nella computer vision con numerose applicazioni, tra cui la robotica, la realtà aumentata e la guida autonoma.

La capacità di stimare la profondità da un'unica immagine bidimensionale è fondamentale per comprendere e interagire con ambienti tridimensionali, ma il problema è intrinsecamente complesso e sotto-determinato, richiedendo avanzati strumenti di apprendimento automatico.

I primi lavori sulla stima della profondità monoculare utilizzavano formulazioni basate su MRF (Markov Random Fields) [51], semplici assunzioni geometriche [52] o metodi non parametrici [53].

Tuttavia, con l'introduzione delle reti neurali convoluzionali (CNN), è stato possibile apprendere rappresentazioni più ricche e gerarchiche direttamente dai dati.

Questi progressi hanno reso evidente la necessità di dataset ampi e diversificati per addestrare modelli capaci di generalizzare a contesti non visti.

Nonostante i progressi, le soluzioni esistenti sono limitate da bias significativi nei dataset disponibili, che spesso catturano solo specifici tipi di ambienti (ad esempio, indoor o outdoor) e utilizzano modalità di acquisizione con vincoli intrinseci, come scanner laser, sensori RGB-D o annotazioni manuali.

Inoltre, l'acquisizione di dati di profondità accurati su larga scala è un compito oneroso, complicato da costi elevati e difficoltà tecniche nell'acquisizione di scene dinamiche.

3.5.1 Obiettivo del modello MiDaS

Il modello MiDaS affronta queste problematiche attraverso un approccio innovativo che combina dati da fonti eterogenee e integra una serie di strumenti avanzati [5]:

- Funzioni di perdita invarianti a scala e traslazione: queste funzioni consentono al modello di addestrarsi su dataset con annotazioni di profondità non uniformi, come disparità relativa, profondità assoluta e altri formati;
- Miscelazione controllata di dataset: utilizzando un'ottimizzazione multi-obiettivo, il modello bilancia efficacemente l'influenza di dataset complementari, migliorando la robustezza e la capacità di generalizzazione;
- Dataset 3D Movies: MiDaS introduce un nuovo dataset costruito utilizzando film stereoscopici, che forniscono milioni di immagini di alta qualità e diversificate, superando le limitazioni dei dataset tradizionali (Fig. 3.25).

3.5.2 Architettura del modello

L'architettura di MiDaS [5] è progettata per affrontare le sfide principali della stima della profondità monoculare, introducendo innovazioni sia nella scelta della rappresentazione dei dati che nelle strategie di addestramento, rendendo il modello altamente robusto e generalizzabile.

Rappresentazione dei Dati

Il modello predice la disparità inversa anziché la profondità assoluta, una scelta che lo rende compatibile con dati annotati in modo diverso (es. profondità assoluta, relativa o disparità).

Questa rappresentazione permette di trattare dataset con scale diverse senza dover normalizzare i valori in anticipo.

Per gestire la variazione di scala e traslazione nei dati, gli autori introducono una funzione di perdita invariante rispetto a scala e traslazione.

Questo approccio consente al modello di generalizzare meglio su dataset mai visti.

Struttura della Rete

La rete utilizza un'architettura multi-scala basata su ResNet, un'architettura consolidata per il riconoscimento delle immagini [2].

L'encoder della rete è pre-addestrato su ImageNet [31], una scelta che migliora significativamente le prestazioni, dato che le capacità dell'encoder per compiti generici di visione artificiale influenzano direttamente la qualità delle stime di profondità.

Funzioni di perdita

La funzione di perdita, progettata per affrontare le ambiguità relative a scala e traslazione, è definita come:

$$\mathcal{L}_{ssi}(\hat{\mathbf{d}}, \hat{\mathbf{d}}^*) = \frac{1}{2M} \sum_{i=1}^M p(\hat{\mathbf{d}}_i - \hat{\mathbf{d}}_i^*) \quad (3.3)$$

dove:

- $\hat{\mathbf{d}}, \hat{\mathbf{d}}^*$ sono versioni scalate e traslate della prediction e della ground truth;
- p rappresenta la funzione di perdita specifica;
- M è il numero di pixel con ground truth valida

Gli autori propongono tre varianti di funzioni di perdita ($\mathcal{L}_{ssi}$):

- Mean Squared Error (MSE) invariante a scala e traslazione $p(x) = p_{mse} = x^2$ ($\mathcal{L}_{ssimse}$);
- MAE invariante a scala e traslazione $p(x) = p_{mae} = |x|$ ($\mathcal{L}_{ssimae}$);
- Perdita con trimming ($\mathcal{L}_{ssitrim}$). In questa variante, vengono ignorati i residui più grandi (outlier) in ogni immagine.

$$\mathcal{L}_{ssitrim}(\hat{\mathbf{d}}, \hat{\mathbf{d}}^*) = \frac{1}{2M} \sum_{j=1}^{U_m} p_{mae}(\hat{\mathbf{d}}_j - \hat{\mathbf{d}}_j^*) \quad (3.4)$$

con $|\hat{\mathbf{d}}_j - \hat{\mathbf{d}}_j^*| \leq |\hat{\mathbf{d}}_{j+1} - \hat{\mathbf{d}}_{j+1}^*|$ e $U_m = 0.8M$

Le ultime due loss sono state introdotte perchè $\mathcal{L}_{ssimse}$ non è robusta agli outlier.

Per definire la perdita completa, è stato adattato il termine di corrispondenza del gradiente multi-scala e invariante rispetto alla scala [54] allo spazio delle disparità.

Questo termine incentiva le discontinuità ad essere nitide e a coincidere con le discontinuità presenti nella e ground truth.

Il termine di corrispondenza del gradiente è definito come:

$$\mathcal{L}_{reg}(\hat{\mathbf{d}}, \hat{\mathbf{d}}^*) = \frac{1}{M} \sum_{k=1}^K \sum_{i=1}^M (|\nabla_x R_i^k| + |\nabla_y R_i^k|) \quad (3.5)$$

dove:

- $R_i^k = \hat{\mathbf{d}}_i - \hat{\mathbf{d}}_i^*$ è la differenza tra le mappe di disparità a livello k ;
- K è il numero di livelli di scala considerati.

La funzione di perdita finale per un training set l è:

$$\mathcal{L}_l = \frac{1}{N} \sum_{n=1}^{N_l} \mathcal{L}_{ssi}(\hat{\mathbf{d}}^n, (\hat{\mathbf{d}}^*)^n) + \alpha \mathcal{L}_{reg}(\hat{\mathbf{d}}^n, (\hat{\mathbf{d}}^*)^n) \quad (3.6)$$

dove:

- N_l è la dimensione del training set;
- $\alpha = 0.5$ è un peso empirico per il termine di regolarizzazione.

Gli effetti delle diverse funzioni di perdita sulle performance del processo di validazione sono mostrati nella Fig. 3.24.

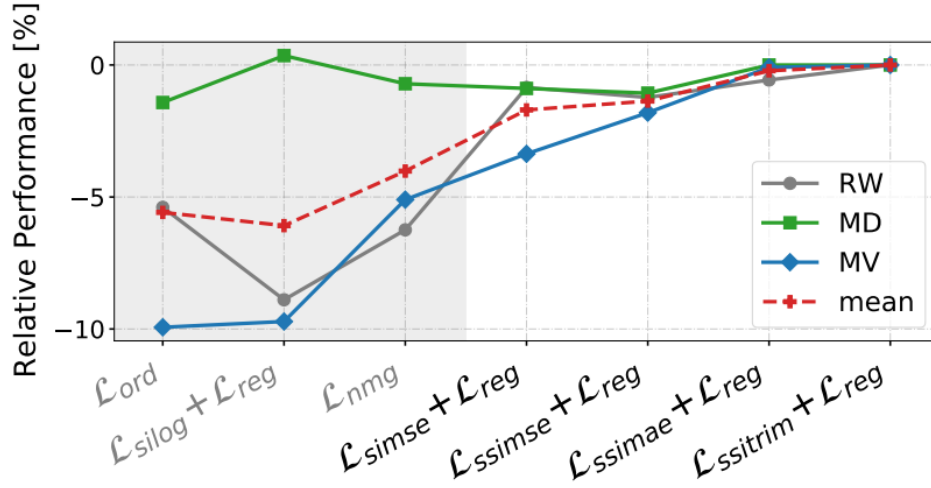


Figura 3.24: Prestazioni relative delle diverse funzioni di perdita (valori più alti indicano prestazioni migliori) con la perdita migliore, $\mathcal{L}_{ssitrim} + \mathcal{L}_{reg}$, utilizzata come riferimento. Tutte le quattro funzioni di perdita proposte (area bianca) superano le attuali funzioni di perdita allo stato dell'arte (area grigia).[5]

Miscelazione dei dataset

Le strategie di mixing dei dataset descritte in MiDaS [5] sono fondamentali per ottenere modelli generali in grado di generalizzare su scenari mai visti (zero-shot cross-dataset transfer).

MiDaS affronta il problema di combinare dati provenienti da diverse fonti, ciascuna con caratteristiche e annotazioni diverse, utilizzando due approcci principali: il mixing naive e l'ottimizzazione multi-task.

- **Mixing naive:** il metodo più semplice per combinare dataset, suddivide uniformemente i campioni nei minibatch durante il training. Supponendo di avere L dataset distinti e un batch size B , seleziona B/L campioni da ciascun dataset per formare un minibatch. Questa distribuzione assicura che ogni dataset sia rappresentato in egual misura nel processo di training, indipendentemente dalle sue dimensioni.
- **Ottimizzazione multi-task:** un approccio più rigoroso. L'apprendimento su ciascun dataset è definito come un compito separato e si cerca un approssimativo ottimo di Pareto tra i dataset (ossia, una soluzione in cui la perdita non può essere ridotta su alcun dataset di training senza aumentarla almeno su uno degli altri) [55].

A differenza del mixing naive che trattano tutti i dataset in modo uniforme, l'approccio multi-task bilancia in modo ottimale l'influenza di ciascun dataset durante l'addestramento, migliorando l'efficacia complessiva del modello.

Dataset Utilizzati

Il modello è addestrato su cinque dataset complementari, ciascuno con caratteristiche uniche (es. scene dinamiche, profondità relativa o assoluta) [5]:

1. **ReDWeb [56] (RW):** un dataset piccolo, fortemente curato, che presenta scene dinamiche e diversificate con ground truth acquisita tramite una baseline stereo relativamente grande.
2. **MegaDepth [54] (MD):** dataset molto più grande, ma mostra principalmente scene statiche. La ground truth è generalmente più accurata nelle regioni di sfondo, poiché è stata utilizzata la ricostruzione stereo multi-view con una larga baseline per l'acquisizione.
3. **WSVD [57] (WS):** consiste in video stereo ottenuti dal web e presenta scene dinamiche e diversificate. Per questo dataset non è fornita una ground truth, pertanto, è stata ricreata seguendo la procedura descritta dagli autori originali.
4. **DIML Indoor [58] (DL):** un dataset RGB-D di scene indoor prevalentemente statiche, acquisite con un Kinect v2.

Mentre come test dataset sono stati scelti 6 dataset in base alla diversità e all'accuratezza della loro ground truth [5]:

1. **DIW [59]:** altamente diversificato, ma fornisce la ground truth solo sotto forma di relazioni ordinali sparse.

2. ETH3D [60]: presenta una ground truth altamente accurata, ottenuta tramite scansione laser su scene statiche.
3. Sintel [61]: offre una ground truth perfetta per scene sintetiche.
4. KITTI [62] e NYU [63]: sono dataset comunemente utilizzati con bias caratteristici.
5. TUM dataset [64]: è stato usato il sottoinsieme dinamico che presenta esseri umani in ambienti indoor [65].

Per completare i dataset esistenti ne è stato creato uno nuovo "3D movies" (MV) [5].

3D movies presenta fotogrammi video di alta qualità in una varietà di ambienti dinamici che vanno da immagini incentrate sugli esseri umani in film di Hollywood basati su storia e dialoghi, a scene naturali con paesaggi e animali in documentari.

L'uso di questo nuovo dataset rappresenta una novità importante, consentendo di accedere a milioni di immagini di alta qualità, migliorando la diversità e la scala dei dati disponibili (Fig. 3.25 e Tab. 3.6).

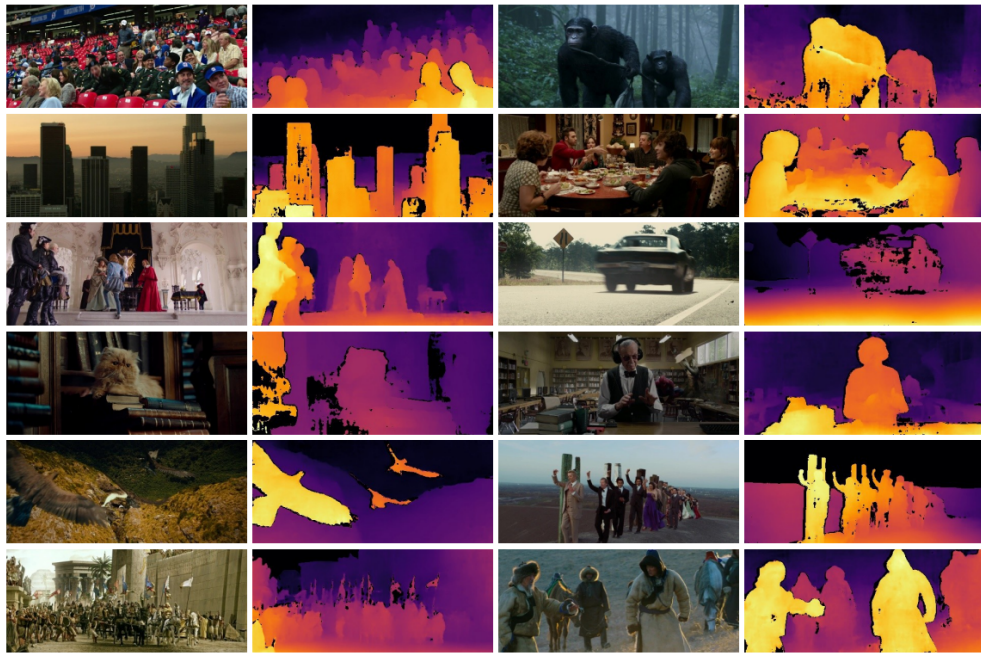


Figura 3.25: Immagini campione dal dataset 3D Movies.[5]

Titolo del film	#frames
Training set	75074
Battle of the Year (2013)	4821
Billy Lynn's Long Halftime Walk (2016)	4178
Drive Angry (2011)	328
Exodus: Gods and Kings (2014)	8063
Final Destination 5 (2011)	1437
A very Harold & Kumar 3D Christmas (2011)	3690
Hellbenders (2012)	120
The Hobbit: An Unexpected Journey (2012)	8874
Hugo (2011)	3189
The Three Musketeers (2011)	5028
Nurse 3D (2013)	492
Pina (2011)	1215
Dawn of the Planet of the Apes (2014)	5571
The Amazing Spider-Man (2012)	5618
Step Up 3D (2010)	509
Step Up: All In (2014)	2187
Transformers: Age of Extinction (2014)	8740
Le Dernier Loup / Wolf Totem (2015)	4843
X-Men: Days of Future Past (2014)	6171
Validation set	3058
The Great Gatsby (2013)	1815
Step Up: Miami Heat / Revolution (2012)	1243
Test set	788
Doctor Who - The Day of the Doctor (2013)	508
StreetDance 2 (2012)	280

Tabella 3.6: Lista dei film e il numero di frame estratti nel dataset 3D Movies.[5]

3.5.3 Prestazioni

Le prestazioni di MiDaS sono state valutate attraverso il protocollo di trasferimento "zero-shot cross-dataset", in cui il modello è testato su dataset mai visti durante l'addestramento [5].

Gli esperimenti dimostrano che l'architettura proposta raggiunge risultati all'avanguardia nella stima della profondità monoculare, con prestazioni significativamente migliori rispetto ai metodi esistenti su sei dataset di test mai visti durante l'addestramento.

Le Figure 3.26 e 3.27 mostrano esempi qualitativi delle stime di profondità prodotte dal modello, evidenziando la sua capacità di generalizzare su scene complesse

e diversificate.



Figura 3.26: In alto: immagini di input. Al centro: mappe di profondità inversa previste da MiDaS. In basso: corrispondenti point cloud renderizzati da un nuovo punto di vista, renderizzate tramite Open3D [66]. Immagini di input dal dataset Microsoft COCO [47], che non è stato visualizzato durante l'addestramento.[5]



Figura 3.27: Confronto qualitativo di MiDaS ai quattro migliori concorrenti sulle immagini dal dataset Microsoft COCO [47]. Dall'alto verso il basso: immagini di input, Xian et al. [56], Wang et al. [57], Li et al. [65], Li & Snavely [54] e MiDaS [5].

Capitolo 4

Soluzione proposte

Nel seguente capitolo verranno descritti ed analizzati i due approcci proposti con particolare attenzione verso l'implementazione, i risultati e i confronti tra le diverse metodologie.

4.1 Approccio 1

In questo approccio ho scelto di usare il modello YOLOv8 [1], in modo tale da avere un modello in grado di effettuare l'instance segmentation delle immagini in input e non semplicemente la classificazione.

Il modello è stato allenato usando FooSeg103 [10] come dataset. Tuttavia, il dataset non può essere usato nella sua versione nativa, ma necessita dei cambiamenti.

Preparazione del dataset

Il dataset FoodSeg103 ha per ogni immagine un'immagine con le corrispettive maschere in scala di grigi dove ogni pixel ha un valore da 0 a 103 in base alla classe di appartenenza, vedi Fig. 4.1. Per poter utilizzare YOLO è necessario ridurre le maschere ad un formato testuale nella forma:

`<class> <center_x> <center_y> <width> <height>`

di conseguenza ho realizzato uno script che andava a creare per ogni immagine un file .txt con le annotazioni delle maschere.

A causa delle dimensioni non troppo generose del dataset non ho applicato uno split del training set in training e validation set, ma ho usato il test set come validation.

