Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Registrazioni vocali per la diagnosi di COVID-19 con Deep Convolutional Neural Networks

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Correlatore:
Dott.
Stefano Pio Zingaro

Presentata da:
Lorenzo Vainigli

Sessione III
Anno Accademico 2019-2020

A tutte
le persone
che mi vogliono bene.

"Se hai tutto sotto controllo, significa che non stai andando abbastanza veloce."

Mario Andretti
ex pilota automobilistico e campione di Formula 1 nel 1978

Sommario

La pandemia da COVID-19 ha portato il mondo scientifico a cercare i metodi migliori per contrastare il veloce diffondersi della malattia. In ambito medico da anni sono in uso tecniche per diagnosticare patologie esaminando i suoni emanati dal corpo come la voce ed il respiro; altre invece sono basate sul riconoscimento di immagini. In questo studio sono state utilizzate le Deep Convolutional Neural Networks per riconoscere pazienti affetti da COVID-19 utilizzando gli spettrogrammi generati dalle registrazioni di colpi di tosse e respiri, raccolti in modalità crowdsourcing attraverso applicazioni mobili e web. I risultati sono promettenti e riescono a pareggiare lo stato dell'arte, certificando che le tecnologie di deep learning per la classificazione di immagini sono un ottimo strumento di supporto alla diagnosi di COVID-19.

Indice

Elenco delle figure

[1] Ayman Mahmoud. *Introduction to Shallow Machine Learning — LinkedIn*. Set. 2019. URL: https://www.linkedin.com/pulse/introduction-shallow-machine-learning-ayman-mahmoud/.

[2] Ayman Mahmoud. *Introduction to Shallow Machine Learning — LinkedIn*. Set. 2019. URL: https://www.linkedin.com/pulse/introduction-shallow-machine-learning-ayman-mahmoud/.

Elenco delle tabelle

[3]Valerio Velardo. *Audio Signal Processing for Machine Learning*. Consultato il 1/02/2021. 2020. URL: `https://www.youtube.com/watch?v=iCwMQJnKk2c&list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0`.

Capitolo 1

Introduzione

La pandemia globale generata dal coronavirus 2 da sindrome respiratoria acuta grave (SARS-CoV-2), che ha come conseguenza la malattia respiratoria acuta da SARS-CoV-2 (COVID-19) ha sconvolto le abitudini della popolazione mondiale e costretto i governi nazionali ad eseguire misure straordinarie nel tentativo di combattere la diffusione del virus in attesa dello sviluppo di vaccini efficaci. Dall'inizio del 2020, ovvero dallo scoppio della pandemia, ad oggi la ricerca scientifica ha cercato metodi per il tracciamento delle persone infettate dal virus applicando tecniche come il *contact tracing* e gli *screening* di massa. La velocità di diffusione di questo virus ha reso però difficile effettuare queste azioni in modo efficace.

I metodi di *deep learning* messi a disposizione dall'intelligenza artificiale e già utilizzati in ambito medico per l'identificazione di svariate patologie possono rivelarsi utili anche per individuare il nuovo COVID-19. Come descritto nel capitolo 3, molti studi hanno cercato di dare una risposta a questa domanda, trovando risvolti positivi. In una di queste ricerche, illustrata nel capitolo 4, alcuni ricercatori dell'Università di Cambridge hanno messo in atto una campagna di *crowdsourcing* utilizzando applicazioni mobili e un form sul loro sito web, al fine di creare un database su cui addestrare dei modelli di intelligenza artificiale a riconoscere casi di COVID-19. Grazie ad un accordo tra l'Università di Cambridge e l'Università di Bologna è stato possibile accedere a questo dataset e provare un metodo alternativo. Più dettagliatamente, il problema su cui si è concentrato questo lavoro di tesi è l'utilizzo delle Convolutional Neural Networks come alternativa ai metodi utilizzati dai ricercatori dell'Università di Cambridge. Utilizzare questo tipo di modelli che si basano sul *deep learning* porta il vantaggio di poter rimuovere l'annosa parte di feature extraction che deve essere effettuata quando si utilizzano approcci di *shallow learning* e che richiede una notevole esperienza nel campo dell'audio processing.

L'idea nasce dalla possibilità di "convertire" i segnali audio in immagini tramite la generazione di spettrogrammi: in questo modo si sposta il problema dal campo dell'audio processing al campo dell'image classification. Esistono tutta una serie di reti neurali addestrate a riconoscere immagini di oggetti di uso comune e che quindi

hanno dei filtri, chiamati "convoluzioni", in grado di estrarre proprio dalle immagini le informazioni, chiamate "feature", che sono utili alla soluzione del problema.

L'interrogativo alla base di questa tesi, nonché quello che ha motivato gli esperimenti presentati in questo testo, è il seguente: possono essere le CNN una valida alternativa ai classificatori Linear Regression e SVM? E se sì, quali risultati sono in grado di ottenere? In questa tesi sono illustrati gli esperimenti condotti per rispondere a queste domande.

Il capitolo successivo (cap. 2) è dedicato alla spiegazione di alcuni concetti importanti per comprendere meglio quanto fatto, in particolare per quanto concerne l'audio processing. Dopo il capitolo 3 dedicato allo stato dell'arte, viene illustrato più nel dettaglio quanto fatto all'Università di Cambridge, nel capitolo 4. I capitoli dedicati ai metodi sono due: il primo (cap. 5) è dedicato all'implementazione e replica dell'esperimento originale, fase necessaria per comprendere meglio come affrontare l'approccio basato su CNN; il secondo (cap. 6), invece, è dedicato alla sperimentazione della nuova metodologia. I risultati finali sono illustrati nel capitolo 7 e confrontati con quelli pubblicati dai ricercatori dell'università inglese. Segue una discussione generale su quanto sperimentato (cap. 8). Nelle appendici si trova del materiale aggiuntivo: dati e grafici prodotti dagli esperimenti effettuati.

Capitolo 2

Nozioni preliminari

In questo capitolo sono descritti i concetti preliminari utili a comprendere meglio gli esperimenti descritti in questo testo, in particolare da dove nascono le tecnologie che sono state utilizzate.

2.1 Intelligenza artificiale e machine learning

L'**intelligenza artificiale** è una disciplina dell'informatica che si occupa di studiare sistemi software che siano in grado di riprodurre alcune carattertiche tipiche degli esseri umani. Ci sono molti problemi complessi che non permettono di applicare l'approccio della programmazione classica, dove si esaminano tutte le possibili soluzioni. L'intelligenza artificiale cerca di creare programmi che siano in grado di esaminare autonomamente lo spazio delle soluzioni, effettuando decisioni sulla base delle informazioni che hanno a disposizione e andando alla ricerca di una soluzione, quando non si conosce bene dove questa possa essere (o di che tipo possa essere).

L'intelligenza artificiale è nata ufficialmente negli anni '50 con la pubblicazione di Turing (1950) [67] e la conferenza di Dartumund (1956) [42], per conoscere poi decenni di oblio a causa delle potenze di elaborazione che ancora non erano adatte a questo tipo di problemi. Negli ultimi decenni l'intelligenza artificiale è tornata alla ribalta e sta rivoluzionando il mondo, in tutti i settori.

Il **machine learining** [58] è quella branca dell'intelligenza artificiale che si occupa di creare sistemi che siano in grado di imparare dalla loro esperienza tramite un sistema di *azione-valutazione-correzione* che si ripete in modo iterativo. Questo approccio si contrappone fortemente a quello di "programmazione classica" (figura 2.1).

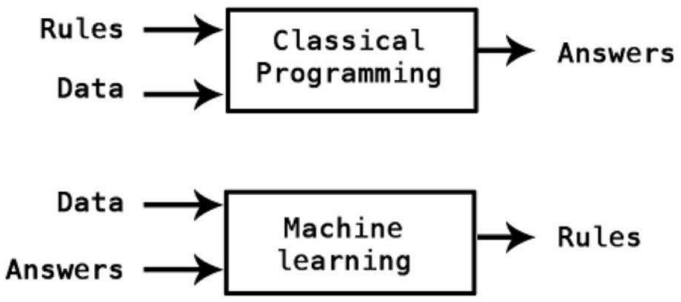


Figura 2.1: Machine learning vs. Programmazione classica.

2.2 Le reti neurali

Le reti neurali sono modelli che vogliono riprodurre la struttura del cervello degli esseri umani. Sono essenzialmente grafi, dove i nodi rappresentato i neuroni e le connessioni tra i nodi rappresentano le sinapsi. L'idea di base da cui sono state create è quella del neurone artificiale [18] e del percettrone [55]. Utilizzano la tecnica di discesa del gradiente e l'algoritmo di backpropagation.

2.3 Shallow learning e deep learning

In questo ambito viene fatta una distinzione tra "apprendimento piatto" (*shallow learning*) e "apprendimento profondo" (*deep learning*). La differenza tra i due approcci risiede nella parte di estrazione delle feature:

- nello **shallow learning** la suddetta parte deve essere effettuata manualmente e quindi richiede una conoscenza del dominio dei dati su cui si vuole sviluppare il processo di appendimento[1];
- nel **deep learning** si delega al modello stesso l'estrazione delle feature e rappresenta un'ottima soluzione nel caso non si possieda una buona conoscenza di quali sono le feature da estrarre.

È statisticamente confermato che i modelli di *shallow learning* ottengono risultati migliori rispetto ai modelli di *deep learning* su dataset limitati se non si utilizzano tecniche di regolarizzazione o di *data augmentation* [50, 60, 69].

2.4 Overfitting

Il principale problema che si incontra nella costruzione di modelli di *machine learning* ha un nome e si chiama "overfitting". Quando un modello viene addestrato

[1] Ayman Mahmoud. *Introduction to Shallow Machine Learning — LinkedIn*. Set. 2019. URL: https://www.linkedin.com/pulse/introduction-shallow-machine-learning-ayman-mahmoud/.

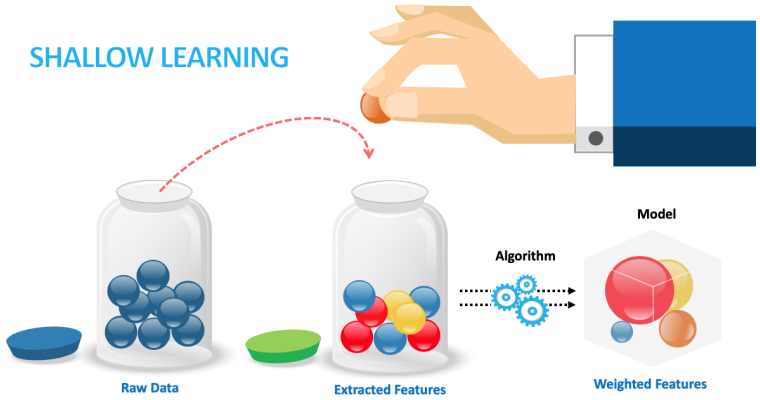


Figura 2.2: Schema di un modello di apprendimento piatto (*shallow learning*)[2].

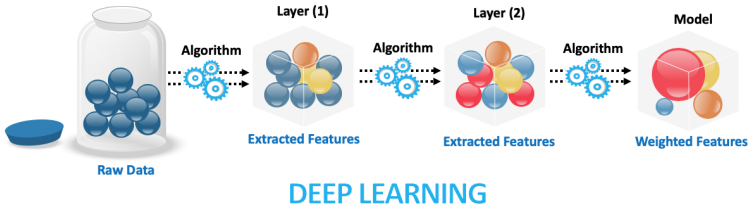


Figura 2.3: Schema di un modello di apprendimento profondo (*deep learning*)[3].

su un insieme di dati chiamato *training set* potrebbe, da un certo punto in poi della fase di addestramento, cercare di "imparare a memoria" i dati che vede piuttosto che utilizzarli per capire quali proprietà debba estrarre. Se questo succede, il modello diventerà troppo specializzato e una valutazione sul *test set* darà risultati scarsi perché il modello non sarà in grado di generalizzare adeguatamente. Dall'altra parte si ha l'*underfitting*, dove il modello non ha imparato abbastanza dal training set. La fase di addestramento deve essere quindi studiata in modo tale da non creare un modello né poco informato (underfitted), né troppo specializzato (overfitted).

2.5 Convolutional Neural Networks

Le Convolutional Neural Networks (o reti neurali convoluzionali) sono un tipo di rete neurale che fa uso di livelli chiamati "convoluzioni" che elaborano un'immagine scomponendola in tante piccole parti e applicando dei filtri al fine di estrarre informazioni che possono essere utili a calcolare l'output finale (*feature* [36]). Rientrano nell'approccio del *deep learning* in quanto la parte di estrazione delle feature è del tutto automatica e si lascia alla rete decidere quali feature sono più importanti e quali meno.

Sono state introdotte con il modello "Inception" nel 2014 in occasione della *ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC 2014)*, definendo di fatto un nuovo standard nel riconoscimento automatico di immagini [63, 37].

Applicazioni di questi modelli si hanno nei problemi di *image classification* [38], *object detection* e *pattern recognition* [20, 45].

Purtroppo questi modelli richiedono un'enorme potenza di elaborazione parallela che spesso è difficile avere a disposizione [72]. Questo è un problema tipico di tutte le reti neurali, ma nelle CNN è accentuato perché le immagini sono input molto grandi.

2.6 Valutazione di modelli predittivi

Ci sono numerose metriche per valutare l'efficacia di modelli di apprendimento automatico e sostanzialmente si basano tutte sul comportamento che il modello ha su dati diversi da quelli sui quali è stato addestrato, poiché all'atto pratico è effettivamente ciò che deve essere in grado di fare, ovvero effettuare buone previsioni su dati che non ha mai visto prima. Questa operazione si chiama "valutazione" e deve fornire una misura della capacità del modello di predire in maniera corretta, una volta che sarà addestrato sull'intero dataset.

2.6.1 Cross validation

Il metodo più diffuso per la valutazione di un modello predittivo si chiama "convalida incrociata" (o *cross validation* in inglese) [9, 62] e l'idea di base è quella di ripetere le fasi di training e di testing con lo stesso modello ma su dati diversi.
In pratica, il dataset viene partizionato e ogni volta vengono scelte partizioni diverse per training set, validation set e test set. Scelto un numero di partizioni k che si vogliono creare (solitamente 5 o 10), si può fare in due modi:

- **Convalida incrociata semplice** (*cross validation*): Si effettuano k iterazioni: all'iterazione i la partizione p_i viene designata come test set e le altre $k-1$ partizioni come training set.
- **Convalida incrociata annidata** (*nested cross validation*) Si effettuano k^2 iterazioni: all'iterazione i la partizione p_i viene designata come test set e le altre $k-1$ partizioni vengono riunite in una sola che a sua volta viene divisa in altre k partizioni, dove una di queste sarà designata come validation set e le altre $k-1$ come training set.

Il primo di questi due concetti è schematizzato in figura 2.4, il secondo in figura 2.5, supponendo $k = 5$.

La convalida incrociata annidata è lo strumento utilizzato per valutare modelli come Logistic Regression o SVM, poiché presentano una varietà di iperparametri non molto elevata. Utilizzare lo stesso approccio con le reti neurali è un problema in

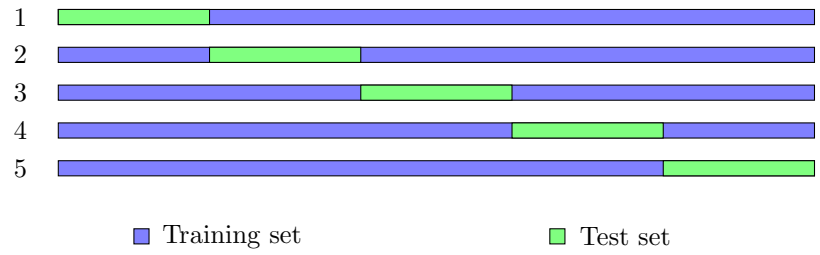


Figura 2.4: Schema di convalida incrociata semplice.

quanto, sebbene si possano selezionare un sottoinsieme di iperparametri (ma quale sarebbe il sottoinsieme da scegliere?), l'uso di un validation set va a sacrificare una parte del training set, aggravando il problema di overfitting quando si hanno pochi dati.

2.6.2 Metriche

Nei problemi di classificazione si hanno quattro indicatori:

- veri positivi (TP): l'elemento è stato correttamente classificato come "vero";
- veri negativi (TN): l'elemento è stato correttamente classificato come "falso";
- falsi positivi (FP): l'elemento è stato erroneamente classificato come "vero";
- falsi negativi (FN): l'elemento è stato erroneamente classificato come "falso".

Questi indicatori vengono combinati per calcolare delle metriche indicative delle prestazioni del modello.

Accuracy

L'accuracy corrisponde al numero di elementi classificati correttamente rispetto al numero totale di elementi presenti.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

La precision corrisponde al numero di elementi che sono stati classificati correttamente come positivi rispetto al numero totale di elementi classificati come positivi.

$$precision = \frac{TP}{TP + FP}$$

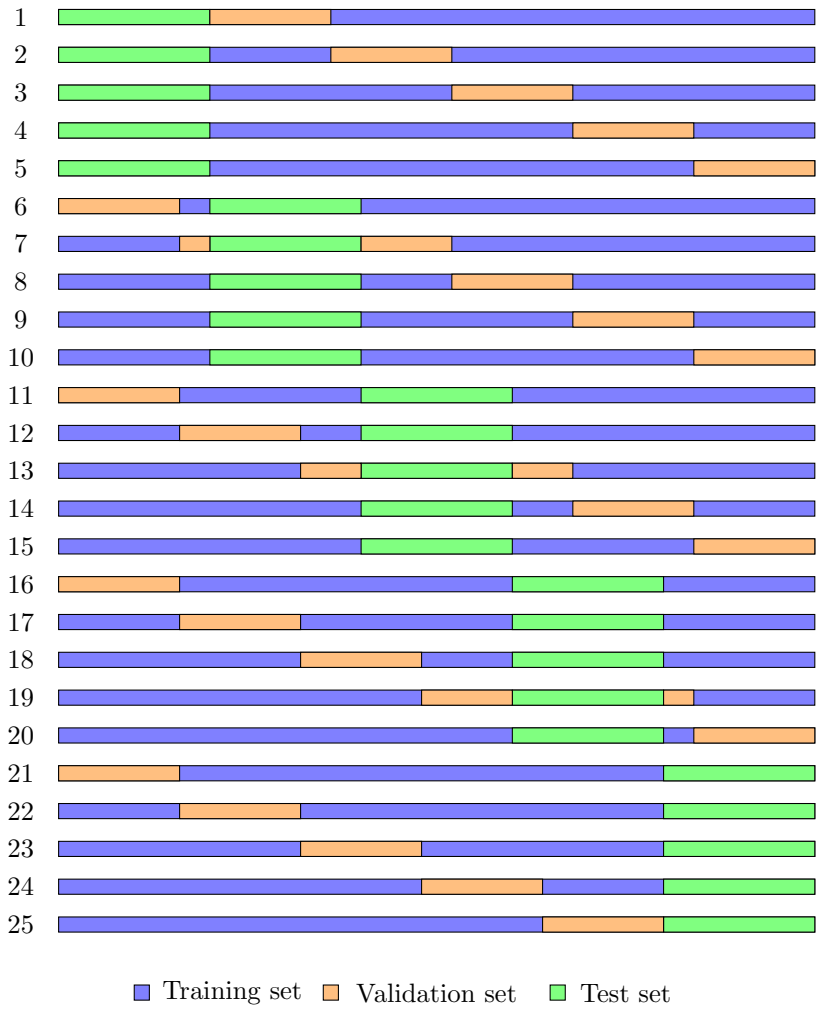


Figura 2.5: Schema di convalida incrociata annidata.

Recall/Sensitivity

La recall (anche chiamata sensitivity) corrisponde al numero di elementi che sono stati classificati correttamente come positivi rispetto al numero totale di elementi positivi presenti.

$$recall/sensitivity = \frac{TP}{TP + FN}$$

Specificity

La specificity corrisponde al numero di elementi che sono stati classificati correttamente come negativi rispetto al numero totale di elementi negativi presenti.

$$specificity = \frac{TN}{TN + FP}$$

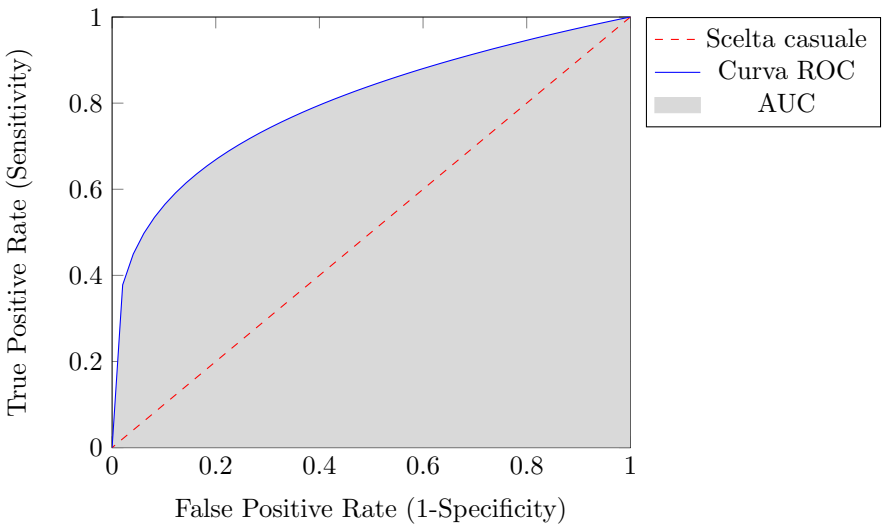


Figura 2.6: Struttura di un grafico ROC-AUC.

Sensitivity e specificity sono molto importati nei modelli sviluppati per la diagnosi di patologie, poiché rispondono alle seguenti domande:

- Sensitivity: quante persone realmente malate sono state identificate come tali?
- Specificity: quante persone sane sono state correttamente identificate come tali?

ROC-AUC

I classificatori binari, per esempio quelli basati su logistic regression, restituiscono un output un valore reale nell'intervallo $[0, 1]$. Questo valore di output grezzo deve servire a generare il valore di output booleano finale, ovvero quello che interessa all'utente. La soluzione utilizzata per convertire il l'output reale in intero è la definizione di una soglia nell'intervallo $[0, 1]$: se il valore è minore della soglia, il risultato sarà 0, mentre se è maggiore (o uguale) alla soglia il risultato sarà 1.

Viene subito da pensare che impostare la soglia a 0.5 sia la soluzione migliore, ma non è così, in modo particolare per i modelli che devono supportare la diagnosi di patologie. Intuitivamente, prima di diagnosticare una malattia è bene avere una certezza maggiore e un valore di 0.6, che darebbe esito positivo, potrebbe non essere abbastanza valido. In questi casi si sceglie di variare il valore della soglia, p.e. a 0.3 o 0.7, per avere un grado di confidenza più elevato.

Come capire, però, qual è il valore giusto per la soglia? Come valutare la bontà del modello indipendentemente da questa scelta? Con i grafici ROC-AUC.

Le curve Receiver Operating Characteristic (ROC) indicano il rapporto che c'è tra True Positive Rate (TPR) e False Positive Rate (FPR) al variare del valore

scelto per la soglia. Il grafico in figura 2.6 mostra la composizione di questo tipo di grafici: una bisettrice indica la situazione che si avrebbe con un classificatore che effettua scelte casuali, dove il tasso di veri positivi è sempre uguale al tasso di falsi positivi. Un classificatore così però sarebbe totalmente inutile. Un buon classificatore, invece, per ogni soglia scelta deve avere un tasso di veri positivi più alto di quello dei falsi positivi; più questo rapporto è alto, più il modello è buono. La metrica appena descritta, ovvero il rapporto tra tasso di veri positivi e tasso di falsi positivi, si indica con il termine Area Under Curve (AUC), il cui significato è quello letterale indicato dal nome. Infatti, più il tasso di veri positivi è maggiore del tasso dei falsi positivi, più la curva "fletterà" verso l'alto e più ampia sarà l'area sotto di essa.

Passiamo adesso ai concetti legati alle registrazioni audio.

2.7 Onde sonore

Le onde sonore sono onde meccaniche che, a differenza delle onde elettromagnetiche che possono propagarsi anche nel vuoto, hanno bisogno di un mezzo solido, liquido o gassoso. Le onde sonore sono generate da una vibrazione degli atomi e delle molecole che compongono il mezzo nel quale l'onda si propaga, che si comprimono e si diradano. Tale vibrazione è creata da l'applicazione di una determinata quantità di energia in uno specifico punto dello spazio. La propagazione nel mezzo è generata dalla deformazione dello stesso e grazie a questo processo l'energia può muoversi nello spazio. Nella figura 2.7 è mostrato graficamente questo concetto. Più le molecole si spostano, più grande sarà l'ampiezza dell'onda e più forte sarà il suono.

Più veloce le molecole vibrano, più alta sarà la frequenza del suono[4].

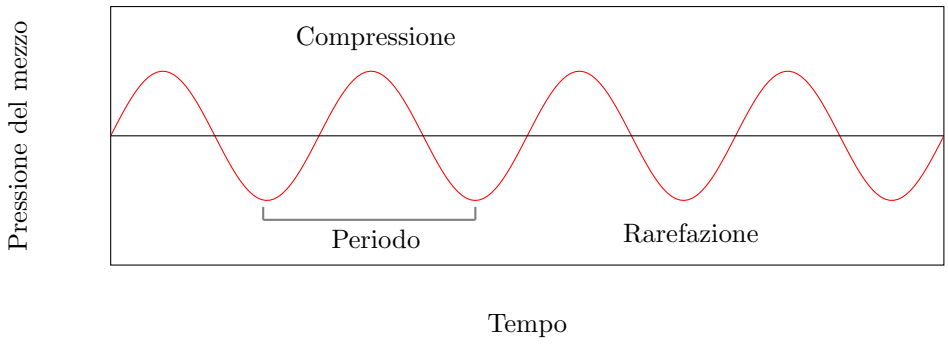


Figura 2.7: Rappresentazione grafica di un'onda sonora.

[4] *Digital audio concepts*. Consultato il 5/02/2021. URL: `https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts`.

2.8 Registrazioni audio

Le registrazioni audio contengono un sorprendente numero di informazioni e, in campo medico, possono essere utilizzate per scoprire patologie nelle persone. Questo è possibile perché i suoni generati dal nostro corpo possono rappresentare biomarcatori per la presenza di malattie come il Parkinson [6, 57] o lo stress post-traumatico [4].

Nei prossimi paragrafi sono mostrate, con l'ausilio di figure, le principali informazioni (o meglio, feature) che possono essere estratte dai file audio grezzi e utilizzate per addestrare modelli di apprendimento automatico. *Librosa* [43] è una libreria open-source per l'elaborazione di file audio in Python.

Frame e hop I *frame* sono i blocchi in cui è diviso il segnale audio digitale, mentre è definita con il nome *hop* la distanza tra i centri di due frame consecutivi.

Forma d'onda La forma d'onda è una rappresentazione grafica che misura l'ampiezza del segnale nel tempo ed è la rappresentazione grafica più semplice di un'onda sonora. Esistono diversi tipi di forma d'onda.

