אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

Applied Deep Learning
O. Azencot
Spring 2023

Assignment 4: **Convolutional Neural Networks**
Deadline: May 28, 5 pm, 2023

In this home assignment, you will implement a classification convolutional neural network and you will view its hidden features via deconvolution. Submission is in pairs. Your submission should include two files: 1) PDF file named `report.pdf` containing the answers to the tasks below, and 2) A compressed container named `code.zip` containing all the code you used. Please submit your work via Moodle. For questions, use the Slack channel, or contact me via email.

1 Background

In the seventh lecture, we learned about convolutional neural networks (CNNs) and their differences with respect to fully connected models. During the code demo, we showed a basic classification example on the CIFAR-10 dataset, which consists of 32×32 RGB images of various objects. In what follows, you will reproduce some of the results shown in class, extend the code, as well as visualize the learned features.

Latent features. In general, we call a latent feature any representation of the model that was obtained using transformations of the original input. For instance, given an image $x \in \mathbb{R}^{3\times32\times32}$, we denote by $z^{(l)}$ its latent representation after applying the l-th layer. In class, we discussed a CNN with two convolutional layers, i.e., a convolution followed by nonlinear activation (ReLU) followed by pooling, see Fig. 1 for an illustration. Thus for any input x, we obtain $z^{(1)}, z^{(2)}$ representing the latent representations after applying the first and second convolutional layers, respectively. In this home assignment, we ignore the fully connected layers that were also part of the network, and we only consider convolutional layers.

Deconvolutional models. One of the main tasks in this home assignment is to visualize latent features. This is based on a well-known paper that uses *deconvolutions* to transform a latent feature into an image. The basic idea is as follows: every convolutional layer is composed of three components–convolution, nonlinear activation, and pooling. These components can be approximately inverted using unpooling, nonlinear activation, and transposed convolution. In practice, you can design a neural network that computes latent features as we showed in class using two convolutional layers. Then, you can implement an approximate inverse using deconvolutional layers, see Fig. 1. Then, your model will have two outputs: 1) the original probabilities vector used for classification $\tilde{y}$, and 2) an image of the same size as the original input, $\tilde{x}$. The new loss will

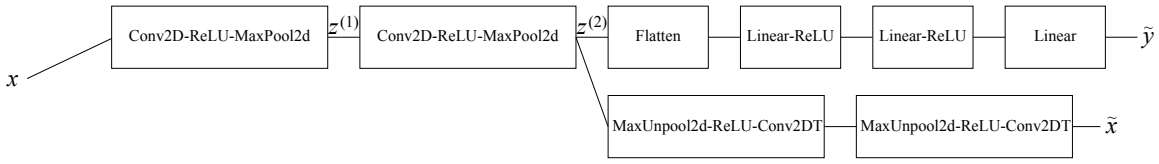

Figure 1: An illustration of the classification and deconvolutional neural network.

be composed of the original cross entropy loss, and a new reconstruction term. Namely, the loss is defined via

$$\mathcal{L}(x, y, \tilde{x}, \tilde{y}) = \mathcal{L}_{\text{ce}}(\tilde{y}, y) + \lambda \mathcal{L}_{\text{rec}}(\tilde{x}, x) \ , \tag{1}$$

where $\lambda \in \mathbb{R}^+$ is a positive weighting scalar, $\mathcal{L}_{\text{ce}}$ is the cross entropy loss we used in class, and $\mathcal{L}_{\text{rec}}(\tilde{x}, x) = \frac{1}{3}\sum_{i=1}^{3} |\tilde{x}_i - x_i|_F^2$ is the mean squared error (MSE) of the original input x_i and its reconstruction $\tilde{x}_i$, where i represents the i-th channel (R, G or B).

2 Tasks

1. **CIFAR-10 classification.** Follow the tutorial link, which includes some of the code we showed in class. Reproduce *all* of the results shown in this tutorial, attach them to your report, and describe what you did.

2. **Deconvolutional Model.** In this task, you will prepare the pipeline for visualizing latent representations $z^{(l)}$ mentioned above. Adapt your network from Task 1 to include two deconvolutional layers and adapt the loss to include a reconstruction component. Re-train your network. Report the classification error (in accuracy) of the test set and show two-three examples of reconstructed images alongside the original images.

3. **Latent Representations Analysis.** Use the trained model from Task 2 on two images, one from the train set and one from the test set. Generate their reconstructions for each and every feature. For instance, $z^{(1)}$ has six different channels, i.e., $z^{(1)} \in \mathbb{R}^{6\times14\times14}$; you should place zeros in five of the channels, reconstruct the image, and attach it to your report. For instance, setting $z^{(1)}[:5] = 0$ will erase the information for the first five channels, and keep the latent representation in the sixth channel. Repeat the above process for all channels in the first convolutional layer, and for three channels (out of 16) in the second convolutional layer. Discuss the results in your report. Are there any meaningful patterns arising in the images you get?

3 Additional Comments

1. Do not attach code to the report. Please follow the guidelines in terms of how to attach code to your submission.

2. A significant portion of the grade is dedicated to the neatness and descriptiveness of the report. You should make all the figures to be as clear as possible.

3. At the same time, the report should not be too long. Please aim for an (at most) 8 page document.

4. If you struggle with computational resources, you are allowed to shrink the CIFAR-10 dataset by a factor of 10, i.e., use only 6000 instead of the total 60,000.

5. To implement the deconvolution network, see the documentation of ConvTranspose2d, and MaxUnpool2d.

6. You can see an example of a deconvolutional network implemented in PyTorch here.