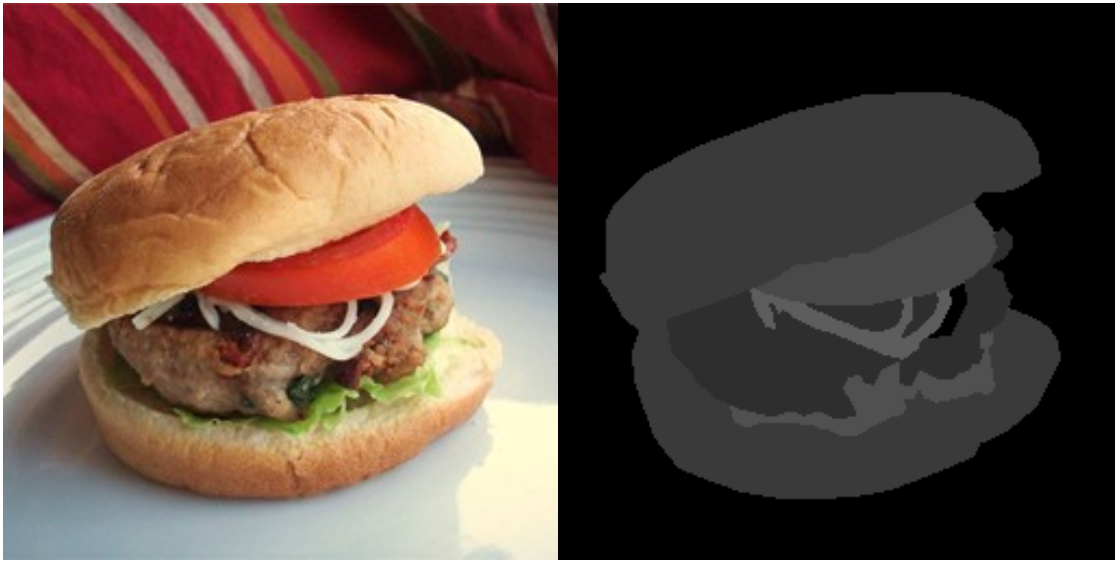


Figura 4.1: Esempio di immagine (sinistra) e maschere (destra) in FoodSeg103.

4.1.1 Risultati

Per testare il modello sono stati effettuati allenamenti con diversi valori del batch-size.

Nella tabella seguente 4.1 sono riportate le metriche sul validation set dopo un training di 200 epoche con batch size 8 ed immagini con dimensione di 480 pixel.

Class	Instances	Box Mask (P)	R	mAP50	mAP50-95)
all	14060	0.396 0.397	0.287 0.284	0.261 0.256	0.212 0.197
candy	49	0 0	0 0	0.0445 0.0345	0.0245 0.0202
egg tart	6	0.278 0.282	0.167 0.167	0.15 0.15	0.15 0.12
french fries	113	0.506 0.52	0.425 0.434	0.421 0.422	0.367 0.348
chocolate	66	0.382 0.404	0.212 0.212	0.217 0.218	0.174 0.173
biscuit	140	0.312 0.276	0.414 0.357	0.295 0.252	0.268 0.228
popcorn	11	0.507 0.429	0.545 0.455	0.571 0.429	0.387 0.374

Class	Instances	Box Mask (P	R	mAP50	mAP50-95)
pudding	1	1 1	0 0	0 0	0 0
ice cream	448	0.521 0.486	0.436 0.402	0.4 0.366	0.318 0.282
cheese butter	214	0.167 0.155	0.0607 0.0561	0.0609 0.0546	0.0345 0.03
cake	245	0.395 0.356	0.376 0.331	0.306 0.224	0.224 0.154
wine	50	0.319 0.316	0.487 0.481	0.333 0.338	0.266 0.242
milkshake	32	0.326 0.329	0.625 0.625	0.487 0.497	0.431 0.413
coffee	61	0.538 0.529	0.754 0.738	0.716 0.712	0.636 0.619
juice	67	0.397 0.398	0.541 0.537	0.444 0.436	0.348 0.323
milk	37	0.543 0.545	0.432 0.432	0.496 0.494	0.394 0.379
tea	6	0.127 0.128	0.167 0.167	0.165 0.165	0.153 0.144
almond	74	0.19 0.179	0.193 0.176	0.112 0.0997	0.0928 0.0707
red beans	34	0.0875 0.0904	0.0294 0.0294	0.0316 0.0316	0.0252 0.0168
cashew	42	0.378 0.392	0.0476 0.0476	0.0783 0.0761	0.051 0.0527
dried cranberries	55	0.208 0.213	0.0364 0.0364	0.107 0.107	0.0762 0.0715
soy	19	0.768 0.775	0.368 0.368	0.442 0.442	0.417 0.398
walnut	84	0.544 0.557	0.226 0.226	0.275 0.276	0.226 0.213
peanut	21	0 0	0 0	0.0416 0.0416	0.0197 0.0184
egg	119	0.339 0.335	0.353 0.345	0.265 0.263	0.193 0.191
apple	86	0.393 0.43	0.302 0.314	0.243 0.263	0.204 0.196

Class	Instances	Box Mask (P)	R	mAP50	mAP50-95)
date	3	1 1	0 0	0.0357 0.0357	0.0297 0.0273
apricot	18	0.0689 0.056	0.0383 0.0311	0.0457 0.048	0.045 0.0415
avocado	36	0.19 0.184	0.139 0.132	0.0892 0.0883	0.0652 0.0664
banana	105	0.529 0.573	0.495 0.524	0.503 0.538	0.432 0.42
strawberry	435	0.648 0.637	0.651 0.637	0.666 0.64	0.53 0.486
cherry	148	0.451 0.508	0.459 0.514	0.365 0.416	0.297 0.293
blueberry	226	0.667 0.689	0.774 0.796	0.791 0.797	0.626 0.583
raspberry	63	0.502 0.534	0.54 0.571	0.469 0.535	0.352 0.361
mango	26	0.12 0.121	0.115 0.115	0.0482 0.0482	0.0343 0.0278
olives	43	0.65 0.727	0.233 0.256	0.286 0.294	0.177 0.182
peach	34	0.234 0.231	0.496 0.486	0.351 0.351	0.297 0.284
lemon	275	0.543 0.547	0.705 0.707	0.663 0.67	0.566 0.551
pear	19	0.338 0.341	0.105 0.105	0.079 0.0779	0.0547 0.0676
fig	9	0.0682 0.0688	0.222 0.222	0.0544 0.0547	0.0419 0.0257
pineapple	96	0.307 0.333	0.26 0.281	0.18 0.184	0.143 0.138
grape	52	0.326 0.34	0.519 0.538	0.469 0.489	0.401 0.37
kiwi	25	0.444 0.45	0.48 0.48	0.405 0.404	0.338 0.295
melon	7	0.478 0.477	0.267 0.266	0.237 0.237	0.197 0.188
orange	114	0.476 0.468	0.491 0.474	0.441 0.433	0.381 0.344

Class	Instances	Box Mask (P	R	mAP50	mAP50-95)
watermelon	20	0.0524 0.171	0.05 0.15	0.0292 0.0606	0.0189 0.0208
steak	513	0.438 0.442	0.474 0.472	0.39 0.386	0.305 0.29
pork	295	0.226 0.22	0.285 0.271	0.172 0.164	0.145 0.134
chicken duck	557	0.413 0.412	0.44 0.433	0.33 0.317	0.264 0.247
sausage	108	0.134 0.14	0.185 0.185	0.0872 0.0889	0.0733 0.0687
fried meat	124	0.216 0.181	0.129 0.105	0.113 0.104	0.0942 0.0877
lamb	46	0.128 0.127	0.156 0.152	0.0895 0.0866	0.0809 0.0668
sauce	432	0.437 0.422	0.454 0.435	0.43 0.403	0.358 0.325
crab	16	0.435 0.455	0.0625 0.0625	0.13 0.13	0.095 0.0646
fish	149	0.267 0.261	0.322 0.309	0.24 0.225	0.211 0.188
shellfish	33	0.304 0.272	0.273 0.242	0.239 0.197	0.173 0.138
shrimp	106	0.414 0.432	0.283 0.292	0.231 0.233	0.162 0.14
soup	31	0.235 0.24	0.355 0.355	0.322 0.308	0.308 0.271
bread	774	0.477 0.471	0.569 0.556	0.478 0.467	0.379 0.348
corn	193	0.807 0.809	0.663 0.663	0.704 0.705	0.647 0.628
hamburg	1	1 1	0 0	0.0398 0	0.00398 0
pizza	30	0.181 0.155	0.233 0.2	0.0841 0.0659	0.0743 0.0471
hanamaki baozi	12	0.102 0.105	0.0833 0.0833	0.0308 0.0308	0.0259 0.0262
wonton dumplings	15	0 0	0 0	0.103 0.103	0.0795 0.085

Class	Instances	Box Mask (P)	R	mAP50	mAP50-95)
pasta	76	0.336 0.345	0.287 0.289	0.228 0.232	0.208 0.203
noodles	162	0.501 0.453	0.457 0.407	0.394 0.339	0.292 0.225
rice	342	0.59 0.589	0.541 0.538	0.529 0.527	0.419 0.413
pie	281	0.305 0.261	0.31 0.263	0.227 0.172	0.189 0.136
tofu	38	0.0788 0.0377	0.061 0.0263	0.0474 0.0471	0.028 0.0255
eggplant	12	1 1	0 0	0.0161 0.0161	0.00804 0.00521
potato	460	0.446 0.451	0.51 0.511	0.46 0.452	0.386 0.366
garlic	30	0.0273 0.0293	0.0333 0.0333	0.0476 0.0484	0.0455 0.0415
cauliflower	116	0.516 0.516	0.509 0.5	0.478 0.479	0.367 0.36
tomato	727	0.618 0.63	0.62 0.626	0.649 0.661	0.52 0.494
kelp	6	1 1	0 0	0 0	0 0
seaweed	10	0.24 0.223	0.072 0.0669	0.177 0.0928	0.103 0.0699
spring onion	136	0.278 0.284	0.11 0.11	0.0736 0.0651	0.0441 0.0328
rape	23	0.205 0.209	0.261 0.261	0.236 0.236	0.229 0.219
ginger	12	0.189 0.193	0.0833 0.0833	0.159 0.159	0.144 0.13
okra	11	0 0	0 0	0.00453 0.00453	0.00223 0.00157
lettuce	376	0.392 0.389	0.397 0.388	0.328 0.314	0.23 0.194
pumpkin	27	0.168 0.201	0.259 0.296	0.168 0.181	0.153 0.161
cucumber	278	0.47 0.482	0.468 0.471	0.436 0.433	0.357 0.346

Class	Instances	Box Mask (P)	R	mAP50	mAP50-95)
white radish	35	0.325 0.328	0.0857 0.0857	0.11 0.11	0.085 0.0859
carrot	766	0.705 0.717	0.631 0.637	0.649 0.657	0.549 0.514
asparagus	206	0.481 0.469	0.356 0.345	0.338 0.324	0.281 0.244
bamboo shoots	7	1 1	0 0	0 0	0 0
broccoli	487	0.764 0.766	0.692 0.692	0.719 0.723	0.619 0.589
celery stick	102	0.499 0.514	0.637 0.637	0.604 0.606	0.488 0.477
cilantro mint	624	0.478 0.48	0.442 0.439	0.405 0.397	0.311 0.272
snow peas	56	0.271 0.27	0.232 0.212	0.156 0.144	0.131 0.118
cabbage	47	0.384 0.375	0.426 0.408	0.34 0.337	0.319 0.3
bean sprouts	25	0.425 0.428	0.12 0.12	0.116 0.117	0.0329 0.059
onion	328	0.27 0.271	0.158 0.155	0.161 0.132	0.0912 0.078
pepper	278	0.308 0.294	0.211 0.198	0.204 0.193	0.138 0.121
green beans	136	0.659 0.644	0.412 0.398	0.494 0.48	0.405 0.382
French beans	240	0.603 0.593	0.417 0.408	0.45 0.435	0.386 0.351
king oyster mushroom	3	0 0	0 0	0 0	0 0
shiitake	105	0.29 0.289	0.229 0.219	0.156 0.159	0.118 0.11
enoki mushroom	5	1 1	0 0	0.0241 0.0241	0.0193 0.0193
oyster mushroom	4	1 1	0 0	0 0	0 0
white button mushroom	66	0.239 0.256	0.242 0.258	0.149 0.156	0.095 0.0944

Class	Instances	Box Mask (P)	R	mAP50	mAP50-95)
salad	11	0.176 0.177	0.0909 0.0909	0.0614 0.0614	0.0549 0.0485
other ingredients	137	0 0	0 0	0.014 0.0104	0.00877 0.00696

Tabella 4.1: Risultati sulle 2135 immagini del validation set per l'intero set (all) e per ciascuna classe. In ogni riga è riportato il numero di istanze e le metriche sia per la box che per la maschera (riga sopra box, riga sotto maschera).

Nella figura 4.2 vi è un esempio visivo dei risultati del modello allenato sul batch 1, mentre la figura 4.3 mostra la confusion matrix normalizzata.



Figura 4.2: Risultati sul batch 1 del validation set. In alto le labels ed in basso le prediction del modello allenato.

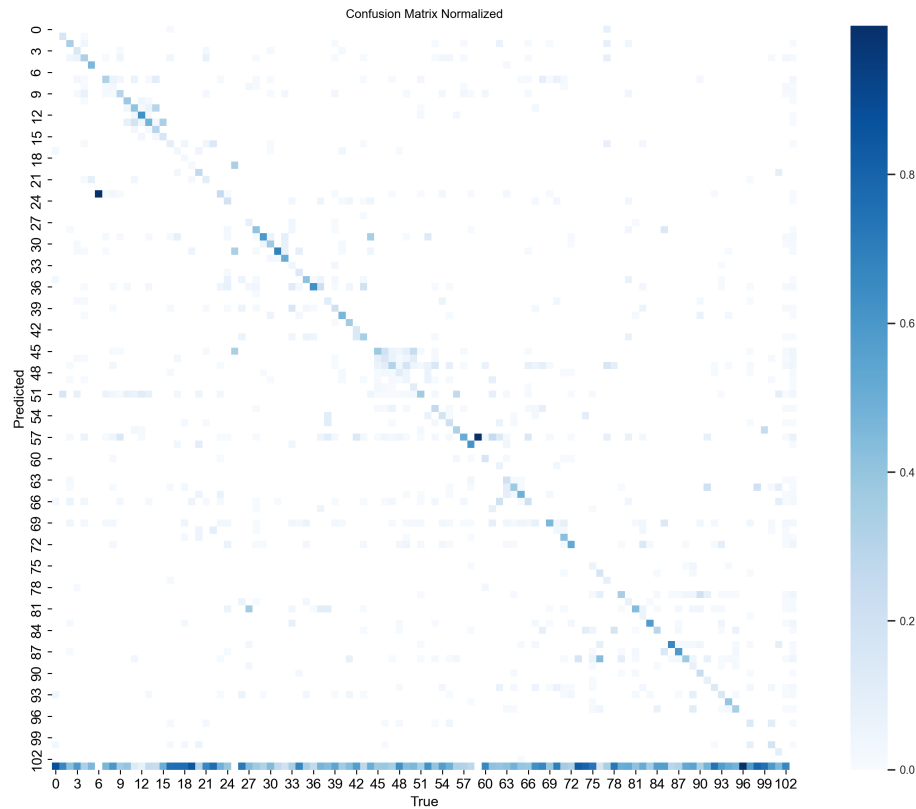


Figura 4.3: Confusion matrix normalizzata modello YOLOv8 allenato per 200 epoche con batch size 8.

Come si può notare, i risultati sono molto deludenti; ciò può essere dovuto a più fattori:

1. Il dataset non ha abbastanza immagini e di conseguenza maschere;
2. Il numero di maschere per classe è altamente sbilanciato.

A ciò si aggiunge che, in genere, il task di instance segmentation viene effettuato su un numero di classi inferiori a quelle presenti nel dataset e su classi che sono ben distinte tra di loro.

Per queste ragioni, ho tentato un nuovo approccio.

4.2 Approccio 2

In questo nuovo approccio i task di classificazione e segmentazione sono ben distinti e separati.

4.2.1 Classificazione delle immagini

Per la classificazione delle immagini, la scelta del dataset è ricaduta su Food-101 [8] ed i modelli testati sono stati ResNet50 [2] e Inception-v3 [3].

I due modelli sono stati allenati usando per entrambi transfer learning con i pesi di ImageNet e sono stati implementati usando le librerie di PyTorch.

In questa fase sono stati allenati diversi modelli; per semplicità ho diviso i risultati riportati in due gruppi.

Gruppo 1

Nella divisione originale le 1000 immagini per classe sono divise in 750 immagini per il training e 250 per il test.

Ho effettuato un'ulteriore divisione delle 750 immagini con percentuali del 75% per il training ed il restante 25% per il validation set (Tab. 4.2).

Set	Numero di immagini
Train	56812
Validation	18938
Test	25250

Tabella 4.2: Distribuzione delle immagini usando uno split del 75% del training set.

Inoltre, le immagini sono state ridimensionate a 224x224 pixel nel caso della ResNet50, mentre hanno una dimensione di 299x299 pixel per l' Inception-v3 e hanno subito una normalizzazione usando la media e la deviazione standard di ImageNet.

I processi di training sono stati effettuati sia usando data augmentation sia senza.

Di seguito sono riportate le trasformazioni usate nel gruppo 1:

- Modifica della luminosità dell'immagine (ColorJitter(brightness=[1,2]) 1 nessuna modifica 2 luminosità raddoppiata);
- Conversione dell'immagine in scala di grigi mantenendo 3 canali (Grayscale(3));
- Rotazione casuale dell'immagine fino a ± 90 gradi (RandomRotation(degrees=90))

Training gruppo 1

Di seguito sono riportati i risultati ottenuti in questa prima fase sia per il modello Inception-v3, sia per ResNet50.