- Onde periodiche: sono onde la cui struttura si ripete nel tempo in modo identico. Possono essere
  - semplici: costituite da una sola onda sinusoidale;
  - complesse: costituite da più onde onde sinusoidali;
- Onde aperiodiche: sono onde la cui struttura non presenta regolarità nel tempo. Possono essere
  - continue: si protraggono nel tempo in modo indefinito (es. rumori continui di sottofondo);
  - transienti: mostrano visibilmente un inizio e una fine (es. suono di una campanello).

In figura 2.8 è mostrato un grafico generato con *librosa* (da notare la somiglianza con la figura 2.7).

Periodo, frequenza e ampiezza d'onda Il periodo è il tempo che intercorre tra due picchi sull'asse dell'ampiezza dell'onda, mentre la frequenza è l'inversa del periodo T, ovvero

$$f = \frac{1}{T}$$

ed è espressa in *Hertz (Hz)*.
L'ampiezza d'onda dà una misura della forza della perturbazione nella pressione dell'aria e si misura in *decibel (dB)*. Graficamente, corrisponde a quanto si discosta

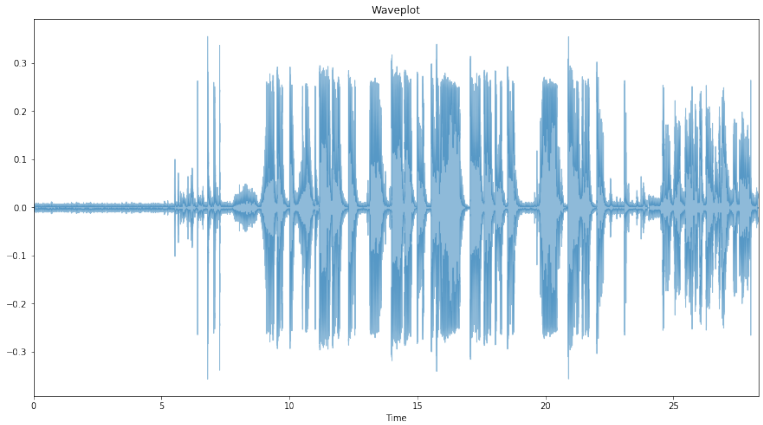


Figura 2.8: Waveplot di una registrazione audio di una persona con pertosse che tossisce.

la traccia dell'onda dalle ascisse. La frequenza definisce l'altezza del suono: un'alta frequenza corrisponde ad un suono alto, mentre l'ampiezza elevata indica quanto un suono intenso.

Ogni specie vivente che possiede un apparato uditivo riesce a percepire suoni che cadono in uno specifico intervallo di frequenza; nella tabella 2.1 sono elencati gli intervalli di frequenza per alcune specie di esseri viventi. L'essere umano è in grado di percepire frequenze non più alte di 20.000 Hz; per questo motivo, le frequenze superiori a questa soglia prendono il nome di "ultrasuoni".

Specie	**Intervallo percepito**
Elefanti	14-12.000 Hz
Esseri umani	20-20.000 Hz
Gatti	48-75.000 Hz
Cani	64-45.000 Hz
Topi	1.000-70.000 Hz
Delfini	75-150.000 Hz
Pipistrelli	7.000-200.000 Hz

Tabella 2.1: Intervallo di frequenze percepito da alcune specie di esseri viventi[5].

Per quanto riguarda la percezione dei suoni da parte nostra, è interessante tenere conto del concetto di *altezza* di un suono (in inglese *pitch*). La nostra percezione delle frequenze di un suono avviene in scala logaritmica, ovvero che noi percepiamo il raddoppio di una frequenza non come tale, ma come se fosse uno step in più rispetto alla frequenza precedente. Possiamo quindi definire l'altezza p di un suono come il logaritmo della sua frequenza f. In particolare, seguendo lo

standard Musical Instrument Digital Interface (MIDI)[6], la formula è:

$$p = 69 + 12 \times \log_2(\frac{f}{440})$$

Intensità e potenza La *potenza* di un suono corrisponde all'ammontare di energia che viene emessa in una singola unità di tempo ed è misurato in watt (W). A questo concetto è legato quello di *intensità*, che misura la potenza in rapporta all'area e viene misurato in watt per metro quadrato (W/m^2). Gli esseri umani possono percepire suono di intensità che vanno da $10^{-12}W/m^2$ (limite minimo) a $10^{-12}W/m^2$ (soglia del dolore).

Spettrogramma Lo spettrogramma (figura 2.9) è una sorta di impronta del segnale audio e descrive l'intensità di un suono in funzione della tempo (ascisse) e della frequenza (ordinate). La differenza di colore nelle varie zone del grafico indica una diversa intensità del suono. Solitamente le frequenze sono rappresentate in scala logaritmica.

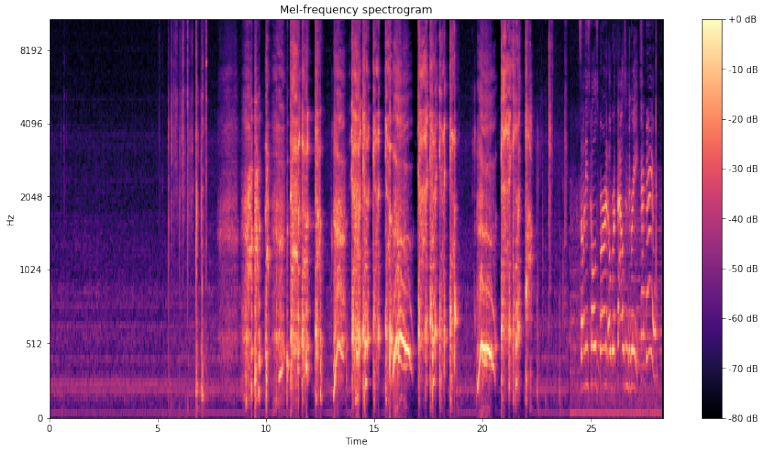


Figura 2.9: Mel-frequency spectrogram di una registrazione audio di una persona con pertosse che tossisce.

[6] *Altezza (acustica) - Wikipedia.* URL: `https://it.wikipedia.org/wiki/Altezza_(acustica)`.

Capitolo 3

Stato dell'arte

La ricerca scientifica si è subito attivata nello studio di questo nuovo virus sconosciuto. Il 2020 è stato un anno ricco di pubblicazioni da parte di ricercatori di tutto il mondo: sul sito del World Health Organization (WHO)[1] si può accedere ad un database di 214.523 articoli che trattano della COVID-19 (alla data del 26 febbraio 2021).

In questo capitolo sono elencate e descritte brevemente le pubblicazioni scientifiche inerenti all'argomento trattato da questo testo, ovvero l'utilizzo di tecniche di intelligenza artificiale e machine learning per la diagnosi di casi da COVID-19.

Il punto di riferimento per la ricerca presentata in questa tesi è il lavoro fatto da alcuni ricercatori dell'Università di Cambridge, i quali hanno creato un modello di classificazione di tipo *shallow learning* per provare a diagnosticare COVID-19 utilizzando per l'apprendimento un dataset creato in *crowdsourcing* [8]. Maggiori dettagli sono presentati nel capitolo 4.

Il suddetto lavoro prende spunto da uno precedente. Pramono et al. [52] hanno proposto un algoritmo per la diagnosi di pertosse che analizza le registrazioni audio di colpi di tosse e "whoop". Gli "whoop" sono insipirazioni simili ad un fischio che avvengono tra un colpo di tosse e l'altro, tipiche della pertosse. L'articolo in questione descrive un algoritmo molto semplice, adatto a dispositivi con moderata potenza computazionale, e si sofferma molto sulla descrizione della parte di feature extraction, indicando quali sono state le procedure per l'estrazione delle informazioni importanti dai file grezzi; informazioni che poi sono state date in input al classificatore. I risultati registrati sono un accuracy del 92% e una precision del 97%.

Imran et al. [33] hanno anticipato i ricercatori di Cambridge effettuando un esperimento dove ci si poneva il problema di utilizzare uno strumento di screening

[1] *Global literature on coronavirus disease*. Consultato il 26/02/2021. WHO. 2020. URL: `https://search.bvsalud.org/global-literature-on-novel-coronavirus-2019-ncov/`.

scalabile sfruttando la tecnologia dei dispositivi mobili. Il dataset utilizzato per allenare i loro modelli era tuttavia molto ridotto, infatti gli autori hanno voluto precisare che lo scopo del loro esperimento è stato quello di studiare la fattibilità dell'uso di tecniche di intelligenza artificiale per distinguere casi di COVID-19. I loro risultati sono stati promettenti.

Bagad et al. [3] hanno utilizzato l'intelligenza artificiale per discriminare tra casi COVID-19 positivi e negativi. Dai file audio sono stati generati degli spettrogrammi che sono stati analizzati da un modello di CNN per predire la presenza dei sintomi della COVID-19 e conseguentemente decidere se il paziente debba essere sottoposto al test RT-PCR.

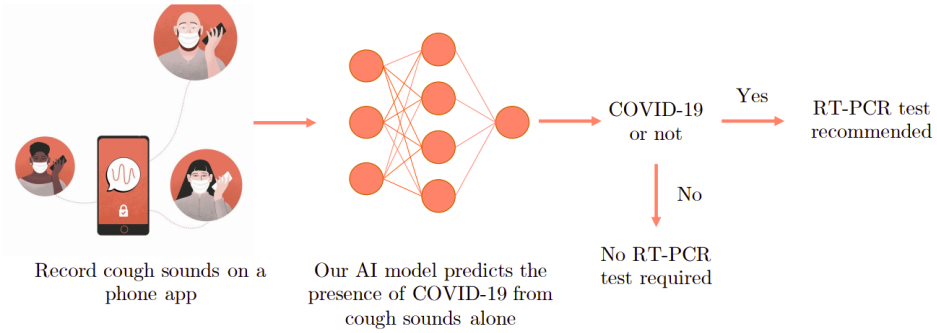


Figura 3.1: Modello di pre-screening basato su intelligenza artificiale [3].

Altri studi [5, 70] hanno utilizzato Dense Convolutional Neural Networks e transfer learning per individuare COVID-19 basandosi sulle radiografie al petto. In particolare, le reti utilizzate sono DenseNet121 e DenseNet201 [31].

Un recente studio [17] ha dimostrato che si possono utilizzare CNN basate su spettrogrammi audio per discriminare tra audio di persone con COVID-19 e audio di persone sane con un'accuracy molto elevata (in questo caso è stato ottenuto un valore di 97,5%). Anche altre ricerche hanno utilizzato questo tipo di approccio per elaborare sorgenti audio [16].

Le CNN sono state utilizzate anche per cercare di prevedere il numero dei casi di persone affette da COVID-19 in Cina [30].

3.1 Convolutional Neural Networks

Sono disponibili gratuitamente, in particolare nella libreria *Keras* [35], tutta una serie di CNN che sono state addestrate e valutate su *ImageNet*. Il grande vantaggio della disponibilità di queste reti è la possibilità di scaricare il file dei pesi creato in fase di addestramento, in modo tale da poter effettuare *transfer learning* e risparmiare molto tempo quando le si addestrano sui nostri dati. Sono disponibili anche le valutazioni ottenute su *ImageNet*, insieme ad altri dati tecnici legati alle

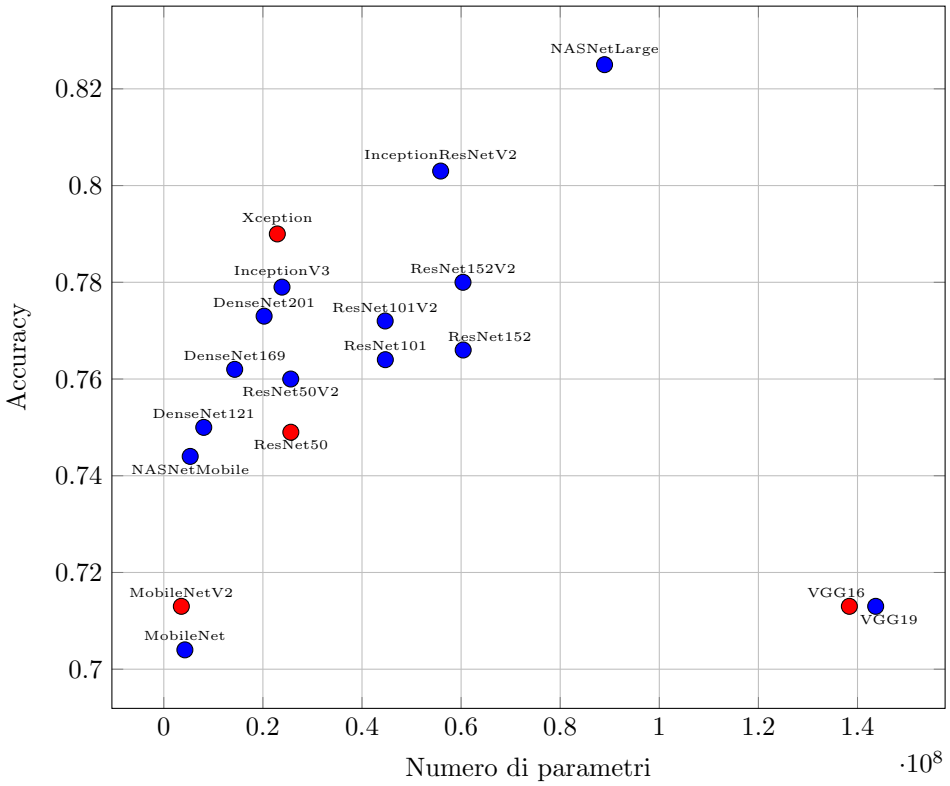


Figura 3.2: Modelli di reti neurali disponibili in *Keras*. Confronto numero di parametri - accuracy.

architetture delle reti (es. numero di parametri e grandezza). In figura 3.2 è mostrato uno scatter plot che mostra il rapporto tra numero di parametri ed accuracy per queste reti, mentre in figura 3.3 lo stesso tipo di grafico mostra il rapporto tra grandezza del file e accuracy.

Nelle prossime sezioni sono presentati i quattro modelli di rete neurale convoluzionale utilizzati: MobileNetV2, ResNet50, VGG16 ed Xception.

3.1.1 Apprendimento residuale

Le architetture di CNN più recenti si basando su un concetto di apprendimento rivisitato, chiamato apprendimento residuale [25, 73]. L'idea che ne sta alla base è quella di aggiungere alla funzione generata dai livelli di apprendimento ($F(x)$) l'input (x), in modo tale da propagare l'informazione sull'input.
Supponiamo di avere due input x e y tali che $F(x) = F(y)$. Procedendo con l'apprendimento classico l'output della funzione F sarebbe lo stesso per due input diversi e ci sarebbe perdita di informazione. Con l'apprendimento residuale, anche se $F(x) = F(y)$ avremo $F(x) + x \neq F(y) + y$. In figura 3.4, tratta da [25], è schematizzato questo concetto.

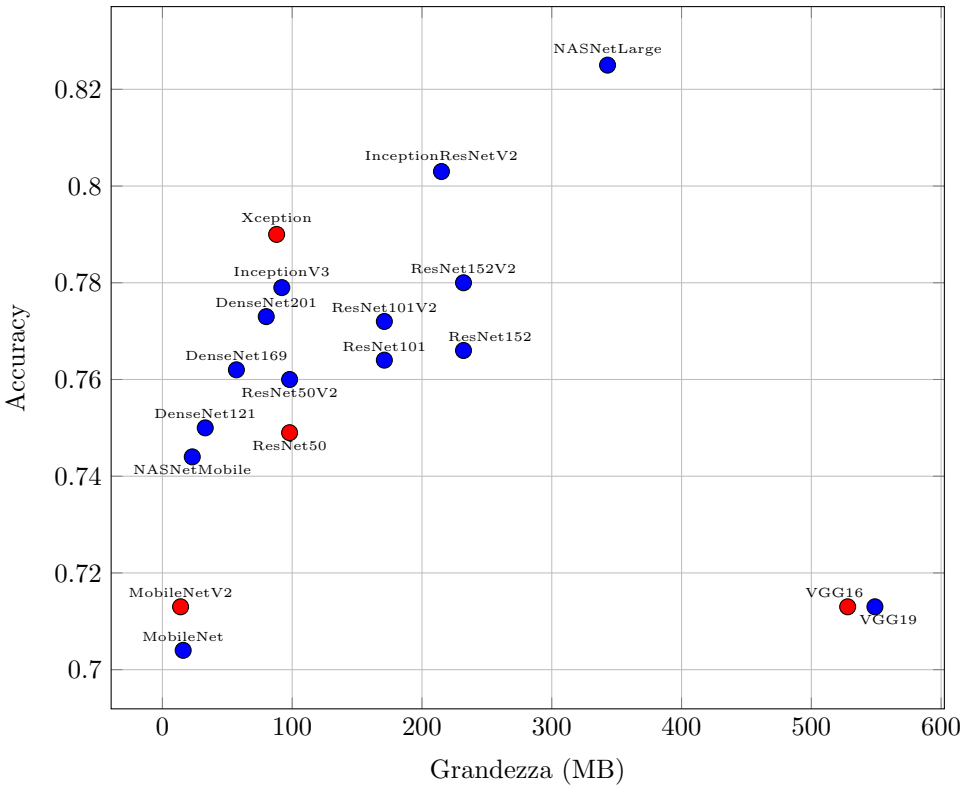


Figura 3.3: Modelli di reti neurali disponibili in *Keras*. Confronto grandezza - accuracy.

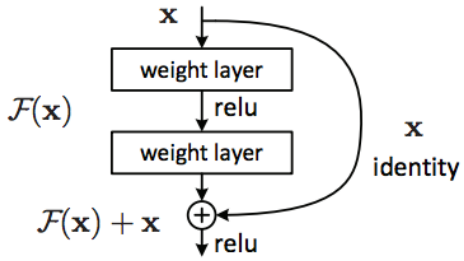


Figura 3.4: Schema del funzionamento dell'apprendimento residuale [25].

3.1.2 MobileNetV2

MobileNetV2 [59] si presenta come un'evoluzione della prima versione [29]. La progettazione di questo tipo di modelli ha come priorità l'occupazione dello spazio in memoria e conseguentemente l'adattabilità a girare su dispositivi con modesta capacità computazionale, come gli smartphone. MobileNetV2 ha infatti un peso di soli 14 MB che la rende la CNN più leggera attualmente presente in letteratura; lo stesso discorso vale per il numero di parametri: 3.538.984[2]. Le performance di MobileNetV2 su ImageNet sono 71% per la top-1 accuracy e 90% per la top-5 accuracy.

3.1.3 ResNet50

ResNet50 è stata introdotta nel 2015, fa parte della famiglia ResNet [26] ed è la CNN con la quale è stato introdotto il concetto di *residual learning*. ResNet50 è molto utilizzata in letteratura e le sue prestazioni sono tutt'ora notevoli, soprattutto se paragonate allo spazio di memoria che occupa (88 MB) e il numero di parametri (25.636.712)[3]. Il benchmark effettuato su ImageNet rivela un 75% in top-1 accuracy e un 92% in top-5 accuracy.

3.1.4 VGG16

VGG16 [61] è la più potente CNN basata sul concetto secondo il quale basta aggiungere livelli di convoluzione per aumentare le prestazioni della rete. Questo approccio, a cui il *residual learning* si propone come alternativa, presenta il problema di richiedere un'enorme quantità di tempo per la fase di addestramento della rete. Il peso in memoria di VGG16 è infatti di 528 MB e il numero di parametri è 138.357.544[4], numeri tremendamente alti che difficilmente vengono giustificati con una top-1 accuracy del 71% e una top-5 accuracy del 90% (i valori sono rilevati, come di consueto, su ImageNet).

[2] *Keras Applications*. URL: `https://keras.io/api/applications/`.
[3] *Keras Applications*. URL: `https://keras.io/api/applications/`.
[4] *Keras Applications*. URL: `https://keras.io/api/applications/`.

3.1.5 Xception

Xception [10] è una variante di Inception [64] ed è stata utilizzata in uno studio per la rilevazione di COVID-19 tramite l'utilizzo di spettrogrammi audio [54]. Se consideriamo il rapporto tra risultati ottenuti e grandezza della rete, Xception è decisamente una delle migliori: in 88 MB di memoria si può salvare un modello che ottiene il 79% in top-1 accuracy e il 95% in top-5 accuracy su ImageNet.

Capitolo 4

Il progetto di *Brown et al.*

Brown et al. [8] è il team di ricercatori dell'Università di Cambridge che ha sviluppato un'applicazione mobile per la raccolta di registrazioni di colpi di tosse e respiri. Il loro obiettivo è creare un modello di classificazione che riesce a distinguere un paziente affetto da COVID-19 da uno sano utilizzando come input le registrazioni audio. In particolare sono stati studiati tre casi:

1. (*COVID-positive vs. non-COVID*) Distinzione tra utenti che hanno dichiarato di essere risultati positivi al COVID-19 e utenti che non hanno dichiarato di essere risultati positivi al COVID-19, hanno una storia medica pulita, non hanno mai fumato e non hanno sintomi;

2. (*COVID-positive with cough vs. non-COVID with cough*) Distinzione tra utenti che hanno dichiarato di essere risultati positivi al COVID-19 e hanno la tosse come sintomo e utenti che hanno dichiarato di non essere risultati positivi al COVID-19, hanno una storia medica pulita, non hanno mai fumato e hanno la tosse;

3. (*COVID-positive with cough vs. non-COVID with asthma cough*) Distinzione tra utenti che hanno dichiarato di essere risultati positivi al COVID-19 e hanno la tosse come sintomo e utenti che hanno dichiarato di non essere risultati positivi al COVID-19, hanno riportato l'asma nella loro storia medica e hanno la tosse.

Per far questo, l'Università di Cambridge ha promosso una campagna di crowdsourcing al fine di costruire un dataset che contenga campioni audio di persone affette da COVID-19 e campioni audio di persone sane da utilizzare per addestrare i modelli. Le informazioni per partecipare sono disponibili sul sito del progetto[1].

Per condurre gli esperimenti con i dati raccolti sono stati utilizzati classificatori di tipo logistic regression e Support Vector Machine (SVM). I risultati registrano

[1] Chloë Brown et al. *COVID-19 Sounds App*. Accessed: 2021-02-01. University of Cambridge. URL: `https://www.covid-19-sounds.org/`.

un valore di AUC dell'80% per i task 1 e 3 e dell'82% per il task 2 senza data augmentation sui file audio. Con data augmentation i risultati sul task 2 sono arrivati all'87% e sul task 3 all'88%. La struttura del modello utilizzato è mostrato in figura 4.1.

Nelle prossime sezioni sono descritti i passi effettuati dai ricercatori per generare i dati numerici, a partire dai file audio grezzi, da poter dare in input ai classificatori.

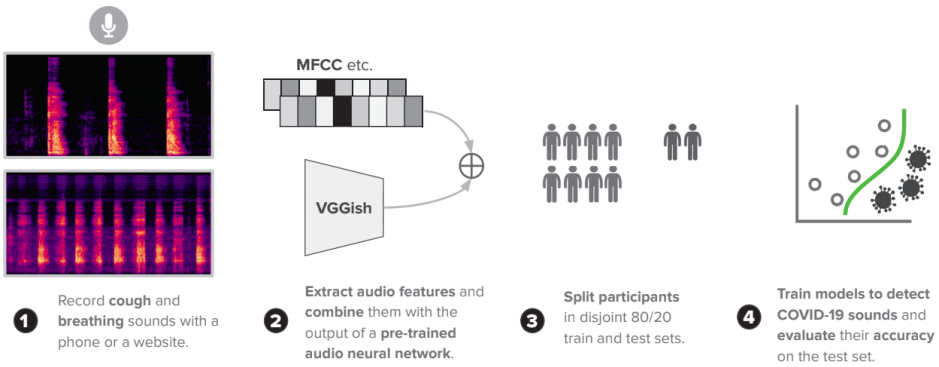


Figura 4.1: Pipeline del modello di machine learning utilizzato.

4.1 Estrazione delle feature

In questo esperimento sono state utilizzate due tipi di feature, quelle estratte manualmente e quelle estratte in modo automatico con *VGGish* [28].

Feature estratte manualmente I file audio sono stati processati con *librosa* [43] e ricampionati (*resampled*, in inglese) allo standard di 22 kHz. Per quanto riguarda le feature estratte, alcune sono a livello di segmento, per le quali viene generato un sono valore per tutto il file audio; mentre altre sono a livello di frame, per le quali viene generato un vettore di valori di dimensione uguale al numero di frame del file audio.

- Feature a livello di segmento: durata, numero di onset, tempo, periodo.

  Durata La misura in secondi della durata del segnale audio.

  Onset Indica l'inizio effettivo di un suono, che può essere non correlato con l'inizio della registrazione.

  Tempo Definisce il ritmo o la velocità di un suono.

- Feature a livello di frame: root-mean-square energy, spectral centroid, frequenza roll-off, rapporto zero-crossing, Mel-Frequency Cepstral Coefficient (MFCC), Δ-MFCC, Δ^2-MFCC.

  Periodo Vedere sezione 2.8.

  RMS Energy Valore che indica l'energia di ogni frame del segnale audio tramite la seguente formula.

  $$RMSE = \frac{1}{N}\sqrt{\sum_{n=0}^{N-1} x(n)^2}$$

  Spectral centroid Rappresenta l'equivalente del centro di massa in uno spettro.

  Roll-off Frequency Fequenza che limita superiormente da parte di segnale dove è contenuta la maggior parte di energia [52].

  Zero-crossing Rate Rappresenta le frequenze che corrispondono ai momenti in cui il segnale cambia di segno (da positivo a negativo).

  MFCC Mel-Frequency Cepstral Coefficients [13].

Per quanto riguarda le feature a livello di frame, sono stati estratti i seguenti indicatori statistici:

- media
- mediana
- root-mean-square
- massimo
- minimo
- 1° quartile
- 3° quartile
- scarto interquartile
- deviazione standard
- skewness
- kurtosis

Ricapitolando, avremo:

- 4 feature di segmento (durata, numero di onset, tempo, periodo), ognuna rappresentata da un singolo valore;
- 4 feature di livello (root-mean-square energy, spectral centroid, frequenza roll-off, rapporto zero-crossing), ognuna rappresentata da 11 indicatori statistici;
- 3 feature MFCC da cui sono stati estratti i primi 13 componenti. Ognuno dei 13 componenti è un vettore su cui sono stati calcolati gli 11 indicatori statistici.

Tutta questa operazione genera un vettore finale, per ogni file audio, di 477 elementi (equazione 4.1).

$$4 + 4 \times 11 + 3 \times 13 \times 11 = 447 \tag{4.1}$$

Feature estratte con *VGGish* *VGGish* è una rete neurale convoluzionale pre-addestrata, pubblicamente accessibile, in grado di effettuare la fase di estrazione delle feature basandosi sugli spettrogrammi. Questo modello è stato addestrato su un grande dataset generato da YouTube.

VGGish divide i campioni audio di 16 kHz in sotto-campioni di 0.96 secondi ciascuno e, per ognuno di questi, restituisce un vettore di 128 valori. Il risultato finale è quindi un vettore di dimensione pari al numero di sotto-campioni da 0.96 s, che contiene sotto-vettori di 128 valori. Una matrice, in sostanza.

