High Level Computer Vision

Exercise 3 | SS 2019

13/05/2019 - Rakshith Shetty
Exercise 3 -- Convolutional Neural Networks

- Implement a simple Convolutional network for CIFAR-10 classification
- Train the network with backpropagation
- Use batch normalization to improve training
- Explore methods to improve generalization
- Use an imagenet pre-trained network to perform transfer learning
Convolutional network architecture
Visualizing network weights

Before training

After training
Batch Normalization

\[ \text{layer} \rightarrow x \rightarrow \hat{x} = \frac{x - \mu}{\sigma} \rightarrow y = \gamma \hat{x} + \beta \]

- $\mu$: mean of $x$ in mini-batch
- $\sigma$: std of $x$ in mini-batch
- $\gamma$: scale
- $\beta$: shift

- $\mu$, $\sigma$: functions of $x$, analogous to responses
- $\gamma$, $\beta$: parameters to be learned, analogous to weights
Early stopping

- Save the best model on the validation set
- Use that as the final model.
Data augmentation

- Common method to increase the available data.
- Encode human knowledge about which transformations the classifiers should be invariant to.
- Careful to choose the augmentations one might actually encounter.
Dropout - Regularization

- One way to control model capacity
- Randomly zero out some activations.
- One hyper-parameter (p=probability of being dropout)
- Builds redundancies and helps units specialize as well.
Transfer learning
Transfer learning with VGG_11_bn

- Keep the conv layers to act as feature extractors
- Get rid of the classifier layers and add new fully connected layers to learn cifar-10 classification.

VGG:

(features): sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace)
  (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (5): ReLU(inplace)
  (6): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): ReLU(inplace)
  (10): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (11): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (12): ReLU(inplace)
  (13): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (15): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (16): ReLU(inplace)
  (17): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (18): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (19): ReLU(inplace)
  (20): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (21): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (22): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (23): ReLU(inplace)
  (24): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (26): ReLU(inplace)
  (27): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (28): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)

(avgpool): AdaptiveAvgPool2d(output_size=(7, 7))

(classifier): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace)
  (2): Dropout(p=0.5)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace)
  (5): Dropout(p=0.5)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
Submission

- Next week, Friday midnight (24/05/2018 23:59)
- Send to rshetty@mpi-inf.mpg.de
- One zip file per team
- Do not send the dataset
- Solutions next tutorial

Questions?