Laboratory 8

Transfer Learning with a Deep Neural Network

Due Date: Midnight Thursday Week 10

Concepts:

- Transfer Learning with a Deep Neural Network
- Image classification

Total Points: 100 points

Objectives:

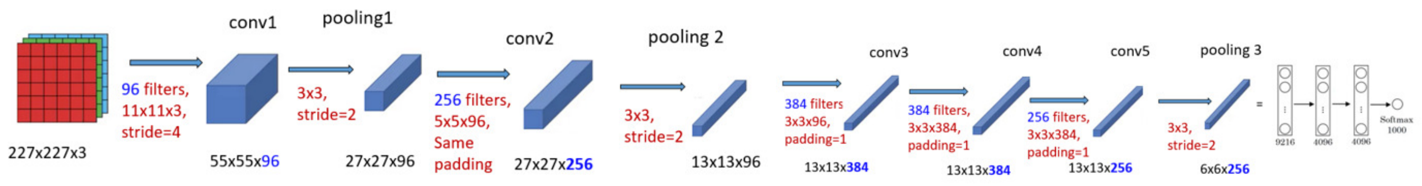
Transfer learning is commonly used in deep learning applications. You can take a pretrained network and use it as a starting point to learn a new task. Fine-tuning a network with transfer learning is usually much faster and easier than training a network with randomly initialized weights from scratch. You can quickly transfer learned features to a new task using a smaller number of training images.

Files Needed:

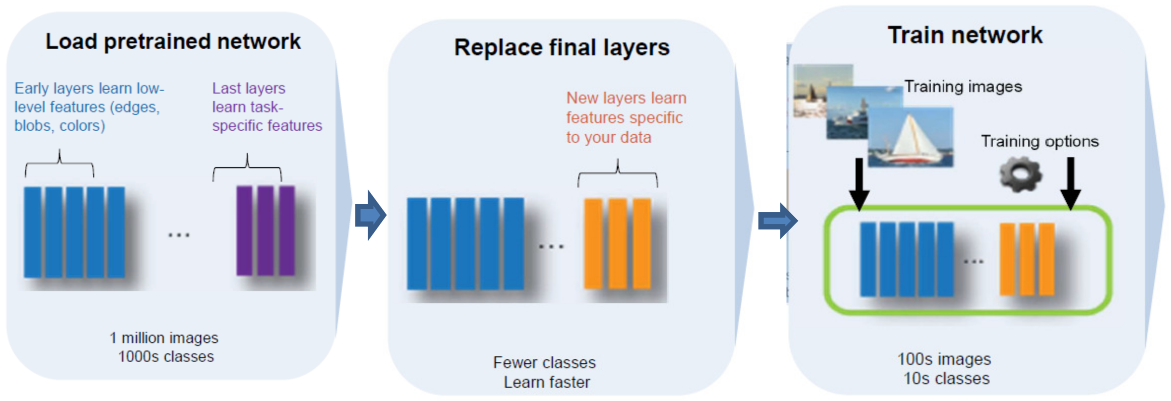
- Lab8.mlx
- Food101 dataset

Assignment: Transfer Learning Using AlexNet

In this lab, you will explore transfer learning using Alexnet. AlexNet has been trained on over a million images and can classify images into 1000 object categories (such as keyboard, coffee mug, pencil, and many animals). The network has learned rich feature representations for a wide range of images. The network takes an image as input and outputs a label for the object in the image together with the probabilities for each of the object categories. AlexNet won ImageNet competition in 2012, achieving highest classification performance. AlexNet has 5 convolutional layers and 3 fully connected layers as shown below:



Transfer learning takes layers from a network trained on a large data set and fine-tune on a new data set. The advantage of transfer learning is that the pretrained network has already learned a rich set of features. These features can be applied to a wide range of other similar tasks. For example, you can take a network trained on millions of images and retrain it for new object classification using only hundreds of images.



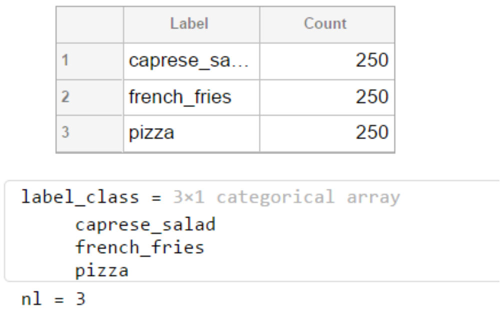
In this lab, you will develop a food classification model by implementing transfer learning using Alexnet. The provided data set is the subset of the Food101 public dataset, available at https://www.vision.ee.ethz.ch/datasets_extra/food-101/

