Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

Sviluppo di CNN per l'analisi di spettri Raman a scopo diagnostico, a partire da reti note utilizzate per Computer Vision

Rodolfo CAROBENE - Mat: 838092 - r.carobene@campus.unimib.it
Emanuele PALUMBO - Mat: 825945 - e.palumbo3@campus.unimib.it

24 gennaio 2023

Indice

Abstract

Durante la pandemia COVID-19 sono state sviluppate diverse tecniche di *Machine Learning* con lo scopo di effettuare una veloce e precisa identificazione di soggetti positivi. Ottimi risultati sono stati raggiunti dall'utilizzo di *Convolutional Neural Networks* (CNN) applicate allo studio di spettri Raman ottenuti da analisi della saliva.

In questo progetto sono stati analizzati diversi modelli di CNN, comparandone le performance con una rete convoluzionale "standard" e già testata come modello di benchmark. In particolare sono state sviluppate delle reti neurali ispirate da modelli notevoli che hanno già dimostrato ottime *performance* in task di *Computer Vision*: ResNet, VGG e DenseNet. Queste reti, sviluppate per immagini tridimensionali, sono state prese a ispirazione per la definizioni di reti adatte allo studio di spettri monodimensionali. Infine, è stata anche sviluppata e testata una rete con input bidimensionale che considera non il singolo spettro come input, ma l'insieme di tutti gli spettri di un singolo paziente.

Le reti a input 1D hanno portato ad *accuracy* del $60 - 70\%$ in classificazione ternaria (COV+ Vs COV- Vs CTRL) e del $70 - 80\%$ in classificazione binaria (COV+ Vs COV- & CTRL). I risultati della rete a input 2D, invece, sono stati molto inferiori: *accuracy* ternaria del 42% e binaria del 69%.

É importante specificare che il dataset utilizzato era molto ridotto e che per questo progetto è stato utilizzata la versione gratuita di GoogleColab: con un dataset più ampio e con più risorse computazionali a disposizione, è probabile che sarebbero state raggiunti migliori risultati.

1 Introduzione

La spettroscopia Raman [1] è una tecnica di analisi dei materiali basata sulla semplice diffusione di luce monocromatica sul materiale stesso. Analizzando le diverse frequenze di luce diffusa, infatti, è possibile ricostruire le componenti principali del materiale in oggetto. Sebbene non sia completamente noto come fenomeno, diversi studi hanno portato a collegare con precisione diffusioni a specifiche frequenze con la presenza di particolari materiali [2].

L'utilizzo della spettroscopia Raman per scopo diagnostico è di evidente interesse: supponendo sia possibile realizzare un metodo di analisi abbastanza preciso, infatti, si avrebbe una diagnosi rapida, assolutamente poco invasiva e facilmente generalizzabile a diverse tipologie di infezione o malattie. Per questo motivo, durante la pandemia COVID-19, si è cercato di sviluppare dei modelli di *Deep Learning* per la classificazione diagnostica di pazienti positivi al Covid o non, tramite l'utilizzo di spettri Raman ottenuti da saliva.

Il progetto in discussione ha origine dall'analisi di un modello di *Deep Learning* (basato su *Convoluzional Neural Networks*) sviluppato per l'analisi e la classificazione di tali spettri. In particolare è stato preso come punto di partenza e riferimento [3], articolo pubblicato su Nature nel 2019 dove si analizza l'uso di una rete convoluzionale proprio per questo problema.

Utilizzando il modello proposto in [3] come punto di riferimento, nel corso di questo progetto sono state sviluppate diverse reti neurali con ispirazioni dal campo della *Computer Vision*, in particolare da ResNet, VGG e DenseNet. Un'ultimo tentativo di nuovo sviluppo è inoltre stato eseguito considerando una rete che considerasse in ingresso molteplici e differenti spettri ottenuti da un solo paziente.

Il report è strutturato come segue: in sezione 2 è riportata un'approfondita descrizione del dataset in analisi, seguita da una descrizione del *data preprocessing* utilizzato e da una spiegazione delle modalità di intepretazione dei risultati[1]; in sezione 3, invece, è descritto lo sviluppo e il testing dei vari modelli utilizzati seguito, in sezione 4, da una fase comparativa; le fila del progetto sono infine tirate nella sezione conclusiva (sezione 5).

Tutti i codici utilizzati sono reperibili alla repository di GitHub [4].

2 Considerazioni generali

2.1 Analisi del Dataset e Preprocessing

Il dataset comprende 2501 spettri Raman provenienti dalla saliva di 101 pazienti. Da ogni paziente sono stati ottenuti 25 diversi spettri, fatta eccezione per tre pazienti con 20 spettri e un ultimo paziente con solo 16. I pazienti sono divisi in tre categorie: pazienti positivi al COVID, identificati come **COV+**; pazienti negativizzati, identificati con **COV-**; e pazienti che non hanno mai contratto l'infezione, identificati con **CTRL**.

La distribuzione dei pazienti fra le diverse classi è riportata in tabella 1.

	COV+	COV-	CTRL
Numero pazienti	30	37	34

Tabella 1: Distribuzione dei pazienti fra le diverse classi. 'COV+' indica pazienti positivi al COVID, 'COV-' pazienti in passato positivi e ora negativi e 'CTRL' indica pazienti mai infetti.

Il database contiene, per ogni spettro, informazione su:

- lo 'user', cioè il codice identificativo per ogni paziente;
- la 'category', quindi la classe a cui appartiene il singolo spettro;
- 'spectra', ovvero i valori delle ordinate dello spettro espresse in unità arbitrarie;
- 'x-axis', ovvero i valori dell'asse delle ascisse dello spettro, espresso in cm^{-1}.

[1] Il fatto che l'analisi venisse fatta su un singolo spettro, ma un singolo paziente potesse avere diversi spettri ha portato alla definizione di due metodi di aggregamento dei risultati.

In seguito una rappresentazione dell'intero dataset e degli esempi di spettri divisi per categorie e paziente.

