

Project 0: Experiments on Deep Learning with Cov. Neural Networks

1 Objective

In the past 3 years, deep learning has become popular and been used widely for pattern classification tasks when large training data become available. This project is designed to provide you with first hand experience on training and testing a typical convolutional neural network (CNN) structure in a task of classifying 10 object classes.

Don't panic! We have not taught this yet, and you don't have to really understand the algorithm and learning method before you play with it. Standard code and data are provided to you. All you need to do is to follow the steps and run some experiments, then report results.

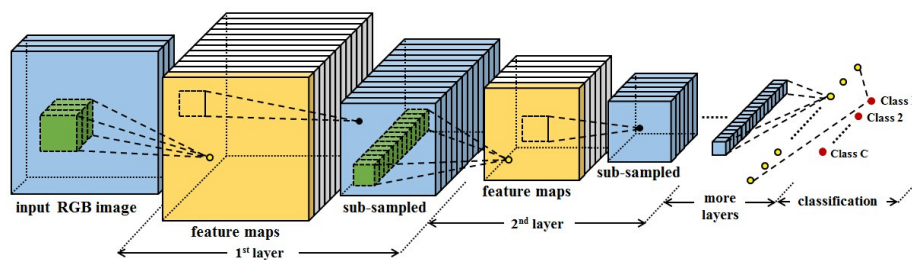


Figure 1: A ConvNet consists of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multi-nomial logistic regression

2 ConvNet

The convolutional neural network (ConvNet) is a specific artificial neural network structure inspired by biological visual cortex and tailored for computer vision tasks. The ConvNet

is a discriminative classifier, defined as a conditional (posterior) probability.

$$p(c|I; w) = \frac{\exp[f_c(I; w) + b_c]}{\sum_{c=1}^C \exp[f_c(I; w) + b_c]} \quad (1)$$

where $c \in \{1, 2, \dots, C = 10\}$ is the class label of an input image I , and $f_c(I; w)$ is the scoring function for each category, and computed by a series of operations in the CNN structure. Figure 1 displays an structure (architecture) of the CNN.

1. **Convolution.** The convolution is the core building block of a ConvNet and consists of a set of learnable filters. Every filter is a small receptive field. For example, a typical filter on the first layer of a ConvNet might have size $5 \times 5 \times 3$ (i.e. 5 pixels width and height, and 3 color channels). Figure 2 illustrates the filter convolution.

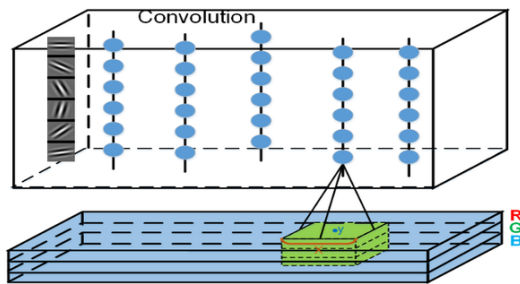


Figure 2: Filter convolution. Each node denotes a 3-channel filter at specific position. Each filter spatially convolves the image.

2. **Pooling.** Pooling is a form of non-linear down-sampling. Max-pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum.
3. **Relu.** Relu is non-linear activation function $f(x) = \max(0, x)$. It increases the non-linear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

LeNet shown in Table 2 is a typical structure design tailored for the CIFAR-10 dataset. The Block1 has 32 filters and the filter size is $5 \times 5 \times 3$. The Block5 is a logistic regression layer, in which the number of filters is the number of classes and the filter size should be the same as the output from Block4. The CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 testing images. Figure 3 shows example images from 10 categories. The objective of classification is to predict the category of each image.

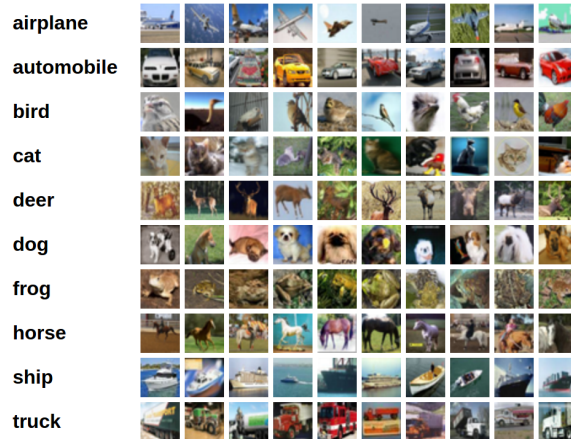


Figure 3: CIFAR-10 Dataset. CIFAR-10 contains 60,000 images in 10 classes.

Block1	Block2	Block3	Block4	Block5
Conv 5x5x3x32	Conv 5x5x32x32	Conv 5x5x32x64	Conv 4x4x64x64	Conv 1x1x64x10
Pooling 3x3	Relu	Relu	Relu	Softmax
Relu	Pooling 3x3	Pooling 3x3		

3 What to do

There are 3 steps for you to do in this project. We will also ask you to run this code again in later projects for comparison.

We will host a Tutorial Session on Tuesday evening.

1. Download CIFAR-10, and learn a LeNet as described in Table 2 on it. Plot the training error and testing error against the training iteration.
2. Keep the Block5, learn a ConvNet only with : a) Block1; b) Block1 and Block2; c) Block1, Block2 and Block3 respectively. Compare the final training errors and testing errors with LeNet in a table. (**Hint:** You need to change the filter size in Block5 to match the output from previous block.)
3. Filter response visualization. Figure 4 displays the visualization of filter responses computed by 32 filters in the first layer. The first color image is the input image, and the rest of 32 heat images are the corresponding filter response maps. The directory of `./code/examples/images/` contains 10 images from each category. For each image, visualize the filter response maps of 32 filters in the first layer. (**Hint:** For each image, you need to preprocess it by subtracting the mean image of CIFAR-10. The mean image is saved in `net.averageImage`. You can use `res = vl_simplenn(net, image)`

to compute filter responses, and all filter responses computed by the first layer filters are saved in `res(2).x`)

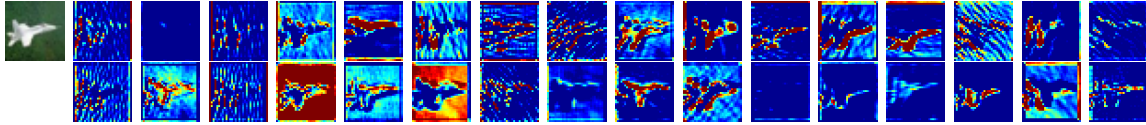


Figure 4: Filter response visualization

4 Code

The code locates at `./code/examples/`, and the code will download CIFAR-10 automatically.

1. `setup.m` is the function to compile the code. You need run `setup.m` first.
2. `cnn_cifar.m` is the main entry function, which gives an example of learning a LeNet.
3. `cnn_train.m` is the training function to train a ConvNet. **Modification:** You design the `draw_figure` function in the line of 82 and 85 to plot the training and testing error. The error is computed by `error_multiclass` function and save in `info.train.error` and `info.val.error`.
4. `cnn_cifar_init.m` defines the structure of ConvNet. **Modification:** You change the structure of ConvNet as required in step 2.