



SAPIENZA
UNIVERSITÀ DI ROMA

Land Classification: Neural Network approaches on satellite images

Facoltà di Ingegneria dell'informazione, informatica e statistica
Corso di Laurea in Informatica

Candidato

Paolo Renzi

Matricola 1887793

Relatore

Prof. Irene Amerini

Correlatore

Dr. Lorenzo Papa

Anno Accademico 2021/2022

Tesi discussa il 22 December 2022
di fronte a una commissione esaminatrice composta da:
Prof. ... (presidente)
Prof. ...
Prof. ...

Land Classification: Neural Network approaches on satellite images
Tesi di Laurea. Sapienza – Università di Roma

© 2022 Paolo Renzi. Tutti i diritti riservati

Questa tesi è stata composta con L^AT_EX e la classe Sapthesis.

Email dell'autore: renzi.1887793@studenti.uniroma1.it

Indice

Abstract	iii
1 Introduzione	1
2 Algoritmi di Deep Learning e soluzioni proposte	3
2.1 Perceptron	4
2.2 Convolutional Neural Network	6
2.2.1 MobileNet	7
2.2.2 EfficientNet	7
2.2.3 ResNet50	8
2.3 Soluzioni proposte	9
2.3.1 Multi Layer Perceptron	9
2.3.2 Rete convoluzionale	10
3 Risultati	11
3.1 Dataset e Dettagli Implementativi	11
3.2 Multi Layer Perceptron	13
3.3 Rete convoluzionale	21
4 Conclusioni e lavori futuri	41
Bibliografia	42

Abstract

I cambiamenti climatici stanno impattando sempre di più le nostre vite cambiando il mondo in cui viviamo. Il modo più efficace di monitorare questi fenomeni a larga scala è quello di osservarli da una prospettiva più ampia possibile e le immagini satellitari ci consentono di fare esattamente questo.

In particolare la missione dell'ESA Sentinel-2 è stata lanciata proprio con questo scopo. I dati catturati dal satellite sono stati resi pubblici in un formato altamente utilizzabile grazie al fatto che sono stati aggiunti al catalogo di dataset forniti dalla libreria Tensorflow. Questo dataset ha immagini di dimensioni 64x64 pixels ognuna con una label associata appartenente ad una delle seguenti 10 classi: Industrial Buildings, Residential Buildings, Annual Crop, Permanent Crop, River, Sea & Lake, Herbaceous Vegetation, Highway, Pasture, Forest che rappresentano 10 tipi di uso del suolo. Inoltre il dataset essendo composto da immagini satellitari sono rappresentate sia come 13 bande spettrali sia come RGB codificate come immagini JPEG.

Sono state scelte queste classi perché consentono di controllare come cambia l'uso del suolo nel corso del tempo, consentendo di fare scelte più oculate nei campi di: cambiamento climatico, agricoltura, soccorsi dopo una calamità naturale, sviluppo ambientale...

Per questo scopo in questa tesi l'obiettivo primario è stato quello di classificare le diverse immagini a seconda delle 10 classi. Come primo approccio è stato usato il Multi layer Perceptron, dopo di che sette reti neurali convoluzionali, una che è stata allenata solo su questo dataset e sei precedentemente addestrate su di un task standard di classificazione di immagini con 1000 classi. Le diverse reti neurali testate sono le seguenti: ResNet50, ResNet50V2, MobileNet, mobileNetV2, EfficientNetB4, EfficientNetB3.

Le diverse reti sono state addestrate sia con immagini RGB che con immagini spettrali.

Infine sono stati provati alcuni metodi di data augmentation ovvero: flip, rotazione e inversione per le immagini con le tredici bande e anche tinta, contrasto, saturazione e luminosità randomici per quelle RGB.

In conclusione dai risultati si può notare che il primo modello testato, il Multi Layer Perceptron, ha avuto un'accuratezza non molto alta con dati solo RGB, paragonabile a quella di approcci di Machine Learning usati nel paper di presentazione del dataset. Al contrario una volta che sono state usate tutte e tredici le bande l'accuratezza è aumentata fino a intorno al 90 %. Inoltre, senza applicare data augmentation, si è riscontrato il fenomeno dell'overfitting dei dati di allenamento mentre applicando alcune tecniche di data augmentation è stato possibile ridurre questo fenomeno.

Le reti convoluzionali invece hanno avuto risultati migliori, già con dati solo RGB infatti avevano un accuratezza di almeno l'80 %. Quando sono state usate invece le immagini spettrali l'accuratezza è salita fino al 97 % circa. In questo caso i metodi di data augmentation hanno avuto come effetto maggiore un aumento di accuratezza, oltre ad aiutare a combattere l'overfitting.

Capitolo 1

Introduzione

La siccità di quest'anno in Europa è stata la peggiore in 500 anni, ciò ha creato e sta continuando a creare problemi di una scala tale da essere visibili dall'orbita e non è solo un modo di dire poiché la missione dell'ESA Sentinel-2 ha aiutato a stimare la portata di tale evento. Si è potuto, infatti, vedere che regioni abbastanza vaste da essere viste dai satelliti sono passate da verde a marrone.

Le immagini satellitari infatti possono essere molto utili per analizzare il tipo di uso del terreno e come esso varia nel tempo permettendo di studiare tali cambiamenti da una prospettiva diversa.

L'automazione di tale compito attraverso l'intelligenza artificiale è fondamentale per rendere però effettivamente praticabile uno studio approfondito di tali immagini per 2 motivi.

Il primo è che anche se è vero che i cambiamenti più grandi possono essere notati anche ad occhio nudo, un sistema automatizzato può notare cambiamenti nei singoli pixel. Il secondo è che è capace di controllare una quantità di dati molto superiore a quella di un qualunque essere umano e usando anche frequenze dello spettro non visibili.

Gli approcci usati per risolvere problemi di questo tipo (ovvero di estrarre informazioni dalle immagini) sono 2:

Machine Learning, il quale è una branca dell'intelligenza artificiale che si occupa di costruire, analizzare e implementare algoritmi che permettono ai computer di apprendere e adattarsi grazie all'uso di modelli statistici usati per analizzare e trarre inferenze da modelli nei dati a partire da feature individuate da esperti di dominio.

Deep Learning, che è un sottoinsieme del Machine Learning, da come è possibile notare in figura 1.1, è essenzialmente una raffinazione di un classe di algoritmi che tentano di imparare una funzione per separare i dati nelle varie classi. La caratteristica principale del Deep Learning è che in genere richiede meno pre-processamento dei dati, poiché automatizza l'estrazione delle feature.

Il Deep Learning si è sviluppato a partire da un particolare algoritmo di Machine Learning chiamato Perceptron. Il Deep in Deep Learning deriva dal fatto che gli algoritmi sono una serie più o meno profonda di nodi ripetibili di cui il Perceptron è il primo ad esser stato sperimentato. Questo tipo di architettura è chiamata Artificial Neural Network(ANN).

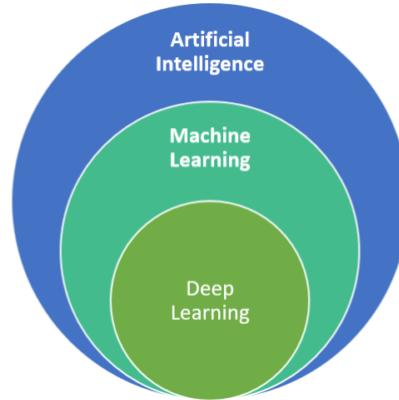


Figura 1.1. Grafico che indica come le varie categorie dell'intelligenza artificiale sono legate

Una classe di algoritmi che si è sviluppata successivamente per la computer vision e altre aree di applicazione, proprio a causa della loro performance, è quella dei Convolutional Neural Network(CNN). La differenza tra una CNN e una ANN è che invece che la moltiplicazione tra matrici usa la convoluzione come operazione, ciò gli consente di estrarre feature dalle immagini in maniera migliore e quindi di imparare meglio.

Le CNN però si è scoperto avere un problema di Overfitting, ovvero tendevano a imparare troppo i casi particolari insiti nei dati usati per allenare la rete e che non erano generali per tutte le immagini della classe. Per ridurre questo problema si sono aggiunti dei nodi che consentono di ridurre il peso di alcuni livelli della rete per ridurre il problema di Overfitting.

Questo ultimo metodo è quello più usato, tanto che nel paper di presentazione del dataset hanno confrontato gli algoritmi più performanti di entrambi i metodi: Support Vector Machine(SVM) e Reti Neurali Convoluzionali(CNN) e hanno osservato che mentre SVM arrivava al massimo al 69% circa le CNN arrivano al 98% di accuratezza.

Nel [Capitolo 2](#) sono spiegati gli algoritmi di Deep Learning che verranno usati in questa tesi.

Nel [Capitolo 3](#) vengono esposti come vengono applicati al problema gli algoritmi e i risultati ottenuti.

Nel [Capitolo 4](#) vengono presentate le conclusioni riassumendo i risultati e spiegando altri possibili esperimenti applicabili a questo problema.

Capitolo 2

Algoritmi di Deep Learning e soluzioni proposte

Per capire cosa sia il Deep Learning bisogna partire dal suo contenitore, ovvero il Machine Learning. Il Machine Learning è una famiglia di algoritmi che riesce ad imparare ad eseguire un task a partire dai dati invece che necessitare una programmazione esplicita.

Alcuni esempi di algoritmi di Machine Learning sono:

- K-Nearest Neighbors(KNN): Algoritmo che predice quale classe ha un nuovo punto in base alla classe di k punti vicini.
- Random Tree: Algoritmo che classifica un nuovo punto in base ad una serie di condizioni sui valori che hanno le feature, riassumibili in un albero.
- Support Vector Machine(SVM): Algoritmo che classifica i nuovi punti in base a quale lato di una linea sono, cercando di mettere la linea il più distante possibile dai punti per avere il margine più grande possibile.
- Perceptron: algoritmo che cerca di dividere con una funzione lineare i punti all'interno del dataset.

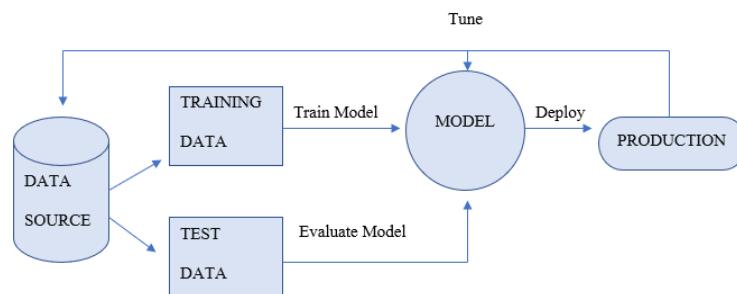


Figura 2.1. Uno schema riassuntivo di ciò che avviene nel Machine Learning

In generale il processo per allenare un algoritmo di Machine Learning parte dai dati; questi vengono divisi in dati usati per l'allenamento e dati usati per verificare l'accuratezza del modello. Per ogni elemento del dataset si modificano i parametri

definiti dell'algoritmo per cercare di migliorarne la performance e se necessario si ripete l'allenamento fino al raggiungimento del risultato migliore possibile.

Per capire in che modo cambiare i parametri si usa un sistema chiamato Discesa del Gradiente. Il suo nome deriva dal fatto che si applica ai casi in cui si può calcolare il gradiente dei parametri rispetto ad una funzione che rappresenta la distanza della predizione attuale rispetto alla risposta corretta chiamata loss. Una volta calcolato il gradiente si usa come bussola per trovare il punto in cui la loss è al valore più basso.

Il problema è che il gradiente indica il verso opposto, quindi si prende l'opposto del gradiente. Inoltre se si prendesse semplicemente il gradiente si potrebbe rischiare o di avvicinarsi troppo lentamente al punto o di oltrepassarlo, quindi lo si scala di un fattore chiamato Learning Rate.

2.1 Perceptron

Il Perceptron è un algoritmo di Machine Learning capace di classificare solo in maniera lineare e in contesto supervisionato.

Per capire come funziona un Perceptron è utile ricordare che tipo di problema riesce a risolvere ovvero la classificazione supervisionata.

I problemi di classificazione supervisionata sono definiti come la classe di problemi nei quali viene data una serie di osservazioni con una classe associata e viene richiesto di saper associare una nuova osservazione alla classe corretta.

I Perceptron risolvono questo problema moltiplicando ogni feature in input con dei pesi, sommando i vari risultati e aggiungendo un altro parametro chiamato Bias, alla fine decide se quei valori delle feature corrispondono alla classe se il risultato è maggiore o uguale di zero. Questa mappatura viene chiamata funzione di attivazione.

I valori dei pesi e del Bias vengono inizializzati randomicamente e successivamente raffinati nella fase di allenamento. Per farlo viene fatta una prova di predizione prendendo ad ogni passo la derivata delle variabili allenabili. Se la predizione è corretta non c'è niente da cambiare e passo all'elemento successivo. Se invece non corrisponde a quella di riferimento si ripercorre il Perceptron al contrario individuando la variabile con derivata più alta e aggiungendo o togliendo il valore della derivata in base al tipo di errore di falso positivo o negativo.

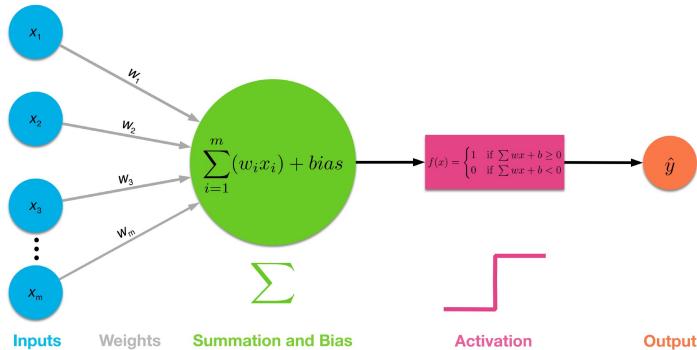


Figura 2.2. Uno schema di come funziona un Perceptron

Il Multi Layer Perceptron invece è l'algoritmo di Deep Learning più semplice, e risolve sempre il problema della classificazione supervisionata, ma riesce ad imparare anche pattern non lineari.

I motivi per cui riesce a fare ciò sono due:

- Collega l'output di un Perceptron con l'input del successivo in un architettura a più livelli.
- La funzione di attivazione finale non è lineare.

Con queste due modifiche è stato mostrato che quest'algoritmo è un approssimatore universale di funzioni. Questo consente agli algoritmi di Deep Learning di poter teoricamente imparare una linea di separazione qualunque tra le varie classi. Questo è vero se la larghezza della rete può tendere ad infinito cosa che non è attuabile.

