

**UNIVERSITÀ DEGLI STUDI DI CAGLIARI**

**FACOLTÀ DI SCIENZE**

Corso di Laurea Triennale in Informatica

**Seed Image Analysis by Deep Learning**

**Relatore Studente**

Prof. Cecilia Di Ruberto Mauro Loddo

Matr. N. 65493

ANNO ACCADEMICO 2019/2020

L'agricoltura è il settore economico più importante di molti

paesi in via di sviluppo.

L'utilizzo della tecnologia nei lavori agricoli

parte dall'inizio del XX secolo, quando l'industria passò da macchinari trainati da animali a macchinari meccanizzati

Il progresso tecnologico ha contribuito ad aumentare le discussioni riguardo la raccolta di grandi quantità di dati che compongono l'attuale informatica

ambientale, considerando che la raccolta dei dati non è un concetto nuovo, specialmente in

contesto della raccolta di dati pubblici.

L’utilizzo più efficiente di queste tecnologie e della digitalizzazione dei dati hanno permesso la creazione di record di grandi dimensioni che possono essere valutati ed analizzati in modo tempestivo e utile.

Questo lavoro di tesi è incentrato sulla ricerca di dataset di semi vegetali e sulla classificazione di essi in base alle famiglie di appartenenza utilizzando deep learning e CNN.

**Indice**

[Introduzione 1](#_Toc54629773)

[1.1 Cos’è l’Image Processing 1](#_Toc54629774)

[1.2 Deep Learning 2](#_Toc54629775)

[Stato dell’arte 3](#_Toc54629776)

[Preprocessing 5](#_Toc54629777)

[Sperimentazione 8](#_Toc54629778)

[4.1 Approccio generale 8](#_Toc54629779)

[4.2 Differenze e somiglianze tra le CNNs 9](#_Toc54629780)

[4.3 Sperimentazione 10](#_Toc54629781)

[4.4 Analisi dei dati del primo dataset 14](#_Toc54629782)

[4.5 Analisi dei dati del secondo dataset 16](#_Toc54629783)

[Conclusioni 18](#_Toc54629784)

[References](#_Toc54629785)

# **Capitolo 1**

# **Introduzione**

## **1.1 Cos’è l’Image Processing**

Noi umani siamo prevalentemente creature che basano la propria vita sulla vista, ci affidiamo quasi esclusivamente alla vista per dare un senso al mondo che ci circonda. Non ci limitiamo ovviamente solo a guardare le cose e classificarle ma ne riconosciamo differenze, caratteristiche e particolari.

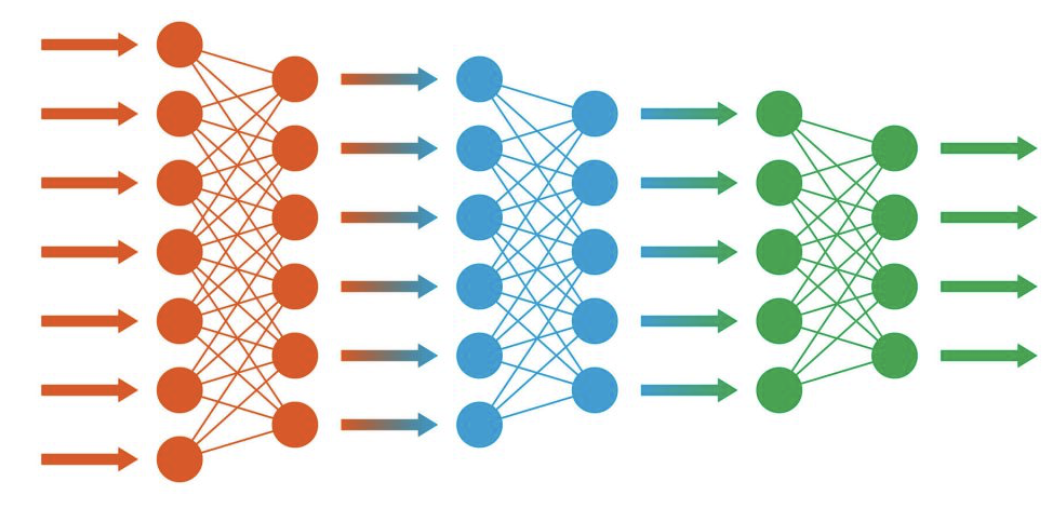
L’Image Processing si occupa di cambiare la natura di un’immagine in modo da sia migliorare le proprie informazioni visuali per l’interpretazione umana, che renderle più consone all’utilizzo che ne vogliono fare programmi e macchinari. Bisogna però precisare che queste due cose spesso non possono andare d’accordo proprio perché un umano preferisce un’immagine nitida, colorata e ricca di dettagli mentre una macchina preferisce avere delle immagini semplici e ordinate.

Le tecniche di Image Processing possono essere molteplici, le più comuni sono quelle che ci permettono di tramutare un’immagine a più canali (ad esempio una RGB) in un’immagine bidimensionale e quindi in scala di grigi; un’altra tecnica molto comune è lo *sharpening* che fa in modo di accentuare gli angoli dell’immagine, quindi utilizzato in maniera appropriata farebbe in modo di rendere più evidenti i particolari di un’immagine leggermente sfocata. Altre tecniche molto importanti sono quelle che ci permettono di ridurre il rumore dall’immagine o di rilevare gli elementi presenti con tecniche di *edge detection* o tramite l’utilizzo di *bounding box* (esattamente la tecnica che vedremo nel capitolo 2 sul preprocessing).

## **1.2 Deep Learning**

Il Deep Learning costituisce una tecnica recente e moderna per l'elaborazione delle immagini e l'analisi dei dati, con risultati promettenti e con un grande potenziale. Poiché il Deep Learning è stato applicato con successo in vari campi, recentemente è entrato anche nel campo dell'agricoltura.

