



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

# Valutazione di algoritmi di Machine Learning per la classificazione di immagini

RELATORE

Prof. Fabio Palomba

Dott. Giammaria Giordano

Università degli Studi di Salerno

CANDIDATO

**Daniele Carangelo**

Matricola: 0512105819

*Questa tesi è stata realizzata nel*



*“Il più grande nemico della conoscenza non è l’ignoranza, è l’illusione della conoscenza.”*

*(Stephen Hawking)*

## **Abstract**

L'intelligenza artificiale (IA) sta conducendo senza alcun dubbio l'innovazione tecnologica degli ultimi anni ricoprendo vari campi della società moderna. Una delle applicazioni più comuni dell'IA è sicuramente la classificazione di immagini utilizzata ad esempio in medicina, automotive, videosorveglianza ecc. Questo lavoro di ricerca è volto a valutare le prestazioni di alcuni algoritmi di Machine Learning quali Support Vector Machine, Random Forest, Reti neurali e Reti neurali convoluzionali sulla classificazione delle immagini utilizzando diverse metriche come: accuracy, loss, F-score ecc. Questi algoritmi attraverso le librerie Python come Tensorflow, keras e Scikit-learn verranno implementati e addestrati con i dataset Fashion-MNIST e CIFAR-10 per poi confrontarne le performance. Da questo confronto emergono valori di precisione e affidabilità nettamente superiori su Fashion-MNIST evidenziando maggiore difficoltà nella classificazione del CIFAR-10. Un altro risultato ottenuto riguarda le reti neurali convoluzionali che tra gli algoritmi testati risultano essere le più performanti nella classificazione di immagini.

Link alla repository GitHub del lavoro svolto:

[https://github.com/daniele-carangelo/image\\_classification](https://github.com/daniele-carangelo/image_classification)

---

# Indice

---

<b>Elenco delle Figure</b>	<b>iv</b>
<b>Elenco delle Tabelle</b>	<b>vi</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Contesto Applicativo . . . . .	1
1.2 Motivazioni e Obiettivi . . . . .	1
1.3 Risultati Ottenuti . . . . .	2
1.4 Struttura della tesi . . . . .	2
<b>2 Background e Stato dell'Arte</b>	<b>4</b>
2.1 Background . . . . .	4
2.1.1 Intelligenza Artificiale . . . . .	4
2.1.2 Machine Learning . . . . .	5
2.1.3 Deep Learning . . . . .	5
2.1.4 Dataset . . . . .	6
2.1.5 Algoritmi per la classificazione di immagini . . . . .	8
2.1.6 Macchina a vettori di supporto . . . . .	9
2.1.7 Random Forest . . . . .	11
2.1.8 Reti neurali . . . . .	12
2.1.9 Reti neurali convoluzionali . . . . .	16

2.2	Stato dell'Arte . . . . .	20
2.2.1	Classificazione di Fashion-MNIST . . . . .	20
2.2.2	CNN con filter size differenti . . . . .	25
2.2.3	Classificazione di CIFAR-10 . . . . .	26
2.2.4	Limiti Stato dell'Arte . . . . .	31
<b>3</b>	<b>Metodologia di ricerca</b>	<b>32</b>
3.1	Obiettivo della ricerca . . . . .	32
3.2	Materiali . . . . .	32
3.3	Metodi . . . . .	33
3.3.1	Caricamento e preparazione dei dati . . . . .	33
3.3.2	Costruzione dei modelli . . . . .	34
3.4	Metodi e Metriche di Valutazione . . . . .	36
<b>4</b>	<b>Risultati</b>	<b>38</b>
4.1	Risultati classificazione con Support Vector Machine . . . . .	38
4.1.1	Test su Fashion-MNIST . . . . .	38
4.1.2	Test su CIFAR-10 . . . . .	39
4.2	Risultati classificazione con Random Forest . . . . .	41
4.2.1	Test su Fashion-MNIST . . . . .	41
4.2.2	Test su CIFAR-10 . . . . .	42
4.3	Risultati classificazione con Reti Neurali . . . . .	43
4.3.1	Test su Fashion-MNIST . . . . .	43
4.3.2	Test su CIFAR-10 . . . . .	45
4.4	Risultati classificazione con Reti Neurali Convoluzionali . . . . .	46
4.4.1	Test su Fashion-MNIST . . . . .	46
4.4.2	Test su CIFAR-10 . . . . .	49
4.5	Confronto finale . . . . .	52
<b>5</b>	<b>Conclusioni</b>	<b>54</b>
5.1	Lavoro svolto . . . . .	54
5.2	Risultati ottenuti . . . . .	55
5.3	Sviluppi futuri . . . . .	55



---

## Elenco delle figure

---

2.1	Modello di Rete Neurale . . . . .	6
2.2	Panoramica del dataset Fashion-MNIST . . . . .	7
2.3	Panoramica del dataset Cifar-10 . . . . .	8
2.4	Support vector machine . . . . .	9
2.5	Differenza tra dati separabili e non separabili linearmente . . . . .	10
2.6	Applicazione del kernel su dati non linearmente separabili . . . . .	11
2.7	Struttura alberi decisionali . . . . .	11
2.8	Funzionamento del Bagging . . . . .	12
2.9	Struttura di un neurone artificiale . . . . .	13
2.10	Modello di Rete Neurale . . . . .	14
2.11	Struttura di un neurone artificiale in dettaglio . . . . .	14
2.12	Grafico della funzione Sigmoide . . . . .	15
2.13	Grafico della funzione ReLU . . . . .	16
2.14	Filtro convoluzionale per linee verticali . . . . .	17
2.15	Filtro convoluzionale per linee orizzontali . . . . .	18
2.16	Operazione di pooling . . . . .	19
2.17	Rete neurale convoluzionale . . . . .	19
2.18	Grafico tempi di addestramento . . . . .	24
2.19	Architettura CNN implementata . . . . .	27

2.20 Architettura AlexNet su ImageNet . . . . .	28
2.21 Architettura LeNet su MNIST . . . . .	29
2.22 Accuratezza del training e test di LeNet-5 . . . . .	30
2.23 Architettura VGG-19 . . . . .	30
3.1 Visualizzazione campione di Fashion-MNIST . . . . .	33
4.1 Matrice di confusione di SVM su F-MINST . . . . .	39
4.2 Matrice di confusione di SVM sul CIFAR-10 . . . . .	40
4.3 Matrice di confusione di Random Forest su F-MINST . . . . .	41
4.4 Matrice di confusione di Random Forest su CIFAR . . . . .	42
4.5 Matrice di confusione della Rete Neurale su F-MINST . . . . .	44
4.6 Matrice di confusione della Rete Neurale sul CIFAR-10 . . . . .	45
4.7 Matrice di confusione della CNN sul F-MNIST per il Modello 1 . . .	47
4.8 Matrice di confusione della CNN sul F-MNIST per il Modello 2 . . .	47
4.9 Matrice di confusione della CNN sul F-MNIST per il Modello 3 . . .	48
4.10 Matrice di confusione della CNN sul CIFAR-10 per il Modello 1 . . .	50
4.11 Matrice di confusione della CNN sul CIFAR-10 per il Modello 2 . . .	50
4.12 Matrice di confusione della CNN sul CIFAR-10 per il Modello 3 . . .	51

---

## Elenco delle tabelle

---

2.1	Architetture Analizzate . . . . .	20
2.2	Architetture Analizzate . . . . .	21
2.3	Confronto metodi di ottimizzazione . . . . .	22
2.4	Accuratezze al variare della batch size . . . . .	22
2.5	Comparazione dei risultati ottenuti . . . . .	23
2.6	Algoritmi di ML confrontati . . . . .	25
2.7	Risultati comparazione con filtri differenti . . . . .	26
2.8	Risultati comparazione con filtri differenti . . . . .	26
3.1	Architetture Analizzate . . . . .	36
4.1	Valori F-score sul F-MNIST con SVM . . . . .	39
4.2	Valori F-score sul CIFAR-10 con SVM . . . . .	40
4.3	Valori F-score sul F-MNIST con Random Forest . . . . .	41
4.4	Valori F-score sul CIFAR-10 con Random Forest . . . . .	43
4.5	Accuracy e loss su F-MNIST con le reti neurali . . . . .	43
4.6	Valori F-score su F-MNIST con le reti neurali . . . . .	44
4.7	Accuracy e loss su CIFAR-10 con le reti neurali . . . . .	45
4.8	Valori F-score sul CIFAR-10 con le reti neurali . . . . .	46
4.9	Accuracy e loss su F-MNIST con le reti neurali convoluzionali . . . . .	46

4.10 Comparazione valori F-score su F-MNIST con CNN . . . . .	49
4.11 Accuracy e loss sul CIFAR-10 con le reti neurali convoluzionali . . . . .	49
4.12 Comparazione valori F-score su CIFAR-10 con CNN . . . . .	52
4.13 Riepilogo SVM . . . . .	52
4.14 Riepilogo Random Forest . . . . .	53
4.15 Riepilogo rete neurale . . . . .	53
4.16 Riepilogo del modello 3 di CNN . . . . .	53

# CAPITOLO 1

---

## Introduzione

---

### 1.1 Contesto Applicativo

L'intelligenza Artificiale (IA) è uno dei trend che sta sicuramente guidando l'innovazione tecnologica e non solo negli ultimi anni. L'obiettivo dell'IA è quello di creare modelli intelligenti in grado di prendere decisioni e apprendere similmente alla mente umana. Uno dei campi dell'Intelligenza artificiale più ambito dai ricercatori è il riconoscimento e la classificazione delle immagini visto l'utilizzo in svariati campi. Alcune delle applicazioni più comuni riguardano ad esempio il riconoscimento di oggetti e volti, sicurezza, guida autonoma ecc. L'obiettivo della classificazione consiste nel trovare caratteristiche all'interno delle immagini utili a collocarle in determinate categorie. Tra gli algoritmi utilizzati per la classificazione di immagini troviamo: Random Forest, Reti Neurali Convoluzionali, Support Vector Machine ecc. A supporto degli algoritmi ci sono dataset creati ad hoc ed utili ad addestrare i modelli per ottenere il risultato desiderato [1].

### 1.2 Motivazioni e Obiettivi

Gli obiettivi principali di questa ricerca sono:

- Costruire modelli di Machine Learning in grado di classificare immagini;
- Analizzare diverse metriche e tecniche per valutare gli algoritmi di classificazione;
- Confrontare i dataset Fashion-MNIST<sup>1</sup> e CIFAR-10<sup>2</sup> valutando la loro difficoltà.

La motivazione principale che ha portato alla scelta di questi obiettivi deriva dal limite incontrato durante l’analisi della letteratura. In particolare, mancano esperimenti dove la valutazione viene effettuata utilizzando diverse metriche di valutazione oltre all’accuratezza. Inoltre, mancano confronti approfonditi tra i dataset F-MNIST e CIFAR-10 utilizzando diversi algoritmi di Machine Learning.

## 1.3 Risultati Ottenuti

L’utilizzo di differenti metriche di valutazione come accuracy, loss, F-score, Matrice di confusione ed oob-error ci hanno aiutato a comprendere come i modelli si sono adattati ai dataset, mostrando i punti di forza e i punti deboli nella classificazione. Nel confronto tra F-MNIST e CIFAR-10 è stato mostrato come il F-MNIST sia più semplice da classificare rispetto al CIFAR-10 ottenendo valori di accuratezza superiori al 90%. I risultati inoltre evidenziano le capacità delle reti neurali convoluzionali nel classificare immagini dimostrandolo anche su dataset complessi come il CIFAR-10.

## 1.4 Struttura della tesi

La tesi è strutturata in cinque capitoli ordinati nel seguente modo:

- **Capitolo 1: Introduzione;**
- **Capitolo 2: Background e stato dell’arte**, tratta gli argomenti presenti nella tesi e gli attuali lavori presenti in letteratura;
- **Capitolo 3: Metodologia di ricerca**, descrive lo svolgimento della ricerca;

---

<sup>1</sup><https://github.com/zalandoresearch/fashion-mnist>

<sup>2</sup><https://www.cs.toronto.edu/~kriz/cifar.html>

- **Capitolo 4: Risultati**, mostra i risultati ottenuti dall'esperimento;
- **Capitolo 5: Conclusioni**, panoramica del lavoro svolto ed i suoi possibili lavori futuri.

# CAPITOLO 2

---

## Background e Stato dell'Arte

---

Questo capitolo è diviso in due sezioni principali: nella prima si approfondiscono i concetti utili a comprendere il lavoro svolto in questa tesi, nella seconda si analizzano i lavori presenti in letteratura con i relativi risultati ottenuti.

### 2.1 Background

#### 2.1.1 Intelligenza Artificiale

L'intelligenza Artificiale (IA) è una disciplina che si occupa di studiare e creare agenti intelligenti capaci di simulare la capacità ed il comportamento dell'intelligenza umana [2].

La definizione di Intelligenza Artificiale comprende diversi aspetti che possiamo sintetizzare in quattro punti principali:

- **Pensare Umanamente:** L'automazione delle attività che associamo al pensiero umano, come il processo decisionale, la risoluzione di problemi e l'apprendimento;
- **Pensare Razionalmente:** Lo studio dei processi di calcolo che rendono possibile percepire, ragionare e agire;

- **Agire Umanamente:** L'operazione eseguita dagli agenti intelligenti ed il suo risultato non è distinguibile da quella eseguita dagli umani;
- **Agire Razionalmente:** Capacità di operare autonomamente, essere in grado di percepire l'ambiente e di raggiungere il miglior risultato o, in condizioni di incertezza, il miglior risultato atteso [2].

### 2.1.2 Machine Learning

Il Machine Learning (ML), sottoinsieme dell'intelligenza artificiale (IA), si occupa della creazione di algoritmi in grado di identificare pattern nei dati. Gli algoritmi di ML sono in grado di apprendere e migliorare le performance in base ai dati che hanno a disposizione [3]. Esistono due approcci principali all'apprendimento automatico:

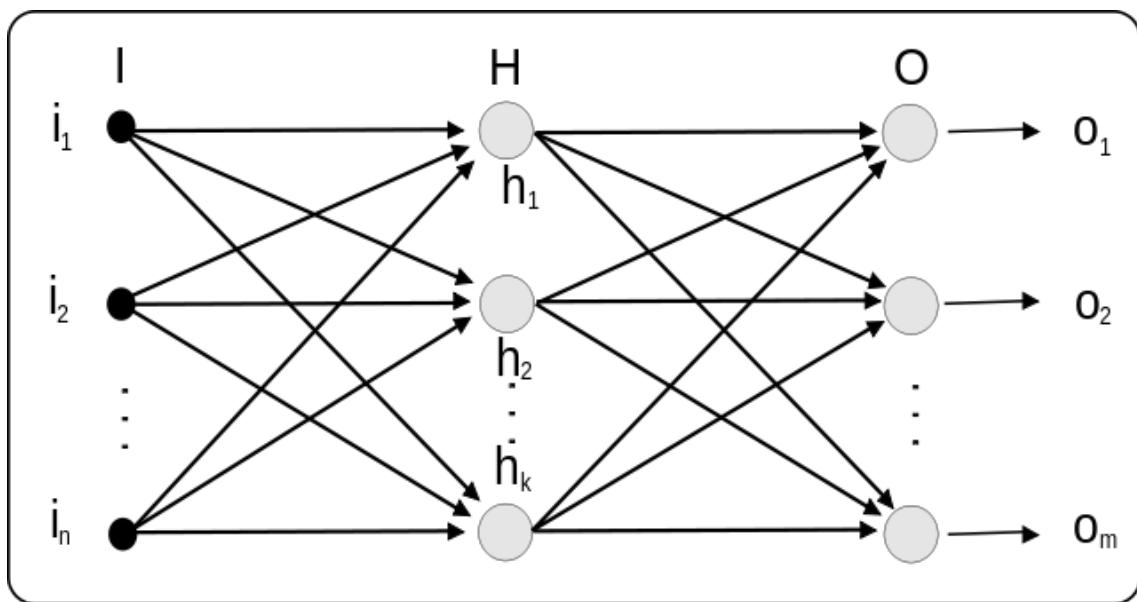
- **Apprendimento supervisionato:** L'algoritmo utilizza set di dati contrassegnati da etichette. Utilizzando ad esempio un dataset di numeri scritti a mano le etichette corrispondono alla cifra scritta. L'obbiettivo di questo metodo di apprendimento è quello di trovare una regola generale che associa ogni input all'output corretto;
- **Apprendimento non supervisionato:** L'approccio non supervisionato è incentrato su dati non etichettati. Un algoritmo di apprendimento non supervisionato ha lo scopo di analizzare i dati in input per trovare strutture e caratteristiche utili a classificare i dati in categorie distinte. Un esempio di utilizzo di algoritmi non supervisionati può essere il raccogliere articoli da diverse testate giornalistiche in categorie comuni come cronaca, politica, gossip ecc [4].

### 2.1.3 Deep Learning

Il Deep Learning è un sottoinsieme del Machine Learning basato sulle reti neurali organizzate in diversi livelli e con diversi parametri. Gli algoritmi di deep learning analizzano in genere immagini, testo, suoni e altri dati per estrarre pattern e caratteristiche utili poi per elaborare previsioni. L'idea del deep learning si basa sulle reti neurali presenti nel cervello umano. Il cervello umano contiene milioni di neuroni connessi tra loro che lavorano insieme per apprendere ed elaborare le informazioni.

Allo stesso modo, le reti neurali artificiali, sono costituite da molti strati di neuroni artificiali detti nodi che lavorano insieme all'interno del computer. I neuroni artificiali sono moduli computazionali che, grazie a calcoli matematici, elaborano un input producendo un risultato che successivamente verrà rielaborato da un altro nodo. Le reti neurali artificiali sono algoritmi di deep learning che utilizzano questi nodi per risolvere problemi complessi [5].