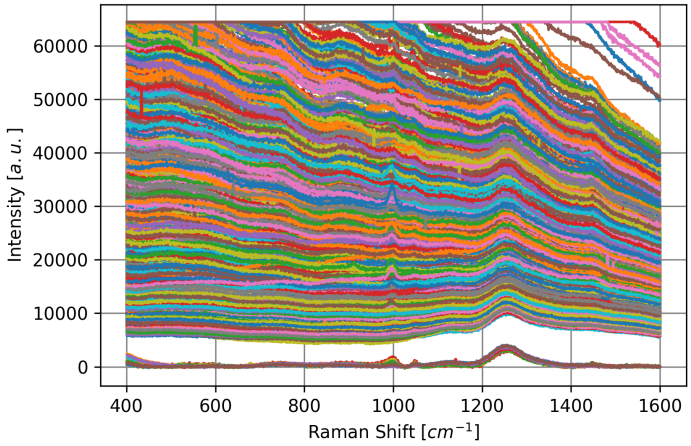
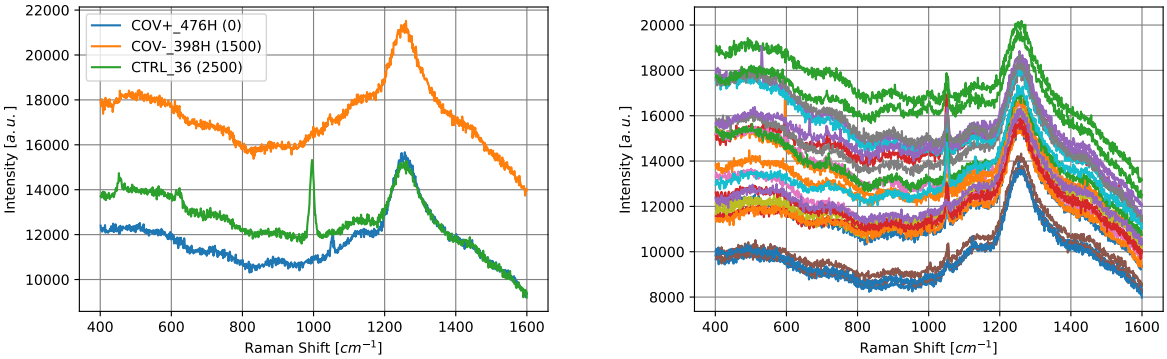


Figura 1: Rappresentazione del dataset completo



(a) (b)

Figura 2: Esempi di spettri dal dataset. In figura (a) un paragone fra tre spettri di diversa categoria; in figura (b) esempio di tutti gli spettri di un singolo paziente.

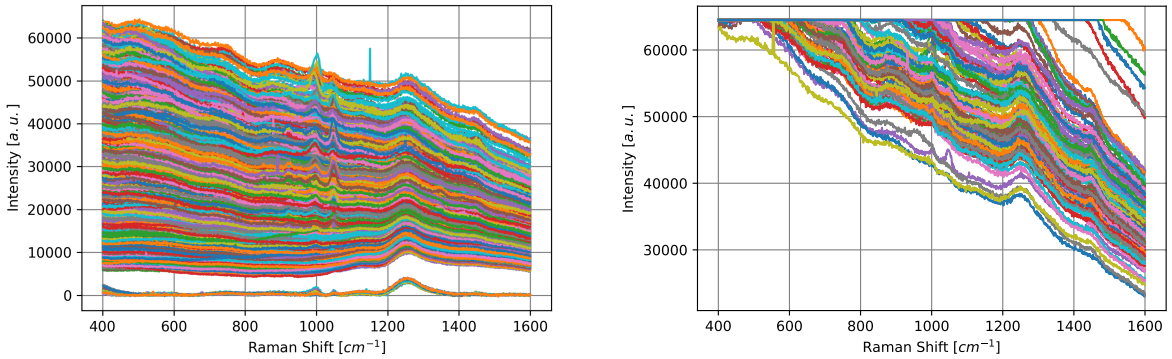
Prima di procedere con la costruzione delle reti neurali è necessario lavorare sui dati in modo da renderli più adatti all'utilizzo di tecniche di machine learning.

2.1.1 Eliminazione degli *outliers*

Il preprocessing inizia con l'eliminazione degli *outliers*, cioè quegli spettri che, almeno in parte, hanno saturato il sistema di acquisizione risultando costanti su parte delle frequenze analizzate. Tali spettri, infatti, non sono utilizzabili nè per l'addestramento nè per il testing di una rete neurale.

Per attuare questo processo è stata implementata una funzione che prende in input il dataset 'raw' ed elimina tutti quegli spettri che raggiungono il livello di saturazione (identificato come $64500\,a.u.$).
Idealmente è anche possibile stabilire una soglia, sull'asse x, prima della quale è possibile accettare valori saturati: se uno spettro ha solo pochissimi punti a $64500\,a.u.$, infatti, non è necessariamente da rimuovere. In ogni caso, dal momento che i picchi più rilevanti di questo problema non sono noti né sono note le rispettive frequenze, si è scelto di non considerare soglie di accettazione (o, equivalentemente, considerare una soglia di $0\,cm^{-1}$).

Questa fase di preprocessing ha portato all'eliminazione di **107 spettri**.



(a) (b)

Figura 3: In figura (a) è rappresentato il dataset senza *outliers*. In figura (b) Sono rappresentati gli *outliers* eliminati.

2.1.2 Pseudo-normalizzazione

In generale è opportuno normalizzare i dataset perché gli algoritmi funzionano intrinsecamente meglio con input numerici $\in [0, 1]$. Per il problema in analisi, tuttavia, la normalizzazione funge anche da strumento per uniformare il dataset. Si nota, infatti, che diversi spettri simili in forma hanno intensità molto diverse: ciò è dovuto alle modalità di presa dati, ma è potenzialmente deleterio per un modello di Deep Learning.

Si è tentato di normalizzare dividendo ogni spettro per il suo valore massimo, ma questa modalità ha portato a un peggioramento dei risultati rispetto all'analisi dei dati puramente *raw*. Questo è probabilmente spiegabile notando che il massimo della maggior parte degli spettri è a $1300\,cm^{-1}$ (picco di fondo), ma gli spettri a intensità maggiore sono anche i più inclinati e hanno il massimo a $400\,cm^{-1}$ (il primo punto); la normalizzazione introduce dunque un bias fittizio.

In seguito a diversi tentativi, si è infine scelto di "normalizzare" tutti gli spettri dividendoli per il loro valore a $400\,cm^{-1}$, facendoli dunque partire tutti da un singolo punto iniziale. In questo modo viene trascurata una eventuale dipendenza dall'inclinazione dello spettro stesso.

