BODY CONDITION SCORE DATABASE

BCS
DATABASE

Problema:

Il monitoraggio delle riserve dei bovini da latte è un'area di ricerca che si sta espandendo. Esso permette di capire, tramite il BCS (Body Condition Score), se il bovino ha necessità di una dieta o meno, in quanto le caratteristiche del prodotto finale sarebbero influenzate dal suo stato di salute. L'indice in questione può essere monitorato tramite immagini digitali e una Neural Network che effettua una regressione. Quest'ultima, tramite i risultati, renderebbe più agevole la comprensione dello stato di salute del bovino, e di conseguenza, permetterebbe di comprendere quali diete siano più adatte o meno.
In conclusione, l'uso di una rete neurale ad hoc per il problema in questione permette di acquisire consapevolezza sulla qualità del prodotto finale e di poter intervenire per migliorarlo e renderlo più equilibrato.

Dataset:

Il dataset conta 207 immagini a colori, corrispondenti a 29 body shapes di differenti bovini, acquisite con differenti condizioni di illuminazione e angolo di rotazione. Non sono state effettuate restrizioni su dimensioni, età, etc. I punti anatomici delle silhouette sono stati etichettati manualmente insieme ai Body Condition Scores calcolati da due differenti esperti del settore. Le immagini sono state acquisite tramite telecamere piazzate all'uscita del robot di mungitura.

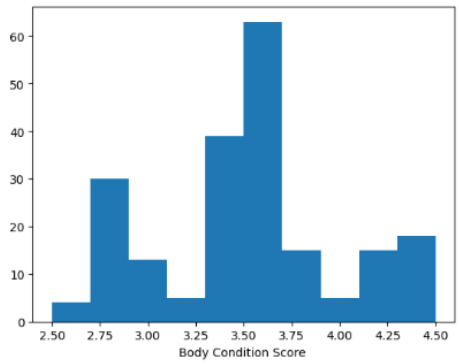
L'immagine in questione rappresenta uno dei 207 sample che il dataset contiene.

Il dataset è stato acquisito tramite BeautifulSoup4, una libreria Python che facilita lo scraping da siti web. Effettuando uno scraping rispetto alla struttura della pagina relativa al dataset è stato possibile estrapolare i nomi dei file immagini e gli indici BCS1 e BCS2. A questo punto è quindi stato costruito un dataset che contiene i nomi delle singole immagini e il relativo indice BCS, calcolato come la media tra i BCS1 e BCS2 forniti dagli esperti del settore.

Successivamente si è notato che all'interno della pagina contenente il dataset, alla fine dello stesso, veniva ripetuto un'altra volta. Per questo motivo è stato necessario effettuare un drop dei duplicati nel dataset Pandas come passo di pulizia dei dati preliminare per evitare qualsivoglia forma di overfitting a priori.

A questo punto è stato istanziato un custom dataset tramite Torch che dalla directory "sample_data/dataset_cows" inserisce le immagini collegate ai rispettivi nomi dei file. Si sono effettuate prove sia sulle immagini a tre canali RGB, sia sulle immagini elaborate in scala di grigi a singolo canale, visto che era necessario estrapolare le sole informazioni sulla silhouette dei bovini.

Il dataset presenta silhouettes con indici abbastanza sparsi nel range [2.5, 4.5] con picchi intorno al valore medio di 3.5.



Istogramma rappresentativo del BCS sul dataset

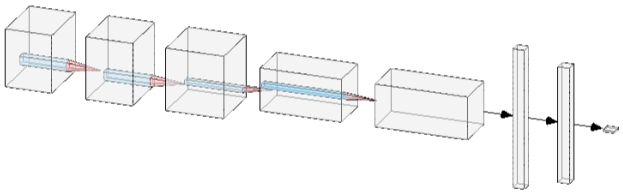
Per quanto riguarda il dataset è stato organizzato in una cartella denominata "dataset_cows" contenente le sole immagini dei bovini, che va inserita all'interno della folder "sample_data" sulla piattaforma Google Colab. Le restanti informazioni quali i BCS e i nomi dei file immagini da collegare al dataset vengono acquisite a run-time tramite il Notebook stesso.

Metodi:

Essendo il dataset composto da immagini si è resa palese la necessità di gestire il problema tramite Convolutional Neural Networks. Esse permettono di apprendere le features più facilmente rispetto a Multi Layer Perceptrons. Nello specifico viene sostituita la moltiplicazione tra matrici con l'operatore di convoluzione. Esso fa scorrere un filtro sull'immagine in input per apprendere le features più rilevanti.

Inizialmente si è tenuta in considerazione come architettura di riferimento MiniAlexNet, che è stata adattata rimuovendo l'ultimo layer originariamente destinato alla classificazione sostituendolo con un regressore lineare. E' rimasta invariata la struttura principale che conta:

- 5 Convolutional Layers;
- 2 Fully Connected Layers



Nello specifico si è effettuato un resize delle immagini che comunque mantenesse il livello di proporzioni rispetto alle immagini originali. Per questo motivo il resize è stato impostato a samples di dimensioni 32x42.