Un concetto nuovo che ha introdotto Multi Layer Perceptron è nel nome stesso ed è l'aggiunta di almeno un layer di Perceptron tra l'input e l'output. In questo modo l'output dei Perceptron non nell'ultimo layer non è direttamente la classe predetta come nel caso del Perceptron semplice, ma la proiezione dei dati dallo spazio di partenza ad un altro spazio in base alla dimensione del layer inserito.

Il layer che prende i dati viene chiamato input layer, mentre i layer interposti vengono chiamati hidden layer, infine il layer che ritorna la classe di appartenenza predetta è il layer di output.

La funzione di attivazione è un concetto presente anche nel Perceptron "standard" ma in quel caso è necessariamente quella descritta nel capitolo precedente. Nel caso del Multi Layer Perceptron invece vi è molta scelta tra le funzioni di attivazione perché vengono usate per scopi diversi.

Principalmente si dividono tra quelle che vengono usate nei layer hidden e quelle usate nel layer di output. Quelle usate nei layer hidden servono per dividere il più possibile i punti che rappresentano le diverse classi in modo da rendere più facile per l'output layer separarli. Quest'ultimo cerca, oltre a separare il più possibile i punti come i layer precedenti, di stabilire la probabilità che appartenga a quella classe. Alcuni esempi di queste funzioni sono in figura 2.4

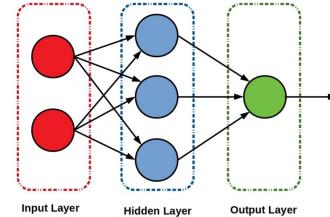


Figura 2.3. Uno schema di come può essere un Multi Layer Perceptron

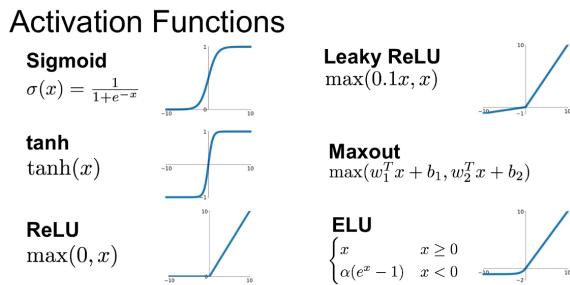


Figura 2.4. Alcune funzioni di attivazione comuni

2.2 Convolutional Neural Network

Uno delle classi di problemi su cui sono state frequentemente usate le reti neurali è la classificazione di immagini. Hanno avuto un discreto successo pur avendo il problema che i dati in input hanno una dimensionalità molto più alta di altre classi di problemi. L'alta dimensionalità dei dati pone la questione che il volume aumenta più velocemente dell'area all'aumentare del numero di dimensioni; questo fatto fa sì che i dati siano più distanti tra di loro e che quindi sia più difficile trovare una funzione di separazione. Per ridurre questo problema si è pensato di usare una operazione molto usata nella manipolazione di immagini: la convoluzione.

La convoluzione è un'operazione che in generale coinvolge 2 liste di numeri, prende la seconda, ne inverte l'ordine e la fa scorrere sulla prima un elemento alla volta prendendo la media pesata degli elementi sovrapposti usando la seconda lista come pesi.

Quando viene applicata tra 2 matrici si sovrappone la seconda (di solito più piccola) alla prima facendola scorrere su ogni elemento calcolando la media pesata degli elementi sovrapposti usando gli elementi della seconda matrice come pesi (vedi figura 2.5).



Figura 2.5. un esempio di applicazione di matrice convoluzionale ad un immagine

La convoluzione viene applicata nelle reti neurali in due tipi di layer:

- I Layer convoluzionali sono i layer in cui viene imparata la feature map ovvero la seconda matrice che viene fatta scorrere sopra la prima. In questo tipo di layer ci sono vari parametri che possono essere modificati come il numero di matrici che vengono create, la dimensione delle matrici e se devono passare su tutti gli elementi o se saltarne alcuni.
- I layer di Pooling invece sono dei layer in cui viene ridotta la dimensionalità del problema. Ci sono due metodi standard per farlo: max e average. Nel caso del max Pooling si prende il massimo valore all'interno di una finestra grande quanto si vuole (definibile tramite un parametro) e si prende il valore massimo. Nel caso dell'average Pooling invece si prende un valore medio sempre all'interno di una finestra come nel caso precedente.

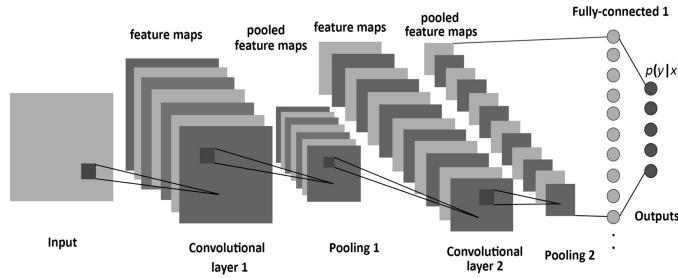


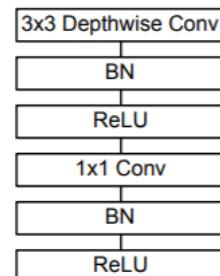
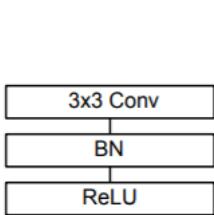
Figura 2.6. un esempio di architettura di una CNN

2.2.1 MobileNet

Uno dei limiti delle reti neurali che nella pratica non ci consentono di considerarle approssimatori universali è la potenza di calcolo finita che abbiamo a disposizione poiché il numero di parametri da ottimizzare sarebbe troppo elevato per gli attuali sistemi computazionali. Le reti neurali convoluzionali sono computazionalmente anche più costose perché i parametri da ottimizzare aumentano come anche il numero di scelte possibili quando si implementa un architettura.

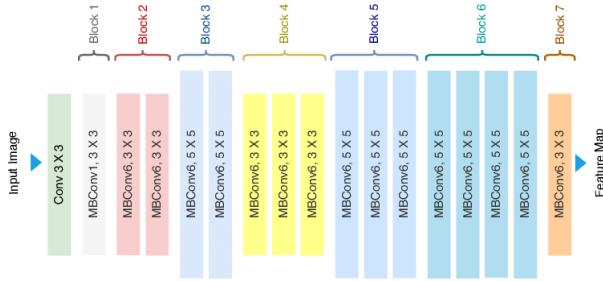
Per cercare di limitare la complessità della computazione senza compromettere troppo l'accuratezza, MobileNet ha il layer di convoluzione diviso in due; quindi invece di applicare la convoluzione sull'immagine pixel per pixel la si applica sui vari canali e poi li convolve tra di loro, riducendo la complessità di computazione nell'ordine di otto/nove volte .

MobileNet ha due tipi di blocchi, il primo tipo che viene usato solo come input poichè è formato da un semplice livello convoluzionale seguito da un livello di normalizzazione(BN) e uno non lineare(ReLU). Il secondo tipo, che è ripetuto fino al layer di output, applica la divisione di cui sopra, quindi applica una convoluzione sui canali, poi una normalizzazione, una non linearità e infine una convoluzione puntuale anch'essa seguita da normalizzazione e non linearità.



2.2.2 EfficientNet

EfficientNet cerca di risolvere lo stesso problema di MobileNet, ovvero quello della quello della limitatezza della risorse di computazione, guardando il problema da un'altra prospettiva.

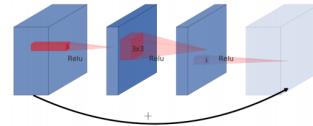
**Figura 2.8.** l'architettura di EfficientNet

EfficientNet usa sempre dei blocchi di convoluzione standard, mantenendo quindi l'accuratezza di questo approccio, ma bilancia in modo intelligente le dimensioni della rete in base alle caratteristiche del problema.

2.2.3 ResNet50

ResNet, al contrario delle due architetture precedenti, non cerca di essere la più efficiente, bensì la più accurata possibile. In particolare cerca di impedire al gradiente usato per ottimizzare le reti di diventare o troppo piccolo o troppo grande, e per farlo implementa un nuovo tipo di layer chiamato layer residuale.

Poiché il problema del gradiente che o esplode o si azzera è di solito causato da un'abbondanza o una mancanza di parametri, il Layer Residuale cerca di imparare quali siano i parametri importanti e se ce ne sono in eccesso, consentendo ai dati di saltare alcuni layer in caso di necessità come si può vedere in figura 2.9. Usare i Layer Residuali consente di creare reti che si adattano meglio al problema perché l'algoritmo può decidere in autonomia quanti dei layer siano effettivamente significativi per risolvere il problema. Un esempio di un architettura di questo tipo è in figura 2.10

**Figura 2.9.** un esempio di architettura di un layer residuale

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

Figura 2.10. un esempio di architettura di una CNN

2.3 Soluzioni proposte

Per risolvere il problema di classificare le immagini nelle dieci classi sono state sperimentate sia alcune reti standard, di cui è stata spiegata l'architettura alla fine della sezione precedente, sia due reti ad hoc sulla cui architettura sono stati testati anche altri parametri.

Le reti ad hoc hanno consentito anche un altro confronto viste le caratteristiche del dataset, ovvero poiché che le reti standard accettano solo immagini RGB nel loro layer di input hanno potuto usare solo quelle bande, invece le reti ad hoc hanno potuto usare tutte le bande spettrali.

la Pipeline di allenamento che è stata usata parte dal caricamento del dataset e la sua divisione in tre parti:

- 80% usato per l'allenamento vero e proprio.
- 10% usato per validazione durante l'allenamento.
- infine un ultimo 10% per test alla fine dell'allenamento.

Successivamente vengono normalizzate le immagini e, se necessario, selezionate le bande RGB. Dopo di ciò viene creata una variabile per tenere il dataset aumentato dalla data augmentation se utile per l'allenamento della rete in questione.

A seguire viene creato il modello e fatto allenare. Una volta finito l'allenamento viene creata la reportistica per controllare il risultato, in particolare viene generato il grafico della loss e dell'accuratezza durante le epoch (cicli sul dataset) e la Confusion Matrix con le predizioni fatte sulla parte di test del dataset.

2.3.1 Multi Layer Perceptron

Per prima cosa viene provato un Multi Layer Perceptron per vedere come il più semplice algoritmo di Deep Learning si comportasse per poi confrontarlo sia all'approccio di Machine Learning con Support Vector Machine che ad approcci basati su reti convoluzionali.

Con i Multi Layer Perceptron, inoltre, vengono fatti altri esperimenti:

- Profondità.
- Layer con un numero di nodi non multiplo di 2.
- RGB oppure tutti e tredici gli spettri.

Per quanto riguarda la profondità vengono provate sia reti di profondità diversa, aggiungendo layer con più nodi all'inizio o con meno alla fine e con una quantità di nodi che raggiunge un minimo al centro e poi riaumenta.

Poiché per convenzione nei layer si usano una quantità di nodi multipli di due, si vuole verificare se questa convenzione ha un qualche motivo insito nel fatto che i dati sono in base due o che l'hardware sia pensato per operazioni in base due oppure se anche i layer con nodi in quantità multipla di dieci funzionano allo stesso modo.

Visto che il dataset mette a disposizione anche altre dieci bande, oltre a quelle del visibile, si vuole controllare quanto sia più informativo avere tutte le bande spettrali a disposizione rispetto a solo quelle visibili.

2.3.2 Rete convoluzionale

Successivamente vengono provate varie reti convoluzionali, La prima delle quali è quella ad hoc. Su questa rete è stato provato principalmente la data augmentation poiché per la parte densa della rete è stata usata la stessa architettura usata precedentemente.

Sulle reti di tipo MobileNet è stato sperimentato l'uso del layer di dropout per regolarizzare il problema poiché altrimenti si aveva un problema di overfitting molto marcato, infatti si è dovuto impostare un dropout del 0.8 per avere risultati buoni.

Sulle reti di tipo EfficientNet invece ci si è concentrati sul learning rate poichè la rete è particolarmente sensibile a questo parametro su questo dataset, infatti il range su cui si è scoperto lavorare bene è intorno a 1e-7, valori più bassi o più alti di molto non riescono ad abbassare l'errore.

Anche sulle reti ResNet si è lavorato sul dropout più per una questione di fine tuning dei parametri piuttosto che di scarsa performance.

Capitolo 3

Risultati

3.1 Dataset e Dettagli Implementativi

Il dataset che è stato scelto ha delle caratteristiche peculiari a causa della sua provenienza. Innanzitutto essendo una raccolta di immagini satellitari non sono semplici immagini RGB, bensì immagini con ben tredici bande dello spettro elettromagnetico. Oltre alle bande del visibile RGB ovvero rosso, verde e blu ci sono altre dieci bande non visibili che hanno caratteristiche diverse.

Le varie bande non hanno tutte la stessa risoluzione, quelle che hanno la risoluzione più grande sono quelle visibili e quella nella parte dell'infrarosso vicino con 10m, seguite dalle altre bande dell'infrarosso con 20m e infine quelle atte al controllo dell'atmosfera con 60m.

Band	[h]	
	Spatial Resolution m	Central Wavelength nm
B01 - Aerosols	60	443
B02 - Blue	10	490
B03 - Green	10	560
B04 - Red	10	665
B05 - Red edge 1	20	705
B06 - Red edge 2	20	740
B07 - Red edge 3	20	783
B08 - NIR	10	842
B08A - Red edge 4	20	865
B09 - Water vapor	60	945
B10 - Cirrus	60	1375
B11 - SWIR 1	20	1610
B12 - SWIR 2	20	2190

Figura 3.1. Tabella con le varie bande con la loro risoluzione e ampiezza.

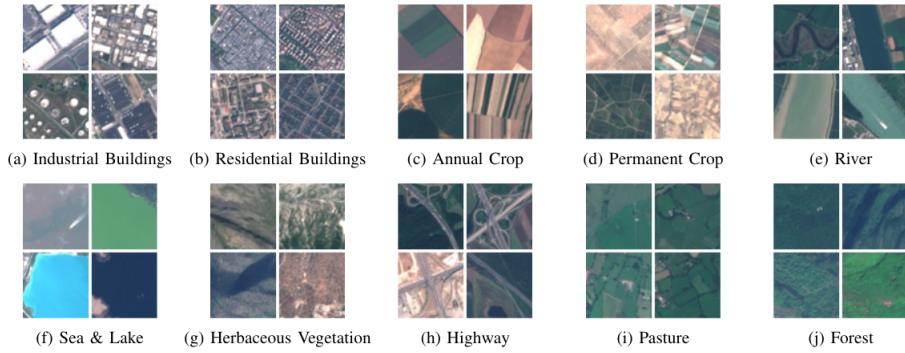


Figura 3.2. Immagini rappresentative delle 10 classi.