2.1.3 Data augmentation

Data la dimensiona ridotta del dataset, è stata implementata la tecnica della *data augmentation*, in due modi differenti.

Un primo metodo consiste nel creare, per ogni spettro presente nel training set, altri 5 con l'aggiunta di rumore gaussiano. La deviazione standard scelta non è molto elevata ($\sigma = 0.05$) in modo da non coprire eventuali picchi potenzialmente essenziali per la classificazione.

Malgrado questo metodo sia più classico, generalmente è stata utilizzata una seconda modalità quasi equivalente: si è aggiunto un layer di GaussianNoise all'inizio della rete neurale, in modo tale gli input venissero ogni volta leggermente modificati.

2.2 Metriche e parametri utilizzati

In tutti i modelli implementati si è scelto di utilizzare come funzione obbiettivo la 'Categorical accuracy', come loss la 'Categorical cross entropy' e come optimizer 'Adam'.

Per la maggior parte dei modelli è stato utilizzato il framework Hyperband [5] per effettuare una scelta guidata degli iperparametri strutturali necessari (numero di filtri, numero di layer...). Maggiori informazioni sono date nelle sezioni specifiche dei modelli.

2.2.1 Classificazione dei pazienti

Nella fase di inferenza, le metriche utilizzate tengono in considerazione i singoli pazienti agglomerando tutti i rispettivi spettri. È dunque necessario definire, anzitutto, come valutare se un paziente appartiene a una delle tre categorie COV+, COV- o CTRL.

L'output dei vari modelli[2] è composto da un vettore a tre dimensioni, in esso è indicata la probabilità che il singolo spettro appartenga a ciascuna delle tre classi. Dati tutti gli spettri di un singolo paziente, per calcolare la categoria di appartenenza del paziente abbiamo definito due metodi:

- Il primo modo è utilizzare una **Majority rule**, in questo caso per ogni spettro si vede a quale classe corrisponde la probabilità maggiore. La classe assegnata al maggior numero di spettri corrisponderà alla classe del paziente in esame.
- Il secondo metodo invece consiste in una **Probability rule**: si considerano tutti gli spettri assieme e si sommano le probabilità per ogni categoria. La classe scelta è quella con il maggior valore della somma delle probabilità.

Chiaramente ci aspettiamo che questi due metodi diano valori abbastanza concordi, ma non è semplice definire a priori quale sia il più efficacie.

[2] Qui vengono considerati solo i modelli a input 1D, per il modello a input 2D la situazione è leggermente diversa e meglio specificata nella sezione dedicata.

2.2.2 Metriche per valutare testing set

A questo punto definiamo le metriche utilizzare per valutare il testing set. La prima è appunto la *categorical accuracy*:

$$\text{Categorical accuracy} = \frac{\text{Previsioni corrette}}{\text{Previsioni totali}} \tag{1}$$

in questo caso il problema presenta una classificazione a tre classi che rivestono la stessa importanza.

Le altre metriche utilizzate, invece, trattano il problema come una classificazione binaria, perché per questo preciso caso si vuole dare una maggiore importanza nel riconoscimento dei casi COV+. Per farlo si studiano le classi COV- e CTRL come una unica, in modo da non tener conto di eventuali errori di riconoscimento della rete tra queste due. Con questo approccio abbiamo definito l'*Accuracy* ("binaria"), la *Sensistivity* e la *Specificity*, rinominandole 'COV+ Accuracy', 'COV+ Sensitivity' e 'COV+ Specificity'.

$$\text{COV+ Accuracy} = \frac{\text{previsioni COV+ corrette}}{\text{previsioni totali}} \tag{2}$$

Definiamo *True positive*, *False positive*, *True negative* e *False negative* in funzione della classe dei positivi COV+, infatti si considerano gli eventi positivi quelli COV+ e negativi le altre classi. In questo modo le metriche rimaste hanno le seguenti definizioni:

$$\text{COV+ Sensitivity} = \frac{\text{COV+ True positive}}{\text{COV+ True positive + COV+ False negative}} \tag{3}$$

$$\text{COV+ Specificity} = \frac{\text{COV+ True negative}}{\text{COV+ True negative + COV+ False positive}} \tag{4}$$

2.3 K-Fold Cross-Validation

Data la dimensione ristretta del dataset, si è ritenuto necessario l'utilizzo di una procedura di K-fold cross-validation. Nell'articolo guida per questo progetto si utilizzava una procedura leave-one-out modificata in modo da funzionare patient-wise: veniva dunque escluso dal training set un paziente alla volta (cioè tutti i suoi 25 spettri).

Per limiti computazionali non è stato possibile utilizzare la medesima procedura ed è stato necessario allargare i vari test-set. In particolare i test-set sono stati sempre composti da un minimo di 3 pazienti (uno per classe) e un massimo di 4 pazienti (uno per classe più uno preso fra COV- e CTRL, le classi con più pazienti).

3 Analisi di diverse tipologie di CNN

In questa sezione siamo partiti col definire il modello di benchmark (semplice CNN) e dall'osservare i suoi risultati sul dataset in analisi. Successivamente abbiamo utilizzato Hyperband per ottimizzare delle nuove reti a input 1D, costruite a partire da reti note in *Computer Vision*. Successivamente all'ottimizzazioe della rete stessa, si è testata la rete sul medesimo dataset.

Per una comparazione dei risultati rimandiamo alla sezione 4.

Per i grafici seguenti, verrà utilizzata sempre la stessa notazione in colore descritta in fig. 4.

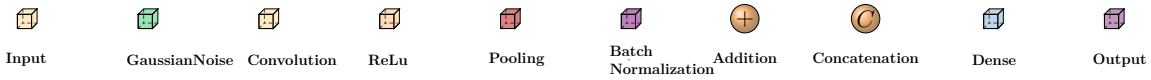


Figura 4: Legenda

3.1 CNN (benchmark)

In questa sezione abbiamo costruito un modello di rete convoluzionale da utilizzare come "benchmark". Questa modello è stato implementato a partire da quello proposto dall'articolo di guida [3].

3.1.1 Definizione del modello

Il modello è composto dai seguenti layers:

- Il primo layer è un GaussianNoise utilizzato per pseudo data-augmentation;
- La parte centrale è composta da 3 layer convoluzionali, i primi 2 con 100 filti, mentre il terzo con 25. La funzione di attivazione scelta è la ReLU.
- A questi si aggiungo due MaxPooling tra il secondo e il terzo e dopo il terzo.
- La rete si chiude con un Flatten layer seguito da 3 Dense layer di dimensione decrescente, anche in questo caso la funzione di attivazione è la ReLU
- La rete si chiude con un output layer di dimensione 3, a cui viene applicata una softmax come funzione di attivazione.

