Spaghetti Detector

Roberto Mirabella

Matricola 0001058899

Abstract

Lo sviluppo tecnologico nel campo della stampa 3D ha reso possibile la creazione di oggetti tridimensionali in modo rapido e personalizzato. Tuttavia, questa innovazione non è priva di sfide: uno dei problemi maggiormente riscontrato riguarda la formazione di strutture collassate, comunemente note come "spaghetti", durante il processo di stampa. Al fine di garantire risultati ottimali, è necessario saper affrontare adeguatamente queste tipologie di problemi.

A tal proposito, l'applicazione "Spaghetti Detector" sfrutta le reti neurali convoluzionali (CNN) per monitorare e valutare la qualità della stampa mediante la telecamera presente nei dispositivi Android. L'applicazione integra un classificatore basato su MobileNetV3, addestrato con un dataset personalizzato, per distinguere efficacemente una stampa corretta da una avente "spaghetti".

Infine, è stato implementato un modulo di detection basato su YOLOv8 small per individuare e localizzare eventuali difetti.

Sommario

Classificatore con MobileNetV3

Nell'ambito della stampa 3D, uno dei principali ostacoli individuati nello sviluppo di applicazioni di rilevamento e classificazione di difetti è la scarsità di dataset specifici. La varietà di difetti che possono verificarsi durante il processo di stampa, unita alla complessità delle immagini catturate, rende cruciale la disponibilità di un set di dati rappresentativo e diversificato.

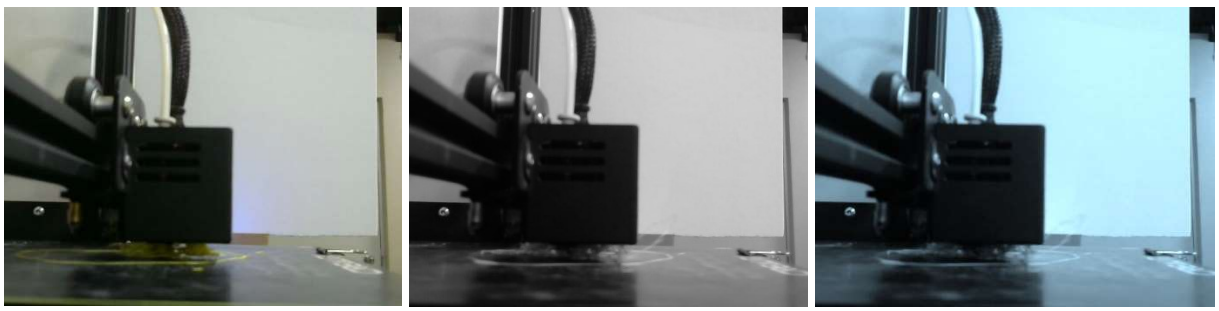
In questo contesto, è emersa la sfida di sviluppare un classificatore robusto in grado di rilevare degli "spaghetti" nelle stampe 3D. Vista la limitata quantità di dati etichettati relativi a difetti specifici delle stampanti, si è optato per il *transfer learning*, una strategia che sfrutta le caratteristiche generali acquisite da una rete preaddestrata su un dataset più ampio per poi adattarlo a un compito specifico, in questo caso, il riconoscimento dei difetti di stampe 3D.

Il modello scelto per questa applicazione è MobileNetV3, noto per la sua efficienza computazionale e la capacità di gestire applicazioni su dispositivi mobili [1].

Dataset

Il dataset impiegato per questa analisi è il "3D-Printer Defected Dataset [2]". Tale dataset si è rivelato essenziale nell'addestramento di un classificatore capace di rilevare la presenza di difetti durante il processo di stampa 3D. Tra i vari tipi di errori considerati, il classificatore si concentra su aspetti critici come la mancata adesione al piatto di stampa, importante per il successo della stampa. Questi difetti includono anche la rottura di supporti o la formazione di "spaghetti", contribuendo così a migliorare la capacità predittiva del modello e a prevenire problematiche significative durante la stampa 3D. Questo dataset, inoltre, contiene 759 foto raffiguranti stampe non andate a buon fine inserite nella cartella Defected e 798 foto di stampe con nessun difetto al momento dell'acquisizione inserite nella cartella No_Defected. Tutte le immagini sono state acquisite mediante un'inquadratura fissa della fotocamera e utilizzando sempre filamenti della stessa colorazione.