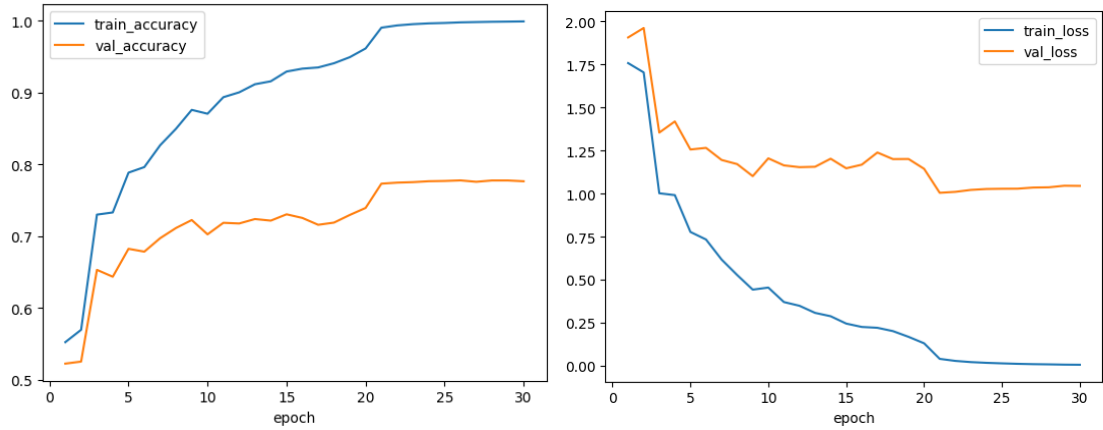


Figura 4.4: Precisione (accuracy) e perdita (loss) training Inception-v3 con: LR=0.0015, batch-size = 32 e usando data augmentation.

Accuracy su validation set	Accuracy su test set
77,66%	82,69%

Tabella 4.3: Miglior valore di accuracy su validation set e test accuracy Inception-v3 con: LR=0.0015, batch-size = 32 e usando data augmentation.

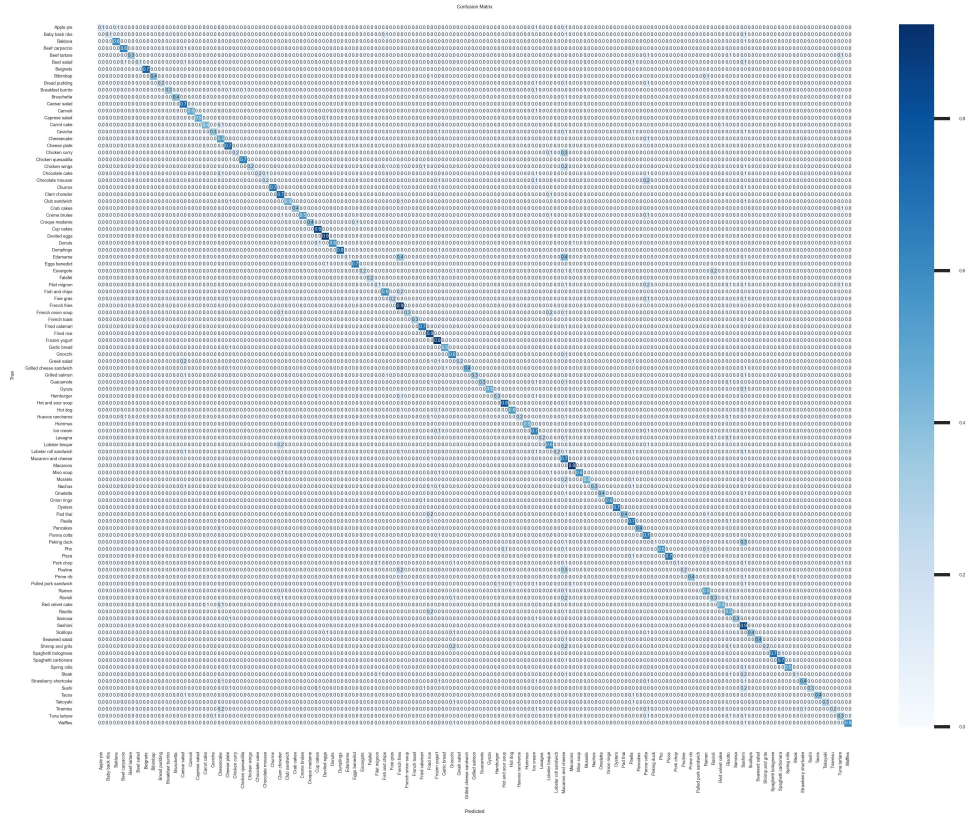


Figura 4.5: Confusion matrix Inception-v3 con: LR=0.0015, batch-size = 32 e usando data augmentation.

Accuracy su validation set	Accuracy su test set
78,40%	82,97%

Tabella 4.4: Miglior valore di accuracy su validation set e test accuracy ResNet50 con: LR=0.015, batch-size = 64 e usando data augmentation.

Gruppo 2

Nel gruppo due ho usato più immagini per il training set, quindi in questo caso solo il 20% del training set è stato usato per ottenere il validation set (Tab. 4.5).

Set	Numero di immagini
Train	60600
Validation	15150
Test	25250

Tabella 4.5: Distribuzione delle immagini usando uno slit del 80% del training set.

Anche le trasformazioni usate sono cambiate per cercare di avere una maggiore variabilità:

1. flip orizzontale casuale delle immagini (`RandomHorizontalFlip(p=1)`);
2. rotazione casuale tra -20 e 20 gradi (`RandomRotation(degrees=20)`);
3. zoom causale tra 1.0 (nessuna modifica) e 1.3 (i grandimento del 30%) (`RandomResizedCrop(size=(224, 224), scale=(1.0, 1.3))`);
4. modifica casuale della luminosità e del contrasto con una variazione massima di 0.2 (`ColorJitter(brightness=0.2, contrast=0.2)`).

Le trasformazioni sono applicate tutte con una probabilità del 75%.

Training gruppo 2

Come nel gruppo 1, di seguito sono riportati i risultati ottenuti su diversi modelli.

Accuracy su validation set	Accuracy su test set
78,06%	82,72%

Tabella 4.6: Miglior valore di accuracy su validation set e test accuracy Inception-v3 con: LR=0.0015, batch-size = 64 e data augmentation.

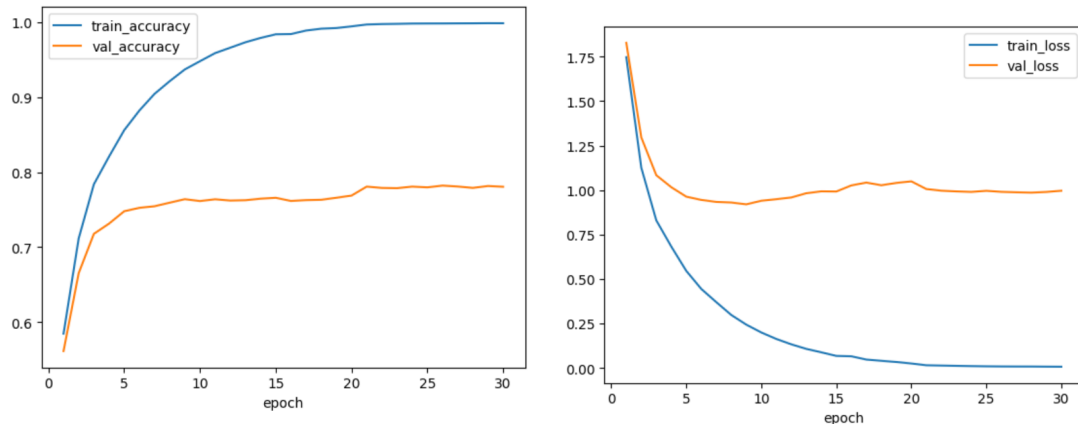


Figura 4.8: Precisione (accuracy) e perdita (loss) training Inception-v3 con: LR=0.0015, batch-size = 64 e data augmentation.

Accuracy su validation set	Accuracy su test set
76,01%	80,28%

Tabella 4.7: Miglior valore di accuracy su validation set e test accuracy ResNet50 con: LR=0.0015 e batch-size = 64 senza data augmentation.

Accuracy su validation set	Accuracy su test set
79,58%	84,10%

Tabella 4.8: Miglior valore di accuracy su validation set e test accuracy ResNet50 con: LR=0.0015, batch-size = 64 e data augmentation.

Soluzione proposte

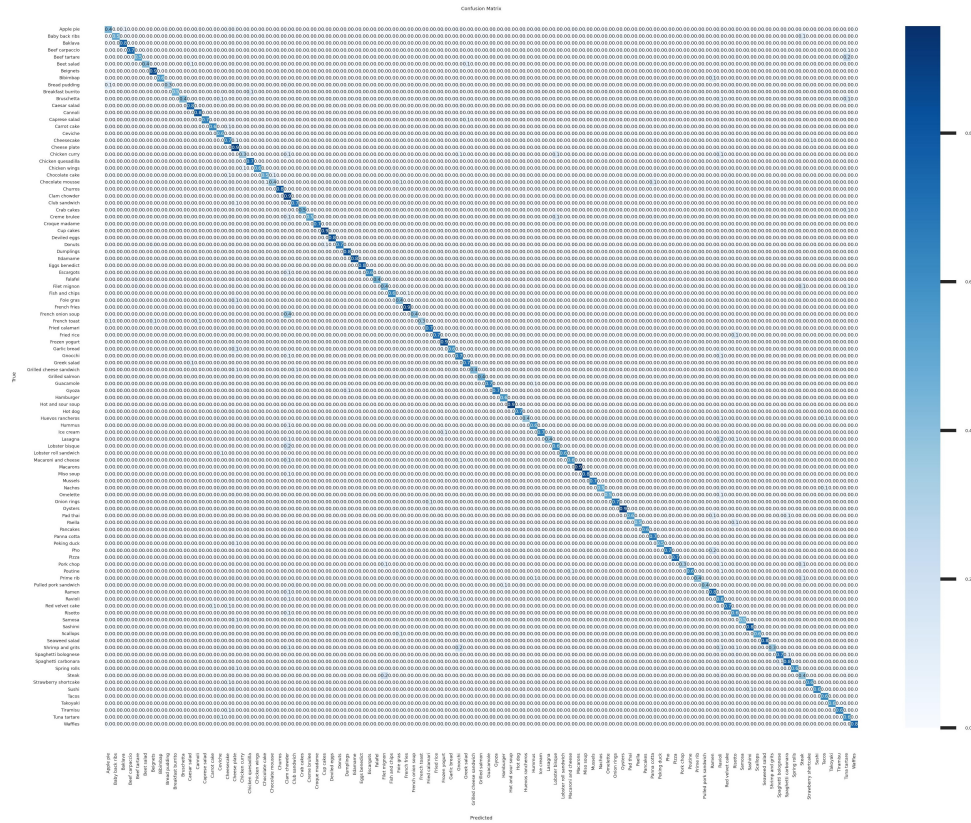


Figura 4.9: Confusion matrix Inception-v3 con: LR=0.0015, batch-size = 32 e data augmentation.

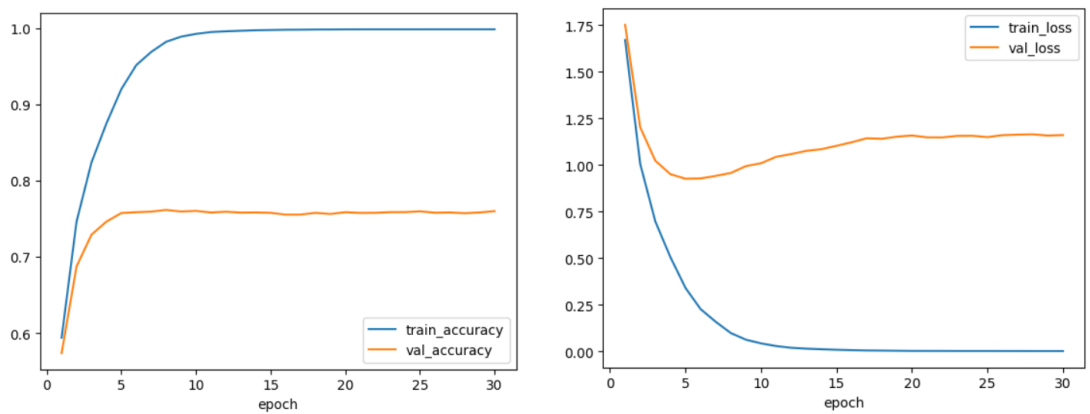


Figura 4.10: Precisione (accuracy) e perdita (loss) training ResNet50 con: LR=0.0015 e batch-size = 64 senza data augmentation.

Soluzione proposte

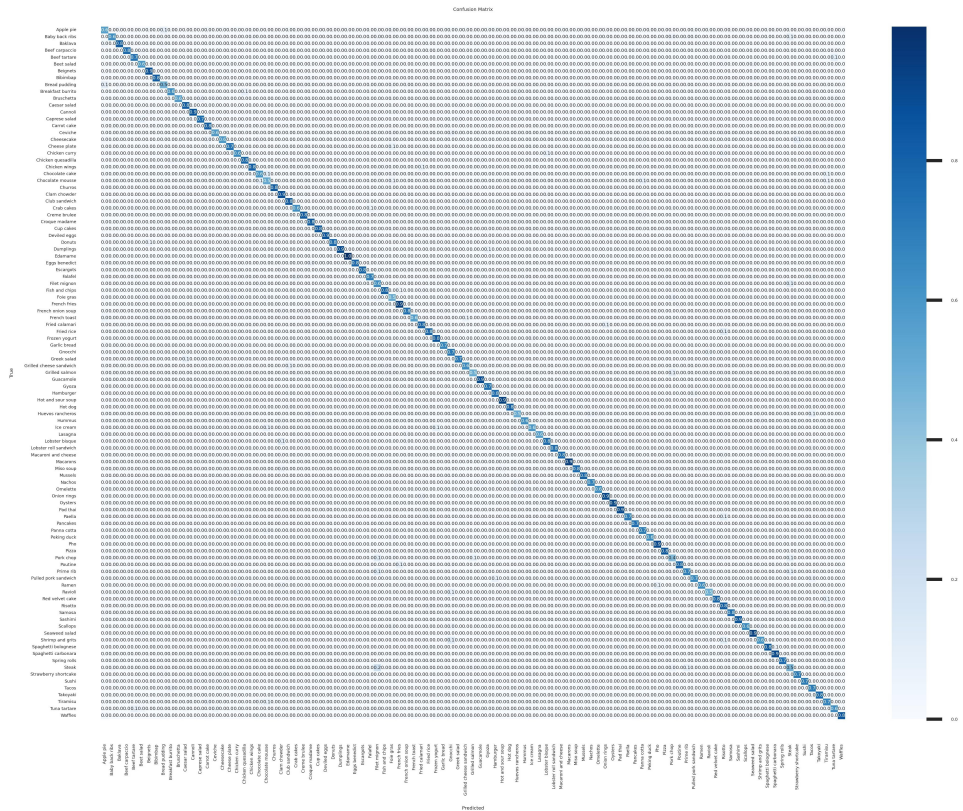


Figura 4.11: Confusion matrix ResNet50 con: LR=0.0015 e batch-size = 64 senza data augmentation.

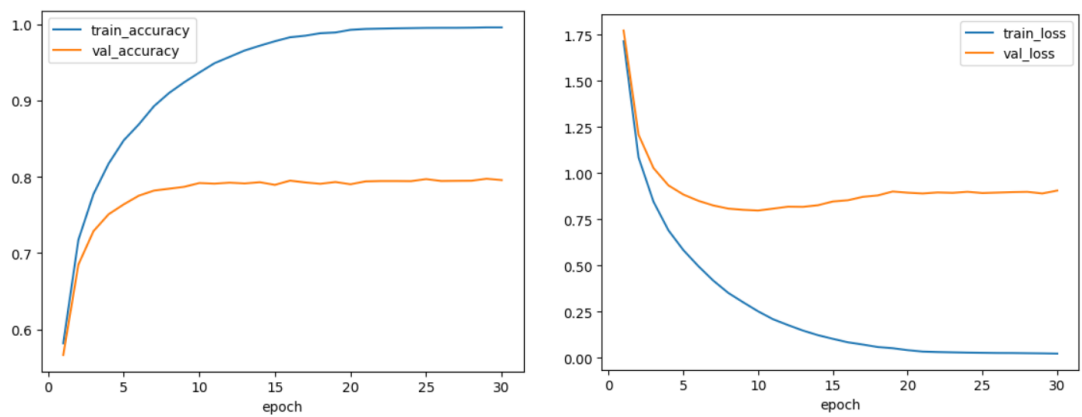


Figura 4.12: Precisione (accuracy) e perdita (loss) training ResNet50 con: LR=0.0015, batch-size = 64 e data augmentation.

Soluzione proposte

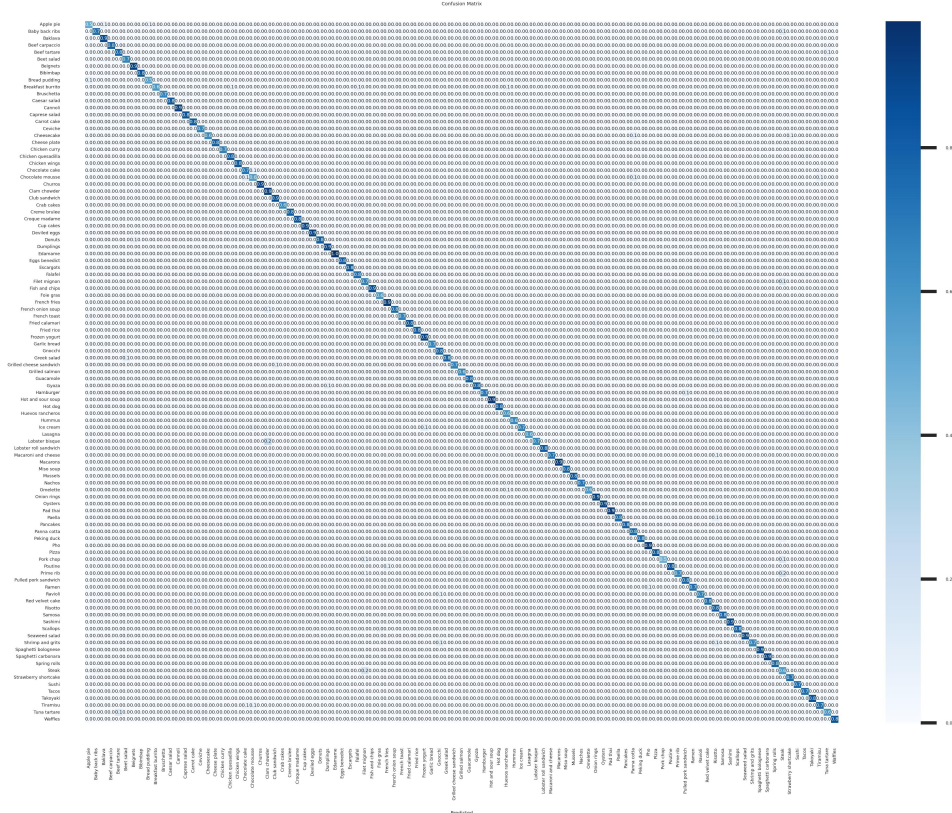


Figura 4.13: Confusion matrix ResNet50 con: LR=0.0015, batch-size = 64 e data augmentation.

