Università degli Studi di Catania
Dipartimento di Matematica ed Informatica
Corso di Laurea Magistrale in Informatica

CLASSIFICAZIONE BINARIA DI CANI E GATTI TRAMITE LOGISTIC REGRESSION E CONVOLUTIONAL NEURAL NETWORK

Prof. Giovanni Maria FARINELLA

Francesco Anastasio

Sommario

Introduzione

1.1 Scopo del progetto

Il progetto si pone l'obiettivo di creare una rete per la classificazione automatica di immagini raffiguranti cani e gatti.

Le immagini saranno estrapolate in modo automatico dal social network Flickr tramite le API messe a disposizione dallo stesso.

Verranno utilizzati i modelli messi a disposizione da PyTorch, LogisticRegression, squeezenet1_0 e AlexNet (Convolutional Neural Network).

1.2 Convolutional Neural Network

Le Convolutional Neural Networks sono una categoria di reti neurali mostratesi molto efficaci per l'image recognition e l'image classification. Le reti neurali classiche non risultano adatte al processamento di immagini in quanto queste possono essere molto grandi, complesse e potrebbero essere necessari parecchi layer. Quando il numero di input e di layer cresce lo fa anche il numero di parametri. L'idea è, quindi, di ridurre il numero di parametri per ogni layer e creare modelli più profondi.

Nelle CNN la moltiplicazione matriciale viene sostituita con la convoluzione. Così facendo ogni nodo di un layer sarà totalmente connesso solo a una regione dell'immagine e i pesi da imparare saranno i valori dei kernel convolutivi.

1.3 PyTorch

PyTorch è una libreria di machine learning per il linguaggio di programmazione Python basata sulla libreria Torch. Inizialmente sviluppata dal gruppo di ricerca

in Intelligenza Artificiale di Facebook, fornisce due features principali ad alto livello:

- Tensor computing con forte accelerazione tramite uso di GPU
- Deep Neural Network costruite su sistema di auto differenziazione

1.4 Flickr

Flickr è una piattaforma che permette a qualsiasi utente, previa registrazione, di caricare e condividere immagini con tutta la community. A differenza di altri Social Network simili, come ad esempio Instagram, Flickr permette di caricare e quindi archiviare immagini con una qualità decisamente superiore permettendo anche ai professionisti di far apprezzare a pieno i propri scatti.

Modelli

2.1 LogisticRegression

La regressione logistica è un algoritmo di classificazione supervisionato, esso, dato un input, restituisce un output binario (nella sua forma base) discreto.

Contrariamente a ciò che si crede, la regressione logistica è un modello di regressione, esso infatti diventa di classificazione con l'inserimento di una funzione a "soglia".

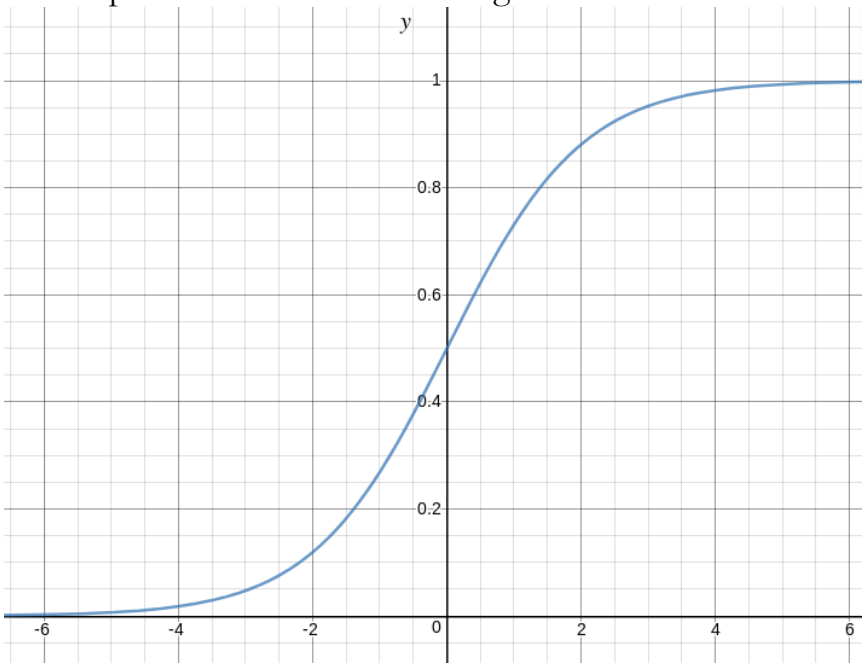
Nello specifico, il modello è:

$$h_\theta(x) = g\big((\theta^T x)\big) dove\ g(z) = \frac{1}{1 + e^{-z}}$$

Combinando le due equazioni il modello finale è il seguente:

$$\frac{1}{1 + e^{\theta^T x}}$$

Il grafico della precedente funzione è il seguente:



La soglia viene scelta di volta in volta in base al problema che si sta affrontando.

2.2 SqueezeNet

2.2.1 Introduzione

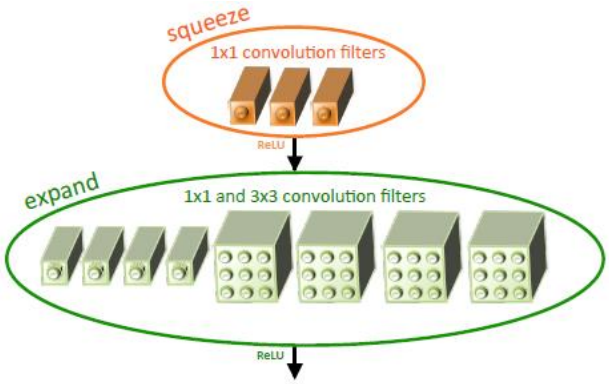
SqueezeNet è una Neural Network rilasciata nel 2016. È stata sviluppata dai ricercatori della DeepScale, University of California, Berkeley e Stanford University. Il goal degli autori è stato creare una rete neurale piccola e con pochi parametri mantenendo la qualità dei risultati.

La rete venne presentata come "in grado di ottenere accuracy ai livelli di AlexNet ma con molti meno parametri". È importante notare che nonostante questa premessa, si tratta di una rete totalmente diversa da AlexNet, avente in comune con essa solo le percentuali di accuracy ottenute sul dataset ImageNet.

2.2.2 Architettura

Fire module:

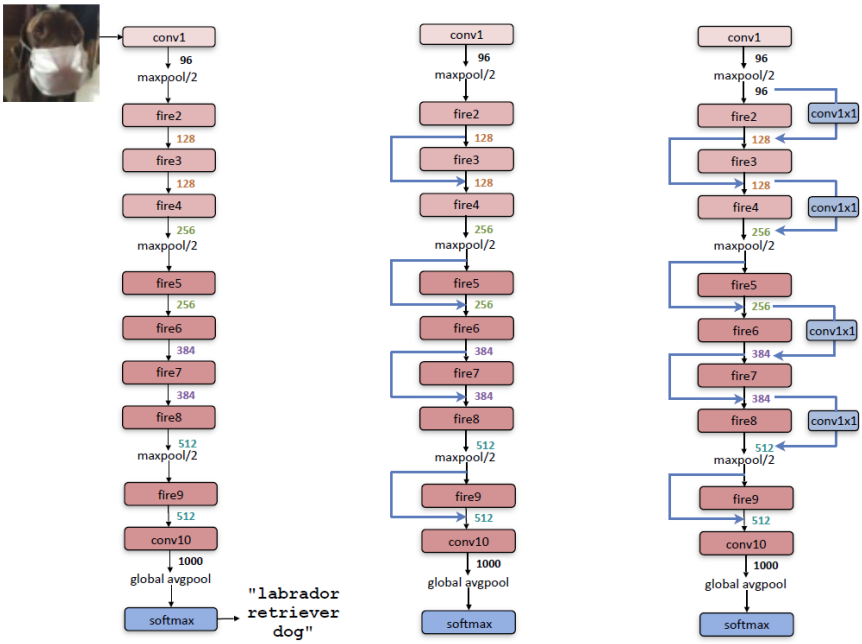
Un modulo che verrà utilizzato nella rete finale. È composto da uno strato di convoluzione che ha solamente kernel 1x1, il cui output viene passato a uno strato convolutivo di espansione formato da un mix di kernel 1x1 e 3x3.