Successivamente si è provato ad implementare due custom CNNs con differente numero di nodi per hidden layer:

- 4-5 Convolutional Layers;
- 2 Fully Connected Layers

Anche in questo caso si sono utilizzate le tecniche di regolarizzazione sopra indicate.

Al variare dei parametri di configurazione dei singoli layers e degli iperparametri si sono raggiunti buoni risultati.

Dal punto di vista dell'algoritmo di ottimizzazione si sono utilizzati:

- Stochastic Gradient Descent
- Adam

Dal punto di vista della regolarizzazione sono state utilizzate:

- Dropout con probabilità p oscillante tra 0.4 e 0.6;
- Data augmentation tramite RandomHorizontalFlip, RandomVerticalFlip, RandomRotation e ColorJitter;
- Batch normalization sia sui convolutional layers sia sui fully connected layers.

La funzione di attivazione utilizzata è stata la ReLU.

Sono inoltre stati fatti test con learning rate schedulers: nello specifico è stato utilizzato uno scheduling di tipo "Step-Decay", che ogni 30 epochs riduce il valore del parametro di apprendimento. Questa scelta è stata dettata dal fatto che il valore della funzione di Loss non riusciva a scendere ulteriormente rispetto a un valore di 0.2, per questo motivo si è preferito provare ad adattare il learning rate per tentare di ottenere risultati ancora più precisi, trattandosi di regressione.

Valutazione:

La misura di valutazione utilizzata per le reti neurali è stata la MSE Loss:

$$\text{MSE Loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

La MSE Loss misura la distanza euclidea tra l'etichetta di ground truth del sample in questione e l'etichetta predetta dal regressore. Quanto più piccolo è il valore di Loss, tanto più preciso è il regressore costruito.

Esperimenti:

Per quanto concerne le reti utilizzate si sono fatti svariati esperimenti, ma sono stati conservati nel Notebook solo quelli che hanno raggiunto risultati migliori. Le reti utilizzate, come già specificato in precedenza, sono di tipo CNN.

Per tutte le reti costruite si è effettuato un resize delle immagini a dimensioni 32x42. Alcune reti sono state addestrate su immagini a tre canali RGB, e altre sono state addestrate su immagini a singolo canale in scala di grigi.

Rete	Learning Rate	Learning Rate Scheduler	Momentum	Weight decay	Epochs	Loss su Training Set	Loss su Test Set
MiniAlexNet	10^{-4}	*Fixed*	0.9	0.1	200	0.209	0.270
MiniAlexNetV2	10^{-3}	*Fixed*	0.5	0.2	200	0.231	0.231
MiniAlexNetV3	10^{-3}	*Step decay*	None	0.1	400	0.161	0.214

Le prime due versioni di MiniAlexNet sono state addestrate per 200 epochs perché si è notato, utilizzando precedentemente un numero maggiore di epochs, che il valore di loss rimaneva nell'intorno di 0.2.

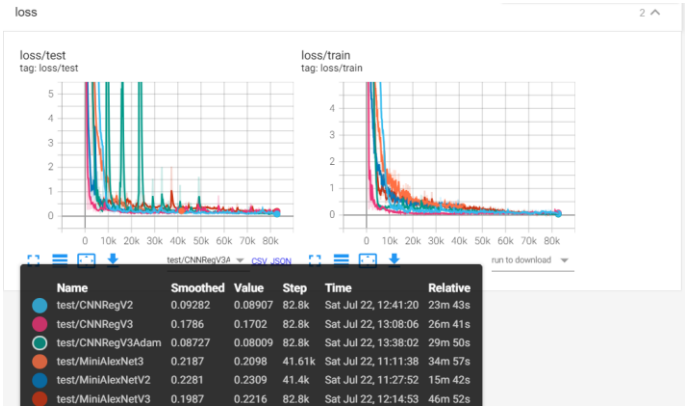
L'ultima rete MiniAlexNetV3 invece, sicuramente anche grazie allo scheduler di learning rate, ha raggiunto risultati migliori, con loss molto più basse. Si è mantenuto un numero di epoche maggiore in funzione del parametro di learning rate decrescente e del fatto che il valore di Loss continuava a diminuire.

Le tre versioni utilizzate di MiniAlexNet differiscono per padding e canali di input: nelle prime due versioni si sono infatti utilizzate immagini RGB, mentre nell'ultima sono state utilizzate immagini in greyscale.

Relativamente all'altro tipo di rete utilizzata, definita come CNNRegressor, si sono utilizzate solamente immagini a tre canali RGB.

