

Visual Information
Processing and Management
2024/2025

Studio e Classificazione Dataset FoodX-251

Pallini Vincenzo - 907303
Pezzotti Mattia - 885965





Analisi Empirica

- Il Dataset **FoodX-251** contiene originariamente **158 846** immagini di cibo proveniente da diverse culture, classificate in 251 classi e divise come segue:
 - 118k immagini per Training Set
 - 12k immagini per Validation Set
 - 28k immagini per Test Set
- Il nostro **Training set** (unlabeled + small) contiene **118 475** immagini
 - Dopo una prima visita empirica, si nota come molte non siano di cibo
- Il nostro **Validation set** contiene **11 994** immagini
 - Dopo una prima visita empirica, sembra che tutte le immagini siano di cibo
- Unendo i due sets otteniamo **130 469** immagini

Small and Unlabeled
Training Set



Prima Analisi

- Unlabeled Training Set
 - Numero immagini: 113 455
 - Numero Classi: 1
- Small Training Set
 - Numero immagini: 5 020
 - Numero Classi: 251



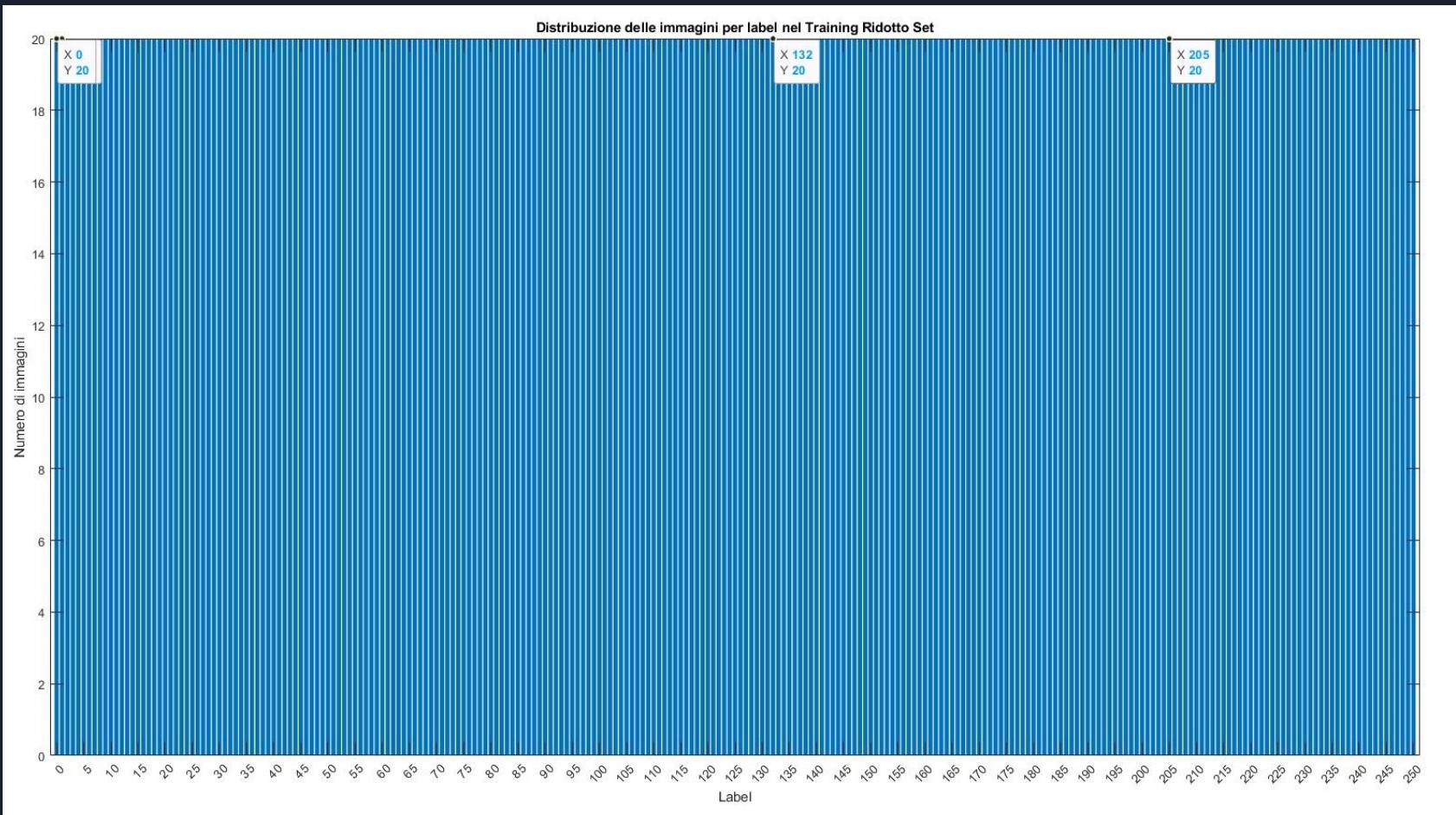
Nome: train_120186.jpg

label: -1



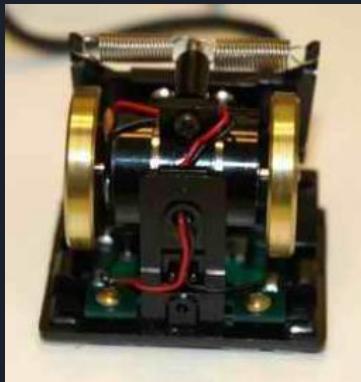
Nome: train_059921.jpg

label: 0 (Macarons)

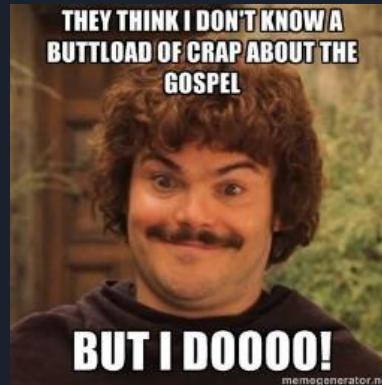


Immagini di non cibo

Approfondendo in modo empirico il Training Set Unlabeled, notiamo come molte immagini non siano effettivamente di cibo



train_005406.jpg



train_005890.jpg



train_102739.jpg

Immagini ambigue

Altre immagini invece, non sono di cibo ma sono collegate al cibo



train_038097.jpg



train_001963.jpg



train_005220.jpg



Pulizia Training Set

Per rimuovere le immagini abbiamo valutato diverse strade:

- Rimozione manuale
 - Scartata a priori in quanto molto laboriosa, e non replicabile facilmente o scalabile.
- Identificazione e rimozione Outlier
 - Scartata in quanto i parametri (media rgb, varianza rgb) non erano descrittori sensibili per distinguere immagini di cibo e non cibo.
- Clustering con K-Means [1]
 - Non richiede etichette, è dipendente dalla qualità delle feature (SVM o Rete Pre-Addestrata) e non completamente automatica.



Clustering

1. L'estrazione delle feature è avvenuta tramite una Pre-Trained Network: **Resnet101**
 - o 1000 Feature per immagine
 - o Più descrittiva di algoritmi come SURF, dato che Resnet ha imparato da un dataset di milioni di immagini.
 - o Più veloce a runtime
2. Effettuata PCA [\[2\]](#) sulle feature per motivi di memoria e prestazioni
 - o Calcolo z-score per ogni vettore
 - o Mantenute 205 features per vettore (90%)
3. Clustering con K-Means
 - o 1000 iterazioni massime
 - o 251 cluster
4. Visualizzazione manuale dei vari cluster
 - o Annotazione dei cluster “non cibo” o “ambigui”
 - o “Rimozione” dei file dal training set

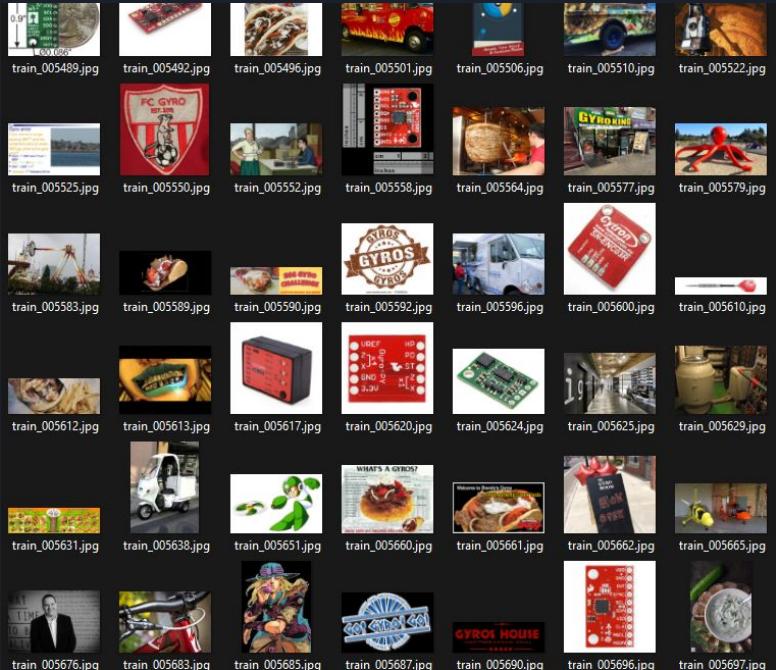
Risultati Finali Clustering

- Training Set Pulito: 111 067 immagini
- Not-Food Set: 7408 immagini

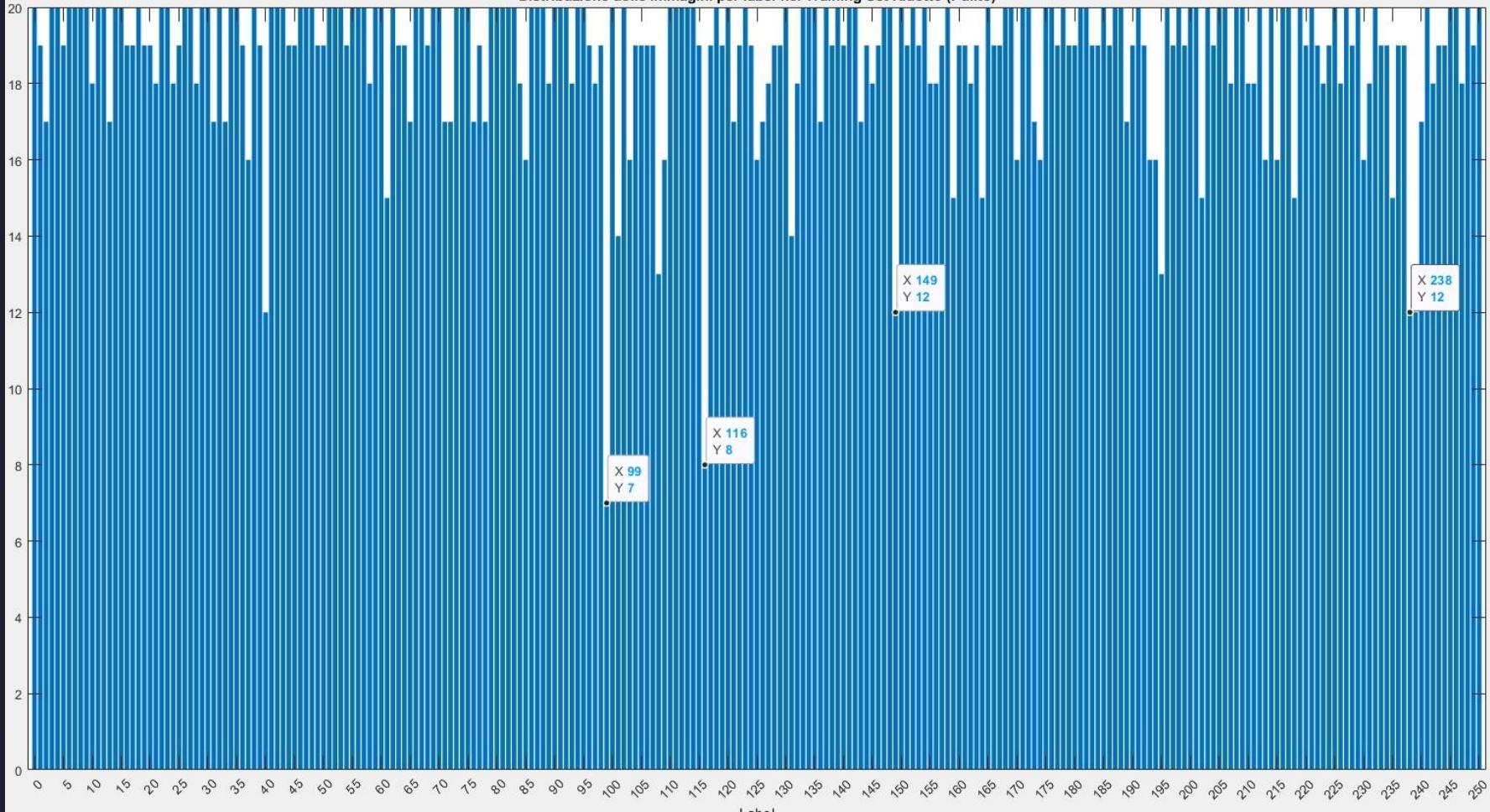
Sicuramente il “**Not-Food Set**” contiene anche immagini di cibo, e sicuramente non sono state rimosse tutte le immagini di non cibo dal training set.