In ultima istanza, per matrice generata automaticamente dal modello, sono stati calcolati le medie e le deviazioni standard per tutti i vettori di dimensione 128, generando un vettore di dimensione 256 (128 × 2). In figura 4.2 è schematizzato questo processo e nelle equazioni 4.2 e 4.3 ne è mostrata la definizione formale.

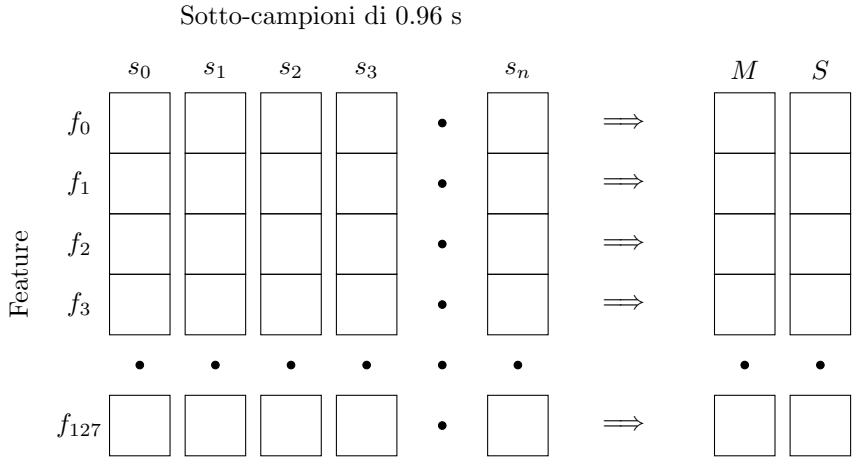


Figura 4.2: Estrazione delle feature con *VGGish*.

$$M_i = \frac{\sum_{j=0}^{n} f_{i,j}}{n} \qquad \text{per } i \in [0...127] \tag{4.2}$$

$$S_i = \sqrt{\frac{\sum_{j=0}^{n-1} (f_{i,j} - M_i)^2}{n}} \qquad \text{per } i \in [0...127] \tag{4.3}$$

Ci sono poi due modi per unire il vettore delle medie M e il vettore delle deviazioni standard S: la prima consiste nella banale concatenazione di M e S (per colonne, eq. 4.4); la seconda prevede l'alternanza dei singoli elementi di M e S (per righe, eq. 4.5).

$$V = M_0, M_1, \ldots, M_{127}, S_0, S_1, \ldots, S_{127} \tag{4.4}$$

$$V = M_0, S_0, M_1, S_1, \ldots, M_{127}, S_{127} \tag{4.5}$$

Ognuna di queste scelte è ugualmente valida e non influenzerà il comportamento del modello di classificazione, purché si effettui la stessa scelta per ogni file audio.

In figura 4.3 sono mostrati i dati estratti, più precisamente i valori di media. I boxplot, che in italiano si possono tradurre con l'espressione "diagrammi a scatola e baffi", sono delle rappresentazioni grafiche utilizzate per descrivere la distribuzione di un campione tramite semplici indici di dispersione e di posizione. Viene rappresentato (orientato orizzontalmente o verticalmente) tramite un rettangolo diviso in due parti, da cui escono due segmenti. Il rettangolo (la "scatola") è delimitato dal primo e dal terzo quartile [56], $q_{1/4}$ e $q_{3/4}$, e diviso al suo interno dalla mediana, $q_{1/2}$. I segmenti (i "baffi") sono delimitati dal minimo e dal massimo dei valori[2]. Si può subito notare marcate differenza in base alla tipologia di registrazione presa in considerazione.

4.2 Principal component analysis

La Principal component analysis (PCA) [19] è una tecnica dell'analisi statistica utilizzata per ridurre la dimensione di un vettore di un vettore di lunghezza arbitraria cercando di perdere meno informazione possibile.

4.3 Risultati

Ai task 1, 2, e 3 sono stati destinati sottoinsiemi del dataset diversi:

- il task n. 1 è stato sperimentato con registrazioni sia di respirazioni sia di colpi di tosse (cough+breath) con 141 esempi da 62 utenti per l'esito positivo (*COVID-positive*) e 298 esempi da 220 utenti per l'esito negativo (*non-COVID*);
- il task n. 2 è stato sperimentato con registrazioni di colpi di tosse (cough) con 54 esempi da 23 utenti per l'esito positivo (*COVID-positive with cough*) e 32 esempi da 29 utenti per l'esito negativo (*non-COVID with cough*);
- il task n. 3 è stato sperimentato con registrazioni di respirazioni (breath) con 54 esempi da 23 utenti per l'esito positivo (*COVID-positive with cough*) e 20 esempi da 18 utenti per l'esito negativo (*non-COVID with asthma cough*).

Per ognuno dei tre task sono stati effettuati tre diverse prove, cambiando la tipologia di feature, per un totale di nove esperimenti.

1. Feature estratte manualmente con PCA = 0.8.

[2] *Diagramma a scatola e baffi - Wikipedia.* URL: `https://it.wikipedia.org/wiki/Diagramma_a_scatola_e_baffi`.

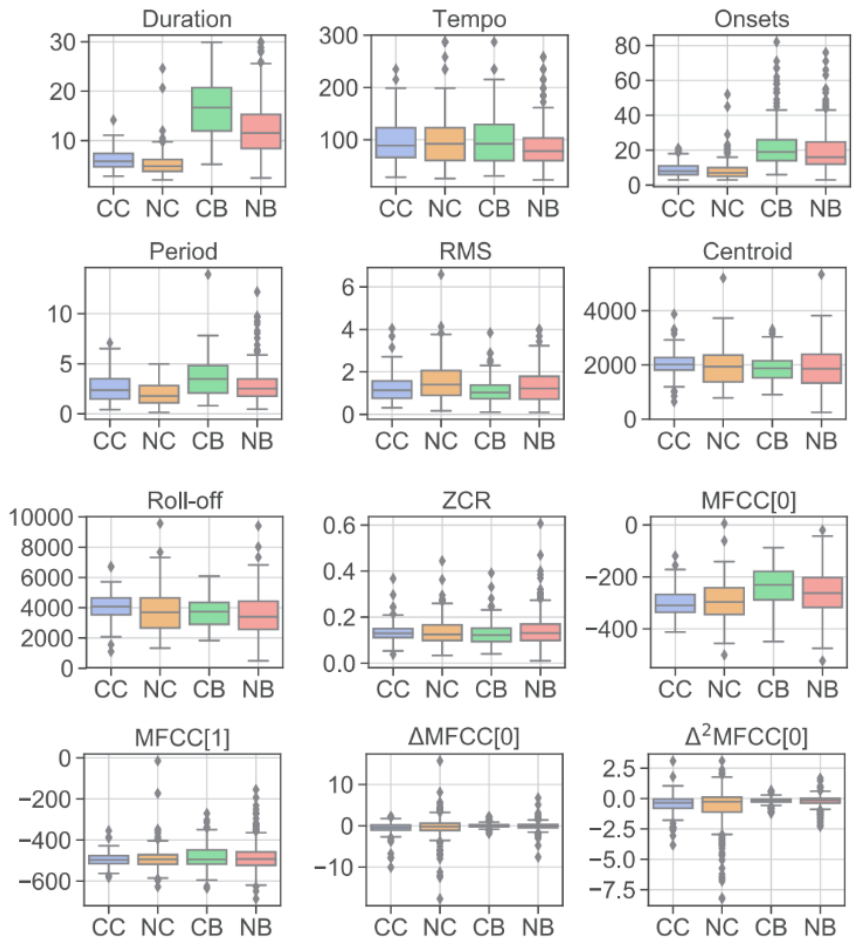


Figura 4.3: Box plot dei valori di media per le feature estratte da registrazioni di respirazioni e colpi di tosse. CC: Tosse COVID; NC: Tosse non-COVID; CB: Respiro COVID; NB: Respiro non-COVID.

2. Feature estratte con VGGish con PCA = 0.95 per i task 1 e 3; PCA = 0.9 per il task 2.

3. Feature estratte manualmente + feature estratte con VGGish con PCA = 0.95 per il task 1; PCA = 0.9 per il task 2 e PCA = 0.7 per il task 3. In particolare:

   - (A) feature VGGish + durata, numero di onset, tempo, periodo;
   - (B) tutte le feature eccetto Δ-MFCC e Δ^2-MFCC.
   - (C) tutte le feature.

I risultati di tutti gli esperimenti sono espressi tramite tre metriche: ROC-AUC, precision e recall (figura 4.4).

Task	Tipo di feature	ROC-AUC	Precision	Recall
1	1	0.71 (0.08)	0.69 (0.09)	0.66 (0.14)
	2	0.78 (0.07)	0.72 (0.08)	0.67 (0.11)
	3(A)	**0.80 (0.07)**	0.72 (0.06)	0.69 (0.11)
2	1	0.65 (0.22)	0.62 (0.20)	0.69 (0.14)
	2	0.82 (0.16)	0.79 (0.16)	0.71 (0.23)
	3(A)	**0.82 (0.18)**	0.80 (0.16)	0.72 (0.23)
3	1	0.76 (0.30)	0.64 (0.29)	0.72 (0.31)
	2	0.72 (0.16)	0.77 (0.22)	0.47 (0.15)
	3(B)	**0.80 (0.14)**	0.69 (0.20)	0.69 (0.26)

Tabella 4.1: Risultati senza data augmentation.

Task	Tipo di feature	ROC-AUC	Precision	Recall
2	1	0.58 (0.22)	0.52 (0.19)	0.85 (0.30)
	2	0.73 (0.20)	0.57 (0.16)	0.92 (0.24)
	3(C)	**0.87 (0.14)**	0.70 (0.15)	0.90 (0.18)
3	1	0.77 (0.18)	0.68 (0.09)	0.90 (0.16)
	2	0.88 (0.15)	0.63 (0.22)	0.82 (0.32)
	3(B)	**0.88 (0.12)**	0.61 (0.22)	0.81 (0.31)

Tabella 4.2: Risultati con data augmentation.

Distinzione di utenti postivi al COVID da utenti sani I risultati del task 1 indicano che il modello creato con due tipi diversi di audio (respirazioni e colpi di tosse) risulta piuttosto accurata con una AUC dell'80% e precision e recall che si aggirano intorno al 70%. La grandezza maggiore del dataset rispetto a quello utilizzato per i task 2 e 3 genera una minore deviazione standard.
È stato anche osservato che unire le feature estratte manualmente con quelle estratte con VGGish genera migliori risultati rispetto al loro uso separato.

Distinzione di tosse COVID da altri tipi di tosse Il miglior risultato ottenuto senza data augmentation mostra una AUC dell'82%, mentre la precision si attesta intorno all'80%, anche questo un buon risultato. La recall equivale circa al 70% e suggerisce che il modello è sì buono, ma anche specializzato sul dataset in oggetto.

Con data augmentation, utilizzata per i task 2 e 3 a causa della presenza di pochi campioni audio, le performance in ROC-AUC sono migliorate del 5% per il task 2 e dell'8% per il task 3. In entrambi i casi la deviazione standard è leggermente diminuita.

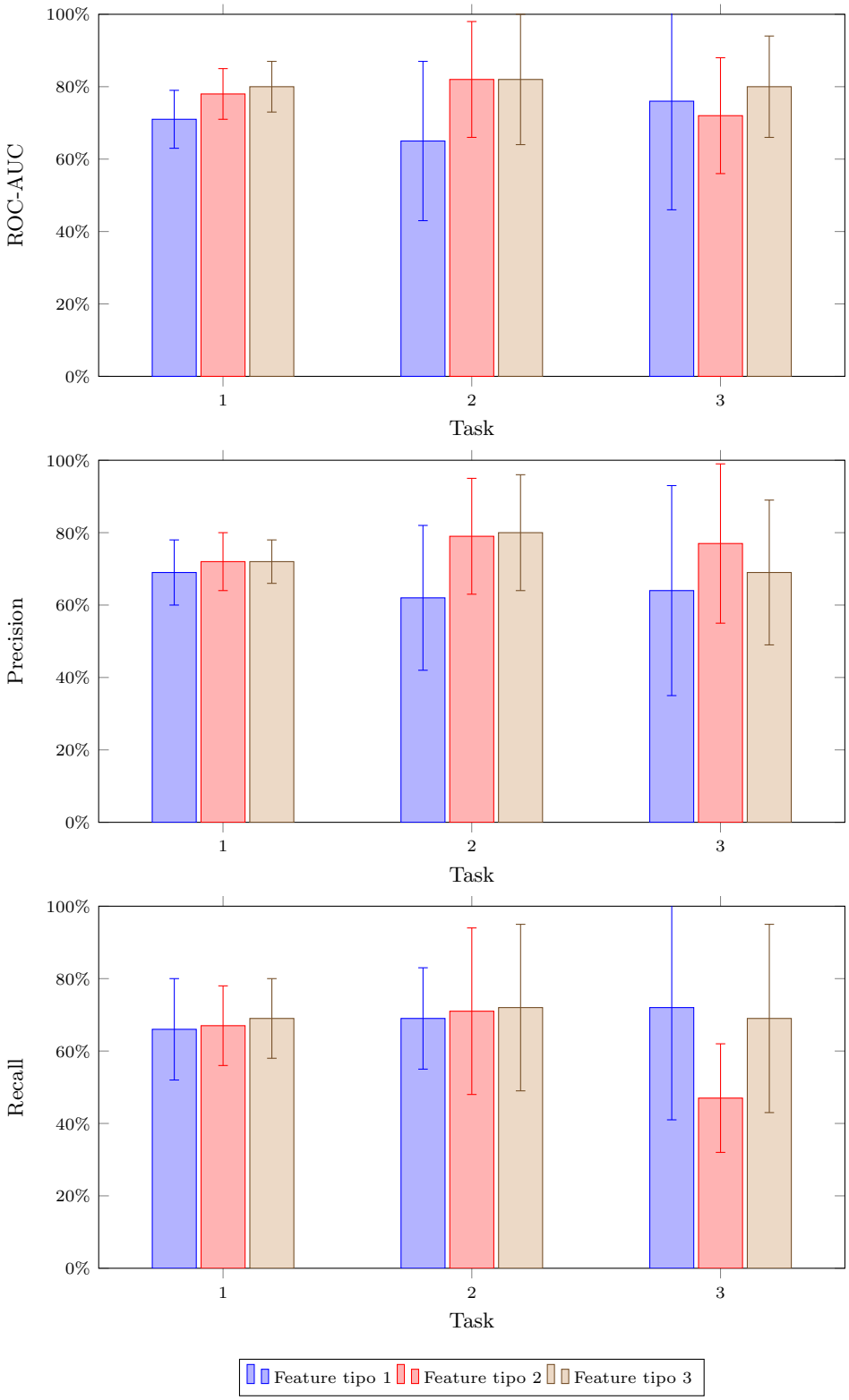


Figura 4.4: Valori di ROC-AUC, precision e recall risultanti gli esperimenti senza data augmentation. Per i task 1 e 2 sono state usate feature di tipo 3(A), mentre per il task 3 feature di tipo 3(B).

Capitolo 5

Metodi - Parte 1

Questo capitolo e il successivo contengono una descrizione dettagliata di tutto quello che è stato il procedimento volto alla replica del progetto. La descrizione dei passi è stata curata in modo da poter essere replicabile.
Qui è illustrato l'iter per la riproduzione dei risultati dei ricercatori dell'Università di Cambridge.

5.1 Software

Gli esperimenti sono stati effettuati utilizzando il linguaggio *Python* sulla piattaforma *Jupyter Notebook*[1] e la libreria *scikit-learn* [51], mentre per la parte di elaborazione dei file audio è stata utilizzata la libreria per *librosa* [43].

5.2 Dati

Grazie ad un accordo con l'Università di Cambridge è stato possibile accedere al dataset delle registrazioni audio, in formato `.wav`, utilizzato nella loro ricerca. L'archivio è composto da 491 coppie di registrazioni (respirazione e colpi di tosse) per un totale di 982 file audio ed organizzato nel seguente modo:

- `covidandroidnocough`: tracce audio di persone positive alla COVID-19 registrate con l'applicazione per Android (64 respirazioni e 64 colpi di tosse).
- `covidandroidwithcough` tracce audio di persone positive alla COVID-19 e che hanno sintomi di tosse registrate con l'applicazione per Android (46 respirazioni e 46 colpi di tosse).
- `covidwebnocough` tracce audio di persone positive alla COVID-19 registrate con l'applicazione web (23 respirazioni e 23 colpi di tosse).

[1] *Project Jupyter*. URL: `https://jupyter.org/`.

- `covidwebwithcough` tracce audio di persone positive alla COVID-19 e che hanno sintomi di tosse registrate con l'applicazione web (8 respirazioni e 8 colpi di tosse).
- `healthyandroidnosymp` tracce audio di persone totalmente sane registrate con l'applicazione Android (137 respirazioni e 137 colpi di tosse).
- `healthyandroidwithcough` tracce audio di persone negative alla COVID-19 ma che hanno sintomi di tosse registrate con l'applicazione Android (8 respirazioni e 8 colpi di tosse).
- `healthywebnosymp` tracce audio di persone totalmente sane registrate con l'applicazione web (161 respirazioni e 161 colpi di tosse).
- `healthywebwithcough` tracce audio di persone negative alla COVID-19 ma che hanno sintomi di tosse registrate con l'applicazione web (24 respirazioni e 24 colpi di tosse).
- `asthmaandroidwithcough` tracce audio di persone negative alla COVID-19 ma che soffrono, o hanno sofferto, di asma e che hanno sintomi di tosse registrate con l'applicazione web (13 respirazioni e 13 colpi di tosse).
- `asthmawebwithcough` tracce audio di persone negative alla COVID-19 ma che soffrono, o hanno sofferto, di asma e che hanno sintomi di tosse registrate con l'applicazione web (7 respirazioni e 7 colpi di tosse).

5.3 Feature extraction

Questa fase è un passo fondamentale di tutto il progetto, poiché bisogna estrarre dai file grezzi le informazioni che servono ai modelli per imparare a classificare.
I ricercatori di cambridge hanno usato due metodi diversi per questa operazione: il primo consiste nell'estrazione tramite l'uso di funzioni di audio processing e di calcolo statistico, che ha richiesto la scrittura di un codice ad hoc; nell'altro è stata usata la rete VGGish[2] per l'estrazione automatica.

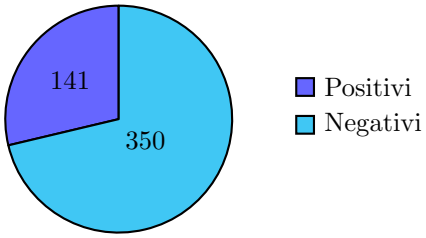
5.3.1 Feature estratte manualmente

Grazie alle funzioni messe a disposizione da *librosa* [43] questa parte è molto semplice.

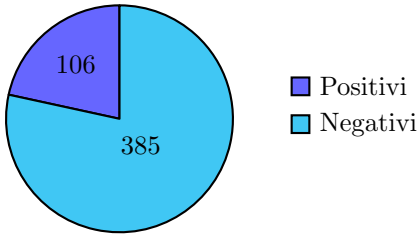
Prima di tutto occorre caricare il file audio, che si può fare con la funzione `load()`.

```
signal, sr = librosa.load(filepath)
```

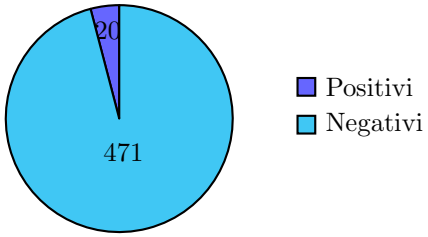
[2] *Tensorflow Models: VGGish.* Consultato il 7/03/2021. URL: `https://github.com/tensorflow/models/tree/master/research/audioset/vggish`.



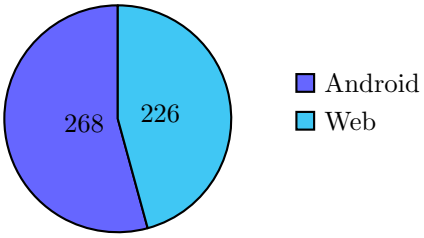
(a) Audio divisi in base alla sintomatologia di COVID-19.



(b) Audio divisi in base alla sintomatologia di tosse.



(c) Audio divisi in base alla sintomatologia di asma.



(d) Audio divisi in base al mezzo di registrazione.

Figura 5.1: Grafico che mostra il raggruppamento dei dati in base ad alcune loro proprietà.

Possiamo così ottenere un *signal*, ovvero un array di interi che descrive il segnale audio. Il secondo parametro restituito è il *sampling rate*. Una volta caricato, dal segnale devono essere tolte le parti silenziose presenti all'inizio e alla fine della registrazione.

```
signal, _ = librosa.effects.trim(signal)
```

Di seguito sono riportati i frammenti di codice utilizzati per estrarre le relative feature dai file audio.

Durata

```
duration = librosa.get_duration(signal)
```

Onset

```
onset_env = librosa.onset.onset_strength(signal, sr=sr)
times = librosa.times_like(onset_env, sr=sr)
onset_frames = librosa.onset.onset_detect(onset_envelope=onset_env, sr=sr)
```

Tempo

```
tempo = librosa.beat.tempo(onset_envelope=onset_env, sr=sr)
```

Periodo

```
period = np.argmax(librosa.stft(signal, n_fft=n_fft))
```

RMS Energy

```
rmse = librosa.feature.rms(signal)[0]
```

Spectral centroid

```
sc = librosa.feature.spectral_centroid(signal)
```

Frequenza Roll-off

```
rolloff = librosa.feature.spectral_rolloff(signal, sr, roll_percent=0.85)
```

Zero-crossing rate

```
zc = librosa.feature.zero_crossing_rate(signal)
```

MFCC

```
mfcc = librosa.feature.mfcc(signal, sr)
mfcc_delta = librosa.feature.delta(mfcc)
mfcc_delta2 = librosa.feature.delta(mfcc, order=2)
```

5.3.2 Feature estratte da VGGish

Le modalità utilizzate per l'estrazione delle feature utilizzando VGGish sono le stesse descritte nella documentazione della repository su GitHub [65]. Il pacchetto contiene uno script Python eseguibile da riga di comando che prende in input il file audio e restituisce un array di interi che rappresentano le feature estratte.

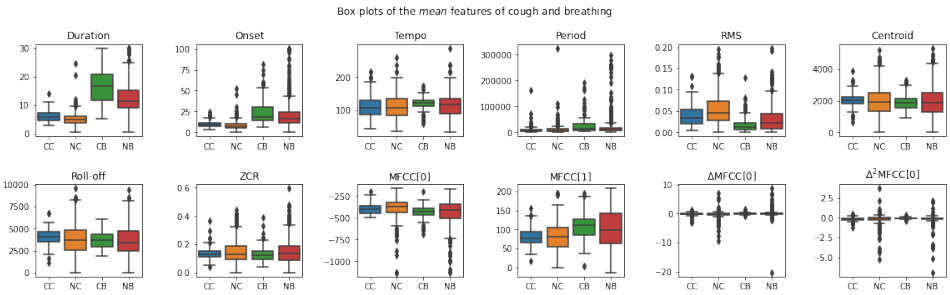


Figura 5.2: Box plot dei valori di media per le feature estratte manualmente (replica dell'esperimento).

5.4 Task

Come è stato descritto precedentemente sono stati creati tre diversi task, per ognuno dei quali è stato creato un sottoinsieme di dati. Ecco quali dati sono stati utilizzati per ogni task:

Task 1

Positivi (Classe 1)	Negativi (Classe 0)
`covidandroidnocough`	
`covidandroidwithcough`	`healthyandroidnosymp`
`covidwebnocough`	`healthywebnosymp`
`covidwebwithcough`	

Task 2

Positivi (Classe 1)	Negativi (Classe 0)
`covidandroidwithcough`	`healthyandroidwithcough`
`covidwebwithcough`	`healthywebwithcough`

Task 3

Positivi (Classe 1)	Negativi (Classe 0)
`covidandroidwithcough`	`asthmaandroidwithcough`
`covidwebwithcough`	`asthmawebwithcough`

5.5 Valutazione

Per stabilire quanto sono buoni i modelli costruiti è importante effettuare la fase di valutazione, nella quale si possono utilizzare numerose metriche. Seguendo l'esperimento di *Brown et al.* sono state utilizzate le metriche: ROC-AUC, precision e recall.

5.5.1 Ottimizzazione degli iperparametri

Nella costruzione di un buon modello di apprendimento automatico la ricerca degli iperparametri migliori è un passo fondamentale. *Scikit-learn* mette da disposizione uno strumento chiamato `GridSearchCV` che permette di effettuare valutazioni ripetuto con lo stesso modello ma con iperaparametri diversi per poi confrontare i risultati e capire quali sono le impostazioni per ottenere le performance migliori. `GridSearchCV` prende in input un dizionario in cui ogni chiave k corrisponde ad un iperparametro del modello che si vuole ottimizzare e il valore associato a k corrisponde all'insieme di valori che si vogliono provare per k.

```
{
...
parametro: [valore1, valore2, ..., valoreN],
...
}
```

Supponiamo quindi che il dizionario contenga p iperparametri, ognuno dei quali contiene n_p valori possibili; le valutazioni del modello che verranno effettuate saranno

$$\prod_{i=0}^{p} n_p$$

ovvero in numero di combinazioni possibili tra tutti gli iperparametri. Infine, per ogni valutazione verrà restituito l'insieme degli iperparametri su cui è stata effettuata e i risultati numerici della valutazione (es. accuracy, ROC-AUC, ecc..).

Un fatto importante da menzionare è che non esiste in assoluto un'impostazione migliore, ma dipende molto dal problema affrontato. Un settaggio ottimo per un tipo di problema potrebbe rivelarsi uno dei peggiori per altri tipi di problema.

Il suddetto metodo è stato applicato ai classificatori *Logistic Regression* e *SVM*. In particolare, per il classificatore *Logistic Regression* sono stati provati i seguenti iperparametri:

- `solver`: *newton-cg*, *lbfgs*, *liblinear*;
- `penalty`: *l2*;
- `C`: *100*, *10*, *1.0*, *0.1*, *0.01*.

mentre per il classificatore SVM sono stati utilizzati i seguenti:

- `kernel`: *linear*, *poly*, *rbf*, *sigmoid*;
- `gamma`: *auto*;
- `C`: *100*, *10*, *1.0*, *0.1*, *0.001*.

5.5.2 Convalida incrociata

La valutazione dei modelli è avvenuta tramite convalida incrociata [9] poiché questo metodo è uno standard per questa fase. Per i classificatori di tipo logistic regression o SVM si può utilizzare la funzione della libreria Scikit-learn [51] `sklearn.model_selection.cross_val_score` mentre per i modelli costruiti con Keras [35] si deve effettuare questa operazione manualmente.

5.6 Risultati

Nella tabella 5.1 sono stati riportati i risultati della replica dell'esperimento.