Nella Figura 2.1 vediamo una rete neurale formata da tre layer: input, hidden e output.



**Figura 2.1:** Modello di Rete Neurale

## 2.1.4 Dataset

### Cosa sono i dataset?

In precedenza, abbiamo visto che al centro di ogni algoritmo di apprendimento ci sono i dati ed in particolare i dataset. Un dataset è una collezione di dati (numeri, testi, immagini, audio) strutturati ad hoc per essere letti ed elaborati da un algoritmo utili ad addestrare modelli predittivi [6]. È possibile creare un proprio set di dati oppure utilizzare dataset già esistenti e pronti all'utilizzo. In questo lavoro i dataset trattati sono: Fashion-MNIST e CIFAR-10. La scelta è ricaduta su questi ultimi essendo tra i dataset più utilizzati come benchmark per algoritmi di Machine Learning.

## Fashion-MNIST

Fashion-MNIST (F-MNIST) è un dataset contenente immagini di capi di abbigliamento presenti su Zalando. Presentato da Xiao et al. [7] membri del reparto di ricerca di Zalando, nasce come sostituto a MNIST<sup>1</sup> [8] in quanto condivide le stesse dimensioni, formato e struttura dei dati. F-MNIST comprende 70.000 immagini, 60.000 per l'addestramento e 10.000 per il test. Ogni immagine è etichettata con la propria tipologia di abbigliamento (camicia, borsa, vestito ecc.) per un totale di 10 classi di abbigliamento. Nella Figura 2.2 viene mostrata una panoramica del dataset con le relative classi.



**Figura 2.2:** Panoramica del dataset Fashion-MNIST

Ogni immagine ha una dimensione di 28x28 per un totale di 784 pixel. Ad ogni pixel è associato un valore da 0 a 255 che indica il grado di luminosità sulla scala di grigi.

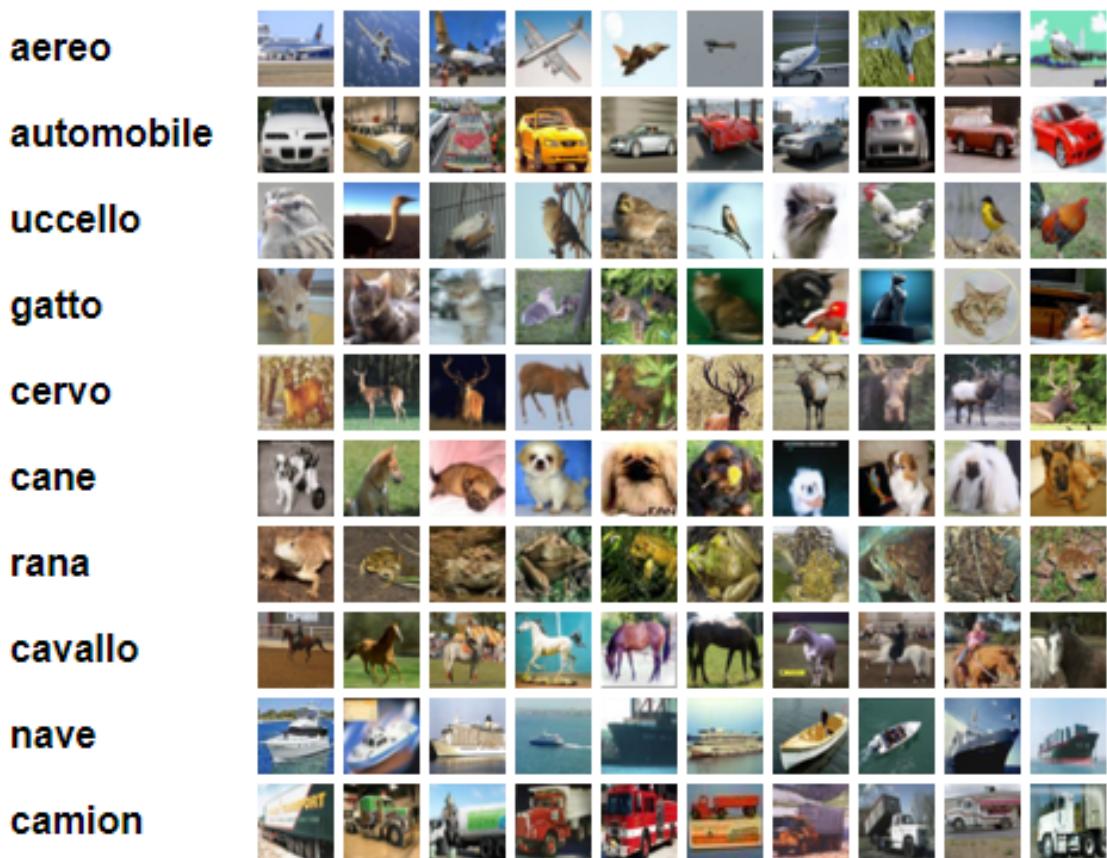
## CIFAR-10

Il Cifar-10 è un dataset introdotto da Krizhevsky [9] nel 2009 che utilizza un sottoinsieme di immagini del "80 million Tiny Images"<sup>2</sup>. Formato da 60.000 immagini

<sup>1</sup>Dataset di cifre numeriche scritte a mano

<sup>2</sup>Dataset formato da 80 milioni di immagini estratte dal Web ed ora deprecato per motivi etici.

a colori con una dimensione di 32x32 pixel divise in 10 classi di appartenenza. Il set di addestramento comprende 50.000 immagini ed il set di test comprende 10.000 immagini. Le classi del dataset sono state scelte in modo da essere mutuamente esclusive. Ad esempio, la classe automobile comprende solo berline, suv, utilitarie e simili, mentre la classe camion comprende solo grandi camion ma nessuna delle due classi includono i camioncini. La differenza sostanziale rispetto ad F-MNIST consiste nelle immagini a colori del Cifar-10. Ad ogni pixel sono associati tre valori che corrispondono ai valori RGB. In Figura 2.3 viene mostrata una panoramica del dataset con le relative classi.



**Figura 2.3:** Panoramica del dataset Cifar-10

### 2.1.5 Algoritmi per la classificazione di immagini

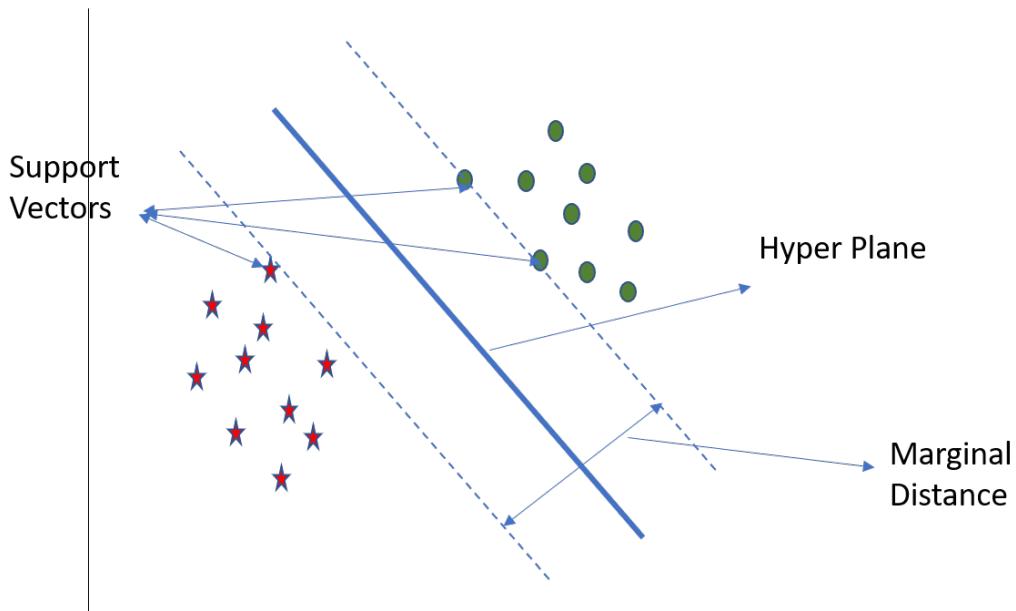
La classificazione delle immagini è uno dei topic di ricerca più di tendenza negli ultimi anni. L'obiettivo degli algoritmi di classificazione consiste nel cercare caratteristiche all'interno dell'immagini utili alla classificazione di queste ultime.

Come sottolineato da Chen et al. [10] gli algoritmi tradizionali di ML si concentrano principalmente sull'impostazione manuale di queste caratteristiche. Gli algoritmi di Deep Learning invece sono in grado di estrapolare queste caratteristiche in modo autonomo senza l'intervento dello sviluppatore.

### 2.1.6 Macchina a vettori di supporto

Una Support Vector Machine (SVM) è una tipologia di algoritmi di apprendimento utilizzati sia nella classificazione che nella regressione. Tra le applicazioni più note per le SVM ci sono la classificazione di immagini e suoni, applicazioni mediche e biologiche [11][12].

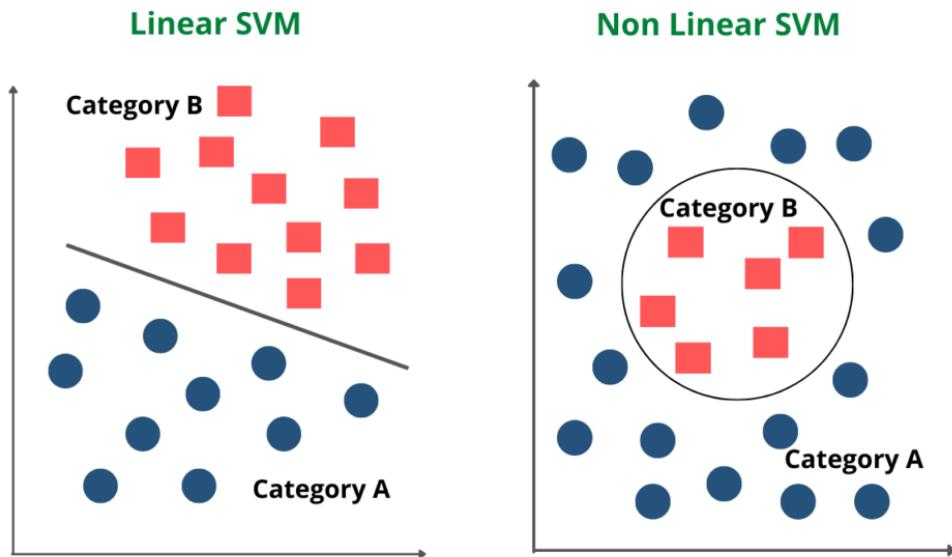
Il funzionamento delle SVM si basa sulla ricerca di un iperpiano in grado di separare i punti appartenenti ad una classe da quelli di altre classi. Nella Figura 2.4 vediamo l'iperpiano scelto che divide le due classi di oggetti.



**Figura 2.4:** Support vector machine

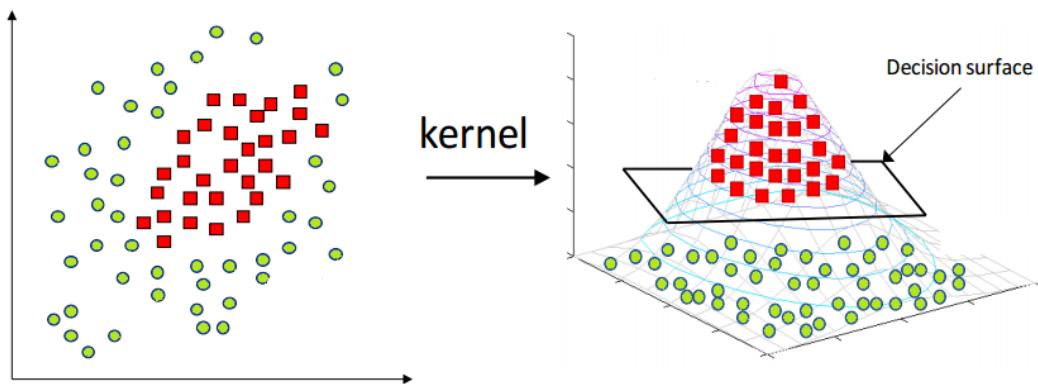
La scelta dell'iperpiano avviene grazie all'aiuto dei vettori di supporto ovvero dei confini tra una classe e l'altra. L'iperpiano selezionato è quello con i margini maggiori. I margini sono le distanze tra le classi ovvero la distanza tra i vettori di supporto. Le SVM vengono ottimizzate massimizzando i margini. Nell'esempio precedente, Figura 2.4, avevamo una classificazione lineare cioè i dati sono distribuiti in modo da

essere separabili linearmente con una retta o in generale iperpiani [12]. Non sempre però ci troviamo di fronte a dati linearmente separabili come vediamo nella Figura 2.5 di seguito.



**Figura 2.5:** Differenza tra dati separabili e non separabili linearmente

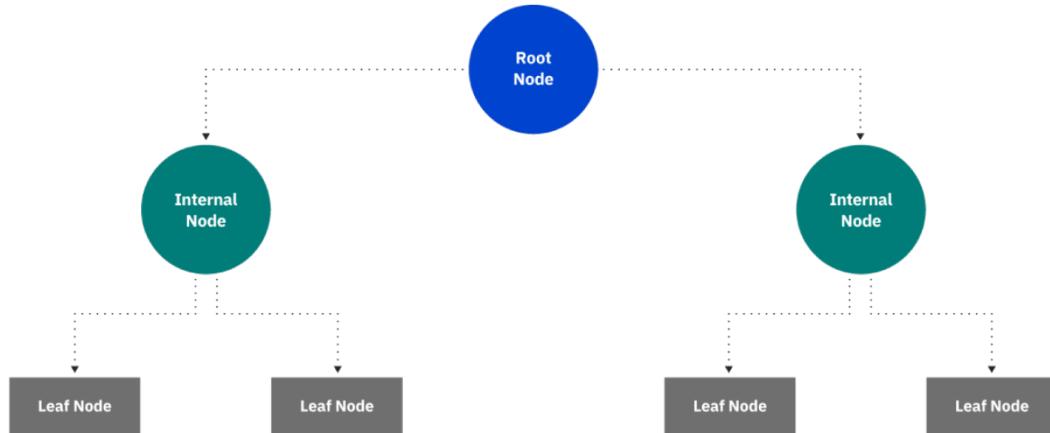
Nel caso di dati non separabili linearmente, una possibile soluzione è quella di mappare i dati in uno spazio a dimensioni maggiori così da poter essere divisi tramite iperpiani. Questa soluzione, seppur fattibile, richiede che venga mappato l'intero dataset in una nuova dimensione, operazione che richiede tempo e risorse di calcolo notevoli. Questa problematica può essere risolta utilizzando un metodo chiamato **Kernel trick** che consente di ottenere lo stesso risultato della mappatura ma senza farlo effettivamente. L'idea consiste nel creare combinazioni tra i dati per poi proiettarle su dimensioni maggiori. Questa proiezione avviene grazie alle cosiddette **Funzioni Kernel** [13]. Nella Figura 2.6 viene mostrato il funzionamento base di una funzione kernel.



**Figura 2.6:** Applicazione del kernel su dati non linearmente separabili

### 2.1.7 Random Forest

Il Random Forest è un algoritmo di classificazione costituito da molti alberi decisionali che appunto formano una foresta. Un albero decisionale è un modello predittivo gerarchico formato da una radice, rami ed un insieme di nodi e foglie come mostrato in Figura 2.7 [14].

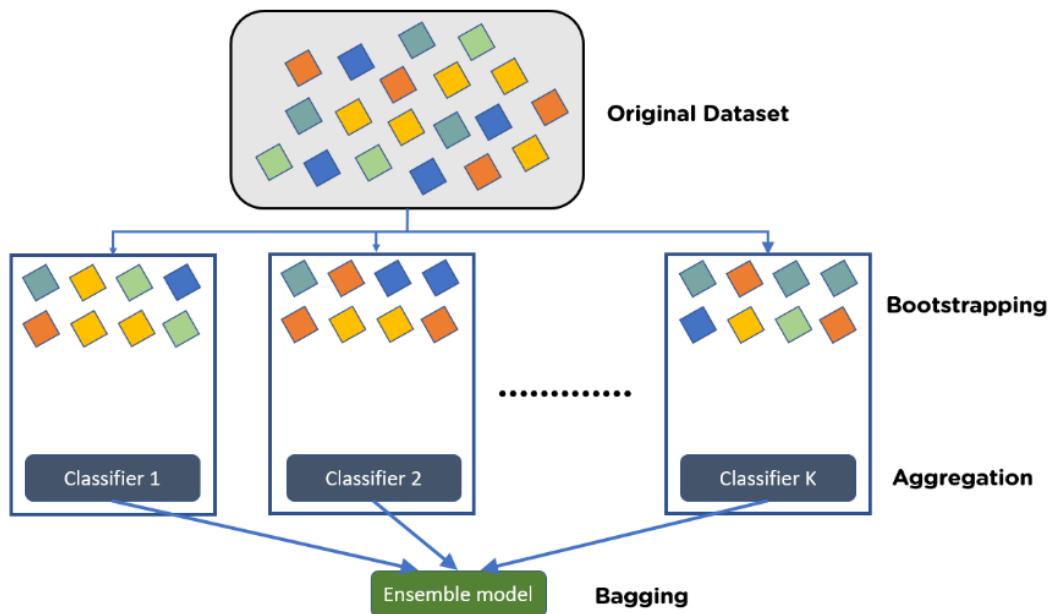


**Figura 2.7:** Struttura alberi decisionali

Nel Random Forest gli alberi decisionali sono aggregati tramite il **Bagging**, un metodo introdotto da Breiman nel 1996 [15]. Il Bagging o bootstrap aggregation, ha tre fasi principali:

- **Bootstrapping:** Questa operazione si occupa di creare diversi sottoinsiemi casuali partendo dal dataset iniziale. Essendo casuali possiamo scegliere anche più volte la stessa istanza;
- **Training:** In questa fase i campioni generati dal bootstrapping vengono addestrati indipendentemente e in parallelo tra loro;
- **Aggregation:** In quest'ultima fase vengono raccolte tutte le previsioni della fase di training e usate per calcolare la previsione finale. Nel caso di regressione viene presa la media di tutte le previsioni, per la classificazione invece, viene scelta la previsione maggiore [16].

