Intelligent Visual Computing [Spring 2022]

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 **15 points** towards your final grade if you are taking the 574 section. **If you are taking the 674 section, multiply your total points by 2/3** (i.e., the assignment counts for 10 points).

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 minutes in the instuctor's 5-year old laptop with a i7-5950HQ). 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 discussed below.

What You Need To Do (10 points in total)

Task A [70%]: Change the starter code in *"model.py"* so that you define the following convnet:

1) a convolution layer with 8 filters applying 7x7 filters on the input image. Set stride to 1 and padding to '0' (no padding). Based on the following layers, think whether you should enable the bias or not for this layer.
2) a layer normalization layer (think about what the parameter *normalized_shape* should be*)*.
3) a leaky ReLu layer with 0.01 'leak' for negative input.
4) a max pooling layer operating on 2x2 windows with stride 2.

5) a depthwise convolution layer with 8 filters applying 7x7 filters on the feature map of the previous layer. Set stride to 2 and padding to 0. Groups should be 8 to enable depthwise convolution. Based on the following layers, think whether you should enable the bias or not for this layer.
6) a layer normalization layer (think again about what the parameter *normalized_shape* should be*)*.
7) a leaky ReLu layer with 0.01 'leak' for negative input.
8) again a max pooling layer operating on 2x2 windows with stride 2.
9) a pointwise convolution layer with 16 filters applying 1x1 filters on the feature map of the previous layer (stride is 1, and no padding). Think whether to use bias or not for this layer. Note: the combination of depthwise and pointwise convolution produces a block known as "depthwise separable convolution."

10) a depthwise convolution layer with 16 filters applying 7x7 filters on the feature map of the previous layer. Set stride to 1 and padding to 0. Groups should be 16 to enable depthwise convolution. Based on the following layers, think whether you should enable the bias or not for this layer.
11) a layer normalization layer (think again about what the parameter *normalized_shape* should be*)*.
12) a leaky ReLu layer with 0.01 'leak' for negative input.
13) again a max pooling layer operating on 2x2 windows with stride 2.
14) a pointwise convolution layer with 32 filters applying 1x1 filters on the feature map of the previous layer (stride is 1, and no padding). Think whether to use bias or not for this layer.

15) a fully connected layer with K outputs, where K is the number of categories. Include the bias. Implement the fully connected layer as a convolutional one (think how and why). Also think whether you should enable the bias or not for this layer.

The starter code already implements a cross-entropy loss that incorporates a softmax transformation on the output of your neural network. The starter code also 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 according to the Kaiming Xe's uniform initialization scheme for conv layers followed by leaky ReLU activations, and Xavier uniform initialization for the pointwise convolution layers and the fully connected layer (i.e., layers not followed by ReLUs). Initialize any biases to 0. Train the network with stochastic gradient descent using 20 epochs, 0.9 momentum, 0.001 learning rate, weight decay 0.001 (as specified in the starter code).

Task B [30%]: 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 invididual 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 (or score) 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 (or score) 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 dataset in your submission!**

dataset.zip	February 10 2022, 4:11 PM
python_starter_code.zip	February 10 2022, 4:10 PM

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Thursday, February 24, 2022, 11:59 PM
Time remaining	14 days 4 hours
Last modified	-
Submission comments	Comments (0)

Add submission

You have not made a submission yet.