Per ragioni di regolarizzazione durante la fase di training sono stati aggiunti dei layer di Dropout per i Dense layer ed è stata utilizzata la tecnica di L2 *weight-penalization*.

Il modello è visibile in fig. 5.

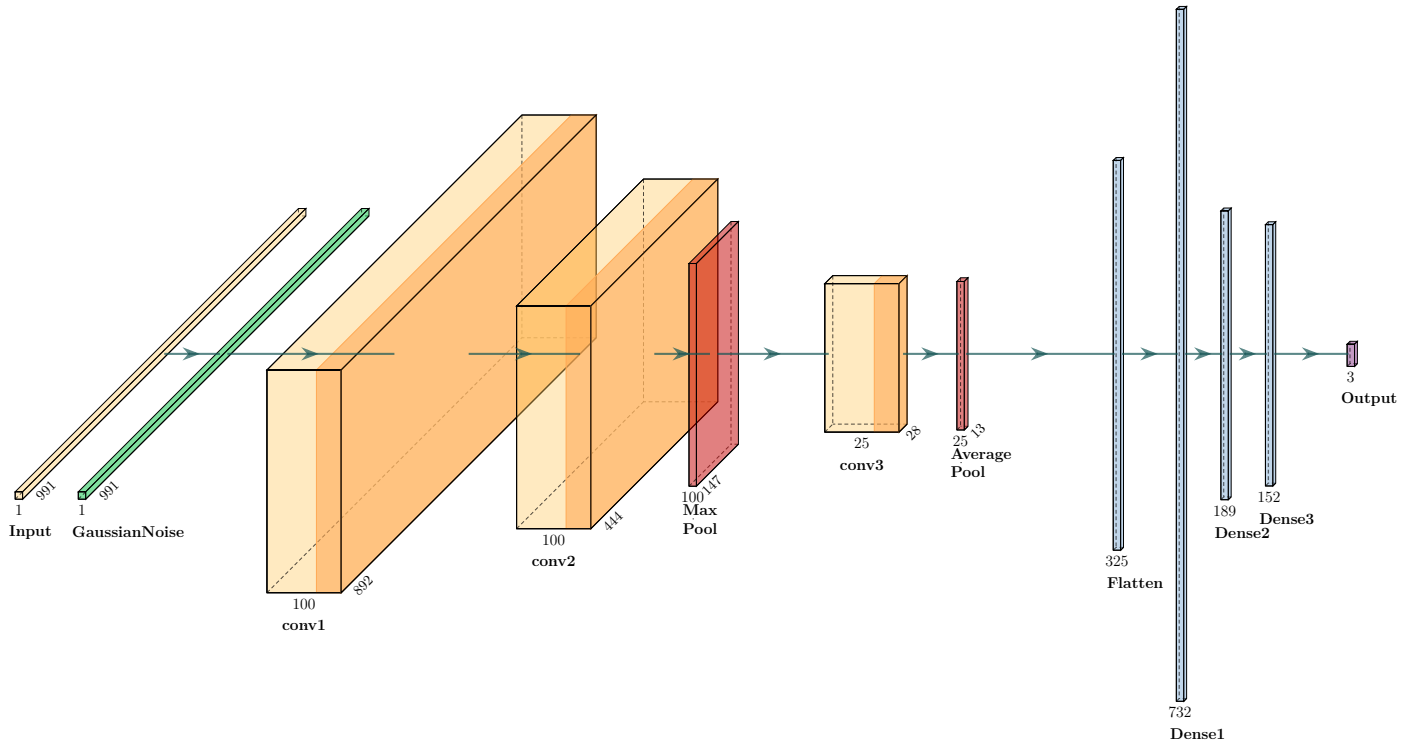


Figura 5: Rappresentazione grafica del modello di Benchmark

3.1.2 Training e testing

Il training del modello è stato compiuto con il layer di GaussianNoise iniziale e con una *data-augmentation* standard, i risultati ottenuti sono stati equivalenti.

Accuracy (%)	COV+ Accuracy (%)	COV+ Sensitivity (%)	COV+ Specificity
58	72	47	85

Tabella 2: Risultati training

3.2 ResNet

Le *Residual Neural Network* (ResNet) [6], sono reti neurali dove, in aggiunta al semplice modello *feedforward*, sono presenti particolari connessioni dette *shortcuts*. Tali connessioni permettono al modello di essere estremamente profondo, con decine di layer successivi, senza incorrere eccessivamente nel problema dei *vanishing gradients* che tipicamente limita la profondità di una rete neurale.
Una ResNet è definita da specifici ResidualBlock che contengono ciascuno un certo numero di layer interni e si ripetono lungo la rete. Le *shortcuts* vanno a sommare all'output di un ResidualBlock il suo input in modo tale da facilitare e velocizzare l'ottimizzazione.

3.2.1 Definizione del modello

Il modello implementato è stato ricavato a partire da ResNet50, una rete convoluzionale profonda 50 layer. Per prima cosa, dunque, è stato necessario modificare la rete in modo tale da considerare un input monodimensionale. Inoltre, ResNet50 considera 4 sezioni con all'interno ciascuna dei ResidualBlock differenti, mentre in questo caso si è scelto di ridurre questo numero a 2 in modo da contenere il numero di parametri.

A questo punto si è utilizzato Hyperband per indentificare i migliori iperparametri strutturali della rete: in particolare il numero di ResidualBlock per le due tipologie e la dimensione dei filtri per il primo layer convoluzionale. Il numero di filtri per ogni rete è stato fissato a 64 per la prima tipologia di ResidualBlock e a 128 per la seconda[3]
In aggiunta a questi parametri si è anche ottimizzato il valore di σ per un layer iniziale di GaussianNoise, utilizzato per pseudo data-augmentation.

I valori utilizzati per l'ottimizzazioni e quelli finali ottenuti sono visibili in tabella 3.
La rete finale, invece, è visibile in fig. 6.