Data la scarsa quantità di esempi e la poca varietà di colori e soggetti, si è deciso di realizzare anche delle versioni Augmented del dataset, così da diversificare ulteriormente la varietà dei dati di training. A tal proposito, sono state effettuate due versioni: una ottenuta dalla conversione delle immagini in scala di grigi e l'altra tramite realizzazione con un filtro seppia.



(A) (B) (C)

Figura 1 (A) Dataset Originale, (B) Scala di grigi, (C) Filtro seppia.

Inoltre, l'intero dataset è stato suddiviso in *train*, *validation* e *test*, rispettivamente contenenti 80%, 10%,10% dei dati, tale da poter testare la rete ottenuta con elementi mai visualizzati in precedenza.

Primo Tentativo con MobileNetV2

Creato il dataset e partizionato nelle sezioni necessarie per il training, inizialmente si era pensato di utilizzare MobileNetV2 come modello di base su cui basare il classificatore. Addestrato questo primo modello poi si è passati all'analisi delle performance provando a quantizzare la rete.

Tuttavia, analizzando i modelli quantizzati nei tre modi possibili messi a disposizione da Pytorch (dynamic, static, aware training) andando a fondere tutti i layer conv2d, batchnorm2d e relu le performance delle reti risultavano essere le medesime.
Infatti, i tempi di inferenza si attestavano intorno ai 25ms su cpu sia per le reti quantizzate sia per il modello non quantizzato.

Transfer Learning

Per implementare il transfer learning su MobileNetV3 per la classificazione degli "spaghetti", è stata adottata un'architettura preaddestrata su ImageNet. L'idea chiave è quella di impiegare le caratteristiche generali apprese da MobileNetV3 durante il preaddestramento su un dataset più esteso e diversificato. Tale approccio è finalizzato ad istruire i layer della rete dedicati alla classificazione effettiva.

Per evitare la perdita delle informazioni apprese durante il preaddestramento, quindi, è stato disabilitato il calcolo dei gradienti per i parametri della parte convoluzionale del modello, congelando di fatto i loro pesi.

Inoltre, è stato personalizzato il classificatore per poterlo adattare allo specifico task di classificazione delle anomalie nelle stampe.

Il nuovo classificatore è stato definito con i seguenti componenti:

- Strato Lineare Iniziale: un primo strato lineare con 512 neuroni, che ha in input l'output generato dalla parte convoluzionale;

- Funzione di Attivazione ReLU: per introdurre non linearità nella rete;

- Strato di Dropout: per mitigare il rischio di overfitting durante il processo di addestramento, è stato inserito uno strato di dropout con probabilità del 50%;

- Strato Lineare Finale: ultimo strato lineare con due neuroni, rappresentanti le due classi di interesse, cioè la stampa corretta e la presenza di difetti.

Training ed Evaluation
Definiti la rete, il dataset e i vari iperparametri di addestramento da utilizzare, si è passati all'addestramento della rete, più nello specifico del classificatore, parte finale della rete neurale. L'addestramento, durato 20 epoche, ha prodotto ottimi risultati di accuratezza sullo specifico dataset (Figura 2).