Nonostante questo, ci riteniamo soddisfatti, dato che, a occhio, la maggior parte delle immagini sporche siano state rimosse.

Controlliamo ora lo **Small Training Set**



Distribuzione delle immagini per label nel Training Set Ridotto (Pulito)



Small Training Set

Notiamo che sono state rimosse dallo **Small 320 immagini** dopo aver effettuato la pulizia.

Per ovviare a questo problema abbiamo selezionato a mano sufficienti immagini da aggiungere allo Small Training Set in modo da raggiungere le originali 20 immagini per classe (5020 in totale).

Per farlo abbiamo ricreato il set, suddividendo in sottocartelle da “riempire”.

📁 0	28/12/2024 04:20 PM	Cartella di file
📁 1	28/12/2024 04:21 PM	Cartella di file
📁 2	28/12/2024 04:22 PM	Cartella di file
📁 3	28/12/2024 04:20 PM	Cartella di file
📁 4	28/12/2024 04:20 PM	Cartella di file
📁 5	28/12/2024 04:24 PM	Cartella di file
📁 6	28/12/2024 04:20 PM	Cartella di file
📁 7	28/12/2024 04:20 PM	Cartella di file
📁 8	28/12/2024 04:20 PM	Cartella di file
📁 9	28/12/2024 04:20 PM	Cartella di file
📁 10	28/12/2024 04:25 PM	Cartella di file
📁 11	28/12/2024 04:20 PM	Cartella di file
📁 12	28/12/2024 04:20 PM	Cartella di file
📁 13	28/12/2024 04:26 PM	Cartella di file
📁 14	28/12/2024 04:20 PM	Cartella di file
📁 15	28/12/2024 04:20 PM	Cartella di file
📁 16	28/12/2024 04:27 PM	Cartella di file
📁 17	28/12/2024 04:28 PM	Cartella di file
📁 18	28/12/2024 04:20 PM	Cartella di file
📁 19	28/12/2024 04:30 PM	Cartella di file
📁 20	28/12/2024 04:31 PM	Cartella di file
📁 21	28/12/2024 04:31 PM	Cartella di file
📁 22	28/12/2024 04:20 PM	Cartella di file
📁 23	28/12/2024 04:20 PM	Cartella di file
📁 24	28/12/2024 04:32 PM	Cartella di file
📁 25	28/12/2024 04:32 PM	Cartella di file
📁 26	28/12/2024 04:32 PM	Cartella di file
📁 27	28/12/2024 04:20 PM	Cartella di file
📁 28	28/12/2024 04:33 PM	Cartella di file
📁 29	28/12/2024 04:20 PM	Cartella di file

Poi ?

Durante la creazione del set ci siamo scontrati con la classe 116 (“**Poi**”). Dovrebbe essere una purea Polinesiana, ma in realtà ci sono delle palline (?)

Effettuando Reverse Search anche queste palline si chiamano Poi.



Fine-grained Food classification





Primi Studi

Approcci esplorati:

- **Unsupervised Learning:** Clustering con feature estratte
- **Semi-Supervised learning**
- **Supervised Learning:** Training rinforzato tramite Pseudo-Labelling
 - Fine-Tuning CNN
 - Estrazione feature con CNN

Clustering

Un primo approccio naïve è stato quello di effettuare nuovamente il clustering con le feature estratte. Questo perché, data la mancanza di una ground truth ampia, fosse il metodo più adatto per creare nuove label.

Questo approccio aveva, ancora prima di partire, due problemi principali:

- Il labelling andava fatto a mano, cosa non scontata per 251 classi
- Non esiste una metrica istantanea per capire se il modello ha diviso in modo corretto

Una volta effettuato il clustering poi è sorto il problema principale: molti cluster non convergevano su nessuna classe in particolare, classi con colori / geometrie simili venivano raggruppate anche se appartenenti a classi diverse





Semi-Supervised learning

Uno degli approcci proposti dagli articoli Mathworks per aumentare il numero di osservazione è il [Label Data Using Semi-Supervised Learning Techniques](#), che si basa su alcuni aspetti dell'apprendimento supervisionato.

Alcune osservazioni di addestramento sono etichettate, ma la maggioranza non è etichettata. I metodi di apprendimento semi-supervisionato cercano di sfruttare la struttura sottostante dei dati per attribuire etichette ai dati non etichettati.

Questo ci è sembrato il metodo più adatto, se non per due ostacoli non indifferenti

- Potrebbe non essere oggetto del nostro corso
- Memoria GPU (almeno per quanto riguarda le funzioni Matlab)

Questo ci ha portato a non sviluppare questo approccio.



Estrazione Features

Leggendo [reti neurali profonde \[3\]](#), un articolo sul sito di Mathworks, troviamo questo suggerimento:

 Suggerimento

Il perfezionamento di una rete neurale spesso offre la massima precisione. Per gli insiemi di dati molto piccoli (meno di circa 20 immagini per classe), è preferibile provare l'estrazione di feature.

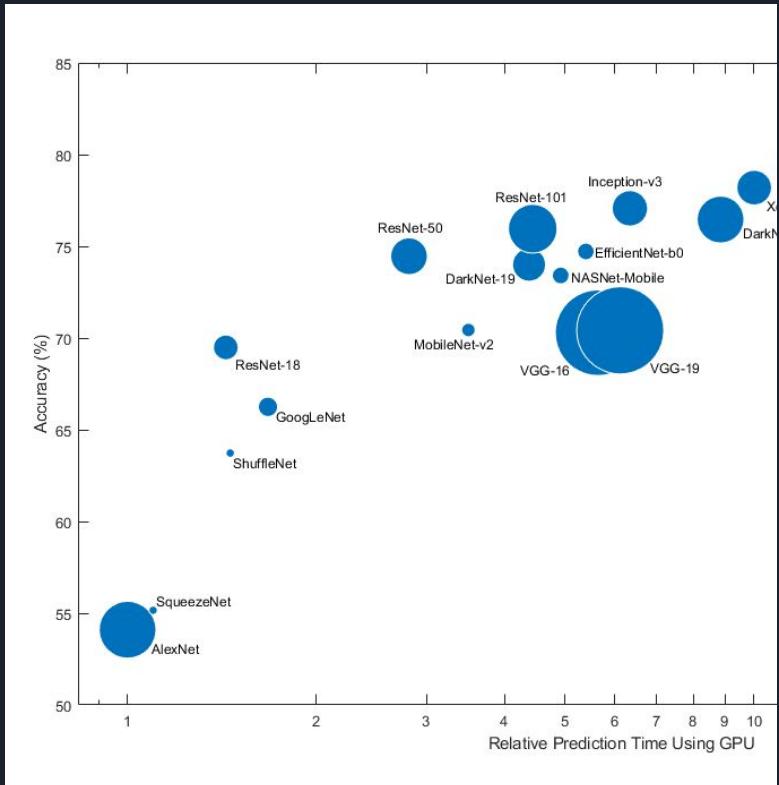
Decidiamo quindi di provare a estrarre le feature dalle immagini, seguendo le guidelines di Mathworks riportate nell'articolo: [Estrazione delle feature dell'immagine utilizzando la rete pre addestrata](#).

Confronto CNN

Dopo aver confrontato le reti neurali tramite [questo articolo Mathworks](#) abbiamo deciso di confrontare principalmente tre reti neurali:

- Resnet50
- Resnet101
 - Non esplorati completamente per problemi di Hardware GPU
- Resnet18

Questo perché dovrebbero essere le migliori in termini di **accuracy/gpuTime**.





Svolgimento e Risultati

1. Feature Extraction a un layer intermedio e uno finale tramite Pre Trained Network
 - o Resnet18, al layer intermedio **res3b_relu** e finale **pool5**
 - o Resnet101, al layer intermedio **res4b9_relu** e finale **pool5**
2. Creazione di un nuovo modello SVM [\[4\]](#) tramite funzione fitcecoc
3. Calcolo accuracy su Validation/Test Set
 - o Resnet18
 - res3b_relu: **7.5%**, training-time: 254 sec, Test-time: 360 sec
 - pool5: **25.2%**, training-time: 278 sec, Test-time: 1012 sec
 - o Resnet101
 - res4b9_relu: **15.4%**, training-time: 21 sec, Test-time: 638 sec
 - pool5: **26.0%**, training-time: 494 sec, Test-time: 913.99 sec

Notiamo che i risultati non sono male, ma prima di procedere, confrontiamo con una CNN fine-tuned



Fine-Tuning CNN

Le reti neurali sono state modificate per effettuare un fine-tuning:

- Modifica dell'ultimo layer Fully Connected per avere un output a 251 classi
- Introduzione di un Dropout Layer al 50% per ridurre overfitting tra il layer di pooling e il layer Fully Connected
- Freeze della backbone (sia per performance sia per fine-tuning)
 - 35 Layer per Resnet 18
 - 141 Layer per Resnet50
 - 301 Layer per Resnet101



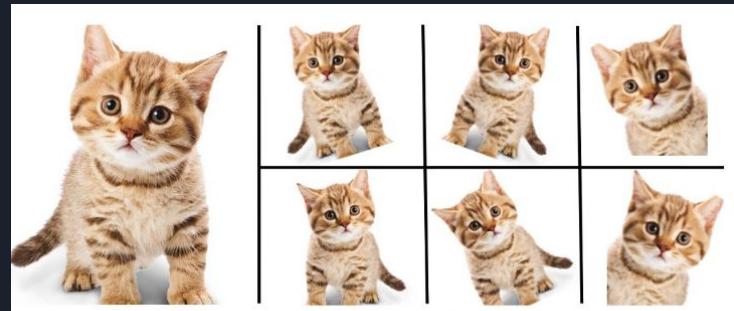
Opzioni Primo Addestramento

- Ottimizzazione: Adam
- Validation Data = augimdsVal
 - Ridurre ulteriormente il Training set avrebbe portato a un peggior addestramento
- Validation Frequency = 10
- Validation Patience = 5
 - Per evitare overfitting
- Mini Batch Size = 128
 - Ridotto per Resnet101 a 64 per problemi di Hardware
- Max Epochs = 50
- Initial Learn Rate = 5e-5
 - Come da Paper originale