	Min	Max	Step	Final value
# ResidualBlock tipo 1	24	1	2	4
# ResidualBlock tipo 2	24	1	2	5
Initial Kernel size	1	12	1	10
GaussianNoise (σ)	0.0	0.1	0.005	0.01

Tabella 3: Iperparametri scelti tramite Hyperband, in particolare sono visibili i limiti scelti per il tuning e i valori finali.

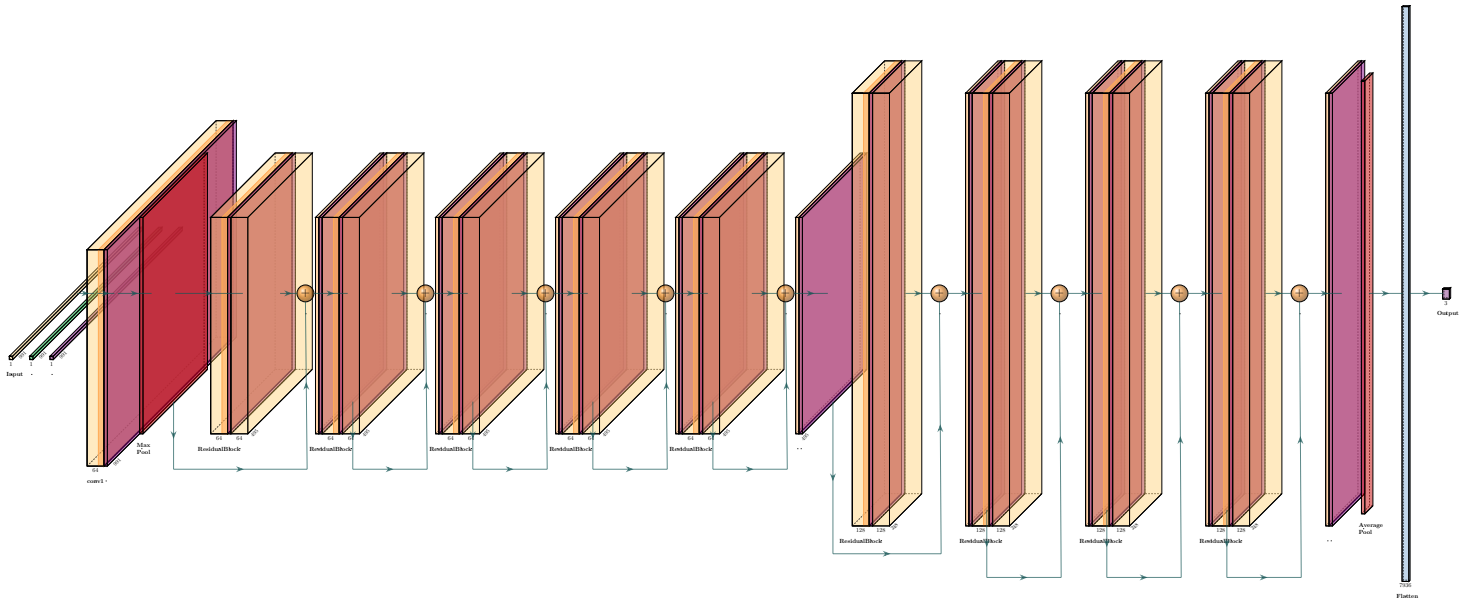


Figura 6: Rappresentazione grafica del modello finale ResNet-inspired

[3]Si è tentato di ottimizzare tramite Hyperband anche questi parametri, ma le risorse computazionali disponibili non erano sufficienti.

É interessante e utile analizzare con più attenzione come è composto un singolo ResidualBlock:

1. Un layer convoluzionale con ReLU come funzione di attivazione;
2. Un layer di BatchNormalization;
3. Un secondo layer convoluzionale con ReLU come funzione di attivazione;
4. Un secondo layer di BatchNormalization;
5. Un layer convoluzionale senza funzione di attivazione;
6. La fine di una *shortcut* che somma l'ultimo layer e l'output della BatchNormalization al punto 2.

La rete così costruita ha **529,477** parametri.

3.2.2 Training e testing

La fase ha portato ai risultati visibili in tabella 4.

	Accuracy (%)	COV+ Accuracy (%)	COV+ Sensitivity (%)	COV+ Specificity
Majority Rule	65	75	63	82
Probability Rule	67	75	60	83

Tabella 4: Risultati training

Rispetto al modello di benchmark illustrato in sezione 3.1 la ResNet utilizzata raggiunge livelli di *accuracy* e *senstivity* migliori con un numero di parametri comparabile.
Si è invece riscontrato un aumento del tempo necessario per il training, senza contare quello utilizzato per l'ottimizzazione degli iperparametri.

3.3 VGG

Le *Visual Geometry Group Neural Networks* (VGG) [7], sono reti neurali sviluppate per l'analisi delle immagini nell'ambito della competizione *imagenet*.

I modelli VGG sono tipicamente mediamente profondi (reti notevoli hanno 16-19 layer) e sono composti da un alto numero di layer convoluzionali inframezzati da MaxPooling layer che vanno a ridurre la dimensione. Le reti VGG terminano poi con due Dense layer anche molto larghi e che tipicamente introducono da soli milioni di parametri.

3.3.1 Definizione del modello

Il modello implementato è stato ricavato a partire da VGG16 (o quantomento è stata presa ispirazione da VGG16).
Per prima cosa, come era stato per ResNet, è stato necessario modificare la rete in modo tale da considerare input monodimensionali.
VGG16 contiene 5 blocchi ciascuno composto da layer convoluzionali con numero di filtri fisso (all'interno del blocco), per tenere contenuto il tempo di ottimizzazione e il numero di parametri si è scelto di considerare 3 soli blocchi. La dimensione dei fltri è stata fissata sempre a (3x3).

A questo punto, si è utilizzato Hyperband per scegliere il numero di filtri per ogni blocco e per identificare il numero di neuroni ottimale per i *Fully connected layer* finali.
In aggiunta a questi parametri si è anche ottimizzato il valore di σ per un layer iniziale di GaussianNoise, utilizzato per pseudo data-augmentation.

I valori utilizzati per l'ottimizzazioni e quelli finali ottenuti sono visibili in tabella 5.
La rete finale, invece, è visibile in fig. 7.

	Min	Max	Step	Final value
# Filtri per blocco 1	8	512	^{2}	32
# Filtri per blocco 2	8	512	^{2}	8
# Filtri per blocco 3	8	512	^{2}	8
# Neuroni per Dense	20	120	10	40
GaussianNoise (σ)	0.0	0.1	0.005	0.06

Tabella 5: Iperparametri scelti tramite Hyperband, in particolare sono visibili i limiti scelti per il tuning e i valori finali.