Task	**Modalità**	**Samples**	**Estimator**	**Features**	**Mean ± std**		
					ROC-AUC	**Precision**	**Recall**
1	C + B	141/295	Logistic Regression	Handcrafted	0.82 (0.04)	0.67 (0.07)	0.62 (0.08)
				VGGish	0.78 (0.05)	0.60 (0.08)	0.65 (0.09)
				Handcrafted + VGGish	**0.85 (0.05)**	0.57 (0.06)	0.83 (0.07)
2	C	27/8	SVM	Handcrafted	0.81 (0.15)	0.77 (0.07)	1.00 (0.00)
				VGGish	**0.97 (0.10)**	0.97 (0.07)	1.00 (0.00)
				Handcrafted + VGGish	0.93 (0.19)	0.98 (0.06)	0.98 (0.06)
3	B	27/10	SVM	Handcrafted	0.60 (0.22)	0.80 (0.11)	0.79 (0.18)
				VGGish	0.60 (0.22)	0.80 (0.11)	0.79 (0.18)
				Handcrafted + VGGish	**1.00 (0.01)**	0.97 (0.06)	1.00 (0.00)

Tabella 5.1: Risultati della replica dell'esperimento di *Brown et al.*

Capitolo 6

Metodi - Parte 2

In questo capitolo è mostrata la parte principale del progetto, dove si è provato ad utilizzare i dati forniti dall'Università di Cambridge applicando il metodo di image classification utilizzando gli spettrogrammi come input per le Deep Convolutional Neural Networks.

6.1 Strumenti

Gli esperimenti sono stati effettuati utilizzando il linguaggio *Python* sulla piattaforma *Jupyter Notebook*[1] e le librerie *Keras* [35] e *Tensorflow* [1] per l'implementazione dei modelli di apprendimento automatico, mentre per la parte di elaborazione dei file audio è stata utilizzata la libreria per *librosa* [43].

Il supporto hardware utilizzato si compone di una GPU Titan X con limite di memoria impostato a 6 GB (50% della capacità) per non intaccare altri esperimenti in ambito accademico che sono stati effettuati nello stesso periodo per scopi di ricerca.

Il lavoro è stato effettuato da remoto utilizzando il protocollo Secure Shell (SSH), seguendo i passi sotto elencati.

- È stata effettuata una connessione dall'host locale all'host remoto;

```
$ ssh [USERNAME]@[IP_HOST]
```

- sull'host remoto è stato avviato l'ambiente *Jupyter Notebook* disabilitando l'utilizzo del browser e indicando la porta 8888 come uscita;

```
$ ssh jupyter notebook --no-browser --port=8888
```

[1] *Project Jupyter*. URL: `https://jupyter.org/`.

- si è effettuato un ridirezionamento dell'indirizzo `localhost:8888` del server sull'indirizzo `localhost:8888` della macchina locale.

```
$ ssh -N -f -L localhost:8888:localhost:8888 [USERNAME]@[IP_HOST]
```

A questo punto è stato possibile accedere a *Jupyter Notebook* con il browser locale accedendo all'indirizzo `https://localhost:8888/` e utilizzando la sua interfaccia grafica proprio come se il programma girasse sulla macchina locale, mentre in realtà la computazione veniva effettuata tutta sul server.

In ausilio è stata utilizzata anche la piattaforma *Google Colab* [22] che mette a disposizione delle GPU *NVIDIA Tesla K80*.

6.2 Generazione degli spettrogrammi

librosa rende molto semplice la generazione degli spettrogrammi grazie alle funzioni che mette a disposizione. La documentazione disponibile, oltre alla descrizione delle funzioni e dei parametri che prendono in input, contiene anche alcuni esempi pratici. Per generare uno spettrogramma in scala Mel (Mel-spectrogram) basta scrivere il seguente codice[2].

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
S_dB = librosa.power_to_db(S, ref=np.max)
img = librosa.display.specshow(S_dB, x_axis='time', y_axis='mel', sr=sr,
    fmax=8000, ax=ax)
```

Infine, per salvare l'immagine come file *png*:

```
plt.axis('off')
fig.savefig(filename + ".png", bbox_inches='tight')
plt.close()
```

6.3 Data augmentation

Molti dataset audio utilizzati per costruire modelli di classificazione presentano il problema della scarsità dei campioni da utilizzare per l'apprendimento. Anche i file audio, come altri tipi di file, possono essere perturbati per generare un dataset più ricco, che è la prima arma per combattere l'overfitting [32, 48, 71].

Grazie alle funzioni messe a disposizione della libreria *librosa*, è possibile effettuare una serie di operazioni per aumentare i dati disponibili: aggiunta di rumore,

[2] *librosa.feature.melspectrogram*. URL: `https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html`.

pitch shift, time shift, time stretch[3]. Queste operazioni permettono di creare quattro nuovi esempi per ogni audio, quintuplicando la grandezza del dataset.
Gli esempi mostrati di seguito sono raffigurati utilizzando uno spettrogramma e un'onda sonora. Il file in questione, che nella figura 6.1 è nella sua forma originale, è stato creato registrando con un comune smartphone.

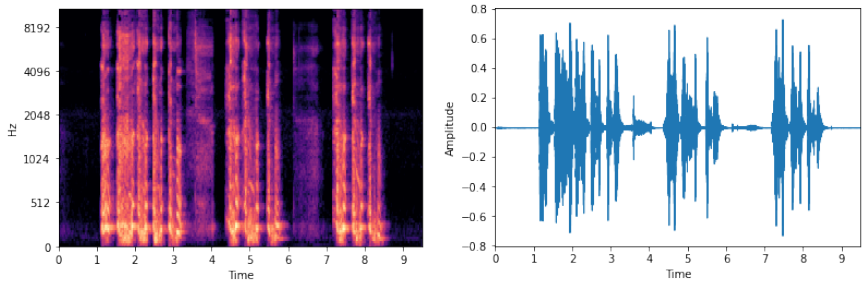


Figura 6.1: Spettrogramma e onda sonora di una persona che tossisce.

6.3.1 Rumore

Aggiungere del rumore gaussiano è un modo per per aggiungere delle variazioni casuali all'input senza però alterarlo nel complesso.

```
signal_n = signal + 0.009 * np.random.normal(0, 1, len(signal))
```

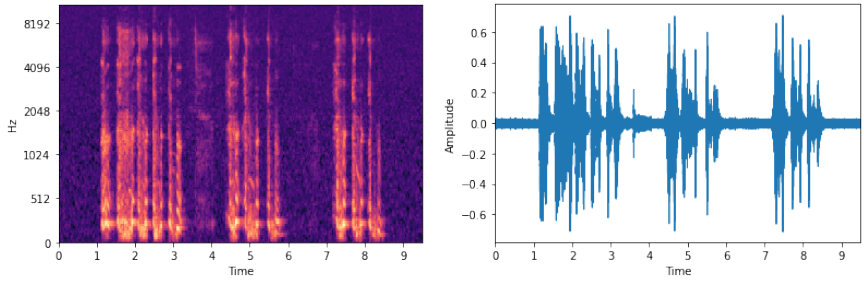


Figura 6.2: Spettrogramma e onda sonora di una persona che tossisce con aggiunta di rumore.

6.3.2 Pitch shift

L'operazione di *pitch shifting* corrisponde alla variazione della frequenza di un suono, senza però cambiare il fattore *tempo*. L'operazione reciproca è il *time shifting*.

[3]Keyur Paralkar. *Audio Data Augmentation in Python*. Consultato il 25/02/2021. URL: https://medium.com/@keur.plkar/audio-data-augmentation-in-python-a91600613e47.

```
signal_ps = librosa.effects.pitch_shift(signal, sr, n_steps=-5)
```

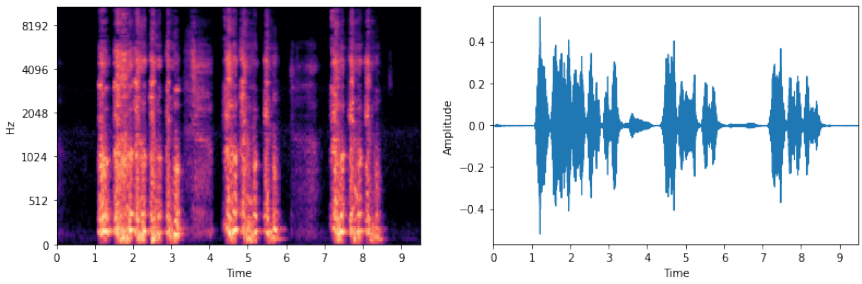


Figura 6.3: Spettrogramma e onda sonora di una persona che tossisce dopo aver effettuato *pitch shifting*.

6.3.3 Time shift

Il *time shifting* è semplicemente uno spostamento del suono sull'asse del tempo.

```
signal_ts = np.roll(signal, int(sr/10))
```

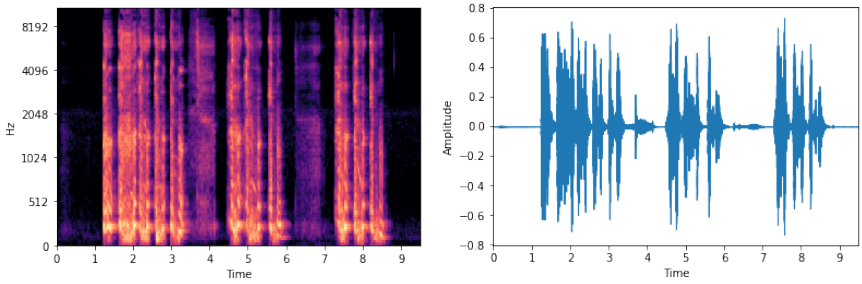


Figura 6.4: Spettrogramma e onda sonora di una persona che tossisce dopo aver effettuato *time shifting*.

6.3.4 Time stretch

Il *time stretching* è un'operazione con la quale si rallenta un suono digitale oppure lo si accelera. Nel primo caso la lunghezza dell'audio aumenta, nel secondo diminuisce.

```
factor = 0.4
signal_tst = librosa.effects.time_stretch(signal, factor)
```

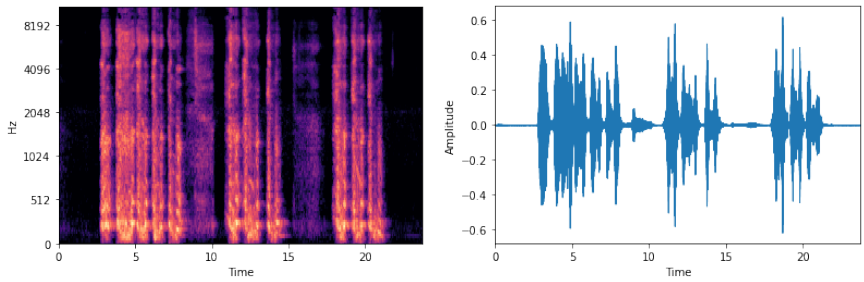


Figura 6.5: Spettrogramma e onda sonora di una persona che tossisce dopo aver effettuato *time stretch*.

6.4 Transfer learning

Seguendo uno dei tutorial ufficiali di *TensorFlow*[4] è stato costruito un modello sequenziale da utilizzare come *wrapper* per i modelli pre-addestrati. Lo schema di tale modello è mostrato in figura 6.6. Fondamentalmente il modello *wrapper* si occupa di due cose:

1. Elaborazione dell'input: il *wrapper* prende in input i dati, li preprocessa tramite apposite funzioni fornite da *Keras* e poi li manda in input alla rete neurale pre-addestrata.

2. Deep learning con feature generate: le feature che vengono estratte dai livelli convoluzionali della rete incapsulata (MobileNetV2, ResNet50, VGG16 o Xception) vengono trasferite ad una serie di *fully connected* layer sui quali viene effettuato il training.

6.5 Ottimizzazione degli iperparametri

I modelli di CNN sono strumenti molto potenti, ma per ottenere da loro le migliori prestazioni è necessario andare alla ricerca degli iperparametri migliori da utilizzare. Non esistono iperparametri migliori in assoluto, ma tutto dipende dal problema affrontato e dai dati utilizzati; questo fatto rende davvero complessa questa operazione.
Sono stati individuate delle impostazioni da poter variare per poi osservare se la valutazione del modello migliora o peggiora, e sono:

- **Neuroni**: numero di neuroni che compongono il livello finale (*dense layer*) della rete.

- **Profondità**: numero di livelli *dense* che compongono la parte finale della rete.

[4] *Transfer learning and fine-tuning — TensorFlow Core*. Consultato il 24/02/2021. URL: `https://www.tensorflow.org/tutorials/images/transfer_learning`.

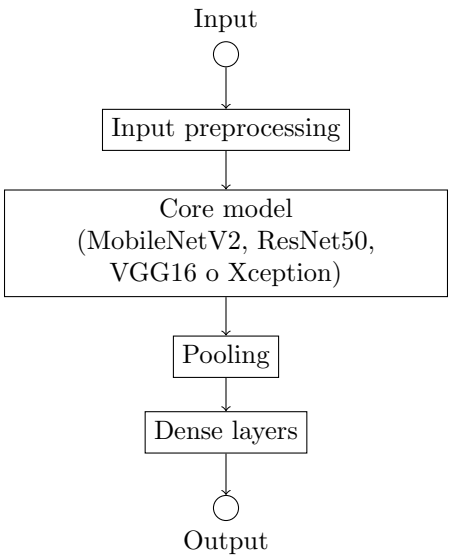


Figura 6.6: Architettura del modello *wrapper*.

- **Topologia**: i livelli finali possono presentare una struttura di tipo quadrato o triangolare, secondo le seguenti definizioni:

  - *struttura quadrata*: sia n il numero di neuroni e p la profondità, la struttura è composta da p livelli di n nodi ciascuno, per un totale di $p * n$ nodi (figura 6.7).
  - *struttura triangolare*: la struttura è composta da p livelli, ma il livello i contiene $p/2^i$ nodi (figura 6.8).

  Per $p = 1$ le due topologie sono equivalenti: si avrà solo un *hidden layer* con n nodi, mentre per $p > 1$ la topologia quadrata conterrà il doppio dei nodi della topologia triangolare.

- **Epoche**: il numero di iterazioni per effettuare la fase di addestramento è un parametro importante. Scegliere il giusto numero di epoche è necessario per evitare due problemi opposti:

  - *underfitting*: si ha quando il modello "non ha imparato abbastanza" ed è naturale che si crei se il numero di epoche impostato è troppo basso;
  - *overfitting*: è il vero problema della fase di addestramento ed è molto facile da riscontrare, in particolare quando si ha un dataset piccolo. Generalmente, però, un elevato numero di epoche porta a creare questo problema. Nella sezione 6.7 sono mostrati i metodi utilizzare per prevenirlo.

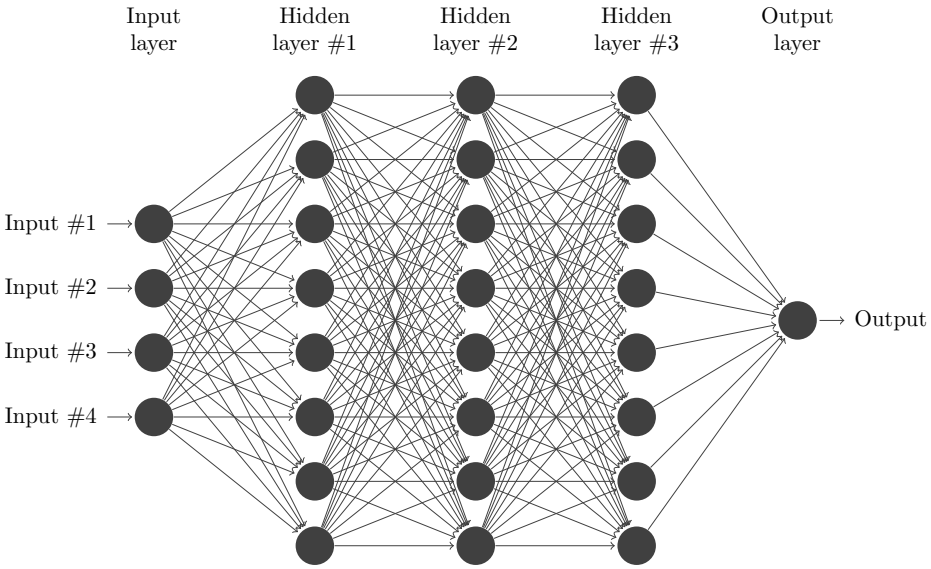


Figura 6.7: Schema di rete neurale con topologia quadrata con $n = 8$ e $p = 3$.

- **Learning rate**: in un algoritmo di ottimizzazione è un iperparametro che determina la variazione ad ogni iterazione quando ci si muove verso un minimo di una funzione obiettivo [44].
- **Batch size**: corrisponde alla quantità di esempi da sottoporre al modello durante una singola iterazione di apprendimento.
- **Dropout**: tecnica grazie alla quale è possibile "disattivare" l'aggiornamento di alcune connessione durante l'apprendimento. Questa tecnica ad ogni iterazione sceglie e "spegne" un sottoinsieme delle connessioni tra un livello e l'altro della rete.
- **Augmentation**: operazione effettuata quando si ha un insieme di dati limitato; i dati vengono perturbati in modo casuale tra un'iterazione e l'altra riuscendo così a simulare un dataset più grande di quello reale.

6.6 Augmentation su spettrogrammi

Per quanto riguarda l'utilizzo di tecniche di *augmentation* sulle immagini, in generale si effettuano rotazioni, riflessioni, capovolgimenti, inversione di colori, applicazione di filtri, ecc..[5]. Questo genere di operazioni è adatto alle fotografie di oggetti reali perché quello su cui il modello di CNN andrà a concentrarsi saranno le forme e i contorni.

[5] *Data Augmentation — TensorFlow Core*. Consultato il 24/02/2021. URL: `https://www.tensorflow.org/tutorials/images/data_augmentation`.

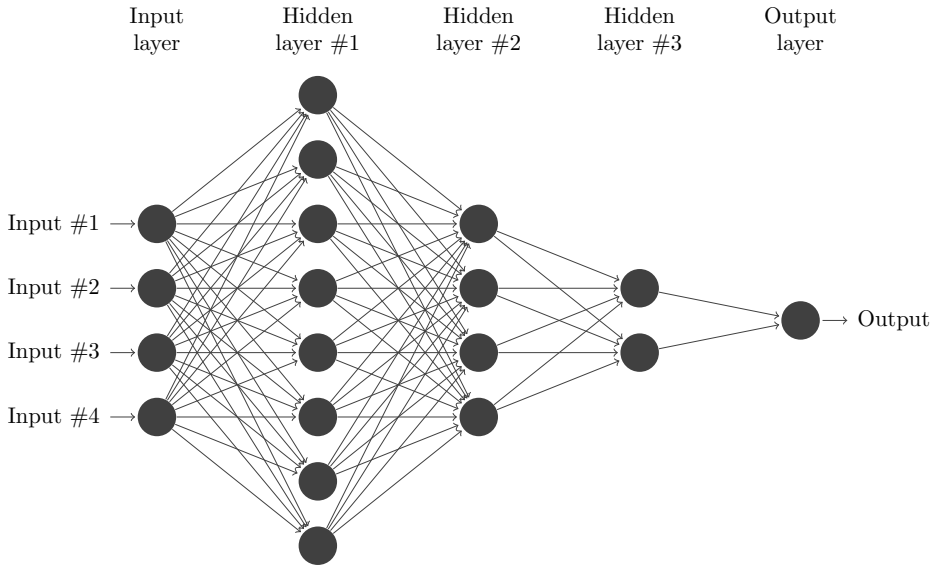


Figura 6.8: Schema di rete neurale con topologia triangolare con $n = 8$ e $p = 3$.

Gli spettrogrammi sono *heatmap*, quindi il tipo di operazioni elencate sopra porterebbe ad una falsificazione dei dati.

- Il colore dei punti indica il livello di intensità del suono: perturbare il colore significherebbe cambiare l'intensità del suono.
- L'asse delle ordinate rappresenta le diverse frequenze d'onda: effettuare uno shift verticale dell'immagine equivarrebbe a cambiare le frequenze dei suoni.
- L'asse delle ascisse rappresenta lo scorrere del tempo durante la manifestazione del suono: effettuare uno shift orizzontale dell'immagine equivarrebbe a cambiare le l'istante temporale in cui si manifesta il suono.

Per quanto riguarda l'ultimo punto si può effettuare una perturbazione con uno shift orizzontale. In fin dei conti, non è fondamentale ai fini della risposta che deve dare il modello se un certo suono appare al tempo t oppure al tempo $t \pm \delta$.

6.7 Prevenzione dell'overfitting

La prevenzione dell'overfitting è una parte cruciale nella costruzione di un modello di *machine learning*. Come scegliere il numero di epoche giuste considerando i pro e i contro descritti nelle sezioni precedenti? Ci sono due metodi.

- **Early stopping**: è un controllo che può essere passato come *callback* alla funzione `fit()` e che ha come parametri:
  - `monitor`: la metrica da monitorare (solitamente è la *loss* sul validation set);

  - `min_delta`: la differenza minima sul parametro monitorato che fa attivare il controllo;
  - `patience`: numero di epoche di tolleranza entro le quali verificare se c'è stato un miglioramento del valore.

  L'*early stopping* si occupa di interrompere il processo di addestramento qualora il valore monitorato non abbia presentato un miglioramento. Molto comune è il fenomeno in cui la *loss* sul validation set aumenta insieme all'aumentare dell'*accuracy* sul training set: un chiaro segno di overfitting.

- **Model checkpoint**: come il precedente, è un controllo che può essere passato come *callback* alla funzione `fit()` e che ha come parametri:
  - `filepath`: percorso del file in cui salvare il modello;
  - `monitor`: la metrica da monitorare (solitamente è la *loss* sul validation set);
  - `mode`: può essere *auto*, *min* o *max* e indica su quale base decidere il modello da salvare.

  Model checkpoint salva sempre il modello migliore. In questo modo è possibile recuperarlo alla fine della fase di addestramento.

In questo progetto non è stato utilizzato un *early stopping* perché spesso capita che questo strumento venga ingannato da un minimo locale. Per quanto concerne l'utilizzo di un *model checkpoint*, è stato scelto anche in questo caso di non ricorrere a questa soluzione poiché è importante sapere qual è il numero giusto di epoche su cui addestrare il modello e dovendo effettuare una valutazione basata su *cross validation*, il numero di epoche deve essere fisso per tutti i round (o fold) della valutazione.

6.8 Valutazione

Quando si effettua un partizionamento dei dati in training, validation e test set si verifica un problema: non c'è la certezza che questo partizionamento sia il migliore possibile; d'altro canto però provare tutti i partizionamenti possibili porterebbe ad un lavoro senza fine.

La soluzione adottata è quella mostrata in figura 6.9: si effettuano più round di addestramento e valutazione del modello, ogni round con partizioni diverse. In questo caso l'ordinamento dei dati è lasciato inalterato, ma si può optare per uno *shuffling randomico*.

L'algoritmo è il seguente:

1. si divide l'intera sequenza di dati in p_t partizioni, dove p_t deve rappresentare la percentuale di splitting utilizzata (per esempio, con uno splitting 80%-20%, p_t sarà 5, cioè 100%/20%);

2. si effettuano p_t round, in ognuno dei quali si designa una diversa partizione p_t come test set, poi

   (a) si prendono le $1 - p_t$ partizioni e le si dividono come al punto 1 (si può cambiare anche la percentuale di splitting) in p_v partizioni;

   (b) si effettuano p_v subround di addestramento e valutazione, ognuno dei quali avrà una delle p_v partizioni designata come validation set e le altre come training set.

Per ognuno dei 25 round sono state utilizzate delle funzioni built-in per calcolare le metriche: per prima cosa sono stati calcolati i valori `Y_pred` che corrispondono ai valori di output predetti dal modello per l'input `X_test`, poi con la funzione `metrics.roc_auc_score()` di *scikit-learn* è stato calcolato il valore di AUC. Infine, per rendere possibile successivamente il disegno del grafico ROC-AUC, è stata utilizzata la funzione `metrics.roc_curve()`, anch'essa di *scikit-learn*, per otteenere i vettori con i valori di falsi positivi, veri positivi e delle soglie.