Augmented Training Set

Dato il ridotto numero di immagini dello Small training Set, si è pensato di aumentare artificiosamente tramite **data augmentation**, in particolare

- Rotazione casuale
- Riflesso orizzontale
- Traslazione orizzontale
- Traslazione verticale
- Zoom casuale



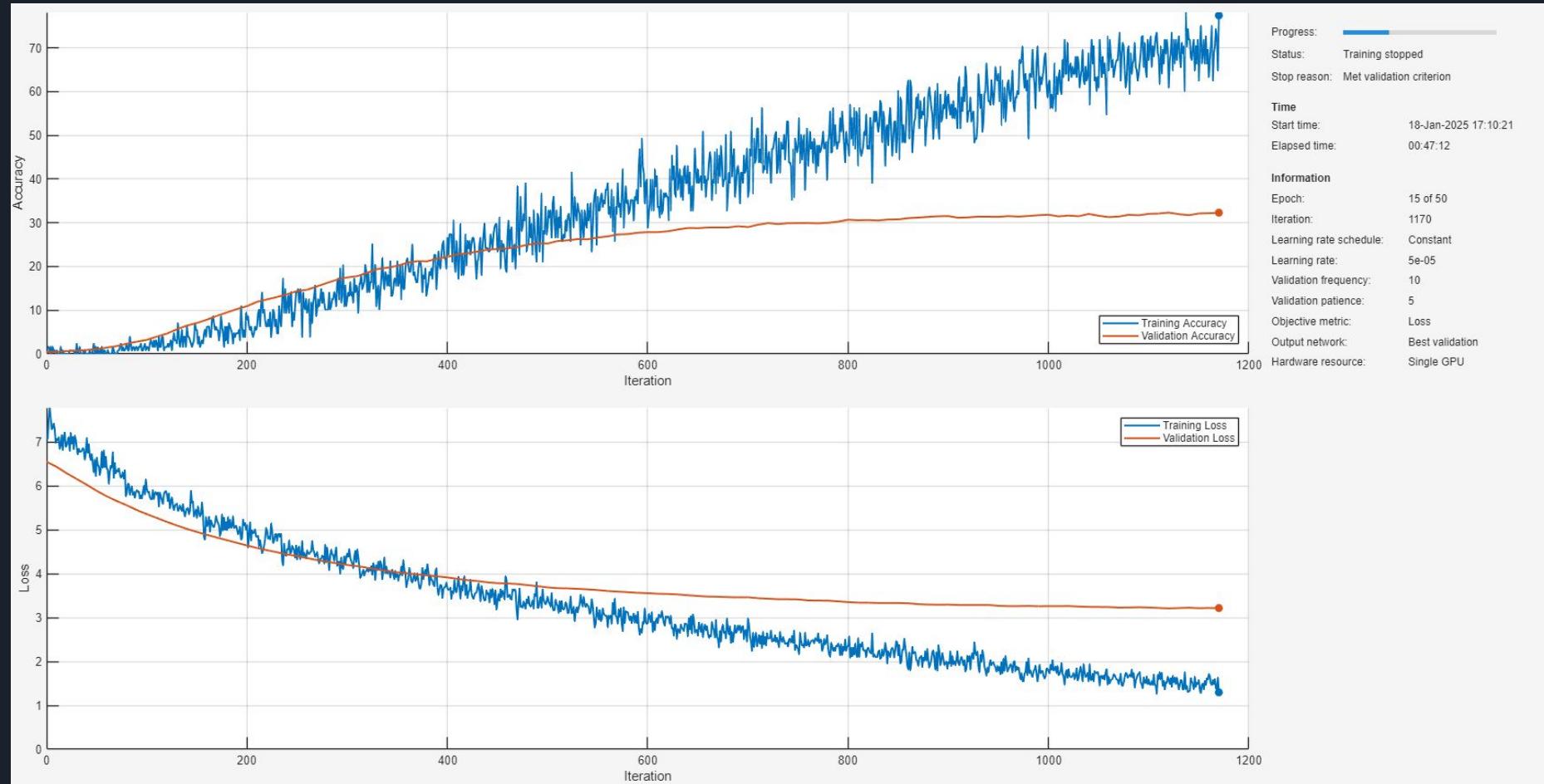
Si sono poi effettuate delle prove con diverso numero di immagini Augmentate.



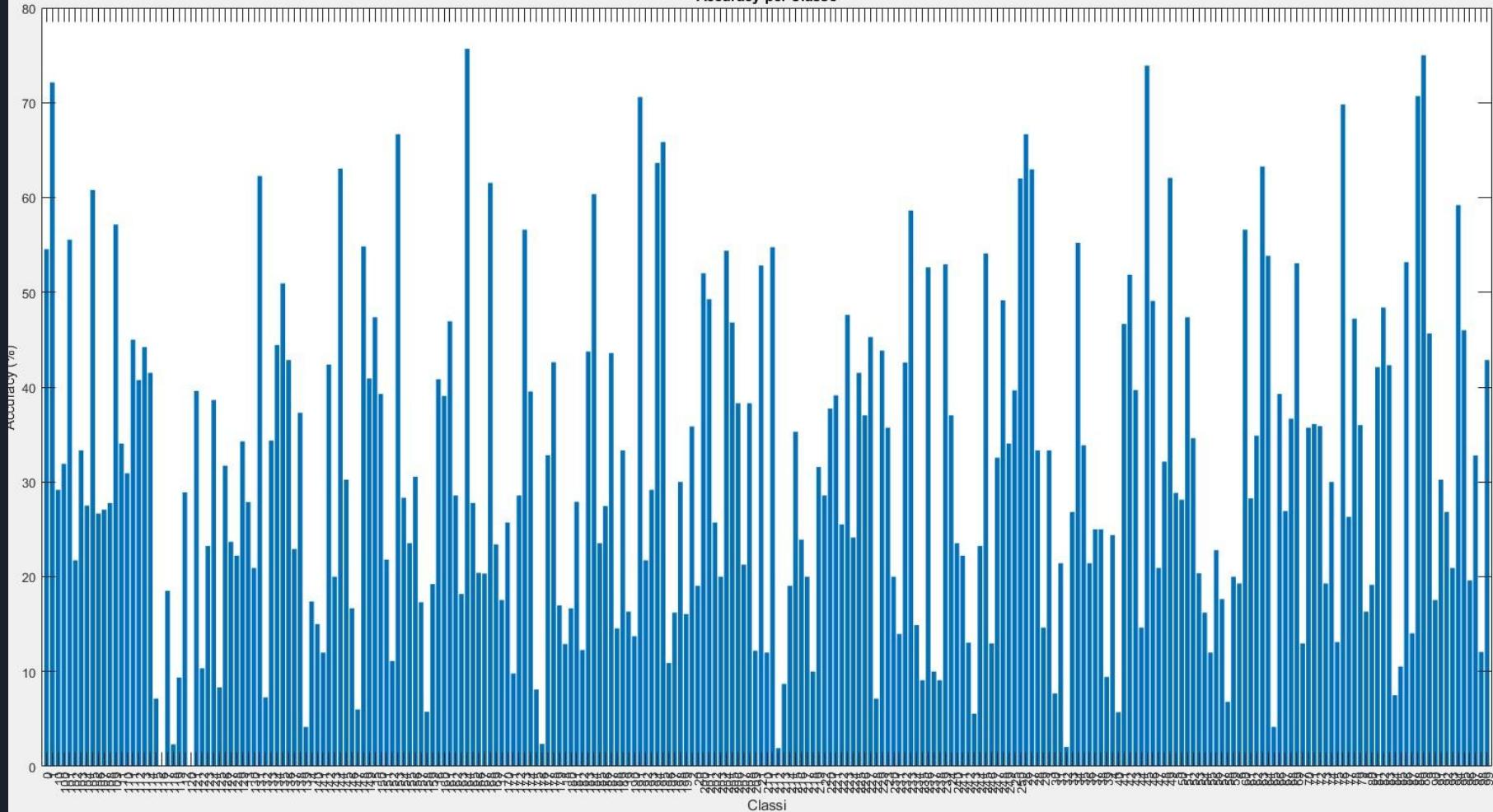
Risultati Primo Round - Resnet18

Tutti e quattro i modelli hanno ottenuto simili risultati:

- No Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: no
 - Accuracy Validation Set: 31.4074%
 - Training-Time: 58 min
- 20 Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: 10040, 40 per classe
 - Accuracy Validation Set: 32.3162%
 - Training Time: 47 min
- 80 Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: 25100, 100 per classe
 - Accuracy Validation Set: 32.1244%
 - Training Time: 54 min
- 280 Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: 75300, 300 per classe
 - Accuracy Validation Set: 31.8076%
 - Training Time: 51 min



Resnet18
Accuracy per Classe





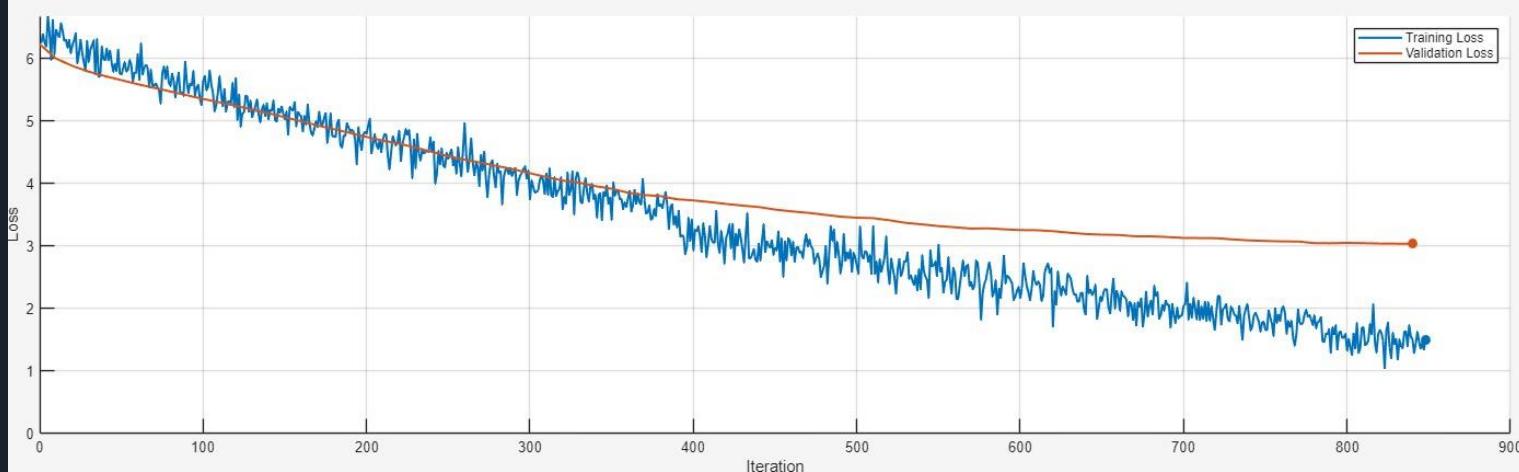
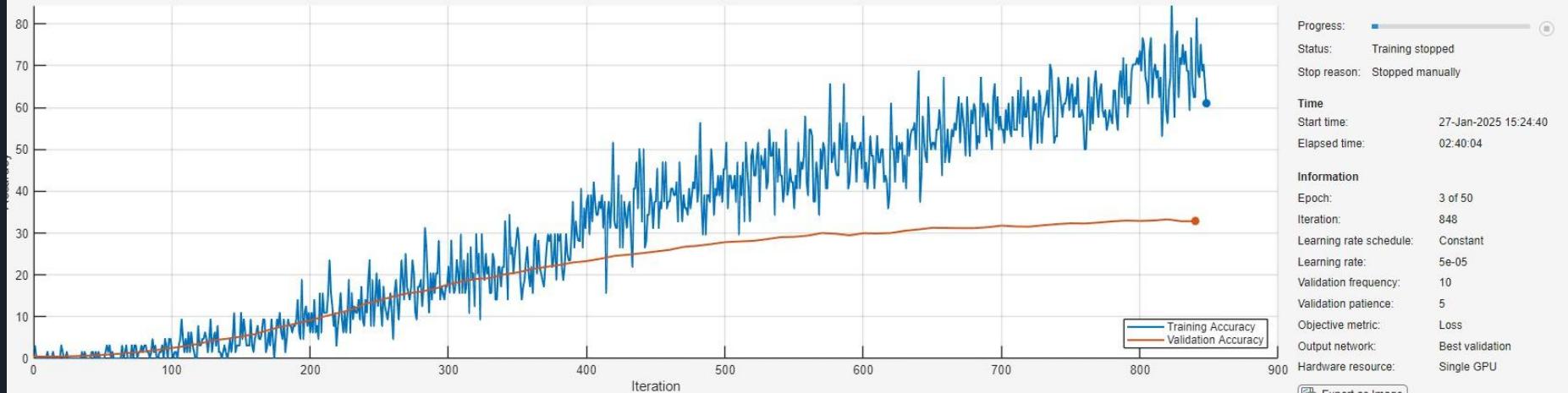
Resnet50 / Resnet101