La rete così costruita ha **45,971** parametri, un numero inferiore di un ordine di grandezza sia rispetto alla rete di benchmark sia alla ResNet analizzata nella precedente sezione.

3.3.2 Training e testing

La fase di training è stata svolta tramite la K-Fold validation come indicato in sezione 2.3 e ha portato ai risultati visibili in tabella 6.

	Accuracy (%)	COV+ Accuracy (%)	COV+ Sensitivity (%)	COV+ Specificity
Majority Rule	74	78	63	88
Probability Rule	77	82	66	90

Tabella 6: Risultati training

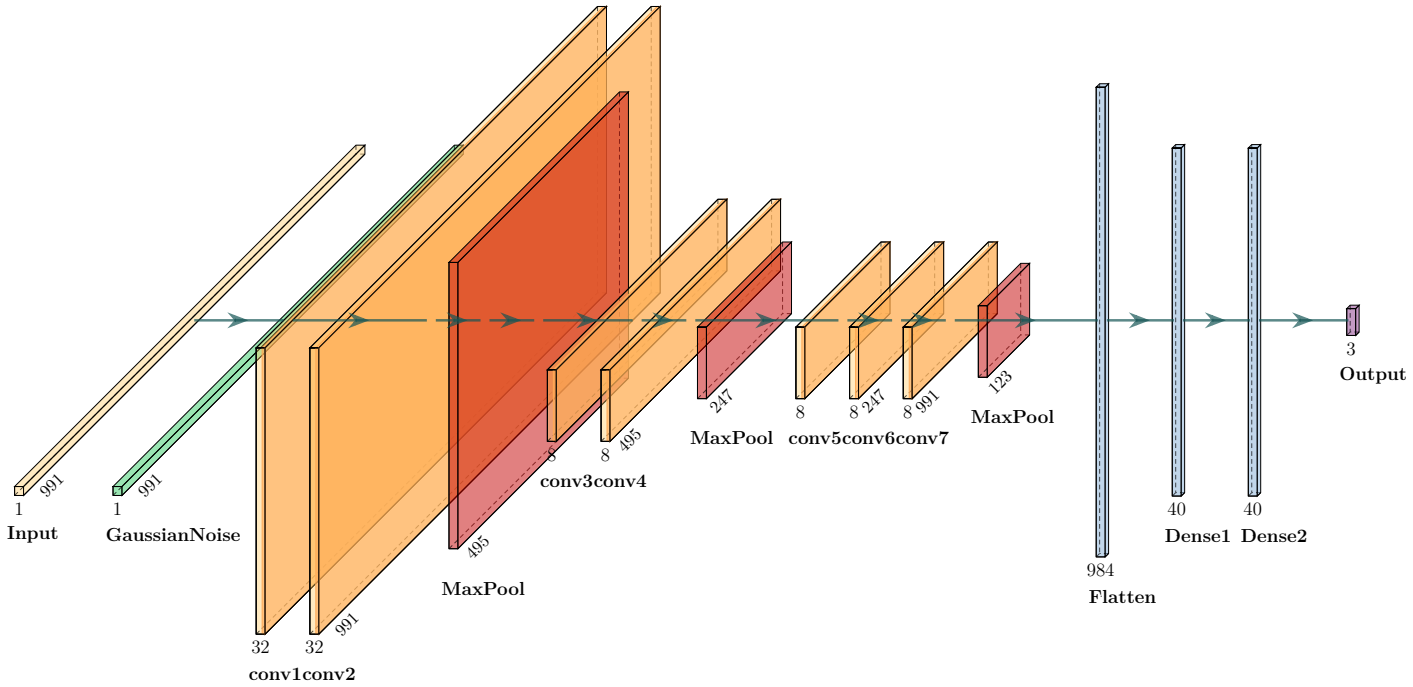


Figura 7: Rappresentazione grafica del modello finale VGG-inspired

Rispetto al modello di benchmark illustrato in sezione 3.1 la rete VGG-inspired utilizzata raggiunge livelli di *accuracy* e *senstivity* migliori con un numero di parametri sensibilmente inferiore.

3.4 DenseNet

Le *Densely Connected Convolutional Neural Networks* (DenseNet) [8] sono reti sviluppate nell'ambito della *Computer Vision* e condividono la medesima idea di fondo di ResNet: introdurre connessioni più corte fra input e output per velocizzare l'ottimizzazione e raggiungere migliori risultati.

ResNet raggiungeva questo obiettivo tramite dei passaggi di Addition (sommando output di layer diversi), invece DenseNet arriva a un equivalente risultato tramite Concatenation layer che aggiungono a un layer non l'output di un layer precedente, ma il layer stesso. In questo modo si stabiliscono delle connessioni "dense": ogni layer di un DenseBlock riceve input da tutti i layer precedenti all'interno del medesimo DenseBlock.

3.4.1 Definizione del modello

I vari modelli DenseNet esistenti sono generalmente consistenti in numero di DenseBlock, ma differiscono per il numero di layer interni al singolo DenseBlock, per questo

abbiamo scelto questo parametro come iperparametro da ottimizzare. Un altro iperparametro da scegliere è il tasso di compressione: i vari DenseBlock hanno infatti una dimensione costante che vieni ridotta fra un blocco e l'altro di un fattore fissato. Infine abbiamo identificato il numero di filtri per layer convoluzionale (valore costante fra i DenseBlock).
Al contrario delle altre reti, DenseNet ha mostrato di generalizzare meglio senza la pseudo data-augmentation ottenuta tramite layer di GaussianNoise.

I valori utilizzati per l'ottimizzazioni e quelli finali ottenuti sono visibili in tabella 7.
La rete finale, invece, è visibile in fig. 8.

	Min	Max	Step	Final value
# Filtri	10	30	2	22
# Layer cnvoluzionali	2	8	2	4
Compression Factor 0.2	0.8	0.1	2	0.5

Tabella 7: Iperparametri scelti tramite Hyperband, in particolare sono visibili i limiti scelti per il tuning e i valori finali.