SqueezeNet Architecture:

- Inizia con un layer di convoluzione seguito da 8 Fire Module e finisce con un layer di convoluzione finale.

- Il numero di kernel per Fire Module viene incrementato gradualmente man mano che si procede nella rete.
- Viene applicato il Max Pooling, con stride 2, dopo il layer conv1, fire4, fire8, conv10.
- Può essere applicato un bypass semplice o un bypass complesso, ispirato a ResNet.



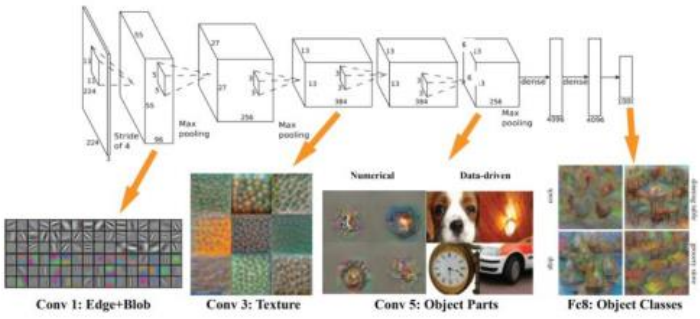
2.3 AlexNet

2.3.1 Introduzione

AlexNet è una Convolutional Neural Network progettata da Alex Krizhevsky. La rete ha vinto il ILSVRC nel 2012. Risolse il problema di classificazione dove l'input era una immagine di una certa classe su 1000 e l'output un vettore di 1000 elementi rappresentanti una distribuzione di probabilità sulle classi. L'input atteso è una immagine di dimensioni 256x256 a 3 canali (RGB).

2.3.2 Architettura

AlexNet è molto più grande delle precedenti CNN usate per risolvere task di Computer Vision. Ha 60 milioni di parametri e 650.000 neuroni.




```
class AlexNet(nn.Module):

    def __init__(self, num_classes=1000):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3,   , kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(96,256 , kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(256, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384,384 , kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), 256 * 6 * 6)
        x = self.classifier(x)
        return x
```

La rete è composta da 5 layer convolutivi e 3 layer fully-connected. Il primo layer convolutivo contiene 96 kernel di dimensione 11x11x3, dove il 3 è dovuto al numero di canali dell'immagine.

All'output di ogni layer convolutivo è applicata la ReLU. I primi due layer convolutivi sono seguiti da layer di Max Pooling, mentre gli altri 3 sono direttamente connessi tra loro. L'ultimo layer convolutivo è pure seguito da un Max Pooling, il quale output viene dato a una serie di due layer fully-connected. Per classificare, in fine, l'output dell'ultimo layer viene passato ad un Softmax. L'overfitting viene ridotto con due fasi di DropOut, che permettono di spegnere e accendere i neuroni con probabilità del 50%.

Implementazione

3.1 Creazione Dataset

Per la creazione del dataset sono state sfruttate le API messe a disposizione da Flickr, sono state scaricate circa 10000 immagini in modo automatico.

Successivamente si è passati alla creazione automatica di due file di testo, train.txt e test.txt per la divisione del dataset in trainingset e testset.

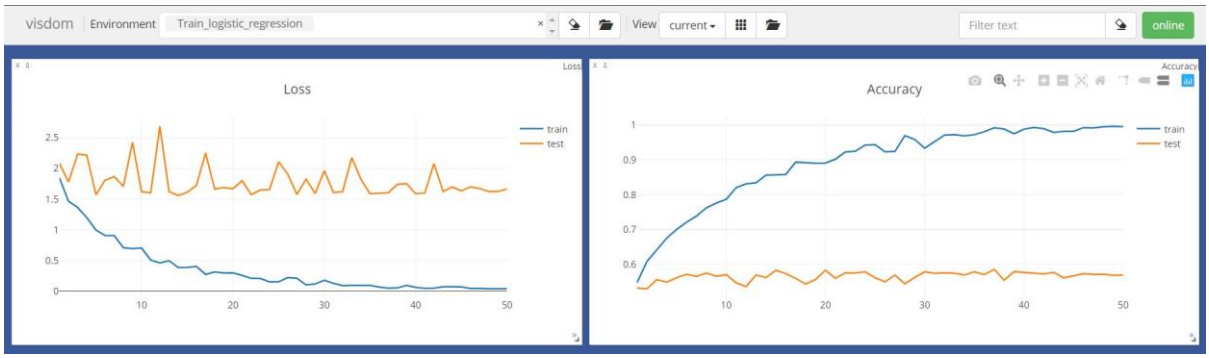
Infine, sono stati creati gli oggetti di tipo DataLoader relativi al trainingset e al testset.

