Deep Learning Onramp:

**1) Introduzione**

Il Deep Learning è una tecnica di machine learning che usa reti neurali per fare predizioni. Vi sono immagini come input e da queste immagini si capiscono caratteristiche e classificazioni.

**2) Usare Reti preallenate**

**Reti neurali convoluzionali:**

Le CNN sono un tipo di reti neurali usate dove gli input hanno una struttura bidimensionale, come le immagini. Sono composte da layer e le informazioni vengono trasferite tra layer. L'output rappresenta il livello di qualità delle predizioni effettuate dalla rete. Le immagini in input vengono trasformate in un set di caratteristiche che gli ultimi layer utilizzano per fare la classificazione. Questo avviene mediante l'architettura della rete, che stabilisce l'ordine dei layer e i vari valori. I layer contengono tanti parametri, detti pesi, che determinano come i layer si comportano quando li vengono passati dei dati. I loro valori sono determinati dal training della rete su certe label. Due reti con la stessa architettura potrebbero comportarsi diversamente a seconda dei dati training che vengono passati a entrambi, perciò dal training dipende la qualità delle predizioni e la precisione di una rete.

**Classificazione:**

Per classificare un'immagine in input in una tra n classi possibili, una rete neurale contiene un output layer di n neuroni, uno per classe. Per ogni neurone la rete calcolerà un valore diverso, che rappresenterà il valore di predizione della rete della probabilità per cui quel neurone corrisponda a ogni classe.

**Schema di una CNN:**

Input image -> Features -> Classification

**3) Gestire collezioni di immagini**

**Introduzione:**

Può capitare di dover gestire una grande quantità di immagini, per questo è utile utilizzare l'Image Datastore. Esso è una variabile che contiene un set di immagini. Una rete neurale può fare predizioni su un intero set di immagini grazie al datastore.

**Gestire le dimensioni:**

Quando si voglioni gestire immagini, bisogna fare in modo che le immagini abbiano una certa dimensione, in modo che matchino correttamente.

**4) Transfer Learning**

**Introduzione:**

La tecnica con cui si prende una rete preallenata, viene modificata e viene allenata con nuovi dati, in modo che nuove immagini vengano riconosciute in certe categorie, è detta Transfer Learning.

**Utilizzo:**

Per utilizzare in modo efficiente il Transfer Learning, occorrono tre cose fondamentali:

* una rete da allenare, coi propri layer, poiché il succo del TL è di utilizzare una rete preallenata per aggiungere nuovi dati da classificare;
* dati per allenare la rete, delle immagini d’esempio per cui ho già delle corrette etichette, nella terminologia tecnica si dice supervised learning. La rete imparerà da questi esempi con una risposta (etichetta) corretta già conosciuta.
* Algoritmi e le loro opzioni, che possono essere ottimizzati tramite dei parametri, come quante immagini di training da usare, numero di iterazioni, learning rate ecc..

Avendo tutte queste cose, utilizzando la funzione trainNetwork(data,layers,options), il training avverrà tranquillamente.

**Gestire imperfezioni delle immagini:**

A volte, le immagini da classificare, potrebbero non essere perfette, cioè possono presentare strane angolature, luminosità non ottima o altro. Per gestire queste situazioni si usa l’augmentedImageDatastore, che permette di trasformare le immagini (dimensioni, rotazioni, traslazioni…) all’interno del datastore e utilizzarle come dati.

**Modificare Layer di rete:**

Deep Learning with MATLAB:

**1) Classificare immagini con reti convoluzionali**

**Introduzione:**

Il deep learning può essere usato per identificare testo, audio o immagini. È possibile poi classificare e prevedere dei dati sequenziali.

Nel deep learning, si usano reti preallenate per classificare immagini; importare e preprocessare immagini o modificare una rete preallenata per il transfer learning.

**4) Network training**

**Introduzione:**

Una rete neurale è un'operazione matematica che prende degli input e restituisce output. Servono tre cose fondamentali per creare una rete: Training data, architettura di rete e opzioni algoritmiche. Il primo provvede esempi di input con corrispondenti output. La seconda dà una struttura generale sulla matematica dietro alla mappatura tra input e output; tuttavia la rete possiede molti parametri conformi: pesi e bias dei layer. Questo significa che in base alla scelta dei parametri, la stessa architettura può mappare gli stessi input in diversi output. Durante l'addestramento di una rete, si usano certi input e output per determinare i valori dei parametri. Questa determinazione dipende dall'errore (loss function) che vi è tra le predizioni della rete e gli output conosciuti. I valori dei parametri con la minima quantità di errore saranno quelli presi in considerazione. L'addestramento di una rete consiste nel minimizzare questa perdita rifinendo i parametri, mediante la tecnica della retropropagazione dell'errore, che consiste, invece di calcolare il valore di loss mediante i parametri, nel calcolare i valori dei parametri tramite la discesa del gradiente, una tecnica che aggiorna i valori dei parametri di una certa quantità lungo la direzione in cui la funzione di perdita decresce il più velocemente possibile, con una matematica complessa e calcolandone la derivata parziale rispetto al parametro considerato.