4.2.2 Training con ResNet50 su fastai

A causa delle dimensioni del dataset ed effettuando gli esperimenti sul mio pc, il processo di ottimizzazione nella scelta degli iper-parametri è risultato molto lento. Ciò mi ha spinto ad utilizzare non più pytorch, ma a passare a Fastai [67] che, pur basandosi su pytorch, mette a disposizione una serie di funzioni che aiutano nella ricerca del valore ottimale del LR con la funzione *lr_find* e permettono di effettuare il training in modalità one-cycle.

Le metriche usate per valutare il modello sono:

- loss (perdita);
- accuracy (precisione);
- top 5 accuracy (precisione considerando i 5 risultati con il valore più alto).

4.2.3 Preparazione del dataset

Di seguito sono riportate tutte le caratteristiche legate al dataset in questo nuovo approccio:

- Split training set originale: 80% training set e 20% validation set (Tab. 4.5);
- Normalizzazione delle immagini con media e deviazione standard di ImageNet su training e validation set;
- trasformazioni specifiche del training set:
 1. flip orizzontale casuale delle immagini (*do_flip=True*);
 2. rotazione casuale tra -20 e 20 gradi (*max_rotate=20*);
 3. zoom casuale tra 1.0 (nessuna modifica) e 1.3 (i grandimento del 30%) (*min_zoom=1.0, max_zoom=1.3*);
 4. distorsione delle immagini fino ad un massimo del 20% (*max_warp=0.2*);
 5. modifica casuale della luminosità e del contrasto con una variazione massima di 0.2 (*max_lighting=0.2*).

Tutte le trasformazioni, sia le affini (rotazione, zoom, traslazione, warping) sia le modifiche di illuminazione e contrasto, hanno una probabilità del 75% di essere applicate (*p_affine=0.75* e *p_lighting=0.75*).

Di seguito viene visualizzato un esempio delle trasformazioni applicate ad un intero batch (Fig. 4.14).

Poichè il modello tendeva a convergere, ho deciso di allenarlo su immagini 224x224 per le prime 20 epoche e poi su immagini 512x512 pixel nelle successive 12 epoche.

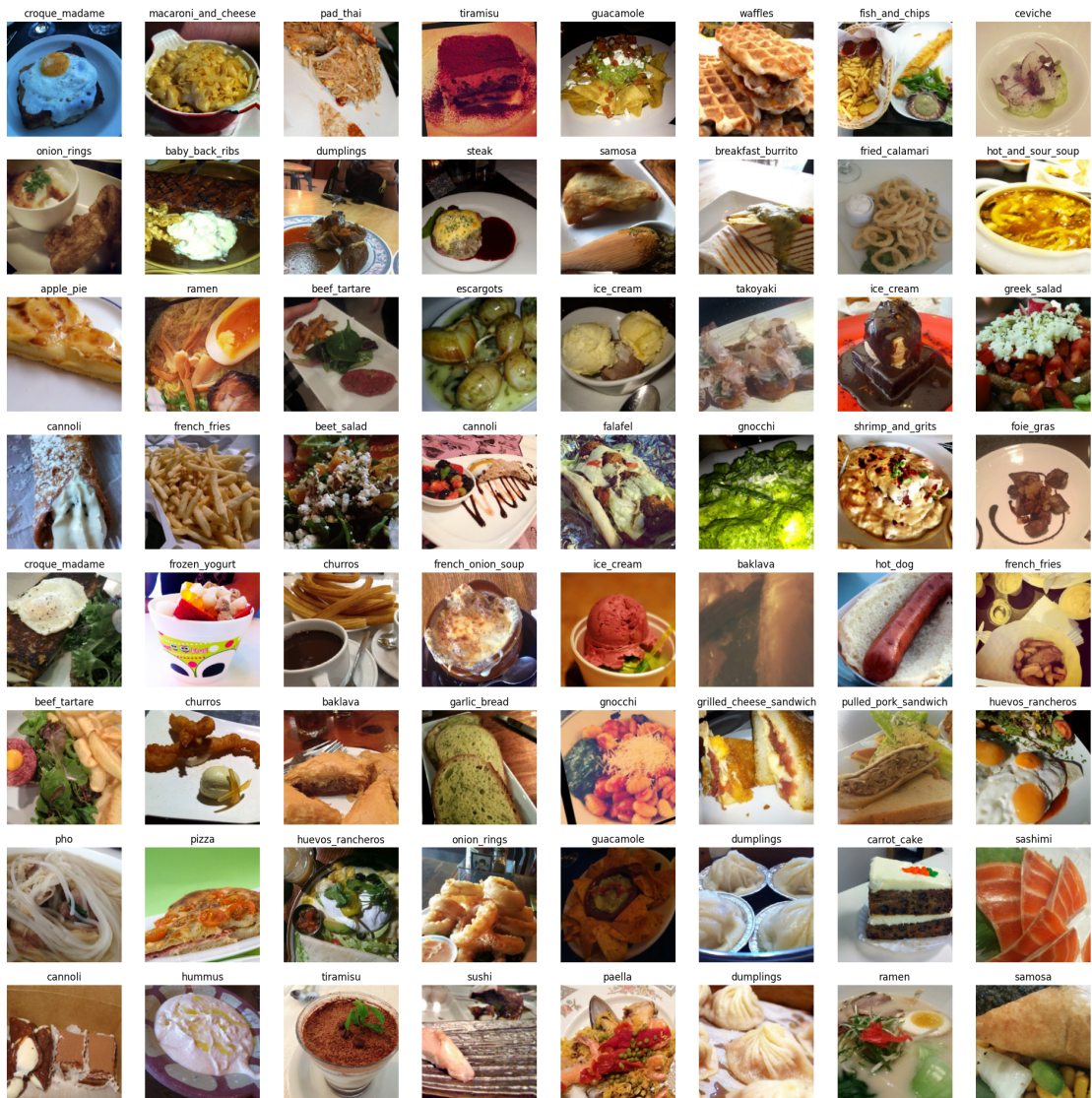


Figura 4.14: Batch da 64 immagini del training set con trasformazioni applicate.

In questo caso, gli esperimenti sono stati ancora più pesanti, se confrontati con quelli precedenti, in termini della quantità di tempo necessario per essere ultimati, in particolare per il training su immagini a 512 pixel. Ciò mi ha spinto ad usare un servizio di GPU hosting quale Vast.ai [68].

4.2.4 Training del modello

L'allenamento sulle immagini ridimensionate a 224 pixel è suddiviso in tre segmenti: i primi due di 8 epoche ognuno e l'ultimo di 4 epoche.

Per ogni segmento ho riportato il valore di learning rate ottimale, l'andamento della loss in forma grafica ed una tabella con l'andamento delle metriche nelle varie epoche.

Segmento 1

Il segmento 1 consiste in un training per 8 epoche usando le immagini di 224x224 pixel.

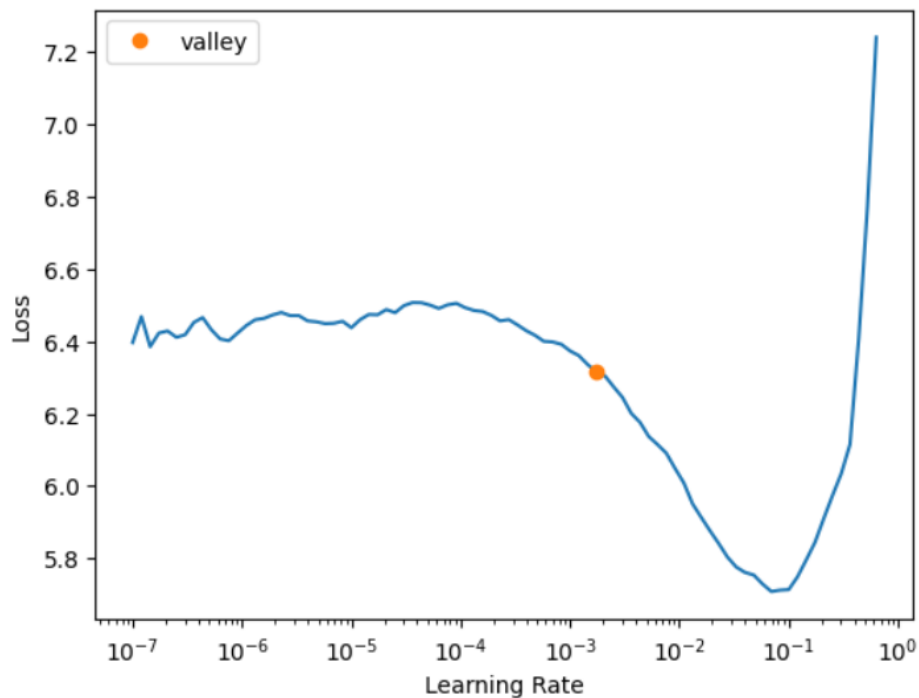


Figura 4.15: Risultato della funzione *lr_find* per il primo segmento.

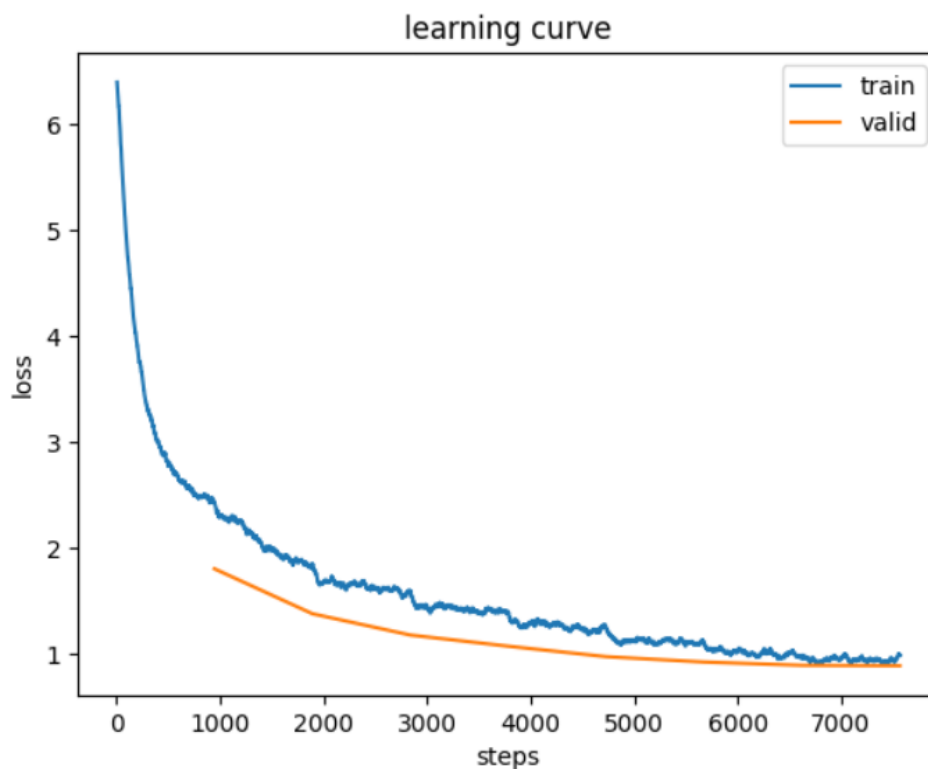


Figura 4.16: Andamento della loss nel primo segmento.

epoch	train_loss	valid_loss	accuracy	top_5_accuracy
1	2.436011	1.802552	0.535116	0.802838
2	1.843460	1.378887	0.638614	0.873399
3	1.593029	1.177917	0.691815	0.895710
4	1.388076	1.072685	0.718482	0.907591
5	1.248857	0.973894	0.742706	0.920066
6	1.107928	0.923303	0.758548	0.927591
7	0.982279	0.892472	0.766139	0.930759
8	0.988512	0.888574	0.767921	0.929967

Tabella 4.9: Andamento del modello nel primo segmento (prime 8 epoche).

Segmento 2

Nel secondo segmento ho continuato ad allenare il modello per altre 8 epoche con il valore del LR consigliato dalla funzione *lr_find*.

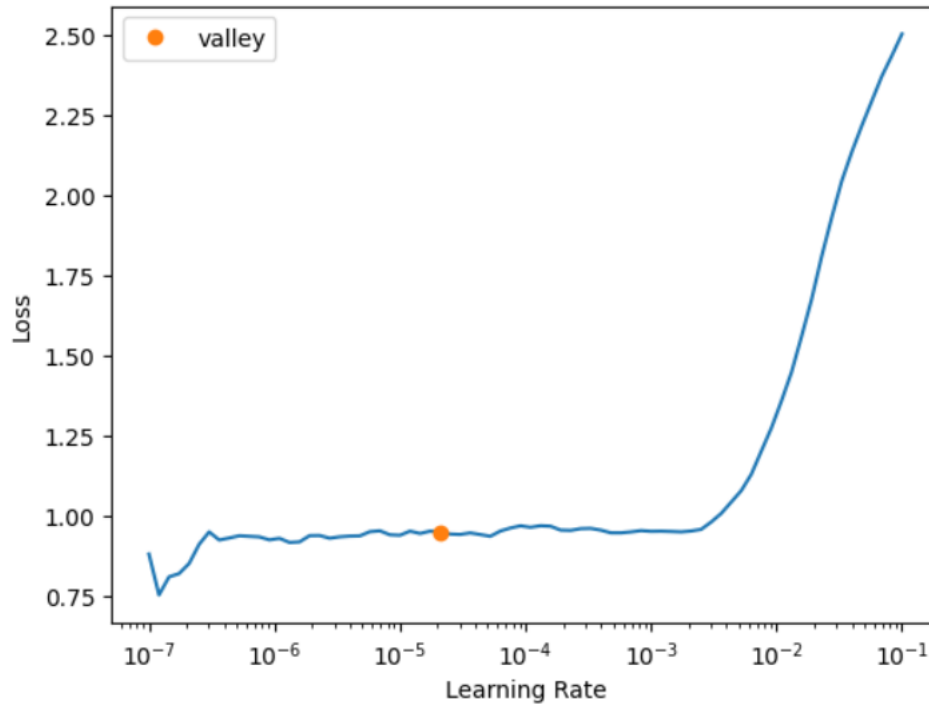


Figura 4.17: Risultato della funzione *lr_find* per il secondo segmento.

epoch	train_loss	valid_loss	accuracy	top_5_accuracy
9	0.943434	0.848450	0.777426	0.934851
10	0.889917	0.801777	0.789505	0.939142
11	0.800551	0.767748	0.801716	0.943564
12	0.699470	0.736252	0.812475	0.946733
13	0.575443	0.720351	0.814521	0.948119
14	0.480284	0.716304	0.820066	0.949571
15	0.423155	0.711077	0.821584	0.949901
16	0.411359	0.713815	0.821980	0.950759

Tabella 4.10: Andamento del modello nel secondo segmento (dalla nona alla sedicesima epoca).

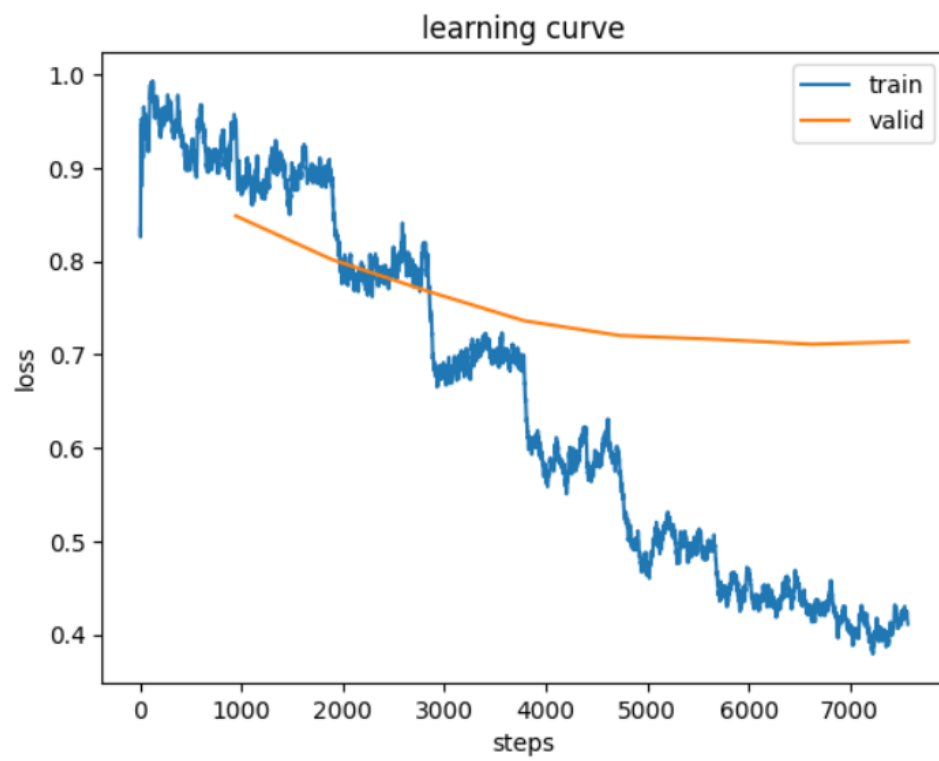


Figura 4.18: Andamento della loss nel secondo segmento.

Segmento 3

Dopo il segmento 2 possiamo notare una tendenza del modello a convergere, infatti se consideriamo i valori dell'accuracy abbiamo dei cambiamenti lievi tra la 14 e la 16 epoca.

Quindi, in questo segmento ho allenato il modello solo per 4 epoche per vedere se tale andamento venisse confermato.

