Progetto di Intelligenza Artificiale

CNN per la classificazione di polmoniti da COVID-19

Mattia Crispino (matricola: 320745)

1 Introduzione

Obiettivo del progetto è la realizzazione di un modello di CNN con conseguente fase di addestramento, validazione e confronto dei risultati con alcuni modelli di rete preaddestrati sulle quali viene eseguito il fine tuning.

Il dataset a disposizione è composto da immagini monocanale di radiografie di polmoni in salute e affetti da COVID-19, BacterialPneumonia e Viral Pneumonia.

Il materiale a disposizione è suddiviso in:

- *NonAugmentedTrain*: collezione delle immagini originali
- *TrainData*: collezione delle immagini della cartella precedente su cui è stato eseguito il processo di Data Augmentation
- *ValData*: insieme delle immagini originali usate come validation set

Le prime due cartelle formano il *training set* mentre l'ultima il *validation set*.

Di seguito vengono mostrati alcuni esempi.

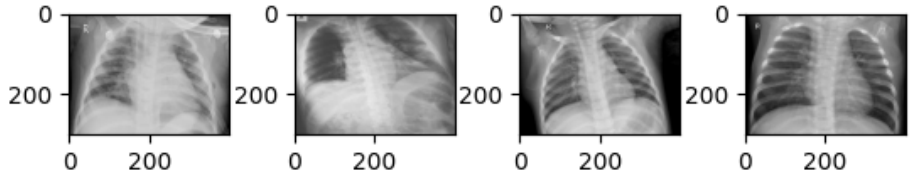
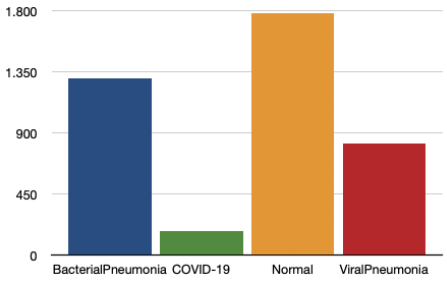


Figura 1: Da sinistra, *BacterialPneumonia*, *COVID-19*, *Normal*, *ViralPneumonia*

Per la realizzazione del progetto sono state utilizzate le librerie *Tensorflow* e *Keras* e l'addestramento delle reti è stato eseguito sfruttando la potenza di calcolo messa a disposizione da *Kaggle.com*.

Un'analisi preliminare del *training set* porta alla seguente suddivisione:



Come si può notare dal grafico sopra vi è un netto sbilanciamento tra il numero di dati delle varie classi del dataset, motivo per cui sono state utilizzate tecniche di preprocessing per rimediare a tale problema.

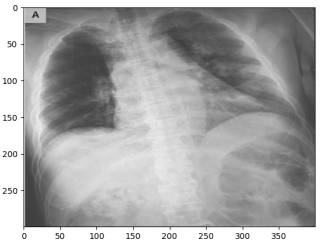
2 Preprocessing

Per il bilanciamento del training set si è impostato come valore di soglia 1000 campioni per classe; per quelle che fossero costituite da un numero maggiore di immagini (*BacterialPneumonia* e *Normal*) sono state selezionate randomicamente 500 immagini dall'insieme *NonAugmentedTrain* e *TrainData*.

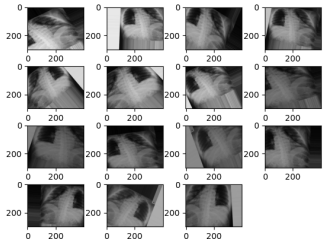
Per quanto riguarda *COVID-19* e *ViralPneumonia* si è adottata la tecnica di DataAugmentation sulle foto originali contenute, rispettivamente, nelle cartelle *NonAugmentedTrain/COVID-19* e *NonAugmentedTrain/ViralPneumonia*; in particolare per la prima, avendo a dispozione 60 campioni, si è realizzato uno script che eseguisse 15 trasformazioni per ogni immagine implementando la classe *ImageDataGenerator* offerta da Keras a cui sono stati passati come parametri:

- brightness_range=[0.5,1.0]
- rotation_range=40
- horizontal_flip=True
- zoom_range=0.2
- shear_range=0.2
- width_shift_range=0.2
- height_shift_range=0.2
- fill_mode='nearest'

Di seguito si mostra un esempio del risultato ottenuto su una immagine.