Il dataset in particolare contiene 27000 immagini abbastanza bilanciate nelle dieci classi(Industrial Buildings, Residential Buildings, Annual Crop, Permanent Crop, River, Sea & Lake, Herbaceous Vegetation, Highway, Pasture, Forest) con circa due/tre mila immagini per classe di dimensione 64x64 pixels.

La funzione di loss che è stata usata si chiama Cross Entropy e misura la distanza che c'è tra la distribuzione predetta e quella vera dei dati sommando la probabilità della distribuzione dei dati con il logaritmo della probabilità della predizione della rete.

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x)$$

Figura 3.3. Formula della funzione di loss Cross Entropy

l'ottimizzatore che è stato usato si chiama Adam ed è un acronimo per Adaptive Moment Estimation, infatti usa il momento del gradiente per essere più efficiente. Adam non segue semplicemente il gradiente in quel punto ma si ricorda del gradiente nel punti precedenti e si fa influenzare anche da quello. Per non farsi influenzare troppo però applica ai gradienti precedenti una funzione di decrescita esponenziale. Inoltre per adattarsi meglio al gradiente nella direzione più importante aggiunge anche una media del quadrato dei gradienti precedenti applicandogli una funzione di decrescita esponenziale .

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[\frac{\delta L}{\delta w_t} \right] v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\delta L}{\delta w_t} \right]^2$$

Figura 3.4. Formula dell'ottimizzatore Adam

Inoltre i dati vengono divisi in tre parti: 80% per il training, ovvero per l'allenamento vero e proprio, 10% per la validation, ovvero un test fatto ad ogni epoca per controllare quanto stia generalizzando il modello e 10% per testare alla fine se il modello generalizza anche su dati mai visti.

Infine vengono provati alcuni metodi di data augmentation ovvero di aumentare la immagine a disposizione del modello nella fase di training modificando quello che già sono presenti nel dataset. In particolare capovolgo, ruoto e inverteo tutte le immagini, in più su quelle RGB posso operare anche sui colori cambiando tinta, saturazione, illuminazione e contrasto.

3.2 Multi Layer Perceptron

Dai risultati di una rete poco profonda, con solo due livelli, su immagini solo RGB (256,128) si può vedere che l'accuratezza è scarsa, anche al di sotto di SVM.

Test Accuracy: 59.56%

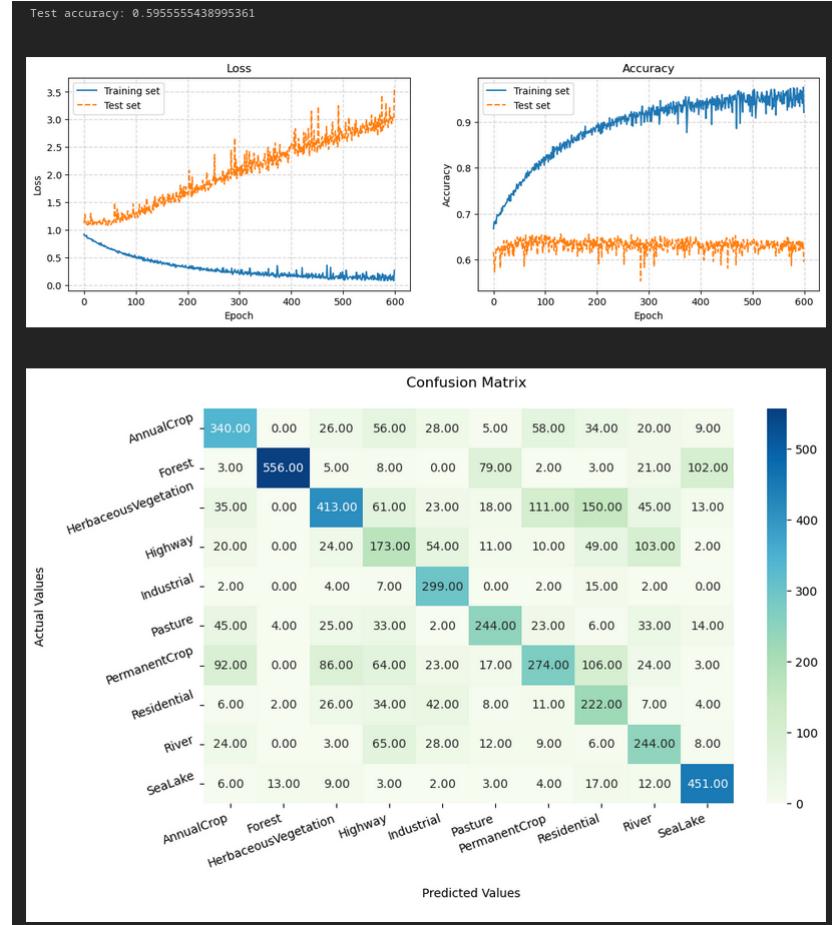


Figura 3.5. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

Dai risultati di una rete più profonda, con cinque livelli, su immagini RGB (1024,512,256,512,1024) si evince che il problema abbia bisogno di più parametri poiché la loss diminuisce considerevolmente.

Test Accuracy: 63.50%

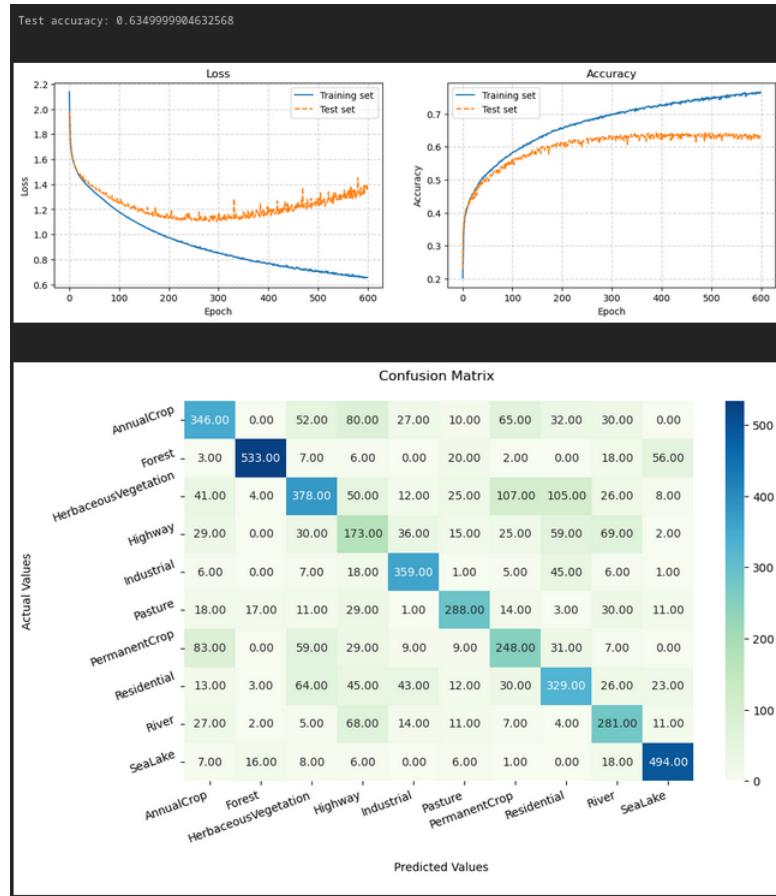


Figura 3.6. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

Dai grafici di una rete sempre a cinque livelli, ma con un numero di nodi sempre decrescente (1024,512,256,128,64) si può notare che è importante cercare di portare il problema più vicino alla dimensione della classificazione infatti l'accuratezza raggiunge quasi il 70% necessario per battere SVM.

Test Accuracy: 68.53%

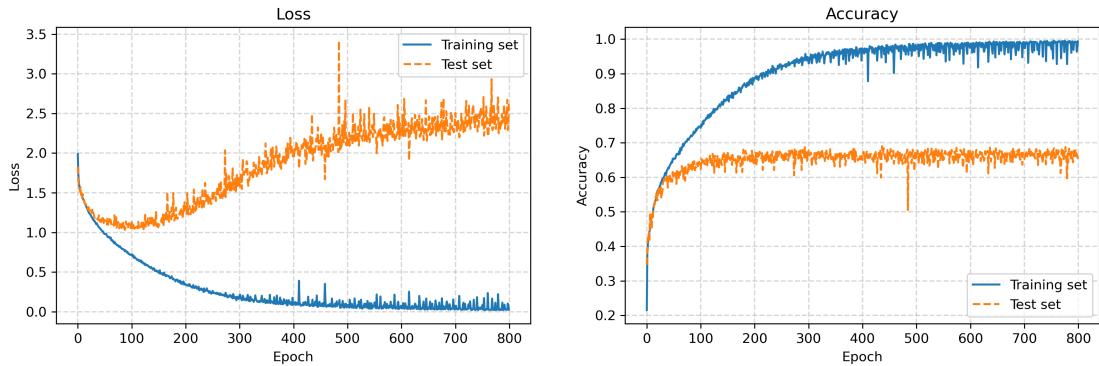


Figura 3.7. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

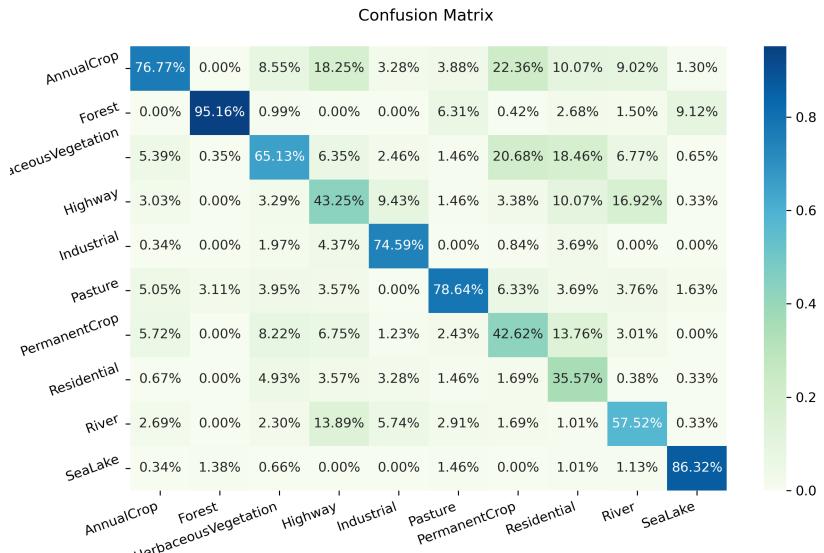


Figura 3.8. Confusion Matrix tra le varie classi.

Dai grafici invece della stessa rete di prima ma con data augmentation applicata possiamo notare l'impatto che ha avuto già così i risultati sono migliori di SVM.

Test Accuracy: 71.26%

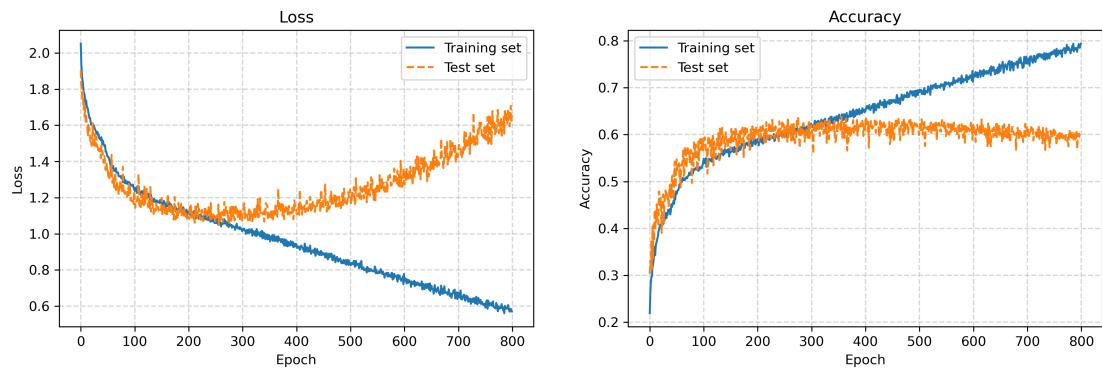


Figura 3.9. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

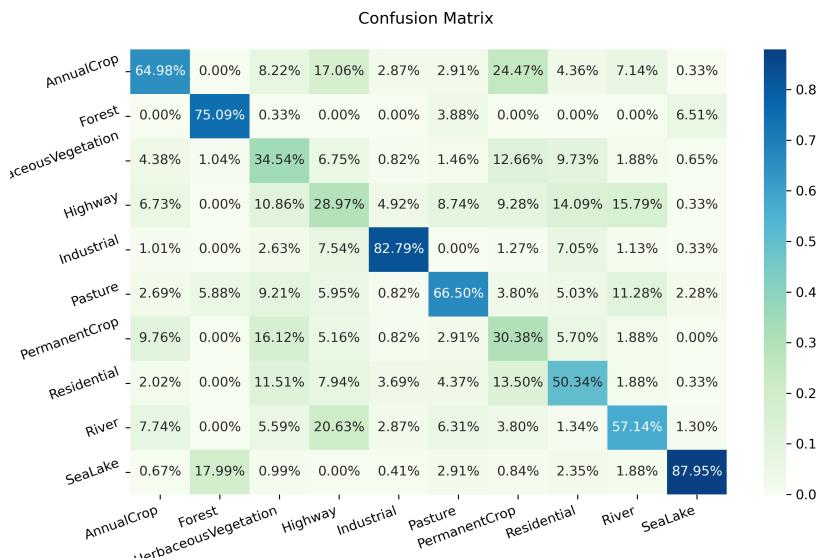


Figura 3.10. Confusion Matrix tra le varie classi.