```
Y_pred = model.predict(X_test)
auc = metrics.roc_auc_score(Y_test, Y_pred)
fpr, tpr, thresholds = metrics.roc_curve(Y_test, Y_pred)
```

6.8.1 Equa rappresentanza delle classi

L'algoritmo illustrato al paragrafo precedente non tiene conto di un ulteriore problema che può facilmente verificarsi e che è effettivamente accaduto durante lo sviluppo di questa fase. Di seguito il problema e la relativa soluzione è spiegata per il una classificazione binaria, ma si può facilmente generalizzare al caso di una classificazione con più di due classi. Se abbiamo il vettore degli output che contiene tutti valori della prima classe all'inizio e tutti i valori della seconda classe alla fine, è facile intuire che utilizzando l'algoritmo di splitting è facile trovarsi con dei set che contengono solo valori per una classe. In questo caso l'altra classe non è rappresentata e gli effetti possono essere diversi a seconda della casistica.

- Il training set non contiene valori per entrambe le classi:
  - se il validation set contiene valori per entrambe le classi, si verificherà un errore in fase di training e verrà sollevata un'eccezione;
  - se il validation set non contiene valori per entrambe le classi, si verificherà un errore in fase di valutazione e verrà sollevata un'eccezione. Questo è il caso più delicato in quanto solo la valutazione del modello può far notare il problema, poiché la fase di addestramento terminerà con un'accuracy del 100% (anche se questo numero dovrebbe sempre essere un campanello d'allarme).

- Uno tra il validation set o il test set non contiene valori per entrambe le classi: la valutazione finale del modello sarà falsata.

Soluzione Si dividono i dati in tante sequenze quante sono le classi, si effettuano gli splitting per ogni singola classe e si ricombinano nel momento in cui bisogna inviare i dati al modello. Con questo metodo, anche se le classi non sono bilanciate, tale sbilanciamento sarà identico, in termini percentuali, sia nel training set, sia nel validation set, sia nel test set. Questo è esattamente il motivo per cui in figura 6.9 sono riportati due grafici, uno per ogni classe.

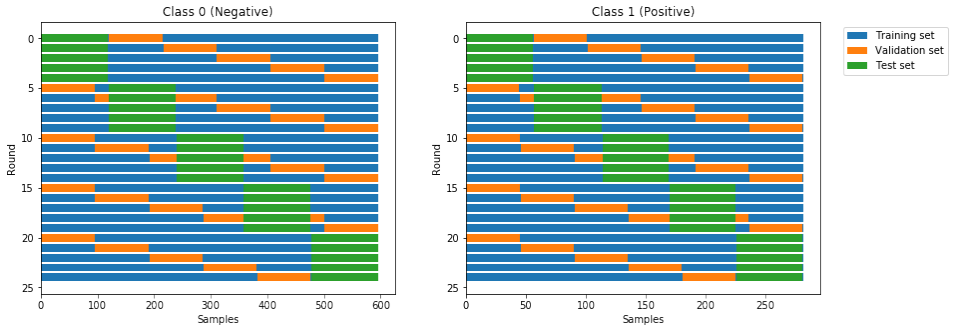


Figura 6.9: Schema di convalida incrociata annidata per un classificatore binario.

6.8.2 Errori da evitare

Lo scopo di creare modelli con l'apprendimento automatico è fare in modo che questi modelli diano risposte affidabili quando dovranno effettuare predizioni su dati che non hanno mai visto in precedenza. Per creare questo presupposto, ci sono dei vincoli fondamentali da tenere in considerazione:

- ad ogni iterazione nessun *sample* deve appartenere a più di un insieme e questo significa che training set, validation set e test set devono essere sempre **insiemi disgiunti**;
- il modello che si addestra deve essere reinizializzato all'inizio di ogni iterazione, altrimenti potrebbe aver effettuato un'iterazione di training su dati che si trovano sul test set;
- se si effettua *data augmentation*, lo si deve fare solo nel training set, altrimenti nel test set si avrebbero dati che non rispondono a quello che ci si aspetterebbe in uno scenario reale. Durante lo sviluppo di questi esperimenti è stato fatto questo errore (e corretto, ovviamente) e i risultati sono stati del 90-95% in ROC-AUC: valori alti ma non veritieri.

Se non si rispettano i vincoli suddetti il modello sarebbe in grado di ottenere buoni risultati in fase di valutazione perché "conoscerebbe" alcune risposte alle domande

su cui viene valutato. Il risultato della valutazione complessiva del modello sarà quindi ottimistico, ovvero migliore rispetto a quello che è nella realtà.

6.9 Curve ROC

Una volta effettuata la valutazione tramite validazione incrociata, si hanno a disposizione 25 curve ROC, una per ogni round di valutazione del modello.
Quando si ha un vettore di risultati il modo migliori per aggregarli è quello di utilizzare i concetti di media e deviazione standard, che grazie alla libreria *numpy* [24] sono facilmente calcolabili su una collezione di vettori. Data una sequenza di m vettori V_i, ciascuno di lunghezza n, si possono ottenere i vettori:

$$M_i = \frac{\sum_{j=0}^{n-1} V_{i,j}}{n} \qquad \text{per } i \in [0...m-1]$$

$$S_i = \sqrt{\frac{\sum_{j=0}^{n-1} (V_{i,j} - M_i)^2}{n}} \qquad \text{per } i \in [0...m-1]$$

dove M è il vettore dei valori medi e S è il vettore delle deviazioni standard (o scarti quadratici medi).

Capitolo 7

Risultati

In questo capitolo sono illustrati i risultati ottenuti eseguendo gli esperimenti con le metodologie presentate nel capitolo precedente e confrontati con i risultati pubblicati dai ricercatori dall'Università di Cambridge.

Premessa I risultati presentati di seguito sono frutto di fasi di addestramento dei modelli attraverso l'uso di una particolare configurazione di iperparametri. Tale configurazione è stata raggiunta dopo aver sperimentato altre configurazioni che non si sono rilevate buone. Condurre una ricerca esaustiva è stato di fatto impossibile a causa dell'elevato numero di variabili che si incontrano durante questo tipo di esperimenti. Pertanto, non è stato provato che questi siano i risultati migliori, anche se il confronto con lo stato dell'arte rivela che sono molto buoni. La configurazione utilizzata per ottenere i risultati è descritta nella sezione 7.1.

Task	**Modalità**	**CNN**	**Mean ± std**			
			ROC-AUC	**Precision**	**Recall**	**Specificity**
1	C + B	MobileNetV2	0.80 (0.02)	0.60 (0.08)	0.65 (0.08)	0.79 (0.08)
		ResNet50	**0.81 (0.05)**	**0.66 (0.07)**	0.65 (0.07)	**0.84 (0.05)**
		VGG16	0.80 (0.02)	0.64 (0.02)	0.66 (0.01)	0.82 (0.02)
		Xception	**0.81 (0.05)**	0.59 (0.06)	**0.67 (0.12)**	0.78 (0.06)
2	C	MobileNetV2	**0.80 (0.14)**	**0.75 (0.12)**	0.83 (0.17)	0.47 (0.37)
		ResNet50	0.79 (0.06)	0.67 (0.04)	**0.87 (0.11)**	0.28 (0.07)
		VGG16	0.71 (0.12)	0.74 (0.11)	0.81 (0.09)	0.46 (0.36)
		Xception	0.62 (0.06)	0.70 (0.10)	0.71 (0.15)	**0.47 (0.25)**
3	B	MobileNetV2	**0.85 (0.15)**	0.83 (0.10)	0.91 (0.13)	0.45 (0.37)
		ResNet50	0.77 (0.09)	**0.84 (0.07)**	0.87 (0.08)	**0.55 (0.21)**
		VGG16	0.84 (0.09)	0.79 (0.03)	**0.89 (0.15)**	0.35 (0.14)
		Xception	0.77 (0.04)	0.79 (0.05)	0.83 (0.16)	0.35 (0.29)

Tabella 7.1: Risultati principali.

Nella tabella 7.1 sono riportati i valori di ROC-AUC, precision, recall e sensitivity ottenuti da ciascuno dei quattro modelli di CNN sui tre task. Ogni metrica,

Task	Mean ± std			
	ROC-AUC	**Precision**	**Recall**	**Specificity**
1	**0.81 (0.04)**	0.62 (0.06)	0.66 (0.07)	**0.81 (0.05)**
2	0.73 (0.10)	0.72 (0.09)	0.81 (0.11)	0.42 (0.26)
3	0.81 (0.09)	**0.81 (0.06)**	**0.88 (0.13)**	0.43 (0.25)

Tabella 7.2: Risultati principali - medie per ogni task.

CNN	Mean ± std			
	ROC-AUC	**Precision**	**Recall**	**Specificity**
MobileNetV2	**0.82 (0.10)**	**0.73 (0.09)**	0.80 (0.13)	**0.57 (0.27)**
ResNet50	0.79 (0.07)	0.72 (0.06)	**0.80 (0.07)**	0.56 (0.11)
VGG16	0.78 (0.08)	0.72 (0.05)	0.79 (0.08)	0.54 (0.17)
Xception	0.73 (0.05)	0.69 (0.07)	0.73 (0.14)	0.53 (0.20)

Tabella 7.3: Risultati principali - medie per ogni CNN.

come è consuetudine, riporta il valore medio e la deviazione standard. I risultati sono stati aggregati nelle tabelle 7.2 e 7.3 che riportano i valori medi per le tre metriche ottenuti, rispettivamente, raggruppando i task e raggruppando le CNN. Infine, in grassetto sono evidenziati i punteggi migliori.

7.1 Configurazione utilizzata

La rete è stata configurata come mostrato in figura 6.6, con un primo *hidden layer* di 256 nodi ed un secondo di 128 nodi. Ad ogni layer è stato applicato un *dropout* di 0.2 ed all'inizio di ogni epoca viene effettuato un *shift* orizzontale dell'input in modalità *wrap* per simulare data augmentation. Il valore di shifting è casuale tra 0 e 0.2 e l'opzione "wrap" permette di recuperare dalla parte opposta dell'immagine i pixel che vanno in overflow. Le impostazioni degli iperparametri utilizzate sono quelle più comuni. In particolare:

- dimensioni dell'immagine: tutte 224×224 tranne per i task 2 e 3, per i quali con Xception è stata usata la dimensione 299×299;
- epoche: 100;
- learning rate: 10^{-3};
- batch size: 32.

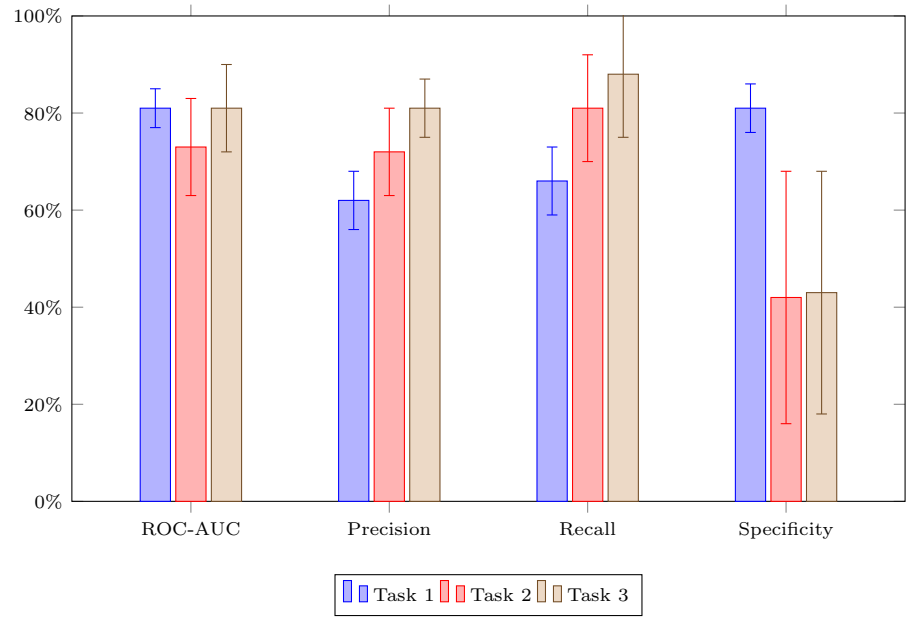


Figura 7.1: Rappresentazione grafica dei risultati principali - medie per ogni task.

7.2 Analisi sui task

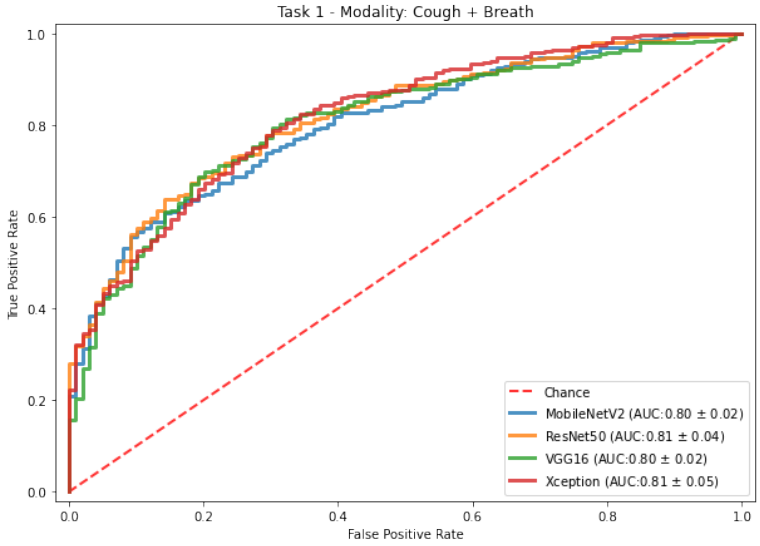


Figura 7.3: Grafico ROC-AUC che illustra i risultati delle CNN sul task 1.

Per quanto riguarda il **task 1**, nonostante l'uso combinato di registrazioni di tosse e di respiri possa sembrare che aggiunga una difficoltà in più, sono stati registrati i risultati migliori. Sicuramente il fatto che i campioni su cui addestrare siano stati di più rispetto agli altri task ha contribuito ad ottenere buone prestazioni. Tutte le

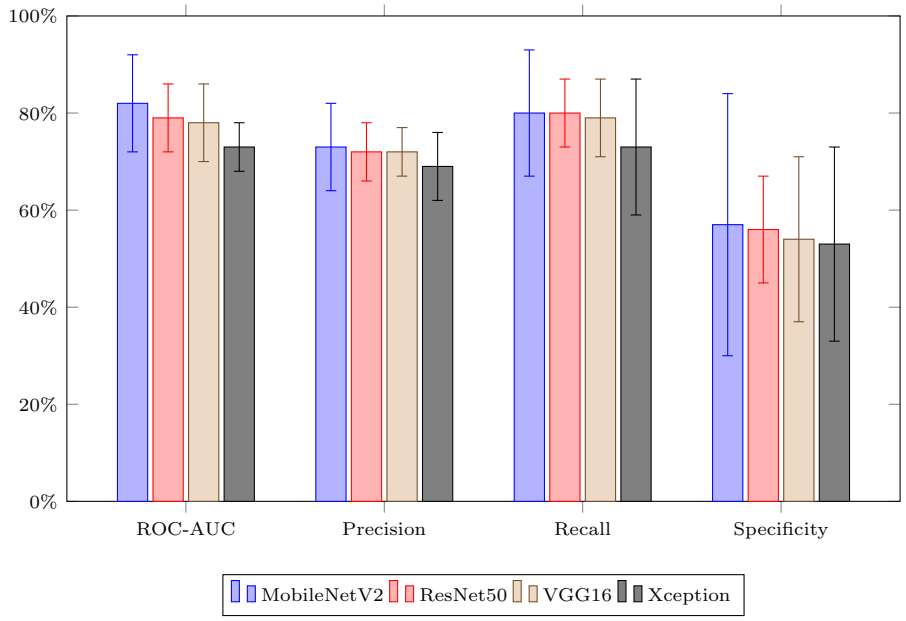


Figura 7.2: Rappresentazione grafica dei risultati principali - medie per ogni CNN.

quattro reti si comportano pressoché allo stesso modo: ResNet ottiene una ROC-AUC migliore rispetto alle altre anche in virtù della minore deviazione standard rispetto ad Xception. Tutto sommato si comportano tutte in modo equivalente.

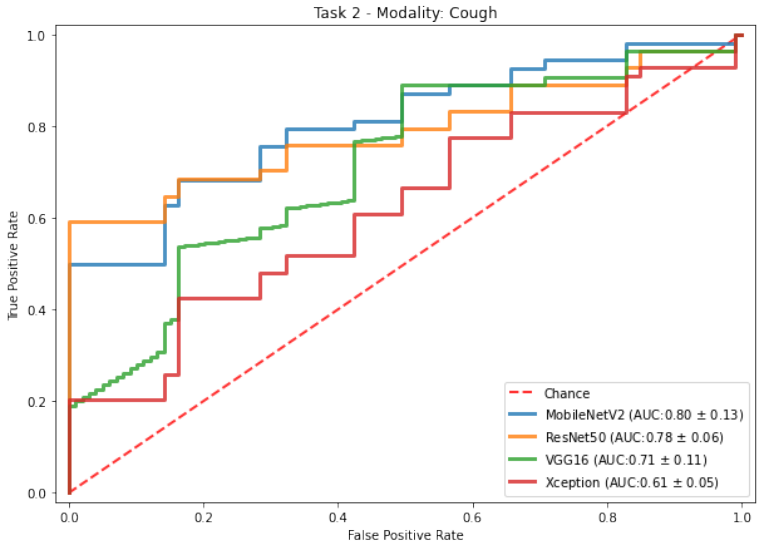


Figura 7.4: Grafico ROC-AUC che illustra i risultati delle CNN sul task 2.

Il **task 2**, che come il task 3 presenta un grafico più "scalinato" a causa della riduzione di elementi del dataset rispetto al task 1, presenta risultati più variabili

rispetto al quest'ultimo. In questo caso si nota una marcata gerarchia tra chi fa bene (MobileNetV2) e chi fa male (Xception).

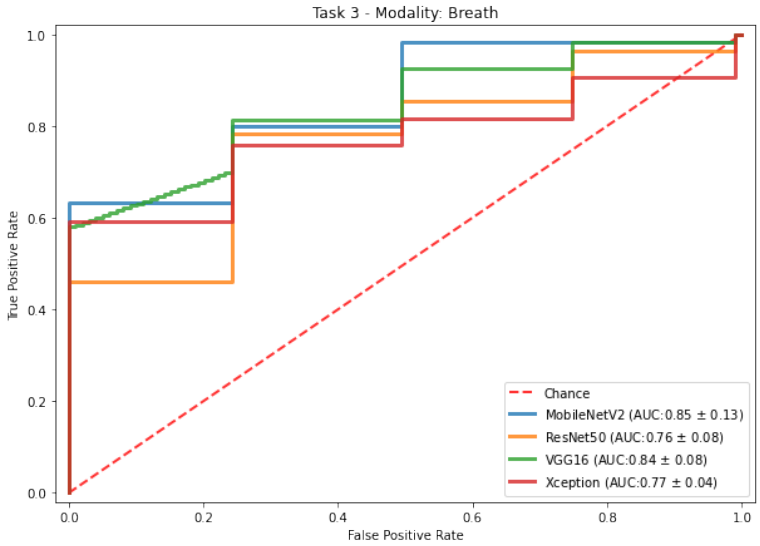


Figura 7.5: Grafico ROC-AUC che illustra i risultati delle CNN sul task 3.

Per il **task 3** si apre una forbice: da una parte due delle reti crescono in ROC-AUC, ma dall'altra parte, le altre due decrescono, quasi dello stesso valore. MobileNetV2 e VGG16 ottengono buoni punteggi, addirittura migliori rispetto al task 1, mentre ResNet50 ed Xception hanno una ROC-AUC che scende attorno al 76-77%.

Le deviazioni standard sono notevoli, ma, come verrà evidenziato dal confronto con i risultati dei ricercatori dell'Università di Cambridge, sono comparabili. Si può notare però che il task 1 presenta delle deviazioni molto piccole, segno che un dataset più ampio può aiutare i modelli ad essere più precisi tra un fold e l'altro nella cross validation.

La tabella 7.2 riporta i valori medi calcolati ragguppando i risultati in base ai task. Si può notare che i migliori valori di ROC-AUC si ottengono per i task 1 e 3 (dove il primo gode di una minore incertezza), mentre per precision e recall il task 3 è quello più affidabile. Questi ultimi valori, sul task 1, sono particolarmente bassi. In specificity la situazione si ribalta ancora, dove il task 1 riesce praticamente a raddoppiare i punteggi del 2 e del 3.

7.3 Analisi sulle CNN

Volendo effettuare una comparazione tra le diverse CNN utilizzate, possiamo prendere come riferimento la tabella 7.3.

MobileNetV2, ovvero il modello da cui si hanno meno pretese vista la sua *lightweight*, è invece in grado di "tenere testa" agli altri modelli ben più pesanti in memoria. Risulta infatti tra i migliori per il task 1 (anche se in questo caso sono tutti equivalenti), ma vince in ROC-AUC nei task 2 e 3 e nella media. Infine, riesce ad avere la migliore precision nella media fra tutti i task.

ResNet50 ha nella recall il suo punto di forza, ottenendo un notevole risultato nel task 2, dove però perde molto in sensitivity.

VGG16, nonostante la sua architettura sia basata su un concetto non più in voga, ottiene discreti risultati, riuscendo a compararsi con le altre. Nel task 3 ottiene la migliore recall.

Xception, nonostante una prestazione equivalente alle altre nel task 1, dove ottiene la migliore recall, fatica molto negli altri due task. L'unico suo punto di forza è la sensitivity nel task 2. Vedendo come migliorano i punteggi tra i task 2-3 e il primo, probabilmente paga più delle altre gli addestramenti su dataset più limitati.

7.4 Comparazione con i risultati di *Brown et al.*

Task	**Modalità**	**Modello**	**Feature**	**Mean ± std**		
				ROC-AUC	**Precision**	**Recall**
1	C + B	Logistic Regression	Handcrafted + VGGish, PCA = 0.95	0.80 (0.07)	0.72 (0.06)	0.69 (0.11)
		ResNet50	Spettrogrammi	0.81 (0.05)	0.66 (0.07)	0.65 (0.07)
		Variazione		**+0.01 (-0.02)**	**-0.06 (+0.01)**	**-0.04 (-0.03)**
2	C	SVM	Handcrafted + VGGish, PCA = 0.9	0.87 (0.14)	0.70 (0.15)	0.90 (0.18)
		MobileNetV2	Spettrogrammi	0.80 (0.14)	0.75 (0.12)	0.83 (0.17)
		Variazione		**-0.07 (0)**	**+0.05 (-0.03)**	**-0.07 (-0.01)**
3	B	SVM	Handcrafted + VGGish, PCA = 0.7	0.88 (0.12)	0.61 (0.22)	0.81 (0.31)
		MobileNetV2	Spettrogrammi	0.85 (0.15)	0.83 (0.10)	0.91 (0.13)
		Variazione		**-0.03 (+0.03)**	**+0.22 (-0.12)**	**+0.10 (-0.18)**

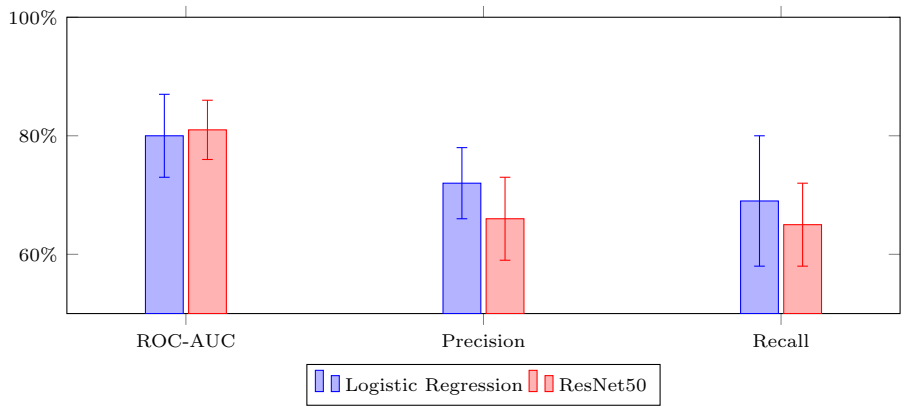
Tabella 7.4: Comparazione dei risultati con quelli di *Brown et al.*

Dal confronto mostrato in tabella 7.4 si può notare che non ci sia grande differenza nei risultati tra l'approccio utilizzato dai ricercatori dell'Università di Cambridge, ovvero estrazione manuale delle feature accompagnata da VGGish e PCA, e l'approccio che prevede l'uso di spettrogrammi con le CNN. Nella tabella sopra menzionata sono confrontati i risultati ottenuti utilizzando l'impostazione che si è rivelata essere migliore rispetto alla ROC-AUC: per *Brown et al.* la soluzione "mista" si è rivelata la migliore in ogni task, mentre per le CNN la scelta migliore è sicuramente MobileNetV2 per i task 2 e 3, mentre per il task 1, nonostante l'identico punteggio in ROC-AUC (con l'identica incerezza), è stata preferita ResNet50 in quanto, complessivamente, fa meglio nelle altre metriche.

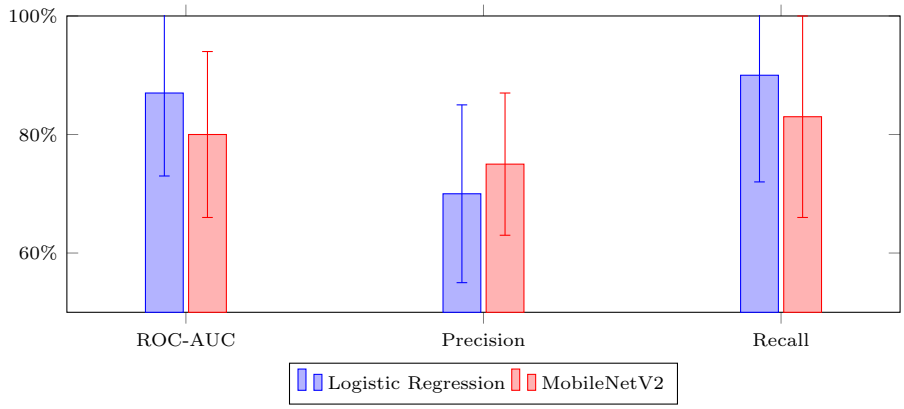
Vediamo in quali casi un approccio risulta migliore dell'altro. Numeri alla mano e osservando il grafico in figura 7.6 non emerge un chiaro "vincitore":

- con l'approccio *Handcrafted + VGGish* ottengono migliori risultati in precision e recall rispetto al task 1, in ROC-AUC e recall rispetto al task 2 e in ROC-AUC rispetto al task 3;
- con l'utilizzo delle CNN si hanno risultati migliori in ROC-AUC rispetto al task 1, in precision rispetto al task 2 e 3 e in recall rispetto al task 3.

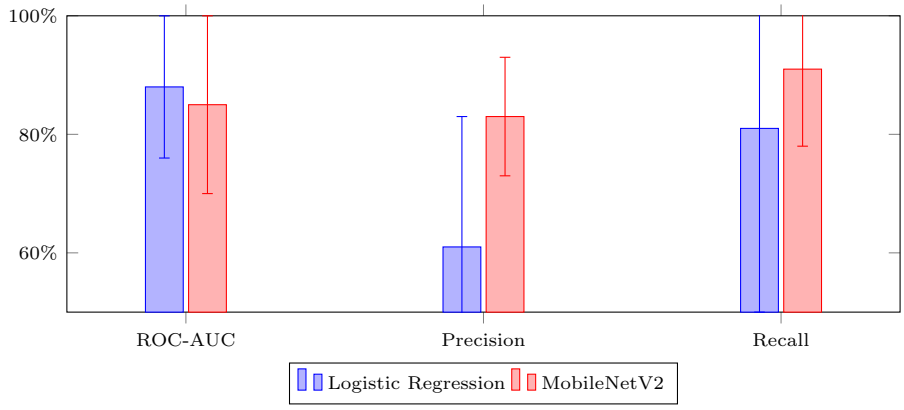
Nel prossimo capitolo è presente qualche considerazione in più.



(a) Task 1



(b) Task 2



(c) Task 3

Figura 7.6: Comparazione grafica dei risultati con quelli di *Brown et al.*

Capitolo 8

Discussione

In questo capitolo sono analizzate alcune questioni inerenti al progetto realizzato, sotto forma di domanda-risposta.

É proprio necessario utilizzare il validation set nella valutazione di una rete neurale? L'obiettivo della valutazione è quantificare la misura di accuratezza, o, più in generale, le performance del modello su dati che non ha mai visto. Questo è il vero scopo del test set. In fase di "consegna", dove deve essere presente il file con la rete addestrata e pronta all'uso, avremo una rete addestrata su tutto il training set e per la quale, ovviamente, saranno stati scelti degli iperparametri. La *nested cross-validation*, però, potrebbe aver effettuato la valutazione scegliendo degli iperparametri diversi ad ogni round, rendendo inaffidabile il valore finale, dato che quelle scelte potrebbero non coincidere con quelle con cui è stato fatto l'addestramento finale. Per questo motivo è stato preferito l'utilizzo di una semplice *cross-validation* [9].

Sono i risultati migliori che si possono ottenere? Sicuramente sono buoni risultati, poiché vanno a pareggiare lo stato dell'arte. Difficile dire se sono i migliori ottenibili: per rispondere a questa domanda dovrebbe essere condotta una ricerca esaustiva che però, richiederebbe molto tempo e risorse.

