Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Dall'IA all'olio: come affinare i sistemi di classificazione della qualità attraverso tecniche di machine learning con l'utilizzo di reti neurali

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Correlatore:
Chiar.mo Prof.
Luca Calamai

Presentata da:
Giosué Cotugno

Sessione I
Anno Accademico 2019-2020

Abstract

Sino ad oggi, per verificare se un olio è extravergine d'oliva o contiene dei difetti, non abbiamo potuto usufruire di molti supporti tecnologici, infatti la tecnica maggiormente utilizzata per la classificazione degli oli è il panel test che consiste nell'assaggio di questi da un gruppo di persone esperte. Recentemente a seguito di analisi oggettive delle molecole costituenti l'olio, sono state introdotte ulteriori metodologie per la classificazione grazie all'utilizzo di analisi statistiche. Questa tesi, dunque, propone un ulteriore metodo di classificazione grazie all'utilizzo delle reti neurali, ossia modelli computazionali composti da neuroni "artificiali" disposti su più livelli che si ispirano ad una rete neurale biologica [1]. Utilizzando i dati sensoriali forniti dal panel test ed i dati molecolari ricavati da analisi chimiche, siamo stati in grado di creare un modello basato sulle reti neurali, capace di predire, con una certa accuratezza, la classe di appartenenza di un olio data la sua composizione molecolare. Le reti su cui si è basata questa sperimentazione sono le seguenti: resnet18, resnet50, mobilenet ed infine una rete neurale costruita ad–hoc per questo esperimento. Le prime tre sono reti convoluzionali, dunque specializzate nel riconoscimento di immagini, mentre la rete personalizzata ha la struttura di una classica rete neurale. Nello specifico le reti resnet18 e resnet50 sono due residual network e sono caratterizzate dal residual block, un ulteriore collegamento che permette di avere come input di un livello, non solo l'output del livello precedente, come avviene nelle comuni reti neurali, ma anche gli output dei livelli superiori. Questa tipologia di rete è stata realizzata per essere usata nel dataset Imagenet ed ha dato ottimi risultati [2]. Mobilenet è una rete che è stata creata per avere un costo computazionale basso senza andare ad intaccare troppo l'accuratezza dei risultati. Questo tipo di rete viene utilizzato principalmente su dispositivi mobili per le loro particolari caratteristiche. Per realizzare ciò

è stato usato un modello ottimizzato ed ha portato ad avere risultati soddisfacenti in campi come il riconoscimento degli oggetti, gli attributi facciali ecc. [3]. L'utilizzo di tutte queste reti ha permesso di comprendere quale sia la miglior tipologia di rete su cui basare ulteriori ricerche nell'ambito della classificazione dell'olio e di comprendere quali siano le loro potenzialità. I risultati ottenuti hanno permesso di concludere che per riuscire a separare differenti classi di olio è sufficiente utilizzare reti neurali classiche e che la ricerca è sulla buona strada per riuscire a trovare uno strumento solido utile a questo scopo.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

L'olio extra–vergine d'oliva è l'olio commestibile di maggiore qualità, che prende questa caratteristica dal suo processo di produzione molto raffinato e dalla bontà delle olive utilizzate. Nel caso in cui vi sia qualche difetto in un olio prodotto è importante essere in grado di riconoscerlo in quanto questi ne caratterizzano la sua classificazione tra olio extra vergine d'oliva (EVOO), olio vergine d'oliva (VOO) e olio vergine d'oliva lampante (LVOO). Questa classificazione viene fatta tramite il panel test che consiste nell'assaggio di più tipologie d'olio riportandone i difetti o le note sensoriali positive in una tabella di punteggio specifica, come regolamentato dall'Unione europea [4]. Esistono diversi possibili difetti che un olio può assumere: rancido, vinoso, ammuffito ecc. e viene considerato extravergine solo se non presenta alcun tipo di difetto ma al più note positive. Nel caso in cui questo presentasse dei difetti con valori inferiori a 3,5 nella tabella di punteggio del panel test, allora esso viene classificato come "olio vergine d'oliva", mentre viene classificato come "olio vergine d'oliva lampante" nel caso in cui i difetti superino la soglia di 3,5. L'utilizzo della spettrometria di massa e della gas cromatografia, due tecniche analitiche, hanno permesso di separare le molecole che sono contenute all'interno degli oli. L'analisi approfondita di questi componenti ha permesso di studiare se vi fossero delle relazioni tra questi ultimi e la categorizzazione dell'olio. Il primo studio documentato lo troviamo all'interno di questo articolo, [5] dove si prova a dare un supporto al panel test per mezzo di analisi statistiche, con dei risultati di classificazione corretti dell'83,5%. Il lavoro svolto all'interno di questa tesi mira a fornire un ulteriore strumento tecnologico in grado di