Una rappresentazione grafica del bagging viene mostrata nella Figura 2.8



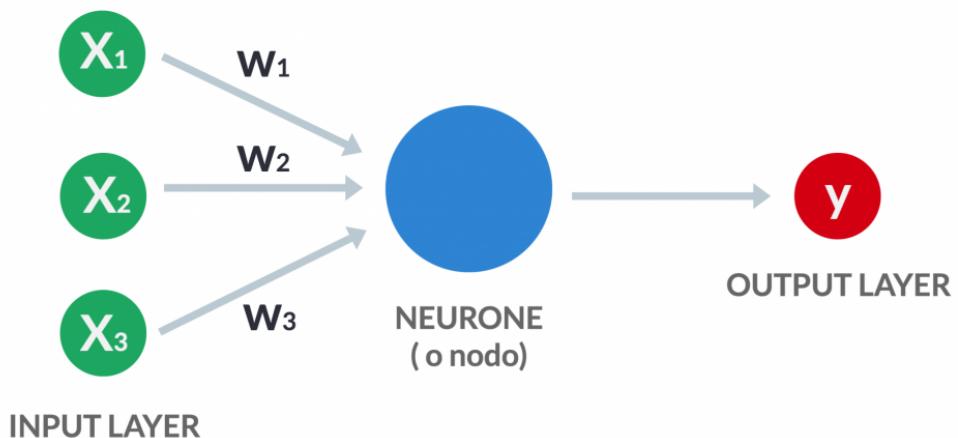
**Figura 2.8:** Funzionamento del Bagging

Il modello Random Forest utilizza il bagging con l'aggiunta di una componente casuale per la scelta delle caratteristiche degli alberi decisionali presenti al suo interno.

### 2.1.8 Reti neurali

Le reti neurali sono modelli computazionali ispirati al comportamento del cervello umano. Una rete neurale è organizzata su più livelli detti layer. Ogni layer ha

all'interno una serie di "neuroni" artificiali detti nodi che comunicano con i layer precedenti e successivi. In Figura 2.9 vediamo una rappresentazione grafica di un neurone artificiale.

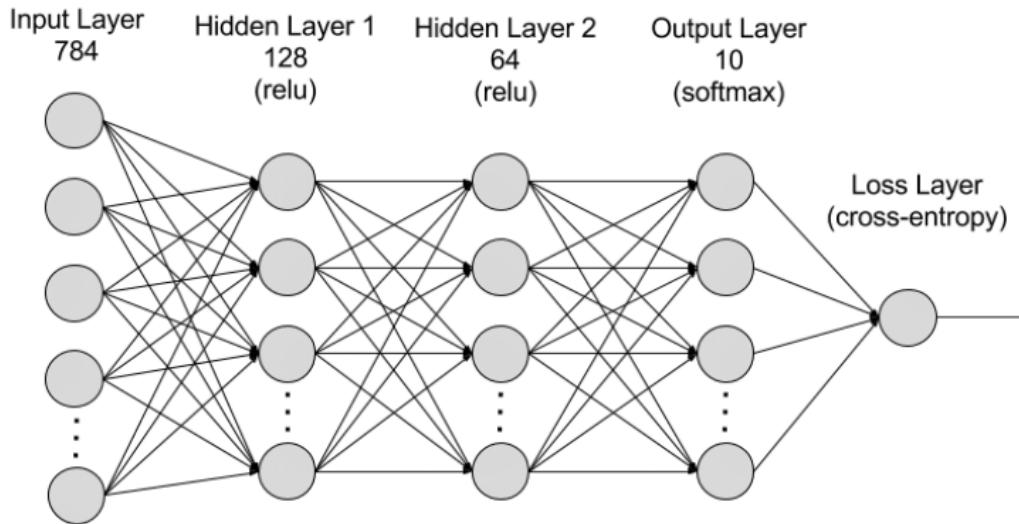


**Figura 2.9:** Struttura di un neurone artificiale

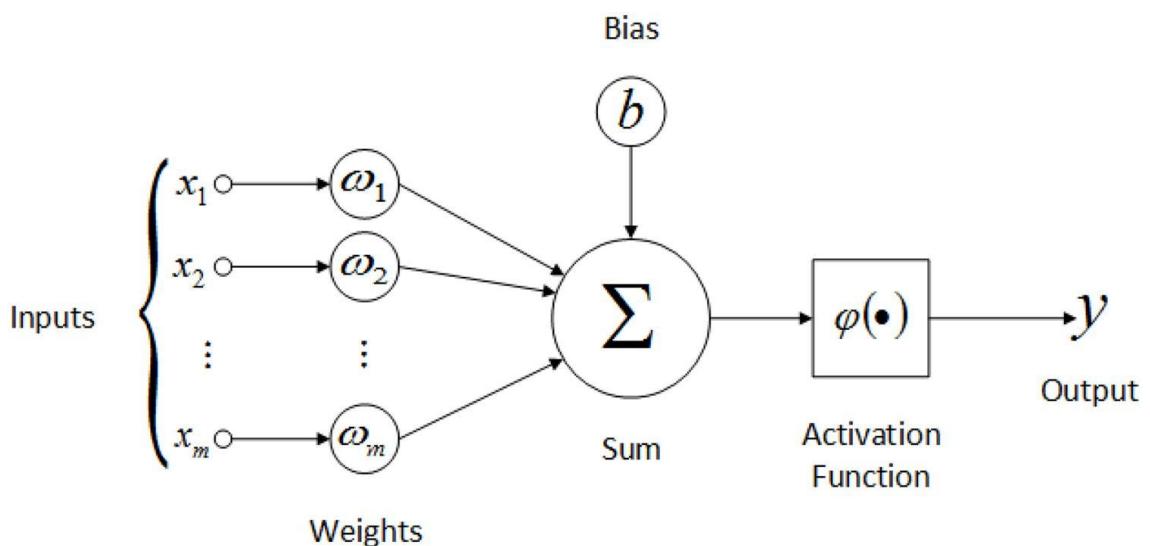
Ogni neurone elabora il segnale ricevuto dal layer precedente per poi inviarlo al successivo così da ottenere informazioni sempre più evolute e dettagliate. Come scrive N.Boldrini [17] in un recente articolo, le reti neurali in linea generale sono formate da tre strati principali:

- **Strato di input:** Riceve i dati in input e li prepara per essere elaborati dai neuroni;
- **Strato nascosto:** Si occupa dell'elaborazione vera e propria e può essere formato anche da più colonne di neuroni;
- **Strato di output:** Si ricevono i dati elaborati nello strato nascosto e vengono adattati per ottenere i risultati attesi.

Nella figura 2.10 vediamo un esempio di rete neurale con tre strati nascosti.

**Figura 2.10:** Modello di Rete Neurale

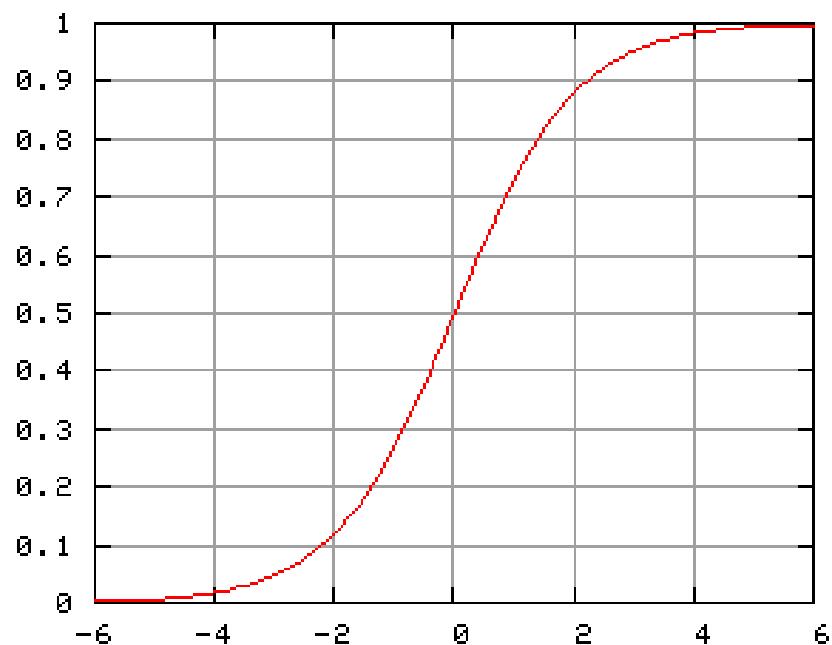
Nel neurone in Figura 2.9 vediamo che riceve una serie di input ( $x_1, x_2, x_3$ ) che sono associati ad alcuni pesi ( $w_1, w_2, w_3$ ). Questi pesi sono alcuni dei parametri che la rete modifica durante l'addestramento per migliorare le predizioni. L'output ottenuto rappresenta la somma ponderata tra i pesi e l'input e successivamente mappata da una funzione di attivazione. Le funzioni di attivazioni vengono utilizzate per limitare l'output del neurone in un certo intervallo altrimenti potrebbe assumere qualsiasi valore [18]. Nella Figura 2.11 che segue viene mostrato un neurone artificiale più in dettaglio.

**Figura 2.11:** Struttura di un neurone artificiale in dettaglio

Esistono diverse funzioni di attivazioni, in seguito vengono mostrate quelle utilizzate in questo esperimento.

### Sigmoide

Sigmoide è una funzione matematica che produce una curva ad "S". Questa funzione limiterà l'output tra 0 e 1 [18]. La Figura 2.12 mostra l'andamento della funzione graficamente.



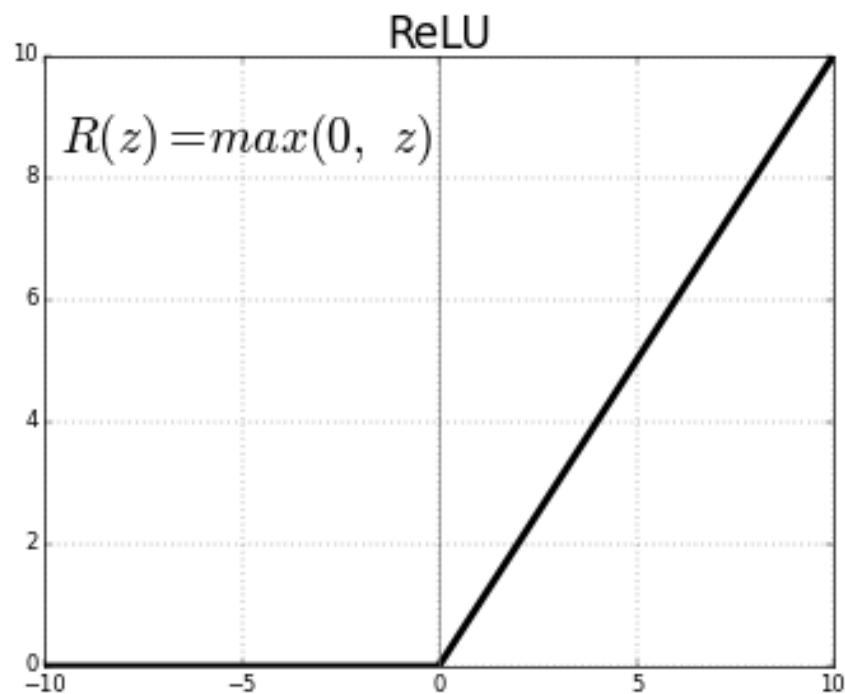
**Figura 2.12:** Grafico della funzione Sigmoide

### Softmax

Softmax è una funzione Sigmoide utilizzata per classificazioni con più classi. La funzione prende in input un vettore di K numeri reali e con una distribuzione di probabilità vengono normalizzati in K probabilità comprese tra 0 e 1 dove la somma di tutte le probabilità sarà uguale a 1. Ad esempio, nella classificazione di F-MNIST vedremo come la funzione softmax, nell'ultimo strato della rete, produrrà una probabilità per ogni classe del dataset per un totale di 10 probabilità dove la somma di queste ultime sarà 1 [18].

## ReLU

ReLU sta per Rectified Linear Unit ed è una delle funzioni di attivazioni più utilizzate, soprattutto nei livelli nascosti delle reti. Ha un intervallo che va da 0 ad infinito come vediamo nella Figura 2.13. In termini di performance è una delle funzioni più efficienti ma può causare la morte di alcuni neuroni perché, per valori negativi, restituisce sempre 0 e quindi il neurone non potrà migliorarsi [18].



**Figura 2.13:** Grafico della funzione ReLU

### 2.1.9 Reti neurali convoluzionali

Una rete neurale convoluzionale o CNN<sup>3</sup> è un'architettura del deep learning simile alle reti neurali viste in precedenza. Le reti neurali convoluzionali sono utili soprattutto per lavorare su immagini, voce e file audio. Una CNN è formata da tre sezioni principali:

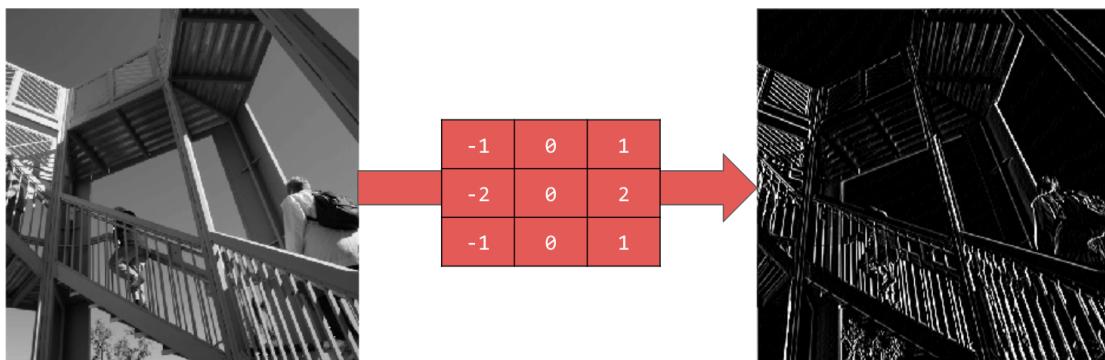
- **Layer di input:** Prepara L'input per essere processato dai livelli successivi;

---

<sup>3</sup>Convolutional Neural Network

- **Layer nascosti:** In questa sezione sono presenti più layer detti convoluzionali che eseguono operazioni sui dati per estrarre caratteristiche applicando una serie di filtri;
- **Layer di output:** L'ultimo livello della rete è un layer di classificazione che si occupa di classificare l'output ottenuto dai layer convoluzionali.

I layer nascosti contengono vari livelli al loro interno, tra i più diffusi abbiamo il livello convoluzionale, il pooling e la funzione ReLU. Il livello convoluzionale è quello più importante ed è quello per cui si differenzia una CNN da una rete neurale classica. Questo livello applica all'input una serie di filtri o kernel<sup>4</sup>, ciascuno di questi estrapola determinate feature nei dati. Ad esempio, processando immagini, il livello convolutivo applica filtri particolari per mettere in risalto o attivare feature come bordi, linee verticali, orizzontali ecc. Ad ogni convoluzione si applica la funzione di attivazione ReLU vista in precedenza che trasmette al layer successivo solo le feature attivate in precedenza [19]. Nella Figura 2.14 viene mostrato l'effetto di un filtro per evidenziare le linee verticali.

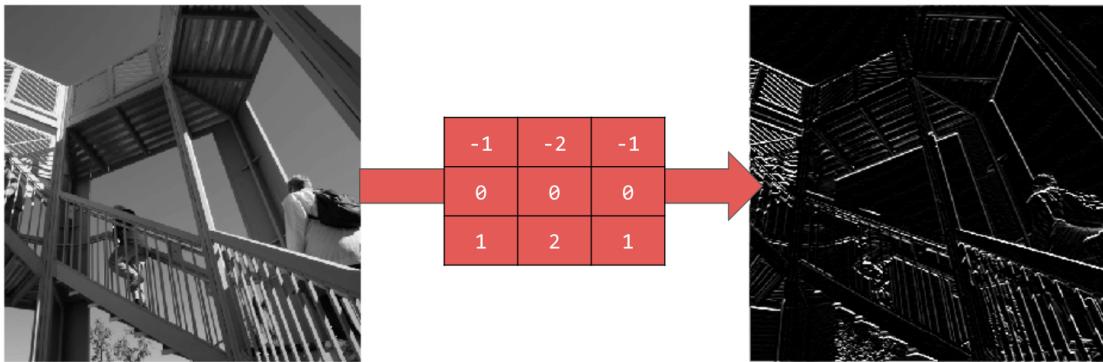


**Figura 2.14:** Filtro convoluzionale per linee verticali

Nella Figura 2.15 invece, vengono messe in risalto le linee orizzontali.