(a) Radiografia originale



(b) Data augmentation

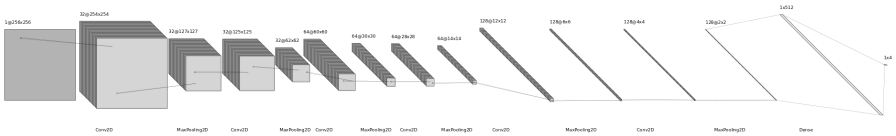
Per quanto riguarda la classe *ViralPneumonia* sono state selezionate in modo casuale 176 campioni e per ognuno di questi è stata eseguita una singola transformazione seguendo la stessa metodologia adottata in precedenza.

Al termine del processo le classi *Normal* e *BacterialPneumonia* risultano essere composte da 1000 elementi mentre *COVID-19* 1041 e *ViralPneumonia* 998.

Durante il caricamento delle immagini per la formazione del training set e del validation set, sono state associate a queste delle label per distinguere le varie classi: *BacterialPneumonia*(0), *COVID-19*(1), *ViralPneumonia*(2) e *Normal*(3).
Infine, ogni immagine viene ridimensionata a 256x256 e normalizzata.

3 CNN

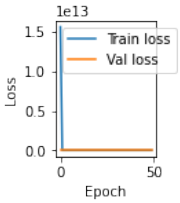
Dopo aver eseguito svariati tentativi con varie strutture, la rete risultante risulta essere così composta:



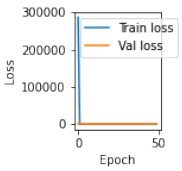
(c) CNN

Al termine dei layer convolutivi è stato inserito un *fully connected layer* composto da 512 neuroni seguito da un dropout di 0.5 (per evitare overfitting) e l'ultimo layer di classificazione composto da un numero di neuroni pari al numero di classi da identificare.

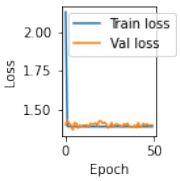
Per la selezione dei parametri di *batch_size* e *learning_rate* sono state realizzate delle funzioni che eseguissero una GridSearch e identificassero i valori ottimali.
Di seguito vengono presentati i risultati di tale ricerca:



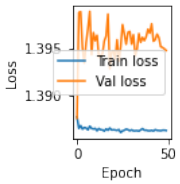
(d) lr = 1



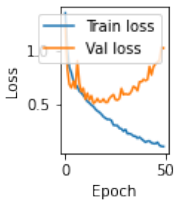
(e) lr = 1E-1



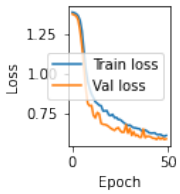
(f) lr = 1E-2



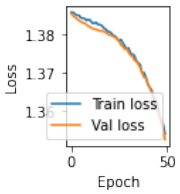
(g) lr = 1E-3



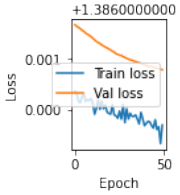
(h) lr = 1E-4



(i) lr = 1E-5



(j) lr = 1E-6



(k) lr = 1E-7

Learning_rate	*Accuracy*
1	0.0091
1E-1	0.3279
1E-2	0.2074
1E-3	0.0091
1E-4	0.6983
1E-5	0.7368
1E-6	0.5860
1E-7	0.0101

Batch size	*Accuracy*
5	0.7551
8	0.7651
16	0.7489
32	0.7338
64	0.7398
128	0.7095

I parametri selezionati risultano essere *learning_rate = 1E-5* e un *batch_size = 8*.

4 Transfer Learning & Fine Tuning

Keras mette a disposizione 26 modelli di rete preaddestrati ed è stato scritto uno script che ha consentito di scaricare ogni modello effettuando una selezione, eliminare la testa di questo inserendone una modificata e addestrarlo mantenendo i layer convolutivi bloccati.
I risultati sono stati salvati in un file *.csv* esterno; l'intero processo ha richiesto circa 2 ore.

model_name	*validation_accuracy*	*validation_loss*	*training_accuracy*	*training_loss*
DenseNet121	0.7773	0.5245	0.8838	0.3038
DenseNet169	0.7641	0.5458	0.9182	0.2262
DenseNet201	0.7388	0.5808	0.9252	0.2098
ResNet101V2	0.6923	0.7584	0.9485	0.1516
ResNet101	0.6973	0.7962	0.6256	0.8862
ResNet152	0.7004	0.7797	0.6286	0.8838
ResNet50V2	0.7277	0.6638	0.9346	0.1873
ResNet50	0.6983	0.7683	0.6315	0.8772
ResNet152V2	0.7155	0.7601	0.9428	0.1680
MobileNet	0.7287	0.6680	0.9230	0.1952
MobileNetV2	0.7307	0.6702	0.9274	0.2029
VGG19	0.7793	0.5378	0.73929	0.6512
VGG16	0.7570	0.5906	0.7296	0.6624