classificare un olio in base alla sua composizione molecolare. Questo strumento, basato su reti neurali e reti neurali convoluzionali, è stato realizzato in Python con l'ausilio della libreria Pytorch, specifica per l'apprendimento automatico. Le prime reti nominate hanno caratteristiche molto differenti rispetto alle ultime siccome queste sono specializzate nel riconoscimento di immagini, mentre le reti neurali comuni, lavorano con input numerici. Nonostante ciò, quelle convoluzionali sono state ugualmente utilizzate poiché esse hanno la possibilità di essere usate con i pesi degli archi inizializzati in maniera non casuale in quanto vengono fornite già addestrate. I pesi degli archi già inizializzati, che caratterizzano la capacità di una rete neurale a risolvere un determinato problema, ci forniscono un buon vantaggio, in quanto avendo pochi dati a disposizione per l'addestramento, i pesi non riuscirebbero ad assumere dei valori che permettano al modello di avere delle buone accuratezze nel classificare i diversi oli. Nei prossimi capitoli si troverà una descrizione dettagliata dei seguenti argomenti: (i) machine learning basato su reti neurali e sulla struttura di una rete neurale e di una rete neurale convoluzionale, (ii) la fase di pre-elaborazione dell'insieme dei dati e la configurazione degli iperparametri delle reti neurali, (iii) la fase di test con un confronto dei risultati ottenuti dalle varie reti ed i risultati ottenuti utilizzando altre tecniche.

Capitolo 2

Machine Learning

In questo capitolo si vuole fare un accenno alle tecnologie utilizzate per la sperimentazione discussa in questa tesi, introducendo il concetto di machine learning, rete neurale e rete neurale convoluzionale utilizzando i documenti[6] [7] [8] come linee guida per la sua redazione.

2.1 Introduzione

Con l'avanzamento tecnologico verificatosi negli ultimi anni, si è venuta a generare una vasta mole di dati riguardante discipline differenti. Il machine learning è un insieme di tecniche che prova a capire questi dati. Le tecniche di cui usufruiamo, ispirate da procedure di calcolo dei sistemi biologici, risultano molto più efficienti di quelle matematiche, in quanto per l'utilizzo di queste ultime, vi è necessaria una completa conoscenza del dominio, mentre per le prime no. Una struttura portante su cui si basa il machine learning e che ha suscitato particolare interesse, è la rete neurale.

2.2 Reti neurali

Il termine rete neurale fa riferimento a sistemi costituiti da unità in grado di attivarsi all'arrivo di un segnale e da linee di interconnessione in grado di propagare l'attivazione ad altre unità. Questa particolare tipologia di sistema si ispira, non a caso, al funzionamento

dei sistemi nervosi biologici ed in particolar modo ai neuroni che fanno parte del cervello. Tale tecnologia permette di essere applicata a svariati campi grazie ad una sua specifica proprietà: essere in grado di apprendere dai dati, con o senza l'ausilio di informazioni aggiuntive. Questa proprietà apre due nuovi scenari di apprendimento:

- Supervisionato: L'obiettivo è quello di predire il valore in uscita sulla base di ciò che si è appreso dai dati.
- Non supervisionato: L'obiettivo è quello di descrivere le relazioni e le associazioni mediante i valori in ingresso.

In questo paragrafo andremo a descrivere i metodi di apprendimento supervisionato essendo l'unica tecnica di cui ci siamo serviti nella fase di sperimentazione.

2.2.1 Metodi di apprendimento supervisionato

Nell'apprendimento supervisionato, oltre l'insieme dei dati, è presente una misura dell'uscita che può essere quantitativa o qualitativa (come nel nostro caso, la classe di appartenenza) chiamata anche label (target). Questa misura è ciò che noi vogliamo predire sulla base di un insieme di caratteristiche o parametri (features) di input. A causa della natura della predizione che vogliamo effettuare, vi è la necessità di utilizzare differenti tipi di apprendimento:

- Regressione: se vogliamo predire output quantitativi.
- Classificazione: se vogliamo predire output qualitativi.

In entrambi i casi si può dire che si sta svolgendo un'attività di approssimazione di funzioni. Andando ad analizzare il primo tipo di apprendimento, concludiamo che questo ha lo scopo di andare a predire il valore di una funzione y(x,w) sulla base di un insieme di addestramento costituito da coppie (x_i, y_i). I valori di uscita dell'insieme di addestramento sono detti "target" e sono di tipo quantitativo ($y \in R^d$). Tutto ciò, è reso possibile tramite la minimizzazione della funzione di errore della somma dei quadrati (figura 2.1).

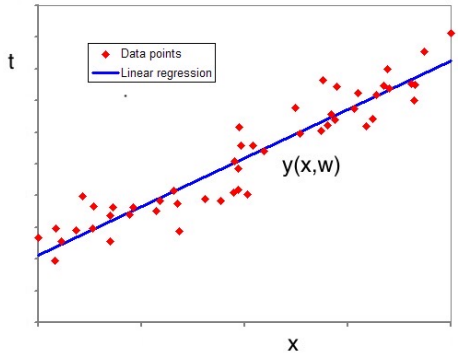

Figura 2.1: Esempio di regressione lineare

Il secondo tipo di apprendimento ha lo scopo di assegnare un input ad una determinata classe sulla base di un insieme di addestramento (training) costituito dalle coppie (x_i, g_i). I valori g sono detti target (ground truth) e sono di tipo qualitativo (si, no, A, B, C). Il risultato permette di scegliere una regola di suddivisione che fraziona lo spazio di input in regioni di decisione, separati da un confine detto anch'esso "di decisione" (figura 2.2).