Il Deep Learning fa parte di una famiglia più ampia di metodi di Machine Learning basati sull’assimilazione di rappresentazioni di dati e indica la branca dell’intelligenza artificiale che fa riferimento agli algoritmi ispirati alla struttura e alla funzione del cervello chiamate reti neurali artificiali. Esso quindi basa il suo funzionamento sulla classificazione e “selezione” di dati più rilevanti per arrivare ad una conclusione. Il Deep Learning è un sistema che sfrutta vari livelli di apprendimento automatico che utilizzano l’output del livello superiore come input di conseguenza le caratteristiche di ogni livello dipendono da quelle del livello superiore.



**Figura 1.1:** *Rappresentazione generica di una rete neurale*

Il Deep Learning quindi sfrutta le reti neurali artificiali, nello specifico il lavoro di tesi si è basato sulle CNNs (Convolutional Neural Networks), uno specifico tipo di reti neurali risultate veramente efficienti nelle aree di riconoscimento e classificazione di immagini. Esistono vari tipi di CNN che differiscono le une dalle altre per il numero di layer e la loro architettura ma tutti quanti hanno in comune un layer di input e dei convolution layer

Il lavoro di tesi è incentrato sulla classificazione di semi tramite Deep Learning e CNNs (Convolutional Neural Network).

**Capitolo 2**

# **Stato dell’arte**

Le applicazioni del Deep Learning nell’agricoltura sono innumerevoli, dall’identificazione di tipi di erbe al riconoscimento di piante, dal conteggio dei frutti alla classificazione del tipo di coltura. L’utilizzo di queste tecniche nell’agricoltura è piuttosto recente dato che la maggior parte dei papers che si possono trovare risalgono a dopo il 2015.

Molti papers si occupano di classificazione e identificazione di aree di interesse, incluso il rilevamento di ostacoli (campo che non fa al caso nostro). Alcuni altri papers si focalizzano su problemi quali il rilevamento di erbacce e la ricerca sul suolo.

Lavori simili a quello di questa tesi possono ad esempio essere quello di HallMcCool, Dayoub, Sunderhauf e Upcroft che nel 2015 hanno utilizzato una CNN definita dall’autore per occuparsi di classificare le foglie provenienti da diverse specie di piante utilizzando un dateset di oltre 1900 immagini divise in 32 specie. Allo stesso modo Sladojevic, Arsenovi, Anderla, Culibrk, e Stefanovic utilizzano nel 2016 la CNN CaffeNet per individuare le malattie delle foglie su un database di circa 4500 immagini. Il Deep Learning utilizzato per l’identificazione e la classificazione è mostrato dal lavoro di Mohanty, Hughes e Salathé che hanno utilizzato AlexNet e GoogleNet identificare 14 specie coltivate e 26 malattie e dal lavoro di classificazione di foglie di banano malate svolto da Amara, Bouaziz, e Algergawy utilizzando la CNN LeNet.

Altri lavori di classificazione sono stati svolti su tipi di raccolto differenti:

Kussul, Lavreniuk, Skakun, e Shelestov hanno lavorato sulla classificazione di colture di grano, mais, semi di soia, girasole e barbabietola da zucchero utilizzando una CNN creata dal’autore; La classificazione di colture oleaginose, ravanelli, orzo, erba e erbacce è stata svolta da Mortensen, Dyrmann, Karstoft, Jørgensen, e Gislum nel 2016 con una versione adattata di VGG16; Rebetez invece ha svolto un lavoro di classificazione di diversi tipi di raccolto tramite immagini UAV ed ha utilizzato una CNN accoppiata con un HistNN (un istogramma RGB).

**Capitolo 3**

# **Preprocessing**

Per quanto riguarda i dataset utilizzati, il primo è stato scaricato dal sito [*www.canada.ca*](https://www.canada.ca/en.html)contenente un enorme database di semi suddivisi per famiglie.  
Questo sito è stato preferito ad altri per la sua grande popolazione di famiglie e la pulizia negli sfondi delle immagini che sono fattori fondamentali per il lavoro richiesto.

Le famiglie scelte per i test sono state selezionate in base al numero di campioni mentre le famiglie contenenti uno scarso numero di immagini sono state scartate da subito.

Per quanto riguarda il secondo dataset, si tratta di un insieme di immagini di semi provenienti dall’orto botanico di Cagliari contenente in totale più di 3300 semi tutti appartenenti alla famiglia delle *Fabaceae* ma suddivisi in circa 115 specie differenti.

Anche da questo dataset sono state scelte le famiglie più numerose ed è stato applicato lo stesso procedimento di preprocessing attuato col primo dataset.

* Il primo passo consiste nella creazione delle maschere binarie dei semi. Questo passaggio è stato reso possibile dalla presenza di uno sfondo abbastanza uniforme delle immagini, infatti il codice utilizzato per la creazione delle maschere fa in modo di individuare il picco più alto dell’istogramma dell’immagine rappresentante i valori dello sfondo ed escluderli dall’immagine secondo un range del 5%

**Immagine che contiene frutta, tavolo, sedendo, piatto

Descrizione generata automaticamenteImmagine che contiene edificio

Descrizione generata automaticamente**

**Figura 3.1:** *Esempio di semi presenti nel dataset e di maschera creata di conseguenza.*

* A questo punto avremo le maschere binarie anche delle immagini contenenti più di un seme; queste maschere verranno utilizzate per individuare i bounding box dei semi e di conseguenza applicati i box alle immagini rgb si otterranno i semi singoli.