---

<sup>4</sup>Filtro usato nelle reti neurali convoluzionali per estrarre le caratteristiche di un'immagine

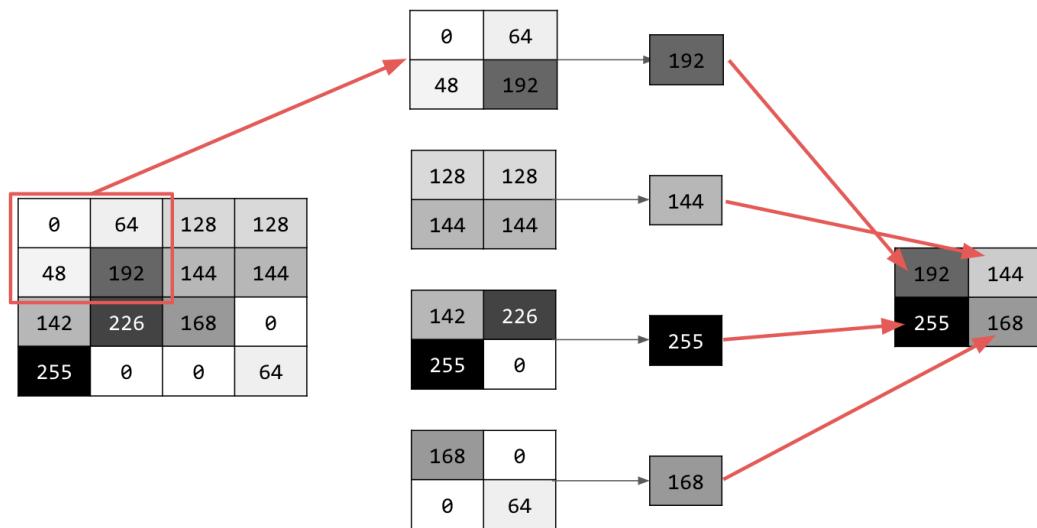


**Figura 2.15:** Filtro convoluzionale per linee orizzontali

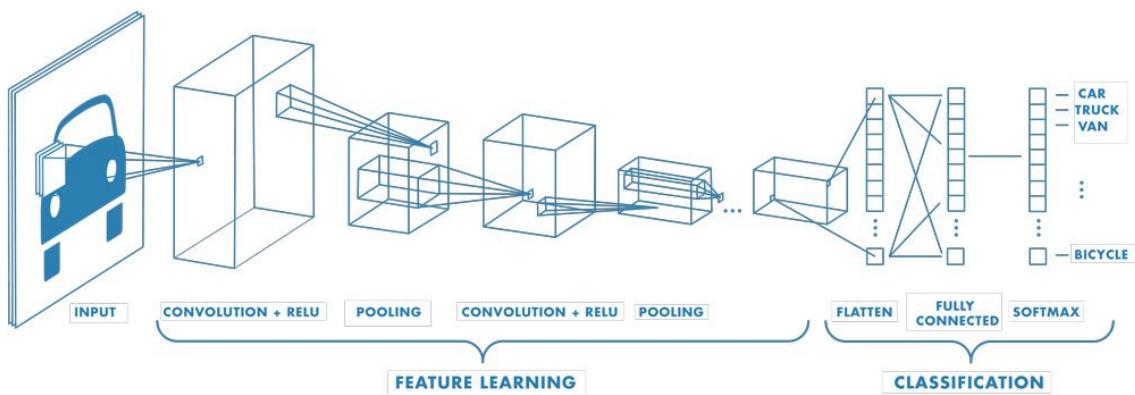
L’ultima operazione all’interno degli strati nascosti è il pooling. La mappatura ottenuta dalla funzione ReLu viene trasmessa al layer di pooling che esegue un downsampling, ovvero una riduzione di dimensionalità, riducendo il numero di parametri mantenendo però le caratteristiche più importanti ricavate in precedenza. Similmente al livello convoluzionale, il layer di pooling applica filtri che non modificano le caratteristiche dell’immagine ma la riducono di dimensione. L’operazione di pooling aiuta a ridurre la complessità e a migliorare le performance. Esistono due principali tipologie di pooling:

- **Pooling massimo:** Il filtro seleziona i valori massimi per inviarli al layer successivo;
- **Pooling medio:** Il filtro calcola il valore medio dell’input da inviare al prossimo layer [19].

In questo esperimento verrà utilizzato il pooling massimo che vediamo in dettaglio nella Figura 2.16.

**Figura 2.16:** Operazione di pooling

Dopo aver estratto le feature nella fase convolutiva, l'architettura passa alla classificazione. In questa fase principalmente si utilizzano layer completamente connessi ovvero strati dove ogni nodo o neurone è connesso con tutti i nodi precedenti e successivi. In questa fase spesso si utilizza la funzione di attivazione softmax per distribuire le probabilità per tutte le classi del dataset in analisi. Nel caso di Fashion-MNIST verranno prodotte dieci probabilità, una per ogni classe di abbigliamento. Nella Figura 2.17 vediamo un'architettura classica di una CNN dove abbiamo due strati convolutivi con i relativi pooling e infine abbiamo la fase di classificazioni.

**Figura 2.17:** Rete neurale convoluzionale

## 2.2 Stato dell’Arte

In questa sezione vengono analizzati i lavori svolti in letteratura nell’ambito della classificazione delle immagini sia su Fashion-MNIST che sul CIFAR-10.

### 2.2.1 Classificazione di Fashion-MNIST

#### Reti neurali convoluzionali applicate al Fashion-MNIST

Nel 2020 Kadam et al. [20] pubblicano un paper dove confrontano diverse architetture di CNN sui dataset MNIST e Fashion-MNIST. Prendiamo in considerazione solo i test effettuati su Fashion-MNIST che andremo ad analizzare in seguito. Nelle tabelle 2.1 e 2.2 vengono mostrate le architetture utilizzate nel paper citato in precedenza.

Model 1	Model 2	Model 3
2 FC <sup>5</sup>	2 CL <sup>6</sup> e 2 FC	3 CL e 2 FC
INPUT:28×28×1 OUTPUT:FC 10 classi	INPUT:28×28×1 OUTPUT:FC 10 classi	INPUT:28×28×1 OUTPUT:FC 10 classi
FC:128 Hidden Neurons	CONV2D:2×2,64 filters POOL:2×2 size DROPOUT: = 0.25 CONV2D :2×2,64 filters DROPOUT: = 0.25 FC:64 Hidden Neurons DROPOUT: = 0.25	CONV2D:2×2,64 filters POOL:2×2 size DROPOUT: = 0.25 CONV2D :2×2,64 filters POOL:2×2 size DROPOUT: = 0.25 CONV2D :2×2,64 filters DROPOUT: = 0.25 FC:64 Hidden Neurons DROPOUT: = 0.25

**Tabella 2.1:** Architetture Analizzate

Model 4	Model 5
4 CL e 2 FC	4 CL e 2 FC
INPUT:28×28×1 OUTPUT:FC 10 classi	INPUT:28×28×1 OUTPUT:FC 10 classi
CONV2D:2×2,64 filters POOL:2×2 size DROPOUT: = 0.25 CONV2D :2×2,64 filters POOL:2×2 size DROPOUT: = 0.25 CONV2D:2×2,64 filters POOL:2×2 size DROPOUT: = 0.25 CONV2D :2×2,64 filters DROPOUT: = 0.25 FC:64 Hidden Neurons DROPOUT: = 0.25	CONV2D:3×3,32 filters CONV2D:3×3,32 filters POOL:2×2 size DROPOUT: = 0.25 CONV2D:3×3,64 filters CONV2D:3×3,64 filters POOL:2x2 size DROPOUT: = 0.25 FC:512 Hidden Neurons DROPOUT: = 0.5

**Tabella 2.2:** Architetture Analizzate

Tutti i test sono stati condotti su Google colaboratory<sup>7</sup> e quindi utilizzando le risorse di calcolo offerte da Google. Per il primo modello di CNN sono stati effettuati diversi test, in particolare sono stati testati i diversi metodi di ottimizzazione come Adam, SGD, RMS ecc. In seguito, vediamo i risultati raggiunti nella Tabella 2.3 applicando diversi optimizer al dataset Fashion-MNIST.

<sup>5</sup>Fully connected: Layer completamente connesso

<sup>6</sup>Layer Convoluzionale

<sup>7</sup>Notebook python online <https://colab.research.google.com/>

Optimizer	Epoche	Train acc.	Test acc.
SGD	50	87,99%	86,53%
RMSprop	50	94,25%	88,76%
Adagrad	50	97,04%	89,62%
Adadelta	60	94,72%	89,00%
Adam	50	97,38%	88,59%

**Tabella 2.3:** Confronto metodi di ottimizzazione

Continuando sempre sul primo modello è stato effettuato un altro test dove viene modificata solo la batch size dei vari training per monitorarne l’accuratezza. Quest’ultimo test è stato effettuato utilizzando come funzione di attivazione la Sigmoid e come optimizer invece Adam. I risultati sono mostrati nella Tabella 2.4.

Batch size	Train acc.	Test acc.
16	93,88%	88,39%
32	94,41%	88,74%
64	94,73%	89,30%
100	94,52%	89,48%
120	94,56%	88,39%
140	94,50%	88,05%
150	94,60%	88,38%
160	96,63%	89,05%
200	94,48%	88,82%

**Tabella 2.4:** Accuratezze al variare della batch size

Come dimostrato da Kadam et al. [20] aumentando la batch size non si ha un sostanziale incremento sull’accuratezza ma si sovraccarica solo la rete richiedendo maggiori risorse di calcolo.

Nella seconda architettura è stato raggiunto il miglior risultato utilizzando un batch size di 128, Softmax come funzione di attivazione e Adam come optimizer. L’accuratezza del training e del testing sono rispettivamente 92,02% e 92,76% addestrando il modello per 50 epoches.

Per il terzo modello i migliori risultati ottenuti sono 93,03% e 93,56% rispettivamente per training e testing. I parametri ottimali utilizzati per raggiungere questi numeri sono gli stessi dell’architettura precedente.

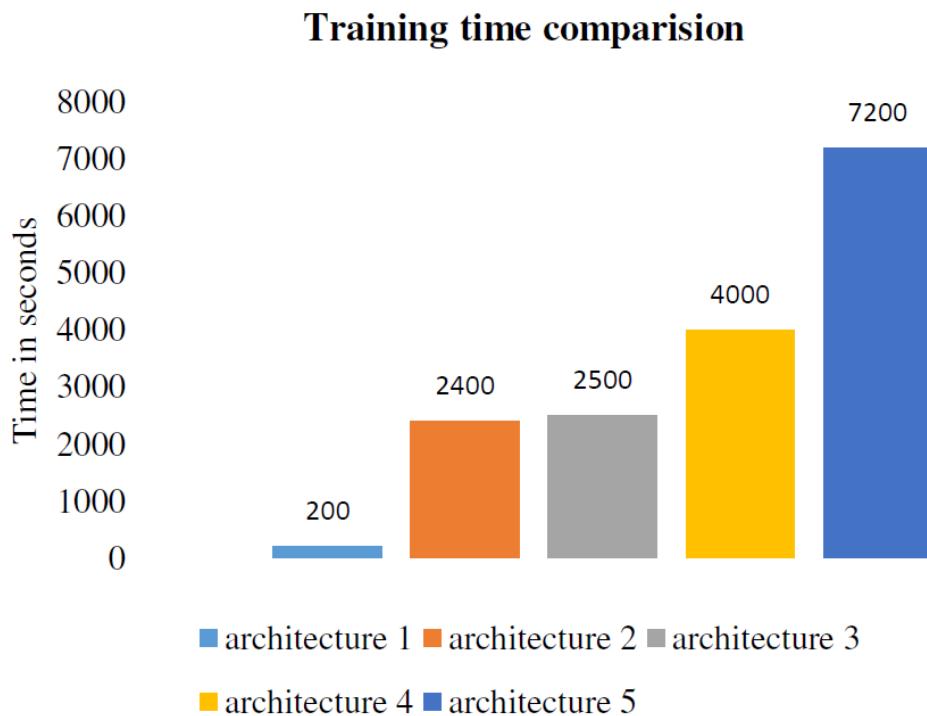
I risultati per la quarta architettura invece sono 93,17% per il training e 92,94% per il testing utilizzando ancora una volta i parametri citati precedentemente.

Per l’ultimo modello preso in analisi è stato modificato l’optimizer utilizzando RMSprop e ottenendo il 92,67% e il 92,86% rispettivamente per il training e il testing. La Tabella 2.5 riepiloga tutti i risultati migliori per ogni modello.

	Train acc.	Test acc.
Model 1	99,60%	89,65%
Model 2	92,02%	92,76%
Model 3	93,09%	93,56%
Model 4	93,17%	92,94%
Model 5	92,67%	92,86%

**Tabella 2.5:** Comparazione dei risultati ottenuti

Nella Figura 2.18 viene mostrato un altro fattore importante per gli algoritmi di ML, il tempo di addestramento. In questo caso vediamo una crescita tra le varie architetture a causa del numero di layer presenti e dalla dimensione dei filtri.



**Figura 2.18:** Grafico tempi di addestramento

In conclusione, l'analisi appena approfondita attesta il terzo modello come il più performante su Fashion-MNIST per quanto riguarda l'accuratezza. Questa architettura, come specificato nella Tabella 2.1, è formata da tre layer convoluzionali con filtri 2x2 e due layer completamente connessi. Un altro elemento sottolineato nel paper riguarda i tempi di training che aumentano con l'aumentare dei layer convoluzionali e soprattutto aumentando la dimensione dei filtri.

### Analisi comparativa degli algoritmi di ML

Il team di sviluppo di Fashion-MNIST ha realizzato un sistema di benchmarking automatico basato su scikit-learn<sup>8</sup> dove vengono confrontati 129 algoritmi di ML ma non di deep learning. I risultati sono stati documentati in parte sul paper di presentazione del dataset [7] ed il resto in un'applicazione cloud su AWS<sup>9</sup>. In questa sezione analizziamo solo i migliori risultati ottenuti da SVM, Random Forest e ANN in termini di accuracy. L'algoritmo che ha raggiunto l'accuratezza migliore è stata

<sup>8</sup>Libreria python per il Machine Learning

<sup>9</sup><http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/>

la macchina a vettori di supporto seguito dal Random Forest ed infine ANN. Nella tabella che segue vengono riassunti i test effettuati con le accuratezze, i parametri ed i relativi tempi di addestramento.

Algoritmo	Parametri	Test acc.	Tempo di training
SVM	"C":10,"kernel":"poly"	89,70%	1:12:39
Random Forest	criterion:"entropy" max-depth:50 n-estimators:100	87,90%	0:08:39
ANN	activation:"relu" hidden-layer-sizes:100	87,70%	0:16:03

**Tabella 2.6:** Algoritmi di ML confrontati

Nei risultati presentati nella Tabella 2.6 notiamo che l’algoritmo SVM risulta più performante in termini di accuratezza ma presenta tempi di addestramento molto superiori rispetto ai restanti. Random Forest e ANN impiegano meno tempo nella fase di training mantenendo comunque un valore di accuratezza accettabile.

## 2.2.2 CNN con filter size differenti

In una pubblicazione di Khanday et al. [21] vengono osservati gli effetti delle reti neurali convoluzionali utilizzando diverse dimensioni dei filtri. Lo studio è stato effettuato sui dataset Fashion-MNIST e CIFAR-10 utilizzando una rete formata da 13 layer. L’esperimento consiste nel testare la rete utilizzando filtri da 3x3, 5x5 e 7x7 lasciando invariato tutto il resto. L’accuratezza sul CIFAR-10 notiamo subito che è maggiore utilizzando un kernel 3x3 e diminuisce utilizzando filtri di taglia maggiore come illustrato nella Tabella 2.7.

Filter size	Train acc.	Test acc.
3x3	94,26%	73,04%
5x5	92,32%	72,97%
7x7	87,72%	63,50%

**Tabella 2.7:** Risultati comparazione con filtri differenti

Il secondo esperimento è stato svolto su Fashion-MNIST utilizzando le stesse configurazioni. Anche in questo caso il filtro 3x3 ha performance, seppur minime, migliori rispetto agli altri kernel di dimensione maggiore. I risultati sono mostrati di seguito.

Filter size	Train acc.	Test acc.
3x3	92,90%	92,68%
5x5	92,60%	92,64%
7x7	91,80%	91,10%

**Tabella 2.8:** Risultati comparazione con filtri differenti

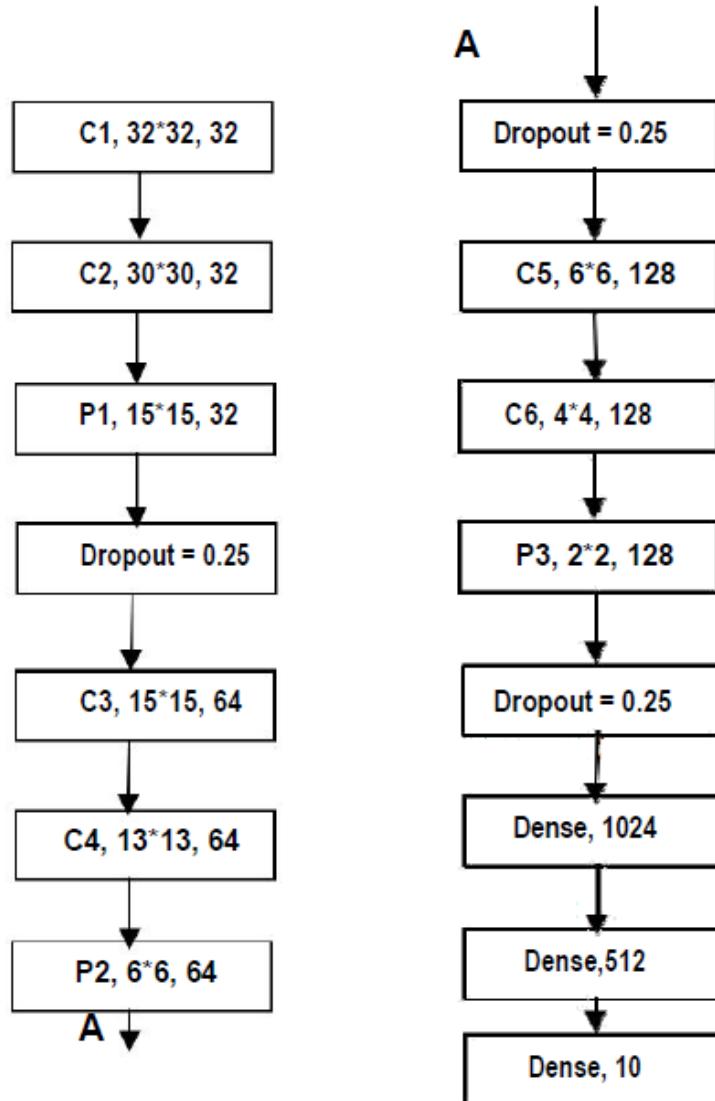
In conclusione, questo paper dimostra che per dataset con queste caratteristiche e dimensioni i filtri di dimensioni minore riescono a estrapolare più dettagli dai dati e quindi ad avere un accuratezze maggiori.