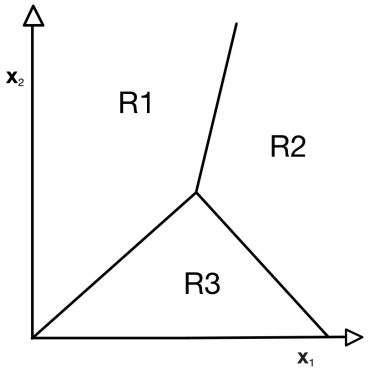

Figura 2.2: Esempio di classificazione

Ognuno di questi confini di decisione è determinato dalla funzione:

$$g(x) = w^{\mathrm{T}}x + w_0$$

dove w è il vettore dei pesi e w_0 è la soglia. Una tale funzione realizza un classificatore a due classi C_1 e C_2 definendo la regola di decisione: se g(x)>0 allora x $\in$ C_1 altrimenti x $\in$ C $_2$ Dunque il confine di decisione è fissato dall'equazione $g(x) = 0$

Avendo n classi si può generalizzare utilizzando una funzione discriminante per ogni classe C_i :

$$g_{\mathrm{i}}(x) = w^{\mathrm{T}}x_{\mathrm{i}} + w_{\mathrm{i}\,0} \text{ per } i = 1, ..., n$$

.

In questo caso la funzione di decisione assegna x a C_i se $g_{\mathrm{i}}(x) > g_{\mathrm{j}}(x)$ con $i \neq j\ \forall i, j$

2.2.2 Struttura delle reti neurali

Come già accennato nell'introduzione di questo capitolo, sappiamo che le reti neurali sono un formalismo che si basa sulla struttura del cervello. Gli elementi fondamentali di questa struttura sono i neuroni, organizzati in livelli e connessi tra di loro di modo che l'output dei neuroni di uno specifico livello costituiscano l'input per quelli del livello successivo. Il funzionamento di un singolo neurone lo si può immaginare come un sistema avente un'entrata (composta da una serie di collegamenti) ed un'uscita dove:

- a_j è il valore di uscita del neurone precedente j
- $W_{j,i}$ è il peso dell'arco che collega il neurone j al neurone i (attuale)
- in_i è il valore della somma pesata degli ingressi
- g è la funzione di attivazione
- a_i è il valore di uscita del neurone i calcolato come $a_i = g(in_i)$

Il neurone prende tutti gli input acquisiti dai neuroni facenti parte del livello precedente e li combina con una somma pesata data dalla formula :

$$in_{\mathrm{i}} = \sum W_{\mathrm{j,i}} a_{\mathrm{i}}$$

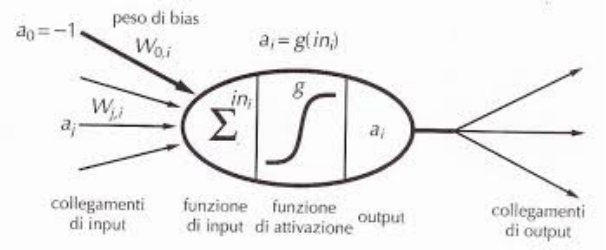

Figura 2.3: Neurone artificiale

Il valore in_i viene dato come input alla funzione di attivazione poiché essa svolge un ruolo di soglia. In passato la funzione di attivazione era definita da una funzione discontinua, mentre oggi si preferisce utilizzare funzioni continue come la sigmoide. L'architettura neurale più semplice è quella costituita da soli due livelli, uno d'entrata ed uno di uscita denominata anche "single-layer perceptron" e questa riuscirà a delimitare due classi linearmente separabili approssimando la funzione lineare (teorema di convergenza del percettrone Rosenblatt,1962). Per delimitare delle classi non linearmente separabili invece si utilizza la rete "multi-layer perceptron" che permette di approssimare sia le funzioni continue che quelle non continue. Entrambe queste architetture sono dette feed-forward in quanto propagano i valori in ingresso verso i livelli successivi mantenendo una sola direzione. Riassumendo notiamo dunque che ogni nodo è collegato da un arco con un peso associato $W_{i,j}$ che viene combinato con il valore inviato dal neurone di partenza. Il valore prodotto dal neurone tramite la funzione di attivazione viene moltiplicato per il peso dell'arco $W_{i,j}$ e sommato con tutti gli altri, dunque ripassato alla funzione di attivazione.

Non esistendo una rete neurale universale in grado di risolvere qualsiasi problema, per ognuno di questi bisognerebbe trovare una combinazione dei pesi degli archi $W_{i,j}$ in grado di risolverlo. Questo compito è computazionalmente improponibile, ma essendo capaci di valutare la bontà dell'uscita della rete in relazione ad ogni ingresso, si è pensato di partire da una configurazione dei pesi casuale per poi fare variazioni leggere su di essi in modo da avvicinarci ad una situazione migliore rispetto a quella attuale. Ripetendo questa iterazione molte volte si hanno buone probabilità di arrivare ad una configurazione che si avvicini alla soluzione del problema. Il procedimento appena descritto prende il