Ogni classificatore è stato allenato diverse volte, andando a modificare di volta in volta il momentum e il learning rate cercando di ottenere i risultati migliori possibili: di seguito sono riportati soltanto i migliori esperimenti per ogni classificatore.

3.2 Risultati con Logistic Regression

Il primo approccio è stato quello di utilizzare un classificatore logistico per predire, data un'immagine di input, se all'interno vi fosse rappresentato un gatto o un cane.

I risultati non sono stati dei migliori, come prevedibile:



L'accuracy di train si attesta sullo 0.99, mentre quella di test non supera lo 0.60, segno che il modello và in overfitting, si specializza troppo e non riesce a generalizzare bene.

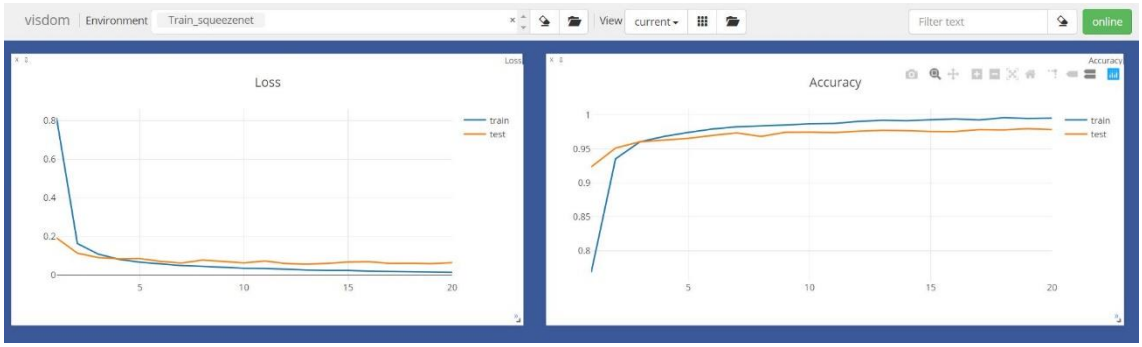
La matrice di confusione risulta molto bilanciata anche negli errori:

```
[[705 530]
 [520 702]]
```

3.3 Risultati con Squeezenet

Visto gli scarsi risultati ottenuti con il modello logistico è stato pensato di utilizzare una rete di tipo CNN, nate con l'intento di analizzare le immagini.

Sono state testate due reti, la prima delle quali è Squeezenet, i risultati sono stati di molto superiori:



L'accuracy di train in questo caso si attesta sullo 0.99, quella di test invece sullo 0.98.

In questo caso non abbiamo overfitting e arriviamo ad accuracy piuttosto alte sul task con una buona generalizzazione.