Immagine che contiene pesce

Descrizione generata automaticamente

**Figura 3.2:** *Esempio di seme ritagliato dall’immagine completa*

* Queste immagini così ottenute avranno il problema di contenere al loro interno delle porzioni dei semi che in precedenza erano troppo vicini al seme che è stato scelto per essere ritagliato col bounding box; per risolvere il problema si riutilizza il codice utilizzato in precedenza per creare le maschere per ottenere nuove maschere con i semi ritagliati.

****

**Figura 3.3:** *Esempio maschera ottenuta dal seme ritagliato*

* Quest’ultime verranno poi passate ad un nuovo script che analizza la presenza degli elementi nelle maschere e rimuove le porzioni che occupano meno spazio riempendone i pixel con valore rgb 000000.

****

**Figura 3.4:** *Esempio di maschera pulita dalla porzione di seme indesiderato*

* Ottenute le nuove maschere pulite non resta che applicarle alle immagini rgb per escludere tutti gli elementi che non rappresentano il seme principale utilizzando l’ultimo script che prende il valore del pixel dello sfondo e lo sostituisce ai valori dei pixel che non combaciano con la maschera

**Immagine che contiene pesce

Descrizione generata automaticamente**

**Figura 3.5:** *Esempio di seme pronto per l sperimentazione*

**Capitolo 4**

# **Sperimentazione**

Questo capitolo verterà sull’approccio utilizzato per la sperimentazione per valutare la CNN migliore

## **4.1 Approccio generale**

Le immagini utilizzate per la sperimentazione hanno tutte in comune il fatto di avere uno sfondo uniforme e di presentare un solo seme per immagine.

Il primo passo per la classificazione è quello di avere un dataset ben fornito e con ogni immagine suddivisa per categoria per poter addestrare la rete. I dataset, per quanto numerosi, vengono sottoposti ad un processo di augmentation; per fare ciò si utilizza la funzione di matlab [*imageDataAugmenter*](https://www.mathworks.com/help/deeplearning/ref/imagedataaugmenter.html#d122e54287)che permette di aumentare a runtime la dimensione di un image datastore popolandolo di immagini ricavate dal datastore originale ma applicandovi delle trasformazioni geometriche come per esempio rotazioni, traslazioni o ridimensionamenti. Questa funzione risulta particolarmente comoda dato che permette di ottenere dataset più grandi senza dover avere migliaia di immagini nel folder utilizzato.

L’imageDataAugmenter è stato inoltre utile per poter ridimensionare le immagini della dimensione richiesta come input dalle varie CNNs dato che non tutte le reti prendono in input immagini della stessa dimensione.

Le immagini presenti nel datastore vengono inoltre suddivise in 3 gruppi, uno per il Training (l’80% delle immagini), uno per la Validation (il 10% delle immagini) e in fine uno per il Testing (il restante 10% delle immagini).

Come suggerisce il nome del gruppo, quello del Training viene utilizzato per addestrare la rete; queste saranno quindi le immagini che verranno passate alla rete come campioni per essere analizzate e da queste la rete otterrà tutte le informazioni necessarie per poter riconoscere le categorie con cui sta lavorando.

Le immagini per la Validation invece sono utilizzate per validare la rete ogni numero di iterazioni prefissate e serviranno per determinare l’accuratezza della validazione; in poche parole la precisione della classificazione sul Validation set.

Il Test set infine viene utilizzato dalla funzione *classify* che prende in input la rete che si è appena creata e il set di immagini che verranno “etichettate” dalla funzione in base alle categorie riconosciute dalla rete.

[Train, Test, Validations] = splitEachLabel(im,0.8,0.1,0.10,'randomized');

Train = augmentedImageDatastore([227 227], Train,'DataAugmentation',imageAugmenter);

Validations = augmentedImageDatastore([227 227], Validations, 'DataAugmentation',imageAugmenter);

Per i primi test svolti sono state utilizzate 6 categorie differenti di semi appartenenti alle famiglie delle:

Amaranthaceae, Apiaceae, Asteraceae, Brassicaceae, Plantaginaceae, Solanaceae

Immagine che contiene fotografia, diverso, tavolo, sedendo

Descrizione generata automaticamente

**Figura 4.1:** *Esempio di semi presenti nel dataset.*

## **4.2 Differenze e somiglianze tra le CNNs**

I test sono stati svolti su diverse CNNs con lo scopo di trovare la migliore sotto vari aspetti. Ogni CNN differisce dalle altre specialmente per il numero di livelli ma sono tutte accomunate dalla presenza di un livello completamente connesso (Fully Connected Layer) e da un livello di convoluzione (Convolutional Layer).

Un Livello Completamente Connesso è un livelo della CNN che prende in input n classi, le moltiplica utilizzando una “matrice di peso” (weight matrix) e aggiunge un vettore di bias; questo è un livello molto importante della rete perché per ogni rete è stato necessario modificare il numero delle classi in input per adattarlo al numero di classi presenti nel dataset.

Un Convolutional Layer invece è un livello di una rete che prende in input un set di immagini 2D e restituisce in output un altro set di immagini 2D; la tecnica della convoluzione è una tecnica che permette di indiviiduare uno specifico pattern all’interno di un’imagine. Quello che fa in sostanza un Convolutional Layer è applicare un filtro ad un’immagine (normalmente il filtro è molto più piccolo dell’immagine intera) e muoverlo per tutta la sua interezza creando così una nuova immagine che sarà più nitida nei punti in cui il filtro combacia e meno nitida nei punti in cui il filtro non combacia. Quindi ogni nodo in un livello di convoluzione possiede uno specifico filtro con un determinato pattern contenuto all’interno di un array.