Invece dai risultati avuti quando si usano tutti e dodici le bande si può vedere l'accuratezza che aumenta drasticamente, anche se si arriva ad un caso di overfitting molto pronunciato, perché l'accuratezza sui dati di allenamento raggiunge quasi il 100% essendo quindi molto più alta di quella di validation

Test Accuracy: 83.72%

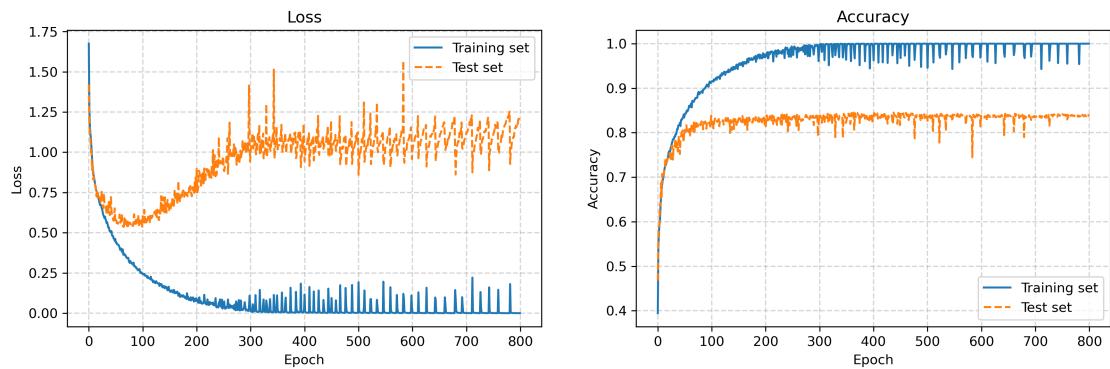


Figura 3.11. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

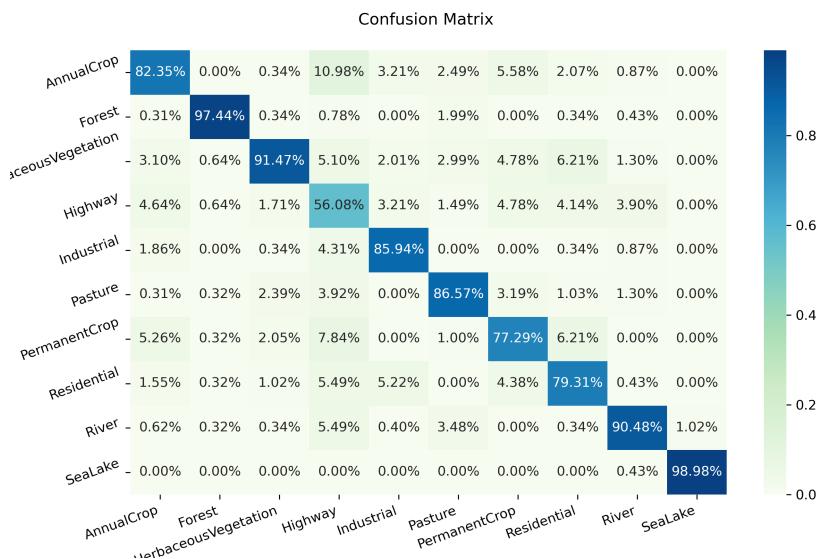


Figura 3.12. Confusion Matrix tra le varie classi.

Applicando data augmentation invece si ottiene un risultato comparabile in questo caso ma senza overfitting.

Test Accuracy: 83.86%

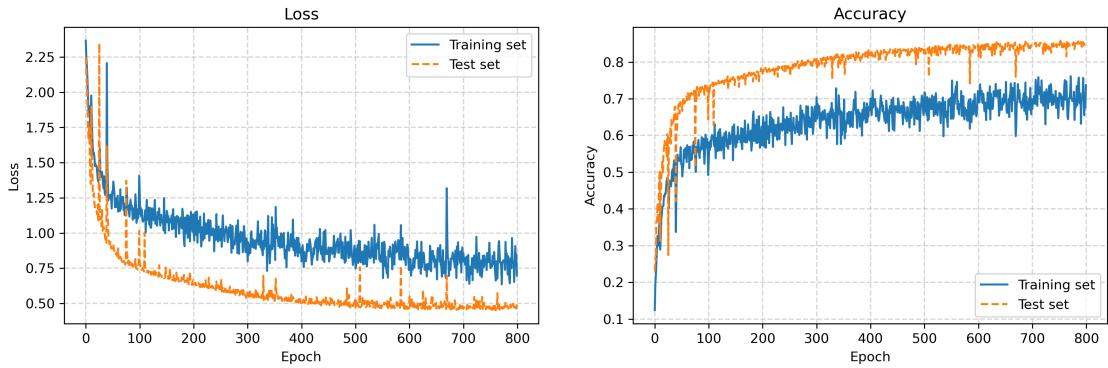


Figura 3.13. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

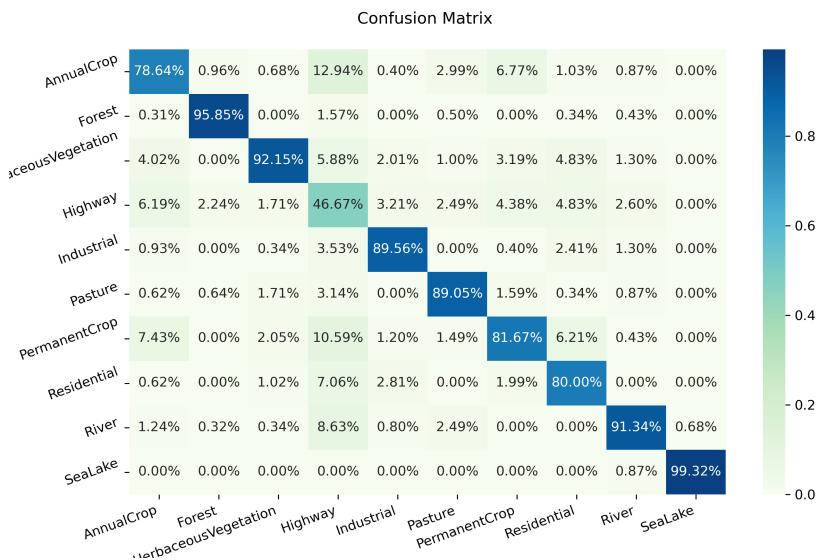


Figura 3.14. Confusion Matrix tra le varie classi.

Questi invece sono risultati di una rete con nodi in base due da confrontare con la prossima con nodi in base dieci.

Test Accuracy: 86.07%

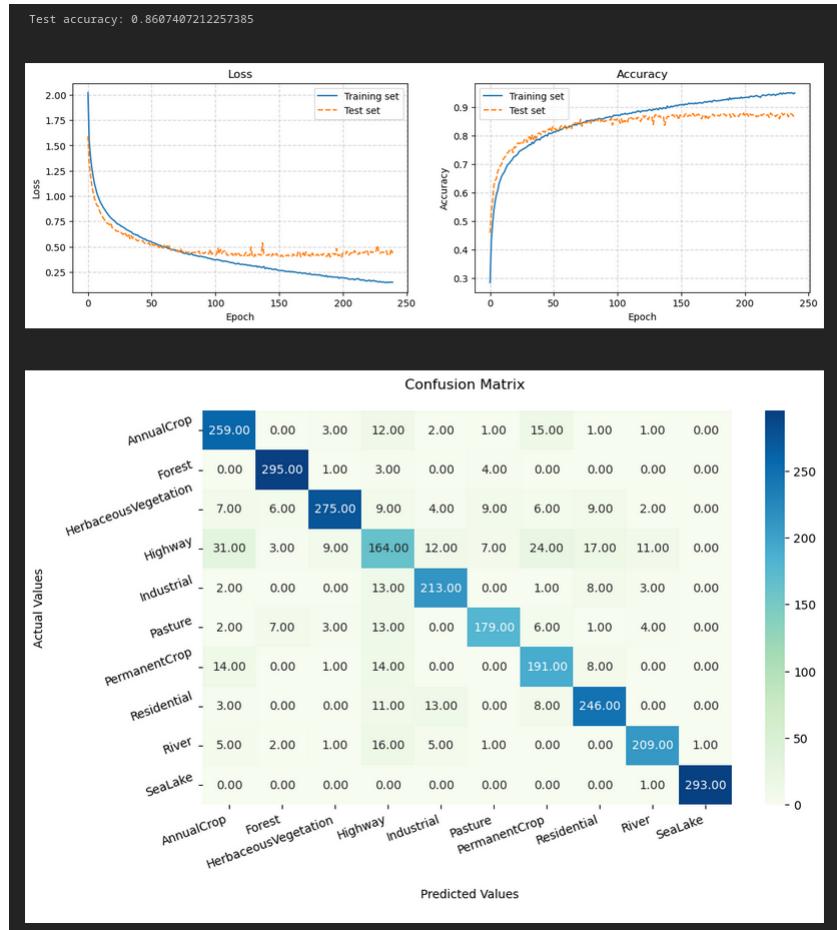


Figura 3.15. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

Questi sono risultati di una rete con nodi in base dieci

Test Accuracy: 89.04%

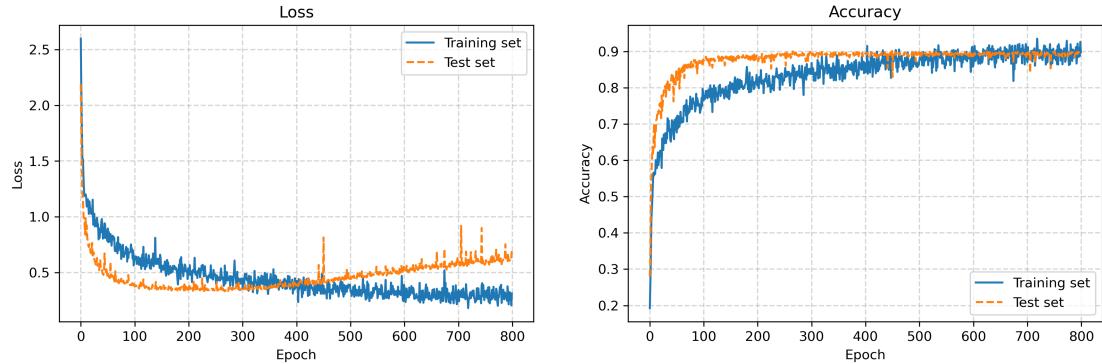


Figura 3.16. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

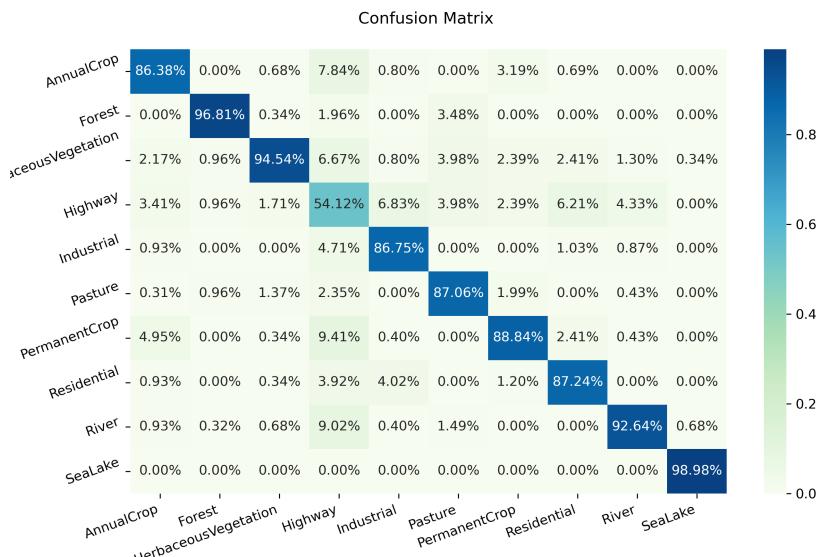


Figura 3.17. Confusion Matrix tra le varie classi.

3.3 Rete convoluzionale

Dai primi risultati su una rete convoluta senza data augmentation su dati RGB si può notare che supera in accuratezza la rete densa nella stessa situazione.

Test Accuracy: 68.76%

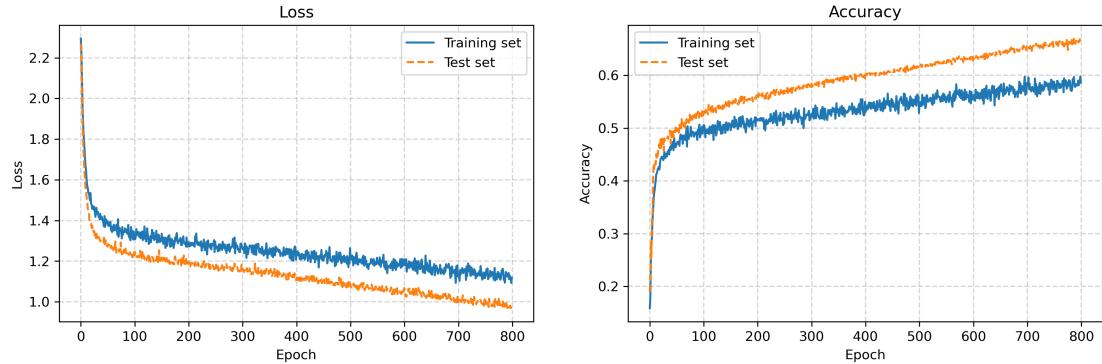


Figura 3.18. Grafici su come si evolve la loss e l'accuratezza nelle varie epocha.

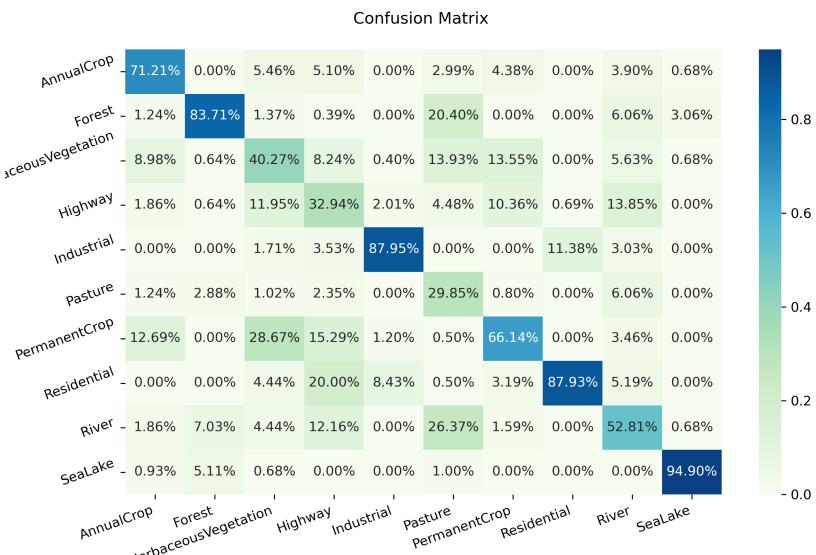


Figura 3.19. Grafici su come si evolve la loss e l'accuratezza nelle varie epocha.

I risultati dopo aver applicato data augmentation invece sono decisamente migliori, si raggiunge un accuratezza di oltre il 90%! Ben superiore ad SVM.