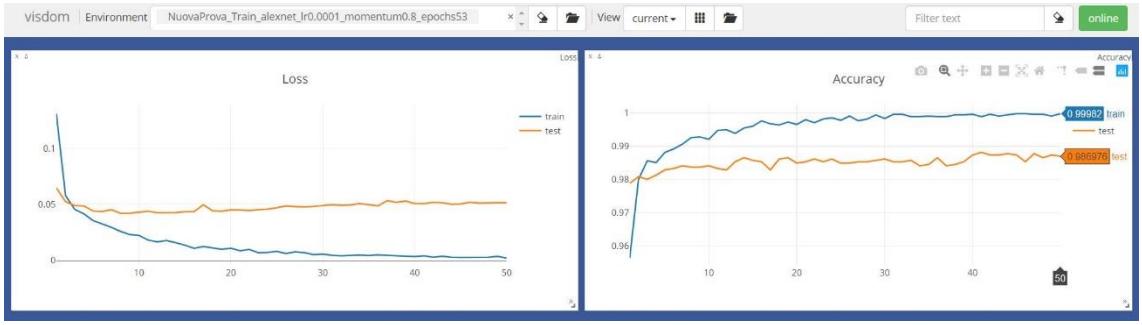
La matrice di confusione conferma quanto indicato dall'accuracy così come tutti gli altri indicatori analizzati:

```
++++++++++ Squeezenet ++++++++++
****Indicatori per dataset di Test****
Accuracy: 0.99
Precision_score-->
0.9873992175945805
Recall_score-->
0.9874063226806788
F1_Score-->
0.987382979022983
Matrice di confusione-->

[[1214   21]
 [  10 1212]]
```

3.4 Risultati con AlexNet

Nonostante gli ottimi risultati è stato deciso di fare un ulteriore tentativo per cercare di aumentare ancora l'accuracy ed è stata utilizzata AlexNet.



In questo caso i risultati sono stati analoghi ai precedenti, leggermente più bassi in termini di millesimi:

```
++++++++++ AlexNet ++++++++++
****Indicatori per dataset di Test****
Accuracy: 0.99
Precision_score-->
0.9869756223619606
Recall_score-->
0.9869756223619606
F1_Score-->
0.9869756223619606
Matrice di confusione-->

[[1219   16]
 [  16 1206]]
```

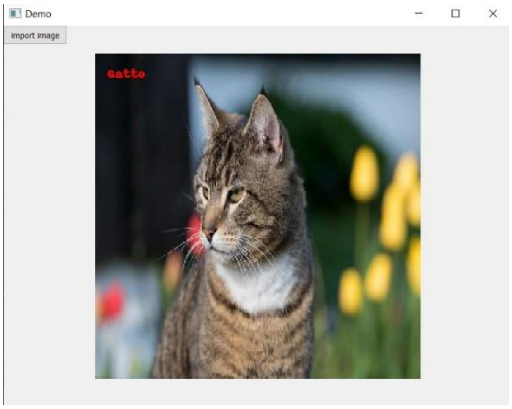
Demo

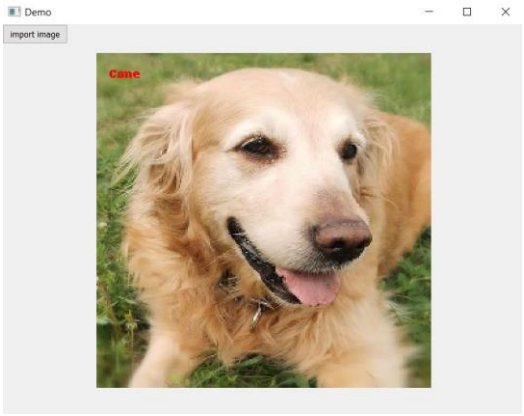
È possibile testare la AlexNet allenata sul dataset base tramite una demo.
La demo si presenta come un'applicazione desktop cross platform, sviluppata con PyQt5, in cui, alla pressione del tasto "import image" si apre una finestra di navigazione che permette di selezionare graficamente l'immagine da testare. Una volta confermata, l'immagine scelta sarà mostrata al centro della finestra principale e su di essa verrà scritto il nome della persona predetta dalla rete.

Per avviare la demo basta aprire un terminale nella cartella “Demo” e lanciare il comando:

- python3 Demo.py

Scaricando le dovute dipendenze qualora non lo si fosse già fatto.





Conclusioni

La prima fase del progetto è stata caratterizzata dalla realizzazione di un dataset con abbastanza immagini da riuscire ad allenare la rete.
Successivamente sono stati provati i tre approcci, andando a modificare di volta in volta parametri come il learning rate e il momentum, per poter ottenere i risultati migliori possibili, dato il task. I migliori valori di accuracy ottenuti sono i seguenti:

- Accuracy di train: 1
- Accuracy di test: 0.99

Il modello con i migliori risultati è stato Squeezenet.