Quale approccio è migliore da utilizzare, quello basato su combinazione delle feature estratte manualmente e combinate su VGGish o quello basato sulle CNN? Come i risultati illustrano, non ci sono grandi differenze e possiamo dire che le soluzioni si equivalgono.

Se consideriamo però la parte di feature extraction, il discorso cambia: nella riproduzione dell'esperimento dei ricercatori dell'Università di Cambridge è emersa la necessità di avere conoscenza di audio processing e quali sono le feature che si possono estrarre da un file audio. Per l'approccio basato sulle CNN questo non serve, o meglio, bisogna solo sapere cos'è uno spettrogramma; l'inconveniente è che i modelli di *deep learning*, rispetto a modelli di *shallow learning*, richiedono un maggiore tempo di addestramento, proprio perché devono capire quali sono le feature importanti.

Quali potenzialità ha questa metodologia? Grandi potenzialità: non è un caso che i modelli di *deep learning* siano sempre più usati, in qualsiasi ambito e guidino lo sviluppo dell'intelligenza artificiale. In ambito medico, è fondamentale poi avere modelli che siano in grado si assistere gli esperti nell'effettuare le diagnosi, poiché molte volte una patologia può essere identificata solo analizzando in modo dettagliato ogni singolo pixel di un'immagine, operazione che le macchine riescono a fare meglio degli esseri umani. È Interessante notare come modelli addestrati su task di riconoscimento di immagini che una persona può effettuare senza problemi (ImageNet) siano in grado di ottenere performance elevatissime in compiti in cui una persona non riuscirebbe ad effettuare (distinguere spettrogrammi).

Questi modelli possono sostituire un esperto e prendere decisioni al suo posto? Non credo che questo sia uno scenario da augurarsi perché penso che la decisione finale spetti sempre ad un essere umano, soprattutto per questioni di responsabilità. Quello che questi modelli possono rappresentare è una sorta di "consulente" in grado di suggerire una soluzione, ma dovrebbe sempre spettare all'esperto il compito di prendere una decisione finale e di metterla in atto.

L'intelligenza artificiale è uno strumento talmente potente da poter cambiare completamente le nostre vite, ma ci sono temi etici molto importanti da affrontare.

Sviluppi futuri

Sono innumerevoli i modi per portare avanti questa ricerca. Volendole elencare alcuni:

- Analizzare i modelli per capire quali sono le feature che sono importanti nel determinare l'esistenza o meno di sintomi di COVID-19.
- Provare diverse configurazioni della rete e diversi iperparametri.
- Provare altre Convolutional Neural Networks.
- Utilizzare diversi dataset presenti in letteratura.

É stato pubblicato nei mesi scorsi un nuovo articolo dello stesso team di ricerca che ha fornito i dati per questo progetto di tesi [23]. Lo studio è molto simile al precedente e persegue gli stessi obiettivi, ma con la differenza che questa volta si analizza anche la voce. Il dataset esaminato è una versione ampliata di quella utilizzata qui, poiché nel tempo sono stati raccolti più campioni. Sarebbe interessante applicare ai nuovi dati un'analisi basata sull'uso delle CNN con la stessa metodologia illustrata in questo testo.

Capitolo 9

Conclusioni

In questo testo è stata presentata una metodologia alternativa per il riconoscimento di casi di COVID-19 basata sull'uso degli spettrogrammi generati da registrazioni di tosse e respiri.

Dopo una breve introduzione, sono stati affrontati dei concetti preliminari utili a comprendere meglio il problema in oggetto (cap. 2) e poi si è passati all'analisi dello stato dell'arte (cap. 3), con un approfondimento particolare per il lavoro di ricerca che ha fatto da base per questo progetto (cap. 4). Una volta inquadrata la situazione, sono stati illustrati i metodi utilizzati per replicare gli esperimenti del team di ricerca dell'Università di Cambridge (cap. 5) e quelli utilizzati per introdurre l'utilizzo di CNN per come soluzione alternativa. Per finire, sono stati presentati i risultati (cap. 7) e discussi (cap. 8).

Questo approccio, già utilizzato per lo stesso scopo con altri dataset, si è dimostrato una valida alternativa a tecniche basate su *shallow learning* come Linear Regression e SVM su campioni audio raccolti in modalità *crowdsourcing*. I risultati ottenuti su un dataset tendenzialmente piccolo per sfruttare le piene potenzialità dei modelli di *deep learning* fanno pensare che in futuro, su questo tipo di problema, questa soluzione possa diventare ancora più solida e potente.

Glossario

Accuracy Rapporto tra il numero di risultati corretti e il numero totale dei risultati ($\frac{TP+TN}{TP+TN+FP+FN}$). xiii, 13, 15, 16, 48

Callback funzione che viene passata come parametro ad un altra funzione, la cui esecuzione avviene al termine della funzione che l'ha ricevuta in input. 46, 47

Funzione obiettivo Una funzione che mappa un evento o dei valori di variabili su un numero reale che intuitivamente rappresenta un "costo" associato all'evento[1]. 45

Heatmap tecnica di visualizzazione di dati che mostra la magnitudine di un fenomeno come colore in due dimensioni. La variazione di colore può essere di tonalità o intensità[2]. 46

Hertz (Hz) Unità di misura della frequenza di un'onda sonora. 1 Hz equivale ad 1 impulso al secondo. 8

Intelligenza artificiale Disciplina che studia l'utilizzo di tecniche informatiche per costruire modelli che emulano i comportamenti della mente umana. xiii, 13, 14

Machine learning Tecnica di programmazione che lascia al programma il compito di costruire le regole algoritmiche, attraverso un processo di apprendimento automatico, che siano in grado di descrivere i dati che gli vengono mandati in input. 13

Onda sinusoidale Onda descritta matematicamente dalla funzione seno[3]. 8

[1] *Loss function - Wikipedia*. Consultato il 24/02/2021. URL: `https://en.wikipedia.org/wiki/Loss_function`.

[2] *Heat map - Wikipedia*. Consultato il 24/02/2021. URL: `https://en.wikipedia.org/wiki/Heat_map`.

[3] *Onda sinusoidale - Wikipedia*. URL: `https://it.wikipedia.org/wiki/Onda_sinusoidale`.

Precision Rapporto tra il numero elementi rilevanti selezionati e il numero totale degli elementi rilevanti ($\frac{TP}{TP+FN}$). 13, 35

Recall Rapporto tra il numero elementi rilevanti il numero totale dei elementi rilevati ($\frac{TP}{TP+FP}$). 35

Test set Insieme di dati utilizzato in fase di addestramento di un modello di apprendimento automatico per effettuare una valutazione complessiva della bontà del modello. 3, 48

Training set Insieme di dati utilizzato in fase di addestramento di un modello di apprendimento automatico per la fase di addestramento vero e proprio, quando il modello esami i dati per capire come deve modificare i propri parametri per adattarsi ai dati. 3, 48

Validation set Insieme di dati utilizzato in fase di addestramento di un modello di apprendimento automatico per la fase di aggiustamento degli iperparametri. 3, 48

Watt (W) Unità di misura della potenza di un'onda sonora. 10

Acronimi

AUC Area Under Curve. xiv, 5, 22, 25, 27, 48

CNN rete neurale convoluzionale. xiv, xv, 14, 15, 17, 44, 46, 52–56

COVID-19 malattia respiratoria acuta da SARS-CoV-2. ix, 1, 13, 14, 18, 21, 29, 31–33

FPR False Positive Rate. 4

MFCC Mel-Frequency Cepstral Coefficient. 22

MIDI Musical Instrument Digital Interface. 10

PCA Principal component analysis. 11, 25

ROC Receiver Operating Characteristic. 4

ROC-AUC Receiver Operating Characteristic - Area Under Curve. ix, xiii, 3–5, 25–27, 35, 36, 48

SARS-CoV-2 coronavirus 2 da sindrome respiratoria acuta grave. 1

SSH Secure Shell. 39

SVM Support Vector Machine. 21, 36, 37

TPR True Positive Rate. 4

WHO World Health Organization. 13

Riferimenti

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[3] Piyush Bagad et al. *Cough Against COVID: Evidence of COVID-19 Signature in Cough Sounds*. Set. 2020.

[4] Debrup Banerjee et al. "A Deep Transfer Learning Approach for Improved Post-Traumatic Stress Disorder Diagnosis". In: nov. 2017, pp. 11–20. DOI: 10.1109/ICDM.2017.10.

[5] Pedro R. A. S. Bassi e Romis Attux. *A Deep Convolutional Neural Network for COVID-19 Detection Using Chest X-Rays*. 2021. arXiv: 2005.01578 [eess.IV].

[6] Luboš Brabenec et al. "Speech disorders in Parkinson's disease: early diagnostics and effects of medication and brain stimulation". In: *Journal of Neural Transmission* 124 (mar. 2017), pp. 1–32. DOI: 10.1007/s00702-017-1676-0.

[8] Chloë Brown et al. "Exploring Automatic Diagnosis of COVID-19 from Crowdsourced Respiratory Sound Data". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020). DOI: 10.1145/3394486.3412865. URL: http://dx.doi.org/10.1145/3394486.3412865.

[9] Gavin Cawley e Nicola Talbot. "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation". In: *Journal of Machine Learning Research* 11 (lug. 2010), pp. 2079–2107.

[10] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: 1610.02357 [cs.CV].

[13] Stan W. Davis e P. Mermelstein. "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Se". In: 1980.

[16] Monika Doerfler e Thomas Grill. "Inside the Spectrogram: Convolutional Neural Networks in Audio Processing." In: lug. 2017. DOI: 10.1109/SAMPTA.2017.8024472.

[17] Rob Dunne, Tim Morris e Simon Harper. *High accuracy classification of COVID-19 coughs using Mel-frequency cepstral coefficients and a Convolutional Neural Network with a use case for smart home devices*. Ago. 2020. DOI: 10.21203/rs.3.rs-63796/v1.

[18] Frederic B. Fitch. "Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133." In: *Journal of Symbolic Logic* 9.2 (1944), 49–50. DOI: 10.2307/2268029.

[19] Karl Pearson F.R.S. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720. eprint: https://doi.org/10.1080/14786440109462720. URL: https://doi.org/10.1080/14786440109462720.

[20] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics* 36 (apr. 1980), 193–202. DOI: 10.1007/BF00344251.

[22] *Google Colab*. Consultato il 7/03/2021. URL: https://colab.research.google.com/.

[23] Jing Han et al. *Exploring Automatic COVID-19 Diagnosis via voice and symptoms from Crowdsourced Data*. 2021. arXiv: 2102.05225 [cs.SD].

[24] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (set. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[25] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[26] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[28] Shawn Hershey et al. *CNN Architectures for Large-Scale Audio Classification*. 2017. arXiv: 1609.09430 [cs.SD].

[29] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[30] Chiou-Jye Huang et al. *Multiple-Input Deep Convolutional Neural Network Model for COVID-19 Forecasting in China*. Mar. 2020. DOI: 10.1101/2020.03.23.20041608.

[31] Gao Huang et al. *Densely Connected Convolutional Networks*. 2018. arXiv: 1608.06993 [cs.CV].

[32] Yeongtae Hwang et al. *Mel-spectrogram augmentation for sequence to sequence voice conversion*. 2020. arXiv: 2001.01401 [cs.LG].

[33] Ali Imran et al. "AI4COVID-19: AI enabled preliminary diagnosis for COVID-19 from cough samples via an app". In: *Informatics in Medicine Unlocked* 20 (2020), p. 100378. ISSN: 2352-9148. DOI: 10.1016/j.imu.2020.100378. URL: http://dx.doi.org/10.1016/j.imu.2020.100378.

[35] Nikhil Ketkar. "Introduction to Keras". In: ott. 2017, pp. 95–109. ISBN: 978-1-4842-2765-7. DOI: 10.1007/978-1-4842-2766-4_7.

[36] A. Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: 2009.

[37] Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (gen. 2012). DOI: 10.1145/3065386.

[38] Steve Lawrence et al. "Face Recognition: A Convolutional Neural Network Approach". In: *Neural Networks, IEEE Transactions on* 8 (feb. 1997), pp. 98 –113. DOI: 10.1109/72.554195.

[42] J. Mccarthy et al. "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence". In: *AI Magazine* 27 (dic. 2006).

[43] Brian McFee et al. "librosa: Audio and Music Signal Analysis in Python". In: gen. 2015, pp. 18–24. DOI: 10.25080/Majora-7b98e3ed-003.

[44] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series. MIT Press, 2012. ISBN: 9780262018029. URL: https://books.google.it/books?id=NZP6AQAAQBAJ.

[45] Hong-Wei Ng et al. "Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning". In: nov. 2015. ISBN: 9781450339124. DOI: 10.1145/2818346.2830593.

[48] Daniel S. Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Interspeech 2019* (2019). DOI: 10.21437/interspeech.2019-2680. URL: http://dx.doi.org/10.21437/Interspeech.2019-2680.

[50] Kitsuchart Pasupa e Wisuwat Sunhem. "A comparison between shallow and deep architecture classifiers on small dataset". In: ott. 2016, pp. 1–6. DOI: 10.1109/ICITEED.2016.7863293.

[51] Fabian Pedregosa et al. *Scikit-learn: Machine Learning in Python*. 2018. arXiv: 1201.0490 [cs.LG].

[52] Renard Pramono, Anas Imtiaz e Esther Rodriguez-Villegas. "A Cough-Based Algorithm for Automatic Diagnosis of Pertussis". In: *PloS one* 11 (set. 2016), e0162128. DOI: 10.1371/journal.pone.0162128.

[54] C. R. Rodriguez et al. “Deep Learning Audio Spectrograms Processing to the Early COVID-19 Detection”. In: *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*. 2020, pp. 429–434. DOI: 10.1109/CICN49253.2020.9242583.

[55] F. Rosenblatt. *The perceptron - A perceiving and recognizing automaton*. Rapp. tecn. 85-460-1. Ithaca, New York: Cornell Aeronautical Laboratory, 1957.

[56] Sheldon M. Ross. *Introductory Statistics (Fourth Edition)*. Fourth Edition. Oxford: Academic Press, 2017, pp. xxi–xxv. ISBN: 978-0-12-804317-2. URL: https://www.sciencedirect.com/book/9780128043172/introductory-statistics.

[57] Betul Sakar, Gorkem Serbes e C. Okan Sakar. “Analyzing the effectiveness of vocal features in early telediagnosis of Parkinson’s disease”. In: *PLOS ONE* 12 (ago. 2017), e0182428. DOI: 10.1371/journal.pone.0182428.

[58] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[59] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

[60] Alexander Schindler, Thomas Lidy e Andreas Rauber. “Comparing Shallow versus Deep Neural Network Architectures for Automatic Music Genre Classification”. In: nov. 2016.

[61] Karen Simonyan e Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[62] M. Stone. “Cross-Validatory Choice and Assessment of Statistical Predictions”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2 (1974), pp. 111–133. DOI: 10.1111/j.2517-6161.1974.tb00994.x. URL: https://doi.org/10.1111%2Fj.2517-6161.1974.tb00994.x.

[63] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2015. URL: http://arxiv.org/abs/1409.4842.

[64] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV].

[65] *Tensorflow Models: VGGish*. Consultato il 7/03/2021. URL: https://github.com/tensorflow/models/tree/master/research/audioset/vggish.

[67] A. M. TURING. "I.—COMPUTING MACHINERY AND INTELLIGENCE". In: *Mind* LIX.236 (ott. 1950), pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL: https://doi.org/10.1093/mind/LIX.236.433.

[69] Joachim Wagner, Jennifer Foster e Josef Genabith. "A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors". In: *Wagner, Joachim and Foster, Jennifer and van Genabith, Josef (2007) A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. In: EMNLP-CoNLL 2007 - Joint Meeting of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning, 28-30 June 2007, Prague, Czech Republic.* (gen. 2007).

[70] Linda Wang e Alexander Wong. *COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest X-Ray Images*. 2020. arXiv: 2003.09871 [eess.IV].

[71] Shengyun Wei et al. "A Comparison on Data Augmentation Methods Based on Deep Learning for Audio Classification". In: *Journal of Physics: Conference Series* 1453 (2020), p. 012085. DOI: 10.1088/1742-6596/1453/1/012085. URL: https://doi.org/10.1088/1742-6596/1453/1/012085.