Resnet 50:

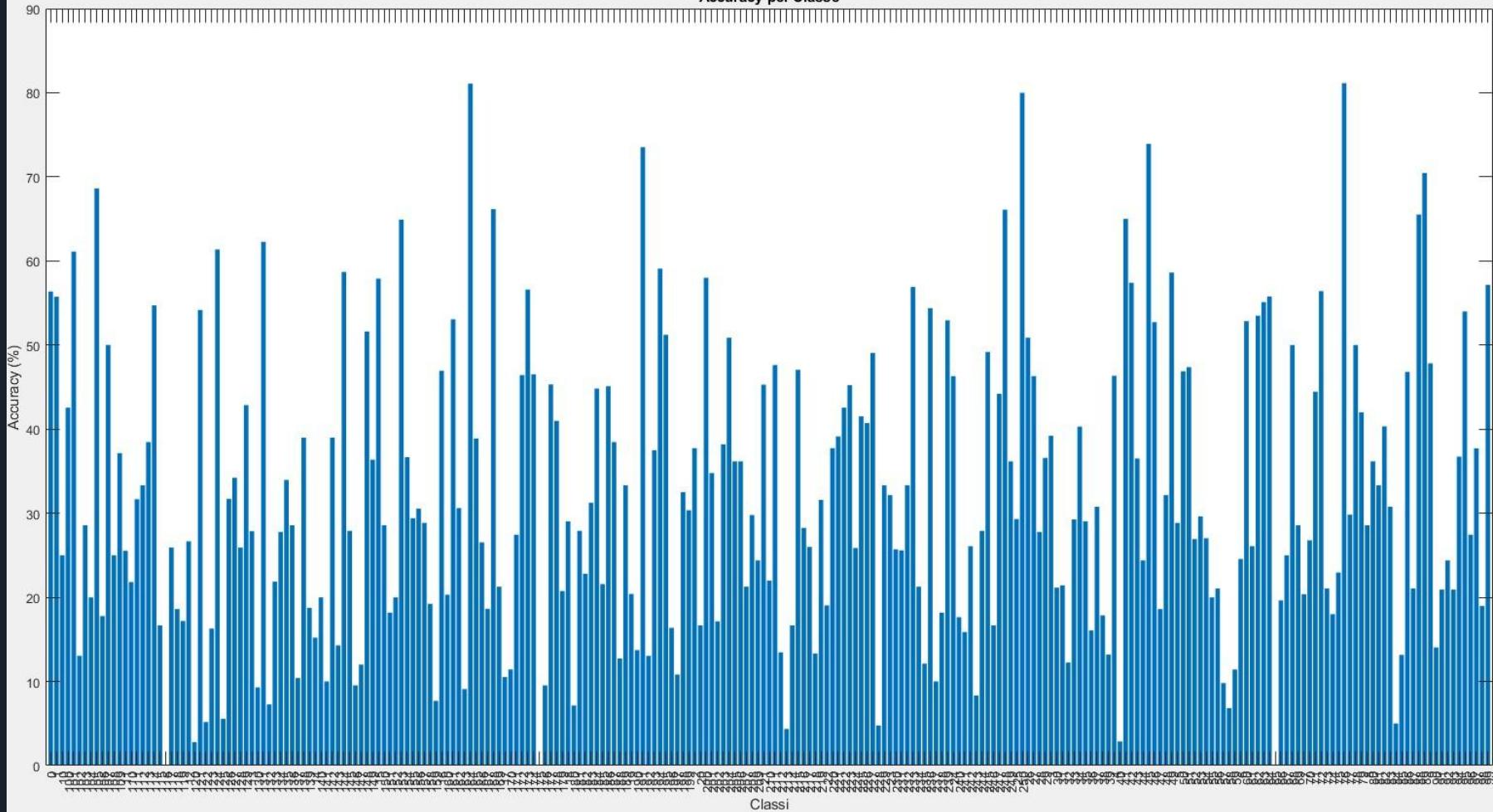
- No Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: no
 - Accuracy Validation Set: 31,0091%
 - Training-Time: 1h 50m (Stopped)
- 80 Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: 25100, 100 per classe
 - Accuracy Validation Set: 32.7414%
 - Training Time: 2h 37m (Stopped)

Resnet 101:

- No Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: no
 - Accuracy Validation Set: 29.4897%
 - Training-Time: 1h 32m (Stopped)
- 80 Augmented:
 - Train Set: 5020, 20 per classe
 - Augmented Set: 25100, 100 per classe
 - Accuracy Validation Set: 32.8498%
 - Training Time: 2h 40m (Stopped)



Resnet50
Accuracy per Classe





Scelta del Metodo

Entrambe le metodologie (Feature Extraction + SVN e CNN Fine Tuned) sono valide visti i risultati di un primo training. Tra i due abbiamo scelto di effettuare Fine Tuning in quanto:

- Risulta più “personalizzabile”, quindi flessibile e adattabile alle nostre esigenze
- Risulta più leggibile il training del modello (trainNet fornisce un grafico, fitecoc no)

Ma soprattutto

- Il tempo di creazione di nuove label per Feature Extraction sull’unlabeled set sia con Resnet18 che con Resnet101 era **estremamente lungo** (3h+ e non aveva ancora terminato)



Ampliamento Small Training Set

I risultati dell'addestramento non sono pessimi, soprattutto per un dataset come FoodX-251 che risulta molto granulare e con classi anche molto simili tra di loro.

Per migliorare l'accuracy però è necessario ampliare il Training Set attingendo dall' Unlabeled Training Set.

Per fare questo ci sono due metodi

- Aggiunta manualmente, estremamente laborioso e non scalabile anche se il più preciso
- **Pseudo-Labeling** automatico, scalabile ma soggetto ad errori



Pseudo-Labeling

Nella sezione “Make Predictions with New Data” di [questo articolo Mathworks](#), viene mostrato come classificare nuove immagini utilizzando una rete pre-addestrata.

Tramite le funzioni **predict** e **scores2label** possiamo assegnare uno score di confidenza alla previsione, in questo modo da tenere solo le previsioni con maggiore confidenza per ampliare lo small training set.

Una volta ampliato, ri-addestriamo la CNN sul nuovo training set e creiamo delle nuove pseudo label fino a quando non abbiamo una dimensione sufficiente del set

Abbiamo scelto di utilizzare Resnet 18 (e Resnet50 solo in parte) in quanto il più leggero, dato che altre reti avrebbero preso troppo tempo e risorse.



Rischi del Pseudo-Labeling

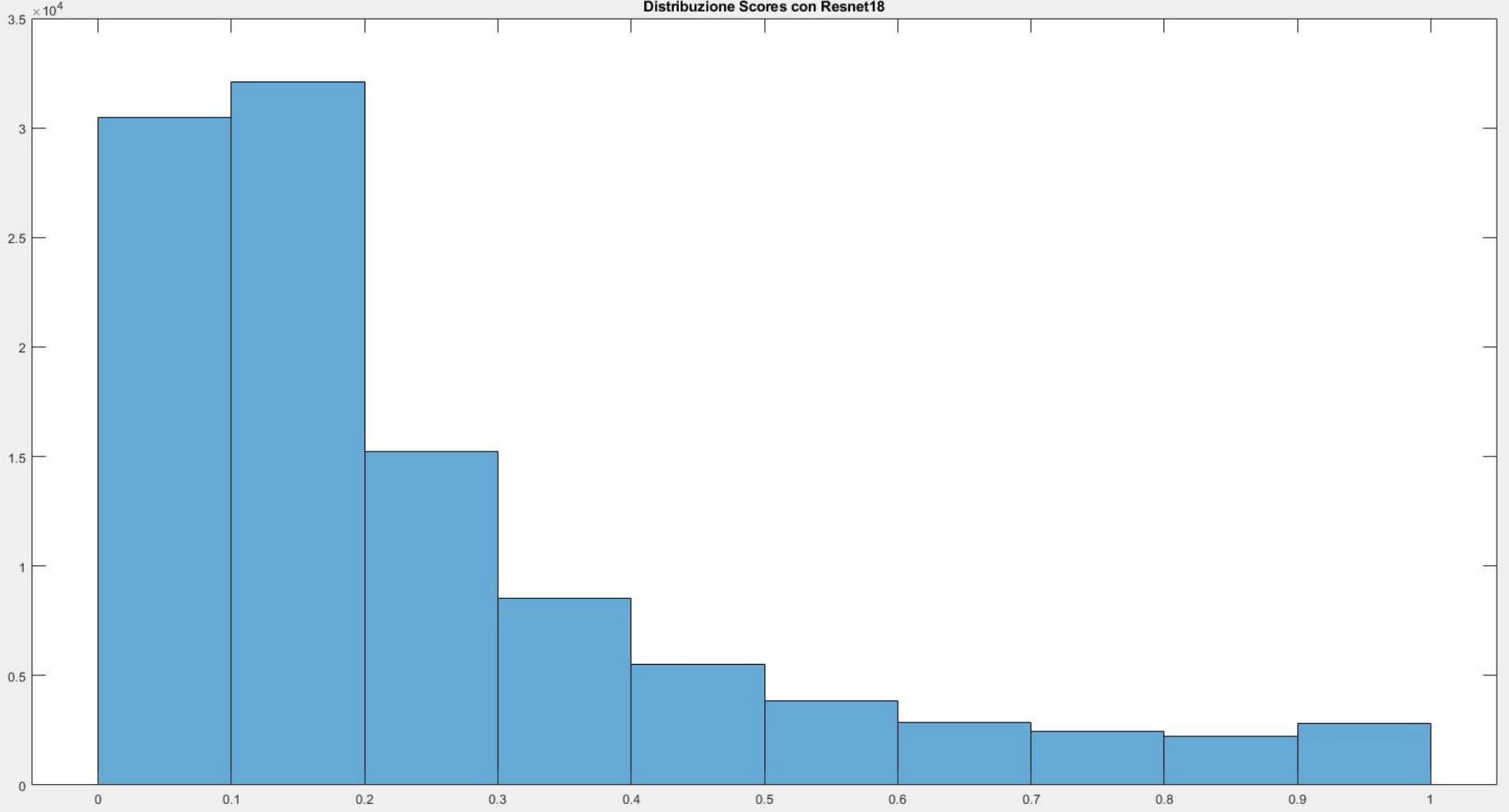
Questo workflow ha dei rischi da tenere in considerazione:

- Una confidence alta non significa necessariamente che la predizione sia corretta. Questo può portare degli errori che si propagano nei vari round di addestramento
- Le nuove label non saranno sicuramente bilanciate sulle 251 classi, c'è bisogno di ridimensionare ogni volta il nuovo Training Set, tramite **oversampling** e **undersampling**.