Immagine che contiene schermo, edificio, luce, largo

Descrizione generata automaticamente

**Figura 4.2:** *Rappresentazione di un possibile filtro applicabile all’immagine.*

Immagine che contiene mammifero, zebra, disegnando

Descrizione generata automaticamenteImmagine che contiene zebra

Descrizione generata automaticamente

**Figura 4.3:** *Esempio di immagine originale e di immagine ottenuta applicando il filtro in figura 4.2*

## **4.3 Sperimentazione**

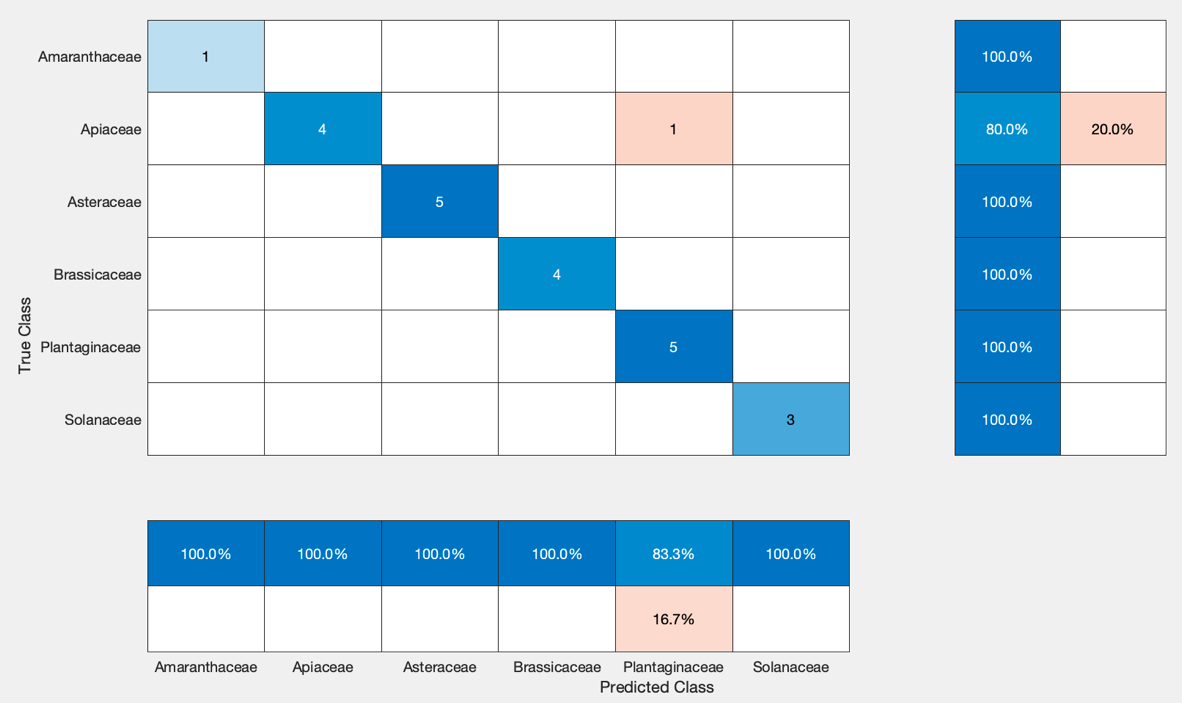
Il lavoro di sperimentazione, come detto in precedenza, è stato svolto su diverse CNNs:

* AlexNet
* ResNet50
* ResNet18
* ResNet101
* GoogleNet
* ShuffleNet
* SqueezeNet
* MobileNetV2
* InceptionV3
* VGG16

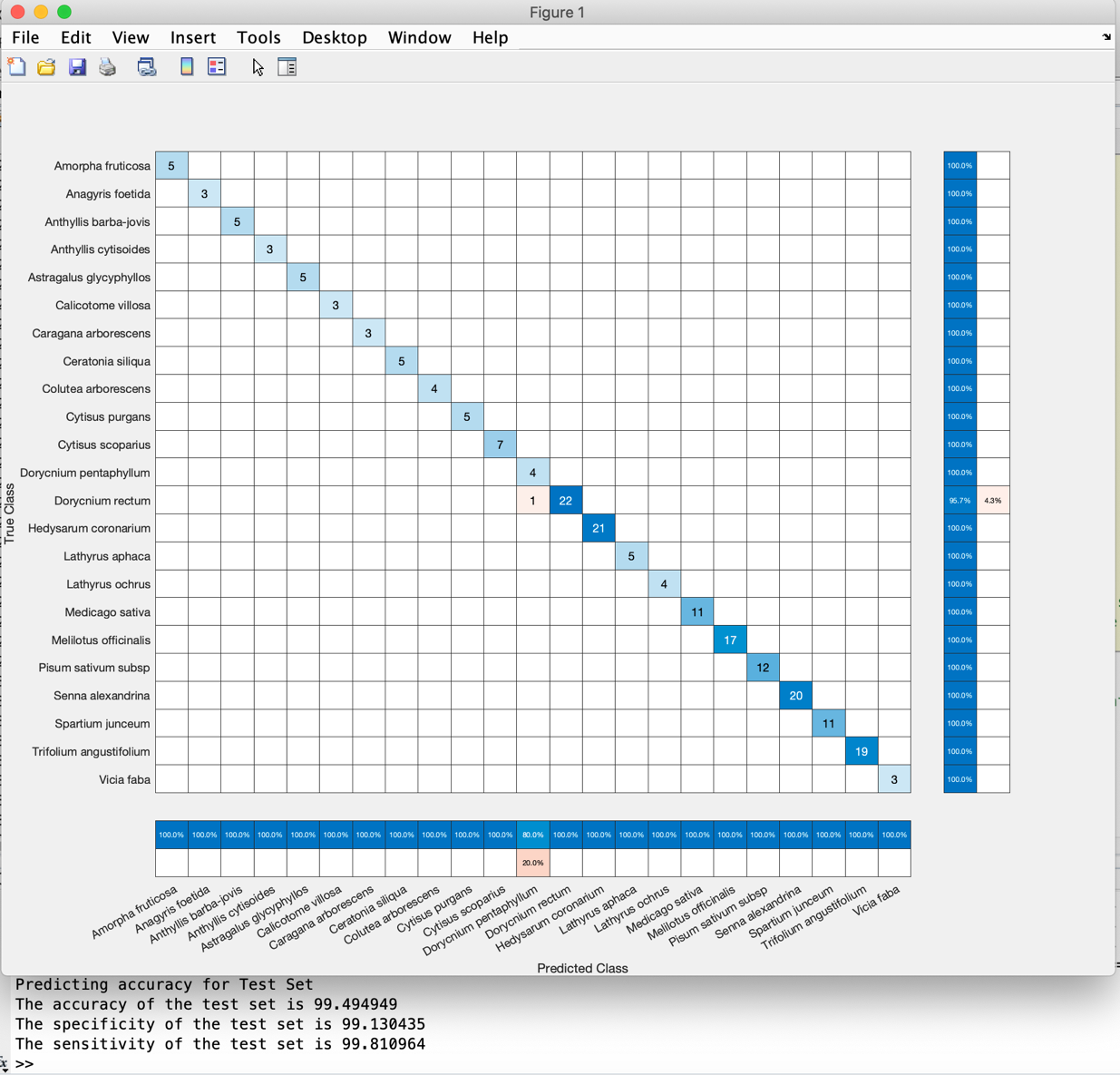
Le varie prove sono state inizialmente svolte utilizzando le stesse opzioni di training per tutte le reti per valutarne le differenze.