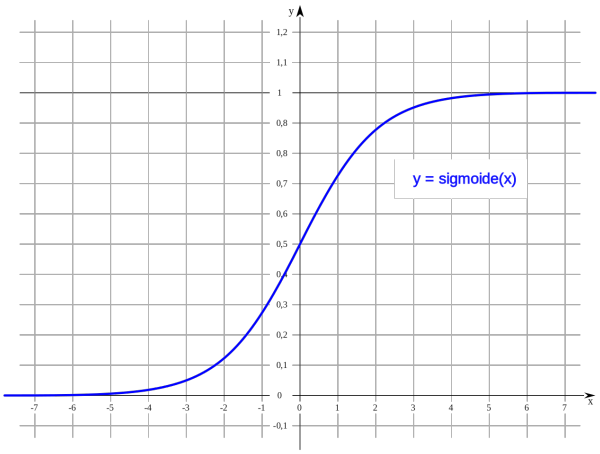

Figura 2.4: Funzione di attivazione sigmoide

nome di error back-propagation che consiste nel modificare i pesi associati agli archi sistematicamente e gradualmente in modo da riuscire ad ottenere un valore di uscita della rete che sia il più vicino possibile a quello aspettato. Essendo impossibile fornire tutti i valori ingresso-uscita, è richiesta alla rete una capacità di generalizzazione. La problematica che può nascere da questa richiesta è quella che la rete non impari ma memorizzi, una delle possibili cause si ha quando viene effettuato un addestramento troppo lungo.

2.3 Reti neurali convoluzionali

Un calcolatore non è in grado di riconoscere oggetti o paesaggi all'interno di un'immagine, in quanto questo le vede solo come matrici di pixel. Per questo motivo sono state create le reti convoluzionali, che riescono con buone probabilità a riconoscerne il contenuto. Questa tipologia di rete è caratterizzata dalla seguente struttura:

- Livello convoluzionale, filtro, passo e padding

Il livello convoluzionale è quello che caratterizza queste reti rispetto a tutte le altre. Di questo livello ve ne possono essere molteplici, ed ognuno ha il compito di ricercare una caratteristica differente nell'immagine. Per riuscire a spiegare in maniera corretta

il funzionamento di questo livello bisogna introdurre il concetto di filtro. Generalmente questo è una piccola matrice di poche righe e colonne che rappresenta la caratteristica che il livello convoluzionale vuole identificare. Dunque si avrà un filtro per ogni livello convoluzionale che riuscirà ad identificare caratteristiche sempre più complesse. Una volta creato il filtro verrà fatta una scansione della matrice di input andando ad analizzare il campo ricettivo su di essa. Questa operazione consiste nell'andare ad analizzare dei sottoinsiemi della matrice iniziale di dimensione pari alla dimensione del filtro. Durante quest'analisi la matrice iniziale viene trasformata attraverso [campo ricettivo (di dimensioni pari a quelle del filtro) prodotto scalare righe per colonne filtro] per poi andare a sommare assieme i valori ottenuti da questo prodotto scalare, ottenendo dal sottoinsieme della matrice selezionato un singolo valore.

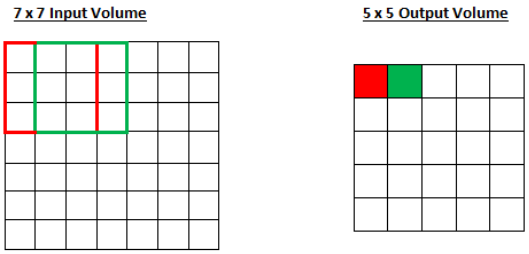

Figura 2.5: Esempio con input 7x7 con filtro 3x3 e passo 1

Questa operazione viene eseguita per tutta la matrice iniziale a partire dal pixel in alto a sinistra. La dimensione dello spostamento che si effettua per analizzare tutta la matrice viene denominato passo, come raffigurato in figura 2.5 dove si ha un passo pari a 1. Con l'applicazione del filtro, seguendo il passo, si otterrà una matrice di attivazione e questa, per come è stato applicato il filtro, sarà sicuramente di dimensione inferiore a quella iniziale. Questo potrebbe portare ad una perdita di informazione specialmente nei bordi e per ovviare a questa problematica si aggiunge uno spessore ai margini della matrice aggiungendo degli zero come raffigurato in figura 2.6.

- Livello di pooling

Il ruolo di questo livello è quello di unire caratteristiche semanticamente simili in un'unica caratteristica. Dato che la posizione delle caratteristiche formanti un pattern può essere

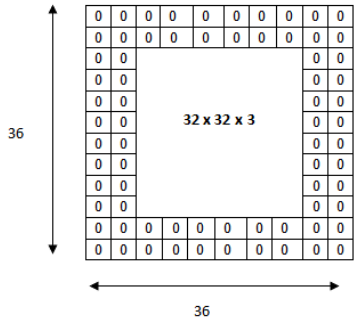

Figura 2.6: Applicazione di zero-padding ad una matrice convoluzionale

diversa da immagine ad immagine, attraverso questo livello è possibile generalizzare ogni caratteristica indipendentemente dalla posizione in cui si trova. L'unità di pooling più utilizzata è il max pooling (figura 2.7) e questa computa il massimo di un gruppo localizzato all'interno di una mappa di attivazione. Esistono altri tipi di pooling come l'average pooling o il sum pooling, tuttavia il max è il preferito in quanto in media riesce a dare risultati migliori.

