Dog and Cat Image Classification

Descrizione del progetto

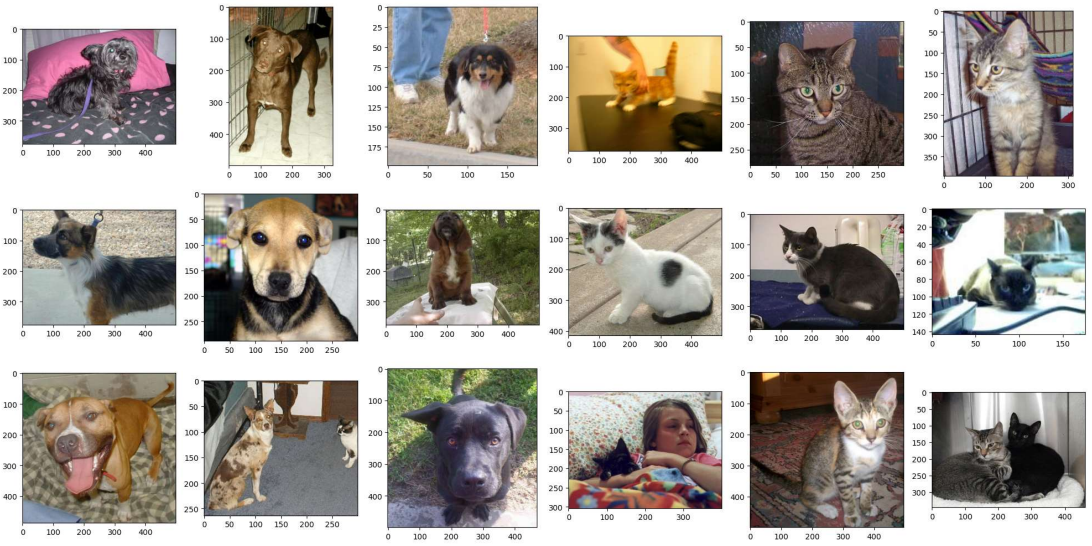
Si ha a disposizione un dataset contenente immagini di animali domestici, tra cui cani e gatti. L' obiettivo è sviluppare un modello di machine learning che possa classificare automaticamente se un'immagine contiene un cane o un gatto. Il modello deve essere in grado di raggiungere un'accuratezza di almeno il 90% nella classificazione.

Analisi e pre-elaborazione dei dati

Il dataset fornito contiene immagini di cani e gatti in cartelle separate chiamate rispettivamente Dog e Cat.

Da un'iniziale analisi osserviamo che il dataset è composto da 12500 immagini di cani e 12500 immagini di gatti per un totale di 25000 immagini. Il dataset è pertanto bilanciato.

Diamo un'occhiata a qualche immagine campione:



Si può osservare che le immagini hanno dimensioni diverse. Per questo motivo nell'elaborazione dei dati si procederà al ridimensionamento delle immagini in dimensione 224x224 pixel.

Si osserva inoltre la presenza di alcune immagini corrotte sia nel dataset dei cani che in quello dei gatti, tali immagini sono state pertanto rimosse.

Si ottiene così un dataset di 12499 immagini di cane e 12499 immagini di gatto.

Preparazione dei dati

Abbiamo utilizzato il ImageDataGenerator di Keras per automatizzare la preparazione dei dati, inclusa la normalizzazione e la divisione del dataset in set di training e validazione. Questo strumento ci ha permesso di ridimensionare tutte le immagini a una dimensione uniforme di 224x224 pixel e di applicare una normalizzazione, dividendo i valori dei pixel per 255 per ottenere valori in un range da 0 a 1. Abbiamo suddiviso i dati in modo che il 20% fosse usato per la validazione e il resto per il training.

Sviluppo del modello

CNN a tre blocchi

Il primo modello utilizzato è una Rete Neurale Convoluzionale (CNN) composta da tre blocchi convoluzionali, ciascuno seguito da uno strato di MaxPooling per ridurre la dimensionalità. Dopo gli strati convoluzionali, i dati vengono appiattiti e passano attraverso uno strato denso con 512 unità e attivazione ReLU, prima di raggiungere lo strato di output con attivazione sigmoidea per la classificazione binaria (cani vs gatti). Il modello è compilato con l'ottimizzatore Adam e la funzione di perdita binary_crossentropy, che è standard per i problemi di classificazione binaria.

Il modello è stato addestrato per 20 epoche, utilizzando i generatori di training e validazione preparati in precedenza. Abbiamo monitorato la perdita e l'accuratezza sia sul set di training che su quello di validazione per valutare i progressi del modello nel tempo.