Test Accuracy: 90.11%

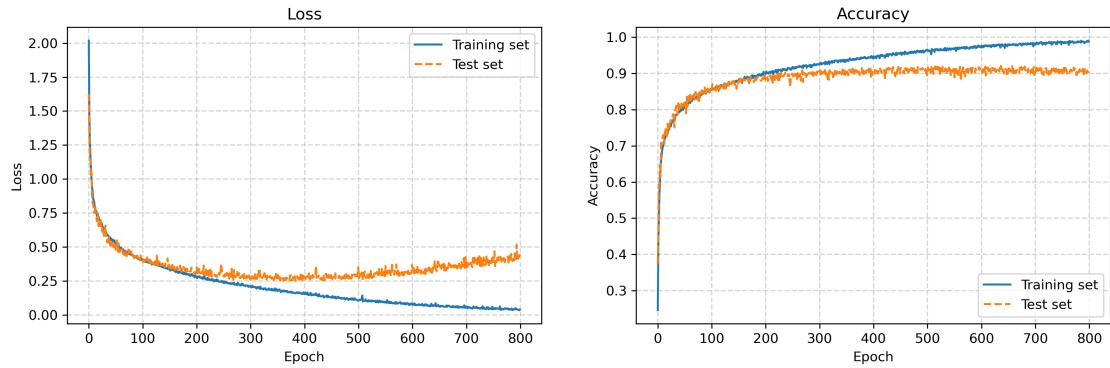


Figura 3.20. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

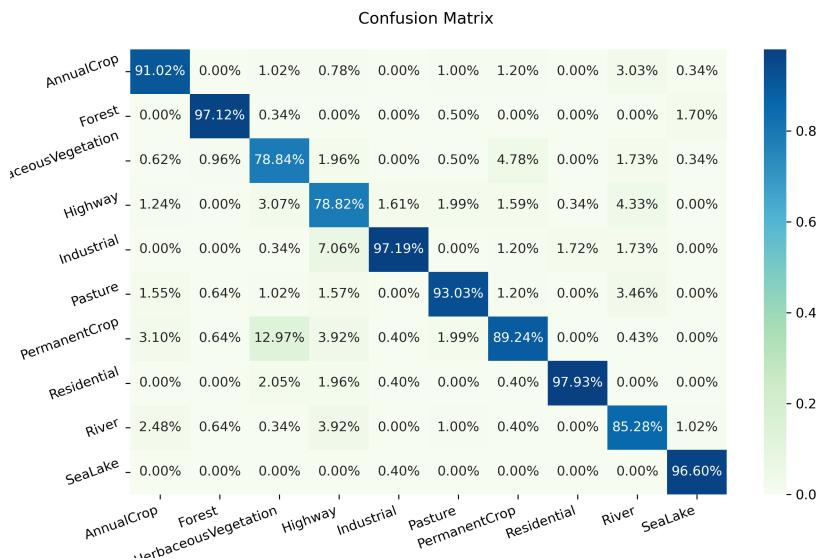


Figura 3.21. Confusion Matrix tra le varie classi.

I risultati invece sono i risultati di una rete convoluzionale applicata a tutti e dodici le bande senza data augmentation sono ancora un leggermente meglio di quelli RGB con data augmentation.

Test Accuracy: 92.06%

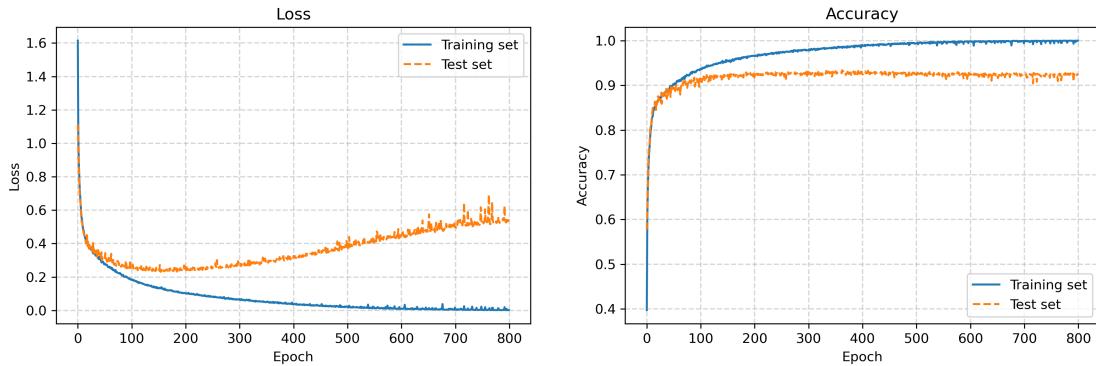


Figura 3.22. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

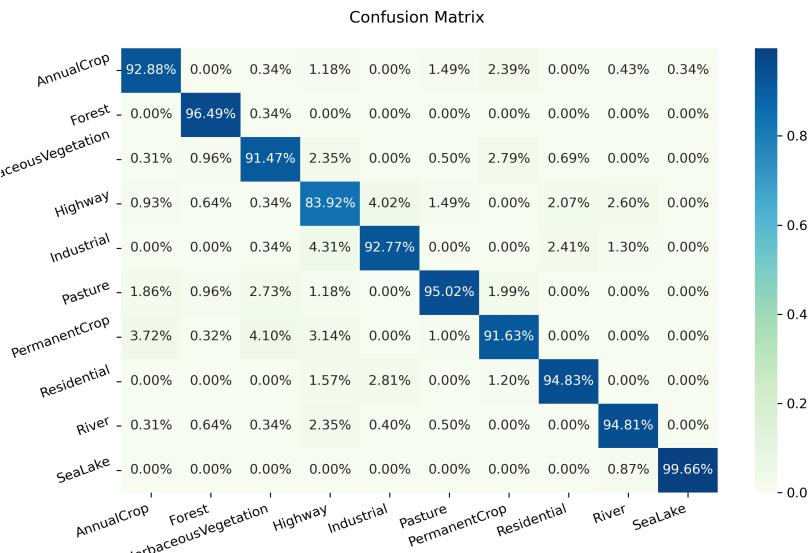


Figura 3.23. Confusion Matrix tra le varie classi.

Dai grafici dei risultati di una rete convoluzionale applicata a tutti e dodici le bande con data augmentation si può vedere che si è arrivati ad un livello di accuratezza molto elevato.

Test Accuracy: 95.08%

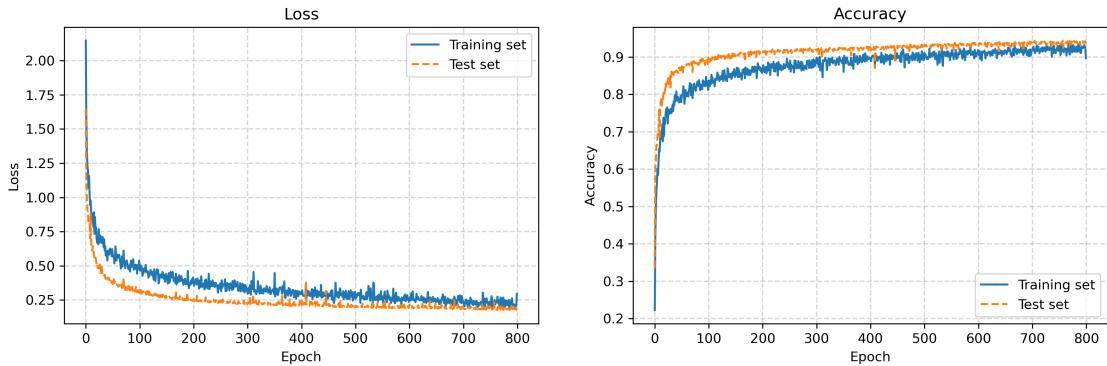


Figura 3.24. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

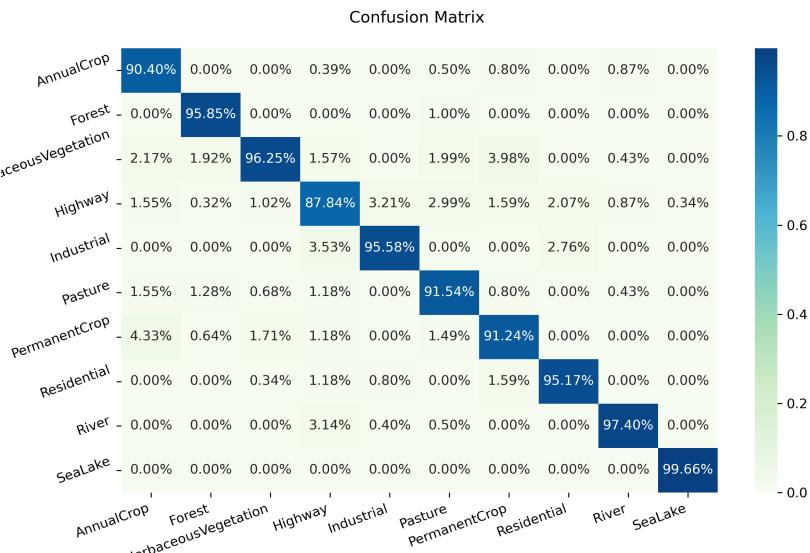


Figura 3.25. Confusion Matrix tra le varie classi.

Da questo grafico invece si possono notare i problemi che si possono avere quando si hanno funzioni di data augmentation troppo aggressive, ovvero che modificano troppo i dati, ovvero che l'allenamento della rete diventa molto meno prevedibile.

Test Accuracy: 95.07%

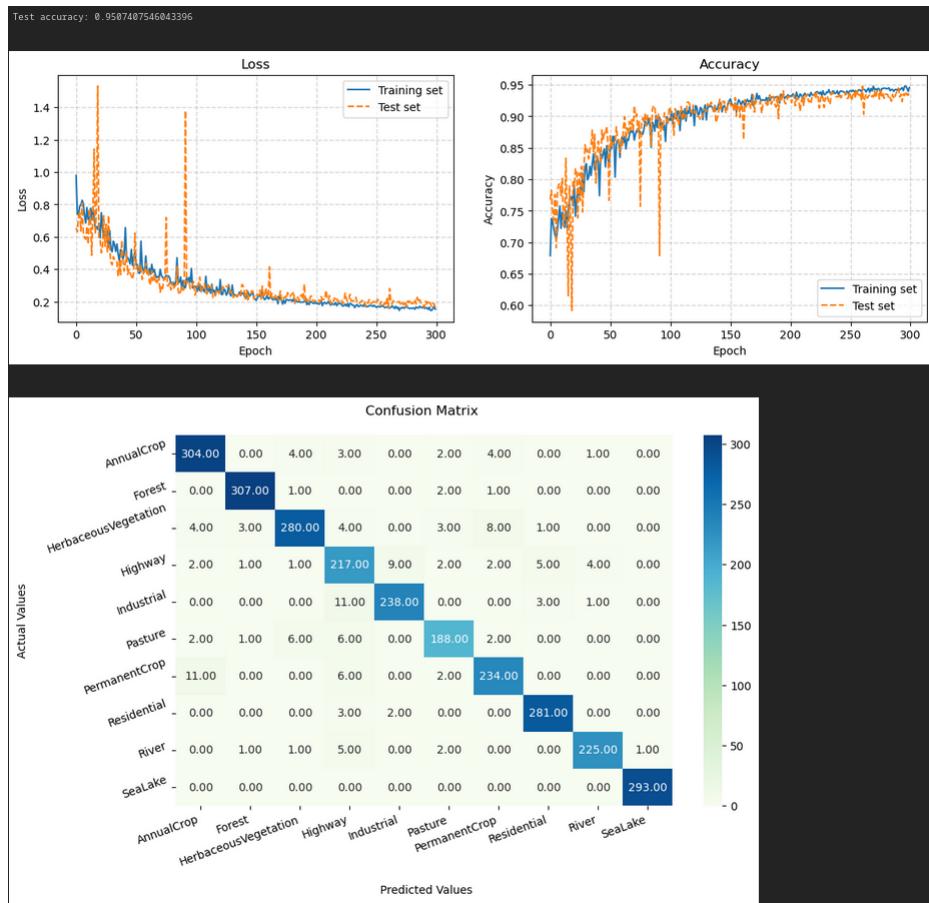


Figura 3.26. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

EfficientB4

All'inizio ci sono stati problemi con il learning rate troppo basso come si può notare da questi grafici ottenuti impostando il learning rate a 1e-8.

Test Accuracy: 22.22%

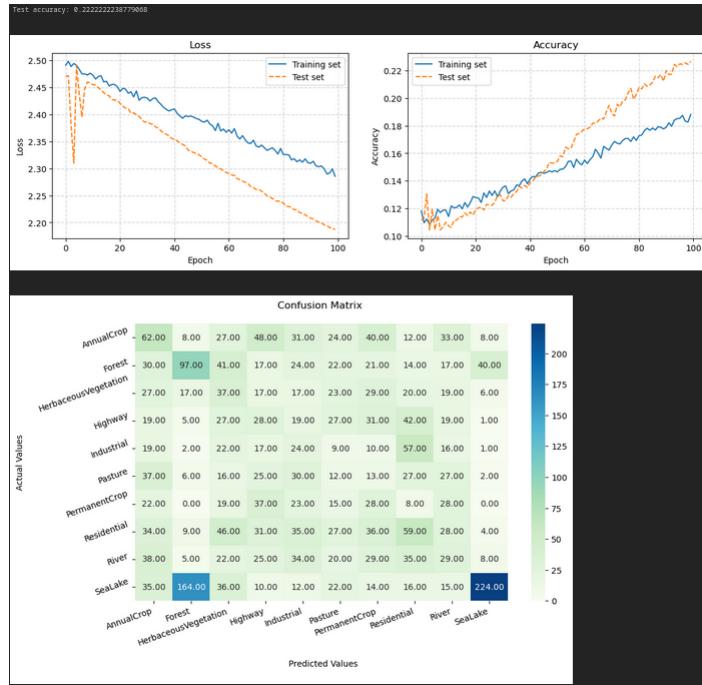


Figura 3.27. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

Dai grafici con learning rate a 1e-6 si può notare che sicuramente è un valore più accettabile del precedente ma ancora non è corretto a causa dei salti troppo eccessivi dell'accuratezza e della loss sulla divisione di test.