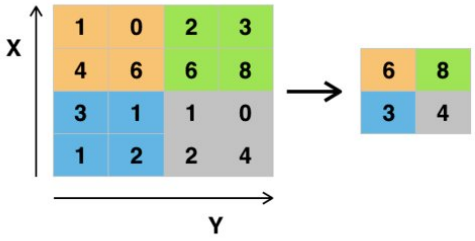

Figura 2.7: Funzionamento maxpool

- Livello full connected

Questo livello è caratterizzato dal numero di nodi di input determinati dall'output del livello precedente ed un numero di nodi di output caratterizzato dal numero di classi su cui le immagini devono convergere. Il numero contenuto all'interno dei nodi di output di questo livello, rappresenta la probabilità che l'immagine di input appartenga a

quella classe. Questo livello funziona andando a vedere gli output del livello precedente, poiché rappresenta le caratteristiche dell'immagine e poi all'interno di questo livello si va a determinare quali di queste caratteristiche siano maggiormente relazionate ad una determinata classe.

Capitolo 3

Il modello

In questo capitolo verrà inizialmente presentato il lavoro svolto sul dataset per renderlo usufruibile dalle strutture da noi utilizzate. Nella seconda sezione verranno descritte le configurazioni delle reti che ci hanno permesso di raggiungere i risultati finali.

3.1 Il dataset

3.1.1 Modifica alle classi di appartenenza

Il dataset, fornito dall'azienda Carapelli Firenze S.p.A., contiene 1300 elementi. Ognuno di essi è composto da 74 colonne, dove le prime due identificano il numero progressivo del campione e la classe di difetto, mentre le altre 72 identificano la concentrazione delle molecole all'interno dell'olio. La classe di ogni olio è stata determinata per mezzo del panel test seguita poi dall'unione delle classi che presentavano difetti simili.

Le 3 classificazione presenti all'interno dell'insieme dei dati sono:

- extravergine d'oliva (ev): se l'olio non presenta difetti
- difetto microbiologico (mi): se l'olio presenta difetti principalmente di tipo vinoso, terroso o umido
- difetto ossidato(ox): se l'olio presenta come difetto principale rancido

Per essere utilizzate dalle reti neurali queste classi sono state mappate in valori discreti:

- 0 per la classe mi
- 1 per la classe ev
- 2 per la classe ox

3.1.2 Modifica della struttura per lavorare con le reti off-the-shelf

Ora il nostro dataset per poter lavorare con le reti off-the-shelf utilizzate, necessita di un ulteriore trasformazione. Questa tipologia di rete lavora unicamente su immagini e ad oggi, non è ancora disponibile nessun altro tipo di rete che lavora con input differenti. Le immagini che queste utilizzano come input sono descritte per mezzo di una matrice 3-dimensionale, dove nel primo spazio troviamo il numero dei canali di colori, mentre negli altri due abbiamo l'altezza e la larghezza dell'immagine. Dato che il nostro dataset ci fornisce, per ogni olio, un vettore 1-dimensionale, vi è la necessita di espanderlo per renderlo compatibile con la tipologia dell'input richiesto. Per realizzare ciò sono state espanse le dimensioni del nostro input e modificate le dimensioni del vettore portandole da 1 a 3. In seguito sono stati duplicati i valori del vettore iniziale sulle dimensioni che sono state precedentemente aggiunte, così da avere come risultato finale della trasformazione una matrice [3x72x72]. Si vuole specificare che questa trasformazione potrebbe provocare dei disturbi nell'apprendimento da parte della rete. Tra i lavori futuri, si consiglia di verificare quanto questo tipo di trasformazione possa incidere sulla predizione ottenuta utilizzando queste reti.

3.1.3 Ulteriori modifiche

Una rete neurale, per apprendere correttamente le relazioni tra i dati e le classi, necessita di un set di dati che abbia a disposizione un numero di esempi il più possibile equilibrato per verificare che un dato appartenga ad una determinata classe rispetto che ad un altra. Data la composizione sbilanciata, come si può vedere in tabella 3.1, sono

state apportate ulteriori modifiche al dataset per provare a migliorare l'accuratezza delle predizioni. Una modifica apportata è stata riequilibrare il numero di oli per ogni classe in modo da renderlo uguale per ognuna di esse. Un'ulteriore modifica che viene apportata permette di testare le classi a coppie e per ogni test, viene eliminata una classe differente rispetto alle precedenti in modo da ripeterlo con combinazioni sempre differenti fino al termine di tutte le classi.

EV	MI	OX
562	513	225

Tabella 3.1: Composizione degli oli per classi di appartenenza

Le modifiche apportate sono:

1. Portare il numero dei campioni per ogni classe al valore della classe con minor numero di dati (tabella 3.2)
2. Eliminare la classe ev (tabella 3.3)
3. Eliminare la classe ox (tabella 3.4)
4. Eliminare la classe mi (tabella 3.5)

EV	MI	OX
225	225	225

Tabella 3.2: Composizione degli oli con classi equilibrate

EV	MI	OX
0	513	225

Tabella 3.3: Composizione del dataset senza gli oli ev

EV	MI	OX
563	513	0

Tabella 3.4: Composizione del dataset senza gli oli ox

EV	MI	OX
563	0	225

Tabella 3.5: Composizione del dataset senza gli oli mi

3.2 Le reti

3.2.1 Reti utilizzate

Come già accennato nell'introduzione, le reti che sono state utilizzate per questa sperimentazione sono:

- resnet50
- resnet18
- mobilenet
- rete personalizzata

Le prime tre reti sono state scelte per le loro particolari caratteristiche, come l'accuratezza quando parliamo della famiglia resnet e l'efficienza per mobilenet. Queste reti pronte all'uso, hanno il vantaggio rispetto ad una rete personalizzata di generalizzare una classificazione con meno passaggi, essendo già state addestrate su altri dati. Questa tecnica si chiama transfer learning e consiste nell'utilizzare reti addestrate con altri dati, dunque con i valori dei pesi degli archi non inizializzati casualmente, e nell'aggiungere un livello all'output per far convergere i dati di input nelle classi utilizzate nel nostro problema [9].