Con tutte è stato utilizzato il solver *Adam* che utilizza l’Adam optimizer; è stato scelto questo solver anziché *sgdm* perché sono stati riscontrati dei cali delle prestazioni utilizzandolo. Per ogni rete sono stati svolti 5 test diversi in modo da poter avere una media dei risultati dato che ogni test differisce dal precedente per la randomicità dei campioni e dell’augmentation.

Al completamento di ogni test viene inoltre mostrata la matrice di confusione generata dalla rete; essa serve a rappresentare la precisione della classificazione della rete: l’asse delle ascisse corrisponde alle classi che sono state predette dalla rete mentre quella delle ordinate corrisponde alle classi vere e proprie che ci aspettiamo. Tutti i valori presenti nella diagonale sono di conseguenza le immagini che sono state correttamente riconosciute, mentre tutti i valori all’esterno della diagonale principale rappresentano degli errori della rete appena addestrata.



**Figura 4.4:** *Esempio di Confusion Matrix ricavata dai primi test*



**Figura 4.5:** *Esempio di Confusion Matrix ricavata dai successivi test*

La Confusion Matrix è usata inoltre per poter calcolare la *Specificità* e la *Sensibilità* della rete:

* La Specificità o Specificity è definita come la proporzione di negativi attuali che sono predetti come negativi (detti anche *True Negative*), ciò implica che ci possano essere anche dei negativi predetti come positivi che vengono detti falsi positivi; la Specificity viene definita quindi come *False Positive Rate.*
* La Sensibilità o Sensitivity è definita come la proporzione di positivi attuali che sono predetti come positivi (detti anche *True Positive*), ciò implica che ci possano essere anche dei positivi predetti come negativi che vengono detti falsi negativi; la Sensitivity viene definita quindi come *True Positive Rate.*

Specificità e Sensibilità sono due dei tre valori utilizzati per valutare l’efficienza della rete neurale; il terzo è l’Accuratezza (detta anche Accuracy) definita come il rapporto tra tutte le previsioni corrette e il totale delle previsioni. Come suggerisce il nome, quanto più una rete è accurata, tanti meno errori di classificazione commetterà. Possiamo quindi dire che una rete neurale con accuratezza del 100% possa classificare con certezza tutte le famiglie utilizzate per il suo addestramento.

