

## Intelligent Visual Computing [Spring 2018]

### Assignment 3: Convnet for 3D shape classification

#### Overview

In this assignment you will learn to use a deep learning package (matconvnet) in order to classify 3D shapes. You will first define a convnet that takes as input rendered images of 3D shapes across different views, then train it on a given shape database using the matconvnet package. Finally, you will use the convnet to predict the category label for each given test shape. This assignment counts for **10 points** towards your final grade.

#### Instructions

In this assignment you will use the [matconvnet](#) package. Please download the [1.0-beta20 version](#) (you may use more recent versions, but at your own risk!). Download also the [starter\\_code.zip](#) and the [3D shape database archive](#). Unzip the matconvnet package inside the starter code folder (i.e., matconvnet's readme should be stored in *starter\_code/matconvnet/README.md*). Unzip the shape database inside the starter code folder (i.e., the 'train' folder should be *starter\_code/dataset/train*). Start matlab, type "*mex -setup*;" and select a C/C++ compiler from your system (visual studio or g++ should work) [note: skip this step if you have previously configured matlab's mex compiler in your machine]. Then go to the *starter\_code* directory and type: "*cd('matconvnet/matlab');*";, then type "*vl\_compilenn*";. Hopefully all matconvnet's C/C++ files will be compiled into the matlab's mex format successfully (if not, try to reconfigure your mex compiler). When you are done, you can start working on the "*trainMVShapeClassifier.m*" and "*testMVImageClassifier.m*" scripts, as required by this assignment.

Note: the convnet you will implement is small, the dataset is small, and can be trained on the CPU (i.e., it takes ~5 min in my laptop). If you want to use the GPU (this is not required by this assignment), first you need to have access to a NVidia graphics card with at least 2GB memory. Then you need to install a CUDA version that is compatible with your graphics driver, and compile matconvnet with the option: "*vl\_compilenn('EnableGpu', 'true');*".

#### What You Need To Do (10 points in total)

**Task A [7 points]:** The starter code in "*trainMVShapeClassifier.m*" defines a simple network that performs multinomial logistic regression. Given a 112x112x1 (*width x height x channels*) intensity image as input, a convolution layer applies filters of size 112x112x1 (i.e., on the whole image). This means that this convolution layer is equivalent to a fully connected layer. The layer has K outputs, where K is the number of shape categories of the input database. The outputs are transformed to a categorical probability distribution through a 'softmax loss' layer, which implements a softmax transformation and at the same time computes the logistic loss function given our training dataset (see more info here).

The starter code parses the data in the given input dataset folder to create the training dataset (read the comments in the beginning of the script for more information about the training data format). It also divides the dataset into a training split and a validation split. Matconvnet implements batch gradient descent to learn the network parameters. It also provides an interactive plot showing how the loss function and classification error behave on the training and validation split during learning.

Execute the starter code as follows:

```
[net, info] = trainMVShapeClassifier('dataset/train/', 'matconvnet');
```

After 20 epochs (i.e., iterations over all training batches), you may observe that the classification error in the validation split is very high (about 35-40%) and does not further decrease much i.e., this simple network does not generalize well.

Your task is to change the network definition so that the network becomes deeper and have more effective layers. Your network should have the following layers:

- a) a convolution layer with 16 filters applying 8x8x1 filters on the input image. It should also include the biases units. Set stride to 2 and padding to '0' (no padding).
- b) a leaky ReLu layer with 0.1 'leak' for negative input.
- c) a max pooling layer operating on 2x2 windows with stride 2.
- d) a convolution layer with 32 filters applying 4x4x16 filters on the feature maps of the previous layer. It should also include the biases units. Set again stride to 2 and padding to 0.
- e) a leaky ReLu layer with 0.1 'leak' for negative input.
- f) again a max pooling layer operating on 2x2 windows with stride 2.
- g) a fully connected layer (implemented as a convolution layer with K filters 6x6x32 [think why], where K is the number of categories). Don't forget to include the biases.
- h) the softmax+loss layer as before

I recommend you to read the examples in *matconvnet/examples/mnist* to see how to define the above layers, [matconvnet's documentation](#), and the comments in the related *matconvnet/matlab/\*.m* and *matconvnet/matlab/simplenn/\*.m* files that serve as wrappers for the above layers. All the layer options are explained in these files. Initialize the weights and biases of the layers according to the guidelines discussed in the class.

Train the network with 20 epochs, 0.9 momentum, 0.001 learning rate, weight decay 1e-4 (as already specified in the starter code). Attach the plot that matconvnet produces after 20 epochs in your report. Submit your revised "trainMVShapeClassifier.m" script. **Please do not include the matconvnet code or the dataset in your submission!**

**Task B [3 points]:** The starter code in "trainMVShapeClassifier.m" takes as input a trained net and a test dataset with rendered images of 3D shapes. It then outputs category predictions per individual rendered image and test error averaged over all images. Execute the starter code as follows:

```
[predicted_labels, testerr] = testMVImageClassifier('dataset/test', 'matconvnet', net, info);
```

Your task is to modify this function such that the function outputs category predictions **per shape** and test error averaged over all the test **shapes**. To predict the category label per shape, experiment with two strategies:

- (a) mean view-pooling: each shape has 12 rendered images (from 12 different views). Average the output probabilities across the 12 images, and select the class with highest average probability as output category per shape.
- (b) max view-pooling: Compute the max instead of average. In other words, for each category compute its maximum probability across the 12 images. Select the class with the highest max probability as output category per shape.





Include the test error for both cases (a) and (b) in your report. Submit the code with mean view-pooling. **Please do not include the matconvnet code or dataset in your submission!**

### Bonus:

**[+1 point]:** Implement any other feature we discussed in class for convnets using matconvnet. The bonus point will be rewarded as long as the average test error decreases with these additional features.

### Submission:

Please follow the **Submission** instructions in the [course policy](#) to upload your zip file to Moodle. The zip file should contain all the Matlab code and a short PDF report. Make sure your program runs in Matlab.

-  [dataset.zip](#) 
-  [starter\\_code.zip](#) 

**Submission status**

Submission status	No attempt
Grading status	Not graded
Due date	Friday, March 2, 2018, 11:55 PM
Time remaining	4 days 11 hours
Last modified	-
Submission comments	► <a href="#">Comments (0)</a>

[Add submission](#)[Make changes to your submission](#)