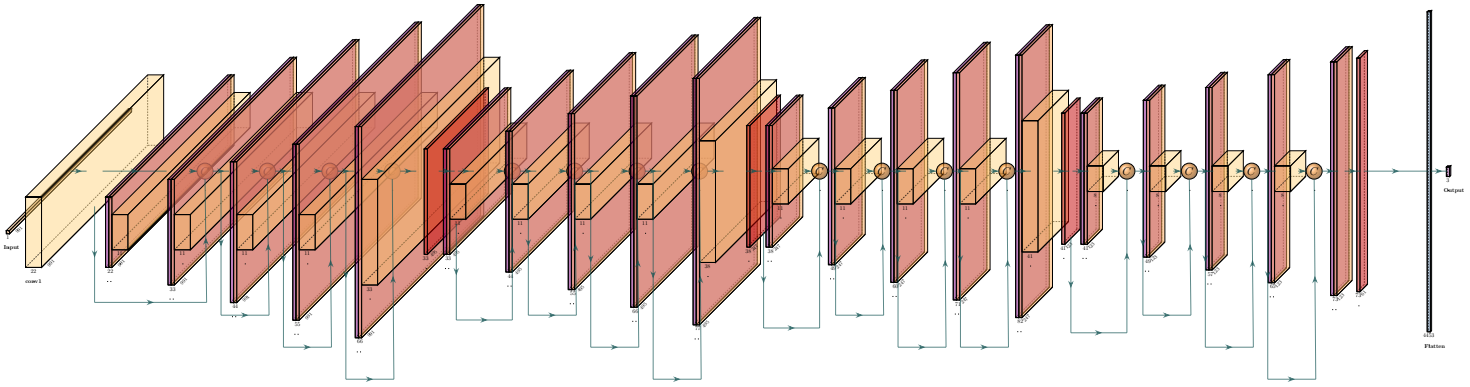


Figura 8: Rappresentazione grafica del modello finale DenseNet-inspired

3.4.2 Training e testing

La fase di training è stata svolta tramite la K-Fold validation come indicato in sezione 2.3 e ha portato ai risultati visibili in tabella 8.

	Accuracy (%)	COV+ Accuracy (%)	COV+ Sensitivity (%)	COV+ Specificity
Majority Rule	47	69	40	83
Probability Rule	49	69	40	83

Tabella 8: Risultati training

3.5 Sviluppo di una rete neurale con input 2D

Tutti i modelli illustrati nelle sezioni precedenti utilizzano come input un singolo spettro Raman monodimensionale; questa modalità è certamente la più intuitiva, ma necessita di particolare attenzione nell'interpretazione dei risultati che vanno prima ottenuti per ogni spettro e poi aggregati per ogni paziente (come illustrato nella sezione 2.2). Un'altra valida possibilità è quella di considerare un input bi-dimensionale ottenuto dall'unione dei 25 spettri di ogni paziente.

3.5.1 Preprocessing specifico

Come illustrato in sezione sezione 2, ad ogni paziente corrisponde un massimo di 25 spettri nel dataset iniziale che a sua volta è composto da 101 pazienti. Dopo l'eliminazione degli *outliers*, tuttavia, non tutti i pazienti risultano ancora avere il medesimo numero di spettri e perciò è necessario applicare un preprocessing specifico in modo tale da ottenere una dimensione di input costante.

In particolare, si è scelto di generare di aggiungere dei dati fittizi per i pazienti che non raggiungevano i 25 spettri.
I nuovi spettri sono stati generati copiando uno spettro valido (non-outlier) già "normalizzato" del paziente in analisi e aggiungendo agli spettri copia del rumore gaussiano con $\sigma = 0.01$.

In questo modo la dimensione di un singolo input è (25, 991, 1).
Il problema che è sorto a questo punto è che il nuovo dataset contiene solo 101 esempi e, siccome è noto che il *deep learning* necessita di una grande mole di dati, si è incorsi in forte problemi di overfitting.

3.5.2 Definizione del modello

Il modello scelto è stato sviluppato con l'idea di mantenere limitato il numero di parametri totale, pur garantendo una certa profondità. La rete è rappresentata graficamente in fig. 9.
In aggiunta a quanto riportato nel grafico, è stata usata una regolarizzazione tramite Dropout (sui Dense layer finali) e L2 (con $l2 = 0.01$).

Tratime Hyperband sono stati identificati i migliori parametri per il Dropout e il layer gaussiano iniziale, le dimensioni dei Dense layer finali e il numero di filtri per blocco. In tabella 9 sono visibili i parametri utilizzati per l'ottimizzazione e i valori finali ottenuti.

	Min	Max	Step	Final value
# Filtri	4	40	2	20
# Dense units (1)	3	128	2	60
# Dense units (2)	3	60	2	20
GaussianNoise (σ)	0.0	0.1	0.01	0.01
Dropout probability	0.0	0.9	0.1	0.4

Tabella 9: Iperparametri scelti tramite Hyperband, in particolare sono visibili i limiti scelti per il tuning e i valori finali.

Analizzando la rete, andando con ordine dal primo all'ultimo layer, troviamo:

- Un GaussianNoise layer ($\sigma = 0.01$) inserito per regolarizzazione e come pseudo data-augmentation;
- Una parte convoluzionale con un alternarsi di layer convoluzionali e averagepooling layers;
- Un flattening layer seguito da due dense layer;
- Un output layer con 3 neuroni finali per la classificazione.

Come funzione di attivazione (per i layer convoluzionali e i dense layer) è stata utilizzata una ReLU, mentre per il layer finale una *softmax*.