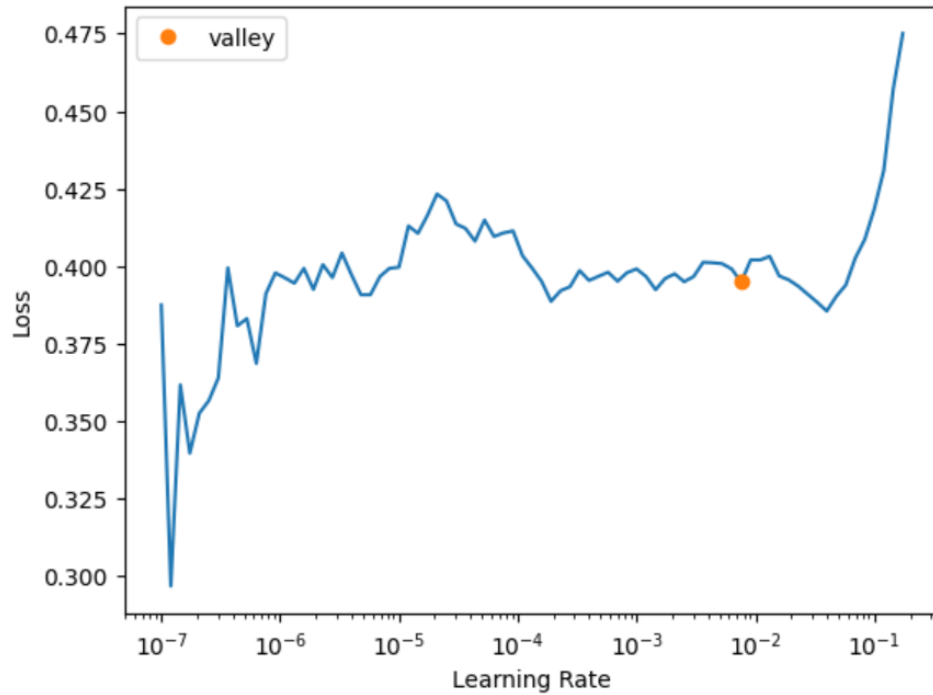


Figura 4.19: Risultato della funzione *lr_find* per il terzo segmento.

epoch	train_loss	valid_loss	accuracy	top_5_accuracy
17	0.379002	0.731765	0.820660	0.950429
18	0.411970	0.741214	0.820330	0.948845
19	0.391681	0.736644	0.822178	0.949505
20	0.402193	0.737916	0.822178	0.950231

Tabella 4.11: Andamento del modello nel terzo segmento (dalla diciassettesima alla ventesima epoca).

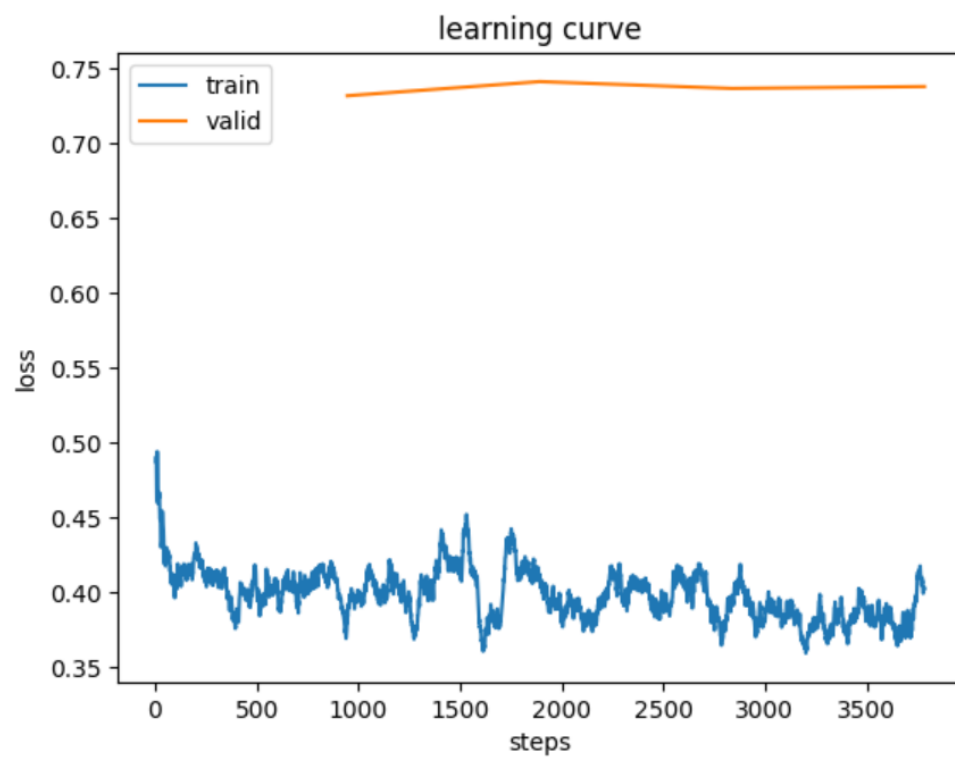


Figura 4.20: Andamento della loss nel terzo segmento.

Segmento 4

Come ci si poteva aspettare, la tendenza del modello a convergere viene confermata dal segmento 3, come si può notare dall'andamento dell'accuracy e della loss (vedi tabella 4.11 e Fig. 4.20).

Quindi, da ora in poi le immagini passeranno da una dimensione di 224x224 pixel ad una dimensione di 512x512 pixel, il tipo di trasformazioni applicate sono le stesse dei segmenti precedenti.

Per semplicità, il conteggio delle epoche riparte da 1.

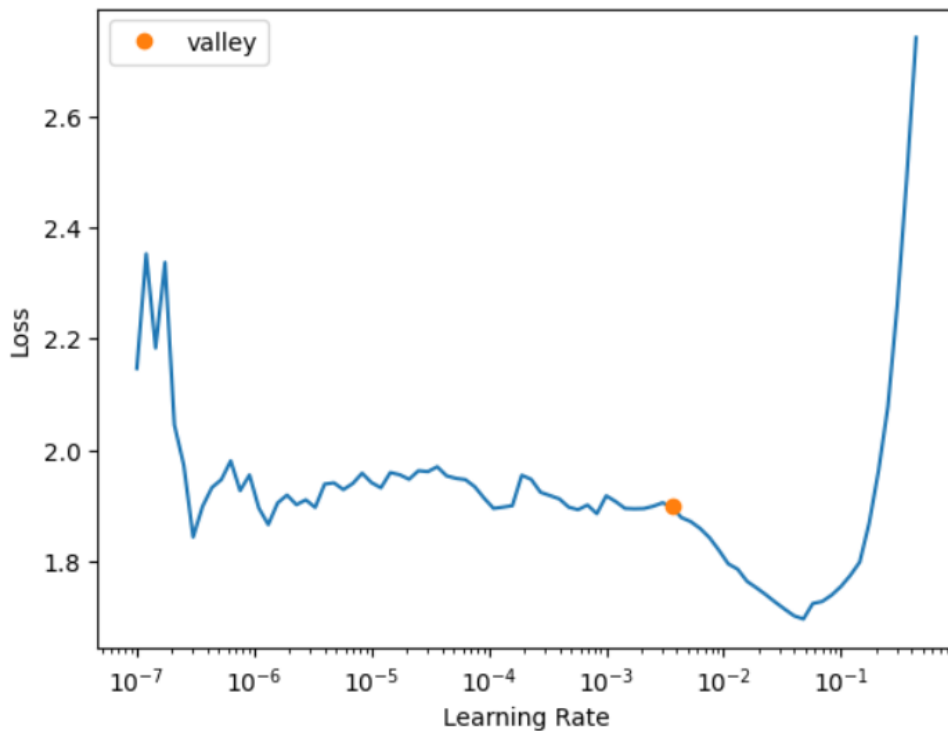


Figura 4.21: Risultato della funzione *lr_find* per il quarto segmento.

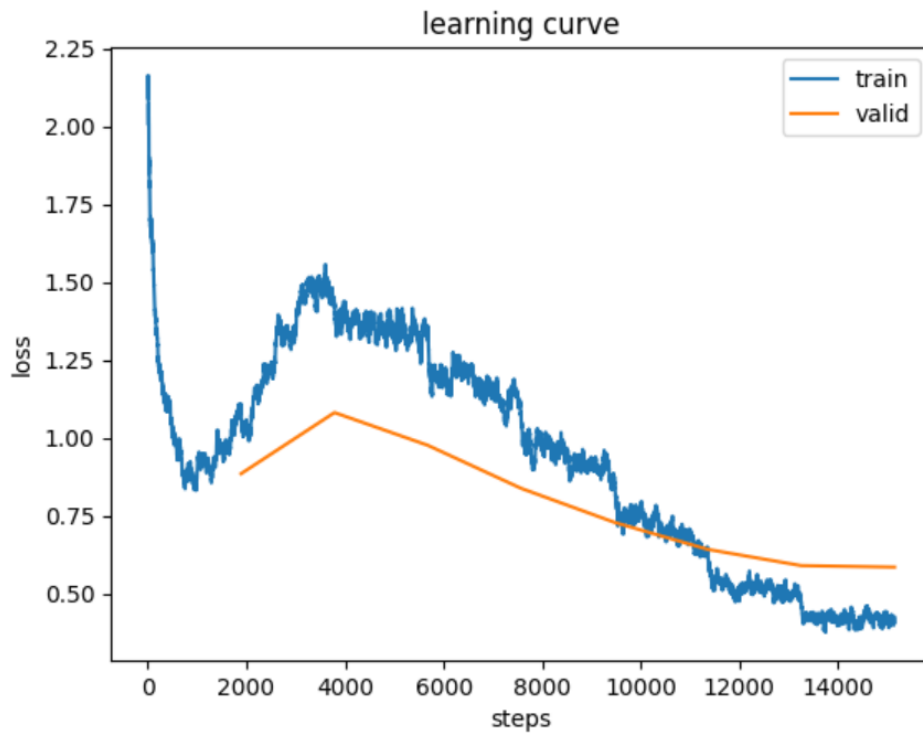


Figura 4.22: Andamento della loss nel quarto segmento.

epoch	train_loss	valid_loss	accuracy	top_5_accuracy
1	1.081589	0.884247	0.776436	0.934323
2	1.425868	1.080256	0.723234	0.912805
3	1.330845	0.975123	0.747987	0.923432
4	1.065678	0.836973	0.786073	0.938812
5	0.864880	0.727788	0.812277	0.951485
6	0.640096	0.641030	0.838482	0.958284
7	0.467898	0.589055	0.853531	0.961386
8	0.410582	0.584098	0.855578	0.963432

Tabella 4.12: Andamento del modello nel quarto segmento (prime 8 epoche).

Segmento 5

Nel segmento 5 si continua ad allenare il modello per 4 epoche abbassando il valore del LR.

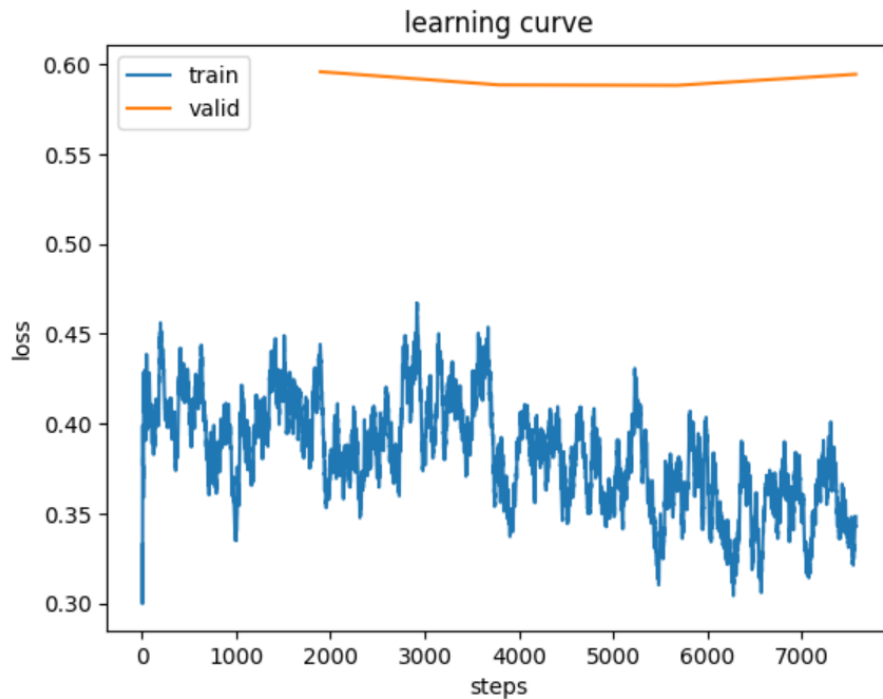


Figura 4.23: Andamento della loss nel quinto segmento.

epoch	train_loss	valid_loss	accuracy	top_5_accuracy
5	0.442609	0.595687	0.856172	0.962310
6	0.386445	0.588376	0.857162	0.962574
7	0.366020	0.588129	0.858944	0.962046
8	0.348239	0.594278	0.858878	0.962244

Tabella 4.13: Andamento del modello nel quinto segmento (dalla quinta alla ottava epoca).

Segmento 6

Ancora una volta si assiste ad una tendenza del modello a convergere, quindi nel sesto segmento, ho deciso di allenare il modello un'ultima volta per 4 epoche.

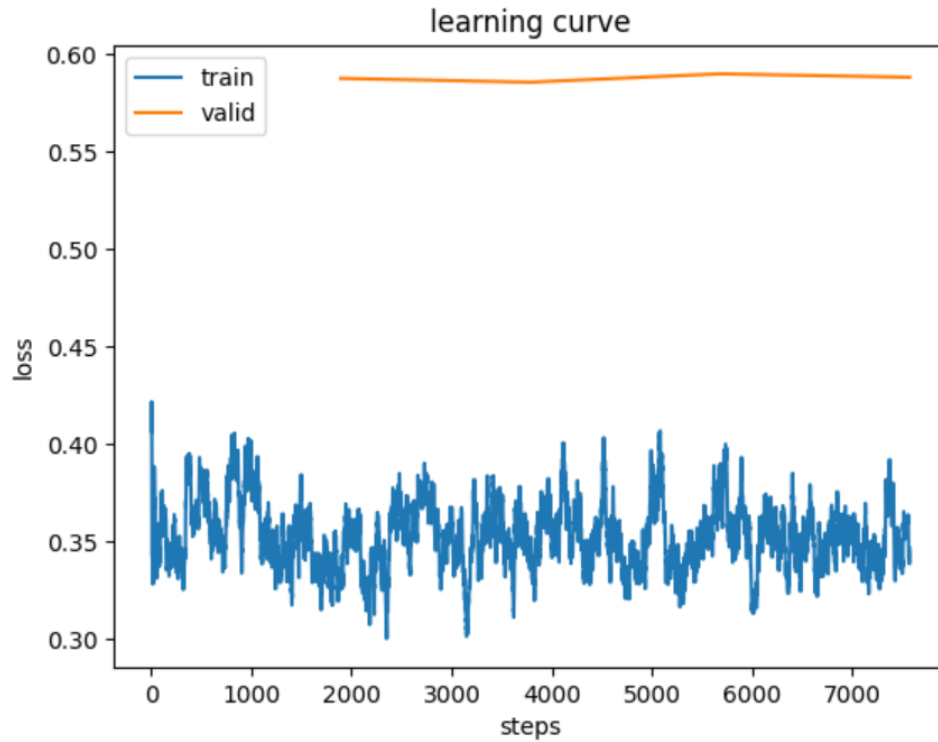


Figura 4.24: Andamento della loss nel sesto segmento.

epoch	train_loss	valid_loss	accuracy	top_5_accuracy
9	0.325627	0.587484	0.858218	0.962838
10	0.337324	0.585538	0.860132	0.962112
11	0.384130	0.589816	0.858152	0.962508
12	0.346784	0.588068	0.858482	0.962640

Tabella 4.14: Andamento del modello nel sesto segmento (dalla nona alla dodicesima epoca).

4.2.5 Test del modello

Il modello appena allenato è stato testato sul test set usando TTA (test time augmentation) e non usandola.

loss	accuracy	top_5_accuracy
0.4072	0.8899	0.9793

Tabella 4.15: Valori di loss, accuracy e top 5 accuracy senza TTA.

accuracy	top_5_accuracy
0.8939	0.9809

Tabella 4.16: Valori di accuracy e top 5 accuracy con TTA.

Come si evince dalle tabelle 4.15 e 4.16, applicare TTA porta ad un leggero miglioramento delle performance, tuttavia, rallenta leggermente il modello nell'effettuare la prediction.

Di seguito è riportata la matrice di confusione (confusion matrix) del modello sul test set (Fig 4.25). La matrice mette in risalto come gli errori più frequenti si riscontrino in classi che si somigliano fortemente come ad esempio la classe "apple pie" che viene classificata erroneamente come "bread pudding" e viceversa.

In tal senso in una applicazione reale una scelta intelligente potrebbe essere fornire all'utente non una sola la classe, ma più classi.