**Accuratezza, perdita e learning rate:**

Per ottenere buoni risultati in una rete neurale, servono molti dati training, ma calcolare la perdita e il gradiente mediante tutti questi dati renderebbe la rete molto costosa, perciò si dividono i set di dati (immagini) in lotti, per ognuno dei quali corrisponderà una certa iterazione. Ogni perdita e gradiente calcolati in ogni lotto saranno un'approssimazione di quelli dell'intero set di dati training. Ogni passo effettuato durante la discesa del gradiente viene detto "learning rate", definibile come la mole di aggiornamenti che vengono effettuati ai parametri dopo ogni passo. È possibile decrementare questo learning rate durante il processo.

Il problema di questo processo è che la funzione di perdita potrebbe variare rapidamente con alcuni parametri ma più lentamente con altri, ciò implica una diversità dei valori dei gradienti. I gradienti con un grande valore fanno cambiare i valori dei parametri in modo irrefrenabile, al contrario quelli con piccolo valore. Si può ovviare a questo problema grazie all'impostazione di una dimensione massima dei gradienti. Inoltre, è possibile eseguire dei passi più lenti in prossimità dei gradienti con grande valore, e viceversa per quelli con piccolo valore, tramite l'impiego di particolari algoritmi (es. RMSProp, Adam).

Si possono generare particolari grafici mediante la funzione plot di Matlab, che mostrano il progresso del training. In ordinate vengono poste l'accuratezza (%) delle immagini di training classificate correttamente dalla rete durante un'iterazione; e la perdita, una misura di quanto sia distante una predizione perfetta dalla capacità della rete; in ascisse il numero di passi (iterazioni), suddivisi in un certo numero di epoche, ovvero il numero di volte cui la rete osserva tutti i set di dati.

**Testing e validazione:**

Solitamente, per validare la performance di una rete, si usano i dati testing. Tuttavia, è possibile usare anche dei dati di validazione. La differenza sostanziale è che i primi si usano dopo il training, i secondi durante il training. I dati di validazione sono utili per scoprire se la rete sia sovradattata. Se la training loss sta decrementando e la validation loss sta aumentando, bisogna fermare il training perché la rete sta imparando dettagli sul training non rilevanti per nuove immagini. Se la validation loss è minore della training loss, la rete è sovradattata.

**5) Migliorare la performance della rete**

**Introduzione:**

È possibile che i dati posseduti, l'architettura e gli algoritmi della rete non siano ottimali per renderla adeguata. Per migliorare la performance, si consiglia di modificare le opzioni di training degli algoritmi; aumentare la quantità di dati training, ma se non si hanno abbastanza dati, è possibile usare l'augmentation (aumento), in Matlab si può gestire quest'ultima tecnica mediante dei datastore. Infine è possibile ottimizzare l'architettura, anche se non se ne ha spesso bisogno. Sia l'accuratezza che il tempo di training sono due caratteristiche fondamentali della performance di una rete. Vi dev'essere un costante bilancio fra alta accuratezza e tempo di training in diminuzione progressiva. Se la rete ha problemi nel training, identificherà tutte le predizioni come undefined. A volte è possibile che il learning rate sia troppo alto e il valore di perdita sia NaN, quindi conviene in questo caso decrementare il suo valore per avere un'accuratezza e una perdita adeguati.

**Guide per le opzioni di training:**

Bisogna avere un valore di learning rate ideale, poiché se troppo piccolo, ci vorrebbe troppo tempo per allenare la rete, e se troppo grande, la perdita potrebbe essere NaN e ci sarebbe un incremento spropositato della perdita all'inizio del training. L'ideale sarebbe renderlo pari al valore massimo per cui la rete funzioni correttamente. La validazione è la chiave per determinare se la rete sia sovradattata, perciò aggiungere il data augmentation o limitare il numero di epoche possono ovviare a ciò. Se la perdita aumenta anche alla fine del training, occorre aumentare il numero di epoche.

**Schema per il miglioramento della performance:**

- Aumentare il learning rate finché non ci siano NaN o punte nella perdita

- Ci sono NaN o grandi punte nella perdita? Sì -> diminuisci il learning rate

- No -> La perdita sta ancora diminuendo alla fine del training? Sì -> aumenta il numero di epoche

- No -> Setta la learn rate schedule a drop at plateau

- La training loss è più bassa della validation loss? Sì -> Usa augment sui training data, aumenta la regolarizzazione

- No -> Classificazione o regressione?

- Classificazione -> bilanciamento o classi di pesi

- Regressione -> Normalizzazione dei dati