Test Accuracy: 70.37%

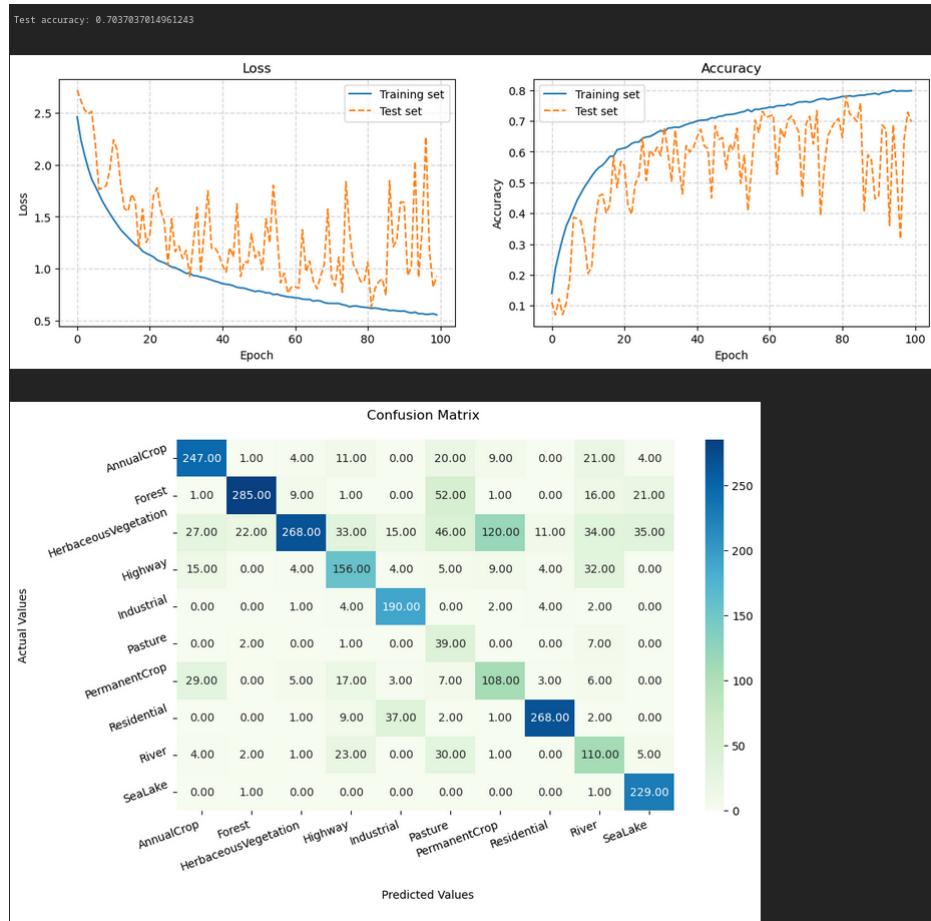


Figura 3.28. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

Dai risultati di EfficientB4 con augmentation.

Test Accuracy: 50.08%

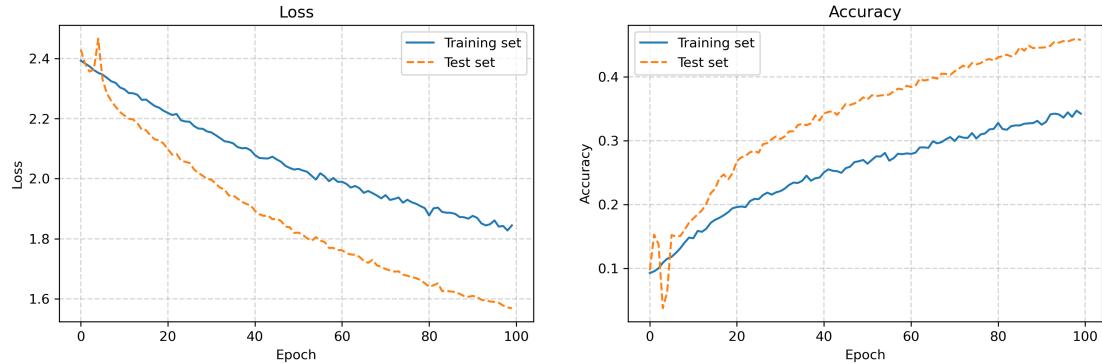


Figura 3.29. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

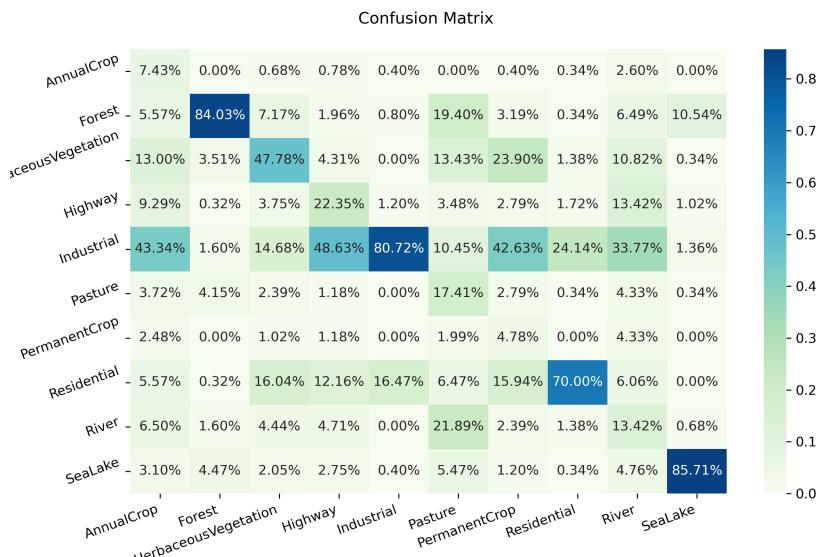


Figura 3.30. Confusion Matrix tra le varie classi.

EfficientB3

dai risultati di EfficientB3 senza augmentation si può evincere che anche la versione B3 con meno parametri non si comporta benissimo su questo dataset.

Test Accuracy: 71.26

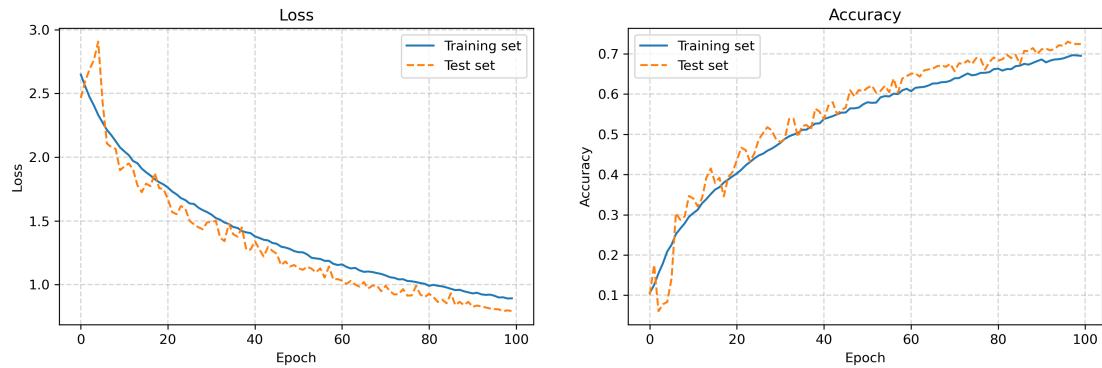


Figura 3.31. Grafici su come si evolve la loss e l'accuratezza nelle varie epocha.

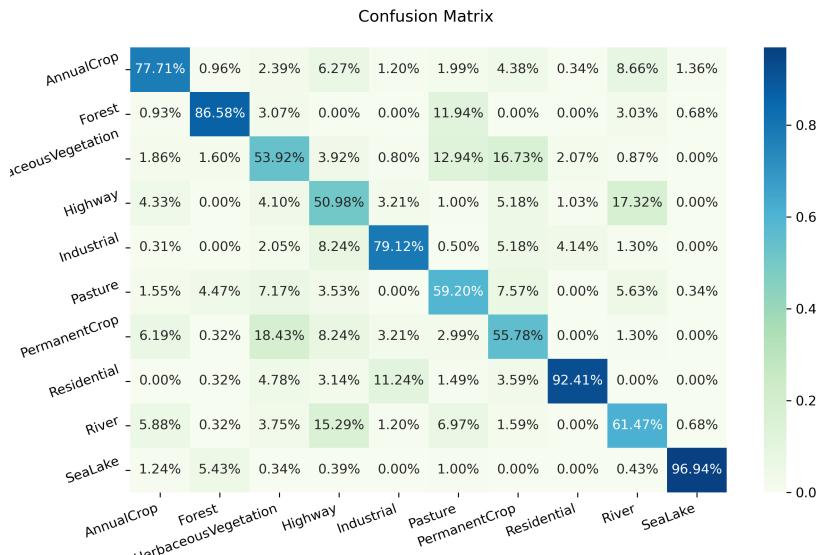


Figura 3.32. Confusion Matrix tra le varie classi.

Dai risultati di EfficientB3 con augmentation si può capire che anche aggiungendo data augmentation la situazione non migliora di molto.

Test Accuracy: 67.92%

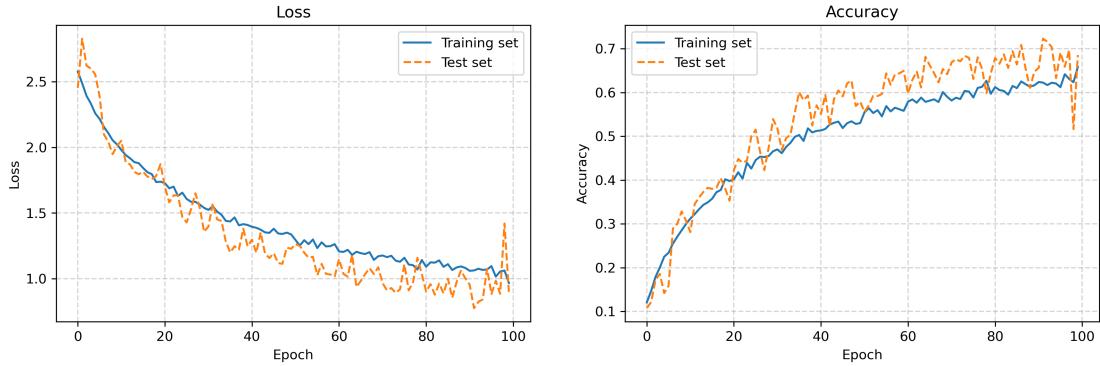


Figura 3.33. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

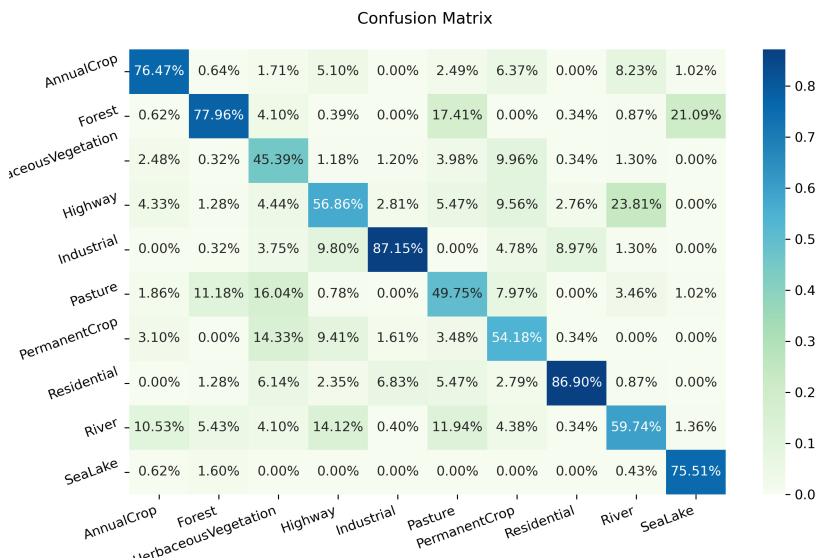


Figura 3.34. Confusion Matrix tra le varie classi.

MobileNet

MobileNet all'inizio ha dato problemi con il dropout, ovvero eccedeva nell'overfitting dei dati di allenamento questi sono i risultati con dropout a 0.5.

Test Accuracy: 95.37%

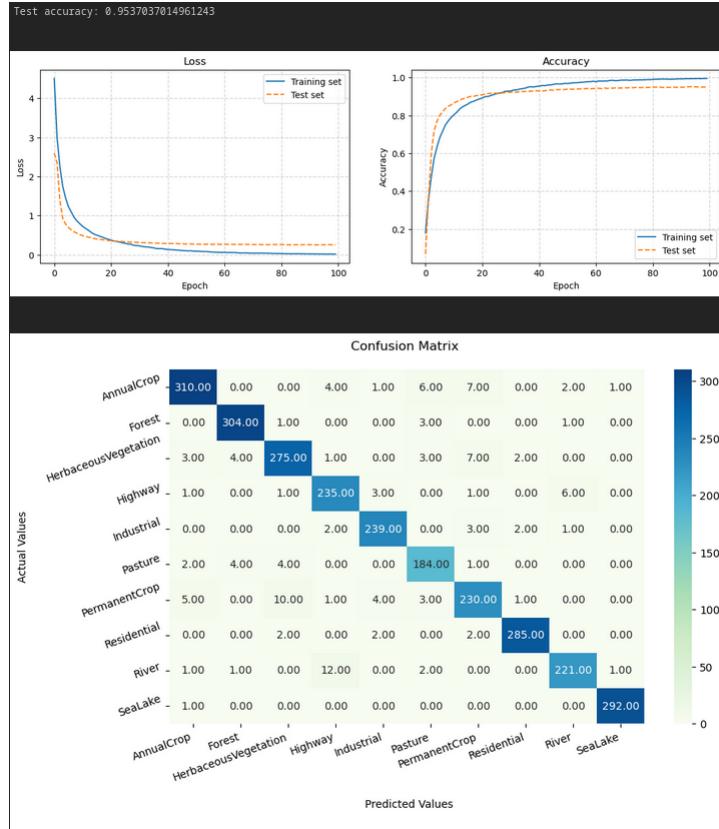


Figura 3.35. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

dai risultati con dropout a 0.75 si può capire che è il problema è ancora l'overfitting e che quindi la direzione è quella di aggiungere ancora dropout

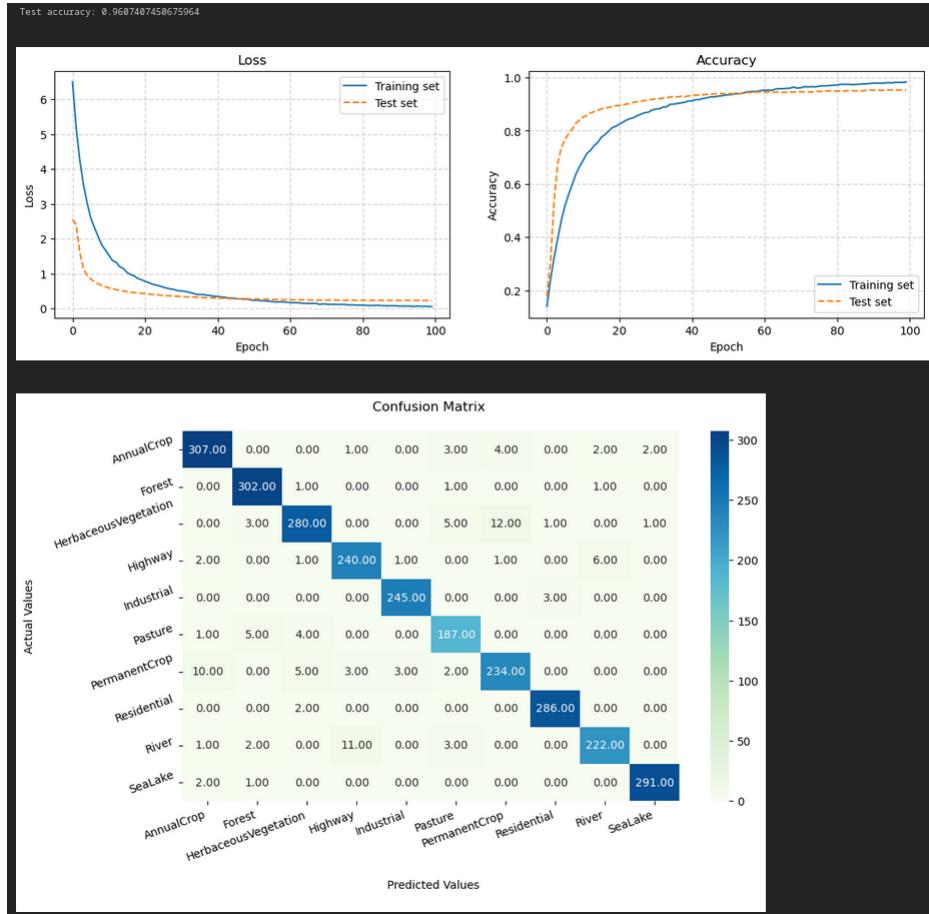


Figura 3.36. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

questi risultati con dropout a 0.8 smentiscono un po i dati precedenti perché non stante sembri che l'overfitting sia minore l'accuratezza sulla divisione di test è maggiore