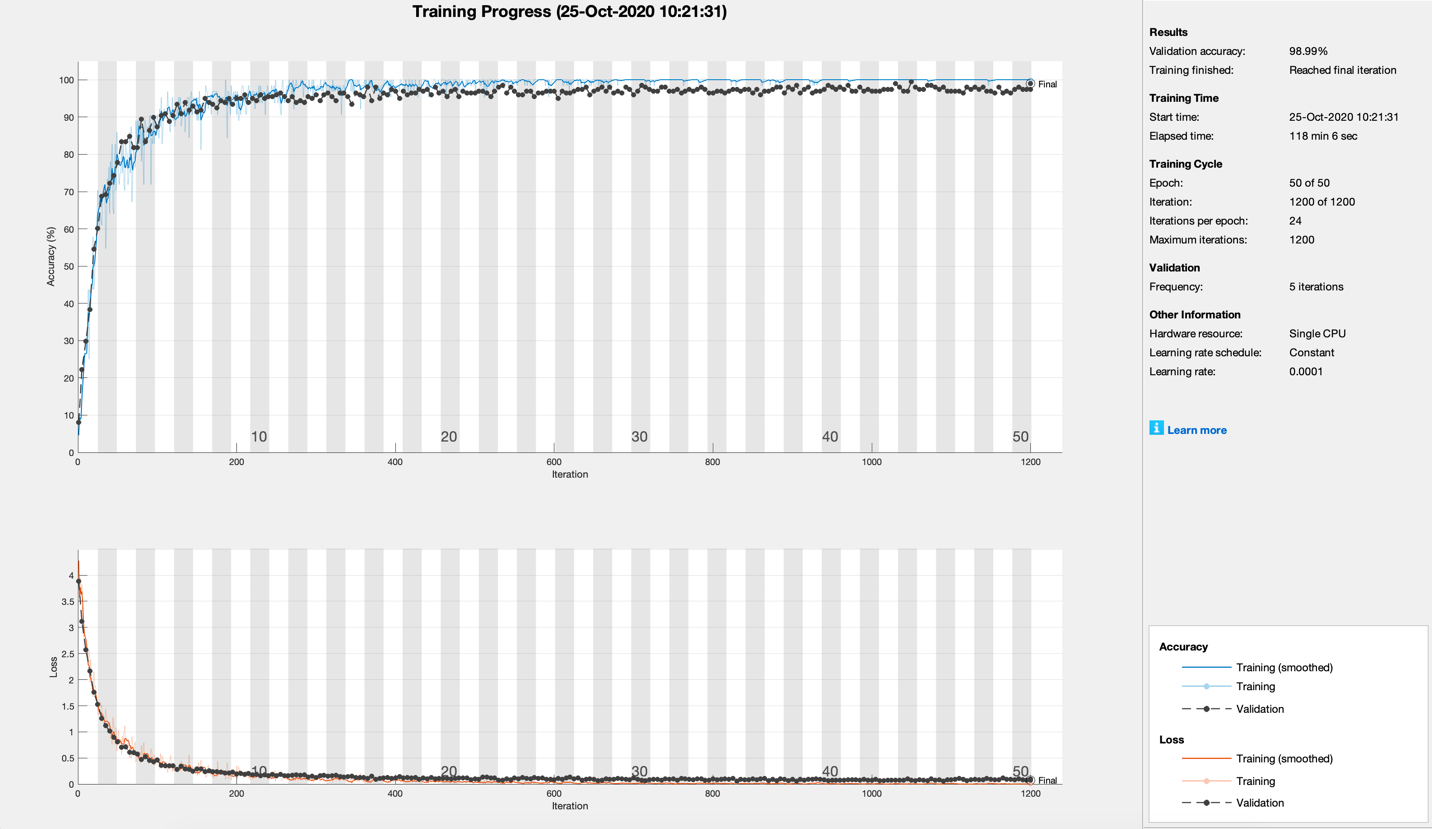
L’ultimo valore che si è andato ad analizzare è la deviazione standard dell’accuratezza, utilizzata proprio per trovare la CNN più stabile tra tutte. La Deviazione Standard è stata ottenuta come radice quadrata della Varianza

*Formula della varianza (M=media, Xi=valori calcolati, N=numero totale dei test)*

Durante ogni test viene mostrato il grafico del training progress, molto utile per poter tenere traccia dell’andamento della rete e del tempo trascorso nonché di tutte le varie opzioni di training; il grafico inoltre è suddiviso in due, nella parte superiore possiamo trovare il grafico dell’accuratezza (espressa in percentuale) con al suo interno la linea blu che sta ad indicare l’andamento del training e la linea nera tratteggiata che indica l’andamento della validazione che presenta un nodo in ogni punto in cui viene svolta. Come si può edere dall’immagine 4.6 la frequenza della validazione risulta di uno ogni 5 iterazioni.

Si può ben immaginare che nel caso del grafico dell’Accuracy, più esso tende a salire e di conseguenza al 100%, più la rete sarà precisa e otterrà buoni risultati.

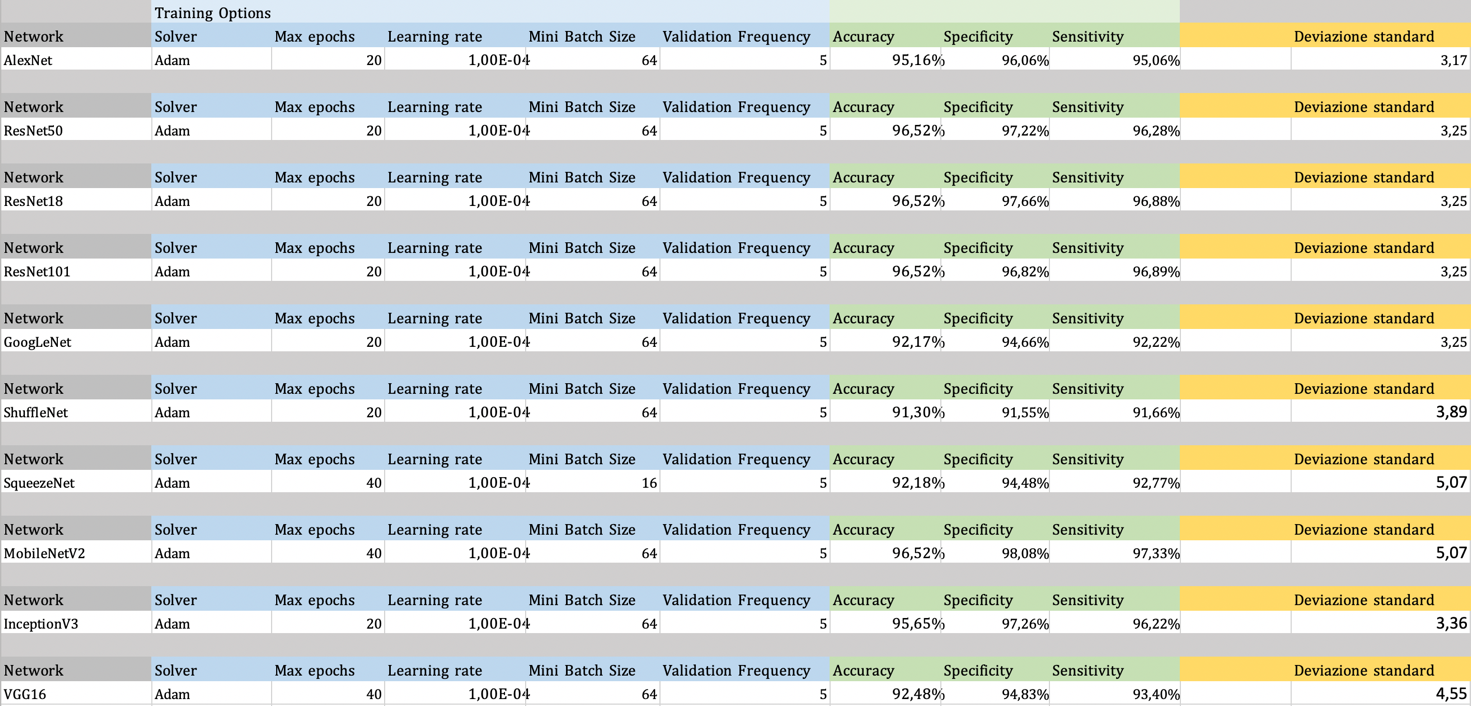
A differenza del primo grafico, il secondo rappresenta la funzione di perdita di ogni mini batch e la perdita del validation set, di conseguenza più essa tende a 0 meglio è.