### 2.2.3 Classificazione di CIFAR-10

#### Reti neurali convoluzionali applicate al CIFAR-10

Nel 2018 Doon et al. [22] implementano una rete neurale convoluzionale che successivamente testano sul dataset CIFAR-10. Nel loro lavoro si può vedere che hanno utilizzato una CNN formata da sei layer convoluzionali, tre di pooling e tre completamente connessi. Le funzioni di attivazioni utilizzate sono ReLU per i strati nascosti e softmax per il layer di output. Per la ridurre l’overfitting è stato usato come

metodo di regolarizzazione il Dropout. Nella Figura 2.19 viene mostrata l'architettura proposta.



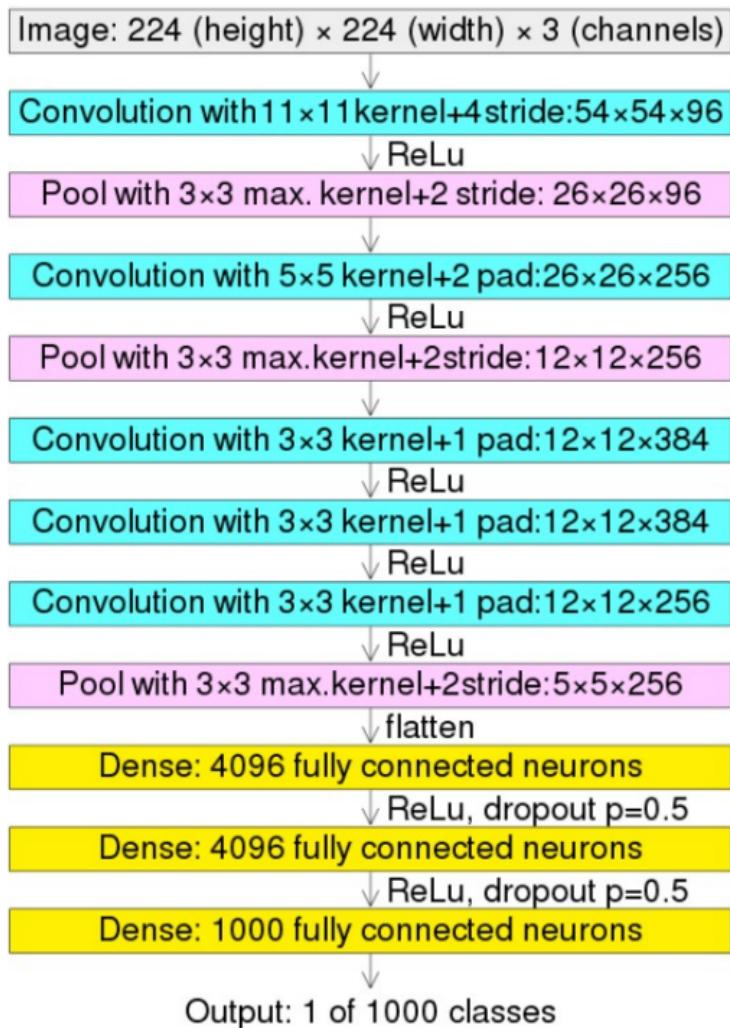
**Figura 2.19:** Architettura CNN implementata

I risultati ottenuti addestrando la rete per 100 epochhe e con un batch size di 256 sono il 90,00% e l'87,57%, rispettivamente per il training e il test.

Zhang nel 2021 pubblica un paper [23] dove invece addestra alcune delle CNN più note nell'ambito del deep learning ovvero AlexNet, LeNet-5 e VGG-NET. AlexNet è una CNN progettata da Alex Krizhevsky et al. [24] inizialmente usata su ImageNet<sup>10</sup> e poi diffusa su altri dataset. La versione utilizzata da Zhang è stata modificata ad hoc

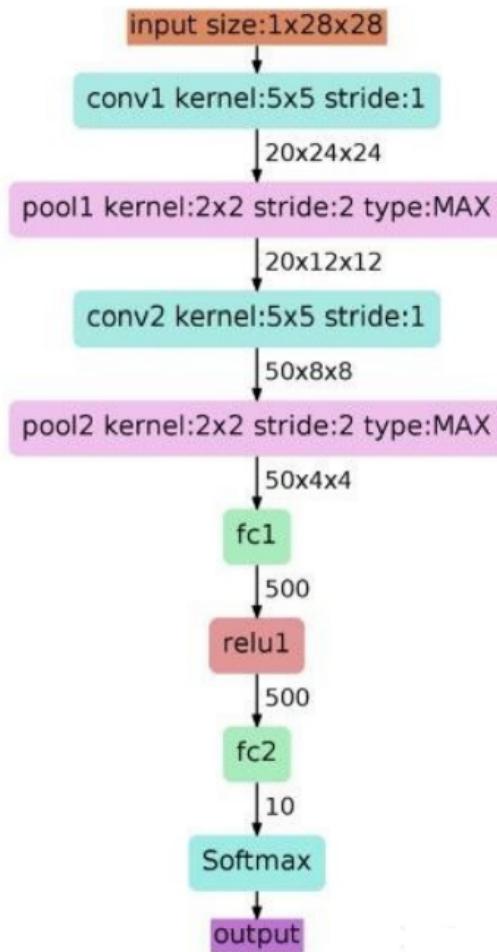
<sup>10</sup>Dataset di immagini ad alta risoluzione con 22,000 categorie diverse.

in modo da renderla compatibile sul CIFAR-10. Le modifiche effettuate riguardano principalmente la riduzione dell’input size da 224x224 a 32x32 e la riduzione della dimensione dei filtri che passano da 11x11 a 3x3. Il risultato raggiunto con questo modello dopo 100 epoche è del 90,00% sul testing. Nella Figura 2.20 è mostrata il modello originale di AlexNet.



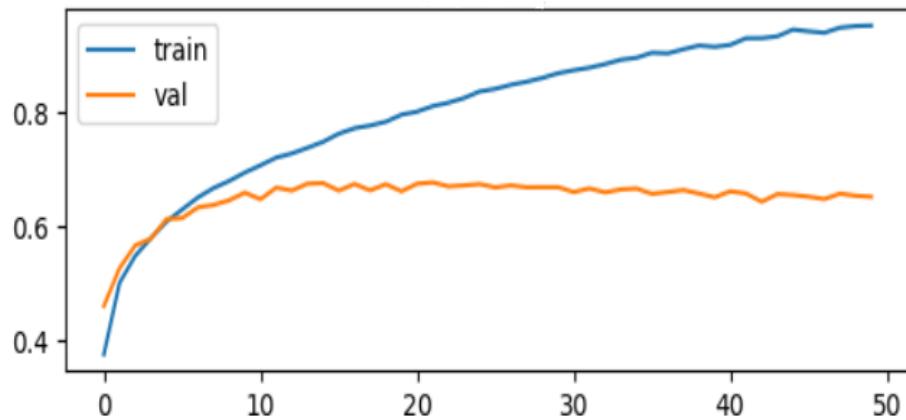
**Figura 2.20:** Architettura AlexNet su ImageNet

LeNet-5 è una rete ideata da LeCun et al. [25] per classificare il dataset MNIST. Anch’essa è stata modificata da Zhang in modo da accettare un input di 32x32x3 rispetto ai 28x28x1 di MNIST. L’architettura di LeNet-5 non è complessa come AlexNet, ad esempio, e la vediamo in dettaglio nella figura 2.21,



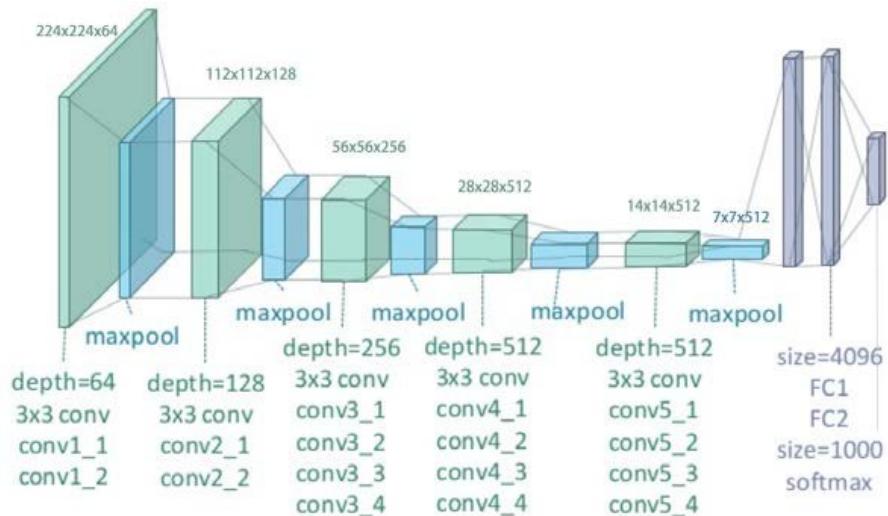
**Figura 2.21:** Architettura LeNet su MNIST

La rete ha raggiunto ottimi risultati nel training arrivando circa al 90,00% ma nel testing dopo appena 10 epoche l'accuratezza si è fermata al 60,00% come vediamo nel grafico successivo. Questo dimostra che la rete seppur raggiunge ottimi risultati con MNIST, in ambienti più complessi soffre molto di overfitting.



**Figura 2.22:** Accuratezza del training e test di LeNet-5

L'ultima rete analizzata da Zhang è la VGG-NET progettata da Simonyan e Zisserman [26] presso il Visual Geometry Group (VGG) all'Università di Oxford. VGG-NET ha molte configurazioni possibili in base ai layer che la formano, quella testata da Zhang è la VGG-19 (Figura 2.23) con appunto 19 layer.



**Figura 2.23:** Architettura VGG-19

Dopo 50 epoche di training sul CIFAR-10, VGG-19 ha raggiunto circa l'80,00% di accuratezza sul testing e circa il 90,00% sul training evidenziando anche in questo caso problemi di overfitting.

Il miglior risultato ottenuto da questo esperimento è stato quindi quello ottenuto da AlexNet con il 90,00% di accuratezza sul testing

Per concludere, Zhang mostra la compatibilità tra vari modelli di CNN con diversi dataset e afferma che è possibile combinare diversi modelli per creare nuove reti neurali ancora più performanti. Sottolinea inoltre che le reti testate in questo esperimento sono modelli molto complessi e richiedono necessariamente risorse come le GPU per addestrarle.

#### 2.2.4 Limiti Stato dell’Arte

Analizzando i lavori svolti in letteratura e mostrati nella Sezione 2.2, possiamo notare come gli algoritmi vengono valutati solo mediante l’accuratezza. Questo approccio in alcuni casi potrebbe non essere sufficiente, ad esempio, utilizzando dataset sbilanciati<sup>11</sup>, l’accuratezza può mostrare valori non veritieri falsando l’effettiva affidabilità di un modello. In questo lavoro di tesi vengono analizzate diverse metriche oltre l’accuratezza come F-score, loss, oob-error e matrici di confusione. Un ulteriore limite incontrato in letteratura riguarda la mancanza di confronti di algoritmi di ML tra F-MNIST e CIFAR-10 in modo approfondito.

---

<sup>11</sup>Il numero degli elementi delle classi non sono distribuiti in modo uniforme.

# CAPITOLO 3

---

## Metodologia di ricerca

---

### 3.1 Obiettivo della ricerca

L'obiettivo della ricerca è orientato alla comparazione di diversi algoritmi di Machine Learning sui dataset Fashion-MNIST e CIFAR-10. La comparazione tratterà le diverse metriche di valutazione di un algoritmo andando oltre alla sola accuratezza di un modello come fatto da gran parte dei lavori presenti in letteratura.

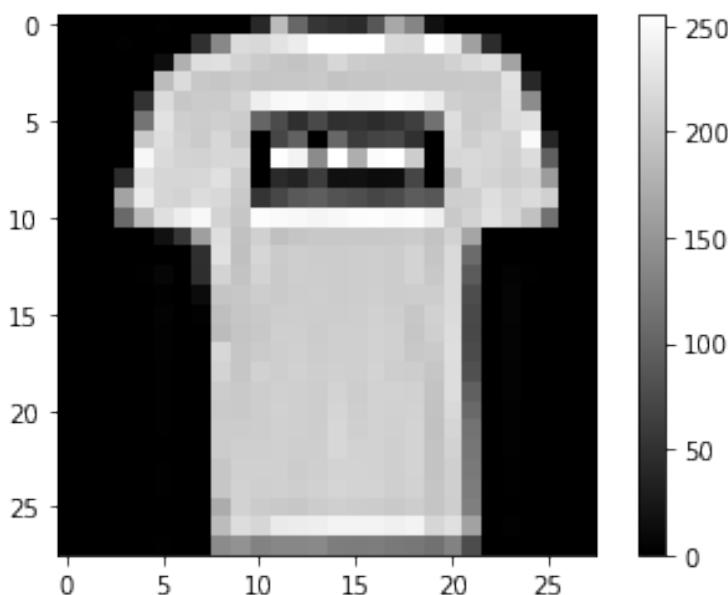
### 3.2 Materiali

L'implementazione degli algoritmi presentati in precedenza è stata supportata da librerie Python come Tensorflow, Keras e scikit-learn. La gestione dei dataset e la visualizzazione di grafici è stata affidata alle librerie Pandas, Matplot e numPy. La scelta dell'ambiente di sviluppo è ricaduta su Google colab per diversi motivi: in primis perché offre risorse di calcolo notevoli per l'esecuzione del codice Python e quindi utile a velocizzare l'addestramento dei modelli di ML, in secundis per la praticità essendo un ambiente già pronto e con le librerie già installate.

## 3.3 Metodi

### 3.3.1 Caricamento e preparazione dei dati

Nella prima fase della ricerca sono stati caricati i dataset Fashion-MNIST e CIFAR-10 direttamente dalla libreria Tensorflow, dopodiché tramite la libreria matplotlib è stata controllata l'integrità dei dati stampando un campione random come mostrato in Figura 3.1.



**Figura 3.1:** Visualizzazione campione di Fashion-MNIST

I dataset dopo il caricamento sono stati suddivisi in due dataset più piccoli rispettivamente per il training e per il test.

Fashion-MNIST partendo da 70,000 record è stato diviso in 60,000 figure per il training e 10,000 per il testing.

Il dataset CIFAR-10 invece, è stato diviso in 50,000 e 10,000 immagini rispettivamente per train e test.

L'ultima operazione effettuata sui dataset è stata la normalizzazione. Questa fase è molto importante perché migliora sia la velocità che l'accuratezza di un modello di ML. L'idea di base è quella di ridurre l'intervallo delle immagini presenti nei dataset da [0,255] a [0,1]. Algoritmi come SVM, reti neurali e CNN dipendono molto dalla normalizzazione.

### 3.3.2 Costruzione dei modelli

In questa sezione vengono specificate le architetture e i parametri utilizzati per la costruzione dei modelli presi in analisi. La scelta dei parametri è stata supportata in parte dalle scelte effettuate in letteratura e approfondite nella Sezione 2.2.

#### SVM

La macchina a vettori di supporto è stata addestrata utilizzando i seguenti parametri:

- **Kernel:** 'rbf'
- **Coefficiente di regolarizzazione (C):** 1
- **Gamma:** 'scale'

Il **Kernel** specifica il tipo di separatore dei dati, il **coefficiente C** indica il parametro per il controllo dell'errore ed infine **Gamma** che rappresenta la curvatura dell'iperpiano.

#### Random Forest

Il modello scelto per il random Forest si compone con i seguenti parametri:

- **n\_estimators:** 128;
- **criterion:** 'entropy';
- **max\_depth:** 20;
- **n\_job:** -1;

In particolare "**n\_estimators**" indica il numero degli alberi presenti nella foresta, "**criterion**" indica la funzione per misurare la qualità di split, "**max\_depth**" è la profondità massima di un albero ed infine "**n\_job**" che indica i processi da eseguire in parallelo (-1 per utilizzare tutte le risorse disponibili).

## Reti Neurali

L'architettura progettata per le reti neurali è formata da quattro layer come mostrato nel codice seguente.

```

1 model = tf.keras.Sequential([
2     tf.keras.layers.Flatten(input_shape=(28, 28)),
3     tf.keras.layers.Dense(128, activation='relu'),
4     tf.keras.layers.Dropout(0.2),
5     tf.keras.layers.Dense(10, activation='softmax')
6 ])

```

In particolare:

- **Flatten:** trasforma l'input da una matrice (28x28 nel caso di F-MNIST) ad un array a singola dimensione;
- **Dense:** FC da 128 neuroni e funzione di attivazione "ReLU";
- **Dropout:** 20% per la riduzione dell'overfitting;
- **Dense:** FC da 10 neuroni e funzione di attivazione "Softmax".

I parametri secondari scelti sono:

- **Optimizer:** "Adam";
- **Loss Function:** "Sparse Categorical Crossentropy".