Test Accuracy: 95.51%

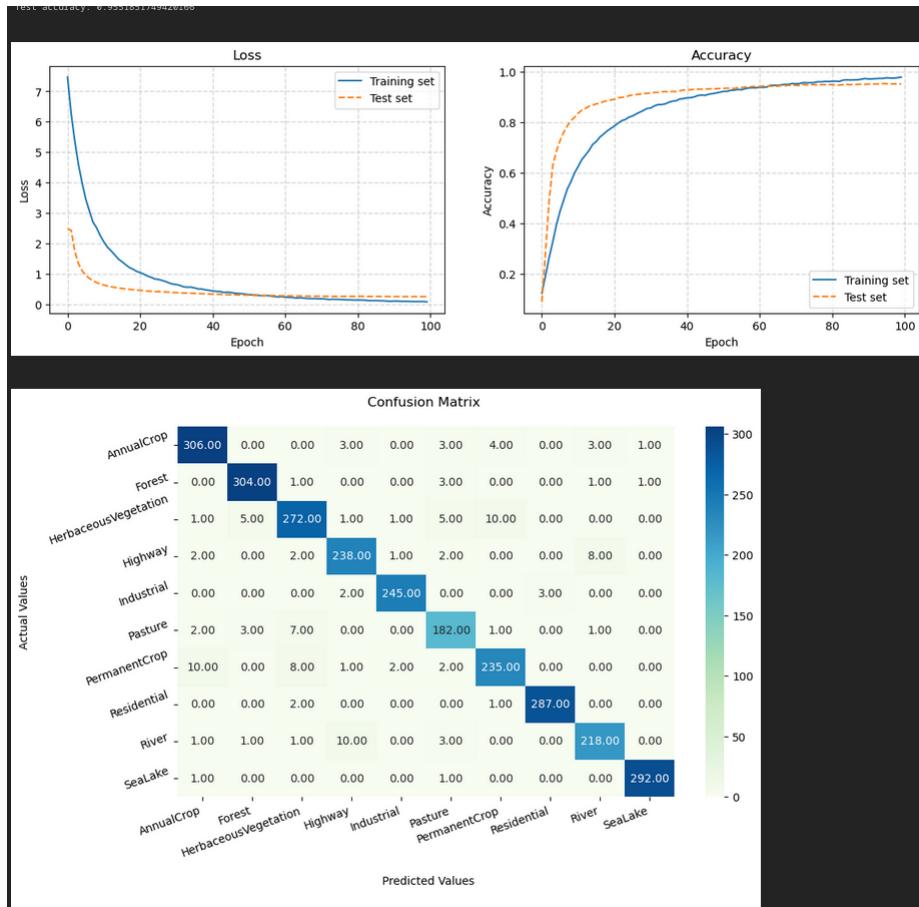


Figura 3.37. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche e la confusion matrix tra le varie classi.

I risultati precedenti erano tutti con data augmentation mentre questi erano i risultati prima di applicare data augmentation.

Test Accuracy: 94.73%

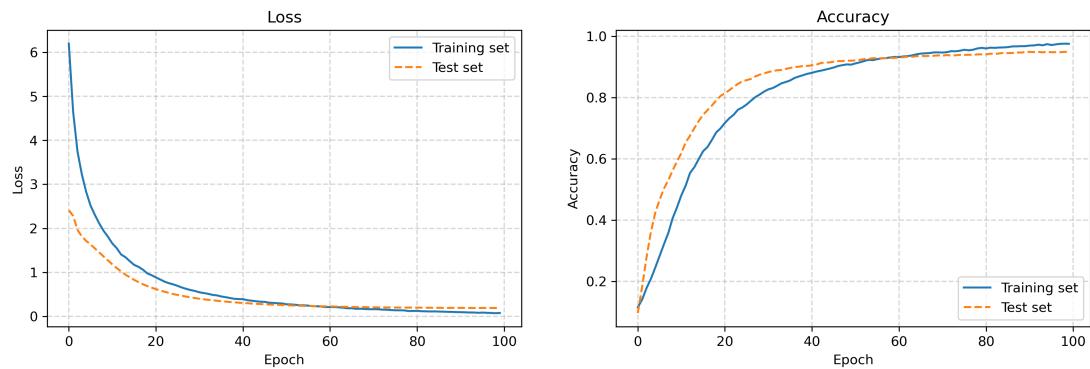


Figura 3.38. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

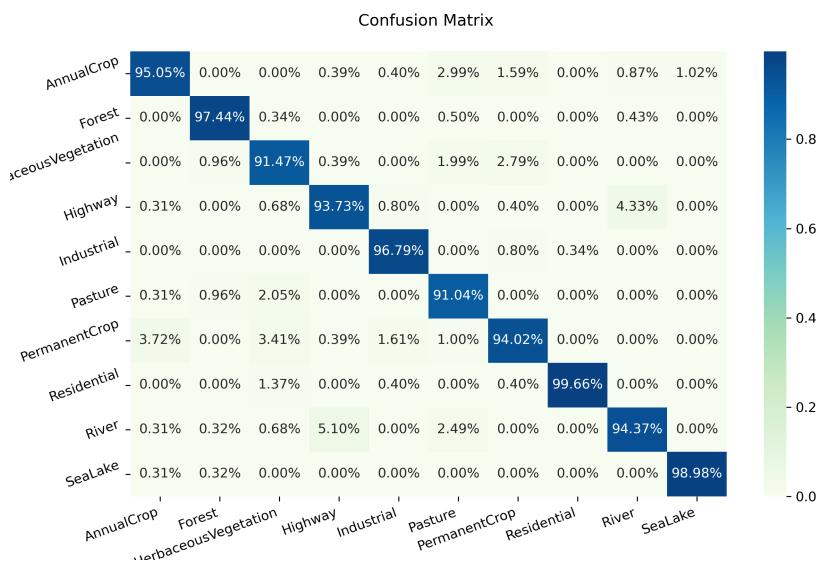


Figura 3.39. Confusion Matrix tra le varie classi.

dai risultati di MobileNetV2 prima di applicare data augmentation si capisce che all'inizio faccia fatica a generalizzare, ma dall'epoca 20 circa riesce ad imparare molto e i risultati sono migliori della versione precedente della rete

Test Accuracy: 95.62%

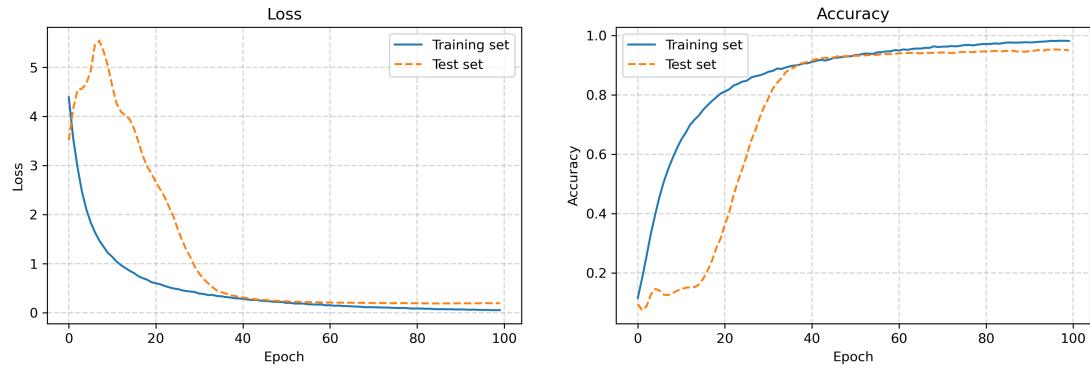


Figura 3.40. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

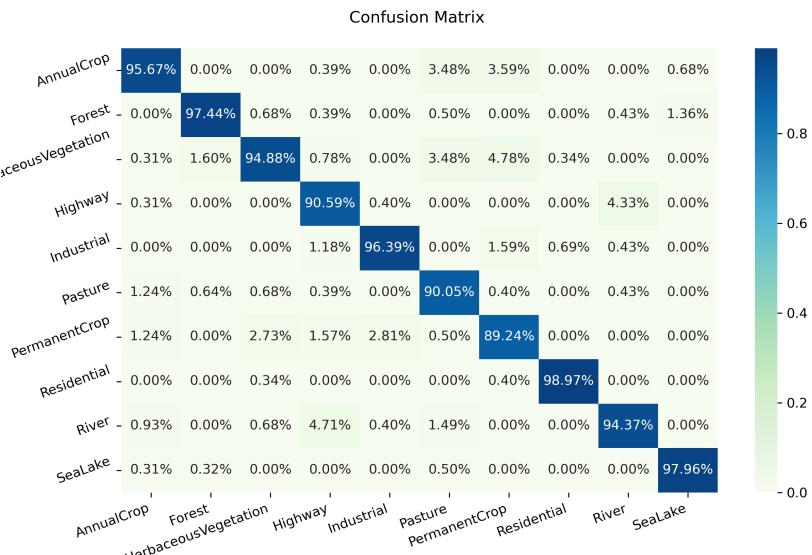


Figura 3.41. Confusion Matrix tra le varie classi.

ResNet

Anche ResNet ha avuto problemi con il dropout qui per esempio viene inizializzato con un valore troppo alto, nonostante ciò ha un ottima accuratezza.

Test Accuracy: 95.26%

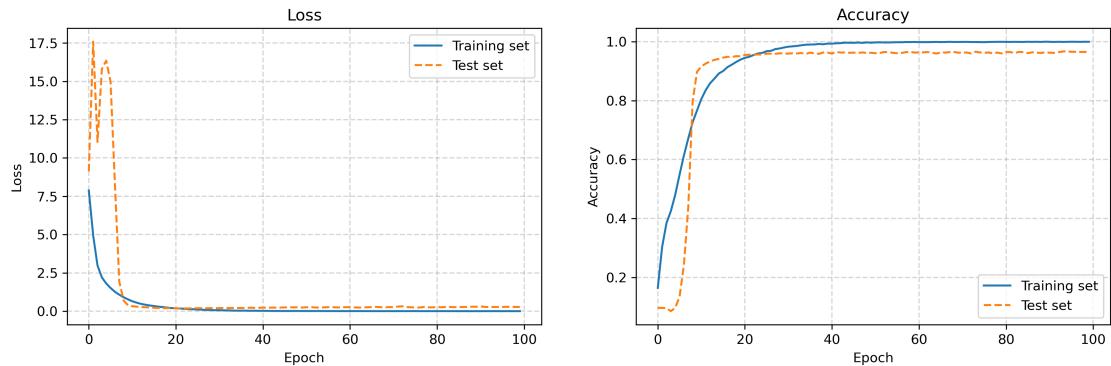


Figura 3.42. Grafici su come si evolve la loss e l'accuratezza nelle varie epocha.

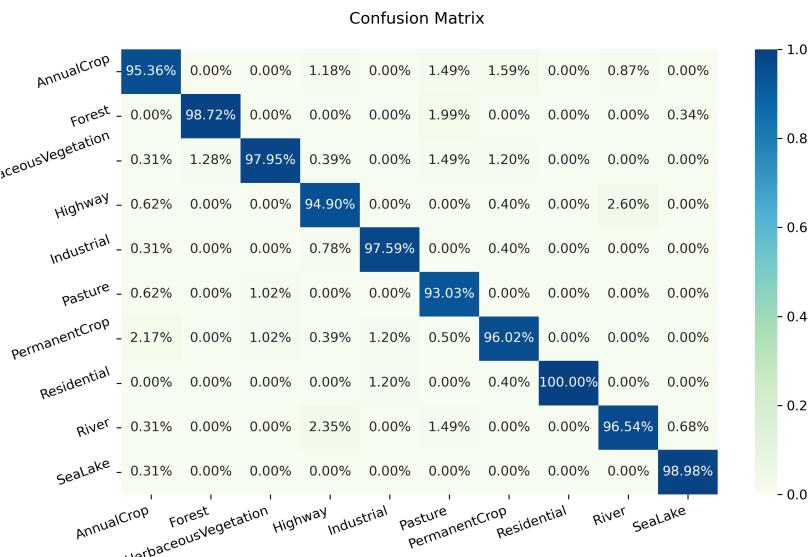


Figura 3.43. Confusion Matrix tra le varie classi.

Questo invece è un valore che è risultato ottimale, ed è 0.875.