Per visualizzare le prestazioni del modello, è stata implementata una funzione summarize_diagnostics che genera grafici della perdita e dell'accuratezza nel corso delle epoche di training e validazione. Questi grafici sono stati salvati come file PNG per un'analisi dettagliata e per facilitare la condivisione dei risultati.

Alla fine del training, il modello ha raggiunto un'accuratezza di validazione del 82.153%, precision del 0.812, recall del 0.838 e F1-score pari a 0.842.

Osservando Figura 1 vediamo chiaramente un problema di overfitting nel modello in questione. Questo si manifesta attraverso un'accuratezza elevata sui dati di addestramento, che dimostra la capacità del modello di apprendere efficacemente le etichette di questi dati. Tuttavia, la stessa performance non si riflette sui dati di validazione, dove l'accuratezza risulta significativamente inferiore. Tale discrepanza evidenzia che il modello non è in grado di generalizzare altrettanto bene su nuovi dati non visti durante l'addestramento.

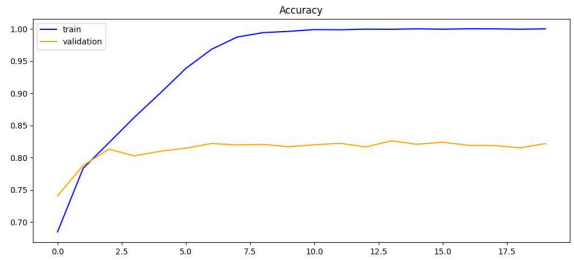


Figura 1

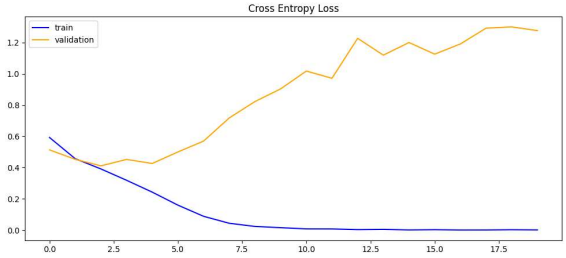


Figura 2

Un altro indicatore di overfitting è l'aumento della cross entropy loss sui dati di validazione al crescere del numero di epoche che possiamo osservare nella Figura 2. Questo significa che, nonostante il modello continui a migliorare la sua performance sui dati di addestramento, diventa progressivamente meno efficace nel prevedere correttamente i dati non incontrati precedentemente. In sintesi, mentre il modello diventa sempre più adatto ai dati di addestramento, la sua capacità di generalizzare su dati esterni si deteriora, un chiaro segno di overfitting.

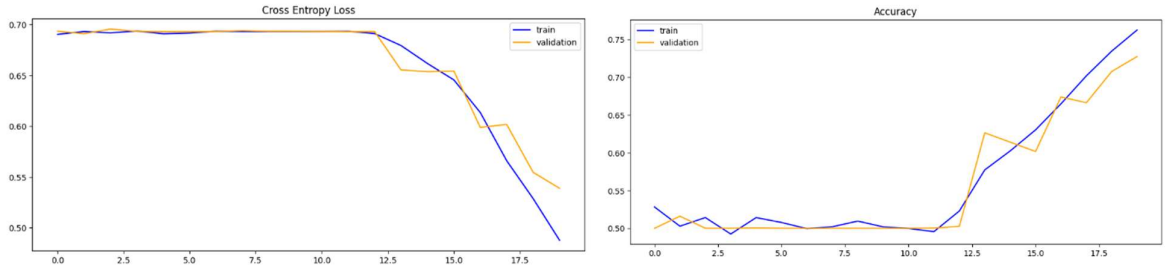
Dropout regularization

Nel tentativo di affrontare il problema dell'overfitting riscontrato nel nostro modello, è stata implementato una tecnica di regolarizzazione nota come dropout. Questo approccio consiste nel disattivare casualmente una percentuale di neuroni durante la fase di addestramento, con l'obiettivo di ridurre la dipendenza del modello dalle specifiche caratteristiche dei dati di addestramento e incoraggiare la generalizzazione su nuovi dati.

Il dropout è stato applicato in diversi punti del modello. Dopo ogni strato di MaxPooling2D, abbiamo inserito uno strato di Dropout con una probabilità di disattivazione del 20% dopo ciascuno strato di pooling. Inoltre, prima dello strato denso finale, abbiamo aumentato la probabilità di dropout al 70% per minimizzare ulteriormente il rischio di overfitting.