1) Load Image Data

The dataset contains 750 color images in 3 categories {'caprese_salad', 'french_fries', 'pizza'}. You will use an imageDataStore object to read images. First, create an ImageDatastore object using the imageDatastore function, specify its properties and then import and process the data using object functions. An imageDatastore automatically labels the images based on folder names and stores the data as an ImageDatastore object.

```
% Example: Load in input images

imagepath =’foodImages/’
imds = imageDatastore(imagepath, 'IncludeSubfolders',true,...
    'LabelSource','FolderNames');
```

Task1: By referencing the Matlab documentation about the imageDatastore() function, display the number of labels in the provided data and count of each category as shown below:



	Label	Count
1	caprese_sa...	250
2	french_fries	250
3	pizza	250

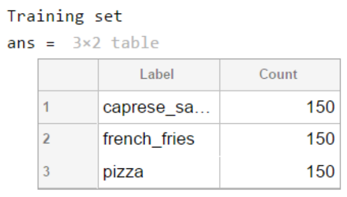
```
label_class = 3×1 categorical array
     caprese_salad
     french_fries
     pizza
nl = 3
```

2) Splitting the data

Before we start training, divide the data into training, validation, and test data sets. You will use 60% of images per category to train, 20% for validation, and specify 20% as a test set to test the food classification model after it has been trained. Use the **splitEachLabel**() function to split the ImageDatastore labels by propositions. See the document for the details about the **splitEachLabel**() function.

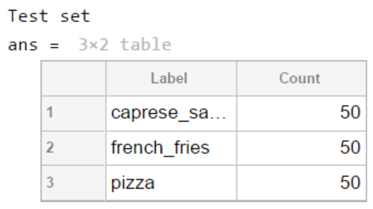
```
%Example Code

[trainDS,testDS, valDS] = splitEachLabel(imds,0.6,0.2,0.2, 'randomized');
```

Task2: Display the count for each category in training, testing, and validation data sets as shown below:



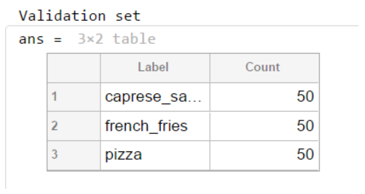
```
Training set
ans =  3×2 table
```

	Label	Count
1	caprese_sa...	150
2	french_fries	150
3	pizza	150



```
Test set
ans =  3×2 table
```

	Label	Count
1	caprese_sa...	50
2	french_fries	50
3	pizza	50



```
Validation set
ans =  3×2 table
```

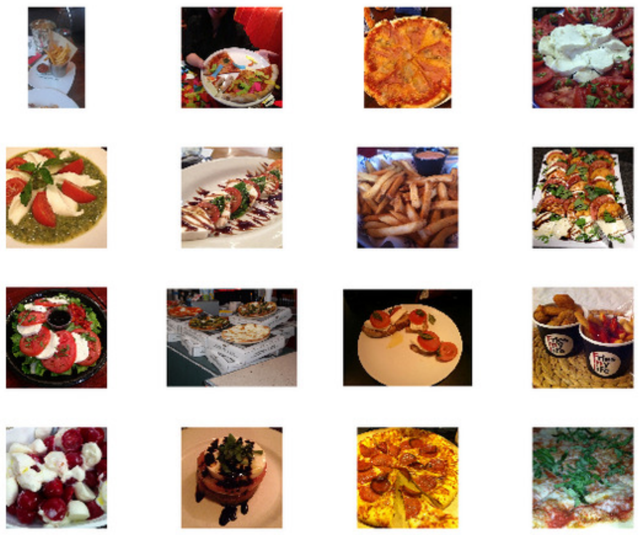
	Label	Count
1	caprese_sa...	50
2	french_fries	50
3	pizza	50

3) **Visualize the Data**

As a first step, the first 16 training examples are visualized. They correspond to images of three different categories.

- Display the randomly selected 16 image samples from the training dataset. Generate 16 numbers in the range of [1, number of the training samples] using the **randperm**() function. Then, extract the image samples from the training data using the indexes generated by the **randperm**() function.

```
%Example code

numTrainImages = numel(trainDS.Labels)
idx = randperm(numTrainImages,16);

figure
for i = 1:16
    subplot(4,4,i)

    %use readimage() function to read images. readimage() is an object function
    % in imageDatastore object

    foodImg = readimage(trainDS,idx(i));
    imshow(foodImg)
end
```



4) **Load the pretrained Alex Network**

Load the pretrained AlexNet neural network. If Neural Network Toolbox™ Model *for AlexNet Network* is not installed, then the software provides a download link. AlexNet is trained on more than one million images and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the model has learned rich feature representations for a wide range of images. To load the pretrained Alexnet,

```
net = alexnet;
```

To display the network architecture, type

```
net.Layers
```

The network has five convolutional layers and three fully connected layers.