Test Accuracy: 96.74%

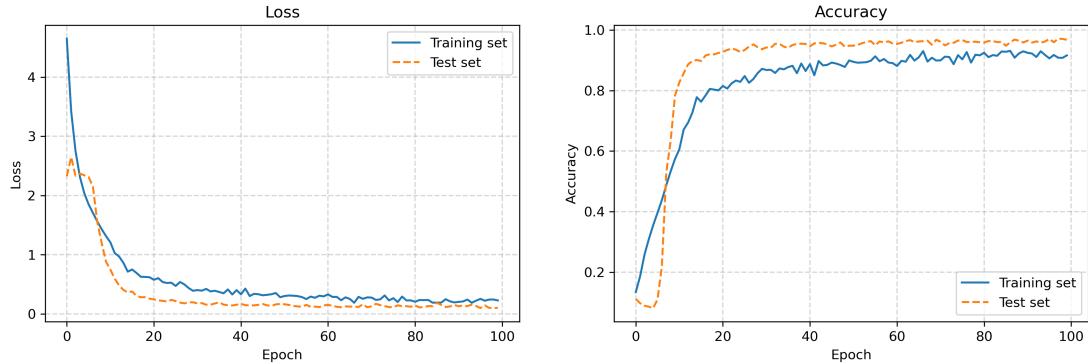


Figura 3.44. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

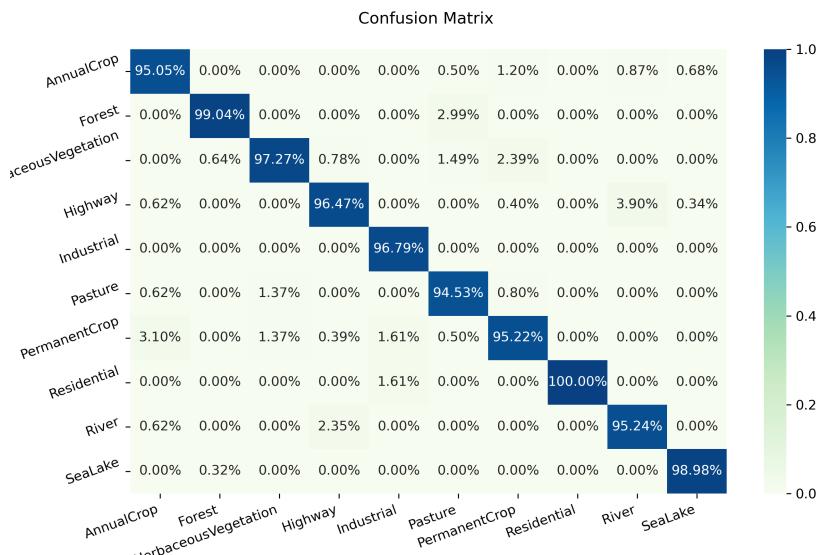


Figura 3.45. Confusion Matrix tra le varie classi.

Questo è un altro set di risultati di ResNet più accurato grazie alla data augmentation

Test Accuracy: 97.08%

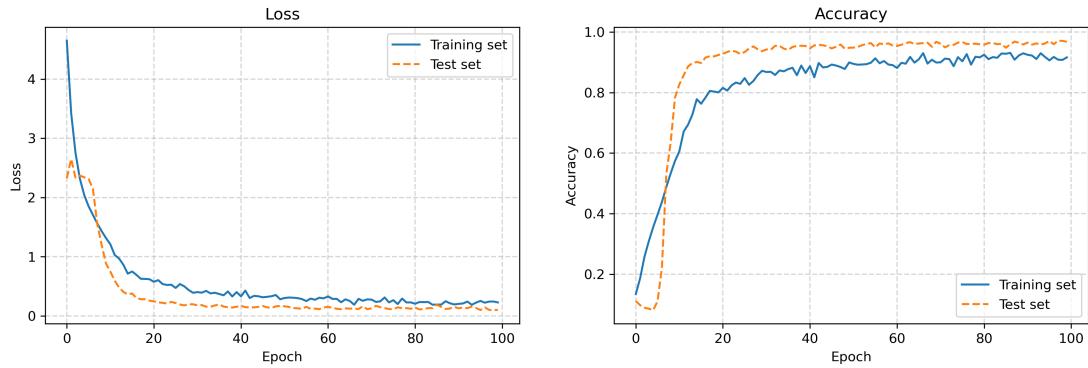


Figura 3.46. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

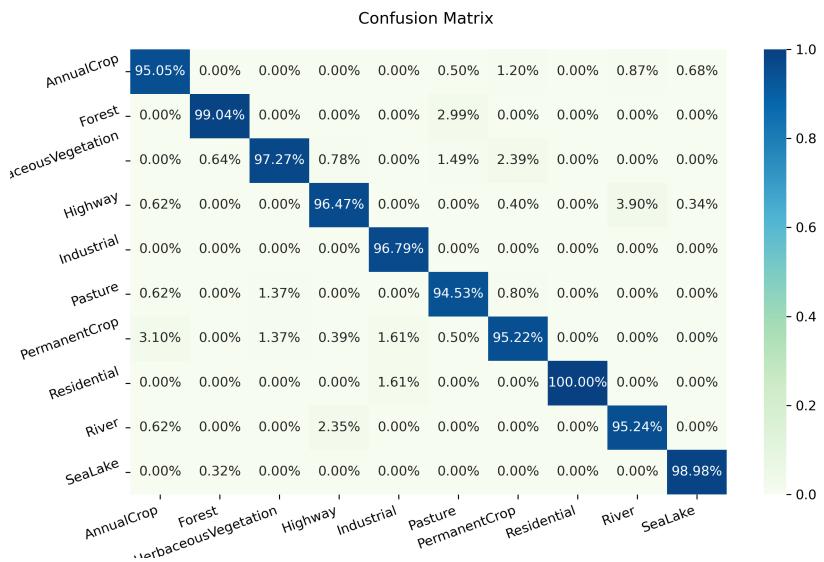


Figura 3.47. Confusion Matrix tra le varie classi.

In questo set di risultati di resnetV2 senza data augmentation si può notare come in questo caso sia inferiore alla sua versione precedente

Test Accuracy: 93.43%

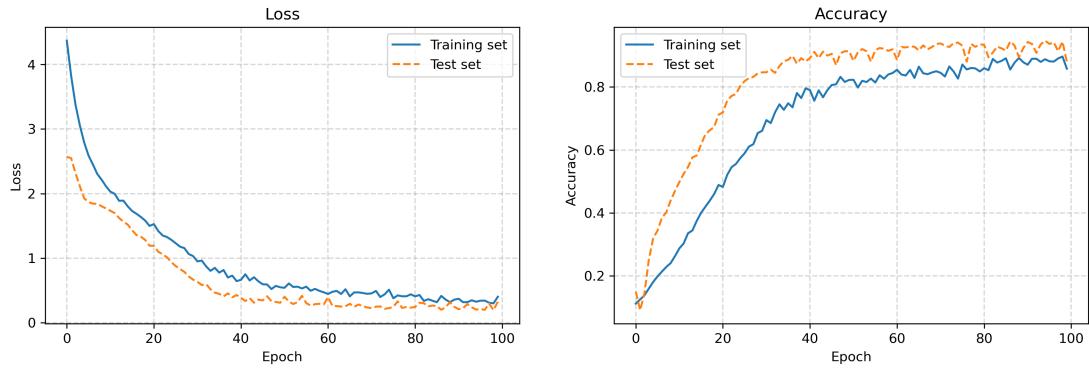


Figura 3.48. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

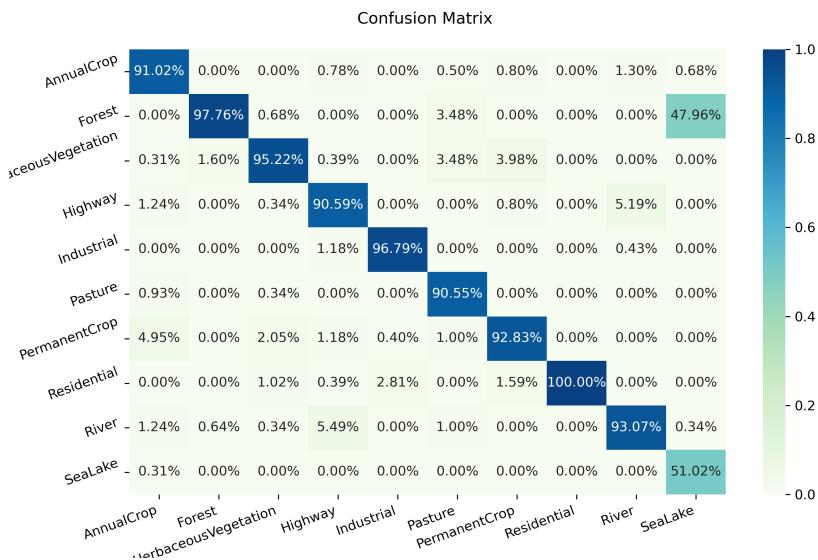


Figura 3.49. Confusion Matrix tra le varie classi.

Questo set di risultati con data augmentation riavvicina ResNetV2 a ResNet ma rimanendo comunque inferiore

Test Accuracy: 95.36%

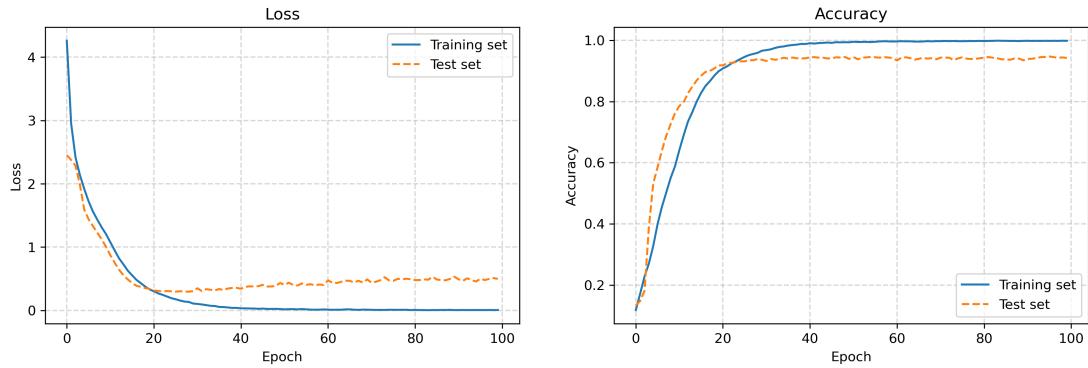


Figura 3.50. Grafici su come si evolve la loss e l'accuratezza nelle varie epoche.

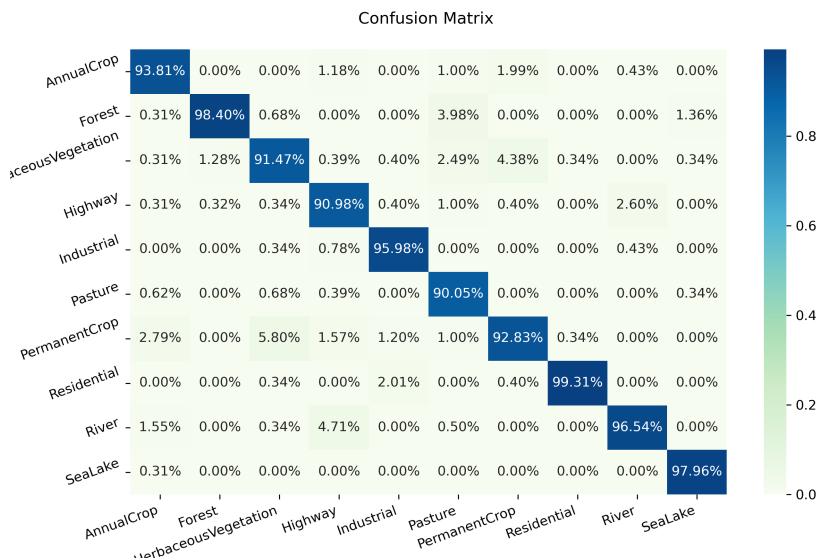


Figura 3.51. Confusion Matrix tra le varie classi.

Capitolo 4

Conclusioni e lavori futuri

In questo progetto sono state usate tecniche di Computer Vision e di Deep Learning per creare un classificatore con dieci classi sul dataset EuroSAT dell'ESA. Inizialmente è stato implementato e testato il Multi Layer Perceptron, successivamente questo algoritmo è stato raffinato, cercando di migliorarne l'accuratezza. In particolare si è resa l'architettura più profonda, portandola da due a cinque livelli, successivamente si sono usate tutte le tredici bande spettrali rispetto alle sole bande RGB usate in precedenza infine si è sperimentato pure con la data augmentation. Questa si compone di diverse modifiche alle immagini quali il flip, la rotazione e inversione, in più su quelle RGB si è potuto operare anche sui colori cambiando tinta, saturazione, illuminazione e contrasto. I risultati ottenuti sono comparabili a quelli ottenuti dagli autori del dataset con metodi di Machine Learning quando si sono usate immagini RGB e hanno ottenuto circa il 10% in più di accuratezza con l'uso di tutte le bande spettrali presenti nel dataset.

Successivamente si è sperimentato lo stesso task facendo uso di algoritmi convoluzionali, in questo caso si è provato principalmente la data augmentation, che è risultata comportarsi in maniera adeguata al compito. I risultati sono stati molto accurati rispetto al Multi Layer Perceptron superandolo di ben il venti/venticinque per cento.

Infine si è sperimentato con varie reti standard come EfficientNet, MobileNet e ResNet. In futuro si vorrebbero effettuare questi ulteriori esperimenti:

- Provare profondità della parte convoluzionale dell'architettura diverse, poiché per questa tesi è stato cambiato solo il numero di matrici e il parametro che controlla quanti elementi va fatta la convoluzione.
- Provare ad usare solo alcune bande per vedere quali sono i gruppi migliori.
- Provare altre reti standard o fare altri esperimenti con il dropout per bilanciare nel miglior modo possibile accuratezza e overfitting e aggiungendo reti dense prima o dopo la parte standard per verificare quanto possono essere di supporto per la parte convoluzionale.
- Provare i modelli addestrati su questo dataset su altri dataset, per testarne la generalizzazione.

Bibliografia

- [1] European Space Agency Sentinel-2's mission, <https://sentinel.esa.int/web/sentinel/missions/sentinel-2>
- [2] European Space Agency Sentinel-2's mission, <https://sentinel.esa.int/web/sentinel/missions/sentinel-2>
- [3] European Space Agency Sentinel-2 mission's dataset on Tensorflow, <https://www.tensorflow.org/datasets/catalog/eurosat>
Notizia Ansa peggior siccità ultimi 500 anni, https://www.ansa.it/canale_scienza_tecnica/notizie/terra_poli/2022/09/06/nellestate-2022-la-peggiore-siccita-deuropa-in-500-anni-bab737df-97c3-4b4a-933a-bd820cf06764.html
- [4] Shigeki, Sugiyama (12 April 2019). Human Behavior and Another Kind in Consciousness: Emerging Research and Opportunities: Emerging Research and Opportunities. IGI Global. ISBN 978-1-5225-8218-2.
- [5] Spiegazione video della convoluzione, <https://www.youtube.com/watch?v=KuXjwB4LzSA>
- [6] Paper di MobileNet, <https://arxiv.org/pdf/1704.04861.pdf>
- [7] Paper di EfficientNet, <https://arxiv.org/pdf/1905.11946.pdf>
- [8] Paper di MobileNet, <https://arxiv.org/pdf/1512.03385.pdf>
- [9] P. Helber, B. Bischke, A. Dengel and D. Borth, "Introducing Eurosat: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification," IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, 2018, pp. 204-207, doi: 10.1109/IGARSS.2018.8519248.
- [10] intuizioni su Adam, <https://www.geeksforgeeks.org/intuition-of-adam-optimizer/>
- [11] intuizioni su Adam, [https://www.gabormelli.com/RKB/Root_Mean_Square_Propagation_Algorithm_\(RMSprop\)](https://www.gabormelli.com/RKB/Root_Mean_Square_Propagation_Algorithm_(RMSprop))