

Intelligent Visual Computing [Spring 2021]

Moodle home / My courses / COMPSCI574_COMPSCI674_137404_SP21 / Week 3: CNNs (cont'd), 3D Deep Learning, Multi-View CNNs / Assignment 2: Multi-View CNN for Shape Classification

Assignment 2: Multi-View CNN for Shape Classification

Overview

In this assignment you will learn to use an existing deep learning library (pytorch) in order to classify 3D shapes based on a multi-view approach. 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. 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

You will need to install the [pytorch](#) and [matplotlib](#) libraries to run the code successfully. Download also the `python_starter_code.zip` and the 3D shape dataset below. Unzip the shape database inside the starter code folder (i.e., the 'train' folder should be `starter_code/dataset/train`). When you are done, you can start working on the `"model.py"`, `"trainMVShapeClassifier.py"` and `"testMVImageClassifier.py"` 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 ~10 min in my laptop). If you want to use a GPU (this is not required by this assignment), first you need to have access to a graphics card with at least 2GB memory, and also enable the function input argument `'cuda=True'` to accelerate the training during the pytorch installation and in the scripts. If you want to see the training loss values for each batch, you can enable the function argument `'verbose=True'` (see `run.py`). For the Task B below (`testMVImageClassifier.py`), you will need to use the input argument `'pooling=mean'` or `'pooling=max'` to switch between two required strategies.

What You Need To Do (10 points in total)

Task A [7 points]: Change the starter code in `"model.py"` so that you define the following convnet:

- ✓ 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) transform the outputs of the previous layer into a categorical probability distribution through a 'softmax loss' layer, which implements a softmax transformation and at the same time computes the softmax loss function given our training dataset.

The starter code parses the data in the given input dataset folder to create the training dataset (read the comments in the provided scripts for more information about the training data format). It also divides the dataset into a training split and a validation split.

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 $1e-4$.

Task B [3 points]: The starter code in `"testMVImageClassifier.py"` 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:

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 (a couple of sentences is OK). Submit the code with mean view-pooling.

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 or TXT for your report. **Please do not include the pytorch libraries or the dataset in your submission!**

-  [dataset.zip](#) 
-  [python_starter_code.zip](#) 

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Thursday, March 4, 2021, 11:59 PM
Time remaining	1 day 15 hours
Last modified	-
Submission comments	 Comments (0) .

Add submission

You have not made a submission yet