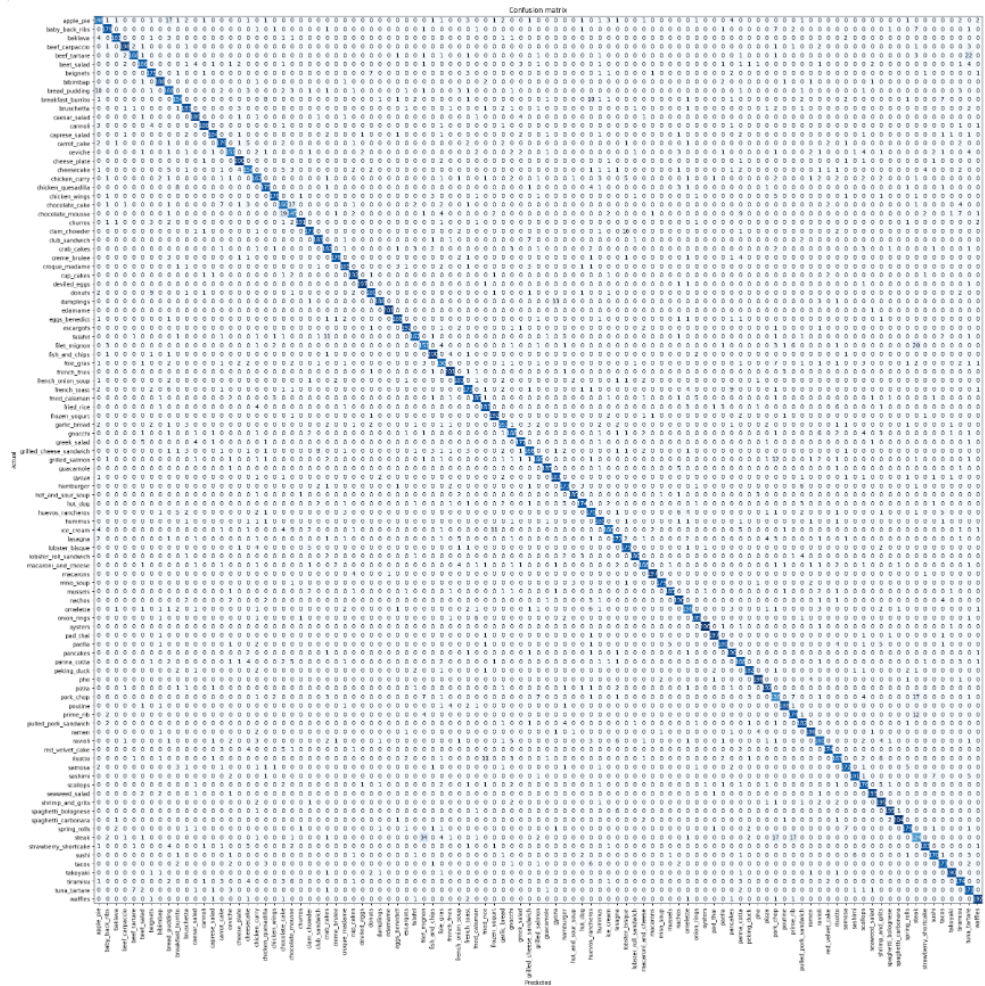


Figura 4.25: Confusion matrix sul test set.

Capitolo 5

Demo

Per mostrare le funzionalità del sistema nella sua interezza è stata realizzata una demo interattiva con una GUI realizzata in Python usando PyQt5 [69].

5.0.1 Caricamento dell'immagine

Il primo passo è ovviamente quello del caricamento dell'immagine che verrà processata dall'applicazione.

Per il caricamento all'utente basterà cliccare sul pulsante "Load image" (Fig. 5.2) che mostrerà una finestra che gli consentirà di caricare un file presente sul proprio pc.

I file devono essere immagini quindi i formati consentiti sono: .jpg, .png e .bmp (Fig. 5.2).

Nel caso di un'implementazione con un'applicazione smartphone, l'acquisizione potrà avvenire direttamente dalla fotocamera.

Una volta scelta l'immagine, questa verrà passata al modello ResNet50 allenato su Fastai, che restituirà le 5 classi con la relativa probabilità. Il risultato viene mostrato nella colonna a sinistra dell'immagine (Fig. 5.3).

5.0.2 Segmentazione

In caso di corretta classificazione, l'utente può selezionare tramite radio box la classe e l'immagine caricata potrà essere segmentata (Fig. 5.4). Per tale funzione ho implementato due possibilità:

- Un approccio automatico attraverso SAM;
- Un approccio manuale.

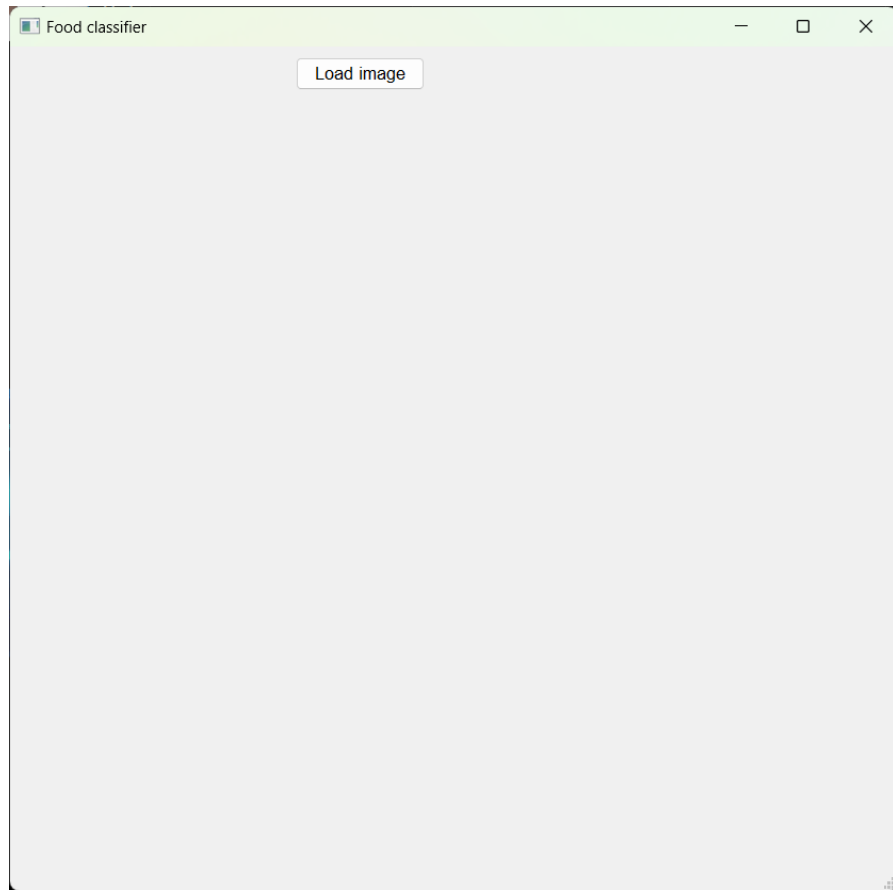


Figura 5.1: Schermata iniziale all'avvio della demo.

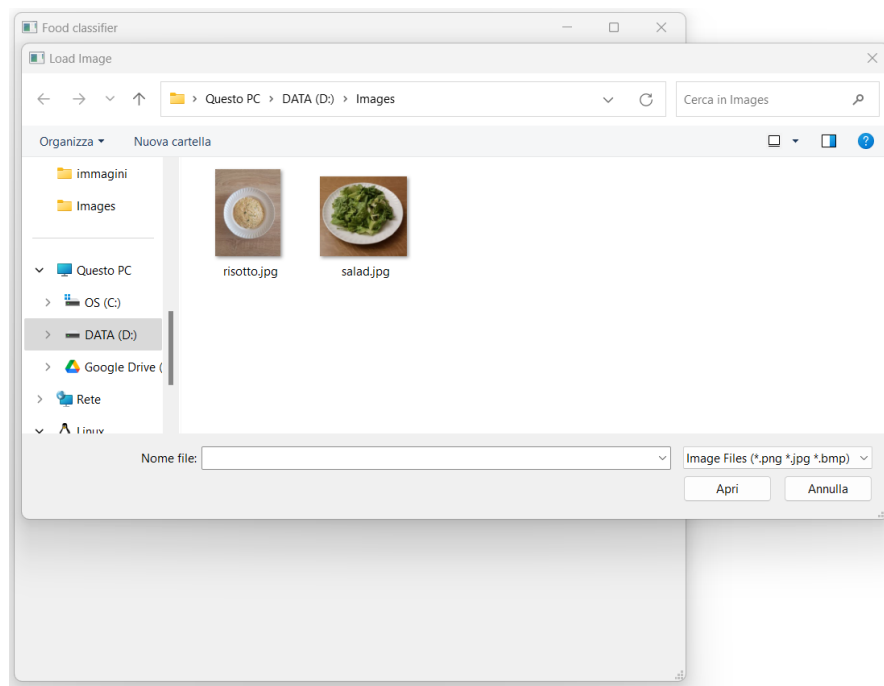


Figura 5.2: Schermata di caricamento dell'immagine dal pc.

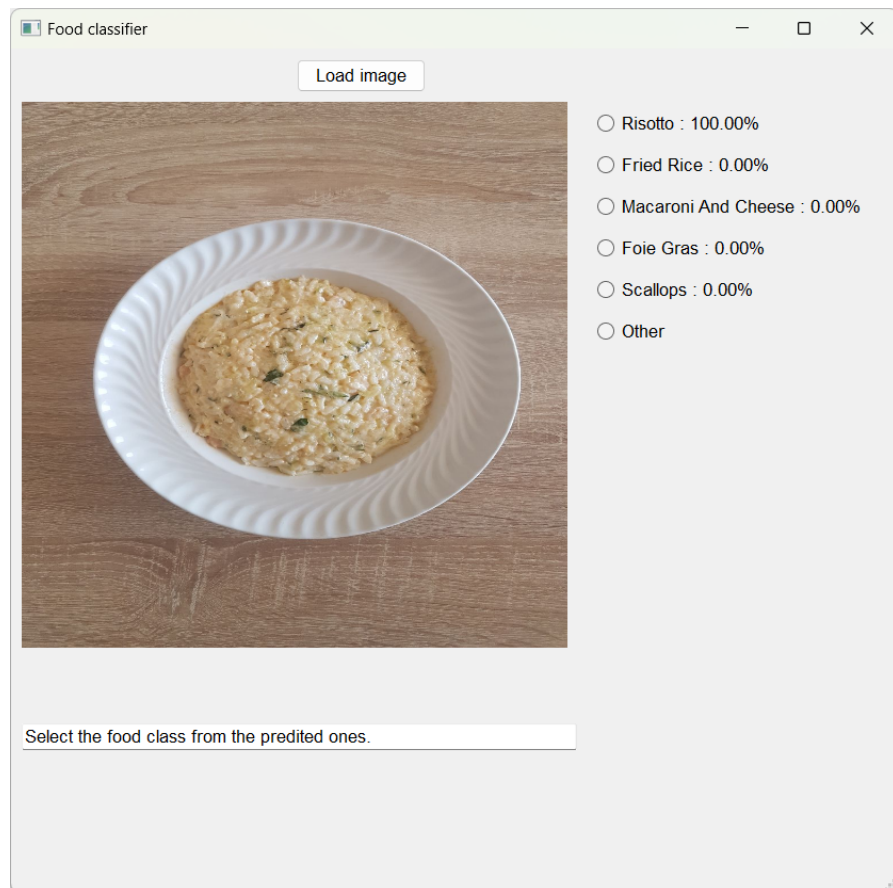


Figura 5.3: Schermata dopo aver caricato l'immagine con le classi predette.

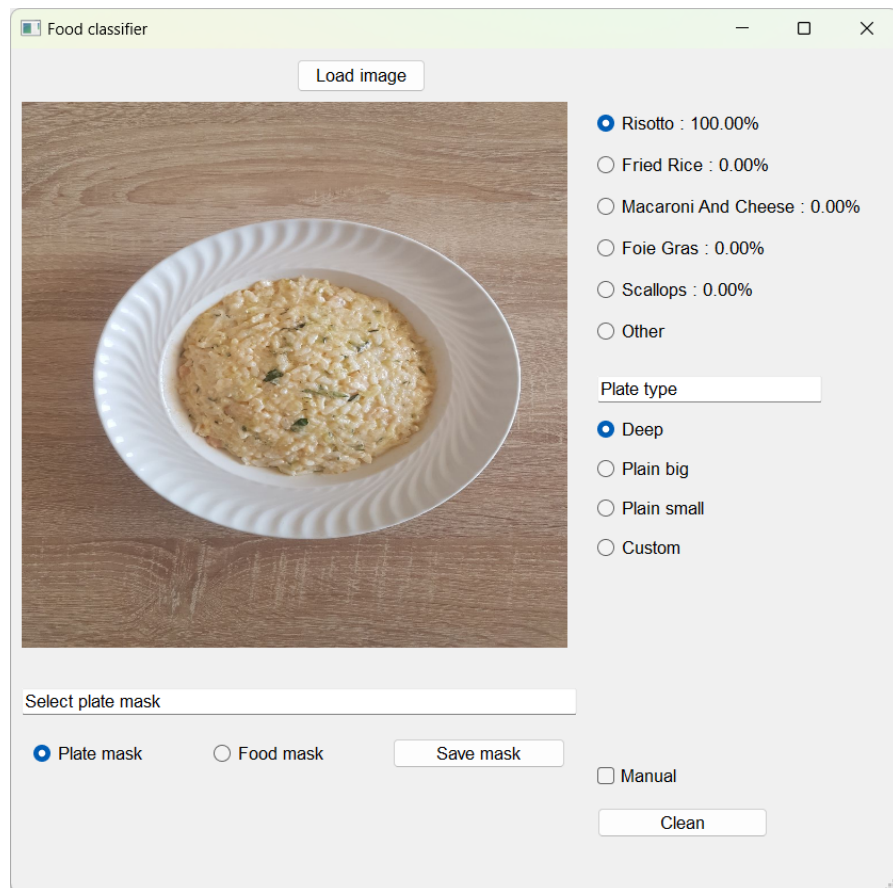


Figura 5.4: Schermata dopo aver selezionato la classe.

Approccio con SAM

Nell'approccio automatico, l'utente potrà utilizzare SAM per segmentare l'immagine caricata. Per tale processo basterà passare con il mouse sull'immagine tenendo premuto il tasto sinistro ed in tempo reale verrà mostrata la maschera ottenuta (Fig. 5.5 e 5.6).

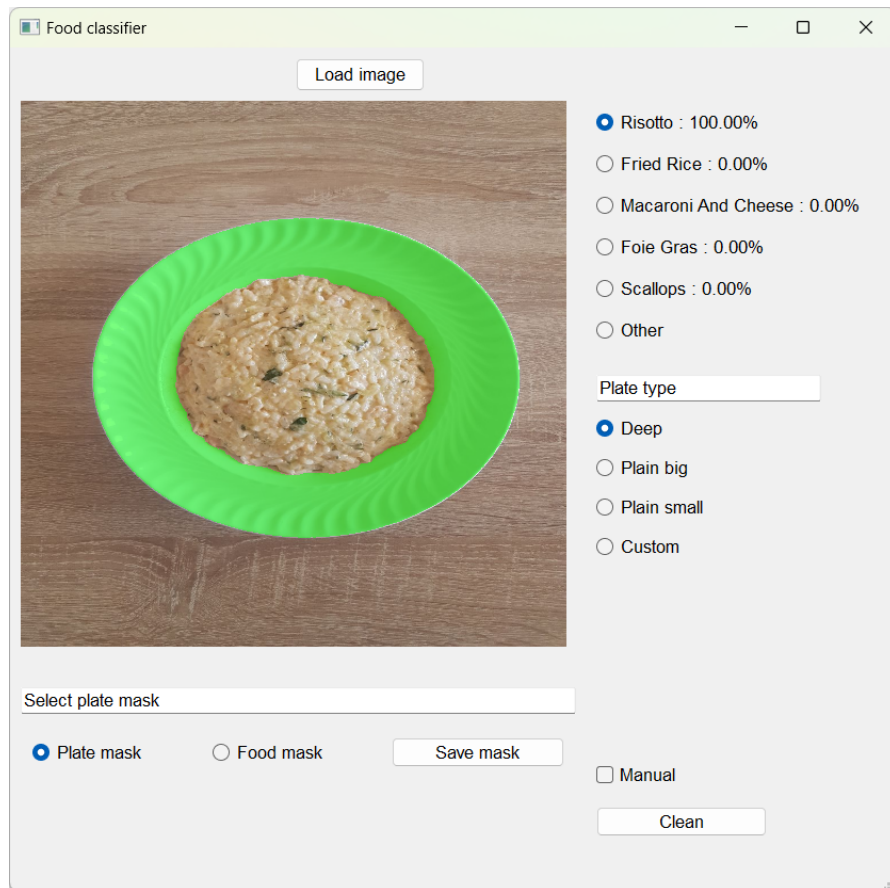


Figura 5.5: Maschera del piatto ottenuta usando SAM.

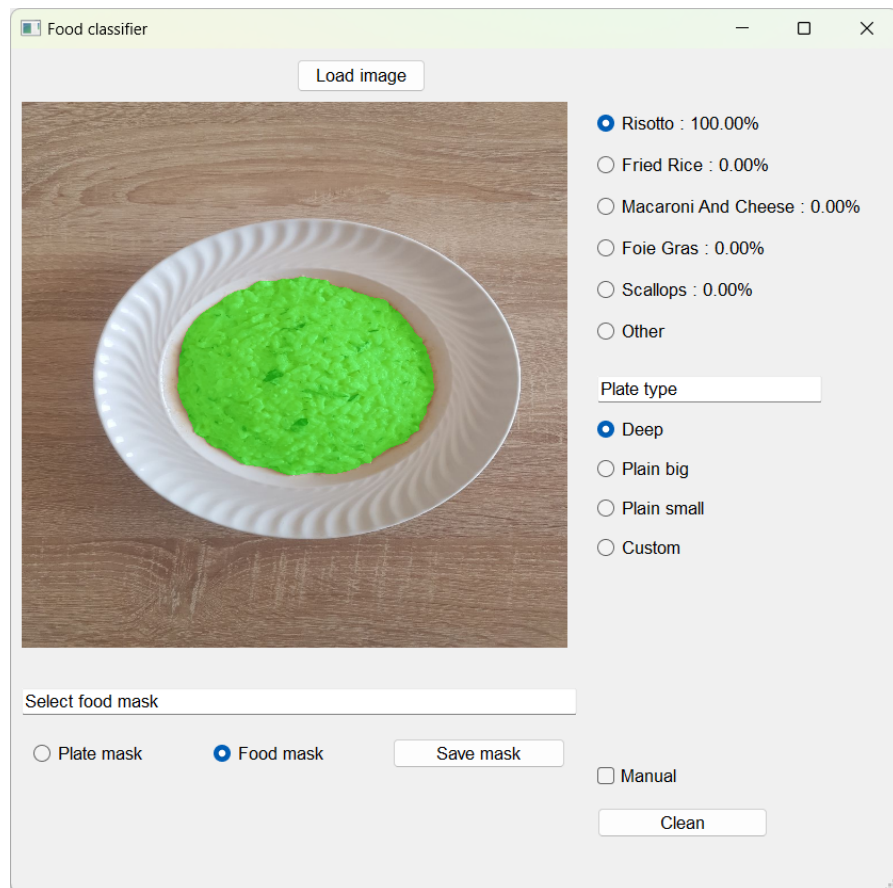


Figura 5.6: Maschera del cibo ottenuta usando SAM.