## Reti Neurali Convoluzionali

Per le CNN essendo più complesse e con un'ampia scelta di parametri, si è optato per la progettazione di diverse architetture, in totale tre. Nella Tabella 3.1 vengono mostrati i layer che compongono i tre modelli implementati in ordine di complessità.

Model 1	Model 2	Model 3
CONV2D:2×2,64 filters POOL:2×2 size DROPOUT: = 0.2	CONV2D:3×3,32 filters POOL:2×2 size DROPOUT: = 0.2	CONV2D:3×3,32 filters POOL:2×2 size DROPOUT: = 0.25
CONV2D :2×2,64 filters POOL:2x2 size FLATTEN	CONV2D :3×3,64 filters POOL: 2x2 size CONV2D :3×3,64 filters FLATTEN	CONV2D :3×3,64 filters POOL:2×2 size DROPOUT: = 0.25
FC:10 Neuroni	FC: 64 Neuroni FC: 10 Neuroni	CONV2D :3×3, 128 filters DROPOUT: = 0.4 FLATTEN FC: 128 Neuroni DROPOUT: = 0.3 FC: 10 Neuroni

**Tabella 3.1:** Architetture Analizzate

Tutte i modelli visti precedentemente utilizzano i seguenti parametri:

- **Optimizer:** "Adam";
- **Funzione di attivazione:** "ReLU" per i layer interni e "Softmax" per l'ultimo layer;
- **Loss Function:** "Sparse Categorical Crossentropy".

## 3.4 Metodi e Metriche di Valutazione

Per valutare le prestazioni e l'affidabilità dei modelli analizzati sono state utilizzate diverse tecniche e metriche. Alcune metriche sono specifiche per alcuni modelli di ML mentre altre sono comuni a tutti gli algoritmi.

Le metriche utilizzate sono le seguenti:

- **Accuracy:** percentuale delle predizioni corrette;

- **Loss:** somma degli errori che il modello ha effettuato;
- **Matrice di confusione:** rappresentazione grafica degli errori commessi dal modello;
- **OOB-error:** utilizzata solo nel Random Forest, misura l'errore dell'algoritmo tramite l'out-of-bag dataset, ovvero la collezione di record non presenti in nessun sottosinsieme di bootstrap;
- **F-score:** metrica che unisce la "Precision" e la "Recall" tramite una media armonica<sup>1</sup>. La Precision indica il rapporto tra veri positivi e tutti i positivi. Utilizzando la classe delle auto del CIFAR-10 ad esempio, la Precision indica il numero di immagini classificate come auto che risultano essere correttamente auto. La Recall invece, indica le immagini che rappresentano auto classificate correttamente come auto.

---

<sup>1</sup>rappresenta il reciproco della media aritmetica dei reciproci

# CAPITOLO 4

---

## Risultati

---

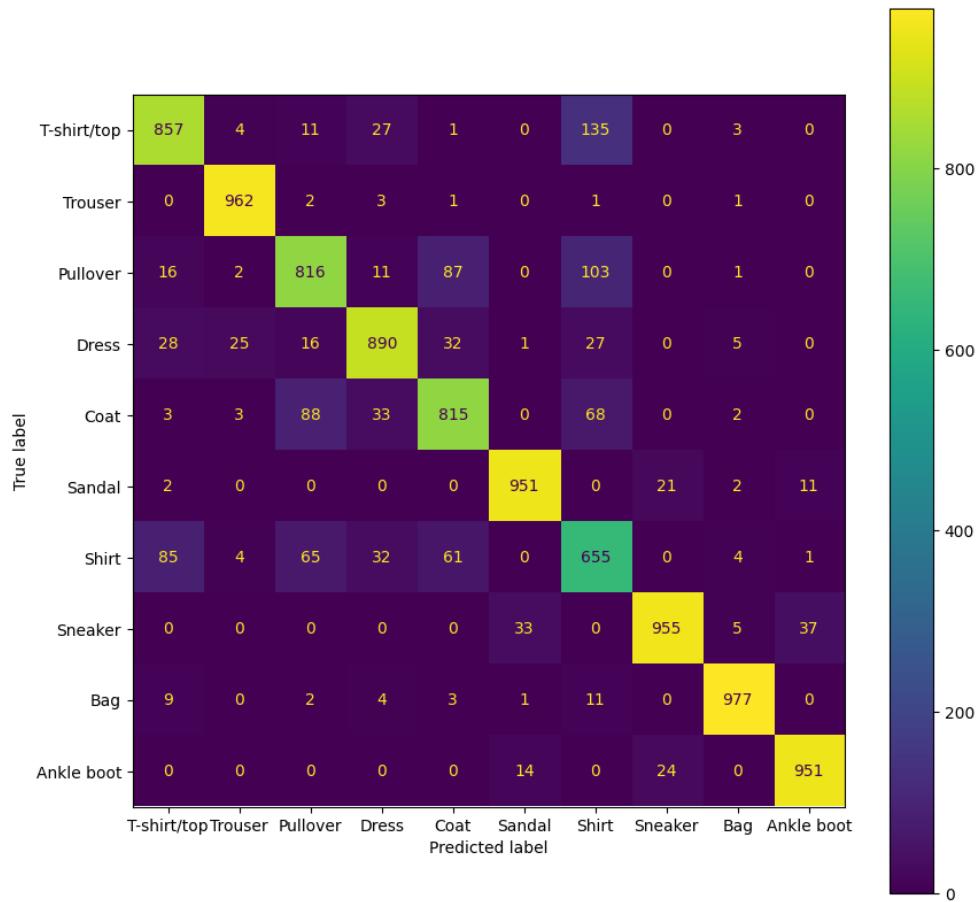
In questo capitolo verranno mostrati tutti i risultati raggiunti con gli algoritmi presentati precedentemente dopo l’addestramento su Fashion-MNIST e CIFAR-10.

### 4.1 Risultati classificazione con Support Vector Machine

#### 4.1.1 Test su Fashion-MNIST

Il risultato ottenuto dalla Support Vector Machine su Fashion-MNIST è del **88,29%** di accuratezza.

La matrice di confusione inerente mostra come il tasso di errore più alto si verifica nella classe 6 (Camicia) che viene classificata come t-shirt o maglione in alcuni casi. Nella Figura 4.1 la matrice di confusione per F-MNIST.


**Figura 4.1:** Matrice di confusione di SVM su F-MINST

Anche la metrica F-score mostra la difficoltà nel classificare la classe 6 come vediamo nella Tabella 4.1.

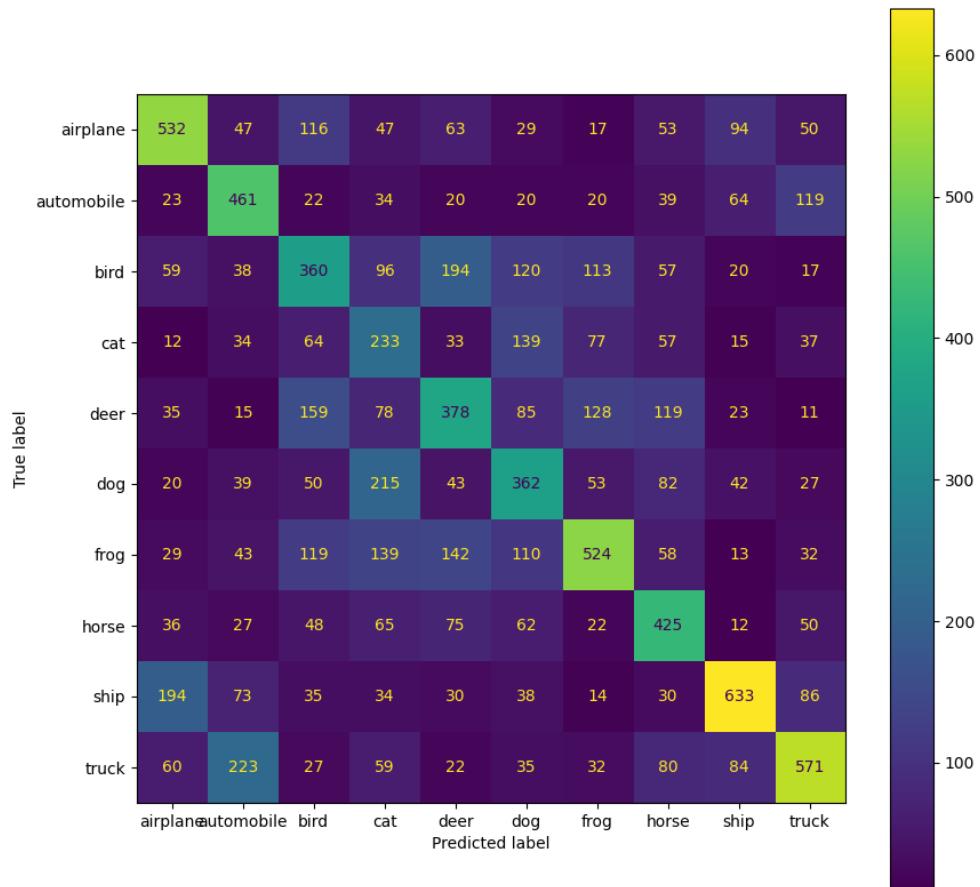
Classe	0	1	2	3	4	5	6	7	8	9
F-score	0,84	0,98	0,80	0,88	0,81	0,96	0,69	0,94	0,97	0,96

**Tabella 4.1:** Valori F-score sul F-MNIST con SVM

### 4.1.2 Test su CIFAR-10

In questo test la dimensione del dataset di training è stata ridotta per ridurre il tempo di addestramento. Sul CIFAR-10 l'accuratezza della SVM scende fino a **44,79%**.

Osservando sia la matrice di confusione in Figura 4.2 e sia la Tabella 4.2 relativa ad F-score si può osservare un problema di classificazione in tutte le classi. Solo la classe 8 (Nave) riesce a superare 600 predizioni corrette su 1000.



**Figura 4.2:** Matrice di confusione di SVM sul CIFAR-10

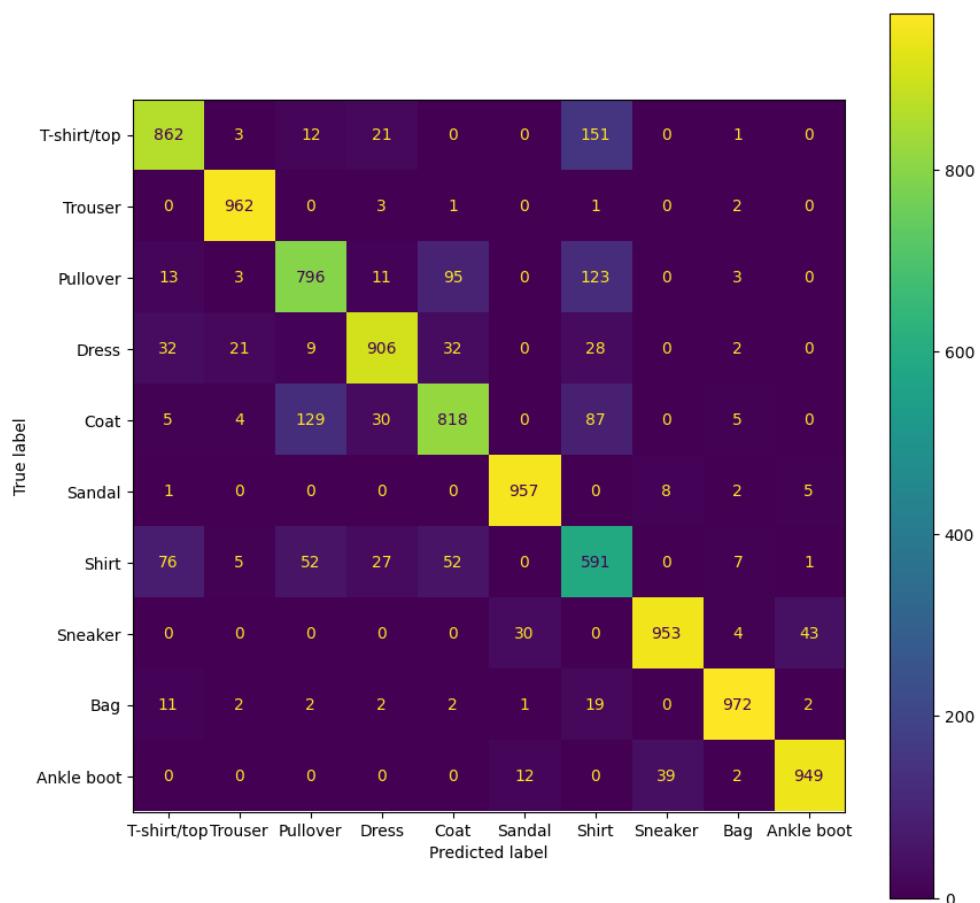
Classe	0	1	2	3	4	5	6	7	8	9
F-score	0,52	0,51	0,35	0,27	0,37	0,37	0,47	0,47	0,58	0,52

**Tabella 4.2:** Valori F-score sul CIFAR-10 con SVM

## 4.2 Risultati classificazione con Random Forest

### 4.2.1 Test su Fashion-MNIST

L'algoritmo Random Forest ha raggiunto un valore di accuratezza pari a **87,66%** nella fase di test. La matrice confusionale nella Figura 4.3 mostra le predizioni corrette per ogni classe su 1000 campioni.



**Figura 4.3:** Matrice di confusione di Random Forest su F-MINST

Insieme ai valori F-score mostrati nella Tabella 4.3 notiamo che la categoria con più errori di classificazione è la classe 6 (Camicia) con 591 predizioni corrette su 1000.

Classe	0	1	2	3	4	5	6	7	8	9
F-score	0,84	0,98	0,78	0,89	0,79	0,97	0,65	0,94	0,97	0,95

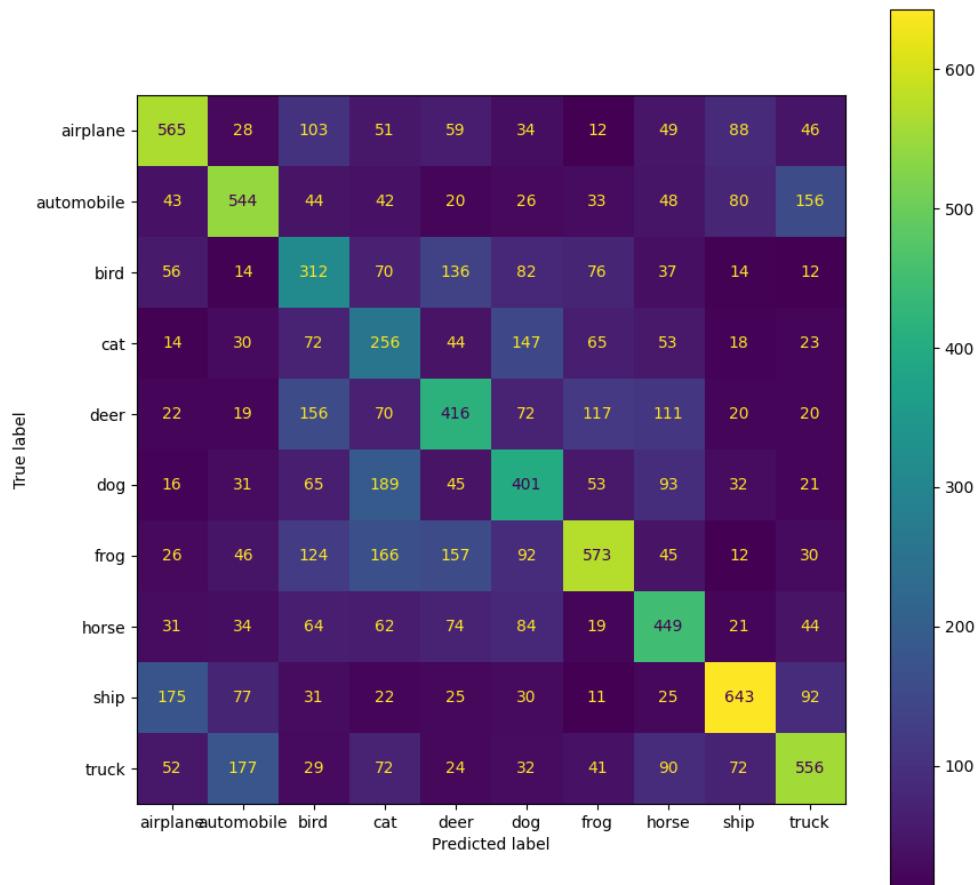
**Tabella 4.3:** Valori F-score sul F-MNIST con Random Forest

Oltre alle metriche di accuratezza ed F-score, sul Random Forest è stato valutato anche l'**oob-error**. Con il dataset F-MNIST l'errore out-of-bag ha un valore di circa **0.11**.

### 4.2.2 Test su CIFAR-10

L'accuratezza ottenuta dall'algoritmo Random Forest sul Cifar-10 è stata pari al 47,15% sul testing.

Nella matrice di confusione mostrata nella Figura 4.4 evidenziamo le problematiche di classificazione con gran parte degli errori che si concentrano nelle classi centrali del dataset come gatto, cane, uccello ecc.



**Figura 4.4:** Matrice di confusione di Random Forest su CIFAR

La stessa problematica viene mostrata anche dalla metrica F-Score, mostrata nella Tabella 4.4, dove solo la classe 8 (Nave) riesce a raggiungere **0,60**.

Classe	0	1	2	3	4	5	6	7	8	9
F-score	0,56	0,53	0,34	0,30	0,41	0,41	0,50	0,48	0,60	0,52

**Tabella 4.4:** Valori F-score sul CIFAR-10 con Random Forest

L'out-of-bag error per questa classificazione è risultato essere circa **0,57**

## 4.3 Risultati classificazione con Reti Neurali

### 4.3.1 Test su Fashion-MNIST

La rete neurale artificiale implementata è stata addestrata su Fashion-MNIST per 30 epoche ed infine valutata sul dataset di test. I risultati raggiunti in termini di accuratezza e loss sono mostrati nella Tabella 4.5.

