Lehrstuhl Computergrafik
Prof. Dr.-Ing. Hendrik P.A. Lensch
Prof. Dr.-Ing. Andreas Geiger

Eberhard Karls Universität Tübingen

Machine Learning in Graphics & Vision

Exercise 3

Release date: Thursday, 21 May 2020 - **Deadline for Homework: Wed, 10 June 2020 - 21:00**

Excercises

Please do not use jupyter-notebooks for these exercises. Create a new file (e.g., `task_3_1_a.py`) for each (sub-)task and provide your obtained text-output as a comment in the particular file when necessary. Submit your overall assignment in the form of a PDF report along with your code files (Preferably in a .zip folder).

3.1 Hands-On Neural Network Layers (3+2+2+3 Points)

Commonly used layers in deep neural networks are: 2D-convolutions (`torch.nn.Conv2d`) and ReLU (`torch.nn.ReLU`). The file `task_3_1_a.py` contains a simple CNN architecture for classification applied to MNIST digits.

a) There are missing parts and two bugs in the provided implementation template. Complete the implementation and remove the bugs. Plot the accuracy curve against the epochs. Which validation-accuracy does your implementation achieves after 15 epochs after completing the implementation?
(**Hint:** Use a TensorBoard-screenshot (preferred) or you can print your data in a log file.)
What is the state-of-the-art accuracy on classifying MNIST digits reported in the literature?

b) Provide a model summary which includes the list of the names and output shapes for each used `torch.nn` layers in your Model. How to compute the output shape of (`torch.nn.Conv2d`) with zero padding (valid) for an input of shape [B, T, T, C] and stride 2 kernel-size 3.
(**Hint:** You can use the `torchsummary` package.)

c) Run the network from **a)** on the Fashion MNIST dataset. Which validation-accuracy do you get after 15 training epochs? Try to improve the network by creating more layers or tweaking the hyper-parameters and explain your findings.

d) We provide 10 images in the template from the Fashion MNIST dataset. Your task is to create an offline predictor function (`task_3_1_d.py`) which runs inference on these images. Which labels are predicted by your network for these images?
(**Hint:** You may want to look at how to save and load a model for inference on PyTorch tutorials.)

3.2 Data Augmentation and Denoising (4 + 4 + 2 Points)

Augmenting inputs helps when the amount of training data is relatively low. Changing the data flow can help to train a neural network on a different task without a time-consuming data collection.

a) Create a dataflow class which "yields" FashionMNIST image pairs (noisy, clean) using `torch.utils.data.Dataset` as presented in the dataflow tutorial. Therefore, fill-out the missing parts in `task_3_2_a.py:NoisyFashionMNIST()` and add uniform distribution noise $(0, 1)$ to the input image. Debug your dataflow by visualizing these image pairs and save the visualization as a JPEG image.

b) A classic task is denoising images. We will use a simple encoder-decoder network π_θ with trainable parameters θ to solve this task. The network π_θ is trained on noisy/clean image pairs to predicts a single denoised image.

$$\text{inference:} \qquad \pi_\theta\colon \underbrace{\mathbb{R}^{H\times W\times C}}_{\text{noisy image}} \to \underbrace{\mathbb{R}^{H\times W\times C}}_{\text{predicted denoised image}}, \quad X \mapsto \pi_\theta(X), \tag{1}$$

$$\text{training:} \qquad \min_\theta \left\| \pi_\theta(X) - X_{\text{gt}} \right\|_2 . \tag{2}$$

Hereby, for π_θ we propose the following structure:

```
- 2D-convolution layer (16 filters, kernel size 3x3, stride 2 and padding=1)
- 2D-convolution layer (16 filters, kernel size 3x3, stride 2 and padding=1)
- 2D-convolution layer (16 filters, kernel size 3x3, stride 1 and padding=1)
- transposed 2D-convolution layer (16 filters, kernel size 3x3, stride 2 and padding=1)
- transposed 2D-convolution layer (16 filters, kernel size 3x3, stride 2 and padding=1)
- 2D-convolution layer (16 filters, kernel size 3x3, stride 1 and and padding=1 no activation!)
```

You are free to implement a more sophisticated version. Please do not forget to add a non-linear activation function (we suggest using `torch.nn.ReLu`). We ask you to implement this network structure in the provided template and train the model. Which *Mean Absolute Error* (MAE)-validation loss do you observe. The Tensorboard will show some results on the training example. Interpret, these results.

c) A common trick is to add a skip-connection by training a the network π'_θ which has the following structure:

$$\pi'_\theta\colon \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{H\times W\times C}, \quad X \mapsto X + \pi_\theta(X). \tag{3}$$

Train the modified network π'_θ separably and compare the result again π_θ in terms of MAE-validation loss. Explain possible benefits from adding such a skip-connection.

Hints

- For Tensorboard implementation, you can refer to the official Pytorch tutorial on Tensorboard basics.
- Look in the tutorials for PyTorch layers explanation and reference links (Conv2D, Maxpool2D, etc.).