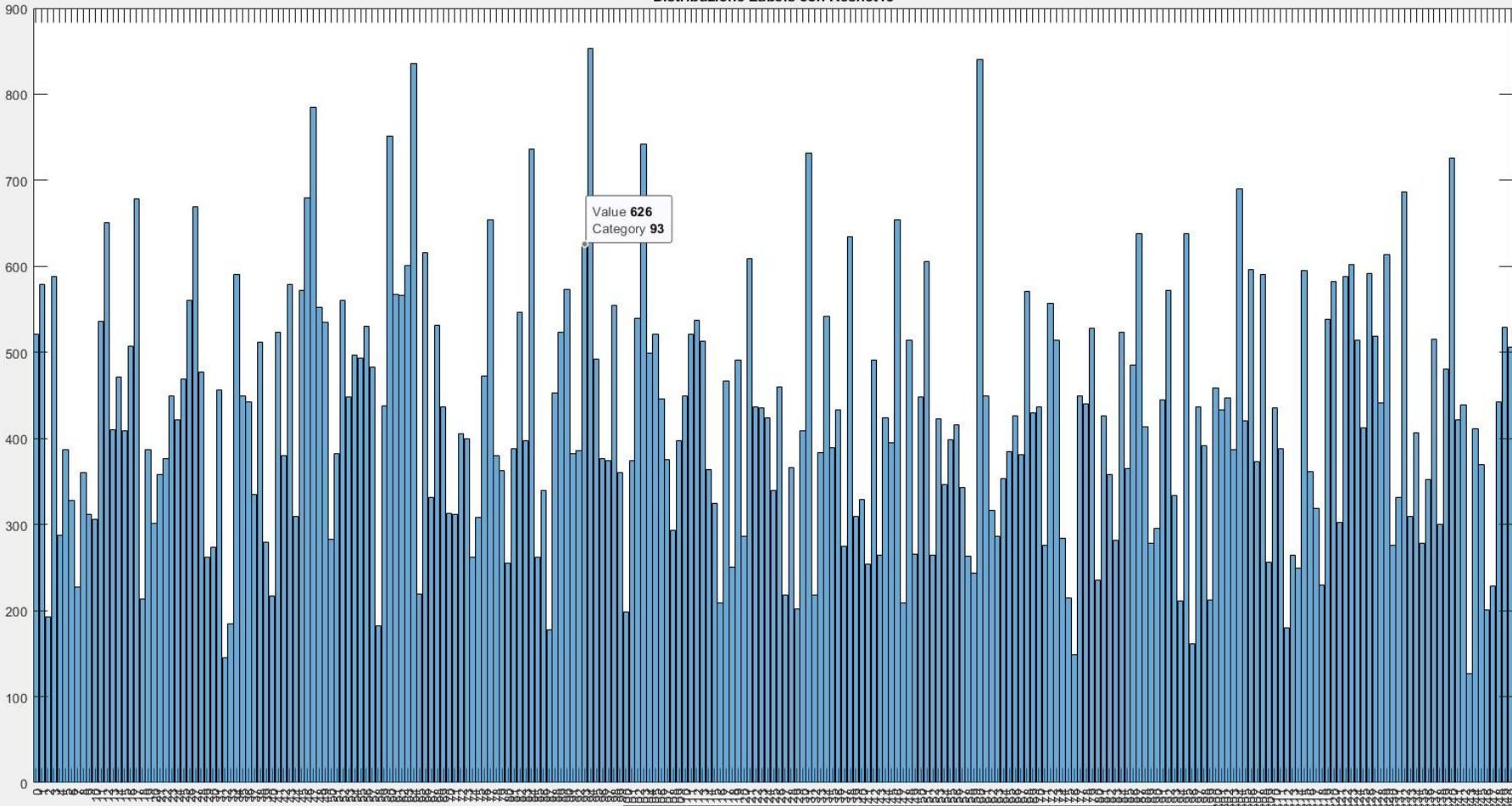
Quello che è stato fatto, dopo aver generato le nuove labels è stato:

- Verificare empiricamente se sono corrette
- Studiare la distribuzione delle classi
- Undersampling delle classi maggioritarie a una soglia scelta
- Oversampling tramite data Augmentation delle classi minoritarie a una soglia scelta
- Rimuovere le nuove label dall' Unlabeled Dataset

Distribuzione Scores con Resnet18



Distribuzione Labels con Resnet18





Risultati Secondo Round - Resnet18

Abbiamo quindi creato le nuove Label che sono state aggiunte al Training Set e abbiamo riaddestrato il modello con i seguenti risultati, riducendo le immagini massime per classe a 100

- Confidence $\geq 90\%$:
 - Train Set: 7135
 - Augmented Set: 25100, 100 per classe
 - Accuracy Validation Set: 33.5168%
- Confidence $\geq 50\%$:
 - Train Set: 14146
 - Augmented Set: 25100, 100 per classe
 - Accuracy Validation Set: 32.2244%
- Confidence $\geq 75\%$:
 - Train Set: 9595
 - Augmented Set: 25100, 100 per classe
 - Accuracy Validation Set: 32.6598%



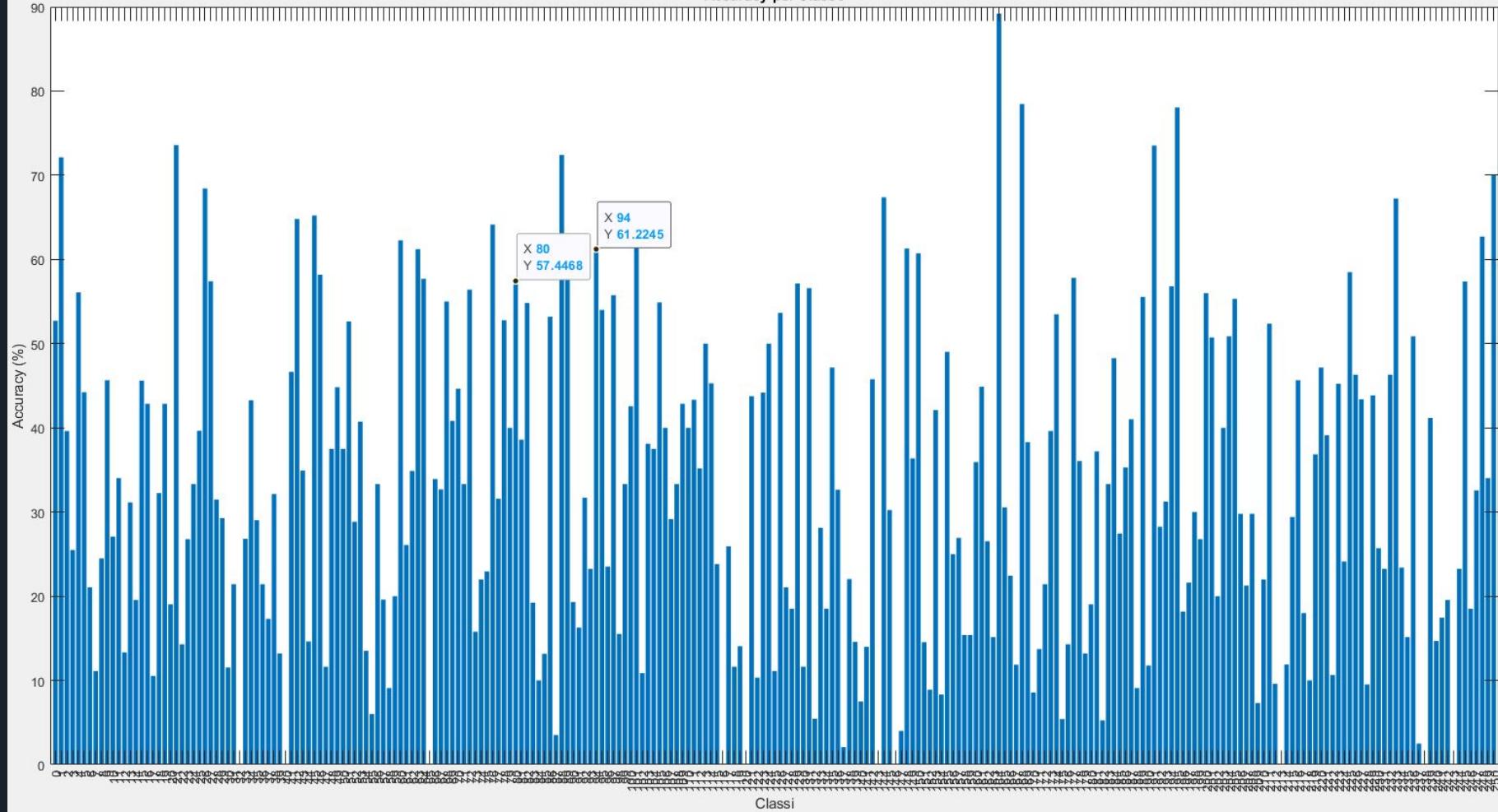
Risultati Terzo Round - Resnet18

- Tenendo immagini con confidence $\geq 90\%$ e riducendo a 180 max per classe:
 - Train Set: 11027
 - Augmented set: 45180, 180 per classe
 - Unlabeled Set: 97241
 - Accuracy Validation Set: 33.5418%