Approccio manuale

Per i casi più complicati ho deciso di dare la possibilità all'utente di tracciare manualmente il contorno del cibo/piatto ed ottenere la corrispondente maschera. Per poter rendere il tracciamento il più facile possibile ho scelto di optare per un tracciamento semi-guidato; infatti l'utente non dovrà tracciare il contorno direttamente, ma gli basterà segnare dei punti sul contorno e potrà concludere il tracciamento premendo il tasto destro del mouse. In questo modo il contorno ottenuto avrà la forma di un poligono.

Ciò, a mio avviso, non solo facilita l'utente, ma rende anche facile ottenere la maschera a partire dal contorno in quanto vi è la certezza che il contorno disegnato abbia la forma di una linea chiusa (il punto finale, quello ottenuto premendo il tasto destro del mouse, coincide con il primo punto scelto dall'utente).

Anche in questo caso in tempo reale l'utente avrà la possibilità di vedere il risultato del suo lavoro (Fig. 5.7).

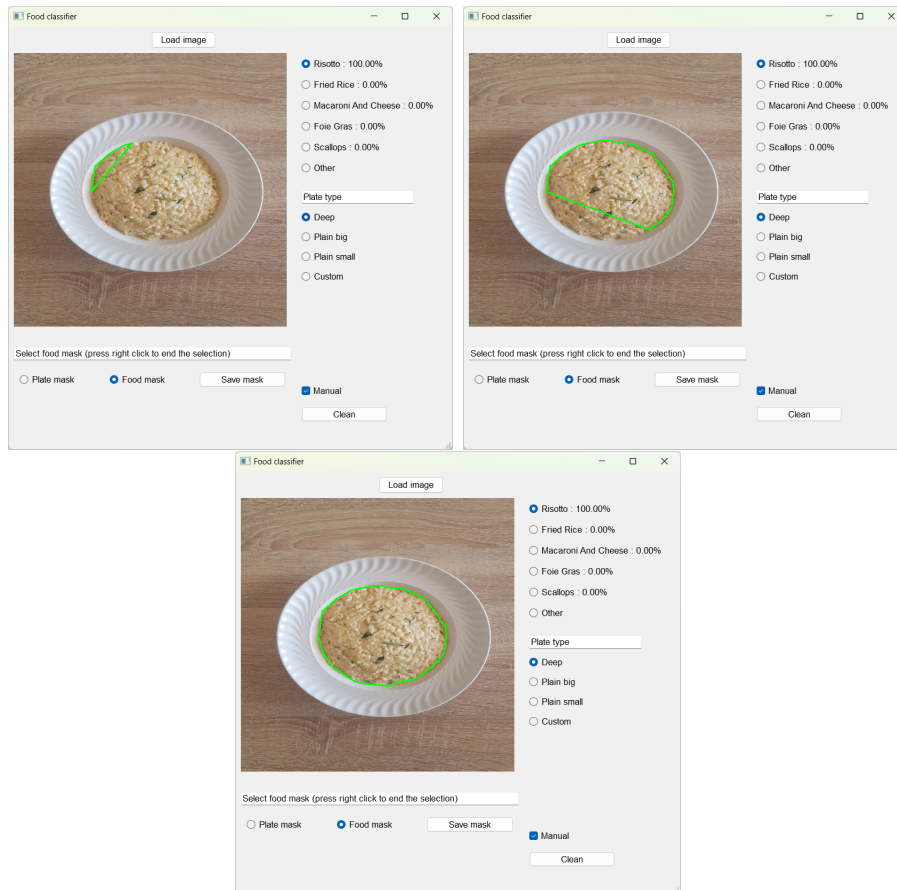


Figura 5.7: Esempio di segmentazione manuale.

5.0.3 Stima del volume

Per poter stimare il volume l'utente dovrà ottenere la maschera del piatto e dell'alimento da stimare.

Nel caso in cui la maschera del piatto sia stata ottenuta in modalità automatica, dovrà essere salvata premendo il tasto "Save mask"; mentre per il cibo l'utente avrà la possibilità di aggiungere più maschere, anche in questo caso per salvare la maschera dovrà essere premuto il tasto "Save mask".

Nel caso di acquisizione manuale della maschera questa verrà automaticamente una volta premuto il tasto destro del mouse.

Per ottenere la maschera del piatto o del cibo basta usare il radio box.

Nel caso in cui la maschera non sia soddisfacente si può facilmente scartare premendo il tasto "Clean"; ciò è possibile a patto che questa non sia stata salvata.

Una volta salvate le maschere del piatto ed almeno una maschera di un cibo comparirà il tasto "Estimate volume" che permetterà all'utente di ottenere una stima del volume di cibo presente nel piatto (Fig. 5.8).

Scelta tipo di piatto

Per poter effettuare una stima più precisa sarà necessario conoscere le caratteristiche fisiche del piatto, per tale ragione ho aggiunto un menù che permette di scegliere la tipologia di piatto che si sta usando.

L'utente potrà scegliere la tipologia di piatto velocemente passando tra:

- Piatto fondo:
 - Altezza: 3,7 cm;
 - Diametro: 22 cm.
- Piatto piano grande:
 - Altezza: 3,1 cm;
 - Diametro: 24,8 cm.
- Piatto piano piccolo:
 - Altezza: 2,3 cm;
 - Diametro: 19,6 cm.

A queste scelte veloci si aggiunge la possibilità di specificare direttamente le caratteristiche del piatto scegliendo la modalità "Custom".

In questo caso si aprirà un menù sottostante dove l'utente potrà inserire le caratteristiche del suo piatto scegliendone altezza e diametro dello stesso (Fig. 5.9).

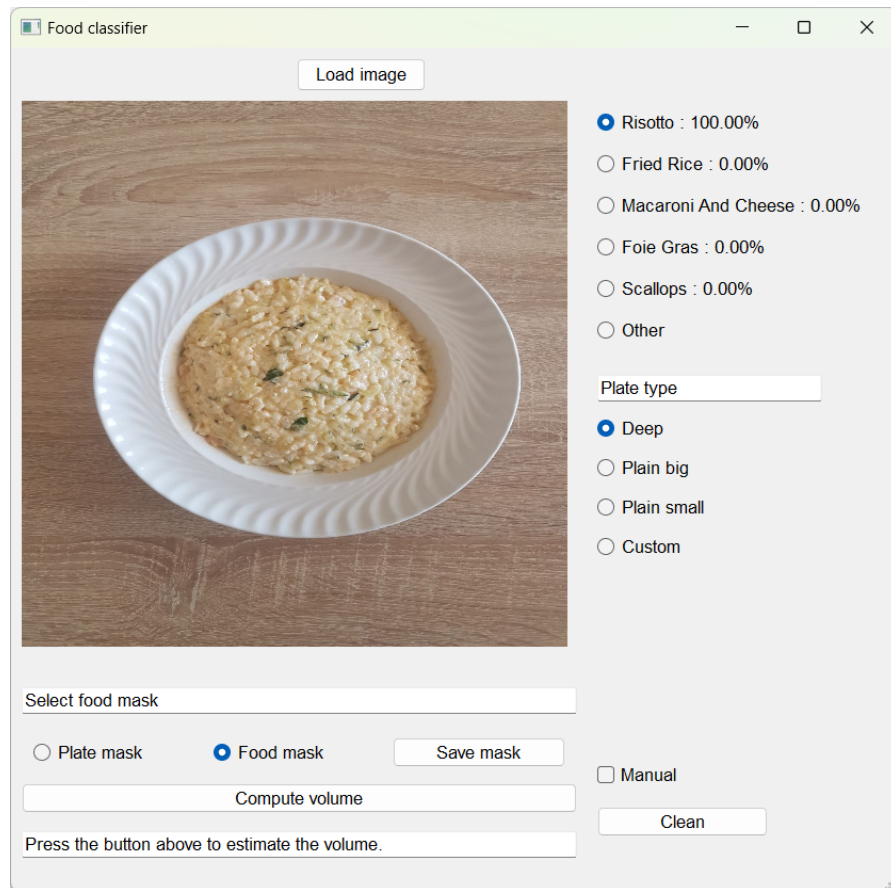


Figura 5.8: Schermata dopo aver selezionato le maschere del piatto e del cibo.

Calcolo del volume

Il volume viene calcolato attraverso una funzione da me implementata.

Questa funzione riceve in ingresso l'immagine originale, le maschere del piatto, una o più maschere di cibi, il diametro e l'altezza del piatto (Fig. 5.10).

Il primo passo per la stima del volume consiste nell'ottenere la mappa di profondità dell'immagine originale attraverso MiDaS [5] (Fig. 5.11). Poi viene calcolata l'area del piatto, il cui valore, conoscendo l'area reale del piatto, verrà usato per ottenere l'area del singolo pixel.

L'area del piatto corrisponde alla circonferenza con diametro pari a quello in ingresso.

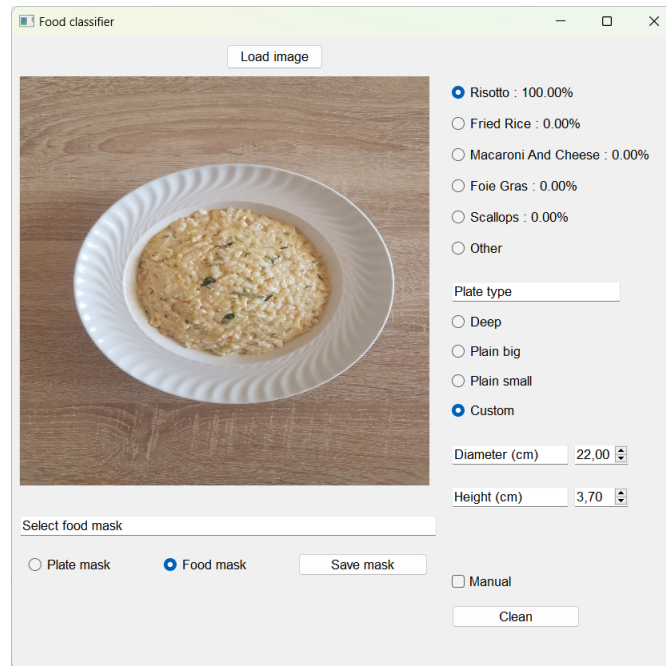


Figura 5.9: Menù che consente di specificare le caratteristiche del piatto.

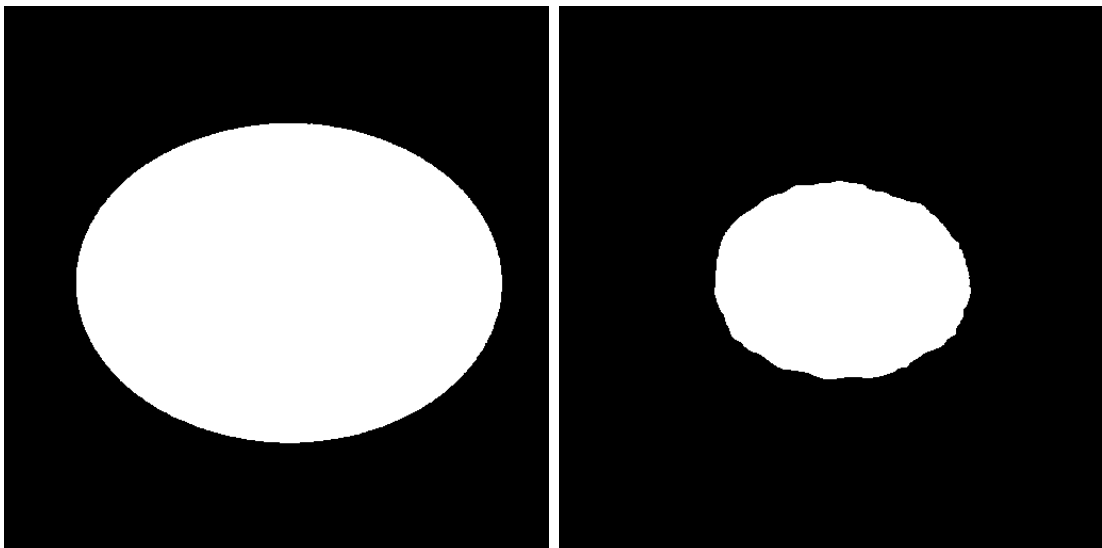


Figura 5.10: Esempio di maschere di piatto (a sinistra) e cibo (a destra) passate in ingresso alla funzione che calcola il volume.



Figura 5.11: Esempio di depth map ottenuta usando MiDaS.

Tramite la maschera del piatto viene ottenuto il suo contorno (Fig. 5.12) e di questo viene calcolata la media dei valori della depth map nei corrispettivi pixel. Il valore ottenuto può, quindi, essere utilizzato per ottenere per ogni maschera di cibi il valore di altezza corrispettivo e calcolare così il volume.

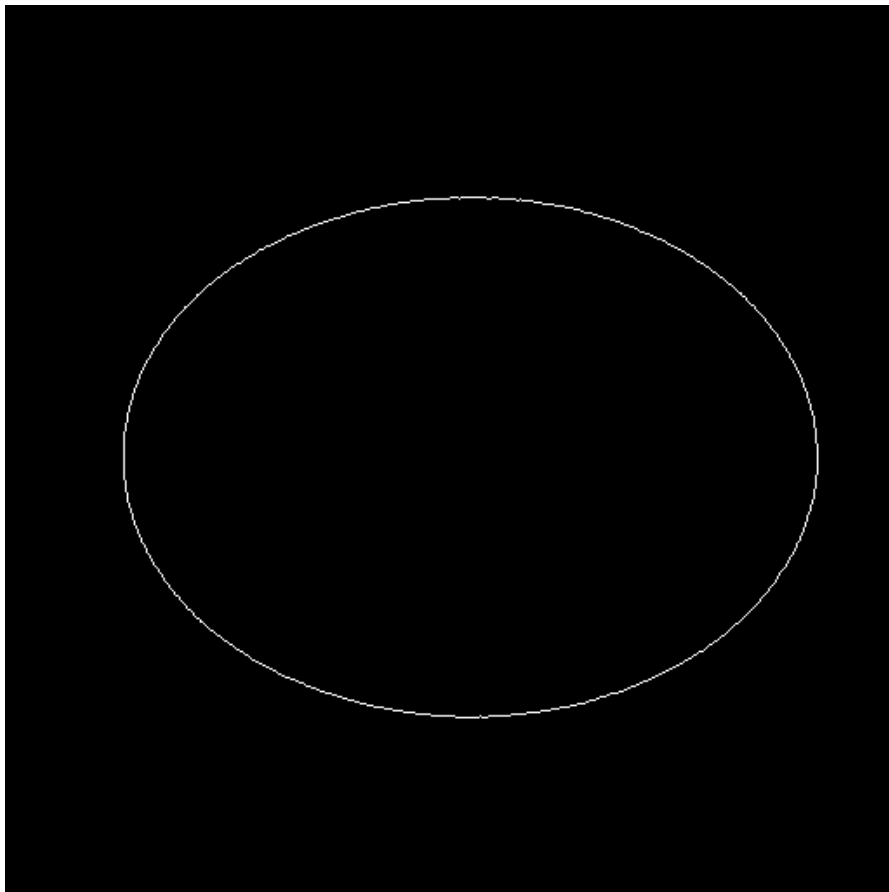


Figura 5.12: Esempio di contorno del piatto.

In particolare, il volume viene calcolato andando a sommare le altezze "scalate" per ogni maschera di cibo.

Nella demo per poter calcolare il volume deve essere premuto il tasto "Estimate volume". Il processo richiede un pò di tempo quindi appare una schermata che invita l'utente ad attendere che il processo finisca ed una volta terminato viene riportato il valore ottenuto (Fig. 5.13).

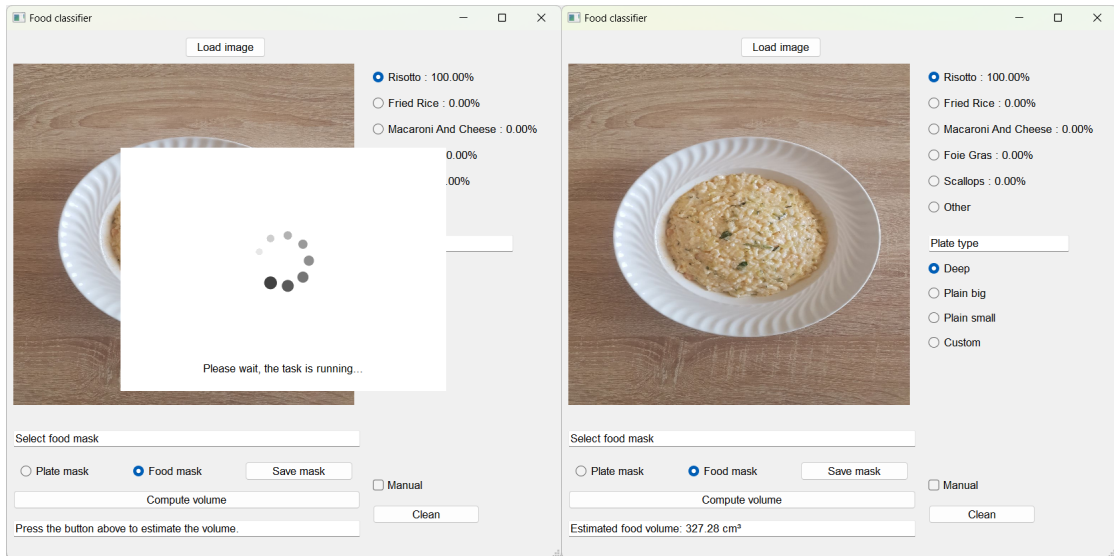


Figura 5.13: Schermata di attesa (sinistra) e nuova schermata che mostra il volume calcolato.

5.0.4 Risultati sul calcolo del volume

In questa sezione viene analizzata la precisione della funzione che calcola il volume. Purtroppo non è presente un dataset che possa essere usato da supporto quindi verranno solo considerate le immagini da me raccolte.

In totale ho raccolto 18 immagini principalmente primi piatti e insalate.

L'errore medio dei volumi calcolati rispetto al valore reale è del 19,89%.

Dai test ho notato come il valore del volume calcolato è fortemente influenzato dall'angolo con cui viene scattata la foto.