Per quanto riguarda la compilazione del modello, abbiamo optato per l'ottimizzatore SGD (Stochastic Gradient Descent) con un learning rate di 0.001 e un momentum di 0.9, sostituendo l'ottimizzatore Adam precedentemente utilizzato. Questa scelta è stata guidata dalla volontà di applicare un approccio più controllato nell'aggiornamento dei pesi, potenzialmente conducendo a una convergenza più stabile e a una migliore generalizzazione.

L'adozione di queste modifiche ha lo scopo di mitigare l'overfitting osservato, promuovendo una maggiore capacità di generalizzazione del modello su dati non visti durante l'addestramento, e riflette un approccio metodico per affrontare una delle sfide comuni nello sviluppo di modelli di machine learning.

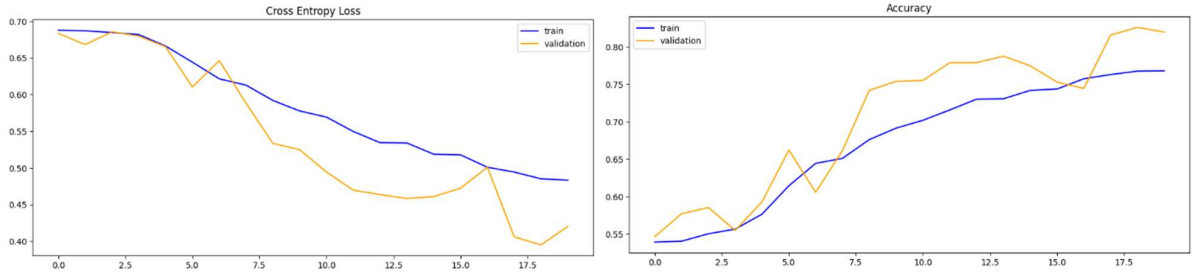


Come possiamo osservare dalle figure il problema di overfitting viene mitigato grazie all'adozione della tecnica di dropout, tuttavia i risultati non sono ancora ottimali, abbiamo un accuratezza di validazione del 72.749%, precision del 0.861, recall del 0.542 e F1-score del 0.666.

Data augmentation

Per aumentare l'accuratezza del modello proviamo ad utilizzare il metodo della data augmentation il metodo di data augmentation, ovvero aumentiamo la varietà delle immagini su cui allenare il modello ruotando o specchiando le immagini di training a disposizione. Il dataset di test è stato invece mantenuto invariato.

L'implementazione della data augmentation ha portato a un miglioramento dell'accuratezza del modello, otteniamo infatti ora un'accuratezza di validazione del 81,973%, precision del 0.854, recall del 0.771 e F1-Score del 0.81

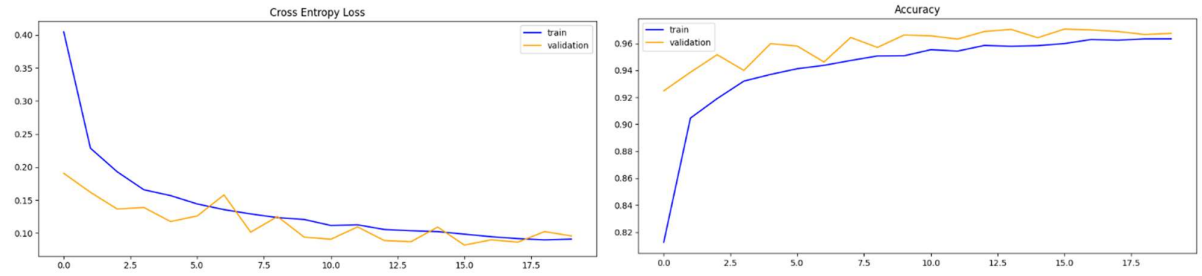


Modello pre-allenato VGG16

Non avendo ancora raggiunto l'accuratezza desiderata proviamo ad utilizzare un modello pre-allenato. La scelta cade sul modello VGG16, noto per la sua efficacia nel riconoscimento di immagini grazie alla sua architettura profonda e alla capacità di catturare caratteristiche complesse.

Nella nostra implementazione, abbiamo caricato il VGG16 con i pesi pre-addestrati, escludendo l'ultimo strato completamente connesso (include_top=False), per adattarlo al nostro compito specifico di classificazione binaria. Abbiamo mantenuto inalterati i pesi delle prime 15 layers della rete, marcandoli come non addestrabili per preservare le caratteristiche generiche apprese durante il pre-addestramento. L'allenamento è stato fatto mantenedo la data augmentation.

Il modello risultante è stato compilato ancora una volta con l'ottimizzatore SGD. Utilizzando questa configurazione il modello ottiene ottimi risultati.



Abbiamo un'accuratezza di validazione pari al 96.739, precision del 0.949, recall del 0.988 e F1-score pari a 0.968.