Fine tuning

Flag di configurazione

Variabili che permettono l'esecuzione selettiva del codice e scelta di alcuni parametri come la rete da utilizzare.

```
% === VARIABILI DI STAMPA ================== %
printTrainingSet = 1;
printTestSet = 1;
printConfMatr = 1;

% === ALTRE VARIABILI ====================== %
numClasses = 2;
```

Caricamento dei dati

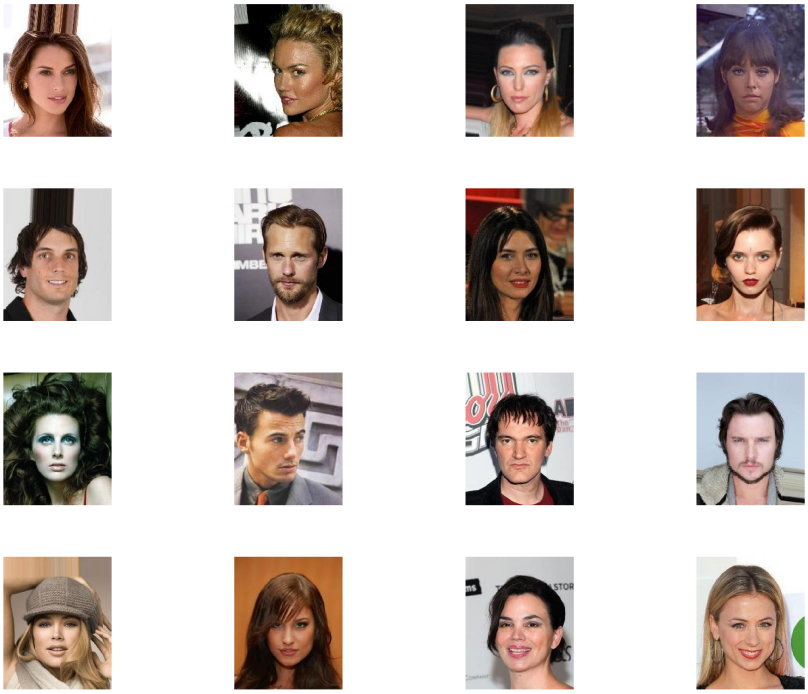
Vengono caricati i dati di training, validation e test set come image datastore. imageDatastore automaticamente applica le label alle classi in base ai nomi delle cartelle delle immagini e salva i dati come oggetti ImageDatastore.

```
% caricamento dei dati
imdsTrain = imageDatastore('dataset/TrainSet/','IncludeSubfolders',true,'LabelSource','
imdsValidation = imageDatastore('dataset/ValSet','IncludeSubfolders',true,'LabelSource'
imdsTest = imageDatastore('dataset/TestSet/','IncludeSubfolders',true,'LabelSource','fo
%numero di dati caricati
numTrainImages = numel(imdsTrain.Labels);
numValImages = numel(imdsValidation.Labels);
numTestImages = numel(imdsTest.Labels);
```

Analisi dei dati

Stampa casuale di alcune delle immagini appartenenti al training set

```
if printTrainingSet == 1
    idx = randperm(numTrainImages,16);
    figure
    for i = 1:16
        subplot(4,4,i)
        I = readimage(imdsTrain,idx(i));
        imshow(I)
    end
end
```



Caricemento rete

```
tic; %init timer
net = alexnet;
```

Ridimensinamento immagini

AlexNet prende immagini di dimensione 227x227, quindi è necessario effettuare un ridimensinamento degli input per far si che siano conformi alla rete. Per ridimensionare automaticamente il training e il test set prima di essere dati come input si utilizza augmentedImageDatastore passando come input, tra le altre cose, la dimensione che devono avere le immagini.

Dopo di che si possono utilizzare questi datastore come argomenti di input.

```
inputSize = net.Layers(1).InputSize;
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);
```

```
YTrain = imdsTrain.Labels;
YTest = imdsTest.Labels;
```

```
YTrain = double(YTrain(:,1));
YTest = double(YTest(:,1));
```

Fine Tuning

Vengono presi tutti i layer della rete eccetto gli ultimi tre. Questi ultimi tre layer della rete preaddestrata sono configurati per 1000 classi, quindi vanno "fine-tunati" per il nuovo problema di classificazione.

Gli ultimi tre layer sono stati rimpiazzati con dei layer fully connected, un softmax layer e un layer di classificazione per l'output. I layer fully connected sono stati impostati per avere la stessa dimensione delle classe nei nuovi dati.

Per imparare più velocemente nei nuovi layer, aumentare i valori di WeightLearnRateFactor e la BiasLearnRateFactor nei fully connected layer.

```
freezedLayers = net.Layers(1:end-3);
layers = [
    freezedLayers
    fullyConnectedLayer(numClasses, ...
        'WeightLearnRateFactor',20, ...
        'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
```