La rete personalizzata è composta da una sequenza di operatori lineari che prendono in ingresso degli input di una determinata dimensione e li restituiscono in output con

un'altra dimensione. Ognuno di questi operatori lineari costituisce un livello nella rete e ad ogni livello della rete si ha che questo è fortemente connesso con il livello precedente e quello successivo. La nostra rete nello specifico è formata da 3 livelli. Tra il primo ed il secondo livello viene utilizzata la funzione ReLU, chiamata anche rettificatore. Questa funzione detta anche di attivazione, restituisce il valore positivo dell'argomento. È stato dimostrato che l'utilizzo di questa funzione permette di ottenere risultati migliori in una DNN (Deep Neural Network) [10]. Tra il secondo ed il terzo livello si utilizza la funzione dropout che previene un sovra adattamento della rete al set di addestramento, andando ad inibire alcuni neuroni, con la conseguente impossibilità di generalizzare il problema in maniera corretta [11].

La struttura della rete personalizzata è la seguente:

```
class OilNet(nn.Module):
  def __init__(self, num_features, num_classes):
    super(OilNet, self).__init__()
    self.classifier = nn.Sequential(
        nn.Linear(num_features, 512),
        nn.ReLU(),
        nn.Linear(512, 64),
        nn.Dropout(.5),
        nn.Linear(64, num_classes)
        )

  def forward(self, x):
    return self.classifier(x)
```

3.2.2 Gli iperparametri

Un compito fondamentale per la riuscita di questo esperimento è stato quello di cercare e trovare iperparametri che riuscissero a fornire dei risultati soddisfacenti sia in fase di apprendimento che in quella di test. Prima di continuare la discussione sulla ricerca di questi si vuole definire cos'è un iperparametro. In machine learning si tratta di un parametro che è stato impostato prima dell'inizio dell'apprendimento e ha il

compito di regolare questa fase, inoltre la loro scelta può incidere sulle prestazioni della classificazione [12]. Tra questi troviamo:

- `batch_size=1`

  È il numero di dati che la rete riceve come input prima di andare ad aggiornare i propri parametri interni. La scelta del valore 1 è stata presa dopo diversi test. Si è vista una maggiore regolarità nell'aumentare dell'accuratezza rispetto ad altri valori testati in precedenza.

- `learning_rate=.0001`

  Quest' iperparametro sta ad indicare lo step size di modifica dei pesi all'interno di una rete neurale. Questi pesi vengono modificati, ad ogni iterazione, per cercare di minimizzare il valore della loss. Durante le varie fasi di test sono stati utilizzati diversi valori di learning rate, ma il learning rate che ha permesso di ottenere accuratezze maggiori è stato il valore 0,0001 [13].

- `optimizer = optim.Adam(model.parameters(), lr=learning_rate)`

  Lo scopo principale di un un algoritmo di machine learning è quello di creare una funzione di ottimizzazione che vada a minimizzare la funzione di loss. Adam è un metodo di ottimizzazione che va autonomamente a modificare il learning rate secondo alcune sue stime interne [14].

- `num_epochs=100`

  È il numero di volte che la rete effettua l'apprendimento sullo stesso set di dati. Questo parametro è stato modificato diverse volte durante l'intera fase di sperimentazione in quanto le reti off-the-shelf di cui si è parlato in precedenza, hanno prestazioni molto differenti rispetto alla rete personalizzata. Basti sapere che per effettuare un addestramento con un num_epoch pari a 60 utilizzando resnet50 ci vuole oltre un'ora, invece utilizzando la rete personalizzata ci vogliono pochi minuti. Il valore per la maggior parte della fase di test di questo iperparametro è 100 ma in alcuni casi varia a seconda delle prestazioni della rete.

- `_, c = np.unique(d.y.numpy()[d_train.indices], return_counts=True)`

```
loss_w = torch.from_numpy(np.min(c) / c)
criterion = CrossEntropyLoss(weight=loss_w)
```

I primi due assegnamenti vanno a calcolare la distribuzione di ogni classe all'interno del set di addestramento. Questa verrà utilizzata all'interno della funzione crossEntropyLoss che ha lo scopo di determinare quanto il modello stia sbagliando rispetto al valore reale (ground truth) [15].

Capitolo 4

La fase di sperimentazione

All'interno di questo capitolo verranno discussi i risultati ottenuti dalla sperimentazione, con un confronto di quelli trovati utilizzando altre tecniche. Seguiranno alcune considerazioni sui modelli usati soffermandosi in particolar modo sulle reti.