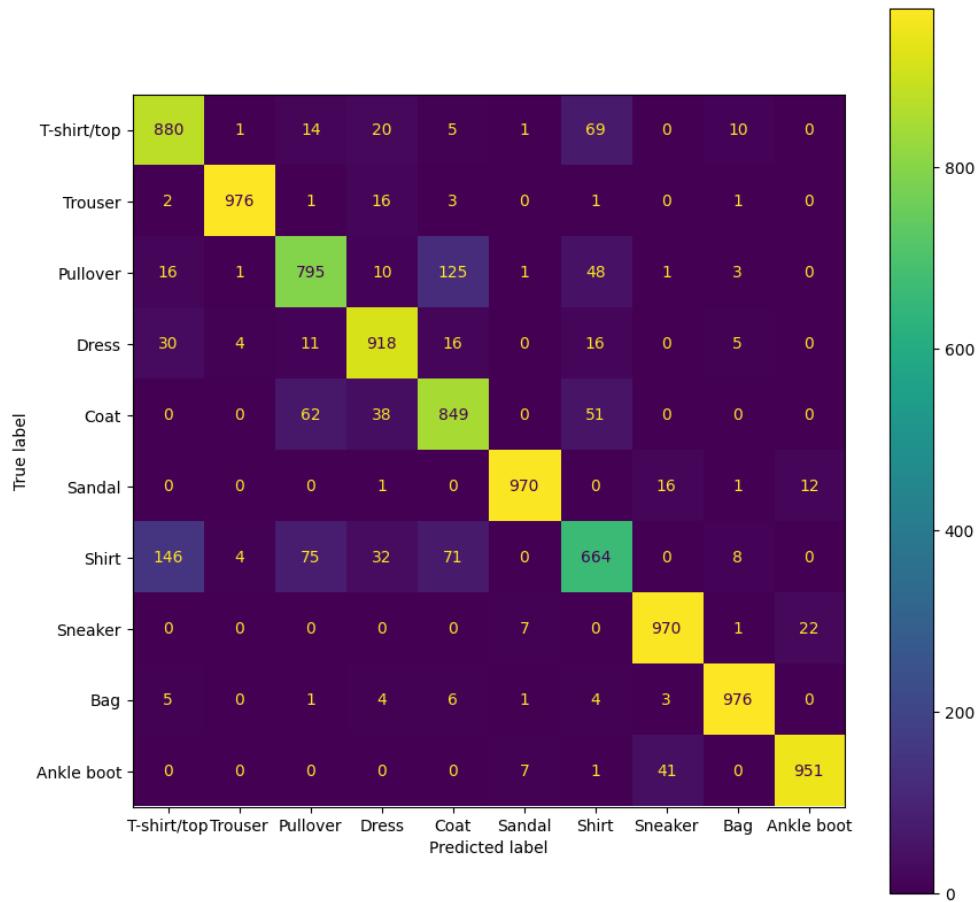
Train accuracy	Train loss	Test accuracy	Test loss
94,67%	0.14	89,49%	0,38

**Tabella 4.5:** Accuracy e loss su F-MNIST con le reti neurali

Si può notare che la rete nonostante la regolarizzazione tramite il Dropout soffre ancora di Overfitting.

La matrice di confusione in Figura 4.5 mostra le predizioni corrette per ogni classe su 1000 campioni. Possiamo notare che le classi riconosciute con maggior difficoltà sono:

- **Classe 2 (Pullover):** 795/1000
- **Classe 6 (Camicia):** 664/1000

**Figura 4.5:** Matrice di confusione della Rete Neurale su F-MINST

Anche analizzando la metrica F-score, che vediamo nella Tabella 4.6, notiamo che i risultati sono in linea con la matrice di confusione e quindi la rete ha difficoltà nel riconoscere alcune classi.

Classe	0	1	2	3	4	5	6	7	8	9
F-score	0,83	0,98	0,78	0,90	0,80	0,97	0,69	0,95	0,96	0,96

**Tabella 4.6:** Valori F-score su F-MNIST con le reti neurali

Nella tabella F-score emerge un valore basso anche nella classe 4 (Cappotto) indicando quindi una precisione minore anche su questa categoria di abbigliamento.

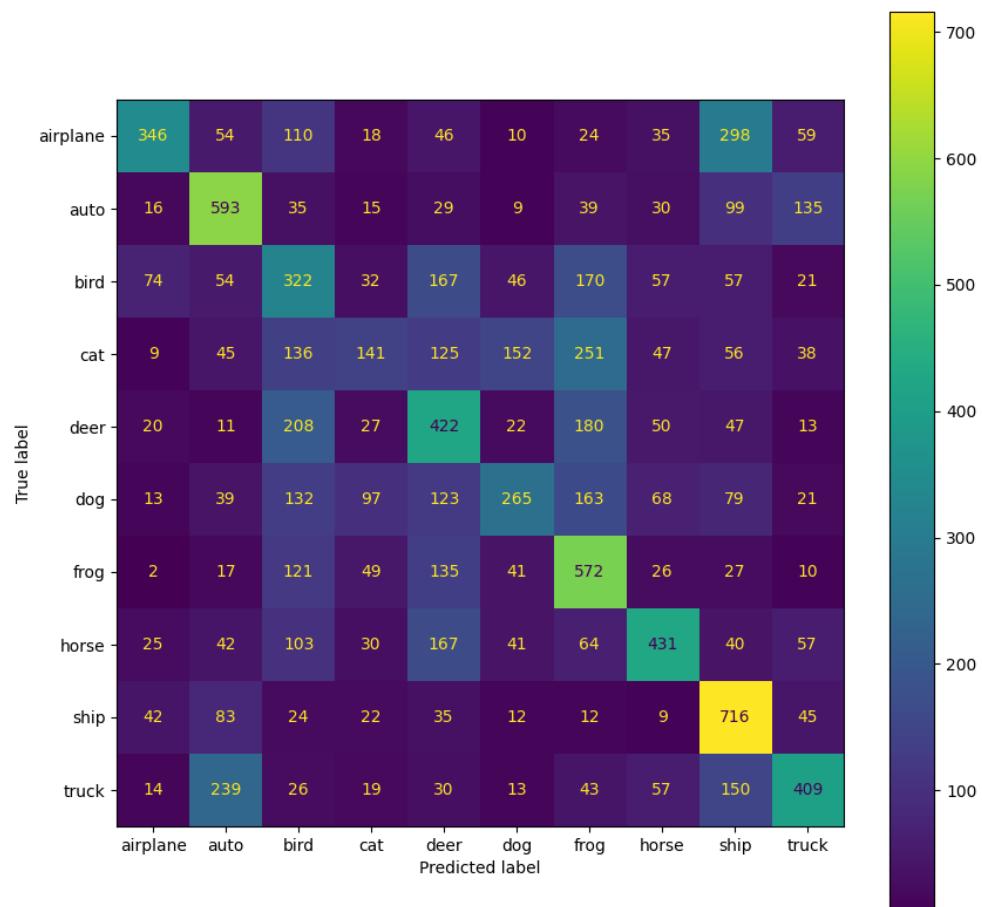
### 4.3.2 Test su CIFAR-10

Sul dataset CIFAR-10 le performance della rete neurale calano notevolmente rispetto a F-MNIST. L'accuratezza si attesta al 45% circa nel testing della rete come vediamo nella Tabella 4.7

Train accuracy	Train loss	Test accuracy	Test loss
47,31%	1.48	44,90%	1,55

**Tabella 4.7:** Accuracy e loss su CIFAR-10 con le reti neurali

Dalla matrice di confusione in Figura 4.6 notiamo che la rete ha problemi di classificazione sulla maggior parte delle classi.



**Figura 4.6:** Matrice di confusione della Rete Neurale sul CIFAR-10

Il problema si presenta anche analizzando il valore F-score dove nessuna classe riesce a superare un valore di 0,60. Nella Tabella 4.8 vengono mostrati tutti i valori della metrica F-score.

Classe	0	1	2	3	4	5	6	7	8	9
F-score	0,50	0,59	0,30	0,32	0,41	0,34	0,49	0,50	0,58	0,45

**Tabella 4.8:** Valori F-score sul CIFAR-10 con le reti neurali

## 4.4 Risultati classificazione con Reti Neurali Convolutionali

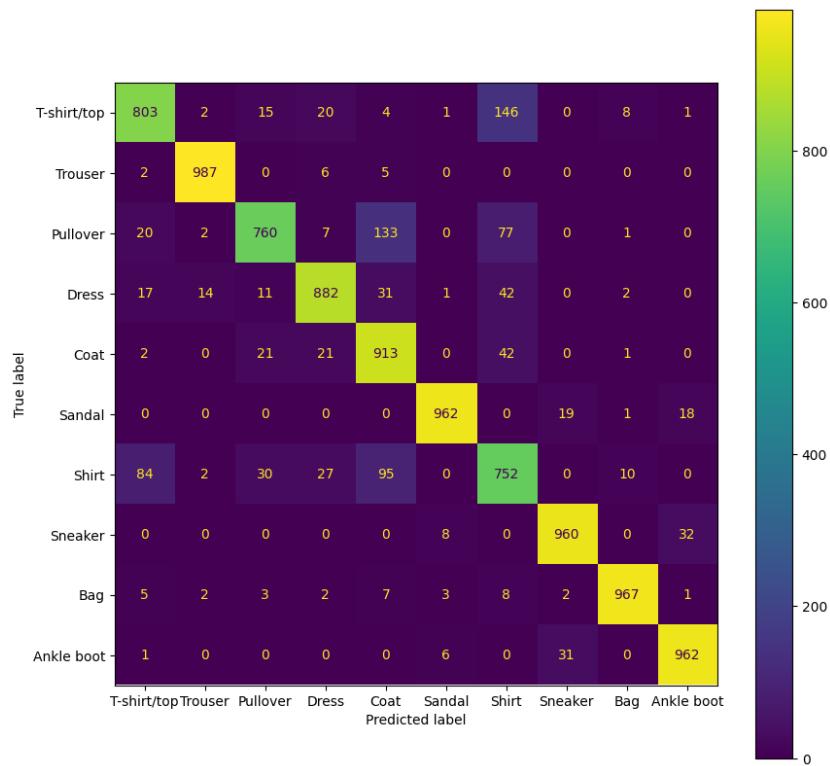
### 4.4.1 Test su Fashion-MNIST

Come anticipato in precedenza, per le reti neurali convoluzionali sono state implementate tre modelli con complessità diverse. Come possiamo vedere nella Tabella 4.9 in tutti i modelli testati è stato raggiunto un valore di accuratezza pari o superiore a 90%.

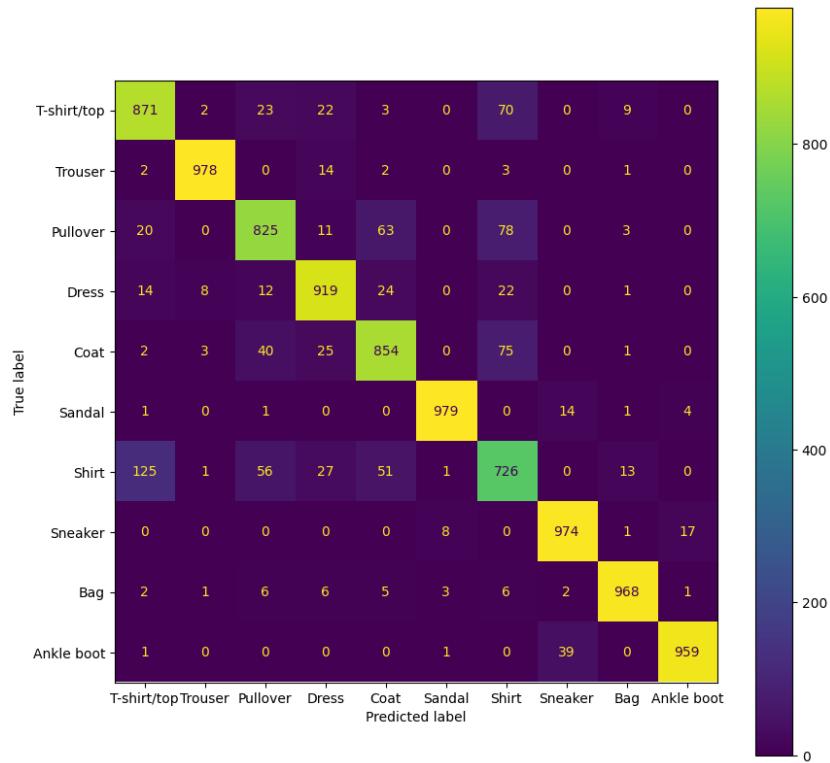
Modello	Train accuracy	Train loss	Test accuracy	Test loss
1	93,90%	0.16	90,23%	0,32
2	96,17%	0.09	90,35%	0,42
3	91,47%	0.22	91,14%	0,24

**Tabella 4.9:** Accuracy e loss su F-MNIST con le reti neurali convoluzionali

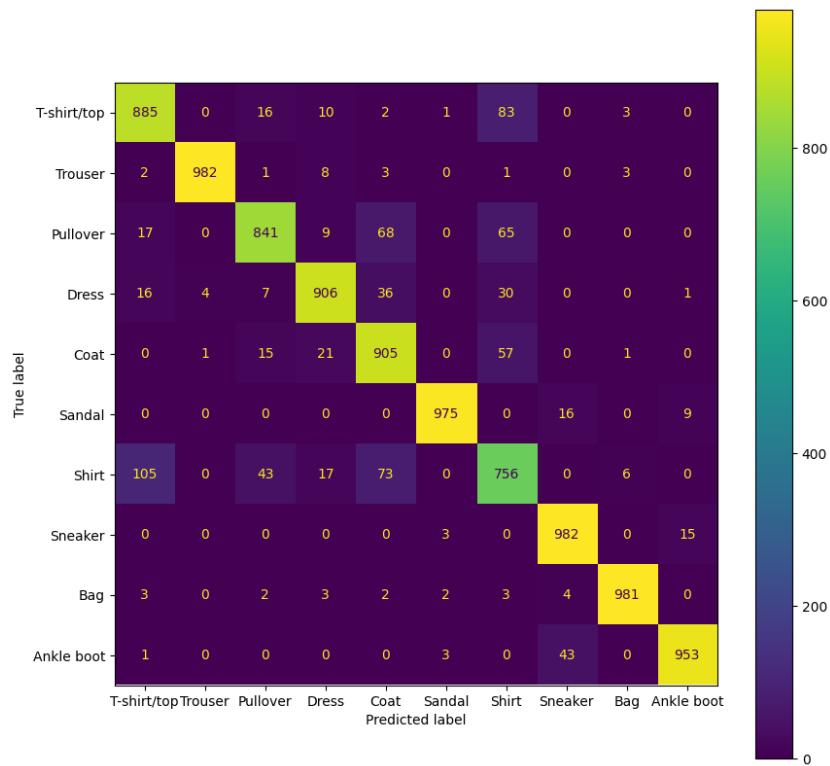
Dalle matrici di confusione generate possiamo notare che l'unica categoria nella quale abbiamo una precisione minore è la classe 6 (Camicia). Alcuni elementi di questa classe, vista la somiglianza, vengono classificati come T-shirt o Pullover. In seguito le tre matrici di confusione.



**Figura 4.7:** Matrice di confusione della CNN sul F-MNIST per il Modello 1



**Figura 4.8:** Matrice di confusione della CNN sul F-MNIST per il Modello 2



**Figura 4.9:** Matrice di confusione della CNN sul F-MNIST per il Modello 3

Le osservazioni fatte in precedenza grazie alle matrici di confusione vengono rispecchiate anche nella metrica dell’F-score. Ancora una volta il valore peggiore è stato ottenuto dalla classe 6. Nella Tabella 4.10 mostriamo tutti i valori F-score di ogni classe per i tre modelli.

Classe	Model 1	Model 2	Model 3
Classe 0	0,84	0,86	0,86
Classe 1	0,98	0,99	0,99
Classe 2	0,86	0,85	0,87
Classe 3	0,90	0,91	0,91
Classe 4	0,85	0,83	0,86
Classe 5	0,97	0,97	0,98
Classe 6	0,72	0,74	0,74
Classe 7	0,96	0,96	0,96
Classe 8	0,98	0,97	0,98
Classe 9	0,96	0,96	0,97

**Tabella 4.10:** Comparazione valori F-score su F-MNIST con CNN

#### 4.4.2 Test su CIFAR-10

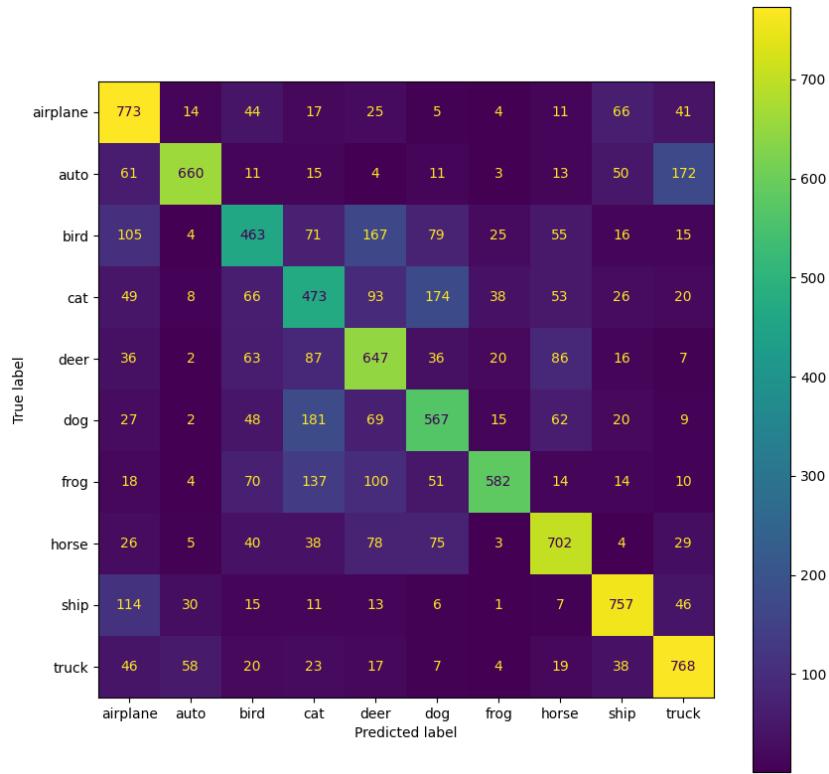
Nella Tabella 4.11 sono elencati i risultati in termini di accuratezza e loss sul CIFAR-10, nei quali possiamo vedere una differenza sostanziale tra i tre modelli analizzati.