Dalla tabella riportata si evince come le reti VGG19 e DenseNet121 siano quelle che offrono un'accuratezza maggiore.
Successivamente si è proseguito con l'addestramento delle due reti; queste sono state inizializzate con i pesi del dataset *imagenet* non includendo la testa, sono stati congelati i layer convolutivi in modo che i pesi non venissero aggiornati durante la prima fase di training ed è stata inserita una testa modificata dalla struttura:

Flatten()
Dense(512, activation="relu")
Dropout(0.5)
Dense(256, activation="relu")
Dropout(0.5)
Dense(4, activation="softmax")

In seguito si è proseguito con la prima fase di training con l'ottimizzatore Adam, si è scongelato uno o più layer convolutivi e si è addestrata nuovamente la rete.
L'intero procedimento è stato poi ripetuto con la rete DenseNet121.

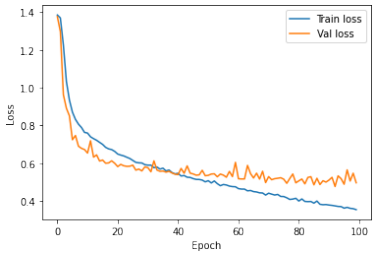
Per migliorare le prestazioni del modello, durante la fase di addestramento si è sfruttata la callback *ReduceLROnPlateau* di Keras impostando:

- *monitor* = 'val_accuracy'
- *min_delta* = 0.005
- *patience* = 3
- *min_lr* = 0.00000001

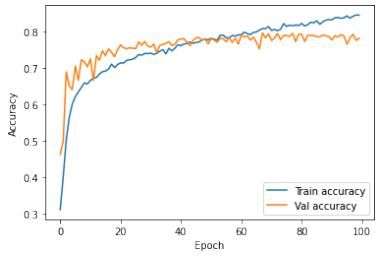
la quale ha permesso di ridurre il learning rate quando vi è un incremento minore di *min_delta* in *val_accuracy* per più di *patience* epoche.

5 Risultati

L'addestramento della CNN ha richiesto circa una decina di minuti e la valutazione ha fornito come risultato un'accuratezza del 78,14%.
Di seguito vengono riportati i grafici dei risultati ottenuti.

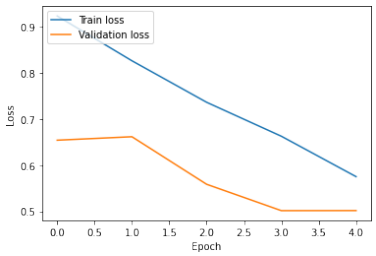


(l) Loss

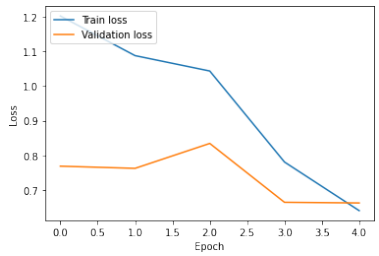


(m) Accuracy

L'addestramento dei modelli preaddestrati invece ha richiesto più tempo, dovuto a causa della complessità delle reti.
Di seguito vengono riportati i risultati ottenuti con la rete VGG19 con, rispettivamente, l'ultimo e gli ultimi due blocchi convolutivi non congelati.



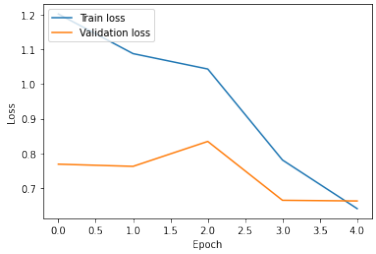
(n)



(o)

L'accuratezza ottenuta è del 77,73% nel primo caso mentre del 79,95% nel secondo.

Di seguito viene presentato il grafico della loss della rete DenseNet121 con i layer dal *conv5_block1_0_bn* non congelati ottenendo un'accuratezza del 73,98%.



(p)

Come si può notare dai risultati, la tecnica del fine tuning ha permesso di raggiungere un'accuratezza leggermente superiore alla CNN costruita solamente con la rete VGG19 a differenza di quanto accade con la DenseNet121, la quale ha peggiorato tale valore.