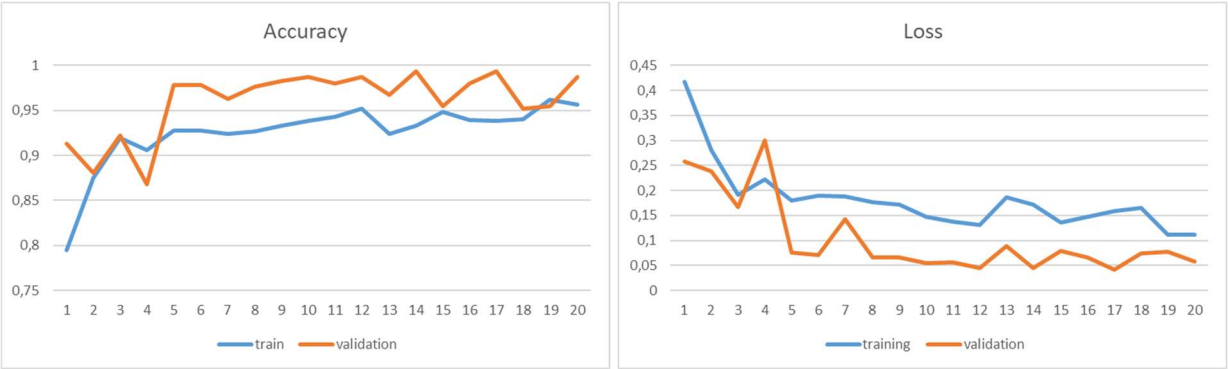


Figura 2 Risultati di accuratezza e loss durante il training.

Una volta addestrata la rete con successo, è stato possibile esportare il modello addestrato ottimizzandolo per i dispositivi mobili, pronto per la piena integrazione nell'ecosistema Android.

Infine, il tempo di inferenza misurato per il modello è parti a 15ms su cpu, miglioramento significativo rispetto alle prime prove con MobileNetV2

Anomaly Detection YOLOv8

La rilevazione di anomalie durante il processo di stampa 3D rappresenta un aspetto cruciale per garantire la qualità del risultato finale. Per raggiungere questo obiettivo, non solo è stato adoperato il classificatore basato su MobileNetV3, ma anche il modello YOLOv8 Small, riconosciuto per la sua efficacia e versatilità nella rilevazione di oggetti [3].

YOLO (*You Only Look Once*) è una famiglia di modelli di rilevazione di oggetti che si caratterizzano per la loro capacità di eseguire l'identificazione in tempo reale con elevate prestazioni. YOLOv8, in particolare, si distingue per la sua efficienza sia in termini computazionali che in termini di precisione rispetto alle versioni precedenti. La versione "Small" è stata scelta per garantire una rapida esecuzione su dispositivi mobili, senza sacrificare l'accuratezza nella rilevazione di piccoli dettagli come gli "spaghetti" nella stampa 3D.

La versione 8 Ultralytics ha introdotto anche la possibilità di utilizzare lo stesso modello non solo per la rilevazione ma anche per segmentazione e classificazione. Tuttavia, le performance di classificazione restano comunque inferiori rispetto ad un modello costruito ad hoc come quello introdotto nel capitolo precedente.

Dataset Personalizzato e Framework
Per addestrare YOLOv8 Small sullo specifico task di rilevazione di anomalie, è stato utilizzato un dataset custom, ovvero il 3D printing failure Computer Vision Project [4]. Questo dataset racchiude immagini di stampe 3D con la presenza di spaghetti/stringing/zits e con i relativi dati delle zone in cui sono presenti tali anomalie.

La ricerca di un dataset abbastanza vario è la chiave per avere un modello affidabile, infatti, in un primo momento si era tentato di utilizzare altri due dataset con la stessa struttura di [4] ma con esempi di qualità molto inferiore. Questo produceva un modello del tutto inaffidabile.

Ultralytics offre un framework trasparente per l'addestramento di reti neurali, semplificando notevolmente il processo di organizzazione e utilizzo dei dati.

I dati devono essere strutturati nel seguente modo:

- train
    - imgs
    - labels
- valid
    - imgs
    - labels
- test (opzionale)
    - imgs
    - labels

Le cartelle imgs racchiuderanno le immagini utilizzate per l'addestramento/testing del modello. Invece, le cartelle labels conterranno un file testuale per ogni immagine nella cartella imgs le cui righe presentano:

- La classe dell'esempio catturato nella corrispettiva foto;
- Bounding Box (regione dello spazio che contiene effettivamente l'esempio).