The first layer, the image input layer, requires input images of size 227-by-227-by-3, where 3 is the number of color channels.

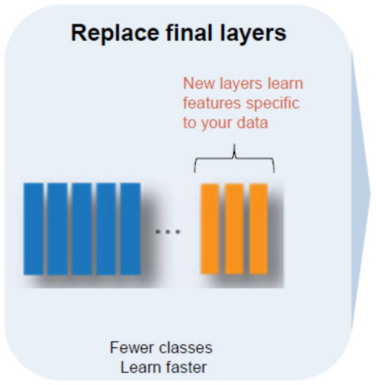
```
inputSize = net.Layers(1).InputSize
inputSize = 1×3

   227   227     3
```

The network requires input images of size 227-by-227-by-3, but input images may have sizes that are different from the pre-trained network, so we want to preprocess images, resize them to be consistent with pre-trained networks.

```
% Example code
```

```
AlexNetImageSize = inputImageSize(1:2);

%we need to resize all of the data. The resize function is specified as ReadFcn
object fucntion in imageDatastore object.

trainDS.ReadFcn = @(loc)imresize(imread(loc), AlexNetImageSize);
testDS.ReadFcn = @(loc)imresize(imread(loc), AlexNetImageSize);
valDS.ReadFcn = @(loc)imresize(imread(loc), AlexNetImageSize);
```

5) **Replace Final Layers**



The last three layers of the pretrained network `net` are configured for 1000 classes. These three layers must be fine-tuned for the new classification problem. Extract all layers, except the last three, from the pretrained network.

```
layersTransfer = net.Layers(1:end-3);
```

Transfer the layers to the new classification task by replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer.

The pre-trained layers at the end of the network are designed to classify 1000 objects. But we need to classify different objects now. So the first step in transfer learning is to replace alter just three of the layers of the pre-trained network with a set of layers that can classify 3 classes. The layers define the network architecture and contain the learned weights. Here we only alter three of the layers. Everything else stays the same.

Specify the options of the new fully connected layer according to the new data. Set the fully connected layer to have the same size as the number of classes in the new data.

```
%Example code

numClasses = numel(categories(trainDS.Labels))
layers = [
            layersTransfer     %pretrained alexNet except last three layers
            fullyConnectedLayer(numClasses)
            softmaxLayer
            classificationLayer];
```

6) Train Network

Now that the network architecture is defined, it can be trained using the new training data. First, specify the training options using the `trainingOptions()` function. For transfer learning, keep the features from the early layers of the pretrained network (the transferred layer weights). To slow down learning in the transferred layers, set the initial learning rate to a small value. When performing transfer learning, you do not need to train for as many epochs. An epoch is a full training cycle on the entire training data set. Specify the mini-batch size and validation data. The software validates the network every `ValidationFrequency` iterations during training. Below is the example of training option. See the Matlab document for the details.

```
%Example code

opts = trainingOptions('sgdm','InitialLearnRate',0.001,...
        'ValidationData',valDS,...
        'Plots','training-progress',...
        'ValidationFrequency', 5,...
        'MaxEpochs',10,...
        'MiniBatchSize', 64,...
        'ValidationPatience', 6);
```

The network training algorithm uses Stochastic Gradient Descent with Momentum (SGDM) with an initial learning rate of 0.001. The training algorithm is run for 10 epochs, etc.

Next, train the network that consists of the transferred and new layers using the trainNetwork() function. By default, `trainNetwork` uses a GPU if one is available (requires Parallel Computing Toolbox™ and a CUDA® enabled GPU with compute capability 3.0 or higher). Otherwise, it uses a CPU.

7) Classify Test Images

Task4: Classify the test images using the trained network. Use the test data to measure the classification accuracy of the network.

Question1: What is the accuracy of the food classification model?

Question2: Any suggestions to improve the performance of the trained net?

Task5: Display four sample test images with their predicted labels as below:

You can work in groups of up to two people

What to Submit to Blackboard:

- **one** ZIP file that includes :
  1) **A report (10 pts): contains answers for Question1 –Question2.**
  2) (90 pts): The live script file, “Lab8.mlx”, in that all outputs are generated either inline or on the right side.

- **Pay attention to the zip file name convention:**
  Lab8_Student1 Lastname_Student2 Lastname.zip
  Ex) Lab8_Green_Smith.zip