Modello	Train accuracy	Train loss	Test accuracy	Test loss
1	75,40%	0.70	65,28%	1,10
2	85,32%	0.40	70,45%	1,08
3	76,30%	0.67	76,35%	0,69

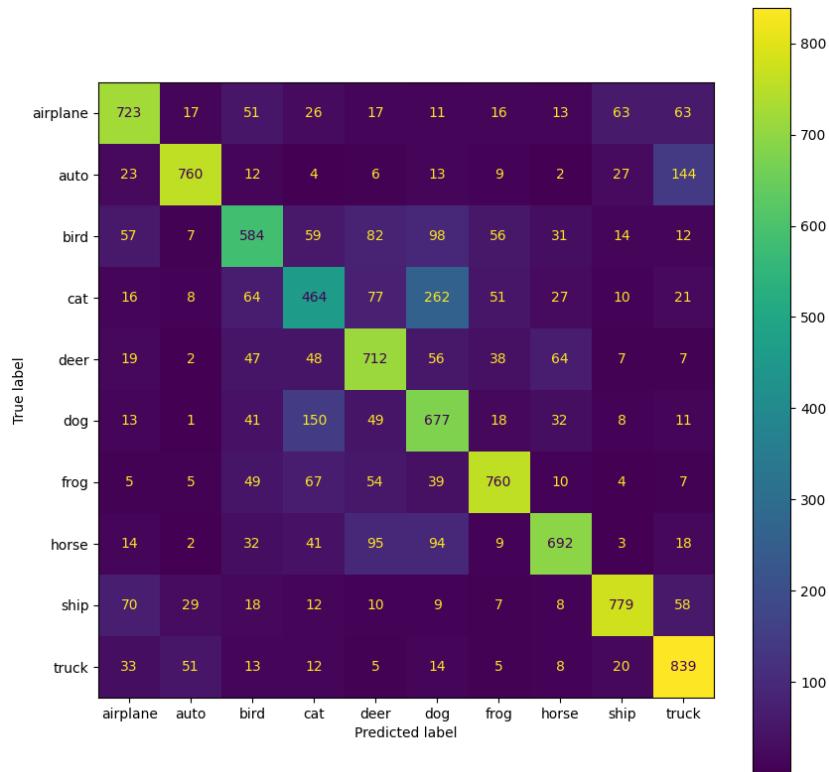
**Tabella 4.11:** Accuracy e loss sul CIFAR-10 con le reti neurali convoluzionali

Dalla Tabella 4.11 possiamo notare come il modello 3 ottiene prestazioni migliori per l'accuracy ed un overfitting nullo.

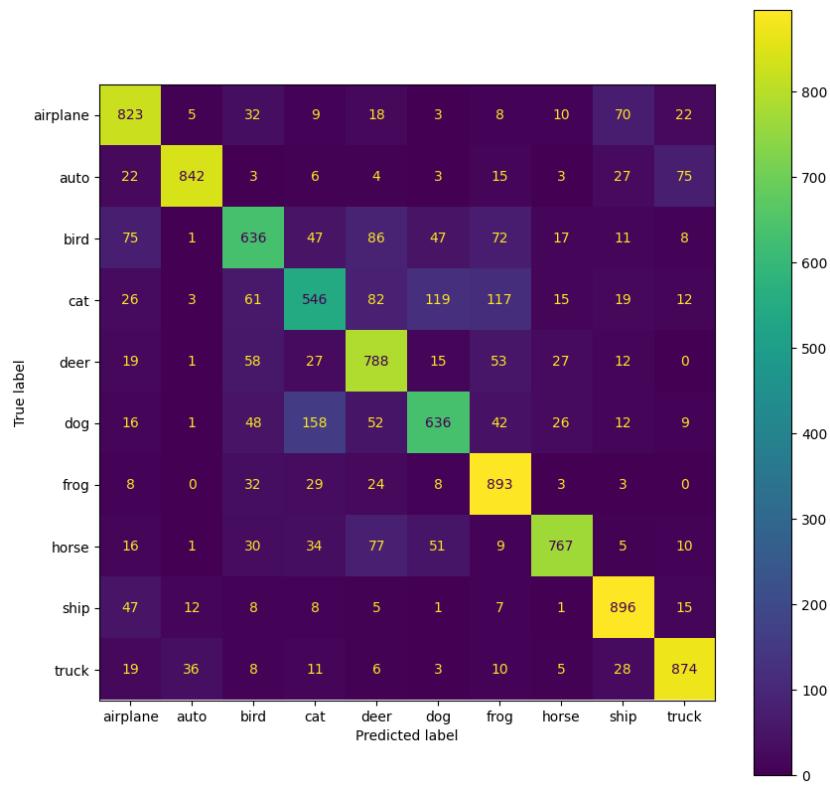
In seguito vengono mostrate le matrici di confusione per i tre modelli.



**Figura 4.10:** Matrice di confusione della CNN sul CIFAR-10 per il Modello 1



**Figura 4.11:** Matrice di confusione della CNN sul CIFAR-10 per il Modello 2



**Figura 4.12:** Matrice di confusione della CNN sul CIFAR-10 per il Modello 3

Dalle matrici di confusione notiamo ancora una volta che le classi più difficili da classificare per il CIFAR-10 sono quelle relative a uccelli, gatti e cani, rispettivamente classe 2, 3, 5.

Nella Tabella 4.12 che segue vengono elencati i valori F-score per ogni modello.

Classe	Model 1	Model 2	Model 3
Classe 0	0,69	0,73	0,79
Classe 1	0,74	0,81	0,89
Classe 2	0,50	0,61	0,66
Classe 3	0,46	0,49	0,58
Classe 4	0,58	0,68	0,74
Classe 5	0,56	0,60	0,67
Classe 6	0,69	0,77	0,80
Classe 7	0,69	0,73	0,82
Classe 8	0,75	0,81	0,86
Classe 9	0,73	0,77	0,86

**Tabella 4.12:** Comparazione valori F-score su CIFAR-10 con CNN

## 4.5 Confronto finale

Nella Tabella 4.13 vengono riassunti i risultati della SVM su entrambi i dataset.

Metrica	F-MNIST	CIFAR-10
Accuracy	88,29%	44,79%
F-score avg	88,3%	44,3%

**Tabella 4.13:** Riepilogo SVM

Nella Tabella 4.14 vengono mostrati i risultati per l'algoritmo Random Forest.

Metrica	F-MNIST	CIFAR-10
Accuracy	87,66%	47,15%
F-score avg	86,80%	46,5%
OOB-ERROR	0,11	0,57

**Tabella 4.14:** Riepilogo Random Forest

Nella Tabella 4.15 viene mostrato un resoconto dei risultati della rete neurale.

Dataset	Train acc.	Train loss	Test acc.	Test loss	F-score avg
F-MNIST	94,67%	0.14	89,49%	0,38	88,2%
CIFAR-10	47,31%	1,48	44,90%	1,55	44,80%

**Tabella 4.15:** Riepilogo rete neurale

Per le reti neurali convoluzionali viene mostrato un riepilogo dei risultati solo del modello 3 che ha avuto performance migliori.

Dataset	Train acc.	Train loss	Test acc.	Test loss	F-score avg
F-MNIST	91,47%	0.22	91,14%	0,24	91,2%
CIFAR-10	76,30%	0.67	76,35%	0,69	76,7%

**Tabella 4.16:** Riepilogo del modello 3 di CNN

# CAPITOLO 5

---

## Conclusioni

---

### 5.1 Lavoro svolto

Il lavoro svolto in questa tesi si è incentrato sulla valutazione degli algoritmi di Machine Learning per la classificazione di immagini utilizzando diverse tecniche e metriche. Gli algoritmi analizzati sono: SVM, Random Forest, Rete neurale artificiale ed infine le reti neurali convoluzionali. Fashion-MNIST e CIFAR-10 sono stati i dataset utilizzati per addestrare e testare questi modelli. Il primo passo di questa tesi è stato quello di analizzare in modo approfondito i lavori presenti in letteratura, dove è possibile osservare che quasi la totalità dei test effettuati vengono valutati solo tramite l'accuratezza. In alcuni casi la sola accuratezza non è sufficiente a valutare l'affidabilità di un algoritmo, soprattutto quando utilizziamo dataset sbilanciati. Il lavoro svolto si è basato sull'analisi e i test di diverse metriche e tecniche di valutazione. L'intera ricerca è stata supportata da librerie Python come Tensorflow, Keras e Scikit-learn. Tutti i modelli sono stati implementati ed eseguiti sul Notebook Python online "Google Colab" che mette a disposizione risorse di calcolo con GPU utili all'addestramento dei modelli. Le metriche utilizzate sono state approfondite nella documentazione ufficiale delle librerie specificate in precedenza.

## 5.2 Risultati ottenuti

Dai risultati ottenuti possiamo osservare innanzitutto che il dataset CIFAR-10 ha un grado di difficoltà maggiore rispetto a Fashion-MNIST a causa delle immagini a colori più complesse e dalla presenza di rumore. Un altro aspetto ottenuto dall’analisi riguarda le prestazioni degli algoritmi, in particolare vediamo che su Fashion-MNIST tutti i modelli si comportano in modo eccellente con risultati equiparabili ma addestrando i modelli sul CIFAR-10 notiamo una differenza sostanziale tra le reti neurali convoluzionali ed i modelli restanti. Utilizzando dataset con difficoltà diverse, come in questo caso, siamo riusciti a mostrare le vere potenzialità delle CNN sulla classificazione di immagini. L’utilizzo di metriche diverse ci ha permesso di capire come i modelli si sono adattati ai dati e dove soffrono maggiormente nella classificazione.

## 5.3 Sviluppi futuri

Dati i risultati della ricerca si potrebbe procedere a:

- Valutare gli Algoritmi utilizzando la metrica delle curve ROC-AUC;
- Testare i dataset F-MNIST e CIFAR-10 utilizzando le architetture di CNN note come AlexNet, LeNet-5 e VGG-NET.

---

## Bibliografia

---

- [1] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis *et al.*, “Deep learning for computer vision: A brief review,” *Computational intelligence and neuroscience*, vol. 2018, 2018. (Citato a pagina 1)
- [2] IBM, “Cos’è l’intelligenza artificiale (ia)?” [Online]. Available: <https://www.ibm.com/it-it/topics/artificial-intelligence> (Citato alle pagine 4 e 5)
- [3] Oracle, “Cos’è il machine learning?” [Online]. Available: <https://www.oracle.com/it/artificial-intelligence/machine-learning/what-is-machine-learning/> (Citato a pagina 5)
- [4] A. AWS, “Cos’è il machine learning.” [Online]. Available: <https://aws.amazon.com/it/what-is/machine-learning/> (Citato a pagina 5)
- [5] ——, “Cos’è il deep learning.” [Online]. Available: <https://aws.amazon.com/it/what-is/deep-learning/> (Citato a pagina 6)
- [6] I. Sydorenko, “What is a dataset in machine learning: Sources, features, analysis,” 2021. [Online]. Available: <https://labelyourdata.com/articles/what-is-dataset-in-machine-learning> (Citato a pagina 6)
- [7] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” 2017. [Online]. Available: <https://arxiv.org/abs/1708.07747> (Citato alle pagine 7 e 24)

- [8] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, vol. 2, 2010. (Citato a pagina 7)
- [9] A. Krizhevsky, “Learning multiple layers of features from tiny images,” *Tech. Rep.*, 2009. (Citato a pagina 7)
- [10] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, “Review of image classification algorithms based on convolutional neural networks,” *Remote Sensing*, vol. 13, no. 22, 2021. [Online]. Available: <https://www.mdpi.com/2072-4292/13/22/4712> (Citato a pagina 9)
- [11] W. S. Noble, “What is a support vector machine?” *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006. (Citato a pagina 9)
- [12] MathWorks, “Support vector machine.” [Online]. Available: <https://it.mathworks.com/discovery/support-vector-machine.html> (Citato alle pagine 9 e 10)
- [13] C. Casadei, “Support vector machine.” [Online]. Available: <https://www.developersmaggiali.it/blog/support-vector-machine/> (Citato a pagina 10)
- [14] IBM, “Cos’è un albero decisionale?” [Online]. Available: <https://www.ibm.com/it-it/topics/decision-trees> (Citato a pagina 11)
- [15] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, pp. 123–140, 1996. (Citato a pagina 11)
- [16] IBM, “Cos’è il bagging?” [Online]. Available: <https://www.ibm.com/it-it/topics/bagging> (Citato a pagina 12)
- [17] N. Boldrini, “Reti neurali: cosa sono e a cosa servono,” 2022. [Online]. Available: <https://www.ai4business.it/intelligenza-artificiale/deep-learning/reti-neurali/> (Citato a pagina 13)
- [18] NetAI, “Guida rapida alle funzioni di attivazione nel deep learning.” [Online]. Available: <https://netai.it/>

- guida-rapida-alle-funzioni-di-attivazione-nel-deep-learning/#page-content  
(Citato alle pagine 14, 15 e 16)
- [19] IBM, "Reti neurali convoluzionali." [Online]. Available: <https://www.ibm.com/it-it/topics/convolutional-neural-networks> (Citato alle pagine 17 e 18)
- [20] S. S. Kadam, A. C. Adamuthe, and A. B. Patil, "Cnn model for image classification on mnist and fashion-mnist dataset," *Journal of scientific research*, vol. 64, no. 2, pp. 374–384, 2020. (Citato alle pagine 20 e 22)
- [21] O. M. Khanday, S. Dadvandipour, and M. A. Lone, "Effect of filter sizes on image classification in cnn: A case study on cifr10 and fashion-mnist datasets," *IAES International Journal of Artificial Intelligence*, vol. 10, no. 4, p. 872, 2021. (Citato a pagina 25)
- [22] R. Doon, T. K. Rawat, and S. Gautam, "Cifar-10 classification using deep convolutional neural network," in *2018 IEEE Punecon.* IEEE, 2018, pp. 1–5. (Citato a pagina 26)
- [23] X. Zhang, "The alexnet, lenet-5 and vgg net applied to cifar-10," in *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE).* IEEE, 2021, pp. 414–419. (Citato a pagina 27)
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. (Citato a pagina 27)
- [25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. (Citato a pagina 28)
- [26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. (Citato a pagina 30)

---

## **Ringraziamenti**

---

**Innanzitutto, un grande ringraziamento al mio relatore Fabio Palomba e al mio tutor Giammaria, sempre disponibili a guidarmi in ogni fase della realizzazione della tesi. Grazie a voi ho acquisito nuove conoscenze in quest'ambito.**

**Ringrazio gli amici incontrati durante questo percorso di studi all'università, chi conosciuto per un esame, chi per un caffè, chi per l'intero percorso, tutti.**

**Ringrazio i miei Genitori che mi sono stati sempre accanto e che in questi anni hanno fatto tantissimi sacrifici per permettermi di raggiungere quest'obiettivo. Questo laurea è dedicata soprattutto a voi e spero di riuscire a portare ulteriori traguardi e soddisfazioni a casa.**

**Ringrazio i miei fratelli Pasquale e Mirko per le passioni in comune che coltiviamo, i consigli sia di studi che di vita ma soprattutto li ringrazio perché insieme alle mie cognate Giusi e Rosaria mi hanno regalato 2 nipotini stupendi, anzi a breve 3!**

**Ringrazio Benedetto, per tutti momenti passati insieme, per i consigli che ci scambiamo, per le innumerevoli chiamate giornaliere per consulenze tecniche ma che mettevano sempre di buon umore, punto di riferimento per i weekend passati a Cerreto.**

---

**Ringrazio Ceres e Marzio che nonostante la distanza ci sono sempre stati, dal primo giorno e non manca mai l'occasione per sentirci con un messaggio o una chiamata!**

**Ringrazio tutti gli amici di Cerreto per tutti i momenti passati insieme lucidi e non... I weekend da Lux, le cene, le serate a ballare e tutti i momenti che verranno. Il venerdì tornavo soprattutto per tutto questo!**

**Ringrazio Marco, conosciuto alle superiori dove fin da subito abbiamo avuto un bellissimo rapporto che grazie all'università è cresciuto sempre di più. In questi anni abbiamo condiviso facoltà, casa, hobby e tanto altro. Vivendo da soli abbiamo affrontato nuovi problemi ed avventure come la sera in cui ti sei chiuso in camera lasciandomi solo a combattere eroicamente contro un pipistrello. Questo traguardo è stato raggiunto anche grazie a te.**

*Questa tesi ha contribuito a piantare un albero in Kenya tramite il progetto Treedom.*

<https://www.treedom.net/it/user/sesalab/event/sesa-random-forest>