Inoltre, per velocizzare il tempo di inferenza del modello, si è deciso di dare in input alla rete un'immagine di dimensioni 224x224 in modo da alleggerire ulteriormente il carico computazionale.

La rete addestrata (40 epoche) ha prodotto i seguenti risultati:

Class	Images	Instances	P	R	mAP50	mAP50-95
all	500	3657	0,554	0,504	0,511	0,304
spaghetti	500	672	0,74	0,874	0,886	0,649
stringing	500	255	0,328	0,361	0,243	0,122
zits	500	2730	0,594	0,277	0,405	0,142

Dove le metriche P, R, mAP50 e mAP50-95 rappresentano:

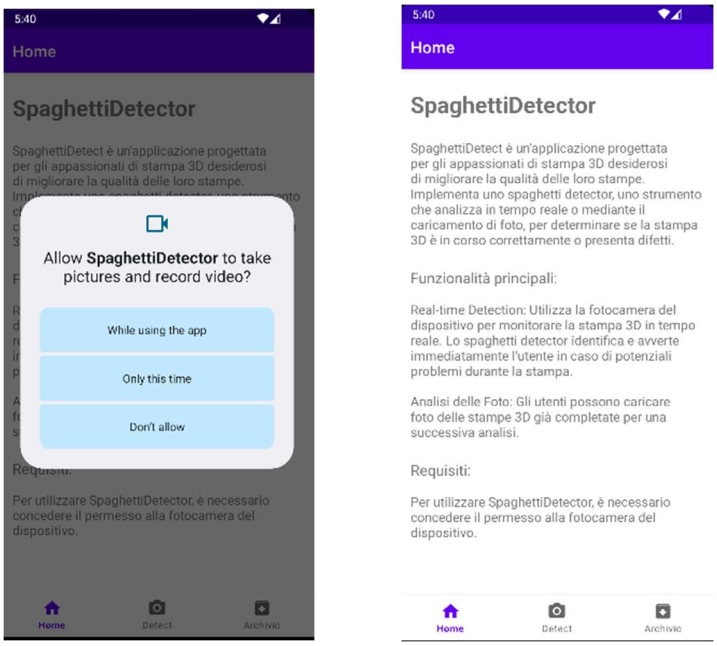
- **P (precisione)**: L'accuratezza degli oggetti rilevati, che indica quanti rilevamenti sono stati corretti.

- **R (Recall)**: La capacità del modello di identificare tutte le istanze di oggetti nelle immagini.
- **mAP50**: precisione media calcolata con una soglia di intersection over union (IoU) di 0,50. È una misura dell'accuratezza del modello considerando solo i rilevamenti "facili".
- **mAP50-95**: la media della precisione media calcolata con soglie di IoU diverse, che vanno da 0,50 a 0,95. Fornisce una visione completa delle prestazioni del modello a diversi livelli di difficoltà di rilevamento.

Dai risultati ottenuti, si evince che il modello sviluppato è in grado di identificare e localizzare le anomalie con discreta precisione. Le altre due classi prese in considerazione, cioè stringing e zits, arricchiscono la varietà dei difetti localizzabili pur non essendo tanto importanti e significativi quanto la classe “spaghetti”.

Applicativo Android

L'applicazione si compone di una MainActivity che gestisce tre fragment distinti con l'ausilio di una navbar. Al primo avvio dell'applicazione viene richiesto il permesso alla fotocamera come mostrato in Figura 3. Inoltre, la MainActivity inizializza tutti i componenti necessari per l'utilizzo dei torch script ottenuti alla fine del training e dell'ottimizzazione del classificatore MobileNetV3 e del rilevatore YOLOv8.



(A) (B)

Figura 3 (A) Primo avvio con richiesta permessi fotocamera. (B) Fragment Home.

Il fragment Home contiene solo informazioni testuali di carattere generale riguardanti l'applicazione (Figura 3B).