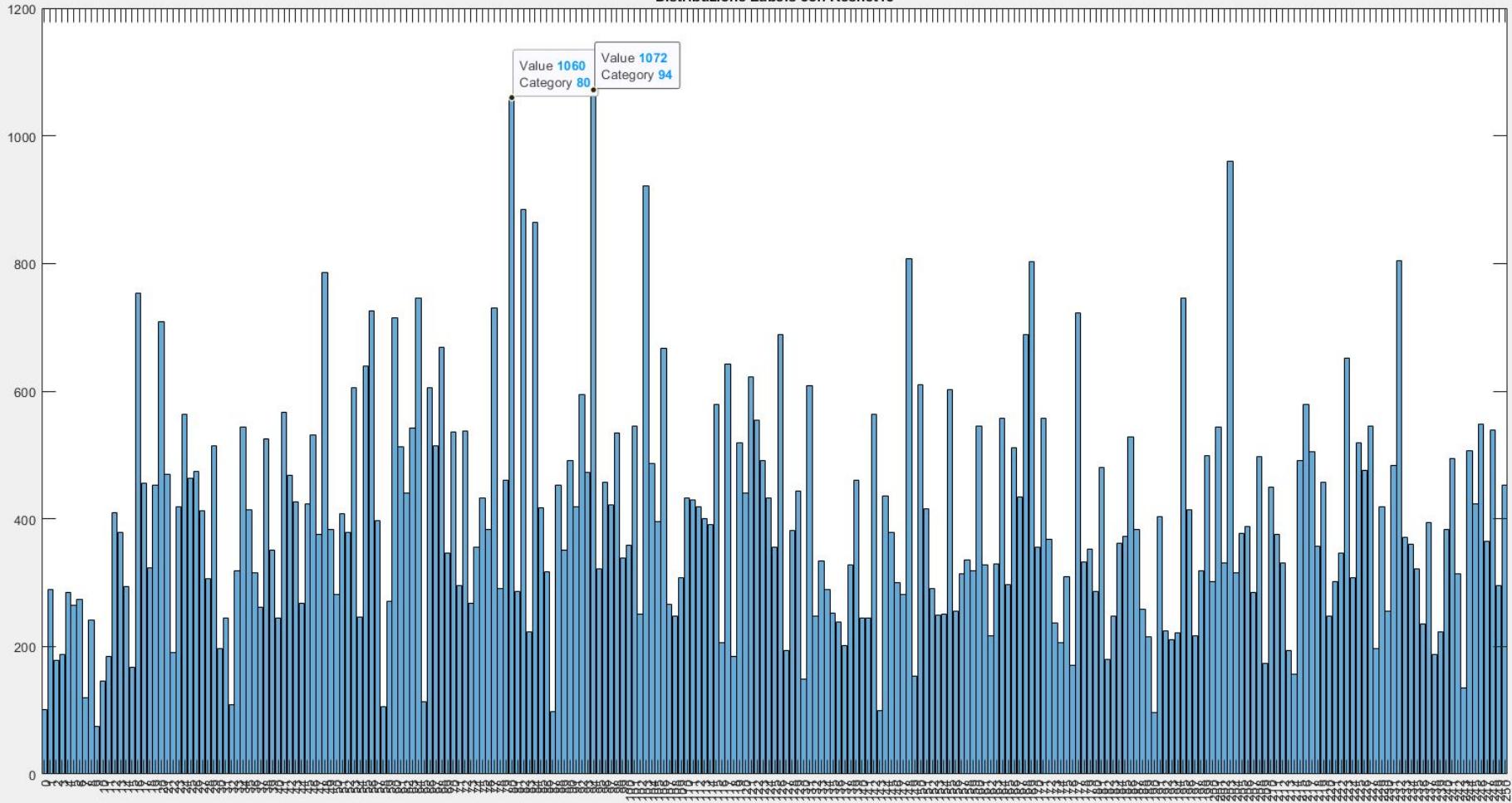
Notiamo come la Validation Accuracy in realtà non cresca molto tra un addestramento e l'altro, questo potrebbe essere dato da:

- Errore nella creazione delle nuove label
- Difficoltà generale nel generalizzare sul dataset
- Modello utilizzato troppo “piccolo”

Accuracy per Classe



Distribuzione Labels con Resnet18





Considerazioni Resnet18

- Notiamo come:
 - Esiste uno sbilanciamento forte verso determinate classi, come per la classe 94 (caesar_salad) labellata quasi 2000 volte (non sappiamo se il Training Set originale fosse bilanciato in realtà, ma possiamo dedurre, dalla accuracy dei “solo” 64% sul validation set, che molte siano errate)
 - Il Modello è molto confidente su poche classi, e poco confidente sulla maggiorparte (molte hanno 0% di accuracy), sintomo probabile del fatto che durante il pseudo-labeling venissero generate label solo per determinate classi.
- Effettuando un ultimo addestramento con tutte le label trovate (facendo oversampling delle classi minoritarie) otteniamo una Validation Accuracy del 29.8733%



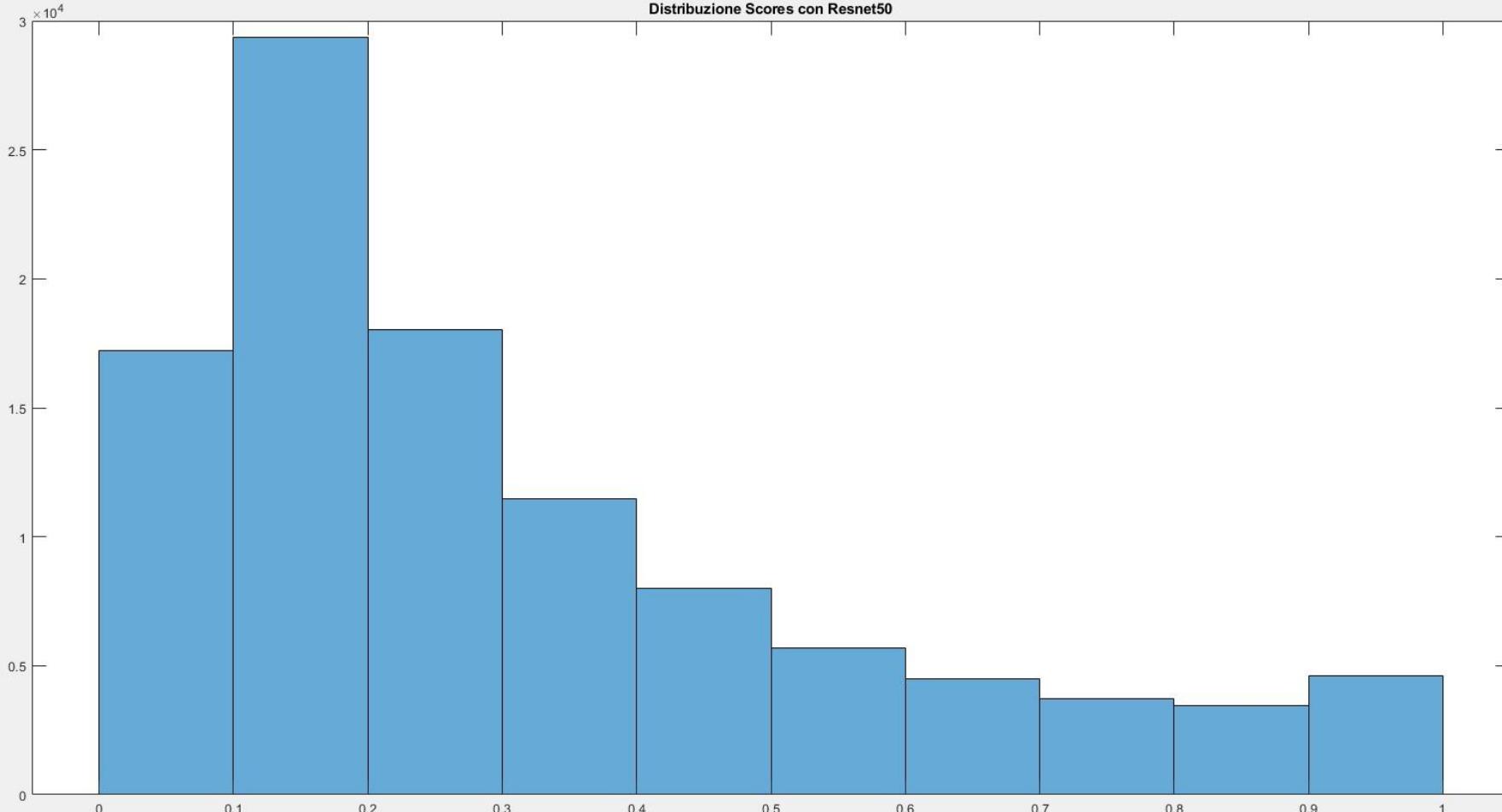
Risultati Secondo Round - Resnet50

Abbiamo effettuato l'addestramento ripartendo dall'inizio, stavolta senza restrizioni sul numero di label per classe. È stato quindi effettuato un oversampling basato sulla classe più numerosa.

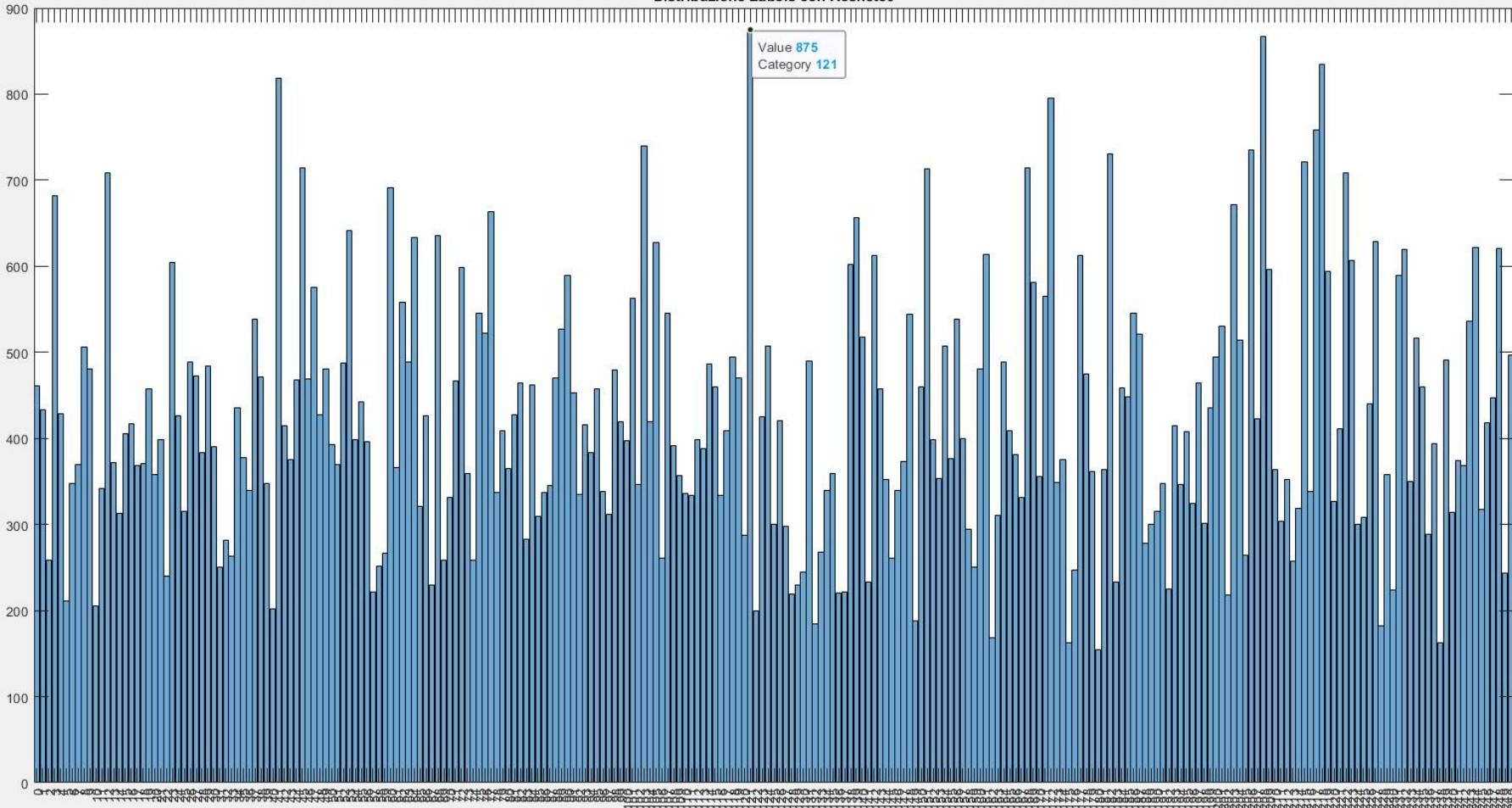
- Confidence $\geq 80\%$:
 - Train Set: 13095
 - Augmented Set: 60240, 240 per classe
 - Accuracy Validation Set: 33.5168%
 - Training Time: 1h 43m (Stopped)

Per motivi computazionali (e per il fatto che apparentemente non ci siano grandi miglioramenti rispetto a Resnet18) non abbiamo effettuato altri cicli di addestramento.