4.1 Risultati ottenuti

La fase di test di questo esperimento si è svolta utilizzando le medesime dimensioni degli insiemi di dati utilizzate da altri team di ricerca , ad esclusione degli ulteriori test che sono stati fatti come già introdotto nella sezione 1.3 del capitolo 1.

Utilizzando l'intero dataset i dati sono stati suddivisi nel seguente modo:

- Set di addestramento composto da 905 elementi
- Set di valutazione composto da 100 elementi
- Set di test composto da 295 elementi

I vari set di dati, tutti indipendenti tra loro, hanno permesso di ottenere i risultati descritti in tabella 4.1.

rete utilizzata	v. medio accuratezza	v. min accuratezza	v. max accuratezza	varianza
personalizzata	70,7%	67,8%	74,2%	4,09
*resnet*50	62,95%	59%	70%	14,46
*resnet*18	42.6%	34,6%	55,9%	54,4
mobilenet	36,9%	31,2%	42%	13,85

Tabella 4.1: Risultati ottenuti utilizzando l'intero dataset.

In seguito è stata verificata l'accuratezza su un sottoinsieme del dataset contenente classi equilibrate (tabella 4.2).Questa ulteriore sperimentazione è stata effettuata in quanto, nonostante venisse utilizzata una loss pesata per la fase di apprendimento, si voleva verificare se l'accuratezza in fase di test, con numero di elementi per ogni classe simile, potesse migliorare.

In questo caso i set di dati utilizzati per le varie fasi di questo test sono:

- Set di addestramento composto da 525 elementi
- Set di valutazione composto da 50 elementi
- Set di test composto da 100 elementi

rete utilizzata	v. medio accuratezza	v. min accuratezza	v. max accuratezza	varianza
personalizzata	66,8%	63%	71%	8,95
*resnet*50	59,25%	49%	69%	78,18
*resnet*18	37,96%	32%	55%	74,16
mobilenet	31%	27%	36%	11,5

Tabella 4.2: Risultati ottenuti utilizzando il dataset con numero di dati per ogni classe equilibrato.

Un'ulteriore sperimentazione effettuata su dei sottoinsiemi del dataset iniziale è stata quella di confrontare coppie di classi. La sperimentazione su questo sottoinsieme è stata molto utile a capire quali erano le classi più difficilmente separabili. I risultati che riportati in tabella 4.3 riportano infatti una difficoltà maggiore a riconoscere le classi

di difetto mi ed ox. Questo probabilmente è causato dal fatto che i difetti in questione hanno caratteristiche simili, quindi difficilmente separabili.

Classi confrontate	Accuratezza	N. train	N. val.	N. test
$EV - MI$	$86,4\%$	800	50	187
$EV - OX$	$82,7\%$	550	50	187
$MI - OX$	$70,3\%$	550	50	138

Tabella 4.3: Risultati ottenuti utilizzando coppie di classi del dataset.

I risultati riportati per questo test sono stati elaborati dalla rete personalizzata, poiché questo è servito solo ad apprendere meglio quali fossero le classi che le reti neurali facessero più fatica a riconoscere. Le accuratezze delle altre reti sono risultate in linea a come visto nei precedenti test, dunque inferiori alla rete personalizzata.

4.2 Confronto dei risultati

Le reti utilizzate per questa sperimentazione hanno tutte caratteristiche diverse tra loro e come potevasi prevedere, hanno dato performance in termini assoluti molto differenti. Durante la fase precedentemente descritta le reti resnet18 e mobilenet hanno avuto performance in termini di accuratezza delle predizioni nettamente inferiori alle altre altre. Un ulteriore confronto che può essere fatto è proprio tra resnet50 e la rete personalizzata in quanto queste due reti, nonostante abbiano peculiarità diverse, si sono comportate in maniera simile sia nella scelta di alcuni iperparametri che in termini di accuratezza. La rete resnet50 ha avuto, in tutte le fasi di test, un'accuratezza leggermente minore rispetto alla rete personalizzata, ciò si può evidenziare facendo riferimento ai valori della tabella 4.1, che differiscono approssimativamente di 7 punti percentuale. Analizzando anche i risultati del test su classi equilibrate, si può notare in tabella 4.2 un'instabilità maggiore ed un'accuratezza delle reti inferiore, che può essere causato dal basso numero di campioni utilizzati per la fase di apprendimento. Grazie ai risultati ottenuti, sfruttando diverse tipologie di rete, siamo in grado di dire che per la classificazione dell'olio non vi è necessità di utilizzare una rete neurale convoluzionale, ma è più

vantaggioso utilizzare reti che separino le classi utilizzando operatori lineari, proprio come fa la nostra rete personalizzata. La ricerca appena esposta, la quale offre metodi per supportare i panel test, può essere paragonata con l'attuale stato dell'arte. Dai risultati ottenuti da questi due metodi si evince una minor accuratezza con l'utilizzo delle reti neurali rispetto a metodi statistici come PCA-LDA. Tuttavia il confronto che abbiamo potuto fare non è completamente accurato dal momento che non abbiamo informazioni precise che riguardano come nell'articolo [5] siano stati definiti i tre insiemi di addestramento valutazione e test. L'unico criterio che abbiamo potuto utilizzare per il confronto è stato quello di utilizzare insiemi con le medesime cardinalità di quelle usate nell'articolo appena citato. Il vantaggio apportato dall'utilizzo dalle reti neurali, nonostante la mancanza di conoscenze del dominio della chimica, permette di fare una classificazione dell'olio in base alle concentrazioni molecolari e di avere di conseguenza una stima della qualità di esso.