**Figura 4.6:** *Esempio di grafico del training progress*

## **4.4 Analisi dei dati del primo dataset**

Come è stato già detto in precedenza, i test sono stati svolti su diverse CNNs utilizzando inizialmente le stesse opzioni di training. Dai primi test svolti, analizzando Accuratezza, Sensibilità e Specificità si è notato che le reti con opzioni di training quali il Numero Massimo di Epoche a 20 e la dimensione del Mini Batch a 64 non si è superata un’accuratezza del 96,52% come si può notare dall’immagine 4.7



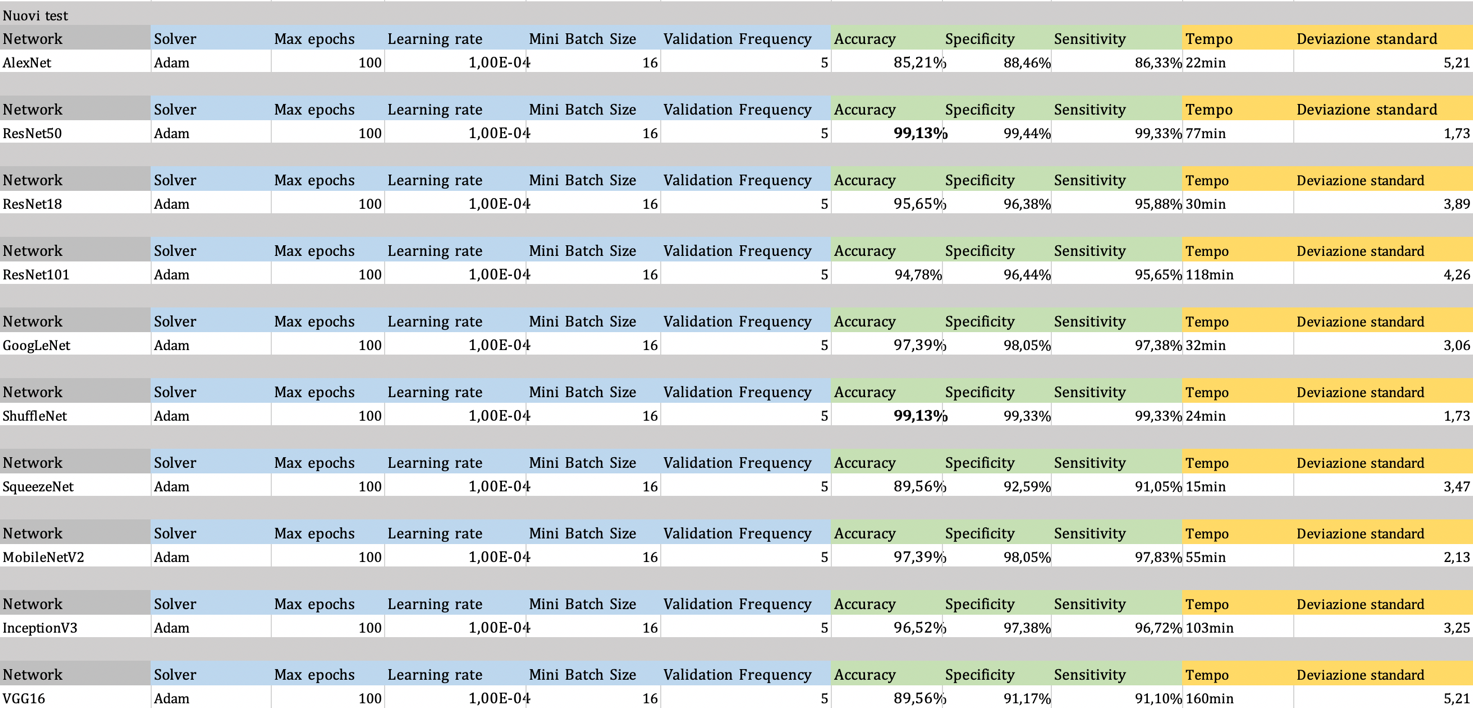
**Figura 4.7:** *Riepilogo dei primi test effettuati*

Queste informazioni sono molto importanti perché ci permettono di capire che le reti con un maggior numero di livelli risultano essere le più accurate e stabili, infatti le varie ResNet e MobileNetV2 sembrano ottenere i risultati migliori ma ResNet è decisamente più stabile, come può dimostrare la deviazione standard.