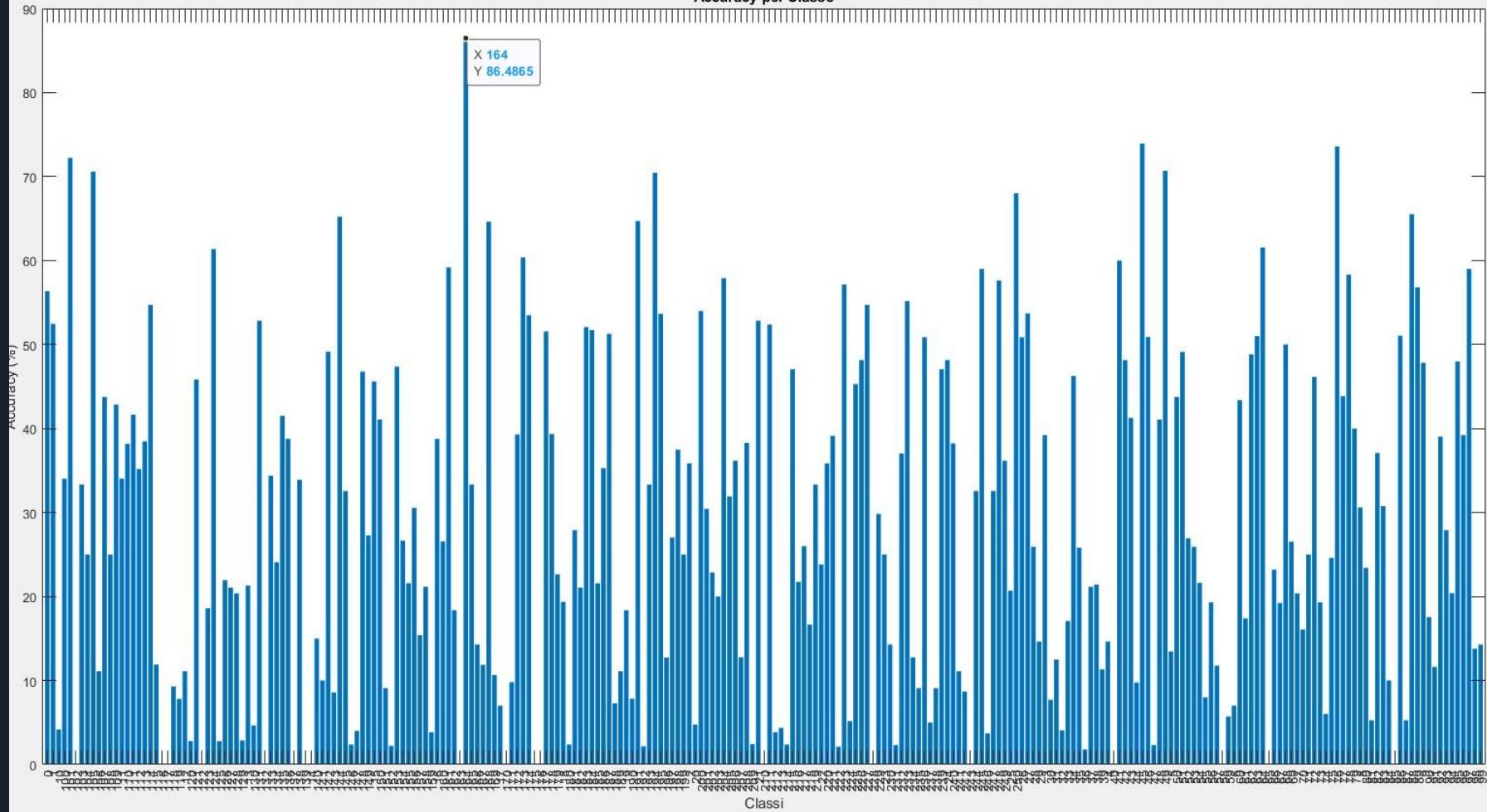
Distribuzione Scores con Resnet50



Distribuzione Labels con Resnet50



Accuracy per Classe



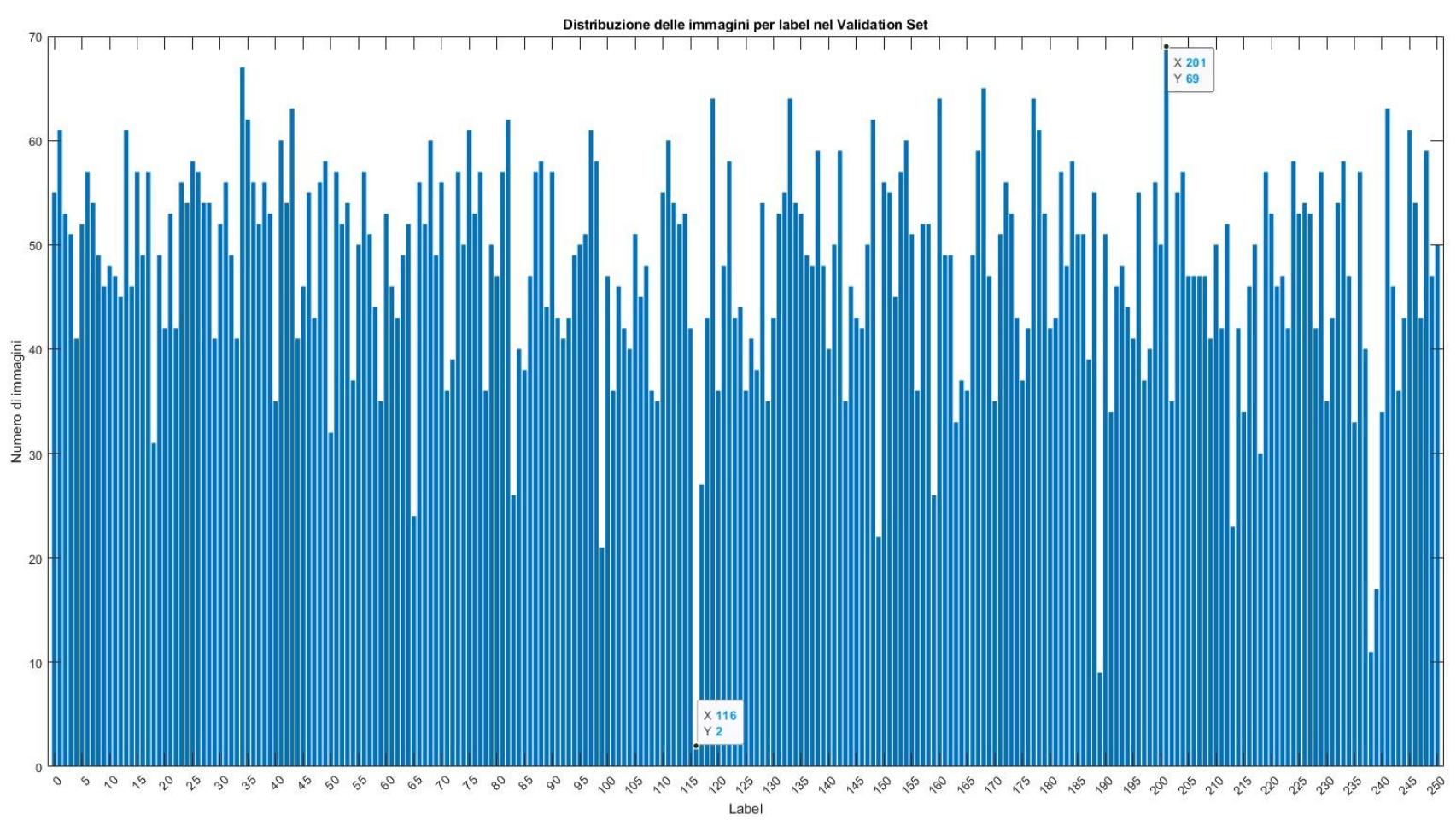


Considerazioni Finali

- Cosa possiamo migliorare?
 - Magari un addestramento diverso, con iperparametri diversi, riesce a creare delle pseudo-label più precise o aumentare l'accuracy generale, e quindi successivi train potrebbero portare ad un aumento significativo dell'accuracy
 - Aumentare il numero di Round di addestramento
 - Utilizzare una Rete più profonda

Degraded
Validation Set

The background features a series of overlapping, diagonal stripes in shades of dark grey, orange, and red. These stripes create a sense of depth and movement across the right side of the slide.





Analisi del Validation set Degradato

L' **obiettivo** di questo lavoro è ottenere le migliori performance di classificazione su un validation set composto da immagini acquisite in condizioni non ottimali.

La **sfida** principale posta dal validation set degradato è rappresentata dalla difficoltà, per un modello di classificazione, di riconoscere le caratteristiche distintive delle classi.

La **soluzione** proposta prevede un'operazione di **data augmentation** sul train set. In particolare, questa operazione consiste nell'identificare e replicare nel train set i difetti presenti nel validation set degradato. In questo modo, il training set viene arricchito con immagini sia originali che degradate.

Questa strategia produce due vantaggi principali:

- aumenta il numero di istanze per ciascuna classe nel train set
- introduce una maggiore variabilità all'interno del train set

Tali effetti contribuiscono a migliorare la capacità di generalizzazione dei modelli di classificazione.



Blur

Il primo artefatto analizzato è il blur, la cui entità è stata misurata utilizzando un algoritmo basato sulla varianza del Laplaciano. La procedura adottata è la seguente:

- Ogni immagine è stata convertita in scala di grigi (se a colori), per semplificare l'elaborazione e uniformare i dati di input
- È stato applicato un filtro Laplaciano all'immagine in scala di grigi. Questo filtro evidenzia i bordi e le variazioni di intensità, elementi sensibili alla presenza di blur
- La **varianza del Laplaciano** è stata calcolata per misurare il livello di sfocatura. Il valore restituito, denominato *blur score*, rappresenta una misura quantitativa (calcolata attraverso la varianza del Laplaciano): valori più bassi indicano immagini più sfocate, mentre valori più alti indicano una maggiore nitidezza. È robusto a variazioni luminose, ma può generare falsi positivi in texture uniformi (es. cieli).
- È stato scelto un *blur score* < 100 , questa scelta è supportata da test empirici condotti su un ampio campione di immagini, che hanno dimostrato come valori inferiori a questa soglia corrispondano a **sfocature evidenti**

Blur

Questa metodologia presenta alcuni vantaggi significativi:

- La misura del blur è completamente automatica e non richiede l'utilizzo di metriche di riferimento (no reference)
- Il calcolo della varianza del Laplaciano si è rivelato efficace nell'identificazione del livello di sfocatura, consentendo una rapida analisi e classificazione delle immagini degradate da blur





Rumore Gaussiano

Molte immagini sono affette da rumore gaussiano. Per stimare il livello di rumore presente, è stato utilizzato l'algoritmo di **Noise Level Estimation** [7], che si basa sulle seguenti fasi:

- Ogni immagine è suddivisa in patch di dimensioni predefinite (ad esempio, 7x7 pixel).
- Si applicano filtri derivativi orizzontali e verticali all'immagine per rilevare le variazioni di intensità tra pixel vicini, evidenziando bordi, gradienti e strutture locali
- L'algoritmo identifica patch con texture deboli, poiché queste sono più adatte per stimare il livello di rumore di fondo
- La stima del livello di rumore (deviazione standard) è effettuata iterativamente, migliorando la selezione delle patch ad ogni ciclo
- L'output dell'algoritmo fornisce una misura quantitativa del livello di rumore per ciascun canale dell'immagine, rappresentata dalla variabile **nlevel**