Tuning delle opzioni

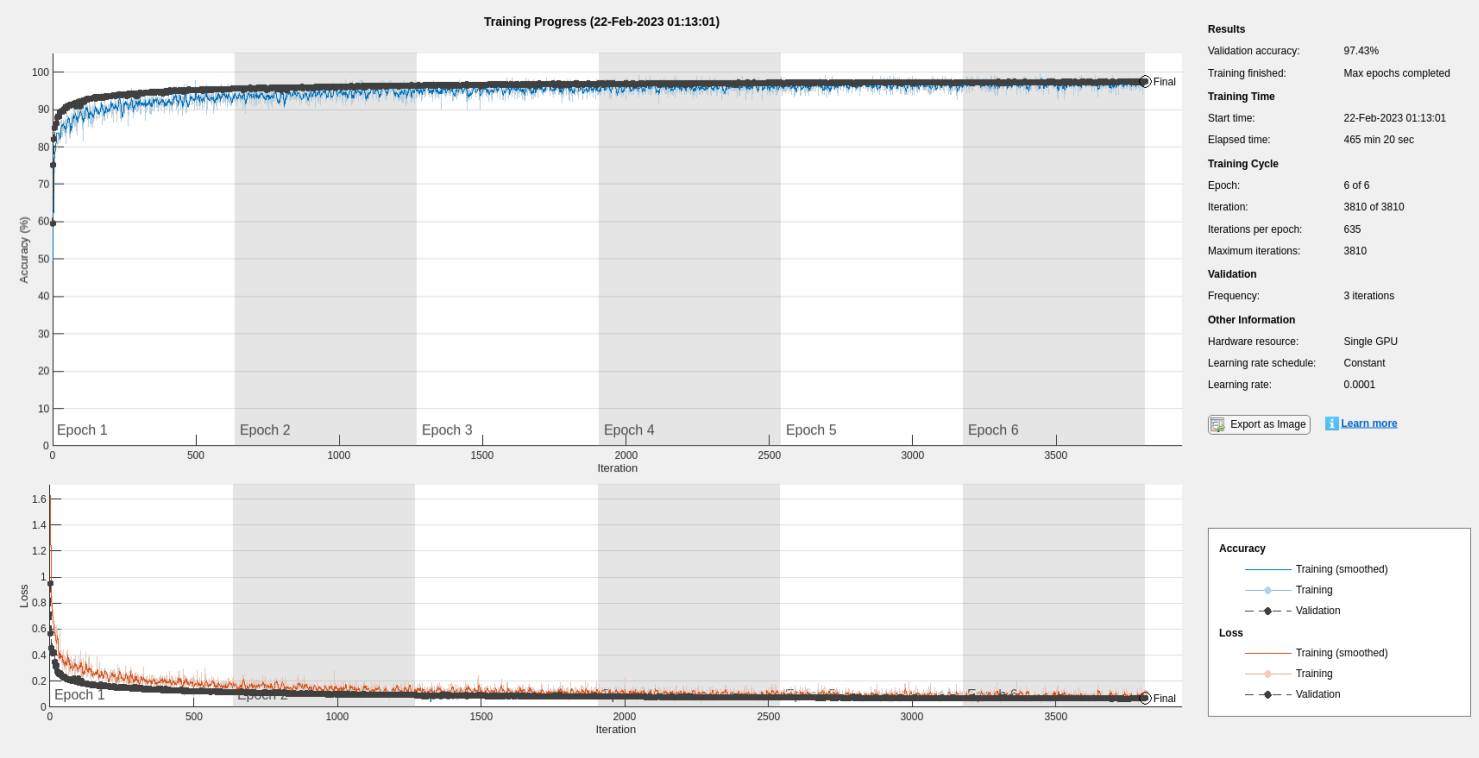
Vengono specificate le opzioni per il training. Per un trasfer learning bisogna mantenere le feature dei vecchi layer del pretrained network. Si addestra il network per poche epoche. Si specifica una mini-batch size e il validation data.

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',256, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress', ...
    'ExecutionEnvironment','gpu');
```

Addestramento della rete

Addestramento della rete usando training set, i layers e le opzioni configurate precedentemente.

```
netTransfer = trainNetwork(augimdsTrain,layers,options);
```



Classificazione

```
[YPred,scores] = classify(netTransfer,augimdsTest);
YPred = double(YPred(:,1));
```

Risultati predizione

Stampa di alcune img del test set con associata la label predetta e quella corretta.

```
if printTestSet == 1
    idx = randi([1 numTestImages],1,12);
    figure
    for i = 1:numel(idx)
        subplot(3,4,i)
        I = readimage(imdsTest,idx(i));
        label = YPred(idx(i));
        corr = YTest(idx(i));
        imshow(I)
        title("#: "+idx(i)+" / Predicted: "+label+" / Correct: "+corr)
    end
end
```


```
printGradMap = 1;
if printGradMap == 1
    figure
    for i = 1:numel(idx)
        subplot(3,4,i)
        I = imresize(readimage(imdsTest,idx(i)),inputSize(1:2));
        label = YPred(idx(i));
        corr = YTest(idx(i));
        imshow(I)
        hold on
        imagesc(gradCAM(netTransfer,I,label),'AlphaData',0.5)
        colormap jet
        title("#: "+idx(i)+" / Predicted: "+label+" / Correct: "+corr)
    end
end
```

Matrice di confusione

Stampa della matrice di confusione. Ogni colonna della matrice rappresenta i valori predetti **(Output Class)** ed ogni riga rappresenta i valori reali **(Target Class)**.

```
if printConfMatr == 1
    cm = confusionmat(YPred, YTest);
    labels = {'Female','Male'};
    figure
    plotConfMat(cm,labels)
end
```

Risultati

```
time = toc;
diff = numel(find(YPred~=YTest));
[M,N] = size(YPred);
tp = M-diff;
accuracy = round(mean(YPred == YTest)*100,2);
disp('Accuracy: '+string(accuracy)+"% - Time Elapsed: "+time+" s - True Positive vs Tot
```

```
Accuracy: 96.59% - Time Elapsed: 28071.5148 s - True Positive vs Total: 19281/19962
```