Un possibile miglioramento potrebbe essere rappresentato dalla costruzione di un dataset di mappe di profondità su immagini di alimenti con il quale allenare il modello MiDaS per rendere più accurata la depth map predetta.

Numero di immagini	Errore medio
18	19,89%

Tabella 5.1: Errore medio sulla stima del volume sulle immagini raccolte.

Capitolo 6

Conclusioni

Il presente lavoro di tesi aveva come obiettivo la realizzazione di un modello in grado di classificare correttamente le immagini di cibi e di stimarne il volume.

In seguito all'analisi di diversi approcci questo compito si è dimostrato particolarmente arduo.

Una delle difficoltà più grandi è la mancanza di dataset, quantomeno pubblici, che potessero essere usati per la stima del volume. Questo ha portato ad un rallentamento del lavoro causa la necessità di dover raccogliere personalmente le immagini e stimarne manualmente il volume.

Tuttavia, il lavoro presentato potrebbe rappresentare un importante punto di partenza per la realizzazione di un sistema più completo che dal calcolo del volume possa poi passare alla stima della quantità di cibo presente.

Ciò potrebbe essere possibile attraverso l'introduzione di un dataset che associ ad ogni ricetta la densità del cibo.

Altri possibili estensioni potrebbero essere rappresentate dall'introduzione di un dataset locale con ricette italiane su cui allenare i modelli proposti.

Appendice A

Categorie Food-101

- Apple pie
- Baby back ribs
- Baklava
- Beef carpaccio
- Beef tartare
- Beet salad
- Beignets
- Bibimbap
- Bread pudding
- Breakfast burrito
- Bruschetta
- Caesar salad
- Cannoli
- Caprese salad
- Carrot cake
- Ceviche
- Cheesecake

- Cheese plate
- Chicken curry
- Chicken quesadilla
- Chicken wings
- Chocolate cake
- Chocolate mousse
- Churros
- Clam chowder
- Club sandwich
- Crab cakes
- Creme brulee
- Croque madame
- Cup cakes
- Deviled eggs
- Donuts
- Dumplings
- Edamame
- Eggs benedict
- Escargots
- Falafel
- Filet mignon
- Fish and chips
- Foie gras
- French fries
- French onion soup

- French toast
- Fried calamari
- Fried rice
- Frozen yogurt
- Garlic bread
- Gnocchi
- Greek salad
- Grilled cheese sandwich
- Grilled salmon
- Guacamole
- Gyoza
- Hamburger
- Hot and sour soup
- Hot dog
- Huevos rancheros
- Hummus
- Ice cream
- Lasagna
- Lobster bisque
- Lobster roll sandwich
- Macaroni and cheese
- Macarons
- Miso soup
- Mussels
- Nachos

- Omelette
- Onion rings
- Oysters
- Pad thai
- Paella
- Pancakes
- Panna cotta
- Peking duck
- Pho
- Pizza
- Pork chop
- Poutine
- Prime rib
- Pulled pork sandwich
- Ramen
- Ravioli
- Red velvet cake
- Risotto
- Samosa
- Sashimi
- Scallops
- Seaweed salad
- Shrimp and grits
- Spaghetti bolognese
- Spaghetti carbonara

- Spring rolls
- Steak
- Strawberry shortcake
- Sushi
- Tacos
- Takoyaki
- Tiramisu
- Tuna tartare
- Waffles

Bibliografia

- [1] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: 2408.15857 [cs.CV]. URL: <https://arxiv.org/abs/2408.15857> (cit. alle pp. ii, 3, 28, 30–32, 49).
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: <https://arxiv.org/abs/1512.03385> (cit. alle pp. ii, 3, 11–16, 41, 58).
- [3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens e Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV]. URL: <https://arxiv.org/abs/1512.00567> (cit. alle pp. ii, 3, 17, 58).
- [4] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: <https://arxiv.org/abs/2304.02643> (cit. alle pp. ii, iii, 3, 32–39).
- [5] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler e Vladlen Koltun. *Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer*. 2020. arXiv: 1907.01341 [cs.CV]. URL: <https://arxiv.org/abs/1907.01341> (cit. alle pp. iii, 3, 40, 41, 43–48, 90).
- [6] *FatSecret*. URL: <https://www.fatsecret.it/> (cit. a p. 2).
- [7] *MyFitnessPal*. URL: <https://www.myfitnesspal.com/it> (cit. a p. 2).
- [8] Lukas Bossard, Matthieu Guillaumin e Luc Van Gool. «Food-101 – Mining Discriminative Components with Random Forests». In: *European Conference on Computer Vision*. 2014 (cit. alle pp. 4, 5, 58).
- [9] Mei Chen, Kapil Dhingra, Wen Wu, Lei Yang, Rahul Sukthankar e Jie Yang. «PFID: Pittsburgh fast-food image dataset». In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. 2009, pp. 289–292. DOI: 10.1109/ICIP.2009.5413511 (cit. a p. 6).

- [10] Xiongwei Wu, Xin Fu, Ying Liu, Ee-Peng Lim, Steven C. H. Hoi e Qianru Sun. *A Large-Scale Benchmark for Food Image Segmentation*. 2021. arXiv: 2105.05409 [cs.CV]. URL: <https://arxiv.org/abs/2105.05409> (cit. alle pp. 7–10, 49).
- [11] Takumi Ege, Wataru Shimoda e Keiji Yanai. «A New Large-scale Food Image Segmentation Dataset and Its Application to Food Calorie Estimation Based on Grains of Rice». In: *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*. MADiMa '19. Nice, France: Association for Computing Machinery, 2019, pp. 82–87. ISBN: 9781450369169. DOI: 10.1145/3347448.3357162. URL: <https://doi.org/10.1145/3347448.3357162> (cit. alle pp. 7, 8).
- [12] Kaimu Okamoto e Keiji Yanai. «UEC-FoodPix Complete: A Large-Scale Food Image Segmentation Dataset». In: *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part V*. Berlin, Heidelberg: Springer-Verlag, 2021, pp. 647–659. ISBN: 978-3-030-68820-2. DOI: 10.1007/978-3-030-68821-9_51. URL: https://doi.org/10.1007/978-3-030-68821-9_51 (cit. alle pp. 7, 8).
- [13] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber e Antonio Torralba. «Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images». In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019) (cit. alle pp. 7, 8).
- [14] Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu e Thomas S. Huang. *CCNet: Criss-Cross Attention for Semantic Segmentation*. 2020. arXiv: 1811.11721 [cs.CV]. URL: <https://arxiv.org/abs/1811.11721> (cit. a p. 9).
- [15] Sixiao Zheng et al. *Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers*. 2021. arXiv: 2012.15840 [cs.CV]. URL: <https://arxiv.org/abs/2012.15840> (cit. a p. 9).
- [16] Alexander Kirillov, Ross Girshick, Kaiming He e Piotr Dollár. *Panoptic Feature Pyramid Networks*. 2019. arXiv: 1901.02446 [cs.CV]. URL: <https://arxiv.org/abs/1901.02446> (cit. a p. 9).
- [17] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth e Bernt Schiele. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. arXiv: 1604.01685 [cs.CV]. URL: <https://arxiv.org/abs/1604.01685> (cit. a p. 9).
- [18] *ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)*. <https://image-net.org/challenges/LSVRC/2015/results.php> (cit. a p. 11).

- [19] A. Krizhevsky I. Sutskever e G. Hinton. «Imagenet classification with deep convolutional neural networks». In: *NIPS*. 2012 (cit. alle pp. 11, 17, 22).
- [20] Karen Simonyan e Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: <https://arxiv.org/abs/1409.1556> (cit. alle pp. 11, 14, 16, 17, 22, 24).
- [21] Kaiming He e Jian Sun. *Convolutional Neural Networks at Constrained Time Cost*. 2014. arXiv: 1412.1710 [cs.CV]. URL: <https://arxiv.org/abs/1412.1710> (cit. a p. 11).
- [22] Rupesh Kumar Srivastava, Klaus Greff e Jürgen Schmidhuber. *Highway Networks*. 2015. arXiv: 1505.00387 [cs.LG]. URL: <https://arxiv.org/abs/1505.00387> (cit. a p. 11).
- [23] Alex Krizhevsky, Vinod Nair e Geoffrey Hinton. *CIFAR-10 (Canadian Institute for Advanced Research)*. 2009. URL: <http://www.cs.toronto.edu/~kriz/cifar.html> (cit. a p. 12).
- [24] Sepp Hochreiter. «Untersuchungen zu dynamischen neuronalen Netzen». In: *Master's thesis, Institut für Informatik, Technische Universität, München* 1 (1991), pp. 1–150 (cit. a p. 12).
- [25] Y. Bengio, P. Simard e P. Frasconi. «Learning long-term dependencies with gradient descent is difficult». In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181 (cit. a p. 12).
- [26] Xavier Glorot e Yoshua Bengio. «Understanding the difficulty of training deep feedforward neural networks». In: *International Conference on Artificial Intelligence and Statistics*. 2010. URL: <https://api.semanticscholar.org/CorpusID:5575601> (cit. a p. 12).
- [27] Yann A. LeCun, Léon Bottou, Genevieve B. Orr e Klaus Robert Müller. «Efficient backprop». In: *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 9–50 (cit. a p. 12).
- [28] Andrew M. Saxe, James L. McClelland e Surya Ganguli. *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*. 2014. arXiv: 1312.6120 [cs.NE]. URL: <https://arxiv.org/abs/1312.6120> (cit. a p. 12).
- [29] Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV]. URL: <https://arxiv.org/abs/1502.01852> (cit. a p. 12).

- [30] Sergey Ioffe e Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: <https://arxiv.org/abs/1502.03167> (cit. alle pp. 12, 22).
- [31] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV]. URL: <https://arxiv.org/abs/1409.0575> (cit. alle pp. 15, 16, 25, 41).
- [32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke e Andrew Rabinovich. «Going deeper with convolutions». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594 (cit. alle pp. 17–25).
- [33] Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: <https://arxiv.org/abs/1506.02640> (cit. alle pp. 26, 27).
- [34] Ross Girshick, Jeff Donahue, Trevor Darrell e Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV]. URL: <https://arxiv.org/abs/1311.2524> (cit. a p. 26).
- [35] Juan Terven, Diana-Margarita Córdova-Esparza e Julio-Alejandro Romero-González. «A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS». In: *Machine Learning and Knowledge Extraction* 5.4 (nov. 2023), pp. 1680–1716. ISSN: 2504-4990. DOI: 10.3390/make5040083. URL: <http://dx.doi.org/10.3390/make5040083> (cit. alle pp. 26, 29).
- [36] *Brief summary of YOLOv8 model structure*. 2023. URL: <https://github.com/ultralytics/ultralytics/issues/189> (cit. a p. 29).
- [37] C.-Y. Wang A. Bochkovskiy e H.-Y. M. Liao. «You only look once, but improved». In: *CVPR 2023*. 2023 (cit. a p. 31).
- [38] J. Redmon e A. Farhadi. «Yolo reborn: Scaling and refining the you only look once architecture.» In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 2023 (cit. a p. 31).
- [39] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: <https://arxiv.org/abs/1912.01703> (cit. a p. 32).
- [40] *roboflow*. <https://roboflow.com/> (cit. a p. 32).

- [41] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: <https://arxiv.org/abs/2010.11929> (cit. a p. 34).
- [42] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár e Ross Girshick. *Masked Autoencoders Are Scalable Vision Learners*. 2021. arXiv: 2111.06377 [cs.CV]. URL: <https://arxiv.org/abs/2111.06377> (cit. a p. 34).
- [43] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: <https://arxiv.org/abs/2103.00020> (cit. a p. 35).
- [44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser e Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: <https://arxiv.org/abs/1706.03762> (cit. a p. 35).
- [45] Alina Kuznetsova et al. «The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale». In: *International Journal of Computer Vision* 128.7 (mar. 2020), pp. 1956–1981. ISSN: 1573-1405. DOI: 10.1007/s11263-020-01316-z. URL: <http://dx.doi.org/10.1007/s11263-020-01316-z> (cit. alle pp. 37, 38).
- [46] Agrim Gupta, Piotr Dollár e Ross Girshick. *LVIS: A Dataset for Large Vocabulary Instance Segmentation*. 2019. arXiv: 1908.03195 [cs.CV]. URL: <https://arxiv.org/abs/1908.03195> (cit. alle pp. 38, 39).
- [47] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: <https://arxiv.org/abs/1405.0312> (cit. alle pp. 38, 47, 48).
- [48] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso e Antonio Torralba. *Semantic Understanding of Scenes through the ADE20K Dataset*. 2018. arXiv: 1608.05442 [cs.CV]. URL: <https://arxiv.org/abs/1608.05442> (cit. a p. 38).
- [49] Pablo Arbelaez, Michael Maire, Charless Fowlkes e Jitendra Malik. «Contour Detection and Hierarchical Image Segmentation». In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.5 (mag. 2011), pp. 898–916. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.161. URL: <http://dx.doi.org/10.1109/TPAMI.2010.161> (cit. a p. 38).
- [50] Yanghao Li, Hanzi Mao, Ross Girshick e Kaiming He. *Exploring Plain Vision Transformer Backbones for Object Detection*. 2022. arXiv: 2203.16527 [cs.CV]. URL: <https://arxiv.org/abs/2203.16527> (cit. a p. 39).

- [51] Ashutosh Saxena, Min Sun e Andrew Y. Ng. «Make3D: Learning 3D Scene Structure from a Single Still Image.» In: *IEEE Trans. Pattern Anal. Mach. Intell.* 31.5 (2009), pp. 824–840. URL: <http://dblp.uni-trier.de/db/journals/pami/pami31.html#SaxenaSN09> (cit. a p. 40).
- [52] Derek Hoiem, Alexei A. Efros e Martial Hebert. «Automatic photo pop-up». In: *ACM Trans. Graph.* 24.3 (lug. 2005), pp. 577–584. ISSN: 0730-0301. DOI: 10.1145/1073204.1073232. URL: <https://doi.org/10.1145/1073204.1073232> (cit. a p. 40).
- [53] Kevin Karsch, Ce Liu e Sing Bing Kang. «Depth Transfer: Depth Extraction from Video Using Non-Parametric Sampling». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (nov. 2014), pp. 2144–2158. ISSN: 2160-9292. DOI: 10.1109/tpami.2014.2316835. URL: <http://dx.doi.org/10.1109/TPAMI.2014.2316835> (cit. a p. 40).
- [54] Zhengqi Li e Noah Snavely. *MegaDepth: Learning Single-View Depth Prediction from Internet Photos*. 2018. arXiv: 1804.00607 [cs.CV]. URL: <https://arxiv.org/abs/1804.00607> (cit. alle pp. 42, 44, 48).
- [55] Ozan Sener e Vladlen Koltun. *Multi-Task Learning as Multi-Objective Optimization*. 2019. arXiv: 1810.04650 [cs.LG]. URL: <https://arxiv.org/abs/1810.04650> (cit. a p. 44).
- [56] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li e Zhenbo Luo. «Monocular Relative Depth Perception with Web Stereo Data Supervision». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 311–320. DOI: 10.1109/CVPR.2018.00040 (cit. alle pp. 44, 48).
- [57] Chaoyang Wang, Simon Lucey, Federico Perazzi e Oliver Wang. *Web Stereo Video Supervision for Depth Prediction from Dynamic Scenes*. 2019. arXiv: 1904.11112 [cs.CV]. URL: <https://arxiv.org/abs/1904.11112> (cit. alle pp. 44, 48).
- [58] Youngjung Kim, Hyungjoo Jung, Dongbo Min e Kwanghoon Sohn. «Deep Monocular Depth Estimation via Integration of Global and Local Predictions». In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 4131–4144. DOI: 10.1109/TIP.2018.2836318 (cit. a p. 44).
- [59] Weifeng Chen, Zhao Fu, Dawei Yang e Jia Deng. *Single-Image Depth Perception in the Wild*. 2017. arXiv: 1604.03901 [cs.CV]. URL: <https://arxiv.org/abs/1604.03901> (cit. a p. 44).

- [60] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys e Andreas Geiger. «A Multi-view Stereo Benchmark with High-Resolution Images and Multi-camera Videos». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2538–2547. DOI: 10.1109/CVPR.2017.272 (cit. a p. 45).
- [61] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley e Michael J. Black. «A Naturalistic Open Source Movie for Optical Flow Evaluation». In: *Computer Vision – ECCV 2012*. A cura di Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato e Cordelia Schmid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 611–625. ISBN: 978-3-642-33783-3 (cit. a p. 45).
- [62] Moritz Menze e Andreas Geiger. «Object scene flow for autonomous vehicles». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3061–3070. DOI: 10.1109/CVPR.2015.7298925 (cit. a p. 45).
- [63] Nathan Silberman, Derek Hoiem, Pushmeet Kohli e Rob Fergus. «Indoor Segmentation and Support Inference from RGBD Images». In: *Computer Vision – ECCV 2012*. A cura di Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato e Cordelia Schmid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760. ISBN: 978-3-642-33715-4 (cit. a p. 45).
- [64] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard e Daniel Cremers. «A benchmark for the evaluation of RGB-D SLAM systems». In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580. DOI: 10.1109/IRoS.2012.6385773 (cit. a p. 45).
- [65] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu e William T. Freeman. *Learning the Depths of Moving People by Watching Frozen People*. 2019. arXiv: 1904.11111 [cs.CV]. URL: <https://arxiv.org/abs/1904.11111> (cit. alle pp. 45, 48).
- [66] Qian-Yi Zhou, Jaesik Park e Vladlen Koltun. *Open3D: A Modern Library for 3D Data Processing*. 2018. arXiv: 1801.09847 [cs.CV]. URL: <https://arxiv.org/abs/1801.09847> (cit. a p. 47).
- [67] Jeremy Howard e Sylvain Gugger. «Fastai: A Layered API for Deep Learning». In: *Information* 11.2 (feb. 2020), p. 108. ISSN: 2078-2489. DOI: 10.3390/info11020108. URL: <http://dx.doi.org/10.3390/info11020108> (cit. a p. 67).
- [68] *Vast.ai*. URL: <https://vast.ai/> (cit. a p. 68).
- [69] *What is PyQt?* 2023. URL: <https://www.riverbankcomputing.com/software/pyqt/intro> (cit. a p. 81).