Rumore Gaussiano

Per **classificare** un'immagine con rumore gaussiano, è stato impostato un $nlevel > 2$. Tale valore è stato scelto in base ad analisi empiriche sul dataset, evidenziando che le immagini con livelli di rumore al di sopra di **questa soglia** presentano una significativa degradazione causata dal rumore gaussiano



Compressione JPEG

Molte immagini analizzate risultano affette da compressione, in particolare da artefatti tipici della compressione JPEG. Si è quindi proceduto a calcolare il livello di compressione utilizzando due metriche di qualità delle immagini: **BRISQUE** [5] e **PIQE** [6]. In particolare:

- per ogni immagine, è stato calcolato il punteggio **BRISQUE** (Blind/Reference-less Image Spatial Quality Evaluator) per determinare la qualità percepita senza riferimento
- contemporaneamente, è stato calcolato il punteggio **PIQE** (Perception-based Image Quality Evaluator), che rileva regioni degradate analizzando patch locali nell'immagine
- Le immagini sono state classificate come **altamente compresse** quando entrambi i punteggi BRISQUE e PIQE superano una **soglia** ($\text{BRISQUE} > 40$ e $\text{PIQE} > 65$). Tale soglia è stata stabilita **empiricamente**, osservando che oltre un certo valore le immagini mostrano in modo evidente artefatti tipici della compressione





Replicazione dei “rumori”

Il tentativo iniziale di restaurare queste immagini ha evidenziato alcune criticità significative:

- La natura aleatoria dei difetti ha rappresentato la prima sfida importante: ogni immagine presentava una combinazione diversa di degradazioni, con intensità variabili e distribuzioni non uniformi. Questa eterogeneità ha reso impraticabile lo sviluppo di un algoritmo automatico di restauro universalmente efficace
- La seconda opzione considerata, il restauro manuale delle singole immagini, si è rivelata irrealizzabile dato il volume considerevole del dataset (circa 12.000 immagini), richiedendo un investimento di tempo e risorse non sostenibile

Si è deciso quindi di studiare e replicare artificialmente i difetti identificati attraverso tecniche di Data Augmentation. Questa strategia ha permesso di affrontare il problema di classificazione del validation set degradato in modo più efficiente e metodologicamente solido



Replicazione dei “rumori”

Per replicare i rumori identificati nel validation set degradato sono state utilizzate le seguenti tecniche di degradazione:

1. **Blur:**
 - Applicato un filtro gaussiano con $\sigma = 3$
 - Il kernel del filtro è stato generato usando `fspecial('gaussian')`
2. **Rumore Gaussiano:**
 - Parametri: media = 0.0, varianza = 0.06
 - Aggiunge rumore bianco gaussiano additivo all'immagine
3. **Compressione JPEG:**
 - Quality factor impostato a 9 (scala 1-100)
 - Simula la degradazione tipica della compressione lossy



Degradazioni sul Train Dataset

Successivamente si è proceduto a replicare le degradazioni sul train set.

Per ogni classe, il train set è stato suddiviso in quattro gruppi:

- 25% delle immagini è stato mantenuto pulito, senza applicare alcun rumore
- 25% delle immagini è stato degradato applicando un rumore di tipo **Blur**
- 25% delle immagini è stato degradato aggiungendo **Rumore Gaussiano**
- 25% delle immagini è stato degradato mediante **Compressione JPEG**

In questo modo, per ciascuna classe, il **75%** delle immagini è stato degradato con diversi tipi di rumore, mantenendo il restante **25%** pulito per garantire una varietà equilibrata di dati.

Esempi di Replicazione





Creazione del “Validation Degradato Ridotto”

Per ottenere un validation set privo, o quasi, di immagini eccessivamente degradate:

- Sono state generate **tre liste** con i nomi delle immagini altamente degradate, classificate in base a:
 - **Alta compressione**
 - **Eccessivo blur**
 - **Rumore gaussiano elevato**
- Queste liste sono state ottenute utilizzando **algoritmi specifici** (precedentemente spiegati) per rilevare i livelli di degrado.
- Le immagini identificate sono state **rimosse** dal validation set degradato, creando il "Validation Ridotto", composto da immagini con degrado minimo o nullo.



Risultati con Resnet18

Train set “Small” Degradato → Val set Pulito (Accuracy ≈ 20.12%):

- Nonostante la dimensione ridotta, il modello riesce a **vedere una varietà di esempi** sufficienti (inclusi quelli puliti) per imparare **caratteristiche utili** a riconoscere le classi sul validation set pulito. Di conseguenza, l'accuratezza sale fino a circa **20%**, un valore nettamente superiore rispetto agli altri esperimenti ma non riesce a generalizzare **efficacemente** al validation set.

Train set Degradato → Val set Pulito (Accuracy ≈ 1.10%):

- Il motivo principale è il **forte mismatch** tra le immagini viste in training (molto degradate) e quelle pulite del validation set, che rende difficile per la rete generalizzare su immagini di qualità diversa.



Risultati con Resnet18

Train set Degradato → Val set Degradato (Accuracy $\approx 0.54\%$):

- Poiché stavolta il validation set è estremamente degradato, con molte immagini quasi irriconoscibili. Anche se il train è su immagini degradate, il livello di “rumore” nel validation può risultare ancora più severo, riducendo ulteriormente la precisione.

Train set Degradato → Val set Degradato “Ridotto” (Accuracy $\approx 0.86\%$):

- Grazie all’eliminazione delle immagini troppo compromesse. Rimuovendo i casi estremi, la rete riesce a classificare meglio, incrementando l’accuratezza complessiva rispetto al **Validation Degradato**.



Riassunto Risultati

Risultati ottenuti usando il modello ResNet18:

Train Set	Validation / Test Set	Validation / Test Accuracy
Small Augmented	Pulito	32.3162%
Full	Pulito	29.8733%
Full	Degrado	18.9178%
Small Degrado	Degrado	20.12%
Full Degrado	Pulito	1.1006%
Full Degrado	Degrado	0.54194%
Full Degrado	Degrado Ridotto	0.8581%



Conclusioni

- Non possiamo sicuramente dire che abbiamo ottenuto dei buoni risultati, né che abbiamo ottenuto dei risultati che possano validare il nostro workflow.
- Nonostante questo ci sentiamo di dire che questo ci sembra valido, e che avremmo ottenuto migliori risultati se:
 - Avessimo effettuato un clustering più puntuale
 - Avessimo utilizzato una rete neurale con più strati, le componenti Hardware dei nostri PC e soprattutto il tempo impiegato da queste per il singolo allenamento ci hanno limitato a reti poco profonde.
 - Avessimo lavorato di più per trovare degli iperparametri migliori
 - Avessimo effettuato pseudo-labeling anche sui Degraded Set
- Ci sentiamo quindi soddisfatti solo in parte del nostro lavoro, soprattutto a causa degli scarsi risultati finali.

Grazie!

per la vostra attenzione



Appendici





Algoritmo K-Means [1]

Algoritmo di clustering non supervisionato utilizzato per suddividere un insieme di dati in k gruppi distinti. In ordine:

1. Si selezionano casualmente k centroidi
2. Ogni punto del dataset viene assegnato al cluster del centroide più vicino, utilizzando la distanza euclidea
3. Per ogni cluster, si calcola la posizione media dei punti assegnati al cluster, aggiornando la posizione del centroide
4. Questo si ripete fino a quando i centroidi convergono oppure viene raggiunto un numero massimo di iterazioni

L'algoritmo restituisce k cluster, con i loro centroidi e i punti assegnati a ciascun cluster



PCA [2]

Tecnica di riduzione della dimensionalità che identifica le componenti principali di un dataset per rappresentarlo in uno spazio di dimensioni ridotte.

- Si **standardizzano** i dati (z-score)
- Si calcola la **matrice di covarianza** del dataset per catturare le relazioni lineari tra le variabili
- Si calcolano gli **autovalori** e gli **autovettori** della matrice di covarianza. Gli autovalori rappresentano la varianza spiegata da ciascuna direzione, mentre gli autovettori definiscono le componenti principali
- Gli autovalori vengono **ordinati**. Gli autovettori associati ai maggiori autovalori corrispondono alle componenti principali più importanti

Nel nostro caso si sono scelte le prime K componenti fino al raggiungimento del **90% della varianza spiegata**



Reti Neurali Profonde [3]

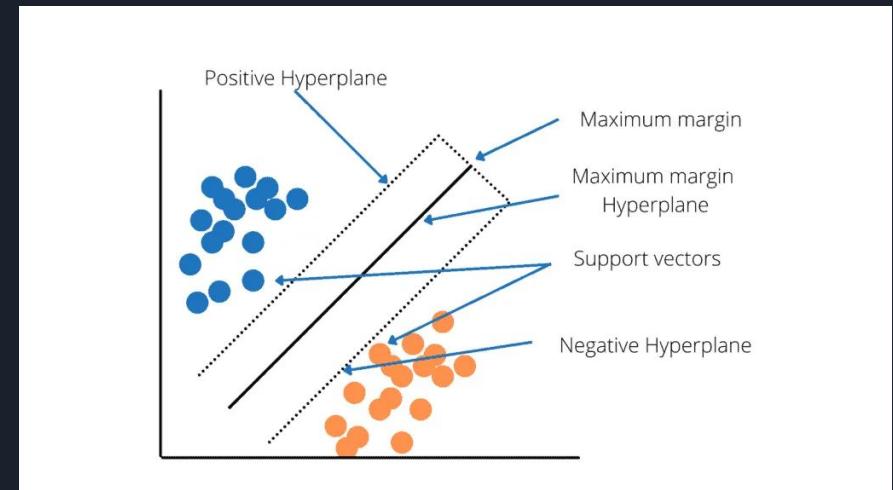
Le reti neurali profonde sono modelli di apprendimento automatico composti da più strati di neuroni artificiali, progettati per apprendere rappresentazioni complesse dai dati.

- **Strati convoluzionali:** Estraggono caratteristiche locali (come bordi e texture nelle immagini) attraverso funzioni kernel o filtri
- **Strati di attivazione:** Introducono non linearità con funzioni come ReLU (Rectified Linear Unit)
- **Strati di pooling:** Ridimensionano i dati riducendo la dimensionalità
- **Strati Fully connected:** Combinano le caratteristiche apprese per fare previsioni.

Support Vector Machine [4]

fitcecoc è una funzione in Matlab che permette di addestrare modelli Error-Correcting Output Codes (ECOC).

Questo metodo scomponete un problema di classificazione multiclasse in molteplici problemi di classificazione binaria più semplici, per poi combinare le soluzioni di questi sottoproblemi per ottenere la classificazione finale.





BRISQUE [5]

BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator):

- È un metodo **No-Reference** che sfrutta la teoria delle **Natural Scene Statistics** (NSS).
- Calcola e analizza i **coefficienti MSCN** (Mean Subtracted Contrast Normalized) per descrivere la distribuzione dei livelli di luminanza normalizzati nell'immagine.
- Questa distribuzione è modellata con curve di tipo **generalized Gaussian** e **asymmetric generalized Gaussian**; lo scostamento dai parametri di riferimento indica il grado di degradazione.
- I **feature vector** estratti vengono poi passati a un **regressore** (ad es. SVM) che produce un unico valore numerico inversamente proporzionale alla qualità percepita.



PIQE [6]

PIQE (Perception-based Image Quality Evaluator):

- Anche questo è un indice **No-Reference** che non richiede immagini di confronto.
- Suddivide l'immagine in **patch** e valuta localmente le **aree degradate**, analizzando l'entità delle distorsioni (ad es. artefatti di compressione, rumore).
- Quantifica la percentuale di patch compromesse e calcola un punteggio finale che rispecchia la qualità globale percepita: un punteggio più alto indica una qualità peggiore.