Dalla navbar posizionata in basso, è possibile cambiare fragment cliccando sulle varie icone.

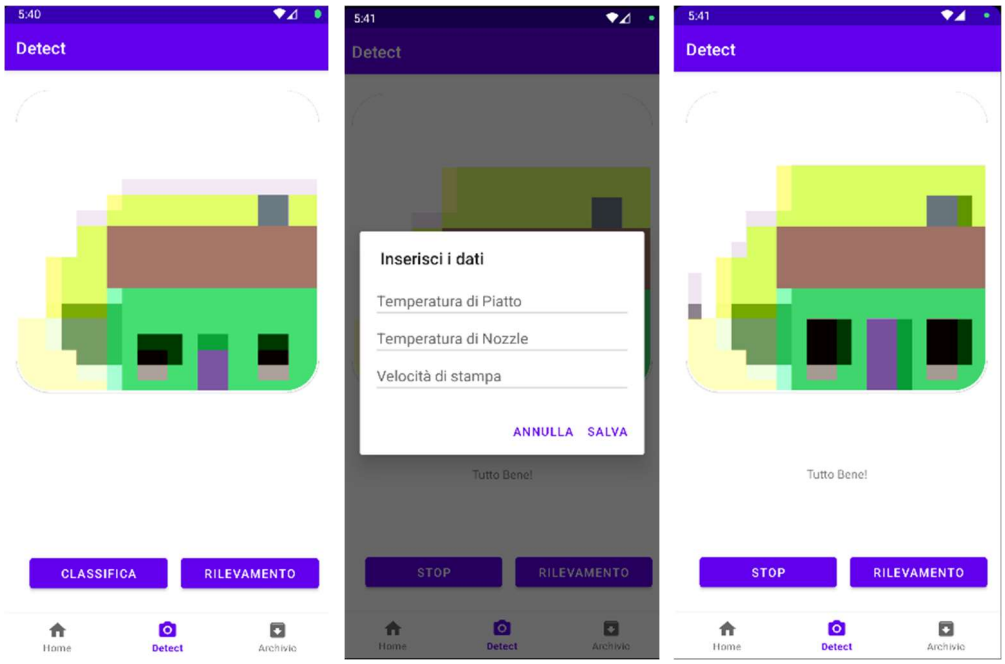
Fragment Detect

Il cuore dell'applicazione risiede nel fragment Detect. Al primo avvio, viene impiegata la fotocamera per inizializzare una Preview di quanto captato dalla fotocamera posteriore dello smartphone.

Nel fragment sono presenti i seguenti pulsanti (Figura 4A): uno adibito all'avvio del classificatore basato su MobilenetV3 e l'altro, invece, destinato all'avvio del Detector YOLO.

Cliccando su "Classifica", appare un Overlay (Figura 4B) in cui viene richiesto l'inserimento dei vari parametri utilizzati per quella specifica stampa 3D. Cliccando su "salva", i dati inseriti nei vari

campi vengono salvati in un file che verrà in seguito utilizzato come archivio di tutte le stampe effettuate.



(A) (B) (C)

Figura 4 (A) Prima visita del fragment Detect. (B) Click su CLASSIFICA. (C) Classificazione in corso.

Una volta chiuso l'Overlay, nel fragment appare un campo testuale in cui viene indicato se l'immagine captata risulta essere priva o meno di anomalie.

Il controllo, e quindi l'inferenza al classificatore, viene effettuata ogni due secondi. Questo comportamento è stato scelto poiché una stampa 3D dura in media diverse ore. Infatti, ogni stampa 3D viene suddivisa in centinaia o migliaia di layer (sezione orizzontale di altezza pari a 0.2 mm dell'oggetto in stampa). Va considerato che per ogni layer la stampante impiega alcuni minuti e un insuccesso si verifica solo quando diversi layer in successione falliscono. Se effettuare un controllo ad ogni frame si rivela inefficiente dal punto di vista energetico e computazionale, al contrario, una verifica ogni manciata di secondi risulta essere non solo più efficiente, ma anche egualmente precisa.