Conclusioni

Il lavoro svolto ci ha permesso di trovare un ulteriore metodo per la classificazione dell'olio. Questo strumento basato sulle reti neurali, si è dimostrato molto affidabile e piuttosto preciso. I test effettuati sulla rete personalizzata hanno prodotto risultati con accuratezze sempre superiori al 63% raggiungendo talvolta picchi del 74%. I test effettuati sulle reti neurali convoluzionali invece, hanno prodotto risultati decisamente peggiori, molto altalenanti e ciò mostra come questa tipologia di rete neurale sia inadatta in quest'ambito rispetto alla rete personalizzata da noi adoperata. Lo svantaggio potrebbe assottigliarsi nel caso si trovasse una trasformazione che non vada ad altererare in alcun modo l'accuratezza della predizione, come potrebbe fare quella da noi applicata per rendere compatibile il dataset all'input richiesto da esse. In generale i risultati ottenuti dalla prima rete citata, offrono un buon punto di partenza per ulteriori ricerche in questo campo, ipotizzando inoltre che con un numero di oli superiore per l'addestramento, in modo da catturare meglio le relazioni tra le molecole, si riuscirebbe ad aumentare l'accuratezza del nostro strumento. Infine si vuole aggiungere che questo progetto potrebbe aprire nuove collaborazioni con altri team di ricerca per riuscire a creare uno strumento che possa dare un paragone di confronto affidabile tra il panel test e i risultati del nostro modello. A tal proposito si presuppone che questo lavoro potrebbe essere portato avanti utilizzando un dataset di dimensioni maggiori per riuscire ad aumentare l'accuratezza del modello, integrando i metodi statistici già utilizzati con il nostro modello e infine avere un supporto da persone esperte nel campo della chimica per provare a diminuire il numero di features su cui il modello deve lavorare per eliminare quelle che creano disturbo nell'apprendimento.

Bibliografia

[1] Marco Gori. Introduzione alle reti neurali artificiali. *Mondo digitale*, 2(8):4–20, 2003.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[4] Commission regulation (eec) no 2568/91 of 11 july 1991 on the characteristics of olive oil and olive-residue oil and on the relevant methods of analysis, 2015.

[5] Lorenzo Cecchi, Marzia Migliorini, Elisa Giambanelli, Adolfo Rossetti, Anna Cane, Fabrizio Melani, and Nadia Mulinacci. Headspace solid-phase microextraction-gas chromatography-mass spectrometry quantification of the volatile profile of more than 1200 virgin olive oils for supporting the panel test in their classification: Comparison of different chemometric approaches. *Journal of Agricultural and Food Chemistry*, 67(32):9112–9120, 2019. PMID: 31314506.

[6] Lorenzo Govoni. Architettura di una rete neurale convoluzionale.

[7] Antonino Staiano. Introduzione alle reti neurali ed applicazioni in campo ambientale.

[8] Zo Angelo Rakotoarivony. Image analysis con tecniche di deep learning.

[9] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[10] Wikipedia contributors. Rectifier (neural networks) — Wikipedia, the free encyclopedia, 2020. [Online; accessed 24-June-2020].

[11] Wikipedia contributors. Dilution (neural networks) — Wikipedia, the free encyclopedia, 2020. [Online; accessed 24-June-2020].

[12] Gang Luo. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5(1):18, 2016.

[13] Wikipedia contributors. Learning rate — Wikipedia, the free encyclopedia, 2020. [Online; accessed 25-June-2020].

[14] Wikipedia contributors. Stochastic gradient descent — Wikipedia, the free encyclopedia, 2020. [Online; accessed 25-June-2020].

[15] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018.

Ringraziamenti

Ringrazio il Prof. Gabbrielli per avermi guidato e supportato in una fase così importante della mia carriera. Un sentito ringraziamento va anche al Dott. Stefano P. Zingaro per i suoi preziosi consigli e la sua pazienza che mi ha permesso di poter scrivere questa tesi. Un altro ringraziamento molto importante va all'azienda Carapelli Firenze S.p.A. per averci fornito il dataset degli oli senza i quali questa ricerca non sarebbe esistita, e ai ricercatori Lorenzo Cecchi, Marzia Migliorini, Elisa Giambanelli, Adolfo Rossetti, Anna cane Fabrizio Melani e Nadia Mulinacci precursori nella ricerca nel campo della classificazione degli oli tramite supporti tecnologici che hanno ispirato il lavoro svolto in questa tesi. Un grande ringraziamento va alla mia famiglia che mi ha sempre supportato nella mia carriera accademica, a Gaia, che mi ha supportato in tutti i momenti, belli e brutti, compresa la scrittura di questa tesi, ai membri della "pyrofile" e a tutti gli altri compagni universitari che ho avuto il piacere di conoscere, che mi hanno aiutato a crescere e a giungere alla fine di questo percorso e per ultimi vorrei ringraziare i miei amici Samuele, Riccardo, Davide e tutte le persone che mi sono state vicino e mi hanno aiutato a diventare chi sono ora. Grazie a tutti.