[72] "Whitepaper GPU-Based Deep Learning Inference : A Performance and Power Analysis". In: 2015.

[73] Saining Xie et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: 1611.05431 [cs.CV].

Ringraziamenti

Cara Università di Bologna,
mai dire "mai", ma sembra che siamo davvero ai titoli di coda. Sono contentissimo di aver raggiunto questo traguardo, ma insieme a questa felicità provo anche un pizzico di nostalgia perché è la fine di un capitolo importante della mia vita. Tuttavia sento che è giustamente giunto il momento di guardare altrove, con un bagaglio personale arricchito da tutto quello che ho potuto vivere in questi anni.

Prima di tutto vorrei ringraziare il Prof. Maurizio Gabbrielli e il Dott. Stefano Pio Zingaro per avermi dato la possibilità di lavorare a questo progetto entusiasmante. Ringrazio il Dott. Zingaro anche per avermi seguito durante lo sviluppo di questo progetto di tesi dandomi numerosi consigli. Non possono non menzionare Antonio Lategano, che in questo ultimo periodo è stato il mio punto di riferimento quando ho avuto bisogno di informazioni.

Rivolgo un grazie a Valerio Velardo perché per merito della sua serie su YouTube[4] sono riuscito ad entrare nell'ostico argomento dell'elaborazione dei segnali audio, ma in particolare a tutti quelli che hanno scritto guide e tutorial sul machine learning perché mi sono state di grande aiuto. Grazie all'azienda Google che, mettendo a disposizione gratuitamente i suoi servizi come TensorFlow e Colab, rende accessibile l'intelligenza artificiale e il machine learning a tutti.

Per me questo lavoro ha un grande valore perché sancisce la conclusione di un percorso iniziato sette anni fa, quando con coraggio e sicuramente un bel po' di sana incoscienza decisi di fare un salto che, col senno di poi, è stata la decisione migliore che potessi prendere. È stato un percorso con tanti alti e bassi. Mi sono reso conto che probabilmente il mondo accademico non rappresenta proprio il mio ideale: studiare mesi ed essere giudicati solo per quello che si riesce a fare in un giorno o in qualche ora non fa per me, ma è un prezzo che sono disposto volentieri a pagare per acquisire la conoscenza che l'università può fornirmi. Devo ammettere però che lavorare su argomenti che hanno le potenzialità per rappresentare il futuro, come le tecnologie utilizzate in questo lavoro di tesi, è una cosa molto bella e entusiasmante.

Adesso i ringraziamenti più sentiti.

[4] Valerio Velardo. *Audio Signal Processing for Machine Learning*. Consultato il 1/02/2021. 2020. URL: `https://www.youtube.com/watch?v=iCwMQJnKk2c&list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0`.

Ringrazio mia mamma, che mi ha permesso di intraprendere questo percorso, mi ha supportato e soprattutto mi ha criticato quando serviva, spronandomi a fare di meglio e ha sempre cercato di non farmi mancare nulla.

Ringrazio mia nonna, grazie alla quale ho capito che certe cose vanno accettate anche se non vanno come si vorrebbe e che molte volte è inutile prendersela, quando invece si può apprezzare meglio quello che si ha.

Ringrazio Dario, mio caro amico da una vita, che da anni con quel suo talento che ha riesce sempre a farmi ridere e vedere le cose con leggerezza, ma soprattutto è sempre pronto a farmi da complice in qualsiasi occasione.

Ringrazio Camilla, un'amica dal cuore grande che purtroppo ho imparato a conoscere veramente solo di recente, ma che si fa sempre trovare pronta nel momento del bisogno, oltre a mettermi a disposizione quella sua straordinaria capacità che ha di capirmi ancor prima che io parli.

Ringrazio quelli che sono stati i miei compagni di corso, i miei coinquilini, in particolare Francesco e Antonio, e tutte le persone con cui ho avuto a che fare durante questo grande percorso, perché, nel bene e nel male, in qualche modo ho tratto da loro gli insegnamenti che mi hanno permesso di diventare la persona che sono.

Un grazie particolare alla città di Bologna perché per me, che venivo dalla piccola Siena, è stata un "nuovo mondo", dove ho potuto mettermi alla prova e dove ho capito davvero cosa posso fare e quali sono i miei limiti, iniziando a riuscire a vedere le cose da punti di vista diversi. Qui ho trovato persone bravissime a cui mi sono potuto ispirare per impegnarmi e migliorarmi; dove ho trovato persone che se hanno un obiettivo non si preoccupano troppo delle numerose difficoltà che possono incontrare cercando di raggiungerlo, ma si rimboccano le maniche e iniziano a lavorare duramente. Qui ho trovato persone che mi hanno fatto capire l'importanza della gentilezza e dell'accoglienza.

Grazie ai ragazzi di Passione Astronomia[5], che in questi ultimi mesi sono stati una grandissima fonte di ispirazione per me e che mi hanno tenuto compagnia con le loro innumerevoli dirette serali, grazie alle quali ho scoperto che nell'Universo esistono delle cose meravigliose, che, purtroppo, per anni ho ignorato. Riuscire a percepire le loro emozioni quando parlano di stelle, pianeti e galassie è una delle cose più affascinanti che abbia mai provato. Il loro impegno nella divulgazione scientifica è lodevole e molto importante per una materia poco compresa come l'astronomia. Da loro ho anche imparato che la scienza può essere una forma d'arte e il metodo scientifico è una delle cose più preziose che abbiamo a disposizione come umanità. Ma soprattutto, osservare il cielo serve a vedere se stessi da una diversa prospettiva.

Devo rivolgere lo stesso ringraziamento al Prof. Alessandro Marchini dell'Osservatorio Astronomico dell'Università di Siena perché anche lui ha giocato un ruolo importante nella mia "scoperta del cielo".

[5] *Passione Astronomia*. Consultato il 2/03/2021. URL: `https://www.passioneastronomia.it/`.

Voglio concludere riprendendo quello che ho scritto nella dedica all'inizio. Questo lavoro, che ho portato avanti con un impegno, ma anche un piacere, mai avuti prima d'ora, è dedicato a tutte le persone che mi vogliono bene. In qualche modo, c'è il loro contributo in tutto questo.

A beautiful adventure has come to the end,
and a new exciting one is going to begin!

Licenza

[6] *Creative Commons — Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale — CC BY-NC-SA 4.0.* Consultato il 7/03/2021. URL: `https://creativecommons.org/licenses/by-nc-sa/4.0/deed.it`.

Appendici

Appendice A

Tabella degli esperimenti

Di seguito è riportata la tabella che contiene i risultati di tutti gli esperimenti effettuati, i quali hanno richiesto un **tempo di calcolo totale di 23 ore e 6 minuti**. Purtroppo, vista il numero di colonne, si è reso necessario spezzare la tabella in più parti, ma ad ogni riga è stato assegnato un identificativo per renderne possibile la lettura.

ID	Task	Modality	Model	Augmentation	Image shape	Neurons	Depth	Shape	Epochs
0	3	B	MobileNetV2	O,N,PS,TSH,TST	224x224	256	2	triangle	100
1	3	B	VGG16	O,N,PS,TSH,TST	224x224	256	2	triangle	100
2	1	CB	Xception	O,N,PS,TSH,TST	224x224	256	2	triangle	100
3	1	CB	ResNet50	O,N,PS,TSH,TST	224x224	256	2	triangle	100
4	2	C	MobileNetV2	O,N,PS,TSH,TST	224x224	256	2	triangle	100
5	1	CB	VGG16	O,N,PS,TSH,TST	224x224	256	2	triangle	100
6	1	CB	MobileNetV2	O,N,PS,TSH,TST	224x224	256	2	triangle	100
7	2	C	ResNet50	O,N,PS,TSH,TST	224x224	256	2	triangle	200
8	2	C	ResNet50	O,N,PS,TSH,TST	224x224	256	2	triangle	300
9	3	B	Xception	O,N,PS,TSH,TST	299x299	256	2	triangle	100
10	3	B	ResNet50	O,N,PS,TSH,TST	224x224	256	2	triangle	100
11	2	C	ResNet50	O,N,PS,TSH,TST	224x224	256	2	triangle	100
12	1	C	ResNet50	O	224x224	256	2	triangle	100
13	1	C	ResNet50	O	160x160	512	2	triangle	200
14	1	C	ResNet50	O	224x224	2048	1	triangle	200
15	1	C	ResNet50	O	224x224	256	2	triangle	100
16	1	C	ResNet50	O	160x160	512	2	triangle	200
17	2	C	VGG16	O,N,PS,TSH,TST	224x224	256	2	triangle	300
18	1	C	ResNet50	O	160x160	256	2	triangle	100
19	1	C	MobileNetV2	O	224x224	256	2	triangle	200
20	1	C	ResNet50	O	224x224	256	1	triangle	50
21	1	C	ResNet50	O	160x160	2048	1	triangle	200
22	1	C	ResNet50	O	160x160	512	1	triangle	200
23	1	C	ResNet50	O	160x160	512	2	triangle	100
24	1	C	Xception	O	299x299	256	2	triangle	100

ID	Task	Modality	Model	Augmentation	Image shape	Neurons	Depth	Shape	Epochs
25	1	C	MobileNetV2	O	224x224	256	2	triangle	200
26	1	C	ResNet50	O	160x160	1024	1	triangle	200
27	1	C	MobileNetV2	O	224x224	256	2	triangle	200
28	1	C	MobileNetV2	O	224x224	256	2	triangle	200
29	1	C	MobileNetV2	O	224x224	256	1	triangle	200
30	1	C	Xception	O	299x299	256	2	triangle	200
31	1	C	MobileNetV2	O	224x224	256	1	triangle	50
32	1	C	ResNet50	O	224x224	256	2	triangle	100
33	1	C	MobileNetV2	O	224x224	512	1	triangle	50
34	1	C	MobileNetV2	O	224x224	256	1	triangle	100
35	1	C	MobileNetV2	O	224x224	64	3	triangle	100
36	1	C	MobileNetV2	O	224x224	256	2	triangle	200
37	1	C	MobileNetV2	O	224x224	256	2	triangle	200
38	1	C	MobileNetV2	O	224x224	256	2	triangle	200
39	1	C	MobileNetV2	O	224x224	256	2	triangle	200
40	1	C	MobileNetV2	O	224x224	256	2	triangle	200
41	1	C	MobileNetV2	O	224x224	256	2	triangle	200
42	1	C	MobileNetV2	O	224x224	512	1	triangle	100
43	1	C	MobileNetV2	O	224x224	128	1	triangle	50
44	1	C	MobileNetV2	O	224x224	512	1	triangle	200
45	1	C	MobileNetV2	O	224x224	256	2	triangle	500
46	1	C	MobileNetV2	O	224x224	256	2	triangle	200
47	1	C	MobileNetV2	O	224x224	256	2	triangle	200
48	1	C	MobileNetV2	O	224x224	1024	1	triangle	50
49	1	C	ResNet50	O	224x224	256	2	triangle	100
50	1	C	MobileNetV2	O	224x224	256	2	triangle	200
51	1	C	MobileNetV2	O	224x224	256	2	triangle	200
52	1	C	MobileNetV2	O	224x224	1024	1	triangle	100
53	1	C	MobileNetV2	O	224x224	256	3	triangle	200
54	1	C	MobileNetV2	O	224x224	1024	1	triangle	200
55	2	C	Xception	O,N,PS,TSH,TST	299x299	256	2	triangle	100
56	2	C	Xception	O,N,PS,TSH,TST	299x299	256	2	triangle	300
57	1	C	MobileNetV2	O	224x224	64	3	triangle	10
58	2	C	Xception	O,N,PS,TSH,TST	299x299	256	2	triangle	100
59	2	C	Xception	O,N,PS,TSH,TST	224x224	256	2	triangle	100

ID	Learning rate	Batch size	Dropout	Aug. shift	Fine tuning	Class weights	Data mixing
0	0.001	32	0.2	0.2	0.0	1	25.0
1	0.001	32	0.2	0.2	0.0	1	79.0
2	0.001	32	0.2	0.2	0.0	1	19.0
3	0.001	32	0.2	0.2	0.0	1	45.0
4	0.001	32	0.2	0.2	0.0	1	84.0

ID	Learning rate	Batch size	Dropout	Aug. shift	Fine tuning	Class weights	Data mixing
5	0.001	32	0.2	0.2	0.0	1	90.0
6	0.001	32	0.2	0.2	0.0	1	85.0
7	0.001	32	0.2	0.2	0.0	1	75.0
8	0.001	32	0.2	0.2	0.0	1	75.0
9	0.001	32	0.2	0.2	0.0	1	84.0
10	0.001	32	0.2	0.2	0.0	1	98.0
11	0.001	32	0.2	0.2	0.0	1	75.0
12	0.001	32	0.2	0.2	0.0	0	
13	0.001	32	0.2	0.2	0.0	1	
14	0.0001	32	0.2	0.2	0.0	1	
15	0.001	32	0.2	0.2	0.0	1	
16	0.0001	32	0.2	0.2	0.0	1	
17	0.001	32	0.2	0.2	0.0	1	11.0
18	0.001	32	0.2	0.2	0.0	1	
19	0.001	32	0.2	0.2	0.0	0	
20	0.001	32	0.2	0.2	0.0	0	
21	0.0001	32	0.2	0.2	0.0	1	
22	0.0001	32	0.2	0.2	0.0	1	
23	0.001	32	0.2	0.2	0.0	1	
24	0.001	32	0.2	0.2	0.0	1	
25	0.001	32	0.2	0.5	0.0	0	
26	0.0001	32	0.2	0.2	0.0	1	
27	1e-05	64	0.2	0.5	0.0	0	
28	1e-05	64	0.2	0.5	0.0	0	
29	0.001	32	0.2	0.2	0.0	0	
30	0.0001	32	0.2	0.2	0.0	1	
31	0.01	32	0.2	0.2	0.0	0	
32	0.001	32	0.2	0.5	0.0	0	
33	0.01	32	0.2	0.2	0.0	0	
34	0.001	32	0.2	0.2	0.0	0	
35	0.001	32	0.2	0.2	0.0	0	
36	0.001	64	0.2	0.5	0.0	0	
37	0.001	64	0.2	0.5	0.0	0	
38	0.001	16	0.2	0.5	0.0	0	
39	0.001	16	0.2	0.5	0.0	0	
40	0.0001	16	0.2	0.5	0.0	0	
41	0.0001	16	0.2	0.5	0.0	0	
42	0.001	32	0.2	0.2	0.0	0	
43	0.01	32	0.2	0.2	0.0	0	
44	0.001	32	0.2	0.2	0.0	0	
45	1e-05	64	0.2	0.2	0.0	0	
46	0.0001	64	0.2	0.5	0.0	0	
47	0.0001	64	0.2	0.5	0.0	0	

ID	Learning rate	Batch size	Dropout	Aug. shift	Fine tuning	Class weights	Data mixing
48	0.01	32	0.2	0.2	0.0	0	
49	0.001	32	0.2	0.2	0.1	1	
50	1e-05	16	0.2	0.5	0.0	0	
51	1e-05	16	0.2	0.5	0.0	0	
52	0.001	32	0.2	0.2	0.0	0	
53	0.001	32	0.2	0.2	0.0	0	
54	0.001	32	0.2	0.2	0.0	0	
55	0.001	32	0.2	0.2	0.0	1	53.0
56	0.001	32	0.2	0.2	0.0	1	53.0
57	0.001	32	0.2	0.2	0.0	0	
58	0.001	32	0.2	0.2	0.0	1	84.0
59	0.001	32	0.2	0.2	0.0	1	74.0

ID	Shuffling seed	Accuracy (mean)	Accuracy (std)	ROC-AUC (mean)	ROC-AUC (std)
0	25.0	0.7847619175910949	0.0848100401292824	0.8486363636363636	0.14689978030802311
1	79.0	0.7438095331192016	0.09714751656049722	0.8395454545454545	0.09403160186214167
2	19.0	0.7404892563819885	0.045711558348054376	0.8143088112437956	0.05052111921241372
3	45.0	0.7756588935852051	0.04086262281362661	0.8138854840181686	0.050102042380267026
4	84.0	0.696895432472229	0.1277413113797464	0.8035064935064934	0.14173261985052688
5	90.0	0.7698933124542237	0.011282148851053142	0.8009643682034217	0.020958659924078686
6	85.0	0.7415673971176148	0.040230423195120366	0.8002564885603365	0.01679241462120863
7	75.0	0.6504085063934326	0.0751858253028191	0.785064935064935	0.0643443550228993
8	75.0	0.7128268003463745	0.12318801543598441	0.7783116883116883	0.0980183041005953
9	84.0	0.7019047737121582	0.04385091954544907	0.7677272727272728	0.044256950707774884
10	98.0	0.7857142925262451	0.09712416446339611	0.765	0.0899896688285259
11	75.0	0.6976307272911072	0.07257482728288588	0.758138528138528	0.07163285754887157
12		0.735282588005066	0.07766136459679822	0.7370965879045949	0.15598073345864094
13		0.6990472197532653	0.05512748533784731	0.7290593081183379	0.16118405297515026
14		0.7038997054100037	0.058556443525696535	0.7171445965879045	0.13983171755795631
15		0.7173584938049317	0.04110884361076522	0.7138962873284908	0.1335068448186306
16		0.6624716877937317	0.05758477628948772	0.7119744788622637	0.15642958880039334
17	11.0	0.6873366117477417	0.08478426929824663	0.7106926406926407	0.11845143910838453
18		0.6762154698371887	0.07795387772646345	0.7085065124822576	0.13042550479291706
19		0.7242271661758423	0.049395047207772366	0.7074423895800284	0.1360123836838659
20		0.7037702560424804	0.0660533810431335	0.7065913139072111	0.13921389156605418
21		0.6989177823066711	0.05964514563752571	0.7064656563969831	0.15677605612213774
22		0.6674333453178406	0.02511145143687881	0.7047626701177256	0.14294857264527985
23		0.6690832138061523	0.08322027222063053	0.7031801230135538	0.14282458447108132
24		0.6853028535842896	0.09225746474081294	0.7008439787370238	0.16222607745910306
25		0.6924327731132507	0.059078504648325335	0.7001994099802398	0.12743334909693052
26		0.6695534348487854	0.0573047273052681	0.6976125073056693	0.14289967329805642
27		0.7037197947502136	0.045416485953204536	0.6960099913723526	0.07589979496419147

ID	Shuffling seed	Accuracy (mean)	Accuracy (std)	ROC-AUC (mean)	ROC-AUC (std)
28		0.7037197947502136	0.045416485953204536	0.6960099913723526	0.07589979496419147
29		0.6924072265625	0.056151689409089885	0.6930534218362974	0.12448823603935018
30		0.7197060704231262	0.06344248860049913	0.6917970554674235	0.14818686826542882
31		0.6899789214134217	0.0478613161931189	0.690165177701706	0.0965580190719446
32		0.7015486121177673	0.0728435630669776	0.6899156717040996	0.15873388521720835
33		0.6948361039161682	0.059450512564278336	0.6855740864434611	0.11229930828905127
34		0.6945510864257812	0.039355853514623625	0.6831965433748017	0.13159251982968626
35		0.7034876108169555	0.05873157549127087	0.6827854777211878	0.12501901462023726
36		0.6695748686790466	0.055573026058177474	0.6811423979293646	0.1308622259414782
37		0.6695748686790466	0.055573026058177474	0.6811423979293646	0.1308622259414782
38		0.67879638671875	0.03700204898264012	0.6799838579499596	0.1026234245150358
39		0.67879638671875	0.03700204898264012	0.6799838579499596	0.1026234245150358
40		0.7037702679634095	0.06192395651510705	0.6788809106342713	0.10154274392471614
41		0.7037702679634095	0.06192395651510705	0.6788809106342713	0.10154274392471614
42		0.701577091217041	0.039074489991718336	0.6787666638835546	0.1100599712298386
43		0.6651083588600158	0.03152219091105889	0.6770560936238903	0.1162259729625023