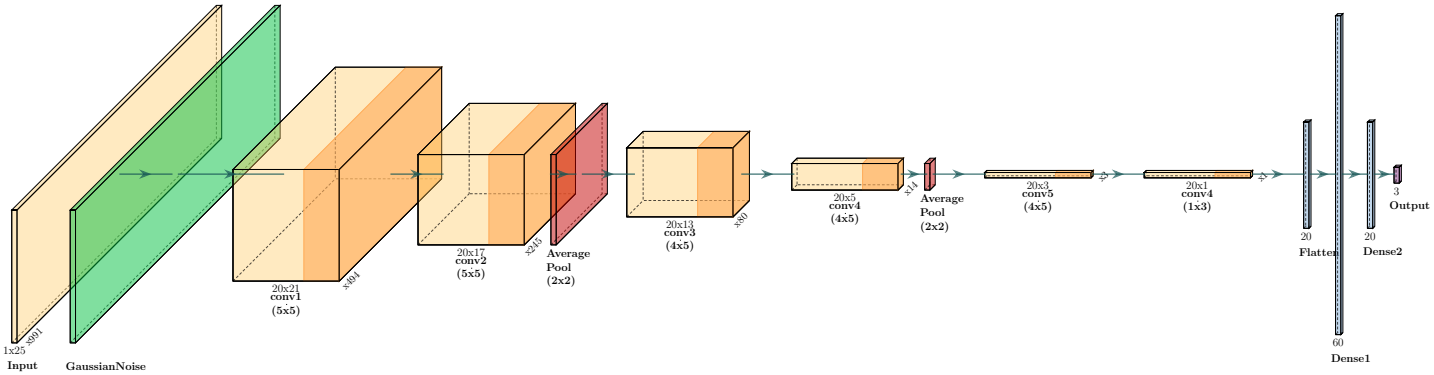


Figura 9: Rappresentazione grafica del modello finale a input 2D

La rete così costruita ha **37,263** parametri. Tale numero potrebbe anche sembrare contenuto, ma considerando che il nostro dataset è composto da solo 101 esempi è comunque eccessivo.

3.5.3 Training e Testing

Il modello è stato allenato tramite K-fold Cross-Validation in modo analogo a quanto già descritto in sezione 2.3, contando però che una cross validation è qui 'naturalmente'

patient-wise. Il numero di epoche per iterazione è stato fissato a 250 con batch_size di 16.

Nonostante tutto, non è stato possibile raggiungere buoni valori di accuratezza. È altamente probabile che ciò sia dovuto, per la maggior parte, alla dimensione molto limitata del dataset siccome il modello riesce a fittare il training set, ma fallisce nella generalizzazione sui test set. I risultati finali sono visibili in tabella 10.

Accuracy (%)	COV+ Accuracy (%)	COV+ sensitivity (%)	COV+ specificity
42	69	16	91

Tabella 10: Risultati training

4 Analisi dei risultati

In tabella 11 sono riportati i risultati di tutte le architetture analizzate. Si è scelto di utilizzare solo la *probability rule* dal momento che ha sempre portato accuracy ternarie migliori, sebbene sempre molto vicine a quelle ottenute dalla *majority rule*.

Si nota che la rete che ha raggiunto le migliori performance è quella VGG-inspired. È necessario, tuttavia, chiarire che nessuno di questi modelli ha probabilmente ottenuto il massimo delle sue performance: in particolare il modello di benchmark. Tale modello, con i parametri dell'ottimizzatore migliori e con l'adeguato preprocessing ha infatti dimostrato di poter raggiungere un'*accuracy* ternaria dell' 89% [3].
Inoltre è l'unico modello a cui, per questo progetto, non è stata tentata un'ottimizzazione degli iperparametri.

	Accuracy (%)	COV+ Accuracy (%)	COV+ Sensitivity (%)	COV+ Specificity
Benchmark	58	72	47	85
Resnet	67	75	60	83
VGG	**77**	**82**	**66**	**90**
DenseNet	49	69	40	83
2D CNN	42	69	16	91

Tabella 11: Analisi dei risultati ottenuti dai diversi modelli (con la probability rule)

5 Conclusioni

Nel corso di questo progetto abbiamo affrontato un problema di classificazione a scopo medico: in particolare per la distinzione di pazienti positivi al covid, negativizzati o mai infetti.

Si è partiti dall'analisi del dataset e dallo sviluppo di una pipeline di preprocessing. In particolare è stato necessario eliminare alcuni spettri (*outlier*) contenenti valori saturati privi di significato e procedere con una pseudo-normalizzazione per uniformare il

dataset. Per il preprocessing è importante notare che quanto svolto per questo progetto è solo minimale e che un preprocessing più approfondito [9] porta probabilmente a migliori *accuracy*.

Il nucleo del progetto, si è invece focalizzato sullo sviluppo e sull'analisi comparativa di diversi modelli di *Deep Learning*: in particolare si sono sviluppate reti ispirate a ResNet, VGG e DenseNet. La comparazione è avvenuta tramite un modello di benchmark ricavato da [3]; tutti i modelli hanno ottenuto risultati almeno compatibili e migliori per la rete VGG-inspired e la ResNet.
Lo sviluppo di una rete a input 2D che considerasse i molteplici spettri di un singolo paziente in contemporanea, infine, ha occupato l'ultima parte del progetto. Tale rete ha dimostrato ottime capacità di adattamento al training set, ma pessime capacità di generalizzazione nella fase di inferenza: è dunque probabile che il limitato numero di paziente nel dataset sia stato il principale fattore limitante.

Per tutte le reti è importante dire che un'analisi più approfondita è necessaria per concludere correttamente la comparazione e decretare in definitiva quale rete sia più appropriata.

In futuro, sarebbe interessante concludere un'ottimizzazione più approfondita delle reti analizzate includendo nell'analisi ulteriori parametri strutturali, ma anche parametri di training come batch_size, numero di epoche, ottimizzatore e rispettivi parametri.
Anche la rete a input 2D è potenzialmente molto interessante, ma un'analisi più approfondita necessita assolutamente di un dataset più ampio o di processi di data-augmentation più specifici.

Riferimenti bibliografici

[1] C. V. Raman and K. S. Krishnan, "A new type of secondary radiation," *Natur*, vol. 121, pp. 501–502, 1928.

[2] K. Virkler and I. K. Lednev, "Forensic body fluid identification: The raman spectroscopic signature of saliva," *Analyst*, vol. 135, pp. 512–517, 2 2010.

[3] C. Carlomagno, "COVID-19 salivary Raman fingerprint: innovative approach for the detection of current and past SARS-CoV-2 infections," *Scientific Reports*, 2021.

[4] C. Rodolfo and P. Emanuele, "aml_project," *GitHub repository*, 2023. [Online]. Available: https://github.com/rodolfocarobene/aml_project

[5] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, and U. Berkeley, "Hyperband: bandit-based configuration evaluation for hyperparameter optimization," *Journal of Machine Learning Research*, 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 12 2015.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 9 2014.

[8] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *CVF Open Access*, 2018.

[9] R. Gautam, S. Vanga, F. Ariese, and S. Umapathy, "Review of multidimensional data processing approaches for raman and infrared spectroscopy," *EPJ Techniques and Instrumentation 2015 2:1*, vol. 2, pp. 1–38, 6 2015.