A questo punto sono stati svolti altri test per cercare di migliorare l’efficienza generale delle reti e si è riscontrato che diminuendo il Mini Batch Size e aumentando il numero di Max Epochs l’efficienza tende ad aumentare per la maggior parte delle reti.

Il Mini Batch è un sottoinsieme del Training Set utilizzato per validare il gradiente della funzione di perdita e per aggiornare i pesi dei noti della rete. Il numero di Max Epochs invece rappresenta il numero Massimo di epoche utilizzate per il training e un’epoca è un intero passo dell’algoritmo di training sull’intero training set.

Di seguito si possono trovare i risultati dei test svolti impostando il numero massimo di epoche a 100 e la dimensione del mini batch a 16



**Figura 4.8:** *Riepilogo dei secondi test effettuati*

A seguito del secondo set di test si può notare che ResNet50 e ShuffleNet raggiungono risultati veramente buoni con una deviazione standard piuttosto basa, ciò le rende sicuramente le reti migliori e più stabili; ShuffleNet inoltrerisulta essere anche una delle più veloci nell’addestramento, il che la rende particolarmente buona.

Altre reti invece, come ad esempio VGG16, nonostante il grande numero di livelli che dovrebbero aumentarne l’efficienza, hanno mostrato un calo delle prestazioni con deviazioni standard piuttosto alte a confermare la loro instabilità.

## **4.5 Analisi dei dati del secondo dataset**

Per il secondo dataset sono stati utilizzati, come detto in precedenza, dei semi provenienti dall’orto botanico di Cagliari, un dataset decisamente più popolato di quello utilizzato per i primi test. Una volta selezionati i semi da utilizzare si è ottenuto un dataset composto da poco meno di 2000 immagini suddivise in 23 classi differenti. Di seguito si possono vedere una serie di campioni randomici provenienti dal dataset.

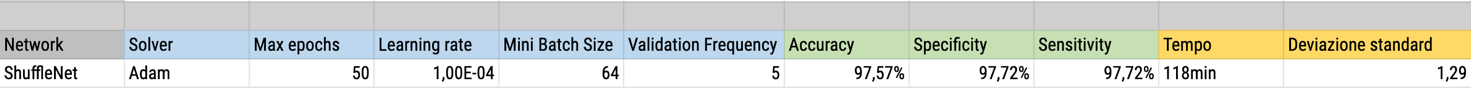
Immagine che contiene cibo, fotografia, sedendo, metallo

Descrizione generata automaticamente

**Figura 4.9:** *Campioni provenienti dal dataset cagliaritano*

I test sono stati svolti utilizzando ShuffleNet e ResNet50, le due reti che sono risultate essere le più stabili ed efficienti con il primo dataset ma le opzioni di training sono state alleggerite per evitare tempi di attesa troppo lunghi dovuti alla dimensione del dataset oltre nove volte maggiore rispetto a quello canadese; per velocizzare i test sono quindi stati utilizzati valori di Max Epochs pari a 50 anziché 100 e valori di Mini Batch Size di 64 anziché 16.

Il training con ShuffleNet ha prodotto risultati promettenti con precisione del 97,57% e deviazione standard abbastanza contenuta di 1,29, persino minore di quella ottenuta dal primo dataset con conseguente maggiore stabilità.



**Figura 4.10:** *Risultati di ShuffleNet*

**Capitolo 5**

# **Conclusioni**

# **Bibliografia**

1. Deep Learning in Agriculture: A Survey - Andreas Kamilaris1 and Francesc X. Prenafeta-Boldú Institute for Food and Agricultural Research and Technology (IRTA).
2. [mathworks.com](https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html) – Options for training deep learning neural network.
3. Machine Learning in Agriculture: A Review - Konstantinos G. Liakos , Patrizia Busato , Dimitrios Moshou, Simon Pearson and Dionysis Bochtis.
4. [ujjwalkarn.me](https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/) - An intuitive explanation of Convolutional Neural Network.
5. [mathworks.com](https://matlabacademy.mathworks.com/R2019b/portal.html?course=mldl#chapter=3&lesson=5&section=2) – Convolutional Layer.
6. An Introduction to Digital Image Processing with Matab – Alasdair McAndrew - Victoria University of Technology

**Elenco delle figure**

**Figura 1.1:** *Rappresentazione generica di una rete neurale*

**Figura 3.1:** *Esempio di semi presenti nel dataset e di maschera creata di conseguenza.*

**Figura 3.2:** *Esempio di seme ritagliato dall’immagine completa*

**Figura 3.3:** *Esempio maschera ottenuta dal seme ritagliato*

**Figura 3.4:** *Esempio di maschera pulita dalla porzione di seme indesiderato*

**Figura 3.5:** *Esempio di seme pronto per l sperimentazione*

**Figura 4.1:** *Esempio di semi presenti nel dataset.*

**Figura 4.2:** *Rappresentazione di un possibile filtro applicabile all’immagine.*

**Figura 4.3:** *Esempio di immagine originale e di immagine ottenuta applicando il filtro in figura 4.2*

**Figura 4.4:** *Esempio di Confusion Matrix ricavata dai primi test*

**Figura 4.5:** *Esempio di Confusion Matrix ricavata dai successivi test*

**Figura 4.6:** *Esempio di grafico del training progress*

**Figura 4.7:** *Riepilogo dei primi test effettuati*

**Figura 4.8:** *Riepilogo dei secondi test effettuati*

**Figura 4.9:** *Campioni provenienti dal dataset cagliaritano*

**Figura 4.10:** *Risultati di ShuffleNet*

**Ringraziamenti**