44		0.6536895394325256	0.0554018538157976	0.6713089671871086	0.11038101173479438
45		0.6719509243965149	0.04639329920856828	0.6700895466310428	0.08274702960741702
46		0.6788219213485718	0.047307551140908316	0.6673402493668421	0.1212190406408013
47		0.6788219213485718	0.047307551140908316	0.6673402493668421	0.1212190406408013
48		0.6831862807273865	0.04810524258322575	0.6648453285463806	0.10440190479401022
49		0.6422748446464539	0.03870229759109205	0.6636427458183741	0.17886494110944412
50		0.680835771560669	0.04844249764421475	0.6601653168573097	0.09920749430011618
51		0.680835771560669	0.04844249764421475	0.6601653168573097	0.09920749430011618
52		0.6630654692649841	0.05620178236542768	0.6573861707160947	0.11221935114016676
53		0.6809123873710632	0.058182299200813886	0.65599670201219	0.1251151333753899
54		0.67184818983078	0.06105149926623073	0.6382278533856558	0.1608572941620787
55	53.0	0.6170751810073852	0.08111905801936241	0.6151948051948052	0.056139191772838645
56	53.0	0.6165032744407654	0.08147763590359172	0.606017316017316	0.07729582397009443
57		0.6539222955703735	0.049676753037451504	0.5921795942222593	0.0891451042334943
58	84.0	0.5617647111415863	0.11410097013953872	0.5794372294372294	0.179594341632289
59	74.0	0.5665849804878235	0.0873390502875941	0.559047619047619	0.08262877511415212

ID	Precision (mean)	Precision (std)	Recall (mean)	Recall (std)
0	0.8330402970314026	0.10163709542537039	0.9054545521736145	0.1319904733068048
1	0.7863247990608215	0.03364960429352295	0.8909090995788574	0.14937888558514073
2	0.588504558801651	0.06327275856828957	0.6669172763824462	0.12167805456655818
3	0.6567569255828858	0.07393764798797126	0.645488727092743	0.0733971087707163
4	0.747802197933197	0.12173871249344902	0.8290909171104431	0.17420420096304068
5	0.6365088224411011	0.022040318533222362	0.6630325794219971	0.014828270416346523
6	0.5997272849082946	0.07939239487049117	0.6456140279769897	0.07839419507804768
7	0.6691774964332581	0.038399522669542545	0.8672727346420288	0.11347261925617863

ID	Precision (mean)	Precision (std)	Recall (mean)	Recall (std)
8	0.704322361946106	0.07690318489144311	0.9272727370262146	0.11853094263996161
9	0.785787558555603	0.04634994981941831	0.829090929031372	0.16191008111324734
10	0.8421212077140808	0.06513742202945222	0.8727272748947144	0.08131156857344954
11	0.7153791427612305	0.04546424995805423	0.8690909147262573	0.1539561000088877
12	0.562900447845459	0.1733485831828192	0.4093596085906029	0.2784772199032528
13	0.5578104615211487	0.13060124770289164	0.5022167474031448	0.2678240370568323
14	0.4577540934085846	0.2372620469861789	0.5073891639709472	0.3512672777302001
15	0.5445503532886505	0.07643081452729378	0.5509852081537246	0.2164106679660989
16	0.4733451426029205	0.12820705052141113	0.5719211846590042	0.3044663772812065
17	0.7415656685829163	0.10549929600447809	0.814545476436615	0.0912720060896854
18	0.4823481142520905	0.13055374880811066	0.6295566588640213	0.28525889306701313
19	0.5670477509498596	0.10435793649480812	0.4386699542403221	0.2247911769312262
20	0.5100453346967697	0.16923111054845127	0.5435960605740547	0.2676599976184665
21	0.5263335168361664	0.0885260864605361	0.5238916136324405	0.2908642709864848
22	0.5064425945281983	0.05214982083948825	0.5655172511935234	0.28450930950312114
23	0.5224608540534973	0.14899451916042866	0.5726601094007492	0.24766130074196965
24	0.5018206298351288	0.14489856290321074	0.552216750383377	0.234472663311038
25	0.4942576289176941	0.10583091377814996	0.44507390111684797	0.26886648919145817
26	0.4735327661037445	0.10125784528109563	0.5519704371690749	0.28021439161135026
27	0.5386003017425537	0.09837365749419463	0.3605911314487457	0.19952015169209228
28	0.5386003017425537	0.09837365749419463	0.3605911314487457	0.19952015169209228
29	0.5097222328186035	0.0965852220401996	0.4660098552703858	0.22469292004263586
30	0.5380952417850494	0.12528409982283875	0.5226601079106331	0.2839770206430449
31	0.483966863155365	0.1000973227081886	0.466502471268177	0.2367947081646887
32	0.5233647406101227	0.1802497388609426	0.38004926815629	0.31133005354624194
33	0.5286972343921661	0.09927468469747924	0.44679802656173706	0.15068054425571042
34	0.4900365769863129	0.08943367805747031	0.40935960561037066	0.25050677416652434
35	0.5004485130310059	0.13566953935584392	0.4088669911026955	0.2604358014739227
36	0.4666296064853668	0.09993335086722482	0.4736453235149384	0.22158810963794506
37	0.4666296064853668	0.09993335086722482	0.4736453235149384	0.22158810963794506
38	0.4769697070121765	0.08271011720951793	0.3389162629842758	0.1851512455713825
39	0.4769697070121765	0.08271011720951793	0.3389162629842758	0.1851512455713825
40	0.4908504515886306	0.16193769433471822	0.4093596048653126	0.2936375857396953
41	0.4908504515886306	0.16193769433471822	0.4093596048653126	0.2936375857396953
42	0.5266391515731812	0.07036460470424845	0.472167482972145	0.27813607592085243
43	0.3942307770252228	0.2014493894341874	0.4450738966464997	0.2601871447813249
44	0.4171551644802093	0.12400653613841316	0.3667487695813179	0.22827795514960905
45	0.44676819443702703	0.1176592237677058	0.3465517207980156	0.2000836972268579
46	0.4833489000797272	0.10672195424407947	0.3674876898527145	0.1801814618157092
47	0.4833489000797272	0.10672195424407947	0.3674876898527145	0.1801814618157092
48	0.4822222173213959	0.09783835790879047	0.3967980295419693	0.17881907985816872
49	0.43055937886238105	0.09892853188260667	0.4874384164810181	0.3067351945064685
50	0.4519065707921982	0.13253001210268808	0.3815271005034447	0.2367972728035785

ID	Precision (mean)	Precision (std)	Recall (mean)	Recall (std)
51	0.4519065707921982	0.13253001210268808	0.3815271005034447	0.2367972728035785
52	0.4690836429595947	0.07159754739030191	0.4022167533636093	0.2211509941371655
53	0.4750792682170868	0.11999884873525253	0.3667487725615501	0.2616053455531476
54	0.42400933504104615	0.11644377690412885	0.3608374424278736	0.2898440289350657
55	0.7015401363372803	0.1038480613640152	0.7054545521736145	0.1462478462147422
56	0.6979853510856628	0.10014060429670651	0.7436363637447357	0.1727751002490437
57	0.4248366087675095	0.3384333683144415	0.11330049484968185	0.06887309196405003
58	0.6391666650772094	0.09389341689127126	0.723636370897293	0.17971511350664052
59	0.6277722358703614	0.05373450446634665	0.7563636422157287	0.09586490475096672

ID	Specificity (mean)	Specificity (std)	Tempo di elaborazione (s)
0	0.45	0.37080992435478316	486.22866320610046
1	0.35	0.13693063937629152	742.2974278926849
2	0.7753081232492998	0.06268621092475059	10486.776430606842
3	0.8372829131652662	0.05252496717106554	10028.94953250885
4	0.4714285714285714	0.36746974515658043	310.19478154182434
5	0.8204341736694676	0.016796244399344896	7957.910123586655
6	0.7870168067226891	0.07714181586964998	4628.796527385712
7	0.28095238095238095	0.06818010030131597	1367.9765560626984
8	0.3476190476190476	0.1342654493321113	2595.7352001667027
9	0.35	0.2850438562747845	1543.2879309654236
10	0.55	0.2091650066335189	814.8870062828064
11	0.3952380952380952	0.24140555915498238	707.0110716819763
12	0.8893220338983051	0.019675622862279918	716.8623471260071
13	0.792316384180791	0.1478412871536184	821.2779958248137
14	0.7959322033898306	0.16827345739023694	1421.5832545757294
15	0.7955932203389829	0.05826625738227634	723.3580899238586
16	0.7048022598870056	0.14121536454022035	824.1006002426147
17	0.45714285714285713	0.35848952457362676	2537.19144654274
18	0.6985875706214689	0.15228079686197174	423.24013781547546
19	0.8594350282485876	0.05580525455161794	602.7812464237212
20	0.7788700564971751	0.12456152467904374	379.6008648872376
21	0.782316384180791	0.11831290883316785	850.673318862915
22	0.7157627118644069	0.15874544059439552	813.5585451126099
23	0.7140677966101696	0.13240054833593126	421.5295307636261
24	0.7485310734463276	0.10365259437712224	1635.7164068222046
25	0.8093220338983051	0.09098711754280102	595.7979588508606
26	0.7255932203389831	0.1447106075851122	839.3111574649811
27	0.8661016949152541	0.05336725675890081	594.2443022727966
28	0.8661016949152541	0.05336725675890081	594.2443022727966
29	0.7989265536723165	0.08451652611663421	616.3463780879973
30	0.8125423728813559	0.11240774285762443	3264.2312908172607

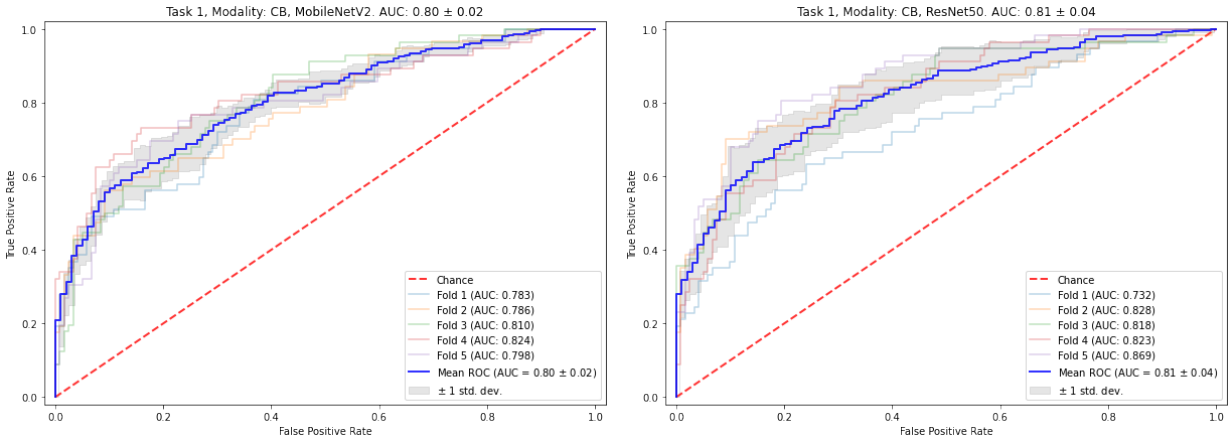
ID	Specificity (mean)	Specificity (std)	Tempo di elaborazione (s)
31	0.7956497175141243	0.07622505405434485	172.16670489311218
32	0.8527683615819208	0.14640013039816407	715.4193544387817
33	0.8124858757062146	0.07785023733363347	176.65826559066772
34	0.8293785310734464	0.07190849835978558	313.92528462409973
35	0.8425988700564974	0.05379510189567839	312.1147320270538
36	0.7622598870056498	0.08163530969782773	592.0731177330018
37	0.7622598870056498	0.08163530969782773	592.0731177330018
38	0.8392655367231638	0.06492112987427671	650.1662902832031
39	0.8392655367231638	0.06492112987427671	650.1662902832031
40	0.8428813559322034	0.10374678278191447	663.1299116611482
41	0.8428813559322034	0.10374678278191447	663.1299116611482
42	0.8092655367231638	0.11719145727568868	324.98670768737793
43	0.7690960451977402	0.09076576702125998	168.1163265705109
44	0.7890395480225989	0.09804081629507183	617.7471323013307
45	0.8259322033898304	0.06390662176691313	1416.9345729351044
46	0.8258757062146893	0.06735960854228902	592.4128131866455
47	0.8258757062146893	0.06735960854228902	592.4128131866455
48	0.8189830508474577	0.027686728753839884	177.62919449806213
49	0.7155367231638418	0.1537606716751027	711.9032378196716
50	0.8224293785310735	0.04823568543670071	678.1753585338593
51	0.8224293785310735	0.04823568543670071	678.1753585338593
52	0.7858757062146893	0.1382696079088122	323.6151685714722
53	0.8290960451977402	0.0949398004310916	614.0630292892456
54	0.8193785310734463	0.08085367563055132	617.7288193702698
55	0.4666666666666666	0.25231355793788485	1786.3691701889038
56	0.4095238095238095	0.29228554360807446	5294.4086124897
57	0.9094350282485876	0.08843776968883911	63.84332847595215
58	0.29047619047619044	0.2975714247612951	1786.719392299652
59	0.24285714285714283	0.11591106137750555	893.375247001648

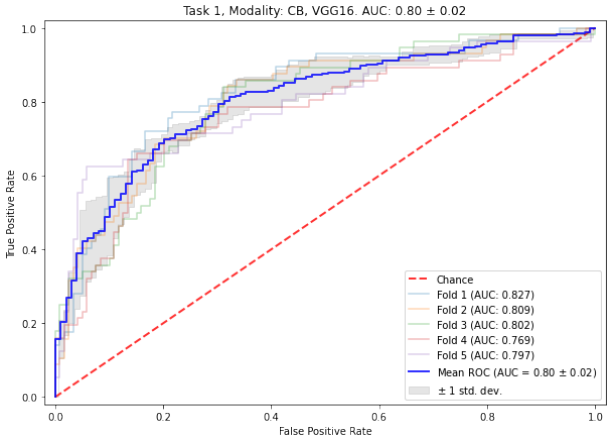
Appendice B

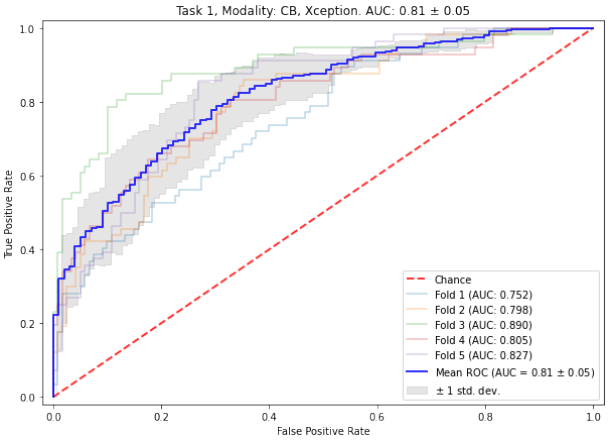
Grafici degli esperimenti

B.1 Curve ROC-AUC

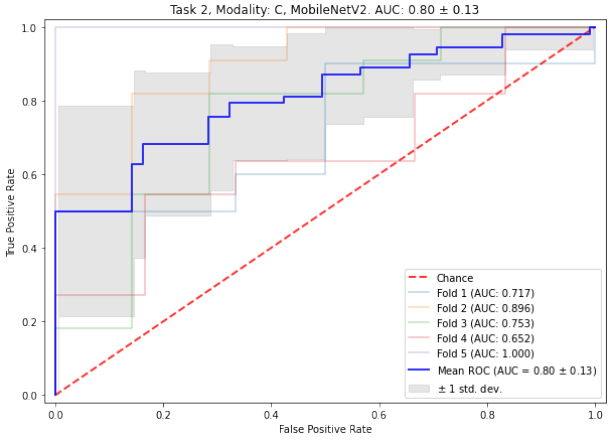
Task 1

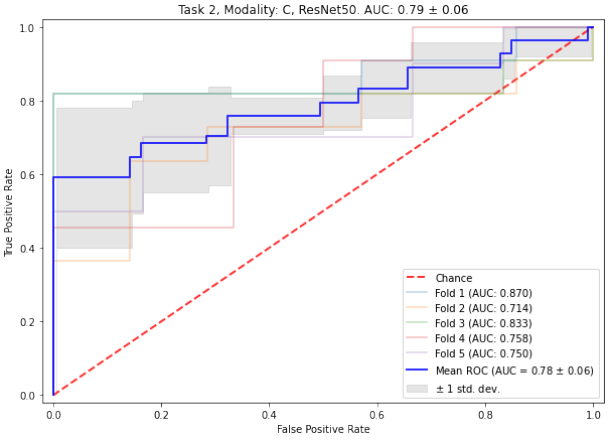


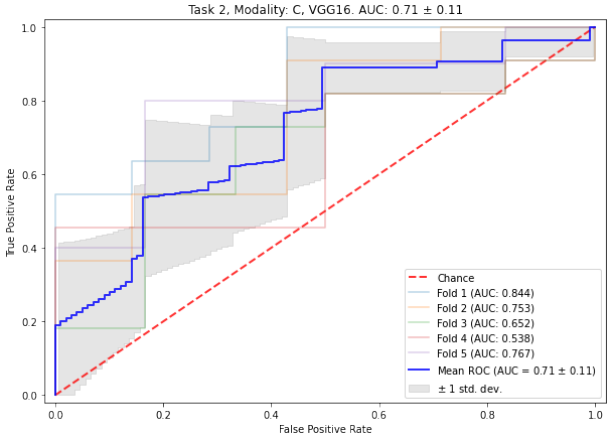

Task 1, Modality: CB, VGG16. AUC: 0.80 ± 0.02
True Positive Rate
False Positive Rate
Chance
Fold 1 (AUC: 0.827)
Fold 2 (AUC: 0.809)
Fold 3 (AUC: 0.802)
Fold 4 (AUC: 0.769)
Fold 5 (AUC: 0.797)
Mean ROC (AUC = 0.80 ± 0.02)
± 1 std. dev.

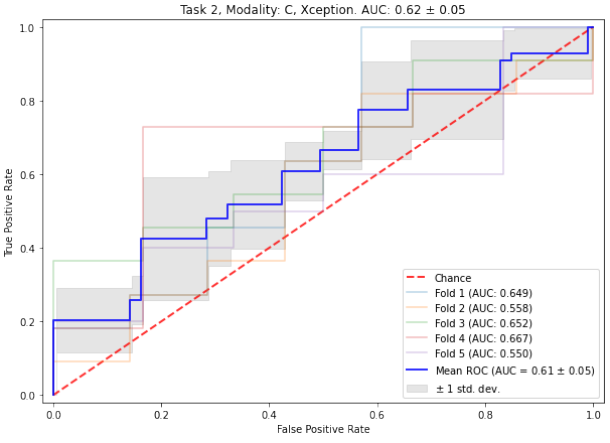

Task 1, Modality: CB, Xception. AUC: 0.81 ± 0.05
True Positive Rate
False Positive Rate
Chance
Fold 1 (AUC: 0.752)
Fold 2 (AUC: 0.798)
Fold 3 (AUC: 0.890)
Fold 4 (AUC: 0.805)
Fold 5 (AUC: 0.827)
Mean ROC (AUC = 0.81 ± 0.05)
± 1 std. dev.

Task 2

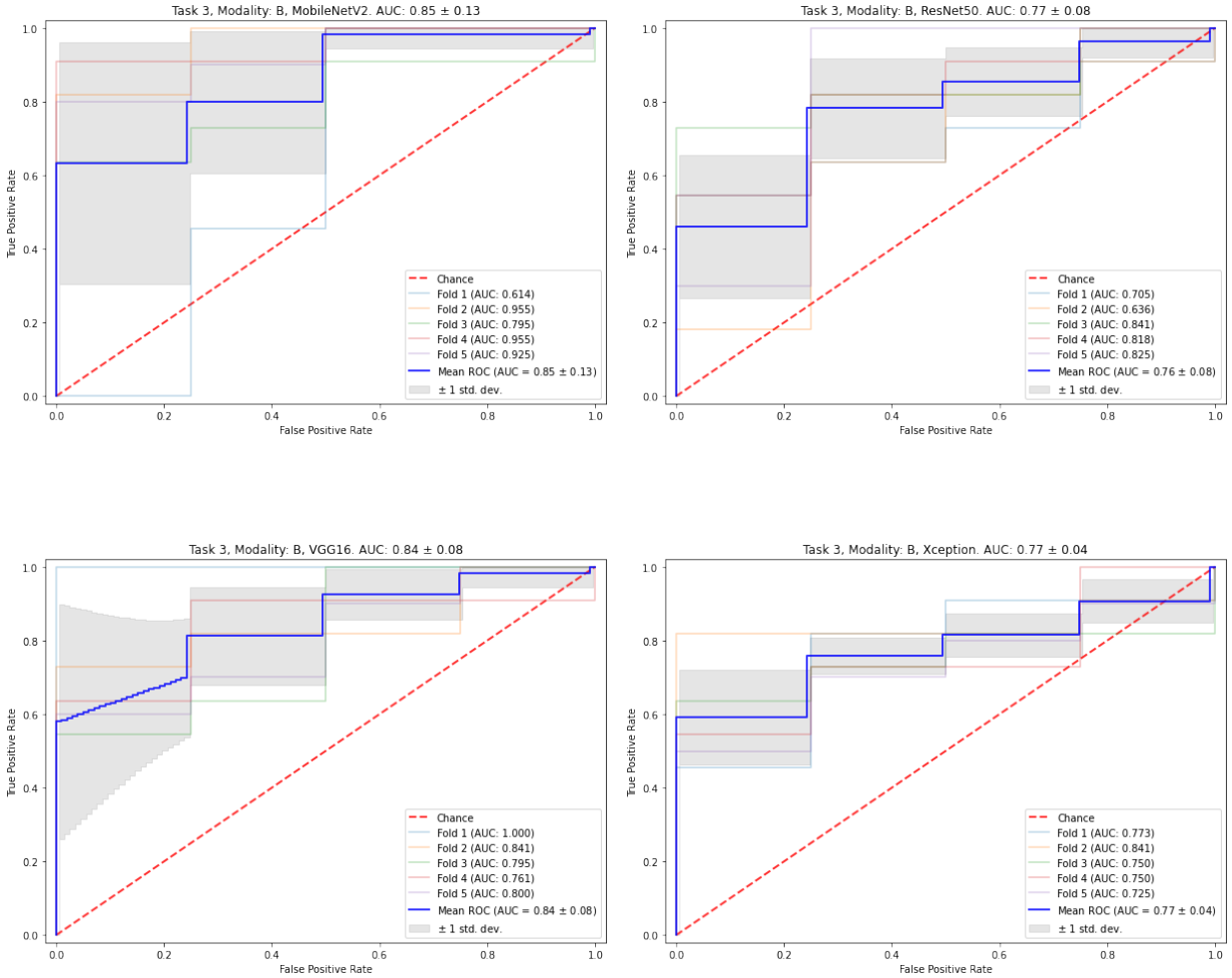

Task 2, Modality: C, MobileNetV2. AUC: 0.80 ± 0.13
True Positive Rate
False Positive Rate
Chance
Fold 1 (AUC: 0.717)
Fold 2 (AUC: 0.896)
Fold 3 (AUC: 0.753)
Fold 4 (AUC: 0.652)
Fold 5 (AUC: 1.000)
Mean ROC (AUC = 0.80 ± 0.13)
± 1 std. dev.


Task 2, Modality: C, ResNet50. AUC: 0.79 ± 0.06
True Positive Rate
False Positive Rate
Chance
Fold 1 (AUC: 0.870)
Fold 2 (AUC: 0.714)
Fold 3 (AUC: 0.833)
Fold 4 (AUC: 0.758)
Fold 5 (AUC: 0.750)
Mean ROC (AUC = 0.78 ± 0.06)
± 1 std. dev.


Task 2, Modality: C, VGG16. AUC: 0.71 ± 0.11
True Positive Rate
False Positive Rate
Chance
Fold 1 (AUC: 0.844)
Fold 2 (AUC: 0.753)
Fold 3 (AUC: 0.652)
Fold 4 (AUC: 0.538)
Fold 5 (AUC: 0.767)
Mean ROC (AUC = 0.71 ± 0.11)
± 1 std. dev.

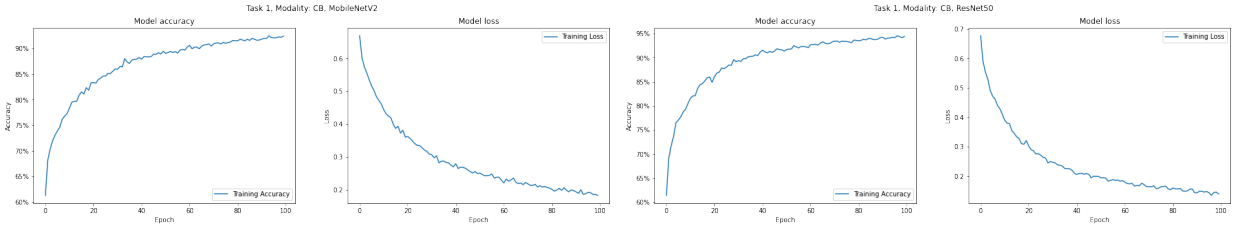

Task 2, Modality: C, Xception. AUC: 0.62 ± 0.05
True Positive Rate
False Positive Rate
Chance
Fold 1 (AUC: 0.649)
Fold 2 (AUC: 0.558)
Fold 3 (AUC: 0.652)
Fold 4 (AUC: 0.667)
Fold 5 (AUC: 0.550)
Mean ROC (AUC = 0.61 ± 0.05)
± 1 std. dev.

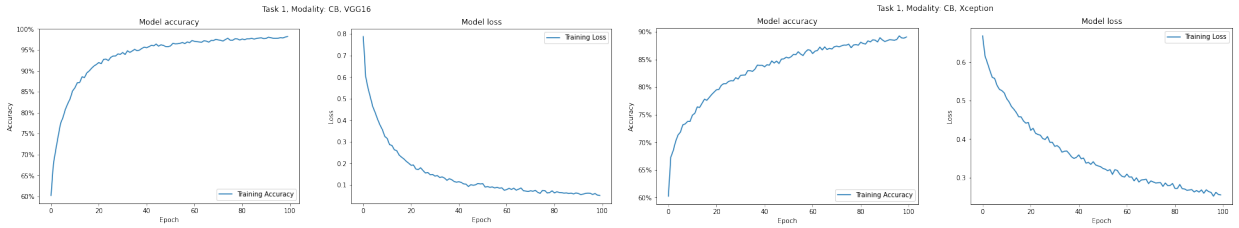
Task 3



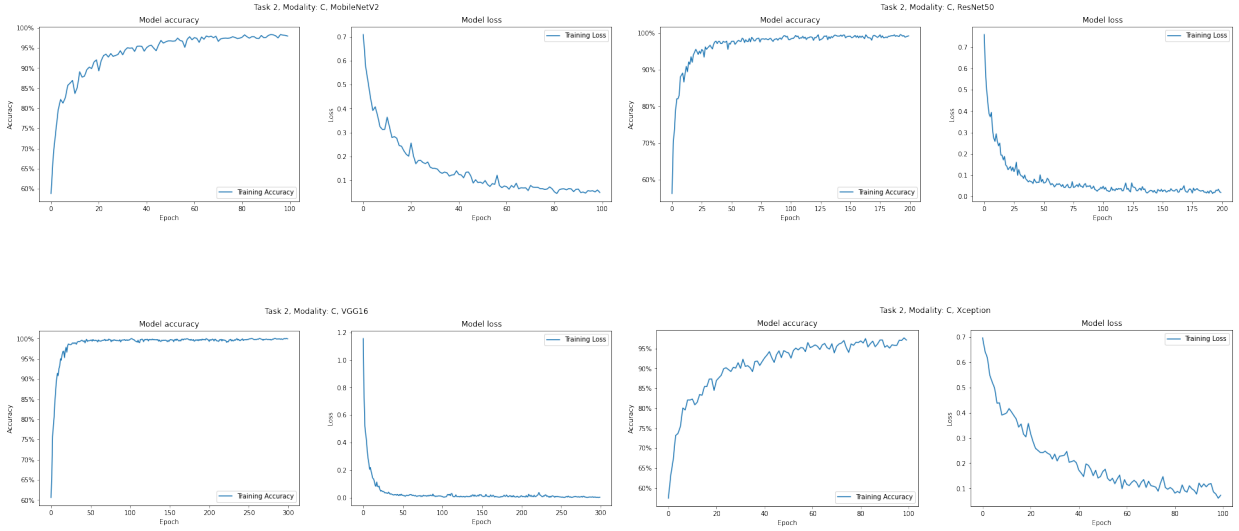
B.2 Curve di learning

Task 1

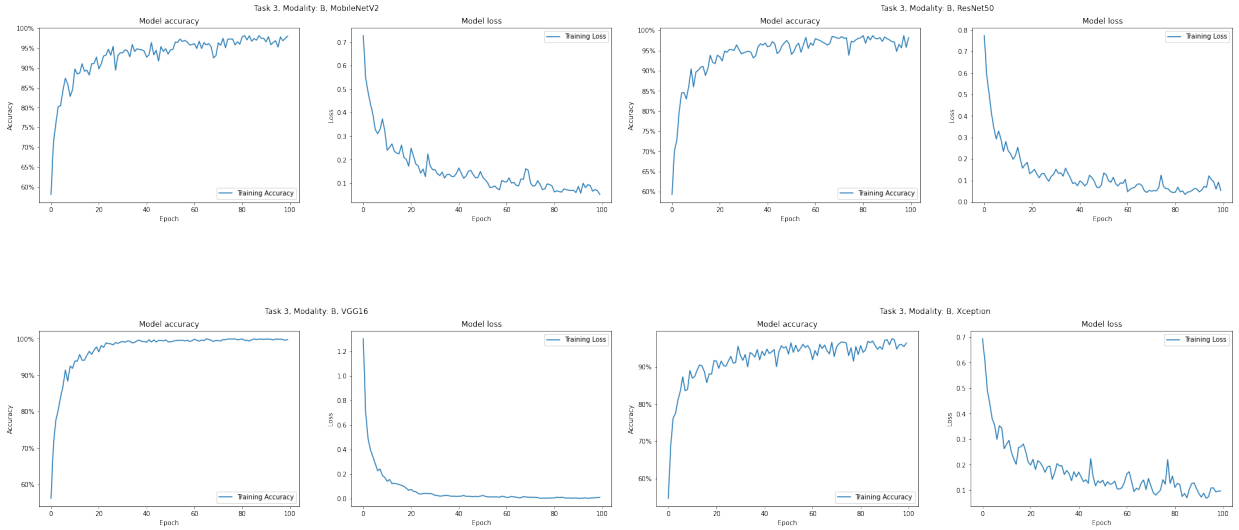



Task 1, Modality: CB, VGG16
Model accuracy
Model loss
Task 1, Modality: CB, Xception
Model accuracy
Model loss
Training Accuracy
Training Loss
Accuracy
Loss
Epoch

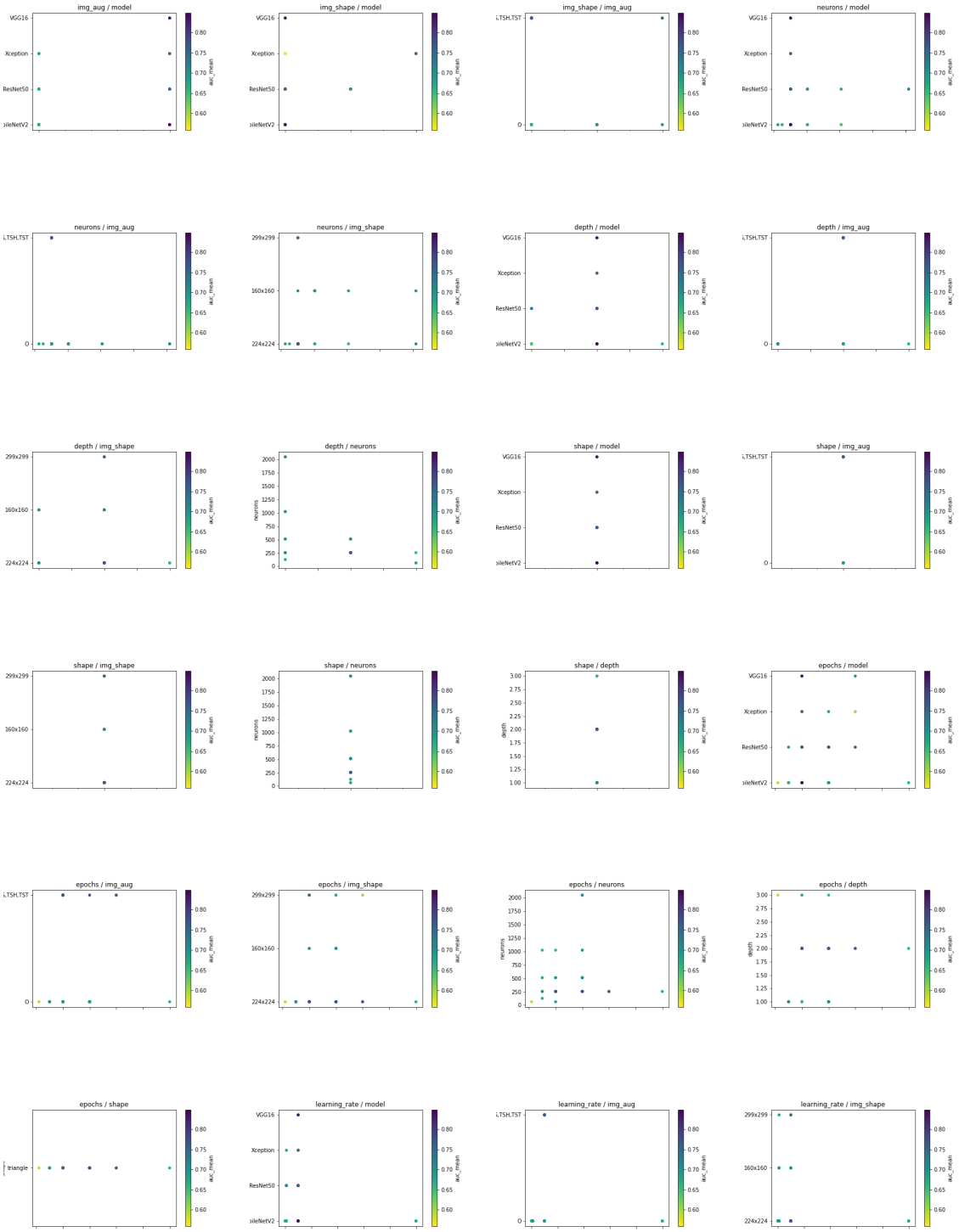
Task 2

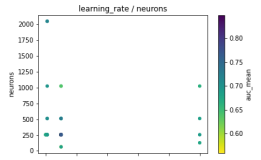

Task 2, Modality: C, MobileNetV2
Model accuracy
Model loss
Task 2, Modality: C, ResNet50
Model accuracy
Model loss
Task 2, Modality: C, VGG16
Model accuracy
Model loss
Task 2, Modality: C, Xception
Model accuracy
Model loss
Training Accuracy
Training Loss
Accuracy
Loss
Epoch

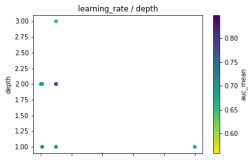
Task 3

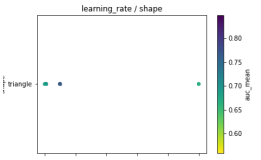

Task 3, Modality: B, MobileNetV2
Model accuracy
Model loss
Task 3, Modality: B, ResNet50
Model accuracy
Model loss
Task 3, Modality: B, VGG16
Model accuracy
Model loss
Task 3, Modality: B, Xception
Model accuracy
Model loss
Training Accuracy
Training Loss
Accuracy
Loss
Epoch

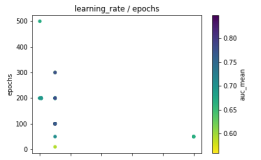
B.3 Scatter plot

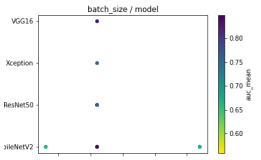


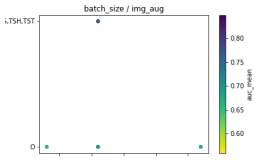

learning_rate / neurons
neurons
auc_mean

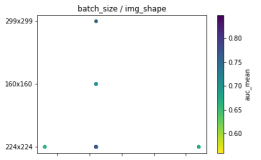

learning_rate / depth
depth
auc_mean

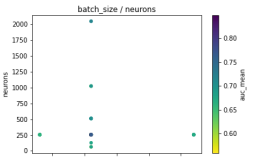

learning_rate / shape
triangle
auc_mean

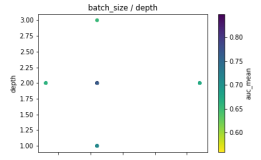

learning_rate / epochs
epochs
auc_mean

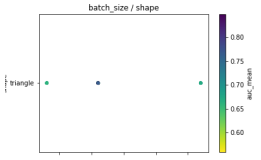

batch_size / model
VGG16
Xception
ResNet50
auc_mean

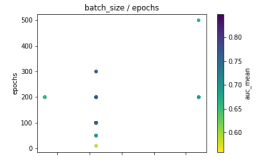

batch_size / img_aug
auc_mean

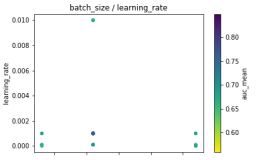

batch_size / img_shape
299x299
160x160
224x224
auc_mean

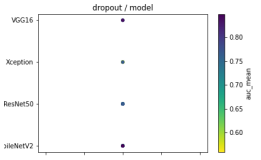

batch_size / neurons
neurons
auc_mean

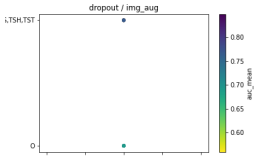

batch_size / depth
depth
auc_mean

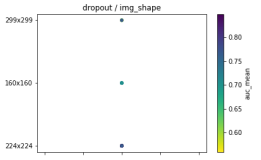

batch_size / shape
triangle
auc_mean

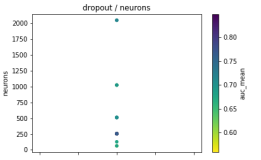

batch_size / epochs
epochs
auc_mean

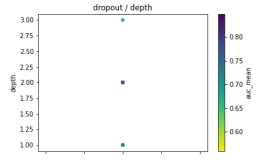

batch_size / learning_rate
learning_rate
auc_mean

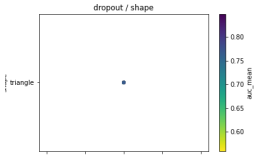

dropout / model
VGG16
Xception
ResNet50
auc_mean

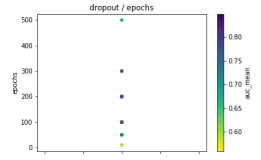

dropout / img_aug
auc_mean

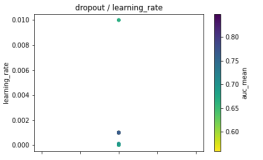

dropout / img_shape
299x299
160x160
224x224
auc_mean

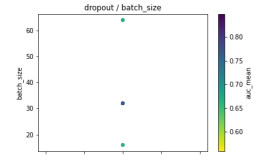

dropout / neurons
neurons
auc_mean

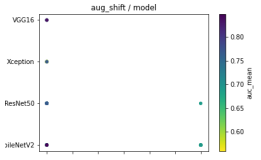

dropout / depth
depth
auc_mean

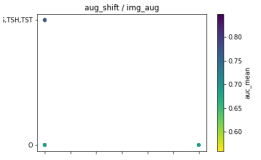

dropout / shape
triangle
auc_mean

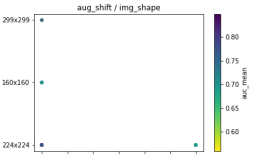

dropout / epochs
epochs
auc_mean

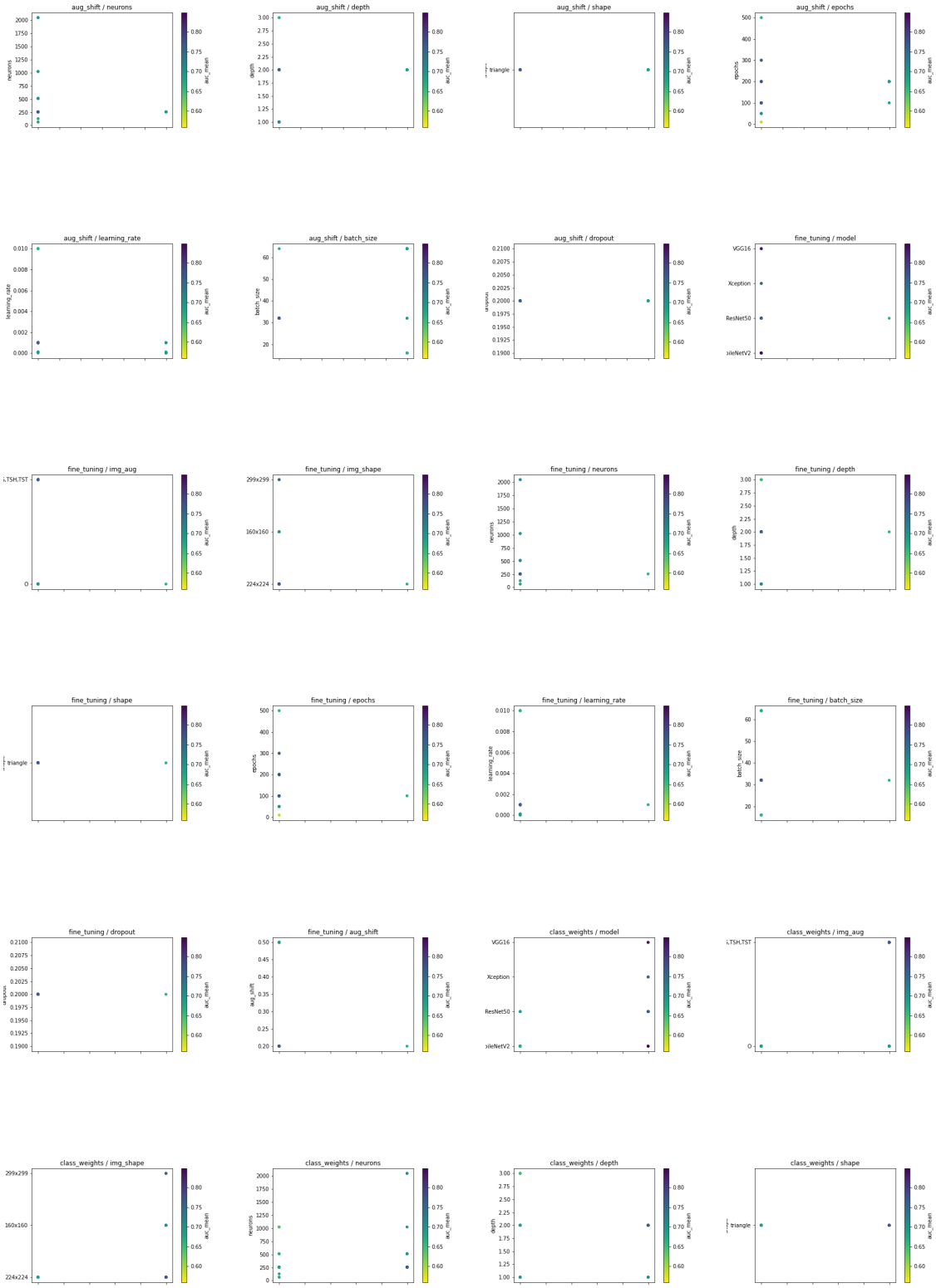

dropout / learning_rate
learning_rate
auc_mean

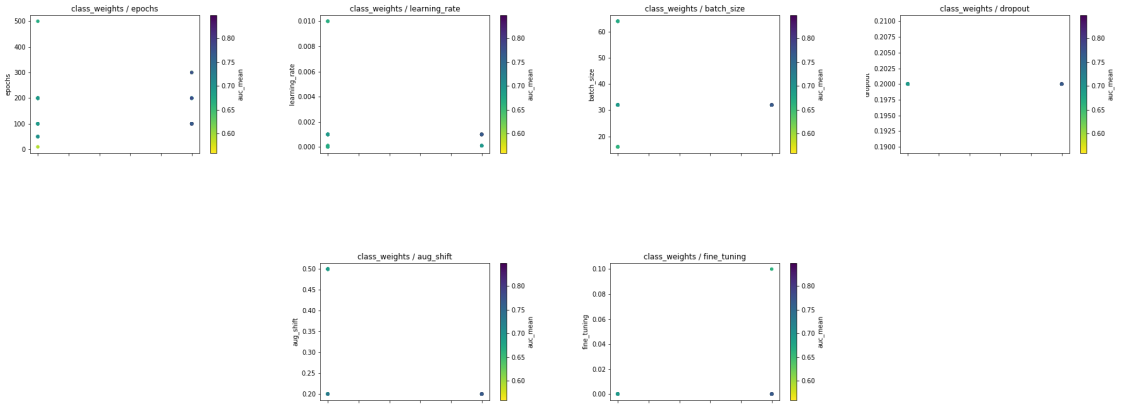

dropout / batch_size
batch_size
auc_mean


aug_shift / model
VGG16
Xception
ResNet50
auc_mean


aug_shift / img_aug
auc_mean


aug_shift / img_shape
299x299
160x160
224x224
auc_mean


aug_shift / neurons
aug_shift / depth
aug_shift / shape
aug_shift / epochs
aug_shift / learning_rate
aug_shift / batch_size
aug_shift / dropout
fine_tuning / model
fine_tuning / img_aug
fine_tuning / img_shape
fine_tuning / neurons
fine_tuning / depth
fine_tuning / shape
fine_tuning / epochs
fine_tuning / learning_rate
fine_tuning / batch_size
fine_tuning / dropout
fine_tuning / aug_shift
class_weights / model
class_weights / img_aug
class_weights / img_shape
class_weights / neurons
class_weights / depth
class_weights / shape
auc_mean


class_weights / epochs
epochs
auc_mean
class_weights / learning_rate
learning_rate
auc_mean
class_weights / batch_size
batch_size
auc_mean
class_weights / dropout
dropout
auc_mean
class_weights / aug_shift
aug_shift
auc_mean
class_weights / fine_tuning
fine_tuning
auc_mean