Cliccando invece sul pulsante "Rilevamento", viene resa visibile una ImageView in cui viene mostrata la medesima immagine catturata dalla fotocamera ma eventualmente con un rettangolo rosso (Figura 5) che va ad indicare la presenza di una delle anomalie imparate durante la fase di training dal modello YOLOv8s.

Analogamente a quando fatto per il classificatore, anche il rilevatore andrà ad effettuare l'inferenza al rilevatore ogni due secondi, sempre per non appesantire inutilmente il carico computazionale che rappresenta l'inferenza ad un modello abbastanza complesso.

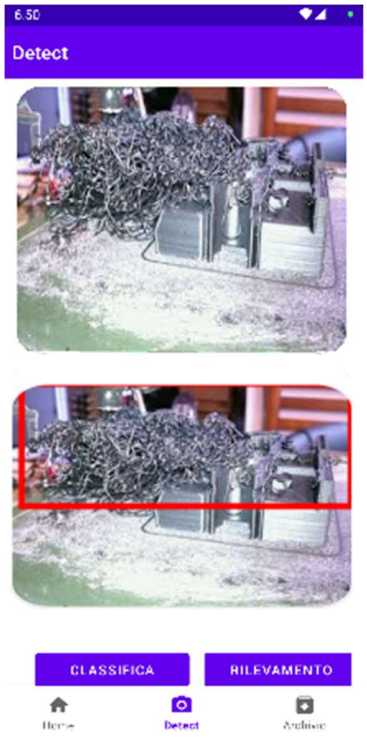


Figura 5 Rilevamento Spaghetti.

Figura 6 Fragment Archivio.

Fragment Archivio

Il fragment archivio contiene uno storico di tutte le impostazioni di stampa che si è deciso di salvare alla comparsa dell'Overlay descritto nella sezione precedente.

Per il momento impostazioni slavate sono quelle più significative, ovvero:

- Timestamp: data in cui è stata effettuata la rilevazione
- Temperature Plate: Temperatura del piatto di stampa
- Temperature Nozzle: Temperatura dell'estrusore
- Print Velocity: velocità della testa di stampa
- Anomly: flag booleano che indica se una stampa si è conclusa con successo o meno

Tutti questi dati vengono scritti e letti su un file. json che viene di volta in volta arricchito con nuovi elementi.

Inoltre, è possibile andare ad eliminare un elemento dall'archivio andando a cliccare l'apposito bottone "Delete". Così facendo verrà non solo rimosso dalla schermata attuale ma anche eliminata l'entry dal file.

Conclusioni

L’applicazione sviluppata permette di monitorare una stampa 3D in maniera efficace ed efficiente. Sia i modelli che l’applicazione però presentano margini di miglioramento; infatti, il principale vincolo presentatosi nell’allenamento di entrambi i modelli adottati è quello della scarsa quantità (e a volte anche scarsa qualità) dei dati. Avere più esempi e soprattutto più esempi diversificati renderebbe i modelli notevolmente più precisi riuscendo a generalizzare meglio nel mondo reale.

Inoltre, l’applicazione potrebbe tener traccia di moltissimi altri parametri di stampa, come l’accelerazione della testa di stampa, velocità del filamento estruso, numero di layer di adesione, presenza del brim, ecc. Sarebbe anche possibile far definire dall’utente con che frequenza si vuole effettuare il controllo andando così a modificare il parametro Delay introdotto nel fragment Detect.

Riferimenti

[1] Searching for MobileNetV3 (https://arxiv.org/pdf/1905.02244.pdf)

[2] 3D-Printer Defected Dataset (https://www.kaggle.com/datasets/justin900429/3d-printer-defected-dataset?select=no_defected)

[3] YOLOv8 (https://docs.ultralytics.com/it/models/yolov8)

[4] 3D printing failure Computer Vision Project (https://universe.roboflow.com/3d-printing-failure/3d-printing-failure)