Rete	Learning Rate	Learning Rate Scheduler	Momentum	Weight decay	Epochs	Loss su Training Set	Loss su Test Set
CNNRegressorV2	3×10^{-4}	*Step decay*	None	0.1	400	0.089	0.113
CNNRegressorV3	10^{-3}	*Fixed*	0.98	0.1	400	0.170	0.127
CNNRegressorV3	10^{-3}	Step decay	None	0.1	400	0.080	0.068

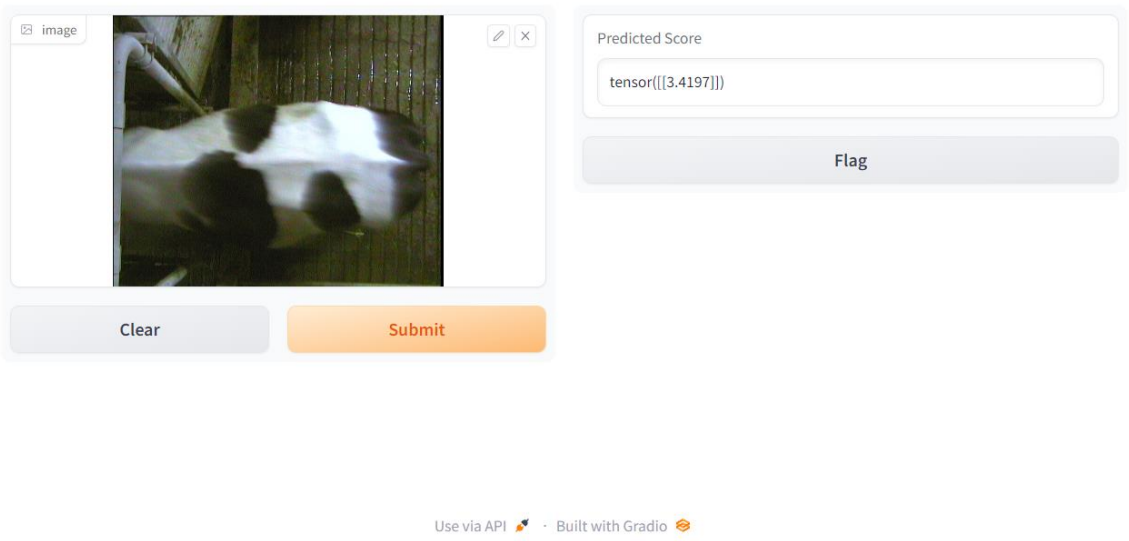
Anche in questo caso la scelta dello step decay ha fornito un improvement significativo dal punto di vista delle prestazioni.



Tramite Tensorboard sono stati visualizzati i risultati di training rappresentati in figura. La rete CNNRegV3Adam è quella che ha ottenuto in assoluto il risultato migliore.

Demo:

Per la realizzazione della demo si è utilizzato Gradio: una libreria Python che permette la realizzazione di una demo di un modello di Machine Learning. Gradio genera una interfaccia grafica in cui si trovano due riquadri: il primo permette il caricamento di un'immagine da elaborare tramite la rete neurale realizzata, mentre il secondo elabora il risultato finale della rete sotto forma di "Predicted Score".

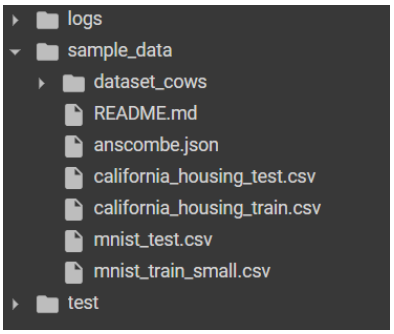


Sono state implementate due demo rappresentative per effettuare il confronto delle prestazioni tra due modelli: MiniAlexNetV3 e CNNRegV3Adam.

Codice:

Il codice è stato scritto sotto forma di notebook tramite la piattaforma Google Colab. Il notebook è organizzato in sottosezioni per facilitare il riconoscimento dei vari step e dei vari training dei modelli neurali proposti come esperimenti.

Per farlo funzionare basta importare su Google Colab il notebook, creare una cartella "dataset_cows" all'interno della cartella "sample_data" e creare una cartella al di fuori di essa dal nome "test" che conterrà i modelli successivamente alle varie fasi di training. Il resto dei dati sarà gestito automaticamente.



Per inizializzare la demo su Gradio è necessario importare i modelli CNNRegV3Adam-400.pth e MiniAlexNetV3-400.pth all'interno della cartella test.

Conclusione:

In definitiva si tengono in considerazione i modelli che hanno riportato Loss minore sul Test Set: CNNRegressorV2 e MiniAlexNetV3.

Il problema di regressione sull'indice corporeo dei bovini da latte ha richiesto varie implementazioni che hanno portato, almeno inizialmente, in linea di massima stesse prestazioni, con una loss che si aggirava intorno al valore di 0.2 sul Test Set perennemente.

Le ultime reti allenate, MiniAlexNetV3 e CNNRegressorV2 hanno portato prestazioni migliori delle altre sicuramente anche grazie al quantitativo di epoche di addestramento più esteso e alla migliore ottimizzazione dei parametri, anche tramite Step Decay learning rate scheduling.

Per migliorare il metodo proposto si potrebbe utilizzare uno scheduling più efficiente in fase di training o provare ad aumentare i dati di training stesso tramite ulteriori tecniche di Data Augmentation per differenziare ulteriormente i sample nella misura in cui non fosse possibile aumentare il quantitativo di dati reali da sfruttare per l'addestramento.